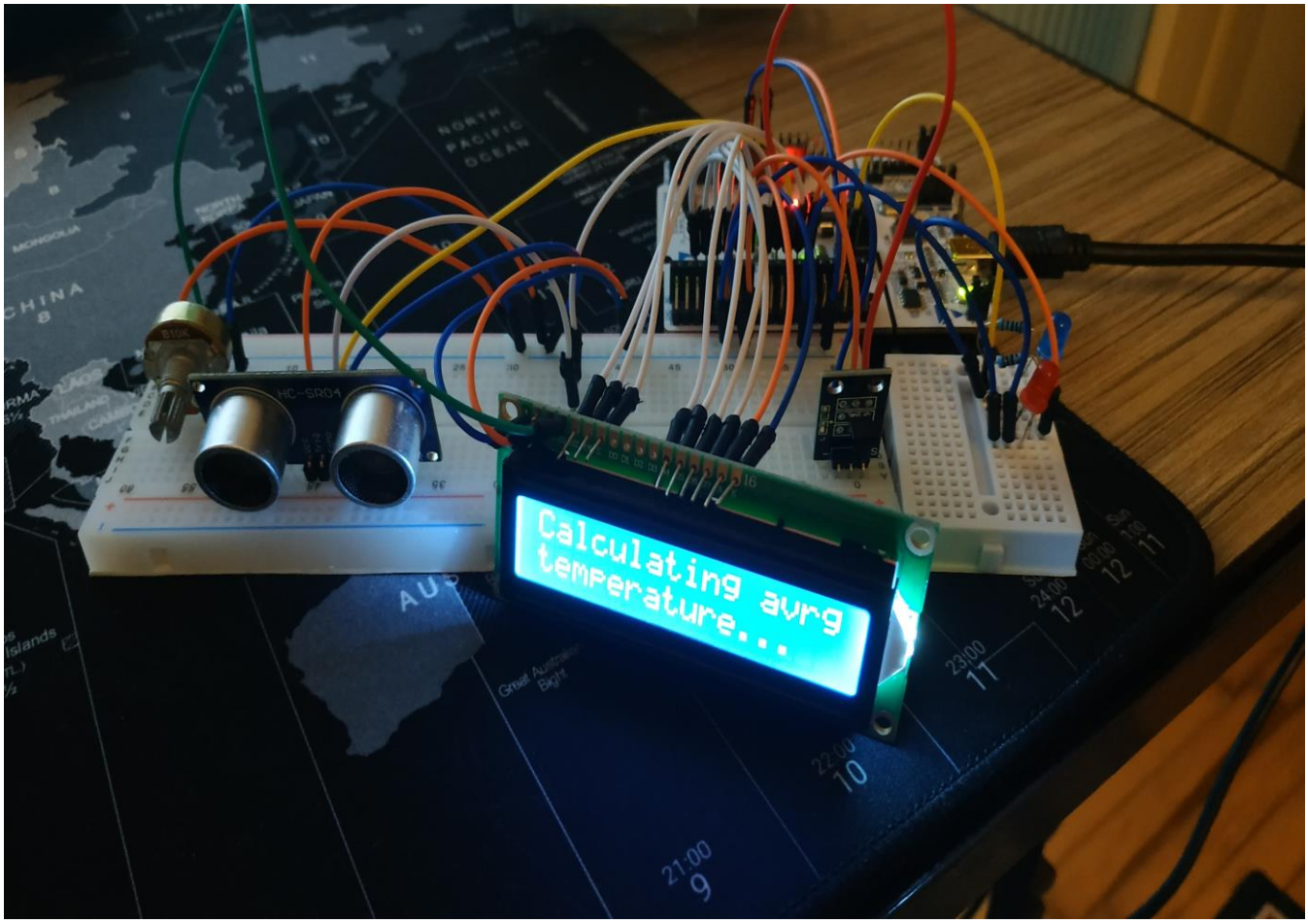
Μικροεπεξεργαστές και περιφερειακά

Εργασία 3

Χωραφάς Χρήστος 8718, Φάββας Αντώνης 8675



Σημείωση: από την εικόνα λείπει μόνο το 3ο led-διακόπτης, καθώς δεν είχαμε, ο κώδικας όμως ελέγχθηκε με τα υπάρχοντα.

