REMOTE VEHICLE INTERFACE

A Computer Engineering senior project by

Travis Johnson & Ben Moon

University of Utah – Fall 2004

Advisor: Al Davis

- Final Documentation -

Table of Contents

Introduction & Motivation.

The RVI Project improves upon commercially available remote start/keyless-entry systems. In addition to a key-ring transmitter, users can use a telephone or WWW interface to start/kill their vehicle's engine or lock/unlock their vehicle's doors. The RVI system employs the same amount of security as a traditional key-ring device, but offers far greater range.

There were two primary motivations for this project. First, key-ring transmitters are limited to a range of usually less than tens of meters. For users that work in large office buildings or users that have campus-like parking this is very inconvenient. On a cold night, it would be nice to start your car when you *begin* your fifteen minute walk to the parking lot. Second, if a user were to lock their keys in the car, then a key-ring transmitter's usefulness has pretty much been undermined. With the RVI system, the user just needs to get to a phone in order to unlock their doors.

Description of User Interfaces.

A user has two options available for interacting with the RVI system. Each interface is treated separately below by giving a walkthrough of a typical transaction.

- ***Telephone (RVI PhoneUI)***

  A user dials the number of the RVI system, and the server answers the call. A voice prompts the user for their 10-digit phone number and 4-digit pin number; these are used for authentication. After being authenticated, the user is presented with a menu of available commands they can send to their vehicle. The

user can select as many of these as they want before terminating the call. Once the command is received from the user, it is immediately forwarded to the vehicle.

- ***World Wide Web (RVI WebUI)***

A user can navigate to *www.remotevi.com*, and click on the Login button. They will then be prompted for a username and password. After entering these, they can either hit Enter on the keyboard or click on the Submit button with their mouse. After logging in, a user stays logged in until they explicitly log out or until they close their web browser. Hence, a user can navigate away from remotevi.com, but when they return they will not be prompted for authentication. An extremely user-friendly menu has four buttons available for clicking on, one for each RVI command (start, kill, lock, & unlock). When a logged-in user clicks a button, the appropriate command is sent to the vehicle.

Description of Design.

This section will have a bulleted entry for each major component of the overall design. Under each entry, a high-level description of that particular component will be given. Source code, schematics, drawings, etc. are available in the various Appendices at the end of this document, all of which are labeled as to what they contain and are listed in the Table of Contents on the cover page.

- ***User Database***

A database was needed to store information about each client receiver. In the end, monetary issues permitted us to demonstrate only one vehicle hooked up to the RVI system, but the system was built to handle far more.

A Microsoft Access database was employed to store personal, authentication, and RF-security information about each individual client. Personal information allowed dynamic web content, which the client could update using the WebUI. Authentication information consisted of the PhoneUI pin number and the WebUI username/password. RF-security information such as PRN seed, offset, and MAC address were also stored (RF security overview can be found in Appendix A).

Connectivity with the database was easily coded in both user interfaces using the System.Data.OleDb class, which is a part of the .NET framework. This class allows for application data caching in order to minimize actual database accesses.

- ***Web User Interface***

The WebUI component is used to host information about the project, and also act as the WWW user interface for sending commands to the vehicle. Obviously it accommodated HTTP requests, but it also used the COM port for output to the RF transmitter.

This component was developed using Microsoft Visual Studio .NET 2003. Graphically, the user was presented with ASP .NET webforms; these offer more functionality and maintainability than traditional HTML. Additionally, ASP code is translated to HTML on the server end to prevent users from gleaning critical system information about services such as database connectivity, cookies, and authentication scheme. C# .NET was used to code the functionality behind the

various buttons/menus on the RVI website. When a user clicks a form button on the website, the RVI server executes the C# routine corresponding to that button.

This one-two punch of C# and ASP .NET made developing the web interface very intuitive. Neither team member had ever worked with ASP, and only one member had (very limited) experience with C#. Still, this component was probably the easiest part of the overall design.

- ***Phone User Interface***

The PhoneUI component is used to accept incoming calls, where it authenticates users and allows them to send commands to the vehicle.

This component used the computer's COM port for output to the RF transmitter, as well as a Supra Express56 voice modem to handle the incoming calls. Like the WebUI, it was also developed using MS Visual Studio .NET 2003 and C# .NET. An ActiveX Control was employed for the low-level TAPI communication (coding this by hand would have been prohibitively time-consuming).

- ***RF Link (TX/RX)***

The RF components in the design were Aerocomm AC4490 transceivers. These are OEM 900 MHz transceivers that use TTL serial communication. They allowed fast develop while maintaining the robustness we would have implemented had we built the RF components ourselves. Each transceiver uses frequency-hopping spread spectrum (FHSS) to eliminate interference in the crowded 902-928 MHz (ISM) license-free band.

On the server end, a 1000mW transceiver was used as a transmitter. On the vehicle end, a 200mW low-power transceiver was used as a receiver. They communicated with each other using what Aerocomm deems “acknowledge mode”. Each transmission is basically acknowledged by the receiver, and up to 5 retries are attempted. This limits packet loss and allows us to notify the user if their command was not successfully received by the vehicle.

Because these parts employ TTL serial communication as their native protocol, conversion from TTL to RS232 was required to connect the transceivers to their respective devices (TX-to-server and RX-to-microcontroller). This was accomplished on the vehicle end with a piggy-back converter from Aerocomm that was made specifically for the low-power transceiver. For the server end, no such part exists, so one had to be made. This just entailed a 3.3V power supply, an RS232-to-TTL converter cable, and a lot of wire-wrapping/soldering.

- ***Microcontroller (Receiver)***

An HCS08 demonstration board served as the brains of the receiver. This board was chosen because of its flexibility, low power consumption, and low cost. It came with Metrowerks CodeWarrior Development Studio, which allowed up to 4KB of C code to be written. The ability to code the behavior of the MCU in C versus HS08 assembly was a huge time-saver.

The microcontroller resides in the car, power by the vehicles battery. It spends it time almost exclusively in wait mode. This is a low-power mode that is exited upon specified interrupts. For this design, an RS232 receive interrupt is the only event that can wake the MCU from wait mode. Upon waking up, it parses

the packet that has been received for authenticity and takes the appropriate action based on the command in the packet. It then returns to wait mode until the next RS232 receive interrupt.

