AN APPLICATION OF MECHATRONICS IN MANUFACTURING WITH OBJECT-ORIENTED PROGRAMMING IN A WINDOWS ENVIRONMENT

M. L. J. Jürgens[1] & D.C. Page[2]

Abstract

This project is part of a program to establish mechatronics knowledge and skills in the Department of Industrial Engineering at the University of Stellenbosch. A low-cost, but accurate mechatronic application was developed by automating a pipe-bending machine. Accelerating software development through object-oriented programming was also investigated. The object-oriented software was developed with a structure that increases the independence between the application object and the data acquisition system. A teach-pendant and large, multicolour displays with interactive buttons were developed to ensure a user-friendly machine. The positioning of the headstock was controlled by a pulsing control algorithm, which achieved an accuracy of ±0.15 degrees and a repeatability of ±0.24 degrees. The design of the machine and software and experimental results are discussed in this paper.

Opsomming

Hierdie projek vorm deel van 'n program om kennis en vaardighede in die aanwending van megatronika in die Departement Bedryfsingenieurswese van die Universiteit van Stellenbosch te vestig. 'n Lae koste, maar akkurate megatronika toepassing is ontwikkel deur 'n pypbuigmasjien te outomatiseer. Versnelling van sagteware-ontwikkeling deur die toepassing van objek-georiënteerde programmering is ook ondersoek. Die objek-georiënteerde sagteware is ontwikkel met 'n struktuur wat onafhanklikheid tussen die toepassings-objek en die dataver-samelingstelsel bevorder. 'n Handkontrole eenheid en groot veelkleurige rekenaarvertoonskerms met interaktiewe drukknoppe om 'n gebruikersvriendelike masjien te vereker is ontwikkel. Die posisionering van die kopstuk word beheer deur 'n puls-algoritme wat 'n akkuraatheid van ±0.15 grade en 'n herhaalbaarheid van ±0.24 grade lewer. Die ontwerp van die masjien en sagteware en eksperimentele resultate word in hierdie artikel bespreek.

[1] Department of Industrial Engineering, University of Stellenbosch
[2] Department of Industrial Engineering, University of Stellenbosch
Telephone: +27 +21 8084236
Fax: +27 +21 8084245
E-mail: dcp@ing.sun.ac.za

1. INTRODUCTION

Judicious employment of automation enables productive manufacturing of high quality products. Mechatronics is an interdisciplinary design philosophy applied to develop systems with a mechanical component, an electronic component and a computer as controller. The premise is to assign functions to the most suitable technology. The core of a mechatronic system or artefact is the intelligent control and adaptability provided by the computer software. Changing the program can change the characteristics of the system. The system can be extended or reduced by simply adding or removing sensors and actuators and changing the program. Products of consistent quality are manufactured due to the repeatability under computer control. Software also gives greater freedom to design uncomplicated and user-friendly human-system interfaces. Mechatronic applications have, however, the disadvantage that software development is very time consuming. A further problem with automation, from the viewpoint of small to medium size manufacturing companies, is the cost involved. A project at the University of Stellenbosch focussed on these areas by developing low-cost automation, according to mechatronic principles for a small manufacturing company [1]. Accelerated software development was also investigated with the use of object-oriented programming. Furthermore, the project focussed on developing user-friendly interfaces for semi-skilled labour.

The industrial partner was a sheet metal fabrication firm with a job shop set-up. In addition to several commercial CNC and manual machines the company employs various hydraulic machines developed in-house. Three manufacturing processes that utilise simple hydraulically powered machines with a single cylinder as actuator were investigated for the project. The first process is the bending of aluminium pipe sections for the manufacturing of baby carrier frames and similar products. The firm uses a simple, manually operated pipe-bending machine, which has to be set up for each different bend in the frame. This requires that a specific bend be applied to a batch of pipes before the machine is set for the next bend. Automation will improve the process in that all the bends can be completed for each pipe before moving to the next pipe, thus reducing the excessive material handling costs of the process. The second process uses a simple, manually operated press break to bend curved sheet-metal components such as radar antenna casings. Automating this process will eliminate the time consuming initial set-up (adjustment of mechanical stops) for each bend. Metal hinges are manufactured in the third process and here automation will improve the accuracy of the process.

