DESIGN AND CONSTRUCTION OF A MICROCONTROLLER-BASED FIVE DEGREE OF FREEDOM ROBOTIC ARM USING SERVO MOTORS

NICHOLUS KARIUKI NDWIGA (B. Ed(Sc.))

I56/CE/28336/2013

A Thesis Submitted in Partial Fulfillment of the Requirements for the Award of the Degree of Master of Science (Electronics and Instrumentation) in the School of Pure and Applied Sciences of Kenyatta University.

AUGUST 2018

DECLARATION

I declare that the work presented in this thesis is my original work and has not been presented for the award of a degree or any other award in any University.

Signature ..... Date 23/10/2018

Nicholus Kariuki Ndwiga - I56/CE/28336/2013

Department of physics

SUPERVISORS

We confirm that the work reported in this thesis was carried out by this candidate under our supervision.

Signature ..... Date 23.10.2018

Dr. Mathew K. Munji

Department of physics

Kenyatta University

Signature ..... Date 23/10/2018

Dr. Willis J. Ambusso

Department of physics

Kenyatta University

DEDICATION

This thesis is dedicated to my wife Scolastica Wambeti and our children.

ACKNOWLEDGEMENTS

I would wish to take this opportunity to sincerely express my appreciation to all my lecturers in the Department of physics, Kenyatta University for their constant support in developing this thesis. Special thanks to my supervisors Dr. Mathew Munji and Dr. Willis Ambusso for believing in me, their guidance and cooperation throughout my research work. I also wish to acknowledge my postgraduate colleagues: Jane Nyaga, Peter Kisavi, Moses Irungu, Dennis Nyabuto and Paul Mwoha for sharing all my moments of joy, laughter and sadness throughout my study days. I wish to again acknowledge the support accorded to me by Eng. John Kiama, Principal Iruma girls' high school Mrs. Esther Koome, Kenya Teachers Service Commission for granting me a 3 months study leave and Kenya National Research Fund for their grant to do my research. My sincere thanks are also extended to the Department of physics laboratory technical staff of Kenyatta University. I would never have been able to accomplish any of my objectives without the support of my most important people; my family. My wife Scolastica Wambeti, my daughter Patience Gakenia and my son Mark Mutugi are my pillars and source of my motivation. Thank you for your patience during the many days I was absent. Thanks to my parents Julius and Jemima, my siblings Simon, Peter, Boniface, Faith and Jackson for looking up to my success that has always given me the challenge to achieve more. Last and not least, I wish to thank the Almighty God for I believe the far I am it has taken his able hands.

