Declaration of originality

I declare that this report entitled — “Design and implementation of double-integral control experimentation board” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature: _________________________
Name: ____________________________
Date: _____________________________
Dedication

To my parents, for their overwhelming care and love …
Acknowledgement

This thesis has benefited greatly from the support of many people, some of whom I would sincerely like to thank here. To begin with, I am deeply grateful to Professor ABDARAHMAN KARRAR for offering me such an interesting topic of investigation.

Finally, but first in my heart, my parents are due my deep gratitude for their continued moral and financial support throughout my studies, the former being of much greater importance. The broad education that I was able to enjoy while growing up has proven invaluable.
Abstract

We will review in this thesis a classical experiment that exemplifies control theory fundamentals and methods of control. This experiment called **double integral system control experimentation board**. An experimentation board was designed and implemented demonstrating all aspects of continuous and discrete approaches that stabilize the control while operating in servo mode. The model contains motor, metal arm, webcam and computer.

MATLAB has been chosen as a controller of the experimentation board, connected to it via a parallel cable & a parallel board. Using MATLAB has provided all the needed functions in order to execute the experiment. Making use of the Background subtraction technique with MATLAB program we could trace the arm by a webcam and laser source.
المستخلص:

سيعرض في هذه الأطروحة تجربة كلاسيكية تمثل اساسيات وطرق التحكم. وهذه التجربة لتصميم لوحة اختبار معملي للنظام تتحكم ذو مكامل مزدوج. تم تصميم لوحة اختيار معملي لتوضيح كل جوانب الطرق المستمرة والمتقطعة التي تجعل التحكم في حالة استقرار بينما تعمل في وضعه المؤارد. النموذج مكون من موتور، ذراع معدني، كاميرا و جهاز حاسب.

وقد تم اختيار الماتلاب كوحدة تحكم في لوحة الاختبار ، ويتصل عبر كبل متوازي منفذ متوازي. وقد وفر برنامج الماتلاب جميع الوظائف المطلوبة من أجل تنفيذ التجربة. بالاستفادة من تقنية الطرح الخلفية مع برنامج الماتلاب امكننا تتبع الذراع بواسطة كاميرا ويب ومصدر الليزر.
# Table of Contents

1 **Introduction** ........................................................................................................................................... 1  
   1.1 Overview: ............................................................................................................................................... 1  
   1.2 Problem Statement ................................................................................................................................. 1  
   1.3 Motivations ............................................................................................................................................ 1  
   1.4 Objective ............................................................................................................................................... 2  
   1.5 Thesis Layout: ...................................................................................................................................... 2  
2 **Literature review** .................................................................................................................................. 3  
   2.1 Computer Control .................................................................................................................................. 3  
   2.2 Computer-Control Theory ..................................................................................................................... 4  
      2.2.1 Time Dependence ............................................................................................................................ 5  
      2.2.2 A Naive Approach to Computer-Controlled Systems .................................................................... 7  
      2.2.3 Deadbeat Control ............................................................................................................................ 11  
      2.2.4 Aliasing .......................................................................................................................................... 13  
   2.3 Discrete-Time Systems .......................................................................................................................... 16  
      2.3.1 Sampling Continuous-Time Signals ................................................................................................. 16  
      2.3.2 Sampling a Continuous-Time State-Space System ......................................................................... 17  
      2.3.3 Zero-Order-Hold Sampling of a System .......................................................................................... 18  
      2.3.4 How to compute $\phi$ and $\Gamma$ .................................................................................................... 21  
   2.4 Integral control system .......................................................................................................................... 22  
   2.5 PID Control .......................................................................................................................................... 23  
      2.5.1 Feedback Control ............................................................................................................................ 24  
      2.5.2 The Three Actions of PID Control .................................................................................................. 25  
   2.6 MATLAB® .......................................................................................................................................... 28  
      2.6.1 Simulink® ...................................................................................................................................... 28
# Table of figures

2.1 Schematic diagram of a computer-controlled system .............................................. Error! Bookmark not defined.
2.2 (a) Block diagram of a digital filter. (b) Step responses (dots) of a digital computer ..... Error! Bookmark not defined.
2.3 A system for controlling the position of the arm of a disk drive .............................................. 7
2.4 Simulation of the disk arm servo with analog (dashed) and computer control (solid) ........... 8
2.5 Simulation of the disk arm servo with computer control having sampling rates .................. 10
2.6 Simulation of the disk arm servo with deadbeat control (solid) ........................................... 12
2.7 Simulation of the disk arm servo with computer control ..................................................... 14
2.8 Simulation of the disk arm servo with analog and computer control ................................. 15
2.9 Block diagram of a continuous-time system connected to A-D and D-A converters .......... 19
2.10 Integral system ................................................................................................................. 23
2.11 Typical component of feedback control loop ................................................................. 25
2.12 Schematic block diagram of feedback control loop ......................................................... 25
2.13 PI controller in automatic reset configuration ................................................................. 27
2.14 The Simulink® Library Browser ....................................................................................... 30
2.15 Integrating the second derivative “ydd” twice to obtain the first derivative “yd” and output “y” ......................................................................................................................... 31
2.16 Dialog box for specifying input step parameter values .................................................... 32
2.17 Simulink® diagram for step response of a second-order system ....................................... 33
2.18 Alternate Simulink® diagram for a second-order system step response ......................... 34
2.19 Dialog box for configuring simulation .............................................................................. 35
2.20 Screen capture of scope output .......................................................................................... 36
2.21 Screen capture of edited scope output ................................................................................ 36
2.22 Simulink® plot of unit step response of a second-order system ........................................ 37
2.23 (a) Background image. (b) A certain frame of the video sequence. (c) Detected foreground objects ........................................................................................................................................ 40
2.24 Shadow detection using different thresholds ...................................................................... 40
2.25 (a) Final shadow detection (shadow pixels are represented by light gray). (b) Foreground objects after shadow removal. (c) Elimination of gaps and isolated pixels through morphological operators ........................................................................................................................................ 43
3.1: System-block diagram ........................................................................................................ 45
3.2: DC motor ........................................................................................................ 47
3.3: DAC08 ............................................................................................................. 47
3.4 Buffer74-244 ................................................................................................. 48
3.5: parallel port .................................................................................................. 49
3.6: webcam ......................................................................................................... 49
3.7: Flow diagram of a generic background subtraction algorithm ..................... 51
3.8: Dc motor Schematic ..................................................................................... 52
3.9: Dc power supply ........................................................................................... 52
1 Introduction

1.1 Overview:

The control of physical systems with a digital computer or microcontroller is becoming more and more common. Examples of electromechanical servomechanisms exist in aircraft, Automobiles, mass-transit vehicles, oil refineries, and paper-making machines. Furthermore, many new digital control applications are being stimulated by microprocessor technology including control of various aspects of automobiles and household appliances. Among the advantages of digital approaches for control is the increased flexibility of the control programs and the decision making or logic capability of digital systems, which can be combined with the dynamic control function to meet other system requirements. In addition, one hardware design can be used with many different software variations on a broad range of products, thus simplifying and reducing the design time.

We will review in this research a classical experiment used in many text books to exemplify control theory fundamentals and methods of control. This experiment called double-integral system control experimentation board; also we will design and implement an experimentation board.

1.2 Problem Statement

- The majority of undergraduate courses in control have labs dealing with time response and control of dc motors. The focus of this thesis is therefore on these lab problems namely, position response, and position control of dc motors.
- This experiment help control student Understanding control theory fundamentals.

1.3 Motivations

The new and effective theories and design methodologies being continually developed in the automatic control field, Proportional–Integral–Derivative (PID) controllers are still by far the most widely adopted controllers in industry owing to the advantageous cost/benefit ratio they are
able to provide. In fact, although they are relatively simple to use, they are able to provide a satisfactory performance in many process control tasks.

1.4 Objective

- The objective of the project is to implement and control a double-integral system.
- To provide an in-depth description of dc motor position control concepts.
- To provide preliminary instruction on how to identify the parameters of a system.
- To show how different parameters and nonlinear effects such as weight of arm and saturation affect the response of the motor.
- To give a better feel for controller design through realistic examples.

1.5 Thesis Layout:

The thesis is decomposed into five chapters:

Chapter 2 (Literature review): this chapter reviews the theory of the Discrete-Time Systems, Computer Control, Integral control system, PID Control, MATLAB\textsuperscript{®} and Background subtraction techniques.

Chapter 3 (Methodology): This chapter describes the tools, requirements, implementation, analysis and verification of the system design plan.

Chapter 4 (results and discussion): This chapter describes the implementation of the tests; the results obtained from the design described in chapter 3 and the discussion and interpolation of the obtained results.

Chapter 5 (Conclusion): this chapter contains the problem faced when implementing the design in both of simulation and hardware implementation, the future work and the recommendations.
2 Literature review

2.1 Computer Control

Practically all control systems that are implemented today are based on computer control. It is therefore important to understand computer-controlled systems well. Such systems can be viewed as approximations of analog-control systems, but this is a poor approach because the full potential of computer control is not used. At best the results are only as good as those obtained with analog control it is much better to master computer-controlled systems, so that the full potential of computer control can be used. There are also phenomena that occur in computer-controlled systems that have no correspondence in analog systems. It is important for an engineer to understand this.

A computer-controlled system can be described schematically as in Fig. 2.1. The output from the process $y(t)$ is a continuous-time signal. The output is converted into digital form by the analog-to-digital (A-D) converter. The A-D converter can be included in the computer or regarded as a separate unit, according to one's preference. The conversion is done at the sampling times, $t_k$. The computer interprets the converted signal, $\{y(t_k)\}$, as a sequence of numbers, processes the measurements using an algorithm, and gives a new sequence of numbers, $\{U(t_k)\}$. This sequence is converted to an analog signal by a digital-to-analog (D-A) converter. The events are synchronized by the real-time clock in the computer. The digital computer operates sequentially in time and each operation takes some time. The D-A converter must, however, produce a continuous-time signal. This is normally done by keeping the control signal constant between the conversions. In this case the system runs open loop in the time interval between the sampling instants because the control signal is constant irrespective of the value of the output.

The computer-controlled system contains both continuous-time signals and sampled or discrete-time signals. Such systems have traditionally been called sampled-data systems, and this term will be used here as a synonym for computer-controlled systems.

