Administrator Application Code

1. Login Form Code

```
//LogIn Form
//by Laura M. Cruz
//802-03-1797
//Icom 5047 Sec 031

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Login : Form
    {
        private System.Data.Odbc.OdbcConnection OdbcCon;
        private System.Data.Odbc.OdbcCommand OdbcCom;
        private System.Data.Odbc.OdbcDataReader OdbcDR;
        private string ConStr = "";
        private string query = "" ;
        private string precinct = "" ;
        private string unit = "" ;
        private string firstname = "" ;
        private string lastname1 = "" ;
        private string lastname2 = "" ;
        private string psswd = "" ;
        private string OfficialID = "" ;

        public Login()
        {
            InitializeComponent();
            // Build the connection string
            ConStr = "DRIVER={MySQL ODBC 5.1
Driver};SERVER=136.145.56.170;PORT=3306;DATABASE=evote;UID=evote;PWD=evote
;OPTION=3";
            //Create connection
            OdbcCon = new System.Data.Odbc.OdbcConnection(ConStr);

            //When form is initialized, fill precinct dropdown box
(comboBox1) with precinct list
            try
            {
                if (OdbcCon.State == ConnectionState.Closed)
                {
                    OdbcCon.Open();
                    query = "SELECT DISTINCT Precinct FROM evote.official
o;";
```

```
                    //Create ODBC Command with necessary query and odbc
connection
                    OdbcCom = new System.Data.Odbc.OdbcCommand(query,
OdbcCon);
                    //ExecuteReader is used when query results in more
than one row
                    OdbcDR = OdbcCom.ExecuteReader();
                    //Adds all available precincts to dropdown list
                    while (OdbcDR.Read())
                    {
                        comboBox1.Items.Add(OdbcDR[0]);
                    }
                    //Always close reader when not in use
                    OdbcDR.Close();
                }
            }

            catch (System.Data.Odbc.OdbcException Ex)
            {
                //Thrown when connection to database is not possible
                MessageBox.Show("No se pudo accesar la base de datos. \r\n
Favor verificar que los campos esten completos con la información correcta
y trate nuevamente. \r\n\r\nMas detalles:\r\n" + Ex.Message, "Error
conectando a la base de datos", MessageBoxButtons.OK,
MessageBoxIcon.Error);

            }
        }


        private void comboBox2_SelectedIndexChanged_1(object sender,
EventArgs e)
        {
            //When unit is selected or changed, clear official drop down
(comboBox3)
            comboBox3.Items.Clear();
            //Sets "unit" to selected unit name
            unit = comboBox2.SelectedItem.ToString();
            query = "SELECT FirstName,FirstLastName,SecondLastName FROM
evote.official o WHERE Precinct='" + precinct + "' AND Unit ='" + unit +
"';";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();

            //Adds First Name, First Last Name , and Second Last Name to
official dropdown list
            while (OdbcDR.Read())
            {
                comboBox3.Items.Add(OdbcDR[0] + " " + OdbcDR[1] + " " +
OdbcDR[2]);
                firstname = (string)OdbcDR[0];
                lastname1 = (string)OdbcDR[1];
                lastname2 = (string)OdbcDR[2];
            }

            //Always close reader when it is not in use
            OdbcDR.Close();
```

```
            //Enables official dropdown list after names have been added
            comboBox3.Enabled = true;

        }


        private void comboBox1_SelectedIndexChanged_1(object sender,
EventArgs e)
        {
            //Clear all related dropdown boxes (units and officials)
            comboBox2.Items.Clear();
            comboBox3.Items.Clear();

            //Sets "precinct" to selected precinct name
            precinct = comboBox1.SelectedItem.ToString();

            query = "SELECT DISTINCT Unit FROM evote.official o WHERE
Precinct ='" + precinct + "';";
            //query = "SELECT DISTINCT Unit FROM evote.official o WHERE
Precinct LIKE ? ;";
            OdbcCom.CommandText = query;
            //MessageBox.Show(query);
            //OdbcParam = new System.Data.Odbc.OdbcParameter();
            //OdbcParam.DbType = DbType.String;
            //OdbcParam.Value = precinct;
            //OdbcCom.Parameters.Add(OdbcParam);

            //OdbcCom.ExecuteNonQuery();

            OdbcDR = OdbcCom.ExecuteReader();
            //Adds units in precinct to dropdown box (comboBox2)
            while (OdbcDR.Read())
            {
                comboBox2.Items.Add(OdbcDR[0]);
            }

            //Always close reader when it is not in use
            OdbcDR.Close();

            //Enable unit dropdown when precinct is selected
            comboBox2.Enabled = true;
        }


        //When official (comboBox3) selection is made password textbox is
enabled
        private void comboBox3_SelectedIndexChanged_1(object sender,
EventArgs e)
        {
            textBox1.Enabled = true;
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            String name = comboBox3.Text;
            string[] words = name.Split(new char[] { ' ' });
            firstname = words[0];
```

```
            lastname1 = words[1];
            lastname2 = words[2];

            //stores provided password
            psswd = textBox1.Text;

            //Stored variables are used in verification query
            query = "SELECT AES_ENCRYPT('"+psswd+"','"+psswd+"') =
Password FROM evote.official o WHERE FirstName='" + firstname + "' AND
FirstLastName='" + lastname1 + "' AND SecondLastName='" + lastname2 + "'
AND Precinct ='" + precinct + "' AND Unit='" + unit + "';";
            //query = "SELECT CAST(AES_DECRYPT(Password,'" + psswd + "')
as CHAR) FROM evote.official o WHERE FirstName='" + firstname + "' AND
FirstLastName='" + lastname1 + "' AND SecondLastName='" + lastname2 + "'
AND Precinct ='" + precinct + "' AND Unit='" + unit + "';";
            OdbcCom.CommandText = query;
            OdbcDR = OdbcCom.ExecuteReader();

            //If record is found in query result
            if (OdbcDR.Read())
            {
                //compares provided password with password in database
                //If passwords match
                if ("1".Equals(OdbcDR[0].ToString()))
                {

                    //close current reader
                    OdbcDR.Close();

                    //obtain Official ID of person that is logging in
                    query = "SELECT OfficialID FROM evote.official o WHERE
FirstName='" + firstname + "' AND FirstLastName='" + lastname1 + "' AND
SecondLastName='" + lastname2 + "' AND Precinct ='" + precinct + "' AND
Unit='" + unit + "';";
                    OdbcCom.CommandText = query;
                    OdbcDR = OdbcCom.ExecuteReader();

                    if (OdbcDR.Read())
                    {
                        OfficialID = OdbcDR[0] + "";

                    }

                    //Creates new Admin form and passes selected precinct
and unit
                    Form Administrator = new Admin(precinct, unit,
OfficialID, this.FindForm());

                    //Erase all variables and close all connections
                    //query = "" ;
                    //precinct = "" ;
                    //unit = "" ;
                    firstname = "" ;
                    lastname1 = "" ;
                    lastname2 = "" ;
                    psswd = "" ;
                    OfficialID = "" ;
```

```
                    //comboBox1.Items.Clear();
                    comboBox2.Items.Clear();
                    comboBox3.Items.Clear();
                    textBox1.ResetText();
                    this.Visible = false;
                    Administrator.ShowDialog();

                }
                else
                {
                    //Shown when incorrect information is provided
                    MessageBox.Show("Información incorrecta");
                }
            }
            else
            {
                //Shown were query returned no results
                MessageBox.Show("No se encontró archivo");
            }


            //Always close reader when it is not in use
            OdbcDR.Close();
        }
    }
}
```

