8η Εργαστηριακή Αναφορά

Ομάδα 20

Παπαδόπουλος Χαράλαμπος 03120199

Στρίφτης Γεώργιος 03121200

Η παρούσα άσκηση επικεντρώνεται στην ανάπτυξη μιας εφαρμογής IoT (Internet of Things) που προσομοιώνει την παρακολούθηση ασθενών σε ένα νοσοκομειακό περιβάλλον. Η εφαρμογή βασίζεται στη σειριακή επικοινωνία UART και στη χρήση του πομποδέκτη WiFi ESP8266 για τη σύνδεση μιας αναπτυξιακής πλακέτας (ntuAboard) με έναν εξυπηρετητή (server). Παράλληλα, χρησιμοποιούνται αισθητήρες για τη συλλογή δεδομένων θερμοκρασίας και πίεσης, τα οποία επεξεργάζονται και αποστέλλονται στον server.

Η υλοποίηση περιλαμβάνει τη δημιουργία ενός payload JSON που περιέχει τις τιμές θερμοκρασίας, πίεσης και την κατάσταση του συστήματος, καθώς και την αποστολή αυτού μέσω του ESP8266. Τα δεδομένα εμφανίζονται σε μια LCD οθόνη για την παρακολούθηση των τιμών σε πραγματικό χρόνο, ενώ η εφαρμογή περιλαμβάνει μηχανισμούς για τον έλεγχο και τη διαχείριση σφαλμάτων.

Ανάλυση

1. **Στόχοι της άσκησης**
    - Εξοικείωση με τη χρήση του πρωτοκόλλου UART για ασύγχρονη σειριακή επικοινωνία.
    - Ενσωμάτωση αισθητήρων για τη μέτρηση φυσικών μεγεθών (θερμοκρασία και πίεση).
    - Ανάπτυξη και αποστολή δομημένων δεδομένων (JSON payload) μέσω WiFi.
    - Υλοποίηση ελέγχων ασφαλείας για την αξιοπιστία της επικοινωνίας και την ακρίβεια των δεδομένων.
2. **Περιγραφή της εφαρμογής**
    - **Διαδικασία επικοινωνίας**: Το ESP8266 προγραμματίζεται ώστε να συνδέεται σε ένα τοπικό δίκτυο WiFi. Στη συνέχεια, λαμβάνει εντολές μέσω UART για τη ρύθμιση παραμέτρων και την αποστολή δεδομένων στον server.
    - **Λήψη και επεξεργασία δεδομένων**: Ο αισθητήρας θερμοκρασίας DS18B20 και το ποτενσιόμετρο που προσομοιώνει την πίεση συλλέγουν δεδομένα. Οι τιμές επεξεργάζονται και προσαρμόζονται στα απαιτούμενα όρια.
    - **Δημιουργία JSON Payload**: Τα δεδομένα μετατρέπονται σε μορφή JSON, περιλαμβάνοντας το όνομα της ομάδας, τις τιμές θερμοκρασίας και πίεσης, καθώς και την κατάσταση (status).
    - **Εμφάνιση δεδομένων**: Η LCD εμφανίζει τις τρέχουσες τιμές και το status, παρέχοντας άμεση οπτική πληροφόρηση.

3. **Χρήση της σειριακής επικοινωνίας UART** Το UART είναι βασικός μηχανισμός της άσκησης, επιτρέποντας την επικοινωνία μεταξύ του ATmega328PB και του ESP8266. Οι βασικοί καταχωρητές ελέγχου και δεδομένων χρησιμοποιούνται για τη ρύθμιση και λειτουργία του UART, με προγραμματισμό για συγκεκριμένο ρυθμό μετάδοσης (baud rate 9600).

4. **Έλεγχος κατάστασης (status)** Το σύστημα ορίζει διάφορες καταστάσεις (OK, CHECK TEMP, CHECK PRESSURE, NURSE CALL) με βάση τα δεδομένα από τους αισθητήρες και τις ενέργειες του χρήστη. Αυτές οι καταστάσεις διασφαλίζουν ότι οι μετρήσεις παρακολουθούνται και αξιολογούνται συνεχώς.

