Individual Lab Report #1

Sensors and Motor Control Lab

Sumit Saxena

Team F: Rescue Rangers

Teammates:

Juncheng Zhang

Karthik Ramachandran

Xiaoyang Liu

October 14, 2016

Contents

1. Individual Progress

1.1. Overview

For the Sensors and Motors lab, my primary role was to develop a Graphical User Interface (GUI) for the system to enable screen-based control and real-time tracking of the system. It was important that the GUI I build is user-friendly and has necessary features to enable the user to realize the full functionality of the system. To ensure that, I came up with the following requirements that the GUI should have and validated them with the team:

The GUI shall:

1. Allow the user to switch between 'Sensor-based' and 'GUI-based' control
2. Display in real-time, the state of all the motors (Position and speed for the DC motor; only position for the stepper and the DC motor)
3. Display in real-time, the readings from all the sensors
4. Allow the user to control the position of all the motors
5. Allow the user to control the speed of the DC motor
6. Allow the user to control the direction of rotation of the Stepper motor and the DC motor
7. Allow the user switch the motors on/off

1.2. Implementation

Since I did not have any prior experience in building GUIs, I evaluated my options for the software packages to use. After careful consideration, I settled on using 'Processing' due to the ease of use and implementation, it offers.

Step 1: Finalize on the state variables and to control and track

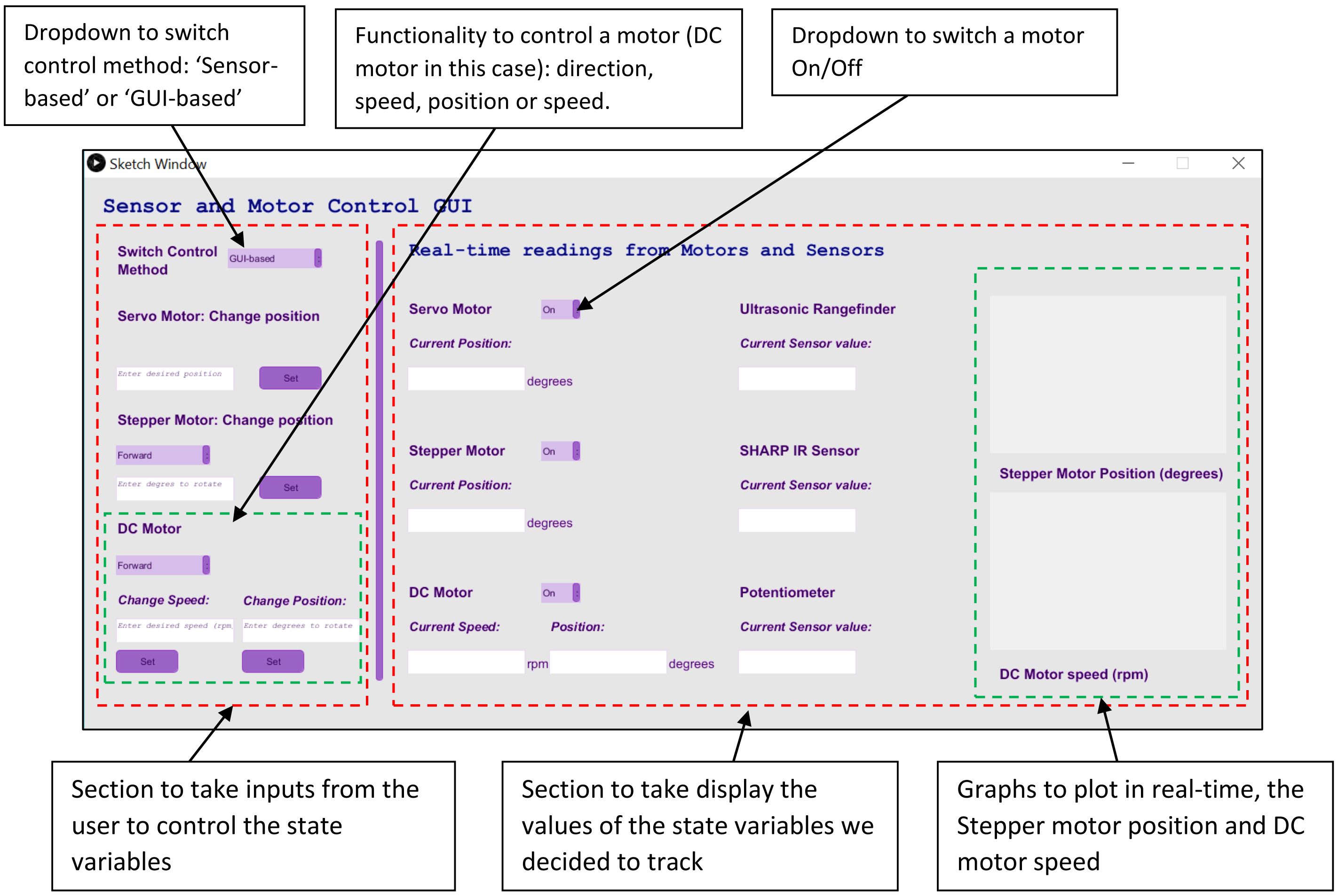
Following from the requirements, it was important to identify clearly the state variables to control and track through the GUI before beginning with the design. I prepared the following list detailing the state variables we needed to control/track for each of the subsystem:

Subsystem	State variables to control	State variables to track
DC Motor	State (On/Off), Speed, Position, Direction	Speed, Position, Direction
Stepper Motor	State (On/Off), Position, Direction	Position, Direction
Servo Motor	State (On/Off), Position	Position
Ultrasonic sensor	N/A	Sensor value
Sharp IR sensor	N/A	Sensor value
Potentiometer	N/A	Sensor value

Table 1: State variables to control and track

Step 2: Design the graphical features of the GUI

I split the interface into two sections: one section allowing the user to give inputs and the other to display all the relevant information. The GUI has two modes: “GUI-based” mode and “Sensor-based” mode. Figures below show the final layout of the graphical interface in the “GUI-based” mode, arrived at after a few iterations:


Dropdown to switch control method: ‘Sensor-based’ or ‘GUI-based’

Functionality to control a motor (DC motor in this case): direction, speed, position or speed.

