UPPSALA
UNIVERSITET

Microcontroller Programming

Report for 1TE663

Weather Station Project (Group 13)

Mohammad El Musleh

March 26, 2019

Contents

1 Introduction

The purpose of the project was to to design and build a weather station which can show temperature and humidity values on any Bluetooth device by using DHT11 (digital temperature and humidity sensor) with 8-bit micro-controller ATmega328p to implement a weather station system that display the temperature degree and humidity percentage through a slave Bluetooth module to a master Bluetooth (can be laptop or smartphone), as well as 3 LED's with different color and Buzzer that can be controlled by the micro-controller through Bluetooth.

2 Implementation

The DHT11 sensor used to get the current temperature and humidity values in the surrounding air. The Bluetooth module (as Slave) used to transmit and receive data (USART) from the microcontroller ATmega328P to Bluetooth master module after the micro-controller receives specific value defined to transmit the value of the temperature or humidity or turn on/off the LED'S or making buzzer beeps for short time.

2.1 Hardware

In this project few hardware components used, they are as follow: micro-controller ATmega328P (running at 1MHz), Bluetooth HC-05 (it operates on 5V, clock speed 1843200), DHT11, Buzzer, LED's (Green, Blue, & Red), 1kΩ resistor (x3), USB AVR Programmer, Jumper wires and Breadboard. The components and the connection are as shown in the Figure 1 below.

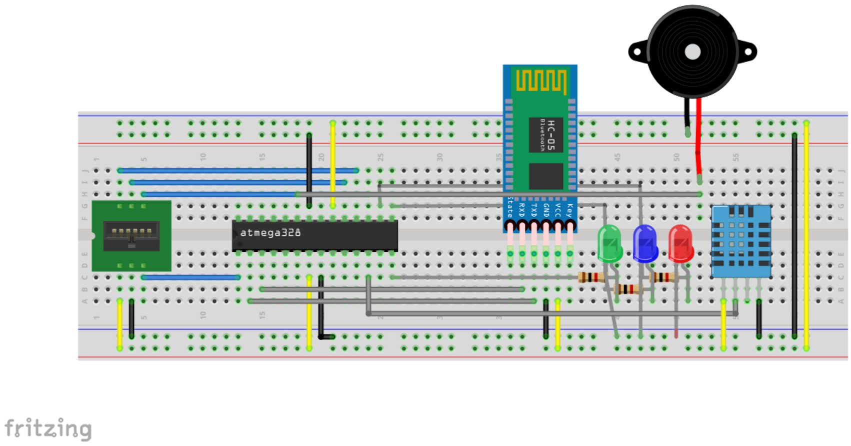

Figure 1: The 2D view of the final implementation by Fritzing program

The black wires are the GND, while yellow wires are the VCC, the blue jumper wire is the connection to program the ATmega328p by the USBASP AVR Programmer USB while the gray jumper wire for signal pins for the other hardware component.

The connection TX goes to RX and RX to TX between the Bluetooth module pins with ATmega328p respectively. The LED's green, blue, red connected to B0, B1, and B2 in micro-controller with a 1KΩ resistor between each connection. The signal pin connection of DHT11 sensor to digital pin D6 and buzzer to C0 pin in ATmega328p.

2.2 Software

The project consists of 4 files (main code, common library, beeps for the buzzer, USART for transmitting and receiving data), the common.h used to apply common call library and define USART as well between the other directory (As shown in the Appendix common.h).

The USART is the same file used in Lab 6 but with some modification. the beeps file responsible for the output sound from the buzzer as tones this done my making many functions to turn on then off while having a delay in micro-second (As shown in the Appendix bees_my_music.h).

All of these files called and get executed in the main file, the main file configured the DHT11 sensor and beeps function for buzzer, the main function implemented as follow first define the pins for LED'S and buzzer as output, calling Bluetooth function, and transmitting short message "Hello, Welcome :)" and short beeps sound from buzzer before starting the while loop.

In the while loop, the ATmega328p wait for receiving a message from Bluetooth and compare the value received in the if else statement condition, for LED'S the value 1, 2, and 3 pressed will turn on the LED light if it pressed again it will turn off. While value 4 and 5 responsible for temperature in Celsius and Fahrenheit and humidity in percentage values. While the last defined value is 6 responsible for giving the beeps song for the buzzer, and the loop keeps repeating with 100 ms delay every time. the main code process is exactly as shown in the Figure 2.