2. Admin Form Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.Threading;
using System.Printing;
using System.Management;
using OLEPRNLib;
using System.Collections.Specialized;

namespace WindowsFormsApplication1
{
    public partial class Admin : Form
    {
        //Contains the Database Address
        private String dbAddress = "DRIVER={MySQL ODBC 5.1
Driver};SERVER=136.145.56.170;PORT=3306;DATABASE=evote;UID=evote;PWD=evote;OP
TION=3";
```

```
        private String UnidadActiva; //Stores the unit selected by the
employee at the login
        private String PrecintoActivo; //Stores the precinct selected by the
employee at the login
        private String CodigoOficial;//Stores the code representing the
official
        private String numElectoral; // Stores the ID of the current voter to
be registered
        private Form LoginForm; // Contains the reference of the Login Form

        //UART information variables
        private String Text = "";
        private String Party = "";
        private String Governor = "";
        private String ResCom = "";
        private int characters = 65;

        //Device Status variables
        private Boolean status1 = false;  // false when device is not
connected
        private Boolean status2 = false;

        // Initializes the intermediate and master keys
        private static IntermediateKey interKeys;
        private static MasterKey masterKeys;

        /**
         *  Method used to initialize an Admin Form with the parameters
selected in the login form
         **/
        public Admin(String Precinto, String Unidad, String Codigo, Form Log)
        {
            InitializeComponent();
            UnidadActiva = Unidad;
            PrecintoActivo = Precinto;
            CodigoOficial = Codigo;
            LoginForm = Log;
            populateCBCasetaHTab();
            VerHistorialButton.Enabled = false;
            populateCBCerrarCaseta();
            populateCBAbrirCaseta();
            interKeys = new IntermediateKey();
            masterKeys = new MasterKey(interKeys);
            VerStatusButton.Enabled = false;
            characters = 65;
            if (!serialPort1.IsOpen)
                try
                {
                    serialPort1.Open();
                }
                catch
                { }
        }

        /**
         * Refresh the status tab when a kiosk is selected
```

```
         **/
        private void comboBox2_SelectedIndexChanged(object sender, EventArgs
e)
        {
            VerStatusButton.Enabled = true;
            TBEstadoSTab.Text = "";
            LBStatus.Items.Clear();
        }

        /**
         *  Updates the voter record with the already voted status
         **/
        private void voted()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; //Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);
            OdbcConnect.Open();
            String hasVotedQ = "UPDATE evote.voter v SET AlreadyVoted=
'1'where ElectoralNumber='" + numElectoral + "';";
            OdbcComm = new System.Data.Odbc.OdbcCommand(hasVotedQ,
OdbcConnect);
            OdbcComm.ExecuteNonQuery();
            OdbcConnect.Close();
        }

        /**
         *  Calls the Verification method
         **/
        private void ButtonVerificar_Click(object sender, EventArgs e)
        {
            Verification();
        }

        /**
         *    Verifies if the voter has not voted already and if the ID is
assgined to the current Precinct and Unit
         *    If the has not voted the asignment field will be activated.
         *    If the voter has already voted or not assigned to that
precinct/unit a message will appear indicating the issue.
         **/
        private void Verification()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; //Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object
            String UnidadAsignadaQ; //Query to get the assigned unit of the
voter
            String PrecintoAsignadoQ; // Query to get the assigned precinct
to the voter
            String alreadyVotedQ; // Query to get if the voter has or not
voted
            String alreadyVoted; // Stores if the voter has voted or not.
Equals to 0 if has not voted, and equals to 1 if already voted
            String UnidadAsignada; //Stores the assigned unit of the voter
```

```
            String PrecintoAsignado; //Stores the assigned precinct of the
voter
            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);
            try
            {
                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Opens the database connection
                    OdbcConnect.Open();

                    // Reads the Numero Electotal ID to be verfied
                    numElectoral = TBNumeroElectoral.Text;

                    if (NumeroElectorValido())
                    {
                        // Looks in the tabase to determine if the voter has
already voted
                        alreadyVotedQ = "SELECT AlreadyVoted FROM voter v
Where ElectoralNumber='" + numElectoral + "';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(alreadyVotedQ, OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        alreadyVoted = OdbcDR[0].ToString();
                        OdbcDR.Close();

                        // Looks in the database the Precinct to which the
voter is assigned
                        PrecintoAsignadoQ = "SELECT Precinct FROM voter v
Where ElectoralNumber='" + numElectoral + "';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(PrecintoAsignadoQ, OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        PrecintoAsignado = OdbcDR[0].ToString();
                        OdbcDR.Close();

                        // Looks in the database the Unit to which the voter
is assigned
                        UnidadAsignadaQ = "SELECT Unit FROM voter v Where
ElectoralNumber='" + numElectoral + "';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(UnidadAsignadaQ, OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        UnidadAsignada = OdbcDR[0].ToString();
                        OdbcDR.Close();

                        //Verifies if the voter has not voted already and if
the voter is assigned to the current unit and precinct
                        if (alreadyVoted.Equals("0") &&
UnidadActiva.Equals(UnidadAsignada) &&
PrecintoActivo.Equals(PrecintoAsignado))
                        {
                            LAutorizacionRTab.Enabled = true;
                            LCasetaRTab.Enabled = true;
```

```
                            CBCasetaRTab.Enabled = true;
                            ButtonAsignar.Enabled = true;
                            populateCBCasetaRTab();
                        }

                        // If the voter has already voted, a warning screen
will appear.
                        else if (alreadyVoted.Equals("1"))
                        {
                            MessageBox.Show("Elector ya ejerció su derecho al
voto en estas elecciones.", "Advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            TBNumeroElectoral.Clear(); // Refresh the Numero
Electoral text box
                        }
                        // If the voter has not voted but is assigned to
another Unit or Precinct a notification screen will appear
                        else
                        {
                            MessageBox.Show("Elector no esta asignado a esta
unidad o precinto.", "Advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                            TBNumeroElectoral.Clear();// Refresh the Numero
Electoral text box
                        }
                    }

                    else
                        MessageBox.Show("Número Electoral no es válido.",
"Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    //Closes the DB Connection
                    OdbcConnect.Close();
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }


        /**
         *  Call the metohd to populate the Caseta combo box
         **/
        private void CBCasetaRTab_SelectedIndexChanged_1(object sender,
EventArgs e)
        {
            populateCBCasetaRTab();
        }


        /**
         * Assigns a kiosk to the voter
```

```
         **/
        private void ButtonAsignar_Click(object sender, EventArgs e)
        {

            try
            {
                System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
                System.Data.Odbc.OdbcCommand OdbcComm; // Database command
object
                System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader
object

