SKETCHING ROBOTIC ARM

A PROJECT REPORT
Presented by

MANUPRASAD K P
RANOOP P R
VISHNU C PANIKKAR
NAVANEETH KRISHNA T S
VYSHAKH T R
RITWIK G

Submitted to

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Government Engineering College, Thrissur

Kerala State, PIN 680 009

MARCH 2015

ABSTRACT

This project presents the design and implementation of a two-link robotic arm which is capable of sketching images. The two linkages provide two degrees of freedom which can be manipulated to locate each pixel in the drawing plane and move accordingly. The linkages are controlled according to kinematic and inverse kinematic equations. The image fed in the computer is converted into pixel form and the inverse kinematic equations are used to calculate the angles that each linkage has to make to successfully trace each pixel.

ACKNOWLEDGEMENT

I express my sincere thanks to **Dr. B Jayanand**, Head of the Department, Electrical and Electronics Engineering for providing the necessary support for this project. In particular, I express my heartfelt gratitude to our project coordinator, **Prof.V Beena** for her moral support and guidance to complete my project on time. I also acknowledge my deep sense of gratitude to all well-wishers and friends who have helped me directly or indirectly to complete this work.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

The manufacturers have always been craving for minimization of human intervention in production processes so as to reduce the cost of production and to increase the quality of products. This dream can be made into reality in this age of automation with the cutting edge technology. While we may provide robots with the capacity to perform useful tasks, we may also want those to be similar to how a human would operate. A robotic arm will be one of the primary examples that would come to one's mind. Such an arm not only operates in various production facilities but also mimics the movement of a human arm.

This project deals with realization of such a robotic arm which could mimic the complex movements of a human arm and performing the task of sketching in minimum time possible. The complexity of sketching an image comes from the accurate positioning of each pixel according to the original image fed to the system. This also follows a recent development in the robotic community, i.e. using robots for artistic and entertainment purposes. The robot follows the same procedure of an artist, by analysing the original image at first and then reproducing it on the canvas. The already existing methods such as image processing, servo motor control etc. are used in carrying out these complex tasks.

2 PROJECT OBJECTIVE

This project aims to realize a robotic arm which can be used to sketch simple images. The arm is envisioned in such a way that, even users without much expertise can operate and redesign it to fit their usage. The project is also aimed at producing the arm at a cheaper price.

To implement this idea, the designed robotic arm needs to have two degrees of freedom. i.e we can control the two linkages of the robotic arm in order to make it sketch an image. Each of the two linkages can move approximately 180 degrees. These movements are combined together to form a large number of possible configurations which in turn can be encoded with specific pixels in the drawing plane. The resolution of image can be controlled by changing the number of divisions in the 180 degrees of each linkage.

3 ROBOTICS

Robotics is the branch of mechanical engineering, electrical engineering and computer science that deals with the design, construction, operation, and application of robots, as well as computer systems for their control, sensory feedback, and information processing.

These technologies deal with automated machines that can take the place of humans in dangerous environments or manufacturing processes, or resemble humans in appearance, behavior, and/or cognition. Many of today's robots are inspired by nature contributing to the field of bio-inspired robotics.

3.1 ROBOTIC ASPECTS

There are many types of robots; they are used in many different environments and for many different uses, although being very diverse in application and form they all share three basic similarities when it comes to their construction:

Robots all have some kind of mechanical construction, a frame, form or shape designed to achieve a particular task. For example, a robot designed to travel across heavy dirt or mud, might use caterpillar tracks. The mechanical aspect is mostly the creator's solution to completing the assigned task and dealing with the physics of the environment around it. Form follows function.

Robots have electrical components which power and control the machinery. For example, the robot with caterpillar tracks would need some kind of power to move the tracker treads. That power comes in the form of electricity, which will have to travel through a wire and originate from a battery, a basic electrical circuit. Even gas powered machines that get their power mainly from gas still require an electrical current to start the gas using process which is why most gas powered machines like cars, have batteries. The electrical aspect of robots is used for movement (through motors), sensing (where electrical signals are used to measure

things like heat, sound, position, and energy status) and operation (robots need some level of electrical energy supplied to their motors and sensors in order to activate and perform basic operations)

All robots contain some level of computer programming code. A program is how a robot decides when or how to do something. In the caterpillar track example, a robot that needs to move across a muddy road may have the correct mechanical construction, and receive the correct amount of power from its battery, but would not go anywhere without a program telling it to move. Programs are the core essence of a robot, it could have excellent mechanical and electrical construction, but if its program is poorly constructed its performance will be very poor or it may not perform at all. There are three different types of robotic programs: remote control, artificial intelligence and hybrid. A robot with remote control programming has a pre-existing set of commands that it will only perform if and when it receives a signal from a control source, typically a human being with a remote control. It is perhaps more appropriate to view devices controlled primarily by human commands as falling in the discipline of automation rather than robotics. Robots that use artificial intelligence interact with their environment on their own without a control source, and can determine reactions to objects and problems they encounter using their pre-existing programming. Hybrid is a form of programming that incorporates both AI and RC functions.

The study of robotics concerns itself with the desire to synthesize some aspects of human function by the use of mechanisms, sensors, actuators, and computers. Obviously, this is a huge undertaking, which seems certain to require a multitude of ideas from various "classical" fields.

Currently, different aspects of robotics research are carried out by experts in various fields. It is usually not the case that any single individual has the entire area of robotics in his or her grasp. A partitioning of the field is natural to expect. At a relatively high level of abstraction, splitting robotics into four major areas seems reasonable: mechanical manipulation, locomotion, computer vision, and artificial intelligence.

3.2 HISTORY OF ROBOTICS

Although the science of robotics only came about in the 20th century, the history of human-invented automation has a much lengthier past. In fact, the ancient Greek engineer Hero of Alexandria, produced two texts, *Pneumatica* and *Automata*, that testify to the existence of hundreds of different kinds of "wonder" machines capable of automated movement. Of course, robotics in the 20th and 21st centuries has advanced radically to include machines capable of assembling other machines and even robots that can be mistaken for human beings.

