בס"ד

דוח מסכם 4

מגישים: תמיר יעקב ורון בניטה.

בס"ד

תיאור המשימה

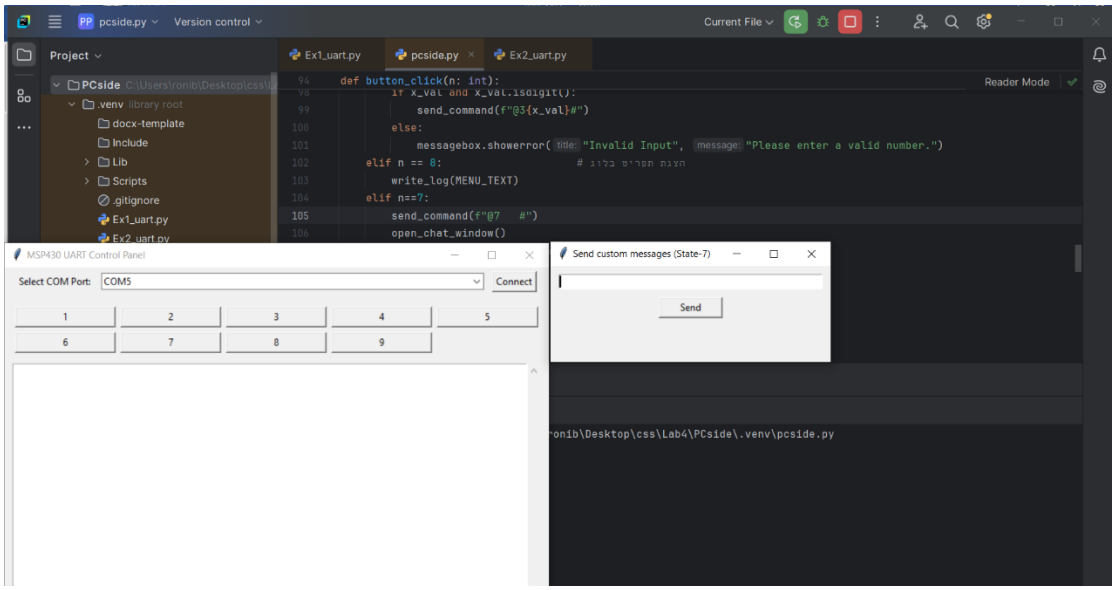
הוסף את הסעיף הבא (מסומן בכחול) לתפריט בדרישת משימת דו"ח מכין (**כך שהדרישה המקורית של התפריט לא תיפגע ותעבוד בצורה מלאה**):

Menu
1. Blink RGB LED, color by color with delay of X[ms]
2. Count up onto LCD screen with delay of X[ms]
3. Circular tone series via Buzzer with delay of X[ms]
4. Get delay time X[ms]:
5. LDR 3-digit value [v] onto LCD
6. Clear LCD screen
7. Get strings from PC side and print them onto LCD
8. Show menu
9. Sleep

המצב מוגדר להסתיים לאחר בחירת מספר של סעיף בתפריט ע"י המשתמש

הסבר: בבחירת סעיף תפריט 7 (בכחול), המשתמש יכול לשלוח רצף מחרוזות באופן טורי להדפסה על גבי מסך ה LCD עד לקבלת תו מסיים (ערך תו המסגרת נתונה להחלטתכם האישית).

בתחילה ניקינו את מצב 7 מהצגת תפריט והתאמנו את הקוד בצד המחשב וצד הבקר כך יתווסף עוד מצב.
בצד הבקר הוספנו עוד $case$, ובצד המחשב הוספנו עוד כפתור בממשק ה-GUI שלנו והתאמנו אותו לדרישת המשימה שיוכל לשלוח מחרוזות.



המשימה מאוד דומה למימוש של מצב 3, נדרשנו להתאמות מינימליות כדי שהסעיף יפעל כמו שצריך.

במצב 3 הגדרנו שברגע שאנחנו נכנסים למצב אנחנו קולטים כל תו ומכניסים אותו לבאפר עד לתו סיום. הגדרנו את הבאפר להיות בגודל של 32, לכן במשימה שלנו כאשר אנחנו נכנסים למצב 7 אנחנו נכנסים שוב למצב של קריאה תו תו כמו במצב ש עד לסיום של הקריאה לפי התו הסיום הרצוי. כאשר אנחנו מסיימים לקרוא אנחנו מדפיסים את הבאפר.

בס"ד

```c
void api_handle_msg(const char* msg)
{
    char temp;

        temp = (msg[0]);
        switch(temp)
```

main.c | bsp.c | bsp.h | app.h

```c
        lcd_clear();
        rec_X();

        break;

    }

    case state4:
    {
        lcd_clear();
        poten_meas();
        break;
    }
    case state5:
    {
        lcd_clear();
        send_love();
        break;
    }
    case state6:
    {
        lcd_clear();
        reset_count();
        break;
    }
    case state7:
    {
    lcd_clear();
    get_string();
    break;
    }
    case state8:
    {
        lcd_clear();
        break;
    }
```

api.c | api.h

```c
        case '3':
        {
            nextstate = state3;
            X = (int)atoi(&msg[1]);
            break;
        }
        case '4':
        {
            nextstate = state4;
            break;
        }
        case '6':
            nextstate = state6;
            break;

        case '7':
        {
            nextstate = state7;
            strcpy(dataString, msg+1);

            break;
        }
        case '8':
            nextstate = state8;

            break;
        case '9':
            nextstate = state0;
            break;
        }
}

void get_string()
{

    lcd_puts(dataString);        //
```

halGPIO.c | LCD.c

```c
}



//*************************************************************
//            UART Interrupt Service Rotine
//*************************************************************
#pragma vector = USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
    char c = UCA0RXBUF;

    if (c == '@') {
        msg_idx = 0;
        msg_in_progress = 1;
    }
    else if ((c == '#') && msg_in_progress) {
        msg_buf[msg_idx] = '\0';
        msg_in_progress = 0;

        // handle the msg
        api_handle_msg((char*)msg_buf);
    }
    else if (msg_in_progress && msg_idx < MSG_BUF_SIZE - 1) {
        msg_buf[msg_idx++] = c;
    }

    __bic_SR_register_on_exit(LPM0_bits); // wake the cpu up
}



void handller_message(void)
{
```

מהלך האלגוריתם

אנחנו מקבלים אינטרפט על ידי המחשב ששולח לנו את הספרה 7, api_handle_msg מעביר אותנו למצב 7 כאשר במצב הזה אנחנו מחכים לקבל את התווים של המחרוזת ועד שאנחנו לא מקבלים את התו סיום אנחנו מכניסים את כל התווים לבאפר.
לאחר שקיבלנו את כל התווים אנחנו מעתיקים את המידע בעזרת $strcpy$ אל תוך המשתנה $dataString$ ולאחר מכן אנחנו מדפיסים אותו.