Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

Αναφορά 3ης Εργαστηριακής Άσκησης

# **Ομάδα Β3**

|  |  |
| --- | --- |
| Φοιτητές | ΑΜ |
|  |  |
| Μαγκλάρας Θεμιστοκλής Παναγιώτης | 1047182 |

**Εισαγωγή:**

Σκοπός της 3ης εργαστηριακής άσκησης είναι να φτιάξουμε ένα ρολόι και να εξοικειωθούμε με τη λειτουργία του. Πρόκειται για έναν έξυπνο τρόπο για να καταλάβουμε πως λειτουργεί το ρολόι χρησιμοποιώντας τα Leds του Whiteboard.

**Πείραμα σε ΑΤ91:**

#include <fcntl.h>

#include <header.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/ioctl.h>

#include <sys/mman.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <unistd.h>

#define LOW 0

#define HIGH 1

unsigned int Cnt\_number = 0;

unsigned int Pause\_Cnt\_number = 0;

unsigned int Cnt\_number4SD = 0;

unsigned int yellow\_blink = 0;

unsigned int yellow\_blink1 = 0;

volatile int press = false;

volatile int Cnt4button = 0;

char do\_reset = false;

int digitalRead(int pin) { return pioa->PDSR & (1 << pin); }

void digitalWrite(int pin, int value)

{

switch (value) {

case LOW:

pioa->CODR = pioa->CODR | (1 << pin) break;

case HIGH:

pioa->SODR = pioa->SODR | (1 << pin) break;

}

}

void upd\_button()

{

press = !digitalRead(2);

Cnt4button = Cnt4button + press;

}

void polling() {

static unsigned int unpressLast = 0;

if (press){

if (millis() - unpressLast >= 100) {

do\_reset = true;

}

} else {

unpressLast = millis();

do\_reset = false;

}

}

void setup() {

pioa->PUER = 0x8;

pioa->ODR = 0x8;

pioa->CODR = 0x7;

pioa->OER = 0x7;

aic->ICCR = (1 << PIOA\_ID);

pioa->IER = 0x01;

pioa->PER = 0xf;

}

void now\_reset() {

Cnt\_number = 0;

Cnt\_number4SD = 0;

yellow\_blink = 0;

yellow\_blink1 = 0;

Cnt4button = 0;

press = false;

do\_reset = false;

Pause\_Cnt\_number = 0;

}

void loop() {

polling();

while (timer0\_millis < 100 \* Cnt\_number) {

if (!(Cnt4button % 2)) {

digitalWrite(3, HIGH);

if (yellow\_blink1)

digitalWrite(4, HIGH);

digitalWrite(5, LOW);

} else {

digitalWrite(5, HIGH);

}

}

Cnt\_number = Cnt\_number + 1;

if (yellow\_blink1)

yellow\_blink1 = yellow\_blink1 + 1;

while (timer0\_millis < 100 \* Cnt\_number) {

if (!(Cnt4button % 2)) {

digitalWrite(3, LOW);

digitalWrite(4, LOW);

digitalWrite(5, LOW);

} else {

digitalWrite(5, HIGH);

}

}

Cnt\_number = Cnt\_number + 1;

if (Cnt\_number % 10 == 0) {

yellow\_blink = Cnt\_number / 10;

yellow\_blink1 = yellow\_blink;

}

if (Cnt\_number > 58 || do\_reset)

now\_reset();

if (!(Cnt4button % 2)) {

Pause\_Cnt\_number = Cnt\_number;

}

}

void FIQ\_handler() {

unsigned int data\_in;

if (fiq & (1 << PIOA\_ID)) {

aic->ICCR = (1 << PIOA\_ID);

data\_in = pioa->PDSR;

if (data\_in & 0x02) {

upd\_button();

}

}

}

int main() {

STARTUP;

setup();

while (getchar() != 'e') {

loop();

}

aic->IDCR = (1 << PIOA\_ID);

CLEANUP;

return 0;

}

**Πείραμα σε Tinkercad:**

Παρακάτω παρατίθεται η εικόνα του κυκλώματος που υλοποιήσαμε:

Εικόνα που περιέχει μέτρο

Περιγραφή που δημιουργήθηκε αυτόματα

Στη συνέχεια παρατίθεται ο σύνδεσμος του κυκλώματος:

<https://www.tinkercad.com/things/5Fx4NRC3PEX-erg-askhsh-3/editel?sharecode=wFJ-9JL1XjOY_lyw-DpVM9hasmdwgItgOGvxuqgOy3M>

Τέλος, παρατίθεται ο κώδικας σε Tinkercad που υλοποιήσαμε:

volatile int press = false;

volatile int Cnt4button = 0;

void upd\_button()

{

press = !digitalRead(2);

Cnt4button = Cnt4button + press;

}

char do\_reset = false;

void polling()

{

static unsigned int unpressLast = 0;

if(press)

{

if(millis() - unpressLast >= 100)

{

do\_reset = true;

}

}

else

{

unpressLast = millis();

do\_reset = false;

}

}

extern volatile unsigned long timer0\_millis;

void setup()