The word *robotics* was inadvertently coined by science fiction author Isaac Asimov in his 1941 story "Liar!" Science fiction authors throughout history have been interested in man's capability of producing self-motivating machines and life forms, from the ancient Greek myth of Pygmalion to Mary Shelley's Dr. Frankenstein and Arthur C. Clarke's HAL 9000. Essentially, a robot is a re-programmable machine that is capable of movement in the completion of a task. Robots use special coding that differentiates them from other machines and machine tools, such as CNC. Robots have found uses in a wide variety of industries due to their robust resistance capabilities and precision function.

- Historical Robotics

  Many sources attest to the popularity of automatons in ancient and medieval times. Ancient Greeks and Romans developed simple automatons for use as tools, toys, and as part of religious ceremonies. Predating modern robots in industry, the Greek God Hephaestus was supposed to have built automatons to work for him in a workshop. Unfortunately, none of the early automatons are extant.

  In the middle Ages, in both Europe and the Middle East, automatons were popular as part of clocks and religious worship. The Arab polymath Al-Jazari (1136-1206) left texts describing and illustrating his various mechanical devices, including a large elephant clock that moved and sounded at the hour, a musical robot band and a waitress automaton that

served drinks. In Europe, there is an automaton monk extant that kisses the cross in its hands. Many other automata were created that showed moving animals and humanoid figures that operated on simple cam systems, but in the 18th century, automata were understood well enough and technology advanced to the point where much more complex pieces could be made. French engineer Jacques de Vaucanson is credited with creating the first successful biomechanical automaton, a human figure that plays a flute. Automata were so popular that they traveled Europe entertaining heads of state such as Frederick the Great and Napoleon Bonaparte.

- Victorian Robots

The Industrial Revolution and the increased focus on mathematics, engineering and science in England in the Victorian age added to the momentum towards actual robotics. Charles Babbage (1791-1871) worked to develop the foundations of computer science in the early-to-mid nineteenth century, his most successful projects being the difference engine and the analytical engine. Although never completed due to lack of funds, these two machines laid out the basics for mechanical calculations. Others such as Ada Lovelace recognized the future possibility of computers creating images or playing music.

Automata continued to provide entertainment during the 19th century, but coterminous with this period was the development of steam-powered machines and engines that helped to make manufacturing much more efficient and quick. Factories began to employ machines to either increase workloads or precision in the production of many products.

- The Twentieth Century to Today

In 1920, Karel Capek published his play R.U.R. (Rossum's Universal Robots), which introduced the word "robot." It was taken from an old Slavic word that meant something akin to "monotonous or forced labor." However, it was thirty years before the first industrial robot went to work.

In the 1950s, George Devol designed the Unimate, a robotic arm device that transported die castings in a General Motors plant in New Jersey, which started work in 1961. Unimation, the company Devol founded with robotic entrepreneur Joseph Engelberger, was the first robot manufacturing company. The robot was originally seen as a curiosity, to the extent that it even appeared on The Tonight Show in 1966. Soon, robotics began to develop into another tool in the industrial manufacturing arsenal.

Robotics became a burgeoning science and more money was invested. Robots spread to Japan, South Korea and many parts of Europe over the last half century, to the extent that projections for the 2011 population of industrial robots are around 1.2 million. Additionally, robots have found a place in other spheres, as toys and entertainment, military weapons, search and rescue assistants, and many other jobs. Essentially, as programming and technology improve, robots find their way into many jobs that in the past have been too dangerous, dull or impossible for humans to achieve. Indeed, robots are being launched into space to complete the next stages of extraterrestrial and extrasolar research.

4 ROBOTIC ARM

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm; the arm may be the sum total of the mechanism or may be part of a more complex robot. The links of such a manipulator are connected by joints allowing either rotational motion (such as in an articulated robot) or translational (linear) displacement. The links of the manipulator can be considered to form a kinematic chain. The terminus of the kinematic chain of the manipulator is called the end effector and it is analogous to the human hand.

4.1 TYPES OF ROBOTIC ARMS

- Cartesian robot / Gantry robot: Used for pick and place work, application of sealant, assembly operations, handling machine tools and arc welding. It's a robot whose arm has three prismatic joints, whose axes are coincident with a Cartesian coordinator.
- Cylindrical robot: Used for assembly operations, handling at machine tools, spot welding, and handling at die casting machines. It's a robot whose axes form a cylindrical coordinate system.
- Spherical robot / Polar robot (such as the Unimate): Used for handling at machine tools, spot welding, die casting, fettling machines, gas welding and arc welding. It's a robot whose axes form a polar coordinate system.
- SCARA robot: Used for pick and place work, application of sealant, assembly operations and handling machine tools. This robot features two parallel rotary joints to provide compliance in a plane.
- Articulated robot: Used for assembly operations, die casting, fettling machines, gas welding, arc welding and spray painting. It's a robot whose arm has at least three rotary joints.

- Parallel robot: One use is a mobile platform handling cockpit flight simulators. It's a robot whose arms have concurrent prismatic or rotary joints.
- Anthropomorphic robot: Similar to the robotic hand Luke Skywalker receives at the end of The Empire Strikes Back. It is shaped in a way that resembles a human hand, i.e. with independent fingers and thumbs.

5 THE MECHANICS AND CONTROL OF ROBOTIC ARM

In the study of robotics, we are constantly concerned with the location of objects in three/ two dimensional space. These objects are the links of the manipulator, the parts and tools with which it deals, and other objects in the manipulator's environment. At a crude but important level, these objects are described by just two attributes: position and orientation. Naturally, one topic of immediate interest is the manner in which we represent these quantities and manipulate them mathematically.

In order to describe the position and orientation of a body in space, we will always attach a coordinate system, or frame, rigidly to the object. We then proceed to describe the position and orientation of this frame with respect to some reference coordinate system.

Any frame can serve as a reference system within which to express the position and orientation of a body, so we often think of transforming or changing the description of these attributes of a body from one frame to another.

5.1 FORWARD KINEMATICS OF ROBOTIC ARM

Kinematics is the science of motion that treats motion without regard to the forces which cause it. Within the science of kinematics, one studies position, velocity, acceleration, and all higher order derivatives of the position variables (with respect to time or any other variable(s)). Hence, the study of the kinematics of manipulators refers to all the geometrical and time-based properties of the motion. Manipulators consist of nearly rigid links, which are connected by joints that allow relative motion of neighbouring links. These joints are usually instrumented with position sensors, which allow the relative position of neighbouring links to be measured. In the case of rotary or revolute joints, these displacements are called joint angles. Some manipulators contain sliding (or prismatic) joints, in which the relative displacement between links is a translation, sometimes called the joint offset.