                //Initialize the connection to the database
                OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Opens the database connection
                    OdbcConnect.Open();

                    try
                    {
                        String casetaAsignada =
CBCasetaRTab.SelectedItem.ToString(); //Stores the kiosk asigned to the voter

                        if (casetaAsignada.Equals("1"))
                        {
                            if (printerStatus().Equals("OK"))
                            {

                                //Added by Laura
                                //Unlocks Voting device
                                try
                                {
                                    if (!serialPort1.IsOpen)
                                    {
                                        serialPort1.Open();
                                    }
                                        serialPort1.Write("0");
                                        //Set the status of the assgined
kiosk as occupied
                                        String casetaAsignadaQ = "UPDATE
evote.kiosk k SET Occupied = '1' where Unit ='" + UnidadActiva +
                                        "' and Precinct='" + PrecintoActivo +
"' and LocalID='" + casetaAsignada + "'";
                                        OdbcComm = new
System.Data.Odbc.OdbcCommand(casetaAsignadaQ, OdbcConnect);
                                        OdbcComm.CommandText =
casetaAsignadaQ;
                                        OdbcComm.ExecuteNonQuery();

                                        //Updates the voter record with the
assigned kiosk and official
                                        String LugarDondeVotoQ = "SELECT kID
FROM evote.kiosk k Where Precinct='" + PrecintoActivo +
```

```
                                            "' and LocalID='" +
casetaAsignada + "';";
                                        OdbcComm = new
System.Data.Odbc.OdbcCommand(LugarDondeVotoQ, OdbcConnect);
                                        OdbcDR = OdbcComm.ExecuteReader();
                                        OdbcDR.Read();
                                        String LugarDondeVoto =
OdbcDR[0].ToString();
                                        OdbcDR.Close();
                                        String SetLugarDondeVotoQ = "UPDATE
evote.voter v SET kID='" + LugarDondeVoto + "'where ElectoralNumber='" +
numElectoral + "';";
                                        OdbcComm = new
System.Data.Odbc.OdbcCommand(SetLugarDondeVotoQ, OdbcConnect);
                                        OdbcComm.ExecuteNonQuery();
                                        String SetOficialQ = "UPDATE
evote.voter v SET OfficialID='" + CodigoOficial + "'where ElectoralNumber='"
+ numElectoral + "';";
                                        OdbcComm = new
System.Data.Odbc.OdbcCommand(SetOficialQ, OdbcConnect);
                                        OdbcComm.ExecuteNonQuery();

                                        //Shows a message to the employee
notifying that the authorization is completed
                                        MessageBox.Show("Autorización
Procesada", "Autorización", MessageBoxButtons.OK,
MessageBoxIcon.Information);

                                        //Refresh the Numero Electoral ID
Text Box
                                        TBNumeroElectoral.Clear();

                                        //Refresh the Kiosk Selection
                                        CBCasetaRTab.Items.Clear();

                                        //Disables the authorization fields
                                        LAutorizacionRTab.Enabled = false;
                                        LCasetaRTab.Enabled = false;
                                        CBCasetaRTab.Enabled = false;
                                        ButtonAsignar.Enabled = false;
                                        CBCasetaRTab.Text = "";

                                        //Closes the database connection
                                        OdbcConnect.Close();
                                    }

                                    catch
                                    {
                                        MessageBox.Show("Caseta seleccionada no
puede ser asignada. Favor verificar el estado de la caseta", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                    }
                                }
                                else
                                    MessageBox.Show("Caseta seleccionada no puede
ser asignada. Favor verificar el estado de la impresora en la caseta.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                        }

                         else
                           MessageBox.Show("Favor seleccionar una caseta o
asegurar que la caseta tenga coneccion", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }

                    catch
                    {
                        MessageBox.Show("Favor seleccionar una caseta o
asegurar que la caseta tenga coneccion", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    }
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        /**
         *  Verifies if the Electoral Number is a valid ID by searching for a
record with that ID
         **/
        private bool NumeroElectorValido()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for a record with the ID
                    String IDQ = "SELECT * FROM voter v Where
ElectoralNumber='" + numElectoral + "';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(IDQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();
                    if (OdbcDR.HasRows)
                    {
                        OdbcConnect.Close();
                        return true;
                    }
```

```
                    else
                    {
                        OdbcConnect.Close();
                        return false;
                    }
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            return false;
        }


        /**
         *  Populates the Caseta Combo Box with the unoccupied kiosks
         **/
        private void populateCBCasetaRTab()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for unoccupied kiosks
                    String casetaQ = "SELECT LocalID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo + "' and Occupied = '0' and Open =
'1';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();

                    //Reads the casetaQ query and populates the combo box
with available (unoccupied) kiosks.
                    while (OdbcDR.Read())
                    {
                        CBCasetaRTab.Items.Add(OdbcDR[0]);

                    }
                    // Show a notice if all the kiosks are occupied
                    if (CBCasetaRTab.Items.Count.ToString().Equals("0"))
```

```
                        MessageBox.Show("Todas las casetas estan ocupadas en
este momento", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    OdbcDR.Close();
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }



        /**
         *  Enables de "Ver historial" button when a kiosk from the drop down
list is selected
         **/
        private void CBCasetaHTab_SelectedIndexChanged(object sender,
EventArgs e)
        {
            LBHistorialHTab.Items.Clear();
            VerHistorialButton.Enabled = true;
        }

        /**
         *  Populates de "Historial" list box with the corresponding history
         **/
        private void populateLBHTab(String NumCaseta)
        {
            String casetaSeleccionada = NumCaseta;//Selected kiosk
            String idCaseta;//ID of the selected kiosk
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for the global ID
                    String casetaQ = "SELECT kID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo + "' and localID ='" +
casetaSeleccionada + "';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
```

```
                    OdbcDR = OdbcComm.ExecuteReader();
                    OdbcDR.Read();
                    idCaseta = OdbcDR[0].ToString();
                    OdbcDR.Close();

                    String voters = "SELECT ElectoralNumber FROM evote.voter
v where kID='" + idCaseta + "' and AlreadyVoted='1';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(voters,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();

                    //Reads the casetaQ query and populates the list with the
history of the kiosk.
                    while (OdbcDR.Read())
                    {
                        System.Data.Odbc.OdbcCommand OdbcComm2; // Database
command object
                        System.Data.Odbc.OdbcDataReader OdbcDR2; // Database
reader object
                        String OficialQ = "SELECT OficialID FROM
evote.voter v where ElectoralNumber='" + OdbcDR[0] + "';";
                        OdbcComm2 = new
System.Data.Odbc.OdbcCommand(OficialQ, OdbcConnect);
                        OdbcDR2 = OdbcComm2.ExecuteReader();
                        OdbcDR2.Read();
                        String idOficial = OdbcDR2[0].ToString();
                        OdbcDR2.Close();

                        System.Data.Odbc.OdbcCommand OdbcComm3; // Database
command object
                        System.Data.Odbc.OdbcDataReader OdbcDR3; // Database
reader object
                        String OficialQ2 = "SELECT FirstName, FirstLastName,
SecondLastName FROM evote.official o where OficialID='" + idOficial + "';";
                        OdbcComm3 = new
System.Data.Odbc.OdbcCommand(OficialQ2, OdbcConnect);
                        OdbcDR3 = OdbcComm3.ExecuteReader();
                        OdbcDR3.Read();
                        String nombreOficial = OdbcDR3[0].ToString() + " " +
OdbcDR3[1].ToString() + " " + OdbcDR3[2].ToString(); OdbcDR3.Close();
                        OdbcDR3.Close();

