Generacija NOW

Naziv projekta: IoT 112

Ime i prezime mentora: Željko Vučković, prof
Kontakt: zeljko.vuckovic@kr.t-com.hr
Naziv ustanove: OŠ Bedekovčina
Adresa: Gajeva 13
Poštanski broj: 49221
Mjesto: Bedekovčina
E-mail ustanove: os-ravnatelj@kr.t-com.hr
Učenici u timu: Lorena Semenski, Gloria Svečnjak, Ivan Pišković, Lana Pripeljaš
Razred: 8.r

Kod za slanje sms na dodir:

/********* Senzorom za dodir možemo aktivirati za slanje SMS poruke hitnoj i bližnjim osobama *********/

```
#include <SoftwareSerial.h>
// Konfigurirali smo serijski port softvera
SoftwareSerial SIM900(7, 8);

void setup(){
pinMode(KEY, INPUT); // Postavili smo pin senzora dodira na način unosa
```

```
}

void sendSMS() {
  // AT naredba za postavljanje SIM900 na SMS način rada
  SIM900.print("AT+CMGF=1\r");
  delay(100);

  // ZAMJENILI SMO X S MOBILNIM BROJEM PRIMATELJA
  // KORISTILI SMO KOD MEĐUNARODNOG FORMATA ZA MOBILNE BROJEVE
  SIM900.println("AT+CMGS=\"+XXXXXXXXXXXX\"");
  delay(100);

  // STAVILI SMO POTREBAM SADRŽAJ SMS PORUKE
  SIM900.println("ime, prezime, krvna grupa, adresa, kontakt bliske osobe");
  delay(100);

  // Završili smo AT naredbu s ^Z, ASCII kodom 26
  SIM900.println((char)26);
  delay(100);
  SIM900.println();
  // Dali smo modulu vremena za slanje SMS-a
  delay(5000);
}

int KEY = 2; // Spojili smo senzor dodira na digitalni pin 2

void loop(){
if(digitalRead(KEY)==HIGH) { // Signal senzora dodira
 // Arduino komunicira sa SIM900 GSM štitom brzinom prijenosa od 19200
 // Provjerili smo da  odgovara brzini prijenosa našeg modula
 SIM900.begin(19200);
 // Dali smo vremena da se GSM štit prijavi na mrežu
 delay(20000);

  // Poslali smo SMS
  sendSMS();
}

}
```

Kod: gps

Arduino IDE

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>
float lat = 28.5458,lon = 77.1703; // stvorili smo varijablu za zemljopisnu
širinu i dužinu objekta
SoftwareSerial gpsSerial(3,4);//rx,tx
LiquidCrystal lcd(A0,A1,A2,A3,A4,A5);
TinyGPS gps; // stvoriti gps object
void setup(){
Serial.begin(9600); // spojeno je serijski
//Serial.println("GPS primljeni signal:");
gpsSerial.begin(9600); // spojili smo GPS senzor
lcd.begin(16,2);
}
void loop(){
  while(gpsSerial.available()){ // provjerili  smo GPS podatke
  if(gps.encode(gpsSerial.read()))// kodirali smo GPS podatake
  {
  gps.f_get_position(&lat,&lon); // dobili smo zemljopisnu širinu i dužinu
  // položaj prikaza
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("GPS Signal");
  //Serial.print("Pozicija: ");
  //Serial.print("zemljopisna širina:");
  //Serial.print(lat,6);
  //Serial.print(";");
```

```
  //Serial.print("zemljopisna dužina:");
  //Serial.println(lon,6);
  lcd.setCursor(1,0);
  lcd.print("LAT:");
  lcd.setCursor(5,0);
  lcd.print(lat);
  //Serial.print(lat);
  //Serial.print(" ");
  lcd.setCursor(0,1);
  lcd.print(",LON:");
  lcd.setCursor(5,1);
  lcd.print(lon);
 }
}
String latitude = String(lat,6);
  String longitude = String(lon,6);
Serial.println(latitude+";"+longitude);
delay(1000);
```

Kod: sim 900

```
/*********
Slanje SMS poruke hitnoj i bližnjim osobama
*********/

#include <SoftwareSerial.h>

// Konfigurirali smo serijski port softvera
SoftwareSerial SIM900(7, 8);

void setup() {
  // Arduino komunicira sa SIM900 GSM štitom brzinom prijenosa od
19200
  // Provjerili smo da  odgovara brzini prijenosa našeg modula
  SIM900.begin(19200);
  // Dali smo vremena da se GSM štit prijavi na mrežu
```

```
  delay(20000);

  // Poslai smo SMS
  sendSMS();
}

void loop() {

}

void sendSMS() {
  // AT naredba za postavljanje SIM900 na SMS način rada
  SIM900.print("AT+CMGF=1\r");
  delay(100);