TABLE OF CONTENTS

TITLE PAGE……i

DECLARATION……ii

DEDICATION……iii

ACKNOWLEDGEMENT……iv

TABLE OF CONTENT……v

LIST OF TABLES……..ix

LIST OF FIGURES……...x

ABBREVIATIONS AND ACRONYMS……...xii

ABSTRACT……...xvi

CHAPTER ONE

INTRODUCTION

1.1 Background of the study……..1

1.2 Statement of the research problem……..2

1.3 Objectives……3

1.3.1 General objective……..3

1.3.2 Specific objectives……3

1.4 Rationale……..4

CHAPTER TWO

LITERATURE REVIEW

2.1 Robotic arm…….5

2.2 PIC18F4550 based robotics…….8

2.3 Haptic technology-based robotic arm……..9

2.4 Human mind controlled robotic arm…….10

CHAPTER THREE

THEORETICAL BACKGROUND

3.1 Introduction……...12

3.2 PIC18F4550 microcontroller…….12

3.2.1 PIC18F4550 memory organization……..13

3.2.1.1 Organization of program memory……...14

3.2.1.2 Organization of data memory……..15

3.2.1.3 Data EEPROM Memory……..17

3.2.2 Reset…….19

3.2.3 Interrupts……...20

3.2.4 Input/Output ports……22

3.2.5 Timer modules……..22

3.2.6 Enhanced Universal Synchronous Asynchronous Receive Transmitter …….23

3.2.7 Oscillator configuration……23

3.3 Servo motor……...25

3.4 Liquid Crystal Display (LCD)……..28

3.5 Pull-up resistor……..29

3.6 PICkit™3 programmer/debugger……..31

3.7 LM2596 Adjustable step-down regulator……..33

CHAPTER FOUR

RESEARCH METHODOLOGY

4.1 Introduction……..34

4.2 Hardware design……34

4.2.1 Power supply…..36

4.2.2 Control circuits…….……...37

4.2.3 Debugging methods……..……....44

4.3 Software design……..……..45

4.3.1 Introduction…….……45

4.3.2 Software installations in the PC……..45

4.3.3 Software for controlling servo motors……46

4.3.4 Software for controlling the robotic arm……47

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1 Introduction……..50

5.2 Hardware results……..……...50

5.2.1 Power supply……..50

5.2.2 Control circuits……50

5.2.3 The robotic arm……..55

5.3 Software results……55

5.3.1 Installed software……55

5.3.2 Software for controlling servo motors...……58

5.3.3 Software for controlling the movements of the robotic arm…59

5.4 Data obtained……59

5.4.1 Timer0 counts……59

5.4.2 ON and OFF PWM signal production method……60

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions……63

6.2 Recommendations……64

REFERENCES……66

APPENDICES……69

Appendix A: PIC18F4550 data sheet.....69

Appendix B: Software for controlling one servo motor using interrupt……70

Appendix C: Software for controlling one servo motor without using interrupt……72

Appendix D: Software for displaying text on the LCD display……73

Appendix E: Software for debugging the system……75

Appendix F: Switch functions to select motors and give appropriate angles……76

Appendix G: Main functions for controlling the movement of the robotic arm…….81

LIST OF TABLES

Table 4.1: The PICkit™3 programmer interface with the PIC18F4550…….39

Table 4.2: The 2x16 LCD interface with PIC18F4550……..39

Table 5.1: Angle moved by the servo motor against the PWM signal……60

Table 5.2: Angle moved by the servo motor against the PWM signal……...…….61

LIST OF FIGURES

Figure 3.1: PIC18F4550 pin diagram showing the 40 pins……..12

Figure 3.2: Figure showing program memory map and stack levels of PIC18F4550….14

Figure 3.3: Schematic showing data memory map and of PIC18F4550…….16

Figure 3.4: Schematic showing special function register map of PIC18F4550……..18

Figure 3.5: Figure showing block diagram of on-chip reset circuit of PIC18F4550…..19

Figure 3.6: Schematic showing program interrupt logic diagram of PIC18F4550...…..21

Figure 3.7: Schematic showing oscillator circuit of PIC18F4550……..24

Figure 3.8: Schematic showing the internal parts of servo motor……..25

Figure 3.9: Schematic showing the principle operation of servo motor……..26

Figure 3.10: Schematic showing the PWM signals to control servo motor……27

Figure 3.11: Schematic showing a 2x16 LCD…….29

Figure 3.12: Schematic showing a pull-up resistor circuit connected to a push button..30

Figure 3.13: Schematic showing the PICkit™3 programmer…….32

Figure 3.14: Figure showing the programming connector of PICkit™3 programmer....32

Figure 3.15: Picture of the LM2596 adjustable voltage regulator……...33

Figure 4.1: Block diagram of the hardware ……..35

Figure 4.2: Block diagram of the power supply……..36

Figure 4.3: Schematic showing the circuit diagram to control one servo motor…….37

Figure 4.4: Schematic showing the circuit diagram to control different servo motors..38

Figure 4.5: Schematic showing the circuit diagram to display onto LCD…..……40

Figure 4.6: Schematic showing the circuit diagram to control robotic arm……41

Figure 4.7: Sketch showing the robotic arm design….....43

Figure 4.8: Schematic showing the software flow chart to control one servo motor…..47

Figure 4.9: Flow chart diagram for the software to control the robotic arm……..…....49

Figure 5.1: Picture showing the power supply section of the circuit……..51

Figure 5.2: Picture showing the PICkit™3 connection to the microcontroller……..51

Figure 5.3: Picture showing the control circuit for one servo motor.....52

Figure 5.4: Picture of the control circuit for three servo motor with push buttons…….52

Figure 5.5: Picture of the control circuit for five servo motor without push button.....53

Figure 5.6: Picture showing the circuit to control the robotic arm……..54

Figure 5.7: Picture of the LCD connections……..……..54

Figure 5.8: Picture of the designed robotic arm……..55

Figure 5.9: Screen shot showing the mikroC Pro and PICkit™3 software on the PC…56

Figure 5.10: Screen shot of the Proteus software for designing circuits on the PC……57

Figure 5.11: Screen shot showing the timer calculator software on the PC……57

Figure 5.12: Graph of pulse ON time against angle of rotation……..62

ABBREVIATIONS AND ACRONYMS

ADC	Analog-digital converter
ALU	Arithmetic logic unit
BOR	Brown-out reset
BSR	Bank select register
CAN	Controller area network
CCP	Capture/compare/PWM
CPU	Central processing unit
CRT	Cathode ray tube
DAC	Digital-analog converter
DOF	Degree of freedom
DSC	Digital signal controller
dsPIC	Digital signal Programmable intelligent computer
ECCP	Enhanced capture/compare/PWM
EEG	Electroencephalography
EEPROM	Electrically erasable programmable read only memory
EPROM	Erasable programmable read only memory

EUSART	Enhanced universal synchronous asynchronous receiver transmitter
FIC	Fuzzy incremental controller
FTDI	Future technology devices international
GPR	General Purpose register
GUI	Graphical user interface
I/O	Input/output
I^2C	Inter-integrated circuit
IC	Integrated circuit
ICSP™	In-circuit serial programming™
IDE	Integrated development environment
ISO	International organization for standardization
ISR	Interrupt service routine
LCD	Liquid crystal display
LED	Light emitting diode
MCLR	Master clear reset
MCU	Micro controller unit
MSSP	Master synchronous serial port

PDIP	Plastic dual in-line package
PIC	Peripheral interface controller
POR	Power on reset
PWM	Pulse width modulation
RAM	Read access memory
RC	Radio controlled
ROM	Read only memory
RUR	Rossum’s universal robots
SCADA	Supervisory control and data acquisition
SFR	Special function register
SIM	Subscriber identity module
SPI	Serial peripheral interface
SPP	Streaming parallel port
SRAM	Static random access memory
FETs	Field effect transistors
TCP	Transmission control protocol
UART	Universal asynchronous receiver/ transmitter