{

Serial.begin(115200);

pinMode(2, INPUT\_PULLUP);

pinMode(3, OUTPUT);

pinMode(4, OUTPUT);

pinMode(5, OUTPUT);

attachInterrupt(digitalPinToInterrupt(2),upd\_button,CHANGE);

}

unsigned int yellow\_blink = 0;

unsigned int yellow\_blink1 = 0;

unsigned int Cnt\_number = 0;

unsigned int Pause\_Cnt\_number = 0;

unsigned int Cnt\_number4SD = 0;

void now\_reset()

{

Cnt\_number = 0;

Cnt\_number4SD = 0;

yellow\_blink = 0;

yellow\_blink1 = 0;

noInterrupts();

Cnt4button = 0;

press = false;

timer0\_millis = 0;

interrupts();

do\_reset = false;

Pause\_Cnt\_number = 0;

}

void loop()

{

polling();

while(timer0\_millis < 100\*Cnt\_number)

{

if(!(Cnt4button%2))

{

digitalWrite(3,HIGH);

if(yellow\_blink1) digitalWrite(4,HIGH);

digitalWrite(5,LOW);

}

else

{

digitalWrite(5,HIGH);

}

}

Cnt\_number = Cnt\_number + 1;

if(yellow\_blink1) yellow\_blink1 = yellow\_blink1 - 1;

while(timer0\_millis < 100\*Cnt\_number)

{

if(!(Cnt4button%2))

{

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

}

else

{

digitalWrite(5,HIGH);

}

}

Cnt\_number = Cnt\_number + 1;

if(Cnt\_number%10 == 0)

{

yellow\_blink = Cnt\_number/10;

yellow\_blink1 = yellow\_blink;

}

if(Cnt\_number >= 59|| do\_reset) now\_reset();

if(!(Cnt4button%2))

{

Pause\_Cnt\_number = Cnt\_number;

}

char a[64];

sprintf(a, "Counter:%d\n", Pause\_Cnt\_number);

Serial.print(a);

}

**Αποτελέσματα:**

Τα αποτελέσματα είναι αυτά τα οποία περιμέναμε. Στην αρχή, το πράσινο λαμπάκι αναβοσβήνει σε κάθε νέο δευτερόλεπτο μέτρησης και άρα αντιστοιχεί στις μονάδες ένδειξης δευτερολέπτων ενώ το κίτρινο λαμπάκι αναβοσβήνει τόσες φορές όσες και η νέα δεκάδα μέτρησης στην οποία εισέρχεται η μέτρηση, δηλαδή αντιστοιχεί στις δεκάδες ένδειξης δευτερολέπτων. Όταν πατήσουμε το διακόπτη, τα 2 αυτά λαμπάκια σταματούν να αναβοσβήνουν και αρχίζει να αναβοσβήνει το κόκκινο λαμπάκι. Αυτό μας γνωστοποιεί ότι βρισκόμαστε στην κατάσταση HOLD. Μόλις, ξαναπατήσουμε το διακόπτη, επανερχόμαστε στην κατάσταση που είχαμε σταματήσει πριν το πρώτο πάτημα του και το κόκκινο λαμπάκι σταματά να αναβοσβήνει. Από εκεί και πέρα το διαδοχικό πάτημα του διακόπτη κάνει εναλλαγή μεταξύ αυτών των 2 καταστάσεων ενώ το κράτημα του για πάνω από 1 δευτερόλεπτο επαναφέρει το ρολόι στη τιμή 0.

**Διορθώσεις:**

Κατά τη διάρκεια υλοποίησης του κώδικα μας, βρεθήκαμε απέναντι σε αρκετές διορθώσεις, ωστόσο οι περισσότερες ήταν συντακτικά λάθη και έτσι αποφασίσαμε να αναλύσουμε αυτές που θεωρούμε κυριότερες. Αρχικά, χρησιμοποιούμε την now\_reset() για να μπορούμε να κάνουμε reset το ρολόι του Arduino. Για το λόγο αυτό κάνουμε χρήση της extern μεταβλητής timer0\_millis. Τέλος, θεωρήμε ισοδύναμες τις συναρτήσεις printf() του AT91 και Serial.print() του Arduino.

**Συμπεράσματα:**

Θεωρούμε πως η άσκηση αυτή ήταν ένας ωραίος τρόπος να δούμε πως μπορούμε να διαχειριστούμε το ρολόι ενός συστήματος. Ειδικότερα, πιστεύουμε πως η υλοποίηση της στο Tinkercad μας βοήθησε ακόμα περισσότερο καθώς ήμασταν σε θέση να δούμε τα φυσικά αποτελέσματα του κώδικα μας.

**ΥΠΟΣΗΜΕΙΩΣΗ:**

Στη 2η αναφορά που υλοποιήσαμε, λόγω λανθασμένης κατανόησης και απροσεξίας από μέρος μας, δεν είχαμε καταλάβει πως μπορούσαμε να σας παραπέμψουμε στα κυκλώματα που είχαμε υλοποιήσει στο Tinkercad χωρίς να τα κάνουμε πρώτα public. Αυτή τη φορά, κατανοήσαμε σωστά τις οδηγίες και έχουμε μοιραστεί το κύκλωμα που υλοποιήσαμε μέσω sharing ενώ αυτό παραμένει private.