

Robix Arm Interface and Robotic Feedback Systems

James Barr
Illinois State University
213 Clay Hall
Normal, IL, 61761
(847) 858-9670
jrbarr@ilstu.edu

ABSTRACT

In this paper I describe the Robix robotic arm, the Robix interface software, and real-life applications of the software.

Categories and Subject Descriptors

B.0 [Hardware]: General – *general hardware*.

General Terms

Documentation, Performance, Design.

Keywords

Hardware, Architecture, Robot, Robix, Feedback System

1. INTRODUCTION

Iris.4 is the name of the mobile robot that the MIND Project is building. It is guided by the central control program (CCP) and mind module software. Iris.4 is also controlled by many other clients that operate via the CCP. The robotic body controlled by the artificial intelligence system provides methods for it to interact with the environment. Iris.4 can receive information from its eyes (the camera) such as the state of a tic-tac-toe board or verbal commands from its ears (the microphone), and it can manipulate that information in many ways. One way, for example, is to use its robotic arm to draw an 'X' in a square on the board, which is detected by its camera.

All of that transferring of information that is done between the many communicating clients is processed via the CCP. The CCP is a combination of a message passer and a server. It is a message passer because it allows for messages to be sent from one client to another, even if the clients are written in two different programming languages. This is possible as long as those languages support socket programming through TCP/IP. The CCP is also a server because it waits for various clients to connect, then allows for their operation. More information about the CCP was written by Bob Arrigo and can be found here.

2. ROBIX ARM

The Robix arm is designed to mimic a person moving their arm and performing various tasks. The design of the arm module also keeps in mind the principles of top down

artificial intelligence. This means that the way the CCP, the robix arm, and the robix interface program communicate with each other is not a model of how humans move their arms, rather it is meant to provide the same functionality of a human's arm. The robix arm should be able to complete tasks similar to the tasks that a human can do. Some examples of things that the robix robotic arm will be used to do are picking up and rearranging blocks and drawing X's and O's with a pen. Humans receive their messages from certain sequences of neurons firing. The robix arm will not receive its messages from a system like that of a human. Instead, it will receive messages that pass through the mind module, the central control program, and a robix interface computer program.

The robix arm is made from many different parts. It consists of a group of erector-set-type metal links that are joined together by servos in different ways in order to provide movement in all three dimensions. The links can be connected with servos in many ways to allow for the arm to move with pitch, yaw, and roll. Each servo is an individual joint of the arm that allows for about one hundred and eighty degrees of movement. A servo has many properties that are a measurement of such things as current position, speed of movement, acceleration of movement, torque being applied to it, and electric current flowing through it. All of the connected servos and links come together to form what is called a pod.

When a human extends his or her arm out straight and moves it to the left, the shoulder, elbow, and wrist do not all rotate at different speeds; instead, the whole arm moves uniformly. One neat feature that is built into the servos is that when they are instructed to move, the motions are automatically smoothed. For example, if the whole arm is swinging left from its base, then the different servos will all change their positions at the same angle per second with respect to the pivot point of rotation. This is done by speeding up how fast the servos move which are farther away from the pivot point of the arm and slowing down the speed of the servos that are closer to the pivot point of the robix arm. The automatic smoothing action allows for the robix arm to move like a

human arm even though how the robix arm moves varies in many ways from how a human arm moves.

The robix arm moves by having the pod, the group of servos and links, execute scripts and macros. Scripts are a collection of macros and instructions for the pod to follow. Different scripts are made up of movements for the servos that tell the motor within the servos to move the attached link a certain number of degrees in a certain direction. An example of a line in a script might be something like “*move 3 to maxpos; wait 5;*”. This tells the pod to move the third servo as far as it can go in the positive direction, which is roughly ninety degrees to the right. There is a semicolon after every command in the script. So, after the pod moves the third servo, it holds it in that position for five seconds. After that, the pod goes on to look for the next command that it should run. A macro is simply a group of movement commands with a name. For example, to use the robix arm in the game tic-tac-toe, the pod would have a tic-tac-toe script. Within that script it could have a macro for drawing an ‘X’ in each of the nine board positions and a macro for drawing an ‘O’ in each of the nine board positions. The scripts and macros can both be saved to a file so that later those same scripts and macros can be reused whenever needed. The robix arm pod can be connected to the Iris.4 mobile robot and controlled and manipulated by using an interface program designed in Java. A pseudocode version of the program can be seen [here](#).

