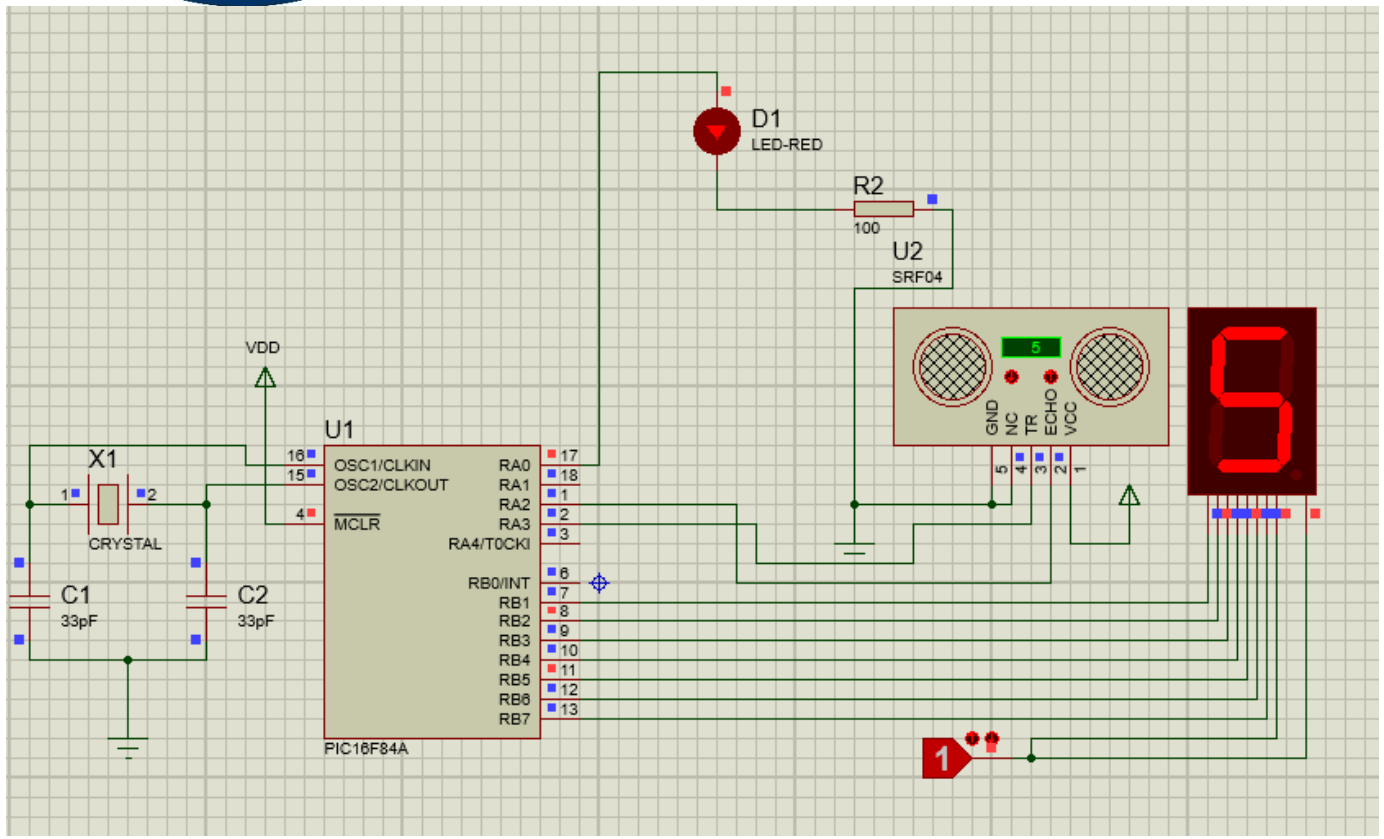
