Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών ΕΜΠ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΑΚΑΔ. ΕΤΟΣ 2024-2025

8ή εργαστηριακή άσκηση.

Ομάδα : 58

Ονοματεπώνυμο : Φέζος Κωνσταντίνος / Α.Μ : 03118076

Ονοματεπώνυμο : Τσουκνίδας Οδυσσέας Αρθούρος Ρήγας / Α.Μ : 03120043

8η Εργαστηριακή Άσκηση

Στην 8η εργαστηριακή, θα χρησιμοποιήσουμε μία από τις USART μονάδες που διαθέτει η πλακέτα NTUA-Board και έναν πομποδέκτη WiFi ESP8266, ώστε να επικοινωνήσουμε με το gateway του εργαστηρίου, παίρνοντας έτσι μία πρώτη ιδέα για το Internet of Things.

Σε πρώτη φάση, θα προσομοιώνουμε με τον υπολογιστή μας τις απαντήσεις του gateway ώστε να βεβαιωθούμε ότι έχουμε υλοποιήσει σωστά την σειριακή επικοινωνία και στο 3ο μέρος θα συνδέσουμε όντως τον πομποδέκτη ώστε να μπορέσουμε να δούμε στο περιβάλλον του εργαστηρίου τα αποτελέσματα της δικτυακής επικοινωνίας.

Για να είναι πιο ευανάγνωστος και διαχειρίσιμος ο κώδικας μας, θα φτιάξουμε ένα header file με όλες τις συναρτήσεις που θα χρησιμοποιήσουμε στην συνέχεια, και ένα source file που θα τις υλοποιεί. Οι περισσότερες από αυτές τις συναρτήσεις είναι συναρτήσεις που έχουμε δει σε προηγούμενες εργαστηριακές ασκήσεις, αλλά θα γράψουμε επιπλέον και κάποιες που θα χειρίζονται την σειριακή επικοινωνία. Οι νέες συναρτήσεις που ορίσαμε φαίνονται παρακάτω :

```
/* Routine: usart_init
Description:
This routine initializes the
usart as shown below.
------- INITIALIZATIONS -------
Baud rate: 9600 (Fck= 16MH)
Asynchronous mode
Transmitter on
Reciever on
Communication parameters: 8 Data ,1 Stop, no Parity
--------------------------------
parameters: ubrr to control the BAUD.
return value: None.*/


void usart_init(unsigned int ubrr){
       UCSR0A=0;
       UCSR0B=(1<<RXEN0)|(1<<TXEN0);
       UBRR0H=(unsigned char)(ubrr>>8);
       UBRR0L=(unsigned char)ubrr;
       UCSR0C=(3 << UCSZ00);
       return;
}

/* Routine: usart_transmit
```

```
Description:
This routine sends a byte of data
5
using usart.
parameters:
data: the byte to be transmitted
return value: None. */

void usart_transmit(uint8_t data){
	while(!(UCSR0A&(1<<UDRE0)));
	UDR0=data;
}

/* Routine: usart_receive
Description:
This routine receives a byte of data
from usart.
parameters: None.
return value: the received byte */

uint8_t usart_receive(){
	while(!(UCSR0A&(1<<RXC0)));
	return UDR0;
}
/*Function that receives a response and returns its length, the package is saved in
buf*/

int usart_receive_buffer(uint8_t *buf, int length){
	int package = 0;
	while(package < length && ((*buf++ = usart_receive()) != BUF_END)) ++package;
	return package;

}

void usart_transmit_buffer(uint8_t *buf, int length) {
	while (length-- > 0) usart_transmit(*buf++);
}

/*function to send commands to the ESP*/
uint8_t esp_send_command(uint8_t command_type, uint8_t *argc, int argv){
	/*Works like syscalls returns 0 on success 1 on fail* -1 on undefined error/
	/*define type of commands*/
	static uint8_t connect[] = "ESP:connect";
	static uint8_t payload[] = "ESP:payload:";
	static uint8_t transmit[] = "ESP:transmit";
	static uint8_t url[]  = "ESP:url:";

	/*define type of returns*/

	static uint8_t success[] = "\"Success\"\n";
	static uint8_t fail[] = "\"Failure\"\n";

	uint8_t buf[50];
	int buf_length;
```

```
	/*Transmit command and its arguments if needed*/
	switch(command_type){
		case CMD_CONNECT:
			usart_transmit_buffer(connect, sizeof(connect)-1);
			break;
		case CMD_URL:
			usart_transmit_buffer(url, sizeof(url) - 1);
			usart_transmit_buffer(argc, argv);
			break;
		case CMD_PAYLOAD:
			usart_transmit_buffer(payload, sizeof(payload)-1);
			usart_transmit_buffer(argc, argv);
			break;
		case CMD_TRANSMIT:
			usart_transmit_buffer(transmit, sizeof(transmit) - 1);
			break;
		default:
			return -1;
			break;
	}

	/*Don't forget new line character as suggested */
	usart_transmit('\n');
	buf_length = usart_receive_buffer(buf, sizeof(buf));
	//usart_transmit('\n');

	/*Now lets see the response from the Gateway*/
	switch(command_type){
		case CMD_CONNECT:
		case CMD_URL:
		case CMD_PAYLOAD:
			if(!strncmp(buf, success, buf_length)) return 0;
			else if(!strncmp(buf, fail, buf_length)) return 1;
			break;
		case CMD_TRANSMIT:
			{
				int ret;
				for(ret = 0; ret < buf_length && ret < argv; ++ret)
argc[ret] = buf[ret];
				return ret;
				break;
			}
		default:
			return -1;
			break;
	}
	return -1;
}
```

