"Industrial Robot, An International Journal", Emerald Group Publishing Limited, Volume 32, Number 6, 2005

Robot-by-voice: Experiments on commanding an industrial robot using the human voice

J. Norberto Pires
Mechanical Engineering Department
and Mechanical Engineering Research Center
(a research center from the Portuguese Foundation for Science and Technology)
University of Coimbra, Portugal

1. Structured Abstract

Purpose
This paper reports a few results of an ongoing research project that aims to explore ways to command an industrial robot using the human voice. This feature can be interesting with several industrial, laboratory and clean-room applications, where a close cooperation between robots and humans is desirable.

Methodology / approach
A demonstration is presented using two industrial robots and a personal computer (PC) equipped with a sound board and a headset microphone. The demonstration was coded using the Microsoft Visual Basic and C# .NET 2003 and associated with two simple robot applications: one capable of picking-and-placing objects and going to predefined positions, and the other capable of performing a simple linear weld on a work-piece. The speech recognition grammar is specified using the grammar builder from the Microsoft Speech SDK 5.1.

The paper also introduces the concepts of text-to-speech translation and voice recognition, and shows how these features can be used with applications built using the Microsoft .NET framework.

Findings
Two simple examples designed to operate with a state-of-the-art industrial robot manipulator are then built to demonstrate the applicability to laboratory and industrial applications. The paper is very detailed in showing implementation aspects enabling the reader to explore immediately from the presented concepts and tools. Namely, the connection between the PC and the robot is explained in detail since it was built using a RPC socket mechanism completely developed from the scratch.

Practical implications
Finally, the paper discusses application to industrial cases where close cooperation between humans and robots is necessary.

Originality / value of the paper
The presented code and examples, along with the fairly interesting and reliable results, indicate clearly that the technology is suitable for industrial utilization.

Keywords: Speech recognition, industrial robotics, distributed software.

2. Introduction

Talking to machines is a thing normally associated with science fiction movies and cartoons and less with current industrial manufacturing systems. In fact, most of the papers about speech recognition start with something related with artificial intelligence, or a science fiction movie, or a robot used in a movie, etc., where machines talk like humans, understand the complex human speech without problems. Nevertheless, industrial manufacturing systems would benefit very much from speech recognition for human-machine interface (HMI) even if the technology is not so advanced. Gains in terms of autonomy, efficiency and agility seem evident. The modern world requires better products at lower prices, requiring even more efficient manufacturing plants because the focus is in achieving better quality products, using faster and cheaper procedures. This means autonomy, having systems that require less operator intervention to operate normally, better human-machine interfaces and cooperation between humans and machines sharing the same workspace as real co-workers.

The final objective is to achieve, in some cases, semi-autonomous systems [1], i.e., highly automated systems that require only minor operator intervention. In many industries, production is closed tracked in any part of the manufacturing cycle, which is composed by several in-line manufacturing systems that perform the necessary operations, transforming the raw materials in a final product. In many cases, if properly designed, those individual manufacturing systems require simple parameterization to execute the tasks they are designed to execute. If that parameterization can be commanded remotely by automatic means from where it is available, then the system becomes almost autonomous in the sense that operator intervention is reduced to the minimum and essentially related with small adjustments, error and maintenance situations [1]. In other cases, a close cooperation between humans and machines is desirable although very difficult to achieve, due to limitations of the actual robotic and automation systems.

The above described scenario puts focus on HMI, where speech interfaces play a very important role because the manufacturing systems efficiency will increase if the interface is more natural or similar to the human way of commanding things. Nevertheless, speech recognition is not a common feature among industrial applications, namely because:

1. The technologies of speech recognition and text-to-speech are relatively new, although they are already robust enough to be used with industrial applications;
2. The industrial environment is very noisy which puts enormous difficulties to automatic speech recognition systems;
3. The industrial systems weren't design to incorporate these types of features, and usually don't have powerful computers especially dedicated to HMI.

Automatic Speech Recognition (ASR) is commonly described as converting speech to text. The reverse process, in which text is converted to speech (TTS), is known as speech synthesis. Speech synthesizers often produce results that are not very natural sounding. Speech synthesis is different from voice processing, which involves digitizing, compressing (not always), recording, and then playing back snippets of speech. Voice processing results are very natural sounding, but the technology is limited in flexibility and is disk storage-space-intensive compared to speech synthesis.

Speech recognition developers are still searching for the perfect human-machine interface, a recognition engine which understands any speaker, interprets natural speech patterns, remains impervious to background noise, and has an infinite

vocabulary with contextual understanding. However, practical product designers, OEMs, and VARs can indeed use today's speech recognition engines to make major improvements to today's markets and applications. Selecting such an engine for any product requires understanding how the speech technologies impact performance and cost factors, and how these factors fit in with the intended application.

That is the approach used in this technical paper. An automatic speech recognition system is selected and used for the purpose of commanding a generic industrial manufacturing cell. The concepts are explained in detail and two test case examples are presented and discussed in a way to show that if certain measures are taken, ASR can be used with great success even with industrial applications. Noise is still a problem, but using a short command structure with a specific word as pre-command string it is possible to reduce enormously the noise effects. The system presented in this paper uses this strategy and was tested with a simple noiseless pick-and-place example, but also with a simple welding application where considerable noise is present.