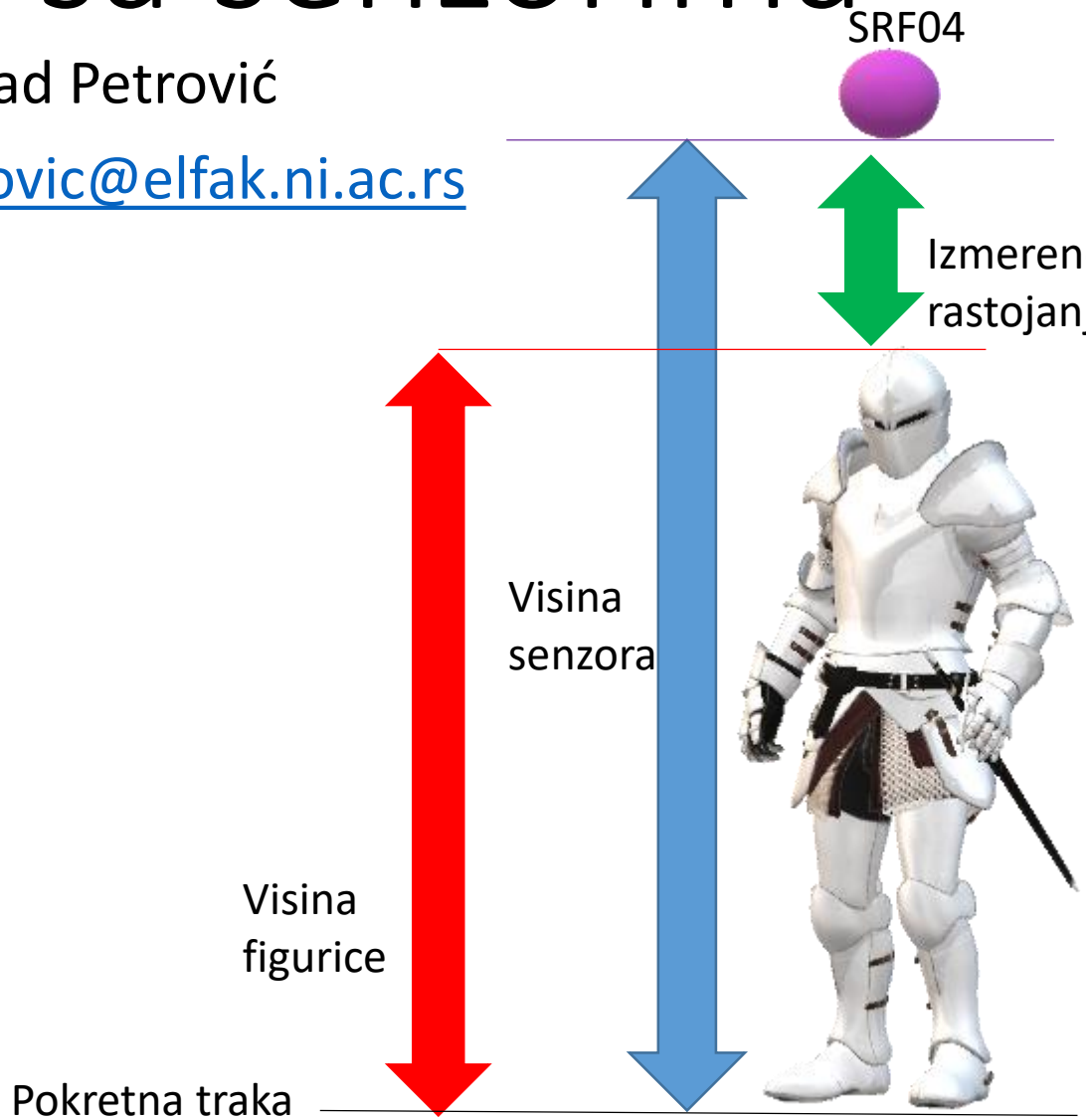
MIKS (2021)
PIC16 - Rad sa senzorima

