Proceedings of the NACCQ 2000
Wellington NZ www.naccq.ac.nz

Use of Embedded Applications in Automatic Loop Tuning

Dr Andy Doonan[1]
Prof Chris Cox[2]

School of Information Technology and
Electrotechnology
Otago Polytechnic
Dunedin, New Zealand
andy@tekotago.ac.nz[1]
University of Sunderland, UK[2]

ABSTRACT

The tuning algorithm is selected based on robustness and ease of application and is described along with the requirements of the software and hardware.

1. INTRODUCTION

This paper describes the development of an embedded system which can be used in the on-line tuning of industrial PID controllers. Three term (PID) control is still widely used in industry because of its simplicity, versatility and since it allows flexible performance over a wide range of operational conditions. Thanks to the fairly recent arrival of the new families of microprocessor based systems these controllers are currently going through an interesting phase of development (Portillo *et al.*, 1998). Improvements in microelectronics together with greatly increased computational power have allowed some manufacturers to offer equipment with additional features such as pretune, gain scheduling and auto-tune. There are many commercial products that offer these plus many other additional features. However, the number of controllers this represents is minute compared to the vast numbers already operating in industry that do not possess these advanced features.

A few years ago a project was initiated to help redress the imbalance by developing some software, initially under DOS, that could provide some of these modern attributes to an otherwise standard controller. The only requirement was that the controller must have a communications port (e.g. RS232, RS422). The result was ***MasterTune*** CAD software environment. The current version of ***MasterTune*** operates under Windows 95. While the basic kernel of the software remains the same, advantage has been taken of the multi-tasking nature of the operating system.

Despite the many advantages of a fully graphical tuning application, there may be times when a full blown pc solution is not desirable. This paper shows how the tuning algotihm can be 'condensed' into a low specification hardware solution allowing a hand held tuning tool to be developed. Such a tool requires the application of an embedded system and its associated microcontroller. Due to its widespread adoption and specifications, the 8051 microcontroller has been selected ensuring many readily available development tools and systems.

This paper is structured as follows: Section 2 presents the basic algorithm which has been selected due to its robustness and ease of application. Section 3 describes the software features and requirements which were implemented in the MasterTune application and are required of the embedded system. Section 4 describes how the tuning algorithm must be 're-cast' so that it is suitable for an embedded system. Test results are presented in Section 5 before concluding remarks in Section 6.

NACCQ
FOSTERING COMPUTING EDUCATION IN NEW ZEALAND

2. RELAY AUTO TUNING

Astrom and Hagglund (1984) described an automatic tuning (auto tuning) method which utilised a relay controller plus an integrator to force the system to oscillate with a constant amplitude, fixed frequency oscillation known as a limit cycle. An important feature of the technique is that changing the magnitude of the relay characteristic can alter the amplitude of the oscillation. Also, the addition of the integrator has the benefit of forcing the limit cycle to be sustained about the set point value and helps to ensure 'bumpless' transfer between tuning and control modes. Much has been published on the relay auto tuning technique so it is sufficient here to stress that the design condition requires that K_c and T_i be chosen such that

$$T_i = \frac{P_U \tan\phi_m}{2\pi} \quad \text{and} \quad K_c = \frac{2V_m P_U \sin\phi_m}{\pi^2 A}$$

Equation 1

where

P_u is the period of the limit cycle oscillation
A is the peak value of the limit cycle appearing at the input to the non-linearity, and,
$_m$ is the phase margin.

A block diagram of the system during tuning is presented as Figure 1.

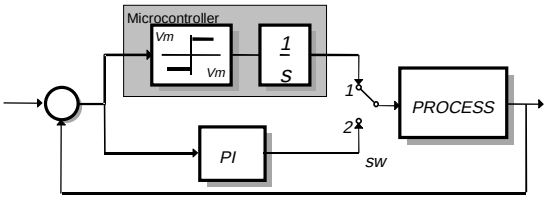

Figure 1 A block diagram of the system during the tuning phase

3. SOFTWARE REQUIREMENTS

CAD software called ***MasterTune*** has been developed (Doonan, 1997) to help in the automatic tuning of PI controllers using the technique described above. Any software implementation of this strategy requires number of distinct stages and each stage a number of parameters which can be adjusted by the user.

