Computer Engineering
Senior Project Proposal

- Remote Vehicle Interface (RVI) -

<u>Team Members</u>

Travis Johnson
tg_johnson@yahoo.com

Ben Moon
bmoon@cs.utah.edu

<u>Project Website</u>

http://www.remoteVI.com

Table of Contents

Introduction.

This project aims to enhance the functionality of commercially available keyless entry/starter devices for vehicles. The next logical steps for devices of this type would be to extend their range and offer the user more ways in which to interact with their vehicle, namely WWW and telephone interfaces. Upon completion of this project, a prototype system will be demonstrated that allows a user to start/kill their engine, lock/unlock their doors, and pop their trunk, all from a much greater distance than current devices allow. This prototype system will approximate a top-down line-of-sight RF signal, much like a satellite-based implementation.

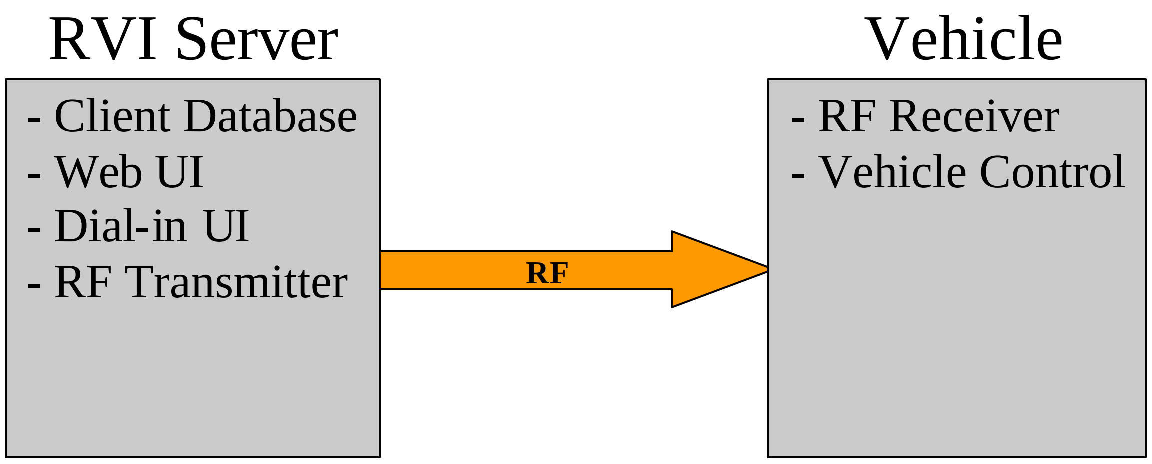
Motivation.

Remote entry/starter devices have become commonplace, and their features have proven to be very desirable. The downfall of current devices is their range. Most key-ring transmitters need to be within tens of meters in order to work, rendering the device practically useless in certain situations. Below are two examples where the RVI system would prove useful, but a current device would not:

> A teaching assistant holding late office hours is getting ready to leave. He arrived on campus at about 9am, and it has been snowing heavily all day. Even if his car is equipped with a remote starter, he cannot use it until he gets to the parking lot on the other side of campus. With the RVI, he can start his car when he leaves the computer lab. By the time the shuttle drops him off in the parking lot, his car is warmed up and ready to go.
>
> The same TA locks his keys and key-ring transmitter in the car after grabbing his snow scraper. Luckily, with the RVI, he can use his cell phone to unlock his doors.

System Overview.

The highest level block diagram consists of a server and a vehicle:



Combining the client database, user interfaces, and transmitter into a single component is a design decision that was made for simplicity and monetary reasons; obviously, for redundancy and fault tolerance, these subcomponents would not be combined. Each component from the block diagram above, as well as the overall system, will be discussed in greater detail in the following sections. The *Components and Tasks* section will enumerate the implementation of each component, and the *Interfaces* section will explain how the components tie together to make the system functional.

Components and Tasks.

- *Client Database* – The RVI Server will require a database to store information about users of the system. The exact information stored will become clear as the rest of the system is explained. This will be accomplished using Microsoft Access. Building the database is pretty straightforward.
- *WWW User Interface* – Users will have the option of logging on to a publicly hosted web site. They will then be able to access the RVI system features. This component will be implemented using Microsoft Internet Information Service (IIS) in conjunction with the .NET framework. The .NET framework has features that

include ASP.NET for dynamic web content and ADO.NET for database connectivity; both are key elements in this project.