Nenad Petrović

nenad.petrovic@elfak.ni.ac.rs



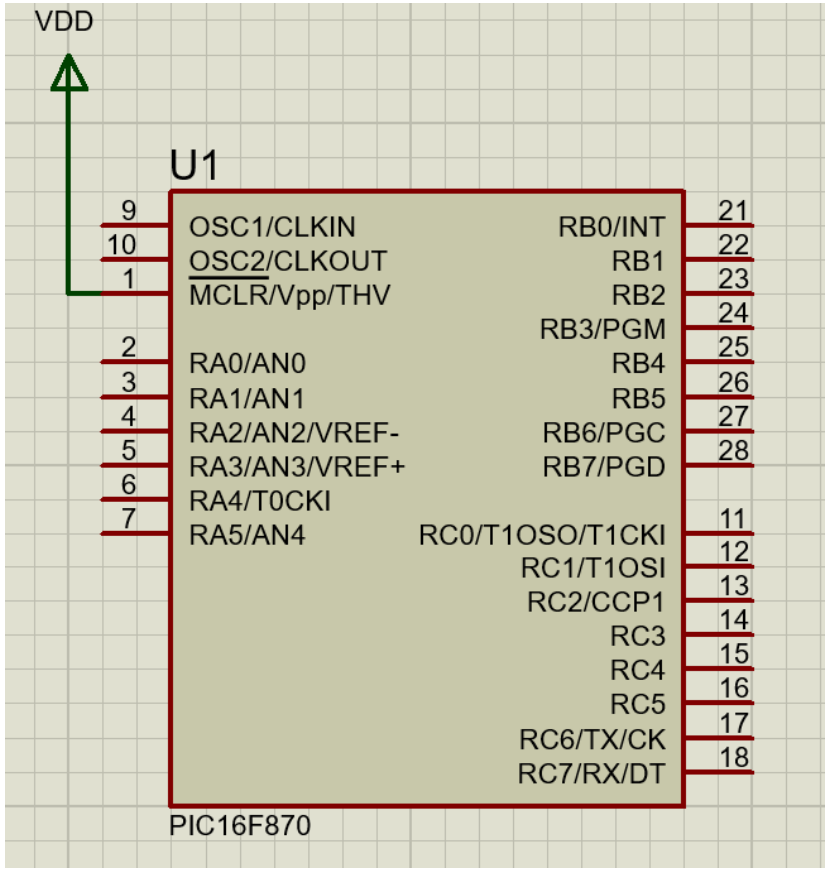


Uvod

- Senzori su uređaji čiji je cilj prikupljanje podataka o spoljašnjoj sredini
  - Temperatura
  - Vlažnost
  - CO_2
  - Prisustvo objekata
- Po vrednosti koju beleže, dele se u dve grupe
  - Analogni – daju analogni signal koji se dalje konvertuje u digitalni (poput napona)
  - Digitalni – direktno daju binarne brojeve
- Mikrokontroleri upravljaju radom senzora u većim sistemima
  - Ugrađeni AD i DA konverteri

PIC16F870

- Noviji i napredniji model iz PIC16 familije – pogodniji za rad sa analognim senzorima
  - Dodatni PORTC (8-bit)
    - PORTC, TRISC
  - 5 kanala na PORTA za 10-bit analogno-digitalnu konverziju
    - Dodatni specijalni registri
      - Konfiguracija: ADCON0, ADCON1
      - Čuvanje rezultata: ADRESH i ADRESL (po 8 bit)
        - *REZULTAT= ADRESL + (ADRESH * 256)*
  - 2 dodatna tajmera
    - TMR0 – 8 bit
    - TMR1 – 16 bit
    - TMR2 - 8 bit
  - capture/compare/PWM
  - USART
  - 2K x 14 FLASH programska memorija
  - 128 bajtova memorije podataka
  - 64 bajtova EEPROM-a



ADCON0

BIT7 BIT0

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE-	-	ADON

Bit	Uloga	
6, 7: ADCS1-ADCS0	**Selekcija takta konverzije** **00 – Fosc/2** **01 – Fosc/8** **10 – Fosc/32** **11 – Frc (takt RC oscilatora)**	
3, 4, 5: CHS2-CHS0	**Selekcija analognog kanala**	
	000 – RA0	**100 – RA5**
	001 – RA1	**101 - NEMA**
	010 – RA2	**110 - NEMA**
	011 – RA3	**111 - NEMA**
2: GO/DONE-	**Ako je ADON=1** **1: AD konverzija je u toku** **0: AD konverzija je završena (automatski nakon završetka konverzije postaje 0)**	
1: -	**Nije implementiran, čita se kao 0**	
0: ADON	**1- AD modul je uključen** **0 – AD modul je isključen**	

