Remote Car Controller

Capstone Design 2
ECE U792

April 20, 2007

Steve Yu
Jason Bergeron
Dennis Cuccaro
David Femino
Raul Insignares
# Table of Contents

- INTRODUCTION AND RELATED WORK ................................................................. 3
- DESIGN SPECIFICATION .................................................................................. 6
- DESIGN OVERVIEW .......................................................................................... 8
  - The Graphical User Interface (GUI) ................................................................. 8
  - The Cell Phone ............................................................................................... 8
  - The BieneRemote128GM Module ................................................................. 9
  - Door Locks .................................................................................................... 9
  - Vehicle Starter ............................................................................................. 10
  - Temperature Sensor .................................................................................... 11
- DESIGN DETAIL ............................................................................................... 12
- DESIGN EVALUATION .................................................................................... 14
- PARTS .............................................................................................................. 16
- REFERENCES ................................................................................................. 18
- APPENDIX A .................................................................................................. 19
- APPENDIX B ................................................................................................. 20
- APPENDIX C ................................................................................................. 26
Introduction and Related Work

Cell phones are becoming an essential part of modern culture these days. Today’s cell phones include different types of features from receiving calls, taking videos/pictures, listening to music, and even surfing the internet. Even though a cell phone provides these functions, cell phone companies continue to add more features.

A typical person carries with them the following items: a cell phone, a wallet and keys. The consumer is attracted to products that provide multiple services. It is usually the cell phone that is the item selected to integrate a variety of services.

There are companies around the world that are working to bring this dream true. MobileLime [3], an American company, is providing users the ability to use their cell phone as a credit card. Another product created by Apple and Motorola integrates Apple’s iTunes, an mp3 media program, onto their cell phone [4].

Nobody has tried to provide a technological solution that will allow a person to use their cell phone to start their vehicle anywhere. We want to provide the user the ability to carry fewer devices without losing functionality of the included devices.

Our product includes a user friendly interface (GUI). The GUI is not only simple in design but small in size as to not occupy all the memory of a cell phone. Our program provides the user a service which monitors the most important aspects of a vehicle.

The main feature we wanted to provide the user is simplicity. We wanted to design a product that is easy to use, easy to maintain, and easily installed in any type of vehicle. We are providing an inexpensive system which will only cost around $600. Since the control board is 75% of the total cost of the product, reducing the cost of the board will lower the overall product cost. The only lasting cost requirement will be the monthly SMS text messaging service through their cell phone provider. Most cell phone companies do not charge more than 5 dollars per month for unlimited SMS text messaging services [5].
OnStar provides two different services that cost $200 and $400 per year. OnStar’s main features are emergency service and roadside assistance [2]. They also provide the user with remote door unlocks and some vehicle diagnosis. To gain access to these services the customer needs to call OnStar directly. OnStar has to then find the user’s information from their computers.

OnStar has two major drawbacks that our product improves on, price and compatibility. Their services have a yearly fee; our product has only a one time fee, plus a prepaid SIM card in the board in the car. The other drawback that OnStar’s service has is that it is not compatible with all vehicle models. The OnStar system comes equipped as a feature in new cars. People that have older vehicle models do not have access to the OnStar service. OnStar is a General Motors service, which can not be installed on other manufacturer’s vehicles. Our product can be installed in any modern automatic vehicle.

There is a very large market in the genre of remote car starters. There is the basic model which provides the user with a set of buttons to turn on/off the vehicle. This model usually has a range of about 500 feet and has a cost around 80 dollars [6]. The problem with this model is with the range. If the user is not within that range, then he/she may not start the vehicle. We are offering a product that can work anywhere the user has cell phone service. Our Remote Car Controller offers more features that will allow the user to monitor the many aspects of the vehicle.
There are also more advanced types of remote car starters. Their costs range from around 150 to 300 dollars. These advanced car starters offer everything that the basic starters offer but with improved range and more features. These features can include the locking/unlocking of doors, timed start, and headlight reminder [8]. The biggest problem with these types of remotes is that they are often bigger and heavier than a cell phone itself. The idea was to get rid of items that had to be carried as opposed to adding them. With the Remote Car Controller, the user has the same features plus more. It adds the ability to monitor some of the other vehicle functions and provides unlimited range. More importantly, it does not require the user to put an additional heavy object in his pocket. It gives a simple application on the cell phone that will allow the same type of control.

