```
/*
 ============================================================================
 Name        : BicycleShed.c
 Author      : Odroid
 Version     :
 Copyright   : Extra Project Code
 Description : The is the main program for the eBike Charging station. The program handless
 the
               comunication between different parts of the charging station. Namely: weather
               station,
               chargers, web server, converters and solar panels temperature measurement.
               The BicycleShed.c
               contains threads which operate on different parts of the system. It is
               crucial that the threads
               terminated after the functions (whether successful or not) becasue otherwise
               they would try to
               operate 'on the same parts of the computer'. Description of the functions or
               parts of the code
               is given on appropriate places.
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <inttypes.h>
#include <modbus.h>
#include <stdint.h>
#include <curl/curl.h>
#include <fcntl.h>
#include <pthread.h>
#include <time.h>
#include <sys/syscall.h>

#include "readDatabase.h"
#include "Victron.h"
#include "Weather.h"
#include "Display.h"
#include "ServerCom.h"
#include "GPIOs.h"


void *function15s();
void *function10m();
void *functionData();
void *functionVictronData();

int main(void) {

    /*LEDBlink();
    puts("done");*/

    int j=1; // for the infinite loop
    int n;// measuring time in seconds



    pthread_t thread1; // thread1 is the thread which is executed most often
    pthread_t thread2; // thread1 is very similar to thread1 but it also writes into server
    pthread_t thread3; //thread for Solar temperature, and writing the data to the server
    pthread_t thread4; // thread for Victron data, similar as thread3
    int iret1,iret2,iret3,iret4;

    while (j<20){
    clock_t start = clock(),diff;

    /*Creating independent threads--each is executing one function
    (this is a bit tricky, if the data are going to be displayed they have to be read,
    therefore some functions are shared among threads)*/
```

```
    iret1 = pthread_create(&thread1,NULL,function15s,NULL);
    if(iret1)
    {
        fprintf(stderr,"Error-pthread_create() return code: %d \n",iret1);
        exit(EXIT_FAILURE);
    }

    printf("pthread_create() was successful, %d \n",iret1);

    pthread_join(thread1,NULL);

    diff = clock()-start; //timing of the process to deliver data when we want to

    printf("the time taken was %d \n",diff/10000);
    n = 15 -diff/10000;
    sleep(n); /*pause before next iteration - in order not to overwhelm the weather
    station!!!*/
    /*NEW THREAD IS CREATED every x seconds to write weather values to the server */
    if(j==6){
        /*read&write weather data*/
        iret2 = pthread_create(&thread2,NULL,function10m,NULL);
        if(iret2)
                {
                    fprintf(stderr,"Error-pthread_create() return code: %d \n",iret2);
                    exit(EXIT_FAILURE);
                }
        pthread_join(thread2,NULL);

    };
    printf("run %d",j);
    if(j==11){
            /*read&write Solar data*/
            iret3 = pthread_create(&thread3,NULL,functionData,NULL);
            if(iret2)
                    {
                        fprintf(stderr,"Error-pthread_create() return code: %d \n",iret2);
                        exit(EXIT_FAILURE);
                    }

        };

    if(j==15){
            /*read&write Victron data*/
            iret3 = pthread_create(&thread4,NULL,functionVictronData,NULL);
            if(iret2)
                    {
                        fprintf(stderr,"Error-pthread_create() return code: %d \n",iret2);
                        exit(EXIT_FAILURE);
                    }
            pthread_join(thread4,NULL);
            j=1;
        };
    printf("j is %d \n",j);
    j++;
    }


    return EXIT_SUCCESS;
}

/*THREAD FUNCTION TO READ CHARGER STATUS FROM THE SERVER. ALSO DISPLAYS THE CURRENT STATUS
OF THE CHARGERS ON THE LOCAL DISPLAY*/
void *function15s()
{

    htmlDisplay();
    printf("Charger function running. was.");
    pthread_exit(NULL); // threads has to be exited to ensure that thread 1 can continue.
    and also memory wise.
}
/*THREAD FUNCTION TO READ&SEND WEATHER DATA also displays the data on the local display*/
```

```
void *function10m()
{
    WeatherDisplay();
    printf("Weather function running. was.");
    pthread_exit(NULL);
}
/*THREAD FUNCTION TO READ&SEND SOLAR PANEL TEMPERATURE*/
void *functionData(){
    /*READ SOLAR TEMP. STATION DATA*/
    uint16_t *solar;
    solar = (uint16_t*)malloc(6*sizeof(uint16_t));
    solar= readHold(); // reading data from registers
    /*SEND THE SOLAR TEMP. DATA TO SERVER*/
    int jj = writeSolarToServer(solar); //send data to server
    free(solar);
    pthread_exit(NULL);
}

/*THREAD FUNCTION FOR VICTRON DATA---uploads data to all 3 Victron Databases*/

void *functionVictronData()
{
/*READ DATA FROM VICTRON AND SEND THEM TO THE SERVER*/
    uint16_t *victron;
        victron = (uint16_t*)malloc(35*sizeof(uint16_t));
        victron= getData(3,38); //this is repeating down, the data are read in three big
        chunks
        int jj = writeVictron1ToServer(victron); //and send data to server
        free(victron);

        /*repeat for second chunk of data--the registers are not prepared to be read from,
        thus this part is commented out*/
/*      victron = (uint16_t*)malloc(44*sizeof(uint16_t));
        victron= getData(259,301);
        int jj1 = writeVictron2ToServer(victron);
        free(victron);

        /*repeat for third chunk of data*/
        victron = (uint16_t*)malloc(8*sizeof(uint16_t));
        victron= getData(778,786);
        int jj2 = writeVictron3ToServer(victron);
        free(victron);
        /*exit thread --- mutex is the reason*/
        sleep(1);
        puts("Last thread. Hope for zero");
        pthread_exit(NULL);
}
```

```
/*
 * Display.c
 *
 *  Created on: Oct 7, 2015
 *      Author: Odroid
 * Description: This part is a big longer, however very straightforward. The
 *              idea is to print the .html each time with new data. This is not
 *              such problem as on Windows. Since on Linux the files which are
 *              opened can be operated on.