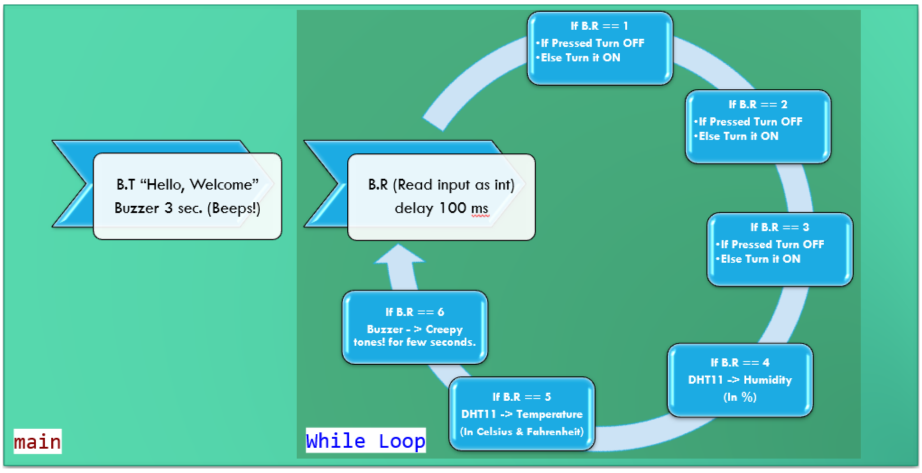

Figure 2: Basic code structure of the program

3 Discussion

The DHT11 sensor was the most challenging part since during debugging notice that the only way to get correct value only after making delay 1 second between each result. In addition, making a nice tone by the buzzer is quite hard as well since any slight change in delay may responsible for a totally new different tone. However, the end result was quite satisfactory as the values are correct and output perform was fast enough.

4 Further improvement

A better user interface would be more useful to show all value instead of making each one alone. Also, add more weather station sensor to more information about the weather forecast (Like Wind speed, a direction sensor, Rain speed sensor, air pressure sensor, light intensive sensor, etc). Adding LCD or Nokia LCD will make it more efficient since no need to keep tracking if from mobile a quick look at the LCD will be enough. Connect the microcontroller to the PC and make API that will generate the value of sensors to an online website, that can access any time. A new 3D printed enclosure will be a perfect touch to end with since it could protect the board from physical damage and not reconnecting a jumper wire every time after moving it. With rechargeable batteries or power bank connected to will make it a truly wireless mobile device (completely wireless).

It's was such a great experience and a lot of fun project. I have learned quite a lot of stuff and became more familiar with a microcontroller and this definitely will not be my last project in the ATmega microcontroller.

5 References

1. Sending Averaged ADC Sample Over USART. (see `http://microchipdeveloper.com/8avr:avg-adc-over-usart`)

2. DHT11 Sensor Interfacing with AVR ATmega16/ATmega32.(see `http://electronicwings.com/avr-ATmega/dht11-sensor-interfacing-with-ATmega16-32`)

3. HC-05 Bluetooth Module Interfacing with AVR ATmega32 (see `https://electronicwings.com/avr-ATmega/hc-05-bluetooth-module-interfacing-with-ATmega1632`)