ADCON1

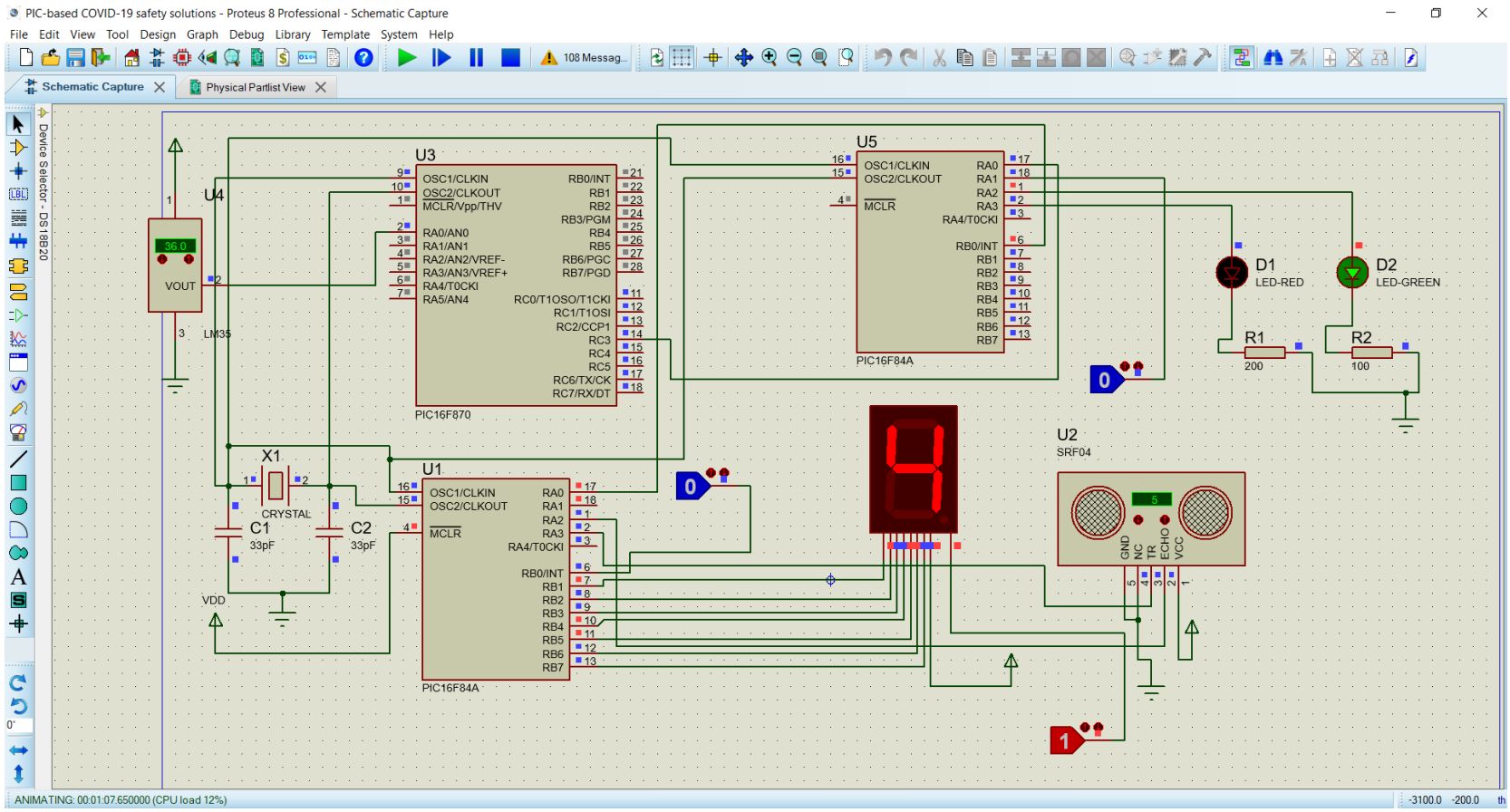
BIT7 BIT0

ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0

Bit	Uloga
7: ADFM	**1-Desno poravnanje – MSB ADRESH su 0** **0-Levo poravnanje – LSB ADRESL su 0**
4, 5, 6: -	**Nisu implementirani**
0, 1, 2, 3: PCFG3-PCFG0	**Bitovi za konfiguraciju portova**
	0000 – RA0-RA4 su analogni, Vref+ je VDD, Vref- je VSS **0001 – RA0-RA2 i RA4 su analogni, Vref+ je RA3, Vref- je VSS** **...**

Primer 1: Sistem za zaštitu od COVID-19

- Upravljačka jedinica
  - PIC16F84A
  - Zelena LED – otvori vrata
  - Crvena LED – zatvori vrata
- Detekcija osobe
  - PIC16F84A
  - Ultrazvučni senzor SRF04
- Merenje temperature
  - PIC16F870
  - Temperaturni senzor LM35
- Detekciju maske (eksterno)
  - Raspberry Pi
  - Kamera
  - Računarski vid