- ***Remote Start/Keyless-Entry***

Ideally, we would have like to build this component ourselves. After shopping some of the commercial systems available, we decided that they offered many bells and whistles that we would like in our system, only we did not have time to build something so substantial. Instead, we opted for a CodeAlarm CA-535 remote start/keyless-entry package; it was purchased/installed at Circuit City.

Integrating with this system was as easy as taking apart the extra key-ring transmitter. Going across the RF twice was not the original plan, but having the key-ring transmitter inside the car and hooked to a constant full 6 volt supply guaranteed reliable transmission.

Design Decision Log.

Date	Description
04/01/2004	C# will be used for UI development
04/01/2004	Access database; easy connectivity w/ C#
04/01/2004	VTapi ActiveX control for Phone UI
04/15/2004	68HC11 Microcontroller for vehicle end
04/15/2004	AC4490-1000 transceivers chosen; best OEM part available
05/01/2004	Changed to HCS08 MCU; better price & power consumption
06/15/2004	Changed to TAPIEx control; better price
07/01/2004	Decided not to build remote start/entry system
10/15/2004	Switched vehicle end to AC4490-200; lower power/piggy-back
11/15/2004	Decided to build RS232 converter for server-end transmitter

Parts List.

Purchased Parts (Qty)	Where	Price
Aerocomm AC4490-1000 (1)	Mouser.com	$86
Aercomm AC4490-200 (2)	Mouser.com	$130
Motorola HCS08 Demo Board (2)	Arrow Electronics	$98
TAPIEx ActiveX Control License (1)	Hotwind Software	$60
Nearson W350 Panel Antenna (1)	Nearson, Inc.	$13
Nearson P467 1/2 Wave Antenna (1)	Nearson, Inc.	$22

Designed Parts	Component Costs
6M RG-174 TM Line	$42
RS232-to-TTL Converter	$70
MCU-to-CA535 connections	$25
Battery pack (for testing)	$20

Citations.

- The JustinIO library (by Justin Harrel at Aciss Systems, Inc.) was used to gain COM port functionality in C# .NET.

Conclusions.

A number of valuable lessons were learned during the course of this project. The main snag we encountered involved the transceivers. Part of it was bad planning, and part of it was issues with the vendor. To put a 3.3V/1500mA part in a low-power setting should have never been the plan, hence the bad planning. The vendor, however, was less than helpful in rectifying the situation, even after he admitted be partially responsible. The effect was about a 3 week set-back, which cost us the coolness factor of demonstrating our finished product at a much longer range than we actually did.

Other snags encountered were all little things. They are listed below with short explanations:

- Tutorial donation – we took it for granted that a shop (any shop) in town would show us how to install a remote starter. Everyone, without fail, cited legal reasons why they didn't want to be liable for us should something go wrong.
- Little parts & big shipping costs – numerous times we found ourselves needing something simple like a power connector or a DB9 gender changer. These parts are less than a dollar, but shipping is usually $8-$9. Great planning could have eliminated some of this.
- C# and the COM port – there is no inherent library in C# for COM port communication. Thanks again to Justin at Aciss Systems, Inc. for posting his serial library in the DevShed forums.

Acknowledgments.

- Endless help in writing code for microcontrollers was received from the discussion boards at www.metrowerks.com

Appendix A – Security Overview

Introduction.
Obviously, I would not be smart to divulge exactly how to hack the RVI system. This document is instead meant to explain the general theory behind the security that has been implemented in the RVI prototype.

Random Numbers.
Like the commercially available remote start/keyless entry systems, the RVI system uses pseudo-random numbers (PRNs). The algorithm employed can generate huge strings of PRNs in the 0 to 255 range without generating sequences. The algorithm is seeded, and each vehicle receiver has its own unsigned 32-bit seed, allowing over 4 billion receivers.

Packets.
Basically two things need to get transmitted: (1) the command that tells receiver what to do, and (2) some sort of authorization based on the unique PRN sequence for that particular receiver. The possibilities here range from the very simple (sending the next PRN in the sequence along with a one byte command) to the more complex (a huge packet with encrypted commands and PRN subsequences). The RVI implementation is explained in the following section.

Implementation.
This is a close example of the actual system, but not the exact implementation (in the spirit of security). A packet will consist of a one byte command, two one byte authentication numbers, and 5 truly random numbers.

G0	C	A0	G1	G2	A1	G3	G4

a. . . Example Packet -

- The command byte is self explanatory.
- The two authentication numbers are a subsequence of the PRN sequence unique to that receiver. The user database, as well as the receiver, stores an offset number that indexes into the PRN sequence. The receiver will accept the next 64 possible two-number sequences to allow up to 63 failed transmissions. The PRN sequence has an odd number of elements to further eliminate patterns.
- The 5 random garbage numbers are there to make pattern recognition even more difficult. Unless you KNOW they're garbage, you must treat them like the other 3 useful bytes.

Calculations.
Suppose a PRN sequence is 2,047 numbers long. Because the number of elements is odd and we are taking the subsequences two at a time, it would take 4,094 transmissions before the pattern started over. Assuming someone used their RVI three times a day, and a bad guy was able to intercept every single transmission, it would still be 3 ¾ years*

before a pattern could begin to be recognizable. And that is provided he knew to disregard the garbage in the packet.

$$\frac{4094\,transmissions}{3\,transmissions/day} = 1364.7\,days \qquad \frac{1364.7\,days}{365\,days/year} \approx 3.74\,years$$

If a bad guy found a way to transmit packets to the vehicle receiver, each byte can have 256 distinct values. Given 8 bytes in a packet, that would yield 256^8 (or 1.84467×10^{19}), possible combinations. One of those is the command to unlock the doors and one of them is the command to start the engine. At 76kbps (9500 bytes/sec.), it would take 1.94176×10^{15} seconds to send all possible commands. This translates to:

$$1.9417 \times 10^{15}\,s \times \frac{1\,\text{min}}{60\,s} \times \frac{1\,hour}{60\,\text{min}} \times \frac{1\,day}{24\,hours} \times \frac{1\,year}{365\,days} = 6.15729 \times 10^{7}\ years$$

a.k.a. “Not very effective”

Appendix B – WebUI Source Code

Start member.aspx

```
<%@ Page language="c#" Codebehind="member.aspx.cs"
AutoEventWireup="false" Inherits="RVIweb.Member" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
<HEAD>
<title>RVI Project</title>
<meta name="GENERATOR" Content="Microsoft Visual Studio .NET 7.1">
<meta name="CODE_LANGUAGE" Content="C#">
<meta name="vs_defaultClientScript" content="JavaScript">
<meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
</HEAD>
<body>
<form id="Form1" method="post" runat="server">
<map id="headmap" name="headmap">
<area shape="RECT" coords="467,115,589,127" href="http://www.utah.edu"
target="_blank"
alt="U of U">
</map>
<div align="center"><img src="/Images/heading.jpg" border="0"
usemap="#headmap"></div>
<div align="center"><img src="/Images/horiz_divider.jpg"
border="0"></div>
<br>
<table width="600" align="center" cellpadding="5" cellspacing="0">
<tr>
<td valign="top" width="130">
<a href="index.htm"><img src="/Images/button_home.jpg"
border="0"></a><br>
<asp:ImageButton ImageUrl="./Images/button_logout.jpg" Runat="server"
id="logoutButton"></asp:ImageButton><br>
<a href="details.htm"><img src="/Images/button_details.jpg"
border="0"></a><br>
<a href="contact.htm"><img src="/Images/button_contact.jpg"
border="0"></a>
</td>
<td valign="top">
<asp:Label Runat="server" id="Label1"></asp:Label>
<hr width="400" color="#ff0000" align="left">
<br>
<font size="+1" face="Arial">Engine commands:<br>
     
<asp:ImageButton ImageUrl="./Images/button_start.jpg" Runat="server"
ID="startButton"></asp:ImageButton>
<asp:ImageButton ImageUrl="./Images/button_kill.jpg" Runat="server"
ID="killButton"></asp:ImageButton>
<br>
Lock Commands:<br>
     
<asp:ImageButton ImageUrl="./Images/button_unlock.jpg" Runat="server"
ID="unlockButton"></asp:ImageButton>
```

```
<asp:ImageButton ImageUrl="./Images/button_lock.jpg" Runat="server"
ID="lockButton"></asp:ImageButton>
<asp:ImageButton ImageUrl="./Images/button_pop.jpg" Runat="server"
ID="trunkButton"></asp:ImageButton>
</font>
</td>
</tr></table></form></body></HTML>
```

Start member.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace RVIweb
{
      /// <summary>
      /// Summary description for Member.
      /// </summary>
      public class Member : System.Web.UI.Page
      {
            protected System.Web.UI.WebControls.ImageButton
ImageButton1;
            protected System.Web.UI.WebControls.ImageButton
Imagebutton3;
            protected System.Web.UI.WebControls.ImageButton
Imagebutton4;
            protected System.Web.UI.WebControls.Label Label1;
            protected System.Web.UI.WebControls.ImageButton
startButton;
            protected System.Web.UI.WebControls.ImageButton killButton;
            protected System.Web.UI.WebControls.ImageButton
unlockButton;
            protected System.Web.UI.WebControls.ImageButton lockButton;
            protected System.Web.UI.WebControls.ImageButton
logoutButton;
            protected System.Web.UI.WebControls.ImageButton
trunkButton;

            private void Page_Load(object sender, System.EventArgs e)
            {
                  // Put user code to initialize the page here

                  //Response.CacheControl = "no-cache";
                  //Response.AddHeader("Pragma", "no-cache");
                  //Response.Expires = -1;
                  this.Label1.Text = "";
```

```
                  if(crypt(Session[crypt("isAuth")].ToString()) ==
"false")
                  {
                        Response.Redirect("authenticate.aspx");
                  }
                  else
                  {
                        OleDbDataReader dr =
DBhelpers.getRecord("SELECT * FROM Clients Where UserName='" +
crypt(Request.Cookies.Get("USERNAME").Value) + "'");
                        if(dr != null && dr.HasRows)
                        {
                              // Get only record
                              dr.Read();

                              this.Label1.Text += "<font size=\"+3\"
face=\"Arial\">" + dr.GetString(dr.GetOrdinal("FirstName")) + " " +
dr.GetString(dr.GetOrdinal("LastName")) + "</font>";
                              this.Label1.Text += "<br />";
                              this.Label1.Text += "<font
face=\"Arial\">" + dr.GetString(dr.GetOrdinal("Email")) + "</font><br
/>";

                              // Closing the dataReader object also
closes the connection
                              dr.Close();
                        }
                  }
            }

            #region Web Form Designer generated code
            override protected void OnInit(EventArgs e)
            {
                  //
                  // CODEGEN: This call is required by the ASP.NET Web
Form Designer.
                  //
                  InitializeComponent();
                  base.OnInit(e);
            }

            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
                  this.logoutButton.Click += new
System.Web.UI.ImageClickEventHandler(this.logoutButton_Click);
                  this.startButton.Click += new
System.Web.UI.ImageClickEventHandler(this.startButton_Click);
                  this.killButton.Click += new
System.Web.UI.ImageClickEventHandler(this.killButton_Click);
                  this.unlockButton.Click += new
System.Web.UI.ImageClickEventHandler(this.unlockButton_Click);
                  this.lockButton.Click += new
System.Web.UI.ImageClickEventHandler(this.lockButton_Click);
```

```
                this.trunkButton.Click += new
System.Web.UI.ImageClickEventHandler(this.trunkButton_Click);
                this.Load += new System.EventHandler(this.Page_Load);

            }
            #endregion

            private void startButton_Click(object sender,
System.Web.UI.ImageClickEventArgs e)
            {
                  this.buttonsHelper('S');
            }

            private void killButton_Click(object sender,
System.Web.UI.ImageClickEventArgs e)
            {
                  this.buttonsHelper('K');
            }

            private void unlockButton_Click(object sender,
System.Web.UI.ImageClickEventArgs e)
            {
                  this.buttonsHelper('U');
            }

            private void lockButton_Click(object sender,
System.Web.UI.ImageClickEventArgs e)
            {
                  this.buttonsHelper('L');
            }

            private void trunkButton_Click(object sender,
System.Web.UI.ImageClickEventArgs e)
            {
                  this.buttonsHelper('T');
            }

            private void logoutButton_Click(object sender,
System.Web.UI.ImageClickEventArgs e)
            {
                  Session[crypt("isAuth")] = crypt("false");
                  Response.Cookies.Get("USERNAME").Value = "";
                  Response.Redirect("index.htm");
            }

            private void buttonsHelper(char cmd)
            {
                  OleDbDataReader dr = DBhelpers.getRecord("SELECT *
FROM Clients Where UserName='" +
crypt(Request.Cookies.Get("USERNAME").Value) + "'");
                  if(dr.HasRows && dr != null)
                  {
                        dr.Read();
                  }
                  else
                  {
                        return;
```

```
                }

                PRNG p = new PRNG((uint)dr.GetInt32(0));
                int offset = dr.GetInt32(dr.GetOrdinal("Offset"));
                byte[] temp = p.get255();
                Random r = new Random();
                CommPort c1 = new CommPort();
                c1.setBaudAndOpen(9600);

                byte[] toSend = {(byte)cmd, (byte)r.Next(256),
temp[offset], temp[(offset+1)%255], (byte)r.Next(256)};

                c1.Write(toSend);
                c1.Close();

                dr.Close();

                offset = (offset+2)%255;

                //this.Label1.Text =
                DBhelpers.updateOffset(offset,
crypt(Request.Cookies.Get("USERNAME").Value));
            }

            private string crypt(string str)
            {
                return Crypt.swapCrypt(str);
            }
      }
}
```

Start DBhelpers.cs

```
using System;
using System.Data;
using System.Data.OleDb;

namespace RVIweb
{
      /// <summary>
      /// Summary description for DBhelpers.
      /// </summary>
      public class DBhelpers
      {
            private static string CONN_STRING =
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:/Inetpub/Data/RVI.mdb";

            public DBhelpers()
            {
                //
                // TODO: Add constructor logic here
                //
            }

            public static OleDbDataReader getRecord(string sqlQuery)
            {
                OleDbConnection Connection = new OleDbConnection();
```

```
                  try
                  {
                        // Open a connection to the database
                        Connection.ConnectionString = CONN_STRING;
                        Connection.Open();

                        // Create an OleDb command,
                        OleDbCommand command = new OleDbCommand();
                        command.Connection = Connection;
                        command.CommandText = sqlQuery;

                        // Execute and return the rows in the data
reader object
                        OleDbDataReader dataReader;
                        dataReader =
command.ExecuteReader(CommandBehavior.CloseConnection);

                        return dataReader;
                  }
                  catch(Exception ex)
                  {
                        Console.WriteLine(ex.Message + "\n");
                        Console.WriteLine(ex.StackTrace);
                        return null;
                  }
            }

            public static void updateOffset(int offset, string
username)
            {
                  OleDbConnection Connection = new OleDbConnection();
                  try
                  {
                        // Open a connection to the database
                        Connection.ConnectionString = CONN_STRING;
                        Connection.Open();

                        // Create an OleDb command,
                        OleDbCommand command = new OleDbCommand();
                        command.Connection = Connection;
                        command.CommandText = "UPDATE Clients SET
Offset='" + offset +"' WHERE UserName='" + username + "'";

                        // Execute and return the rows in the data
reader object
                        command.ExecuteNonQuery();

                        Connection.Close();

                        //return "None";
                  }
                  catch(Exception ex)
                  {
                        //return (ex.Message + "<br />" +
ex.StackTrace);
                  }
            }
```

```
	}
}
```

Start PRNG.cs

```
using System;
namespace RVIweb
{
	/// <summary>
	/// Summary description for PRNG.
	/// </summary>
	class PRNG
	{
		private uint SEED;

		/* uncomment parameters of choice */
		private uint a = 1588635695, m = 4294967291, q = 2, r =
1117695901;
		// static unsigned int a = 1223106847, m = 4294967291U, q =
3, r = 625646750;
		// static unsigned int a = 279470273, m = 4294967291U, q =
15, r = 102913196;
		// static unsigned int a = 1583458089, m = 2147483647, q =
1, r = 564025558;
		// static unsigned int a = 784588716, m = 2147483647, q =
2, r = 578306215;
		// static unsigned int a = 16807, m = 2147483647, q =
127773, r = 2836;
		// static unsigned int a = 950706376, m = 2147483647, q =
2, r = 246070895;

		public PRNG(uint s)
		{
			SEED = s;
		}
		public byte random()
		{
			double temp;

			SEED = a*(SEED % q) - r*(SEED / q);
			temp = ((double)SEED / (double)m);

			return (byte) ((double)temp * ((double)(256)));
		}
		public byte[] get255()
		{
			byte[] result = new byte[255];

			for(int i = 0; i < 255; i++)
			{
				result[i] = this.random();
			}

			return result;
		}
	}
}
```

Appendix C – PhoneUI Source Code

Start Form1.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.OleDb;
using System.Threading;

namespace RVI_PhoneUI
{
	public class Form1 : System.Windows.Forms.Form
	{
		internal System.Windows.Forms.TextBox Txtlog;
		public System.Windows.Forms.ToolTip ToolTip1;
		public System.Windows.Forms.Button BnClose;
		public System.Windows.Forms.Button BnOpen;
		public System.Windows.Forms.ListBox LtLines;
		public System.Windows.Forms.TextBox EdAutoAnswerRingCount;
		public System.Windows.Forms.Label Label1;
		public System.Windows.Forms.Label Label3;
		private AxTAPIEXLib.AxTAPIExCtl mTAPIEx;
		private System.ComponentModel.IContainer components;

		private string clientNumber;
		private string clientPassword;

		enum Mode
		{
			WELCOME = 0,
			PASSWORD,
			AUTHENTICATED,
			GOODBYE
		};

		public Form1()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after
InitializeComponent call
			//
		}

		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
```

```
                        if (components != null)
                        {
                              components.Dispose();
                        }
                  }
                  base.Dispose( disposing );
            }

            #region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
                  this.components = new
System.ComponentModel.Container();
                  System.Resources.ResourceManager resources = new
System.Resources.ResourceManager(typeof(Form1));
                  this.Txtlog = new System.Windows.Forms.TextBox();
                  this.ToolTip1 = new
System.Windows.Forms.ToolTip(this.components);
                  this.BnClose = new System.Windows.Forms.Button();
                  this.BnOpen = new System.Windows.Forms.Button();
                  this.LtLines = new System.Windows.Forms.ListBox();
                  this.EdAutoAnswerRingCount = new
System.Windows.Forms.TextBox();
                  this.Label1 = new System.Windows.Forms.Label();
                  this.Label3 = new System.Windows.Forms.Label();
                  this.mTAPIEx = new AxTAPIEXLib.AxTAPIExCtl();

      ((System.ComponentModel.ISupportInitialize)(this.mTAPIEx)).BeginI
nit();
                  this.SuspendLayout();
                  //
                  // Txtlog
                  //
                  this.Txtlog.Location = new System.Drawing.Point(7,
88);
                  this.Txtlog.Multiline = true;
                  this.Txtlog.Name = "Txtlog";
                  this.Txtlog.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
                  this.Txtlog.Size = new System.Drawing.Size(409, 224);
                  this.Txtlog.TabIndex = 15;
                  this.Txtlog.Text = "";
                  //
                  // BnClose
                  //
                  this.BnClose.BackColor =
System.Drawing.SystemColors.Control;
                  this.BnClose.Cursor =
System.Windows.Forms.Cursors.Default;
                  this.BnClose.Enabled = false;
                  this.BnClose.ForeColor =
System.Drawing.SystemColors.ControlText;
```

```
                this.BnClose.Location = new System.Drawing.Point(336,
48);
                this.BnClose.Name = "BnClose";
                this.BnClose.RightToLeft =
System.Windows.Forms.RightToLeft.No;
                this.BnClose.Size = new System.Drawing.Size(80, 25);
                this.BnClose.TabIndex = 14;
                this.BnClose.Text = "Close All";
                this.BnClose.Click += new
System.EventHandler(this.BnClose_Click);
                //
                // BnOpen
                //
                this.BnOpen.BackColor =
System.Drawing.SystemColors.Control;
                this.BnOpen.Cursor =
System.Windows.Forms.Cursors.Default;
                this.BnOpen.DialogResult =
System.Windows.Forms.DialogResult.Cancel;
                this.BnOpen.ForeColor =
System.Drawing.SystemColors.ControlText;
                this.BnOpen.Location = new System.Drawing.Point(336,
16);
                this.BnOpen.Name = "BnOpen";
                this.BnOpen.RightToLeft =
System.Windows.Forms.RightToLeft.No;
                this.BnOpen.Size = new System.Drawing.Size(80, 25);
                this.BnOpen.TabIndex = 13;
                this.BnOpen.Text = "Open All";
                this.BnOpen.Click += new
System.EventHandler(this.BnOpen_Click);
                //
                // LtLines
                //
                this.LtLines.BackColor =
System.Drawing.SystemColors.Window;
                this.LtLines.Cursor =
System.Windows.Forms.Cursors.Default;
                this.LtLines.ForeColor =
System.Drawing.SystemColors.WindowText;
                this.LtLines.Location = new System.Drawing.Point(8,
24);
                this.LtLines.Name = "LtLines";
                this.LtLines.RightToLeft =
System.Windows.Forms.RightToLeft.No;
                this.LtLines.Size = new System.Drawing.Size(320, 17);
                this.LtLines.TabIndex = 11;
                //
                // EdAutoAnswerRingCount
                //
                this.EdAutoAnswerRingCount.AcceptsReturn = true;
                this.EdAutoAnswerRingCount.AutoSize = false;
                this.EdAutoAnswerRingCount.BackColor =
System.Drawing.SystemColors.Window;
                this.EdAutoAnswerRingCount.Cursor =
System.Windows.Forms.Cursors.IBeam;
```

```
                this.EdAutoAnswerRingCount.ForeColor =
System.Drawing.SystemColors.WindowText;
                this.EdAutoAnswerRingCount.Location = new
System.Drawing.Point(304, 48);
                this.EdAutoAnswerRingCount.MaxLength = 0;
                this.EdAutoAnswerRingCount.Name =
"EdAutoAnswerRingCount";
                this.EdAutoAnswerRingCount.RightToLeft =
System.Windows.Forms.RightToLeft.No;
                this.EdAutoAnswerRingCount.Size = new
System.Drawing.Size(25, 19);
                this.EdAutoAnswerRingCount.TabIndex = 9;
                this.EdAutoAnswerRingCount.Text = "1";
                //
                // Label1
                //
                this.Label1.BackColor =
System.Drawing.SystemColors.Control;
                this.Label1.Cursor =
System.Windows.Forms.Cursors.Default;
                this.Label1.ForeColor =
System.Drawing.SystemColors.ControlText;
                this.Label1.Location = new System.Drawing.Point(8,
8);
                this.Label1.Name = "Label1";
                this.Label1.RightToLeft =
System.Windows.Forms.RightToLeft.No;
                this.Label1.Size = new System.Drawing.Size(113, 17);
                this.Label1.TabIndex = 12;
                this.Label1.Text = "Device";
                //
                // Label3
                //
                this.Label3.BackColor =
System.Drawing.SystemColors.Control;
                this.Label3.Cursor =
System.Windows.Forms.Cursors.Default;
                this.Label3.ForeColor =
System.Drawing.SystemColors.ControlText;
                this.Label3.Location = new System.Drawing.Point(168,
48);
                this.Label3.Name = "Label3";
                this.Label3.RightToLeft =
System.Windows.Forms.RightToLeft.No;
                this.Label3.Size = new System.Drawing.Size(136, 17);
                this.Label3.TabIndex = 10;
                this.Label3.Text = "Auto Answer after Ring";
                //
                // mTAPIEx
                //
                this.mTAPIEx.Enabled = true;
                this.mTAPIEx.Location = new System.Drawing.Point(8,
48);
                this.mTAPIEx.Name = "mTAPIEx";
                this.mTAPIEx.OcxState =
((System.Windows.Forms.AxHost.State)(resources.GetObject("mTAPIEx.OcxSt
ate")));
```

```
                this.mTAPIEx.Size = new System.Drawing.Size(30, 30);
                this.mTAPIEx.TabIndex = 16;
                this.mTAPIEx.Visible = false;
                this.mTAPIEx.OnDTMFTimeOut += new
AxTAPIEXLib._ITAPIExEvents_OnDTMFTimeOutEventHandler(this.mTAPIEx_OnDTM
FTimeOut);
                this.mTAPIEx.OnDisConnected += new
AxTAPIEXLib._ITAPIExEvents_OnDisConnectedEventHandler(this.mTAPIEx_OnDi
sConnected);
                this.mTAPIEx.OnPlayBackComplete += new
AxTAPIEXLib._ITAPIExEvents_OnPlayBackCompleteEventHandler(this.mTAPIEx_
OnPlayBackComplete);
                this.mTAPIEx.OnDebug += new
AxTAPIEXLib._ITAPIExEvents_OnDebugEventHandler(this.mTAPIEx_OnDebug);
                this.mTAPIEx.OnDTMF += new
AxTAPIEXLib._ITAPIExEvents_OnDTMFEventHandler(this.mTAPIEx_OnDTMF);
                this.mTAPIEx.OnConnected += new
AxTAPIEXLib._ITAPIExEvents_OnConnectedEventHandler(this.mTAPIEx_OnConne
cted);
                this.mTAPIEx.OnRing += new
AxTAPIEXLib._ITAPIExEvents_OnRingEventHandler(this.mTAPIEx_OnRing);
                //
                // Form1
                //
                this.AutoScaleBaseSize = new System.Drawing.Size(6,
14);
                this.ClientSize = new System.Drawing.Size(424, 333);
                this.Controls.Add(this.mTAPIEx);
                this.Controls.Add(this.BnClose);
                this.Controls.Add(this.BnOpen);
                this.Controls.Add(this.LtLines);
                this.Controls.Add(this.EdAutoAnswerRingCount);
                this.Controls.Add(this.Label1);
                this.Controls.Add(this.Label3);
                this.Controls.Add(this.Txtlog);
                this.Font = new System.Drawing.Font("Verdana", 8.25F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                this.Name = "Form1";
                this.Text = "RVI PhoneUI";
                this.Load += new
System.EventHandler(this.Form1_Load);

      ((System.ComponentModel.ISupportInitialize)(this.mTAPIEx)).EndIni
t();
                this.ResumeLayout(false);

            }
            #endregion

            [STAThread]
            static void Main()
            {
                  Application.Run(new Form1());
            }

            private void Form1_Load(object sender, System.EventArgs e)
```

```
            {
                  mTAPIEx.initialize(); //Initialize
                  //for(int i=0;i<mTAPIEx.Lines.Count;i++)
                  //{
                  //
      LtLines.Items.Add(mTAPIEx.Lines.Item(i).DeviceName);
                  //}
                  LtLines.Items.Add(mTAPIEx.Lines.Item(0).DeviceName);
                  this.Txtlog.Text += "Line Caps: " +
mTAPIEx.Lines.Item(0).Caps.Call_Features_Str + "\r\n";

                  BnOpen_Click(null, null);
            }

            private void BnOpen_Click(object sender, System.EventArgs
e)
            {
                  //for(int i=0;i<mTAPIEx.Lines.Count;i++)
                  //
      mTAPIEx.Lines.Item(i).Open(TAPIEXLib.LINECALLPRIVILEGE.CALLPRIVIL
EGE_OWNER);

      //mTAPIEx.Lines.Item(0).Open(TAPIEXLib.LINECALLPRIVILEGE.CALLPRIV
ILEGE_OWNER);
                  mTAPIEx.Lines.Item(0).Open();
                  BnOpen.Enabled = false;
                  BnClose.Enabled = !BnOpen.Enabled;
            }

            private void BnClose_Click(object sender, System.EventArgs
e)
            {
                  mTAPIEx.CloseAll();
                  BnOpen.Enabled = true;
                  BnClose.Enabled = !BnOpen.Enabled;
            }

            private void mTAPIEx_OnDebug(object sender,
AxTAPIEXLib._ITAPIExEvents_OnDebugEvent e)
            {
                  this.Txtlog.Text = Txtlog.Text + e.msg +"\r\n";
                  this.Txtlog.SelectionStart = this.Txtlog.Text.Length;
                  this.Txtlog.ScrollToCaret();
            }

            private void mTAPIEx_OnRing(object sender,
AxTAPIEXLib._ITAPIExEvents_OnRingEvent e)
            {
                  int idefringcount =
System.Convert.ToInt32(EdAutoAnswerRingCount.Text);
                  if(e.ringCount >= idefringcount)
                  {
                        e.m_Call.Answer();
                  }
            }
```

```
            private void mTAPIEx_OnDTMFTimeOut(object sender,
AxTAPIEXLib._ITAPIExEvents_OnDTMFTimeOutEvent e)
            {
                  e.m_Call.DTMF_TimeOut = 0; //stop DTMF time out
                  e.m_Call.StopPlayBack();
                  e.m_Call.PlaybackFile("timeout.wav");
            }

            private void mTAPIEx_OnConnected(object sender,
AxTAPIEXLib._ITAPIExEvents_OnConnectedEvent e)
            {
                  TAPIEXLib.TAPICall c = e.m_Call;

                  this.clientPassword = "";
                  this.clientNumber = "";
                  c.User_Data1 = Mode.WELCOME;
                  c.PlaybackFile("C:\\wav\\welcome.wav");
            }

            private void mTAPIEx_OnDTMF(object sender,
AxTAPIEXLib._ITAPIExEvents_OnDTMFEvent e)
            {
                  TAPIEXLib.TAPICall c = e.m_Call;
                  string temp = c.DigitsReceived;

                  if((Mode)c.User_Data1 != Mode.GOODBYE)
                  {
                        c.StopPlayBack(); //Stop the current
menu/prompt
                  }

                  switch((Mode)c.User_Data1)
                  {
                        case Mode.WELCOME:
                              if(temp.Length == 10)
                              {
                                    this.clientNumber = temp;
                                    c.User_Data1 = Mode.PASSWORD;

      c.PlaybackFile("C:\\wav\\password.wav");
                              }
                              break;

                        case Mode.PASSWORD:
                              if(temp.Length == 14)
                              {
                                    this.clientPassword =
(""+temp[10]+temp[11]+temp[12]+temp[13]);
                                    if(this.isAuth())
                                    {
                                          c.User_Data1 =
Mode.AUTHENTICATED;

      c.PlaybackFile("C:\\wav\\menuTop.wav");
                                    }
                                    else //i.e. NOT Authenticated
                                    {
```

```
                                    c.User_Data1 = Mode.GOODBYE;

      c.PlaybackFile("C:\\wav\\authError.wav");
                                    }
                              }
                              break;

                        case Mode.AUTHENTICATED:
                              if(temp[temp.Length-1] == '1')
                              {
                                    sendCommand('S');
                              }
                              else if(temp[temp.Length-1] == '2')
                              {
                                    sendCommand('S');
                              }
                              else if(temp[temp.Length-1] == '3')
                              {
                                    sendCommand('L');
                              }
                              else if(temp[temp.Length-1] == '4')
                              {
                                    sendCommand('U');
                              }
                              else if(temp[temp.Length-1] == '5')
                              {
                                    c.User_Data1 = Mode.GOODBYE;

      c.PlaybackFile("C:\\wav\\goodbye.wav");
                              }
                              break;

                        case Mode.GOODBYE:
                              break;
                  }
            }

            private void mTAPIEx_OnPlayBackComplete(object sender,
AxTAPIEXLib._ITAPIExEvents_OnPlayBackCompleteEvent e)
            {
                  string temp = e.m_Call.ActivePlayBackFile;
                  if(temp == "C:\\wav\\authError.wav" || temp ==
"C:\\wav\\goodbye.wav")
                  {
                        e.m_Call.Drop();
                  }
            }

            private void mTAPIEx_OnDisConnected(object sender,
AxTAPIEXLib._ITAPIExEvents_OnDisConnectedEvent e)
            {
                  e.m_Call.Drop();
            }
            private bool isAuth()
            {
```

```
                  OleDbDataReader dr = DB.getRecord("SELECT * FROM
Clients WHERE Phone=" + this.clientNumber + " AND Pin=" +
this.clientPassword);
                  if(dr != null)
                  {
                        dr.Close();
                        return true;
                  }
                  else
                  {
                        return false;
                  }
            }

            private bool sendCommand(char cmd)
            {
                  OleDbDataReader dr = DB.getRecord("SELECT * FROM
Clients WHERE Phone=" + this.clientNumber);
                  if(dr != null)
                  {
                        dr.Read();
                  }
                  else
                  {
                        return false;
                  }

                  PRNG p = new PRNG((uint)dr.GetInt32(0));
                  int offset = dr.GetInt32(dr.GetOrdinal("Offset"));
                  byte[] temp = p.get255();
                  Random r = new Random();
                  CommPort c1 = new CommPort();
                  c1.setBaudAndOpen(9600);

                  byte[] toSend = {(byte)cmd, (byte)r.Next(256),
temp[offset], temp[(offset+1)%255], (byte)r.Next(256)};

                  c1.Write(toSend);
                  c1.Close();

                  dr.Close();

                  offset = (offset+2)%255;

                  DB.updateOffset(offset, this.clientNumber);

                  return true;
            }
      }
}
```

Appendix D – Microcontroller Source Code

Start main.c

```
#include <hidef.h>            /* for EnableInterrupts macro */
#include <MC9S08GB60.h>  /* include peripheral declarations */

#include "INT.h"
#include "rands.inc"

// Function prototypes
void SCI2_init(void);
Bool checkAuth(void);
void delay(int);

// Variables
ulong RXtimeout;
byte buffer[5];
byte bufCount;
byte offset;

// Interrupt Handlers ------------------------------------------
interrupt Vsci2rx void onSCI2_RX(){
      if(SCI2S1_RDRF == 1 && bufCount < 5) {
            RXtimeout = 5000;
            buffer[bufCount++] = SCI2D;
      }
}

// Main Function -----------------------------------------------
void main(void) {
      EnableInterrupts; //assembly macro

      ICGC1_CLKS = 0; // set ICG to SCM mode (4MHz bus, 8Mhz clock)
      SOPT_COPE = 0;  // disable COP
      PTFDD = 0xF;      // direct channels 3 thru 0 to data-register
      PTFD  = 0x0;      // ch3 thru ch0 = 0   (1 = ~80mV. 0 = ~3V)

      SCI2_init();

      RXtimeout = 0;
      bufCount  = 0;
      offset    = 254;

      for(;;) {

            //PTFD_PTFD3 = 0;

            while(RXtimeout > 0) {
                  if(bufCount == 5) {
                        // handle received command ------
                        if(checkAuth()) {
                              switch((char)buffer[0]) {
                                    case 'L':
                                          PTFD_PTFD0 = !PTFD_PTFD0;
                                          delay(2);
```

```
                                    PTFD_PTFD0 = !PTFD_PTFD0;
                                    break;
                              case 'U':
                                    PTFD_PTFD1 = !PTFD_PTFD1;
                                    delay(10);
                                    PTFD_PTFD1 = !PTFD_PTFD1;
                                    break;

                              case 'S':
                                    PTFD_PTFD2 = !PTFD_PTFD2;
                                    delay(2);
                                    PTFD_PTFD2 = !PTFD_PTFD2;
                                    delay(2);
                                    PTFD_PTFD2 = !PTFD_PTFD2;
                                    delay(2);
                                    PTFD_PTFD2 = !PTFD_PTFD2;
                                    break;
                              case 'K':
                                    PTFD_PTFD3 = !PTFD_PTFD3;
                                    delay(2);
                                    PTFD_PTFD3 = !PTFD_PTFD3;
                                    break;
                        }
                  }
                  //-------------------------------
                  bufCount = 0;
            } else {
                  RXtimeout--;
            }
      }// end while loop

      bufCount = 0;

      //PTFD_PTFD3 = 1;

      // Go to sleep
      asm(WAIT);

  } // loop forever
}

// Helper & Init Functions -------------------------------------
// Checks received data in the buffer for authentication
Bool checkAuth(void) {
  int i;
  int end = ((int)offset)+64;

  for(i = offset; i < end; i+=2) {
      if(authNums[i%255] == buffer[2] && authNums[(i+1)%255] ==
buffer[3]) {
            offset = (byte)((i+2) % 255);
            return 1;
      }
  }
  return 0;
}
```

```
//Delay Function
void delay(int half_seconds) {
      ulong scale = half_seconds * 10000;
      int i;
      for(i = 0; i < scale; i++) {
        //do nothing
        __RESET_WATCHDOG();
      }
      return;
}

// Initializes the SCI2 port
void SCI2_init(void) {
      SCI2BDH = 0;      //
      SCI2BDL = 26;     // Sets BAUD rate to 4e6/(16 * _26_) ~= 9600

      SCI2C1 = 0b00001100;
      /*              ||||||||
                      |||||||+- No meaning because parity disabled
(0=even parity)
                      ||||||+-- Parity Disabled
                      |||||+--- Idle character bit count starts after
_stop_ bit
                      ||||+---- Address-mark wakeup
                      |||+----- 8-bit mode
                      ||+------ No meaning, since LOOPS=0
                      |+------- SCI clocks run in wait mode
                      +-------- Not Looped
      */
      SCI2C2 = 0b00101100;
      /*              ||||||||
                      |||||||+- Send Break (0=normal operation)
                      ||||||+-- Receiver wakeup control (0=normal
operation)
                      |||||+--- RX enabled
                      ||||+---- TX enabled
                      |||+----- Idle line interrupt disabled
                      ||+------ RX interrupt enabled
                      |+------- TX-complete interrupt dsiabled
                      +-------- TX interrupt disabled
      */
      SCI2C3 = 0b00000000;
      /*              ||||||||
                      |||||||+- Parity error interrupt disabled
                      ||||||+-- Framing error interrupt disabled
                      |||||+--- Noise error interrupt disabled
                      ||||+---- Overrun interrupt disabled
                      |||+----- 0 --> _no use_
                      ||+------ TXDIR
                      |+------- T8
                      +-------- R8
      */

      PTCDD_PTCDD1 = 0;
      PTCPE_PTCPE1 = 1;
}
```

Start rands.inc

```
byte authNums[] =
      {94, 31, 210, 176, 44, 220, 129, 166, 90, 188, 163, 23, 173, 225,
244, 54, 194,
      243, 254, 20, 180, 6, 65, 147, 126, 19, 30, 123, 125, 17, 151,
77, 106, 202, 163,
      104, 157, 61, 95, 66, 190, 68, 36, 215, 172, 240, 65, 16, 153,
178, 81, 37, 168,
      151, 18, 144, 63, 188, 11, 63, 10, 48, 88, 187, 34, 185, 76, 82,
175, 49, 21, 167,
      239, 160, 21, 231, 68, 103, 100, 3, 85, 163, 45, 176, 170, 8, 38,
103, 165, 193,
      194, 207, 102, 216, 187, 198, 184, 47, 151, 193, 106, 125, 20,
82, 193, 119, 206,
      191, 206, 36, 149, 233, 128, 109, 114, 143, 46, 176, 176, 215,
196, 23, 78, 139,
      61, 76, 246, 147, 207, 11, 166, 163, 17, 228, 6, 144, 8, 73, 202,
154, 135, 12, 20,
      177, 134, 178, 62, 220, 118, 98, 245, 100, 168, 251, 158, 57,
223, 226, 69, 159,
      97, 203, 25, 179, 80, 194, 229, 108, 51, 177, 130, 101, 190, 105,
212, 216, 250,
      216, 153, 92, 195, 148, 223, 118, 186, 116, 115, 235, 241, 80,
224, 18, 78, 62,
      154, 111, 105, 156, 193, 39, 115, 182, 215, 245, 177, 130, 173,
129, 239, 189, 7,
      119, 218, 103, 228, 58, 184, 187, 46, 72, 83, 188, 4, 127, 105,
77, 195, 219, 110,
      232, 203, 216, 195, 138, 171, 236, 58, 150, 169, 132, 68, 232,
52, 168, 171, 33,
      246, 4, 24, 160, 138, 207, 169, 64, 123};
```

Appendix E – Block Diagrams

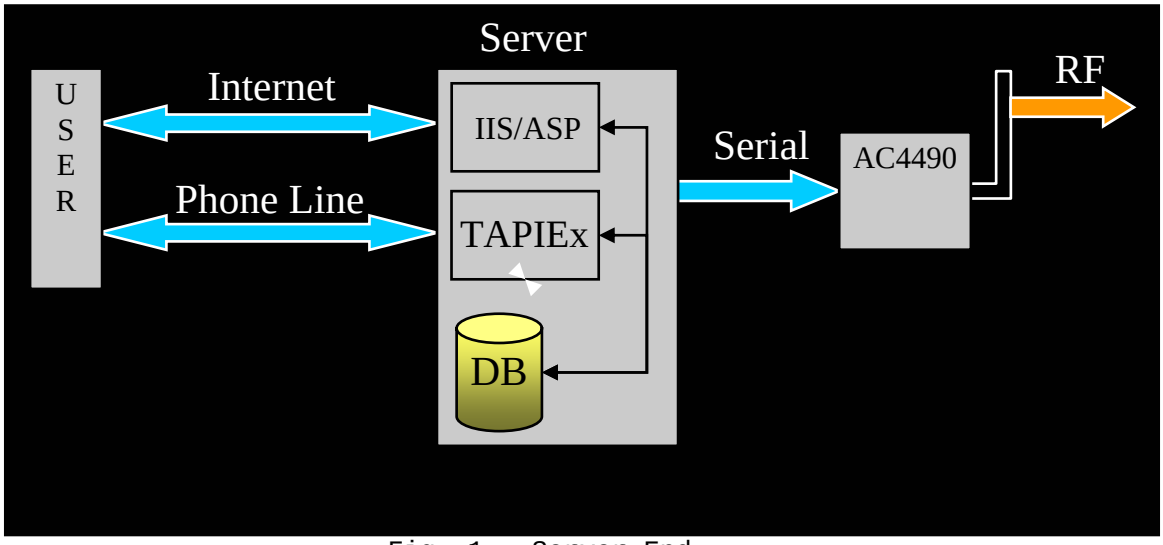

Fig. 1 - Server End

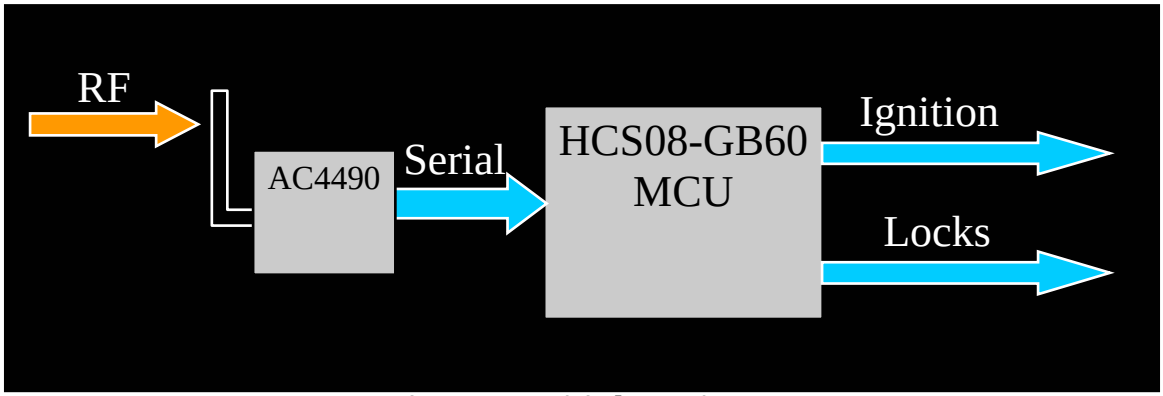

Fig. 2 – Vehicle End