 */

#include <stdio.h>
#include <stdlib.h>

#include "Weather.h"
#include "ServerCom.h"

void htmlDisplay(){
    FILE  * fp;
    /*READ WEATHER STATION DATA*/
    uint16_t *weather = readInput();
    /*READ SERVER TO GET INFORMATION ABOUT CHARGER STATUS*/
    int i;
    int charger[7];
    for (i=0;i<6;i++){
    char *b=NULL;
    b  = getChargerState(i+1);
    charger[i]= atoi(b);}
    /*CREATE HTML TO HAVE IMPORTANT INFO*/
    fp = fopen("file.html","w+");

    //fprintf(fp, "Content-type: text/html\n\n");
    fprintf(fp, "<html><title>Hello</title><body>\n");
    fprintf(fp,"<head>\n");
    fprintf(fp,"<meta http-equiv='refresh' content='300'>\n");
    fprintf(fp,"<link rel='stylesheet' type='text/css' href='mystyle.css'>\n");
    fprintf(fp,"<script src='go-debug.js'></script>\n");
    fprintf(fp,"</head>\n");
    fprintf(fp,"<body>\n");


    fprintf(fp,"<div id='header'>\n");
    fprintf(fp,"<div class='background'>\n");
    fprintf(fp,"<h1>TU Delft Bicycle shed</h1>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"</div>\n");

    fprintf(fp,"<div id='nav'>\n");
    fprintf(fp,"About<br>\n");
    fprintf(fp,"Weather<br>\n");
    fprintf(fp,"Charger<br>\n");
    fprintf(fp,"</div>");

    fprintf(fp,"<div id='section'>\n");
    fprintf(fp,"<h2>About</h2>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"Solar eBike charging station is station which is power neutral. It is able
    to store solar energy and charge up to 4 electric bikes  and a single scooter.\n");
    fprintf(fp,"</p>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"This is very much fun so fun.\n");
    fprintf(fp,"</p>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='myDiagramDiv'>\n");
    fprintf(fp,"<script>\n");
    fprintf(fp,"var $ = go.GraphObject.make;\n");
    fprintf(fp,"var myDiagram = \n");
    fprintf(fp,"$(go.Diagram, 'myDiagramDiv', \n");
    fprintf(fp,"{\n");
    fprintf(fp,"initialContentAlignment: go.Spot.Center,\n");
    fprintf(fp,"'undoManager.isEnabled': true,\n");
```

```
    fprintf(fp,"layout: $(go.TreeLayout, \n");
    fprintf(fp,"{ angle: 90, layerSpacing: 25})\n");
    fprintf(fp,"});\n");
    fprintf(fp,"myDiagram.nodeTemplate =\n");
    fprintf(fp,"$(go.Node, 'Horizontal',\n");
    fprintf(fp,"$(go.Picture,\n");
    fprintf(fp,"{ margin: 10, width: 40, height: 40, background: 'white' },\n");
    fprintf(fp,"new go.Binding('source')),\n");
    fprintf(fp,"$(go.TextBlock,\n");
    fprintf(fp,"'Default Text',  \n");
    fprintf(fp,"{ margin: 12, stroke: 'black', font: 'bold 10px sans-serif' },\n");
    fprintf(fp,"new go.Binding('text', 'name'))\n");
    fprintf(fp,");\n");
    fprintf(fp,"var model = $(go.TreeModel);\n");
    fprintf(fp,"model.nodeDataArray =\n");
    fprintf(fp,"[ { key: '1',       name: 'Solar', source: 'solar.png' },\n");
    fprintf(fp,"{ key: '3', parent: '2', name: 'Battery', source: 'battery.GIF' },\n");
    fprintf(fp,"{ key: '2', parent: '1', name: 'Converter',  source: 'converter.gif' },\n");
    fprintf(fp,"{ key: '3', parent: '2', name: 'Charger',  source: 'Charger.png' }\n");
    fprintf(fp,"];\n");
    fprintf(fp,"myDiagram.model = model;\n");
    fprintf(fp,"</script>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='section2'>\n");
    fprintf(fp,"<h2>Weather</h2>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"Here will be the info from Weather station. Like\n");
    fprintf(fp,"<div id= 'tabloid'> \n");
    fprintf(fp,"<table id='myTable'>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<th>Name</th>\n");
    fprintf(fp,"<th>Value</th>\n");
    fprintf(fp,"<th>Name</th>\n");
    fprintf(fp,"<th>Value</th>\n");
    fprintf(fp,"</tr>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<td>Air Temperature</td>\n");
    fprintf(fp,"<td> %d C</td>\n",*(weather)/10);
    fprintf(fp,"<td>Global Irradiation</td>\n");
    fprintf(fp,"<td>%d </td>\n",*(weather+3)/10);
    fprintf(fp,"</tr>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<td>Relative Humidity</td>\n");
    fprintf(fp,"<td>%d %</td>\n",*(weather+1)/10);
    fprintf(fp,"<td>Air Pressure</td>\n");
    fprintf(fp,"<td>%d kPa</td>\n",*(weather+4)/10);
    fprintf(fp,"</tr>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<td>Wind Speed</td>\n");
    fprintf(fp,"<td>%d m/s</td>\n",*(weather+2)/10);
    fprintf(fp,"<td>Absolute Precipitation</td>\n");
    fprintf(fp,"<td>%d </td>\n",*(weather+5)/100);
    fprintf(fp,"</tr>\n");
    fprintf(fp,"</table>\n");
    fprintf(fp,"</div> \n");
    fprintf(fp,"</p>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='section3'>\n");
    fprintf(fp,"<h2>Charger</h2>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"<div id='myDiagramDiv2'>\n");
    fprintf(fp,"<script>\n");
    fprintf(fp,"var $ = go.GraphObject.make;\n");
    fprintf(fp,"var myDiagram2 =\n");
    fprintf(fp,"$(go.Diagram, 'myDiagramDiv2',\n");
    fprintf(fp,"{\n");
    fprintf(fp,"initialContentAlignment: go.Spot.Center,\n");
    fprintf(fp,"'undoManager.isEnabled': true,\n");
    fprintf(fp,"layout: $(go.TreeLayout,\n");
    fprintf(fp,"{ angle: 90, layerSpacing: 25})\n");
    fprintf(fp,"});\n");
```

```
    fprintf(fp,"myDiagram2.nodeTemplate =\n");
    fprintf(fp,"$(go.Node, 'Horizontal',\n");
    fprintf(fp,"$(go.Picture,\n");
    fprintf(fp,"{ margin: 10, width: 40, height: 40, background: 'white' },\n");
    fprintf(fp,"new go.Binding('source')),\n");
    fprintf(fp,"$(go.TextBlock,\n");
    fprintf(fp,"'Default Text',\n");
    fprintf(fp,"{ margin: 12, stroke: 'black', font: 'bold 10px sans-serif' },\n");
    fprintf(fp,"new go.Binding('text', 'name'))\n");
    fprintf(fp,");\n");
    fprintf(fp,"var model2 = $(go.TreeModel);\n");
    fprintf(fp,"model2.nodeDataArray =\n");
    fprintf(fp,"[ { key: '1',       name: 'Converter', source: 'converter.gif' },\n");
    if((charger[0])==0){
    fprintf(fp,"{ key: '2', parent: '1', name: 'Charger', source: 'Wireless.png' },\n");}else{
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger', source: 'Wirelesson.png' },\n");
    }
    if((charger[1])==0){
    fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png' },\n");}else{
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png' },\n");
    }
    if((charger[2])==0){
    fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png' },\n");}else{
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png' },\n");
    }
    if((charger[3])==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }
        if((charger[4])==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }
    if((charger[5])==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }
    /*if(*(b+6)==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }*/

    fprintf(fp,"];");
    fprintf(fp,"myDiagram2.model = model2;\n");
    fprintf(fp,"</script>\n");
    fprintf(fp,"</div>\n");

    fprintf(fp,"</p>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='footer'>\n");
    fprintf(fp,"Copyright © TU Delft\n");
    fprintf(fp,"</div>\n");

    fprintf(fp,"</body> </html>\n");
    fclose(fp);

    /*FREE ALLOCATED MEMORY*/
    //free(f);
}

/* It may seem strange that we have to function almost identical. However,
    the reason lies in different timing for sending data to the server. To
    ensure that the screen is always updated with relevant information, the
```

```
    thread which also sends data, updates the screen as well.