3. Automatic speech recognition system and strategy

From the several continuous speech ASR technologies available, based on personal computers, the Microsoft Speech Engine [2] was selected because it integrates very well with the operating systems we use for HMI, manufacturing cell control and supervision (Windows XP/NT/2000). The Microsoft Speech Application Programming Interface (SAPI) was also selected, along with the Microsoft's Speech SDK (version 5.1), to develop the speech and text to voice software applications [2]. This API provides a nice collection of methods and data structures that integrate very well in the .NET 2003 framework [11], providing a very interesting developing platform that takes advantage of the computing power available from actual personal computers. Finally, the Microsoft's SAPI 5.1 works with several ASR engines, which gives some freedom to developers to choose the technology and the speech engine to use.

Grammars define the way the ASR recognizes the speech from the user. When a sequence included in the grammar is recognized, the engine originates an event that can be handled by the application to perform the planned actions. The SAPI provides the necessary methods and data structures to extract the relevant information from the generated event, so that proper identification and details are obtained. There are three ways to define grammars: using XML files, using binary configuration files (CFG) or using the grammar builder methods and data structures. XML files are a good idea to define grammars if a compiler and converter is available like in the SDK 5.1. In the examples provided in this paper, the grammar builder methods were used to programmatically construct and modify the grammar.

The strategy used here takes in consideration that there should be several robots in the network, running different applications. In that *scenario*, the user needs to identify the robot first, before sending the command. The following strategy is used,

1. All commands start with the word "Robot";
2. The second word identifies the robot by a number: one, two, etc.
3. The following words constitute the command and the parameters associated with a specific command.

Consequently, the grammar used is composed by a "*TopLevelRule*" with a predetermined *initial state*, i.e., the ASR system looks for the pre-command word

"Robot" as a pre-condition to any recognizable command string. The above mentioned sequence of words constitutes the *second level rules*, i.e, they are used by the *TopLevelRule* and aren't directly recognizable. A rule is defined for each planned action. As a result, the following represents the defined syntax of commands:

Robot *number command parameter_i*

where "robot" is the pre-command word, *number* represents the robot number, *command* is the word representing the command to send to the robot, and *parameter_i* are *i* words representing the parameters associated with the *command*.

Another thing considered was safety. Each robot responds to "*hello*" commands, and when asked to "*initialize*" the robots require voice identification of *username* and *password* to be able to give to the user the proper access rights. Since the robots are connected to the calling PC using a RPC sockets [3-5] mechanism, the user must "initialize" the robot to start using its remote services, which means that an RPC connection is open, and must "terminate" the connection when no more actions are needed. A typical session would look like,

User: Robot one hello.
Robot: I am listening my friend.
User: Robot one initialize.
Robot: You need to identify to access my functions.
Robot: Your username please?
User: Robot one <username>.
Robot: Correct.
Robot: Your password please?
User: Robot one <password>.
Robot: Correct.
Robot: Welcome again <username>. I am robot one. Long time no see.

Sequence of commands here. Robot is under user control.

User: Robot one terminate.
Robot: See you soon <username>.

In the following sections, two simple examples are given to demonstrate how this voice command mechanism is implemented, and how the robot controller software is designed to allow these features.

4. Pick-and-place and robotic welding examples

The following examples take advantage of developments done in the Industrial Robotics Laboratory, of the Mechanical Engineering Department of the University of Coimbra around robot remote access for command and supervision [5-8]. Briefly, two industrial robots connected to an Ethernet network are used. The robot controllers (ABB S4CPlus) are equipped with RPC servers that enable user access from the network, offering several interesting services like variable access services, IO access services, programs and files access services and system status services [9]. The new versions of the ABB controller, named IRC5, are equipped with a TCP/IP sockets API [9], enabling users to program and setup TCP/IP sockets servers in the controller. For that reason, the ideas presented here can be easily transported to the new IRC5 controller with no major change.

If calls to those services are implemented in the client PC, it is fairly easy to develop new services. The examples presented here include the ActiveX PCROBNET2003 [5] that implement the necessary methods and data structures (see Table 1) to access all the services available from the robot controller.

The basic idea is simple and not very different from the concept used when implementing any remote server. If the system designer can access robot program variables, then he can design his own services and offer them to the remote clients. A simple SWITCH-CASE-DO cycle, driven by a variable controlled from the calling client, would do the job:

```
switch (decision_1)
{
    case 0: call service_0; break;
    case 1: call service_1; break;
    case 2: call service_2; break;
    ...
    case n: call service_n; break;
}
```

Table 1. Methods and properties of the software component PCROB NET2003 [10]