The mixture of different types of signals sometimes causes difficulties. In most cases it is, however, sufficient to describe the behavior of the system at the sampling instants. The signals are then of interest only at discrete times. Such systems will be called discrete-time systems.
Discrete-time systems deal with sequences of numbers, so a natural way to represent these systems is to use difference equations.

![Schematic diagram of a computer-controlled system](image)

Figure 2.1 Schematic diagram of a computer-controlled system.

### 2.2 Computer-Control Theory

Using computers to implement controllers has substantial advantages. Many of the difficulties with analog implementation can be avoided. For example, there are no problems with accuracy or drift of the components. It is very easy to have sophisticated calculations in the control law, and it is easy to include logic and nonlinear functions. Tables can be used to store data in order to accumulate knowledge about the properties of the system. It is also possible to have effective user interfaces.

A schematic diagram of a computer-controlled system is shown in Fig. 2.1. The system contains essentially five parts: the process, the A-D and D-A converters, the control algorithm, and the clock. Its operation is controlled by the clock. The times when the measured signals are converted to digital form are called the sampling instants; the time between successive samplings is called the sampling period and is denoted by $h$. Periodic sampling is normally used, but there are, of course, many other possibilities. For example, it is possible to sample when the
output signals have changed by a certain amount. It is also possible to use different sampling periods for different loops in a system. This is called multirate sampling.

### 2.2.1 Time Dependence

The presence of the clock in Fig. 2.1 makes computer-controlled systems time-varying. Such systems can exhibit behavior that does not occur in linear time-invariant systems. Time dependence in digital filtering

A digital filter is a simple example of a computer-controlled system. Suppose that we want to implement a compensator that is simply a first-order lag. Such a compensator can be implemented using A-D conversion, a digital computer, and D-A conversion.

The first-order differential equation is approximated by a first-order difference equation. The step response of such a system is shown in Fig. 2.2. The figure clearly shows that the sampled system is not time-invariant because the response depends on the time when the step occurs. If the input is delayed, then the output is delayed by the same amount only if the delay is a multiple of the sampling period.

The phenomenon illustrated in Fig. 2.2 depends on the fact that the system is controlled by a clock (compare with Fig. 2.1). The response of the system to an external stimulus will then depend on how the external event is synchronized with the internal clock of the computer system. Because sampling is often periodic, computer-controlled systems will often result in closed-loop systems that are linear periodic systems. The phenomenon shown in Fig. 2.2 is typical for such systems.
Figure 2.2 (a) Block diagram of a digital filter. (b) Step responses (dots) of a digital computer implementation of a first-order lag for different delays in the input step (dashed) compared with the first sampling instant. For comparison the response of the corresponding continuous-time system (solid) is also shown.
2.2.2 A Naive Approach to Computer-Controlled Systems

We may expect that a computer-controlled system behaves as a continuous time system if the sampling period is sufficiently small. This is true under very reasonable assumptions. We will illustrate this with an example.

2.2.2.1 Controlling the arm of a disk drive

A schematic diagram of a disk-drive assembly is shown in Fig. Hi. Let \( J \) be the moment of inertia of the arm assembly. The dynamics relating the position \( y \) of the arm to the voltage \( u \) of the drive amplifier is approximately described by the transfer function:

\[
G(s) = \frac{k}{js^2}
\]

Where \( k \) is a constant

The purpose of the control system is to control the position of the arm so that the head follows a given track and that it can be rapidly moved to a different track. It is easy to find the benefits of improved control. Better track-keeping allows narrower tracks and higher packing density. A faster control system reduces the search time.

In this example we will focus on the search problem, which is a typical servo problem. Let \( U_c \) be the command signal and denote Laplace transforms with capital letters. A simple servo controller can be described by:

\[
U(s) = \frac{bK}{a} U_c(s) - K \frac{s + b}{s + a} Y(s)
\]

Figure 2.3: A system for controlling the position of the arm of a disk drive.
2.2

Figure 2.4: Simulation of the disk arm servo with analog (dashed) and computer control (solid). The sampling period is $h = \frac{0.2}{\omega_0}$.

This controller is a two-degree-of-freedom controller where the feedback from the measured signal is simply a lead-lag filter. If the controller parameters are chosen as

\[ a = 2\omega_0 \]
\[ b = \frac{\omega_0}{2} \]
\[ K = 2\frac{I\omega_0^2}{k} \]

A closed system with the characteristic polynomial

\[ P(s) = s^3 + 2\omega_0s^2 + 2\omega_0^2s + \omega_0^3 \]

This system has a reasonable behavior with a settling time to 5% of $5.52/\omega_0$. See Fig. 2.4. To obtain an algorithm for a computer-controlled system, the control law given by (1.2) is first written as:
This control law can be written as:

$$U(s) = \frac{bK}{a} U_c(s) - KY(s) + K \frac{a - b}{s + a} Y(s) = K \left( \frac{b}{a} U_c(s) - Y(s) + X(s) \right)$$

To obtain an algorithm for a control computer, the derivative $\frac{dx}{dt}$ is approximated with a difference. This gives:

$$\frac{x(t + h) - x(t)}{h} = -ax(t) + (a - b)y(t)$$

The following approximation of the continuous algorithm is then obtained:

$$u(t_k) = K \left( \frac{a}{b} u_c(t_k) - y(t_k) + x(t_k) \right)$$

$$x(t_k + h) = x(t_k) + h \left( (a - b)y(t_k) - ax(t_k) \right)$$

Arm position $y$ is read from an analog input. Its desired value $u_c$ is assumed to be given digitally. The algorithm has one state, variable $x$, which is updated at each sampling instant. The control law is computed and the value is converted to an analog signal. The program is executed periodically with period $h$ by a scheduling program. Because the approximation of the derivative by a difference is good if the interval $h$ is small, we can expect the behavior of the computer-controlled system to be close to the continuous-time system which shows the arm positions and the control signals for the systems with $h = \frac{0.2}{\omega_o}$. Notice that the control signal for the computer-controlled system is constant between the sampling instants. Also notice that the difference between the outputs of the systems is very small. The computer-controlled system has slightly higher overshoot and the settling time to 5% is a little longer $\frac{5.7}{\omega_o}$ instead of $\frac{5.5}{\omega_o}$ the difference between the systems decreases when the sampling period decreases.

We have thus shown that it is straightforward to obtain an algorithm for computer control simply by writing the continuous-time control law as a differential equation and approximating the derivatives by differences. The example indicated that the procedure seemed to work well if the
sampling period was sufficiently small. The overshoot and the settling time are, however, a little larger for the computer-controlled system.

2.3

Figure 2.5: Simulation of the disk arm servo with computer control having sampling rates (a) $h = \frac{0.5}{\omega_0}$ and (b) $h = \frac{1.08}{\omega_0}$. For comparison, the signals for analog control are shown with dashed lines.
2.2.3 Deadbeat Control

Controlling the arm of a disk drive example seems to indicate that a computer-controlled system will be inferior to a continuous-time example. We will now show that this is not necessarily the case. The periodic nature of the control actions can be actually used to obtain control strategies with superior performance.

2.2.3.1 Disk drive with deadbeat control

Consider the disk drive in the previous example. Figure 2.6 shows the behavior of a computer-controlled system with a very long sampling interval $h = \frac{1.4}{\omega_0}$. For comparison we have also shown the arm position, its velocity, and the control signal for the continuous controller used in disk drive example. Notice the excellent behavior of the computer-controlled system. It settles much quicker than the continuous-time system even if control signals of the same magnitude are used. The 5% settling time is $\frac{2.34}{\omega_0}$, which is much shorter than the settling time $\frac{5.5}{\omega_0}$ of the continuous system. The output also reaches the desired position without overshoot and it remains constant when it has achieved its desired value, which happens in finite time. This behavior cannot be obtained with continuous-time systems because the solutions to such systems are sums of functions that are products of polynomials and exponential functions. The behavior obtained can be also described in the following way; the arm accelerates with constant acceleration until it is halfway to the desired position and it then decelerates with constant retardation. The control strategy used has the same form as the control strategy in disk drive Example, that is:

$$u(t_k) = t_0u_c(t_k) + t_1u_c(t_{k-1}) - s_0y(t_k) - s_1y(t_{k-1}) - r_1y(t_{k-1})$$

The parameter values are different. When controlling the disk drive, the system can be implemented in such a way that sampling is initiated when the command signal is changed. In this way it is possible to avoid the extra time delay that occurs due to the lack of synchronization of sampling and command signal changes. The example shows that control strategies with different behavior can be obtained with computer control. In the particular example the response
time can be reduced by a factor of 2. The control strategy in this example is called deadbeat control because the system is at rest when the desired position is reached. Such a control scheme cannot be obtained with a continuous-time controller.

2.4

**Figure 2.6:** Simulation of the disk arm servo with deadbeat control (solid). The sampling period is \( h = \frac{1.4}{\omega_0} \). The analog controller from disk drive example is also shown (dashed).
2.2.4 Aliasing

One property of the time-varying nature of computer-controlled systems was illustrated in Fig. 2.2. We will now illustrate another property that has far reaching consequences. Stable linear time-invariant systems have the property that the steady-state response to sinusoidal excitations is sinusoidal with the frequency of the excitation signal. It will be shown that computer-controlled systems behave in a much more complicated way because sampling will create signals with new frequencies. This can drastically deteriorate performance if proper precautions are not taken.

Sampling creates new frequencies Consider the systems for control of the disk drive arm discussed previously. Assume that the frequency \( \omega_0 \) is 1 rad/s, let the sampling period be \( h = \frac{0.5}{\omega_0} \). And assume that there is a sinusoidal measurement noise with amplitude 0.1 and frequency 12 rad/s. There is clearly a drastic difference between the systems. For the continuous-time system, the measurement noise has very little influence on the arm position. It does, however, create substantial control action with the frequency of the measurement noise. The high-frequency measurement noise is not noticeable in the control signal for the computer-controlled system, but there is also a substantial low-frequency component.