USAR	Urban search and rescue
USB	Universal serial bus

ABSTRACT

Integration of robotic arms into working tasks has currently increased magnificently in performing the very repetitive, dangerous or difficult tasks. Typically, a robotic arm is a mechanical arm that is programmable to mimic the behavior of a human arm in terms of how it functions. Computers and microcontrollers have widely been used in the control of robotic arms with the help of sensors, levers, buttons, wireless devices, just to mention but a few. More advanced technology has lately revolutionized their control, ranging from the haptic technology using accelerometers to human-brain control through noninvasive technology. One of the areas robotic arms are used in our day to day life is in land movers like in excavators, bulldozers, backhoes, front loaders and trenching machines. The available arms for these devices need personnel throughout their operations to control and manipulate their movements using gears, levers, pistons, pedals and sometimes buttons. There is a great need to complement their movement so that they can autonomously operate once they are powered. This research focused on the design, implementation and control of a robotic arm with five degree of freedom (DOF) using servo motors. It was designed to entirely operate by itself in a repetitive routine. A control circuit based on a PIC18F4550 microcontroller interfaced with a servo motor was built and a suitable software for the control of the rotation of motor developed. The control circuit was used to send appropriate Pulse Width Modulation (PWM) signals to different motors to produce the desired rotation. In this study five servo motors were employed to realize the robotic arm. Three servos were utilized to control the body motion including base, shoulder, and elbow and two servos were used for the motion of end effector, the wrist and the gripper. The software for the control of rotation of the motors was done using C programming language. The codes were developed and debugged using the mikroC PRO for PIC Integrated Development Environment (IDE). PICkit™3 in-circuit programming module was used to upload the program to the microcontroller through PICkit™3 programmer software. The materials were assembled and joined to construct the robotic arm which was tested in the University laboratory to demonstrate repetitive picking, lifting and dropping of objects of specific weight from one place to another without the influence of the operator.

CHAPTER ONE

INTRODUCTION

1.1 Background of the study

Robots are increasingly being integrated into performing repetitive, dangerous or difficult tasks. Robots can be used in many fields of operations including but not limited to; constructions of roads and buildings, performing repetitive tasks in industries, loading and offloading in transport, drilling, picking tools and minerals in mining, move cameras that capture videos from elevated, crowded and dangerous environment, install power cables and picking and placing medical equipment for easier surgery operations in hospitals. Fundamentally, robots should mimic human motion, and to enable them do so they have features which include foot and arms. A robot possibly will take a sense of aptitude or imagination by itself through mimicking a natural look or systematizing actions. Robotic arms can be autonomous or manually controlled (Ajayi *et al.*, 2007). Control of robotic arms has been done through computer terminals (Dylan *et al.*, 2004), joysticks, graphical user interface (Siti, 2011), accelerometers (Ashutosh, 2013), sensors (Khairul, 2009) and tentatively interfacing them with internet (Robert, 2011) so that they can be remotely controlled from any part of the world (ISO 8373-1994).

Recently there has been more advanced ways of controlling robotic arm among them being haptic technology (Bhanu *et al.*, 2012), electromyography and human mind. According to Wikipedia, haptic technology provides a tactile feedback which creates the sense of touch by applying forces, vibrations or motion to the user. It gives the feel of touch, sense and force. It's detected through an accelerometer that measures the acceleration of a moving or

vibrating body. In the application of electromyography, muscle tension generates a signal which can be transmitted to a computer using a wire or via Bluetooth to control a robotic arm (Florentinus *et al.*, 2016). The human mind controlled robotic arm uses a noninvasive technique where just by imagining moving an arm, one is able to manipulate a robotic arm. This means that people who have lost their mobility through paralysis can perform their routine activities using this invention. Microcontrollers have played a big role in the control of robots where they are programmed in such a way that the robot will perform a specific task repetitively until the end of the program or keep on repeating a task a given number of times until power is switched off (Timothy, 2007). One of the areas robotic arms are used in our day to day life is in the land movers. For instance, skid steers, backhoes, bulldozers, excavators, front loaders and trenching machines have an arm that does its activities. The activities these equipment are adopted to depend primarily on the end effector. This can be either for gripping, digging, pushing, trenching, lifting or hitting, just to mention but a few. These arms need personnel throughout their operation to control and manipulate their movements using gears, levers, pistons, pedals and sometimes buttons.

1.2 Statement of the research problem

Human beings are often tasked with activities that are very difficult and tedious to perform. Others demands for unwavering concentration to produce the same results over and over again. Some of these activities include and not limited to; landscaping and site clean-up in building and road constructions, loading and off-loading construction materials onto the transport trucks, lifting heavy construction materials like metal chases and poles, digging

holes and heavy demolitions. Robotic arms are therefore needed to work in such areas to make it easy, fast and efficient. Bulldozers, excavators and backhoe loaders are some of the robots available that have arms that help in these activities. The arms in these robots need personnel throughout the activity to give it instructions using gears, levers and sometimes buttons. There is therefore a need to compliment the movement of the robotic arm involved in such operations so that it can automatically perform its functions without the influence or interference of a human being. This arm will be programmed to pick and place objects from a fixed position repetitively without the influence of an operator.