                        LBHistorialHTab.Items.Add(OdbcDR[0] + "       " +
nombreOficial);

                    }

                    OdbcDR.Close();
                    CBCasetaHTab.Text = "";//clears the text on the drop down
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
        }

        /**
         *  Populates the kiosk drop down of the "Historial" tab
         *  Populates the kiosk drop dowm of the "Status" tab
         **/
        private void populateCBCasetaHTab()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for kiosks
                    String casetaQ = "SELECT LocalID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo + "';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();

                    //Reads the casetaQ query and populates the combo box
with available kiosks.
                    while (OdbcDR.Read())
                    {
                        CBCasetaHTab.Items.Add(OdbcDR[0]);
                        CBCasetaSTab.Items.Add(OdbcDR[0]);
                    }

                    OdbcDR.Close();
                }

            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }


        /**
         *   Ends the session and returns to the login screen
         **/
        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
```

```
        {
            this.Visible = false;
            LoginForm.Visible = true;
            UnidadActiva = "";
            PrecintoActivo = "";
            serialPort1.Close();
        }

        /**
         *  Provides infomation about the product
         **/
        private void sobreEVoteToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            MessageBox.Show("eVote \r\nVersion 1.0 \r\nCopyright 2008 por
JSAL. Todos los derechos reservados", "Sobre eVote", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }


        /**
         *  Disables the close button to avoid the user closing the
application with logging off
         **/
        private void Admin_Load(object sender, EventArgs e)
        {
            this.ControlBox = false;
        }


        /**
         *   Populates the "Historial" list box with the corresponding
history after "Ver Historial" is clicked
         **/
        private void button1_Click(object sender, EventArgs e)
        {
            LBHistorialHTab.Items.Clear();
            populateLBHTab(CBCasetaHTab.SelectedItem.ToString());
        }


        /*
         *   Call the populateCBCerrarCaseta function when the field is
clicked
         **/
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            populateCBCerrarCaseta();
        }

        /**
         *  Populates the "Cerrar Caseta" drop down list of the "Registrar"
Tab
         **/
        private void populateCBCerrarCaseta()
        {
```

```
            //CBCerrarCaseta.Items.Clear();
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for open kiosks
                    String casetaQ = "SELECT LocalID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo + "'and Open = '1';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();

                    //Reads the casetaQ query and populates the combo box
with open kiosks.
                    while (OdbcDR.Read())
                    {
                        CBCerrarCaseta.Items.Add(OdbcDR[0]);
                    }

                    OdbcDR.Close();
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }


        /**
         *  Populates the "Abrir Caseta" drop down list of the "Registrar"
Tab
         **/
        private void populateCBAbrirCaseta()
        {
            System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);
```

```
            try
            {

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Open DB Connection
                    OdbcConnect.Open();

                    //Looks for open kiosks
                    String casetaQ = "SELECT LocalID FROM evote.kiosk k where
Unit='" + UnidadActiva + "' and Precinct='" +
                        PrecintoActivo + "'and Open = '0';";
                    OdbcComm = new System.Data.Odbc.OdbcCommand(casetaQ,
OdbcConnect);
                    OdbcDR = OdbcComm.ExecuteReader();

                    //Reads the casetaQ query and populates the combo box
with open kiosks.
                    while (OdbcDR.Read())
                    {
                        CBAbrirCaseta.Items.Add(OdbcDR[0]);
                    }

                    OdbcDR.Close();
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        /*
         *   Call the populateCBAbrirCaseta function when the field is
clicked
         **/
        private void CBAbrirCaseta_SelectedIndexChanged(object sender,
EventArgs e)
        {
            populateCBAbrirCaseta();
        }


        /**
         *  Closes the selected kiosk when the "Cerrar" button is pressed
         **/
        private void OKCerrarCaseta_Click_1(object sender, EventArgs e)
        {
            try
            {
                System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
                System.Data.Odbc.OdbcCommand OdbcComm; // Database command
object
```

```
                //Initialize the connection to the database
                OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Opens the database connection
                    OdbcConnect.Open();

                    try
                    {
                        String casetaSeleccionada =
CBCerrarCaseta.SelectedItem.ToString(); //Stores the kiosk asigned to the
voter

                        //Set the status of the selected kiosk as closed
                        String casetaSeleccionadaQ = "UPDATE evote.kiosk k
SET Open = '0' where Unit ='" + UnidadActiva +
                            "' and Precinct='" + PrecintoActivo + "' and
LocalID='" + casetaSeleccionada + "'";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(casetaSeleccionadaQ, OdbcConnect);
                        OdbcComm.CommandText = casetaSeleccionadaQ;
                        OdbcComm.ExecuteNonQuery();

                        //Closes the database connection
                        OdbcConnect.Close();
                        CBCerrarCaseta.Text = "";
                        CBCerrarCaseta.Items.Clear();
                        populateCBCerrarCaseta();
                        CBAbrirCaseta.Items.Clear();
                        populateCBAbrirCaseta();
                        CBCasetaRTab.Items.Clear();
                        populateCBCasetaRTab();
                        if (serialPort1.IsOpen)
                        {
                            serialPort1.Write("#");
                            serialPort1.DiscardInBuffer();
                        }
                    }

                    catch
                    {
                        MessageBox.Show("Favor seleccionar una caseta",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
```

```
        /**
         *  Opens the selected kiosk when the "Abrir" button is pressed
         **/
        private void OKAbrirCaseta_Click(object sender, EventArgs e)
        {
            try
            {
                System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
                System.Data.Odbc.OdbcCommand OdbcComm; // Database command
object
                System.Data.Odbc.OdbcCommand OdbcComm2; // Database command
object

                //Initialize the connection to the database
                OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Opens the database connection
                    OdbcConnect.Open();

                    try
                    {
                        String casetaSeleccionada =
CBAbrirCaseta.SelectedItem.ToString(); //Stores the kiosk asigned to the
voter

                        //Set the status of the selected kiosk as open
                        String casetaSeleccionadaQ = "UPDATE evote.kiosk k
SET Open = '1' where Unit ='" + UnidadActiva +
                                "' and Precinct='" + PrecintoActivo + "' and
LocalID='" + casetaSeleccionada + "'";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(casetaSeleccionadaQ, OdbcConnect);
                        OdbcComm.CommandText = casetaSeleccionadaQ;
                        OdbcComm.ExecuteNonQuery();

                        //De-Occupy a kiosk when it is opened
                        String casetaOcupadaQ = "UPDATE evote.kiosk k SET
Occupied = '0' where Unit ='" + UnidadActiva + "' and Precinct='" +
PrecintoActivo + "' and LocalID='" + casetaSeleccionada + "'";
                        OdbcComm2 = new
System.Data.Odbc.OdbcCommand(casetaOcupadaQ, OdbcConnect);
                        OdbcComm2.CommandText = casetaOcupadaQ;
                        OdbcComm2.ExecuteNonQuery();