Function	Brief Description
open	Opens a communication line with a robot (RPC client)
close	Closes a communication line.
motor_on	Go to Run State
motor_off	Go to Standby State
prog_stop	Stop running program
prog_run	Start loaded program
prog_load	Load named program
prog_del	Delete loaded program
prog_set_mode	Set program mode
prog_get_mode	Read actual program mode
prog_prep	Prepare Program to Run (Program Counter to begin)
pgmstate	Get Program Controller State
ctlstate	Get Controller State
oprstate	Get Operational State
sysstate	Get System State
ctlvers	Get Controller Version
ctlid	Get Controller ID
robpos	Get current robot position
read_xxxx	Read variable of type xxxx (there are calls for each type of variable defined in RAPID [10])
read_xdata	Read user defined variables
write_xxx	Write variable of type xxxx (there are calls for each type of variable defined in RAPID [10])
write_xdata	Write user defined variables
digin	Read digital input
digout	Set digital output
anain	Read analog input
anaout	Set analog output

4.1. Pick-and-place example

For example, consider a simple pick-and-place application. The robot, equipped with a two finger pneumatic gripper, is able to pick a piece from one position

(called origin) and deliver it to other position (called final). Both positions are placed on top of a working table (Figure 1).

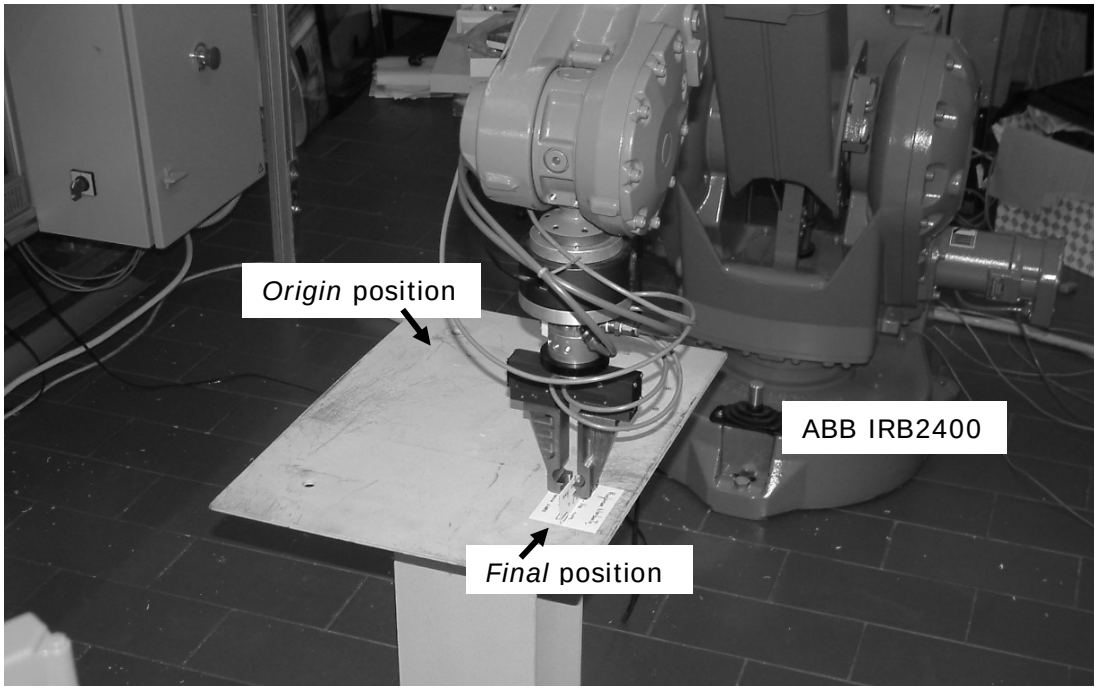


Figure 1: Working table for the simple pick-and-place application.

The robot can be commanded to open/close the gripper, approach origin/final position (positions 100 mm above of origin/final position, respectively), move to origin/final position, and move to "home" (safe position away from the table.). This is a very simple example, but sufficient to demonstrate the voice interface. Figure 2 shows a simplified version of the server software running on the robot controller.

To be able to send any of those commands using the human voice, the following grammar was implemented:

TopLevelRule = "Robot"	pre-command word
Rule 0 = "one hello"	check if robot is there
Rule 1 = "one initialize"	ask robot to initialize (open client)
Rule 2 = "one master"	rule defining username "master"
Rule 3 = "one masterxyz"	password of username "master"
Rule 4 = "one open"	open the gripper
Rule 5 = "one close"	close the gripper
Rule 6 = "one motor on"	put robot in run state
Rule 7 = "one motor off"	put robot stand-by state
Rule 8 = "one program run"	start program
Rule 9 = "one program stopt"	stop program
Rule 10 = "one approach origin"	call service 94
Rule 11 = "one approach final"	call service 93
Rule 12 = "one origin"	call service 91
Rule 13 = "one final"	call service 92
Rule 14 = "one home"	call service 90
Rule 14 = "one terminate"	release robot access (close client)

```
PROC main()
    TPErase; TPWrite "Example Server ...";
    p1:=CRobT(\Tool:=trj_tool\WObj:=trj_wobj);
    MoveJ p1,v100,fine,trj_tool\WObj:=trj_wobj;
    decision1:=123;
    WHILE TRUE DO
      TEST decision1
        CASE 90:
          MoveJ home,v200,fine,tool0;
          decision1:=123;
        CASE 91:
          MoveL final,v200,fine,tool0;
          decision1:=123;
        CASE 92:
          MoveL origin,v200,fine,tool0;
          decision1:=123;
        CASE 93:
          MoveJ Offs(final, 0,0,100),v200,fine,tool0;
          decision1:=123;
        CASE 94:
          MoveJ Offs(origin, 0,0,100),v200,fine,tool0;
          decision1:=123;
     ENDTEST
    ENDWHILE
ENDPROC
```

Figure 2: Simple pick-an-place server implemented in RAPID [10].