Dropdown to switch a motor On/Off

Section to take inputs from the user to control the state variables

Section to take display the values of the state variables we decided to track

Graphs to plot in real-time, the Stepper motor position and DC motor speed

Figure 1: GUI layout in the 'GUI-based' mode illustrating various functionalities

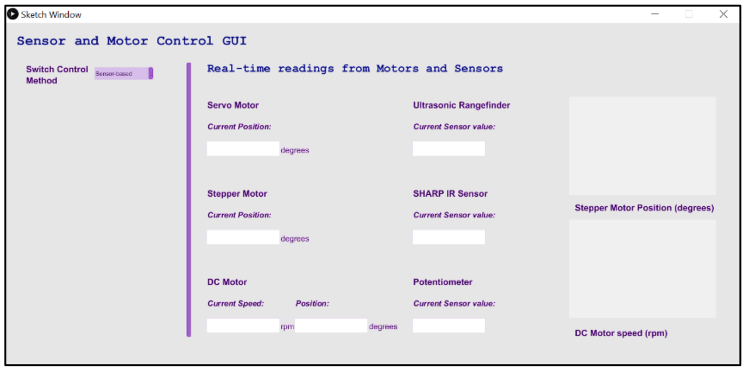


Figure 2: GUI layout in 'Sensor-based' mode

As shown in Figure 2 above, functionality for user input is completely removed from the GUI when operating in the ‘Sensor-based’ mode, except just for the functionality to switch the control mode. This prevents any confusions while operating.

Step 3: Defining the communication protocol

This was an important step in order to have reliable communications. After a discussion with Karthik, who was responsible for writing the firmware side of the communication interface, I defined the protocol in the following manner:

1. Every command/dataset is exchanged in form of a delimited string containing information about the parameters which completely define the action to take or information to display on the GUI
2. Commands from ‘Processing’ to the microcontroller are of two types:
    a) **Switch control mode:** the string has the following structure in this mode:
    “<bom>,<control_mode_to_execute>,<eom>”,
    where <control_mode_to_execute> = 0 for ‘Sensor-based’ and 1 for ‘GUI-based’
    For example,
    “bom,sensor-based,eom’ would be used to switch to ‘Sensor-based’ mode.
    b) **Control the motor:** the string has the following structure in this mode:
    “<bom>,<motor_label>,<desired_state>,<desired_position>,<desired_direction>,<desired_speed>,<eom>”,
    <motor_label>= “m_ser” or “m_stp” or “m_dcm”
    <desired_state>= 0 for ‘Off’ or 1 for ‘On’; ‘na’ for no action
    <desired_position>= numeric value (desired position in case of servo; desire degrees to rotate in case of stepper and DC motors); ‘na’ for no action
    <desired_direction>= 0 for reverse and 1 for forward; ‘na’ for no action
    <desired_speed>= numeric value of the desired speed in rpm; ‘na’ for no action
    For example,
    “bom,m_ser,1,180,na,na,eom” would rotate the servo to 180 degree position
    “bom,m_stp,1,180,1,na,eom” would rotate the stepper by 180 degrees in the forward direction
    “bom,m_dcm,1,na,0,60,eom” would make the DC motor to rotate at 60 rpm in forward direction
3. Data from the sensors/motors after initial processing is passed to the ‘Processing’ program in form of a single delimited string containing information about the states of all the motors and sensors, together. The structure was similar to that described for the commands above, just clubbed together for all the sensors and motors in a single string. Following is the resultant structure:

“<bom>,<servo_motor_state(on/off)>,<servo_motor_position>,<na>,<na>,<stepper_motor_state(on/off)>,<stepper_motor_position>,<stepper_motor_direction>,<na>,<DC_motor_state(on/off)>,<DC_motor_position>,<DC_motor_direction>,<DC_motor_speed>,<Ultrasonic_sensor_state(on/off)>,<Ultrasonic_sensor_value>,<na>,<na>,>,<Infrared_sensor_state(on/off)>,<Infrared_sensor_value>,<na>,<na>,<Potentiometer_state(on/off)>,<Potentiometer_value>,<na>,<na>,<eom>”

Step 3: Writing the logic into a program

The logic to take inputs from user, pass commands to microcontroller, and display information on the GUI was properly structured and then coded in ‘Processing’.

Ensuring robustness:

- The code was tested numerous times to ensure reliability and consistency
- Checks were applied in the user-editable text fields to ensure that the user enters only numeric values
- Other text fields not supposed to be edited by the user were disabled

Step 4: Testing with the complete system

The GUI was tested with the complete system numerous times to ensure proper information flow, reliability and consistency.

1.3. Conclusion:

The project was a success. All the components of the system was working really well. The control and tracking through the GUI was flawless. We could easily see the control of DC motor and Stepper motor in action, real-time on the graphs plotted on the GUI.

2. Challenges:

I faced multiple challenges during the project:

1. Since I was new to GUI development, the primary challenge for me was learning and making things work really fast. Initially, it was difficult for me to assess which environment could be the best for me without any prior practical experience. I did research online and talked to people to figure out which could be the better option.
2. After settling on using ‘Processing’ also, I had difficulty in choosing the library to use to create the GUI. I initially tried the ‘ControlP5’ library but found it difficult to work with and then moved to ‘G4P’ which only allowed simple graphical tools. Though this cost me

some time, the learning I gained was very useful as later, I used 'ControlP5' also to bring in some nice graphs.

Working on 'Processing' was a great learning experience. I achieved things that I had not at all expected at the outset. It is a really easy-to-use tool to help implement things in a short period of time, effectively.