                        //Closes the database connection
                        OdbcConnect.Close();

                        CBAbrirCaseta.Text = "";
                        CBAbrirCaseta.Items.Clear();
                        populateCBAbrirCaseta();
                        CBCerrarCaseta.Items.Clear();
                        populateCBCerrarCaseta();
                        CBCasetaRTab.Items.Clear();
                        populateCBCasetaRTab();
```

```
                    }

                    catch
                    {
                        MessageBox.Show("Favor seleccionar una caseta",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }


        /**
         *  Verfies the status of the printer for a selected kiosk
         **/
        private string printerStatus()
        {
            string[] ErrorMessageText = new string[8];

            ErrorMessageText[0] = "servicio requerido";//"service requested";
            ErrorMessageText[1] = "no conectado";//"offline";
            ErrorMessageText[2] = "papel atascado";//"paper jammed";
            ErrorMessageText[3] = "puerta abierta";//"door open";
            ErrorMessageText[4] = "no tiene tinta";//"no toner";
            ErrorMessageText[5] = "nivel de tinta bajo";//"toner low";
            ErrorMessageText[6] = "no tiene papel";//"out of paper";
            ErrorMessageText[7] = "nivel de papel bajo";//low paper";

            int DeviceID = 1;
            int Retries = 1;
            int TimeoutInMS = 2000;
            string CommunityString = "public";

            string IPAddressOfPrinter = "";

            if (CBCasetaSTab.SelectedItem.ToString().Equals("1"))
                IPAddressOfPrinter = "136.145.57.108";//capstone 2
            else if (CBCasetaSTab.SelectedItem.ToString().Equals("2"))
                IPAddressOfPrinter = "136.145.57.243";//capstone

            if (!IPAddressOfPrinter.Equals(""))
            {
                try
                {
                    // Create instance of COM object
                    OLEPRNLib.SNMP snmp = new OLEPRNLib.SNMP();

                    // Open the SNMP connect to the printer
                    snmp.Open(IPAddressOfPrinter, CommunityString, Retries,
TimeoutInMS);
```

```
                    // The actual Warning/Error bits
                    uint WarningErrorBits =
snmp.GetAsByte(String.Format("25.3.5.1.2.{0}", DeviceID));

                    // The actual Printer Status
                    uint StatusResult =
snmp.GetAsByte(String.Format("25.3.2.1.5.{0}", DeviceID));
                    // uint Result2 =
snmp.GetAsByte(String.Format("25.3.5.1.1.{0}", DeviceID));

                    snmp.Close();

                    string Result1Str = "";
                    switch (StatusResult)
                    {
                        case 2: Result1Str = "OK";
                            break;
                        case 3: Result1Str = "Advertencia: ";//"Warning: ";
                            break;
                        case 4: Result1Str = "Bajo prueba: ";//"Being Tested:
";
                            break;
                        case 5: Result1Str = "No disponible para ningun uso:
";//"Unavailable for any use: ";
                            break;
                        default: Result1Str = "Status desconocido, código:
";//"Unknown Status Code : " + StatusResult;
                            break;
                    }

                    string Str = "";
                    if ((StatusResult == 3 || StatusResult == 5))
                    {
                        int Mask = 1;
                        int NumMsg = 0;
                        for (int i = 0; i < 8; i++)
                        {
                            if ((WarningErrorBits & Mask) == Mask)
                            {
                                if (Str.Length > 0)
                                    Str += ", ";
                                Str += ErrorMessageText[i];
                                NumMsg = NumMsg + 1;
                            }
                            Mask = Mask * 2;
                        }
                    }
                    return Result1Str + Str;
                }
                catch
                {
                    return "Impresora está apagada o desconectada";
                }
            }

        else
```

```
                return "Información sobre impresora no
disponible.";//"Printer information is not available.";
        }

        /**
         *  Populates the printer status field
         **/
        private void populateStatus()
        {
            LBStatus.Items.Clear();
            LBStatus.Items.Add(printerStatus());
        }

        /**
         *  Refresh the status tab
         **/
        private void VerStatusButton_Click(object sender, EventArgs e)
        {
            populateStatus();
            populateEstadoDevice();
        }

        //Added by Laura
        //Code for receiving UART information

        private void separateText(String completeText)
        {
            Party = completeText.Substring(1, 3);
            Governor = completeText.Substring(6, 28);
            ResCom = completeText.Substring(36, 28);
        }

        /**
         *   Verifies if the data received from the device is complete
         * */
        private Boolean receivedHeaders(String T)
        {
            Boolean complete = false;

            //Only performs necessary check if string if of appropiate length
            if (T.Length == 65)
            {
                //Get characters in header and footer position
                String H1 = T.Substring(0, 1);
                String H2 = T.Substring(4, 1);
                String H3 = T.Substring(5, 1);
                String H4 = T.Substring(34, 1);
                String H5 = T.Substring(35, 1);
                String H6 = T.Substring(64, 1);

                //Check that characters are headers and footers
                if (H1.Equals("1") && H2.Equals("1") && H3.Equals("2") &&
H4.Equals("2") && H5.Equals("3") && H6.Equals("3"))
                {
                    //Text is complete
                    complete = true;
                }
```

```
            }

            //Return wether text is complete or not
            return complete;
        }

        /**
         * Receives the information from the device
         * Send information to print
         * Updates the database after the printing is sucessfull
         **/
        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            int read = serialPort1.ReadChar();
            string letter = Char.ConvertFromUtf32(read);

            if (letter.Equals("#"))
            {
                try
                {
                    if(!serialPort1.IsOpen)
                    serialPort1.Open();
                }
                catch
                {
                    status1 = false;
                }
                serialPort1.Write("#");
                serialPort1.DiscardInBuffer();
            }

            //Total number of characters: 59 letters or spaces plus 6 start
and end characters

            if (characters > 0 && letter != "#")
            {
                Text = Text + letter;
                characters--;
            }

            //If all 65 characters have been obtained

            if (characters == 0)
            {
                //separate into different variables
                separateText(Text);
                characters = 65;

                //If correct text was received sent "OK" (@) to evote device
                if (receivedHeaders(Text))
                {
                    //IF CHECK PRINT
                    //ELSE NO PRINT THEN ERROR
                    //ENVIARLO A IMPRIMIR
                    //Send results to database, erase all variables and close
all connections after vote is sent*****
```

```
                    System.Data.Odbc.OdbcConnection OdbcConnect4; // Database
connection object
                    System.Data.Odbc.OdbcCommand OdbcComm4; // Database
command object
                    System.Data.Odbc.OdbcDataReader OdbcDR4; //Databse reader
object

                    OdbcConnect4 = new
System.Data.Odbc.OdbcConnection(dbAddress);
                    OdbcConnect4.Open();

                    String votoGobQ = "";

                    String printParty = " ";
                    String printGobernador = " ";
                    String printComisionado = " ";

                    string TEMPKEY = "SELECT CAST(AES_DECRYPT(MasterKey,'" +
interKeys.getIntermediateKeyByPrecinct(PrecintoActivo) + "')as BINARY) FROM
evote.masterkey WHERE ID='" + interKeys.getPrecinctIndex(PrecintoActivo) +
"'";