Appendix

main.c Main file code:

```c
/*
ATmega328|P

 Reset  PC6|1   28|PC5
 B.RX   PD0|2   27|PC4
 B.TX   PD1|3   26|PC3
        PD2|4   25|PC2
        PD3|5   24|PC1
        PD4|6   23|PC0        Buzzer
VCC     Vcc|7   22|Gnd        GND
GND     Gnd|8   21|Aref
        PB6|9   20|AVcc       VCC
        PB7|10  19|PB5        SCK
        PD5|11  18|PB4        MISO
DHT11   PD6|12  17|PB3        MOSI
        PD7|13  16|PB2        LED
LED     PB0|14  15|PB1        LED
*/

#include "common.h"
#include "HAL_USART.h"
#include "bees_my_music.h"

// DHT11 global variable
#define DHT11_PIN PIND6
uint8_t c=0,I_RH,D_RH,I_Temp,D_Temp,CheckSum;

char start_wel[]= "Hello, Welcome :)";
char error[]= "ERROR";
char Hum_pt[]= "Humidity = ";
char Temp_pt[]= "Temperature = ";
char Hum_siu[]= " %";
char Temp_c_siu[]= " C";    //Celsius
char Temp_f_siu[]= " F";    //Fahrenheit

bool pressed_f1 = false, pressed_f2 = false, pressed_f3 = false;

/*----------DHT11/Begin----------*/
void Request(){             //send start pulse/request
    DDRD |= (1<<DHT11_PIN);
    PORTD |= (1<<DHT11_PIN);    /* set to on pin */
    _delay_ms(1000);
    PORTD &= ~(1<<DHT11_PIN);   /* set to low pin */
    _delay_ms(20);              /* wait for 20ms */
    PORTD |= (1<<DHT11_PIN);    /* set to high pin */
}

void Response(){            //receive response from DHT11
    DDRD &= ~(1<<DHT11_PIN);
    while(PIND & (1<<DHT11_PIN));
    while((PIND & (1<<DHT11_PIN))==0);
    while(PIND & (1<<DHT11_PIN));
}

uint8_t Receive_data(){     //receive data
    for (int q=0; q<8; q++){
        while((PIND & (1<<DHT11_PIN)) == 0);    /* check received bit 0 or 1
             */
        _delay_us(30);
        if(PIND & (1<<DHT11_PIN))               /* if high pulse is greater
             than 30ms */
        c = (c<<1)|(0x01);                      /* then its logic HIGH */
        else                                    /* otherwise its logic LOW */
        c = (c<<1);
        while(PIND & (1<<DHT11_PIN));
    }
    return c;
}
```

```
void DTH11_final_code(int Temp_or_hum){
    char data[5];
    Request();              /* send start pulse */
    Response();             /* receive response */
    I_RH=Receive_data();    /* store first eight bit in I_RH */
    D_RH=Receive_data();    /* store next eight bit in D_RH */
    I_Temp=Receive_data();  /* store next eight bit in I_Temp */
    D_Temp=Receive_data();  /* store next eight bit in D_Temp */
    CheckSum=Receive_data();/* store next eight bit in CheckSum */

    if ((I_RH + D_RH + I_Temp + D_Temp) != CheckSum)
    {
        Transmit_string(error);
        USART_SendLine();
    }

    else
    {
        //1 mean humidity in %
        if(Temp_or_hum == 1){
            itoa(I_RH,data,10); /* Integer to string conversion */
            Transmit_string(Hum_pt);
            Transmit_string(data);
            Transmit_string(".");
            itoa(D_RH,data,10);
            Transmit_string(data);
            Transmit_string(Hum_siu);
            USART_SendLine();
        }
        //2 mean temp. in C and F
        else if(Temp_or_hum == 2){
            itoa(I_Temp,data,10);
            Transmit_string(Temp_pt);
            Transmit_string(data);
            Transmit_string(".");
            itoa(D_Temp,data,10);
            Transmit_string(data);
            Transmit_string(Temp_c_siu);
            USART_SendLine();

            Transmit_string(Temp_pt);
            itoa(CheckSum,data,10);
            Transmit_string(data);
            Transmit_string(Temp_f_siu);
            USART_SendLine();
        }
    }
    _delay_ms(500);
}
/*----------DHT11/End----------*/

/*----------Buzzer/Begin----------*/
void mid_state(){
    beep_2();
    beep_7();
    beep_2();   //lazier fire
    beep_8();
    beep_1();   //IT exit
    beep_2();
    beep_7();
    beep_3();
    beep_5();
    beep_6();
    beep_1();
    beep_2();
    beep_7();
    beep_3();
    beep_5();
    beep_6();
    beep_2();
    beep_7();
    beep_2();
    beep_8();
}
```

```
void inc_state(){
    beep_1();
    beep_2();
    beep_3();
    beep_4();
}
void speaker_info(){
    DDRC |= (1 << PINC0);
    PORTC |= (1 << PINC0);
    _delay_ms(100);
    PORTC &= ~(1 << PINC0);
    inc_state();
}
/*----------Buzzer/End----------*/

int main(void){
    DDRB |= 1 << PINB0;         // pin 0 of PORTB as output
    DDRB |= 1 << PINB1;         // pin 1 of PORTB as output
    DDRB |= 1 << PINB2;         // pin 2 of PORTB as output
    USART_Begin();
    int Data_in;
    Transmit_string(start_wel);
    USART_SendLine();
    speaker_info();
    _delay_ms(100);
    while(1){
        Data_in= USART_Receive();
        _delay_ms(100);
        //USART_SendByte(Data_in);
        if(Data_in == '1'){
            if(pressed_f1 == false){        //if false not On
                Transmit_string("Green LED -> ON");
                PORTB |= 1 << PINB0;        // switch PB0 to 1
                pressed_f1 = true;
            }
            else {
                Transmit_string("Green LED -> OFF");
                PORTB &= ~(1 << PINB0);     // switch PB0 to OFF
                pressed_f1 = false;
            }
            USART_SendLine();
        }
        else if(Data_in == '2'){
            if(pressed_f2 == false){            //if false not On
                Transmit_string("Blue LED -> ON");
                PORTB |= 1 << PINB1;        // switch PB1 to 1
                pressed_f2 = true;
            }
            else {
                Transmit_string("Blue LED -> OFF");
                PORTB &= ~(1 << PINB1);     // switch PB1 to OFF
                pressed_f2 = false;
            }
            USART_SendLine();
        }
        else if(Data_in == '3'){
            if(pressed_f3 == false){        //if false not On
                Transmit_string("Red LED -> ON");
                PORTB |= 1 << PINB2;    // switch PB2 to 1
                pressed_f3 = true;
            }
            else {
                Transmit_string("Red LED -> OFF");
                PORTB &= ~(1 << PINB2);     // switch PB2 to OFF
                pressed_f3 = false;
            }
            USART_SendLine();
        }
        else if(Data_in == '4'){
            Transmit_string("|---- [Humidity] ----|");
            USART_SendLine();
            DTH11_final_code(1);
            _delay_ms(500);
        }
        else if(Data_in == '5'){
            Transmit_string("|---- [Temperature] ----|");
            USART_SendLine();
            DTH11_final_code(2);
            _delay_ms(500);
        }
        else if(Data_in == '6'){
            Transmit_string("|---- [Buzzer] ----|");
            USART_SendLine();
            Transmit_string("Enjoy!!!");
            mid_state();
            USART_SendLine();
            Transmit_string("Unfortunately, it's over now! play again!!");
        }
        _delay_ms(100);
    }
}
```

