ANEXOS

ANEXO A

A.1. FIRMWARE PARA TARJETA ARDUINO

El firmware o código fuente que se carga a la tarjeta Arduino se presenta a continuación.

A.1.1 Declaración de Variables, Librerías y Configuración

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[]={0x90, 0xA2, 0xDA, 0x0E, 0xDF, 0xE7};//colocamos la mac de arduino
IPAddress ip(192,168,122,50);// Asignamos la direccion ip de la red que queremos
EthernetServer server(23); // Asignamos el puerto de comunicación

char caracterPC=' ';
String comando=" ";
char caracterIn=' ';
String respuesta=" ",dato=" ";
boolean respuestaCompleta=false;
String TipoDato="",cabDato=" ";
int i=0;


void setup()
{
  Ethernet.begin(mac,ip);//Cargamos la direccion IP y la mac para el modo servidor Ethernet
  server.begin();//Sepone en marcha el modo servidor
  Serial.begin(9600);// se configura la velocidad de comunicacion serial
  Serial.print("serial inicializado");

}
```

Figura A.1 Librerías, variables y configuración.

En la figura A.1 se observa las librerías necesarias para desarrollar el firmware, la SPI y la Ethernet que nos permiten utilizar la tarjeta como servidor.

En el Loop se realiza en primera instancia un siclo while cuando hay conexión con algún socket del App de java, dentro del cual se capturan los datos que llegan al puerto Ethernet, se imprimen en el serial y se reinicia la variable encargada de recibir los datos. En la figura A.2 se muestra la parte del código que realiza lo descrito anteriormente.

```
void loop()
{

  EthernetClient client=server.available();//Crea un objeto EthernetClient para encender el servidor
  if (client==true)//Se pregunta si existe el servidor
  {
    while(client.connected())//Mientras el servidor este conectado a un socket
    {
      if(client.available())// y si el cliente esta activo se ingresa al proceso de
      {                     //recibir datos por el puerto ethernet
        caracterPC='-';
        while (caracterPC!='\n' )//Mientras la variable caracter no encuentre un retorno de carro o enter
        {

           caracterPC=client.read();//Se lee lo que hay en el buffer ethernet
           comando = comando+caracterPC; //y se asigna a la variable comando
        }//cuando termina de recibir toda la palabra enbiada desde el App termina el siclo

        Serial.print(comando);// se imprime el dato enviado desde el App en el puerto serial
        delay(10);// Se esperan 10 milisegundos
        comando="";// se reinicia la variable
           }
```

Figura A.2 Código para lectura de datos del puerto Ethernet.

Luego de ejecutar el código anterior y dentro del mismo siclo while se revisa la conexión con el transmisor a través del puerto serie, si hay conexión se leen los datos demarcados en el formato ANSI VT100 delimitados por los corchetes“[”. Cundo un dato está completo se envía a la función “tipoRespuesta” que se describe más adelante, el resultado de la función es asignada a una variable y esta enviada por el puerto Ethernet al App. Por último se reinician las variables temporales. En la figura A.3 se observa el código fuente de la descripción anterior.

```
  if (Serial.available()>0)//si el serial esta activo y tiene datos por recibir
      {
       caracterIn=(char)Serial.read();// se lee cada caracter en la varible caracterIn
       if(caracterIn!='['){ // si el caracter no es el final de linea ([)
       respuesta+=caracterIn;//en la variable respuesta se guarda cada caracter que entre
       }else// si es igual la respuesta es completada
         {
          respuestaCompleta=true;
         }
      if(respuestaCompleta==true)// si la respuesta es completa, se envia el dato a la funcion
        {                        // tipoRespuesta la cual verifica si el dato es un parametro.
         dato=tipoRespuesta(respuesta);
         TipoDato="";

         client.println(dato);// Si la respuesta es un parametro se imprime el nuevo formato por el purto Ethernet sino no se imprime nada.

         respuesta="";
         respuestaCompleta=false;
        }
        }
 }

}
```

Figura A.3 Código para lectura de datos del puerto serie.

La función “tipoRespuesta” recibe como parámetro el dato recibido por el puerto serie, este parámetro es comparado en sus 4 o 6 primeros caracteres con otros específicos que representan datos de información útil como valores de estado, potencia, voltajes etc. Si se cumple alguna condición se crea una nueva variable y se le asigna una cabecera que identifica el tipo de dato, luego con un siclo se asigna la carga útil de los datos llegados por el puerto serie y por último se agrega el tamaño de la cabecera. Este nuevo dato encapsulado en una nueva trama es retornado por la función. A continuación en la figura A.4 se muestra el código de la función “tipoRespuesta”.

```
String tipoRespuesta(String a){// esta funcion compara los datos que se pasan como parametros
                               // con las opciones que se muestran.
  if(/*a.charAt(0)=='['&*/a.charAt(0)=='8'&a.charAt(1)==';'&a.charAt(2)=='6'&a.charAt(3)=='2'&a.charAt(4)=='H'){
    TipoDato="@COR";//Corriente del sistema
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }// Si el dato es correcto se crea una cabecera TipoDato y sele agrega solo la carga util de los datos pasados como parametro
    TipoDato+=4;//por ultimo se añade el tamaño de la cabecera
  }else
  if(a.charAt(0)=='7'&a.charAt(1)==';'&a.charAt(2)=='6'&a.charAt(3)=='2'&a.charAt(4)=='H'){
    TipoDato="@VOL";//Voltage del sistema
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='6'&a.charAt(1)==';'&a.charAt(2)=='6'&a.charAt(3)=='2'&a.charAt(4)=='H'){
    TipoDato="@RAF";//RAdio frecuencia estado
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='8'&a.charAt(1)==';'&a.charAt(2)=='2'&a.charAt(3)=='6'&a.charAt(4)=='H'){
    TipoDato="@EFI";//Eficiencia del transmisor
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);

    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='7'&a.charAt(1)==';'&a.charAt(2)=='2'&a.charAt(3)=='6'&a.charAt(4)=='H'){
    TipoDato="@POR";//Potencia reflejada
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='6'&a.charAt(1)==';'&a.charAt(2)=='2'&a.charAt(3)=='6'&a.charAt(4)=='H'){
    TipoDato="@POE";//Potencia entregada
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='1'&a.charAt(1)=='1'&a.charAt(2)==';'&a.charAt(3)=='1'&a.charAt(4)=='0'&a.charAt(5)=='H'){
    TipoDato="@AL1";//Alarma 1
    for(i=6;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='1'&a.charAt(1)=='0'&a.charAt(2)==';'&a.charAt(3)=='1'&a.charAt(4)=='0'&a.charAt(5)=='H'){
    TipoDato="@AL2";//Alarma 2
```

```
    for(i=6;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='9'&a.charAt(1)==';'&a.charAt(2)=='1'&a.charAt(3)=='0'&a.charAt(4)=='H'){
    TipoDato="@AL3";//Alarma 3
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='8'&a.charAt(1)==';'&a.charAt(2)=='1'&a.charAt(3)=='0'&a.charAt(4)=='H'){
    TipoDato="@AL4";//Alarma 4
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='7'&a.charAt(1)==';'&a.charAt(2)=='1'&a.charAt(3)=='0'&a.charAt(4)=='H'){
    TipoDato="@AL5";//Alarma 5
    for(i=5;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=4;
  }else
  if(a.charAt(0)=='2'&a.charAt(1)=='2'&a.charAt(2)==';'&a.charAt(3)=='2'&a.charAt(4)=='1'&a.charAt(5)=='H'){
    TipoDato="@TFE";//Temperatura Fan Speed


    for(i=6;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=6;
  }else
  if(a.charAt(0)=='2'&a.charAt(1)=='2'&a.charAt(2)==';'&a.charAt(3)=='6'&a.charAt(4)=='6'&a.charAt(5)=='H'){
     TipoDato="@POC23";
    for(i=6;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=6;
  }else
  if(a.charAt(0)=='2'&a.charAt(1)=='3'&a.charAt(2)==';'&a.charAt(3)=='6'&a.charAt(4)=='6'&a.charAt(5)=='H'){
     TipoDato="@POC24";
    for(i=6;i<a.length();i++){
    TipoDato+=a.charAt(i);
    }
    TipoDato+=6;
  }

