

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ФКСУ**

**Технологичен практикум**

***Документация***

***на***

***„***Хронометър с дисплей. Спиране, пускане и зануляване през комуникационенканал.***“***

**Разработили:** Екип №1

**Група:** 46

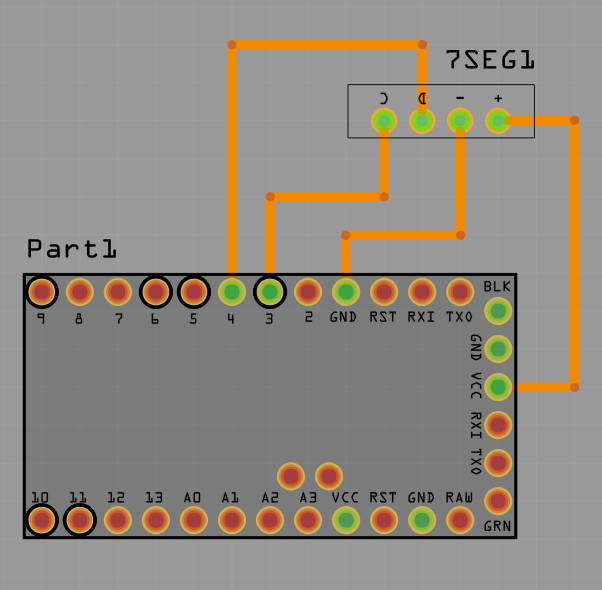
Съдържание

1. Въведение
2. PCB
3. Schematic
4. BreadBoard
5. Хардуерна реализация
6. Софтуерна реализация
7. Бъдеща разработка
8. Source code
9. Low level design
10. Снимки на прототипа

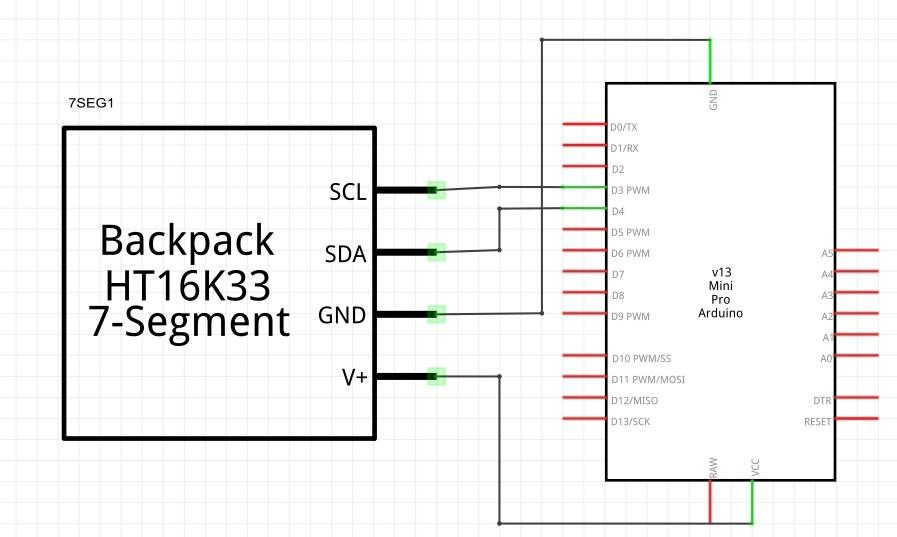
1.Въведение

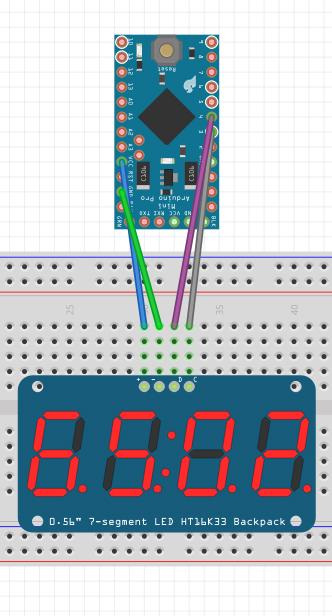
Целта на проекта е реализация на хронометър, който има функционалност - пускане, спиране и зануляване. Проекта трябва да се реализира чрез хардуерния продукт Arduino

2.PCB



3.Schematic



4.BreadBoard

5.Хардуерна реализация

Хронометъра се състои от 2 сложни компонента дисплей “4 digit catalex” и Arduino Pro Mini платка. Комукацията става чрез входовете на дисплейя - “CLK” и “DIO” и аналоговите изходите на ардуино платката “4” и “3”.

6.Софтуерна реализация

Реализацията на проекта се осъществява през средата за разработка на Arduino и програмния език C++. Използвани са библиотеки: “TimerOne“ служи за броенето на секунди, “ТМ1637” служи за изписване на числатата на дисплейя, “EEPROM “служи за запазване на времената в eeprom памета на Arduino и “avr/pgmspace”. След като устройството зареди трябва да се пусне serial monitor-а на Arduino платформата чрез който се управлява хронометъра.

7.Бъдеща разработка

Заданието на проекта позволява голяма гъвкавост за развитие, благодарение на това ние реализирахме две допълнителни функционалности: запазване на времето в eeprom паметта и показване на запазените времена. След това реализирахме още един проект, който се състои от 16x2 liquid crystal display и 4 бутона за управление - спиране, пускане, продължаване и рестартиране на броча.

