```
Programmet anvÃ¤nder sig av biblikteken frÃ¥n uppgift 6 och dessa antas finnas i
 /usr/local/lib
Den Ã¤r snarlik den textbaserade electrotest frÃ¥n uppgift 6 men denna har ett g
rafiskt grÃ¤nssnitt som anvÃ¤nder
GTK+ 2.0 (rekommendation frÃ¥n kursboken). Kompileras med make all som skapar de
n kÃ¶rbara filen electrotestgtk.
```

```
#Makefile for electrotestgtk
all:      electrotestgtk

electrotestgtk:     electrogui.o coupling_box.o calculate_box.o voltage_box.o
        gcc electrogui.o calculate_box.o voltage_box.o coupling_box.o -o electro
testgtk -L/usr/local/lib -lresistance -lpower -lcomponent -Wl,-R/usr/local/lib -
lm `pkg-config --cflags --libs gtk+-2.0`

electrogui.o:       electrogui.c electrogui.h
        gcc -c electrogui.c -o electrogui.o `pkg-config --cflags --libs gtk+-2.0
`

coupling_box.o:   coupling_box.c coupling_box.h
        gcc -c coupling_box.c -o coupling_box.o `pkg-config --cflags --libs gtk+
-2.0`

voltage_box.o:     voltage_box.c voltage_box.h
        gcc -c voltage_box.c -o voltage_box.o `pkg-config --cflags --libs gtk+-2
.0`

calculate_box.o:           calculate_box.c calculate_box.h
        gcc -c calculate_box.c -o calculate_box.o `pkg-config --cflags --libs gt
k+-2.0`
```

```
/*
 * @file calculate_box.h
 * @author David TÃ¶yrÃ¤
 * @date 26 april 2019
 * @brief A gtk box containing the calculate functionality of electrotest.
 */

#ifndef CALC_BOX_H_
#define CALC_BOX_H_
#include <gtk/gtk.h>
#include "electrogui.h"
#include <stdlib.h>
#include <string.h>

/**
 * @brief Constructor of the lower GUI part.
 *
 * Contains the calculate button and the output of power and resistance.
 *
 * @param gui A struct that contains pointers to all gui parts and to the
 * array which contains the resistor values.
 * @return GtkWidget* pointer to the widget with the calculate box.
 *
 */
extern GtkWidget* calc_result_box_new(struct gui_comp* gui);
#endif
```

```
/*
 * @file coupling_box.h
 * @author David TÃ¶yrÃ¤
 * @date 26 april 2019
 * @brief A gtk box containing the coupling options of serial or parallel.
 */
#ifndef COUPLING_BOX_H_
#define COUPLING_BOX_H_
#include <gtk/gtk.h>
#include "helper_functions.h"


/**
 * @brief Constructor for the Resistor Coupling Gui part
 *
 * The coupling takes an input of either s or p, it is undefined
 * for something else.
 *
 * @return GtkWidget* pointer to the new created Gui part.
 *
 */
extern GtkWidget* coupling_box_new(void);
#endif
```

```
/*
 * @file electrogui.h
 * @author David TÃ¶yrÃ¤
 * @date 26 april 2019
 * @brief headerfile for main program that assembles and runs the GUI.
 */
#ifndef ELECTROGUI_H_
#define ELECTROGUI_H_
#include <gtk/gtk.h>
#include "coupling_box.h"
#include "voltage_box.h"
#include "libresistance.h"
#include "libpower.h"
#include "libcomponent.h"

/**
 * @brief Struct with pointers to all gui and data parts
 *
 * The struct contains pointers to the three GtkWidget gui
 * parts and the float array that contains the current resistor
 * values.
 */
struct gui_comp {
        GtkWidget* voltage_box;
        GtkWidget* coupling_box;
        GtkWidget* resistor_box;
        GtkWidget* calc_result_box;
        float* resistor_values;
};

/**
 * @brief Searches for a gtklabel by a name from a parent widget.
 *
 * Recursively traverses the list of child widgets from a parent
 * until it has found the widget by a name that is searched.
 *
 * @param parent widget to start searching from.
 * @param name string that represents name of the widget being searched
 * @return gtkwidget matching the name
 */
GtkWidget* find_label(GtkWidget* parent, const gchar* name);

/**
 * @brief Constructor for the three resistor fields GUI.
 *
 * This GUI part contains three resistor input field.
 *
 * @return GtkWidget Contains the constructed GUI part.
 */
GtkWidget* resistor_box_new(void);

/**
 * @brief Updates the value_array with values from the GUI
 *
 * The Function assumes a gfloat array of size 3.
 *
 * @param resistor_box The GUI part to access the user input.