3.1 Connection to the Process

A physical connection to the process the be tuned can be considered to be the most direct (and perhaps in terms of software) the simplest method. However, from the point of view of both the user of the software and the process engineer this approach is less than satisfactory. A better solution is to use the existing plant connections and to exploit the communications ability of modern industrial process controller regulating the loop. This means that instead of making direct electrical connections to the plant it is required to simply plug a cable into the serial port of the controller (Figure 2).

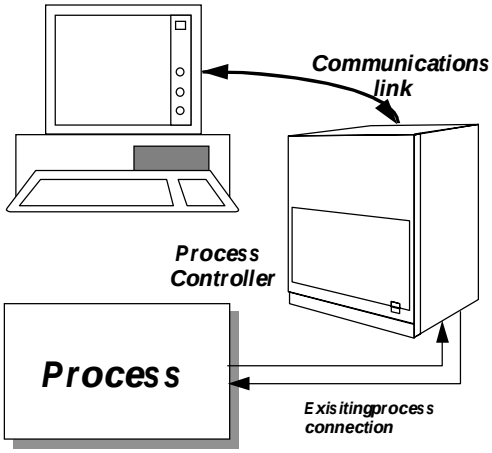

Figure 2 Hardware connection

3.2 Test Parameters

The values of various test parameters used during the tuning phase are automatically set to default values by interrogating the controller and using existing settings. These values are used by the software to provide 'sensible' tuning parameters however, if the user so chooses these values can be modified to suit individual requirements. The parameters that can be modified include

- *The percentage overshoot:* the amount of overshoot the closed loop system will exhibit when compensated by the tuned PI controller. The percentage overshoot is used to determine the phase margin required by the design equations (see Equation 1).

- *The relay characteristics:* the amplitude (to control the size of the limit cycle oscillations during tuning) and the hysteresis (used to prevent false switching caused by noisy signals).
- *Constraints:* control over the allowable level variations of both the process and manipulated variables.

3.3 Process Analysis

The process analysis phase is the preliminary step of the tuning procedure. Here an open loop step test is conducted and a first order plus dead time model automatically calculated using a characteristic area approach (Nishikawa *et al.*, 1984) If, as recommended, the fast tune procedure is by passed in favour of the relay feedback autotuning, then the relay characteristics must be set. The process analysis phase calculates a value for the relay height that will produce a limit cycle with amplitude approximately equal to that set in the 'Set test parameters' phase above. Additionally, the software will analyse the level of noise imposed on the process variable and use this value to suggest a level of hysteresis.

This stage relieves the process operator of the rather complex task of specifying the relay characteristic. The operator only needs to specify the tolerable process variable swing. Further, it should be noted that this stage might only be required when tuning a process for the first time. In subsequent sessions, the relay parameters can be entered directly based on previously recorded values. If the time delay is not appreciable in relation to the time constant the software recommends that a PI controller be used. This fast tune facility uses the transfer function obtained in the process analysis phase to calculate the controller settings using some empirical formula, typically those recommended by Cox *et al*. (1987). This feature can be used to get a control loop working satisfactorily in a very short period of time.

3.4 AutoTuning

When the autotune in invoked, the relay and integrator force the process variable to oscillate with the specified amplitude. After a stable limit cycle is exhibited by the process, a PI controller is calculated using the technique given in Section 2 which will result with the required level of overshoot being observed during compensated evaluation tests.

3.5 Evaluation

Once the appropriate controller has been selected and tuned, the evaluation stage can be implemented. Here the controller is put into automatic mode and, if desired, a step change can be induced in the set-point (see Figure 7). This, as well as each other stage in the tuning cycle, can be logged and later 'uploaded' to a pc to provide a full record of the work completed.

4. EMBEDDED APPLICATIONS

The use of embedded systems is now widespread in the electronics industry. The replacement of numerous discrete components with a microcontroller enables simple post-deployment modifications being just one of the advantages of such an approach. Size and power consumption are also advantages. But what is an embedded system? An embedded system is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function. In some cases, embedded systems are part of a larger system or product, as is the case of an anti-lock braking system in a car.