*/
void WeatherDisplay(){
    FILE  * fp;
    /*READ WEATHER STATION DATA*/
    uint16_t *weather;
    weather = (uint16_t*)malloc(6*sizeof(uint16_t));
    weather= readInput();
    /*SEND THE WEATHER DATA TO SERVER*/
    int jj = writeWeatherToServer(weather);
    /*READ SERVER TO GET INFORMATION ABOUT CHARGER STATUS*/
    int i;
    int charger[7];
    for (i=0;i<6;i++){
    char *b=NULL;
    b  = getChargerState(i+1);
    charger[i]= atoi(b);}
    /*CREATE HTML TO HAVE IMPORTANT INFO*/
    fp = fopen("file.html","w+");

    //fprintf(fp, "Content-type: text/html\n\n");
    fprintf(fp, "<html><title>Hello</title><body>\n");
    fprintf(fp,"<head>\n");
    fprintf(fp,"<meta http-equiv='refresh' content='300'>\n");
    fprintf(fp,"<link rel='stylesheet' type='text/css' href='mystyle.css'>\n");
    fprintf(fp,"<script src='go-debug.js'></script>\n");
    fprintf(fp,"</head>\n");
    fprintf(fp,"<body>\n");


    fprintf(fp,"<div id='header'>\n");
    fprintf(fp,"<div class='background'>\n");
    fprintf(fp,"<h1>TU Delft Bicycle shed</h1>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"</div>\n");

    fprintf(fp,"<div id='nav'>\n");
    fprintf(fp,"About<br>\n");
    fprintf(fp,"Weather<br>\n");
    fprintf(fp,"Charger<br>\n");
    fprintf(fp,"</div>");

    fprintf(fp,"<div id='section'>\n");
    fprintf(fp,"<h2>About</h2>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"Solar eBike charging station is station which is power neutral. It is able
    to store solar energy and charge up to 4 electric bikes  and a single scooter.\n");
    fprintf(fp,"</p>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"This is very much fun so fun.\n");
    fprintf(fp,"</p>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='myDiagramDiv'>\n");
    fprintf(fp,"<script>\n");
    fprintf(fp,"var $ = go.GraphObject.make;\n");
    fprintf(fp,"var myDiagram = \n");
    fprintf(fp,"$(go.Diagram, 'myDiagramDiv', \n");
    fprintf(fp,"{\n");
    fprintf(fp,"initialContentAlignment: go.Spot.Center,\n");
    fprintf(fp,"'undoManager.isEnabled': true,\n");
    fprintf(fp,"layout: $(go.TreeLayout, \n");
    fprintf(fp,"{ angle: 90, layerSpacing: 25})\n");
    fprintf(fp,"});\n");
    fprintf(fp,"myDiagram.nodeTemplate =\n");
    fprintf(fp,"$(go.Node, 'Horizontal',\n");
    fprintf(fp,"$(go.Picture,\n");
    fprintf(fp,"{ margin: 10, width: 40, height: 40, background: 'white' },\n");
    fprintf(fp,"new go.Binding('source')),\n");
    fprintf(fp,"$(go.TextBlock,\n");
    fprintf(fp,"'Default Text',  \n");
    fprintf(fp,"{ margin: 12, stroke: 'black', font: 'bold 10px sans-serif' },\n");
```

```
    fprintf(fp,"new go.Binding('text', 'name'))\n");
    fprintf(fp,");\n");
    fprintf(fp,"var model = $(go.TreeModel);\n");
    fprintf(fp,"model.nodeDataArray =\n");
    fprintf(fp,"[ { key: '1',       name: 'Solar', source: 'solar.png' },\n");
    fprintf(fp,"{ key: '3', parent: '2', name: 'Battery', source: 'battery.GIF' },\n");
    fprintf(fp,"{ key: '2', parent: '1', name: 'Converter',  source: 'converter.gif' },\n");
    fprintf(fp,"{ key: '3', parent: '2', name: 'Charger',  source: 'Charger.png' }\n");
    fprintf(fp,"];\n");
    fprintf(fp,"myDiagram.model = model;\n");
    fprintf(fp,"</script>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='section2'>\n");
    fprintf(fp,"<h2>Weather</h2>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"Here will be the info from Weather station. Like\n");
    fprintf(fp,"<div id= 'tabloid'> \n");
    fprintf(fp,"<table id='myTable'>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<th>Name</th>\n");
    fprintf(fp,"<th>Value</th>\n");
    fprintf(fp,"<th>Name</th>\n");
    fprintf(fp,"<th>Value</th>\n");
    fprintf(fp,"</tr>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<td>Air Temperature</td>\n");
    fprintf(fp,"<td> %d C</td>\n",*(weather)/10);
    fprintf(fp,"<td>Global Irradiation</td>\n");
    fprintf(fp,"<td>%d </td>\n",*(weather+3)/10);
    fprintf(fp,"</tr>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<td>Relative Humidity</td>\n");
    fprintf(fp,"<td>%d %</td>\n",*(weather+1)/10);
    fprintf(fp,"<td>Air Pressure</td>\n");
    fprintf(fp,"<td>%d kPa</td>\n",*(weather+4)/10);
    fprintf(fp,"</tr>\n");
    fprintf(fp,"<tr>\n");
    fprintf(fp,"<td>Wind Speed</td>\n");
    fprintf(fp,"<td>%d m/s</td>\n",*(weather+2)/10);
    fprintf(fp,"<td>Absolute Precipitation</td>\n");
    fprintf(fp,"<td>%d </td>\n",*(weather+5)/100);
    fprintf(fp,"</tr>\n");
    fprintf(fp,"</table>\n");
    fprintf(fp,"</div> \n");
    fprintf(fp,"</p>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='section3'>\n");
    fprintf(fp,"<h2>Charger</h2>\n");
    fprintf(fp,"<p>\n");
    fprintf(fp,"<div id='myDiagramDiv2'>\n");
    fprintf(fp,"<script>\n");
    fprintf(fp,"var $ = go.GraphObject.make;\n");
    fprintf(fp,"var myDiagram2 =\n");
    fprintf(fp,"$(go.Diagram, 'myDiagramDiv2',\n");
    fprintf(fp,"{\n");
    fprintf(fp,"initialContentAlignment: go.Spot.Center,\n");
    fprintf(fp,"'undoManager.isEnabled': true,\n");
    fprintf(fp,"layout: $(go.TreeLayout,\n");
    fprintf(fp,"{ angle: 90, layerSpacing: 25})\n");
    fprintf(fp,"});\n");
    fprintf(fp,"myDiagram2.nodeTemplate =\n");
    fprintf(fp,"$(go.Node, 'Horizontal',\n");
    fprintf(fp,"$(go.Picture,\n");
    fprintf(fp,"{ margin: 10, width: 40, height: 40, background: 'white' },\n");
    fprintf(fp,"new go.Binding('source')),\n");
    fprintf(fp,"$(go.TextBlock,\n");
    fprintf(fp,"'Default Text',\n");