 return TipoDato;//Al terminar la comparacion se retorna la nueva trama para ser transmitida via ethernet.
}
```

Figura A.4 Código de la función tipoRespuesta.

ANEXO B

B.1. CONFIGURACIÓN DE EQUIPO 1 COMO AP WDS

El equipo PS2 que se encuentra en La Rejoya, conectado a la tarjeta Arduino, se configura como AP, para ingresar a la configuración del firmware de Ubiquiti se conecta un PC al PS2 por medio de un cable UTP como se describe en el capítulo 3 en el ítem 3.4.2. Se configura la dirección del PC para que este en la misma red del PS2, luego se abre un navegador web y se digita la dirección IP del PS2 que por defecto es 192.168.1.20. Se ingresa a una interfaz gráfica que muestra varias opciones, en la pestaña Link Setup se configura el wireless mode como Access Point WDS y en WDS Peers se introduce la dirección MAC del PS2 que se configurará como Estación WDS, en el campo SSID se ingresa el nombre que identifica la red inalámbrica, se coloca el código de país en Country Code, se escoge el canal de comunicación y por último se escoge el valor de la potencia de salida, para este caso es de 26 dBm. En la figura B.1 se observa la configuración realizada para el AP del SGCR.

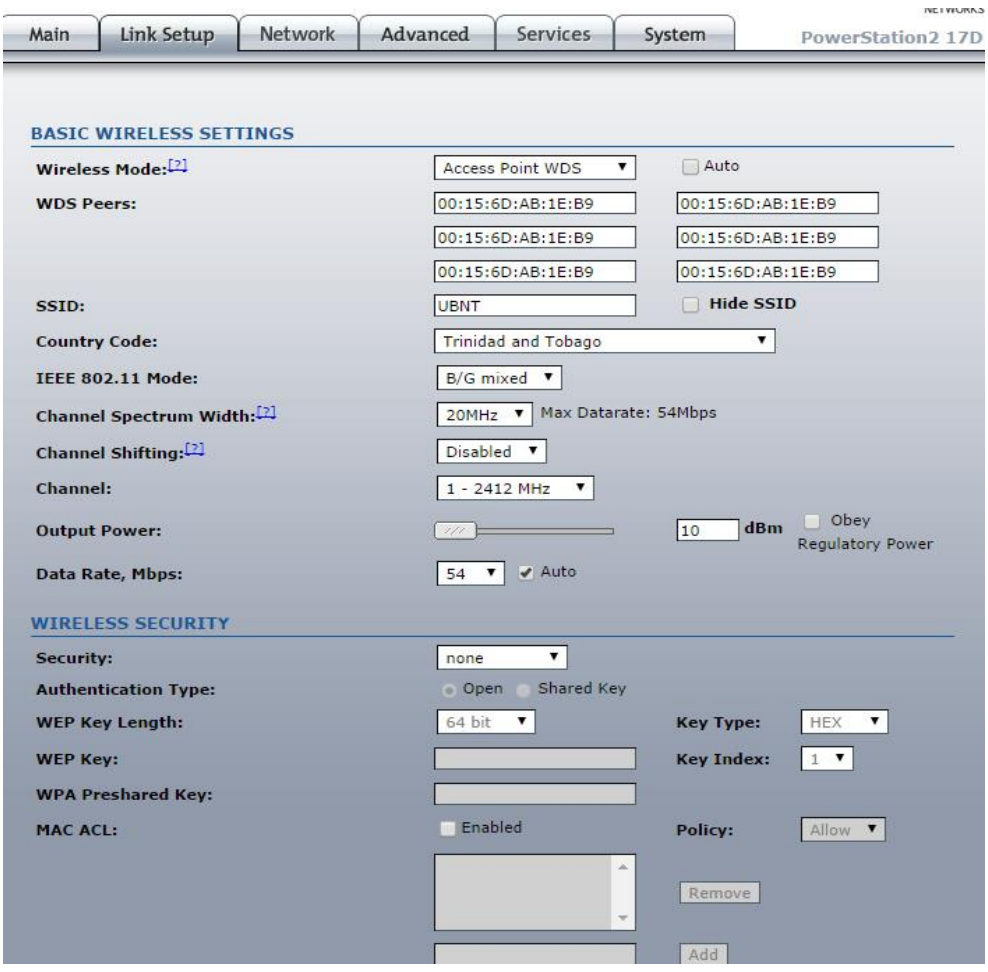


Figura B.1 Configuración del enlace para el AP.

En la pestaña Network, se configura el modo de red en Network Mode con Bridge, además se ingresa la dirección IP, la máscara de red, una Gateway y en los de DNS se coloca 8.8.8.8 y 8.8.4.4. En la figura B.2 se muestra la configuración echa para el AP.

Figura B.2 Configuración de la red para el AP.

B.2. CONFIGURACIÓN DE EQUIPO 2 COMO ESTACIÓN WDS

El PS2 ubicado en la Universidad del Cauca, se configura como Estación WDS, de igual manera que el PS2 anterior se necesita conectar un PC e ingresar a modo configuración con la dirección IP por defecto. A diferencia de la configuración del AP en la Estación se ingresa la dirección MAC del AP WDS. La configuración de este PS2 como Estación WDS se ilustra en la figura B.3.

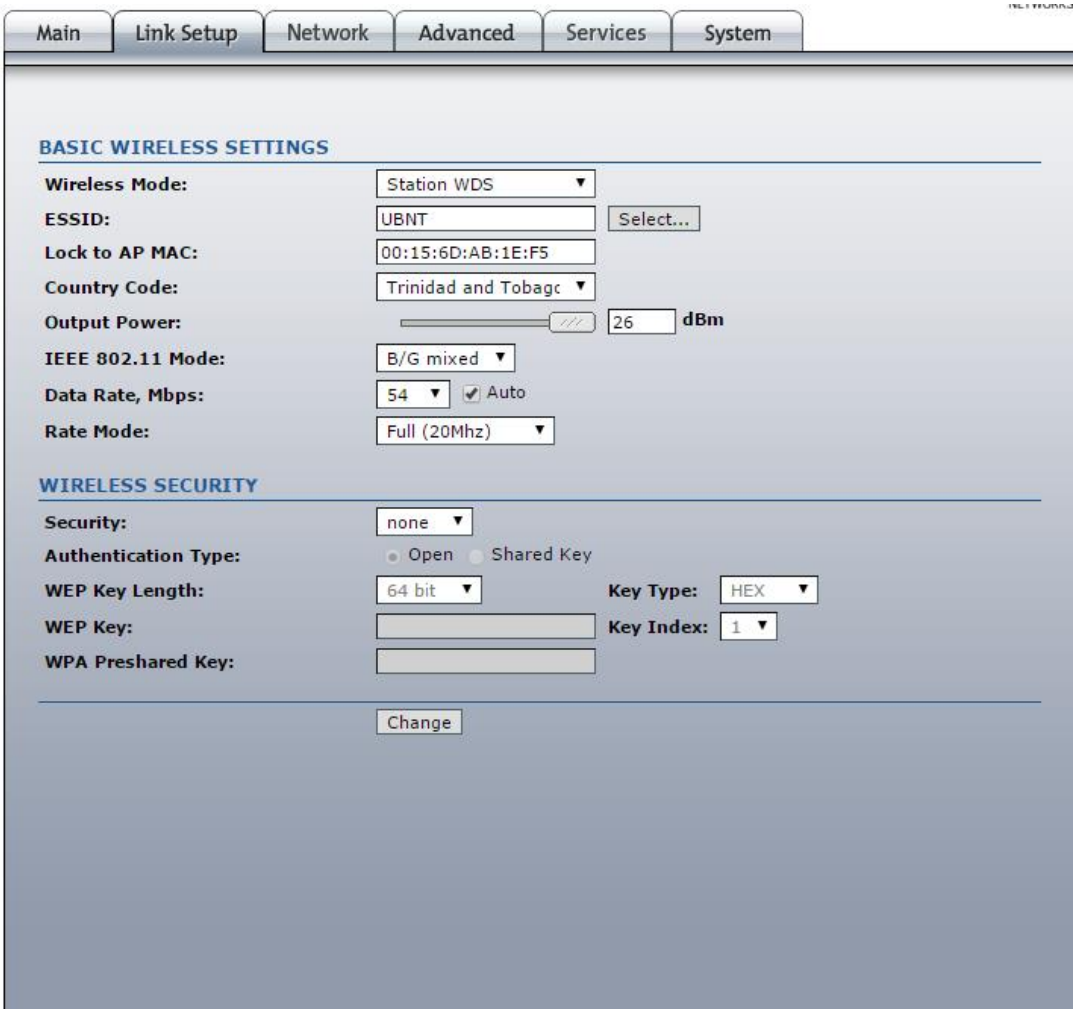


Figura B.3 Configuración del enlace para la Estación.

La configuración de la red es similar a la del AP, solo cambia la dirección IP. En la figura B.4 se observa la configuración.

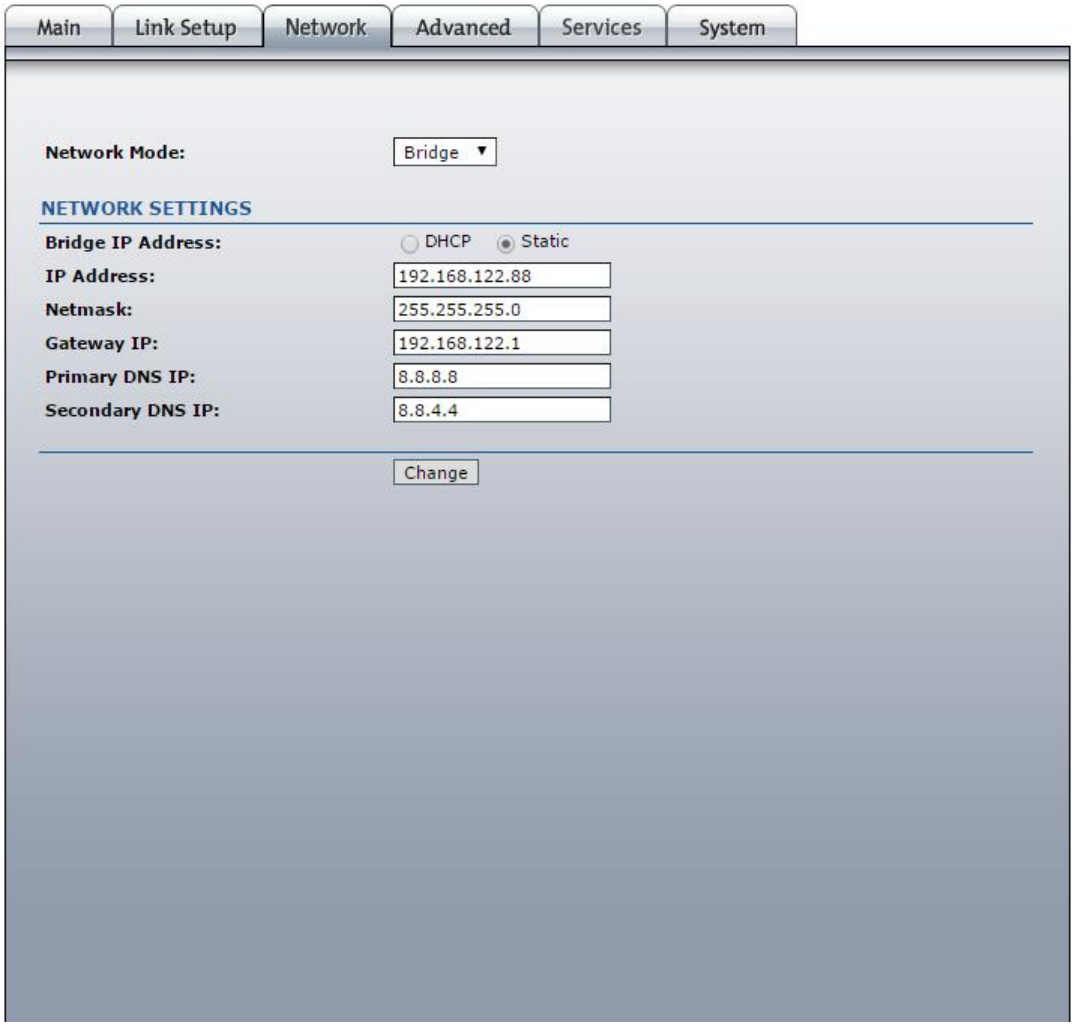


Figura B.4 Configuración de la red para la Estación.

ANEXO C

C.1. APP DE JAVA

El App del SGCR se desarrolla en Netbeans utilizando librerías que permiten comunicación por socket y librerías para gestionar bases de datos. Debido a la cantidad de código fuente, solo se describirá el código más importante desarrollado.

C.1.1 Conexión a Base de Datos

Para realizar la gestión de la base de datos se crea una clase conexionDB la cual se conecta a la base de datos unicaucaEstereo con el usuario root y contraseña root, creada con MySQL. A continuación en la figura C.1 se muestra la configuración de conexión.