3. Teamwork:

The amount of work needed and explicit categorization of work made it easy to distribute work amongst the team members. Following was the division of work:

- Juncheng Zhang:
    - Develop hardware and firmware for PID control of the DC motor using feedback from encoder
    - Develop hardware and firmware to use potentiometer for velocity and position control of the DC motor
- Karthik Ramachandran:
    - Develop hardware and firmware to control the Servo motor
    - Develop hardware and firmware to use the Ultrasonic sensor for position control of the servo motor
    - Integrate firmware side of code for all the systems and write firmware side of communication interface
- Sumit Saxena:
    - Develop GUI for the system
    - Define communication protocol and write GUI side of communication interface
- Xiaoyang Liu:
    - Develop hardware and firmware to control the Stepper motor
    - Develop hardware and firmware to use the SHARP IR sensor for position control of the stepper motor
    - Integrate hardware of all the individual subsystems into one

The amount of collaboration required to make the project live and work reliably was immense. Though the work was well divided, we worked as a team to tackle the really difficult problems. Specifically, we faced a lot of issues while integrating the firmware side of code of different subsystems into a single code and while developing reliable PID control for the DC motor. We worked well together on these problems to overcome them as quickly as possible.

4. Future plans:

Following are the tasks I plan to do until the next ILR:

- Plan the local search strategy from the perspective of the thermal imaging camera we will use
    - I will look into specifications of the various kinds of thermal imaging cameras available in the market and try to figure out the operating altitudes, fields of view and resolutions these cameras offer
    - This sensor information will help me figure out what altitudes would be optimal for the drone to operate during different kinds of navigation – point to point navigation and navigation during local search around a point

Karthik and Juncheng will work on the same problem, but for long range microphones and RGB cameras, respectively. Once we will have different strategies possible for all these sensors in one place, we will try to find out which configurations of these sensors work well together and what should our navigation strategy be for the best set of sensors' configurations found.

