ELEC-C5070 – Elektroniikkapaja

Loppuraportti
GSM-käynnistin

Syksy 2016 - Ryhmä numero 41

Tuomas Manninen

1 JOHDANTO

Projektin tarkoituksena oli rakentaa sulautettu järjestelmä, jonka suunnitteluvaiheet tehdään niin pitkälle valmiiksi tuotteeksi asti kuin mahdollista.

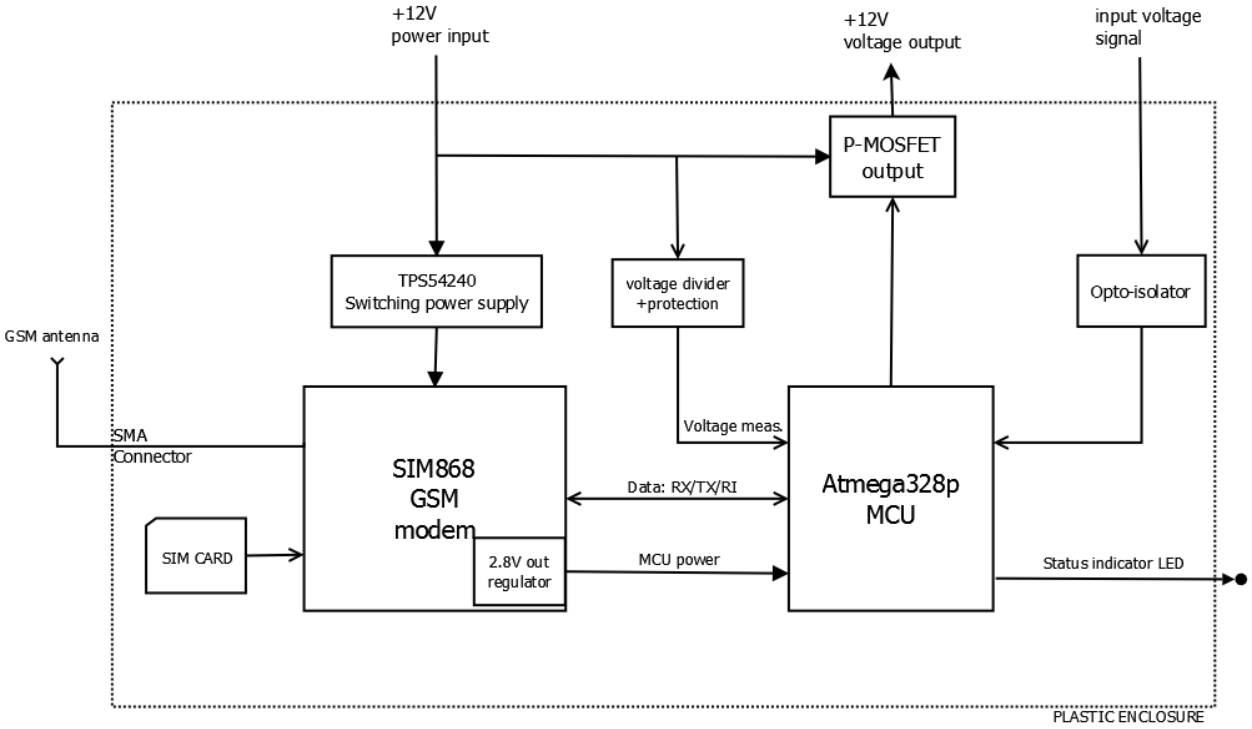
Valmistunut lopputulos on ”GSM-käynnistin”-niminen jälkiasennettava laite, joka mahdollistaa erilaisten laitteiden päälle kytkemisen matkapuhelimen soiton avulla. Laiteen avulla on tarkoitus kytkeä erityisesti auton esilämmitintä, mutta sitä tulisi voida soveltaa pienin muutoksin myös esimerkiksi saunan, porttien, ilmanvaihdon yms. kytkentäsignaalina. Erilaista signaalia tarvitsevien laitteiden ajaminen onnistuu kytkemällä laitteen 12V ulostulo sähköturvalliseen relepakettiin.

Laite tarvitsee toimiakseen 6-20V jännitelähteen ja yhteyden matkapuhelinverkkoon.