    fprintf(fp,"{ margin: 12, stroke: 'black', font: 'bold 10px sans-serif' },\n");
    fprintf(fp,"new go.Binding('text', 'name'))\n");
    fprintf(fp,");\n");
    fprintf(fp,"var model2 = $(go.TreeModel);\n");
```

```
    fprintf(fp,"model2.nodeDataArray =\n");
    fprintf(fp,"[ { key: '1',       name: 'Converter', source: 'converter.gif' },\n");
    if((charger[0])==0){
    fprintf(fp,"{ key: '2', parent: '1', name: 'Charger', source: 'Wireless.png' },\n");}else{
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger', source: 'Wirelesson.png' },\n");
    }
    if((charger[1])==0){
    fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png' },\n");}else{
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png' },\n");
    }
    if((charger[2])==0){
    fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png' },\n");}else{
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png' },\n");
    }
    if((charger[3])==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }
        if((charger[4])==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }
    if((charger[5])==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }
    /*if(*(b+6)==0){
        fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Charger.png'
        },\n");}else{
            fprintf(fp,"{ key: '2', parent: '1', name: 'Charger',  source: 'Chargeron.png'
            },\n");
        }*/

    fprintf(fp,"];");
    fprintf(fp,"myDiagram2.model = model2;\n");
    fprintf(fp,"</script>\n");
    fprintf(fp,"</div>\n");

    fprintf(fp,"</p>\n");
    fprintf(fp,"</div>\n");
    fprintf(fp,"<div id='footer'>\n");
    fprintf(fp,"Copyright © TU Delft\n");
    fprintf(fp,"</div>\n");

    fprintf(fp,"</body> </html>\n");
    fclose(fp);

    /*FREE ALLOCATED MEMORY*/
    free(weather);
}
```

```
/*
 * readDatabase.c
 *
 *  Created on: Oct 1, 2015
 *      Author: Odroid
 * Description: This is file contains function to operate on the database of Odroid.
 *              However, these functions are not currently used. They are left as a
 *              reference in case in the future database on the Odroid will be useful.
 */



#include <stdio.h>
#include <stdlib.h>
#include </usr/include/mysql/mysql.h>
#include <stdint.h>

void readData(){
    /*Declaration of variables, be so kind do not EDIT this part*/
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    int num_fields;
    int i;
    char *server = "localhost";
    char *user = "root";
    char *password = "odroid";
    char *database = "odroid";
    conn = mysql_init(NULL);
    /*end of super important stuff*/
    /*CONNECTING*/
    if ( !mysql_real_connect(conn, server, user, password, database, 0, NULL,0)){
        printf(stderr,"%s\n",mysql_error(conn));
        exit(1);
    }
    /*query mysql*/
    if (mysql_query(conn,"SHOW tables")){
        printf(stderr,"%s\n",mysql_error(conn));
        exit(1);
    }
    /*initiate res & show info*/
    res = mysql_use_result(conn);
    printf("MYSQL database victron says: \n");
    while ((row = mysql_fetch_row(res)) != NULL){
        printf("%s \n",row[0]);
    }
    /*show contents of the database*/
    if (mysql_query(conn,"SELECT*FROM VictronTable")){
        printf(stderr,"%s\n",mysql_error(conn));
        exit(1);
    }
    res = mysql_store_result(conn);
    num_fields = mysql_num_fields(res);
    while ((row = mysql_fetch_row(res)) != NULL){
        for (i=0;i<num_fields;i++){
            printf("%s \n",row[i] ? row[i]:"NULL");
        }
    }
    /*mischief managed --> CLOSE CONNECTION*/
//  mysql_free_result(free);
    mysql_close(conn);

    puts("lol");
}
/*!!!!!!!!!!!!!!!!!NEW FUNCTION FOR WRITING!!!!!!!!!!!!!!!!!!!*/
void writeData(int START, int END, uint16_t *tab_register){
    /*Declaration of variables, be so kind do not EDIT this part*/
        MYSQL *conn;
        MYSQL_RES *res;
        MYSQL_ROW row;
        int num_fields;
```

```
        int i;
        char *server = "localhost";
        char *user = "root";
        char *password = "odroid";
        char *database = "odroid";
        conn = mysql_init(NULL);
        /*end of super important stuff*/

        /*CONNECTING*/

        if ( !mysql_real_connect(conn, server, user, password, database, 0, NULL,0)){
            printf(stderr,"%s\n",mysql_error(conn));
            exit(1);
        }
        /*query mysql*/
        if (mysql_query(conn,"SHOW tables")){
            printf(stderr,"%s\n",mysql_error(conn));
            exit(1);
        }
        /*initiate res & show info*/
        res = mysql_use_result(conn);
        printf("MYSQL database odroid says: \n");
        while ((row = mysql_fetch_row(res)) != NULL){
            printf("%s \n",row[0]);
        }
        /*write to the DATABASE*/
        char buf[50];
        char buf2[50];

        for(i=0;i<=END-START;i++){

            sprintf(buf,"UPDATE victron SET VALUE= %d", tab_register[i]);
            sprintf(buf2," WHERE Address= %d", START+i);
            strcat(buf,buf2);
            char *m;
            m = strchr(buf,NULL);
            int e = (int)(m-buf);
            char buf3[e];
            for(i=0;i<=e;i++){
                buf3[i]=buf[i];
            }

            printf(" \n %s \n",buf3);

            if (mysql_query(conn,buf3)){
            printf(stderr,"%s\n",mysql_error(conn));
            exit(1);
            }
        }

    /*  res = mysql_store_result(conn);
        num_fields = mysql_num_fields(res);
        while ((row = mysql_fetch_row(res)) != NULL){
            for (i=0;i<num_fields;i++){
                printf("%s \n",row[i] ? row[i]:"NULL");
