ΑΝΑΦΟΡΑ 3ΗΣ ΕΡΓΑΣΙΑΣ

Χρήστος Σιδηράς, ΑΕΜ: 9647
Παναγιώτα Χήτα, ΑΕΜ: 9747

ΕΙΣΑΓΩΓΗ

Στην τρίτη εργασία στο μάθημα Μικροεπεξεργαστές και Περιφερειακά δημιουργήθηκαν τέσσερα αρχεία, ένα με το κύριο κομμάτι του κώδικα, συμπεριλαμβανομένων και των συναρτήσεων που ζητούνται έτσι ώστε να διαβάζουμε τα δεδομένα θερμοκρασίας από τον αισθητήρα *dht11* σύμφωνα με το datasheet που δόθηκε και τα αρχεία *delay.c, delay.h, delay_cycles.s* που αξιοποιήθηκαν για τους timers,

Σχετικά με τα τελευταία αρχεία αξίζει να σημειωθεί ότι το delay_cycles.s περιέχει τον κώδικα σε arm ο οποίος ενσωματώνεται μέσα στον κώδικα του *delay.c*, με τον εξής τρόπο (χρησιμοποιήθηκε και στο πρώτο εργαστήριο)

```
extern void delay_cycles(unsigned int cycles);
```

και έχει αποκλειστικό στόχο να υλοποιήσει συναρτήσεις καθυστέρησης.

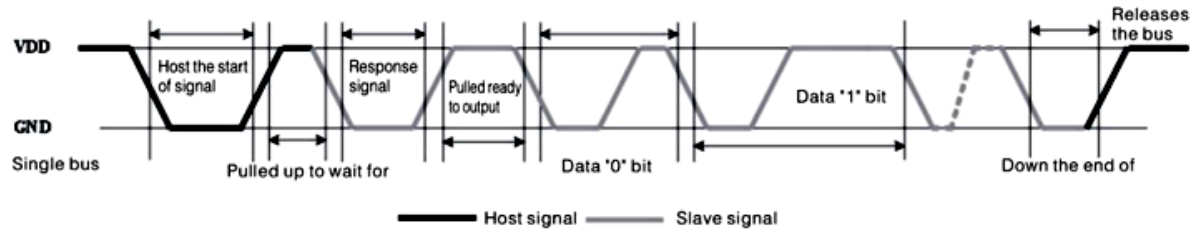
Functions for reading data from dht11

Προκειμένου να διαβαστούν τα δεδομένα από τον αισθητήρα, να σταλούν δηλαδή στον μικροελεγκτή μας έπρεπε να υλοποιηθούν οι παρακάτω συναρτήσεις βασιζόμενοι στο datasheet του dht11. Αναλυτικά:

```
void DHT11_init(void)
{
        gpio_set_mode(P_DHT, Input);
}
```

Η συνάρτηση αυτή καθορίζει την λειτουργία ενός ακροδέκτη gpio, το πρώτο όρισμα αντιπροσωπεύει ποιο ακροδέκτη gpio θα τον ορίσουμε σαν είσοδο - input, διότι θα χρησιμοποιηθεί για λήψη σημάτων/δεδομένων από τον αισθητήρα επιτρέποντας στον μικροελεγκτή να διαβάσει τα δεδομένα θερμοκρασίας που μεταδίδονται.

Όλες οι συναρτήσεις χτίστηκαν με βάση αυτό το μοντέλο και το κατάλληλο delay time που αναδεικνύονταν στο datasheet.



Αρχικά προετοιμάζει ο μικροελεγκτής το περιβάλλον έτσι ώστε να μπορεί να δεχθεί τα δεδομένα από τον αισθητήρα. Η συνάρτηση *DHT_start()* αρχικά στέλνει σήμα στο dht11, θέτει την τιμή του ακροδέκτη σε χαμηλή τιμή (0) για να ξεκινήσει το σήμα εκκίνησης. Καθυστερεί για περίπου 20 ms και ανεβάζει την τιμή του ακροδέκτη στο 1 περιμένοντας την απάντηση από τον αισθητήρα.
Στην συνέχεια η συνάρτηση *DHT_check_response(void)* μέσω τον κατάλληλων καθυστερήσεων ελέγχει αν έχει σταλεί κάποια απάντηση: if(!(gpio_get(P_DHT))), όταν το DATA Single-Bus βρίσκεται στο επίπεδο χαμηλής τάσης, αυτό σημαίνει ότι το DHT στέλνει την απάντηση σήμα.