1.3 Objectives

1.3.1 General objective

To design, implement and control a five degree of freedom robotic arm using servo motors based on a PIC microcontroller.

1.3.2 Specific objectives

i) To design a control circuit using a PIC microcontroller and interface it with a servo motor.

ii) To develop and implement software using C programming language for controlling the rotation of a servo motor.

iii) To use the microcontroller circuit to send the appropriate signals to different motors to produce the required rotation.

iv) To assemble and join the materials in construction of the robotic arm.

v) To test and implement the robotic arm.

1.4 Rationale

This research work is aimed at designing and constructing a robotic arm that does not need an operator throughout its operation time. Through the software the robotic arm works by itself once the power is put on. It repetitively picks, lift and drop objects of specific weight from one place to another without the influence or interference by the operator. This is an attempt to compliment the movement of an arm mounted on a land mover, for example an excavator so that it can do its operations without involving a human being.

CHAPTER TWO

LITERATURE REVIEW

2.1 Robotic arm

A robotic arm functions like a human arm either entirely by itself or being a part of a super complex robotic system (Ramaiah *et al.*, 2011). The task or the application to which the robotic arm is intended for is determined by the part called end effector (Aakash, 2012). Arid *et al*. (2013) designed a robotic arm that incorporated a simple linkage actuation mechanism, AC motor and spur gears in a threaded shaft arrangement. It was designed in such a way that the gripper performs the basic functions of picking, holding and grasping of objects by means of a DC motor.

Asmarashid *et al*. (2009) developed a single degree of freedom (DOF) arm robot that has similar function with other 3 to 6 DOF arm robot. It used a magnetic actuator as a gripper and the operations are controlled using PIC microcontroller and used servo motor for single movement operation. Khairul (2009) designed a pick and place robotic arm of 1DOF using PIC microcontroller. Input devices such as infrared sensors send a signal to PIC and the PIC responds accordingly. The response involves turning ON or OFF an output signal to some of output devices such as servo motor and switches.

Araian (2014) designed and implemented a 5DOF robotic arm using servo motors that were controlled using PIC16F877A microcontroller. Every motor behavior was determined by a button that corresponds to its interface in the microcontroller as programmed to generate the required pulse width modulated (PWM) signal for the corresponding servo motor. This

means that in the entire process of the robot activity a human being must be present to control it.

Siti (2011) designed a robotic arm controller using MATLAB. It was designed to be used to move either to the left and right or to lift an object. It used a PIC16F877A microcontroller circuit as the basic circuits to control the servo motor. MATLAB was used as a graphical user interface (GUI) for controlling the movement of this robot. A PC was used to send instructions through GUI to the microcontroller, an issue that makes it necessary for an operator to be present throughout its operation.

Dylan *et al*. (2004) designed a robotic arm using stepper motors that were computer controlled through the computer's parallel port. A separate circuit with 4-bit shift registers for each motor was used and the control software was written using Microsoft Visual C++ that creates a GUI and a DB-25 parallel port was used for communicating with the robot via the electronics.

Ashutosh (2013) proposed and designed a robotic arm controlled by natural human arm movements whose data was acquired through the use of accelerometers. This arm was based on ATmega32 and ATmega640 microcontrollers along a PC for signal processing. These were interfaced to each other to communicate serially. The robotic arm imitated the movements of a human arm.

Midhun *et al*. (2015) designed and developed a wireless control of pick and place robotic arm vehicle with a soft catching gripper that was designed to avoid extra pressure on the suspected object (like bomb) for safety reasons. It was controlled by an android application for remote operation. Bluetooth device was connected to the microcontroller to drive

motors via motor driver IC for necessary operation. Remote operation was achieved by any smart phone/tablet etc., with android operating system (OS), upon a GUI based touchscreen operation.

Robert (2011) designed and developed an application that provided remote control of an AL5A robotic arm connected to a low-powered, small central processing unit (CPU) embedded system, called an eBox. The program communicated via a client/server application involving TCP protocol. The eBox acted as a server and received servo transmission from the client to control the robotic arm while transmitting video back. It had a spatial map that allowed the user to plot specific destination coordinates in one set of mouse commands for the necessary rotation.

Babul *et al*. (2012) designed a wireless mobile robotic arm to pick and place objects controlled using wireless PS2 controller. It was based on Arduino Mega board interfaced with the wireless controller to the mobile robotic arm. The control of the actuator was by PWM signal. Depending on the controller's programmed instructions, the arm moves forward, reverse as well as turning right and left. Alan *et al* (2006) designed a toy excavator controlled remotely using GUI that displays measured and calculated information from the excavator and visual representation. It allow for operation when out of sight using two linear joysticks interfaced to PC. Commands were relayed by use of wireless transmission to hardware driving motors. Sensors were used to monitor track and arm joints angles.

2.2 PIC18F4550 based robotics

PIC18F4550 microcontroller has widely been used in intelligent control, automation and robotics. Dirman *et al.* (2012) designed an autonomous robot based on a PIC18F4550 as a basic development of an autonomous mobile robot for air duct or corridor cleaning. Fuzzy incremental controller (FIC) was used to control the robot through the use of ultrasonic sensors that were sensing the distance from the wall. The FIC analysis the signal generated by the sensors and uses it to vary the speed of the DC motors that in turn moves the robot appropriately.

