

Prototype System for Radio Patching for Law Enforcement in a Mobile Environment

Ivan Elhart¹⁾, Andrew L. Kun²⁾, W. Thomas Miller, III³⁾
University of New Hampshire, Dept. of Electrical & Computer Engineering

Abstract—The mobile radio is one of the most used devices in a police cruiser. It is used to coordinate and dispatch officers on patrol. In the case of crisis, officers have to use additional radios to communicate with other agencies. Usually, this introduces interacting with different manual radio interfaces. Operating several radios at the same time is a very complex task. This complexity is especially taxing when officers have to interact with multiple radios while they are driving.

We outline an approach that has been developed and integrated into the Project54 system. Our prototype is based on affordable off-the-shelf devices and supports the patching of multiple radios in a mobile environment such as a police cruiser. The prototype utilizes the main advantages of Project54 such as integration of multiple police devices and use of a speech user interface. This allows operating several different radio devices using only a speech user interface (a microphone, speaker, and push-to-talk button).

I. INTRODUCTION

While driving police cruisers, police officers often need to communicate with people from multiple organizations at the same time, especially in times of crisis. For example, they may have to talk to other police officers, fire and rescue personnel and federal agents. The much-discussed interoperability problem is that all these organizations do not necessarily use compatible radios that can exchange information with each other [1].

Currently, there are many hardware solutions for this problem on the market. However they are very expensive and sometimes developed for specific radio systems. Because of their price alone, these solutions are not adequate for small public safety agencies with a limited budget.

One of the inexpensive solutions for the radio interoperability problem is the brute force method that is often used. In this method, police officers use more than one radio device at the same time in the cruiser. However, using

different radio types or radios made by different companies involves substantial training because most of them use different user interfaces. Also, operating multiple radios may require adjusting multiple radio parameters, and this may be very distracting, especially during emergency situations, which is exactly when multiple radios may be the most useful.

Also, there is the issue of driver distraction while operating the vehicle. Although the radio is one of the most frequently used devices in the police cruiser, it is also a source of distraction from the primary task of driving. Using the manual user interface that is currently the standard on radio devices can degrade driving performance significantly. Thus, it is likely that operating more than one radio using manual user interface would be even more demanding and distracting. However, in previous work we have shown that interacting with a police radio via a speech user interface does not degrade driving performance [2].

A solution to this problem would be a system integrated into the Project54 system [3]. The Project54 system is a highly integrated in-vehicle hardware/software system whose main goal is to improve the safety and functionality of police cruisers. It is a completely computer based system that simplifies the interaction with in-vehicle electronic devices and allows officers to control them using a speech user interface (SUI) or a graphic user interface (GUI). Through a physical network and the use of open hardware and software standards, the system communicates with all common police devices such as the lights, siren, radio, and GPS unit.

In this paper, we present a prototype PC-based system for radio patching that has been developed as a part of the Project54 system. Our prototype is designed for radio patching in a mobile environment such as a police cruiser. It allows for the usage of several different radios in a cruiser and provides a single user interface for controlling them.

¹⁾ivan.elhart@unh.edu, ²⁾andrew.kun@unh.edu, ³⁾tom.miller@unh.edu

II. PROTOTYPE SYSTEM

The prototype system can be divided into two main parts: software and hardware. The hardware block diagram of the prototype system is shown in Figure 1, while the block diagram of the software solution is presented in Figure 2. Here we will explain how our prototype system's hardware and software components were integrated into Project54 and how a single user interface was used.

A. Hardware design

At the center of the system are a PC and an audio switch card (Figure 1). The PC is connected to the radio devices through the Common Intelligent Device Bus Interface (CIDBI) which utilizes standard CAN 2.0 signaling. This interface is used for the radios monitoring and controlling (e.g. monitoring of the busy indication and controlling of the push-to-talk signal, changing channels, etc.). On the other side, the radios are connected to the audio switch card through an off-the-shelf radio interface card. This device offers control of the input and output signal levels and also provides an alternative automatic push-to-talk signal and control of its duration. The remote radios are used for testing requirements.