Both the basic and advanced remotes work with batteries. This means that they will need to be replaced about every 2 to 3 months. The Remote Car Controller does not need an additional battery since it is an application on the cell phone. The user will just have to recharge its cell phone as it normally does.

There is one more competitor that is worth mentioning. It is not on the market yet, but is in the process of getting a patent. Sarah Dodge, a 17 year old high school student, invented a cell phone car starter system. Her starter system uses two cell phones. One of the cell phones has to be connected and inside the car at all times. The person then calls the cell phone inside of the car, inputs a password and then the car will start. But this system has two problems. First, the user needs to be paying for an extra cell phone and service on top of their monthly bill. Another option is to have a prepaid phone inside of the car but with each time the phone in the car is called, a minute will be used up. Second, the user will need to extract the cell phone from inside of the car when it is time to recharge it. If it is not recharged, it won’t be able to start the car. The Remote Car Controller does not have this problem since the receiver inside recharges automatically with the car battery. This way the Remote Car Controller can hold a charge for 2 months without the car being started. There is also an added benefit in the fact that the user does not need to be paying for an extra phone number, just the SMS service.

With the features we have included in our project, it is easy to see how our competition fails. Our product provides the user a useful service and more importantly it has a low cost, is not limited in range, and can be installed in any vehicle.
Design Specification

The Remote Car Controller uses a cell phone that resides on the GSM 1800/850 network with SMS text messaging enabled. The phone is then able to communicate with the BieneRemote128GM unit. The user is able to send a text message from his phone to the device in his vehicle. This text message is interpreted by the device and the pre-declared operation will take place. The device has a few configured outputs. These outputs are going to be starting the vehicle, arming the alarm system, locking the vehicle, turning on the ventilation system, and toggling any of above operations. The unit will also turn off the vehicle after a preset time, such as 15 minutes with no other operation. The unit will also be able to send text messages as alerts to the user with certain information such as the current temperature of the vehicle, whether the vehicle is running, and when the vehicle has been turned off. Confirmation messages will also be sent whenever any task has been completed (i.e. the vehicle has been un-locked.)

The main objective of the Remote Car Controller is to start a person’s vehicle from a remote location. The distance that it will work at is seemingly unlimited as the signal will be carried through cell phone towers to reach the vehicle. The vehicle will then be started and a text message will be sent back to the cell phone to inform of the success or failure of the start.

Besides the main function of starting the vehicle, the Remote Car Controller will also be able to monitor many aspects of the vehicle’s health, and relay those measurements via a text message right back to the cell phone. It can measure the engine temperature so the person will not have to wonder when the vehicle has actually warmed up. The temperature will be measured by a temperature sensor will be inside the engine bay and will relay the change in temperature to the BieneRemote. Once the engine temperature reaches a certain level that the user has set, the BieneRemote will send a SMS text to the cell phone informing the driver that their vehicle is warmed up. All of these options will help a person keep an eye on their vehicle even if they cannot see it.

The last couple of functions of the vehicle will help with safety. A person will be able to control the power locks and arm/disarm the alarm of the vehicle. This will provide one extra fail-safe in order to prevent getting locked out of the vehicle. The monitoring of the alarm will allow the vehicle to send a text message when the alarm goes off so it can warn of a possible intruder.

The unit can be set up to send alerts to up to three different phones. It can also be set to only receive commands from a list of numbers. This can be beneficial if there is more than one owner of the vehicle, or in case the user’s cell phone is stolen, the vehicle will then not be at risk.

There will be no need to adjust the hardware of the cell phone, as it uses GSM and SMS to communicate with the vehicle. The only addition will be a software interface that can be run so the data that the phone receives from the vehicle will be able to be interpreted and displayed in a presentable fashion. The product will also work with almost all modern automatic transmission vehicles so there will be no compatibility issues.
Unlike some services such as OnStar, there is no monthly fee associated with the service. The only ongoing cost will be the price of the text messages the vehicle will send, but that is very minimal in comparison. There will be a one time installation of the unit into the vehicle. It will have to be electrically wired and set up to monitor all of the diagnostics. Once this is completed, no regular maintenance will be required.
**Design Overview**

![Diagram of a circuit board with connections to Starter, Locks, Temperature Sensor, and Brakes]

**The Graphical User Interface (GUI)**

The user interface is written in C# and communicates through the cell phone to the BieneRemote module in the vehicle, which then communicates to the different devices in the vehicle.

**The Cell Phone**

The cell phones that are compatible with the BieneRemote module are GSM-based phones, and it must be operate on the GSM 850/1800 network with SMS capabilities. SMS capability is a must because this is how the cell phone will send and receive text messages to and from the BieneRemote module. The cell phone must be able to download the GUI we have created or be able to connect to a PC via a data sync cable. The phone must then be able to run the GUI we have created.
**The BieneRemote128GM Module**

The BieneRemote128GM Module is capable of holding five different cell phone numbers that are stored on the 3v SIM card located on the module. Only the numbers stored on the SIM card can send or receive messages to the BieneRemote module, thus keeping unwanted messages away and allowing only certain people access to the vehicle. We will be using four outputs on the module: The first output communicates with door locks, the second can manipulate the vehicle's alarm, the third output will be used to communicate with the vehicle's ventilation system, and the last will be used to start the vehicle. One input on the module will be used to obtain the vehicle's current running temperature; this temperature can be relayed to the user and appear on the users' GUI on the cell phone. The module will last for up to a month without the vehicle being used before the vehicle's battery is at risk of being too low to start the vehicle. The BieneRemote also has two LED's on the module that indicate when the system is in use, and if there are any problems with the system. The module comes with its own GUI that is loaded on a PC. This software is used to setup the module for use. The GUI gives us the ability to initialize the module’s control over the vehicle and the ability to set certain thresholds. This will mainly be used for the included temperature sensor that we will be using to monitor the vehicle's running temperature. The module will be stored in the vehicle preferably under the dash or secured in the vehicle's trunk. We will have to extend the included GSM Antenna so that it will receive the best signal. The antenna will be secured either in the rear window or the front windshield.

**Door Locks**

The module is able to communicate with the vehicle’s locks and is able to either lock or unlock the vehicle depending on the commands the user sends. The module will then send a text message to the users’ GUI confirming whether the doors have been locked or unlocked. This is all done, by attaching the BieneRemote to the main power wires of the vehicle's locking system.

**Locking Doors**

Send Text → Set Output 2 → Applies 12V to lock wire → Sets on Output2 → Resets Output 2
**Unlocking Doors**

Send Text → Set Output 3 → Applies 12V to lock wire → Sets on Output3 → Resets Output 3

**Vehicle Starter**

The module is able to communicate with the starter relay in the vehicle that relays a signal to the vehicle’s ignition system and starter. These two connections are the two connections that make it possible for the module to start the vehicle. The user is then sent a text message confirming that the vehicle has started. The module will monitor how long the vehicle has been running and after 15 minutes of idle time without the vehicle being entered, the vehicle will be shutdown and the user will be notified that the vehicle has been shutdown via a text message.
Temperature Sensor

The module is able to receive information from the attached temperature sensor. This sensor is connected to one of inputs on the module. The temperature sensor information is then relayed to the user via a text message sent to the GUI. The module is able to identify three different temperature thresholds: low, high, and normal. These threshold values are setup on the module via the modules software. This will allow us to customize the settings for different types of vehicles. When the vehicle has reached the normal operating temperature the module will send a text message to the user notifying them of this.
Design Detail

The BieneRemote that is used to communicate between the cell phone and car must be programmed. The data sheet for the board that we use is shown in Appendix A. With the power that it consumes, it is possible for the Remote to run idle for up to a month before it may begin to affect the battery on startup. A rough board layout is shown below.

![Board Layout](image)

This shows where all the inputs and outputs are located on the board. The pin-out diagrams are also all given for the P1 connection (antenna) and for P2 – P6 (analog inputs and outputs).

The Remote was programmed through a serial port that is built into each board. Biene Electronics provided software with the board that enabled us to get the functionality of each input and output to be where we want it to be. Biene Electronics was very helpful and generous with information about the workings of the board.

The Remote communicates to the cell phone by using phone numbers that are programmed into its own SIM card, and can hold up to five external phone numbers. Text messages can then be sent to these four phones as alerts to the car’s status. The Remote will be set up to only take commands from the numbers programmed into its SIM card. This prevents the Remote from getting bombarded with text messages from outside users, and possibly starting the car at an unwanted time. It also only knows the phone numbers it should be sending alerts to as well.

The next major component that is included is the car starter relay. This had to be set up to be controlled by one of the outputs from the Remote. All starters have five outputs; there’s one that connects to the ignition, one that grounds the remote starter when running, one that connects to the starter wire, on to the remote starters output, and the last pin runs to the starter wire from the key.
A minor component consists of the temperature sensor. This is a feature built into the BieneRemote and the temperature sensor will be attached to the engine block for engine temperature or inside the car for internal temperature. This makes it possible for the sensor to relay information to the BieneRemote. We set gauges of normal, high, and low, and when the car gets into the level of either too high or too low, a SMS message is sent to the user, alerting them of the abnormal temperature of their engine.

And the final major component is the user interface (GUI). This software allows the user to activate certain functions on our receiver that is placed in the car. The user will run the program on their phone, and then select the desired feature they would like to execute. Once they’ve selected an option from the screen, the program sends the appropriate text message to the receiver in the car. The receiver interprets these messages and executes the desired function. The functions that the program can execute are starting and stopping the car. It can also lock and unlock the car doors, as well as check the cars temperature. The receiver can also send confirmation text messages back to the user if they desire. The code for the GUI can be found in Appendix B. The code for the SMS text messaging for the program can be found in Appendix C. Below are a few pictures of what the program looks like.
Design Evaluation

We were able to successfully get all of the major components of our design to work the way we wanted to. The programming of the BieneRemote was relatively easy because of the software provided by the company. We were able to add our cell phone numbers onto the board so we would receive the text messages sent as status updates. Setting the voltage outputs was also done with the software. Once we had confirmed which outputs/inputs would be connected to each component, programming of the board was complete. Outputs for the car starter and door locks were successful. To test these outputs, we connected LED lights on our Proto-Board, and when we sent a text message to turn on the output the LED would light up in about 15-30 seconds depending on cell service.

The BieneRemote that we purchased was the most advanced board offered at the time. The biggest difference over the other boards is that this one came with a temperature sensor on the board. Unfortunately, the temperature sensor supplied on the board needed a specific temperature probe only available in Europe. We were able to find a distributor of the probe, with the cost being about $45 each, but the shipping was over $100! We decided to pass on the purchase of the probe, and built our own temperature sensor on the Proto-Board and use a different temperature probe we purchased for just $10. We were able to get the temperature probe calibrated based loosely on a ratio of voltage to degrees, and were able to get a status update of temperature automatically sent to our cell phones when the temperature got either too hot or too cold. The only difference we would make to improve the design is just purchase the more simple board without the temperature sensor built in because we didn’t use it at all.

Below is a picture of the BieneRemote on the right, and our circuit schematic on the Proto-Board on the left. The temperature sensor we created is located on the left side of the Proto-Board. The LED’s we tested the outputs with can also be seen on the right side of the Proto-Board.
However, we did run into some problems with another component of our design, the GUI. We were able to write a cell phone program coded in C#, that had the text message commands configured for one-click access. The program would run fine on our PC and everything simulated correctly, but we could not get the program onto a cell phone. We tried to put the program onto a Windows Mobile enabled phone, but the phone would not allow the program to run because it wasn’t a certified code. We tried many different solutions to get certificates for the program, but all of them failed. This was a disappointment for the group because the GUI would have made it extremely easy for the user to control aspects of their car from their phone. This problem did not hinder our product, however, because we were able to send the text message strings to the board manually. It just made things a little more difficult.

A prime concern we have is security. The BieneRemote feature that allows it only to talk to phone numbers programmed into its SIM card is great, but it won’t stop experienced hackers from getting into the Remote. Development of some sort of firewall protection of the chip must be explored, but we did not have the time during the semester to explore the options. We must also protect against a number “spoof” which is when someone hacks their phone so it looks like it is coming from one of the programmed numbers in the SIM card. Finding solutions to these security issues is something we could possibly look into sometime in the future; more for satisfaction to ourselves that we created a product that is secure.
Parts

The following is a list of parts that we need to develop the prototype:

- **Transmitter**
  
  - **Cell Phone with SMS Service**
    
    We used one of the member’s cell phone which has an unlimited SMS service. The cell phone we used had a PC connection, and we were able to load the GUI onto it and operate the program.
    
    This part did not affect the cost for the final product or prototype as this is a requirement that the user must have before buying our product.

- **Cell Phone Interface**
  
    We developed the program using C#. It didn’t add any cost to the prototype as we are going to be in charge of it. But, for the final product it will consist around 20% of the final cost.

- **Receiver**
  
    - **BieneRemote128GM-4A/Pt1000**
      
      This is the main board that is in charge of all the operations inside of the car. This board is made by Biene Electronics and we bought it from the Ecuador’s distributor, we chose this location as it is the closest to Boston. The board includes the antenna and the software application to set it up.
      
      The board’s cost is 467.68 dollars and the shipping’s cost is 182.75 dollars. This is the main cost in the prototype as this is the only item that we need to purchase, and as a consequence it will be major part of the cost on the final product.

    - **SIM Card**
      
      The SIM Card is in charge of connecting the board to the GSM cell phone network. We purchased a prepaid SIM card from Cingular Wireless for $25. Each text message costs $0.10, thus providing us with 250 text messages.
- Car

We used Jason’s Jeep to develop the prototype and film the live demo we presented in the Capstone competition. The Jeep we used had a car starter already installed in it, so there was no need to buy an additional car starter and need to install it in the car.

- Wires and Electric Tools

We need wire to connect the components inside of the car and the tools to work on the electronics. We are going to use the supplies that are given to us in the Capstone Lab. This will not add any additional cost to the prototype and either to the final product.

Prices

<table>
<thead>
<tr>
<th>Prototype</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BieneRemote128GM</td>
<td>$400</td>
</tr>
<tr>
<td>Prepaid SIM Card</td>
<td>$25*</td>
</tr>
<tr>
<td>Temperature Sensor</td>
<td>$10</td>
</tr>
<tr>
<td>Circuit Parts and Wiring</td>
<td>$5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Production Estimates</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BieneRemote128GM</td>
<td>$300</td>
</tr>
<tr>
<td>Prepaid SIM Card</td>
<td>$25*</td>
</tr>
<tr>
<td>Temperature Sensor</td>
<td>$10</td>
</tr>
</tbody>
</table>

* Each text message is $0.10 per message with Cingular Wireless
References


## Appendix A

### BieneRemote128GM Hardware Specification

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>GSM900/1800/1900; BieneRemote128GM-PCS</td>
<td>GSM850/900/1800/1900; BieneRemote128GM-QUAD</td>
<td></td>
</tr>
<tr>
<td>Command and data transmission</td>
<td>SMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal GSM modem</td>
<td>Telit GM862-PCS / GM862-QUAD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIM card reader</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIM card type</td>
<td>Phase 2 GSM11.14 - SIM 3V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firmware</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Digital input:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of digital inputs</td>
<td>6 (+1 optional)</td>
<td>6 (+1 optional)</td>
<td>6 (+1 optional)</td>
</tr>
<tr>
<td>- Transistor digital input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Events digital inputs</td>
<td>(*&quot;0&quot;: 0...+1V; &quot;1&quot;: +1.5...+12V without external limited resistor)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Digital signal filter</td>
<td>6 (+1 optional)</td>
<td>2kms - 9sec</td>
<td></td>
</tr>
<tr>
<td><strong>Temperature sensor inputs:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature sensor</td>
<td>Pt1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Pt1000 inputs</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature events inputs</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature sensor type</td>
<td>Pt1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature range</td>
<td>-99 to +149 °C</td>
<td>-146.2 °F to +300 °F</td>
<td></td>
</tr>
<tr>
<td>Events Temperature range</td>
<td>-99 to +99 °C</td>
<td>-99 °F to +99 °F</td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Analog input:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of analog inputs</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>- Input</td>
<td>0...+5V / 0...+10V, 100MΩ</td>
<td>0...+5V / 0...+10V, 200kΩ</td>
<td>0-20mA / 4-20mA</td>
</tr>
<tr>
<td>- Optional</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Protection</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>- Analog event inputs</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>ADC resolution</td>
<td>10 bits</td>
<td>10 bits</td>
<td></td>
</tr>
<tr>
<td><strong>Output:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of outputs</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- MOSFET Open Drain outputs</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Relay outputs</td>
<td>(MOSFET S5T5NF20V, 20V max)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Logger (optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timing Interval</td>
<td>30-60 sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data transfer</td>
<td>RS232 port / GPRS (file on server)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logging data</td>
<td>Date/ Time, analog data, digital data</td>
<td>Date/ Time, temperature, analog data, digital data</td>
<td></td>
</tr>
<tr>
<td>Line in logging data file</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Supply</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required Power supply</td>
<td>External -12 VDC stabilized</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power requirement</td>
<td></td>
<td>70mA typ, 800mA(rms) max, 2A peak during transmission</td>
<td></td>
</tr>
<tr>
<td>Voltage regulator</td>
<td>Internal voltage regulator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power protection</td>
<td>Reverse-polarity and overvoltage protection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environmental Conditions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal operational temp. range</td>
<td>-10...+55°C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extreme operational temp. range</td>
<td>-20...+70°C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physical parameter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Board dimension</td>
<td>100x52mm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

#region Using directives
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using Microsoft.Wireless;
#endregion

namespace RemoteController
{
    /// <summary>
    /// This Form is for the Remote Controller Main Form.
    /// </summary>
    public class RemoteController : System.Windows.Forms.Form
    {
        private Panel panel1;
        private Label label1;
        private Label label2;
        private Label label3;
        private Label label4;
        private Label label7;
        private MainMenu mainMenu1;
        private MenuItem menuItem1;
        private MenuItem menuItem2;
        private MainMenu mainMenu2;
        private VScrollBar vScrollBar2;
        public Microsoft.Wireless.SMS thisSMS;

        public RemoteController()
        {
            InitializeComponent();
        }

        protected override void Dispose(bool disposing)
        {
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        private void InitializeComponent()
        {
            this.panel1 = new System.Windows.Forms.Panel();
            this.label7 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.mainMenu1 = new System.Windows.Forms.MainMenu();
            this.menuItem1 = new System.Windows.Forms.MenuItem();
            this.menuItem2 = new System.Windows.Forms.MenuItem();
            this.mainMenu2 = new System.Windows.Forms.MainMenu();
            this.vScrollBar2 = new System.Windows.Forms.VScrollBar();
            this.thisSMS = new Microsoft.Wireless.SMS();
            this.panel1.SuspendLayout();
            this.SuspendLayout();

            // Form 1 Design Time Code

            // Form 1 compiled time code

            this.SuspendLayout();
        }

        #endregion
    }
}
this.menuItem1 = new System.Windows.Forms.MenuItem();
this.menuItem2 = new System.Windows.Forms.MenuItem();
this.mainMenu2 = new System.Windows.Forms.MainMenu();
//
// panel1
//
this.panel1.Controls.Add(this.label1);
this.panel1.Controls.Add(this.label2);
this.panel1.Controls.Add(this.label3);
this.panel1.Controls.Add(this.label4);
this.panel1.Controls.Add(this.label7);
this.panel1.Location = ((System.Drawing.Point)(resources.GetObject("panel1.Location")));
this.panel1.Visible = ((bool)(resources.GetObject("panel1.Visible")));
//
// label7
//
this.label7.Font = ((System.Drawing.Point)(resources.GetObject("label7.Location")));
this.label7.Size = ((System.Drawing.Size)(resources.GetObject("label7.Size")));
this.label7.Text = resources.GetString("label7.Text");
this.label7.Visible = ((bool)(resources.GetObject("label7.Visible")));
//
// label4
//
this.label4.Text = resources.GetString("label4.Text");
//
// label3
//
this.label3.Location = ((System.Drawing.Point)(resources.GetObject("label3.Location")));
this.label3.Text = resources.GetString("label3.Text");
//
this.label3.Size = 
this.label3.Text = resources.GetString("label3.Text"); 
this.label3.TextAlign = 
this.label3.Visible = 
((bool)(resources.GetObject("label3.Visible"))); 
// // label2 
this.label2.Font = 
this.label2.Location = 
((System.Drawing.Point)(resources.GetObject("label2.Location"))); 
this.label2.Size = 
this.label2.Text = resources.GetString("label2.Text"); 
this.label2.TextAlign = 
((System.Drawing.ContentAlignment)(resources.GetObject("label2.TextAlign"))); 
this.label2.Visible = 
((bool)(resources.GetObject("label2.Visible"))); 
// // label1 
this.label1.Font = 
this.label1.Location = 
((System.Drawing.Point)(resources.GetObject("label1.Location"))); 
this.label1.Size = 
this.label1.Text = resources.GetString("label1.Text"); 
this.label1.TextAlign = 
this.label1.Visible = 
((bool)(resources.GetObject("label1.Visible"))); 
// // mainMenu1 
// this.mainMenu1.MenuItems.Add(this.menuItem1); 
this.mainMenu1.MenuItems.Add(this.menuItem2); 
// // menuItem1 
// this.menuItem1.Enabled = 
((bool)(resources.GetObject("menuItem1.Enabled"))); 
this.menuItem1.Text = resources.GetString("menuItem1.Text"); 
this.menuItem1.Click += new System.EventHandler(this.menuItem1_Click); 
// // menuItem2 
// this.menuItem2.Enabled = 
((bool)(resources.GetObject("menuItem2.Enabled"))); 
this.menuItem2.Text = resources.GetString("menuItem2.Text");

22
this.menuItem2.Click += new System.EventHandler(this.menuItem2_Click);

//
// RemoteController
//
this.Controls.Add(this.panel1);
this.Font = ((System.Drawing.Font)(resources.GetObject("$this.Font")));
this.Location = ((System.Drawing.Point)(resources.GetObject("$this.Location")));
this.Menu = this.mainMenu1;
this.Text = resources.GetString("$this.Text");
this.KeyDown += new System.Windows.Forms.KeyEventHandler(this.RemoteController_KeyDown);

System.Windows.Forms.KeyEventHandler(this.RemoteController_KeyDown);
}

#endregion

static void Main()
{
    Application.Run(new RemoteController());
}

private void menuItem1_Click(object sender, System.EventArgs e)
{
    this.Close();
}

private void menuItem2_Click(object sender, System.EventArgs e)
{
    //System.Windows.Forms;
}

private void RemoteController_KeyDown(object sender, KeyEventArgs e)
{
    {
        // Soft Key 1
        // Not handled when menu is present.
    }
    {
        // Soft Key 2
        // Not handled when menu is present.
    }
    {
        // Up
    }
    {
        // Down
    }
    {
        // Left
    }
    {
        // Right
    }
}
{
    // Right
}

{
    // Enter
}

{
    // 1
    // Start Vehicle
    for (int i = 0; i < 100000; i++)
    {
        SMS.SendMessage("6178995006", "setou4");
    }
}

{
    // 2
    // Stop Vehicle
    for (int i = 0; i < 100000; i++)
    {
        SMS.SendMessage("6178995006", "setou3");
    }
}

{
    // 3
    // Lock Vehicle
    for (int i = 0; i < 100000; i++)
    {
        SMS.SendMessage("6178995006", "setou1");
    }
}

{
    // 4
    // Un-lock Vehicle
    for (int i = 0; i < 100000; i++)
    {
        SMS.SendMessage("6178995006", "setou2");
    }
}

{
    // 5
    // Vehicle Temperature
    for (int i = 0; i < 100000; i++)
    {
        SMS.SendMessage("6178995006", "setin4");
    }
}
{
}
{
}
{
  // 8
}
{
  // 9
}
{
  // *
}
{
  // 0
}
{
  // #
}
using System;
using System.Text;
using System.Runtime.InteropServices;

namespace Microsoft.Wireless {
    /// <summary>
    /// Identifies the phone number type specified.
    /// </summary>
    public enum AddressType {
        /// <summary>Unknown phone number type.</summary>
        Unknown,
        /// <summary>International phone number.</summary>
        International,
        /// <summary>National phone number.</summary>
        National,
        /// <summary>Network-specific phone number.</summary>
        NetworkSpecific,
        /// <summary>Subscriber phone number.</summary>
        Subscriber,
        /// <summary>Alphanumeric phone number.</summary>
        Alphanumeric,
        /// <summary>Abbreviated phone number.</summary>
        Abbreviated
    }

    /// <summary>
    /// Information about the phone number.
    /// </summary>
    public struct PhoneAddress {
        /// <summary>The address type.</summary>
        public AddressType AddressType;
        /// <summary>The phone number in string format.</summary>
        public String Address;
    }

    /// <summary>
    /// Short Message Service.
    /// </summary>
    public class SMS {
        private static string SMS_MSGTYPE_TEXT = "Microsoft Text SMS Protocol";
        private static long SMS_MODE_SEND = 0x00000002;
        private static long SMS_OPTION_DELIVERY_NONE = 0x00000000;
        private static long SMS_OPTION_DELIVERY_NO_RETRY = 0x00000001;
        private static long PS_MESSAGE_OPTION_NONE = 0x00000000;

        private enum SMS_DATA_ENCODING {
            SMSDE_OPTIMAL = 0,
private enum PROVIDER_SPECIFIC_MESSAGE_CLASS
{
    PS_MESSAGE_CLASS0 = 0,
    PS_MESSAGE_CLASS1,
    PS_MESSAGE_CLASS2,
    PS_MESSAGE_CLASS3,
}

private enum PROVIDER_SPECIFIC_REPLACE_OPTION
{
    PSRO_NONE = 0,
    PSRO_REPLACE_TYPE1,
    PSRO_REPLACE_TYPE2,
    PSRO_REPLACE_TYPE3,
    PSRO_REPLACE_TYPE4,
    PSRO_REPLACE_TYPE5,
    PSRO_REPLACE_TYPE6,
    PSRO_REPLACE_TYPE7,
    PSRO_RETURN_CALL,
    PSRO_DEPERSONALIZATION,
}

private struct TEXT_PROVIDER_SPECIFIC_DATA
{
    public IntPtr dwMessageOptions;
    public PROVIDER_SPECIFIC_MESSAGE_CLASS psMessageClass;
    public PROVIDER_SPECIFIC_REPLACE_OPTION psReplaceOption;
}

[DllImport("sms.dll")]  // Calls the native SMS API functions.
private static extern IntPtr SmsOpen(String ptsMessageProtocol, IntPtr dwMessageModes, ref IntPtr psmshHandle, IntPtr phMessageAvailableEvent);

[DllImport("sms.dll")]  // Calls the native SMS API functions.
private static extern IntPtr SmsSendMessage(IntPtr smshHandle, IntPtr psmsaSMSCAddress, IntPtr psmsaDestinationAddress, IntPtr pstValidityPeriod, byte[] pbData, IntPtr dwDataSize, byte[] pbProviderSpecificData, IntPtr dwProviderSpecificDataSize, SMS_DATA_ENCODING smsdeDataEncoding, IntPtr dwOptions, IntPtr psmsmidMessageID);

[DllImport("sms.dll")]  // Calls the native SMS API functions.
private static extern IntPtr SmsClose(IntPtr smshHandle);

/// <summary>
/// Sends a SMS message to the phone number specified.  
/// </summary>
/// <param name="sPhoneNumber">The phone number of the recipient</param>
/// <param name="sMessage">The message to send.</param>
unsafe public static void SendMessage(string sPhoneNumber, string sMessage)
IntPtr hSms = IntPtr.Zero;

try
{
    IntPtr res = SmsOpen(SMS_MSGTYPE_TEXT, (IntPtr)SMS_MODE_SEND, ref hSms, IntPtr.Zero);
    if (res != IntPtr.Zero)
        throw new Exception("Could not open SMS.");

    Byte[] bDest = new Byte[516];
    fixed (byte* pAddr = bDest)
    {
        byte* pCurrent = pAddr;
        Marshal.WriteInt32((IntPtr)pCurrent, (int)AddressType.Unknown);
        pCurrent += 4;

        foreach (byte b in Encoding.Unicode.GetBytes(sPhoneNumber))
            Marshal.WriteByte((IntPtr)pCurrent, b);

        // The data for the TEXT_PROVIDER_SPECIFIC_DATA
        byte[] ProvData = new Byte[12];

        byte[] bMessage = Encoding.Unicode.GetBytes(sMessage);
        int nMsgSize = bMessage.Length;

        res = SmsSendMessage(hSms, IntPtr.Zero, (IntPtr)pAddr, IntPtr.Zero, bMessage, (IntPtr)nMsgSize, ProvData, (IntPtr)ProvData.Length, SMS_DATA_ENCODING.SMSDE_OPTIMAL, (IntPtr)SMS_OPTION_DELIVERY_NONE, IntPtr.Zero);

        if (res != IntPtr.Zero)
            throw new Exception("SMS send failed.");
    }
}
finally
{
    if (hSms != IntPtr.Zero)
        SmsClose(hSms);
}