We have thus made the striking observation that sampling creates signals with new frequencies; this is clearly a phenomenon that we must understand in order to deal with computer-controlled systems. At this stage we do not wish to go into the details of the theory; let it suffice to mention that sampling of a signal with frequency \( \omega \) creates signal components with frequencies:

\[
\omega_{\text{sampled}} = n\omega_s \pm \omega
\]
Figure 2.7: Simulation of the disk arm servo with computer control. The frequency $\omega_0$ is 1, the sampling period is $h=0.5$, and there is a measurement noise $n = 0.1 \sin 12t$. 
Figure 2.8: Simulation of the disk arm servo with analog and computer control. The frequency $\omega_0$ is 1, the sampling period is $h = 0.5$, and there is a measurement noise $n = 0.1 \sin 12t$.

(a) Continuous-time system; (b) sampled-data system.
2.3 Discrete-Time Systems

A key idea is to show how a continuous-time system can be transformed into a discrete-time system by considering the behavior of the signals at the sampling instants. The computer receives measurements from the process at discrete times and transmits new control signals at discrete times. The goal then is to describe the change in the signals from sample to sample and disregard the behavior between the samples. The use of difference equations then becomes a natural tool. It should be emphasized that computer-oriented mathematical models only give the behavior at the sampling points—the physical process is still a continuous-time system. Looking at the problem this way, however, will greatly simplify the treatment.

2.3.1 Sampling Continuous-Time Signals

According to dictionaries, sampling means “the act or process of taking a small part or quantity of something as a sample for testing or analysis”. In the context of control and communication, sampling means that a continuous-time signal is replaced by a sequence of numbers, which represents the values of the signal at certain times.

Sampling is a fundamental property of computer-controlled systems because of the discrete-time nature of the digital computer. Consider, for example, the system shown in Fig. 2.1. The process variables are sampled in connection with the analog conversion and then converted to digital representation for processing. The continuous-time signal that represents the process variables is thus converted to a sequence of numbers, which is processed by the digital computer. The processing gives a new sequence of numbers, which is converted to a continuous-time signal and applied to the process. In the system shown in Fig. 2.1, this is handled by the D-A converter. The process of converting a sequence of numbers into a continuous-time signal is called signal reconstruction.

For the purpose of analysis, it is useful to have a mathematical description of sampling. Sampling a continuous-time signal simply means to replace the signal by its values in a discrete
set of pointe. Let $Z$ be the positive and negative integers $Z = \{\cdots -1, 0, 1, \cdots\}$ and let $\{t_k; k \in Z\}$ be a subset of the real numbers called the sampling instants. The sampled version of the signal $f$ is then the sequence. Sampling is a linear operation. The sampling instants are often equally spaced in time, that is, $t_k = kh$. This case is called periodic sampling and $h$ is called the sampling period, or the sampling time.

The corresponding frequency $f_s = \frac{1}{h} \text{ (Hz)}$ or $\omega_s = \frac{2\pi}{h} \text{ (rad/s)}$ is called the sampling frequency. It is also convenient to introduce a notation for half the sampling frequency:

$$f_N = \frac{1}{2h} \text{ (Hz)} \text{ or } \omega_N = \frac{\pi}{h} \text{ (rad/s)},$$

which is called the Nyquist frequency.

More complicated sampling schemes can also be used. For instance, different sampling periods can be used for different control loops. This is called multirate sampling and can be considered to be the superposition of several periodic sampling schemes.

The case of periodic sampling is well understood. Most theory is devoted to this case, but systems with multirate sampling are becoming more important because of the increased use of multiprocessor systems. With modern software for concurrent processes, it is also possible to design a system as if it were composed of many different processes running asynchronously. There are also technical advantages in using different sampling rates for different variables.

### 2.3.2 Sampling a Continuous-Time State-Space System

A fundamental problem is how to describe a continuous-time system connected to a computer via A-D and D-A converters. Consider the system shown in Fig. 2.9. The signals in the computer are the sequences $\{u(t_k)\}$ and $\{y(t_k)\}$. The key problem is to find the relationship between these sequences. To find the discrete-time equivalent of a continuous-time system is called sampling a continuous-time system. The model obtained is also called a stroboscopic model because it gives a relationship between the system variables at the sampling instants only. To obtain the desired descriptions, it is necessary to describe the converters and the system. Assume that the continuous-time system is given in the following state-space form:

$$\frac{dx}{dt} = Ax(t) + Bu(t)$$
\[ y(t) = Cx(t) + Du(t) \]

The system has \( r \) inputs, \( p \) outputs, and is of order \( n \).

### 2.3.3 Zero-Order-Hold Sampling of a System

A common situation in computer control is that the D-A converter is so constructed that it holds the analog signal constant until a new conversion is commanded. This is often called a zero-order-hold circuit. It is then natural to choose the sampling instants, \( t_k \), as the times when the control changes. Because the control signal is discontinuous, it is necessary to specify its behavior at the discontinuities. The convention that the signal is continuous from the right is adopted. The control signal is thus represented by the sampled signal

\[ (u(t_k); k = \{ \ldots, -1, 0, 1, \ldots \}) \]

The relationship between the system variables at the sampling instants will now be determined. Given the state at the sampling time \( t_k \), the state at some future time \( t \) is obtained by solving state-space form equation. The state at time \( t \), where \( t_k \leq t \leq t_{k+1} \), is thus given by:

\[
x(t) = e^{A(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{A(t-s)}Bu(s)\,ds
\]

\[
= e^{A(t-t_k)}x(t_k) + \int_{t_k}^{t} e^{A(t-s)}\,d\bar{s}Bu(t_k)
\]

\[
= e^{A(t-t_k)}x(t_k) + \int_{0}^{t-t_k} e^{A(s)}\,ds Bu(t_k)
\]

\[
= \phi(t,t_k)x(t_k) + \Gamma(t,t_k)u(t_k)
\]
Figure 2.9 Block diagram of a continuous-time system connected to A-D and D-A converters.

The second equality follows because \( u \) is constant between the sampling instants. The state vector at time \( t \) is thus a linear function of \( x(t_k) \) and \( u(t_k) \). If the A-D and D-A converters in Fig. 2.9 are perfectly synchronized and if the conversion times are negligible, the input \( u \) and the output \( y \) can be regarded as being sampled at the same instants. The system equation of the sampled system at the sampling instants is then:

\[
x(t_{k-1}) = \Phi(t_{k+1}, t_k)x(t_k) + \Gamma(t_{k+1}, t_k)u(t_k)
\]

\[
y(t_k) = Cx(t_k) + Du(t_k)
\]

Where

\[
\Phi(t_{k+1}, t_k) = e^{A(t_{k+1}-t_k)}
\]

\[
\Gamma(t_{k+1}, t_k) = \int_0^{t_{k+1}-t_k} e^{As}dsB
\]

The relationship between the sampled signals thus can be expressed by the linear difference equation. Notice that the above equation does not involve any approximations. It gives the exact values of the state variables and the output at the sampling instants because the control signal is constant between the sampling instants. The model in above equations is therefore called a zero-order-hold sampling of the system in fig. (2.9). This system can also be called the zero-order-hold equivalent of fig. (2.9).
In most cases \( D = 0 \). One reason for this is because in computer-controlled systems, the output \( y \) is first measured and the control signal \( u(t_k) \) is then generated as a function of \( y(t_k) \). In practice it often happens that there is a significant delay between the A-D and D-A conversions. However, it is easy to make the necessary modifications. The state vector at times between sampling points is given by the state at time \( t \) equation. This makes it possible to investigate the inter-sample behavior of the system. Notice that the responses between the sampling points are parts of step responses) with initial conditions, for the system. This implies that the system is running in open loop between the sampling points.

For periodic sampling with period \( h \), we have \( t_k = k h \) and the previous model simplifies to the time-invariant system.

\[
\dot{x}(kh + h) = \Phi(x(kh) + \Gamma u(kh))
\]

\[
y(kh) = Cx(kh) + Du(kh)
\]

Where

\[
\Phi = e^{Ah}
\]

\[
\Gamma = \int_0^h e^{As}dsB
\]

It follows from above that

\[
\frac{d\Phi(t)}{dt} = A\Phi(t) = \Phi(t)A
\]

\[
\frac{d\Gamma(t)}{dt} = \Phi(t)B
\]

The matrices \( \Phi \) and \( \Gamma \) therefore satisfy the equation

\[
\frac{d}{dt} \begin{pmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{pmatrix} = \begin{pmatrix} \Phi(t) & \Gamma(t) \\ 0 & I \end{pmatrix} \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix}
\]

Where \( I \) is a unit matrix of the same dimension as the number of inputs.

The matrices \( \Phi(h) \) and \( \Gamma(h) \) for the sampling period \( h \) therefore can be obtained from the block matrix.
2.3.4 How to compute $\phi$ and $\Gamma$

The calculations required to sample a continuous-time system are the evaluation of a matrix exponential and the integration of a matrix exponential. These can be done in many different ways, for instance, by using the following:

- Numerical calculation in MATLAB® or MATRIXX®.
- Series expansion of the matrix exponential.
- The Laplace transform-the Laplace transform of $\exp(At)$ is $(sl - A)^{-1}$.
- Cayley-Hamilton's theorem.
- Transformation to diagonal or Jordan forms.
- Symbolic computer algebra, using programs such as Maple® and Mathematica®.

Calculations by hand are feasible for low-order systems, $n \leq 2$, and for high-order systems with special structures. One way to simplify the computations is to compute:

$$
\varphi = \int_0^h e^{As} ds = I h + \frac{Ah^2}{2!} + \frac{A^2 h^3}{3!} + \ldots + \frac{A^i h^{i+1}}{(i+1)!} + \ldots
$$

The matrices $\phi$ and $\Gamma$ are given by

$$
\phi = I + A\varphi \\
\Gamma = \varphi B
$$

Computer evaluation can be done using several different numerical algorithms in MATLAB® or MATRIXX®.
2.4 Integral control system

Difficult process dynamics such as significant dead time prevent use of large gains, steady state error performance may be unacceptable when human process operation notice the existence of steady state errors due to changes in desired value and for disturbance they can correct for these by changing the desired value (set-point) or the controller output bias until the error disappears. This is called manual reset. Integral control is means of removing steady state errors without the need for manual reset and in fact it is sometimes called automatic reset.