The presented rules were introduced into a new grammar using the grammar builder included in the Microsoft Speech API (SAPI) [2]. The following (Figure 3) shows how that can be done, using the Microsoft Visual Basic .NET2003 compiler.

```
TopRule = Grammar.Rules.Add("TopLevelRule", SpeechLib.SpeechRuleAttributes.SRATopLevel Or
SpeechLib.SpeechRuleAttributes.SRADynamic, 1)
ListItemsRule = Grammar.Rules.Add("ListItemsRule", SpeechLib.SpeechRuleAttributes.SRADynamic,
2)
AfterCmdState = TopRule.AddState
m_PreCommandString = "Robot"
TopRule.InitialState.AddWordTransition(AfterCmdState, m_PreCommandString, " ", , "", 0, 0)
AfterCmdState.AddRuleTransition(Nothing, ListItemsRule, "", 1, 1)
ListItemsRule.Clear()

ListItemsRule.InitialState.AddWordTransition(Nothing, "one hello", " ", , "one hello", 0, 0)
...
Grammar.Rules.Commit()
Grammar.CmdSetRuleState("TopLevelRule",SpeechLib.SpeechRuleState.SGDSActive)
RecoContext.State() = SpeechLib.SpeechRecoContextState.SRCS_Enabled
```

Figure 3: Adding grammar rules and compiling the grammar using SAPI in Visual Basic .NET2003.

After committing and activating the grammar, the ASR listens for voice commands and generates speech recognition events when a programmed command is recognized. The correspondent event service routines execute the commanded strings. Figure 4 shows the shell of the application built in Visual Basic .NET 2003 to implement the voice interface for this simple example. Two robots are listed in the interface. The robot executing the simple pick-and-place example is robot one (named *Rita*).

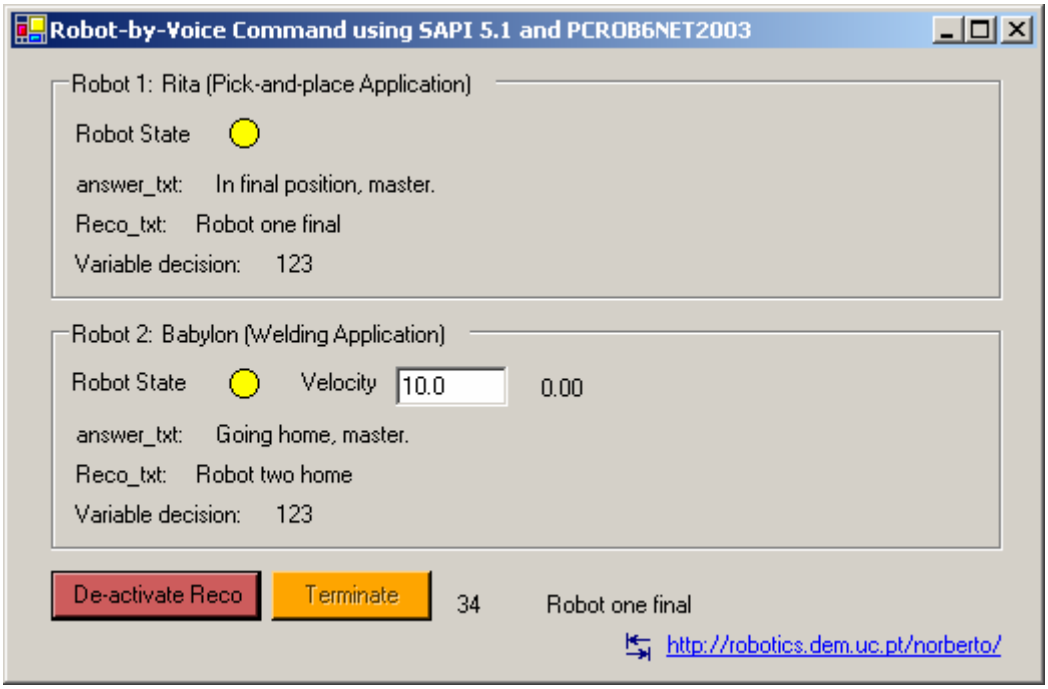


Figure 4: Shell of the voice interface application used to command the robot.

With this interface activated the following sequence of commands (admitting that the logging procedure was already executed) will take the robot from the "home" position, pick the work object at the origin position, deliver it to the final position, return to "home" and release the robot control.