The audio switch card can be controlled by the PC using a USB connection and it has four input and six output channels. The card can be programmed to patch any input channel to any of the output channels. Multiple inputs can be patched to the same output and one input can be patched to multiple outputs at the same time.

For a physical push-to-talk the AirClick remote control is employed and its receiver is directly connected to the PC through a USB port. The microphone, speaker and AirClick represent the hardware part of the single user interface.

In the prototype system one input/output pair is used to connect the audio switch card to the microphone (In 4) and speaker (Out 4). One input/output is also routed back to the PC sound system and the Windows Multimedia drivers (Out 3 and In 3). Two input/output pairs are used to connect the system to audio sources and sinks. These audio source/sink pairs are the radios (In 1 and 2 are connected to the radio speaker outputs and Out 1 and 2 are connected to the radio microphone inputs). The two outputs (Out 5 and Out 6) of the prototype system are not connected to any audio sinks.

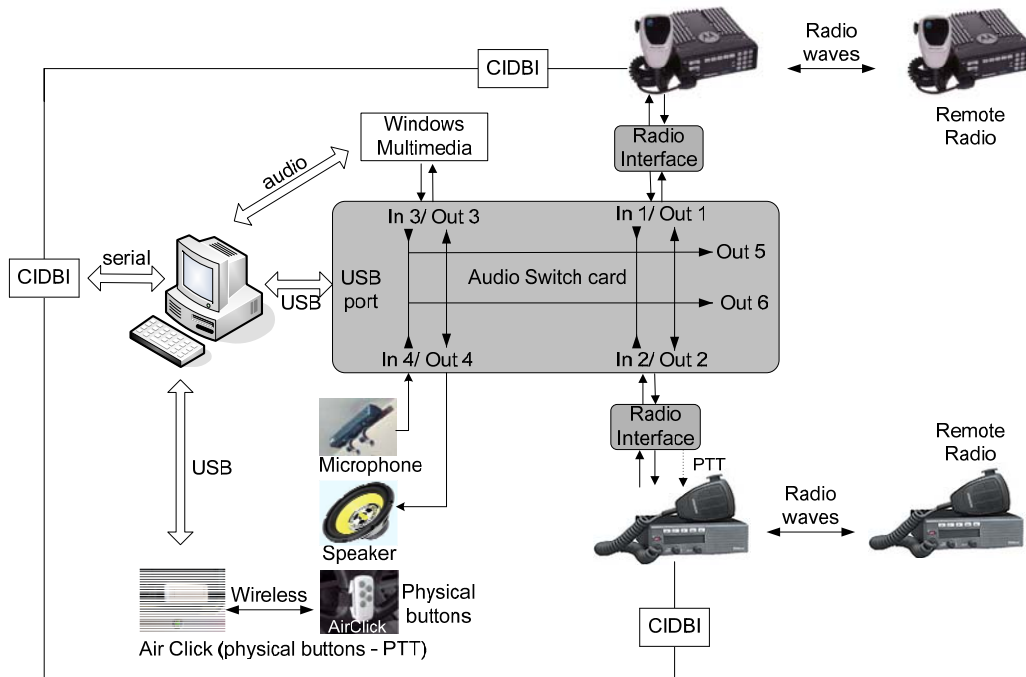


Figure 1 - Hardware block diagram of the prototype system

The input/output pair's positions, where the devices are connected to the audio switch card, are not fixed and they can be changed. All configuration parameters related to the audio switch card, routing, switching and patching audio signals, utilization of physical buttons and the radio monitoring and controlling can be maintained using the software part of the system.

B. Software implementation

The Project54 software part runs on a PC (Figure 1) and consists of components called applications (Figure 2). The applications can communicate with each other using text messages routed through a core piece of software. Most applications control a piece of hardware. Thus, in Figure 2 the Push-To-Talk Application operates with the AirClick remote device. The AirClick is used as the push-to-talk button for initiating the radio transmission, as well as for signaling the beginning and end of a speech command to Project54. The AirClick has multiple buttons and two different AirClick buttons are used for radio transmissions and for speech commands. Speech is handled by the Speech I/O Application. The Speech I/O Application interacts with the speech synthesis and speech recognition components. These two components are used for processing input voice commands and providing voice feedback, respectively. The Radio Control Application monitors and controls radio settings (e.g. channel and volume settings, busy indication,

and push-to-talk signal) for a given radio. Different Radio Control Applications for different police radios have been developed.