Integral control can be used by itself or in combination with other control modes; and the most common mode is proportional plus integral control. Although integral control is very useful for removing or reducing steady-state errors; it has the undesirable side effects of reducing response speed and degrading stability. The reduction in speed is most readily seen in the time domain, where a step input (sudden change) to an integrator causes a ramp output, a much more gradual change. Stability degradation is most apparent in the frequency domain (Nyquist criterion) where the integrator reduce phase margin by giving an additional $90^0$ degree of phase lag at every frequency. An integrating effect will naturally appear in a system element (actuator, process, etc.) other than the controller these integrators can also be effective in reducing steady state errors. Although controllers with a single integrator are most common, double and occasionally triple integrators are useful for the more difficult steady-state errors problems.
2.5 PID Control

A Proportional–Integral–Derivative (PID) controller is a three-term controller that has a long history in the automatic control field, starting from the beginning of the last century (Bennett, 2000). Owing to its intuitiveness and its relative simplicity, in addition to satisfactory performance which it is able to provide with a wide range of processes, it has become in practice the standard controller in industrial settings. It has been evolving along with the progress of the technology and nowadays it is very often implemented in digital form rather than with pneumatic or electrical components. It can be found in virtually all kinds of control equipment’s, either as a stand-alone (single-station) controller or as a functional block in Programmable Logic Controllers (PLCs) and Distributed Control Systems (DCSs). Actually, the new potentialities offered by the development of the digital technology and of the software packages has led to a significant growth of the research in the PID control field: new effective tools have been devised for the improvement of the analysis and design methods of the basic algorithm as well as for the improvement of the additional functionalities that are implemented with the basic algorithm in order to increase its performance and its ease of use.

The success of the PID controllers is also enhanced by the fact that they often represent the fundamental component for more sophisticated control schemes that can be implemented...
when the basic control law is not sufficient to obtain the required performance or a more complicated control task is of concern.

### 2.5.1 Feedback Control

The aim of a control system is to obtain a desired response for a given system. This can be done with an open-loop control system, where the controller determines the input signal to the process on the basis of the reference signal only, or with a closed-loop control system, where the controller determines the input signal to the process by using also the measurement of the output (i.e., the feedback signal).

Feedback control is actually essential to keep the process variable close to the desired value in spite of disturbances and variations of the process dynamics, and the development of feedback control methodologies has had a tremendous impact in many different fields of the engineering. Besides, nowadays the availability of control system components at a lower cost has favoured the increase of the applications of the feedback principle (for example in consumer electronics products).

The typical feedback control system is represented in Figure 2.10. Obviously, the overall control system performance depends on the proper choice of each component. From the purposes of controller design, the actuator and sensor dynamics are often neglected (although the saturation limits of the actuator have to be taken into account) and the block diagram of Figure 2.11 is considered, where P is the process, C is the controller, F is a feed forward filter, r is the reference signal, e = r − y is the control error, u is the manipulated (control) variable, y is the process (controlled) variable, d is a load disturbance signal and n is a measurement noise signal.
2.5.2 The Three Actions of PID Control

Applying a PID control law consists of applying properly the sum of three types of control actions: a proportional action, an integral action and a derivative one. These actions are described singularly hereafter.

2.5.2.1 Proportional Action

The proportional control action is proportional to the current control error, according to the expression:
\[ u(t) = K_p e(t) = K_p (r(t) - y(t)) \]

Where \( K_p \) is the proportional gain. Its meaning is straightforward, since it implements the typical operation of increasing the control variable when the control error is large (with appropriate sign). The transfer function of a proportional controller can be derived trivially as:

\[ C(s) = K_p \]

With respect to the On–Off controller, a proportional controller has the advantage of providing a small control variable when the control error is small and therefore to avoid excessive control efforts. The main drawback of using a pure proportional controller is that it produces a steady-state error. It is worth noting that this occurs even if the process presents an integrating dynamics (i.e., its transfer function has a pole at the origin of the complex plane), in case a constant load disturbance occurs. This motivates the addition of a bias (or reset) term \( u_b \), namely,

\[ u(t) = K_p e(t) + u_b \]

The value of \( u_b \) can be fixed at a constant level (usually at \( \frac{u_{max} + u_{min}}{2} \)) or can be adjusted manually until the steady-state error is reduced to zero. It is worth noting that in commercial products the proportional gain is often replaced by the proportional band \( PB \), that is the range of error that causes a full range change of the control variable.

\[ PB = \frac{100}{K_p} \]

### 2.5.2.2 Integral Action

The integral action is proportional to the integral of the control error, i.e., it is:

\[ u(t) = K_i \int_0^t e(\tau)d\tau \]

Where \( K_i \) is the integral gain. It appears that the integral action is related to the past values of the control error. The corresponding transfer function is:

\[ C(s) = \frac{K_i}{s} \]

The presence of a pole at the origin of the complex plane allows the reduction to zero of the steady-state error when a step reference signal is applied or a step load disturbance occurs. In
other words, the integral action is able to set automatically the correct value of $u_b$ so that the steady-state error is zero. This fact is better explained in Figure 2.12, where the resulting transfer function is:

$$C(s) = K_p \left(1 + \frac{1}{T_i s}\right)$$

For this reason the integral action is also often called automatic reset. Thus, the use of a proportional action in conjunction to an integral action, i.e., of a PI controller, solves the main problems of the oscillatory response associated to an On–Off controller and of the steady-state error associated to a pure proportional controller.

It has to be stressed that when integral action is present, the so-called integrator windup phenomenon might occur in the presence of saturation of the control variable.

\[\text{Fig. 2.13: PI controller in automatic reset configuration}\]

\[\text{2.5.2.3 Derivative Action}\]

While the proportional action is based on the current value of the control error and the integral action is based on the past values of the control error, the derivative action is based on the predicted future values of the control error. An ideal derivative control law can be expressed as:

$$u(t) = K_d \frac{de(t)}{dt}$$

Where $K_d$ is the derivative gain. The corresponding controller transfer function is:

$$C(s) = K_d s$$
In order to understand better the meaning of the derivative action, it is worth considering the first two terms of the Taylor series expansion of the control error at time $T_d$ ahead:

$$
e(t + T_d) \approx e(t) + T_d \frac{de(t)}{dt}$$

If a control law proportional to this expression is considered

$$u(t) = K_p \left( e(t) + T_d \frac{de(t)}{dt} \right)$$

This naturally results in a PD controller. The control variable at time $t$ is therefore based on the predicted value of the control error at time $(t + T_d)$. For this reason the derivative action is also called anticipatory control, or rate action, or pre-act.

It appears that the derivative action has a great potentiality in improving the control performance as it can anticipate an incorrect trend of the control error and counteract for it. However, it has also some critical issues that make it not very frequently adopted in practical cases.

### 2.6 MATLAB®

MATLAB is a powerful computing system for handling the calculations involved in scientific and engineering problems. The name MATLAB® stands for MATrix LABoratory, because the system was designed to make matrix computations particularly easy.

One of the many things you will like about MATLAB (and which distinguishes it from many other computer programming systems, such as C++ and Java) is that you can use it interactively. This means you type some commands at the special MATLAB prompt, and get the answers immediately. The problems solved in this way can be very simple, like finding a square root, or they can be much more complicated, like finding the solution to a system of differential equations.

#### 2.6.1 Simulink®
Simulink® it is similar in many ways to its predecessors such as CSMP (Continuous System Modeling Program), ACSL (Advanced Continuous Simulation Language), TUTSIM (Twente University of Technology Simulator), MATRIX-X, STELLA, and EASY5. The major advantage of Simulink stems from its tight integration with MATLAB®, the data analysis and visualization program with its own structured programming language. The numerous MATLAB toolboxes in diverse areas of engineering, science, and business extend the capabilities of Simulink.

In addition to the toolboxes, there are a number of Simulink block sets that extend Simulink into various disciplines such as aerospace, communications, signal processing, image processing, and so forth.

### 2.6.1.1 BUILDING A SIMULINK® MODEL

To begin our introduction to Simulink, we will demonstrate the procedure for creating a model of a simple system and run the model to obtain useful information about its dynamic response.

### 2.6.1.2 SIMULINK® LIBRARY

The Simulink library contains blocks for representing the mathematical models of commonly occurring components in dynamic systems. The blocks are grouped in sub-libraries according to function. The standard Simulink sub-libraries are shown in the left pane of Figure 5.1. The blocks residing in the selected “Continuous” sub-library are shown in the right pane. The “Integrator” block is selected, and there is a brief description of it in the top pane. The transfer function, $1=\frac{1}{s}$, is used to designate the integrator.

Building a Simulink model of a system consists of selecting the appropriate blocks and connecting them in a way that represents the mathematical model. Inputs, when present, are implemented using blocks from the “Sources” sub-library, which can generate a host of input signals. Simulation output is saved and displayed using various blocks such as “Scopes,” “XY Graphs,” and “Displays” from the “Sinks” sub-library.
First Simulink model will simulate the dynamics of the linear second-order system model. The differential equation is:

$$\frac{d^2}{dt^2} y(t) + 2\zeta \omega_n \frac{d}{dt} y(t) + \omega_n^2 y(t) = K \omega_n^2 u(t)$$

Assuming for the moment that the second derivative term $d^2 y=dt^2$ is present in a new model window, it can be twice integrated as shown in Figure 2.13 where “ydd,” “yd,” and “y” are the Simulink variable names. The “Integrator” blocks are dragged or copied from the “Continuous” Sub-library into the model window.

By inspection of the second derivative term is a linear combination of the input $u(t)$, the output $y(t)$, and its first derivative $dy=dt$. The Simulink library browser allows us to search the standard sub-libraries for the blocks needed to “build” the second derivative and, thus, complete the Simulink model.

![Simulink Library Browser](image)

2.12

Figure 2.43: The Simulink® Library Browser.
Figure 2.15: Integrating the second derivative “ydd” twice to obtain the first derivative “yd” and output “y”.

The system parameters K, ωₙ, and ζ and the literal constant “2” are generated using a “Constant” block found in the “Sources” sub-library. The “Math” sub-library provides the additional blocks for addition and multiplication of the signals.

We have yet to specify an input or forcing function, assuming there is one. For now, let us pick a simple step input applied at t = 0. Looking in the “Sources” sub-library, the step input can be implemented with a “Constant” or “Step” block; however, the latter is more flexible should we later decide to delay the time at which the step is applied.
Numerical values of the system parameters are set by selecting the individual blocks and typing in the appropriate values in a properties dialog box. Some Simulink blocks contain several parameters, all of which should be specified or else the default values will be used. For example, the “Step” block generally requires values for “Step time,” “Initial value,” and “Final value” as shown in Figure 2.15, and the “Integrator” block requires an “Initial condition.”

