



Universitatea „Transilvania” din Brașov

Facultatea de Inginerie Electrică și Știința Calculatoarelor

Tehnologia Informației

## Cântar electronic de mână

Realizat de:

Vlad MEȘCO

TI 4301

Îndrumător:

conf. dr. ing. Dan FLOROIAN

*Universitatea „Transilvania” din Brașov*

*Aprilie 2013*

## Enunțarea temei

Se cere a fi realizat un sistem cu microcontroller AVR pentru afișare pe LCD.

### ***Tema detaliată***

Pentru acest proiect s-a ales implementarea unui cântar de mână pentru bagaje în vederea verificării respectării limitelor de greutate impuse de diversele companii aeriene.

Se vor folosi un senzor de presiune pentru cântărirea greutății, un LCD pentru afișarea de informații, 5 butoane pentru controlul sistemului, două LEDuri pe post de martori luminoși și un microcontroller AVR pentru achiziția, prelucrarea și afișarea datelor.

### ***Specificații***

Cerințe:

- Cântărirea obiectului (maxim 30kg -- această greutate este peste limitele obișnuite<sup>[1]</sup>).
- Posibilitatea setării greutății maxime admise înaintea unei utilizări
- Afișarea greutății și aprinderea unui martor de depășire a greutății maxime permise în cadrul unei utilizări

Descrierea unui caz tipic de utilizare:

1. Utilizatorul apasă butonul de reset al dispozitivului
2. Setează greutatea maximă dorită, în limita senzorului de presiune
3. Activarea secvenței de trecere în mod Cântărire
4. Așa bagajul de cârligul cântarului
5. Este achiziționată greutatea, aceasta este afișată pe ecranul LCD; greutatea e evaluată iar dacă este mai mare decât cea setată la punctul 2, se aprinde un martor de depășire a greutății admise.
6. Se repetă de la pasul 4
  - În cazul apăsării butonului de reset se reia de la pasul 2
  - În cazul apăsării butonului de OFF, dispozitivul intră în mod sleep
  - În modul Cântărire este activat LEDul care indică că dispozitivul se află în acest mod

## Studiu de implementare

Pe ADC0, pin cu intrare analogă, este legat senzorul de greutate.

LCD-ul este legat pe PC6:0 și PD6:0. De la LCD nu sunt folosiți pinii de DOTs (pentru virgule).

Pe pinul PC7 este legat un LED numit WEIGHMODE. Pe pinul PD7 e legat un LED numit OVERWEIGHT.

Pe pinii PB0, PB1, PB3 sunt legate butoanele +, - și WEIGH. Pe pinul INT2 e legat butonul OFF. Butonul de RESET cauzează resetarea sistemului și este legat la pinul de ~RESET.

Butonul OFF deconectează LCD-ul și senzorul de presiune și trece sistemul în modul sleep. Acesta poate fi trezit prin RESET.

Butoanele + și - cresc sau scad greutatea de referință cu o unitate. O unitate reprezintă un kg. Apăsarea oricăruia dintre ele trece sistemul în modul „setare greutate de referință”.

Acționarea butonului WEIGH face trecerea din modul „setare” în modul „cântărire” sau din modul „cântărire” în modul „setare”.

În modul „setare greutate de referință” e afișată greutatea de referință salvată. Butoanele + și - cresc respectiv scad greutatea cu o unitate în intervalul [1..30].