                    // Generate the queries to update the database and the
variables to send the summary to print for the governor option
                    if (Governor.Equals("anibal acevedo vila         "))
                    {

                        votoGobQ = "INSERT INTO evote.candidatevotes (CanID)
VALUES(AES_ENCRYPT('1','TEMPKEY'))";
                        printGobernador = "Anibal Acevedo Vila";
                    }
                    else if (Governor.Equals("luis fortuno               "))
                    {
                        votoGobQ = "INSERT INTO evote.candidatevotes (CanID)
VALUES(AES_ENCRYPT('2','TEMPKEY'))";
                        printGobernador = "Luis Fortuño";
                    }
                    else if (Governor.Equals("edwin irizarry mora        "))
                    {
                        votoGobQ = "INSERT INTO evote.candidatevotes (CanID)
VALUES(AES_ENCRYPT('3','TEMPKEY'))";
                        printGobernador = "Edwin Irizarry Mora";
                    }
                    else if (Governor.Equals("rogelio figueroa garcia    "))
                    {
                        votoGobQ = "INSERT INTO evote.candidatevotes (CanID)
VALUES(AES_ENCRYPT('4','TEMPKEY'))";
                        printGobernador = "Rogelio Figueroa Garcia";
                    }
                    else
                    {
                        //Caso del write in
                        votoGobQ = "INSERT INTO evote.writein (Name,Position)
VALUES(AES_ENCRYPT('" + Governor + "','TEMPKEY'), 'Gobernador')";
                        printGobernador = Governor;
                    }
```

```
                    String votoResComQ = "";
                    // Generate the queries to update the database and the
variables to send the summary to print for the commisioner option
                    if (ResCom.Equals("alfredo salazar            "))
                    {
                        votoResComQ = "INSERT INTO evote.candidatevotes
(CanID) VALUES(AES_ENCRYPT('5','TEMPKEY'))";
                        printComisionado = "Alfredo Salazar";
                    }
                    else if (ResCom.Equals("pedro pierluisi           "))
                    {
                        votoResComQ = "INSERT INTO evote.candidatevotes
(CanID) VALUES(AES_ENCRYPT('6','TEMPKEY'))";
                        printComisionado = "Pedro Pierluisi";
                    }
                    else if (ResCom.Equals("jessica martinez birriel    "))
                    {
                        votoResComQ = "INSERT INTO evote.candidatevotes
(CanID) VALUES(AES_ENCRYPT('7','TEMPKEY'))";
                        printComisionado = "Jessica Martinez Birriel";
                    }
                    else if (ResCom.Equals("carlos alberto velazquez    "))
                    {
                        votoResComQ = "INSERT INTO evote.candidatevotes
(CanID) VALUES(AES_ENCRYPT('8','TEMPKEY'))";
                        printComisionado = "Carlos Velazquez";
                    }
                    else
                    {
                        //Caso del write in
                        votoResComQ = "INSERT INTO evote.writein
(Name,Position) VALUES (AES_ENCRYPT('" + ResCom + "','TEMPKEY'),
'ComisionadoResidente')";
                        printComisionado = ResCom;
                    }



                    String votoPartyQ = "";

                    // Generate the queries to update the database and the
variables to send the summary to print for the party option
                    if (Party.Equals("ppd"))
                    {
                        votoPartyQ = "INSERT evote.partyvotes (pID)
VALUES(AES_ENCRYPT('1','TEMPKEY'))";
                        printParty = "PPD";
                    }
                    else if (Party.Equals("pnp"))
                    {
                        votoPartyQ = "INSERT evote.partyvotes (pID)
VALUES(AES_ENCRYPT('2','TEMPKEY'))";
                        printParty = "PNP";
                    }
```

```
                    else if (Party.Equals("pip"))
                    {
                        votoPartyQ = "INSERT evote.partyvotes (pID)
VALUES(AES_ENCRYPT('3','TEMPKEY'))";
                        printParty = "PIP";
                    }
                    else if (Party.Equals("ppr"))
                    {
                        votoPartyQ = "INSERT evote.partyvotes (pID)
VALUES(AES_ENCRYPT('4','TEMPKEY'))";
                        printParty = "PPR";
                    }


                    //If print is sucesfully update the database
                    if (printResults(printParty, printGobernador,
printComisionado))
                    {
                        //retrieve master key
                        OdbcComm4 = new System.Data.Odbc.OdbcCommand(TEMPKEY,
OdbcConnect4);
                        OdbcComm4.CommandText = TEMPKEY;
                        OdbcDR4 = OdbcComm4.ExecuteReader();
                        OdbcDR4.Read();
                        TEMPKEY = "";
                        TEMPKEY = OdbcDR4.GetString(0);
                        OdbcDR4.Close();

                        //update db
                        votoGobQ = votoGobQ.Replace("TEMPKEY", TEMPKEY);
                        OdbcComm4 = new
System.Data.Odbc.OdbcCommand(votoGobQ, OdbcConnect4);
                        OdbcComm4.CommandText = votoGobQ;
                        OdbcComm4.ExecuteNonQuery();
                        votoGobQ = "";

                        votoResComQ = votoResComQ.Replace("TEMPKEY",
TEMPKEY);
                        OdbcComm4 = new
System.Data.Odbc.OdbcCommand(votoResComQ, OdbcConnect4);
                        OdbcComm4.CommandText = votoResComQ;
                        OdbcComm4.ExecuteNonQuery();
                        votoResComQ = "";

                        if (votoPartyQ != "")
                        {
                            votoPartyQ = votoPartyQ.Replace("TEMPKEY",
TEMPKEY);
                            OdbcComm4 = new
System.Data.Odbc.OdbcCommand(votoPartyQ, OdbcConnect4);
                            OdbcComm4.CommandText = votoPartyQ;
                            OdbcComm4.ExecuteNonQuery();
                            votoPartyQ = "";

                        }

                        voted();
```

```
                    }

                    Text = "";
                    letter = "";
                    Governor = "";
                    Party = "";
                    ResCom = "";

                    //Mark kiosk as unoccupied
                    System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
                    System.Data.Odbc.OdbcCommand OdbcComm; // Database
command object


                    //Initialize the connection to the database
                    OdbcConnect = new
System.Data.Odbc.OdbcConnection(dbAddress);

                    if (OdbcConnect.State == ConnectionState.Closed)
                    {
                        // Opens the database connection
                        OdbcConnect.Open();

                        try
                        {
                            //Set the status of the selected kiosk as closed
                            String casetaOcupadaQ = "UPDATE evote.kiosk k SET
Occupied = '0' where Unit ='1' and Precinct='San Juan 1' and LocalID='1'";
                            OdbcComm = new
System.Data.Odbc.OdbcCommand(casetaOcupadaQ, OdbcConnect);
                            OdbcComm.CommandText = casetaOcupadaQ;
                            OdbcComm.ExecuteNonQuery();
                        }
                        catch
                        {
                            MessageBox.Show("Error desocupando caseta",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        }
                    }

                    Governor = "";
                    Party = "";
                    ResCom = "";
                    TEMPKEY = "";
                    OdbcConnect4.Close();

                    serialPort1.DiscardInBuffer();
                    serialPort1.Write("@");

                }
                else
                {
                    status1 = false;
                    serialPort1.Write("%");
```

```
                }

            }
        }

        /**
         *  Populates the device status for a selected kiosk
         **/
        private void populateEstadoDevice()
        {

            try
            {
                if (serialPort1.IsOpen)
                    status1 = true;
                else
                    serialPort1.Open();
                    status1 = true;
            }
            catch
            {
                status1 = false;
            }

            if (CBCasetaSTab.SelectedItem.ToString().Equals("1"))
            {
                if (status1 == false)
                    TBEstadoSTab.Text = "Desconectado";
                else if (status1 == true)
                    TBEstadoSTab.Text = "Conectado";
                else
                    TBEstadoSTab.Text = "Desconocido";
            }

            else if (CBCasetaSTab.SelectedItem.ToString().Equals("2"))
            {
                if (status2 == false)
                    TBEstadoSTab.Text = "Desconectado";
                else if (status2 == true)
                    TBEstadoSTab.Text = "Conectado";
                else
                    TBEstadoSTab.Text = "Desconocido";
            }
            else
                TBEstadoSTab.Text= "Desconocido";


        }

        /**
         * Prints the ballot
         **/
        private Boolean printResults(String party, String governor, String
commisioner)
        {
```

```
            System.Data.Odbc.OdbcConnection OdbcConnect;// Database
connection object
            System.Data.Odbc.OdbcCommand OdbcComm; // Database command object
            System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader object

            //Initialize the connection to the database
            OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

            try
            {


                if (OdbcConnect.State == ConnectionState.Closed)
                {

                    // Open DB Connection
                    OdbcConnect.Open();

                    //Reads the selection for "Gobernador" and "Comisionado
Residente"
                    //gobernador = CBGobernadorPTab.SelectedItem.ToString();
                    //comisionado =
CBComisionadoPTab.SelectedItem.ToString();

                    String partido;
                    String gobernador; // Stores the selection for the
"Gobernador" position
                    String comisionado; // Stores  the selection for the
"Comisionado Residente" position
                    partido = party;
                    gobernador = governor;
                    comisionado = commisioner;

                    String partidoCode = " ";
                    String gobernadorCode = " ";
                    String comisionadoCode = " ";

                    // If the gobernador is selected from one of the provided
options looks for the code assigned for the barcode
                    if (partido.Equals("PPD") || partido.Equals("PNP") ||
partido.Equals("PIP") || partido.Equals("PPR"))
                    {
                        String partidoQ = "SELECT Barcode FROM
`evote`.`party`where pName= '" + partido + "';";
                        OdbcComm = new System.Data.Odbc.OdbcCommand(partidoQ,
OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        partidoCode = OdbcDR[0].ToString();
                        OdbcDR.Close();
                    }

                    if (gobernador.Equals("Anibal Acevedo Vila") ||
gobernador.Equals("Luis Fortuño") ||
                        gobernador.Equals("Edwin Irizarry Mora") ||
gobernador.Equals("Rogelio Figueroa Garcia"))
                    {
```

```
                        char[] space = { ' ' };
                        string[] gobernadorA = new string[2];
                        gobernadorA = gobernador.Split(space);
                        String gobernadorQ = "SELECT Barcode FROM
`evote`.`candidate`where FirstName = '" + gobernadorA[0] +
                            "' and FirstLastName = '" + gobernadorA[1] +
"';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(gobernadorQ, OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        gobernadorCode = OdbcDR[0].ToString();
                        OdbcDR.Close();
                    }

                    if (comisionado.Equals("Alfredo Salazar") ||
comisionado.Equals("Pedro Pierluisi") ||
                        comisionado.Equals("Jessica Martinez Birriel") ||
comisionado.Equals("Carlos Velazquez"))
                    {
                        char[] space = { ' ' };
                        string[] comisionadoA = new string[2];
                        comisionadoA = comisionado.Split(space);
                        String comisionadoQ = "SELECT Barcode FROM
`evote`.`candidate`where FirstName = '" + comisionadoA[0] +
                            "' and FirstLastName = '" + comisionadoA[1] +
"';";
                        OdbcComm = new
System.Data.Odbc.OdbcCommand(comisionadoQ, OdbcConnect);
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
                        comisionadoCode = OdbcDR[0].ToString();
                        OdbcDR.Close();

                        if (comisionado.Equals("Carlos Velazquez"))
                            comisionado = "Carlos Alberto Velazquez";
                    }

                    //Reference objects
                    object oMissing = System.Reflection.Missing.Value;
                    object oEndOfDoc = "\\endofdoc";  //end of file,
predifined bookmark

                    //Starts Word Application and create a new document.
                    Microsoft.Office.Interop.Word.Application oWord = new
Microsoft.Office.Interop.Word.Application();
                    Microsoft.Office.Interop.Word.Document oDoc = new
Microsoft.Office.Interop.Word.Document();
                    //Word._Application oWord; //2003 reference
                    //Word._Document oDoc;//2003 reference
                    //oWord = new Word.Application();
                    oWord.Visible = false; // hides the document, runs in
back without anyone seeing what it contains
                    oDoc = oWord.Documents.Add(ref oMissing, ref oMissing,
ref oMissing, ref oMissing);
```

```
                    //Insert a text line with paragraph spacing at the
beginning of the document.
                    Microsoft.Office.Interop.Word.Paragraph oPara1;
                    //Word.Paragraph oPara1;//2003 reference
                    oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);//Adds
a new paragraph to the contents of the document
                    oPara1.Range.Text = "Resultados de su votación";//Adds
text to the paragraph
                    oPara1.Range.Font.Bold = 1; // Set the text font as BOLD
                    oPara1.Range.Font.Size = 12; // Set the font size to 12
                    oPara1.Format.SpaceAfter = 24;    //Space left between
lines, 24pt
                    oPara1.Range.InsertParagraphAfter();

                    //Sets the range of the document using the end of file
bookmark
                    object oRng = oDoc.Bookmarks.get_Item(ref
oEndOfDoc).Range;

                    //Insert a 7 x 3 table to enter the results
                    Microsoft.Office.Interop.Word.Table oTable;
                    // Word.Table oTable; //Table object //2003 reference
                    Microsoft.Office.Interop.Word.Range wrdRng =
oDoc.Bookmarks.get_Item(ref oEndOfDoc).Range;
                    //Word.Range wrdRng = oDoc.Bookmarks.get_Item(ref
oEndOfDoc).Range; //Sets the range of the text for the table using the end of
file bookmark //2003 reference
                    oTable = oDoc.Tables.Add(wrdRng, 7, 3, ref oMissing, ref
oMissing); // Adds the table to the document
                    oTable.Range.ParagraphFormat.SpaceAfter = 10;//Space left
between line after the text block