Μόλις η DHT στείλει το σήμα απόκρισης, ανεβάζει την τάση και τη διατηρεί για 80us, προετοιμάζεται για τη μετάδοση δεδομένων, για αυτό αν περάσει στην if θα θέσουμε ότι η απάντηση είναι 1, άρα έχει σταλεί, αλλιώς -1.

```
delay_us(80);
if(gpio_get(P_DHT))
```

Περιμένει όσο η τιμή του ακροδέκτη είναι υψηλή, υποδεικνύοντας ότι το DHT11 εξακολουθεί να είναι απασχολημένο με την επεξεργασία του αιτήματος και στο τέλος επιστρέφει εάν υπάρχει απάντηση ή όχι. Μετά ακολουθεί η μετάδοση αυτών των δεδομένων.

```
delay_us(40);
if (!gpio_get(P_DHT))   // if the pin is low
{
      i&= ~(1<<(7-j));   // write 0 (26-28us)
}
else i|= (1<<(7-j));  // if the pin is high, write 1 (70us)
while (gpio_get(P_DHT));
```

Η συνάρτηση *DHT11_read()* αφού περιμένει τον ζητούμενο χρόνο περιμένοντας να πάει ο ακροδέκτης στο 1 και μετά στο 0, πρακτικά ξεκινά να διαβάζει τα δεδομένα. Το μήκος του χρόνου του επόμενου σήματος υψηλής στάθμης τάσης καθορίζει εάν το bit δεδομένων είναι "0" ή "1".

Τέλος, η συνάρτηση *DHT_GetTemperature(void)* καλεί την *DHT11_start()* έτσι ώστε να στείλει αίτημα για να ξεκινήσει να μετρά την θερμοκρασία, μετά την *DHT11_check_response* και μέσω της *DHT11_read* διαβαζει και αποθηκεύει σε μεταβλητές τα δεδομένα όπου ανά 8 bit αντιπροσωπεύουν την υγρασία(ακέραιο μέρος και δεκαδικό), θερμότητα(ακέραιο μέρος και δεκαδικό) και το τελευταίο για έλεγχο αν έχει γίνει σωστά η μεταφορά των δεδομένων.

```
if((Rh_1 + Rh_2 + Temp_1 + Temp_2) == sum_byte5) {
      return (float)(Temp_1 + (float)Temp_2/10.0);
}
else {
      return -1000.0; // for error
}
```

```
void toggle_led_isr(){
      do{
            int temporary_counter = counter;
            if (counter % 2){
                  gpio_toggle(P_LED_R);
                  } else {
                  gpio_toggle(P_LED_R);}
            if(counter == period - 1)
                  break;
            while(counter == temporary_counter );
      }while(counter < period);
}
```

Led/button isr functions

Οι συναρτήσεις προκειμένου να αναβοσβήνει το led έχουν ξανα υλοποιηθεί και σε προηγούμενα εργαστήριο όποτε θα αναφερθούμε μόνο στην *toggle_led_isr()* η οποία αναβοσβήνει το led με την βοήθεια ενός timer ανά 1 sec.

Ο counter αυτός αυξάνεται μέσα στην συνάρτηση *timer_callback_isr()* όπου αυξάνεται κατά ένα κάθε sec.

Προκειμένου να υλοποιηθούν σωστά ανάλογα με τις συχνότητες που ζητούνται σε κάθε περίπτωση έγινε χρήση ενός timer handler, όπου μετράει ανά ένα sec και αυξάνει έναν counter++;
Ανάλογα με το στάδιο της εργασίας που βρισκόμαστε έχουν οριστεί τρία mode έτσι ώστε να χειριστούμε ποιά συχνότητα κάθε φορά είναι η σωστή για να διαβάζονται τα δεδομένα θερμότητας. Όπως φαίνεται και από την υλοποίηση της main παρακάτω (ένα κομμάτι του κώδικα παρουσίαζεται) στα δεξιά.

```
float temp;
period = 2;
do{
      if(counter >= period){
            counter = 0;
            temp = DHT_GetTemperature();
            control_led_temp(temp);
do{
      if (counter >= period){
            counter = 0;
            temp = DHT_GetTemperature();
            control_led_temp(temp);
            __WFI(); }
}while(mode == 2);

do{
      if (counter >= period){
            counter = 0;
            temp = DHT_GetTemperature();
      control_led_temp(temp);
            __WFI(); }
}while(mode == 3);
```

Το AEM το εισάγει ο χρήστης μέσω της uart όπως είχε γίνει και στο lab2, μπορεί να είναι 4ψήφιο ή και 5ψήφιο και αμέσως μετά απενεργοποιούνται όλα τα uart_interupts μέσω της συνάρτησης NVIC_DisableIRQ(USART2_IRQn);
Τέλος, υλοποιείται και μιά συνάρτηση *control_led_temp(float temp)*, η οποία ελέγχει την θερμοκρασία που λαμβάνεται από τον αισθητήρα και ανάλογα την τιμή καλεί τα interrupt handler των leds.