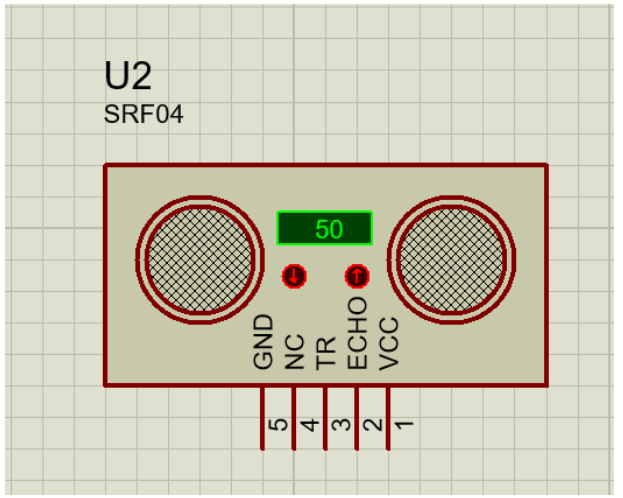
Upravljačka logika

- PIC16F84A, asemblerski kod
  - RA1 – signal da li nosi masku: 1-NE, 0-DA
  - RA0 – da li je temperatura normalna : 1-NE 0-DA
- Kada dođe nova osoba (RB0 interrupt)
  - If(mask (RA1) and temperature_ok (RA0))
    - Then: Otvori vrata – zeleni LED (RA2)
    - Else: Zatvori vrata – crveni LED (RA3)

```
Start:
   banksel TRISA
   movlw 0x03
   movwf TRISA
   movlw 0x01
   movwf TRISB

   banksel PORTA
   clrf PORTA
   clrf PORTB

   clrf INTCON
   movlw 0xB0
   movwf INTCON

   loop:
      goto loop

InterruptCode:
               //SAVE_CONTEXT

               BTFSS INTCON,INTF
               goto skip

               btfsc PORTA,0
               goto violation
               btfsc PORTA,1
               goto violation
               bsf PORTA,2
               bcf PORTA, 3
               goto no_violation
violation:
               bsf PORTA, 3
               bcf PORTA, 2
no_violation:
               bcf INTCON, INTF
skip:
EndInt:        //RESTORE_CONTEXT
               retfie

End
```

Merenje rastojanja uz pomoć ultrazvučnog senzora

- PIC16F84A, XC 8 kod, SRF04 ultrazvučni senzor
- Protokol
  - Aktivacija senzora signalom ‘’1’’ (trajanja oko 10 µs) preko trigger pina TR
  - Čekamo da ECHO pređe iz 0 u 1
  - Senzor generiše 8 impulsa od 40KHz
  - Ako postoji prepreka na putu, odbiće se ultrazvučni talasi
  - ECHO će imati vrednost “1” onoliko koliko dugo su putovali
  - Nakon toga ECHO ponovo postaje 0
  - Trajanje visokog signala se meri i koristi za izračunavanje rastojanja
    - ***d*** – rastojanje između objekta i senzora
    - ***2d*** – ukupno rastojanje koje pređu ultrazvučni talasi
    - Brzina zvuka u vazduhu je 340 m/s, što je ekvivalentno 29.412 µs/cm
    - ***trajanje_jedinice*** = 5 µs, ***brojač*** – broj vremenskih jedinica trajanja povratnog putovanja ultrazvučnih talasa



