

Voice Alert system using PC

Noboru Yamamoto, KEK, Tsukuba, JAPAN

Abstract

Flexible voice alert system was build combining existing softwares in a very short time. When an alert condition occurs in the system, EPICS alarm handler program send UDP packet including alert message to a server program running on a Macintosh computer. The server program uses TextSpeech program to read it out as a human voice. Both client and server programs were written in Python programming language.

1 SYSTEM OVERVIEW

The voice alert system described in this paper is the system notifies alert conditions to the operator with the vocal alert messages.

Main component of the system is a software on PC (Macintosh computer running MacOS 8) receives an alert message from a control system over a network and read it as a vocal output. This functionality is supported by the standard function included in the MacOS 8, MacinTalk. The program can accept any string of english words and convert it into synthetic voice sound. This adds the great flexibility to the system, i.e. no pre-recorded voice or sentences are required.

The server uses TCP/UDP as a communication mechanism. A clients program simply send an appropriate string as an UDP packet. No complicated protocol is required for this communication.

The both server and client programs are written in a scripting language "Python"[2]. Python was chosen because: 1) it runs on several platforms including Macintosh and Unixes. 2) its standard library includes socket interface library for network communication, and 3)Macintosh version of Python supports MacinTalk library.

Clean syntax and support of object oriented programing in the language also make Python an attractive choice.

The system was tested combining with EPICS[1] ALH(ALarm Handler) program. However, any program which can run the client program with an alert message within it, can be used. After this test system, voice alert system for KEK-LINAC and KEKB ring operation system was build and has been used in the KEKB control system.

2 PYTHON: OBJECT ORIENTED INTERPRETER LANGUAGE

According to the WWW home page [3] for Python language,

"Python is an interpreted, interactive, object-oriented programming language. It is often com-

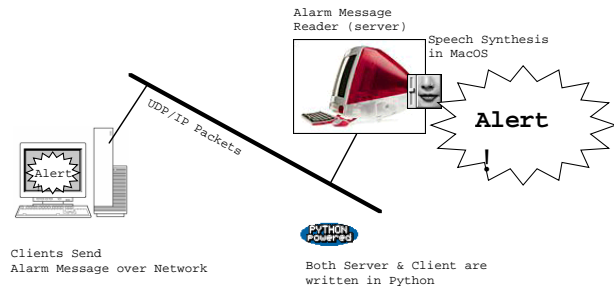


Figure 1: System overview of a vocal alert system

pared to Tcl, Perl, Scheme or Java. ” “Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC). New built-in modules are easily written in in C or C++. Python is also usable as an extension language for applications that need a programmable interface. ”

“Python is copyrighted but freely usable and distributable, even for commercial use. ”

Despite the domination of Perl and Tcl in recent script language world, Python is getting popularity because of it cleanness and ease of use. In KEKB control system, Python is used for developing applications for operator interfaces.

The following segment of codes is a complete python program which generate “Hello World” message on a terminal.

```
#!/python

def Hello():
    print ``Hello World``

Hello()
```

The first line of the code is a comment in Python. The following two lines define a function named Hello. The function Hello() has no argument and print out the string on the standard output. A block structure of the function body is represented by indentation and has no begin-end or braces. The last line makes a call to Hello() and prints out the messages.

Python supports list of objects and mapping (associative array) as a primitive data type in the language. Use of these

types, especially mapping data type, and a built-in garbage collector allows users to concentrate on solving the problem rather than programming.

Python standard library supports socket interface for network communication. The following code segment illustrates the use of socket library in the Python. It is a client program for this system. It just send a message to a specified host using UDP socket. The function socket() creates a socket object. The sendto method of the socket object is used to send message at a specified address.

```
def client():
    if len(sys.argv) < 3:
        usage()
    host = sys.argv[2]
    if len(sys.argv) > 3:
        port = eval(sys.argv[3])
    else:
        port = ECHO_PORT
    line = sys.stdin.readline()
    if line:
        send_message(line, host, port)

def send_message(line, host='', port=ECHO_PORT):
    addr = (host, port,)
    s = socket(AF_INET, SOCK_DGRAM)
    if line:
        s.sendto(line, addr)
```

3 MACINTOSH SPEECH MANAGER INTERFACE

Macintosh Speech Manager (formally known as MacinTalk) is a name of library in MacOS for voice synthesis. It consists of two parts, voice synthesis part and sentence analysis part. Corresponding to these components, Python library for Mac speech manager includes two objects, a voice object and a speech channel object. A Macintosh system may have several voices, each of which has different characteristics. GetIndVoice() function creates a voice object specified by its argument.

A speech channel is an object that the Speech Manager uses when processing text. It is associated with a particular voice and particular speech attributes, pitch and rate. To convert text string into a vocal sound, SpeakText() method of Speech Channel object is used.

```
import sys, select, time
from socket import *
import macspeech

ECHO_PORT = 50000 + 7
BUFSIZE = 1024

def InitVoices():
    global spChan
    spChan = macspeech.GetIndVoice(1).NewChannel()

def server():
    port = ECHO_PORT
```

```
s = socket(AF_INET, SOCK_DGRAM)
s.bind('', port)
while 1:
    global spChan
    ready = select.select([s.fileno()], \
                          [], [], 0.5)
    if (ready[0] <> []):
        data, addr = s.recvfrom(BUFSIZE)
        spChan.Stop()
        spChan.SpeakText(data)
```

The server receives UDP datagram from a socket when data is available. It just passes this data to Speech Manager and MacOS takes care of the rest.

In the test we configured EPICS Alarm Handler program (ALH) to send alarm messages to the voice alarm server when some of EPICS channels generate alarm.

All of these developments have been done in a very short period, a single business day.

4 CONCLUSION

Using the standard functionality built in to the Mac OS, we could write a voice alert system in a very short period (just a single day). Choice of Tool, Python, was also important for this rapid development. Its rich set of library functions and data types reduces a need to write codes.

Cost of this system is also quite low, you can get Python and its library for free.

Although improvement of sound quality in the synthesized voice, this mechanism can be used in the daily operation of KEKB accelerators.

5 REFERENCES

- [1] B. Dalesio et al. "Distributed Software Development in the EPICS Collaboration," in Proceedings of the 1995 International Conference on Accelerator and Large Experimental Physics Control Systems, M. Crowley-Milling, P. Lucas and P. Schoessow, Eds., pp 360 - 366. ; "EPICS home page", <http://epics.aps.anl.gov/asd/controls/epics/Epics-Documentation/WWWPages/>
- [2] M. Lutz, "Programming Python", O'Reilly & Associates, Inc. USA, 1996; "Python Home Page"
- [3] The URL of the official Python home page is <http://www.python.org/>.