            }
        }*/
        /*mischief managed --> CLOSE CONNECTION*/
        /*mysql_free_result(free);*/
        mysql_close(conn);

        puts("chaha");
}
/*!!!!!!!!!!!!!!!!!NEW FUNCTION FOR WRITING WEATHER DATA!!!!!!!!!!!!!!!!!!!*/
void writeWeatherData(uint16_t *tab_register){
    /*Declaration of variables, be so kind do not EDIT this part*/
        MYSQL *conn;
        MYSQL_RES *res;
        MYSQL_ROW row;
        int num_fields;
        int i;
```

```
        int START = 1;
        int END = 5;
        char *server = "localhost";
        char *user = "root";
        char *password = "odroid";
        char *database = "odroid";
        conn = mysql_init(NULL);
        /*end of super important stuff*/

        /*CONNECTING*/

        if ( !mysql_real_connect(conn, server, user, password, database, 0, NULL,0)){
            printf(stderr,"%s\n",mysql_error(conn));
            exit(1);
        }
        /*query mysql*/
        if (mysql_query(conn,"SHOW tables")){
            printf(stderr,"%s\n",mysql_error(conn));
            exit(1);
        }
        /*initiate res & show info*/
        res = mysql_use_result(conn);
        printf("MYSQL database odroid says: \n");
        while ((row = mysql_fetch_row(res)) != NULL){
            printf("%s \n",row[0]);
        }
        /*write to the DATABASE*/
        char buf[50];
        char buf2[50];

        for(i=0;i<=4;i++){

            sprintf(buf,"UPDATE weather SET value= %d", tab_register[i]);
            sprintf(buf2," WHERE address= %d", 1+i);
            strcat(buf,buf2);
            char *m;
            m = strchr(buf,NULL);
            int e = (int)(m-buf);
            char buf3[e];
            for(i=0;i<=e;i++){
                buf3[i]=buf[i];
            }

            printf(" \n %s \n",buf3);

            if (mysql_query(conn,buf3)){
            printf(stderr,"%s\n",mysql_error(conn));
            exit(1);
            }
        }

        /*mischief managed --> CLOSE CONNECTION*/
        /*mysql_free_result(free);*/
        mysql_close(conn);

        puts("chaha");
}

int* readCharger(int *f){
/*Declaration of variables, be so kind do not EDIT this part*/
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;
    int num_fields;
    int i;
    static int ress[7];
    char *server = "localhost";
    char *user = "root";
    char *password = "odroid";
    char *database = "odroid";
    conn = mysql_init(NULL);
```

```
    /*end of super important stuff*/
    /*CONNECTING*/
    if ( !mysql_real_connect(conn, server, user, password, database, 0, NULL,0)){
        printf(stderr,"%s\n",mysql_error(conn));
        exit(1);
    }
    /*query mysql*/
    if (mysql_query(conn,"SHOW tables")){
        printf(stderr,"%s\n",mysql_error(conn));
        exit(1);
    }
    /*initiate res & show info*/
    res = mysql_use_result(conn);
    printf("MYSQL database odroid says: \n");
    while ((row = mysql_fetch_row(res)) != NULL){
        printf("%s \n",row[0]);
    }
    /*show contents of the database*/
    if (mysql_query(conn,"SELECT*FROM charger")){
        printf(stderr,"%s\n",mysql_error(conn));
        exit(1);
    }
    res = mysql_store_result(conn);
    num_fields = mysql_num_fields(res);
    int a =0;
    while ((row = mysql_fetch_row(res)) != NULL){
        for (i=0;i<1;i++){//num_fields
            printf("%s \n",row[i] ? row[i]:"NULL");
            char look[5];
            int l=sprintf(look,"%s \n",row[i] ? row[i]:"NULL");
            puts(look);
            char *m=strchr(look,'1');
        if(m!=NULL){
            ress[a]=1;
        }else ress[a]=0;
        }
    a=a+1;
    }
    /*int ress2[7];
    int a=0;
    for (i=0;i<50;i++){
        if (ress[i]==0){
            ress2[a]=0;
            a=a+1;
        }else if (ress[i]==1){
            ress2[a]=1;
            a=a+1;
        }
    }*/

    /*mischief managed --> CLOSE CONNECTION*/
//  mysql_free_result(free);

    mysql_close(conn);

    puts("lol");

    f=&ress;

    return f;
}
```

```
/*
 * ServerCom.c
 *
 *  Created on: Oct 13, 2015
 *      Author: Odroid
 *      Description: Routines to communicate with the Server.
 *                   All communication with the server is performed via
 *                   http links. This is maybe not the best way. However, it is quite
 *                   secure and also easy to implement. The amount of data is not so
 *                   large to use FTP or other means of communication.
 */


#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <inttypes.h>
#include </usr/local/include/modbus/modbus.h>
#include <stdint.h>
#include <curl/curl.h>
#include <sys/syscall.h>
#include <fcntl.h>
/*--------------------TEST-is the internet connection O.K. ? ----------------------------*/
int writeToServer(){
    CURL *curl  = curl_easy_init();
    if (curl){
        CURLcode res;

        curl_easy_setopt(curl,CURLOPT_URL,"http://solarpoweredbikes.tudelft.nl/test/php/testing.php?a=3&b=4");
        res     = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
}
    return 1;
}
/*-----------------WEATHER DATA - sends data from Weather station-------------------------*/
int writeWeatherToServer(uint16_t *weather){
    CURL *curl  = curl_easy_init();
    if (curl){
        CURLcode res;
        char link[100];
        int vys =
        sprintf(link,"http://solarpoweredbikes.tudelft.nl/test/php/Pavel.php?w1=%d&w2=%d&w3=%d&w4=%d&w5=%d&w6=%d",*(weather)/10,*(weather+1)/10,*(weather+2)/10,*(weather+3)/10,*(weather+4)/10,*(weather+5)/10);///10 is the scalefactor.