                    //Insert text into the table
                    oTable.Cell(1, 1).Range.Text = "BOLETO ESTATAL";
                    oTable.Cell(3, 1).Range.Text = "PARTIDO";
                    oTable.Cell(3, 2).Range.Text = "NOMBRE DEL PARTIDO";
                    oTable.Cell(3, 3).Range.Text = "CODIGO DE BARRA";
                    oTable.Cell(5, 2).Range.Text = "NOMBRE DEL CANDIDATO";
                    oTable.Cell(5, 3).Range.Text = "CODIGO DE BARRA";
                    oTable.Cell(6, 1).Range.Text = "GOBERNADOR";
                    oTable.Cell(7, 1).Range.Text = "COMISIONADO RESIDENTE";

                    //Set the font of some of the cell to be BOLD
                    oTable.Cell(1, 1).Range.Font.Bold = 1;
                    oTable.Cell(3, 1).Range.Font.Bold = 1;
                    oTable.Cell(3, 2).Range.Font.Bold = 1;
                    oTable.Cell(3, 3).Range.Font.Bold = 1;
                    oTable.Cell(5, 2).Range.Font.Bold = 1;
                    oTable.Cell(5, 3).Range.Font.Bold = 1;
                    oTable.Cell(6, 1).Range.Font.Bold = 1;
                    oTable.Cell(7, 1).Range.Font.Bold = 1;

                    //Insert the selection into the table
                    oTable.Cell(3, 2).Range.Text = partido;
                    oTable.Cell(3, 2).Range.Font.Italic = 1;
                    oTable.Cell(3, 3).Range.Text = partidoCode;
```

```
                    oTable.Cell(3, 3).Range.Font.Name =
"IDAutomationHC39M";//set the font of the cell to be barcode
                    oTable.Cell(6, 2).Range.Text = gobernador;
                    oTable.Cell(6, 2).Range.Font.Italic = 1;
                    oTable.Cell(6, 3).Range.Text = gobernadorCode;
                    oTable.Cell(6, 3).Range.Font.Name =
"IDAutomationHC39M";//set the font of the cell to be barcode
                    oTable.Cell(7, 2).Range.Text = comisionado;
                    oTable.Cell(7, 2).Range.Font.Italic = 1;
                    oTable.Cell(7, 3).Range.Text = comisionadoCode;
                    oTable.Cell(7, 3).Range.Font.Name =
"IDAutomationHC39M";//set the font of the cell to be barcode


                    //Print the document
                    object copies = "1";
                    object pages = "1";
                    object range =
Microsoft.Office.Interop.Word.WdPrintOutRange.wdPrintCurrentPage;
                    //object range = Word.WdPrintOutRange.wdPrintCurrentPage;
//2003 reference
                    // object items =
Word.WdPrintOutItem.wdPrintDocumentContent;
                    object items =
Microsoft.Office.Interop.Word.WdPrintOutItem.wdPrintDocumentContent;
                    //object pageType = Word.WdPrintOutPages.wdPrintAllPages;
//2003 reference
                    object pageType =
Microsoft.Office.Interop.Word.WdPrintOutPages.wdPrintAllPages;
                    object oTrue = true;
                    object oFalse = false;

                    oWord.ActiveDocument.PrintOut(ref oTrue, ref oFalse, ref
range, ref oMissing, ref oMissing, ref oMissing,
                        ref items, ref copies, ref pages, ref pageType, ref
oFalse, ref oTrue,
                        ref oMissing, ref oFalse, ref oMissing, ref oMissing,
ref oMissing, ref oMissing);

                    //Closes the document without saving it
                    object doNotSaveChanges =
Microsoft.Office.Interop.Word.WdSaveOptions.wdDoNotSaveChanges;
                    oDoc.Close(ref oFalse, ref oMissing, ref oMissing);


                    //Pause to send the information to the printer before the
application is closed
                    while
(GetPrintJobsCollection(@"\\naiboa.ecenet.ece.uprm.edu\HP LaserJet 4010 PCL -
Capstone") > 0)
                    {
                        Thread.Sleep(200);
                        if (kioskIsClosed())
                            return false;
                    }

                    //Closes the Microsoft Word Application
```

```
                    oWord.Quit(ref doNotSaveChanges, ref oMissing, ref
oMissing);

                    return true;
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }

            return false;
        }

        /**
         * Verifies if the kiosk is closed
         **/
        private Boolean kioskIsClosed()
        {

            try
            {
                System.Data.Odbc.OdbcConnection OdbcConnect; // Database
connection object
                System.Data.Odbc.OdbcCommand OdbcComm; // Database command
object
                System.Data.Odbc.OdbcDataReader OdbcDR; // Database reader
object
                String cerrado = "";

                //Initialize the connection to the database
                OdbcConnect = new System.Data.Odbc.OdbcConnection(dbAddress);

                if (OdbcConnect.State == ConnectionState.Closed)
                {
                    // Opens the database connection
                    OdbcConnect.Open();

                    try
                    {
                        //String casetaSeleccionada =
CBCerrarCaseta.SelectedItem.ToString(); //Stores the kiosk asigned to the
voter
                        //Set the status of the selected kiosk as closed
                        String casetaSeleccionadaQ = "SELECT Open FROM
evote.kiosk k where Unit='" + UnidadActiva + "' and Precinct='" +
                            PrecintoActivo + "' and LocalID='1'";

                        OdbcComm = new
System.Data.Odbc.OdbcCommand(casetaSeleccionadaQ, OdbcConnect);
                        OdbcComm.CommandText = casetaSeleccionadaQ;
                        OdbcComm.ExecuteNonQuery();
                        OdbcDR = OdbcComm.ExecuteReader();
                        OdbcDR.Read();
```

```
                        cerrado = OdbcDR[0].ToString();
                        OdbcDR.Close();
                        if (cerrado == "0")
                            return true;
                        else
                            return false;

                    }

                    catch
                    {
                        MessageBox.Show("Error imprimiendo", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return false;
                    }
                }
            }

            catch
            {
                MessageBox.Show("No se pudo establecer conección con el
servidor", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

            return false;
        }

       /**
        * Look for the print queue, returns the number of pending jobs
        **/
       public static int GetPrintJobsCollection(string printerName)
       {
           StringCollection printJobCollection = new StringCollection();
           string searchQuery = "SELECT * FROM Win32_PrintJob";

           /*searchQuery can also be mentioned with where Attribute,
               but this is not working in Windows 2000 / ME / 98 machines
               and throws Invalid query error*/

        ManagementObjectSearcher searchPrintJobs =
                     new ManagementObjectSearcher(searchQuery);
           ManagementObjectCollection prntJobCollection =
searchPrintJobs.Get();
           foreach (ManagementObject prntJob in prntJobCollection)
           {
               System.String jobName =
prntJob.Properties["Name"].Value.ToString();

               //Job name would be of the format [Printer name], [Job ID]

               char[] splitArr = new char[1];
               splitArr[0] = Convert.ToChar(",");
               string prnterName = jobName.Split(splitArr)[0];
               string documentName =
prntJob.Properties["Document"].Value.ToString();
             if (prnterName.Equals(printerName))
             {
```

```
                    printJobCollection.Add(documentName);
                }
            }
            return printJobCollection.Count;
        }


        private void serialPort1_ErrorReceived(object sender,
System.IO.Ports.SerialErrorReceivedEventArgs e)
        {
            status1 = false;
        }
    }
}
```