Microprocessors and Peripherals 3rd Lab.

Driver implementation for DHT11 module on the STM32 nulceo microcontroller.

Authors: Konstantinidis Paschalis, Tzouvaras Evangelos.
Team: 5

Contents

Lab Description: 1
First Request: 1
Second Request: 2
Third Request: 3
Fourth, Fifth, Sixth Request: 3
Seventh Request: 4
Testing and problems: 5

Lab Description:

The third lab assignment requires for its implementation, the use of all the programming techniques we learned in the previous labs plus the use of timers. By combining the above we should write a driver in the Keil uVision environment for the DHT11 module and get some temperature measurements. The DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. Moreover, the lab requires the use of UART, the user button and the on board led.

First Request:

We should ask the user to give as his AEM through UART interface one time in the beginning of the program.

Main code:

```
133  int main(){
134      // UART Initialization
135      uart_init(115200);                                          // Set the baud rate
136      uart_enable();                                                // Enable the uart communication
137      uart_print("Please give me your AEM\n\n");                  // A message to user
138
139      uart_set_rx_callback(AEM_isr);              // Set the uart callback function
140
```

AEM_isr:

```
53   // ISR to process the user's AEM
54   void AEM_isr(uint8_t rx_data){
55       //Save the AEM until the enter character was recognized
56       if(rx_data != 10 && (rx_data >=48 && rx_data <=57)){
57           AEM[i] = rx_data;                                       // Store the next rx data
58           i++;                                                    // Go to the next index of the AEM array
59       }
60       else if(rx_data == 10){
61           i--;
62           AEM_sum = (AEM[i] - 48) + (AEM[i-1] - 48);              // Add the two last digits of the AEM
63           i = 0;                                                  // Reset the index array
64       }
65       else{
66           uart_print("Wrong AEM!!\n");
67           AEM_sum = 2;
68       }
69   }
70
```

Second Request:

We should read the temperature value that sensor gives us with a constant period of 2 seconds which we will determine with the use of a timer.

Main code:

```
147      // Timer init
148      timer_init(CPU_CLOCK);                                  // Set up the timer at 1sec
149      timer_enable();                                           // Enable the timer
150      timer_set_callback(timer_isr);
151
```

Timer_isr:

```
94   // ISR that the timer calls
95   void timer_isr(){
96       Temp_counter++;
97       Print_counter++;
98
99       // Read the temperature from the sensor
100      if(Temp_counter == Temp_counter_max){          //Temp_counter_max has been initialize to 2
101          temperature = read_Temp_DHT11();           // Read the temperature
102          Temp_counter = 0;                          // Reset the timer counter
103      }
104
```

Reading temperature value:

At this point we implemented the driver of the DHT11 module in order to get the measurement. More information can be found in the DHT_11.c and DHT_11.h files.

Third Request:

In this step with the use of an ISR we should print through UART the temperature and the sampling rate of the temperature sensor. We implemented the code inside the timer_isr.

Time_isr:

```
94   // ISR that the timer calls
95   void timer_isr(){
96       Temp_counter++;
97       Print_counter++;
98
99       // Read the temperature from the sensor
100      if(Temp_counter == Temp_counter_max){              //Temp_counter_max has been initialize to 2
101          temperature = read_Temp_DHT11();              // Read the temperature
102          Temp_counter = 0;                             // Reset the timer counter
103      }
104
105      // Print the temperature value
106      if(Print_counter == Print_counter_max){            //Print_counter_max has been initialize to 2
107          uart_print("Temperature: ");
108          sprintf(temp, "%d", temperature);
109          uart_print(temp);
110          uart_print(" oC");
111          uart_print("\n");
112          uart_print("Sensor sampling rate: 2sec\n");
113          Print_counter = 0;                            // Reset the print counter
114      }
115
```

Fourth, Fifth, Sixth Request:

After printing the temperature and the sampling rate we should control the on board led based on some conditions. If the temperature is above 25 Celsius an ISR will turn the led on. If the temperature is lower than 20 Celsius an ISR will turn the led off. Finally, if the temperature is between 20 – 25 Celsius the led will toggle per 1 second.

Main:

```
143      // Peripherals' init
144      LED_init();                                                    // Initialize the LED
```

Led_init:

```
32   // Function to initialize the LED state
33   void LED_init(){
34       gpio_set_mode(USER_LED, Output);
35       gpio_set(USER_LED, LED_OFF);
36   }
37
```

Inside the timer_isr when we get the temperature value, we control the led based the conditions.

```
116         // Control the LED based on the temp value
117         if(temperature > HIGH_TEMP)
118                 high_Temp();
119             else if(temperature < LOW_TEMP)
120                 low_Temp();
121             else
122                 med_Temp();
123
124     }
125
```

Above 25:

```
38  // ISR for temperatures grater than 25oC
39  void high_Temp(){
40      gpio_set(USER_LED, LED_ON);                 // Switch on the LED
41  }
42
```

Below 20:

```
48  //ISR for temperatures lower than 20oC
49  void low_Temp(){
50      gpio_set(USER_LED, LED_OFF);
51  }
52
```

Between 20 and 25:

```
43  // ISR for temperatures between 20 and 25oC
44  void med_Temp(){
45      gpio_toggle(USER_LED);
46  }
47
```

Seventh Request:

At this last point we should check if the button is pushed. If so an ISR will be called which will change the display rate of the temperature measurement in the serial communication.

- First time the button is pressed the display rate should change according to the sum of the last two digits of the AEM.

- After the first time, each time the button is pressed we should check if the time the button is pressed is odd or even. If it is odd the display period should be 3 seconds, else it should be 3 seconds.

Main:

```
145     Button_init();                                          // Initialize the button
146
```

Button_init:

```
126 //Function to initialize the button state
127 void Button_init(){
128     gpio_set_mode(USER_BUTTON, PullUp);
129     gpio_set_trigger(USER_BUTTON, Rising);
130     gpio_set_callback(USER_BUTTON, Button_isr);
131 }
132
```

Button_isr:

```
71  // ISR if the button was pressed
72  void Button_isr(){
73      button_pressed ++;                      // If the ISR was called, then means that the button was pressed
74      Print_counter = 0;                      // Reset the counter value
75
76      if(button_pressed == 1){                // At the first time that the button was pressed
77          Print_counter_max = AEM_sum;        // Now the timer period will depend from the AEM value
78      }
79      else{                                   // All the other times that the button was pressed
80          if(button_pressed % 2 == 0){        // If the button pressed count is even number
81              Print_counter_max = 4;
82          }
83          else{                               // Else if the button pressed count is odd number
84              Print_counter_max = 3;
85          }
86      }
87
88      uart_print("\nDisplay rate: ");
89      sprintf(display_rate, "%d", Print_counter_max);
90      uart_print(display_rate);
91      uart_print(" sec\n\n");
92  }
93
```

Testing and problems:

We had a minor difficulty finding the optimal way and time durations in order to be possible for us to read the waveform from the DHT_11 module. Once we got the sensor to operate smoothly, we then test our code implementation by creating the desired environment temperature. That was achieved by putting the sensor into ice and by putting it close to a lighter.