User: Robot one approach origin.
Robot: Near origin, master.
User: Robot one open.
Robot: Tool open master.
User: Robot one origin.
Robot: In origin position master.
User: Robot one close.
Robot: Tool close master.
User: Robot one approach origin.
Robot: Near origin, master.
User: Robot one approach final.
Robot: Near final, master.
User: Robot one final.
Robot: In final position, master.
User: Robot one approach final.
Robot: Near final, master.
User: Robot one home.
Robot: In home position, master.
User: Robot one terminate.
Robot: See you soon master.

The speech event routine, running on the voice interface application is called when any of the rules defined in the working grammar is recognized. For example, when the "motor on" rule is identified the following routine is executed,

```
If ok_command_1 = 1 And (strText = "Robot one motor on") Then
   result1 = Pcrobnet2003.MotorON2(1)
   If result1 >= 0 Then
      Voice.Speak("Motor on, master.")
```

```
            ans_robot_1.Text() = "Motor ON, master."
         Else
            Voice.Speak("Error executing, master.")
            ans_robot_1.Text() = "Error executing, master."
         End If
      End If
```

To give another example, when the move to "origin" rule is recognized the following routine is executed,

```
      If ok_command_1 = 1 And (strText = "Robot one origin") Then
         Dim valor As Integer
         valor = 92
         result1 = Pcrobnet2003.WriteNum2("decision1", valor, 1)
         If result1 >= 0 Then
            Voice.Speak("In origin position, master.")
            ans_robot_1.Text() = "In origin position, master."
         Else
            Voice.Speak("Error executing, master.")
            ans_robot_1.Text() = "Error executing, master."
         End If
      End If
```

4.2. Robotic welding example

The welding example presented here extends slightly the functionality of the simple server presented in Figure 2, just by adding another service and the necessary routines to control the welding power source. The system used for this demonstration is composed by an industrial robot ABB IRB1400 equipped with the robot controller ABB S4CPlus, and a MIG/MAG welding power source (ESAB LUA 315A). The work-piece is placed on top of a welding table, and the robot must approach point 1 (called origin) and perform a linear weld from that point until point 2 (called final). The system is presented in Figure 5. The user is able to command the robot to,

1. Approach and reach the point origin (P1);
2. Approach and reach the point final (P2);
3. Move to "home" position;
4. Perform a linear weld from point P1 (origin) to point P2 (final);
5. Adjust and read the value of the welding velocity.

These actions are only demonstration actions selected to show further details about the voice interface to industrial robots. To implement the simple welding server it is enough to add the following welding service to the simple server presented in Figure 2,

```
      CASE 94:
         weld_on;
         MoveL final,v200,fine,tool0;
         weld_off;
         decision1:=123;
```

where the routine "weld_on" makes the necessary actions to initiate the welding arc [8], and the routine "weld_off" performs the post welding actions to finish the welding and terminate the welding arc [8].

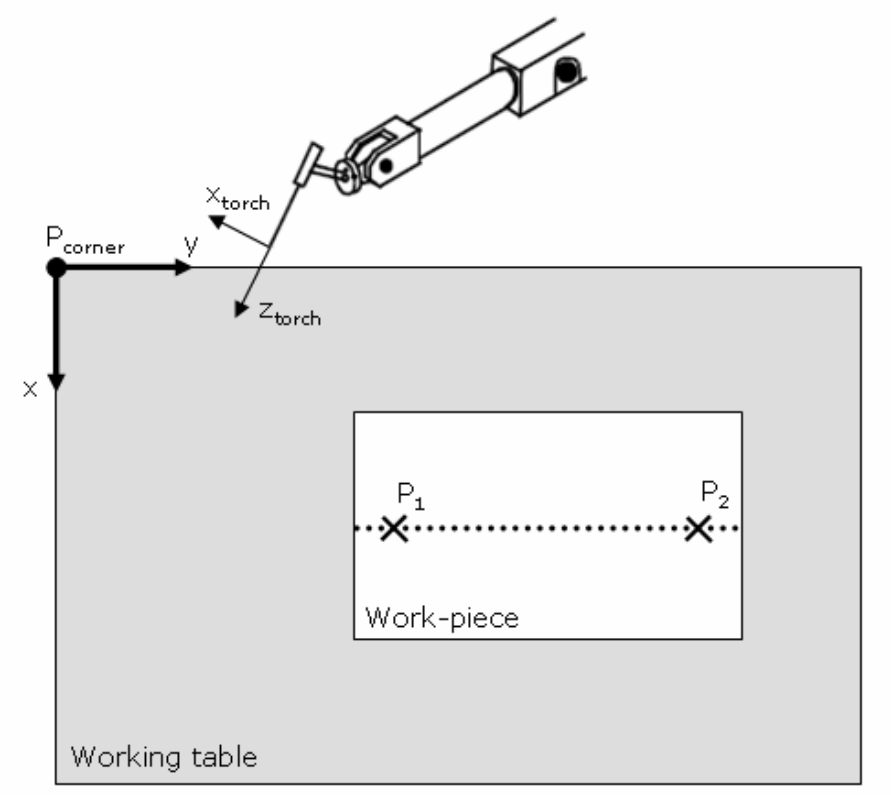
The welding server is running in robot 2 (named *babylon*), and is addressed by that number from the voice interface application (Figure 6). To execute a linear weld from P1 to P2, at 10mm/s, the user must command the following actions (after logging to access the robot, and editing the velocity value in the voice interface application – Figure 6) using the human voice:

User: Robot 2 approach origin.
Robot: Near origin master.
User: Robot 2 origin.
Robot: In origin position master.
User: Robot 2 velocity.
Robot: Velocity changed master.
User: Robot 2 weld.
Robot: I am welding master.
User: Robot 2 approach final.
Robot: Near final master.