 * @value_array Float array of size 3 that contains the resistor values
 */
gfloat* update_resistor_values(GtkWidget* resistor_box, gfloat *value_array);
#endif
```

```
#ifndef _LIBCOMPONENT_H
#define _LIBCOMPONENT_H
    int e_resistance(float orig_resistance, float *res_array);
#endif
```

```
#ifndef _LIBPOWER_H_
#define _LIBPOWER_H_

float calc_power_r(float voltage, float resistance);
float calc_power_i(float voltage, float current);

#endif
```

```
#include <stdio.h>
#include <stdlib.h>
float calc_resistance(int count, char conn, float *array);
```

```
#ifndef VOLTAGE_BOX_H_
#define VOLTAGE_BOX_H_
#include <gtk/gtk.h>
#include "helper_functions.h"

/**
 * @brief Constructor for voltage gtk view
 *
 * Constructs the voltage_box GUI element which contains
 * the entry field for setting the voltage.
 *
 * @return GtkWidget
 */
extern GtkWidget* voltage_box_new(void);

#endif
```

```
/*
 * @file calculate_box.c
 * @author David TÃ¶yrÃ¤
 * @date 26 april 2019
 * @brief A gtk box containing the calculate functionality of electrotest.
 */

#include <gtk/gtk.h>
#include "calculate_box.h"
#include "voltage_box.h"
#include "coupling_box.h"
#include "helper_functions.h"
#include "electrogui.h"


GtkWidget* result_power;
GtkWidget* result_resistance;

/**
 * @brief on click listener for the calculate button
 * calculates resistance and power with the library functions.
 *
 * @param button calculate button
 * @param gui A struct that contains pointers to all gui parts and to the
 * array which contains the resistor values.
 *
 */
void button_clicked (GtkWidget *button, struct gui_comp* gui) {

        //Get data from voltage and coupling boxes
        update_resistor_values((*gui).resistor_box, (*gui).resistor_values);
        float voltage = atof(gtk_entry_get_text(GTK_ENTRY(find_label((*gui).volt
age_box, "voltage"))));
        char coupling;
        if(strcmp(gtk_entry_get_text(GTK_ENTRY(find_label((*gui).coupling_box, "
coupling"))), "S") == 0
          || strcmp(gtk_entry_get_text(GTK_ENTRY(find_label((*gui).coupling_box,
"coupling"))), "s") == 0)
        {
                coupling = 'S';
                printf("coupling = %s\n", gtk_entry_get_text(GTK_ENTRY(find_label((*
gui).coupling_box, "coupling"))));
        }
        else if (strcmp(gtk_entry_get_text(GTK_ENTRY(find_label((*gui).coupling_
box, "coupling"))) , "P")
          || strcmp(gtk_entry_get_text(GTK_ENTRY(find_label((*gui).coupling_box,
"coupling"))) , "p"))
        {
                coupling = 'P';
                printf("coupling = %s\n", gtk_entry_get_text(GTK_ENTRY(find_label((*
gui).coupling_box, "coupling"))));
        }

        //Calculate resistance and power
        float resistance = calc_resistance(3, coupling, (*gui).resistor_values);
        float power =  calc_power_r(voltage, resistance);

        //Convert result to strings
        char res_resistance[10];
        sprintf(res_resistance, "%.2f",resistance);

        char res_power[10];
```



```
        sprintf(res_power, "%.2f",power);

        //Publish results
        gtk_entry_set_text(GTK_ENTRY(result_resistance), res_resistance);
        gtk_entry_set_editable(GTK_ENTRY(result_resistance), FALSE);
        gtk_entry_set_text(GTK_ENTRY(result_power), res_power);
        gtk_entry_set_editable(GTK_ENTRY(result_power), FALSE);

}



/**
 * @brief Constructor of the lower GUI part.
 *
 * Contains the calculate button and the output of power and resistance.
 *
 * @param gui A struct that contains pointers to all gui parts and to the
 * array which contains the resistor values.
 * @return GtkWidget* pointer to the widget with the calculate box.
