Μικροεπεξεργαστές και Περιφερειακά

Εαρινό Εξάμηνο 2023

3η Εργασία/Εργαστήριο

Στην παρούσα εργασία υλοποιήσαμε έναν driver(ένα αρχείο C και ένα header) που ελέγχει την επικοινωνία του αισθητήρα με τον μικροεπεξεργαστή και ένα αρχείο C με τη main συνάρτηση.

Όσων αφορά τον **driver**:

- **Header(dht11.h)**: Ορίσαμε το pin που συνδέεται με τον αισθητήρα. Υλοποιήσαμε ένα struct που περιέχει τους τύπους δεδομένων που λαμβάνει από τον αισθητήρα και αποθηκεύονται σε μια μεταβλητή και ορίσαμε τις συναρτήσεις που θα χρησιμοποιηθούν για την επικοινωνία με τον αισθητήρα.

```
typedef struct {
  uint8_t humidity;
  uint8_t temperature;
  Pin pin;
} dht11_TypeDef;

void dht11_init(dht11_TypeDef *dht11, uint8_t hum, uint8_t tmp, Pin pin);
uint8_t dht11_read_byte(void);
uint8_t dht11_read(dht11_TypeDef *dht11);
uint8_t dht11_start(void);
```

- **C file(dht11.c)**:

void dht11_init(dht11_TypeDef *dht11, uint8_t hum, uint8_t tmp, Pin pin): Αρχικοποιεί την μεταβλητή dht11 με τα δεδομένα που δίνονται κατά την κλήση της συνάρτησης. Ακόμη, κάνει το pin mode σε output και ο μικροεπεξεργαστής στέλνει σήμα high voltage.

uint8_t dht11_start(void): Ξεκινά την επικοινωνία μεταξύ του μικροεπεξεργαστή και του αισθητήρα. Αρχικά ο μικροεπεξεργαστής στέλνει σήμα μηδέν(gpio_set(DHT11_PIN, 0);) και το κρατάει εκεί για 18ms. Μετά, στέλνει σήμα 1 και το κρατά για 20us. Έπειτα μετατρέπει το pin mode σε input και περιμένει μέχρι να δει το σήμα μηδέν, κι όταν αυτό ισχύει ξεκινά η επικοινωνία.

```
gpio_set(DHT11_PIN, 0);
delay_ms(18);

gpio_set(DHT11_PIN, 1);
delay_us(20);

gpio_set_mode(DHT11_PIN, Input);
```

```
while(1){
if(gpio_get(DHT11_PIN) == 0){
    break;
  }
}

delay_us(80);
```

```
if(gpio_get(DHT11_PIN) == 0){
  /*uart_print("start2 = 0\r\n");*/
  return 0;
}
while(1){
  if(gpio_get(DHT11_PIN) == 0){
    break;
  }
}
```

Περιμένει 80us, τη διάρκεια που ο αισθητήρας στέλνει σήμα μηδέν. Στη συνέχεια, ελέγχει αν το σήμα παραμένει μηδέν μετά την πάροδο των 80us και αν ισχύει, η συνάρτηση επιστρέφει 0. Αλλιώς κάνει delay άλλα 80us και επιστρέφει 1 για να ενημερώσει τον μικροεπεξεργαστή πως είναι έτοιμος να στείλει δεδομένα.

uint8_t dht11_read_byte(void): Διαβάζει ένα byte από τα δεδομένα που στέλνει ο αισθητήρας. Χρησιμοποιούμε ένα loop που επαναλαμβάνεται για 8 φορές(για κάθε bit του byte). Αρχικά, υλοποιήσαμε μία while που περιμένει το σήμα να γίνει high, δηλαδή να περάσουν τα 50us που το σήμα μένει στο μηδέν για να ειδοποιήσει ο αισθητήρας τον μικροεπεξεργαστή ότι στέλνει 1 bit. Έπειτα, περιμένει άλλα 28us, το χρόνο που ο αισθητήρας κρατά το σήμα σε high για να σταλεί το bit 0.

```
while(1){
  if(gpio_get(DHT11_PIN)){
    break;
  }
}
  delay_us(28);
```

```
if(gpio_get(DHT11_PIN)){

    data |= (1 << (7-j));

    delay_us(42);
  }
```

Στη συνέχεια, ελέγχεται αν το pin είναι ακόμα σε high voltage, δηλαδή το bit που στέλνεται είναι το 1. Αν ισχύει, το bit 1 κάνει αριστερή ολίσθηση όσες φορές χρειάζεται για να φτάσει στη σωστή θέση του byte, κάτι που επιτυγχάνεται αν αφαιρέσουμε από το 7 τη μεταβλητή j που χρησιμοποιείται για το loop και περιμένει 42s ώστε το σήμα να γίνει ξανά μηδέν και να σταλεί το επόμενο bit. Τέλος, επιστρέφει τη μεταβλητή data που περιέχει το byte.

uint8_t dht11_read(dht11_TypeDef *dht11): Ξεκινάει το διάβασμα των δεδομένων και αποθηκεύει τα αποτελέσματα στις μεταβλητές της dht11. Αρχικά, μέσα σε μία if καλεί τη συνάρτηση dht11_start και αν επιτρέψει 1, αποθηκεύει τα 5 byte που στέλνει ο αισθητήρας σε 5 μεταβλητές, χρησιμοποιώντας για το καθένα τη συνάρτηση dht11_read_byte. Αν επιστρέψει μηδέν, σημαίνει πως δεν ξεκίνησε η επικοινωνία.

```
uint8_t tmp_b2 = dht11_read_byte();
uint16_t check_sum = dht11_read_byte();
```

```
sum = hum_b1 + hum_b2 + tmp_b1 + tmp_b2;
if (check_sum == sum ) {
  dht11->humidity = hum_b1;
  dht11->temperature = tmp_b1;
```

Έπειτα, κάνει ξανά output το pin και κάνει το σήμα 1(free status) για την επόμενη επικοινωνία. Τέλος, ελέγχουμε αν το πέμπτο byte είναι ίσο με το άθροισμα των των υπολοίπων byte για να δούμε αν η επικοινωνία ήταν επιτυχής και περνάμε τα δεδομένα στις μεταβλητές της dht11.