8.1. Soure code - проект “iStopWatch”

#include <EEPROM.h>

#include <TimerOne.h>

#include <avr/pgmspace.h>

#include "TM1637.h"

#define ON 1

#define OFF 0

int8\_t TimeDisp[] = {0x00,0x00,0x00,0x00};

unsigned char ClockPoint = 1;

unsigned char Update;

unsigned char microsecond\_10 = 0;

unsigned char second;

unsigned char \_microsecond\_10 = 0;

unsigned char \_second;

unsigned int eepromaddr;

boolean Flag\_ReadTime;

#define CLK 3

#define DIO 4

TM1637 tm1637(CLK,DIO);

void setup()

{

Serial.begin(9600);

tm1637.set(BRIGHT\_TYPICAL);

tm1637.init();

Timer1.initialize(10000);

Timer1.attachInterrupt(TimingISR);

Serial.println("Please send command to control the stopwatch:");

Serial.println("s - start");

Serial.println("p - pause");

Serial.println("l - list the time");

Serial.println("w - write the time to EEPROM ");

Serial.println("r - reset");

}

void loop()

{

char command;

command = Serial.read();

switch(command)

{

case 's':stopwatchStart();Serial.println("Start timing...");break;

case 'p':stopwatchPause();Serial.println("Stopwatch was paused");break;

case 'l':readTime();break;

case 'w':saveTime();Serial.println("Save the time");break;

case 'r':stopwatchReset();Serial.println("Stopwatch was reset");break;

default:break;

}

if(Update == ON)

{

TimeUpdate();

tm1637.display(TimeDisp);

}

}

void TimingISR()

{

microsecond\_10 ++;

Update = ON;

if(microsecond\_10 == 100){

second ++;

if(second == 60)

{

second = 0;

}

microsecond\_10 = 0;

}

ClockPoint = (~ClockPoint) & 0x01;

if(Flag\_ReadTime == 0)

{

\_microsecond\_10 = microsecond\_10;

\_second = second;

}

}

void TimeUpdate(void)

{

if(ClockPoint)tm1637.point(POINT\_ON);

else tm1637.point(POINT\_ON);

TimeDisp[2] = \_microsecond\_10 / 10;

TimeDisp[3] = \_microsecond\_10 % 10;

TimeDisp[0] = \_second / 10;

TimeDisp[1] = \_second % 10;

Update = OFF;

}

void stopwatchStart()

{

Flag\_ReadTime = 0;

}

void stopwatchPause()

{

TCCR1B &= ~(\_BV(CS10) | \_BV(CS11) | \_BV(CS12));

}

void stopwatchReset()

{

stopwatchPause();

Flag\_ReadTime = 0;

\_microsecond\_10 = 0;

\_second = 0;

microsecond\_10 = 0;

second = 0;

Update = ON;

}

void saveTime()

{

EEPROM.write(eepromaddr ++,microsecond\_10);

EEPROM.write(eepromaddr ++,second);

}

void readTime()

{

Flag\_ReadTime = 1;

if(eepromaddr == 0)

{

Serial.println("The time had been read");

\_microsecond\_10 = 0;

\_second = 0;

}

else{

\_second = EEPROM.read(-- eepromaddr);

\_microsecond\_10 = EEPROM.read(-- eepromaddr);

Serial.println("List the time");

}

Update = ON;

}

8.2 Source code - проект 2

#include <Wire.h>

#include <LCD.h>

#include <LiquidCrystal\_I2C.h>

#define I2C\_ADDR 0x3F

#define LED\_OFF 0

#define LED\_ON 1

LiquidCrystal\_I2C lcd(I2C\_ADDR, 4, 5, 6, 0, 1, 2, 3, 7, NEGATIVE);

const int startButton = 2;

const int pauseButton = 3;

const int resumeButton = 4;

const int resetButton = 5;

bool startScreen = true;

int seconds = 0;

int minutes = 0;

char buf[50];

void setup()

{

lcd.begin (16,2);

lcd.setBacklight(LED\_ON);

pinMode(startButton, INPUT);

pinMode(pauseButton, INPUT);

Serial.begin(9600);

}

void loop()

{

if(startScreen){

lcd.setCursor(0,0);

lcd.print("00:00");

lcd.setCursor(0,1);

lcd.print("str pas res rest");

startScreen = false;

}

if((digitalRead(startButton)) == HIGH) startTimer();

}

int startTimer(){

startScreen = true;

lcd.setCursor(0,0);

minutes = 0;

seconds = 0;

while(1){

if((digitalRead(pauseButton)) == HIGH){

while(1){

if((digitalRead(resumeButton)) == HIGH) break;

if((digitalRead(resetButton)) == HIGH) return 0;

}

}

if((digitalRead(resetButton)) == HIGH)return 0;

lcd.clear();

lcd.setCursor(0,0);

sprintf(buf, "%0.2d:%0.2d",minutes, seconds);

lcd.print(buf);

lcd.setCursor(0,1);

lcd.print("str pas res rest");

seconds++;

delay(1000);

if(seconds == 59){

minutes++;

seconds = 0;

}

}

}

9. Low level design 

10.Снимки на прототипа