Συμπληρωματικά με τις παραπάνω χρησιμοποιούμε όλες τις συναρτήσεις από τις εργαστηριακές ασκήσεις 3 μέχρι και 7 που μας επέτρεπαν τη χρήση

περιφερειακών και μικροελεγκτών της πλακέτας όπως η LCD οθόνη ή το Port Expander.

Ζήτημα 8.1

Στο πρώτο ζήτημα της άσκησης, καλούμαστε να γράψουμε κώδικα ο οποίος χρησιμοποιεί την UART και προσπαθεί να συνδέσει τον ESP8266 στο δίκτυο, και ανάλογα με την απάντηση που θα λάβει θα τυπώνει Success ή Fail στην LCD οθόνη. Στην συνέχεια, θα δίνει εντολή στον πομποδέκτη να επικοινωνήσει με συγκεκριμένο url, και θα τυπώνει αντίστοιχα Success ή Fail στην οθόνη.

```
/*
 * main.c
 *
 * Created: 12/2/2024 1:00:14 PM
 *  Author: kosta
 */

#define F_CPU 16000000UL
#include "header81.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdbool.h>
#include <string.h>

int main(void){

      /* Set up lcd screen and usart */

    twi_init();
    PCA9555_0_write(REG_CONFIGURATION_0, 0x00);
    lcd_init();
    _delay_ms(100);             //Maybe for the LCD
    lcd_clear_display();
      usart_init(103);

      uint8_t url[] = "\"http://192.168.1.250:5000/data\"";
      char success_1_msg[] = "1.Success";
      char fail_1_msg[] = "1.Fail";
      char success_2_msg[] = "2.Success";
      char fail_2_msg[] = "2.Fail";

      while(1){
            _delay_ms(1500);
            lcd_clear_display();
            uint8_t ret = esp_send_command(CMD_CONNECT, 0, 0);

            /* Check response and print return message */
            if(ret == 0){
                  for(int i = 0; i < strlen(success_1_msg); i++){
```

```
                        lcd_data(success_1_msg[i]);
                }
        }
        else if(ret == 1){
                for(int i = 0; i < strlen(fail_1_msg); i++){
                        lcd_data(fail_1_msg[i]);
                }
        }
        else{
                lcd_data('!');
        }

        /* Send command to connect to url */
        _delay_ms(1500);
        lcd_clear_display();

        ret = esp_send_command(CMD_URL, url, sizeof(url) - 1);
        /* Check response and print return message */
        if(ret == 0){
                for(int i = 0; i < strlen(success_2_msg); i++){
                        lcd_data(success_2_msg[i]);
                }
        }
        else if(ret == 1){
                //lcd_clear_display();
                for(int i = 0; i < strlen(fail_2_msg); i++){
                        lcd_data(fail_2_msg[i]);
                }
        }
        else{
                //lcd_clear_display();
                lcd_data('!');
        }
    }
}
```