        puts(link);
        curl_easy_setopt(curl,CURLOPT_URL,link);
        res     = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
}
    return 1;
}
/*----------------------SOLAR TEMPERATURE - sends data from Solar panels temperature measurement------------------*/
int writeSolarToServer(uint16_t *solar){
    CURL *curl  = curl_easy_init();
    if (curl){
        CURLcode res;
        char link[300];
        int vys =
        sprintf(link,"http://solarpoweredbikes.tudelft.nl/test/php/SolarTemperature.php?w1=%d&w2=%d&w3=%d&w4=%d&w5=%d&w6=%d",*(solar),*(solar+1),*(solar+2),*(solar+3),*(solar+4),*(solar+5));///10 is maybe the scalefactor,who knows.
        puts(link);
        curl_easy_setopt(curl,CURLOPT_URL,link);
        res     = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
}
    return 1;
```

```
}
/*----------------------SOLAR TEMPERATURE REGISTERS -- for different type of registers.
currently not used, left as a reference---------*/
/*int writeVictron1ToServer(uint16_t *solar){
    CURL *curl  = curl_easy_init();
    if (curl){
        CURLcode res;
        char link[100];
        int vys =
        sprintf(link,"http://solarpoweredbikes.tudelft.nl/test/php/SolarTemperature.php?w1=%d&
        w2=%d&w3=%d&w4=%d&w5=%d&w6=%d",*(solar),*(solar+1),*(solar+2),*(solar+3),*(solar+4),*(
        solar+5));///10 is maybe the scalefactor,who knows.
        puts(link);
        curl_easy_setopt(curl,CURLOPT_URL,link);
        res     = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
}
    return 1;
}*/
/*----------------------CHARGER STATES  - reads the values of the chargers from the
server-------------------*/
function_pt(void *ptr, size_t size,size_t nmemb,void *ress){
    char **response_ptr = (char**)ress;
    /*the response should be a string*/
    *response_ptr = strndup(ptr,(size_t)(size *nmemb));
    //return nmemb;

}
int getChargerState(int n){
    CURL *curl  = curl_easy_init();
    //char a;
    char look[100];
    char *ress = NULL;


    if (curl){
            CURLcode res;
            int vys =
            sprintf(look,"http://solarpoweredbikes.tudelft.nl/test/php/PavelCharger.php?n=%d",
            n);
            curl_easy_setopt(curl,CURLOPT_URL,look);
            curl_easy_setopt(curl,CURLOPT_HTTPGET,1);
            curl_easy_setopt(curl,CURLOPT_FOLLOWLOCATION,1);
            curl_easy_setopt(curl,CURLOPT_WRITEFUNCTION,function_pt);
            curl_easy_setopt(curl,CURLOPT_WRITEDATA,&ress);
            res = curl_easy_perform(curl);
            curl_easy_cleanup(curl);
    }

    return ress;
}
/*
 * The number of registers in Victron system is simply not read as one chunk of data.
 * The reason is that the register values are 1-39 and 239-301 and 771-786. These cannot
 * be read on one time. In order not to cause stack smashing the reading&sending was divided
 * into three parts.
 */
/*----------------------VICTRON DATA chunk no.1----------------------*/
int writeVictron1ToServer(uint16_t *victron){
    CURL *curl  = curl_easy_init();
    if (curl){
        CURLcode res;
        char link[400];
        int vys = sprintf(link,"http://solarpoweredbikes.tudelft.nl/test/php/Victron1.php"
                "?w1=%d&w2=%d&w3=%d&w4=%d&w5=%d&w6=%d&w7=%d&w8=%d&w9=%d&w10=%d&w11=%d"
                "&w12=%d&w13=%d&w14=%d&w15=%d&w16=%d&w17=%d&w18=%d&w19=%d&w20=%d&w21=%d"
                "&w22=%d&w23=%d&w24=%d&w25=%d&w26=%d&w27=%d&w28=%d&w29=%d&w30=%d&w31=%d"

                "&w32=%d&w33=%d&w34=%d&w35=%d",*(victron),*(victron+1),*(victron+2),*(victron+
                3)
                ,*(victron+4),*(victron+5),*(victron+6),*(victron+7),*(victron+8),*(victron+9)
```

```
                ,*(victron+10),*(victron+11),*(victron+12),*(victron+13),*(victron+14),*(victr
                on+15)

                ,*(victron+16),*(victron+17),*(victron+18),*(victron+19),*(victron+20),*(victr
                on+21)

                ,*(victron+22),*(victron+23),*(victron+24),*(victron+25),*(victron+26),*(victr
                on+27)

                ,*(victron+28),*(victron+29),*(victron+30),*(victron+31),*(victron+32),*(victr
                on+33)
                ,*(victron+34));
        curl_easy_setopt(curl,CURLOPT_URL,link);
        res     = curl_easy_perform(curl);
        curl_easy_cleanup(curl);

}
    return 1;
}
/*----------------------VICTRON DATA chunk no.2----------------------*/
int writeVictron2ToServer(uint16_t *victron){
    CURL *curl  = curl_easy_init();
    if (curl){
        CURLcode res;
        char link[500];
        int vys = sprintf(link,"http://solarpoweredbikes.tudelft.nl/test/php/Victron2.php"
                "?w1=%d&w2=%d&w3=%d&w4=%d&w5=%d&w6=%d&w7=%d&w8=%d&w9=%d&w10=%d&w11=%d"
                "&w12=%d&w13=%d&w14=%d&w15=%d&w16=%d&w17=%d&w18=%d&w19=%d&w20=%d&w21=%d"
                "&w22=%d&w23=%d&w24=%d&w25=%d&w26=%d&w27=%d&w28=%d&w29=%d&w30=%d&w31=%d"

                "&w32=%d&w33=%d&w34=%d&w35=%d&w36=%d&w37=%d&w38=%d&w39=%d&w40=%d&w41=%d&w42=%d
                "
                "&w43=%d",*(victron),*(victron+1),*(victron+2),*(victron+3)
                ,*(victron+4),*(victron+5),*(victron+6),*(victron+7),*(victron+8),*(victron+9)

                ,*(victron+10),*(victron+11),*(victron+12),*(victron+13),*(victron+14),*(victr
                on+15)

                ,*(victron+16),*(victron+17),*(victron+18),*(victron+19),*(victron+20),*(victr
                on+21)

                ,*(victron+22),*(victron+23),*(victron+24),*(victron+25),*(victron+26),*(victr
                on+27)

                ,*(victron+28),*(victron+29),*(victron+30),*(victron+31),*(victron+32),*(victr
                on+33)

                ,*(victron+34),*(victron+35),*(victron+36),*(victron+37),*(victron+38),*(victr
                on+39),*(victron+40)
                ,*(victron+41),*(victron+42));
        curl_easy_setopt(curl,CURLOPT_URL,link);
        res     = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
}
    return 1;
}
/*----------------------VICTRON DATA chunk no.3----------------------*/
int writeVictron3ToServer(uint16_t *victron){
    CURL *curl  = curl_easy_init();