Florentinus *et al.* (2016) designed a robot arm controlled by muscle tension based on electromyography and PIC18F4550. In their presentation, they suggested that the muscle tension generate a signal which can be transmitted to a computer using wire or via Bluetooth. They presented on the application of electromyography commonly known as guges muscle tension on the movement of robotic arm with aid of a PIC18F4550. Abdul *et al.* (2012) developed a gripper that could autonomously adjust to pick and place objects. The gripper used a photoelectric sensor as the source of its instructions for its operation. The sensor determines the size of the objects and give a feedback to the PIC18F4550 that hence give signal in term of voltage to motor driver which in turn converts this signal to PWM to DC motor. Generally, this gripper is able to identify shape and size of an object and adjust accordingly.

Eega *et al.* (2008) designed a prototype system of systems swarm robotics that is self-contained, powered and governing. This robot is run by five PIC18F4550 microcontrollers for master control and depth, thruster control, sonar, accelerometers and remote control and

an inter-board communications system. The master unit oversees the communications between the other units. Each thruster has a motor controller that controls the thruster's power using PWM which in turn controls the robot's depth, forward, backward movement and turning. This robotic swarm can be used in urban search and rescue (USAR) in collapsed building or man-made structures after a catastrophic event such as earthquake or bombing.

Savas *et al.* (2013) developed a microcontroller-based robotics and supervisory control and data acquisition (SCADA) experiments. In their paper they described how SCADA and robotics experiments can be conducted at a cheaper price in school of control and automation. They implemented this with a set-up that consisted of a fluid tank, a Cartesian robot with a three-axis robotic arm and serial, parallel, USB, and TCP/IP communication ports. PIC18F4550 board was used to implement the USB port for graphical programming using LabVIEW and the USB-based Cartesian robot through a general purpose step motor driver card for real-time controlling and monitoring for a three-axis system.

2.3 Haptic technology-based robotic arm

Reshamwala *et al.* (2015) presented a paper to review the use of haptic technology in controlling robot arm. They suggested that a combination of visual display and haptic technology can be used to train people for tasks requiring hand-eye coordination, such as robotic tele-surgery and space-ship maneuvering as well as for games in which you feel as well as see your interactions with image. Bhanu *et al*. (2012) designed and implemented a robotic arm based on haptic technology with servo motors with a torque of 11kg controlled using ATMEGA-328 microcontroller using Arduino programming. The input was from an

arm made of polycarbonate fitted with haptic gloves for virtual environment capable of haptic interaction and potentiometers with certain angle of rotation. The potentiometers detected the angle of rotation and signals were sent to the microcontroller accordingly.

Amey *et al.* (2013) designed and developed a robot that moved using wireless system by recognizing hand motion that was controlled by haptic technology for virtual environment and human-machine systems capable of haptic interaction. The project was divided into haptic gloves called transmitter and robot side called receiver. The transmitter consisted of haptic sensors, AVR 8535 microcontroller and ST3654-CC2500 ZigBee based trans/receiver module while the receiver consisted of an antenna, CC2500 RF transceiver, AVR 8535 microcontroller, H-bridge, gear motor and robotic arm.

Aishwarya *et al.* (2016) controlled a robotic arm through simple human gestures using haptic sensors that was done wirelessly. The implementation was placed on wheels to facilitate mobility. A transmitter with haptic sensors is fitted over the user's hands and the analog output sent to receiver through ZigBee to receiver where it is converted to digital using AVR controller and used to manipulate the arm by use of servo motors in the joints. A camera is mounted on this arrangement for controlling and monitoring purposes.

2.4 Human mind controlled robotic arm

Bin *et al.* (2016) developed a robotic arm that could reach for objects based in noninvasive electroencephalogram. This robotic arm is controlled using thoughts in the mind. Just by imagining moving an arm, individuals manipulate the motion of the robotic arm. The noninvasive technology, called electroencephalography (EEG) based brain-computer interface, records weak electrical activity of the subject's brain through a specialized, high-

tech EEG cap filled with 64 electrodes and converts the “thoughts” into action by advanced signal processing and machine learning. The brain-computer interface technology works due to the geography of the motor cortex, the area of the cerebrum that governs movement. When humans move, or think about a movement, neurons in the motor cortex produce tiny electric currents. Thinking about a different movement activates a new assortment of neurons. The brain-computer interface was achieved by trying to differentiate the neurons assortments by advance processing of these signals.

It is quite evidence that most of the robotic arms available need an individual to control it in its entire time of operation through buttons, levers, GUI interface in a PC, Bluetooth devices, touch screen, muscle tension, accelerometers or human minds. The other available robotic arms make use of sensors to make decisions in their area of operations. This project is aimed at designing and constructing a robotic arm that will not need influence of personnel throughout its operation. Through the software developed, the arm repetitively picks, lift and drop loads from one place to another once the power is put on.

CHAPTER THREE

THEORETICAL BACKGROUND

3.1 Introduction

This chapter gives a detailed view of PIC18F4550 microcontroller internal architecture and its operation. Other devices highlighted are the servo motor, Liquid Crystal Display (LCD), pull-up resistor and a relay.