```
19  public class conexionDB {
20      persona pers=new persona();
21     Connection con=null;
22     ResultSet rs;
23     public Connection conexion(){
24     try{
25     Class.forName("com.mysql.jdbc.Driver");
26     con= DriverManager.getConnection("jdbc:mysql://localhost/unicaucaEstereo","root","root");
27     System.out.println("conexion establecida.");
28
29     }catch(ClassNotFoundException | SQLException e)
30     {
31     System.out.println("error conexion establecida.");
32     JOptionPane.showMessageDialog(null, "error conexion establecida"+ e);
33     }
34      return con;
35     }
36     public Connection getConnection(){
37        return con;
38     }
```

Figura C.1 Conexión base de datos.

Para validar los usuarios que se registran en la base de datos se utiliza una función llamada “ingresar” que se encarga de comparar los datos ingresados desde la App con la base de datos. En la figura C.2 se ilustra el código encargado de realizar la petición de comparación a MySQL.

```
public boolean ingresar(String user, String contra)
 {
     Object[][] res = this.select("usuario", " nombreUsuario , contraseña ", " nombreUsuario='"+user+"' AND contraseña='"+contra+"' ")
     if( res.length > 0)
     {
         pers.setNombre( res[0][0].toString() );
         pers.setContraseña( res[0][1].toString() );

         return true;
     }
     else
         return false;
 }
```

Figura C.2 Código de comparación para ingreso.

Para actualizar la base de datos se utiliza la función “actualizar” la cual compara un identificador dado con el registro de la base de datos y actualiza el nuevo valor. A continuación en la figura C.3 se muestra el código de la función.

```
public void actualizarR(String valor,int a){
    Statement st=null;


     try {
              st=(Statement)con.createStatement();
              st.executeUpdate("UPDATE parametrostransmisor SET valor='"+valor+"'WHERE idParametro="+a+";");


     } catch (SQLException ex) {
         Logger.getLogger(conexionDB.class.getName()).log(Level.SEVERE, null, ex);
     }


 }
```

Figura C.3 Función de actualizar base de datos.

Por último con la función “leer” se obtiene la información actualizada de la base de datos para procesarla. En la figura C.4 se ilustra la función “leer”.

```
   public String leer(int a){
   Statement st=null;
   String Dato="";
   try {
              st=(Statement)con.createStatement();
              rs= st.executeQuery("select valor from parametrostransmisor WHERE idParametro="+a+";");
              if(rs.next()){
              Dato=rs.getString(1);
                      }
       } catch (SQLException ex) {
           Logger.getLogger(conexionDB.class.getName()).log(Level.SEVERE, null, ex);
       }
       return Dato;
}
```

Figura C.4 Función leer.

C.1.2. Conexión a Servidor por Socket

Para realizar la conexión al servidor de Arduino el App de java utiliza los socket, para lo cual se desarrollan funciones en la clase UnicaucaEstereo que permiten la conexión y desconexión. A continuación en las figuras C.5 y C.6 se muestran las funciones de conexión y desconexión.

```
42      void conectar() {
43
44          try {
45              socket = new Socket("192.168.122.50", 23);
46              out = new PrintWriter(socket.getOutputStream(), true);
47              in = new BufferedReader(new InputStreamReader(
48                      socket.getInputStream()));
49
50              timer = new Timer(1900, new ActionListener() {
51
52                  @Override
                    public void actionPerformed(ActionEvent e) {
54                      throw new UnsupportedOperationException("Not supported yet.");
55                  }
56
57              });
58
59              timer.start();
60
61          } catch (UnknownHostException e) {
62              System.err.println("No se ha podido conectar con la ip.");
63              System.exit(1);
64          } catch (IOException e) {
65              System.err.println("Couldn't get I/O for "
66                      + "the connection");
67              System.exit(1);
68          }
```

Figura C.5 Función de conexión al socket.

```
3       void desconectar() {
4           try {
5               timer.stop();
6               in.close();
7               out.close();
8               socket.close();
9               //setGUIOff();
0
1           } catch (IOException ex) {
2               Logger.getLogger(UnicaucaEstereo.class.getName()).log(Level.SEVERE, null, ex);
3           }
4
5       }
```

Figura C.6 Función de desconexión al socket.

C.1.3. *Login*

La clase *Login* se crea con un objeto de base de datos por medio del cual hace comparación de usuarios registrados para permitir o negar en acceso a la App. En la siguiente figura se muestra el constructor de Login.