For our prototype, we created the Audio Switch with ASIO Support component (ASwAS) and the Audio Control Application. The ASwAS component implements a set of functions to allow the controlling of the audio switch card [4]. The functions from the ASwAS call functions from the ASIO API level which in turn is an abstraction of the hardware architecture and calls functions from the ASIO driver. The specifications for the ASIO API level functions can be found in the Steinberg SDK [5].

The Audio Control Application provides a GUI and SUI for controlling the audio switch card. The audio switch card is represented with switch matrix configuration where each element of the matrix corresponds to the actual connection between a selected input and output. On the GUI, next to the switch matrix are radio status indications (busy and transmit) and indications which button on the AirClick is pressed.

For example, when the Audio Control Application receives a push-to-talk message from the Push-To-Talk Application (push-to-talk button on AirClick is pressed), it redirects the message to the corresponding Radio Control Application (chosen radio) depending on the current audio switch configuration.

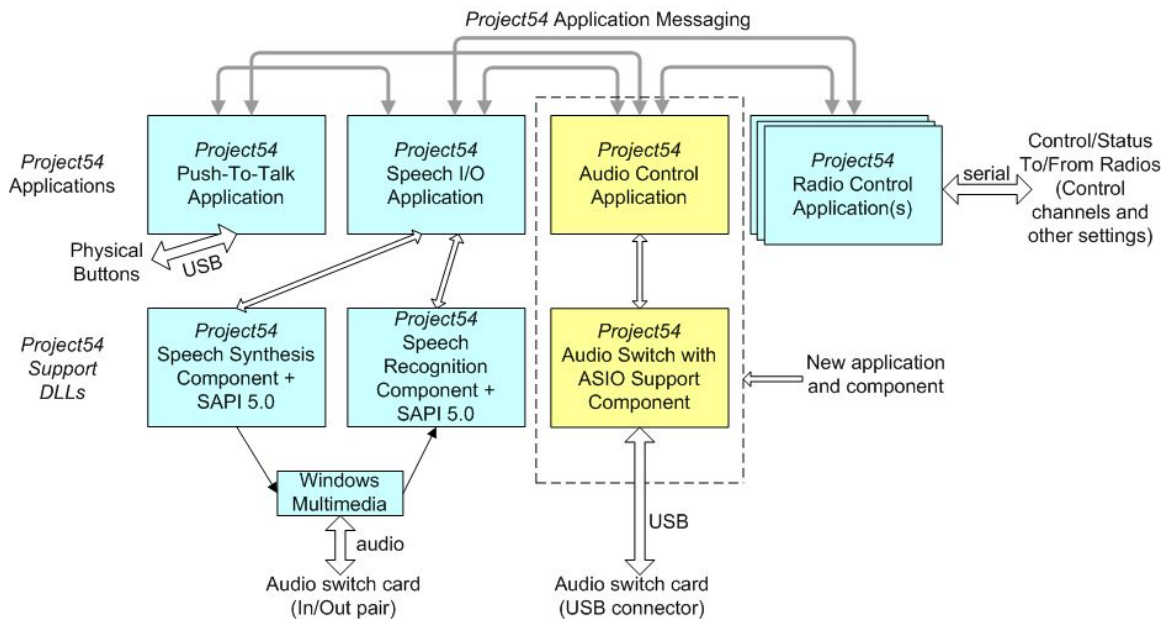


Figure 2 - Software block diagram of the prototype system

Also, the Audio Control Application indicates on the GUI that an AirClick button is pressed. If element (4, 1) in the switch matrix is selected active, then the audio signal from the microphone (In 4) will be routed to the audio input of the first radio (Out 1). Once the Radio Control Application receives the push-to-talk message, it sends a new message to the radio through the CIDBI. When the radio receives the message it starts transmitting and sends feedback transmission indication to the Radio Control Application. This indication is sent to the Audio Control Application and displayed on the GUI.

