Electrical and Computer Engineering
Aristotle University Of Thessaloniki

Microprocessors and Peripherals
Report 3

Meliopoulos Marios #9680
Papadakis Konstantinos Fotios #10371
27/5/24

1.1 Assignment info

This assignment is implemented in a main.c file where all the necessary methods and processes take place, along with the instructors' complementary drivers(.h and .c files).

1.2 Problems

The temperature and humidity sensor was quite a complex peripheral to work with. Even though the timing and the information crucial to its functions were present inside the datasheet, the way of communicating it to the user was not very efficient compared to other methods such as pseudocode or any other practical way. Great help to us were previous attempts from other people online who have made this sensor work with different hardware in the past (mainly Arduino but also some stm32 microcontrollers).

atoi(buff)’s output included numbers that we deleted using backspace. They seem to persist despite our attempts to go back and override the previous buffer positions.

2.1 UART input

We initially modified the range of input types of characters that can be given as input to include numbers and all the other ASCII codes except letters and printable characters. On the bottom of the UART code, we also added a check for when no number was given. If the input is valid we add the buffer’s last two digits for when we need it later on.

2.2 DHT_11 sensor initialization

In order to receive information from our sensor MCU needs to send a start signal. First, we set the pin’s mode to output, we set the pin to 0, wait 18ms, set it to 1, wait 20us and finally, we set its mode to input.

2.3 DHT_11 check response

This is a signal sent by the sensor to make sure everything works well. We first wait 40us so that we can reach the middle of the 80us constant low voltage period. If the pin is 0 then we wait 80us to reach the middle of the next, this time, 80us constant high voltage period. If the pin is 1 then we have successfully sent a signal to the MCU. If not then something went wrong. Finally, we wait until the pin is 0 to continue with receiving the actual data.

2.4 DHT_11 read

The last step of this sensor's operation is receiving the temperature and humidity information. To do that we wait until the sensor's pin is 1, then we wait 40us and check the pin's value again. After checking the value we store it inside i and wait for the pin to go to 0. We have to traverse all 8 bits one by one and check if they are 0 or 1. When having done so, we return i.

2.5 Toggle LED

The function that toggles the LED simply toggles the value of the pin associated with the LED light and then calls a function that prints the current state of the LED.

2.6 Touch ISR

This is a very simple function that updates a public variable called touch_pressed, to become 1 every time the touch sensor gets activated. The button's behaviour is explained later on at the while(1) segment of the report.

2.7 Temperature and humidity sampling

We set up a timer which can only count up to 1 second. Thus to count intervals longer than 1 second we need to set up a global counter that gets updated every time our timer interrupt, which is set to count 1 second, activates. Every time it gets called we add 1 to the counter and when the intended ticking frequency and the counter are the same our code gets executed. That code fetches our sensor's readings and prints them for us along with the ticking speed. When we need to also print the humidity level we just have to set a public variable to 1 and that part of the code gets executed as well.

2.8 while(1)

The while loop we enter after all the prework we have done includes 2 parts. The first part checks the current temperature and configures the LED's behavior according to what was asked of us and the second part checks if the touch_pressed variable has become 1 which would mean that we have activated the touch sensor. If that is the case we take note of how many times it has been activated and configure the sampling frequency if humidity gets printed as instructed.