```
21          */
22      public Login() {
23          initComponents();
24          co=new conexionDB();
25          co.conexion();
26      }
27
```

Figura C.7 Constructo de la case Login.

Luego de construida la clase, se capturan los datos ingresados por el usuario y se hace uso de la función “ingresar” de la clase conexionDB para compararlos en la base de datos, si están registrados se ingresa al menú, de lo contrario, se ilustra un mensaje de error. En la figura C.8 se muestra el código de validación.

```
    private void jbLoginActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String usuario,contraseña;
        usuario=jtUsuario.getText();
        contraseña=jpContra.getText();
        if (co.ingresar(usuario, contraseña))
        {
         Menu MiMenu=new Menu();
         MiMenu.setVisible(true);
         this.setVisible(false);
        }else{
         jTextArea1.setText("Contraseña o usuario incorrectos\n pruebe de nuevo");
         jtUsuario.setText("");
         jpContra.setText("");
        }
    }
```

Figura C.8 Validación de usuarios.

C.1.4 Menú

En menú primero se debe establecer la conexión al servidor por medio del socket, por lo que se crea el objeto de tipo UnicaucaEstereo. En la figura C.9 se muestra la acción de conectar de menú, la desconexión se hace al salir del menú.

```
266     private void jbConectarActionPerformed(java.awt.event.ActionEvent evt) {
267         // TODO add your handling code here:
268         ConexionArduino=new UnicaucaEstereo();
269
270         hilo.start();
271         if(hilo!=null){
272         ConexionArduino.conectar();
273         ConexionArduino.out.println("21i00");
274         setGUIOn();
275         jlConexion.setText("Conectado al Transmisor");
276         }else{
277         jlConexion.setText("no arranco el hilo");
278         }
279     }
```

Figura C.9 Conexión al servidor.

Luego de establecer la conexión con el servidor, se elige una opción en el menú, las cuales son Conectar, Estado, Alarmas, Control y Salir, al presionar cualquier botón se hace visible la interfaz escogida o se sale del sistema si se presiona el botón Salir. Para guardar los valores que llegan al App se utilizan las etiquetas para ubicar los datos en la base de datos, se utiliza el siguiente fragmento de código ilustrado en las figuras C.10 y C.11.


```
public void run() {
    if(hilo!=null){
     while(true){
     String majIn,tipoParametro="",sacarDato="",sql="";
     int tamanoString;
     char tamanoCab;


     try {
          Thread.sleep(100);
        majIn = ConexionArduino.in.readLine();
        tamanoString= majIn.length();
        tamanoCab= majIn.charAt(tamanoString-1);


       if (tamanoCab=='4'){
        for (int i=0;i<=3;i++){
         tipoParametro+=majIn.charAt(i);
        }
        sacarDato=sacar(majIn,4);

       switch (tipoParametro){
          case "$COR":
              conn.actualizarR(sacarDato,1);
               JOptionPane.showMessageDialog(null, sacarDato);

              break;
          case "$VOL":
              conn.actualizarR(sacarDato,2);
                JOptionPane.showMessageDialog(null, sacarDato);
              break;
          case "$RUE":
              conn.actualizarR(sacarDato,3);
               JOptionPane.showMessageDialog(null, sacarDato);
              break;
          case "$EXI":
              conn.actualizarR(sacarDato,4);
               JOptionPane.showMessageDialog(null, sacarDato);
              break;
          case "$PCR":
              conn.actualizarR(sacarDato,5);
                 JOptionPane.showMessageDialog(null, sacarDato);
              break;
          case "$PCE":
              conn.actualizarR(sacarDato,6);

               JOptionPane.showMessageDialog(null, sacarDato);
              break;
          case "$AI1":
               conn.actualizarR(sacarDato,7);
               JOptionPane.showMessageDialog(null, sacarDato);
              break;
          case "$AI2":
              conn.actualizarR(sacarDato,8);