The number of degrees of freedom that a manipulator possesses is the number of independent position variables that would have to be specified in order to locate all parts of the mechanism. This is a general term used for any mechanism. For example, a four-bar linkage has only one degree of freedom (even though there are three moving members).

In the case of typical industrial robots, because a manipulator is usually an open kinematic chain, and because each joint position is usually defined with a single variable, the number of joints equals the number of degrees of freedom. At the free end of the chain of links that make up the manipulator is the end- effector. Depending on the intended application of the robot, the end-effector could be a gripper, a welding torch, an electromagnet, or another device. We generally describe the position of the manipulator by giving a description of the tool frame, which is attached to the end-effector, relative to the base frame, which is attached to the non-moving base of the manipulator. A very basic problem in the study of mechanical manipulation is called forward kinematics. This is the static geometrical problem of computing the position and orientation of the end-effector of the manipulator. Specifically, given a set of joint angles, the forward kinematic problem is to compute the position and orientation of the tool frame relative to the base frame. Sometimes, we think of this as changing the representation of manipulator position from a joint space description into a Cartesian space description.

5.2 INVERSE KINEMATICS OF ROBOTIC ARM

Given the position and orientation of the end-effector of the manipulator, calculating all possible sets of joint angles that could be used to attain this given position and orientation is said to be the process of inverse kinematics. This is a fundamental problem in the practical use of manipulators.

This is a rather complicated geometrical problem that is routinely solved thousands of times daily in human and other biological systems. In the case of an artificial system like a robot, we will need to create an algorithm in the control computer that can make this calculation. In some ways, solution of this problem is the most important element in a manipulator system. We can think of this problem

as a mapping of "locations" in 3-D Cartesian space to "locations" in the robot's internal joint space. This need naturally arises anytime a goal is specified in external 3-D space coordinates. Some early robots lacked this algorithm—they were simply moved (sometimes by hand) to desired locations, which were then recorded as a set of joint values (i.e., as a location in joint space) for later playback.

Obviously, if the robot is used purely in the mode of recording and playback of joint locations and motions, no algorithm relating joint space to Cartesian space is needed. These days, however, it is rare to find an industrial robot that lacks this basic inverse kinematic algorithm. The inverse kinematics problem is not as simple as the forward kinematics one. Because the kinematic equations are nonlinear, their solution is not always easy (or even possible) in a closed form. Also, questions about the existence of a solution and about multiple solutions arise.

Study of these issues gives one an appreciation for what the human mind and nervous system are accomplishing when we, seemingly without conscious thought, move and manipulate objects with our arms and hands. The existence or nonexistence of a kinematic solution defines the workspace of a given manipulator. The lack of a solution means that the manipulator cannot attain the desired position and orientation because it lies outside of the manipulator's workspace.

5.3 MECHANICS OF THE PROPOSED ROBOTIC ARM

The proposed project work has two degrees of freedom which can be manipulated to accommodate and locate points in two-dimensional plane. This naturally leads to a need for representing positions of parts, of tools, and of the mechanism itself. To define and manipulate mathematical quantities that represent position and orientation, we must define coordinate systems and develop conventions for representation.

6 BLOCK DIAGRAM

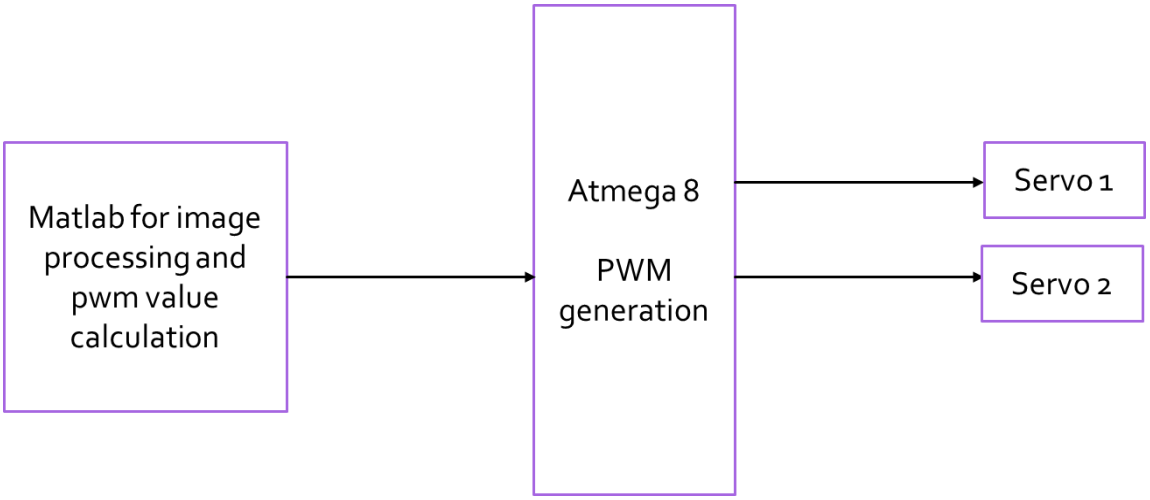


Figure 6.1 Block diagram

6.1 MATLAB

In this project Matlab is mainly used for two purposes. First one is to for finding the total sketching area of our robotic arm and the second one is to process the image the user inputs and calculate the required angles to achieve the motion.

In the first step using the mathematical model of the robotic arm total reachable sketching area of that particular arm is calculated and plotted. From this a convenient area is selected for our sketching purposes.

In the second step an image is processed and the angles required at each joint to move the pen to filled pixels are calculated using reverse kinematics equations. From this pwm values are calculated and a look up table is generated.

6.2 ATMEGA 8

Atmega 8 (in this project we have used an atmega 8 based development board) is used for the generation of required pwm values. That means the pwm values in the look up table is generated using atmega 8. These generated pwm values are fed in to servo motors for angle control. Atmega 8 is used to so that we can produce accurate pwm signals.