III. TESTING

The system was developed and tested in laboratory conditions. The lab setup is shown in Figure 3. The radios (Figure 3- items 1 and 4) are connected to the audio switch (item 5) through the radio interface modules (items 2 and 3). Also, the radios are connected to the PC using CIDBI boxes (items 6, 7, and 8). The first box (item 6) is connected to the PC using a serial port interface, while the other two boxes are connected to the radios. The user interface for the system consists of the AirClick device (item 9), microphone (item 10), and the speaker (item 11).

The radios manufactured by different companies (Motorola and E.F. Jonson) are used to prove full radio patching in the system. Another two radios (they are not shown in the figure)

are connected to the radios (items 1 and 4) via radio waves. Those additional radios are included for proof of concept to show the signal is properly transmitted and the push-to-talk signal functions.

For the push-to-talk solution, we considered two approaches. In the first approach the radio interface module was used. The module has a push-to-talk-circuit which waits for the input audio signal to place the push-to-talk signal. This introduces a push-to-talk delay time measured between the time when the audio input signal is received and the time when the push-to-talk signal is set up. This delay is affected by the audio signal characteristics (especially volume level). It is in the range of 2 to 300 ms and for some situations it causes loss of the initial part of the audio signals. Because of the potentially large delays and loss of signal, the radio interface module is only used as a temporary solution for radios that do not support software controlled push-to-talk signaling (which is rarely the case).

The second approach uses the advantage of the Project54 software and the CIDBI. Whenever the Radio Control Application receives a push-to-talk message from other applications, it sends a new message to the radio through the CIDBI as a signal to start transmitting. In this approach, two push-to-talk paths were considered and delay times were measured.

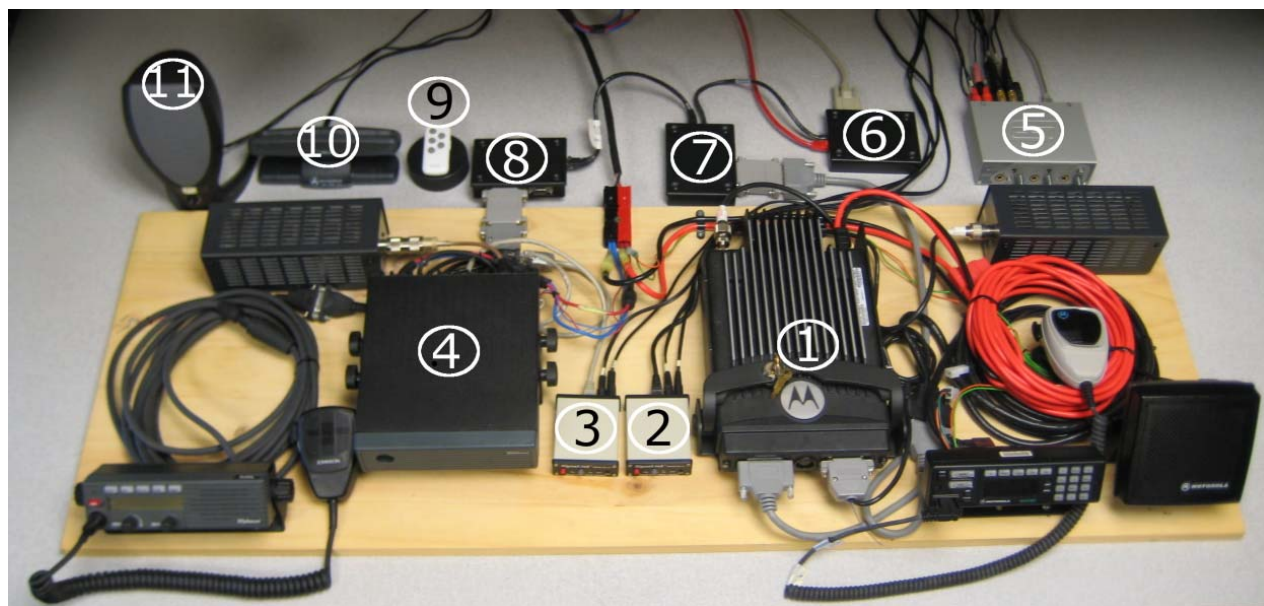


Figure 3 - Lab setup for testing the prototype system

These paths are for the radio patching and when transmission is initiated using the AirClick. In the case of radio patching the push-to-talk delay is around 55 ms. This is measured between the time when one radio detects transmission and sends the busy signal to the PC and the time when another radio receives a message from the PC to start transmission. When the AirClick is used for initiating the transmission the push-to-talk delay is almost constant around 155 ms. This is measured between the time when an AirClick button is pressed and the time when the radio starts transmission. These push-to-talk delays do not affect audio signal propagation and during the tests they did not decrease the system performance.