Ζήτημα 8.2

Στο δεύτερο ζήτημα, καλούμαστε να επεκτείνουμε τον παραπάνω κώδικα ώστε να προσομοιώσουμε ένα απλό ψηφιακό σύστημα ενός νοσοκομείου. Συγκεκριμένα, αφού επικοινωνήσουμε επιτυχώς με τον server (που στο συγκεκριμένο ζήτημα τον προσομοιώνουμε απλά με τον υπολογιστή μας), θα του στέλνουμε payload το οποίο θα περιέχει μετρήσεις θερμοκρασίας και πίεσης ενός ασθενούς (τις οποίες θα προσομοιώνουμε με το θερμόμετρο DS18B20 που είδαμε στην 7η εργαστηριακή άσκηση, και με ένα ποτενσιόμετρο αντίστοιχα), αλλά και μία τιμή status η οποία θα διαμορφώνεται ως εξής:

1) Αν πατηθεί στο πληκτρολόγιο το πλήκτρο που αντιστοιχεί στο τελευταίο ψηφίο της ομάδας μας, δηλαδή 8 τότε το status να γίνεται “NURSE CALL”, ενώ αν πατηθεί στη συνέχεια η δίεση # τότε το status να γίνεται OK εκτός αν συμβαίνει κάτι από τα 2 η 3.
2) Αν η πίεση είναι πάνω από 12 ή κάτω από 4 τότε το status να γίνεται CHECK PRESSURE.
3) Αν η θερμοκρασία είναι κάτω από 34 ή πάνω από 37 τότε το status να γίνεται CHECK TEMP.
4) Αν δεν έχει συμβεί κανένα από τα παραπάνω τότε το status να γίνεται OK.

```
/*
 * main.c
 *
 * Created: 12/2/2024 7:11:06 PM
 *  Author: kosta
 */


#define F_CPU 16000000UL
#include "header81.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdbool.h>
#include <string.h>

#define CHK_TEMP 1
#define CHK_PRESSURE 2
#define CALL_NRS 3
#define STATUS_OK 0

int main(void){
    twi_init();
    PCA9555_0_write(REG_CONFIGURATION_0, 0x00);
      PCA9555_0_write(REG_CONFIGURATION_1, 0xF0); //PORT1[7:3] = input, [3:0] =
output
    lcd_init();
    _delay_ms(100);             //Maybe for the LCD
    lcd_clear_display();
    usart_init(103);
      adc_init();
      /*Get a measurement from the potensiometer and adjust accordingly*/

      char check_temp[] = "CHECK TEMP";
      char check_pressure[] = "CHECK PRESSURE";
      char ok[] = "OK";
      char call_nurse[] = "NURSE CALL";

      char payload[1024]; // Adjust size as needed for larger payloads
      char temp[256];     // Temporary buffer for intermediate formatting
```

```
uint8_t url[] = "\"http://192.168.1.250:5000/data\"";
char success_1_msg[] = "1.Success";
char fail_1_msg[] = "1.Fail";
char success_2_msg[] = "2.Success";
char fail_2_msg[] = "2.Fail";

start:
_delay_ms(1500);
lcd_clear_display();
uint8_t ret = esp_send_command(CMD_CONNECT, 0, 0);

/* Check response and print return message */
if(ret == 0){
	for(int i = 0; i < strlen(success_1_msg); i++){
		lcd_data(success_1_msg[i]);
	}
}
else if(ret == 1){
	for(int i = 0; i < strlen(fail_1_msg); i++){
		lcd_data(fail_1_msg[i]);
	}
	goto start;
}
else{
	lcd_data('!');
	goto start;
}

/* Send command to connect to url */
_delay_ms(1500);
lcd_clear_display();

ret = esp_send_command(CMD_URL, url, sizeof(url) - 1);
/* Check response and print return message */
if(ret == 0){
	for(int i = 0; i < strlen(success_2_msg); i++){
		lcd_data(success_2_msg[i]);
	}
}
else if(ret == 1){
	//lcd_clear_display();
	for(int i = 0; i < strlen(fail_2_msg); i++){
		lcd_data(fail_2_msg[i]);
	}
	goto start;
}
else{
	//lcd_clear_display();
	lcd_data('!');
	goto start;
}

_delay_ms(3000);
lcd_clear_display();