Αλγόριθμος:

- Για την μέτρηση θερμοκρασίας χρησιμοποιήθηκε timer interruption, ο οποίος «σκάει» 5 φορές για μία μέτρηση (timer_init(1000000)).
- Για να διαβαστεί η τιμή του σένσορα θερμοκρασίας, διαβάσαμε το datasheet του online και δράσαμε κατάλληλα. Πρώτα πρέπει να στείλουμε ένα σήμα για κάποια μικροδευτερόλεπτα στον σένσορά και έπειτα παίρνουμε την απόκριση του. Από εκεί και πέρα χρειάζονται κάποια write commands, με κατάλληλα delays ενδιάμεσα και τέλος ένα read, που θα μου δώσει τα 2 bytes του scratchpad. Το datasheet: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
- Για την μέτρηση της απόστασης μέσω του σένσορα εγγύτητας, μετρήσαμε τον χρόνο για τον οποίο μένει σε κατάσταση high το pin echo και μετατρέψαμε αυτόν τον χρόνο σε απόσταση. Στην εργασία αυτή αποφασίσαμε να ακολουθήσουμε μια λίγο πιο μπακάλικη προσέγγιση και μετρήσαμε κύκλους μέσω μιας while λουπας, για όσο χρόνο είνια high το echo pin, χρησιμοποιώντας τον debugger για να δούμε τους αριθμούς κύκλων για μία λούπα μέσω του καταχωρητή States. Έτσι γλιτώσαμε λίγες γραμμές κώδικα, αφού τον «καλό» τρόπο αξιοποιήσαμε στην 2^{η} εργασία με χρήση της Assembly.

Κώδικας:

- **ultrasonic_init()**: ενεργοποιεί τον αισθητήρα εγγύτητας, στέλνοντας ένα σήμα για τουλάχιστον 10 us.

```
// Sensor initializations to start communication
void ultrasonic_init() {
    gpio_set(P_TRIGGERED, 1);
    delay_us(12);
    gpio_set(P_TRIGGERED, 0);
}
```

- **timer_isr():** Κρατάμε ελάχιστο τον χρόνο που σπαταλείται εδώ, αυξάνοντας απλά τον counter των interruptions.

- **Calculate_avrg():** για κάποιο λόγο το queue έπρεπε να το αρχικοποιήσουμε με 25 θέσεις για να βγάζει σωστό ΜΟ θερμοκρασίας, εξού και το 25 στην for.

```
void timer_isr() {
    counter++;
}

// Calculate average temperature
float calculate_avrg(Queue *q) {
    int i, sum = 0;
    for(i = 0; i < 25; i++) {
        sum += q->data[i];
    }
    return ((float)sum / 24);
}
```

- Όλες οι μέθοδοι επικοινωνίας και αρχικοποίησης με τους σένσορες.

```
uint8_t tempsensor_init() {
    uint8_t response = 0;
    gpio_set_mode(P_rec, Output);
    gpio_set(P_rec, 0);
    delay_us(480);

    gpio_set_mode(P_rec, Input);
    delay_us(80);
    if(!(gpio_get(P_rec))) response = 1;
    else response = 0;
    delay_us(400);
    return response;
}

// Write method
void tempsensor_write(uint8_t data) {
    gpio_set_mode(P_rec, Output);                // se
    for (int i = 0; i < 8; i++) {
        if ((data & (1<<i))!=0) {                    //
            gpio_set_mode(P_rec, Output);        // se
            gpio_set(P_rec, 0);
            delay_us(1);

            gpio_set_mode(P_rec, Input);         // se
            delay_us(60);
        }
        else {
            gpio_set_mode(P_rec, Output);
            gpio_set(P_rec, 0);
            delay_us(60);

            gpio_set_mode(P_rec, Input);
        }
    }
}

// Read method
uint8_t tempsensor_read() {
    uint8_t value = 0;
    gpio_set_mode(P_rec, Input);
    for (int i = 0; i < 8; i++) {
        gpio_set_mode(P_rec, Output);
        gpio_set(P_rec, 0);
        delay_us(2);

        gpio_set_mode(P_rec, Input);             // se
        if(gpio_get(P_rec)) {
            value |= 1<<i;
        }
        delay_us(60);
    }
    return value;
}
```

