POLITECHNIKA POZNAŃSKA

WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI

INSTYTUT ROBOTYKI I INTELIGENCJI MASZYNOWEJ

ZAKŁAD STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

SPRAWOZDANIE

SYSTEMY MIKROPROCESOROWE (LABORATORIUM)
[WARIE_2021-22_AIR_DZ_1_5_D_LUCZAK_21/22]

INTERFEJS KOMUNIKACYJNY I2C
(CYFROWY CZUJNIK NATĘŻENIA ŚWIATŁA)

(TEMAT ZAJĘĆ)

KAROL DĘBSKI

(AUTOR I: KAROL.DEBSKI@STUDENT.PUT.POZNAN.PL)

FORMA ZAJĘĆ: LABORATORIUM

PROWADZĄCY:

DR INŻ. DOMINIK ŁUCZAK

DOMINIK.LUCZAK@PUT.POZNAN.PL

POZNAŃ *15-11-2021* 9-45

(*DATA I GODZINA ZAJĘĆ*)

Spis treści

Zadanie #1

1.1 Specyfikacja

Program co jedną sekundę dokonuje odczytu z czujnika BH1750. Do weryfikacji poprawnego działania programu zostanie użyty SWV.

1.2 Implementacja

Listing 1 Funkcja inicjalizująca pracę czujnika BH1750

```
void BH1750_Init()
{
          uint8 t command;
          command=BH1750_POWER_ON;
          HAL_I2C_Master_Transmit(&hi2c3, BH1750_ADDRESS, &command, 1, 1000);
          command=BH1750_CONTINOUS_H_RES_MODE;
          HAL I2C Master Transmit(&hi2c3, BH1750 ADDRESS, &command, 1, 1000);
          BH1750 TypeDef BH1750 sensor={.interface type="i2c",.mode="CH1"};
}
```

Listing 2 Funkcja dokonująca odczytu z czujnika

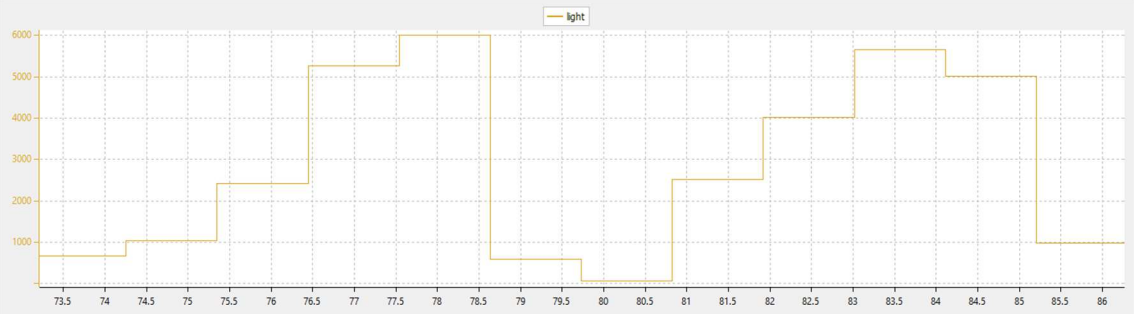
```
float BH1750_Read()
{
          float light = 0.0;
          uint8_t sample_array[2];
          HAL_I2C_Master_Receive(&hi2c3, BH1750_ADDRESS, sample_array, 2, 1000);
          uint16 t sample=0;
          sample=sample array[0];
          sample=sample<<8;
          sample=sample|sample_array[1];
          light=((float)sample)/1.2;
          return light;
}
```

Listing 3 Uchwyt do czujnika

```
typedef struct{
          char interface_type[3];
          char mode[3];
}BH1750_TypeDef;
```

Praca czujnika jest inicjowana poprzez włączenie go i ustawienie trybu pracy. Odbywa się to przez przesłanie komend na wskazany adres. Funkcja dokonująca odczytu odbiera dwa bajty danych z czujnika, przekształca je według wzoru a następnie zwraca wartość natężenia światła w lux'ach.

1.3 Wynik testów



Rys. 1 Podgląd SWV

1.4 Wnioski

Program działa poprawnie, kierując źródło światła na czujnik zwiększa się wartość zmiennej light co można zaobserwować na wykresie.