bees_my_music.h The tones for buzzer.

```
#ifndef BEES_MY_MUSIC_H_
#define BEES_MY_MUSIC_H_

void delay_us(int us){
    for (int i = 0; i < us; i++){
        _delay_us(1);
    }
}

void beep_main(){
    for (int i = 100; i < 200; i+=5) {
        PORTC = 0x01;
        delay_us(i);
        PORTC = 0x00;
        delay_us(500);
    }
}

void beep_1(){
    for (int i = 0; i < 100; i++) {
        PORTC = 0x01;
        delay_us(20);
        PORTC = 0x00;
        delay_us(980);
    }
}

void beep_2(){
    beep_main();
    _delay_ms(300);
    beep_main();
    _delay_ms(200);
    beep_main();
}

void beep_3(){
    _delay_ms(150);
    for (int i = 0; i < 100; i++) {
        PORTC = 0x01;
        delay_us(20);
        PORTC = 0x00;
        delay_us(980);
    }

    _delay_ms(100);

    for (int i = 0; i < 200; i++) {
        PORTC = 0x01;
        delay_us(67);
        PORTC = 0x00;
        delay_us(600);
    }
}

void beep_4(){
    _delay_ms(150);
    for (int i = 0; i < 100; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(200);
    }

    _delay_ms(100);

    for (int i = 0; i < 200; i++) {
        PORTC = 0x01;
        delay_us(97);
        PORTC = 0x00;
        delay_us(300);
    }
}

void beep_5(){
    _delay_ms(150);
    for (int i = 0; i < 100; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(i+30);
    }

    _delay_ms(100);

    for (int i = 0; i < 200; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(200-i);
    }
}

void beep_6(){
    _delay_ms(150);
    for (int i = 0; i < 100; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(i+40);
    }

    _delay_ms(100);

    for (int i = 0; i < 200; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(100+i);
    }
}

void beep_7(){
    _delay_ms(150);
    for (int i = 0; i < 100; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(300);
    }

    _delay_ms(100);

    for (int i = 0; i < 200; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(200);
    }
}
void beep_8(){
    _delay_ms(150);
    for (int i = 0; i < 100; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(600);
    }

    _delay_ms(100);

    for (int i = 0; i < 200; i++) {
        PORTC = 0x01;
        delay_us(1);
        PORTC = 0x00;
        delay_us(30);
    }
}
#endif /* BEES_MY_MUSIC_H_ */
```

common.h The common file for all files.

```
#ifndef COMMON_H_
#define COMMON_H_

#define F_CPU 1000000UL

#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>    //bool

/* Declaration for Bluetooth*/
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define BAUDRATE ((F_CPU)/(BAUD*8UL)-1) // set baud rate value for UBRR
#define MYUBRR FOSC/16/BAUD-1



#endif /* COMMON_H_ */
```

usart.c & usart.h For transmit and receiving data. (same from Lab 6 files)