```

```
	reset:
	while(keypad_to_ascii() != '8'){
		/*Get a measurement from thermometer as in exercise 7*/
		uint16_t temperature = get_temperature_reading();
		uint8_t patient_temperature_int = integer_part(temperature) + 12;
		uint8_t patient_temperature_dec = decimal_part(temperature);

		int status = 0;

		/*Get ADC reading*/
		uint16_t patient_pressure_adc = adc_read();
		int patient_pressure = patient_pressure_adc / 51;

		if(patient_temperature_int > 36 || patient_temperature_int < 34){
			status = CHK_TEMP;
		}

		if(patient_pressure > 12 || patient_pressure < 4){
			status = CHK_PRESSURE;
		}

		// Start the JSON payload
		snprintf(payload, sizeof(payload), "");

		// Add each JSON object to the payload, these are similar for all status
		snprintf(temp, sizeof(temp), "{\"name\": \"temperature\",\"value\":
\"%d.%d\"},",patient_temperature_int, patient_temperature_dec);
		strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

		snprintf(temp, sizeof(temp), "{\"name\": \"pressure\",\"value\":
\"%d\"},", patient_pressure);
		strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

		snprintf(temp, sizeof(temp), "{\"name\": \"team\",\"value\": \"58\"},");
		strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);


		/* Send appropriate payload */
		switch(status){
			case STATUS_OK:
				snprintf(temp, sizeof(temp), "{\"name\":
\"status\",\"value\": \"OK\"}");
				strncat(payload, temp, sizeof(payload) - strlen(payload) -
1);
				// Close the JSON array
				strncat(payload, "]", sizeof(payload) - strlen(payload) -
1);
				send_three_to_screen(patient_temperature_int);
				lcd_data('.');
				send_three_to_screen(patient_temperature_dec);
				lcd_data(' ');
				lcd_data('\xdf');
				lcd_data('C');
				lcd_data(' ');
```

```
                        send_three_to_screen(patient_pressure);
                        lcd_change_line();
                        for(int i = 0; i < strlen(ok); i++){
                              lcd_data(ok[i]);
                        }
                        break;
                  case CHK_TEMP:
                        snprintf(temp, sizeof(temp), "{\"name\":
\"status\",\"value\": \"CHECK TEMP\"}");
                        strncat(payload, temp, sizeof(payload) - strlen(payload) -
1);

                        send_three_to_screen(patient_temperature_int);
                        lcd_data('.');
                        send_three_to_screen(patient_temperature_dec);
                        lcd_data(' ');
                        lcd_data('\xdf');
                        lcd_data('C');
                        lcd_data(' ');
                        send_three_to_screen(patient_pressure);
                        lcd_change_line();
                        for(int i = 0; i < strlen(check_temp); i++){
                              lcd_data(check_temp[i]);
                        }
                        break;
                  case CHK_PRESSURE:
                        snprintf(temp, sizeof(temp), "{\"name\":
\"status\",\"value\": \"CHECK PRESSURE\"}");
                        strncat(payload, temp, sizeof(payload) - strlen(payload) -