  // ZAMJENILI SMO X S MOBILNIM BROJEM PRIMATELJA
  // KORISTILI SMO KOD MEĐUNARODNOG FORMATA ZA MOBILNE
BROJEVE
  SIM900.println("AT+CMGS=\"+XXXXXXXXXXXX\"");
  delay(100);

  // STAVILI SMO POTREBAM SADRŽAJ SMS PORUKE
  SIM900.println("ime, prezime, krvna grupa, adresa, kontakt bliske
osobe");

  delay(100);

  // Završili smo AT naredbu s ^Z, ASCII kodom 26
  SIM900.println((char)26);
  delay(100);
  SIM900.println();
  // Dali smo modulu vremena za slanje SMS-a
  delay(5000);
}
```

Kod: sensor za dodir

```
/*********
Senzorom za dodir možemo aktivirati za slanje SMS poruke ili za info o
gps lokaciji
*********/


int ledPin = 13; // Spojili smo LED na pin 13
int KEY = 2; // Spojili smo senzor dodira na digitalni pin 2
void setup(){
pinMode(ledPin, OUTPUT); // Postavili smo LEDPin na izlazni način rada
pinMode(KEY, INPUT); // Postavili smo pin senzora dodira na način unosa
}
void loop(){
if(digitalRead(KEY)==HIGH) { // Proučili smo signal senzora dodira
digitalWrite(ledPin, HIGH); // Kada je senzor dodira HIGH, uključili smo
LED
}
else{
digitalWrite(ledPin, LOW); // a kada je senzor dodira LOW, tada smo
isključili LED
}
}
```

Kod: animation

```
int animationFrames = 7;

int animationDelays[] = { 2000, 2000, 2000, 2000, 2000, 2000, 2000 };

// Animacija je dizajnirana za 8x8 piksela
uint8_t animation[][16] = {
  { 0xc0, 0x3, 0x30, 0xc, 0xc, 0x30, 0xfc, 0x3f, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30,
0xc, 0x30 },
  { 0xfc, 0x3, 0xc, 0xc, 0xc, 0xc, 0xfc, 0x3, 0x3c, 0x0, 0xcc, 0x0, 0xc, 0x3, 0xc, 0xc
},
  { 0xfc, 0x3, 0xc, 0xc, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0xc, 0xc, 0xfc,
0x3 },
```

```
  { 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0x30, 0xc,
0xc0, 0x3 },
  { 0xf0, 0x3, 0xc0, 0x0, 0xc0, 0x0, 0xc0, 0x0, 0xc0, 0x0, 0xc0, 0x0, 0xc0, 0x0,
0xf0, 0x3 },
  { 0xc, 0x30, 0xfc, 0x30, 0xcc, 0x30, 0xc, 0x33, 0xc, 0x33, 0xc, 0x3c, 0xc, 0x3c,
0xc, 0x30 },
  { 0xc0, 0x3, 0x30, 0xc, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0xc, 0x30, 0x30, 0xc,
0xc0, 0x3 }
};
```

Kod: TimerOne

```
/*
 *  Uslužni programi za prekide i PWM za 16-bitni Timer1 na ATmega168/328
 *  Izvorni kod Jesse Tanea za http://labs.ideo.com  kolovoz 2008
 *  Izmijenili u ožujku 2009. Jérôme Despatis i Jesse Tane za podršku za
ATmega328
 *  Izmijenili su u lipnju 2009. Michael Polli i Jesse Tane kako bi popravili bug u
setPeriod() koji je uzrokovao zaustavljanje mjerača vremena
 *  Izmijenjen u lipnju 2011. od strane Lex Talionisa kako bi se dodala funkcija za
čitanje mjerača vremena
 *  Izmijenio Andrew Richards u listopadu 2011. kako bi izbjegli određene
probleme:
 * - Dodajte (duge) dodjele i cast u TimerOne::read() kako biste osigurali da
izračuni koji uključuju tmp, ICR1 i TCNT1 nisu skraćeni
 * - Osigurajte da su pristupi 16-bitnim registrima atomski - izvodite s
onemogućenim prekidima prilikom pristupa
 * - Ukloni globalno omogućavanje prekida (sei()) - može se izvoditi unutar
rutine prekida)
 * - Onemogući prekide dok je TCTN1 == 0. Tablica s podacima o ovome je
nejasna, ali eksperiment pokazuje da prekid preljeva
 * zastavica se postavlja dok je TCNT1 == 0, što rezultira fantomskim prekidom.
Može se postaviti na 1, ali postaje netočno
 * u vrlo kratkom trajanju
 * - startBottom() dodan za pokretanje brojača na 0 i rukovanje svim
omogućavanjem prekida.
 * - start() izmijenjen da omogući prekide
 * - restart() izmijenjen tako da pokazuje na startBottom()
 * Izmijenjeno u 19:26 u nedjelju, 9. listopada 2011., autor Lex Talionis
 * - preimenovao start() u resume() kako bi odražavao njegovu stvarnu ulogu
```