               JOptionPane.showMessageDialog(null, sacarDato);
```

Figura C.10 Verificación y envío de etiquetas.

Figura C.11 Verificación y envío de etiquetas.

ANEXO D

D.1. MANUAL DE USUARIO

Figura D.1 Interfaz de Inicio de sesión.

1. Usuario	En este campo el usuario del sistema debe ingresar un nombre de usuario.
2. Contraseña	El usuario del sistema debe ingresar una contraseña.
3. Campo texto	En este espacio el sistema, de ser necesario, mostrará un texto diciéndole al usuario que los datos ingresados en los campos 1 y/o 2 son erróneos.
4. Salir	El botón Salir permite al usuario salir del sistema.
5. Entrar	El botón Entrar se presiona una vez se haya ingresado los datos de los campos 1 y 2. Si los datos ingresados son correctos, el sistema dará acceso al usuario al menú del sistema, de lo contrario se activará el campo texto.

Tabla D.1 Componentes interfaz de inicio.

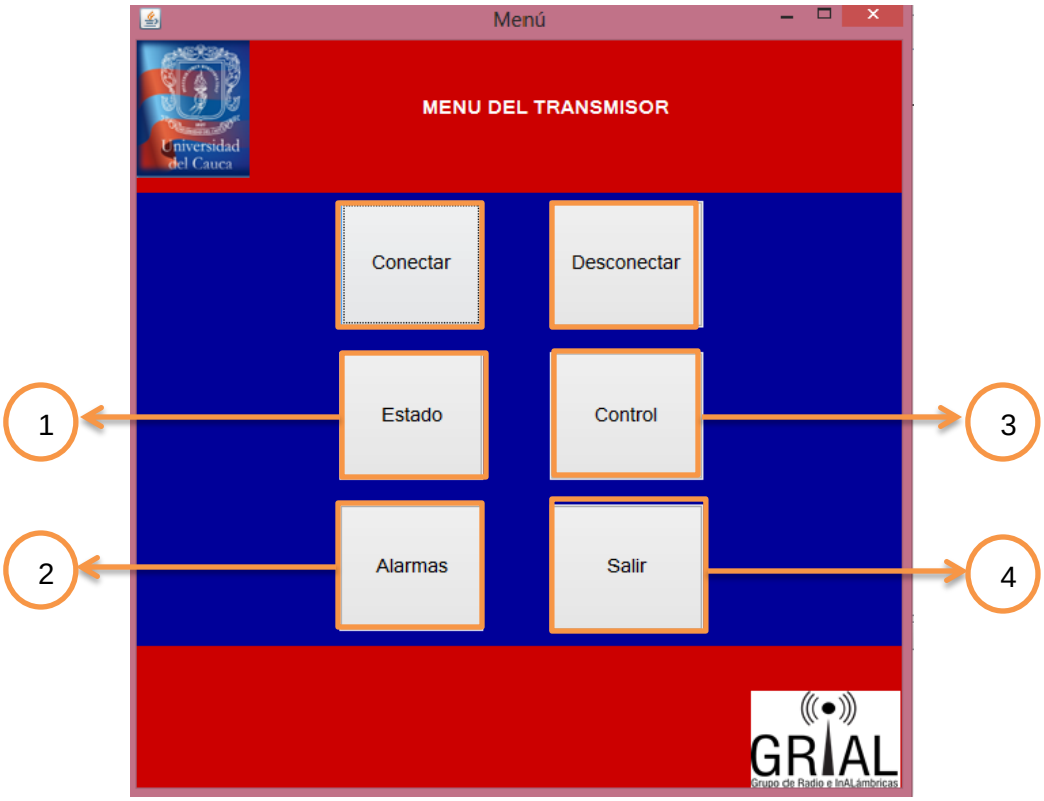


Figura D.2 Interfaz de Menú.

1. Estado	El botón Estado abre una interfaz donde se puede observar o modificar el estado del transmisor.
2. Alarmas	Este botón abre la interfaz para observar cinco alarmas que generó el transmisor.
3. Control	Control abre una interfaz que permite escoger tanto el modo de potencia entregada como el o los valores de potencia que entregará el transmisor las 24 horas.
4. Salir	El botón Salir permite al usuario salir del sistema.

Tabla D.2 Componentes interfaz Menú.

Figura D.3 Interfaz Estado.

1. Estado	La pestaña Estado abre una interfaz donde se puede observar o modificar el estado del transmisor.
2. Temperatura	La pestaña Temperatura abre una interfaz en la cual se observan las temperaturas más relevantes del transmisor.
3. On Air	Al accionar este botón se enciende el transmisor. Si el transmisor ya está encendido, el botón estará opaco.
4. St-By	St-By hace que el transmisor quede en estado de espera o reposo. Si el transmisor ya está en Stand by, el botón estará opaco.
5. Apagar	El botón Apagar apaga el transmisor. Si el este ya está apagado, el botón estará opaco y solo se podrá encender el transmisor.
6. Menú	Con este botón se regresa a la interfaz del menú del transmisor.

Tabla D.3 Componentes interfaz Estado.

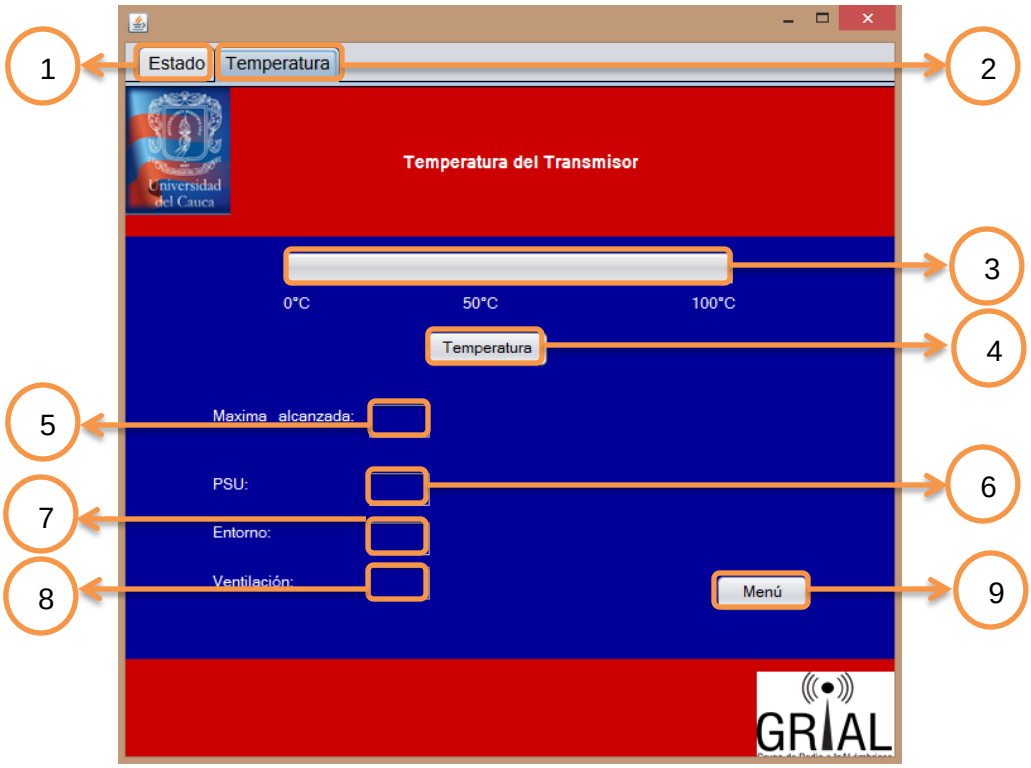


Figura D.4 Interfaz Temperatura.

1. Estado	La pestaña Estado abre una interfaz donde se puede observar o modificar el estado del transmisor.
2. Temperatura	La pestaña Temperatura abre una interfaz en la cual se observan las temperaturas más relevantes del transmisor.
3. Barra de progreso temperatura	En esta barra se ilustra la temperatura general del transmisor.
4. Botón Temperatura	Al accionar este botón, la barra de progreso temperatura (3) se empieza a llenar hasta llegar al valor de temperatura del transmisor.
5. Máxima alcanzada	Es la temperatura más alta que registra o registró el transmisor.
6. PSU	Es la temperatura de la fuente de alimentación.
7. Entorno	Temperatura del entorno.
8. Ventilación	Temperatura que proporciona la ventilación del transmisor.
9. Menú	Con este botón se regresa a la interfaz del menú del transmisor.

Tabla D.4 Componentes interfaz Temperatura.

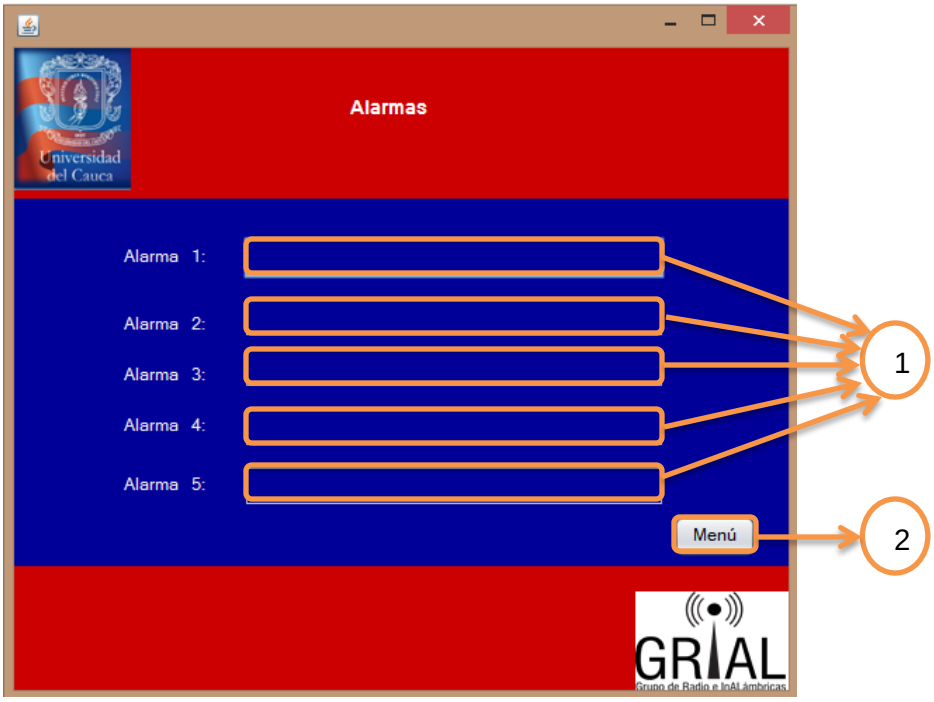


Figura D.5 Interfaz Alarmas.

1. Campos Alarmas	En estos campos se registran las cinco alarmas que el transmisor ha generado recientemente.
2. Menú	Con este botón se regresa a la interfaz del menú del transmisor.

Tabla D.5 Componentes interfaz Alarmas.

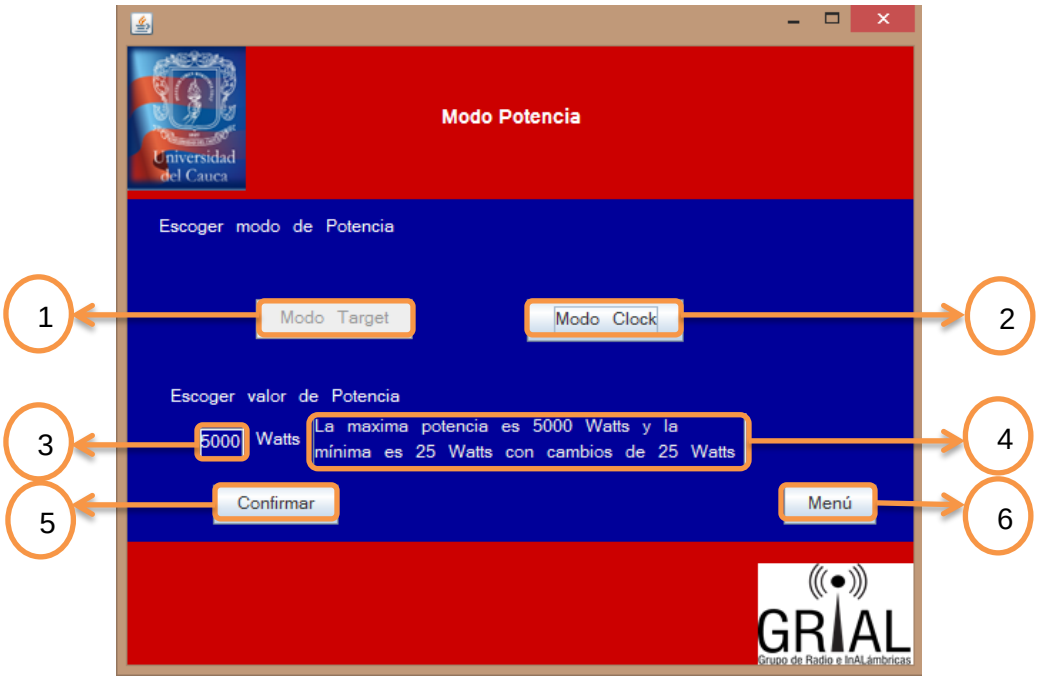


Figura D.6 Interfaz Control.

1. Modo Target	Al presionar este botón, se hacen visibles 3,4 y 5. Lo anterior para modificar la potencia entregada por el transmisor en el modo target, que significa que el transmisor entregará el mismo valor de potencia las 24 horas.
2. Modo Clock	Este botón abre la interfaz para modificar la potencia entregada en modo clock.
3. Campo Edición Potencia	En este campo se modifica el valor de la potencia entregada. El valor asignado debe estar entre 25 W y 5000 W con saltos de 25 W.
4. Campo de advertencia	En este campo se advierte que el valor de potencia entregada debe estar en un rango permitido por el transmisor.
5. Confirmar	Al accionar este botón, se establece en el transmisor el valor de potencia entregada que se asignó en 3.
6. Menú	Con este botón se regresa a la interfaz del menú del transmisor.

Tabla D.6 Componentes interfaz Control.

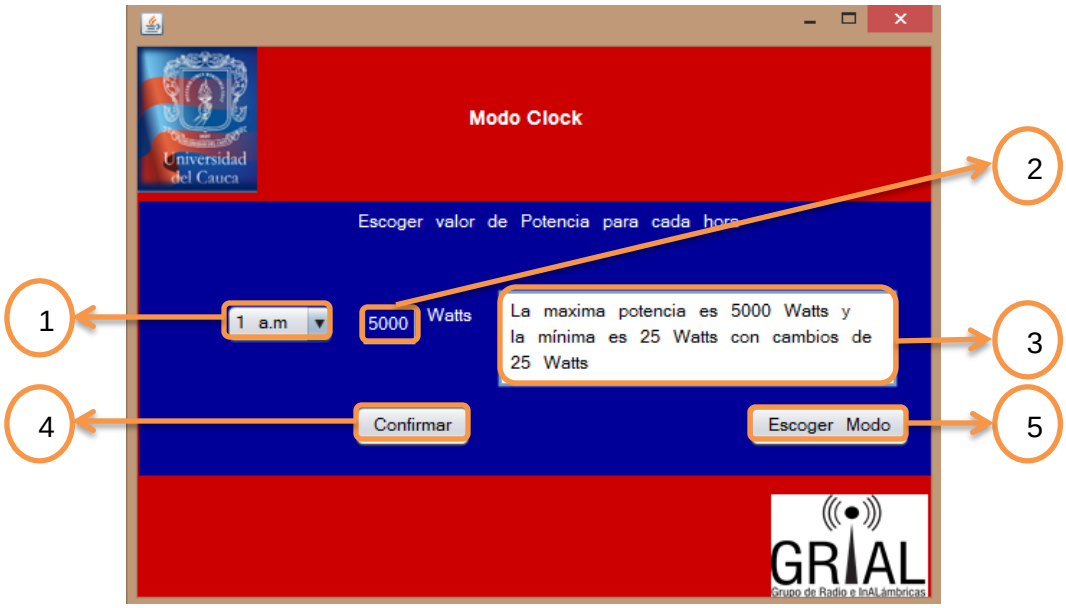


Figura D.7 Interfaz Modo Clock.

1. Hora del día	En este campo se escoge la hora del día a la cual se le va a asignar el valor de potencia entregada.
