Εργαστήριο Μικροϋπολογιστών: 8η Εργαστηριακή Άσκηση (2023)

Ομάδα 64

Ιωάννου Κωνσταντίνος AM:03119840

Μυτιληναίος Γιώργος AM:03119841

```
662: void connect_esp()
663: {
664:
665:     int x,i;
666:
667:     usart_trasmit_string("ESP:connect\n");
668:
669:     x = usart_receive_string() ;
670:
671:    if ( x == 10)
672:    {
673:         i = 0;
674:        char ar[12] = "1.Success\n";
675:        while(ar[i]!= '\n') {
676:            lcd_data(ar[i]) ;
677:            i++;}
678:       _delay_ms(2000);
679:    }
680:
681:     if ( x != 10 )
682:     {
683:         lcd_command(0x01);
684:        i = 0;
685:        char ar[12] = "1.Fail\n";
686:        while(ar[i]!= '\n')
687:        {
688:            lcd_data(ar[i]) ;
689:            i++;
690:         }
691:         _delay_ms(2000);
692:    }
693:
694:   usart_trasmit_string("ESP:url:
\"http://192.168.1.250:5000/data\"\n");
695:
696:   x = usart_receive_string() ;
697:
698:   if ( x == 10 )
699:   {
700:        lcd_command(0x01);
701:        i = 0;
702:        char ar[12] = "2.Success\n";
703:        while(ar[i]!= '\n')
704:         {
705:            lcd_data(ar[i]) ;
706:            i++;
707:         }
708:         _delay_ms(2000);
709:   }
710:
711:     if ( x != 10 )
712:     {
713:         lcd_command(0x01);
714:         i = 0;
715:         char ar[12] = "2.Fail\n";
716:         while(ar[i]!= '\n')
717:         {
718:             lcd_data(ar[i]) ;
719:             i++;
720:         }
721:         _delay_ms(2000);
722:     }
723:
724: }
```

Ζήτημα 8.1:

Για την επικοινωνία του μικρουπολογιστή με τον ESP8266 χρησιμοποιούμε την συνάρτηση connect_esp() στην αρχή της while της int main. Για να στέλνουμε μηνύματα με τον ESP χρησιμοποιείται η συνάρτηση usart_transmit_string όπου στέλνει ένα string με το μήνυμα που θέλουμε να στείλουμε.

```
275: void usart_trasmit_string(const char *txt)
276: {
277:    while (*txt) {
278:    usart_transmit(*txt++);
279:   }
280: }
```

Έπειτα περιμένουμε μήνυμα από τον ESP να βεβαιώσει ότι έλαβε το μήνυμα μέσο της usart_receive_string.

```
282: int  usart_receive_string()
283: {
284:    const char txt[20] ="\"Success\"\n" ;
285:    int x =10;
286:    char a ="Q";
287:    int i =0;
288:    while(a!='\n') {
289:        a=usart_receive();
290:       if(a != txt[i]) x=  9;
291:       i++;
292:    }
293:   return x;
294: }
```

Περιμένουμε να μας στείλει την τιμή 10 άμα το μήνυμα λήφθηκε με επιτυχία, αλλιώς τυπώνουμε στην LCD 1.Fail. Αν λάβουμε το 10 όμως τότε τυπώνει 1.Success. Έπειτα δίνουμε το url στο οποίο θα στείλουμε το payload. Ανάλογα αν λάβουμε 10 ή όχι θα τυπώσουμε 2.Success ή 2.Fail αν στάλθηκε σωστά τότε μπορούμε να προχωρήσουμε στην 8.2.

```
0970:    int main()
0971:    {
0972:
0973:        DDRC |= 0b00000000; // input for ADC
0974:        ADCSRA = 0b10000111;
0975:        ADMUX = 0b01000000; //POT0
0976:
0977:        twi_init();
0978:        PCA9555_0_write(REG_CONFIGURATION_0,
0x00); // for extedn
0979:        PCA9555_0_write(REG_CONFIGURATION_1,
0xF0); //11110000  keyboard-> IO1_0-IO1_3 output  and
IO1_4-IO1_7 input
0980:
0981:        lcd_init();
0982:        int i ;
0983:        flag = 0; //  flag : OK=0 , CN =1 ,CT =2 ,CP=3
0984:        int prev_flag=0;
0985:        usart_init(103);
0986:       while(1)
0987:       {
0988:           lcd_command(0x01);
0989:           connect_esp();
0990:
0991:           flag = 0;
0992:
0993:           int x = find_temp();
0994:           conv_temp(x);     // make temp_buffer
0995:
0996:           pressure_check();  // make pres_buffer
0997:
0998:           _delay_ms(1000);
0999:
1000:           lcd_command(0x28);  //first line
1001:           lcd_command(0x01);
1002:           lcd_command(0b10000000);
1003:
1004:           for(i=0;i<6;i++) { lcd_data(temp_buffer[i]) ;} //
pirnt on lCD temp
1005:
1006:           lcd_data('C');
1007:           lcd_data(' ');
1008:           lcd_data(' ');
1009:
1010:           for(i=0;i<5;i++) { lcd_data(pres_buffer[i])
;}//print on LCD pressure
1011:
1012:           lcd_data('c');
1013:           lcd_data('m');
1012:           lcd_data('c');
1013:           lcd_data('m');
1014:
1015:           //prev_flag =1 -> stay in Nurce Call
1016:           //prev_flag =0 -> check states  for OK,Check
Temp,Check Press
1017:           //find_status();
1018:           char key = key_press();
1019:           if(key == '4') prev_flag = 1;
1020:           if(key=='#') prev_flag = 0;
1021:           if (prev_flag == 1) flag =1;
1022:
1023:           lcd_command(0b11000000); // second line
1024:
1025:           find_and_print_status();
1026:           _delay_ms(1000);
1027:           trasnmit_payload();
1029:         }
1030:       }
```