Figure 5 shows the voice interface when robot 2 is actually welding along with a user equipped with a handset microphone to send voice commands to the robot. The code associated with the welding command is,

```
If ok_command_2 = 1 And (strText = "Robot two weld") Then
   Dim valor As Integer
   valor = 95
   result1 = Pcrobnet2003.WriteNum2("decision1", valor, 2)
   If result1 >= 0 Then
      Voice.Speak("I am welding, master.")
      ans_robot_2.Text() = "I am welding, master."
   Else
      Voice.Speak("Error executing, master.")
      ans_robot_2.Text() = "Error executing, master."
   End If
End If
```

The code above writes the value 95 to the variable "decision1", which means that the service "weld" is executed (check Figure 2).



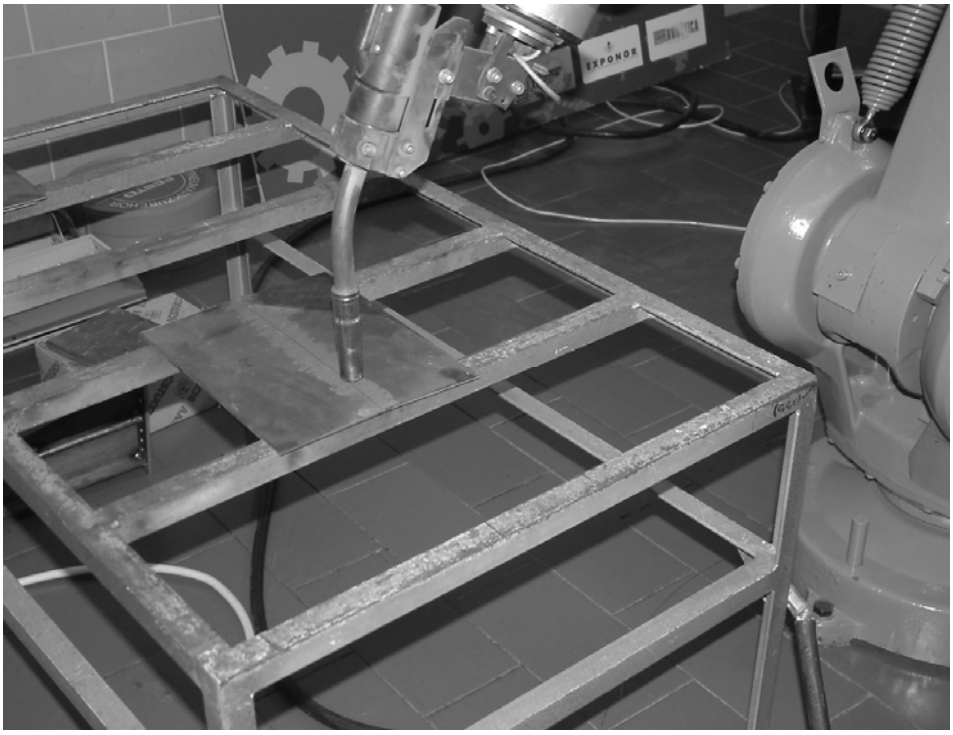


Figure 5: Simple welding application used for demonstration.