Figure 2.16 shows a Simulink diagram for simulation of the unit step response of the second-order system. The choice of Simulink blocks and their location in a Simulink diagram is not unique. The appearance or layout of blocks depends to a large extent on individual user preferences. Some prefer that the diagram be the most economical in terms of Simulink blocks used. Others are more concerned with layout style, striving to make the diagram visually appealing.

Often times, the mathematical model of the system is available in block diagram form, as in the case of a control system. A Simulink diagram of the system will be strikingly similar, especially when Simulink blocks for modeling actual system components are available.
An alternate Simulink diagram for the second-order system in Equation is shown in Figure 2.17. A “Gain” block with a parameter value equal to the product $2\zeta \omega_n$ replaces the “Product” block in the inner feedback loop and the three constant blocks feeding it. Another “Gain” block is inserted in the outer feedback loop with a parameter value numerically equal to $\omega_n^2$ replacing the “Product” and “Constant” blocks in Figure 2.16. The third “Gain” block is employed to multiply the input $u(t)$ by $K\omega_n^2$, further reducing the number of blocks required.

FIGURE 2.17: Simulink® diagram for step response of a second-order system.
2.6.1.3 Running a Simulink® model

The Simulink model is similar to a conventional block diagram of a system. For a system with analog components, it embodies the algebraic and differential equations of the continuous-time math model. For inherently discrete-time systems, the Simulink model encapsulates algebraic and difference equations governing the system’s behavior. Simulink models of hybrid systems containing analog and discrete-time components implement solutions to algebraic, differential, and difference equations.

A computer program is created from the Simulink model to solve the equations that comprise the mathematical model of the system. Some of its functions include initialization of state variables, calculation of state derivatives, solution of algebraic equations, updating the state variables, and calculation of the system’s outputs. Simulink offers a variety of numerical integrators to advance the continuous-time state vector over an integration step. The user has the
option of choosing a particular integrator and step size (applicable for fixed-step size algorithms), tolerances for satisfying accuracy requirements, the simulation start and stop times, and exchanging simulation data with MATLAB via The MATLAB Workspace.

Clicking on "Simulation" in the model window menu followed by "Configuration Parameters" leads to a dialog box like the one shown in Figure 5.6 where the simulation is configured according to the user’s preferences as previously described. The improved Euler integrator (Heun’s method) with a fixed-step size of 0.01 s and simulation time of 5 s has been selected.

![Dialog box for configuring simulation.](image)

**FIGURE 2.19:** Dialog box for configuring simulation.

After configuring the simulation, the "Simulation" pull-down menu is reopened and "Start" is selected. The simulation terminates when the simulation time reaches the selected "stop time" of 5s.

The simplest way to view simulation output is to select one of the scopes and observe the time history of its input. The output of the second integrator "y" is displayed in several ways, as shown in Figures 2.19 through 2.21. Figure 2.19 is a screen capture of the scope labeled "y(t)" after running the simulation and viewing the scope output by double clicking on it.
2.18

FIGURE 2.19: Screen capture of scope output.

2.19

FIGURE 2.21: Screen capture of edited scope output.
2.6.2 Background subtraction techniques

Tracking moving objects in video sequence is an important problem in computer vision, with applications several fields, such as video surveillance and target tracking. Most techniques reported in the literature use background subtraction techniques to obtain foreground objects, and apply shadow detection algorithms exploring spectral information of the images to retrieve only valid moving objects.

A relevant problem in computer vision is the detection and tracking of moving objects in video sequences. Possible applications include surveillance, traffic monitoring and athletic performance analysis, among others.

In applications using fixed cameras with respect to the static background (e.g. stationary surveillance cameras), a very common approach is to use background subtraction to obtain an initial estimate of moving objects. Basically, background subtraction consists of comparing each new frame with a representation of the scene background: significative differences usually correspond to foreground objects. Ideally, background subtraction should detect real moving objects with high accuracy, limiting false negatives (objects pixels that are not detected) as much
as possible; at the same time, it should extract pixels of moving objects with the maximum responsiveness possible, avoiding detection of transient spurious objects, such as cast shadows, static objects, or noise.

In particular, the detection of cast shadows as foreground objects is very common, producing undesirable consequences. For example, shadows can connect different people walking in a group, generating a single object (typically called blob) as output of background subtraction. In such cases, it is more difficult to isolate and track each person in the group.

There are several techniques for shadow detection in video sequences, and the vast majority of them are based on color video sequences. Although color images indeed provide more information for shadow detection.

2.6.2.1 The proposed algorithm

- The background model of W4 and propose a small improvement to the model.
- Novel method for shadow segmentation of foreground pixels, based on normalized cross-correlations and pixel ratios.

2.6.2.1.1 Background Scene Modeling

W4 uses a model of background variation that is a bimodal distribution constructed from order statistics of background values during a training period, obtaining robust background model even if there are moving foreground objects in the field of view, such as walking people, moving cars, etc. It uses a two stage method based on excluding moving pixels from background model computation. In the first stage, a pixel wise median filter over time is applied to several seconds of video (typically 20-40 seconds) to distinguish moving pixels from stationary pixels (however, our experiments showed that 100 frames ≈ 3.3 seconds are typically enough for the training period, if not too many moving objects are present). In the second stage, only those stationary pixels are processed to construct the initial background model. Let $V$ be an array containing $N$ consecutive images, $V^k(i,j)$ be the intensity of a pixel $(i,j)$ in the $k$-th image of $V$, $\sigma (i, j)$ and $\lambda (i, j)$ be the standard deviation and median value of intensities at pixel $(i, j)$ in all
images in $V$, respectively. The initial background model for a pixel $(i, j)$ is formed by a three-dimensional vector: the minimum $m(i, j)$ and maximum $n(i, j)$ intensity values and the maximum intensity difference $d(i, j)$ between consecutive frames observed during this training period. The background model $B(i, j) = [m(i, j), n(i, j), d(i, j)]$, is obtained as follows:

$$
\begin{bmatrix}
    m(i, j) \\
    n(i, j) \\
    d(i, j)
\end{bmatrix} =
\begin{bmatrix}
    \min_z V^z(i, j) \\
    \max_z V^z(i, j) \\
    \max_z |V^z(i, j) - V^{z-1}(i, j)|
\end{bmatrix}
$$

Where $z$ are frames satisfying $|Vz(i, j) - \lambda(i, j)| \leq 2\sigma(i, j)$. This condition guarantees that only stationary pixels are computed in the background model, i.e., $Vz(i, j)$ is classified as a stationary pixel.

After the training period, an initial background model $B(i, j)$ is obtained. Then, each input image $I_t(i, j)$ of the video sequence is compared to $B(i, j)$, and a pixel $(i, j)$ is classified as a background pixel if:

$$I_t(i, j) - m(i, j) \leq k\mu \quad \text{or} \quad I_t(i, j) - n(i, j) \leq k\mu$$

Where $\mu$ is the median of the largest inter frame absolute difference image $d(i, j)$, and $k$ is a fixed parameter (the authors suggested the value $k = 2$). It can be noted that, if a certain pixel $(i, j)$ has an intensity $m(i, j) < I_t(i, j) < n(i, j)$ at a certain frame $t$, it should be classified as background (because it lies between the minimum and maximum values of the background model). However, above equation may wrongly classify such pixel as foreground, depending on $k, \mu, m(i, j)$ and $n(i, j)$. For example, if $\mu = 5, k = 2, m(i, j) = 40, n(i, j) = 65$ and $I_t(i, j) = 52$, also above Equation would classify $I_t(i, j)$ as foreground, even though it lies between $m(i, j)$ and $n(i, j)$. To solve this problem, we propose an alternative test for foreground detection, and classify $I_t(i, j)$ as a foreground pixel if:

$$I_t(i, j) > (m(i, j) - k\mu) \quad \text{and} \quad I_t(i, j) > (n(i, j) - k\mu)$$

Figure 2.22 illustrates an example of background subtraction (using $k = 2$, as in all other examples in this paper). The background image (median of frames across time) is shown in Figure 1(a), a certain frame of the video sequence is shown in Figure 1(b), and detected
foreground objects are shown in Figure 1(c). It can be noticed that two kinds of shadows were detected: on the left, shadow was caused by obstruction of indirect light; on the right, shadow was produced by direct sunlight blocking.

2.21

Figure 2.23: (a) Background image. (b) A certain frame of the video sequence. (c) Detected foreground objects.

2.22

Figure 2.24: Shadow detection using different thresholds $L_{ncc}$. (a) $L_{ncc} = 0.90$ (b) $L_{ncc} = 0.95$ (c) $L_{ncc} = 0.98$ produced by direct sunlight blocking.

2.6.2.2 Shadow identification

In shadowed regions, it is expected that a certain fraction $\alpha$ of incoming light is blocked. Although there are several factors that may influence the intensity of a pixel in shadow, we assume that the observed intensity of shadow pixels is directly proportional to incident light; consequently, shadowed pixels are scaled versions (darker) of corresponding pixels in the background model. As noticed by other authors, the normalized cross correlation (NCC) can be
useful to detect shadow pixel candidates, since it can identify scaled versions of the same signal. In this work, we use the NCC as an initial step for shadow detection, and refine the process using local statistics of pixel ratios, as explained next.

2.6.2.2.1 Detection of shadow pixel candidates

Let $B(i,j)$ be the background image formed by temporal median filtering, and $I(i,j)$ be an image of the video sequence. For each pixel $(i,j)$ belonging to the foreground, consider a $(2N+1) \times (2N+1)$ template $T_{ij}$ such that $T_{ij}(n,m) = I(i+n, j+m)$, for $-N \leq n \leq N$, $-N \leq m \leq N$ (i.e. $T_{ij}$ corresponds to a neighborhood of pixel $(i,j)$). Then, the NCC between template $T_{ij}$ and image $B$ at pixel $(i,j)$ is given by:

$$NCC(i,j) = \frac{ER(i,j)}{EB(i,j)ET_{ij}}$$

Where

$$ER(i,j) = \sum_{n=-N}^{N} \sum_{m=-N}^{N} B(i+n, j+m)T_{ij}(n,m)$$

$$EB(i,j) = \sqrt{\sum_{n=-N}^{N} \sum_{m=-N}^{N} B(i+n, j+m)^2}$$

$$ET_{ij} = \sqrt{\sum_{n=-N}^{N} \sum_{m=-N}^{N} T_{ij}(n,m)^2}$$

For a pixel $(i,j)$ in a shadowed region, the NCC in a neighboring region $T_{ij}$ should be large (close to one), and the energy $ET_{ij}$ of this region should be lower than the energy $EB(i,j)$ of the corresponding region in the background image. Thus, a pixel $(i,j)$ is pre-classified as shadow if:

$$NCC(i,j) \geq L_{ncc} \text{ and } ET_{ij} < EB(i,j)$$

Where $L_{ncc}$ is a fixed threshold.