APPENDIX

```
//'Processing' Code:

import g4p_controls.*;

import processing.serial.*;

import java.awt.Font;

import controlP5.*;


Serial myPort;

int drp_operation_state=1;

int drp_ser_state=1;

int drp_stp_state=1;

int drp_dcm_state=1;

int drp_usr_state=1;

int drp_irs_state=1;

int drp_for_state=1;


String string_to_send;


String des_ser_pos;

String des_stp_pos;

String des_dcm_pos;

String des_dcm_speed;


int des_ser_dir;

int des_stp_dir;

int des_dcm_dir;
int i;


ControlP5 cp5;
Chart dc_vel;
Chart stp_degrees;



public void setup(){
 size(1500, 700, JAVA2D);
 createGUI();
 customGUI();
 // Place your setup code here
 String portName = Serial.list()[0];
 myPort = new Serial(this, portName, 9600);

 drp_ser.setSelected(1);
 drp_stp.setSelected(1);
 drp_dcm.setSelected(1);
 drp_usr.setSelected(1);
 drp_irs.setSelected(1);
 drp_for.setSelected(1);
 drp_operation.setSelected(1);
 drp_ser_direction.setSelected(1);
 drp_stp_direction.setSelected(1);
 drp_dcm_direction.setSelected(1);
 lab_servo.setFont(new Font("SanSerif", Font.PLAIN, 18));
 label1.setFont(new Font("SanSerif", Font.PLAIN, 16));
```

```
lab_stepper.setFont(new Font("SanSerif", Font.PLAIN, 18));
label2.setFont(new Font("SanSerif", Font.PLAIN, 16));
lab_dc.setFont(new Font("SanSerif", Font.PLAIN, 18));
label3.setFont(new Font("SanSerif", Font.PLAIN, 16));
label4.setFont(new Font("SanSerif", Font.PLAIN, 16));
label5.setFont(new Font("SanSerif", Font.PLAIN, 16));
label6.setFont(new Font("SanSerif", Font.PLAIN, 16));
label7.setFont(new Font("SanSerif", Font.PLAIN, 18));
label8.setFont(new Font("SanSerif", Font.PLAIN, 16));
label9.setFont(new Font("SanSerif", Font.PLAIN, 18));
label10.setFont(new Font("SanSerif", Font.PLAIN, 16));
label11.setFont(new Font("SanSerif", Font.PLAIN, 18));
label12.setFont(new Font("SanSerif", Font.PLAIN, 16));
label13.setFont(new Font("Monospaced", Font.PLAIN, 28));
label14.setFont(new Font("SanSerif", Font.PLAIN, 18));
label15.setFont(new Font("SanSerif", Font.PLAIN, 18));
label16.setFont(new Font("SanSerif", Font.PLAIN, 18));
label17.setFont(new Font("SanSerif", Font.PLAIN, 18));
label18.setFont(new Font("SanSerif", Font.PLAIN, 16));
label19.setFont(new Font("SanSerif", Font.PLAIN, 16));
label20.setFont(new Font("SanSerif", Font.PLAIN, 16));
label21.setFont(new Font("Monospaced", Font.PLAIN, 24));
label22.setFont(new Font("SanSerif", Font.PLAIN, 16));
label23.setFont(new Font("SanSerif", Font.PLAIN, 18));
label24.setFont(new Font("SanSerif", Font.PLAIN, 18));
txt_set_ser.setFont(new Font("Monospaced", Font.PLAIN, 10));
txt_set_stp.setFont(new Font("Monospaced", Font.PLAIN, 10));
```

```
txt_set_dcm_speed.setFont(new Font("Monospaced", Font.PLAIN, 10));
txt_set_dcm_position.setFont(new Font("Monospaced", Font.PLAIN, 10));
txt_servo_deg.setFont(new Font("Monospaced", Font.PLAIN, 18));
txt_stepper_deg.setFont(new Font("Monospaced", Font.PLAIN, 18));
txt_dc_deg.setFont(new Font("Monospaced", Font.PLAIN, 18));
txt_dc_rpm.setFont(new Font("Monospaced", Font.PLAIN, 18));
txt_usr.setFont(new Font("Monospaced", Font.PLAIN, 18));
txt_irs.setFont(new Font("Monospaced", Font.PLAIN, 18));
txt_pot.setFont(new Font("Monospaced", Font.PLAIN, 18));
button1.setEnabled(false);
txt_servo_deg.setEnabled(false);
txt_stepper_deg.setEnabled(false);
txt_dc_deg.setEnabled(false);
txt_dc_rpm.setEnabled(false);
txt_usr.setEnabled(false);
txt_irs.setEnabled(false);
txt_pot.setEnabled(false);

drp_ser_direction.setVisible(false);
lab_ser_warning.setVisible(false);
lab_stp_warning.setVisible(false);
lab_dcm_warning.setVisible(false);
drp_usr.setVisible(false);
drp_irs.setVisible(false);
drp_for.setVisible(false);

des_stp_dir=1;
```

```
  des_dcm_dir=1;



  frameRate(100);
  string_to_send = "bom,gui-based,eom\n";
  myPort.write(string_to_send);

  cp5 = new ControlP5(this);
  cp5.printPublicMethodsFor(Chart.class);
  dc_vel = cp5.addChart("DC motor velocity")
        .setPosition(1150, 400)
        .setSize(300, 200)
        .setRange(0, 120)
        .setView(Chart.BAR) // use Chart.LINE, Chart.PIE, Chart.AREA, Chart.BAR_CENTERED
        ;

  dc_vel.getColor().setBackground(color(255, 100));

  stp_degrees = cp5.addChart("Stepper motor degrees rotated")
        .setPosition(1150, 150)
        .setSize(300, 200)
        .setRange(0, 10000)
        .setView(Chart.BAR) // use Chart.LINE, Chart.PIE, Chart.AREA, Chart.BAR_CENTERED
        ;

  stp_degrees.getColor().setBackground(color(255, 100));
}
```

```
public void draw(){
  background(230);

  String inString = myPort.readStringUntil('\n');

  if (inString != null) {
    String trimmedString =  inString.trim();
    String readings[] = trimmedString.split(",");

    if (readings[0].equals("bom") && readings[readings.length - 1].equals("eom") &&
readings.length == 27) {

      switch(readings[1]){
        case "na": break;
        case "0" :{ drp_ser.setSelected(0); break; }
        case "1" :{ drp_ser.setSelected(1); break; }
        default: break;
      }
      switch(readings[5]){
        case "na": break;
        case "0" :{ drp_stp.setSelected(0); break; }
        case "1" :{ drp_stp.setSelected(1); break; }
        default: break;
      }
      switch(readings[9]){
        case "na": break;
```

```
          case "0" :{ drp_dcm.setSelected(0); break; }
          case "1" :{ drp_dcm.setSelected(1); break; }
          default: break;
        }

        txt_servo_deg.setText(readings[2]);
        txt_stepper_deg.setText(readings[6]);
        txt_dc_deg.setText(readings[10]);
        txt_dc_rpm.setText(readings[12]);
        txt_usr.setText(readings[14]);
        txt_irs.setText(readings[18]);
        txt_pot.setText(readings[22]);

        i = i+1;
        String step = "Step "+i;
        dc_vel.addDataSet(step);
        dc_vel.updateData(step, int(readings[12]));
        stp_degrees.addDataSet(step);