All three applications can be automated by computerised position control of a linear hydraulic cylinder. The elements of the control was designed as portable units since equipment had to be shared among the processes to make automation viable from a cost perspective. A pipe-bending machine, similar to the machine at Calculus Products, was designed and manufactured at the University of Stellenbosch and was used for development in the laboratory.

2. OPERATION OF THE PIPE-BENDING MACHINE

The machine, shown in Figures 1 and 2, consists of a vertical rotational headstock, a guide and mandrel (for guiding the pipe) and a double acting linear hydraulic cylinder. These parts are mounted on a basic framework. The mandrel, which has a slightly smaller diameter than the inner diameter of the pipe, prevents the pipe from collapsing. A manual clamp on the rotational

headstock secures the pipe during the bending action. The cylinder is connected to a continuous chain, which runs around a sprocket on the headstock and a free running sprocket on the other end.

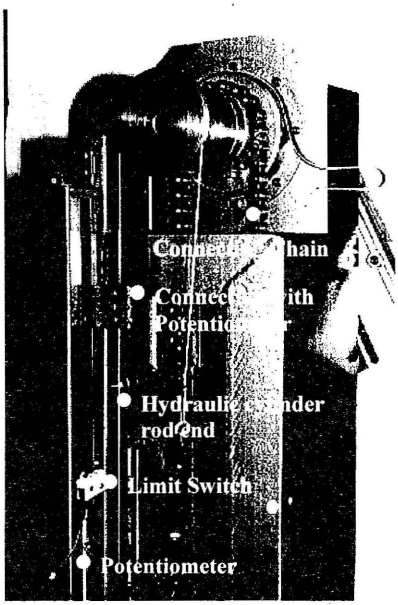

Figure 1 – Side view: Actuator and sensors

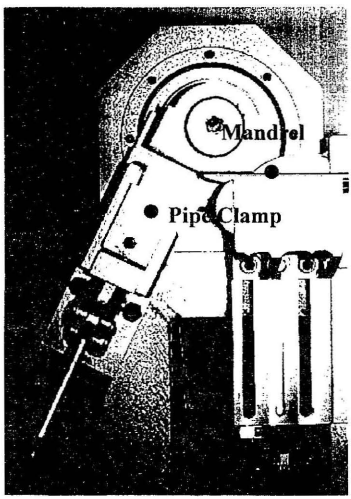

Figure 2 – Side view: Headstock

The manual bending machine is operated as follows:

1. The machine is set for the desired bend by adjusting mechanical stops.
2. An aluminium pipe is inserted over the mandrel and clamped to the rotating headstock.
3. The pipe is bent through the desired angle by activating the cylinder that rotates the headstock.
4. The pipe is removed from the machine and the next pipe is inserted.

Steps two to four are repeated for the whole batch before the stops are set for the next bend and the whole batch is reloaded. Automating the headstock, so that a series of bends can be programmed, will drastically reduce material handling.

The automated process runs as follows:

1. An aluminium pipe is inserted over the mandrel and clamped to the rotating headstock.
2. The pipe is bent through the desired angle by activating the cylinder that rotates the headstock.

3. The pipe is unclamped, adjusted lengthwise for the next bend and rotated through the correct angle (assisted with mechanical stops) if a bend is required in a different plane.

Steps two and three are repeated until the required bends are completed, after which the pipe is replaced with the next pipe. Every pipe is thus inserted only once, drastically reducing material handling.

The following elements were used in the automation of the bending machine. A hydraulic power pack, a simple three position, solenoid/solenoid operated, spring centred hydraulic valve, a flow control valve, a linear potentiometer as a displacement transducer and a PC with a low-cost analog-digital/digital-analog (AD/DA) card for data acquisition. A diagram of the system is shown in Figure 3.