 *
 */
GtkWidget* calc_result_box_new(struct gui_comp* gui){

        GtkWidget* lower_vbox;
        GtkWidget* valign;
        GtkWidget* halign;
        GtkWidget* hbox;
        GtkWidget* button;
        GtkWidget* result_resistance_box;
        GtkWidget* result_power_box;

        //Make main box for the calculate function
        lower_vbox = gtk_vbox_new(FALSE, 5);
        valign = gtk_alignment_new(0, 1, 0, 0);
        gtk_container_add(GTK_CONTAINER(lower_vbox), valign);
        hbox = gtk_hbox_new(TRUE, 3);

        //Make button and add callback funtion to it
        button = gtk_button_new_with_label("Calculate");
        gtk_widget_set_size_request(button, 70, 30);
        gtk_container_add(GTK_CONTAINER(hbox), button);


        g_signal_connect(GTK_OBJECT(button), "clicked",
                         GTK_SIGNAL_FUNC(button_clicked), (gpointer *) gui);

        //Make replacement resistance box
        result_resistance_box = gtk_vbox_new(FALSE, 2);
        result_resistance = gtk_entry_new();
        gtk_entry_set_editable(GTK_ENTRY(result_resistance), FALSE);
        GtkWidget *label = gtk_label_new("Replacement Resistance (Ohm)");
        GtkWidget *vbox = gtk_vbox_new(TRUE, 2);

        gtk_container_add(GTK_CONTAINER(vbox), label);
        gtk_container_add(GTK_CONTAINER(vbox), result_resistance);
        gtk_container_add(GTK_CONTAINER(result_resistance_box), vbox);
        gtk_container_add(GTK_CONTAINER(hbox), result_resistance_box);

        //Make power box
        result_power_box = gtk_vbox_new(FALSE, 2);
        result_power = gtk_entry_new();
```

```
        gtk_entry_set_editable(GTK_ENTRY(result_power), FALSE);
        GtkWidget *power_label = gtk_label_new("Power(W)");
        GtkWidget *powervbox = gtk_vbox_new(TRUE, 2);

        gtk_container_add(GTK_CONTAINER(vbox), power_label);
        gtk_container_add(GTK_CONTAINER(vbox), result_power);
        gtk_container_add(GTK_CONTAINER(result_power_box), powervbox);
        gtk_container_add(GTK_CONTAINER(hbox), result_power_box);

        halign = gtk_alignment_new(0, 0, 0, 0);
        gtk_container_add(GTK_CONTAINER(halign), hbox);

        gtk_box_pack_start(GTK_BOX(lower_vbox), halign, FALSE, FALSE, 0);

        return lower_vbox;

}
```

```c
/*
 * @file coupling_box.c
 * @author David TÃ¶yrÃ¤
 * @date 26 april 2019
 * @brief A gtk box containing the coupling options of serial or parallel.
 */
#include <gtk/gtk.h>
#include "coupling_box.h"



/**
 * @brief Constructor for the Resistor Coupling Gui part
 *
 * The coupling takes an input of either s or p, it is undefined
 * for something else.
 *
 * @return GtkWidget* pointer to the new created Gui part.
 *
 */
GtkWidget* coupling_box_new(void)
{
        GtkWidget *coupling_box;
        GtkWidget* entry;
        entry = gtk_entry_new();
        gtk_widget_set_name(entry, "coupling");

        coupling_box = gtk_vbox_new(FALSE, 2);
        GtkWidget *label = gtk_label_new("Coupling S or P: ");
        GtkWidget *hbox = gtk_hbox_new(TRUE,2);

        gtk_container_add(GTK_CONTAINER(hbox), label);
        gtk_container_add(GTK_CONTAINER(hbox), entry);
        gtk_container_add(GTK_CONTAINER(coupling_box), hbox);


        return coupling_box;
}
```

```c
/*
 * @file electrogui.c
 * @author David TÃ¶yrÃ¤
 * @date 26 april 2019
 * @brief main program that assembles and runs the GUI.
 */
#include "electrogui.h"
#include <stdio.h>
#include <string.h>
#include "helper_functions.h"
#include "calculate_box.h"
#include <stdlib.h>
#include "coupling_box.h"
#include "voltage_box.h"



char* const resistor_names[] = { "res1", "res2", "res3" };
char* const resistor_labels[] = {"1:", "2:", "3:"};

/**
 * @brief Searches for a gtklabel by a name from a parent widget.
 *
 * Recursively traverses the list of child widgets from a parent
 * until it has found the widget by a name that is searched.
 *
 * @param parent widget to start searching from.
 * @param name string that represents name of the widget being searched
 * @return gtkwidget matching the name
 */
GtkWidget* find_label(GtkWidget* parent, const gchar* name)
{
        //Check if name matches
        if (g_ascii_strcasecmp(gtk_widget_get_name((GtkWidget*)parent), (gchar*)
name) == 0)
        {
                return parent;
        }


    if (GTK_IS_BIN(parent))
        {
        GtkWidget *child = gtk_bin_get_child(GTK_BIN(parent));
        return find_label(child, name);
    }


        //Get list of children and call this function again.
    if (GTK_IS_CONTAINER(parent))
        {
        GList *children = gtk_container_get_children(GTK_CONTAINER(parent));
        do {
                GtkWidget* widget = find_label(children->data,name);
                if (widget != NULL) {
                        return widget;
                }
        } while ((children = g_list_next(children)) != NULL);
    }
    return NULL;
}

/**
```



```c
 * @brief Constructor for the three resistor fields GUI.
 *
 * This GUI part contains three resistor input field.
 *
 * @return GtkWidget Contains the constructed GUI part.