$$d = \frac{brojač \cdot trajanje\_jedinice}{2} \cdot \frac{1}{29.412}[cm]$$

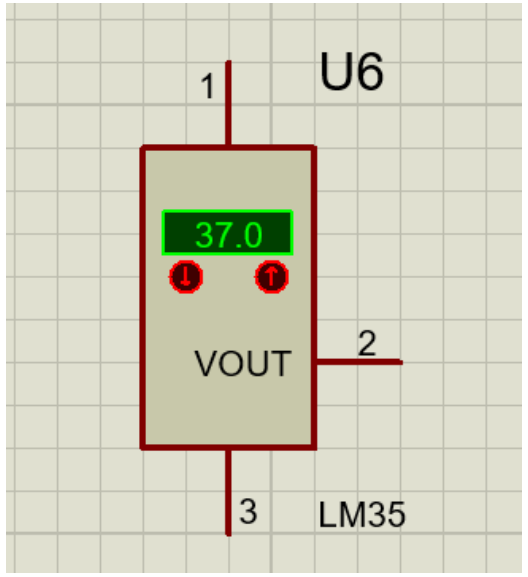
Detekcija osobe - implementacija

- EEPROM
  - Maksimalni broj osoba na lokaciji 0x03
- If (*rastojanje_osobe<granično_rastojanje*)
  - Then: If(*broj_osoba<max_osoba*)
    - Then: Poslati “1” preko RA0 upravljačkoj jedinici na njen RB0 pin
    - Else: Čekati izlazak osobe (RB0 prekid)

```
void main(void)
{
   init_eeprom();
   max_persons=read_eeprom(0x03);
   PORTA=0x00;
   PORTB=0x00;

   TRISB=0x01;
   TRISA=0x04;

   show_digit(num_persons);
   __delay_ms(3000);

   INTCON=0xB0;

   while (1)
   {
      PORTAbits.RA0=1;
      PORTAbits.RA1=1;

      duration=0;
      //Sending trigger signal to sonar
      PORTAbits.RA3=1;
      __delay_us(10);
      PORTAbits.RA3=0;

     //Waiting for echo signal
      while(!PORTAbits.RA2);
      while(PORTAbits.RA2)
      {
       __delay_us(5);
       duration++;
      }
      distance_cm=(duration*5)/29;
      duration=0;

      if(distance_cm<8 && num_persons<max_persons){
        num_persons++;
        PORTAbits.RA0=1;
        __delay_ms(1000);
        PORTAbits.RA0=0;
      }
      show_digit(num_persons);
      __delay_ms(2000);
   }
}

void interrupt intcode(){
   if(INTCONbits.INTF==1)
   {
      num_persons--;
     INTCONbits.INTF=0;
      show_digit(num_persons);
   }
}
```

Merenje temperature uz pomoć LM35 senzora

- PIC16F870, XC 8 code, LM35 analogni temperaturni senzor povezan preko pina za A/D konverziju
  - RA0 pin - 10-bit A/D konverzija
  - Specijalni registri registeri koji drže pročitanu vrednost - ADRESL i ADRESH
- Postupak merenja
  - Ova vrednost se prvo konvertuje u napon množenjem sa 4.88
  - Konačno se rezultat deli sa 10 da bi se napon konvertovao u temperaturu (+10 mV/°C linearni faktor skaliranja)