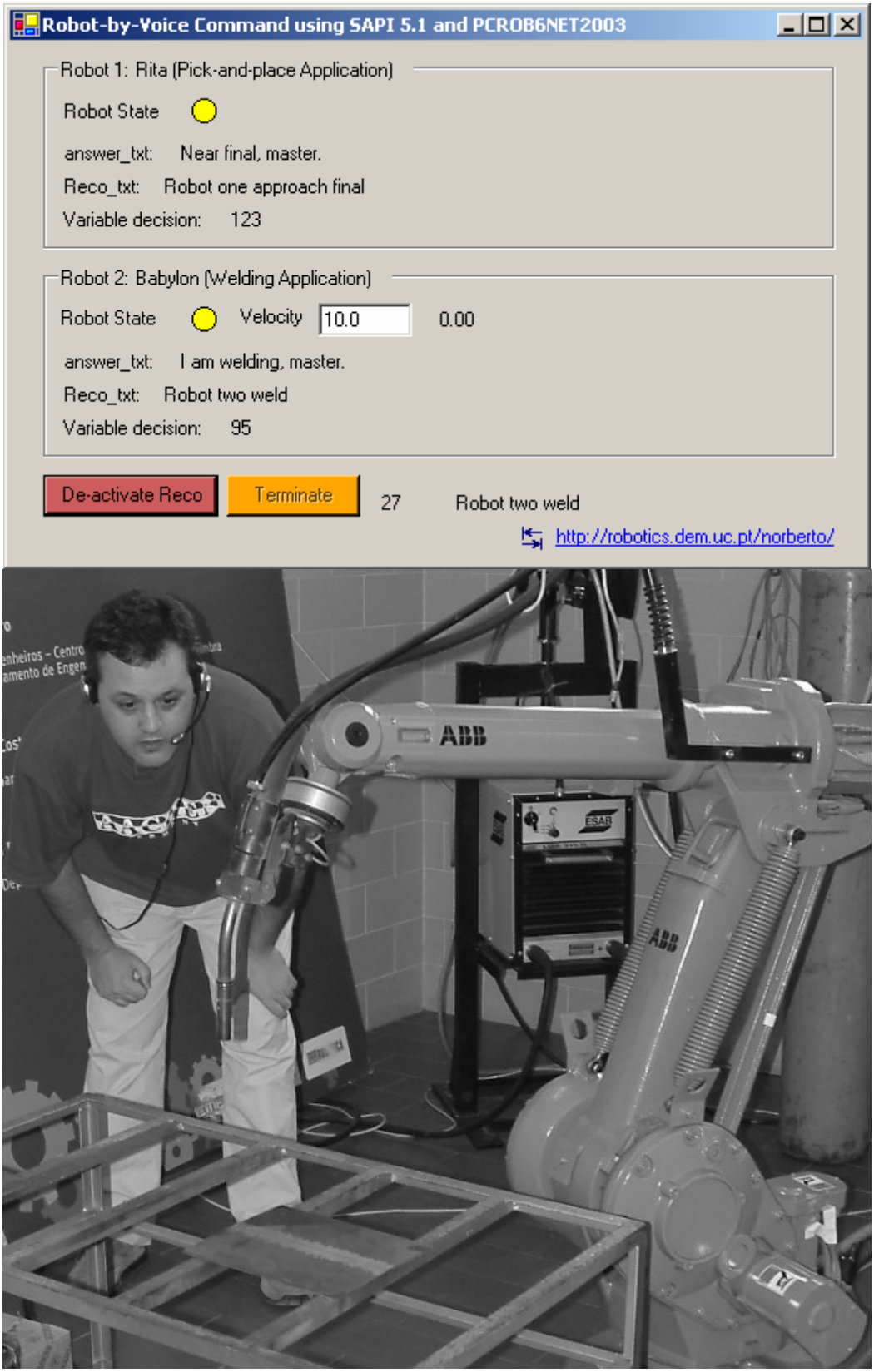


Figure 6: Shell of the voice interface application showing the welding operation, and a user (author of this paper) commanding the robot using a headset microphone.

4.2.1 Adjusting Process Variables

During the welding process it may be needed to adjust process variables like the welding velocity, welding current, the welding points, etc. This means that the voice interface must allow users to command numerical values that are difficult to recognize with high accuracy. Furthermore, it is not practical to define fixed rules for each possible number to recognize, which means that dictation capabilities must be active when the user wants to command numbers. To avoid noise effects, and consequently erroneous recognition, a set of rules were added to enable dictation only when necessary, having the rule strategy defined above always active. Consequently, the following rules were added for robot 2 (the one executing the welding example):

Rule V1 = "two variables"	enables access to variables
Rule V2 = "two variables out"	terminates access to variables
Rule V3 = "two <variable_name>"	enables access to <variable_name>
Rule V4 = "two <variable_name> lock"	terminates access to <variable_name>
Rule V5 = "two <variable_name> read"	reads from <variable_name>
Rule V6 = "two <variable_name> write"	writes to <variable_name>

Rules V1 and V2 are used to activate/deactivate the dictation capabilities, which will enable the easy recognition of numbers in decimal format (when the feature is activated, a white dot appears in the program shell – Figure 7). Rules V3 and V4 are used to access a specific variable. When activated, each number correctly recognized is added to the text box associated with the variable (a blinking led appears in the program shell – Figure 7). Deactivating the access, the value is locked and can be written to the robot program variable under consideration. The rules V5 and V6 are used to read/write the actual value of the selected variable from/to the robot controller.