        stp_degrees.updateData(step, int(readings[6]));



      }

    }
  }
```

```
public void customGUI(){

}


//gui tab code:


/* =========================================================

 * ====                   WARNING                   ===

 * =========================================================

 * The code in this tab has been generated from the GUI form

 * designer and care should be taken when editing this file.

 * Only add/edit code inside the event handlers i.e. only

 * use lines between the matching comment tags. e.g.


 void myBtnEvents(GButton button) { //_CODE_:button1:12356:

     // It is safe to enter your event code here

 } //_CODE_:button1:12356:


 * Do not rename this tab!

 * =========================================================

 */


public void textfield1_change1(GTextField source, GEvent event) {
//_CODE_:txt_servo_deg:553474:

} //_CODE_:txt_servo_deg:553474:


public void textfield1_change2(GTextField source, GEvent event) {
//_CODE_:txt_stepper_deg:449646:
} //_CODE_:txt_stepper_deg:449646:


public void textfield1_change3(GTextField source, GEvent event) {
//_CODE_:txt_dc_rpm:215636:

} //_CODE_:txt_dc_rpm:215636:


public void textfield1_change4(GTextField source, GEvent event) { //_CODE_:txt_usr:924713:

} //_CODE_:txt_usr:924713:


public void textfield1_change5(GTextField source, GEvent event) { //_CODE_:txt_irs:459733:

} //_CODE_:txt_irs:459733:


public void textfield1_change6(GTextField source, GEvent event) { //_CODE_:txt_pot:406254:

} //_CODE_:txt_pot:406254:


public void drp_operation_change(GDropList source, GEvent event) {
//_CODE_:drp_operation:536135:


  drp_operation_state = drp_operation.getSelectedIndex();


  if (drp_operation_state ==0) {

  string_to_send = "bom,sensor-based,eom\n";

  myPort.write(string_to_send);

  //println(string_to_send);

  label15.setVisible(false);

  label16.setVisible(false);

  label17.setVisible(false);
```

```
label18.setVisible(false);

label20.setVisible(false);

drp_ser_direction.setVisible(false);

drp_stp_direction.setVisible(false);

drp_dcm_direction.setVisible(false);

txt_set_ser.setVisible(false);

txt_set_stp.setVisible(false);

txt_set_dcm_speed.setVisible(false);

txt_set_dcm_position.setVisible(false);

cmd_set_ser.setVisible(false);

cmd_set_stp.setVisible(false);

cmd_set_dcm_speed.setVisible(false);

cmd_set_dcm_pos.setVisible(false);

drp_ser.setVisible(false);

drp_stp.setVisible(false);

drp_dcm.setVisible(false);


}

else if (drp_operation_state ==1) {

string_to_send = "bom,gui-based,eom\n";

myPort.write(string_to_send);

//println(string_to_send);

label15.setVisible(true);

label16.setVisible(true);

label17.setVisible(true);

label18.setVisible(true);
```

```
    label20.setVisible(true);
    drp_ser_direction.setVisible(true);
    drp_stp_direction.setVisible(true);
    drp_dcm_direction.setVisible(true);
    txt_set_ser.setVisible(true);
    txt_set_stp.setVisible(true);
    txt_set_dcm_speed.setVisible(true);
    txt_set_dcm_position.setVisible(true);
    cmd_set_ser.setVisible(true);
    cmd_set_stp.setVisible(true);
    cmd_set_dcm_speed.setVisible(true);
    cmd_set_dcm_pos.setVisible(true);
    drp_ser.setVisible(true);
    drp_stp.setVisible(true);
    drp_dcm.setVisible(true);


    }


} //_CODE_:drp_operation:536135:


public void textfield1_change7(GTextField source, GEvent event) {
//_CODE_:txt_set_ser:700573:
} //_CODE_:txt_set_ser:700573:


public void set_ser_clicked(GButton source, GEvent event) { //_CODE_:cmd_set_ser:241633:
```

```
des_ser_pos= txt_set_ser.getText();


int len_inputString;

len_inputString = des_ser_pos.length();

int k=0;

int count_invalid_ascii = 0;

char dig_k;

int dig_k_ascii;


 while(k<(len_inputString-1)){

  dig_k = des_ser_pos.charAt(k);

  println (int(dig_k));

  if (int(dig_k)<48 || int(dig_k)>57){

   count_invalid_ascii=1;

   break;

  }

  k++;

 }


 if (count_invalid_ascii>0){

  lab_ser_warning.setVisible(true);

 }

 else {

  lab_ser_warning.setVisible(false);

  string_to_send = "bom,m_ser,1,"+ des_ser_pos +",1,na,eom\n";

  myPort.write(string_to_send);
```

```
    }


} //_CODE_:cmd_set_ser:241633:


public void textfield1_change8(GTextField source, GEvent event) {
//_CODE_:txt_set_stp:851015:

} //_CODE_:txt_set_stp:851015:


public void drp_ser_change(GDropList source, GEvent event) {
//_CODE_:drp_ser_direction:868128:

  des_ser_dir= drp_ser_direction.getSelectedIndex();


} //_CODE_:drp_ser_direction:868128:


public void drp_stp_change(GDropList source, GEvent event) {
//_CODE_:drp_stp_direction:426702:

   des_stp_dir= drp_stp_direction.getSelectedIndex();

} //_CODE_:drp_stp_direction:426702:


public void set_stp_clicked(GButton source, GEvent event) { //_CODE_:cmd_set_stp:897329:


  des_stp_pos= txt_set_stp.getText();


  int len_inputString;

  len_inputString = des_stp_pos.length();

  int k=1;

  int count_invalid_ascii = 0;
  char dig_k;

  int dig_k_ascii;


    while(k<(len_inputString-1)){

     dig_k = des_stp_pos.charAt(k);

     if (int(dig_k)<48 || int(dig_k)>57){

      count_invalid_ascii=1;

      break;

     }

     k++;

    }


    if (count_invalid_ascii>0){

     lab_stp_warning.setVisible(true);

    }

    else {

     lab_stp_warning.setVisible(false);

     string_to_send = "bom,m_stp,1,"+ des_stp_pos +","+ des_stp_dir+ ",na,eom\n";

     myPort.write(string_to_send);

    }


} //_CODE_:cmd_set_stp:897329:


public void textfield1_change12(GTextField source, GEvent event) {
//_CODE_:txt_set_dcm_speed:626067:

} //_CODE_:txt_set_dcm_speed:626067:
```

```
public void set_dcm_speed_clicked(GButton source, GEvent event) {
//_CODE_:cmd_set_dcm_speed:960054:


 des_dcm_speed= txt_set_dcm_speed.getText();


 int len_inputString;

 len_inputString = des_dcm_speed.length();

 int k=1;

 int count_invalid_ascii = 0;

 char dig_k;

 int dig_k_ascii;