$$ADC\_Resolution = \frac{5[v]}{2^{10}-1}$$

$$V_{OUT} = ADC\_Resolution \cdot ADRES$$

$$Temperature = V_{OUT} / 10mV$$

Merenje temperature osobe

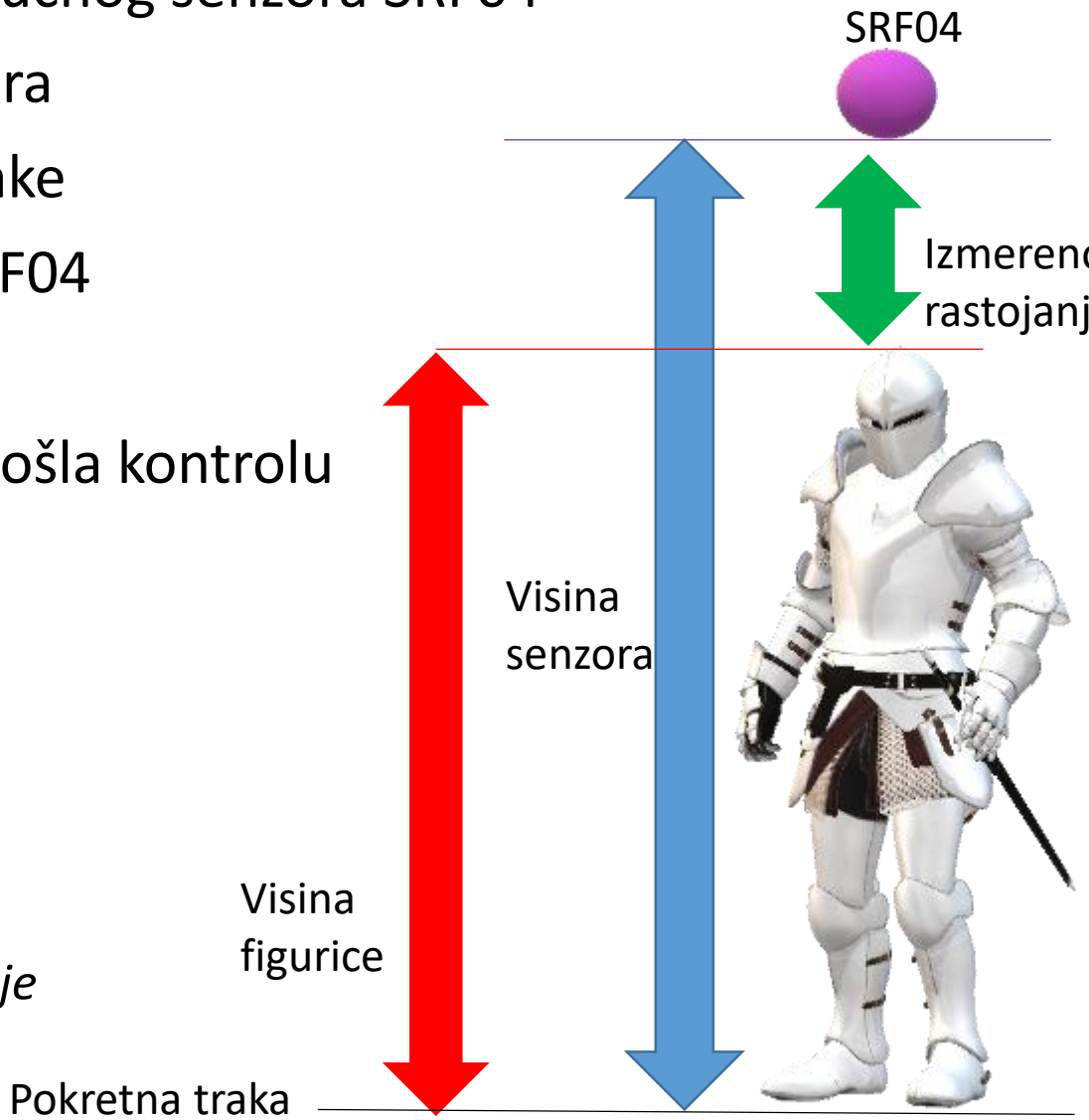
- If (*temperatura>37*)
  - Then: if (*broj_osoba<max_osoba*)
    - Then
      - Poslati signal “1” preko RC3 upravljačkoj jedinici na njen RA1
  - Else
      - Poslati signal “0” preko RC3 upravljačkoj jedinici na njen RA1

```
void ADC_Initialize()
{
  ADCON0 = 0b01000001; //ADC ON and Fosc/16 is selected
  ADCON1 = 0b11000000; // Internal reference voltage is selected
}

unsigned int ADC_Read(unsigned char channel)
{
  ADCON0 &= 0x11000101; //Clearing the Channel Selection Bits
  ADCON0 |= channel<<3; //Setting the required Bits
  __delay_ms(2); //Acquisition time to charge hold capacitor
  GO_nDONE = 1; Initializes A/D Conversion
  while(GO_nDONE); //Wait for A/D Conversion to complete
  return ((ADRESH<<8)+ADRESL);
}

void main(void)
 {
    ADC_Initialize();
    TRISB=0xFF;
    TRISA=0xFF;
    TRISC=0x00;
    PORTA=0x00;
    PORTC=0x00;
    INTCON=0xB0;

   while (1)
   {

    temp = (ADC_Read(0)*4.88);
    temp = (temp/10.00);
    if(temp>37)
       PORTCbits.RC3=1;
      else
       PORTCbits.RC3=0;

     __delay_ms(1000);
   }
 }
```