Zadanie #2

1.1 Specyfikacja

Program wysyła aktualne natężenie światłą padające na czujnik co 1 sekundę. Do zweryfikowania poprawnego działania programu zostanie użyty terminal na PC.

1.2 Implementacja

Listing 4 Funkcja zwrotna dla TIM10

```
void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim){
        if(htim->Instance == TIM10){
                light=BH1750_Read();
                sprintf((uint8_t *)msg, "%0.2f\r\n",light);
                start_send=1;
        }
}
```

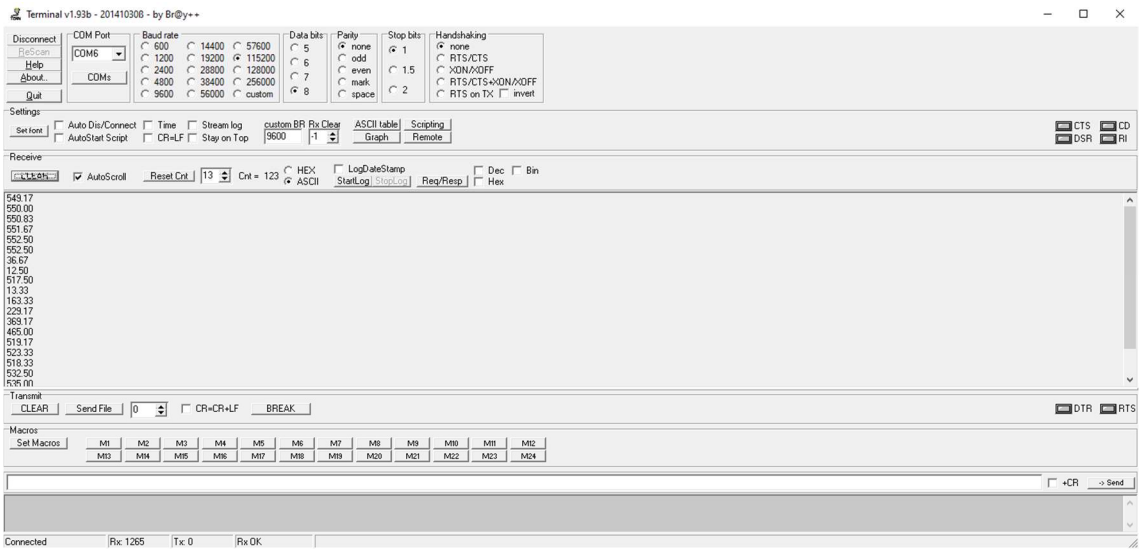
Listing 5 Główna pętla programu

```
while (1)
  {
  if(start_send){
        HAL_UART_Transmit(&huart2, msg, strlen(msg),500);
        start_send=0;
  }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
```

Po nastąpieniu przerwania na timerze 10 zostaje wyliczone natężenie światła padające na czujnik. Obliczone natężenie jest zamieniane na tekst a także ustawiana jest zmienna zezwalająca na wysłanie natężenia światła przez UART w głównej pętli programu.

1.3 Wynik testów



Rys. 2 Podgląd terminalu na PC

1.4 Wnioski

Program działa poprawnie wysyłając natężenie światła do terminalu na PC

1.5 Pytania

W jakim zakresie zmienia się wynik pomiaru? – Zmienia się od 1 do 65535 lux'ów.

Jakimi środkami możemy wpłynąć na wynik pomiaru? – Naświetlając czujnik.

Zadanie #3

1.1 Specyfikacja

Program wysyła do terminalu na PC wartość natężenia światła padającego na czujnik. W trakcie działania programu jest możliwa zmiana odświeżania wysyłania odczytu. Do weryfikacji działania programu zostanie zamieszczony link do materiału video.