```
 * - preimenovano startBottom() u start(). Ovo razbija neki stari kod koji
očekuje da će nastaviti s brojanjem gdje je stao
 *
 * Ovaj program je besplatan softver: možete ga redistribuirati i/ili mijenjati
 * pod uvjetima GNU Opće javne licence koju je objavio
 * Free Software Foundation, bilo verziju 3 Licence, ili
 * (po vašoj želji) bilo koju kasniju verziju.
 *
 * Ovaj program se distribuira u nadi da će biti koristan,
 * ali BEZ IKAKVOG JAMSTVA; čak i bez impliciranog jamstva za
 * PRILIKA ZA PRODAJU ili PRIKLADNOST ZA ODREĐENU NAMJENU. Vidi
 * GNU Opća javna licenca za više pojedinosti.
 *
 * Trebali ste dobiti kopiju GNU Opće javne licence
 * zajedno s ovim programom. Ako ne, pogledajte
<http://www.gnu.org/licenses/>.
 *
 * Pogledajte Google Code projekt http://code.google.com/p/arduino-
timerone/ za najnovije
*/
#ifndef TIMERONE_h
#define TIMERONE_h

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

#define RESOLUTION 65536    // Timer1 je 16-bitni

class TimerOne
{
  public:

    //svojstva
    unsigned int pwmPeriod;
    unsigned char clockSelectBits;
                                          char oldSREG;                    // Za
zadržavanje statusnog registra dok je onemogućen

    // metode
    void initialize(long microseconds=1000000);
```

```
    void start();
    void stop();
    void restart();
                                            void resume();
                                            unsigned long read();
    void pwm(char pin, int duty, long microseconds=-1);
    void disablePwm(char pin);
    void attachInterrupt(void (*isr)(), long microseconds=-1);
    void detachInterrupt();
    void setPeriod(long microseconds);
    void setPwmDuty(char pin, int duty);
    void (*isrCallback)();
};

extern TimerOne Timer1;
#endif
```

Kod: program

```
#include <TimerOne.h>
#include "animation.h"

// Pin spojen na LAT modula
int latchPin = 8;
// Pin spojen na CLK modula
int clockPin = 12;
// Pin spojen na DI modula
int dataPin = 11;

// Broj koji označava kada treba prijeći na sljedeći okvir
unsigned long nextImage = 0;
int animationIndex = 0;
byte brightnesses[64];
int M[8];
//--------------------------------------------------------------------

void setup() {
  // postavili smo pinove na izlaz tako da možemo kontrolirati pomak
  pinMode(latchPin, OUTPUT);
```

```
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  if(animationIndex >= animationFrames)
  {
    // ponovno pokretanje indeksa animacije
    animationIndex = 0;
  }
  else{
    //load Delay time for Image
    nextImage = animationDelays[animationIndex];

    // učitaj sliku (pretvoreno)
    for(int i=0; i<64; i++) {
      brightnesses[i] = (animation[animationIndex][i/4] >> (i%4*2)) & B00000001;
      M[i/8] |= (brightnesses[i] << (i%8)) ;
    }

    //Ažuriranje slike
    screenUpdate(nextImage);
    animationIndex ++;

    //clear M[]
    for(int i=0; i<(8); ++i) {
      M[i]=0;
    }
  }

}


void screenUpdate(unsigned long frametime)
{ // function to display image

    unsigned long startime=millis();
  while(millis()-startime<frametime)
  {
    byte row = B10000000; // row 1
```

```
   for (byte k = 0; k < 8; k++)
   {
     digitalWrite(latchPin, LOW); // otvoreni zasun spreman za primanje
podataka
     shiftIt(~row); // red binarni broj
     shiftIt(M[k]); // LED niz (obrnuti)

     // Zatvorili smo zasun, slali podatke iz registara u matricu
     digitalWrite(latchPin, HIGH);
     row = row>> 1; // bitshift right
   }
  }
}



void shiftIt(byte dataOut) {
  // Shift out 8 bits LSB first, on rising edge of clock
  boolean pinState;

  //clear shift register read for sending data
  digitalWrite(dataPin, LOW);
  // za svaki bit u dataOut poslali smo bit
  for (int i=0; i<8; i++) {
    // postavili smo clockPin na LOW prije slanja bita
    digitalWrite(clockPin, LOW);
    // ako je vrijednost DataOut i (logičko AND) bitmaska
    // istinito onda smo postavili pinState na 1 (HIGH)
    if ( dataOut & (1<<i) ) {
      pinState = HIGH;
    }
    else {
      pinState = LOW;
    }
    // postavili smo dataPin na HIGH ili LOW ovisno o pinState
    digitalWrite(dataPin, pinState);
    //send bit out on rising edge of clock
    digitalWrite(clockPin, HIGH);
    digitalWrite(dataPin, LOW);
  }
```

```
  digitalWrite(clockPin, LOW); //stop shifting
}
```

Slike tima:


CONTROL

Slika opreme: