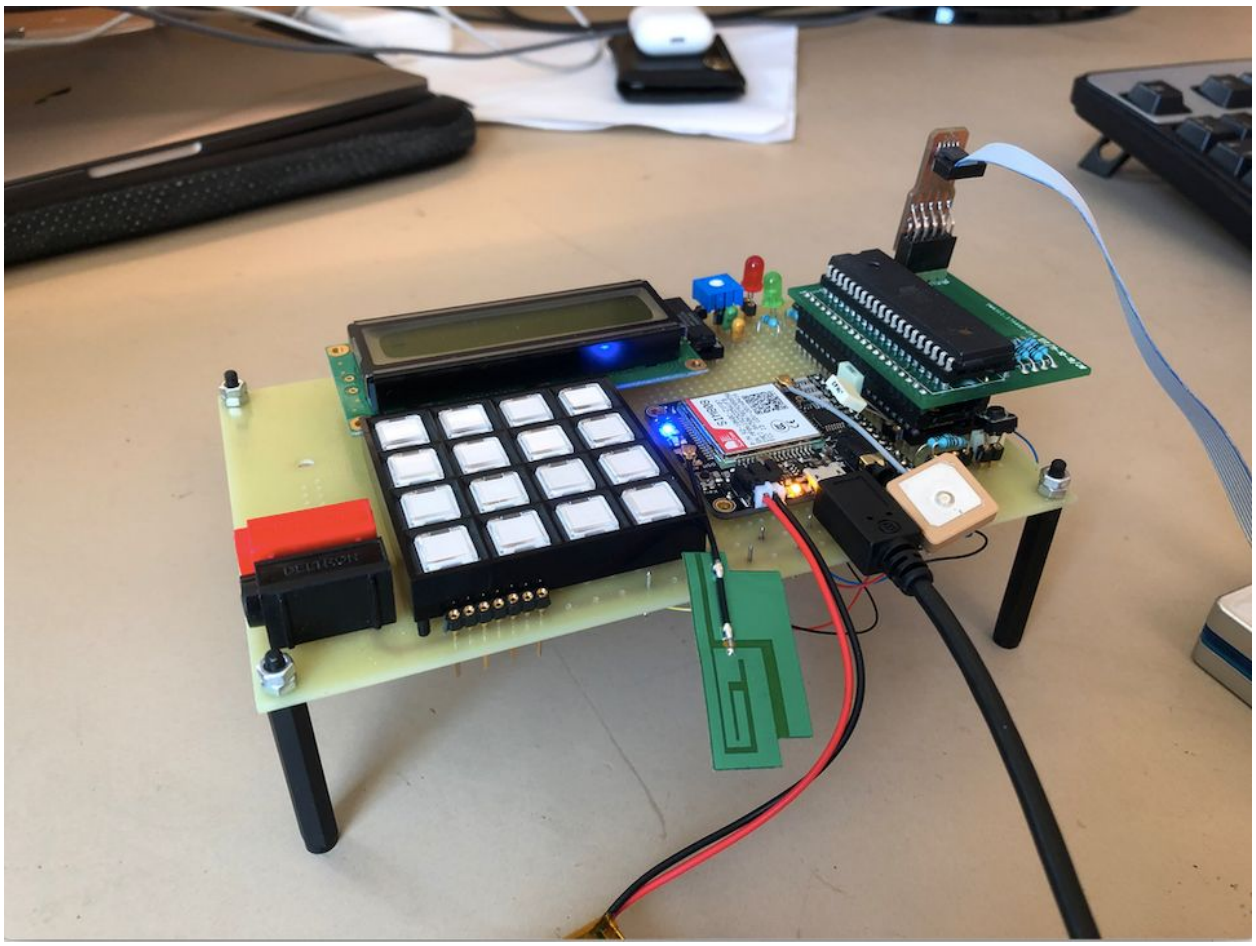
TRACK-A-CAT

Projektarbete i Digitala Projekt, Lunds Tekniska Högskola

Studenter: Erik Busk, Markus Nilsson, Viktor Stenwall
Handledare: Christoffer Cederberg och Bertil Lindvall
Kurs: EITF12
Datum: 2019-05-21



Abstract

This report describes the process of developing the hardware and software of a cat-tracker. The report is part of a project within Digital Projects at Lund University. Firstly, a circuit diagram of all necessary connections and hardware was made. The model was wired and soldered according to the circuit diagram previously mentioned. The hardware was then tested in order to guarantee that all the correct wiring had been made. Thirdly all the necessary coding was conducted in Atmel Studio coded in C to make all hardware communicate and deliver desired results. Thereafter the product was tested in a practical test. The GNSS-tracker used in this project was SIM808 with an external antenna. The SIM808 module communicated with the microprocessor through UART. The project uses JTag for debugging. The result was a product that meet all the requirements. For similar projects conducted in the future it should be considered that it is difficult to test the tracker indoor due to lack of GNSS connection.

Innehållsförteckning

1.Inledning

1.1 Projektbeskrivning

Att koppla samman hårdvara och mjukvara är av största värde för att skapa användbara produkter. I kursen digitala projekt skall en prototyp för en produkt byggas så att den fungerar, med både hårdvara och mjukvara. Syftet med detta projekt är att konstruera en tracker ämnad för en katt. Med trackern ska man kunna markera en hemposition där katten bor samt ställa in en maximal distans från hempositionen som katten får befinna sig. Om katten rör sig längre bort än den maximala distansen ska prototypen varna genom att tända en röd lampa. För att åstadkomma detta används en tracker som håller reda på positionen av katten, samt diverse funktioner för att avgöra huruvida katten är på rymmen. Programmet kodas i Atmel Studio med programspråket C. För felsökning används JTag.

1.2 Kravställning

1. När Track-A-Cat först sätts på skall den gröna dioden blinka till (en sekund) som en startsignal.
2. Track-A-Cat skall kunna läsa av kattens nuvarande position och sätta en hemposition.
3. Genom att trycka knappen “SD” skall prototypen kunna sätta maxdistansen katten får gå från hempisitionen.
4. En lampa på prototypen skall lysa rött då katten är längre bort från hempositionen än maxdistansen. Detta för att uppmärksamma om katten gått för långt hemifrån. Om katten befinner sig inom gränsen för maxdistansen skall den gröna lampan lysa.
5. Prototypen skall även ha funktioner för att testa trackern och hårdvarans funktion så den är tillförlitlig då katten släpps iväg.

2. Teori

2.1 Hårdvara

2.1.1 Processor

Processorn är komponenten som styr alla andra komponenter i modellen. Processen styrs genom att man programmerar i Atmel studios och sedan använder en JTAG ICE för att debugga modellen.

Den processorn som används är ATMega16 med 40 pinnar, vilket täcker Track-A-Cats behov. ATMega16 är en MPU (Micro Processing Unit). Den kan ta emot digitala signaler och omvandla analoga signaler till digitala med en Analog till digital converter.

2.1.2 Display

En display av typen GDM1602K från Xiamen ocular optics Co används för att kunna skriva ut meddelanden och för att visa de siffror som användaren skrivit in. Displayen kan skriva ut 32 synliga tecken samtidigt, vilket är tillräckligt för Track-A-Cats behov. Displayen är en alfanumerisk LCD-display och kontrasten är justerbar med hjälp av en inkopplad resistor.

2.1.3 Tracker

Trackern som används är SIM808. Den har funktioner för bl.a att skicka nuvarande koordinater, testa uppkoppling etc. Den kommunicerar med processorn med hjälp av UART och kan använda två olika satellitnavigationssystem, GPS och GNSS. För projektet används GNSS eftersom i detta system skrivs koordinaterna på ett format som var lättare att räkna på.

2.1.4 Knappsats

En 16 knappars knappsats används vilket nästan täcker Track-a-cats behov. Egentligen krävdes 17 knappar men eftersom endast 16 knappar finns tillgängliga på knappsatsen har knappen för att skriva siffran 9 tagits bort och blivit en hempositionsknapp. Knappsatsen kopplas till processorn via en encoder som omvandlar input från knappsatsen till binära värden för att programmet skall veta vilken knapp som trycks ner. Encodern som används är MM54C922.

2.1.5 JTag

En JTag fungerar som en länk mellan datorn och den konstruerade hårdvaran och används bland annat för testning, debugging och felsökning. Utöver detta används den för att föra över kod som skrivs på dator till prototypens processor.

2.1.6 Dioder

En röd och en grön diod används för att signalera att apparaten är påslagen och ifall katten är i (grön) eller utanför det tillåtna området (röd). För att dessa ska förses med lämpligt strömflöde har de kopplats till en resistans.

2.1.7 Strömförsörjning och jord

Hårdvaran förses med ström via en micro-USB-kabel samt ett 5V batteri och jordas via minuspolen på batteriet.

2.1.8 Övrigt:

Utöver det som tidigare nämnts behövs ett antal motstånd, kondensatorer och reset-knappar för att all hårdvara skall fungera korrekt, se appendix 6.1.

2.2 Mjukvara

För att styra hårdvaran så har ett antal klasser med källkod skrivits. Detta kapitel är uppdelat på liknande sätt som klasserna i koden.

2.2.1 Tracker (sim808.c samt uart.c)

Trackern kommunicerar med processorn via UART, därav måste ett antal inställningar göras för att de skall kommunicera korrekt. Därefter kopplas trackern till närmaste basstation och GPS/GNSS-funktionen sätts på. Då den returnerar “Location fix” kan rätt koordinater skickas med GNSS. Både koordinater och svar på om trackern har kontakt skickas som en sträng som även innehåller annan information än bara den som behövs för Track-a-cat. På grund av detta måste koordinaterna klippas ut från den sträng som erhålls. I varje steg måste anropen kallas enligt reglerna för UART för att sedan få svar och ett “OK” om kommandot genomförts korrekt, vilket också kan klippas ut från

strängen som returneras. Det måste även finnas kod för att beräkna huruvida katten befinner sig inom tillåtet avstånd. Detta görs genom att jämföra koordinaterna för hempositionen med aktuell koordinat. För att kunna jämföra med maxavståndet som anges i meter omräknas avståndet mellan koordinaterna i meter. Om det är mindre än kattens maximalt tillåtna avstånd (default 100m) så lyser den gröna dioden, annars den röda.

2.2.2 Knappar (keyboard.c)

Se appendix 6.2 för knapparnas funktioner. Då en knapp trycks ner returneras ett unikt värde till processorn. Genom att skapa olika case där varje värde associeras med en knapp, och för varje case genomföra ett eller flera kommandon, kopplas knappsatsen till de rätta funktionerna.

2.2.3 Display (display.c)

Displayen måste först initieras genom att göra ett antal inställningar. Dessa innefattar bland annat att aktivera båda textraderna och sätta markören längst till vänster. Därefter går det att skriva ut på skärmen.

2.2.4 Lysdioder (led.c)

Lysdioderna skall kodas så att de lyser enligt kravställningen. Gröna dioden skall lysa då programmet startas samt då katten befinner sig innanför maxavståndet från hempunkten. Den röda dioden skall programmeras så att den lyser då katten befinner sig utanför maxavståndet.

2.2.5 Huvudprogram (main.c)

I huvudprogrammet anropas metoder för att initiera displayen och SIM808. Dessutom sätts en hemposition samt ett antal variabler skapas. Därefter inleds en loop som varje varv inväntar knapptryckningar och signaler, uppdaterar aktuella koordinaten och jämför om katten är inom tillåtet avstånd från hempunkten.

3. Genomförande

3.1 Kopplingsschema

För genomförandet gjordes ett kopplingsschema över hårdvaran där respektive komponents datablad användes för att avgöra vilka pinnar som skulle utnyttjas, vilka ytterligare komponenter som kondensatorer och motstånd som skulle användas, samt vilka kopplingar som skulle göras. Se appendix 6.1 för kopplingsschemat.

3.2 Montering

Alla komponenter sattes fast på en platta, löddes fast där nödvändigt och virades enligt kopplingsschemat. Därefter testades hårdvaran genom att konstruera enkla program, samt att använda debugger och logikpenna.

3.3 Programmering

Klasser för respektive avsnitt i kapitel 2 kodades. Först initierades displayen, sedan knappsatsen och dioderna. Efter det upprättades kommunikation med SIM808-komponenten via UART och slutligen kodades huvudprogrammet så allt samverkade.

3.3 Praktiskt test

För att testa programvaran startades produkten, alla funktionstester på trackern gjordes och den fick tid att fixera koordinater nära ett fönster. Inledningsvis sattes test-koordinater för att se att algoritmerna fungerade. Därefter sattes hempunkten som test-koordinat, men nuvarande position som den faktiska trackerns koordinat, varefter tester gjordes med olika max-avstånd. Slutligen testades produkten utomhus med både koordinaterna som verkliga. Ett maxavstånd om 20m sattes och produkten förflyttades 20m, varefter lampan lyste rött. Testet upprepas med andra avstånd. Därefter betraktades prototypen som fungerande.

4. Resultat

Track-A-Cat fungerade som förväntat vid det praktiska testet, vid olika maxavstånd och olika hempunkter lös dioderna i enlighet med kravställningen. Alla funktioner som presenterades i kravställningen fungerade. Precision av trackern var inte mer än ungefär 3 meter så den minsta maxdistansen som modellen fungerar godtyckligt för är 3 meter. Under början av testfasen klipptes resetknappen då det troddes att den orsakade fel. Knappen kopplades inte tillbaka igen efter detta så resetfunktionen fungerar ej.

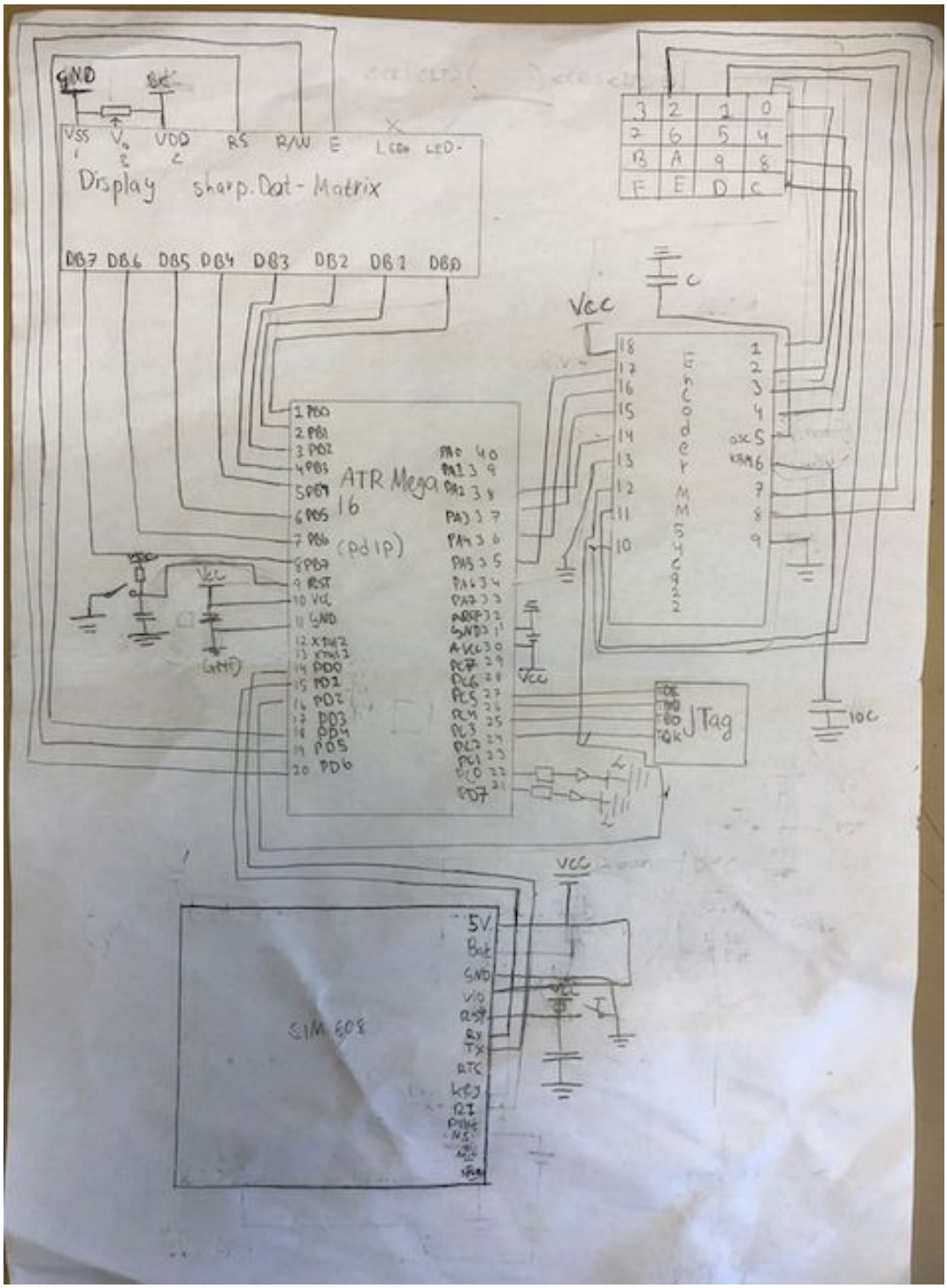
5. Diskussion och slutsats

Projektet var både utmanande och stundvis frustrerande men även mycket lärorik. Det var framförallt tolkning av datablad samt SIM808 komponenten som var det svåraste. När man väl satte sig in i projektet och kom över den första kunskaps-tröskeln blev allting lättare.

Det hade behövts fler knappar eftersom det krävdes fler tester för att få igång SIM808 komponenten än förväntat. I samråd med handledare slopades en sms-funktion som fanns med i konceptstadiet då knapparna tog slut. Det var även lite utmanande att få GNSS:en att funka då den inte har täckning inomhus, samt en fixeringstid som varrierar. Den fick därför hängas ut genom fönstret och bidrog till en del väntetid. Projektet var stundtals utmanande i och med att större delarna av ett datablad inte är intressant för en specifik applikation. Ingen i gruppen hade tidigare programmerat i C, och kunskaperna kring hårdvaran var ytterst grunda. Emellertid gick det framåt genom att ta en sak i sänder och med handledning. På det hela taget ett lyckat projekt där alla studenter lärt sig mycket.

6. Appendix

6.1 Kopplingsschema



6.2 Knappfunktioner

1	2	3	S1
4	5	6	S2
7	8	HP	S3
Clr	0	Ent	SD

- Nummer 0-8 skriver numret på displayen samt sparar numret i en vektor. Numret i vektorn används för att sätta tillåtna maxdistans från hempunkten.

- Clr tömmer displayen och vektorn ovan.

- Ent registrerar att användaren skrivit färdigt och ändrar maxdistansen katten får förflytta sig från hempunkten.

- HP sätter hempositionen till de koordinater där användaren befinner sig just nu.

- S1 testar att UART-kommunikationen funkar genom att se att SIM808 komponenten kan kommunicera med mikroprocessorn. Returnerar OK om allt fungerar.

- S2 sätter på GPS:en, returnerar ok när den är påslagen.

- S3 testar om GPS:en har en positionsfix. Returnerar “1” om den har en positionsfix.

- SD väljs då användaren vill välja maxdistans från hempunkten. Meddelande visas på skärmen om hur användaren skall gå till väga.

```
/*
 * gps_tracker.c
 *
 * Created: 2019-05-15 20:00:56
 * Author : er4853bu-s
 */

#include <avr/io.h>
#include "uart.h"
#include "display.h"
#include "sim808.h"
#include "keyboard.h"
#include "led.h"
#include <stdio.h>

#define TRUE     1
#define FALSE    0

extern volatile double home_longitude = 0;
extern volatile double home_latitude = 0;

extern volatile double current_longitude;
extern volatile double current_latitude;


// Uart stuff ----------------------------------------------------
extern volatile char rx_data[100];
extern volatile char position_data[100];

// Sim808 stuff ---------------------------------------------------

double maximum_distance = 100;
double distance_from_home;
extern volatile uint8_t which_vector;

// Keyboard stuff
-----------------------------------------------------
volatile uint8_t val;
volatile uint8_t btn_pressed = FALSE;
char a[20] = {-1}; //Array med inputs frÂn anv‰ndaren
int numInputs; //Antalet inputs frÂn anv‰ndaren
int number; //Om man ska cleara raden sÂ ‰r den 0


// Debug variables
-----------------------------------------------------
volatile uint8_t dummy;

extern volatile uint8_t cnt;

void clearEntryArray(){
        for(int i = 0; i<numInputs; i++){
                a[i] = -1;
        }
```

```
        numInputs = 0;
}

void check_on_the_loose(){

                distance_from_home = distance(home_latitude,
home_longitude, current_latitude, current_longitude);

                if(distance_from_home>maximum_distance){
                        led_green_off();
                        led_red_on();
                }else{
                        led_red_off();
                        led_green_on();
                }
}

int main(void)
{

        led_init();
        uart_init();
        disp_init();
        keyboard_init();

        led_green_on();
        _delay_ms(1000);
        led_green_off();

        sei(); // Enable global interrupts.

        sim808_check_all_str();
        disp_cmd(0b00000001);
        disp_printString("PRESS HP TO SET HOME POSITION ");
        sim808_set_home_position();



    while (1) {

                sim808_update_position();

                check_on_the_loose();

                if(btn_pressed == TRUE) {

                        btn_pressed = FALSE;

                        if(number==0){
                                disp_cmd(0b00000001);
                                number++;
                        }

                        switch(val) {
```

```
case BUTTON_1  :

disp_printString("1");
a[numInputs] = 1;
numInputs++;

break;

case BUTTON_2 :

disp_printString("2");
a[numInputs] = 2;
numInputs++;

break;

case BUTTON_3 :

disp_printString("3");
a[numInputs] = 3;
numInputs++;

break;

case BUTTON_4 :

disp_printString("4");
a[numInputs] = 4;
numInputs++;

break;

case BUTTON_5 :

disp_printString("5");
a[numInputs] = 5;
numInputs++;

break;

case BUTTON_6 :

disp_printString("6");
a[numInputs] = 6;
numInputs++;

break;

case BUTTON_7 :

disp_printString("7");
a[numInputs] = 7;
numInputs++;
```

```
                                        break;

                                        case BUTTON_8 :

                                        disp_printString("8");
                                        a[numInputs] = 8;
                                        numInputs++;

                                        break;

                                        case BUTTON_9 :

                                        //M≈STE IMPLEMENTERAS
                                        sim808_set_home_position();
                                        _delay_ms(200);
                                        disp_printString("Home position
has been set");
                                        _delay_ms(2000);
                                        disp_cmd(0b00000001);

                                        break;

                                        case BUTTON_0 :

                                        disp_printString("0");
                                        a[numInputs] = 0;
                                        numInputs++;

                                        break;

                                        case SIM_OK_STR1 :

                                        sim808_check_str1();
                                        _delay_ms(1000);
                                        disp_cmd(0b00000001);

                                        break;

                                        case SIM_OK :

                                        sim808_check_str();
                                        _delay_ms(1000);
                                        disp_cmd(0b00000001);

                                        break;

                                        case SET_DISTANCE :

                                        //Set max distance
                                        disp_cmd(0b00000001);
                                        number = 0;
                                        disp_printString("Type distance(m)
& press enter");
```

```
                                        break;

                                        case POSITION :

                                        //sim808_check_str2();
                                        _delay_ms(1000);
                                        disp_cmd(0b00000001);
                                        sim808_check_str3();

                                        break;

                                        case CLEAR :

                                        disp_cmd(0b00000001);
                                        clearEntryArray();

                                        break;

                                        case ENTER :

                                        //skicka infon till metod
                                        number = 0;
                                        disp_cmd(0b00000001);

                                        int i = 0;
                                        maximum_distance = 0;
                                        for (i = 0; i < (numInputs); i++)
                                        maximum_distance = 10 *
maximum_distance + a[i];

                                        clearEntryArray();

                                        disp_printString("Max distance has
been set");
                                        _delay_ms(1000);
                                        disp_cmd(0b00000001);

                                        break;


                                        default :

                                                dummy++;


                                }

                                val = NO_BTN;

                        }


                }
```

```
}


ISR(USART_RXC_vect) {

        if(which_vector == 0){
                rx_data[cnt] = UDR;
                cnt++;
        }else{
                position_data[cnt] = UDR;
                cnt++;
        }




}

ISR(INT0_vect) {

        val = (PINA & (0b00111100)) >> 2;
        dummy++;
        btn_pressed = TRUE;


}
```

```
/*
 * sim808.c
 *
 * Created: 2019-05-15 17:25:52
 *  Author: er4853bu-s
 */
#include "sim808.h"


char str[] = "AT";
char str1[] = "AT+CGNSPWR=1";//Power on
char str2[] = "AT+CGNSSEQ=\"RMC\"";//Parses the NMEA sentence
related to GPRMC
char str3[] = "AT+CGNSINF";//Location


char gps_str[] = "AT";
char gps_str1[] = "AT+CGPSPWR=1";//Power on
char gps_str2[] = "AT+CGPSRST=0";//Cold restart
char gps_str3[] = "AT+CGPSSTATUS?";//Status fix location?
char gps_str4[] = "AT+CGPSINF=0";//Location
```

```
volatile char rx_data[100];
volatile uint8_t cnt;

volatile char position_data[100];

volatile double home_longitude;
volatile double home_latitude;

volatile double current_longitude;
volatile double current_latitude;

double deg2rad(double);
double rad2deg(double);

char latitude[30];
char longitude[30];

volatile uint8_t which_vector;




//Updatera current position
void sim808_update_position(){
        sim808_reset_position_data();

        which_vector = 1;

        uart_send_string(str3);
        uart_transmit(_CR);
        _delay_ms(100);


        int i = 0;
        int j = 0;


        for(int k = 0; k < 30; k++) {
                latitude[k] = 0;
                longitude[k] = 0;
        }


        while(position_data[i] !=','){
                i++;
        }
        i++;
        while(position_data[i] !=','){
                i++;
        }
        i++;
        while(position_data[i] !=','){
                i++;
```

```
}
j = i;
i++;

while(position_data[i] !=','){
        i++;
}


int l = 0;
for(int k = (j+1); k<i; k++){
        latitude[l] = position_data[k];
        l++;
}

j = i;
i++;

while(position_data[i] !=','){
        i++;
}

l = 0;
for(int k = (j+1); k<i; k++){
        longitude[l] = position_data[k];
        l++;
}

double d = 0;
d = pow(10,1)*(latitude[0]-0x30);
d = d + (latitude[1]-0x30);
d = d + pow(10,-1)*(latitude[3]-0x30);
d = d + pow(10,-2)*(latitude[4]-0x30);
d = d + pow(10,-3)*(latitude[5]-0x30);
d = d + pow(10,-4)*(latitude[6]-0x30);
d = d + pow(10,-5)*(latitude[7]-0x30);
d = d + pow(10,-6)*(latitude[8]-0x30);
d = d + pow(10,-7)*(latitude[9]-0x30);
d = d + pow(10,-8)*(latitude[10]-0x30);

double e = 0;
e = pow(10,1)*(longitude[0]-0x30);
e = e + (longitude[1]-0x30);
e = e + pow(10,-1)*(longitude[3]-0x30);
e = e + pow(10,-2)*(longitude[4]-0x30);
e = e + pow(10,-3)*(longitude[5]-0x30);
e = e + pow(10,-4)*(longitude[6]-0x30);
e = e + pow(10,-5)*(longitude[7]-0x30);
e = e + pow(10,-6)*(longitude[8]-0x30);
e = e + pow(10,-7)*(longitude[9]-0x30);
e = e + pow(10,-8)*(longitude[10]-0x30);


current_latitude = d;
```

```
        current_longitude = e;

        cnt = 0;

        which_vector = 0;

}



double sim808_check_all_str(){

        sim808_check_str(); //AT
        sim808_check_str1(); //Power on
        sim808_check_str2(); //Parses the NMEA sentence related to
GPRMC
        sim808_check_str3(); //Status; location fix available?

        cnt = 0;

        return 0;
}

void sim808_resetArray() {
        for(int i = 0; i < 50; i++) {
                rx_data[i] = 0;
        }
}

void sim808_reset_position_data() {
        for(int i = 0; i < 50; i++) {
                position_data[i] = 0;
        }
}

void sim808_check_str(){
        sim808_resetArray();
        uart_send_string(str);
        uart_transmit(_CR);
        _delay_ms(100);
        sim808_response_str_1_2();
        cnt = 0;
}

void sim808_check_str1(){
        sim808_resetArray();
        uart_send_string(str1);
        uart_transmit(_CR);
        _delay_ms(1000);
        sim808_response_str_1_2();
        cnt = 0;
}
```

```
void sim808_check_str2(){
        sim808_resetArray();
        uart_send_string(str2);
        uart_transmit(_CR);
        _delay_ms(1000);
        sim808_response_str_1_2();
        cnt = 0;
}

void sim808_check_str3(){
        sim808_resetArray();
        uart_send_string(str3);
        uart_transmit(_CR);
        _delay_ms(1000);
        sim808_response_str3();
        cnt = 0;
}

void sim808_check_str4(){
        sim808_resetArray();
        uart_send_string(gps_str4);
        uart_transmit(_CR);
        _delay_ms(1000);
        sim808_update_current_position();
        cnt = 0;
}

void sim808_set_home_position(){
        sim808_update_position();
        _delay_ms(1000);
        home_latitude = current_latitude;
        home_longitude = current_longitude;

}

double sim808_to_double(const char* s, int start, int stop) {
    unsigned long long int m = 1;
    double ret = 0;
    for (int i = stop; i >= start; i--) {
        ret += (s[i] - '0') * m;
        m *= 10;
    }
    return ret;
}

void sim808_update_current_position(){

        sim808_check_str3();

        int i = 0;
        int j = 0;
        char latitude[30];
        char longitude[30];
```

```
for(int k = 0; k < 30; k++) {
        latitude[k] = 0;
        longitude[k] = 0;
}


while(rx_data[i] !=','){
        i++;
}
i++;
while(rx_data[i] !=','){
        i++;
}
i++;
while(rx_data[i] !=','){
        i++;
}
j = i;
i++;

while(rx_data[i] !=','){
        i++;
}


int l = 0;
for(int k = (j+1); k<i; k++){
        latitude[l] = rx_data[k];
        l++;
}

j = i;
i++;

while(rx_data[i] !=','){
        i++;
}

l = 0;
for(int k = (j+1); k<i; k++){
        longitude[l] = rx_data[k];
        l++;
}

double d = 0;
d = pow(10,1)*(latitude[0]-0x30);
d = d + (latitude[1]-0x30);
d = d + pow(10,-1)*(latitude[3]-0x30);
d = d + pow(10,-2)*(latitude[4]-0x30);
d = d + pow(10,-3)*(latitude[5]-0x30);
d = d + pow(10,-4)*(latitude[6]-0x30);
d = d + pow(10,-5)*(latitude[7]-0x30);
d = d + pow(10,-6)*(latitude[8]-0x30);
d = d + pow(10,-7)*(latitude[9]-0x30);
```

```
        d = d + pow(10,-8)*(latitude[10]-0x30);

        double e = 0;
        e = pow(10,1)*(longitude[0]-0x30);
        e = e + (longitude[1]-0x30);
        e = e + pow(10,-1)*(longitude[3]-0x30);
        e = e + pow(10,-2)*(longitude[4]-0x30);
        e = e + pow(10,-3)*(longitude[5]-0x30);
        e = e + pow(10,-4)*(longitude[6]-0x30);
        e = e + pow(10,-5)*(longitude[7]-0x30);
        e = e + pow(10,-6)*(longitude[8]-0x30);
        e = e + pow(10,-7)*(longitude[9]-0x30);
        e = e + pow(10,-8)*(longitude[10]-0x30);


        current_latitude = d;
        current_longitude = e;

}


void sim808_response_str_1_2(){

        int i = 0;
        int j = 0;
        char resp[100];
        for(int k = 0; k < 100; k++) {
                resp[k] = 0;
        }

        while (rx_data[i] != '\0'){
                i++;
        }

        while(rx_data[i] !=_CR){
                i--;
        }
        j = i;

        while(rx_data[i] !=_LF){
                i--;
        }

        int l = 0;
        for(int k = (i+1); k<j; k++){
                resp[l] = rx_data[k];
                l++;
        }

        disp_printString(resp);

}

void sim808_response_str3(){
```

```
        int i = 0;
        while (rx_data[i] != ','){
                i++;
        }
        i++;

        displ_printChar(rx_data[i]);
        _delay_ms(1000);
        disp_cmd(0b00000001);

}

double distance(double lat1, double lon1, double lat2, double lon2)
{
        // The math module contains
        // a function named toRadians
        // which converts from degrees
        // to radians.
        lon1 = deg2rad(lon1);
        lon2 = deg2rad(lon2);
        lat1 = deg2rad(lat1);
        lat2 = deg2rad(lat2);

        // Haversine formula
        double dlon = lon2 - lon1;
        double dlat = lat2 - lat1;
        double a = pow(sin(dlat / 2), 2) +
        cos(lat1) * cos(lat2) *
        pow(sin(dlon / 2),2);

        double c = 2 * asin(sqrt(a));

        // Radius of earth in
        // kilometers. Use 3956
        // for miles
        double r = 6371000;
        double s = (c * r);
        // calculate the result
        return s;
}

double deg2rad(double deg) {
        return (deg * pi / 180);
}

double rad2deg(double rad) {
        return (rad * 180 / pi);
}
```

```
/*
 * display.c
 *
 * Created: 2019-05-15 17:19:06
 *  Author: er4853bu-s
 */


#include "display.h"

void disp_cmd(char c){
        _delay_ms(5);
        PORTB=c;
        PORTD |= (1<<E); // E hˆg
        PORTD &= ~(1<<RS); // RS lÂg
        PORTD &= ~(1<<RW); // R/W lÃg
        PORTD &= ~(1<<E); // E lÂg
        PORTD |= (1<<E); // E hˆg
}

void displ_printChar(char c){
        _delay_ms(2);
        PORTB = c;
        PORTD |= (1<<E); // E hˆg
        PORTD |= (1<<RS); // RS high
        PORTD &= ~(1<<RW); // RW lÃg
        PORTD &= ~(1<<E); // E lÂg
        PORTD |= (1<<E); // E hˆg
}

void disp_init(){
        //Starta displayen och s‰tt markˆren l‰ngst till v‰nster
och clear
        DDRD |= 0b01110000;
        DDRB = 0b11111111;
        disp_cmd(0b00111000);//FUNCTION SET
        disp_cmd(0b00001111);//Display on
        disp_cmd(0b000000010); // return home
        disp_cmd(0b00000001);//Clear display
}

void disp_printString(char list[]){
        int i = 0;
        while(list[i] !='\0'){
                if(i==16){
                        disp_cmd(0xC0);
                }
                displ_printChar(list[i]);
                i++;
        }
}
```

```
/*
 * keyboard.c
 *
 * Created: 2019-05-15 17:41:04
 *  Author: er4853bu-s
 */

#include "keyboard.h"

extern volatile uint8_t val;

void keyboard_init(){
        MCUCR |= (1 << ISC01) | (1 << ISC00);
        GICR  |= (1 << INT0);
}
```

```
/*
 * uart.c
 *
 * Created: 2019-05-15 17:22:36
 *  Author: er4853bu-s
 */

#include "uart.h"

void uart_init() {
        UCSRB |=(1<<RXCIE)|(1<<RXEN)|(1<<TXEN);
        UBRRL  = 51;
}

unsigned char uart_Receive( void ) {
        /* Wait for data to be received */
        while ( !(UCSRA & (1<<RXC)));
        /* Get and return received data from buffer */
        return UDR;
}

void uart_transmit( unsigned char data ) {
        /* Wait for empty transmit buffer */
        while ( !( UCSRA & (1<<UDRE)));
        /* Put data into buffer, sends the data */
        UDR = data;
}

void uart_send_string(char s[])
{
        int i =0;
```

```
        while (s[i] != 0x00)
        {
                uart_transmit(s[i]);
                i++;
        }
}
```

```
/*
 * led.c
 *
 * Created: 2019-05-15 20:36:01
 *  Author: er4853bu-s
 */

#include "led.h"

void led_init() {

        LED_DDR_GREEN    |= (1 << LED_GREEN);
        LED_DDR_RED              |= (1 << LED_RED);

}

void led_green_toggle() {

        LED_PORT_GREEN ^= (1 << LED_GREEN);

}

void led_green_on(){

        LED_PORT_GREEN |= (1 << LED_GREEN);

}

void led_green_off() {

        LED_PORT_GREEN &= ~(1 << LED_GREEN);

}

void led_red_toggle() {

        LED_PORT_RED ^= (1 << LED_RED);

}

void led_red_on(){

        LED_PORT_RED |= (1 << LED_RED);
```

```
}

void led_red_off() {

        LED_PORT_RED &= ~(1 << LED_RED);

}
```

```
/*
 * sim808.h
 *
 * Created: 2019-05-15 17:25:40
 *  Author: er4853bu-s
 */


#ifndef SIM808_H_
#define SIM808_H_

#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include "uart.h"
#include "display.h"
#include <stdbool.h>
#include <math.h>

#endif /* SIM808_H_ */

double sim808_check_all_str();
void sim808_resetArray();
void sim808_check_str();
void sim808_check_str1();
void sim808_check_str2();
void sim808_check_str3();
void sim808_set_home_position();
void sim808_response_str_1_2();
void sim808_response_str3();
double distance(double lat1, double lon1, double lat2, double lon2);
double deg2rad(double deg);
double rad2deg(double rad);
void sim808_update_current_position();
void sim808_update_position();
void sim808_reset_position_data();


#define _CR              0xD
```

```
#define _LF              0xA
#define pi 3.14159265358979323846
```

```
/*
 * display.h
 *
 * Created: 2019-05-15 17:18:51
 *  Author: er4853bu-s
 */


#ifndef DISPLAY_H_
#define DISPLAY_H_

#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>

#define E                6
#define RS               4
#define RW               5

void disp_cmd(char c);
void displ_printChar(char c);
void disp_init();
void disp_printString(char list[]);



#endif /* DISPLAY_H_ */
```

```
/*
 * keyboard.h
 *
 * Created: 2019-05-15 17:41:22
 *  Author: er4853bu-s
 */


#ifndef KEYBOARD_H_
#define KEYBOARD_H_
```

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>

#define BUTTON_1         12
#define BUTTON_2         4
#define BUTTON_3         8
#define BUTTON_4         14
#define BUTTON_5         6
#define BUTTON_6         10
#define BUTTON_7         13
#define BUTTON_8         5
#define BUTTON_9         9
#define BUTTON_0         7
#define SIM_OK_STR1     2
#define SIM_OK                   0
#define SET_DISTANCE     3
#define POSITION         1
#define CLEAR                    15
#define ENTER                    11
#define NO_BTN                   255

void keyboard_init();

#endif /* KEYBOARD_H_ */
```

```
/*
 * uart.h
 *
 * Created: 2019-05-15 17:22:25
 *  Author: er4853bu-s
 */


#ifndef UART_H_
#define UART_H_

#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>

void uart_init();
unsigned char uart_Receive( void );
void uart_transmit( unsigned char data );
void uart_send_string(char s[]);
```

```
#endif /* UART_H_ */
```

```
/*
 * led.h
 *
 * Created: 2019-05-15 20:36:19
 *  Author: er4853bu-s
 */


#ifndef LED_H_
#define LED_H_

#include <avr/io.h>

#define LED_GREEN        (7)
#define LED_RED          (0)

#define LED_PORT_GREEN   PORTD
#define LED_PORT_RED     PORTC

#define LED_DDR_GREEN    DDRD
#define LED_DDR_RED              DDRC

void led_init();
void led_green_toggle();
void led_green_on();
void led_green_off();
void led_red_toggle();
void led_red_on();
void led_red_off();

#endif /* LED_H_ */
```