    if (curl){
        CURLcode res;
        char link[400];
        int vys = sprintf(link,"http://solarpoweredbikes.tudelft.nl/test/php/Victron3.php"

                "?w1=%d&w2=%d&w3=%d&w4=%d&w5=%d&w6=%d&w7=%d",*(victron),*(victron+1),*(victron
                +2),*(victron+3)
                ,*(victron+4),*(victron+5),*(victron+6),*(victron+7));
        curl_easy_setopt(curl,CURLOPT_URL,link);
        res     = curl_easy_perform(curl);
        curl_easy_cleanup(curl);
```

```
}
    return 1;
}
```

```c
/*
 * Victron.c
 *
 *  Created on: Oct 1, 2015
 *      Author: Odroid
 *      Description : This part contains functions to read data from the Victron system. The
 contex is Modbus/TCP
 */


#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <inttypes.h>
#include </usr/local/include/modbus/modbus.h>
#include <stdint.h>

uint16_t* getData(int start, int end){
    /*DEFINE VARIABLES*/
    #define LOOP             1
    #define SERVER_ID        0 /*used to be 17*/
    #define ADDRESS_START    start
    #define ADDRESS_END      end
    // START THE MODBUS CONTEX
    modbus_t *ctx;
    /*Memory management variables*/

    uint16_t *tab_rp_registers;
    /*Variables for register manipulation*/
    int nb;
    int rc;
    int addr;
    int slave;
    int i;
    int loop = ADDRESS_END-ADDRESS_START;

    /*TCP Connection*/
    ctx = modbus_new_tcp("192.168.1.3",502);
    modbus_set_debug(ctx,TRUE);
    slave = modbus_set_slave(ctx,0);

    if (modbus_connect(ctx) == -1){
        fprintf(stderr,"Connection Failed %s \n",modbus_strerror(errno));
        modbus_free(ctx);
        exit(1);
    }

    /*After the we have connection allocate some memory*/
    nb = loop;


    tab_rp_registers = (uint16_t *) malloc(nb * sizeof(uint16_t));
    memset(tab_rp_registers, 0, nb * sizeof(uint16_t));



    addr=ADDRESS_START;


    /*Now try to read something*/
    rc = modbus_read_registers(ctx,addr,loop,tab_rp_registers);
    if (rc != loop){
        printf("ERROR : did not read properly") ;
        printf("Messed up on address : %d\n",addr);
    //  goto exception;
    /*
     * The exception is still left there in case in future problems with this part occure.
     */
    }
```

```
    else {
        for (i=0; i<loop; i++) {
            printf("%" PRIu16 "\n",tab_rp_registers[i]);}
    }
    /*exception: // ERROR TIMEOUT IN CONNECTION
        //sleep(5);
        for (i=0;i<43;i++){
            a[i]=1+30*i;
        }
        tab_rp_registers = &a;
        puts('I have been in exception');
    //getchar();
    /*free what we can free*/
    // free(tab_rq_registers); deal with this shit
    // free(tab_rw_rq_registers);

    /*always close the connection*/
    modbus_close(ctx);
    modbus_free(ctx);

return tab_rp_registers;
}
```

```c
/*
 * Weather.c
 *
 *  Created on: Oct 6, 2015
 *      Author: odroid
 * Description: The routines to get the data from the weather
 *              station and the Adam 4015-T. These are merged in one file
 *              since both use the same RTU-TCP contex.
 */



#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <inttypes.h>
#include </usr/local/include/modbus/modbus.h>
#include <stdint.h>

#include "Weather.h"

uint16_t* readHold(){
#define SERVER_ID 1 //normaly in header file. However, there are two devices, thus it is here.

const uint16_t UT_REGISTERS_ADDRESS = 0x0001;
/* Raise a manual exception when this address is used for the first byte */
const uint16_t UT_REGISTERS_NB = 5;
const uint16_t UT_REGISTERS_TAB[] = { 0x0001, 0x0002, 0x0003, 0x0004, 0x0005 };
/* If the following value is used, a bad response is sent.
   It's better to test with a lower value than
   UT_REGISTERS_NB_POINTS to try to raise a segfault. */

const uint16_t UT_INPUT_REGISTERS_NB = 0x6;
    // ------------DECLARATIONs--------------------//
    uint8_t *tab_rp_bits;
    uint16_t *tab_rp_registers;
    modbus_t *ctx;
    int i;
    int nb_points;
    int rc;


    /*BE SURE TO ASSIGN PROPER NUMBER OF USB !!!! THIS IS VERY CRUCIAL !! ALSO BEWARE OF
    CHANGING NUMBERS OF SERVERS*/
    ctx = modbus_new_rtu("/dev/ttyUSB1", 9600,'N',8,1);
    if (ctx == NULL){
        printf(stderr,"Unable to create libmodbus contex\n");
        //exit 1;
    }
    int serial = modbus_rtu_set_serial_mode(ctx,0);

    printf("Serial mode is %d \n",serial);

    modbus_set_debug(ctx,TRUE);
    modbus_set_error_recovery(ctx,MODBUS_ERROR_RECOVERY_LINK |
    MODBUS_ERROR_RECOVERY_PROTOCOL);

    modbus_set_slave(ctx, SERVER_ID);
    serial = modbus_rtu_get_serial_mode(ctx);

    printf("Serial mode is %d \n",serial);

    if (modbus_connect(ctx) == -1){
        fprintf(stderr,"Connection failed: %s\n",modbus_strerror(errno));
        modbus_free(ctx);
        //exit 1;
    }
```

```
    //Allocate and initialize memory to store registers
    nb_points = (UT_REGISTERS_NB > UT_INPUT_REGISTERS_NB) ? UT_REGISTERS_NB :
    UT_INPUT_REGISTERS_NB;
    tab_rp_registers = (uint16_t *) malloc(nb_points * sizeof(uint16_t));
    memset(tab_rp_registers, 0, nb_points * sizeof(uint16_t));


    printf("UNIT TESTING \n"); // just to know we are running, these operation might be a
    bit slower
    printf("TEST READ \n");

    rc = modbus_read_registers(ctx,UT_REGISTERS_ADDRESS, UT_REGISTERS_NB, tab_rp_registers);
    printf("modbus read registers: ");
    if(rc!=UT_REGISTERS_NB){
        printf("FAILED (nb points %d) \n",rc);
        modbus_free(ctx);
        goto close;
    }
    else{
        for (i=0; i< UT_REGISTERS_NB; i++){
    printf("OK, value %d \n",tab_rp_registers[i]);
        }
        modbus_free(ctx);
        return tab_rp_registers;
    }
    modbus_free(ctx);
    modbus_close(ctx);
    return tab_rp_registers;

    close:
    return tab_rp_registers;
    /*Free memory*/
    free(tab_rp_bits);
    free(tab_rp_registers);
    /*Close connection*/
    modbus_close(ctx);
    modbus_free(ctx);

}

/*
 * This function is for the Solar Temperature measurement Adam 4015-T utility. Please be
 * aware of the number of USB port and the server_id. This in case of unattention may cause
 * waste of time.