1);

                        send_three_to_screen(patient_temperature_int);
                        lcd_data('.');
                        send_three_to_screen(patient_temperature_dec);
                        lcd_data(' ');
                        lcd_data('\xdf');
                        lcd_data('C');
                        lcd_data(' ');
                        send_three_to_screen(patient_pressure);
                        lcd_change_line();
                        for(int i = 0; i < strlen(check_pressure); i++){
                              lcd_data(check_pressure[i]);
                        }
                        break;
                  default:
                        lcd_data('!');
                        break;
            }

            esp_send_command(CMD_PAYLOAD, payload, strlen(payload));
            _delay_ms(3000);
            lcd_clear_display();
            continue;

      nurse:
```

```
            while(keypad_to_ascii() != '#'){

                  uint16_t temperature = get_temperature_reading();
                  uint8_t patient_temperature_int = integer_part(temperature) + 12;
                  uint8_t patient_temperature_dec = decimal_part(temperature);

                  /*Get ADC reading*/
                  uint16_t patient_pressure_adc = adc_read();
                  int patient_pressure = patient_pressure_adc / 51;
                  snprintf(payload, sizeof(payload), "");

                  // Add each JSON object to the payload, these are similar for all
status
                  snprintf(temp, sizeof(temp), "{\"name\":
\"temperature\",\"value\": \"%d.%d\"},",patient_temperature_int,
patient_temperature_dec);
                  strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

                  snprintf(temp, sizeof(temp), "{\"name\": \"pressure\",\"value\":
\"%d\"},", patient_pressure);
                  strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

                  snprintf(temp, sizeof(temp), "{\"name\": \"team\",\"value\":
\"58\"},");
                  strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

                  snprintf(temp, sizeof(temp), "{\"name\": \"status\",\"value\":
\"NURSE CALL\"}");
                  strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);
                  status = CALL_NRS;

                  send_three_to_screen(patient_temperature_int);
                  lcd_data('.');
                  send_three_to_screen(patient_temperature_dec);
                  lcd_data(' ');
                  lcd_data('\xdf');
                  lcd_data('C');
                  lcd_data(' ');
                  send_three_to_screen(patient_pressure);
                  lcd_change_line();
                  for(int i = 0; i < strlen(call_nurse); i++){
                        lcd_data(call_nurse[i]);
                  }
                  esp_send_command(CMD_PAYLOAD, payload, strlen(payload));
                  _delay_ms(3000);
                  lcd_clear_display();
            }
            status = STATUS_OK;
            goto reset;
      }
      goto nurse;
}
```