2. Campo Edición Potencia	En este campo se modifica el valor de la potencia entregada. El valor asignado debe estar entre 25 W y 5000 W con saltos de 25 W.
3. Campo de advertencia	En este campo se advierte que el valor de potencia entregada debe estar en un rango permitido por el transmisor.
4. Confirmar	Al accionar este botón, se establece en el transmisor el valor de potencia entregada que se asignó en 3.
5. Escoger Modo	Con este botón se regresa a la interfaz de Control del transmisor.

Tabla D.7 Componentes interfaz Modo Clock.

ANEXO E: MANUAL TÉCNICO

E.1. INTRODUCCIÓN

El personal técnico encargado de realizar gestión al transmisor de la emisora Unicauca Estéreo, debe realizar evaluaciones periódicas del estado del equipo utilizando la información que éste proporciona. Para facilitar dicho proceso es necesario implementar un sistema de gestión y control remoto que permita monitorear y controlar a través de una aplicación software parámetros del transmisor como los voltajes, corrientes, temperaturas, potencias y estados.Dicho sistema se compone de una tarjeta Arduino que funciona como servidor y obtiene la información relevante del transmisor, un enlace inalámbrico que es el medio de comunicación, y una aplicación software. A continuación se presenta el manual técnico del sistema en general, empezando con el hardware requerido para montar el servidor, la forma de implementar el firmware en él mismo, requerimientos y descripción del software.

E.2. PROPÓSITO

La finalidad de este documento es mostrar todas las características, funcionalidades y alcances del Sistema de Gestión y Control Remoto para la emisora Unicauca Estéreo. Este documento va dirigido al personal técnico encargado de gestionar dicho sistema.

E.3. ALCANCE

El sistema propuesto permitirá realizar el monitoreo en tiempo real de parámetros del transmisor de FM de la emisora Unicauca Estéreo como temperatura, potencia, alarmas, corrientes, voltajes y estados. Además, permitirá realizar control de parámetros como la potencia entregada y el estado de funcionamiento.

E.4. DESCRIPCIÓN GENERAL

La funcionalidad del producto consiste en realizar peticiones de información al transmisor de FM de Unicauca Estéreo, con el fin de obtener parámetros relevantes, como los antes expuestos, para mostrarlos en una interfaz gráfica. También permite enviar comandos de control al transmisor para modificar los estados de funcionamiento y modificar la potencia entregada.

E.5. CARACTERÍSTICA DE USUARIO

Tipo de usuario	Administrador
Formación	Administrador/Técnico
Habilidades	Conocimientos avanzados en equipo de cómputo y mantenimiento del transmisor FM
Actividades	Realizar el monitoreo y el control de los parámetros del transmisor

E.6. RESTRICCIONES

En este ítem se mostrará las restricciones que se deben tener en cuenta para la implementación del sistema, para lo cual se dividen en restricciones de aplicación software en java, de servidor en tarjeta Arduino y del enlace inalámbrico. Además, para cada restricción anterior se generan restricciones de hardware, software y conexión.

Aplicación Java

Hardware:

1. Computador personal.
2. Sistema:
   Cualquier SO con máquina virtual de java.
3. Intel Pentium @ 2Ghz, mayor o similar.

Software:

1. Servidor MySQL.
2. JDK java.

Conexión:

1. Puerto Ethernet 10/100 Mb.

Servidor

Hardware:

1. Tarjeta Arduino Ethernet.
2. Conector DB9 de Rs232.
3. Programador USB de Arduino.

Software:

1. IDE de Arduino.

Conexión:

1. Puerto de comunicación serie.
2. Puerto Ethernet 10/100 Mb.
3. Puerto USB de programación.

Enlace Inalámbrico

Hardware:

1. Equipos Power Station 2 de Ubiquiti.
2. Cable UTP para exteriores.
3. Conectores PoE.

Software:

1. Firmware Air OS.

Conexión:

1. Puerto Ethernet 10/100 Mb.

E.7. SUPOSICIONES Y DEPENDENCIAS

El sistema de gestión y control remoto funciona en tres escenarios, la conexión al transmisor, el enlace inalámbrico y la aplicación de java. Este interactúa con su entorno a través de interfaces, para comunicarse con el transmisor a través de comunicación serie y con el cliente o administrador a través de la interfaz gráfica.

Se requiere que el administrador del sistema tenga un computador personal con el JDK de java y el servidor de bases de datos MySQL 6.2. El administrador debe tener conocimiento de los parámetros que entrega el transmisor y la forma de modificarlos e interpretarlos.

E.8. EVOLUCIÓN PREVISIBLE DEL SISTEMA

Se plantea que el sistema necesite mejoras futuras orientadas a optimizar la interacción con el administrador con respecto a los siguientes puntos:

Incrementar los parámetros gestionados

El administrador podrá monitorear y controlar más parámetros además de los ya establecidos.