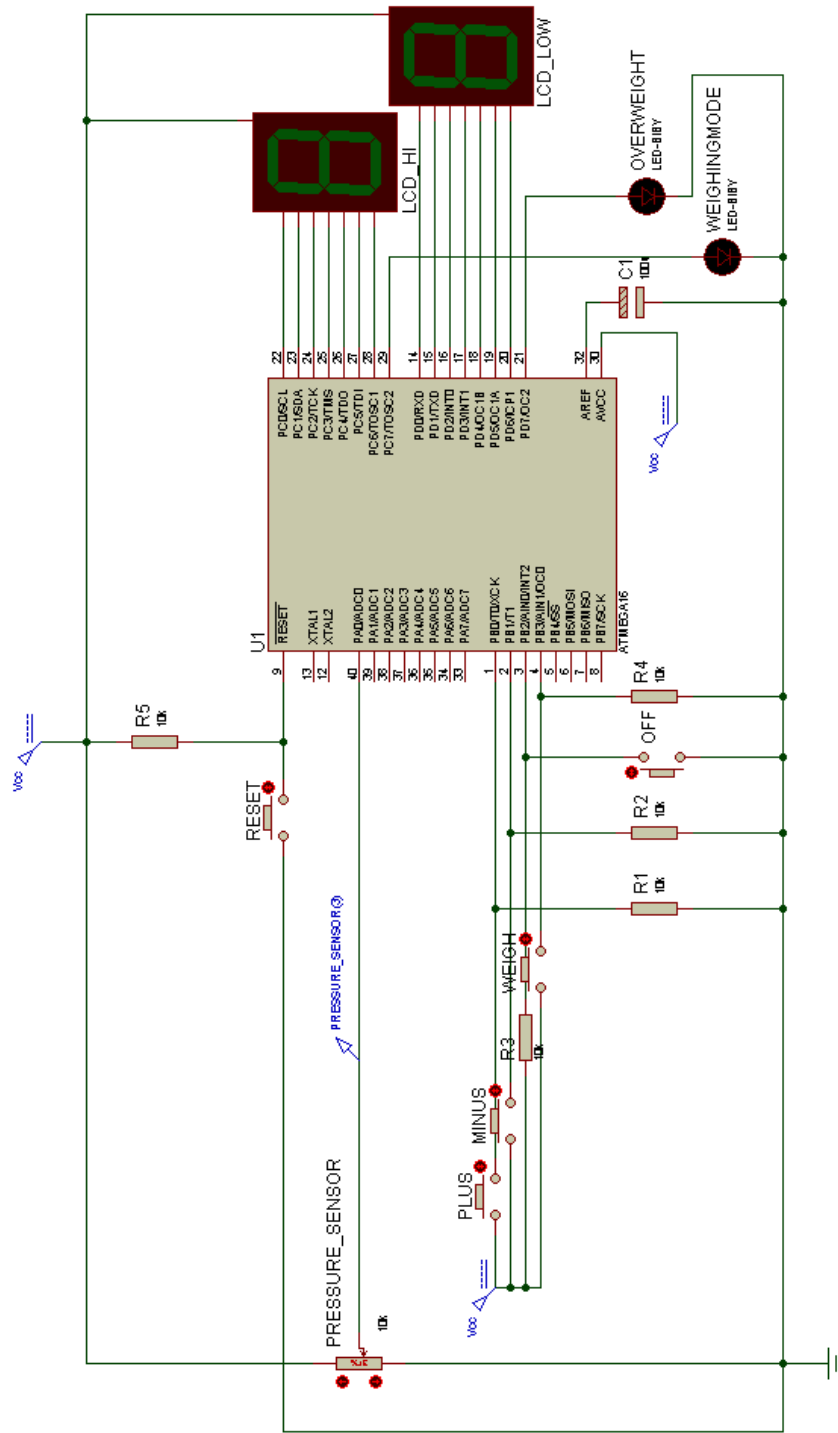
În modul „cântărire” e afișată greutatea citită de senzorul de presiune, este aprins LED-ul WEIGHMODE care semnifică că dispozitivul se află în modul „cântărire”. Dacă greutatea este mai mare decât greutatea salvată în modul „setare” se aprinde LED-ul OVERWEIGHT.

### **BOM**

- 1x Atmel AVR ATmega16<sup>[5]</sup>
- 1x CloverDisplay LCD S5080<sup>[2]</sup> (2x7seg)
- 1x FlexiForce SEN-08685 pressure sensor<sup>[8]</sup> (upward 100lbs)
- 2x LED
- 5x push buttons (RESET, OFF, +, -, WEIGH)
- 5x 10kOhm resistors
- 1x 100uF capacitor

# Schemă de montaj

## Schemă



## Listare cod

```
/*
 * scales.c
 *
 * Created: 17/04/2013 20:47:06
 * Author: Vlad Mesco
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>

#define ST_INPUT 0
#define ST_WEIGHING 1
#define ST_SLEEP 0xFF

#define NUMBER_MAX 30
#define NUMBER_MIN 5

unsigned char state;
unsigned char number;

#define DEFINE_PRINT_FUNC(WHICH, PORT) \
void print##WHICH(unsigned int x) \
{ \
    switch(x) { \
        case 0: \
            PORT = (PORT & 0x80) | (0x7F & ~0xBF); \
            break; \
        case 1: \
            PORT = (PORT & 0x80) | (0x7F & ~0x86); \
            break; \
        case 2: \
            PORT = (PORT & 0x80) | (0x7F & ~0xDB); \
            break; \
        case 3: \
            PORT = (PORT & 0x80) | (0x7F & ~0xCF); \
            break; \
        case 4: \
            PORT = (PORT & 0x80) | (0x7F & ~0xE6); \
            break; \
        case 5: \
            PORT = (PORT & 0x80) | (0x7F & ~0xED); \
            break; \
        case 6: \
            PORT = (PORT & 0x80) | (0x7F & ~0xFC); \
            break; \
    }
```

```

        case 7: \
            PORT = (PORT & 0x80) | (0x7F & ~0x87); \
            break; \
        case 8: \
            PORT = (PORT & 0x80) | (0x7F & 0x0); \
            break; \
        case 9: \
            PORT = (PORT & 0x80) | (0x7F & 0x10); \
            break; \
        case 'E': \
            PORT = (PORT & 0x80) | (0x7F & 0x6); \
            break; \
        case 'r': \
            PORT = (PORT & 0x80) | (0x7F & ~0x50); \
            break; \
        default: \
            PORT = (PORT & 0x80) | (0x7F & 0x6); \
    } \
}

ISR(INT2_vect, ISR_BLOCK)
{
    cli();
    if(GIFR & (1 << INTF2)) {
        state = ST_SLEEP;
        PORTC = 0x7F;
        PORTD = 0x7F;
        ADCSRA = 0x0; // turn off ADC
        GIFR &= ~(1 << INTF2);
    }
    sei();
}

unsigned int readAnalog()
{
    // Start conversion by setting ADSC in ADCSRA Register
    ADCSRA |= (1<<ADSC);
    // wait until conversion complete ADSC=0 -> Complete
    while (ADCSRA & (1<<ADSC));
    // Get ADC the Result
    return ADCW;
}

DEFINE_PRINT_FUNC(Lo, PORTD)
DEFINE_PRINT_FUNC(Hi, PORTC)

// print a double-digit number
void output(unsigned int x)
{
    if(x < 100) {