5. **Αποστολή δεδομένων στον server** Η τελική λειτουργία περιλαμβάνει την αποστολή του JSON payload στον server μέσω του ESP8266. Η επιτυχία ή αποτυχία της επικοινωνίας καταγράφεται και εμφανίζεται στην LCD.

```
#define F_CPU 16000000UL

#include<avr/io.h>
#include<avr/interrupt.h>
#include<util/delay.h>
#include <stdio.h>
#include <string.h>

#include "setup_LCD_PEX.h"
#include "setup_TWI.h"
#include "setup_OWI.h"
#include "setup_USART.h"
#include "setup_KEYBOARD.h"

static uint8_t temp_l, temp_h;
extern uint16_t pressed_keys;

typedef enum {
    NURSE_CALL,          // 0
    OK,                  // 1
    CHECK_TEMP,          // 2
    CHECK_PRESSURE       // 3
} STATUS_t;

void comms();
const char* status_to_string(STATUS_t s);
void transmit_state(double temperature, double pressure, STATUS_t
status);
void transmit_payload(double temperature, double pressure, int team,
STATUS_t status);
STATUS_t check_button();
double measure_temp();
double read_adc();
void transmit_esp();

void setup() {
    twi_init();
    _delay_ms(50);  // Delay 500 ms for better stability
    PCA9555_0_write(REG_CONFIGURATION_0, 0x00); // Set as output
    lcd_init();
```

```
    _delay_ms(50);
    lcd_clear_display();
    _delay_us(100);
    usart_init(103);
    ADMUX = 0b01000000;   // ADC right-adjusted, select ADC2
    ADCSRA = 0b10000111;  // Enable ADC with a prescaler of 128
    setup_keyboard();
    _delay_ms(50);  // Delay 500 ms for better stability

}

int main()
{
    setup();
    double pressure;
    STATUS_t status;
    double temperature;
    int team = 20;


    while(1){
        lcd_clear_display();
        lcd_set_cursor(0, 0);

        comms();

        status = check_button();
        pressure = read_adc();
        temperature = measure_temp();

        if((pressure < 4 || pressure > 12) && status != NURSE_CALL)
            status = CHECK_PRESSURE;
        else if ((temperature <= 34.0 || temperature >= 37.0) && status
!= NURSE_CALL)
            status = CHECK_TEMP;

        transmit_state(temperature, pressure, status);
        transmit_payload(temperature, pressure, team, status);
        transmit_esp();

    }
}

double measure_temp()
{
    int16_t temperature = 0;
    if(one_wire_reset()) {
        one_wire_transmit_byte(0xCC);
        one_wire_transmit_byte(0x44);
        while(!one_wire_receive_bit()) {
            //busy waiting
        }
        if(one_wire_reset()) {
            one_wire_transmit_byte(0xCC);
            one_wire_transmit_byte(0xBE);
            temp_l = one_wire_receive_byte();
            temp_h = one_wire_receive_byte();
```

```
        }
        else {
            temp_l = 0x00;
            temp_h = 0x80;
        }
    }
    else {
        temp_l = 0x00;
        temp_h = 0x80;
    }

    temperature = (temp_h & 0b00000111) << 8;
    temperature |= temp_l;
    double result = ((double)temperature * 0.0625) + 10.0;


    return result;
}

double read_adc()
{
    double adc;
    double output;
    ADCSRA |= (1 << ADSC);
    while (ADCSRA & (1 << ADSC));
    adc = ADC;
    output = (adc * 20) / 1024;
    return output;
}

void comms()
{
    char message1[] = "ESP:connect";
    char message2[] = "ESP:url:\"http://192.168.1.250:5000/data\"";
    char c;
    for(int i = 0; i < strlen(message1); i++) {
        usart_transmit(message1[i]);
        _delay_us(10);
    }
    usart_transmit('\n');

    lcd_data('1');
    lcd_data('.');
    do {
        c = usart_receive();
        if(c != '\n')
            lcd_data(c);
    } while(c != '\n');


    lcd_set_cursor(1, 0);

    for(int i = 0; i < strlen(message2); i++) {
        usart_transmit(message2[i]);
        _delay_us(10);
    }
    usart_transmit('\n');
```

```
    lcd_data('2');
    lcd_data('.');
    do {
        c = usart_receive();
        if(c != '\n')
            lcd_data(c);
    } while(c != '\n');
}

const char* status_to_string(STATUS_t s) {
    switch (s) {
        case OK: return "OK";
        case NURSE_CALL: return "NURSE CALL";
        case CHECK_TEMP: return "CHECK TEMP";
        case CHECK_PRESSURE: return "CHECK PRESSURE";
    }
    return "OK";
}

void transmit_state(double temperature, double pressure, STATUS_t status)
{
    char pres_out[5];
    char status_out[16];
    char temp_out[5];
    lcd_clear_display();
    lcd_data('T');
    lcd_data(':');
    sprintf(temp_out, "%.1f", temperature);
    for (int i = 0; i < strlen(temp_out); i++)
        lcd_data(temp_out[i]);

    lcd_data(' ');

    lcd_data('P');
    lcd_data(':');
    snprintf(pres_out, 5, "%.2f", pressure);
    for (int i = 0; i < strlen(pres_out); i++)
        lcd_data(pres_out[i]);

    lcd_set_cursor(1, 0);
    snprintf(status_out, 16, status_to_string(status));
    for (int i = 0; i < strlen(status_out); i++)
        lcd_data(status_out[i]);

    _delay_ms(2000);
}

void transmit_payload(double temperature, double pressure, int team,
STATUS_t status)
{
    char payload[256] = {0};
    char c;
    snprintf(payload, 256,
             "ESP:payload:[{\"name\": \"temperature\",\"value\":
\"%.2f\"},"
             "{\"name\": \"pressure\",\"value\": \"%.2f\"},"
```

```
            "{\"name\": \"team\",\"value\": \"%d\"},"
            "{\"name\": \"status\",\"value\": \"%s\"}]",
            temperature, pressure, team, status_to_string(status));

    for(int i = 0; i < strlen(payload); i++) {
        usart_transmit(payload[i]);
        _delay_us(10);
    }
    usart_transmit('\n');

    lcd_clear_display();
    lcd_data('3');
    lcd_data('.');
    do {
        c = usart_receive();
        if(c != '\n')
            lcd_data(c);
    } while(c != '\n');

    _delay_ms(2000);
    return;
}

void transmit_esp()
{
    char message[] = "ESP:transmit";
    char c;

    for(int i = 0; i < strlen(message); i++) {
        usart_transmit(message[i]);
        _delay_us(10);
    }
    usart_transmit('\n');


    lcd_set_cursor(1, 0);
    lcd_data('4');
    lcd_data('.');
    do {
        c = usart_receive();
        if(c != '\n')
            lcd_data(c);
    } while(c != '\n');

    _delay_ms(1000);
}

STATUS_t check_button()
{
    int num = 0;
    STATUS_t status = OK;
    int del = 0;
    while(del < 500) {
        scan_keypad_rising_edge();
        if (pressed_keys != 0) {
            num = keypad_to_ascii();
            if( num == 48)
```

```
                status = NURSE_CALL;
                else if(num == 35)
                status = OK;
            }
        _delay_us(100);
        del++;
    }

    return status;
}
```