Mejorar la conexión entre servidor y aplicación java

Mejorar la conexión por socket entre el servidor Arduino y la aplicación de java, teniendo en cuenta las excepciones que no se tuvieron en cuenta en el actual desarrollo.

E.9. REQUISITOS COMUNES DE INTERFACES

En este documento se muestran los requisitos técnicos para la implementación del sistema.

E.10. INTERFACES HARDWARE

Para trabajar con el sistema se necesita implementar el servidor de Arduino, conectado al transmisor FM por puerto serie RS232, además debe conectarse a través de la interfaz Ethernet con los equipos Ubiquiti utilizados para el enlace inalámbrico.

Para conectarse al servidor como cliente administrador, se debe montar la aplicación java en un computador con interfaz Ethernet para la conexión con los equipos Ubiquiti.

Los equipos Ubiquiti se conectan tanto al servidor Arduino como a la aplicación java utilizando los puertos PoE y cable UTP para exteriores.

E.11. INTERFACES DE SOFTWARE

Para satisfacer las necesidades de la emisora Unicauca Estéreo se tiene en cuenta los siguientes puntos:

- El sistema se conecta al servidor MySQL que contiene la base de datos.
- El sistema se conecta a través del uso de sockets Cliente servidor.
- Se implementa la máquina virtual de java para independencia del S.O.
- Se utiliza un registro de inicio de sesión.
- Se utiliza un menú de navegación.

E.12. INTERFAZ DE COMUNICACIÓN

Las interfaces de comunicación serán:

- Ethernet.
- Serie Rs232.
- Interfaz WiFi.

La conexión serie se hace a través de un conector DB9 que se acopla a los pines de comunicación serie de la tarjeta Arduino, las interfaces Ethernet e inalámbrica estan integradas en los equipos Ubiquiti y en la tarjeta Arduino.

E.13. REQUISITOS FUNCIONALES

Conexión a base de datos.

El sistema verifica el acceso a la base de datos cuando esta está disponible.

Inicio de sesión.

Luego de conectarse a la base de datos se debe identificar el cliente administrador por medio de un nombre de usuario y una contraseña.

Conexión a servidor.

La conexión al servidor se debe realizar con sockets a través de un cliente creado en la aplicación java.

Menú de navegación

Al momento de realizar las conexiones, se permite seleccionar parámetros para monitoreo y control. Se muestran submenús de estado, control, alarmas, temperaturas y modo de entrega de potencia.

Monitoreo de parámetros

Los datos se obtienen desde el transmisor FM a través del servidor Arduino para ser enviados a la aplicación de java. Está actualiza los datos que llegan en la base de datos del sistema y desde la base de datos captura los valores y los muestra en las interfaces gráficas.

Control de parámetros.

Cada opción de controlar parámetros envía, al ser seleccionada, comandos en formato ANSI VT100 para modificar el estado de los mismos.

E.14. DIAGRAMA GENERAL

El diagrama general se presenta a continuación, dividido de dos partes: el modelo de entidad de relación y el diagrama de contexto. El primero es una herramienta para el modelado de datos de un sistema de información, que en este caso se incluye al sistema general de monitoreo y control, y el segundo que delimita el sistema en general, permitiéndolo distinguirse del entorno con el que se relaciona.

Modelo de entidad-relación.

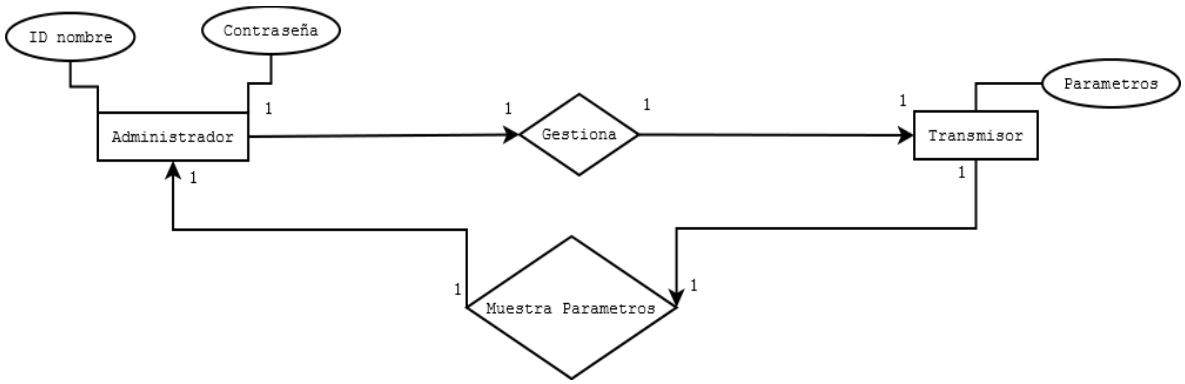
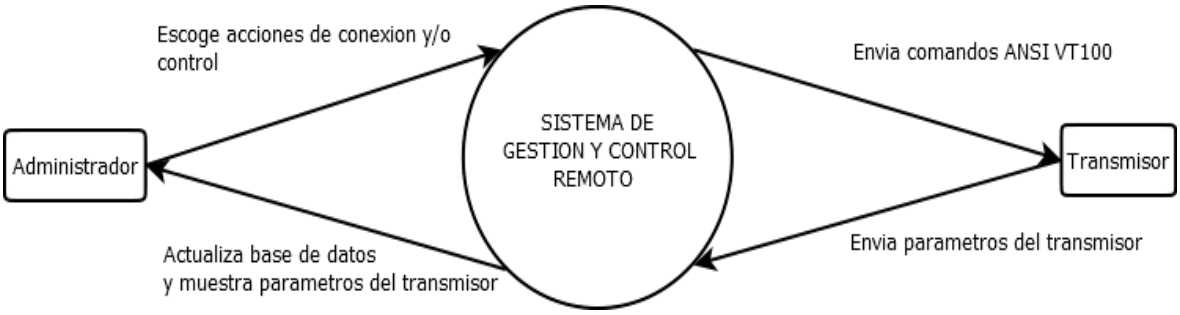


Diagrama de contexto



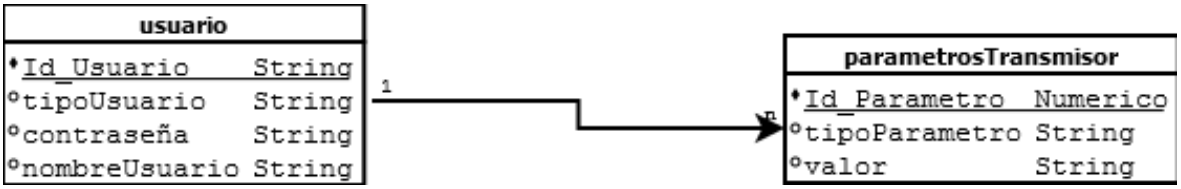
E.15. DICCIONARIO DE DATOS

A continuación se muestra un conjunto de metadatos que contiene las características de los datos que se van a utilizar en el sistema.

Nombre de la tabla	Nombre del atributo	Contenido	Tipo	Llave
usuario	Id_nombreusuario	Identificador del usuario	String	primaria
	TipoUsuario	Tipo de usuario	String	
	Contraseña	Contraseña de ingreso	String	
parametrosTransmisor	Id_Parametro	Identificador del parámetro	Numérico	primaria
	TipoParametro	Nombre del parámetro	String	
	Valor	Valor actual del parámetro	String	

E.16. DIAGRAMA RELAICIONAL

A continuación se muestra el diagrama relacional de la base de datos del sistema con el objetivo de mostrar un modelo de datos basado en la lógica de predicados.



E.17. DEFINICIÓN DE VARIABLES DE AMBIENTE Y LIBRERÍAS

Librerías utilizadas en el IDE de Arduino para el desarrollo del Firmware.

Nombre de acceso	Descripción	Variables	Definición de variables
SPI.h	Esta librería permite la comunicación de la tarjeta Arduino con otros dispositivos a través del protocolo de comunicación serie SPI.	• Serial.print() • Serial.available() • Serial.read()	• Se utiliza para enviar los datos de forma serial. • Guarda un dato tipo int con el que se comprueba si hay datos para leer en el puerto serie. • Guarda datos en buffer para ser leídos.