```

```

        unsigned int hi = x / 10;
        unsigned int lo = x % 10;

        printHi(hi);
        printLo(lo);
    } else {
        printHi('E');
        printLo('r');
    }
}

// handle buttons
void buttons()
{
    if(PINB & (1 << PINB0)) {
        state = ST_INPUT;
        number = (number + 1) % (NUMBER_MAX - NUMBER_MIN + 1);
    } else if(PINB & (1 << PINB1)) {
        state = ST_INPUT;
        number = (NUMBER_MAX - NUMBER_MIN + 1 + number - 1) %
(NUMBER_MAX - NUMBER_MIN + 1);
    } else if(PINB & (1 << PINB3)) {
        state = !state;
    }
}

// PORTA is analog input
// PORTB is button inputs
// PORTC, PORTD for LCDs
int main(void)
{
reset:
    sei(); // set global interrupt enable

    DDRB = 0xF0; // lower 4 pins are inputs
    PORTB = 0x04; // activate pull-up

    // port c & d are output
    DDRC = 0xFF; // most significant bit is weighing mode
    PORTC = 0x7F;
    DDRD = 0xFF; // most significant bit is overweight led output
    PORTD = 0x7F;

    // only set ADEN when using it, otherwise it uses too much power
    // set up analog input
    // ADSP0:2 set division factor from 1 to 128
    // Set ADCSRA Register in ATMegal68
    // ADEN <- enable
    // ADIF <- free running mode
    ADCSRA = (1 << ADEN) | (1 << ADIF) | (1 << ADPS2) | (1 << ADPS1)

```

```

| (1 << ADPS0);
  // Set ADMUX Register in ATMegal68
  // REFS0 <- make vcc reference
  // ADLAR <- left shift ADCW which is 10bit
  // lower 4 bits select positive input pin ; 0000 = ADC0 which is
what I want
  ADMUX = (1 << REFS0);
  //ADMUX |= 0x10; // no gain

  MCUCSR = (0 << ISC2); // falling edge on INT2
  GICR = (1 << INT2); // activate interrupt on INT2

  state = ST_INPUT;
  number = 23 - NUMBER_MIN;

  while(1)
  {
    buttons();
    switch(state) {
      case ST_INPUT:
        PORTC &= 0x7F;
        PORTD &= 0x7F;
        output(number + NUMBER_MIN);
        break;
      case ST_WEIGHING: {
        // (r + 5) / 10 <- rounding, gain was x10 so remove
that
        // divide by 2 because it inputs 5V @50kg
        unsigned short readValue = (readAnalog() + 5) / 20;
        output(readValue);
        if(readValue > number + NUMBER_MIN) {
          PORTD |= 0x80;
        } else {
          PORTD &= 0x7F;
        }
        PORTC |= 0x80;
        break; }
      case ST_SLEEP:
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);
        sleep_mode();
        break;
      default:
        goto reset;
    }
    _delay_ms(200);
  }
}

```



## Bibliografie

### ***Referințe***

1. <http://www.airline-baggage-fees.com/>

### ***LCD***

2. <http://www.cloverdisplay.com/pdf/S5080.pdf> -- LCD datasheet

### ***AVR***

5. <http://www.atmel.com/Images/doc2466.pdf> -- ATmega16

### ***Senzori de presiune***

6. <http://www.tekscan.com/pdf/FlexiForce-Sensors-Manual.pdf> -- manual de utilizare pentru senzor de presiune
7. <https://www.sparkfun.com/tutorials/389> -- descriere utilizare senzor de presiune FlexiForce (montaj, conversie, etc)
8. <https://www.sparkfun.com/products/8685> -- pressure sensor product page