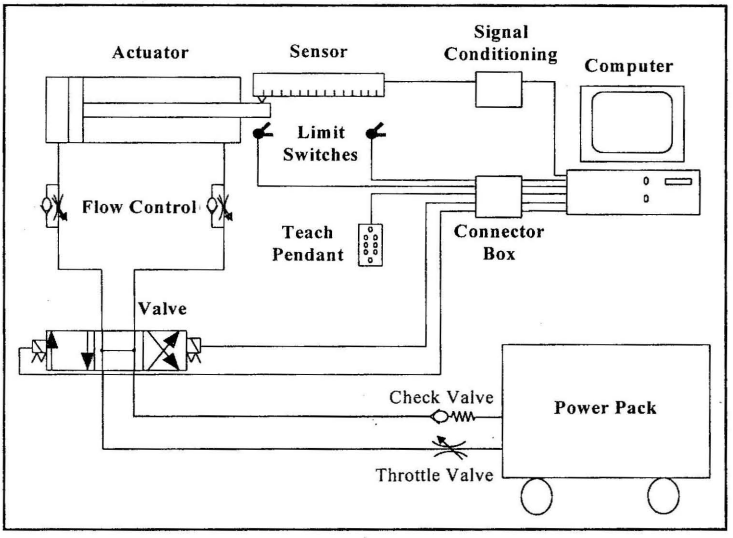

Figure 3 - Diagram of control system

3. DEVELOPMENT OF OBJECT-ORIENTED SOFTWARE

Software for the application was developed in C++ in a Windows environment according to object-oriented principles. A traditional approach to software development concentrates on *actions* that are specific to the environment of the application. In practice this means that software cannot be re-used and that development of new applications effectively starts from scratch. The object-oriented approach focuses on developing objects that describe an entity as a whole. All the functionality and characteristics of the entity are developed in the object. An object can only communicate via set messages with a fixed format, protecting the object against corruption. Furthermore, the code of the object is encapsulated and a user only needs to understand the set protocol and messages to use the object. The object is thus independent of the environment and it is possible to re-use the code in different applications.

An object-oriented approach can be applied to a mechatronic system as a whole by dividing the system into three main objects, namely a hardware object, a virtual hardware object and the application object. The objects communicate in a serial fashion with the virtual hardware object between the hardware object and the application (refer to Figure 4). The approach is explained with an example where the mechatronic software uses an AD/DA-card for interfacing with the manufacturing process.

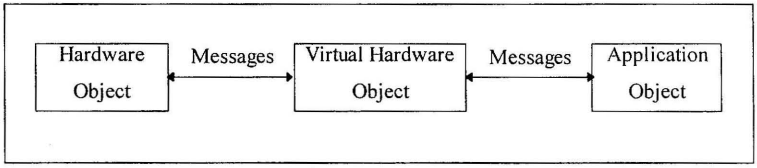

Figure 4 - Schematic presentation of software

The AD/DA-card (hardware) object is the object that directly controls the physical AD/DA-card. This object is AD/DA-card specific, which implies that a different object must be developed for each different type of AD/DA-card. Once an object is developed for a specific AD/DA-card it can be used with any application that uses the virtual AD/DA-card object. The AD/DA-card object uses all the functionality of the AD/DA-card through a range of methods. Each of the methods can be activated through a specific message that is defined in the interface of the virtual AD/DA-card object. The generation of a response to a message is, however, completely hidden by the AD/DA-card object. The virtual AD/DA-card specifies *what* functionalities an AD/DA-card must deliver and the AD/DA-card object provides this functionality for the specific type of card.

The purpose of the virtual AD/DA-card object is to ensure compatibility and interchangeability between different application objects and AD/DA-cards. The virtual AD/DA-card object describes a general AD/DA-card. It considers only the functionality that should be available in any AD/DA-card. It describes the input needed to activate the functionality of a general AD/DA-card and the responses that are expected from the card. The description is accessible to the application through the message protocol that conveys data and commands to and from the application.

The concept of a general AD/DA-card implies that differences between real AD/DA-cards must be equalised. AD/DA-cards, however, differ vastly in complexity. Some cards provide only basic functionality, while others provide more sophisticated functionality. It will be possible to write objects for all cards that are compatible with a virtual AD/DA-card object but much of the complex functionality will have to be implemented with software in simple cards. This implies that the simple cards will perform with less speed than complex cards.