 */
uint16_t* readInput(){

    #define SERVER_ID 2


/* If the following value is used, a bad response is sent.
   It's better to test with a lower value than
   UT_REGISTERS_NB_POINTS to try to raise a segfault. */
    const uint16_t UT_REGISTERS_NB = 6;
    const uint16_t UT_INPUT_REGISTERS_ADDRESS = 0x0022;
    const uint16_t UT_INPUT_REGISTERS_NB = 0x6;
    const uint16_t UT_INPUT_REGISTERS_TAB[] = { 0x0022, 0x000D, 0x002D, 0x001D, 0x0011,
    0x0030 };

    const float UT_REAL = 916.540649;
    const uint32_t UT_IREAL = 0x4465229a;

    // ------------DECLARATIONs--------------------//
    static uint16_t *tab_rp_registers;
    modbus_t *ctx;
    static int a[6];
    int i;
    int nb_points;
    int rc;
```

```
    ctx = modbus_new_rtu("/dev/ttyUSB0", 9600,'N',8,1);
    if (ctx == NULL){
        printf(stderr,"Unable to create libmodbus contex\n");
        return 0;
    }
    int serial = modbus_rtu_set_serial_mode(ctx,0);

    printf("Serial mode is %d \n",serial);

    modbus_set_debug(ctx,TRUE);
    modbus_set_error_recovery(ctx,MODBUS_ERROR_RECOVERY_LINK |
    MODBUS_ERROR_RECOVERY_PROTOCOL);

    modbus_set_slave(ctx, SERVER_ID);
    serial = modbus_rtu_get_serial_mode(ctx);

    printf("Serial mode is %d \n",serial);

    if (modbus_connect(ctx) == -1){
        fprintf(stderr,"Connection failed: %s\n",modbus_strerror(errno));
        modbus_free(ctx);
        return 0;
    }



    //Allocate and initialize memory to store registers
    nb_points = (UT_REGISTERS_NB > UT_INPUT_REGISTERS_NB) ? UT_REGISTERS_NB :
    UT_INPUT_REGISTERS_NB;
    tab_rp_registers = (uint16_t *) malloc(nb_points * sizeof(uint16_t));
    memset(tab_rp_registers, 0, nb_points * sizeof(uint16_t));

    printf("UNIT TESTING \n");
    printf("TEST READ \n");

    rc = modbus_read_input_registers(ctx,UT_INPUT_REGISTERS_ADDRESS, UT_INPUT_REGISTERS_NB,
    tab_rp_registers);
    printf("modbus read input registers: ");

    if(rc!=UT_INPUT_REGISTERS_NB){
        printf("FAILED (nb points %d) \n",rc);
        free(ctx);
        goto exception;
        //return 0;
    }
    else{
        for (i=0; i< UT_REGISTERS_NB; i++){
    printf("OK, value %d \n",tab_rp_registers[i]);
        }
        free(ctx);
        return tab_rp_registers;
    }
    /*
     * The exception are handled as goto's. It is not ideal solution. However,
     * I didn't see any better. And this type of exception should not cause
     * memory problems.
     */
    exception: // ERROR TIMEOUT IN CONNECTION - here the exception is really needed
    //sleep(5);
    for (i=0;i<6;i++){
        a[i]=1+30*i;
    }
    tab_rp_registers = &a;
    //free(ctx);
    return tab_rp_registers;


}
```

```c
/*
 * GPIOs.c
 *
 *  Created on: Oct 22, 2015
 *      Author: odroid
 * Description: This is just to show how to operate on the GPIOs.
 */


#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <string.h>
#include <time.h>

#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <wiringSerial.h>
#include <lcd.h>


#define DATA_UPDATE_PERIOD  100 // 100ms

#define PORT_ADC1   0   // ADC.AIN0

static int adcValue = 0;

static int ledPos = 0;

const int ledPorts[] = {
    24, // GPIOX.BIT0(#97)
    23, // GPIOX.BIT11(#108)
    22, // GPIOX.BIT3(#100)
    21, // GPIOX.BIT4(#101)
    14, // GPIOX.BIT8(#105)
    13, // GPIOX.BIT9(#106)
    12, // GPIOX.BIT10(#107)
    3,  // GPIOX.BIT18(#115)
    2,  // GPIOX.BIT19(#116)
    0,  // GPIOY.BIT8(#88)
    7,  // GPIOY.BIT3(#83)

    1,  // GPIOY.BIT7(#87)
    4,  // GPIOX.BIT7(#104)
    5,  // GPIOX.BIT5(#102)
    6,  // GPIOX.BIT6(#103)
    10, // GPIOX.BIT20(#117)
    26, // GPIOX.BIT2(#99)
    11, // GPIOX.BIT21(#118)
    27, // GPIOX.BIT1(#98)
};

#define MAX_LED_CNT sizeof(ledPorts) / sizeof(ledPorts[0])
 /*System initialize*/
int system_init(void)
{
    int i;
    i=0;
    // GPIO Init(LED Port ALL Output)
    for(i = 0; i < MAX_LED_CNT; i++)    pinMode (ledPorts[i], OUTPUT);

    return  0;
 }

/*Read ADC value*/
void boardDataUpdate(void)
{
    int i;
```

```
    // adc value read
    if((adcValue = analogRead (PORT_ADC1)))    {
        ledPos = (adcValue * MAX_LED_CNT * 1000) / 1024;
        ledPos = (MAX_LED_CNT - (ledPos / 1000));
    }
    else
        ledPos = 0;

    //  LED Control
   for(i = 0; i < MAX_LED_CNT; i++)    digitalWrite (ledPorts[i], 0); // LED All Clear
    for(i = 0; i < ledPos;      i++)    digitalWrite (ledPorts[i], 1); // LED On
}

void LEDBlink()
{
    static int timer = 0;
    int i;
    int j;

    wiringPiSetup();
    if (system_init()<0)
    {
        printf(stderr,"Sth. went wrong. \n");
    }
    j=1;
    for (i=0;i<15;i++)
    {
         digitalWrite(1,LOW); //turn off
        sleep(1);
         digitalWrite(1,HIGH);//turn on
        sleep(1);
    }
}
```