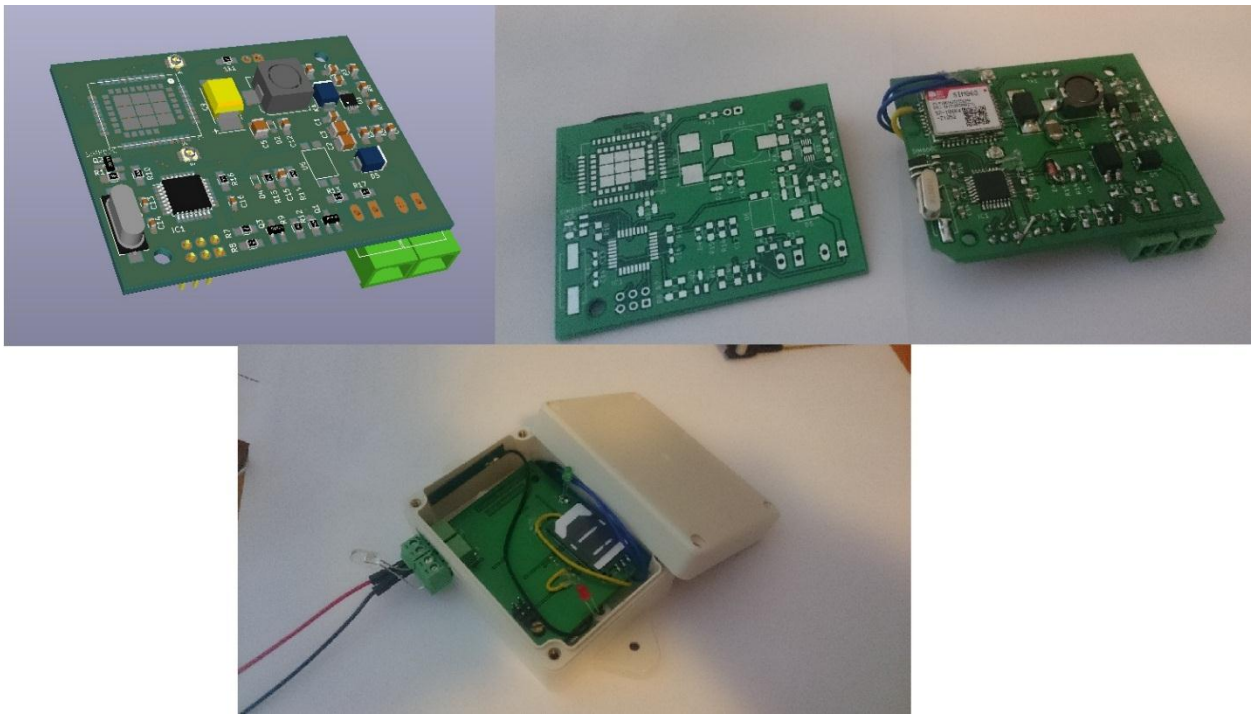
2 TOTEUTETTU LAITTEISTO

Kuvassa 1 on esitetty laitteen lohkokaavio. Laitteen ulkoiset kytkennät on toteutettu Phoenix-liittimien avulla, jossa on ruuviterminaali käyttäjää varten. Laitteen sulake (2A lattasulake) asetetaan syöttöjohtoon käyttäjän haluamaan kohtaan, joka myös mahdollistaa laitteen kytkemisen tarvittaessa irti järjestelmästä.

Laite toimii vain 2G verkossa ja vaatii siihen soveltuvan SIM-kortin.



Kuva 1: Toteutunut lohkokaavio



Kuva 2: Piirilevysuunnitelma ja toteutunut laite

Kuvassa 2 on esitetty laitteen eteneminen suunnitelmasta piirilevyyn, kasaamiseen ja kokoamiseen.

Piirilevyn koko on 64x50mm ja kotelon koko on 83x57mm. Kotelo on avattavissa ruuveilla ja on roisketiivis ympäristöltä. Laite kiinnitetään tasaiselle pinnalle muovisten kiinnityskorvakoiden avulla. Laitteen piirikaavio on esitetty liitteessä 3.

3 MITTAUKSET JA TESTAUS

Yleismittarilla mitattuna laitteen virrankulutus on lepotilassa 1.6mA @12V. Tähän ei ole kuitenkaan ole laskettu mukaan GSM-modeemin vetämiä hetkittäisiä virtapiikkejä, joten se ei ole keskimääräinen virrankulutus. GSM-modeemi on asetettu toimimaan virransäästötilassa, jolloin sen sarjaväylä vaatii herätteen [2]. Lisäksi mikrokontrolleri asetetaan unitilaan noin 5 sekunnin toimettoman ajan jälkeen.

Ohjelmiston suorituskyvyn testauksessa modeemille lähetettiin tuntemattomia komentoja, joihin laite reagoi odotetusti resetoimalla itsensä. Ohjelmiston suorituskyky varmistetaan Watchdog-ajastimella, joka resetoi järjestelmän jos sitä ei nollata 4 sekunnin sisällä, eli jos ohjelmisto jumittaa mistä tahansa syystä tuntemattomaan tilaan.

Lämpötilan ja ympäristön testaus suoritettiin pakastimen lämpötilassa. Tärinätestausta ei voinut suorittaa koska kiinnitysreiät eivät sijainneet oikeilla paikoilla, joten piirilevyä ei saatu kiinni koteloon. Säänkestävyys tulisi suorittaa suojalakatun piirilevyn avulla.

Ulostuloa tulee käyttää vain signaalijännitteenä, sillä virrankesto ei ole kovin suuri. Tämänhetkisen piirilevyn P-MOS kestää vain alle 100mA virtaa, joten sillä ei voida ajaa suoraan releen käämiä.

Modeemin ja mikrokontrollerin välinen tiedonsiirron robustisuus ei ole paras mahdollinen, sillä rautapohjaista vuonohjausta ei kytketty. Tällä hetkellä ohjelmisto hylkää kaikki bitit jotka saapuvat muun prosessoinnin aikana, eikä siis voi asettaa GSM-modeemia jonotustilaan.

4 JOHTOPÄÄTÖKSET

Työ saavutti perustoimivuudeltaan asetetut vaatimukset. Laitteen soittotoiminto toimi odotetusti, sekä onnistuneesti kytki Webaston päälle ja sulki luurin toiminnan varmistamisen merkiksi. Tekstiviestitoiminto toimi odotetusti, eli palautti lämpötilan ja akun jännitteen. Lisätoiminnoista GPS:n käyttäminen jäi mahdottomaksi, koska piirilevystä jäi GPS:n käynnistysnapa piirin alapuolelta kytkemättä. Ohjelmiston viimeistely jäi heikoksi, sillä ohjelmointilaitteessani debugaaminen ei onnistu, joten koodin kehittäminen, vianetsintä ja toiminnan varmistaminen on todella hidasta.

Projektin edetessä huomasin, että alustan monipuolistamiseksi voisi olla hyödyllisempää käyttää laajempaa mikrokontrollerialustaa. Esimerkiksi STM32F1xx-sarjan ARM-mikrokontrollerit olisivat sopineet lähes samaan hintaluokkaan, mutta lisänneet esimerkiksi CAN-väylän ja laajemmat prosessointimahdollisuudet. Projektin valmistumisen kannalta oli kuitenkin hyvä valita tuttu softausympäristö, jolla suunnitellun piirilevyn perustoimivuus voitiin varmistaa. Mikrokontrollerin vaihto tässä vaiheessa on vielä yksinkertainen muutos.

5 LÄHTEET

1. SIMCOM SIM868, *Quad-Band GSM/GPRS module*, http://simcomm2m.com/UploadFile/TechnicalFile/SIM868_Hardware_Design_V1.01.pdf
2. Atmel ATmega328, *8-bit AVR microcontroller*, http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf
3. TI TPS54240, *3.5-V to 42-V Step-Down DC - DC Converter*, http://www.ti.com/lit/ds/symlink/tps54240.pdf
4. SIM800 Series Serial Port Application https://cdn-shop.adafruit.com/product-files/1946/SIM800+Series_Serial+Port_Application+Note_V1.01.pdf
5. SIM800 Series AT Command manual https://cdn-shop.adafruit.com/datasheets/sim800_series_at_command_manual_v1.01.pdf

LIITE 1: TYÖNJAKO JA MITÄ OPITTIIN

Opin huomattavan paljon AVR C-ohjelmoinnista, piirilevyn ja piirikaavion suunnittelusta, vianetsinnästä, datalehtien lukemisesta sekä tyypillisistä huolimattomuusvirheistä. Lähtötasolta olen jonkin verran kokenut harrastetasolla digitaalielektroniikkaa käyttävien laitteiden rakentelusta ja suunnittelusta.