- *Dial-in User Interface* – Users will have the option of calling into the RVI system from any telephone, land-line or cellular, where they will then have access to the RVI system features. This component will be realized by writing a program that (1)accepts incoming calls, (2)provides menus and submenus, (3)authenticates users, and (4)recognizes DTMF tone inputs from the user. To expedite the authoring of this program, an ActiveX Control called VTapi will be used. VTapi provides a plethora of built-in functions that perform many of the above modem related tasks.
- *RF Transmitter* – This component resides on the server end and transmits packets to the receiver at the vehicle end. Its necessity in this system should be pretty clear-cut at this point. A commercially available 900 MHz transceiver will be used in lieu of building this component. The exact device is listed in the *Bill of Materials* section.
- *RF Receiver* – This component will reside on the vehicle end and accept packets from the transmitter on the server end. It will be the same model as the transmitter.
- *Vehicle Control* – A component is necessary that will make sense out of the packets received on the vehicle end. Since this component will need to act as the middle-man between the receiver and the vehicle's systems, a microcontroller unit (MCU) makes the most sense. The MCU needs to have at least two ports for communicating with other devices, and it needs enough storage to accommodate the code that will be written to implement its combinational logic. The exact model of the MCU is listed in the *Bill of Materials* section.

Interfaces.

- *User to Web/Dial-in UI* – The two user interfaces provided will be the only ways for the user to interact with the RVI system. The inner workings of this system should never concern the user; from their perspective, they can control their vehicle with the push of a button or the click of a mouse.
- *Web UI to Database* – The Web UI will need to retrieve/update information stored in the Access database. This can be accomplished using the ADO.NET library which provides functionality for database connectivity/manipulation.
- *Dial-in UI to Database* – As above, the Dial-in UI will also need to retrieve/update information stored in the Access database. And, again, the ADO.NET library will provide this functionality.
- *RVI Server to RF Transmitter* – The commercial transceiver being used as the transmitter is capable of serial communication via an RS-232 cable. The information to be transmitted will be sent to the transmitter from the RVI Server via the server's serial port.
- *RF Transmitter to Receiver* – The transmitter converts the information to be transmitted into RF packets and transmits them. The receiver needs to recognize these transmissions and accept the RF packets. This functionality is built into commercial transceivers being used for the transmitter/receiver components.
- *RF Receiver to Microcontroller Unit* – Once RF packets are received and converted back to useful command data, that data needs to get to the MCU. This narrowed component choices, but both the receiver and MCU are capable of similar serial communication.

- *MCU to Vehicle* – After getting commands from the receiver, the MCU will decide how to manipulate its connections with the vehicle's lock/ignition systems. Since the exact implementation of these systems is beyond the scope of our academic studies, interfacing with them is yet to be nailed down. It is noteworthy, however, that the MCU provides up to 56 GPIO (general-purpose input/output) signals. This should be more than robust enough to implement the interface once the vehicle's exact implementation is understood.

Security Overview.

The security measures being employed in the RVI system are very similar to the key rings devices available today. The user database will store a seed and an offset for a random number generator algorithm. The MCU on the vehicle end will store the same algorithm with the seed hard coded into it, as well as the last successful offset value. Using this scheme makes intercepting RF transmissions (for underhanded reasons) useless. Even if a thief could take apart and understand an intercepted packet, it would give him virtually no insight about the next random number.

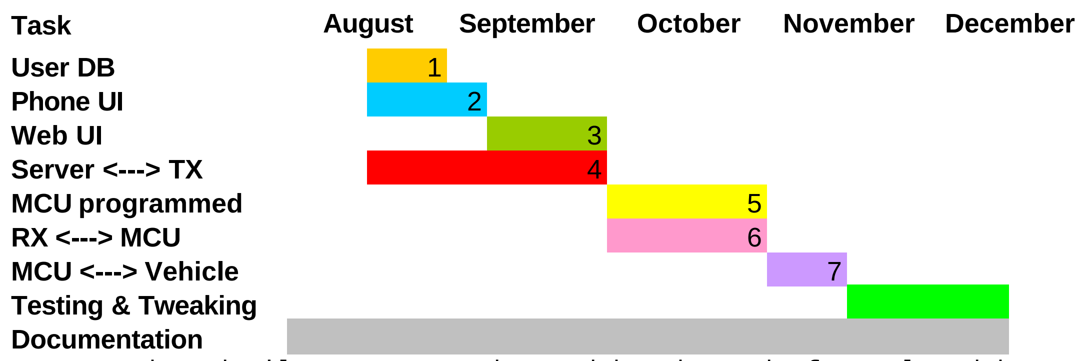
Testing and Integration Strategy.

During the implementation of each component listed in the *Components and Tasks* section, numerous unit tests will be performed to ensure that each component is ready for integration. While completing the component and subsystem integrations described in the *Interfaces* section, more testing will be done to uncover unforeseen system bugs. Like any large system, the RVI system depends on each and every subsystem and component working like it supposed to. This includes accounting for all corner cases and revising interface specifications (when necessary) while carrying out unit/integration testing.

Group Communication Plan.

Many of the negative issues encountered while working in larger groups should be avoidable since this is a two-person project. Specific team meetings will not be necessary as the team members have multiple classes together in the fall semester. Meetings with the Senior Project Advisor, Al Davis, will be scheduled regularly in order to demonstrate milestones and/or discuss project related issues should they arise. The team will also maintain a project website <*http://www.remoteVI.com*> that will have information regarding the project, including all current component and system documentation.