If $L_{ncc}$ is low, several foreground pixels corresponding to moving objects may be misclassified as shadows. On the other hand, selecting a larger value for $L_{ncc}$ results in less false
positives, but pixels related to actual shadows may not be detected. In fact, the influence of the threshold $L_{ncc}$ for shadow detection can be observed in Figure 2.23. This Figure illustrates the application of our shadow detector in the foreground image of Figure 2.22 (c) using $N = 4$, for different thresholds $L_{ncc}$. Black pixels are foreground pixels, and gray pixels correspond to shadowed pixels according to previous Equation. Our experiments indicated that choosing $L_{ncc} = 0.95$ results in a good compromise between false positives and false negatives, and that $N = 4$ is a good neighborhood size.

2.6.2.2.2 Shadow refinement

The NCC provides a good initial estimate about the location of shadowed pixels, by detecting pixels for which the surrounding neighborhood is approximately scaled with respect to the reference background. However, some background pixels related to valid moving objects may be wrongly classified as shadow pixels. To remove such false positives, a refinement stage is applied to all pixels that satisfy previous Equation. The proposed refinement stage consists of verifying if the ratio $I(i, j)/B(i, j)$ in a neighborhood around each shadow pixel candidate is approximately constant, by computing the standard deviation of $I(i, j)/B(i, j)$ within this neighborhood. More specifically, we consider a region $R$ with $(2M+1)\times(2M+1)$ pixels (we used $M = 1$ in all experiments) centered at each shadow pixel candidate $(i, j)$, and classify it as a shadow pixel if:

$$\text{std}_R \left( \frac{I(i,j)}{B(i,j)} \right) < L_{\text{std}} \quad \text{and} \quad L_{\text{low}} \leq \left( \frac{I(i,j)}{B(i,j)} \right) < 1$$

Where $\text{std}_R \left( \frac{I(i,j)}{B(i,j)} \right)$ is the standard deviation of quantities $I(i, j)/B(i, j)$ over the region $R$, and $L_{\text{std}}$, $L_{\text{low}}$ are thresholds. More precisely, $L_{\text{std}}$ controls the maximum deviation within the neighborhood being analyzed, and $L_{\text{low}}$ prevents the misclassification of dark objects with very low pixel intensities as shadowed pixels. To determine values for $L_{\text{std}}$ and $L_{\text{low}}$, we conducted the following experiment. We printed a chart with several gray tones and analyzed its pixel values under direct sunlight, building a background model. We evaluated these pixels across time, when a moving cloud caused progressive light occlusion, and computed values $\text{std}_R \left( \frac{I(i,j)}{B(i,j)} \right)$. Experimentally obtained values were $L_{\text{std}} = 0.05$ and $L_{\text{low}} = 0.5$ (however, we believe that further
studies on the selection of $L_{\text{std}}$ and $L_{\text{low}}$ are needed). It should be noticed that in sunny days shadows may be very strong, and information about pixel intensity in the umbra may be completely lost. In such cases, $I(i, j)/B(i, j)$ is usually very small, and shadows may be misclassified as valid foreground objects. Also, we apply morphological operators to foreground pixels after shadow removal, to complete empty spaces and remove isolated pixels. We apply sequentially a closing and an opening operator with a $5\times5$ diamond-shaped structuring element.

Stauder and colleagues also used the local variance for shadow detection. However, they did not compare each frame of the video sequence with a background model; in their approach, pixel ratios were computed for consecutive frames, which may cause erroneous detection in rotating objects. An example of the shadow refinement technique applied to the initial shadow detection of Figure 2.23 is depicted in Figure 2.24(a). In this Figure, darker gray pixels correspond to the initial shadow detection, and lighter gray pixels correspond to the final shadow detection. Figure 2.24(b) shows all foreground pixels after shadow removal, and Figure 2.24(c) shows the final result after applying morphological operators.

![Image](image_url)

**Figure 2.25**: (a) Final shadow detection (shadow pixels are represented by light gray). (b) Foreground objects after shadow removal. (c) Elimination of gaps and isolated pixels through morphological operators.

In this work, we improved an existing method for background subtraction and proposed a novel technique for shadow detection in grayscale video sequences. In our approach, the normalized cross-correlation is applied to foreground pixels, and candidate shadow pixels are obtained. A refinement process is then applied to further improve shadow segmentation.
Experimental results showed that the proposed technique performs well in video sequences containing strong shadows (occlusion of direct sunlight) and weak shadows (occlusion of indirect light), being suited for both indoor and outdoor applications. Other shadow detection techniques based on grayscale images assume smooth gradient variation in shadowed regions, and are more appropriate to indoor applications only. With the proposed technique, persons walking close to each other connected by shadows can be successfully tracked individually.
3 Methodology

3.1 Overview

This project was designed hardware and software to make a **Dc motor with arm operating in servo mode**.

The arm installed with Dc motor shaft so that it can move freely on an axle. The problem of controlling a Dc motor in this manner is classical to control system, since the system is inherently unstable. This type of system is also difficult to control manually, and therefore calls for use of electronic controls.

In this project, a control algorithm will be developed and implemented digitally using a computer as a controller and webcam as a position sensor and a working demonstration will be built.

Ultimately, this project will show the effectiveness of a digital control system to stabilize a Dc motor quickly, and it will demonstrate the robustness of the controlled system to unexpected disturbances.

3.2 Features and Objectives

- **Dc motor Control**. The goal of this project is to develop a control system that will keep a Dc motor stop smoothly in set point position. The angle from Dc position and set point position will provide the feedback with which the system will determine how to move the arm. The system will be able to maintain arm stability when it is started at set point position and respond to disturbances to the arm angle.

- **Webcam as sensor Position**. It is important that the webcam can read arm position to provide feedback the system. Making use of the MATLAB program became possible to use it as position sensor. By making a tracing program for webcam to trace laser beam.
3.3 **Hardware and Software Used**

The hardware will be built using the following major components. The specific selection of these components will be discussed later.

- Metal rod for arm.
- DC motor.
- Personal computer with MATLAB® application.
- Parallel cable.
- DAC08.
- Buffer 74-244
- Power supply for the DAC, buffer and motor.
- Webcam.
- Laser source.

The control algorithms were implemented in MATLAB. The program will handle two main signals, which will be further discussed in the Technology section.

- Motor Control.
- Webcam feedback.
3.4 **System block diagram**

The hardware is basically composed of two level elements: the webcam level and the Controller level. The controller model includes the motor itself, and it is the source of an arm position signal which is sent to the controller via webcam model which drives the motors so as to set point position. The Controller determines the appropriate motor input in real-time, based on the values received for the webcam position.

![System block diagram](image)
3.5 Design Concept

The concept of this project is the demonstration of the application of control theory to an arm with Dc motor controller. Therefore we selected the arrangement that would most clearly develop this concept. The design of a Dc motor control demonstration can be broken down into two parts: hardware and software.

Similar projects have used stand-alone devices requiring a motor, and controller to be built into one machine. However, research on some of these projects evidenced that many extraneous factors are added to the system when this physical arrangement is used. Primarily, the great addition of weight increases the load on the motor, which impairs its ability to respond quickly to disturbances.

A MATLAB program controller is used to drive the DC motors. The MATLAB program designed to drive a DC motor forward or backward at any voltage from 0 to the maximum voltage. The program determines the angle and the direction of the set point.

The arm has two main physical requirements: a metal arm that is free to rotate in a plane X and a laser source which installed on it.

3.5.1 Hardware Design

The electrical component of our design needs to receive a signal from the webcam as an input, take this input to perform calculations, and use the output of the calculations to drive a DC motor.

3.5.1.1 Motor

Our initial design included a 12V DC motor to drive the arm. We chose a dc motor because it was available immediately and it is easy to use. Therefore, a traditional DC motor is more accurate arm model. It also provides for smooth motion at high speeds and rapid change in speed. Also, it would enable us to always know the position of the arm based on the installed laser. The problem with a Dc motor is that it is not designed for a project with our requirements; it not provides very precise movement.
3.5.1.2 DAC\(^1\) (digital to analog convertor)

To drive a DC motor from parallel port must have convert digital signals to analog dc signals, so DAC chip are used.

The DAC0800 series are monolithic 8-bit high-speed current-output digital-to-analog converters (DAC) featuring typical settling times of 100 ns. When used as a multiplying DAC, monotonic performance over a 40 to 1 reference current range is possible. The DAC0800 series also features high compliance complementary current outputs to allow differential output voltages of 20 Vp-p with simple resistor loads. The reference-to-full-scale current matching of better than ±1 LSB eliminates the need for full-scale trims in most applications while the nonlinearities of better than ±0.1% over temperature minimizes system error accumulations.

\(^1\) See appendix B
3.5.1.3 Buffer 74AHC244

Octal buffer\(^2\) use as protection for parallel port.

![Buffer 74AHC244](image)

Figure 3.4: Buffer74-244

3.5.1.4 Parallel port

The Parallel Port\(^3\) is the most commonly used port for interfacing homemade projects. This port will allow the input of up to 9 bits or the output of 12 bits at any one given time, thus requiring minimal external circuitry to implement many simpler tasks. The port is composed of 4 control lines, 5 status lines and 8 data lines. It's found commonly on the back of your PC as a D-Type 25 Pin female connector. There may also be a D-Type 25 pin male connector. This will be a serial RS-232 port and thus, is a totally incompatible port.

\(^2\) See appendix C
\(^3\) See appendix D
3.5.1.5 Webcam

A webcam is a video camera that feeds its images in real time to a computer or computer network, often via USB, Ethernet, or Wi-Fi.