Ζήτημα 8.3

Στο τελευταίο ζήτημα θα συνδέσουμε κατάλληλα τον πομποδέκτη στην πλακέτα μας και θα επεκτείνουμε τον κώδικα μας ώστε να ζητάμε από τον ESP8266 να στείλει στον server του εργαστηρίου το payload (εντολές payload και transmit) και στην συνέχεια να τυπώνουμε την απάντηση του server στην LCD οθόνη. Αν όλα έχουν υλοποιηθεί σωστά, μπορούμε να παρατηρήσουμε στον προβολέα του εργαστηρίου τα δεδομένα που του στείλαμε.

```
/*
 * main.c
 *
 * Created: 12/2/2024 7:11:06 PM
 *  Author: kosta
 */


#define F_CPU 16000000UL
#include "header81.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdbool.h>
#include <string.h>

#define CHK_TEMP 1
#define CHK_PRESSURE 2
#define CALL_NRS 3
#define STATUS_OK 0

int main(void){
    twi_init();
    PCA9555_0_write(REG_CONFIGURATION_0, 0x00);
       PCA9555_0_write(REG_CONFIGURATION_1, 0xF0); //PORT1[7:3] = input, [3:0] =
output
    lcd_init();
    _delay_ms(100);             //Maybe for the LCD
    lcd_clear_display();
    usart_init(103);
       adc_init();
       /*Get a measurement from the potensiometer and adjust accordingly*/

       char check_temp[] = "CHECK TEMP";
       char check_pressure[] = "CHECK PRESSURE";
       char ok[] = "OK";
       char call_nurse[] = "NURSE CALL";

       char payload[1024]; // Adjust size as needed for larger payloads
       char temp[256];     // Temporary buffer for intermediate formatting
       uint8_t transmit_response[100];
```

```
uint8_t url[] = "\"http://192.168.1.250:5000/data\"";
char success_1_msg[] = "1.Success";
char fail_1_msg[] = "1.Fail";
char success_2_msg[] = "2.Success";
char fail_2_msg[] = "2.Fail";
char success_3_msg[] = "3.Success";
char fail_3_msg[] = "3.Fail";

start:
_delay_ms(1500);
lcd_clear_display();
uint8_t ret = esp_send_command(CMD_CONNECT, 0, 0);

/* Check response and print return message */
if(ret == 0){
	for(int i = 0; i < strlen(success_1_msg); i++){
		lcd_data(success_1_msg[i]);
	}
}
else if(ret == 1){
	for(int i = 0; i < strlen(fail_1_msg); i++){
		lcd_data(fail_1_msg[i]);
	}
	goto start;
}
else{
	lcd_data('!');
	goto start;
}

/* Send command to connect to url */
_delay_ms(1500);
lcd_clear_display();

ret = esp_send_command(CMD_URL, url, sizeof(url) - 1);
/* Check response and print return message */
if(ret == 0){
	for(int i = 0; i < strlen(success_2_msg); i++){
		lcd_data(success_2_msg[i]);
	}
}
else if(ret == 1){
	//lcd_clear_display();
	for(int i = 0; i < strlen(fail_2_msg); i++){
		lcd_data(fail_2_msg[i]);
	}
	goto start;
}
else{
	//lcd_clear_display();
	lcd_data('!');
	goto start;
}

_delay_ms(3000);
```

```
	reset:
	lcd_clear_display();
	while(keypad_to_ascii() != '8'){
		/*Get a measurement from thermometer as in exercise 7*/
		uint16_t temperature = get_temperature_reading();
		uint8_t patient_temperature_int = integer_part(temperature) + 10;
		uint8_t patient_temperature_dec = decimal_part(temperature);

		int status = 0;

		/*Get ADC reading*/
		uint16_t patient_pressure_adc = adc_read();
		int patient_pressure = patient_pressure_adc / 51;

		if(patient_temperature_int > 36 || patient_temperature_int < 34){
			status = CHK_TEMP;
		}

		if(patient_pressure > 12 || patient_pressure < 4){
			status = CHK_PRESSURE;
		}

		// Start the JSON payload
		snprintf(payload, sizeof(payload), "[");

		// Add each JSON object to the payload, these are similar for all status
		snprintf(temp, sizeof(temp), "{\"name\": \"team\",\"value\": \"58\"},");
		strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

		snprintf(temp, sizeof(temp), "{\"name\": \"temperature\",\"value\":
\"%d.%d\"},",patient_temperature_int, patient_temperature_dec);
		strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

		snprintf(temp, sizeof(temp), "{\"name\": \"pressure\",\"value\":
\"%d\"},", patient_pressure);
		strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);
		/* Send appropriate payload */
		switch(status){
			case STATUS_OK:
				snprintf(temp, sizeof(temp), "{\"name\":
\"status\",\"value\": \"OK\"}");
				strncat(payload, temp, sizeof(payload) - strlen(payload) -
1);
				break;
			case CHK_TEMP:
				snprintf(temp, sizeof(temp), "{\"name\":