  while(k<(len_inputString-1)){

   dig_k = des_dcm_speed.charAt(k);

   if (int(dig_k)<48 || int(dig_k)>57){

    count_invalid_ascii=1;

    break;

   }

   k++;

  }


  if (count_invalid_ascii>0){

   lab_dcm_warning.setVisible(true);

  }

  else {
      lab_dcm_warning.setVisible(false);

      string_to_send = "bom,m_dcm,1,na,"+ des_dcm_dir+","+des_dcm_speed+ ",eom\n";

      myPort.write(string_to_send);

    }



} //_CODE_:cmd_set_dcm_speed:960054:


public void textfield1_change10(GTextField source, GEvent event) {
//_CODE_:txt_set_dcm_position:674107:

} //_CODE_:txt_set_dcm_position:674107:


public void drp_dcm_change(GDropList source, GEvent event) {
//_CODE_:drp_dcm_direction:868614:

   des_dcm_dir= drp_dcm_direction.getSelectedIndex();

} //_CODE_:drp_dcm_direction:868614:


public void set_dcm_pos_clicked(GButton source, GEvent event) {
//_CODE_:cmd_set_dcm_pos:930818:


  des_dcm_pos= txt_set_dcm_position.getText();


  int len_inputString;

  len_inputString = des_dcm_pos.length();

  int k=1;

  int count_invalid_ascii = 0;

  char dig_k;
  int dig_k_ascii;


   while(k<(len_inputString-1)){
    dig_k = des_dcm_pos.charAt(k);
    if (int(dig_k)<48 || int(dig_k)>57){
      count_invalid_ascii=1;
      break;
    }
    k++;
   }


   if (count_invalid_ascii>0){
    lab_dcm_warning.setVisible(true);
   }
   else {
    lab_dcm_warning.setVisible(false);
    string_to_send = "bom,m_dcm,1,"+ des_dcm_pos + ","+des_dcm_dir+ ",na,eom\n";
    myPort.write(string_to_send);
   }



} //_CODE_:cmd_set_dcm_pos:930818:


public void drp_usr_change(GDropList source, GEvent event) { //_CODE_:drp_usr:306460:
  int temp;
  temp= drp_usr.getSelectedIndex();
  if (drp_usr_state != temp && (temp == 0 || temp == 1)) {

    drp_usr_state = temp;

    string_to_send = "bom,s_usr,"+ drp_usr_state +",0,na,na,eom\n";
    myPort.write(string_to_send);
  }
} //_CODE_:drp_usr:306460:

public void drp_irs_change(GDropList source, GEvent event) { //_CODE_:drp_irs:328196:
  int temp;
  temp= drp_irs.getSelectedIndex();

  if (drp_irs_state != temp && (temp == 0 || temp == 1)) {

    drp_irs_state = temp;

    string_to_send = "bom,s_irs,"+ drp_irs_state +",0,na,na,eom\n";
    myPort.write(string_to_send);
  }
} //_CODE_:drp_irs:328196:

public void drp_for_change(GDropList source, GEvent event) { //_CODE_:drp_for:429334:
  int temp;
  temp= drp_for.getSelectedIndex();
  if (drp_for_state != temp && (temp == 0 || temp == 1)) {

    drp_for_state = temp;

    string_to_send = "bom,s_for,"+ drp_for_state +",0,na,na,eom\n";
    myPort.write(string_to_send);

  }
} //_CODE_:drp_for:429334:

public void drp_ser_on(GDropList source, GEvent event) { //_CODE_:drp_ser:348150:

    drp_ser_state = drp_ser.getSelectedIndex();

    string_to_send = "bom,m_ser,"+ drp_ser_state +",0,na,na,eom\n";
    myPort.write(string_to_send);

} //_CODE_:drp_ser:348150:

public void drp_stp_on(GDropList source, GEvent event) { //_CODE_:drp_stp:519247:

    drp_stp_state =  drp_stp.getSelectedIndex();

    string_to_send = "bom,m_stp,"+ drp_stp_state +",0,na,na,eom\n";
    myPort.write(string_to_send);
```

```
} //_CODE_:drp_stp:519247:

public void drp_dcm_on(GDropList source, GEvent event) { //_CODE_:drp_dcm:574413:

   drp_dcm_state = drp_dcm.getSelectedIndex();

   string_to_send = "bom,m_dcm,"+ drp_dcm_state +",0,na,na,eom\n";
   myPort.write(string_to_send);

} //_CODE_:drp_dcm:574413:

public void textfield1_change9(GTextField source, GEvent event) {
//_CODE_:txt_dc_deg:836902:
} //_CODE_:txt_dc_deg:836902:

public void button1_click1(GButton source, GEvent event) { //_CODE_:button1:670886:
} //_CODE_:button1:670886:



// Create all the GUI controls.
// autogenerated do not edit
public void createGUI(){
  G4P.messagesEnabled(false);
  G4P.setGlobalColorScheme(GCScheme.BLUE_SCHEME);
```

```
G4P.setCursor(ARROW);

surface.setTitle("Sketch Window");

lab_servo = new GLabel(this, 410, 150, 300, 32);

lab_servo.setText("Servo Motor");

lab_servo.setTextBold();

lab_servo.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

lab_servo.setOpaque(false);

txt_servo_deg = new GTextField(this, 410, 240, 150, 30, G4P.SCROLLBARS_NONE);

txt_servo_deg.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

txt_servo_deg.setOpaque(true);

txt_servo_deg.addEventHandler(this, "textfield1_change1");

label1 = new GLabel(this, 560, 248, 80, 20);

label1.setText("degrees");

label1.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label1.setOpaque(false);

lab_stepper = new GLabel(this, 410, 330, 300, 32);

lab_stepper.setText("Stepper Motor");

lab_stepper.setTextBold();

lab_stepper.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

lab_stepper.setOpaque(false);

txt_stepper_deg = new GTextField(this, 410, 420, 150, 30, G4P.SCROLLBARS_NONE);

txt_stepper_deg.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

txt_stepper_deg.setOpaque(true);

txt_stepper_deg.addEventHandler(this, "textfield1_change2");

label2 = new GLabel(this, 560, 428, 80, 20);

label2.setText("degrees");
```

```
label2.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label2.setOpaque(false);

lab_dc = new GLabel(this, 410, 510, 300, 32);

lab_dc.setText("DC Motor");

lab_dc.setTextBold();

lab_dc.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

lab_dc.setOpaque(false);

txt_dc_rpm = new GTextField(this, 410, 600, 150, 30, G4P.SCROLLBARS_NONE);

txt_dc_rpm.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

txt_dc_rpm.setOpaque(true);

txt_dc_rpm.addEventHandler(this, "textfield1_change3");

label3 = new GLabel(this, 560, 607, 34, 20);

label3.setText("rpm");

label3.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label3.setOpaque(false);

label4 = new GLabel(this, 410, 200, 150, 20);

label4.setText("Current Position:");

label4.setTextBold();

label4.setTextItalic();