The heart of the embedded system, the microcontroller can be thought of as being a computer on a single chip, that is a CPU combined with RAM, firmware (usually EPROM and/or EEPROM) and sometimes an Analog to Digital Convertor (ADC) and a Digital to Analog Convertor (DAC). It is this self-containment which enables the microcontroller to be used in the embedded system. There are many families of microcontroller available, major manufacturers such as Intel, Philips and Hitachi each produce their own proprietary internal architecture generally significantly different to the other manufacturers. Also, the features offered by each manufacture differ, some microcontrollers offer on-board PWM generators, ADC, DAC, RS232, etc.

A disadvantage of the microcontroller is the general scarcity of resources, there will be a small amount of memory and generally slow clock speeds. This restriction has implication for the software which must be executed in that code should be as compact as possible and very limited user input facilities are available (no GUI!).

4.1 Choice of Microcontroller

For this project, the 8051 series of microcontroller was chosen and in particular the devices manufactured by ATMEL This device is utilises flash memory to hold the

program code and therefore removes the need to erase the old program during the development cycle (as required by older EPROM based devices). This facility greatly reduces the development time. The 8051 design has been around since the 1980's and as such has plenty of support products readily available. Due to the complexity of the program code required for this project a C compiler is required and is readily obtainable for this device.

4.2 Hardware Design

With the design goal of a developing a hand held device, hardware design is of importance especially that facilitating user IO. As well as the microcontroller, the following components were required.

2 line LCD screen	used to communicate messages to the user
keypad	used to allow the user to modify system parameters
RS232 line driver	required to interface to serial communication ports
EEPROM	used for the non-volatile storage of parameters, results, etc.

Table 1 A list of hardware components

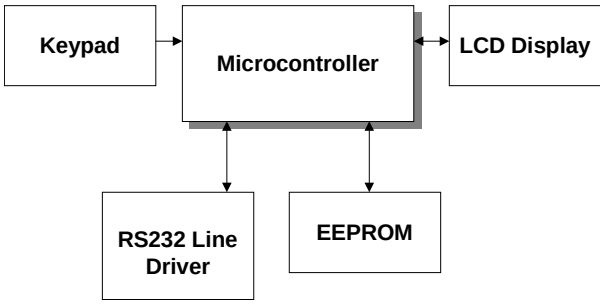

Figure 3 Hardware components required

4.3 Software Design

The functionality of the autotuner described above can be decomposed into a number of numerical algorithms.

- Noise level detector
- Step response evaluator
- Relay/integrator simulation
- PI parameter calculation

One approach to the solution of this type of problem where a discrete number of steps should be followed in a given sequence is to use a 'finite state machine'

Implementation involves the management of the flow of the numerical algorithms. This flow is managed by activating or deactivating the algorithms depending upon a number of conditions. The program implements a state flow diagram (see Figure 4) whose states are formally described by a global variable S. The states and transistions are briefly presented below.

During S = 0, the noise level of the error signal is analysed. This is used in determining the amount of hysteresis necessary for the relay to exhibit. Condition C0 is true when 20 samples have been analysed.

In state S = 1, the open loop step is applied. Condition C1 is true when steady state has been determined.

In state S = 2, relay tuning is performed. Condition C2 is true when the amplitude of successive peaks of the resulting limit cycle are within a certain percentage of each other.

In state S = 3, the PI parameters are calculated. Condition C3 is true once these parameters have been determined.

In state S = 4, the PI algorithm is active. Condition C4 becomes true when the 're-tune button' is selected.

5. TESTING

To cater for the absence of exisitng plant data recording equipment, software has also been developed which will down load recorded data from the hand held unit and display it in a graphical environment. These graphs can then be used for comparison and archiving purposes. Figure 5 shows the results of the process analysis phase presented to the process operator. The resulting model output is overlaid onto the process reaction curve allowing an informed judgement to be made about the quality of the model obtained.