\"status\",\"value\": \"CHECK TEMP\"}");
				strncat(payload, temp, sizeof(payload) - strlen(payload) -
1);
				break;
			case CHK_PRESSURE:
				snprintf(temp, sizeof(temp), "{\"name\":
\"status\",\"value\": \"CHECK PRESSURE\"}");
				strncat(payload, temp, sizeof(payload) - strlen(payload) -
1);
```

```
                        break;
                default:
                        break;
        }

        strncat(payload, "]", sizeof(payload) - strlen(payload) - 1);
        ret = esp_send_command(CMD_PAYLOAD, payload, strlen(payload));
        if(ret == 0){
                for(int i = 0; i < strlen(success_3_msg); i++){
                        lcd_data(success_3_msg[i]);
                }
        }
        else if(ret == 1){
                for(int i = 0; i < strlen(fail_3_msg); i++){
                        lcd_data(fail_3_msg[i]);
                }
                goto reset;
        }
        else{
                lcd_data('!');
                goto reset;
        }

        _delay_ms(3000);
        lcd_clear_display();

        ret = esp_send_command(CMD_TRANSMIT, transmit_response,
sizeof(transmit_response) -1);
        lcd_data('4');
        lcd_data('.');
        for(int i = 0; i < ret; i++){
                lcd_data(transmit_response[i]);
        }


        _delay_ms(3000);
        lcd_clear_display();
        continue;

nurse:
        while(keypad_to_ascii() != '#'){

                uint16_t temperature = get_temperature_reading();
                uint8_t patient_temperature_int = integer_part(temperature) + 12;
                uint8_t patient_temperature_dec = decimal_part(temperature);

                /*Get ADC reading*/
                uint16_t patient_pressure_adc = adc_read();
                int patient_pressure = patient_pressure_adc / 51;
                snprintf(payload, sizeof(payload), "[");

                // Add each JSON object to the payload, these are similar for all
status
```

```
                    snprintf(temp, sizeof(temp), "{\"name\":
\"temperature\",\"value\": \"%d.%d\"},",patient_temperature_int,
patient_temperature_dec);
                    strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

                    snprintf(temp, sizeof(temp), "{\"name\": \"pressure\",\"value\":
\"%d\"},", patient_pressure);
                    strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

                    snprintf(temp, sizeof(temp), "{\"name\": \"team\",\"value\":
\"58\"},");
                    strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);

                    snprintf(temp, sizeof(temp), "{\"name\": \"status\",\"value\":
\"NURSE CALL\"}");
                    strncat(payload, temp, sizeof(payload) - strlen(payload) - 1);
                    strncat(payload, "]", sizeof(payload) - strlen(payload) - 1);
                    status = CALL_NRS;
                    ret = esp_send_command(CMD_PAYLOAD, payload, strlen(payload));
                    if(ret == 0){
                          for(int i = 0; i < strlen(success_3_msg); i++){
                                lcd_data(success_3_msg[i]);
                          }
                    }
                    else if(ret == 1){
                          for(int i = 0; i < strlen(fail_3_msg); i++){
                                lcd_data(fail_3_msg[i]);
                          }
                          goto reset;
                    }
                    else{
                          lcd_data('!');
                          goto reset;
                    }

                    _delay_ms(3000);
                    lcd_clear_display();

                    ret = esp_send_command(CMD_TRANSMIT, transmit_response,
sizeof(transmit_response) -1);
                    lcd_data('4');
                    lcd_data('.');
                    for(int i = 0; i < ret; i++){
                          lcd_data(transmit_response[i]);
                    }

                    _delay_ms(3000);
                    lcd_clear_display();

              }
              status = STATUS_OK;
              goto reset;
        }
        goto nurse;
}
```