



Wydział Elektryczny

POLITECHNIKA WARSZAWSKA

Technika mikroprocesorowa i systemy wbudowane

Wykorzystując mikrokontroler MSP430F169 oraz moduł GPS NEO-6M napisać program wyświetlający na wyświetlaczu LCD dane pobrane z modułu GPS

Zespół: **Bernard Kościewicz, Michał Wołowicz, Kamil Cuber**

Data wykonania i oddania: 15.06.2023

Prowadzący: **mgr inż. Grzegorz Wrona**

Ocena:

Opis programu

Konfiguracja

Pierwszym zadaniem było nawiązanie połączenia Moduł GPS - MSP430f169, więc uruchomiliśmy na płytce protokół komunikacyjny UART na mikro-BUS-1. Z pomocą prowadzącego skonfigurowaliśmy wejścia i wyjścia multipleksera, a następnie włączliśmy moduły przesyłu i odczytu. W końcu ustaliliśmy baud rate mikrokontrolera, na identyczny jak modułu GPS; na podstawie odczytów z oscyloskopu. W celu późniejszego odbioru danych, włączliśmy przerwania protokołu UART.

```
int main(void) {
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT P2DIR = 0xFF;
    P3SEL |= 0xC0; // P3.6,7 = USART1
    P3DIR |= 0x01; // P3.0 wyjście do MUXa
    P3OUT_bit.P0 = 0x00;
    P4DIR |= 0x80;
    P4OUT &= ~0x80; // P4.0 wyjście do MUXa

    ME2 |= UTXE1 + URXE1; // włączenie TXD/RXD
    UCTL1 |= CHAR; // 8-bit

    UTCTL1 |= SSEL1; // UCLK = SMCLK
    UBR01 = 0x50;
    UBR11 = 0x00;
    UCTL1 &= ~SWRST;
    IE2 |= URXIE1; // przerwania od uart
    __bis_SR_register(GIE); // ustawienie GIE
```

Rysunek 1: Fragment kodu odpowiedzialny za konfigurację

Odczyt i przetworzenie danych

W trakcie przerwania od UART, elementy odebranej od GPS ramki zostają przekazane do listy. Gdy lista się zapełniła uruchamiane są funkcje przetwarzania danych.

```
#pragma vector=USART1RX_VECTOR
_interrupt void usart1_rx (void)
{
    char data = RXBUF1; // get received data
    static int t=0;
    rdbuf1[t]=data;
    t++;
    if(t>=256) t=0;
    rdbuf1_length++; // increment length
    while (!(IFG2 & UTXIFG1)); // wait until buffer is empty
    TXBUF1 = 0x55;
    if(rdbuf1_length == MAX_BUFFER_SIZE - 1) { // if buffer is full, process data
        processReceivedData();
        transferData(); // Copy data to new_array
        rdbuf1_length = 0; // reset buffer length
    }
}
```