3. ROBIX INTERFACE

The robix arm interface manages the movement of the arm pod. It is the connection between the CCP and the robix arm and also between any other module and the robix arm. Any messages that are sent to the arm client are dealt with by the interface. The program starts off by connecting to the robix arm hardware and the CCP and then loading the appropriate script that contains the macros that are needed. The scripts are saved in the memory storage of the nexus object. The nexus is the controller of the pod, scripts, and physical connection to the hardware of the robix arm. Once the nexus connects to the robix arm hardware, called an “usbor”, the program looks for a pod. After the pod is found, it looks for any servos that are attached. Lastly, after the servos are found, the nexus assigns the external script file to the pod so that the pod can run various macros. Now the robix arm is physically ready for use and must connect to the CCP.

The robix interface program sends a message to the CCP in order to connect to it. A message consists of text fields for a sender, receiver, message type, message flags, and a payload size. It also may contain a payload of information. After the CCP accepts the robix arm’s

connection, the interface program loops itself and waits until a message is sent to it. The message may be sent from the CCP, Mind Module, or even another module. The message will contain a flag that will tell the interface program which macro that it is supposed to execute. The interface then tells the pod to run the selected macro from the script that has been assigned to its library. After the macro has been completed, the program goes back into the loop and waits for another message telling it what to do next.

A change that could be made to the robix arm interface program is improving its feedback from the environment. For example, think of what is involved in a human picking up a block. First, the arm goes to the position so that the open, outstretched hand can reach the block. Next, the arm lowers and grasps the block with its hand. Finally, the arm moves the hand with the block away to a new location. Now, compare this to how the robix arm would operate. Similarly, the servos begin by moving so that the open gripper claw(the hand) is above the block. Then, the arm brings the gripper down to the block and closes the gripper on the block. Finally, the servos move the block to a different location. But how does the robix arm know how hard to grasp the block without dropping the block or breaking itself just as a human hand knows exactly how hard to squeeze its hand?

One such method for doing this would involve using a combination of two servo properties: the measure of the current flowing through the servo and the measure of the torque being applied to the servo. It would then be possible to examine the value of the current flowing through the servo and make the moving gripper stop from grasping any harder when the value was higher than a certain threshold. The measure of the current would help tell how hard the gripper arm is squeezing because it would let us calculate the amount of resistance offered by the gripper arm. The resistance provided by the gripper arm could be measured by dividing the power that controls the robix arm, measured in volts, by the amount of current flowing through each servo, measured in amps. That way, when the gripper arm had a firm hold on the block but still squeezed further, the amount of current needed to flow through that servo would increase and pass the threshold value of current for the servo, prompting the termination of the grasping action.

4. FTF SYSTEMS

There are many other real-world applications of robotic feedback systems that are similar to the one that was just discussed. An example of this is the Fault-Tolerant Feedback System (FTF) created in the Lyndon B. Johnson Space Center in Houston, Texas. The FTF system is designed to catch any errors in the movement of any of

NASA's robotic arms. It is important for NASA to maintain control of its robotic arms since it works closely with humans and delicate materials. The FTF system is put in between the motion controller and servo controller to detect errors in the motion so that it can block the sent message to the servos for them to carry out a specific motion ("Fault-Tolerant...").

5. CONCLUSIONS

The FTF system provides the same functionality as the proposed change to the robix robotic arm and to a human's arm. All three are similar in the fact that via their own respective methods, each system manages its

arm by getting feedback from the environment and, in a sense, error checking its movements. The FTF system does its checking by comparing its movements to a mathematical model to detect transducer failures. The robix arm does its feedback checking by measuring current through its servos to see if something is blocking its movement. Both systems attempt to reproduce the same types of error-proof movements of a human arm, just not via the same methods and checks.

6. REFERENCES

- [1] Nasatech. *Fault-Tolerant Feedback System for Robot Control*. Nasatech. 6 May 2005.
<http://nasatech.com/Briefs/Oct99/MS22591.html>