Ζήτημα 8.2:

Για να βρούμε τις τιμές θερμοκρασίας κι πίεσης του ασθενή χρησιμοποιούμε τις συναρτήσεις που φτειάξαμε στην προηγούμενες σειρές ασκήσεων (find_temp, conv_temp, pressure_check) με την μόνη αλλαγή πώς αποθηκεύουμε τις τιμές που βρίσκουμε σε δύο global πίνακες temp_buffer και pres_buffer για την θερμοκρασία κι την πίεση αντίστοιχα και τα τυπώνουμε στην LCD. Στο τέλος θα βάλουμε τα buffers στο payload που θα στείλουμε. Για να ελέγχουμε το status του ασθενή χρησιμοποιούμε τις μεταβλητές flag και prev_flag. Η μεταβλητή flag καθορίζει την τιμή του status που θα τυπωθεί στην LCD και το payload.

Αν δεν υπάρχει πρόβλημα με τις μετρήσεις τότε:

flag = 0: OK

Αν πατηθεί το κουμπί 4 στο πληκτρολόγιο τότε:

flag = 1: NURSE CALL

Αν η θερμοκρασία δεν είναι >34 και <37 τότε:

flag = 2: CHECK TEMP

Αν η πίεση δεν είναι <12 και >4 τότε:

flag = 3: CHECK PRESSURE

Η prev_flag διατηρεί την τιμή flag = 1 αν έχει πατηθεί το κουμπί 4. Το κάνουμε αυτό ώστε να μην αλλάξει η τιμή της flag αν δεν πατηθεί πρώτα το κουμπί # όπου θα αλλάξει την prev_flag = 0 και μετά η flag μπορεί να αλλάξει ανάλογα τις τιμές. Ανάλογα την τελική τιμή της flag όταν μπαίνει το πρόγραμμα στην find_and_print_status() καταχωρείτε η τιμή του status στον status_buffer όπου μετα θα τυπωθεί στην LCD και στο payload.

```
874: void trasnmit_payload() {
875:     int i;
876:
877:
usart_trasmit_string("ESP:payload:[{\"name\":
\"temperature\",\"value\": ");
878:
879:     usart_transmit("");
880:     for(i=0;i<6;i++) {
usart_transmit(temp_buffer[i]) ;}
881:     usart_transmit("");
882:
883:     usart_trasmit_string("},{\"name\":
\"pressure\",\"value\": ");
884:
885:     usart_transmit("");
886:     for(i=0;i<5;i++) {
usart_transmit(pres_buffer[i]) ;}
887:     usart_transmit("");
888:
889:     usart_trasmit_string("},{\"name\":
\"team\",\"value\": \"64\"},{\"name\":
\"status\",\"value\": ");
890:
891:     usart_transmit("");
892:
893:     i=0;
894:     while(status_buffer[i]!= '\n'){
usart_transmit(status_buffer[i]); i++;}
895:
896:     usart_transmit("");
897:     usart_trasmit_string("}]\n");
898:
899: ////////////////////////for 3.Success
900:     lcd_command(0x01);
901:
902:     int  x = usart_receive_string() ;
903:
904:     if ( x == 10 )
905:     {
906:         lcd_command(0x01);
907:         i = 0;
908:         char ar[12] = "3.Success\n";
909:         while(ar[i]!= '\n') {
910:             lcd_data(ar[i]) ;
911:             i++;}
912:         _delay_ms(2000);
913:     }
914:
915:     if ( x != 10 )
916:     {
917:         lcd_command(0x01);
918:         i = 0;
919:         char ar[12] = "3.Fail\n";
920:         while(ar[i]!= '\n') {
921:             lcd_data(ar[i]) ;
922:             i++;}
923:             _delay_ms(2000);
924:         _delay_ms(2000);
925:     }
```

Ζήτημα 8.3:

Το payload κατασκευάζεται βήμα βήμα στην payload_transmit. Ο ESP θεωρεί πως ένα μήνυμα έχει τελειώσει μόνο όταν εμφανίζεται το /n οπότε στέλνουμε με την usart_transmit_string() σε κομμάτια ότι θέλουμε να στείλουμε και στο τελευταίο μήνυμα βάζουμε το /n για να του πούμε πως τελειώσαμε το μήνυμα μας. Στην συνέχεια περιμένουμε να λάβουμε βεβαίωση πως το μήνυμα στάλθηκε και ανάλογα το αποτέλεσμα στέλνουμε 3.Success ή 3.Fail.

Τέλος στέλνουμε στον ESP την εντολή transmit για να στείλει το payload στον σέρβερ και διαβάζουμε την απάντηση που μας στέλνει πίσω (περιμένουμε να δούμε 200 OK) και τυπώνουμε το αποτέλεσμα στην οθόνη.

```
926: ////////////////////////for transmit 200-OK
927:
928:     lcd_command(0x01);
929:     _delay_ms(50);
930:     usart_trasmit_string("ESP:transmit\n");
931:     char test[15];
932:     test[0]=usart_receive();
933:
934:     int  j=0;
935:     while(test[j]!= '\n') {
936:         j++;
937:         test[j]=usart_receive();
938:     }
939:
940:     _delay_ms(500);
941:     for(int i =0; i<j;i++){
942:         lcd_data(test[i]);
943:     }
944:
945:     _delay_ms(2000);
946:  }
```

Αφού γίνουν όλα αυτά πάει ξανά στην αρχή του while κι ξαναρχίζει την διαδικασία.