1.2 Implementacja

Listing 6 Funkcja zwrotna dla przerwania na UART2

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
        if(huart->Instance==USART2){
                sscanf(msg2,"%5c%d",&tmp string,&tmp freq);
                if(!strcmp(tmp_string,"freq=") && tmp_freq>0 && tmp_freq<10){
                        ARR=(10000/tmp freq)-1;
                }
                HAL_UART_Receive_IT(&huart2, msg2, 6);
        }
}
```

Listing 7 Główna pętla programu

```
  while (1)
  {
          HAL TIM SET AUTORELOAD(&htim10, ARR);
        if(start_send){
                HAL_UART_Transmit(&huart2, msg, strlen(msg),500);
                start_send=0;
        }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
```

Po wykryciu przerwania na USART2 analizowane są przyjęte dane (msg2). Jeśli dane to „freq=” plus liczba od 1 do 9 to ustawiana jest zmienna ARR by na jej podstawie tak zmodyfikować rejestr AutoReload timera by ten pracował z określoną częstotliwością.

1.3 Wynik testów

https://drive.google.com/file/d/1_3sTvDIh6si4ceMJrBgum6xDuxivZTyH/view?usp=sharing

(23 sekundy)

1.4 Wnioski

Na materiale video widać, że po wysłaniu komendy, zmieniana jest częstotliwość odświeżania wysyłania komunikatów.

Zadanie #4

1.1 Specyfikacja

Program na komendę „print_on” zaczyna cyklicznie wysyłać na terminal pomiar z czujnika. Po otrzymaniu „print_off” zaprzestaje wysyłania pomiaru z czujnika. Gdy dostanie komendę „logger=*n*” zaczyna zbierać tyle odczytów z czujnika ile wynosi liczba. Formatuje je do formatu JSON i wysyła na terminal. Do weryfikacji poprawnego działania programu zostanie nagrany ekran terminala na PC.

1.2 Implementacja

Listing 7 Funkcja zwrotna dla USART2

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
          if(huart->Instance==USART2){
                    if(bufforRx[bufRxIndex]== 0x0D){
                              if(!tooLongData){
                                        dataReceived=1;
                              }
                    while(bufRxIndex<11 && !tooLongData
                              bufforRx[bufRxIndex]=0;
                              bufRxIndex++;
                    }
                    bufRxIndex=0;
                    tooLongData=0;
                    }else{
                              bufRxIndex++;
                              if(bufRxIndex>10){
                                        bufRxIndex=0;
                                        tooLongData=1;
                              }
                    }
          HAL_UART_Receive_IT(&huart2, &bufforRx[bufRxIndex], 1);
          }
}
```

Listing 8 Główna pętla programu

```
while (1)
  {
          if(dataReceived){
                    strncpy(tmp string,"print on",20);
                    print_on=!strcmp(bufforRx,tmp_string);
                    strncpy(tmp_string,"print_off",20);
                    print off=!strcmp(bufforRx,tmp string);
                    memset(tmp string, 0, 20);
                    collectData=sscanf(bufforRx,"%7c%d",&tmp string,&tmp logger);
                    if(!strcmp(tmp_string,"logger=") && tmp_logger>1 && tmp_logger<101 &&
collectData){
                              for(uint16_t i=0;i<tmp_logger;i++){
                                        light=BH1750 Read();
                                        data[i]=light;
                              }

                              memset(json_data, 0, 1000);
                              strcat(json_data,"[");
                              for(uint16_t i=0;i<tmp_logger;i++){
                                        snprintf(tmp, 10, "%.2f,", data[i]);
                                        if(tmp_logger==i+1){
                                                  snprintf(tmp, 10, "%.2f", data[i]);
                                                  strcat(json_data,tmp);
                                        }else{
                                                  snprintf(tmp, 10, "%.2f,", data[i]);
                                                  strcat(json data,tmp);
                                        }
                              }
                              strcat(json_data,"]\r\n");
                              HAL UART Transmit(&huart2, json data,
strlen(json data),1000);
                    }
                    dataReceived=0;
          }
          if(print_on){
                    light=BH1750_Read();
```

```
                sprintf(msg, "%0.2f\r\n",light);
                HAL_UART_Transmit(&huart2, msg, strlen(msg),500);
                HAL_Delay(500);
        }
        if(print_off){
                print_on=0;
        }
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
```

Gdy komenda tekstowa zaczyna docierać do USART2, wywoływany jest callback. Odczytuje on znak po znaku nadchodzącą komendę. Gdy wszystkie dane zostaną zebrane i są zgodne z założeniami zmienna dataReceived jest ustawiana na 1. W głównej pętli programu najpierw analizowane są dane które przeszły przez USART2. Jeśli są to komendy „print_on" lub „print_off" to ustawiane są odpowiedni zmienne które będą pozwalały na wykonanie funkcjonalności związanej z tymi komendami. Jeżeli zostanie odebrana komenda „logger=*n*" zostanie uruchomiony proces zbierania *n* odczytów z czujnika a następnie sformatowania ich do formatu JSON. Gotowy tekst JSON zostaje wysłany na terminal.