Also, we measured the audio signal path delay time (latency) between the all audio switch inputs and outputs [6]. This is the difference between the time when the audio signal is presented to an audio switch input and the time when the audio signal is detected on an output (e.g. In 1 and Out 2). The results show a constant audio latency of only 23.5 ms, which cannot be noticed in the radio communication system and is much less than 150 ms specified by ITU-T [7]. The audio latency times on the analog radio interface modules are negligible because they are less than 1 ms.

The Audio Control Application was tested under various scenarios using log file analysis [8], [9]. Every user action, radio transmission, the system configuration change, and system responses were logged. Once the tests were done, the analysis of the log file proved that system was reacting correctly to user actions and radio transmissions.

IV. CONCLUSION AND FUTURE WORK

Radio patching is the cheapest and fastest solution to the interoperability problem in police cruisers. However, this solution introduces a problem of operating additional radio devices inside the cruiser, each with its own user interface. This can be distracting for officers as they drive and perform other necessary tasks.

Our prototype is integrated into the Project54 system, which allows use of a single common user interface (a microphone, speaker, and push-to-talk button) to operate several different radios. The result is a system that should not only be less confusing but less distracting to officers as they carry out their assigned tasks within the police cruiser.

The system was developed and tested in laboratory conditions. Two different radios made by different companies were used in a proof of concept experiment. Propagation of the push-to-talk signal was monitored through the entire system as well as various audio signals. Test results showed that the push-to-talk delay and audio latency are constant and small enough to have no effect on the system's performance. Also, it was clearly shown that the system supports full radio patching without any limitations on what is routed where and how audio signals are mixed.

As the next step, we will examine the impact that the system has on the police officer's driving performance through driving simulator studies. Finally we will deploy the system in a real world scenario to test its operation. This will require setup in an actual police cruiser with at least two radio systems.

ACKNOWLEDGMENT

This work was supported by the U.S. Department of Justice under grant 2006DDBXK099.

REFERENCES

- [1] NTFOI, "Why Can't We Talk? Working Together To Bridge the Communications Gap To Save Lives," U.S Department of Justice, 2005.
- [2] Z. Medenica and A. L. Kun, "Comparing the influence of two user interfaces for mobile radios on driving performance," *Driving Assessment 2007*, Stevenson, WA, 2007.
- [3] A. L. Kun, W. T. Miller, III, and W. H. Lenharth, "Computers in police cruisers," *Pervasive Computing, IEEE*, vol. 3, no. 4, pp. 34-41, 2004.
- [4] A. Bouin and B. Lengrand, "Audio Switch," University of New Hampshire, ECE Department, Tech. Rep. ECE.P54.2006.6, Sept.2006.
- [5] Steinberg, "Steinberg Audio Streaming Input Output Specification: Development Kit 2.1," Steinberg Media Technologies GmbH, 2005.
- [6] K. MacMillan, M. Droettboom, and I. Fujinaga, "Audio Latency Measurements of Desktop Operating Systems," *Proceedings of the International Computer Music conference*, 2001.
- [7] ITU, "ITU-T G.144 General recommendations on the transmission quality for an entire international telephone connection," Geneva, 2003.
- [8] J.H. Andrews, "Testing using Log File Analysis: Tools, Methods, and Issues," In 13th Annual International Conference on Automated Software Engineering (ASE'98), Honolulu, Hawaii, 1998.
- [9] J. Z. Gao, H.-S. J. Tsao, and Y. Wu, *Testing And Quality Assurance For Component-Based Software*. Norwood, MA: ARTECH HOUSE, INC, 2003.