label4.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label4.setOpaque(false);

label5 = new GLabel(this, 410, 380, 150, 20);

label5.setText("Current Position:");

label5.setTextBold();

label5.setTextItalic();

label5.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
```

```
label5.setOpaque(false);

label6 = new GLabel(this, 410, 560, 150, 20);

label6.setText("Current Speed:");

label6.setTextBold();

label6.setTextItalic();

label6.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label6.setOpaque(false);

label7 = new GLabel(this, 830, 150, 300, 32);

label7.setText("Ultrasonic Rangefinder");

label7.setTextBold();

label7.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label7.setOpaque(false);

label8 = new GLabel(this, 830, 200, 250, 20);

label8.setText("Current Sensor value:");

label8.setTextBold();

label8.setTextItalic();

label8.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label8.setOpaque(false);

txt_usr = new GTextField(this, 830, 240, 150, 30, G4P.SCROLLBARS_NONE);

txt_usr.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

txt_usr.setOpaque(true);

txt_usr.addEventHandler(this, "textfield1_change4");

label9 = new GLabel(this, 830, 330, 300, 32);

label9.setText("SHARP IR Sensor");

label9.setTextBold();

label9.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
```

```
label9.setOpaque(false);
label10 = new GLabel(this, 830, 380, 250, 20);
label10.setText("Current Sensor value:");
label10.setTextBold();
label10.setTextItalic();
label10.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label10.setOpaque(false);
txt_irs = new GTextField(this, 830, 420, 150, 30, G4P.SCROLLBARS_NONE);
txt_irs.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
txt_irs.setOpaque(true);
txt_irs.addEventHandler(this, "textfield1_change5");
label11 = new GLabel(this, 830, 510, 300, 32);
label11.setText("Potentiometer");
label11.setTextBold();
label11.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label11.setOpaque(false);
label12 = new GLabel(this, 830, 560, 250, 20);
label12.setText("Current Sensor value:");
label12.setTextBold();
label12.setTextItalic();
label12.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label12.setOpaque(false);
txt_pot = new GTextField(this, 830, 600, 150, 30, G4P.SCROLLBARS_NONE);
txt_pot.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
txt_pot.setOpaque(true);
txt_pot.addEventHandler(this, "textfield1_change6");
```

```
label13 = new GLabel(this, 21, 18, 800, 32);
label13.setText("Sensor and Motor Control GUI");
label13.setTextBold();
label13.setOpaque(false);
label14 = new GLabel(this, 40, 80, 150, 100);
label14.setTextAlign(GAlign.LEFT, GAlign.TOP);
label14.setText("Switch Control Method");
label14.setTextBold();
label14.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label14.setOpaque(false);
drp_operation = new GDropList(this, 182, 90, 120, 100, 3);
drp_operation.setItems(loadStrings("list_536135"), 0);
drp_operation.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
drp_operation.addEventHandler(this, "drp_operation_change");
label15 = new GLabel(this, 40, 140, 300, 70);
label15.setText("Servo Motor: Change position");
label15.setTextBold();
label15.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label15.setOpaque(false);
txt_set_ser = new GTextField(this, 40, 240, 150, 30, G4P.SCROLLBARS_NONE);
txt_set_ser.setPromptText("Enter desired position");
txt_set_ser.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
txt_set_ser.setOpaque(true);
txt_set_ser.addEventHandler(this, "textfield1_change7");
cmd_set_ser = new GButton(this, 222, 240, 80, 30);
cmd_set_ser.setText("Set");
cmd_set_ser.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

cmd_set_ser.addEventHandler(this, "set_ser_clicked");

label16 = new GLabel(this, 40, 291, 300, 32);

label16.setText("Stepper Motor: Change position");

label16.setTextBold();

label16.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label16.setOpaque(false);

txt_set_stp = new GTextField(this, 40, 380, 150, 30, G4P.SCROLLBARS_NONE);

txt_set_stp.setPromptText("Enter degres to rotate");

txt_set_stp.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

txt_set_stp.setOpaque(true);

txt_set_stp.addEventHandler(this, "textfield1_change8");

drp_ser_direction = new GDropList(this, 40, 200, 120, 100, 3);

drp_ser_direction.setItems(loadStrings("list_868128"), 0);

drp_ser_direction.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

drp_ser_direction.addEventHandler(this, "drp_ser_change");

drp_stp_direction = new GDropList(this, 40, 340, 120, 100, 3);

drp_stp_direction.setItems(loadStrings("list_426702"), 0);

drp_stp_direction.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

drp_stp_direction.addEventHandler(this, "drp_stp_change");

cmd_set_stp = new GButton(this, 222, 379, 80, 30);

cmd_set_stp.setText("Set");

cmd_set_stp.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

cmd_set_stp.addEventHandler(this, "set_stp_clicked");

label17 = new GLabel(this, 40, 429, 300, 32);

label17.setText("DC Motor");
```

```
label17.setTextBold();
label17.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label17.setOpaque(false);
txt_set_dcm_speed = new GTextField(this, 40, 560, 150, 30, G4P.SCROLLBARS_NONE);
txt_set_dcm_speed.setPromptText("Enter desired speed (rpm)");
txt_set_dcm_speed.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
txt_set_dcm_speed.setOpaque(true);
txt_set_dcm_speed.addEventHandler(this, "textfield1_change12");
cmd_set_dcm_speed = new GButton(this, 40, 600, 80, 30);
cmd_set_dcm_speed.setText("Set");
cmd_set_dcm_speed.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
cmd_set_dcm_speed.addEventHandler(this, "set_dcm_speed_clicked");
label18 = new GLabel(this, 40, 520, 150, 32);
label18.setText("Change Speed:");
label18.setTextBold();
label18.setTextItalic();
label18.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label18.setOpaque(false);
txt_set_dcm_position = new GTextField(this, 200, 560, 150, 30, G4P.SCROLLBARS_NONE);
txt_set_dcm_position.setPromptText("Enter degrees to rotate");
txt_set_dcm_position.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
txt_set_dcm_position.setOpaque(true);
txt_set_dcm_position.addEventHandler(this, "textfield1_change10");
drp_dcm_direction = new GDropList(this, 40, 480, 120, 100, 3);
drp_dcm_direction.setItems(loadStrings("list_868614"), 0);
drp_dcm_direction.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
drp_dcm_direction.addEventHandler(this, "drp_dcm_change");

cmd_set_dcm_pos = new GButton(this, 200, 600, 80, 30);

cmd_set_dcm_pos.setText("Set");

cmd_set_dcm_pos.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

cmd_set_dcm_pos.addEventHandler(this, "set_dcm_pos_clicked");

drp_usr = new GDropList(this, 1050, 155, 50, 100, 3);

drp_usr.setItems(loadStrings("list_306460"), 0);

drp_usr.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

drp_usr.addEventHandler(this, "drp_usr_change");

drp_irs = new GDropList(this, 1050, 335, 50, 100, 3);

drp_irs.setItems(loadStrings("list_328196"), 0);

drp_irs.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

drp_irs.addEventHandler(this, "drp_irs_change");

drp_for = new GDropList(this, 1050, 515, 50, 100, 3);

drp_for.setItems(loadStrings("list_429334"), 0);

drp_for.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

drp_for.addEventHandler(this, "drp_for_change");

drp_ser = new GDropList(this, 580, 155, 50, 100, 3);

drp_ser.setItems(loadStrings("list_348150"), 1);

drp_ser.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

drp_ser.addEventHandler(this, "drp_ser_on");

drp_stp = new GDropList(this, 580, 335, 50, 100, 3);

drp_stp.setItems(loadStrings("list_519247"), 0);

drp_stp.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

drp_stp.addEventHandler(this, "drp_stp_on");

drp_dcm = new GDropList(this, 580, 515, 50, 100, 3);
```

```
drp_dcm.setItems(loadStrings("list_574413"), 0);
drp_dcm.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
drp_dcm.addEventHandler(this, "drp_dcm_on");
label19 = new GLabel(this, 590, 560, 150, 20);
label19.setText("Position:");
label19.setTextBold();
label19.setTextItalic();
label19.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label19.setOpaque(false);
txt_dc_deg = new GTextField(this, 590, 600, 150, 30, G4P.SCROLLBARS_NONE);
txt_dc_deg.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
txt_dc_deg.setOpaque(true);
txt_dc_deg.addEventHandler(this, "textfield1_change9");
label20 = new GLabel(this, 199, 521, 150, 32);
label20.setText("Change Position:");
label20.setTextBold();
label20.setTextItalic();
label20.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
label20.setOpaque(false);
label21 = new GLabel(this, 410, 75, 730, 32);
label21.setText("Real-time readings from Motors and Sensors");
label21.setTextBold();
label21.setOpaque(false);
label22 = new GLabel(this, 740, 607, 80, 20);
label22.setText("degrees");
label22.setLocalColorScheme(GCScheme.PURPLE_SCHEME);
```

```
label22.setOpaque(false);

button1 = new GButton(this, 370, 80, 10, 560);

button1.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

button1.addEventHandler(this, "button1_click1");

label23 = new GLabel(this, 1160, 615, 200, 30);

label23.setText("DC Motor speed (rpm)");

label23.setTextBold();

label23.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label23.setOpaque(false);

label24 = new GLabel(this, 1160, 360, 300, 30);

label24.setText("Stepper Motor Position (degrees)");

label24.setTextBold();

label24.setLocalColorScheme(GCScheme.PURPLE_SCHEME);

label24.setOpaque(false);

lab_ser_warning = new GLabel(this, 172, 191, 192, 40);

lab_ser_warning.setTextAlign(GAlign.CENTER, GAlign.MIDDLE);

lab_ser_warning.setText("Invalid input! Enter numeric values between 0 and 180");

lab_ser_warning.setTextBold();

lab_ser_warning.setLocalColorScheme(GCScheme.RED_SCHEME);

lab_ser_warning.setOpaque(false);

lab_stp_warning = new GLabel(this, 172, 331, 192, 40);

lab_stp_warning.setTextAlign(GAlign.CENTER, GAlign.MIDDLE);

lab_stp_warning.setText("Invalid input! Enter numeric values between 0 and 180");

lab_stp_warning.setTextBold();

lab_stp_warning.setLocalColorScheme(GCScheme.RED_SCHEME);

lab_stp_warning.setOpaque(false);
```

```
  lab_dcm_warning = new GLabel(this, 172, 471, 192, 40);
  lab_dcm_warning.setTextAlign(GAlign.CENTER, GAlign.MIDDLE);
  lab_dcm_warning.setText("Invalid input! Enter numeric values");
  lab_dcm_warning.setTextBold();
  lab_dcm_warning.setLocalColorScheme(GCScheme.RED_SCHEME);
  lab_dcm_warning.setOpaque(false);
}

// Variable declarations
// autogenerated do not edit
GLabel lab_servo;
GTextField txt_servo_deg;
GLabel label1;
GLabel lab_stepper;
GTextField txt_stepper_deg;
GLabel label2;
GLabel lab_dc;
GTextField txt_dc_rpm;
GLabel label3;
GLabel label4;
GLabel label5;
GLabel label6;
GLabel label7;
GLabel label8;
GTextField txt_usr;
GLabel label9;
GLabel label10;

GTextField txt_irs;

GLabel label11;

GLabel label12;

GTextField txt_pot;

GLabel label13;

GLabel label14;

GDropList drp_operation;

GLabel label15;

GTextField txt_set_ser;

GButton cmd_set_ser;

GLabel label16;

GTextField txt_set_stp;

GDropList drp_ser_direction;

GDropList drp_stp_direction;

GButton cmd_set_stp;

GLabel label17;

GTextField txt_set_dcm_speed;

GButton cmd_set_dcm_speed;

GLabel label18;

GTextField txt_set_dcm_position;

GDropList drp_dcm_direction;

GButton cmd_set_dcm_pos;

GDropList drp_usr;

GDropList drp_irs;

GDropList drp_for;
GDropList drp_ser;

GDropList drp_stp;

GDropList drp_dcm;

GLabel label19;

GTextField txt_dc_deg;

GLabel label20;

GLabel label21;

GLabel label22;

GButton button1;

GLabel label23;

GLabel label24;

GLabel lab_ser_warning;

GLabel lab_stp_warning;

GLabel lab_dcm_warning;
```