- Μέσα στην **main():** Αρχικά ενεργοποιούμε τον σένσορα εγγύτητας και μετράμε κύκλους για όσο το echo pin ειναι high. Έπειτα μετατρέπουμε σε χρόνο και απόσταση σύμφωνα με έτοιμους τύπους. Το 170 προκύπτει επειδη η ταχύτητα του ήχου ειναι 340m/s αλλά το κύμα διανύει 2 φορές την ίδια απόσταση. Στην συνέχεια τσεκάρουμε με μια if αν υπάρχει πολύ κοντά κάποιο εμπόδιο.

```
timer_enable();
while(1) {
    cycles = 0;                              // Re-initializ
    duration = 0.0f;
    distance = 0;
    leds_set(1);
    lcd_set_cursor(0,0);

    // Start getting signal from ultrasonic sensor
    ultrasonic_init();
    while(!(gpio_get(P_ECHO)));
    while(gpio_get(P_ECHO)) {
        cycles += 33;                        // check debugg
    }
    duration = (float) cycles / SystemCoreClock;
    distance = duration * 170 * 100;         // measured dis

    if(distance < min_distance) {
        lcd_print("Current oC:");            // Print curren
        sprintf(rx_tmp, " %i  ", (int)temp);
        lcd_print(rx_tmp);

        lcd_set_cursor(0,1);
        lcd_print("Average oC:");            // Print averag
        sprintf(rx_tmp, " %i  ", (int)avrg);
        lcd_print(rx_tmp);
    }
```

- Μέτρηση θερμοκρασίας: Αν ο counter των interruptions γίνει 5, σημαίνει πως πέρασαν 5 δευτερόλεπτα και άρα ξεκινάει νέα μέτρηση.

```
// Start getting signal from temperature sensor
if(counter == 5) {
    counter = 0;
    leds_set(0);

    sres = tempsensor_init();
    delay_ms(1);
    tempsensor_write(0xCC);              // skip ROM
    tempsensor_write(0x44);              // convert t
    delay_ms(200);

    sres = tempsensor_init();
    tempsensor_write(0xCC);              // skip ROM
    tempsensor_write(0xBE);              // Read Scratch-pad

    tmp_byte1 = tempsensor_read();
    tmp_byte2 = tempsensor_read();
    temp = (float)((tmp_byte2<<8)|tmp_byte1)/16; // measured
```