The application employs the virtual AD/DA-card object to manipulate the AD/DA-card. The type and specific attributes of the AD/DA-card are encapsulated by the virtual AD/DA-card object and are thus hidden from the application. The methods provided by the virtual AD/DA-card object are the total functionality available to the application. This functionality is used by the application to acquire data and to control the mechatronic system. The application is responsible for transforming and manipulating the information (provided by the data acquisition) and representing it in an appropriate format to the human user. Other applications, developed compatible to the virtual AD/DA-card object, can employ the data acquisition system without developing new software.

Total mechatronic systems can thus be developed more rapidly because no data acquisition systems need to be developed. New developers also do not need extensive knowledge of AD/DA-card and data acquisition systems. They only have to master the use of the basic and higher methods provided by the virtual AD/DA-card object.

4. MAN/MACHINE INTERFACE

The man/machine interface was designed with the emphasis on ease of use as the industrial partner employs semi-skilled labour. The partner also required that the machine be programmed with the standard G-code language as a number of CNC machines were already in operation at the firm.

The machine can be controlled by using a keyboard, a mouse or a teach pendant. The teach pendant was specifically developed for use in the factory as the pendant is easier to operate. The teach pendant can be used in the teach-mode to create a G-code program by "teaching" the machine. A program to control a successive sequence of bends are captured under teach pendant control by completing all the bends on one product. These motions are converted to a G-code program that can be edited by a G-code editor developed as part of the software. New programs can also be developed from scratch in the G-code editor or the programs can be developed off-line. By using a simulation option the control software can also test a new program.

The user-friendliness of the software is achieved by using screens with large multicolour displays. The displays change colour to show the different modes of the machine and on-screen buttons are used instead of type-in commands. A typical screen is shown in Figure 5. The teach pendant or the mouse can be used to activate buttons on the screen when running the program.

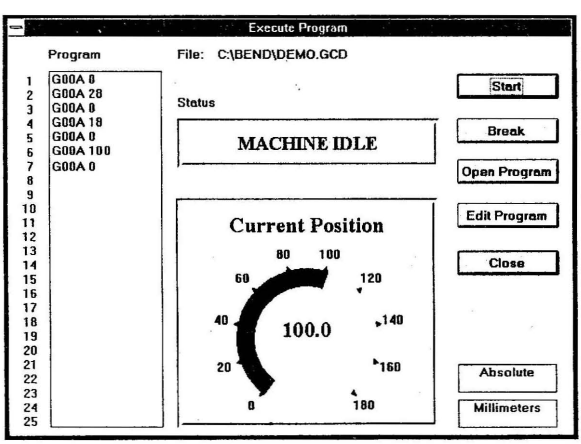

Figure 5 - Typical screen of control software

5. CONTROL ALGORITHM

A modified ON/OFF control scheme[2] is used and is explained in Figure 6. This method of control has four parameters that must be set: a pulseband, deadband, pulse period and the pulse width. A

constant signal is applied when the error signal (difference between target position or setpoint and actual position) is bigger than the pulseband and a pulsing signal is generated when the error falls between the pulseband and deadband. The pulse period and pulse width determine the pulsing signal. In the specific application a pulse width of half the pulse period was used. The advantages of modified on-off control are that this type of control is independent of the computing speed of the controller, as the signal is off when calculations are required. Secondly, modified ON/OFF control also enables accurate control with simple, low-cost equipment.

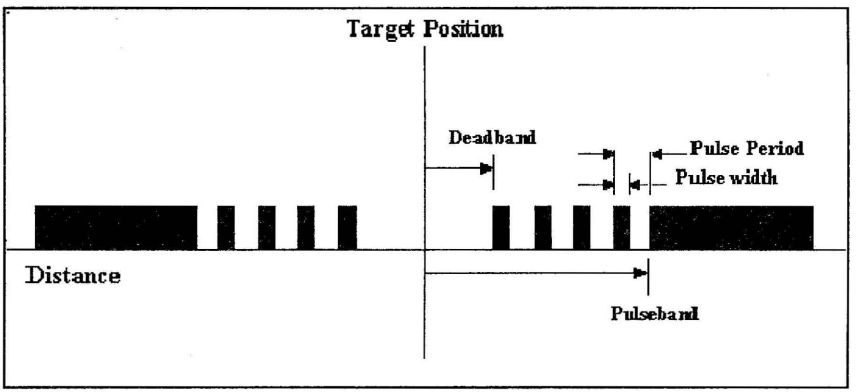

Figure 6 - Modified on-off control

6. ACCURACY AND COST

Tests were executed to determine the accuracy of the machine with different deadband, pulseband and pulse width values. This was accomplished by executing a number of "move" commands and measuring the distance between the target and the actual position where the headstock stopped. Note that displacement was measured by the potentiometer (used as sensor) and not independently on the headstock (the real rotation).