1.3 Wynik testów

https://drive.google.com/file/d/1dfk35T64d-4TCTkDfEqutYoklNMOCJIV/view?usp=sharing

(24 sekundy)

1.4 Wnioski

Program działa zgodnie z założeniami.

Zadanie #5

1.1 Specyfikacja

Program zmienia wypełnienie sygnału PWM docierającego do LEDa za pomocą komend odbieranych z terminala. Do weryfikacji poprawnego działania programu zostanie zamieszczony materiał video.

1.2 Implementacja

Listing 9 Główna pętla programu

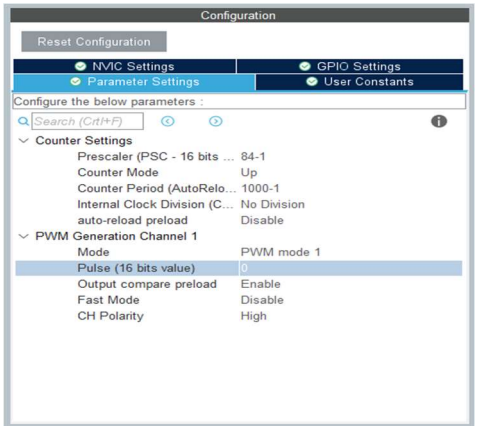
```
while (1)
  {
            if(dataReceived){
                      sscanf(bufforRx,"%5c%f",&tmp_string,&tmp_float);
                      if(!strcmp(tmp_string,"PWM1=") && tmp_float>=1 && tmp_float<=100){
                                tmp_int=(int)(tmp_float*10);
                                __HAL_TIM_SET_COMPARE(&htim10,TIM_CHANNEL_1,tmp_int);
                      }
                      dataReceived=0;
            }
            light=BH1750_Read();
            sprintf(msg, "%0.2f\r\n",light);
            HAL_UART_Transmit(&huart2, msg, strlen(msg),500);
            HAL_Delay(1000);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
```

Listing 10 Funkcja zwrotna dla przerwania na USART

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
          if(huart->Instance==USART2){
                    if(bufforRx[bufRxIndex]== 0x0D){
                              if(!tooLongData){
                                        dataReceived=1;
                              }
                    while(bufRxIndex<11 && !tooLongData){//wyzerowanie konca bufor w
przypadku krotszej komendy
                              bufforRx[bufRxIndex]=0;
                              bufRxIndex++;
                    }
                    bufRxIndex=0;
                    tooLongData=0;
                    }else{
                              bufRxIndex++;
                              if(bufRxIndex>10){
                                        bufRxIndex=0;
                                        tooLongData=1;
                              }
                    }
          HAL_UART_Receive_IT(&huart2, &bufforRx[bufRxIndex], 1);
          }

}
```



Rys 3 Konfiguracja timera

Gdy komenda tekstowa zaczyna docierać do USART2, wywoływany jest callback. Odczytuje on znak po znaku nadchodzącą komendę. Gdy wszystkie dane zostaną zebrane i są zgodne z założeniami zmienna dataReceived jest ustawiana na 1. W pętli głównej jeśli zmienna dataReceived jest równa 1 to przystępuje się do analizy odebranych danych. Jeśli mają one format „PWM1=*n*" to zmieniany jest rejestr ARR timera by ustawić wypełnienie PWM zgodnie z liczbą „n". Następnie dokonywany jest odczyt z czujnika i jest on wysyłany na terminal.

1.3 Wynik testów

https://drive.google.com/file/d/1_wWB1r61ay7pLPoN_aFT9oOYnKL8_qRg/view?usp=sharing

(25 sekund)

1.4 Wnioski

Program działa poprawnie, reaguje na komendy.

1.5 Pytania

W jakim zakresie zmienia się wynik pomiaru?

- Od 30 do 300 lux'ów dla danych warunków podczas nagrywania video.

Podsumowanie

Zadanie 4 było bardzo czasochłonne.