3.2 PIC18F4550 microcontroller

PIC18F4550 shown in figure 3.1 is a 40pin PDIP 8-bit microcontroller of PIC18 family. The PIC18 family of microcontrollers with Nano Watt technology is based on Harvard architecture and is produced by Microchip Technology Inc.

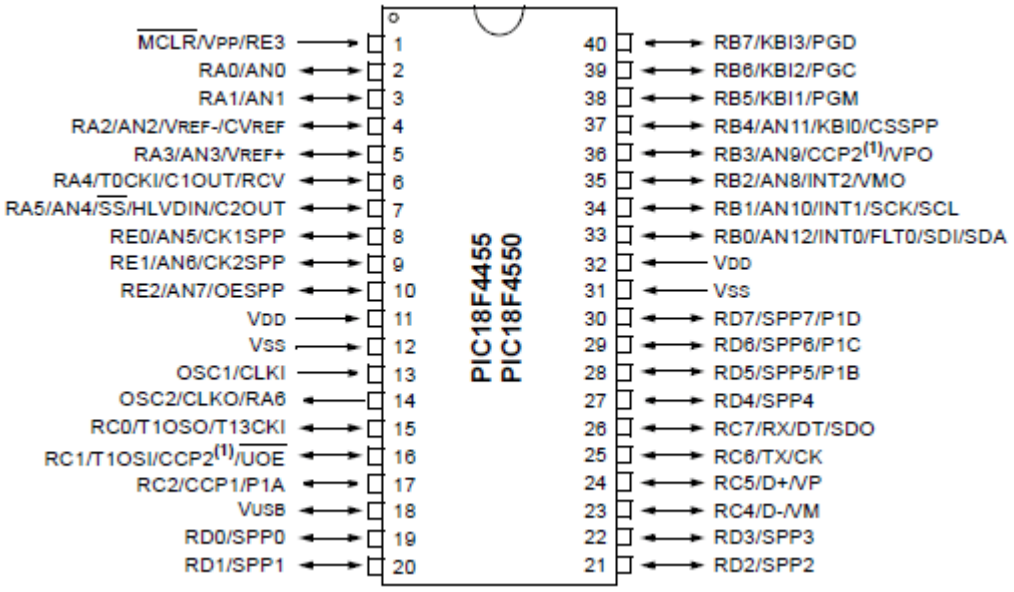

Figure 3.1: PIC18F4550 pin diagram showing the 40 pins (www.microchip.com, 15th January 2016).

PIC18F4550 is based on 16-bit instruction set architecture. It consists of 32kb flash memory, 2KB SRAM and 256 bytes EEPROM. It consist of 5 input/output (I/O) ports namely PORTA, PORTB, PORTC, PORTD and PORTE. It is a standard device which has its flash memory enhanced to accommodate a minimum of 4.2V and a maximum of 5.5V VDD. It can allow both internal and external clock sources on a frequency varying from 31 kHz to 48 MHz.

It has inbuilt modules which includes; Analogue-to-Digital (A/D), Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART), Master Synchronous Serial Port (MSSP) in both I2C and SPI modes, Streaming Parallel Port (SPP), Universal Serial Bus (USB) peripheral, Capture/Compare/PWM (CCP), Enhanced Capture/Compare/PWM (ECCP), timer and interrupts.

3.2.1 PIC18F4550 memory organization

PIC18F4550 has the following types of memory;

- Program
- Data RAM
- Data EEPROM

To allow concurrent access to both data and program memory spaces, the device has separate busses. Practically, to address and access the data EEPROM, a set of control registers must be used since it is regarded as a peripheral device.

3.2.1.1 Organization of program memory

Program memory in PIC18 microcontrollers is implemented in a 21-bit program counter. It can address a 2-Mbyte program memory space as shown in figure 3.2. ‘0s’ will be returned whenever a location between the upper boundary of the physically implemented memory and the 2-Mbyte address is accessed. PIC18F4550 has 32kbytes flash memory that can store up to 16384 single-word instructions. It has reset vector address at 0000h and the Interrupt vector addresses are at 0008h and 0018h.

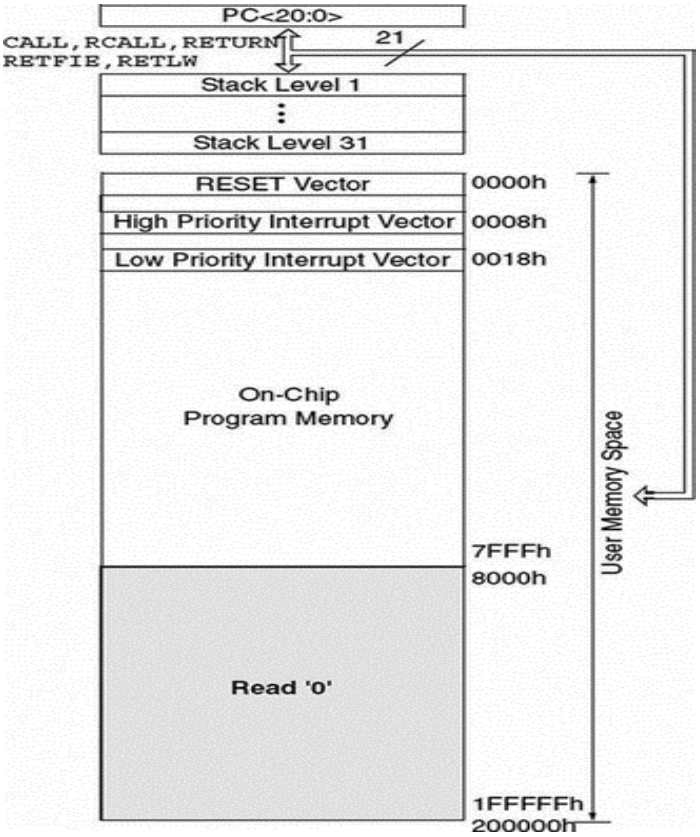

Figure 3.2: Schematic showing program memory map and stack levels of PIC18F4550 (www.microchip.com, 15th January 2016).