Primer 2: Kontrola kvaliteta igračaka

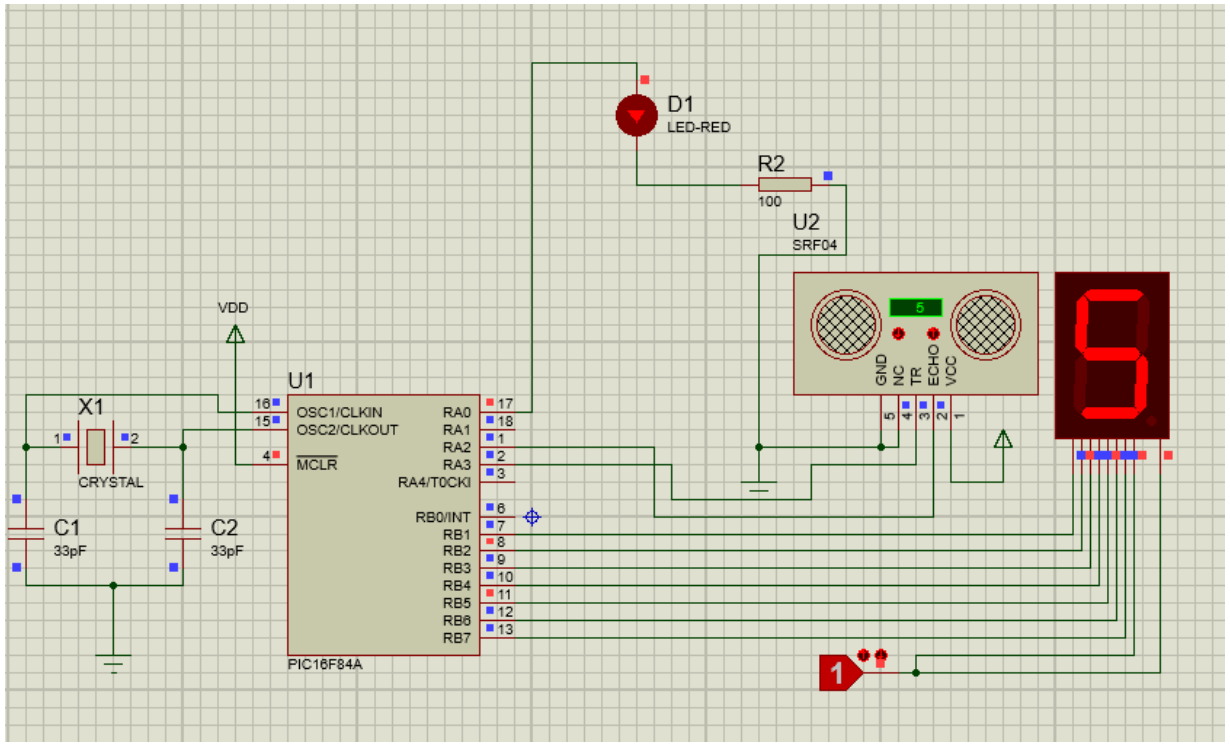
- Projektovati i implementirati sistem za kontrolu kvaliteta igračaka na proizvodnoj traci uz pomoć PIC16F84A mikrokontrolera i ultrazvučnog senzora SRF04
- Na svake 2 sekunde dolazi nova igračka ispod senzora
- Senzor je postavljen na visini 10 cm od pokretne trake
- Implementirati merenje visine igračke uz pomoć SRF04
- Visinu svake igračke prikazati na 7s displeju
- Ukoliko je visina igračke između 6 i 7 cm, onda je prošla kontrolu
  - Ukoliko nije u tom opsegu, upaliti crvenu LED diodu

Visina figurice=Visina senzora – Izmereno rastojanje



Rešenje

- RA0: crvena LED dioda
- RA3: izlazni, jer se preko njega šalje trigger
- RA2: ulazni, jer on prima ECHO
- R1-RB7: 7s displej




```
#include <xc.h>
#include <htc.h>
#include <stdlib.h>
__CONFIG(FOSC_HS & WDTE_OFF & PWRTE_ON & CP_OFF);
#define _XTAL_FREQ 8000000 //8Mhz

void prikaz(int cifra);

int vreme;
int visina;
int izmereno;

void main(void)
{
   visina=0;
   vreme=0;
   izmereno=0;

   PORTA=0x00;
   PORTB=0x00;
   prikaz(visina);

   TRISB=0x00;
   TRISA=0x04;

   while (1)
   {
       __delay_ms(2000);
      PORTAbits.RA3=1; //trigerujemo senzor
      __delay_us(10);
      PORTAbits.RA3=0;

      while(!PORTAbits.RA2) //cekamo echo
      {
      }

      while(PORTAbits.RA2) // echo sa senzora
      {
          __delay_us(5); //zbog frekvencija
          vreme++;
      }
       izmereno=(vreme*5)/29;
       vreme=0;
       visina=10-izmereno;
       prikaz(visina);
      if(visina>7 || visina <6)
          PORTAbits.RA0=1;
      else
          PORTAbits.RA0=0;
   }
}

void prikaz(int cifra){
   static const int kodovi[]={0x80,0xF2,0x48,0x60,0x32,0x24,0x04,0xB0,0x00,0x20};
   if(cifra<10)
      PORTB=kodovi[cifra];
}
```

Dodatni materijali

- N. Petrović, “Prototyping PIC16-based COVID-19 Indoor Safety Solutions within Microcomputer Systems Course”, IEEESTEC – 13th Student Projects Conference, Niš, Srbija, November 2020, pp. 185-189.
  - https://www.researchgate.net/publication/345037230_Prototyping_PIC16-based_COVID-19_Indoor_Safety_Solutions_within_Microcomputer_Systems_Course
- GitHub projekat
  - https://github.com/penenadpi/pic16covidsafety