Laitteen kasaamisen työnjako oli sopivasti ryhmitelty segmentteihin. Alustavan piirikaavion suunnittelun ja prototyypin jälkeen suoritetaan ohjelmointi ja piirilevysuunnittelu rinnakkain. Tämän jälkeen laite on kasattavissa ja ohjelmiston kehitystä voidaan jatkaa valmistetulla piirilevyllä.

Tavoitteeni oli syventää osaamistani sulautetun järjestelmän suunnittelussa ja käyttää Arduinon ja nettioppaiden sijaan aitoa kehitysympäristöä ja etsiä tarvittava informaatio datalehdistä. Sähköpaja-kurssista poiketen en tehnyt myöskään vain prototyyppiä, vaan tavoite oli luoda kokonaisuutena valmistettavissa oleva tuote.

Suurin oppimiskokemus tuli piirikaavion viimeistelyssä. Piirikaaviosta johtuvien huolimattomuusvirheiden korjaamiseen piirilevyltä meni noin 3 päivää, jonka olisi voinut välttää tunnin kestävällä tarkastuksella. Myös paljon hyödyllisiä lisäkytkentöjä jäi kokonaan kytkemättä. Kytkennän yksinkertaisuuden vuoksi erehdyin jättämään lopputarkastuksen väliin, joka oli suuri virhe. Jatkossa tulen tarkistamaan jokaisen vedon, sekä varmistan tarvitaanko niihin esimerkiksi ylös- tai alasvetoja.

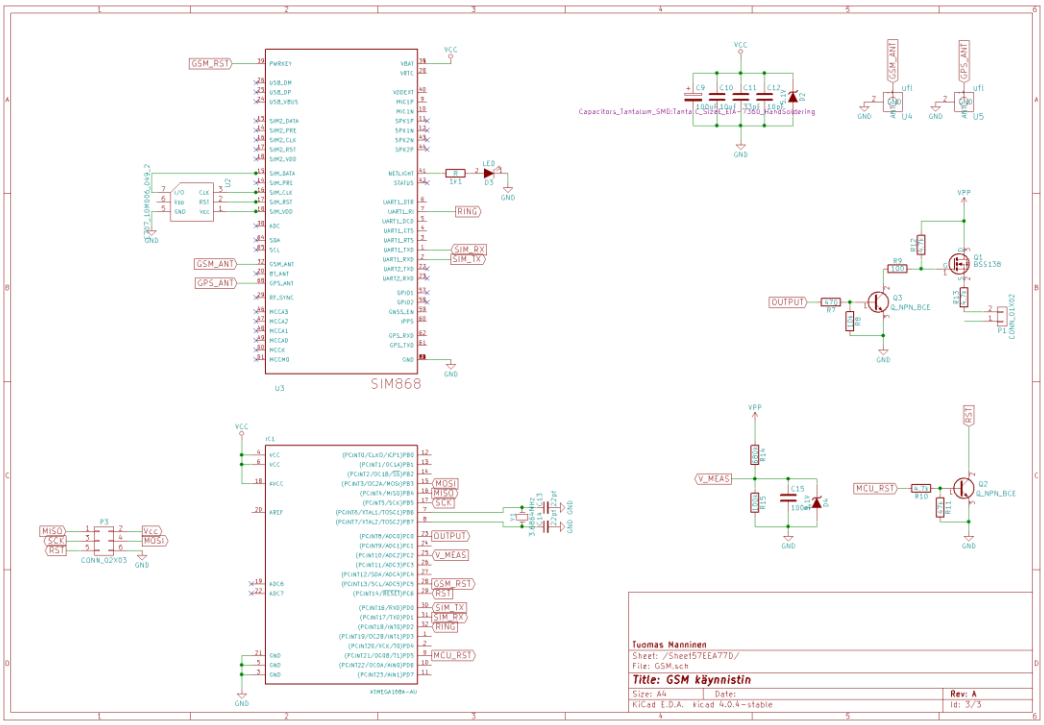
Huomasin myös, että vianetsinnän helpottamiseksi on kannattavaa vetää reilusti ylimääräisiä ulostuloja ainakin merkkivaloille ja datansiirtoportteille, jotta näiden toiminnan voi tarkistaa ilman muutoskytkentöjä.

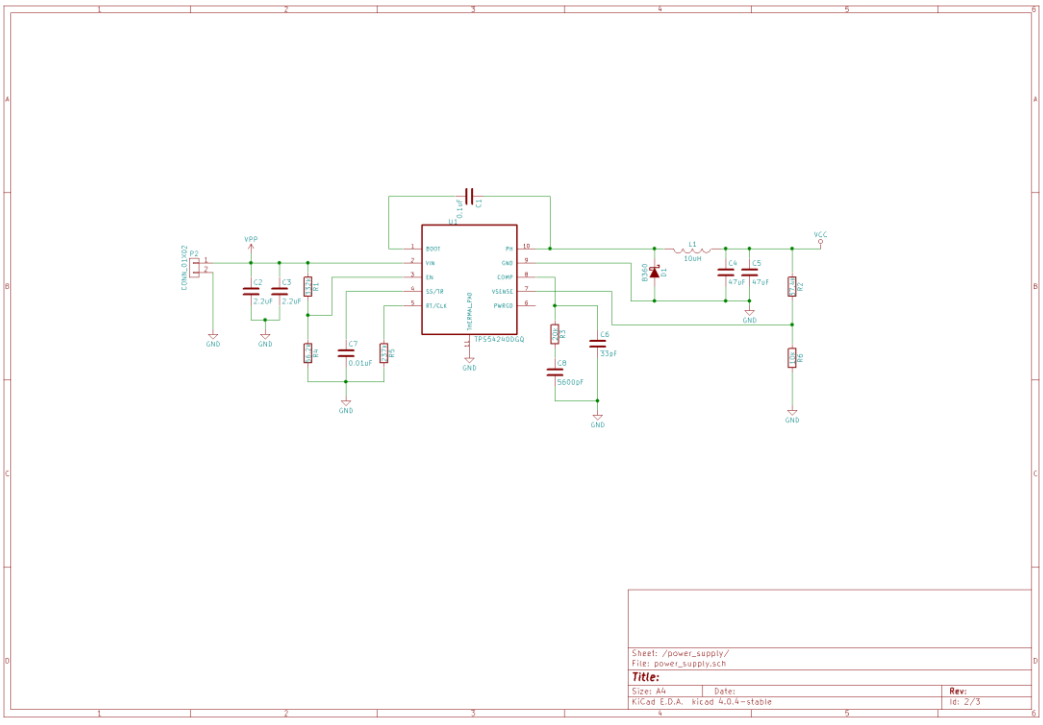
Hyödyllistä oli myös saada käytännön kokemusta LGA-pintaliitoskomponentin juottamisesta reflow-uunilla, jota en ole aikaisemmin päässyt tekemään.

Parannettavaa on edelleenkin projektin ajanhallinnassa. Suurin osa työstä tuli tehtyä lähinnä vain väli- ja loppuesittelyä edeltävällä viikolla. Vaikka aikataulu oli tehty, sen vapaaehtoinen seuraaminen ei muiden pakollisten aikataulujen rinnalla onnistunut.

Projektin suurin oppimiskokemus tuli saatua eniten kaikissa pieleen menneissä osuuksissa. Elektroniikan suunnittelua syventävät tavoitteeni onnistuivat mielestäni hyvin, sillä opin juuri sen mitä lähdin tavoittelemaan.

LIITE 2: PIIRIKAAVIO





LIITE 3: OHJELMISTOLISTAUS

```
#define F_CPU 3686400L
#define BAUD 9600
#define BRC (F_CPU/16/BAUD-1)
#define BUF_SIZE 64
#define PIN_GSMRST PINC5
#define PIN_OUT PINB5
#define PIN_LED PIND3

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>
#include <util/delay.h>
#include <string.h>
#include <stdlib.h>
#include <avr/sleep.h>

void init_AVR();
void send_data(const char* data);
void send_char(const char data);
char get_status(char* msg);
char get_ctrlz();
char get_response(char* response);
void clear_buffer();
void timer1_mode(const char mode);
char send_SMS();
char get_sms();
void avr_sleep();

//global variables
char* phone_num="+358505532822";
char set_output=0;

//volatiles for ISR
volatile char rx_buffer[BUF_SIZE];
volatile unsigned char data_index;
volatile unsigned char command_ready;
volatile unsigned int timer_count;

//setup commands
//AT+CMGF=1 :SMS text mode


//AT+IFC=1,1 software flow control
//send_char(19); XOFF
//send_char(17); XON

int main(void)
{
	init_AVR();

	send_char(27);

	do{
		send_data("AT+CREG?\r");
		_delay_ms(2000);
	} while(!get_response("0,1"));

	send_data("AT+CMGF=1\r");
	get_response("OK");

	send_data("AT+CNMI=2,2,0,0,0\r");
	get_response("OK");

	send_data("AT+CSCLK=2\r");
	get_response("OK");

	send_data("AT+CNETLIGHT=0\r");
	get_response("OK");

	PORTC |= (1 << PIN_GSMRST); //disable gsm reset
	PORTD |= (1 << PIN_LED);


	/* MAIN LOOP */
	while (1)
	{
		wdt_reset();

		if (command_ready){
			command_ready=0;

			if (strstr((char*)rx_buffer,"RING")){
				PORTB |= (1 << PIN_OUT);
				timer1_mode(1);
				set_output=1;
			}
```

```
                                        if (strstr((char*)rx_buffer,"+CMT")){
                                                PORTB |= (1 << PIN_OUT);
                                        }
                                }

                                if (set_output==1){
                                        if (timer_count==2){
                                                send_data("ATH\r\n"); //hang up
                                                if (get_response("OK")) timer_count++; //exit if condition
                                                EIMSK |= (1 << INT0); // Enable INT0
                                        }

                                        if (timer_count>12){
                                                PORTB &= ~(1 << PIN_OUT);
                                                timer1_mode(1);
                                                set_output=0;
                                        }
                                }
                                else{
                                        if (timer_count>4) avr_sleep();
                                }
                        }
}

void avr_sleep(){
                        PORTD &= ~(1 << PIN_LED);

                        set_sleep_mode(SLEEP_MODE_PWR_DOWN);
                        cli();

                        sleep_enable();
                        sei();
                        sleep_cpu();
                        sleep_disable();

                        sei();

                        timer1_mode(1);
                        PORTD |= (1 << PIN_LED);
}

char get_sms(){

                        char cmd_buffer[BUF_SIZE];
                        strcpy(cmd_buffer,(const char*) rx_buffer);
                        data_index=0;

                        send_data(cmd_buffer);

                        //char* ptr=strchr(cmd_buffer,'\n')+1; // find ',' from +CMTE: 0,26.17

                        return 1;

                        //+CMT:"+358505532822","","16/12/06,22:44:42+08"\nnumero1
}

void clear_buffer(char* buffer){
                        memset((char*)&buffer[0], 0, sizeof(*buffer));
}

void send_char (char data)
{
                        while ( !( UCSR0A & (1<<UDRE0)) ); // Wait for empty transmit buffer
                        UDR0 = data;  // Send data
}

void send_data(const char *data){
                        while(*data) send_char(*data++);
}

ISR(USART_RX_vect){
                        //wait for data
                        while ( !(UCSR0A & (1<<RXC0)) );

                        if (!(UCSR0A & ((1<<FE0) | (1<<DOR0) | (1<<UPE0)))) //check errors for frame, overrun, parity
                        {

                                if (data_index>BUF_SIZE-1) data_index=0;

                                rx_buffer[data_index] = UDR0;

                                if (rx_buffer[data_index]=='\r'){
                                        command_ready=1;
                                }
                                else{
                                        data_index++;
                                }
                        }
}

//timer 1 interrupt
```

```
ISR(TIMER1_OVF_vect) {
		timer_count++;
}

//RING interrupt
ISR(INT0_vect) {
		PORTD |= (1 << PIND3);
		EIMSK &= ~(1 << INT0); // Disable INT0
}

char get_response(char* response){

		unsigned int timeout=0;

		do{
			_delay_ms(500);
			if (++timeout>8 || strstr((char*)rx_buffer,"ERROR")) return 0;
		}
		while(!strstr((char*)rx_buffer,response));

		data_index=0;

		return 1;
}


char send_SMS(){
		char status_msg[30];
		get_status(status_msg);

		char msg[30];
		strcpy(msg,"AT+CMGS=\"");
		strcat(msg, phone_num);
		strcat(msg, "\"");

		send_data(msg);
		send_data("\r\n");
		get_response(">");

		send_data(status_msg);

		send_char(27);

		return get_response("OK");
}


char get_status(char* msg){

		//get temperature
		send_data("AT+CMTE?\r\n");
		char check=get_response("OK");

		if (check==0) return 0;

		char temp_str[3]="";
		char volt_str[3]="";
		char* ptr=strchr((const char*)rx_buffer,',')+1; // find ',' from +CMTE: 0,26.17

		strncpy(temp_str, ptr, 2); //get temp string

		//get ADC
		ADMUX=0x00; //PC0

		ADCSRA |= (1<<ADSC);

		while(ADCSRA & (1<<ADSC));

		int voltage=ADCW; //12.5V => 490 adc value
		itoa(voltage,volt_str,10);

		char buf[30];
		strcpy(buf,"Lampotila: ");
		strcat(buf, temp_str);
		strcat(buf,"c \nAkun jannite: ");
		strcat(buf, volt_str);

		strcpy(msg,buf);

		return 1;
}



void init_AVR(){

		/* INIT WDT		*/
		/*
		//set up WDT interrupt
		WDTCSR = (1<<WDCE)|(1<<WDE);
		//Start watchdog timer with 4s prescaler
		WDTCSR = (1<<WDIE)|(1<<WDE)|(1<<WDP3); //(1<<WDP0)
```

```
	*/
	MCUSR = 0; // clear reset flags
	wdt_disable();

	cli(); // disable global interrupts

	//wdt_enable(WDTO_4S);

	/* INIT PORTS */

	//DDRx: 1 output, 0 input

	//B: 5 OUTPUT
	DDRB = (1 << PIN_OUT);
	PORTB = 0x00;

	//C: 0 MEAS(in), 1 IN_SIG(in), 5 GSM_RST(out)
	DDRC = (1 << PIN_GSMRST);
	PORTC = 0x00;

	//D: 3 TEST_LED(out), 2 RING(in)
	//D3
	DDRD = (1 << PIN_LED);

	//D2 (ring interrupt)
	DDRD &= ~(1 << PIND2); //set to input
	PORTD |= (1 << PIND2); //set to 0: tristate


	EIMSK &= ~(1 << INT0); // Disable INT0
	EIMSK |= (1 << INT0);  // Enable INT0
	// trigger in low level, ISC0:00


	/* INIT ADC */

	ADMUX = (1<<REFS0); // AREF = AVcc

	// ADC Enable and prescaler of 128
	// 3.6MHz/32 = 112500Hz
	ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS0);


	/* INIT TIMER */

	TIMSK1 = (1 << TOIE1); // enable Timer1 overflow interrupt:
	timer1_mode(1);


	/* INIT USART */

	//Set baud rate
	UBRR0H = (BRC>>8);
	UBRR0L = BRC;

	//Enable transmitter and receiver and RX complete interrupt
	UCSR0B = (1<<TXEN0)|(1<<RXEN0)|(1<<RXCIE0);

	//Set frame format: 8N1
	UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);

	sei(); // enable global interrupts
}


//0: off & reset, 1: on
void timer1_mode(const char mode){
	timer_count=0;
	TCCR1A = 0;
	TCCR1B = 0; //stop timer
	TCNT1 = 0; //reset timer

	if (mode==1)
	{
		TCCR1B |= (1 << CS10) | (1 << CS11); //start timer
		//duration: 1/(f/prescaler)*2^timer_bits
	}
}
```