6.3 SERVO

Servo is the actuator which realizes the actual movement. Two servos are used to realize the angles in the two joints of our robotic arms. Using servo we can achieve very precise angle control.

7 MATHEMATICAL MODELING

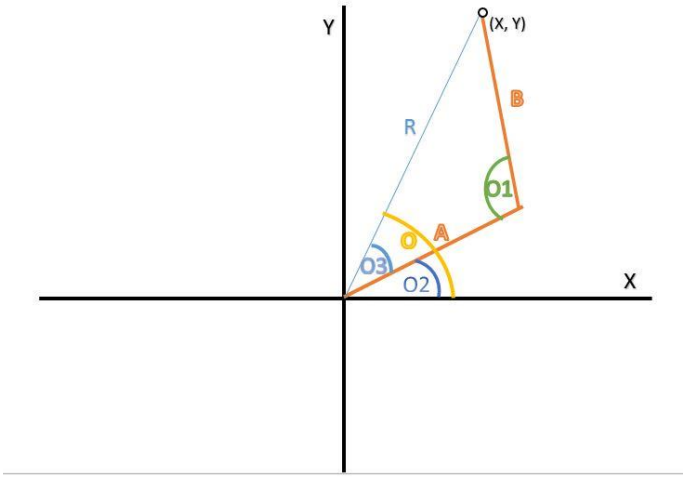


Figure 7.1 Mathematical model of arm

A&B are the lengths of arms

$$A=10cm$$

$$B=10cm$$

O1 andO2 are angles achieved by motion of servo

$$O1=0to180^{o}$$

$$O2=0to180^{o}$$

Now using this values we can find out co-ordinates of points where our robotic arm can reach

For that first

$$R^2=A^2+B^2-2*A*B*\cos(O1)$$

From this equation we obtain R

Then

$$O3=\sin^{-1}(B*\sin(O1)/R)$$

So total angle

$$O=O2+O3$$

Therefore

$$X=R*\cos(O)$$

$$Y=R*\sin(O)$$

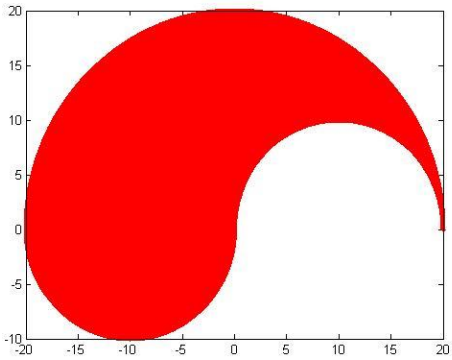


Figure 7.2 Total working area plotted using matlab

8 PROCESSING

8.1 SELECTING WORKING AREA AND CREATING LOOK UP TABLES

Next step is to select a suitable working area. That means a good rectangular place in our total sketching area where we can place our paper and do the drawing.

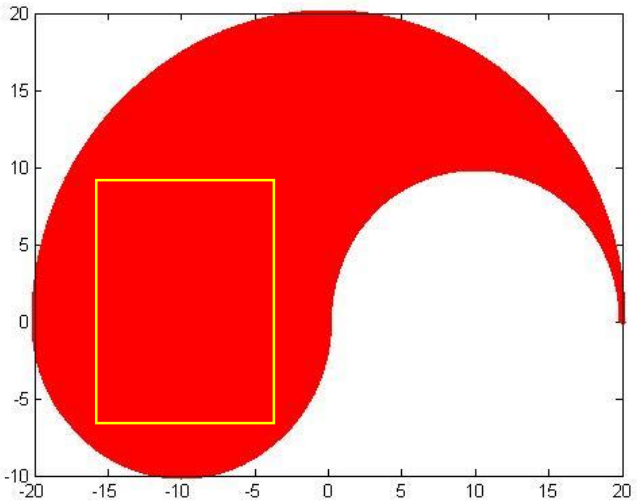


Figure 8.1 Selected working area

In this case we selected

$$X = -16 \text{ to } -4$$

$$Y = -6 \text{ to } 8$$

Now in Matlab a code was written with X and Y having above values with a resolution of .5. And two matrixes with

Number of rows = Number of variables in Y

Number of columns = Number of variables in X

These matrices will be as same size as that of the image that we are gonna input. That means our image will have X-Y pixels. Now each of this matrices are

used for each servo motors. Cells in this matrices are equivalent to the pixels in the picture. Using the inverse kinematics equations we find the angles required at each joint to move our pen to a particular pixel and this angles are stored in our matrices at corresponding position of the pixels. Likewise angles for all the pixels are calculated and stored. Now these matrices are used a look up table for our further processing. How these look up tables are utilized is explained below.

Consider a picture with pixel size 4*4. We will be processing black and white images. Now let's assume all the pixels in this picture is dark. Then when we process this picture in Matlab we will get a matrix like below for representing the picture in binary(Picture in binary form will have 0 for a dark pixel and 1 for a white pixel).

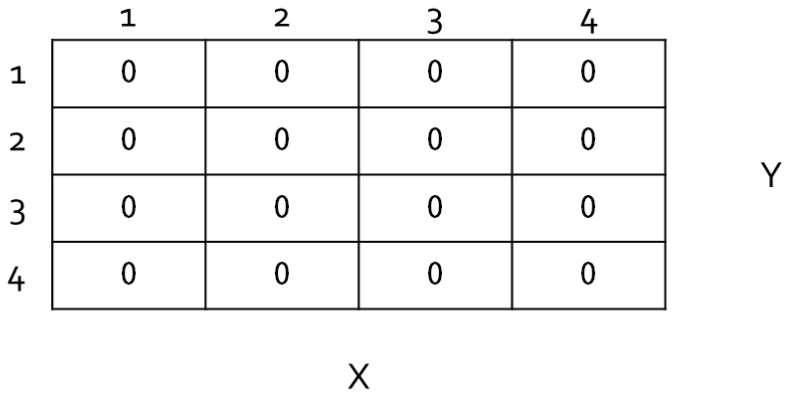


Figure 8.2 Matrix representing a dark picture

Now the other two matrices are created from this picture and to find out angles required to move our arm to a particular pixel all we have to do is use the look up table like below.