To fetch an instruction for execution the program counter (PC) that stores the address of the next instruction specifies the address. The PC is 21 bits wide and is contained in the PCL register that can be read from and written onto. The PCH register contains the PC<15:8> bits that cannot be read or written and the content of PCLATH and PCLATU which are transferred to the PC by any operation that writes PCL. The microcontroller addresses the program in bytes where instructions are stored as either two bytes or four bytes in the program memory. The storage of instruction words is such that the least significant byte is always stored in a program memory location with an even address (Lsb=0). The PC increments in steps of 2 and the LSB will always read '0' as it maintains alignment with instruction boundaries.

3.2.1.2 Organization of data memory

The PIC18F4550 implements its data memory as static RAM. Each register in data memory has a 12-bit address lines which results to 4096 bytes of data memory. The memory space partitions are in such a way that it has as many as 16 banks which contain 256 bytes each as shown in figure 3.3. The PIC18F4550 implement only 8 complete banks for a total of 2048 bytes. The data memory has both Special Function Registers (SFRs) and General Purpose Registers (GPRs) with SFRs being used for control and status of the controller and peripheral functions whereas GPRs are for data storage and scratchpad operations in user's application. A '0s' is returned whenever an unimplemented location is read.

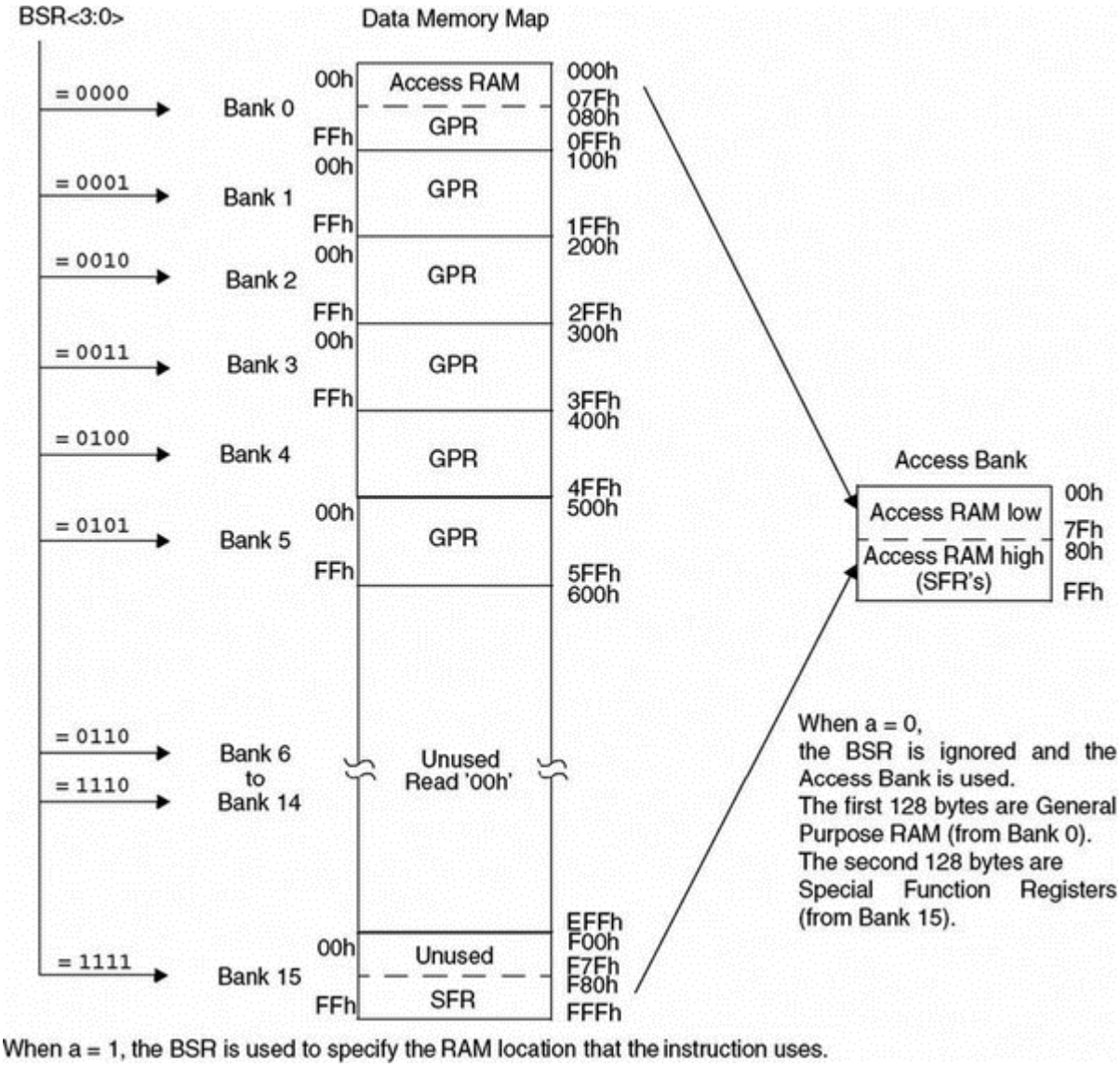

Figure 3.3: Schematic showing data memory map of PIC18F4550 (www.microchip.com, 15th January 2016).