Ethernet.h	Esta librería proporciona elementos necesarios para la comunicación por el puerto Ethernet.	• Mac. • Ip. • Server.	• Guarda la dirección Mac de la tarjeta Arduino. • Guarda la dirección IP proporcionada a la tarjeta Arduino. • Guarda el puerto de nivel 4 nivel de transporte.

Librerías utilizadas en el IDE de NetBeans para el desarrollo de la app de java.

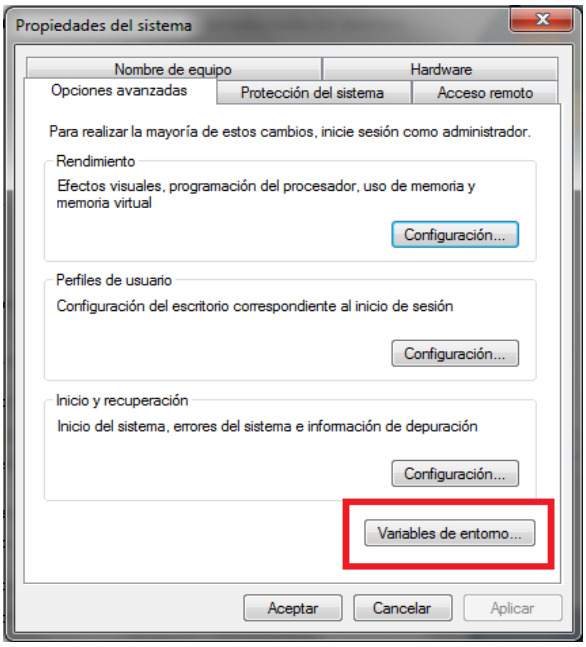
Nombre de acceso	Descripción	Variables	Definición de variables
Java.sql.*	Estas librerías permiten realizar la conexión a una base de datos relacional.	• Class.forName • DriverManager.getConnection	• Determina el driver con el que se realiza la conexión a una base de datos MySQL. • -Ubica la base de datos creada y el login y contraseña.
javax.swing.JOptionPane	Esta librería permite mostrar ventanas con alternativas.	JOptionPane.showMessageDialog()	Crea la ventana con el parámetro incluido.

java.util.logging.Level java.util.logging.Logger	Estas librerías permiten el control de excepciones.	SQLException	Contra excepciones generadas al trabajar con bases de datos.
java.awt.event.ActionEvent	Esta librería permite capturar eventos y realizar acciones.		
java.io.BufferedReader	Esta librería permite guardar datos en buffer para luego entregar la palabra completa.	BufferedReader	Guarda la información que ingresa por una interface en un buffer.
java.io.PrintWriter	Esta librería permite enviar información por determinadas interfaces.	PrintWriter	Permite el envió de datos por interfaces como la Ethernet.
java.net.Socket	Esta librería proporciona elementos con los que se crea un socket de comunicación.	Socket	Con dos parámetros (dirección IP y puerto de capa 4) establece un cliente socket.
java.sql.PreparedStatement	Esta librería permite enviar estamentos o líneas SQL a la base de datos desde la conexión echa por la App de java.		Las variables de esta librería permiten crear los estamentos SQL y enviarlos al gestor de base de datos para ejecutarlos.

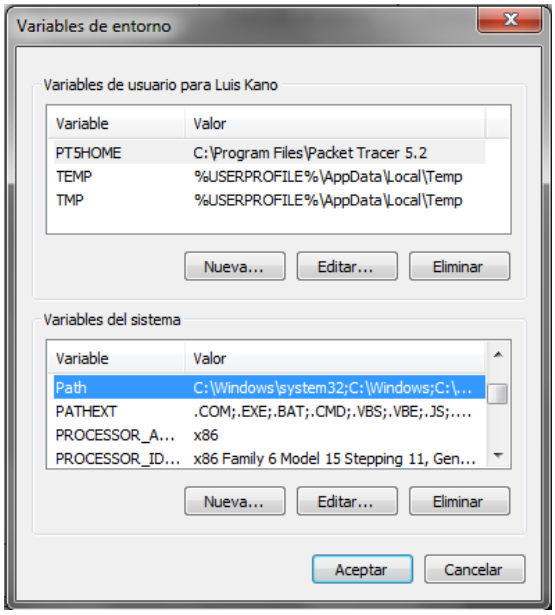
E.18. PROGRAMAS ESPECIALES Y DE AMBIENTE

Los IDE de Arduino y NetBeans en los cuales se desarrolla la aplicación java y el firmware de Arduino. Además, para el desarrollo de la aplicación es necesario montar en el computador residente el JDK de java y configurarlo así como el servidor de MySQL.

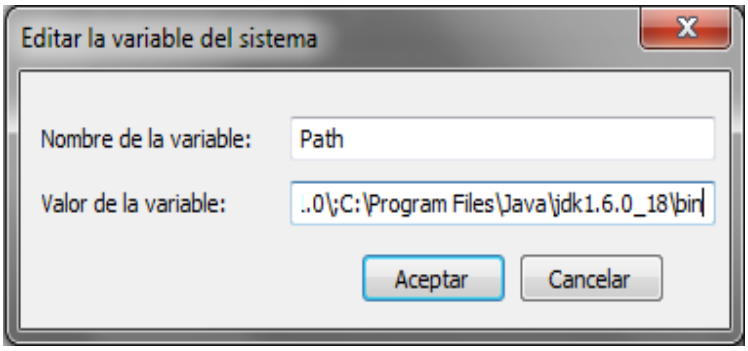
Para trabajar con java se instala el JDK que se puede descargar de internet a través de la página de ORACLE. Una vez instalado el JDK se debe configurar las variables de entrono para permitir la compilación desde Windows. Para Windows 7 se ingresa a Mi PC seguido de la opción propiedades, opciones avanzadas, luego se ingresa a la opción variables de entorno como se muestra en la siguiente figura.



En la ventana de variables de entorno mostrado en la siguiente figura, en el ítem variables del sistema se escoge la variable Path.

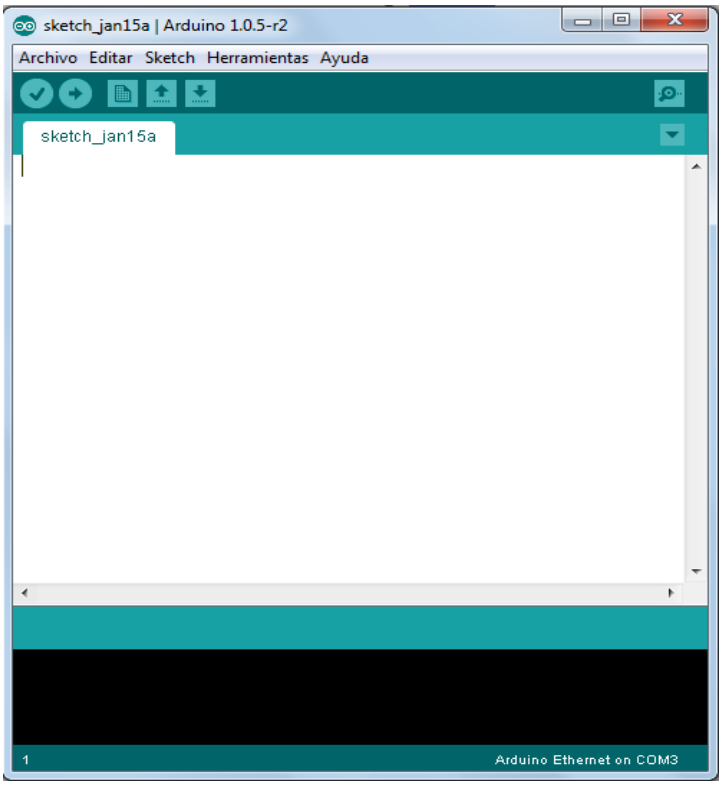


Cuando se escoge la opción editar y en la ventana que aparece se ingresa la ruta del instalador del JDK, esto se hace sin borrar los otros datos, solo se coloca un punto y coma que se para la instrucción. En la siguiente figura se muestra la configuración de la variable Path.



Luego de instalar el JDK de java se debe instalar el servidor de MySQL 6.2, para esto se ingresa a la página oficial de MySQL y se descarga el instalador.

Además de los anteriores programas, se describe a continuación el IDE de Arduino con el que se desarrolla el firmware de la tarjeta que sirve de servidor.



El IDE permite desarrollar tareas como verificar, compilar y cargar el código, crear, abrir y guardar proyectos o sketch. Para desarrollar un sketch se debe configurar en la pestaña de herramientas el puerto serial de comunicación que asigna Windows al IDE, además se debe escoger el tipo de tarjeta con la que se trabajará, en este caso Ethernet. En la pestaña sketch se escogen las librerías necesarias y se hace compilación y verificación de código.

Cuando se crea el código fuente y no se encuentran errores en la compilación, este se carga a la tarjeta Arduino. Una vez cargado el código, este se ejecuta en un bucle infinito.

El código se divide en dos procesos, el primero en ejecutarse es el *set up* que se utiliza para inicialización y configuración de variables, el segundo es el *loop* que se ejecuta después del *set up* de forma infinita. En la siguiente figura se observa el esquema básico de un código en el IDE de Arduino.