Rysunek 2: Wektor przerwania UART

Moduł GPS przesyłał nam ramkę danych, według standaru NMEA, w postaci znaków ASCII. Nasz kod miał wykryć sekwencję rozpoczętą "\$GPGGA", po czym zaczynał zczytywać odpowiednie dane do zmiennych.

```
void transferData() {
    for(int i = 0; i < MAX_BUFFER_SIZE; i++) {
        xtract[i] = rdbuf1[i];
    }
}

void parseGGA() {
    // Find the GGA sentence in the buffer
    char* ggaStart = strstr((const char*)xtract, "$GPGGA");

    if (ggaStart != NULL) {
        char term[MAX_BUFFER_SIZE];
        char NS[1], EW[1];

        sscanf(ggaStart, "$GPGGA,%9[^,],%14[^,],%1[^,],%14[^,],%1[^,]", &term, &latitude, &NS, &longitude, &EW);

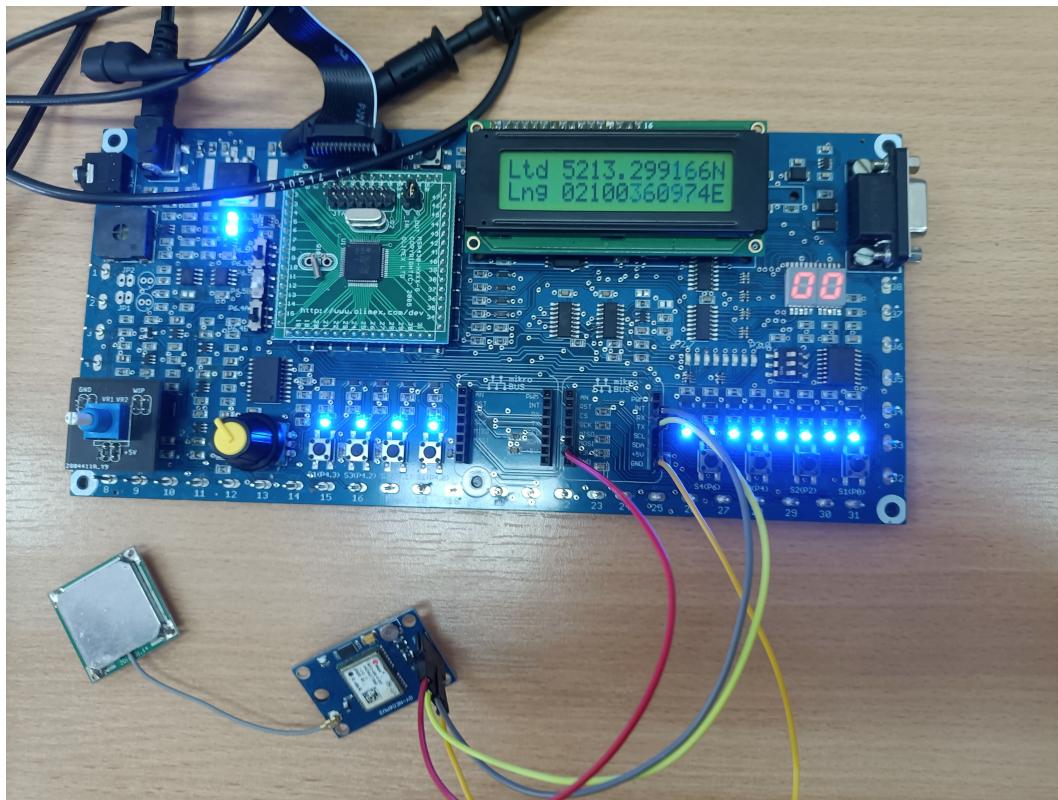
        lcd_go(1,4);
        lcd_write(latitude);
        lcd_go(1,15);
        lcd_write(NS);
        lcd_go(2,4);
        lcd_write(longitude);
        lcd_go(2,15);
        lcd_write(EW);
    }
}

void processReceivedData() {
    // Null-terminate the received data for string functions
    rdbuf1[rdbuf1_length] = '\0';
    // Parse the GPS data
    parseGGA();
}
```

Rysunek 3: Funkcje przetwarzające dane i wyświetlające dane na LCD

Wyświetlenie na LCD

Po przygotowaniu danych oraz implementacji funkcji inicjalizującej LCD, utworzonej przez prowadzącego przekazujemy dane do funkcji wyświetlającej je na LCD.



Rysunek 4: Wynik na ekranie LCD

Wnioski

Główna trudnością była synchronizacja baud rate mikrokontrolera z modułem GPS, bo w rzeczywistości nie pokrywała się z wartościami wyznaczonymi analitycznie. Cel osiągneliśmy dzięki pomiarom z oscyloskopu i ręcznym dobraniem właściwej przepustowości danych.

Kolejnym wyzwaniem był właściwy odczyt danych z listy - przypisanie do zmiennych odpowiednich wyrażeń i pominięcie zbędnych danych oraz znaków interpunkcyjnych.