 */
GtkWidget* resistor_box_new(void){

        GtkWidget* upperrightvbox;
        GtkWidget* resistor_frame;

        upperrightvbox = gtk_vbox_new(FALSE, 5);


        for(int i = 0; i < 3; i++)
        {
                GtkWidget* entry = gtk_entry_new();
                gtk_widget_set_name(entry, resistor_names[i]);
                GtkWidget *label = gtk_label_new(resistor_labels[i]);
                GtkWidget *hbox = gtk_hbox_new(TRUE,2);

                gtk_container_add(GTK_CONTAINER(hbox), label);
                gtk_container_add(GTK_CONTAINER(hbox), entry);
                gtk_container_add(GTK_CONTAINER(upperrightvbox), hbox);
        }

        resistor_frame = gtk_frame_new("Resistors (Ohm)");
        gtk_container_add(GTK_CONTAINER(resistor_frame), upperrightvbox);

        return resistor_frame;
}

/**
 * @brief Updates the value_array with values from the GUI
 *
 * The Function assumes a gfloat array of size 3.
 *
 * @param resistor_box The GUI part to access the user input.
 * @value_array Float array of size 3 that contains the resistor values
 */
gfloat* update_resistor_values(GtkWidget* resistor_box, gfloat *value_array){

        for(int i = 0; i < 3; i++){
                value_array[i] = (gfloat) atof(gtk_entry_get_text(
                GTK_ENTRY( (GtkWidget *) find_label(resistor_box, resistor_names
[i])));
        }

        return value_array;
}

/**
 * @brief closeApp callback function
 *
 * Function to close main GTK application window.
 *
 * @param window, pointer to the main window widget
 */
void closeApp ( GtkWidget *window) {
  gtk_main_quit();
}
```

```
/**
 * @brief main function for the electrotest gui.
 *
 * Uses GTK+ v2.0 library for the graphical interface. Creates the meain window
 * and then the gtk widgets one by one and adds them to the main window.
 *
 */
gint main (gint argc, gchar *argv[]) {

        GtkWidget *window;
        GtkWidget *global_vbox;
        GtkWidget *upper_left_vbox;
        GtkWidget *upper_hbox;

        struct gui_comp gui;
        struct gui_comp* gui_pt = &gui;
        gfloat resistor_values[3] = {1, 1, 1};

        gtk_init(&argc, &argv);

        //Create main frame
        window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
        gtk_window_set_title(GTK_WINDOW(window), "Electrotest");
        gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
        gtk_window_set_default_size(GTK_WINDOW(window),400, 250);
        gtk_container_set_border_width(GTK_CONTAINER(window), 10);

        global_vbox = gtk_vbox_new(FALSE, 5);

        //Create boxes for the data I/O
        gui.voltage_box = voltage_box_new();
        gui.coupling_box = coupling_box_new();
        gui.resistor_box = resistor_box_new();
        gui.calc_result_box = calc_result_box_new(gui_pt);
        gui.resistor_values = resistor_values;

        //Add coupling and voltage fields
        upper_left_vbox = gtk_vbox_new(FALSE, 5);
        gtk_container_add(GTK_CONTAINER(upper_left_vbox), gui.voltage_box);
        gtk_container_add(GTK_CONTAINER(upper_left_vbox), gui.coupling_box);


        upper_hbox = gtk_hbox_new(FALSE, 5);
        gtk_container_add(GTK_CONTAINER(upper_hbox),upper_left_vbox);
        gtk_container_add(GTK_CONTAINER(upper_hbox),gui.resistor_box);
        gtk_container_add(GTK_CONTAINER(global_vbox), upper_hbox);


    gtk_container_add(GTK_CONTAINER(global_vbox), gui.calc_result_box);
    gtk_container_add(GTK_CONTAINER(window), global_vbox);

    g_signal_connect(G_OBJECT(window), "destroy",
          G_CALLBACK(gtk_main_quit), G_OBJECT(window));

    gtk_widget_show_all(window);

    gtk_main();

    return 0;

}
```

```c
#include "voltage_box.h"

/**
 * @brief Constructor for voltage gtk view
 *
 * Constructs the voltage_box GUI element which contains
 * the entry field for setting the voltage.
 *
 * @return GtkWidget
 */
GtkWidget *voltage_box_new(void)
{
        GtkWidget* voltageInfo = gtk_entry_new();
        gtk_widget_set_name(voltageInfo, "voltage");

        GtkWidget* voltage_box = gtk_vbox_new(FALSE, 2);
        GtkWidget *label = gtk_label_new("Voltage (V): ");
        GtkWidget *hbox = gtk_hbox_new(TRUE,2);

        gtk_container_add(GTK_CONTAINER(hbox), label);
        gtk_container_add(GTK_CONTAINER(hbox), voltageInfo);
        gtk_container_add(GTK_CONTAINER(voltage_box), hbox);

        return voltage_box;
}
```