Design laboratory

Project - Weather station

Light intensity and UART communication

Autors:
Michał Hasior
Mariusz Więcławek

Lecturer: dr inż. **Jacek Stępień**

1. Ligth sensor

To measure light intensity we used an ADC converter built in the evaluation board “ATmega328PB XPLAINED mini”, then we need to rescale information. This operation mainly depends on the two parameters. First is the resolution of ADC, second is a reference voltage(V_{ref}). In our project, we have 10-bit ADC and V_{ref} = 5V.

Below are some code fragments with their detailed description:

```
89          /******************** ADC2 LIGHT INTENSITY MEASUREMENT**********************/
90          // ADC2 LIGHT INTENSITY MEASUREMENT
91          meas2 = ADC_MEASURE(PORTC2);
92
93          // ADC2 LIGHT INTENSITY AS A VOLTAGE
94          meas2 = (5 * meas2) / 1024;         // We need to rescale the result because we have 10bit-ADC and Vref=5V
95          meas2 *= 100;                         // we need to change the number format from X,XX  to  XXX
96          div_t divmod2 = div(meas2,100);     // Function div() write to structure div_t an integer part and fractional part
97
98          //WRITE MEASUREMENTS TO STRUCTURE       //DATA STRUCTURE DEFINE IN EEPROM.h
99          Data_Measurements.I_sensor2 = divmod2.quot;         // write integer part
100         Data_Measurements.D_sensor2 = divmod2.rem;          // write fractional part
101
```

Photo no. 1 - part od code to the measurement of light intensity (source: own code in file main.c)

```
69  /** Result type for function div(). */
70  typedef struct {
71      int quot;                       /**< The Quotient. */
72      int rem;                        /**< The Remainder. */
73  } div_t;
```

Photo no. 2 - definition of the structure div_t (source: standard library stdlib.h)

```
14
15  void ADC_INIT_AVCC(void)
16  {
17      ADCSRA |= (1<<ADEN); //ADEN: ADC Enable (starting the ADC)
18
19      //ADPS2:0: set prescaler, prescale= 128 // We need frequency in a good interval which we can find in the datesheet.
20      ADCSRA |= (1<<ADPS0)|(1<<ADPS1)|(1<<ADPS2);
21
22      // AVCC = 5V  -> it will be our reference voltage
23      ADMUX  |=  (1<<REFS0); //choose reference voltage
24  }
25
26  uint16_t ADC_MEASURE(uint8_t channel)
27  {
28      ADMUX = (ADMUX & 0xF8) | channel;       //choose channel
29      ADCSRA |= (1<<ADSC);    //start a single convertion
30      while(ADCSRA & (1<<ADSC)); //wait to end of convertion
31      return ADC;
32  }
```

Photo nr. 3 - functions for using the ADC (source: own code in file adc.c)

2. USART

We use the UART port to display data on the terminal in the computer or on the phone using the HC-06 bluetooth module. To initialize the USART, we need to determine the baud rate and use the formula below to calculate the UBRR parameters:

```
23  #define F_CPU 16000000UL // Clock Speed
24  #define BAUD 9600
25  #define MYUBRR F_CPU/16/BAUD-1
```

Photo no. 4 - part od code to calculate UBRR parameter (source: own code in file main.c)

Then use the calculated value in the USART initialization function:

```
16  void USART_Init( unsigned int ubrr) // Initialization UART
17  {
18      UBRR0H = (unsigned char)(ubrr>>8); //Set baud rate
19      UBRR0L = (unsigned char)ubrr;
20
21      UCSR0B = (1<<RXEN0)|(1<<TXEN0); // Enable receiver and transmitter
22
23      UCSR0C = (3<<UCSZ00); // Set frame format: 8data, 1stop bit
24  }
```

Photo no. 5 - part od code to initialize USART (source: own code in file USART.c)

```
26  void USART_PutC( char data ) // function that sends a single character
27  {
28      while ( !( UCSR0A & (1<<UDRE0)) ); // Wait for empty transmit buffer
29      UDR0 = data; // Put data into buffer, sends the data
30  }
```

Photo no. 6 - part od code to send a single character (source: own code in file USART.c)

```
32  void USART_PutS( char * s) // function that sends a string
33  {
34      while( *s ) USART_PutC( *s++ );
35  }
```

Photo no. 7 - part od code to send a string (source: own code in file USART.c)

```
37  void USART_PutInt(uint16_t number, uint8_t radix) // send the number and specify what format we want (bin/dec/hex)
38  {
39      char buf[17];
40      itoa(number,buf,radix); //bin/dec/hex
41      USART_PutS(buf);
42  }
```

Photo no. 8 - part od code to send the integer in chosen format (source: own code in file USART.c)

```
44  void USART_PutS_P(const char *s)    //sends string from RAM to UART
45  {
46      register char c;
47      while((c = pgm_read_byte(s++) )) USART_PutC(c);
48  }
```

Photo no. 9 - part od code to send string from RAM to UART (source: own code in file USART.c)

```
50  unsigned char USART_Receive( void ) // get and return received data from buffer
51  {
52      while ( !(UCSR0A & (1<<RXC0)) );    // Wait for data to be received
53      return UDR0;
54  }
```

Photo no. 10 - part od code to return received data from buffer (source: own code in file USART.c)

3. Testing

n	Humidity [%]	Temperature [°]	RAIN_VOLTAGE [V]	LIGHT_VOLTAGE [V]
1	46.0	23.1	4.34	0.36
2	46.0	23.1	3.71	0.35
3	46.0	23.1	4.26	1.84
4	46.0	23.1	1.48	4.83
5	46.0	23.2	2.66	0.53
6	46.0	23.2	2.95	1.6
7	46.0	23.4	3.33	1.21
8	46.0	23.4	3.66	0.30

Photo no. 11 - display information on the terminal (source: own photo)

We used a flashlight and examined the behaviour of the light sensor.