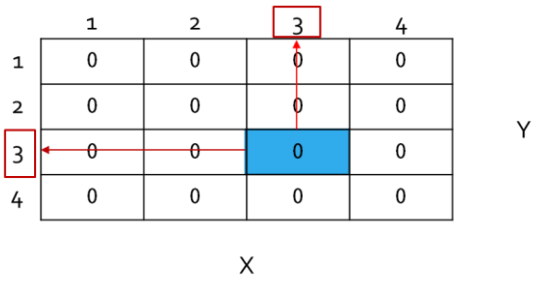


Figure 8.3 Selected pixel in image

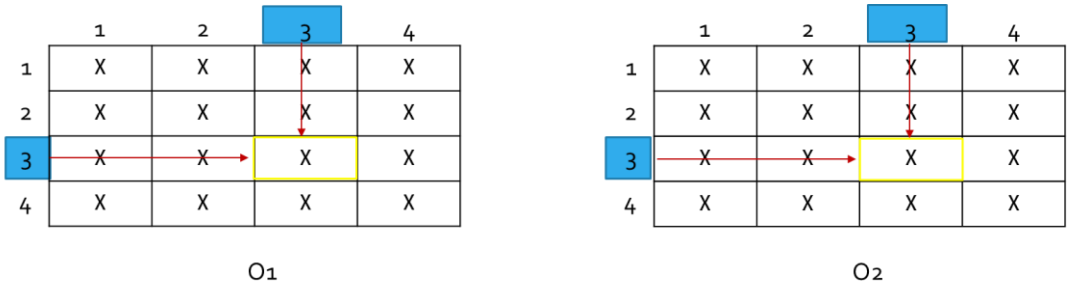


Figure 8.4 Finding the angles using look up table

8.2 HOW MOTION IS ACHIVED

Images are saved as pixels. Using image processing tools in Matlab we can convert the image in to a logical 2 dimensional array of ones and zeroes.

One represents a white pixel in picture and zero represents a filled part in picture.

By analysing this property we can find out where to move our robotic arm. And by calculating angles required we can control the servos and thus achieve the motion of arm.

Now selecting row number and column number of box filled with zero we can find O1 and O2 angles.

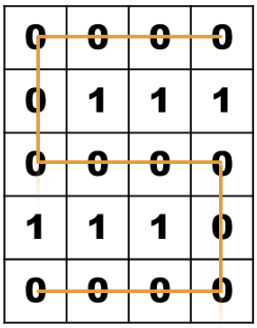


Figure 8.5 Example for how a picture is represented in binary

8.3 CONTINUITY CHECKING ALGORITHAM

In addition to the above process we have added an extra logic to track continues lines and draw it continuously. For this we used a recursive function called check. It checks for the continuity as explained below

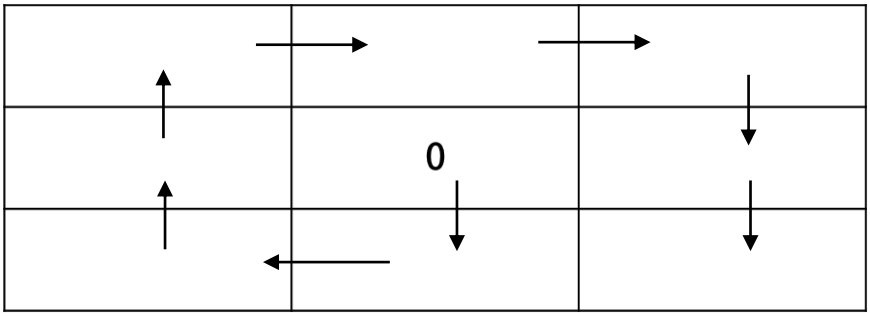
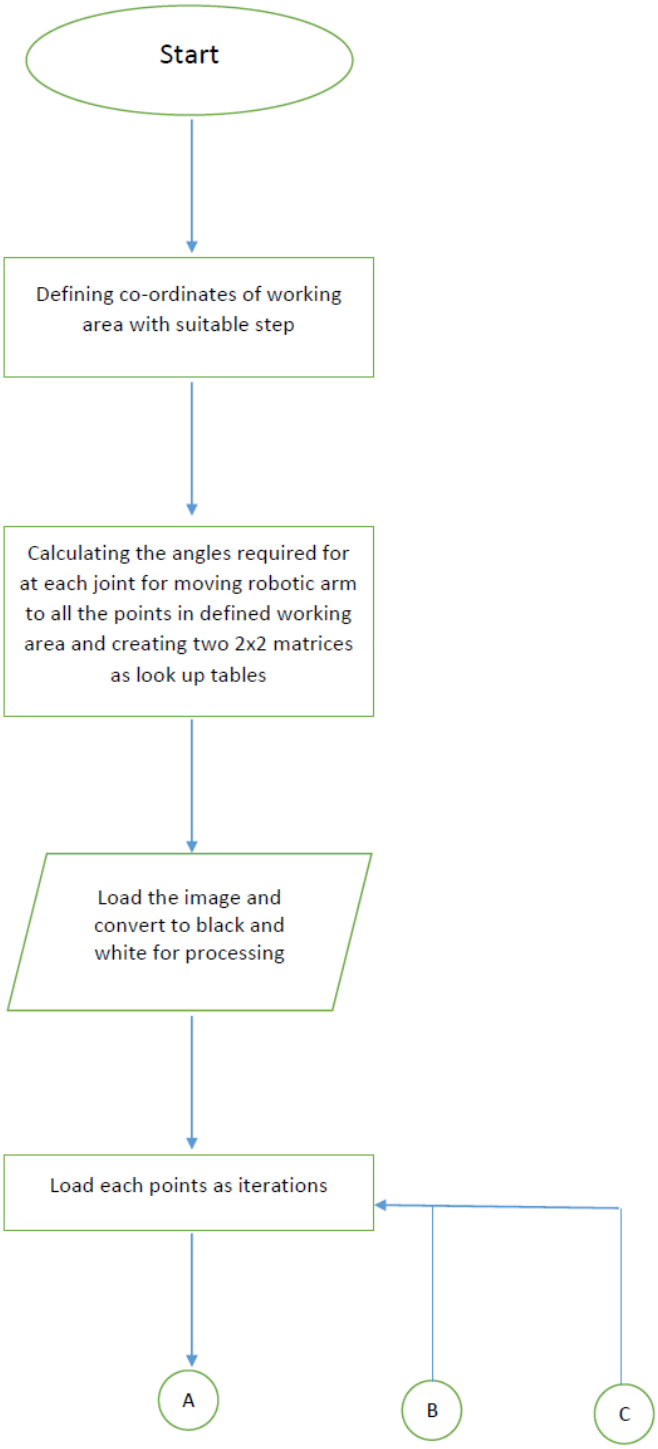


Figure 8.6 Continuity checking

In this function once dark pixel is found it will check if there is another dark pixel in the adjacent cells of it in the order above shown. If there is another dark pixel this process will repeat and continue till a pixel without any adjacent pixels are found. A flag variable is used to check if the pixel is already checked or not. If a pixel is already checked it will be skipped in this process.