Figure 3.6: webcam
3.5.2 Software Design

3.5.2.1 MATLAB®

To perform system control by using Simulink features, MATLAB can drive the motor through parallel port via parallel cable. Also MATLAB receive webcam feedback through USB (universal serial bus).

MATLAB provide all function needed to make sure system is controlled. The program will handle the storing of variables within the main loop of the program and performing the appropriate calculations on these values. A main requirement of the software is speed. The main loop must finish extremely quickly.

3.5.2.2 Background subtraction

Making use of this technology we able to trace the arm.

3.5.2.3 Background subtraction Algorithm

- It must be robust against changes in illumination.
- It should avoid detecting non-stationary background objects such as swinging leaves, lights and shadow cast by moving objects.
- Finally, its internal background model should react quickly to changes in background such as starting and stopping of vehicles.
Figure 3.7: Flow diagram of a generic background subtraction algorithm.

• **Advantages:**
  
  – A different “threshold” is selected for each pixel.
  
  – These pixel-wise “thresholds” are adapting by time.
  
  – Objects are allowed to become part of the background without destroying the existing background model.
  
  – Provides fast recovery.

• **Disadvantages:**
  
  – Cannot deal with sudden, drastic lighting changes.
  
  – Initializing the Gaussians is important (median filtering).
  
  – There are relatively many parameters, and they should be selected intelligently.
3.6 The Design implementation

Firstly, we implemented the external circuit board as shown in fig. 3.8. This circuit receive digital signal from parallel port as input to buffer and DAC, then it’s converted to analog output to drive the motor. Also voltage supply as shown in fig.3.9, to operate the buffer and the DAC chips was implemented.

Secondly, we wrote MATLAB function\(^4\) to trace the position of the leaser, to determine movement of the arm.

![Dc motor Schematic](image1)

**Figure 3.8: Dc motor Schematic**

![Dc power supply](image2)

**Figure 3.9: Dc power supply**

\(^4\) The MATLAB function code is illustrated in Appendix A
4 The Results and Discussion

4.1 Introduction

The implementation divided in two sections hardware and software. The results are obtained from both types of implementations are shown and discussed in this chapter. The hardware results are compared with the simulation results and with theoretical results. Finally, the results should determine to which standard the whole project fulfills the objective.

4.2 Design implementation test:

- Code was written to accomplish the testing in section 3.7
- Code was written to implement the PID control algorithm in discrete time.
- Output from parallel port also was checked.

4.3 Design implementation limitations

The specific hardware and software selections in this project will depend on the quality of performance in each aspect. The hardware is mainly affected by DAC. The DAC was not working as expected, thus the final analog signal was not obtained practically. Finding more time and suitable resistors values it could be easily implemented.

4.4 Discussion Summary

This control algorithm therefore requires the measurement of position. The initial design technology using a Dc motor wouldn’t allow us measure of position. However, alternatives for accurate measurement of position using encoders proved to be highly expensive. Sonar or optical sensors, though less expensive, were seen to create additional difficulties, such as nonlinear output and low resolution. Thus it was out of the financial and temporal scope of this project to implement position measurement. As a result, it was impractical to implement state feedback control.
5 Conclusion & recommendations

5.1 Conclusion

This project has an immediate application in engineering education, for it presents a physical demonstration of modern control techniques which are taught in various fields of engineering. Specifically, the project will impress upon students the quality of the PID control method in stabilizing an inherently unstable system.

This project could also be extended to be the focus of control laboratory exercises through the use of modifiable control parameters. The main constants used in each control algorithm implemented could be made modifiable by the student, perhaps through the use of simple knobs on a user console. This would allow the students to experiment with the effects of changing each value, and it would demonstrate the value of calculation in determining optimal values.

5.2 Recommendation

It is proposed to add features to the current Dc motor arm Control System that will enhance the functionality and quality of the system. One enhancement is the inclusion of several different controllers, such as state feedback, LEAD and LAG controllers. This would allow the user to compare the performance of the control algorithms. This feature would build upon and enhance the current pedagogical value of the system.

Second enhancement we found that a common method of positioning controlling a DC motor is with a PWM operated H-bridge. However, alternatives for accurate measurement of position using encoders proved to be highly expensive.
References

Appendix A

This appendix contains the function code written to perform the system control algorithm.

```matlab
function z = trace(ref,cam,set_point_degree)

set_point_pexil=set_point_degree*4+100;
dio=digitalio('parallel');
addline(dio,0:7,'out');
set(cam,'TriggerRepeat',Inf);

start(cam);
while(1)
    pic=getsnapshot(cam);

    diff_image = pic - ref;
    gray_image = rgb2gray(diff_image);
    binary_image = gray_image > 20;
labeled_image = bwlabel(binary_image, 8);

    blobMeasurements = regionprops(labeled_image,gray_image, 'MaxIntensity');
ar=[blobMeasurements.MaxIntensity];
m=max(ar);
idx = find([blobMeasurements.MaxIntensity] == m);
```
blobMeasurements = regionprops(labeled_image, 'centroid');

x_coordinate = blobMeasurements(idx).Centroid(1);
y_coordinate = blobMeasurements(idx).Centroid(2);

z=[x_coordinate  y_coordinate]

error=set_point_pexil - x_coordinate

parralel_value=error*40;

str = dec2bin(parralel_value);

putvalue(dio,str);
Appendix B

This appendix contain specifications of DAC08

### ELECTRICAL CHARACTERISTICS

$V_s = \pm 15\, \text{V}, I_{\text{SS}} = 2.0\, \text{mA}, -55^\circ \text{C} \leq T_s \leq +125^\circ \text{C}$ for DAC08/DAC08A, $0^\circ \text{C} \leq T_s \leq +70^\circ \text{C}$ for DAC08E and DAC08H, $-40^\circ \text{C}$ to $+85^\circ \text{C}$ for DAC08C, unless otherwise noted. Output characteristics refer to both $I_{\text{OS}}$ and $I_{\text{OUT}}$.

#### Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>DAC08A/DAC08B</th>
<th>DAC08E</th>
<th>DAC08B</th>
<th>DAC08C</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution, Monotonicity</td>
<td>$\Delta L$</td>
<td>$\Delta L = 0.1$</td>
<td>$\pm 0.039$</td>
<td>$\pm 0.039$</td>
<td>$\pm 0.039$</td>
<td>$\pm 0.039$</td>
<td>Bits</td>
</tr>
<tr>
<td>Nonlinearity</td>
<td>$L_{\text{NL}}$</td>
<td>$L_{\text{NL}} = 0$</td>
<td>$L_{\text{NL}} = 0$</td>
<td>$L_{\text{NL}} = 0$</td>
<td>$L_{\text{NL}} = 0$</td>
<td>$L_{\text{NL}} = 0$</td>
<td>%FS</td>
</tr>
<tr>
<td>Setting Time</td>
<td>$T_{\text{S}}$</td>
<td>$T_{\text{S}} = 25^\circ \text{C}$</td>
<td>$T_{\text{S}} = 25^\circ \text{C}$</td>
<td>$T_{\text{S}} = 25^\circ \text{C}$</td>
<td>$T_{\text{S}} = 25^\circ \text{C}$</td>
<td>$T_{\text{S}} = 25^\circ \text{C}$</td>
<td>ns</td>
</tr>
<tr>
<td>Propagation Delay Per Bit</td>
<td>$T_{\text{PD}}$</td>
<td>$T_{\text{PD}} = 25^\circ \text{C}$</td>
<td>$T_{\text{PD}} = 25^\circ \text{C}$</td>
<td>$T_{\text{PD}} = 25^\circ \text{C}$</td>
<td>$T_{\text{PD}} = 25^\circ \text{C}$</td>
<td>$T_{\text{PD}} = 25^\circ \text{C}$</td>
<td>ns</td>
</tr>
<tr>
<td>All Bits Switched</td>
<td>$B_{\text{ALL}}$</td>
<td>$B_{\text{ALL}} = 0$</td>
<td>$B_{\text{ALL}} = 0$</td>
<td>$B_{\text{ALL}} = 0$</td>
<td>$B_{\text{ALL}} = 0$</td>
<td>$B_{\text{ALL}} = 0$</td>
<td>ns</td>
</tr>
<tr>
<td>Full-Scale Tempo</td>
<td>$T_{\text{FS}}$</td>
<td>$T_{\text{FS}} = 500, \text{kHz}$</td>
<td>$T_{\text{FS}} = 500, \text{kHz}$</td>
<td>$T_{\text{FS}} = 500, \text{kHz}$</td>
<td>$T_{\text{FS}} = 500, \text{kHz}$</td>
<td>$T_{\text{FS}} = 500, \text{kHz}$</td>
<td>ppm/°C</td>
</tr>
<tr>
<td>Output Voltage Compliance (True Compliance)</td>
<td>$V_{\text{OC}}$</td>
<td>$V_{\text{OC}} = 2, \text{V}$</td>
<td>$V_{\text{OC}} = 2, \text{V}$</td>
<td>$V_{\text{OC}} = 2, \text{V}$</td>
<td>$V_{\text{OC}} = 2, \text{V}$</td>
<td>$V_{\text{OC}} = 2, \text{V}$</td>
<td>V</td>
</tr>
<tr>
<td>Full Range Current</td>
<td>$I_{\text{FS}}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>mA</td>
</tr>
<tr>
<td>Full Range Symmetry</td>
<td>$I_{\text{FS}}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>mA</td>
</tr>
<tr>
<td>Zero-Current Range</td>
<td>$I_{\text{FS}}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>$I_{\text{FS}} = 2, \text{mA}$</td>
<td>mA</td>
</tr>
<tr>
<td>Output Current Noise</td>
<td>$I_{\text{OC}}$</td>
<td>$I_{\text{OC}} = 2, \text{mA}$</td>
<td>$I_{\text{OC}} = 2, \text{mA}$</td>
<td>$I_{\text{OC}} = 2, \text{mA}$</td>
<td>$I_{\text{OC}} = 2, \text{mA}$</td>
<td>$I_{\text{OC}} = 2, \text{mA}$</td>
<td>mA</td>
</tr>
<tr>
<td>Logic Input Levels</td>
<td>$V_{\text{IL}}$</td>
<td>$V_{\text{IL}} = 0, \text{V}$</td>
<td>$V_{\text{IL}} = 0, \text{V}$</td>
<td>$V_{\text{IL}} = 0, \text{V}$</td>
<td>$V_{\text{IL}} = 0, \text{V}$</td>
<td>$V_{\text{IL}} = 0, \text{V}$</td>
<td>V</td>
</tr>
<tr>
<td>Logic Input Current</td>
<td>$I_{\text{IL}}$</td>
<td>$I_{\text{IL}} = 0, \text{mA}$</td>
<td>$I_{\text{IL}} = 0, \text{mA}$</td>
<td>$I_{\text{IL}} = 0, \text{mA}$</td>
<td>$I_{\text{IL}} = 0, \text{mA}$</td>
<td>$I_{\text{IL}} = 0, \text{mA}$</td>
<td>mA</td>
</tr>
<tr>
<td>Logic Input Swing</td>
<td>$V_{\text{IL}}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>mA</td>
</tr>
<tr>
<td>Logic Input Swing</td>
<td>$V_{\text{IL}}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>$V_{\text{IL}} = 2.0, \text{V}$</td>
<td>mA</td>
</tr>
<tr>
<td>Reference Bias Current</td>
<td>$I_{\text{FB}}$</td>
<td>$I_{\text{FB}} = 2, \text{mA}$</td>
<td>$I_{\text{FB}} = 2, \text{mA}$</td>
<td>$I_{\text{FB}} = 2, \text{mA}$</td>
<td>$I_{\text{FB}} = 2, \text{mA}$</td>
<td>$I_{\text{FB}} = 2, \text{mA}$</td>
<td>mA</td>
</tr>
<tr>
<td>Reference Input Slew Rate</td>
<td>$dV/dt$</td>
<td>$dV/dt = 200, \text{V/s}$</td>
<td>$dV/dt = 200, \text{V/s}$</td>
<td>$dV/dt = 200, \text{V/s}$</td>
<td>$dV/dt = 200, \text{V/s}$</td>
<td>$dV/dt = 200, \text{V/s}$</td>
<td>mA/μs</td>
</tr>
<tr>
<td>Power Supply Sensitivity</td>
<td>$PSL_{\text{SS}}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>mA/V</td>
</tr>
<tr>
<td>Power Supply Sensitivity</td>
<td>$PSL_{\text{SS}}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>$PSL_{\text{SS}} = 4.5, \text{V}$</td>
<td>mA/V</td>
</tr>
</tbody>
</table>

#### Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>DAC08A/DAC08B</th>
<th>DAC08E</th>
<th>DAC08B</th>
<th>DAC08C</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Supply Current</td>
<td>$I_{\text{P}}$</td>
<td>$V_{\text{SS}} = \pm 5, \text{V}, I_{\text{P}} = 1.0, \text{mA}$</td>
<td>$I_{\text{P}} = 1.0, \text{mA}$</td>
<td>$I_{\text{P}} = 1.0, \text{mA}$</td>
<td>$I_{\text{P}} = 1.0, \text{mA}$</td>
<td>$I_{\text{P}} = 1.0, \text{mA}$</td>
<td>mA</td>
</tr>
<tr>
<td>Power Dissipation</td>
<td>$PD$</td>
<td>$V_{\text{SS}} = \pm 5, \text{V}, I_{\text{P}} = 1.0, \text{mA}$</td>
<td>$PD = 33, \text{mW}$</td>
<td>$PD = 33, \text{mW}$</td>
<td>$PD = 33, \text{mW}$</td>
<td>$PD = 33, \text{mW}$</td>
<td>mW</td>
</tr>
</tbody>
</table>

1-B
DAC08 pin connection:

![DAC08 Pin Connection Diagram]

*Figure 2. 16-Lead Dual In-Line Package (Q and P Suffixes)*
Appendix C

This appendix contains Buffer 74LS244 Dc characteristic’s, operating condition and limiting values.

**DC CHARACTERISTICS**

**74AHC family**

Over recommended operating conditions; voltages are referenced to GND (ground = 0 V).

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>$V_{CC} (V)$</th>
<th>$T_{amb} (^{\circ}C)$</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>OTHER</td>
<td>MIN.</td>
<td>TYP.</td>
<td>MAX.</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>HIGH-level input voltage</td>
<td>2.0</td>
<td>1.5</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
<td>2.1</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td>3.85</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>LOW-level input voltage</td>
<td>2.0</td>
<td>–</td>
<td>–</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
<td>–</td>
<td>–</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td>–</td>
<td>–</td>
<td>1.65</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>HIGH-level output voltage; all outputs $V_{I} = V_{IH}$ or $V_{IL}$; $I_{O} = -50 \mu A$</td>
<td>2.0</td>
<td>1.9</td>
<td>2.0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
<td>2.9</td>
<td>3.0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5</td>
<td>4.4</td>
<td>4.5</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>HIGH-level output voltage $V_{I} = V_{IH}$ or $V_{IL}$; $I_{O} = -4.0 \ mA$</td>
<td>3.0</td>
<td>2.53</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5</td>
<td>3.94</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>LOW-level output voltage; all outputs $V_{I} = V_{IH}$ or $V_{IL}$; $I_{O} = 50 \mu A$</td>
<td>2.0</td>
<td>–</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
<td>–</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5</td>
<td>–</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>LOW-level output voltage $V_{I} = V_{IH}$ or $V_{IL}$; $I_{O} = 4 \ mA$</td>
<td>3.0</td>
<td>–</td>
<td>–</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5</td>
<td>–</td>
<td>–</td>
<td>0.36</td>
</tr>
<tr>
<td>$I_{I}$</td>
<td>input leakage current $V_{I} = V_{CC}$ or GND</td>
<td>5.5</td>
<td>–</td>
<td>–</td>
<td>0.1</td>
</tr>
<tr>
<td>$I_{OZ}$</td>
<td>3-state output OFF current $V_{I} = V_{IH}$ or $V_{IL}$; $V_{O} = V_{CC}$ or GND</td>
<td>5.5</td>
<td>–</td>
<td>–</td>
<td>±0.25</td>
</tr>
<tr>
<td>$I_{Q}$</td>
<td>quiescent supply current $V_{I} = V_{CC}$ or GND; $I_{O} = 0$</td>
<td>5.5</td>
<td>–</td>
<td>–</td>
<td>4.0</td>
</tr>
<tr>
<td>$C_{I}$</td>
<td>input capacitance</td>
<td>–</td>
<td>–</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>
### RECOMMENDED OPERATING CONDITIONS

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>CONDITIONS</th>
<th>74AHC</th>
<th>74AHCT</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>DC supply voltage</td>
<td>2.0 5.0 5.5 4.5 5.0 5.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIL</td>
<td>input voltage</td>
<td>0 – 5.5</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VOL</td>
<td>output voltage</td>
<td>0 – VCC</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tamb</td>
<td>operating ambient temperature range</td>
<td>see DC and AC characteristics per device</td>
<td>-40 25 +85 -40 25 +85 °C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t&lt;sub&gt;rr&lt;/sub&gt; (ΔV/Δf)</td>
<td>input rise and fall rates</td>
<td>V&lt;sub&gt;CC&lt;/sub&gt; = 3.3 V ±0.3 V</td>
<td>– – 100 – –</td>
<td>ns/V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>V&lt;sub&gt;CC&lt;/sub&gt; = 5 V ±0.5 V</td>
<td>– – 20 – –</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134); voltages are referenced to GND (ground = 0 V).

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>CONDITIONS</th>
<th>MIN.</th>
<th>MAX.</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC</td>
<td>DC supply voltage</td>
<td>–0.5 +7.0</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIL</td>
<td>input voltage range</td>
<td>–0.5 +7.0</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I&lt;sub&gt;IL&lt;/sub&gt;</td>
<td>DC input diode current</td>
<td>V&lt;sub&gt;IL&lt;/sub&gt; = 0.5 V; note 1</td>
<td>–20 mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I&lt;sub&gt;OL&lt;/sub&gt;</td>
<td>DC output diode current</td>
<td>V&lt;sub&gt;OL&lt;/sub&gt; &lt; -0.5 V or V&lt;sub&gt;OL&lt;/sub&gt; &gt; V&lt;sub&gt;CC&lt;/sub&gt; + 0.5 V; note 1</td>
<td>±20 mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I&lt;sub&gt;OL&lt;/sub&gt;</td>
<td>DC output source or sink current</td>
<td>-0.5 V &lt; V&lt;sub&gt;OL&lt;/sub&gt; &lt; V&lt;sub&gt;CC&lt;/sub&gt; + 0.5 V</td>
<td>±25 mA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>DC V&lt;sub&gt;CC&lt;/sub&gt; or GND current</td>
<td>– – 75 mA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tamb</td>
<td>storage temperature range</td>
<td>–65 +150 °C</td>
<td>°C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td>power dissipation per package</td>
<td>for temperature range: -40 to +125 °C; note 2</td>
<td>– 500 mW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pin connection:

![Pin connection diagram](image-url)
Appendix D

This appendix contains interfacing parallel port:

Pin Assignments of the D-Type 25 pin Parallel Port Connector.

<table>
<thead>
<tr>
<th>Pin No (D-Type 25)</th>
<th>Pin No (Centronics)</th>
<th>SPP Signal</th>
<th>Direction In/out</th>
<th>Register</th>
<th>Hardware Inverted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>nStrobe</td>
<td>In/Out</td>
<td>Control</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Data 0</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Data 1</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Data 2</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>Data 3</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Data 4</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Data 5</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Data 6</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Data 7</td>
<td>Out</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>nAck</td>
<td>In</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>Busy</td>
<td>In</td>
<td>Status</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>Paper-Out</td>
<td>In</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PaperEnd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>Select</td>
<td>In</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>nAuto-Linefeed</td>
<td>In/Out</td>
<td>Control</td>
<td>Yes</td>
</tr>
<tr>
<td>15</td>
<td>32</td>
<td>nError / nFault</td>
<td>In</td>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>31</td>
<td>nInitialize</td>
<td>In/Out</td>
<td>Control</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>36</td>
<td>nSelect-Printer</td>
<td>In/Out</td>
<td>Control</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nSelect-In</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18 - 25</td>
<td>19-30</td>
<td>Ground</td>
<td>Gnd</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>