Schedule Flow.



- Numbered milestones are denoted by the end of a colored bar
- The 4 columns spanning each month represent the weeks

Milestones.

1. (9/1/04) Access database has been built. This can be demonstrated by firing up MS Access and showing off the database.
2. (9/8/04) Dial-in UI is working. This can be demonstrated by calling the server and walking through different menus/submenus while monitoring database and serial port activity.

3. (10/1/04) Web UI is working. This can be demonstrated by logging onto the website and navigating through different options while monitoring database and serial port activity.
4. (10/1/04) RVI Server is communicating with the transmitter, and the transmitter is sending RF packets. This can be demonstrated using a network analyzer (NA) in the Microwave Lab (MEB 2275).
5. (11/1/04) The MCU code is authored and tested. This can be demonstrated by sending dummy commands to the MCU and monitoring the appropriate GPIO signals.
6. (11/1/04) Receiver and MCU are communicating and transmitted RF packets are arriving at the MCU's input port as RVI commands. This can be demonstrated much like (5) above, but with real commands.
7. (11/15/04) The MCU's GPIO signals are controlling the appropriate vehicle systems. This can be demonstrated by letting Al try to break the RVI system ☺…

Risk Assessment.

- *User database* – Virtually no risk associated with this task.
- *Web UI* – Low risk. The team has experience with Web design utilizing ASP and ADO.NET.
- *Dial-in UI* – Medium risk. Neither team member has experience authoring code that could be used to address this user interface. While the VTapi ActiveX Control is pretty straightforward and simple to use, its robustness has yet to be determined.
- *Transmitter/Receiver* – Medium Risk. The white papers for this device make communicating with it sound very simple. As the team has yet to actually get our hands on one, this ease of use remains to be confirmed.

- *MCU programming* – Low Risk. The team has experience writing assembly code for Motorola-based MCUs, and the particular MCU being used comes with a C-compiler that should significantly reduce the time to write/debug the MCU code.
- *Vehicle Control* – High Risk. Interfacing the MCU with the appropriate vehicle systems is foreign territory for both team members. To remedy this, we are attempting to get a company that installs similar devices to give us a crash course in how current systems work. A fall-back plan for interfacing the MCU directly with the vehicle is to interface the MCU with a commercially available, short-range entry/starter device that could be installed with greater ease.

Bill of Materials.

<u>RVI Server</u>

- Standard PC
- Already have this component
    - Windows XP Professional
    - Voice Modem
    - NIC
    - Serial port

<u>Web UI</u>

- Microsoft IIS
    - Comes with Windows XP Professional
    - ASP.NET-enabled
        - Provides dynamic content
    - ADO.NET-enabled
        - Provides database connectivity

<u>Dial-in UI</u>

- VTapi ActiveX Control
- Already have this component
- Available from SoftCab, Inc.
    - http://www.softcag.com
    - Free license for non-commercial use

Transmitter / Receiver

- Aerocomm AC4490-1000
- http://www.aerocomm.com/OEM/AC4490.htm
- Primary Vendor: AvNet Electronics
    - Part #: AEROAC4490-1000-3
    - Lead Time: None (In-stock)
    - Unit Cost: $75.00
    - Quantity: 3
    - Total Cost: $225.00 + Shipping
- Secondary Vendor: AeroComm
    - Part #: AC4490-1000
    - Lead Time: None (In-stock)
    - Unit Cost: $68.00
    - Quantity: 3
    - Total Cost: $204.00 + Shipping
    - Will only sell directly if AvNet cannot deliver for some reason

Microcontroller Unit

- Motorola HCS08-based
- Model #: MC9S08GB60
- Primary Vendor: Motorola Semiconductors
    - Part #: DEMO9S08RG60
    - Lead Time: None (In-stock)
    - Unit Cost: $49.95
    - Quantity: 2
    - Total Cost: $99.90 + Shipping
- Secondary Vendor: Arrow Electronics
    - Part #: DEMO908GB60
    - Lead Time: None (In-stock)
    - Unit Cost: $52.95
    - Quantity: 2
    - Total Cost: $105.90 + Shipping

Vendor List.

AvNet Electronics

- Website: http://www.avnet.com
- Contact: None (yet)

AeroComm

- Website: http://www.aerocomm.com
- Contact: Mike Rumph, Western Region Sales Manager
- Arrangements: Mike would not give academic samples, and he preferred that I buy parts from his distributor. Since AvNet is the only distributor, Mike agreed to sell us parts directly if we have any trouble with AvNet.

Motorola Semiconductors

- Website: http://e-www.motorola.com
- Contact: Brian Miller, Academic Sales

Arrow Electronics

- Website: http://www.arrow.com
- Contact: None