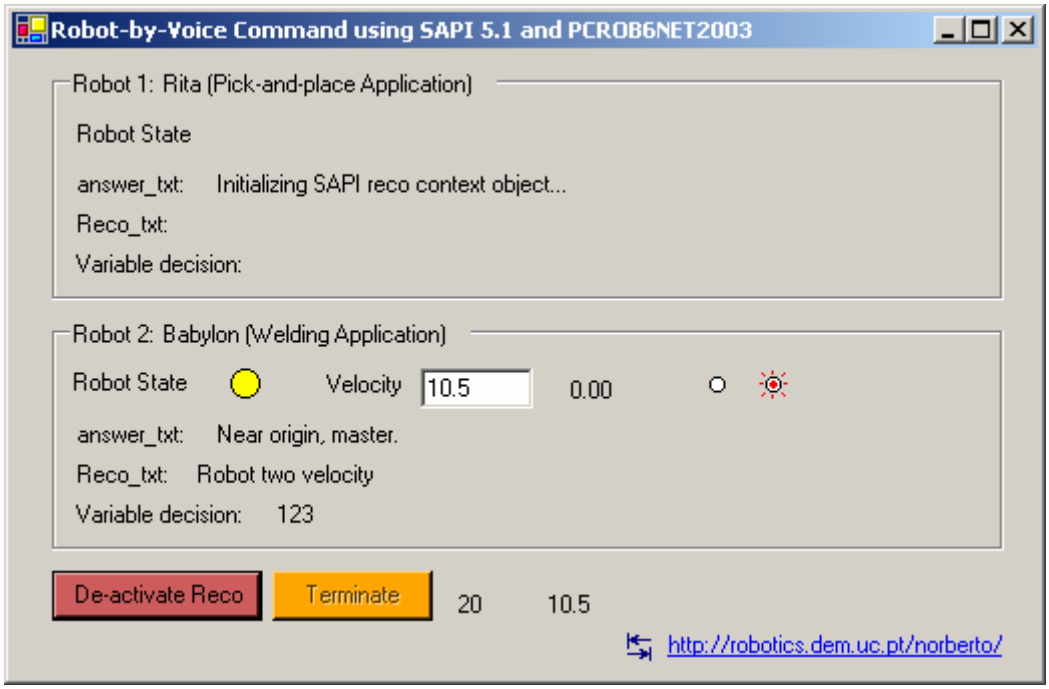


Figure 7: Accessing variables in the robot controller.

As an example, to adjust the welding velocity the following code is executed after the correspondent rule is recognized,

```
If ok_command_2 = 1 And (strText = "Robot two velocity write") Then
   Dim valor As Double
   Dim velocity as Integer
    valor = velocity.Text()
   result1 = Pcrobnet2003.WriteSpeed("velocity", valor, 2)
   If Result11 >= 0 Then
      Voice.Speak("Welding velocity changed, master.")
      ans_robot_2.Text() = "Welding velocity changed, master."
   Else
      Voice.Speak("Error executing, master.")
      ans_robot_2.Text() = "Error executing, master."
   End If
End If
```

Since the voice interface was designed to operate with several robots, two in the present case, the user may send commands to both robots using the same interface which is potentially interesting.

Using speech interfaces is a big improvement to HMI systems, because of the following reasons:

1. Speech is a natural interface, similar to the "interface" we share with other humans, that is robust enough to be used with demanding applications. That will change drastically the way humans interface with machines;
2. Speech makes robot control and supervision possible from simple multi-robot interfaces. In the presented cases common PC's were used, along with a quite normal noise-suppressing headset microphone;
3. Speech reduces the amount and complexity of different HMI interfaces, usually developed for each application. Since a PC platform is used, which carry currently very good computing power, ASR systems become affordable and very simple to use.

The experiments performed with this interface worked extremely well, even when high noise was involved namely during welding applications, which indicates clearly that the technology is suitable to use with industrial applications where human-machine cooperation is necessary or where operator intervention is minimal.

5. Conclusion

In this paper a voice interface to command robotic manufacturing cells was designed and presented. The speech recognition interface strategy used was briefly introduced and explained. Two selected industrial representative examples were presented and fully demonstrated, with the objective of clarifying the potential interest of these human-machine interfaces for industrial applications.

Details about implementation were presented in a way to enable the reader to immediately explore from the discussed concepts and examples. Since a personal computer platform is used, along with standard programming tools (Microsoft Visual Studio .NET2003 and Speech SDK 5.1) and an ASR system freely available (SAPI 5.1), the whole implementation is affordable even for SME utilization.

The presented code and examples, along with the fairly interesting and reliable results, indicate clearly that the technology is suitable for industrial utilization.

6. References

[1] Pires, JN, "Semi-autonomous manufacturing systems: the role of the HMI software and of the manufacturing tracking software", Elsevier and IFAC Journal Mechatronics, to appear 2005.
[2] Microsoft Speech Application Programming Interface (API) and SDK, Version 5.1, Microsoft Corporation, http://www.microsoft.com/speech
[3] Bloomer J., "Power Programming with RPC", O'Reilly & Associates, Inc., 1992.
[4] RAP, Service Protocol Definition, ABB Flexible Automation, 1996 - 2004.
[5] Pires, JN, "PCROBNET2003, an ActiveX Control for ABB S4 Robots", Internal Report, Robotics and Control Laboratory, Mechanical Engineering Department, University of Coimbra, April 2004.
[6] Pires, JN, "Complete Robotic Inspection Line using PC based Control, Supervision and Parameterization Software", Elsevier and IFAC Journal Robotics and Computer Integrated Manufacturing, Vol. 21, Nº1, 2005
[7] Pires, JN, "Handling production changes on-line: example using a robotic palletizing system for the automobile glass industry", Assembly Automation Journal, MCB University Press, Volume 24, Number 3, 2004.
[8] Pires JN, et al, "Welding Robots", Springer-Verlag, UK, 2005.
[9] ABB IRC5 Documentation, ABB Flexible Automation, 2005.
[10] ABB RAPID Programming Manual, ABB Flexible Automation, 2005.
[11] Microsoft Studio .NET 2003, TechNet On-line Documentation, Microsoft Corporation, http://www.microsoft.com, 2003.