8.4 FLOWCHART REPRESENTING THE VARIOUS STEPS IN THE OPERATION OF THE ROBOTIC ARM



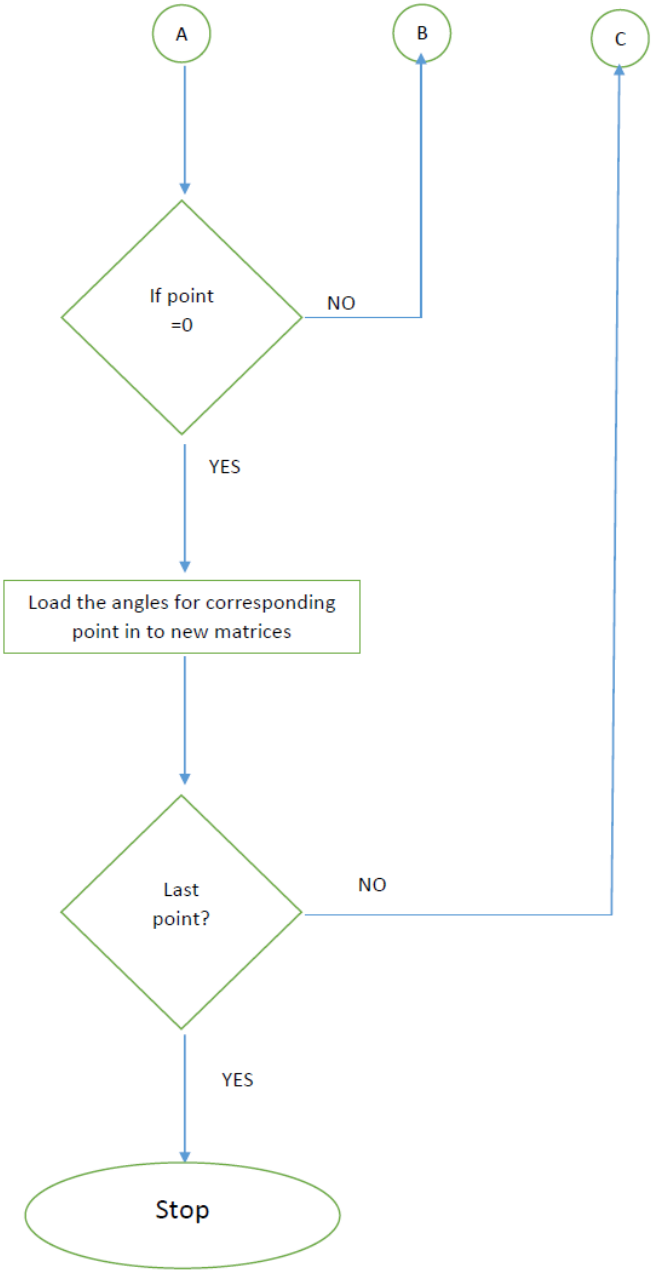


Figure 8.7 Flowchart representing the various steps in the operation of the robotic arm

9 PWM GENERATION

PWM signal is generated with the help of Atmega 8 development board. Atmega 8 was configured in fast PWM mode with frequency 50Hz using 16 bit counter. Atmega 8 is used with 12MHz crystal. In the program for PWM generation, look up tables created using Matlab was loaded and PWM values for servos are fetched.

The Fast PWM mode is based on single-slope operation. In single slope operation, the register TCNTn counts from bottom value to maximum value and its value resets to zero. The counting starts again from bottom. The register OCRn compares the value with the TCNTn register constantly. If the timer is configured in non-inverting mode, PWM output pin (OCn) goes low when the value of the above two registers matches. The OCn pin becomes high when the TCNTn register reaches at bottom value. In inverting mode OCn pin behaves opposite to non-inverting mode. For timer 0 fast PWM mode, following table shows the functionality of COM 1x[1:0] bits.

COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level)
1	1	Set OC1A/OC1B on Compare Match (Set output to high level)

In this project we have used Clear on Compare Match mode. Also we will be using the timer in Fast PWM mode with ICR1 for TOP value. For this WGM13,12,11 register bits are made high. Different modes we can select is listed below

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation[1]	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Prescaler was set to 8 by enabling CS11 bit. Different prescaler configurations are listed below

CS12	CS11	CS10	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

9.1 CALCULATING TOP VALUE

For PWM output frequency we have the equation

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

We have

$f_{clk\_I/O}$ = 12MHz

N = 8

We need the output frequency at 50Hz. So by solving the equation we get value of ICR1 as 29999

10 ADVANTAGES AND LIMITATIONS

The main advantage of the proposed robotic arm is the low cost of components. The mechanical structure is constructed of low cost aluminium plates and other easily available items. The servo motors used here are also less costly thus bringing down the total cost of production of such a robotic arm. The power consumption of the robotic arm is considerably less because of the usage of servomotors.

The construction of the arm is also easy, as the structure of the robotic arm is very simple. The structure is made in such a way that it can be easily dismantled and assembled. The program used is based on basic mathematical equations; hence it can be altered according to the needs. The use of development board for the microcontroller considerably reduces the complexities of burning the microcontroller. Because of these factors, the maintenance of the arm becomes easy.

There are many limitations in the finished robotic arm, which may have risen due to the limitations in money and time. One of them is the vibration of the mechanical structure during the operation. This may be due to the instability of the joints in the aluminium frame which was used for the construction of the robotic arm. The arm is prevented from drawing multiple unconnected images in a single sketch because of the absence of a third motion control system to move the pen up or down.