- **Εσωτερικά** αυτής της if πρέπει να ελέγξουμε αν η λίστα με τις μετρήσεις είναι γεμάτη, αν ναι τότε αυτό σημαίνει πως έχουμε 24 μετρήσεις και θέλουμε τον MO (σε αυτή την περίπτωση απενεργοποιούμε τα interruptions για 10s), αλλιώς κάνουμε απλά enqueuer και τσεκάρουμε την κατάσταση της θερμοκρασίας, εμφανίζοντας κατάλληλα μηνύματα.

```
if(queue_is_full(&rx_queue)) {
    __disable_irq();
    avrg = calculate_avrg(&rx_queue);
    rx_queue.tail = 0;              // reset the queue to st
    qres = queue_enqueue(&rx_queue, temp);  // store the 1st

    lcd_clear();
    lcd_set_cursor(0,0);
    lcd_print("  Average temp");
    lcd_set_cursor(0,1);

    sprintf(rx_tmp, "     %ioC     ", (int)avrg);
    lcd_print(rx_tmp);
    delay_ms(10000);
    __enable_irq();
}
else {
    qres = queue_enqueue(&rx_queue, temp);

    // Checks for temperature and distance
    if(temp >= hot_temp) {
        gpio_set(P_led_r, 1);
        gpio_set(P_led_g, 1);       // A switch turns on (le

        lcd_clear();
        lcd_set_cursor(0,0);
        lcd_print("  Hot Hot Hot!");
    }
    else if(temp < cool_temp) {
        gpio_set(P_led_b, 1);

        lcd_clear();
        lcd_set_cursor(0,0);
        lcd_print(" Is it snowing?");
    }
    else {
        gpio_set(P_led_r, 0);
        gpio_set(P_led_g, 0);
        gpio_set(P_led_b, 0);

        lcd_print("Calculating avrg");
        lcd_set_cursor(0,1);
        lcd_print("temperature...");
    }
}
```

***Λεπτομέρειες*:**

- Επίτηδες έχουμε βάλει το led του board να αναβοσβήνει σε συγχρονισμό με το led του temperature module για να φαίνεται ξεκάθαρα το πότε γίνεται μέτρηση.
- Επειδή δεν δώθηκαν λεπτομέρειες στο πότε ακριβώς και για πόσο χρόνο θα εμφανίζονται κάποια μηνύματα στο LCD, έχουμε τα εξής:
    1) Αν εντοπιστεί εμπόδιο σε κάποια απόσταση, εμφανίζεται κατευθείαν το μήνυμα με τις θερμοκρασίες και παραμένει μέχρι την επόμενη μέτρηση (δηλαδή μαξ 5 δευτερόλεπτα).
    2) Τα μηνύματα που σχετίζονται με την κατάσταση της θερμοκρασίας (hot, cold) εμφανίζονται μόνο όταν γίνει μέτρηση και εφόσον ικανοποιούν τις αντίστοιχες συνθήκες και σταματούν, όταν δεν τις ικανοποιούν, στην επόμενη. Οπότε τα led είναι αναμένα για αντίστοιχο χρόνο.
    3) Ο τρόπος που δουλεύει ο υπολογισμός MO είναι: γίνεται πρώτα η μέτρηση θερμοκρασίας. Αν η λίστα όμως είναι γεμάτη, υπολογίζω το MO, επαναφέρω το tail στην αρχή και μετά κάνω enqueue αυτή την τιμή, η οποία θα είναι και η πρώτη μέτρηση των επόμενων 2 λεπτών.
- Για την εμφάνιση τιμών στο LCD χρησιμοποιήσαμε την sprint() της βιβλιοθήκης stdio.h.

Προβλήματα που αντιμετωπίσαμε:

Κύριο πρόβλημα ήταν το πως θα γίνει η μέτρηση θερμοκρασίας, αρχικά δοκιμάσαμε να το κάνουμε με uart επικοινωνία, αλλά καταλήξαμε σε αδιέξοδο, καθώς ο σένσορας έχει μόνο ένα pin για data. Μετά καταλάβαμε ότι έπρεπε να δώσουμε έμφαση στο datasheet και με λίγη βοήθεια από το ίντερνετ καταφέραμε να πάρουμε μέτρηση. Για τον σένσορα εγγύτητας καταλήξαμε στο ότι δεν χρειαζόμαστε ADC, μιας και αυτά που μας έβγαζε στην οθόνη ήταν ότι να ναι και επίσης καταλάβαμε ότι αυτό που βγάζει το echo pin είναι απλά μια high κατάσταση, που εμείς πρέπει να υπολογίσουμε για πόσο χρόνο παραμένει έτσι.