The results show that, as can be expected, a narrow deadband improves accuracy and repeatability. The machine keeps on pulsing until the position is within the deadband. There are, however, limitations on how narrow the deadband can be. If the band is too narrow, the machine will pulse an excessive number of times, slowing down the process and reducing the lifetime of the valve. In this specific application another problem is that the design of the machine must be such that pipes are bent in one direction only. On overshoot, an attempt to bend the metal in the other direction by returning to the deadband only absorbs elastic spring-back. A first order type of response without overshoot and with zero steady-state error is required from this system. A too narrow deadband will cause overshoot (if the second last position is near the deadband, the last pulse can move the headstock over the narrow deadband). The average error is also influenced by the width of the pulseband. The pulseband should be such that the average error is minimised (close to zero), but that excessive pulsing is not required.

From experiments the best accuracy was achieved with a deadband of 0.15 degrees, a pulseband of three degrees and a pulse width of five milliseconds. The best absolute accuracy of the system was

therefore ± 0.15 degrees (the width of the deadband). The repeatability by statistical definition of ±3 standard deviations was ±0.24 degrees. This accuracy amply satisfied product requirements.

A similar application was developed at the Rensselaer Polytechnic Institute in Troy, N.Y [2] where a fixturing device was positioned by two hydraulic cylinders, controlled by ON/OFF control valves utilising a modified ON/OFF control algorithm. An important difference between the two applications is that the Rensselaer application measured position with a highly accurate linear incremental encoder with no electrical noise, in comparison to the inexpensive linear potentiometer with the inherently high noise level used locally. The Rensselaer application used a pulse period of 24 ms and a pulse width of 12 ms. With these settings an accuracy of between ±0.19 mm and ±0.05 mm was achieved. This compares reasonably well with the ±0.15 to ±0.24 degree accuracy of the bending machine (machine designed such that one millimetre converts to one degree).

In a preceding project at Stellenbosch [3] on the same bending machine an accuracy of ±0,02 degrees was achieved. With this project an accurate linear incremental encoder and a high performance proportional control valve was used. The problem with this application was cost. In 1995 terms the previous application cost R20 000 (computer excluded), in comparison to the R4 000 cost of new system.

7. CONCLUSIONS

The system satisfies the objectives of low cost and acceptable accuracy.

From the software perspective valuable lessons were learnt. For this type of application the researchers experienced that Object-Oriented programming improves programmer productivity when objects are re-used, but it also increases software complexity when applied to simple applications. With traditional approaches the programmer designs software for the current environment (software and hardware). The developer of objects on the other hand must develop the object for a wide range of environments which make the design more complex. Currently object-oriented programming is natural with objects (or device drivers) supplied with the hardware. In the previous project Pascal was used in a DOS environment. In a current follow-on project a high-level development system, LabVIEW, is used. Generally, it seems as if the total programming time remains roughly the same with the different programming technologies. The quality of the program, however, improves significantly. Programs are much more stable with better user interfaces with newer technologies.

With the current project the technology developed is being transferred to industry.

8. REFERENCES

[1] Jürgens M L J, 1998, *An application of mechatronics in manufacturing with object-oriented programming in a windows environment*, Unpublished Master's Thesis in Industrial Engineering, University of Stellenbosch, 142pp.

[2] Kurz K, Craig, K, Wolf B, Stoli F, 1994, Developing a Flexible Automated Fixturing Device, *Mechanical Engineering*, 116(7), pp.59-63.

[3] Coetzee J H, 1994, *Applications in Computerised Automation using Linear Displacement Transducers,* Unpublished Master's Thesis in Industrial Engineering, University of Stellenbosch, 223pp.