The most prominent of the limitations is the inaccuracy of sketches. This inaccuracy is linked to the usage of lower priced servo motors which do not have a feedback system in order to detect and correct errors. The usage of motors with feedbacks requires additional programming too.

11 POSSIBLE IMPROVEMENTS

Better mechanical structure can be use in order to improve the stability and thus remove the vibration of the robotic arm. The sketching can be improved by the usage of servomotors with feedback systems. The feedback can be programmed to reduce the error in rotation and thus increase the accuracy of the sketch.

A third servo motor can be used to control the up and down motion of the pen. Such an operation will ensure the sketching of multiple unconnected images in a single drawing. The whole system can be made user friendly and attractive by reducing the need of human intervention in the whole process. This can be achieved by removing the need for copying the pwm values from the output of the matlab program to the program for the microcontroller. Such a communication between the two programs will require more work in the backend.

12 CONCLUSION

A prototype of a sketching robotic arm was made. The mathematical model of the arm was simulated and tested using matlab. The pwm values generated from the matlab program was fetched to the microcontroller- atmega8. The microcontroller was used to control the servo motors and these servo motors were in turn used as the control for the robotic arm .the arm was tested and the image drawn by the arm was compared to those fed in the program. Apart from some negligible errors the arm worked properly.

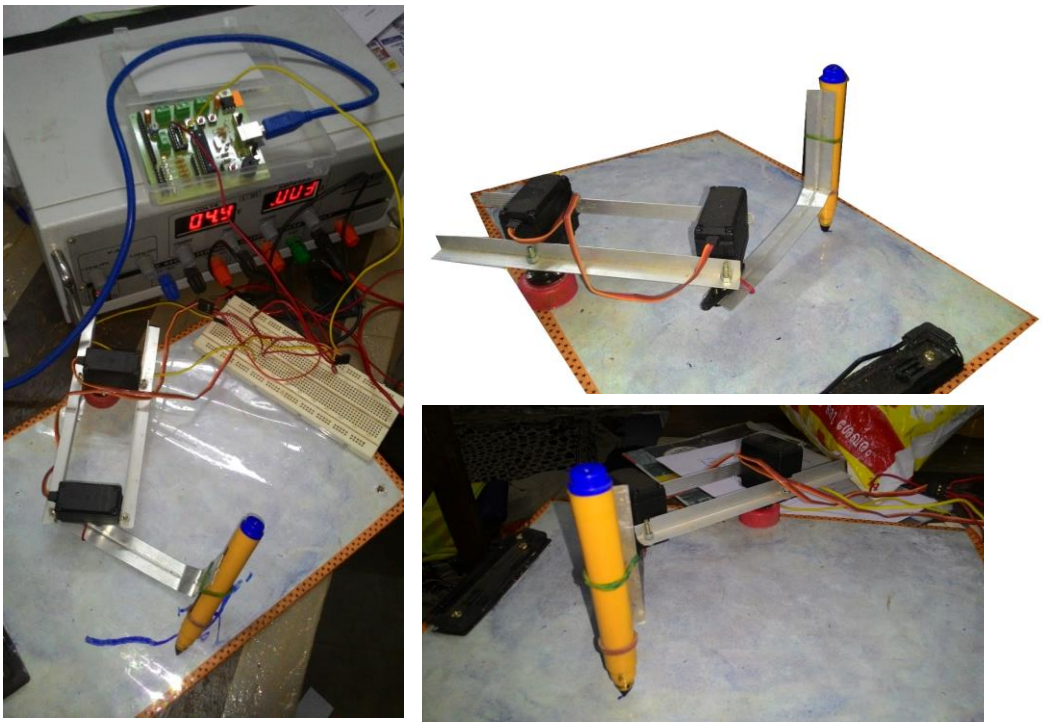


Figure 12.1 Working model of the project

13 FUTURE SCOPE

The program and the mechanical structure can be improved in order to modify it as a 3D printer. The complexity of such a system will be much higher and will be beyond the scope of a project. The drawing application of this robotic arm can be altered into several other applications requiring similar motion. Automated welding is one of such applications. It does not require much alteration of the robotic arm. But the addition of the third servo motor (for the up and down motion of the welding tool) will be inevitable.

In a much larger scale, the same logic can be used for the movement of loads in industries and factories.

14 APPENDIX

14.1 MATLAB

MATLAB(**mat**rix **lab**oratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, Fortran and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolicengine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

In 2004, MATLAB had around one million users across industry and academia.[3]MATLAB users come from various backgrounds of engineering, science, and economics. MATLAB is widely used in academic and research institutions as well as industrial enterprises.

14.2 SERVO MOTOR

Figure 14.1 servo motor

A servomotor is a rotary actuator that allows for precise control of angular position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system. Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

As the name suggests, a servomotor is a servomechanism. More specifically, it is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is some signal, either analogue or digital, representing the position commanded for the output shaft. The motor is paired with some type of encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.

More sophisticated servomotors measure both the position and also the speed of the output shaft. They may also control the speed of their motor, rather than always running at full speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

14.3 ATMEGA8

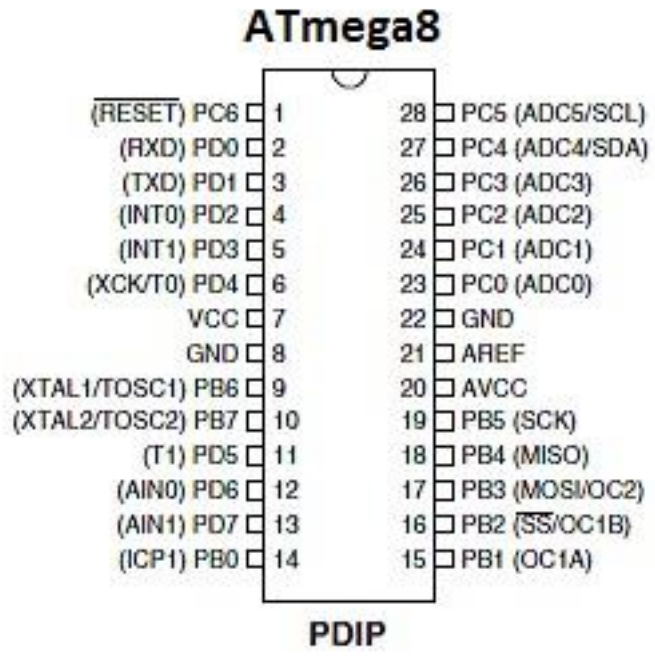


Figure 14.2 Atmega8 Pin diagram

The Atmel AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega8 provides the following features: 8 Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1 Kbyte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two- wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops

the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning.