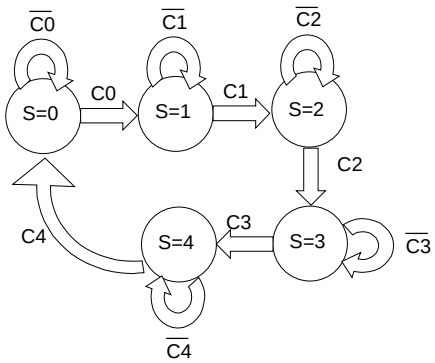

Figure 4 State transistion diagram of PI autotuner

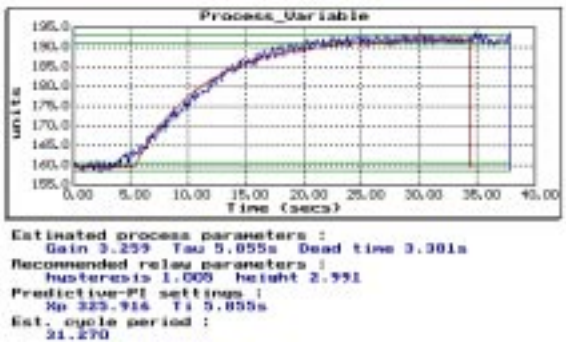

Figure 5 Typical process analysis result from a laboratory scale flow rig

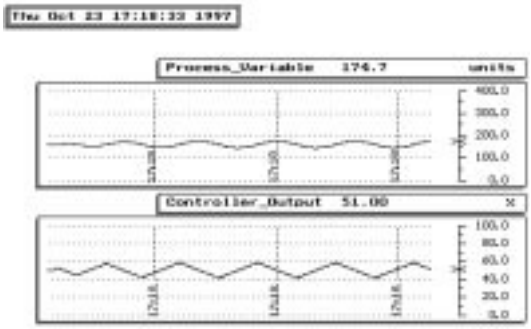

Figure 6 CAD software screen dump showing Autotuning in process

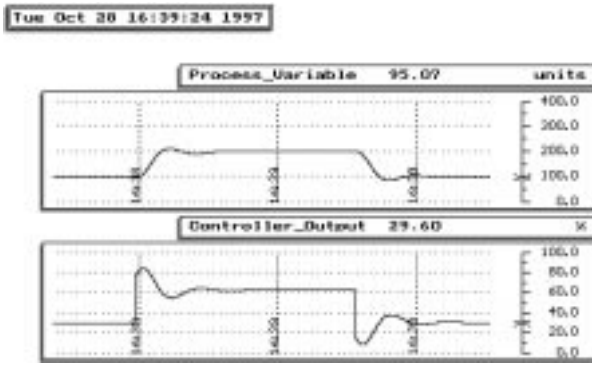

Figure 7 CAD software screen dump showing evaluation of PI controller

Figure 6 and Figure 7 shows data recorded during the tuning and evaluation phases.

6. CONCLUSION

The PI(D) controller is still the dominant system used in current industrial practice. Much has been published about the design of PI(D) regulators, however, the majority of the strategies rely on accurate process information not normally available to the commissioning engineer installing systems on 'real plant'. For this reason, alternative methods have been sought which still provide good controller settings based upon imprecise process knowledge. Success with the Astrom-Hagglund tuning method in the MasterTune software application has lead the authors to investigate other potential areas of exploit. Athough the features offered by a fully graphical interface can be desirable to a user, portability is also advantageous and can be obtained from an embedded system. This paper has also shown that with some modification, the tuning kernel can also be easily implemented into such a system resulting in an handheld unit being developed.

7. REFERENCES

Astrom K.J. And Hagglund T., (1984), Automatica, 20, 645-651. 'Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins'.

Cox C.S., Daniel P.R. And Lowdon A., (1997), Control Eng. Practice, Vol 5, No 10, pp 1463-1472, 'Quicktune: A Reliable Automatic Strategy for Determining PI And PPI Controller Parameters Using A Fopdt Model'.

Doonan A.F., (1997), Phd Thesis : The Development, Evaluation and Implementation Of True Digital Control, University Of Sunderland.

Nishikawa Y., Sannomiya N., Ohat T. And Tanaka H., (1984), Automatica 20 1984, 'A Method of Autotuning PID Parameters'.

Portillo J., Marcos M., Orive D., Lopez F.And Perez F., (1998), Pid_Atc: A Real-Time Tool for PID Control and Auto-Tuning, Pre-Prints 5th Ifac Workshop On Algorithms And Architectures For Real-Time Control, AARTC'98.