Όσον αφορά τον **κώδικα υλοποίησης της εργασίας**:

Οι αρχικοποιήσεις της uart, του διακόπτη και των leds , καθώς και οι συναρτήσεις των callbacks τους υλοποιήθηκαν με τον ίδιο τρόπο με την προηγούμενη εργασία. Αρχικά, περιμένουμε μέχρι να δοθεί το AEM χρησιμοποιώντας ένα endless loop που κάνει break όταν το flag της uart_rx_isr γινεται 1. Τότε ξεκινάει να λειτουργεί ο timer.

Για τον timer, αρχικοποιήθηκε με περίοδο 2s.

```
timer_init(1000000);
timer_set_callback(timer_isr);
timer_enable();
```

Όταν καλείται η **timer_isr**, επειδή ο timer είναι του 1s, ορίζεται ένας counter που αυξάνεται κατά 1 κάθε φορά που χτυπάει ο timer. Όταν η τιμή του φτάσει στην επιθυμητή περίοδο, our_timer_period, αλλάζει η τιμή του getTempFlag, δηλαδή το flag που ειδοποιεί πως ήρθε η στιγμή να πάρουμε θερμοκρασία, σε 1 και μηδενίζει τον counter.

```
if(counter < (int)our_timer_period){
  counter++;
}
```

```
else if(counter == (int)our_timer_period){
  counter = 0;
  getTempFlag = 1;
}
```

Μέσα στο endless loop της main, ελέγχεται αρχικά το flag του διακόπτη. Αν είναι 1, αυξάνεται ο μετρητής που κρατά τον αριθμό πού έχει πατηθεί ο διακόπτης. Αν ο διακόπτης έχει πατηθεί 1 φορά, παίρνουμε τα 2 τελευταία ψηφία του AEM από την rx.queue και αλλάζουμε την περίοδο του timer στο άθροισμα τους.

```
if(switchCount == 1){
  i = rx_queue.data[aemCount - 1] - '0';
  j = rx_queue.data[aemCount - 2] - '0';
  our_timer_period = (uint32_t)(i + j);
```

Έπειτα ελέγχουμε ο αριθμός των φορών που έχει πατηθεί ο διακόπτης είναι μονός ή ζυγός και αλλάζουμε την περίοδο του σε 3s ή 4s αντίστοιχα και το flag γίνεται μηδέν για να χρησιμοποιηθεί ξανά.

```
else if(switchCount % 2 != 0){
  our_timer_period = 3;
}
```

```
else if(switchCount % 2 == 0){
  our_timer_period = 4;
}
switchFlag = 0;
```

Στη συνέχεια, ελέγχουμε αν το getTempFlag είναι 1 και αν ισχύει, καλούμε την dht11_read ώστε να ξεκινήσει η επικοινωνία με τον αισθητήρα, την συνάρτηση tmp() και μηδενίζουμε το getTempFlag.

```
if(getTempFlag == 1){
  dht11_read(&dht11);
  tmp();
  getTempFlag = 0;
}
```

Η συνάρτηση **tmp()** ελέγχει την τιμή της θερμοκρασίας. Αν είναι μεταξύ των 20 και 25 βαθμών Κελσίου, κάνει το tmpFlag=2, αν είναι μικρότερη των 20 βαθμών, κάνει το tmpFlag=0, ενώ αν είναι μεγαλύτερη των 25 βαθμών κάνει το tmpFlag=1. Τέλος, καλεί τη συνάρτηση **print_isr()**.

```
if(((int)dht11.temperature) >= 20 && ((int)dht11.temperature) <= 25){
     tmpFlag = 2;       /*blink LED if temp is between 20?C and 25?C*/
   }
```

Η συνάρτηση **print_isr()** εκτυπώνει την θερμοκρασία και τον ρυθμό δειγματοληψίας.

```
void print_isr(void){
  char s1[100], s2[100];
  sprintf(s1, "The temperature is %d degrees Celsius \r\n", (int)dht11.temperature);
  uart_print(s1);
  sprintf(s2, "The sampling rate is %d \r\n", our_timer_period);
  uart_print(s2);
}
```

Πίσω στο endless loop της main, ελέγχεται το tmpFlag. Αν είναι 1 ενεργοποιεί το led, ενώ αν είναι 0 το απενεργοποιεί με την εντολή gpio_set. Αν το tmpFlag είναι 2, χρησιμοποιούμε μία while που λειτουργεί όσο το flag παραμένει 2 και δεν πατιέται ο διακόπτης. Μέσα χρησιμοποιούμε την εντολή gpio_toggle για να αλλάζει την κατάσταση του led και μια delay_ms για να γίνεται κάθε 1s.

```
else if(tmpFlag == 2){
    while(tmpFlag ==2 && switchFlag == 0){
      gpio_toggle(P_LED_R);
      delay_ms(1000);
    }
```

Σιταρίδης Παναγιώτης ΑΕΜ: 10249
Τσιτσάνου Άννα ΑΕΜ: 10051