The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash Section will continue to run while the Application Flash Section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications. The ATmega8 is supported with a full suite of program and system development tools, including C compilers, macro assemblers, program simulators, and evaluation kits

14.3.1 16-BIT TIMER/COUNTER1

The 16-bit Timer/Counter unit allows accurate program execution timing (event management),

Wave generation, and signal timing measurement. The main features are:

- **True 16-bit Design (i.e., allows 16-bit PWM)**
- **Two Independent Output Compare Units**
- **Double Buffered Output Compare Registers**
- **One Input Capture Unit**
- **Input Capture Noise Canceler**

- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch-free, Phase Correct Pulse Width Modulator (PWM)**
- **Variable PWM Period**
- **Frequency Generator**
- **External Event Counter**
- **Four Independent Interrupt Sources (TOV1, OCF1A, OCF1B, and ICF1)**

14.3.2 FAST PWM MODE

The *fast Pulse Width Modulation* or fast PWM mode (WGM13:0 = 5, 6, 7, 14, or 15) provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the Compare Match between TCNT1 and OCR1x, and set at BOTTOM. In inverting Compare Output mode output is set on Compare Match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct and phase and frequency correct PWM modes that use dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8-, 9-, or 10-bit, or defined by either ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0003), and the maximum resolution is 16-bit (ICR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

$$R_{FPWM} = \frac{log(TOP + 1)}{log(2)}$$

In fast PWM mode the counter is incremented until the counter value matches either one of the fixed values 0x00FF, 0x01FF, or 0x03FF (WGM13:0 = 5, 6, or 7), the value in ICR1 (WGM13:0 = 14), or the value in OCR1A (WGM13:0 = 15). The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 38. The figure shows fast PWM

mode when OCR1A or ICR1 is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT1 slopes represent compare matches between OCR1x and TCNT1. The OC1x Interrupt Flag will be set when a Compare Match occurs.

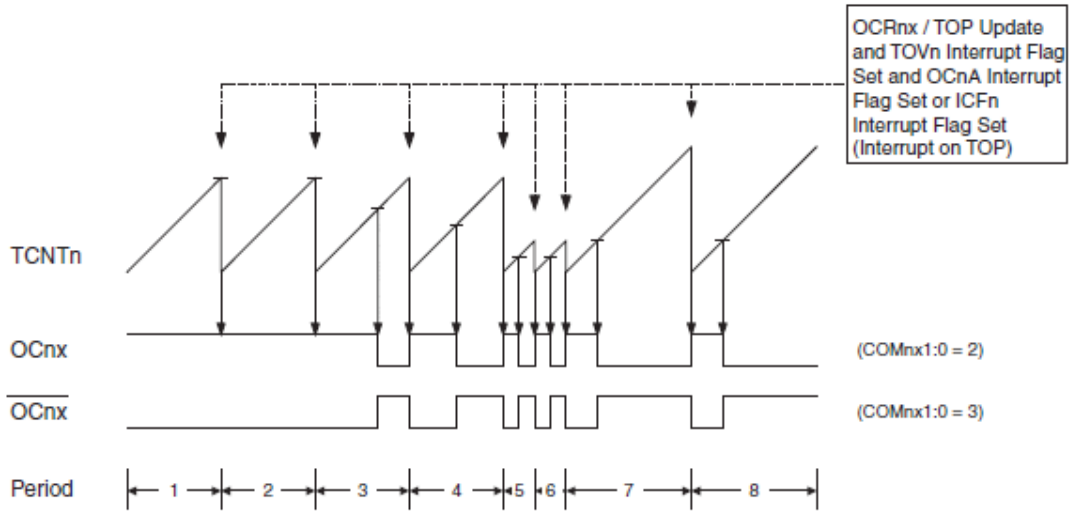


Figure 14.3 Fast PWM mode, Timing Diagram

The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OCF1A or ICF1 Flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or ICR1 is used for defining the TOP value. If one of the interrupts are enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the Compare Registers. If the TOP value is lower than any of the Compare Registers, a Compare Match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCR1x Registers are written.

The procedure for updating ICR1 differs from updating OCR1A when used for defining the TOP value. The ICR1 Register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the

Compare Match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the Compare Match can occur. The OCR1A Register, however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the Buffer Register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TCNT1 is cleared and the TOV1 Flag is set.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed (by changing the TOP value), using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In fast PWM mode, the compare units allow generation of PWM waveforms on theOC1x pins. Setting the COM1x1:0 bits to 2 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1:0 to 3. See Table 37 on page 98. The actual OC1x value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OC1x Register at the Compare Match between OCR1x and TCNT1, and clearing (or setting) the OC1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x equal to TOP will result in a constant high or low output (depending on the polarity of the output set by the COM1x1:0 bits.)

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC1A to toggle its logical level on each Compare Match (COM1A1:0 = 1). This applies only if OCR1A is used to define the TOP value (WGM13:0 = 15). The waveform generated will have a maximum frequency of fOC1A = fclk_I/O/2 when OCR1A is set to zero (0x0000). This feature is similar to the OC1A toggle in CTC mode, except the double buffer feature of the Output Compare unit is enabled in the fast PWM mode.

15 REFERENCE

- John J. Craig, “Introduction to Robotics, Mechanics and Control”, Pearson Education International,2005

- Sylvain Calinon, Julien Epiney and Aude Billard, “A Humanoid Robot Drawing Human Portraits”, IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2005)

- Shweta Patil and Sanjay Lakshminarayan, “Position Control of Pick and Place Robotic Arm”, EIE’s 2nd Intl’ Conf.Comp., Energy, Net., Robotics and Telecom,2012

- R. Brooks, “Flesh and Machines”, Pantheon Books, New York, 2002.