In order to operate across all the banks in the entire data memory the microcontroller has the provision of direct, indirect or indexed addressing modes by the instruction set and its architecture. To ensure that commonly used registers are accessible in one complete cycle, PIC18 controllers make use of an access bank which occupies an equivalent of 256 byte

memory capacity. For the microcontroller to control the desired operation of the devices, the CPU and peripheral modules uses special function registers (SFRs). The SFRs extend from the top of data memory downwards to occupy the top segment of bank 15, from F60h to FFFh. The SFRs is primarily divided into ALU, Resets and Interrupts that are concerned with core running of the controller and those related to the peripheral functions as shown in figure 3.4. The status register has the arithmetic status of ALU and just like any other SFRs, is accessed as an operand in any instruction. If the status register is the destination for an instruction that affect the Zero flag bit, Digital Carry flag bit, Carry flag bit, Overflow flag bit or Negative flag bit, the results of the instruction are not written rather the status register is updated according to the instruction performed.

3.2.1.3 Data EEPROM memory

The data EEPROM stores program data on long term basis. It is indirectly addressed through the SFRs and enjoys the direct mapping to neither of the register file nor the program memory space. It uses EECON1, EECON2, EEDATA and EEADR in reading and writing to it as well as to program memory. One of the qualities enjoyed while using EEPROM data memory is the fact that it can endure constant erasing and writing process while at the same time keep its storage capability intact. When an instruction to write onto a certain memory location is received, the previous contents are lost. The time taken to write to a memory location is regulated by a timer within the chip that varies from one chip to another as well as by the changes in the voltage and temperature. In order to read data from a given memory location, the user is required to write its address to the EEADR register,

clear the EEPGD control bit (EECON1<7>) and then set control bit RD(EECON1<0>). This data will be available on the next instruction cycle. To write on to an EEPROM data location, the user is required to write its address to the EEADR register and the data is written to the EEDATA register.

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(1)	FBFh	CCPR1H	F9Fh	IPR1	F7Fh	UEP15
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	CCPR1L	F9Eh	PIR1	F7Eh	UEP14
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	CCP1CON	F9Dh	PIE1	F7Dh	UEP13
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR2H	F9Ch	—(2)	F7Ch	UEP12
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR2L	F9Bh	OSCTUNE	F7Bh	UEP11
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—(2)	F7Ah	UEP10
FF9h	PCL	FD9h	FSR2L	FB9h	—(2)	F99h	—(2)	F79h	UEP9
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	—(2)	F78h	UEP8
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	ECCP1DEL	F97h	—(2)	F77h	UEP7
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS	F96h	TRISE(3)	F76h	UEP6
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD(3)	F75h	UEP5
FF4h	PRODH	FD4h	—(2)	FB4h	CMCON	F94h	TRISC	F74h	UEP4
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	UEP3
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA	F72h	UEP2
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—(2)	F71h	UEP1
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	—(2)	F70h	UEP0
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—(2)	F6Fh	UCFG
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—(2)	F6Eh	UADDR
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(3)	F6Dh	UCON
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(3)	F6Ch	USTAT
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC	F6Bh	UEIE
FEAh	FSR0H	FCAh	T2CON	FAAh	—(2)	F8Ah	LATB	F6Ah	UEIR
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA	F69h	UIE
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—(2)	F68h	UIR
FE7h	INDF1(1)	FC7h	SSPSTAT	FA7h	EECON2(1)	F87h	—(2)	F67h	UFRMH
FE6h	POSTINC1(1)	FC6h	SSPCON1	FA6h	EECON1	F86h	—(2)	F66h	UFRML
FE5h	POSTDEC1(1)	FC5h	SSPCON2	FA5h	—(2)	F85h	—(2)	F65h	SPPCON(3)
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	—(2)	F84h	PORTE	F64h	SPPEPS(3)
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	—(2)	F83h	PORTD(3)	F63h	SPPCFG(3)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	SPPDATA(3)
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	—(2)
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA	F60h	—(2)

Note 1: Not a physical register.
2: Unimplemented registers are read as '0'.
3: These registers are implemented only on 40/44-pin devices.

Figure 3.4: Schematic showing special function register map of PIC18F4550 (www.microchip.com, 15th January 2016)

3.2.2 Reset

PIC18F4550 has various kinds of Reset as shown in figure 3.5. The master clear reset (MCLR) is among them. RE3 pin of the controller that is also multiplexed as MCLR pin gives an avenue for the device reset to be triggered externally once the pin is held low.

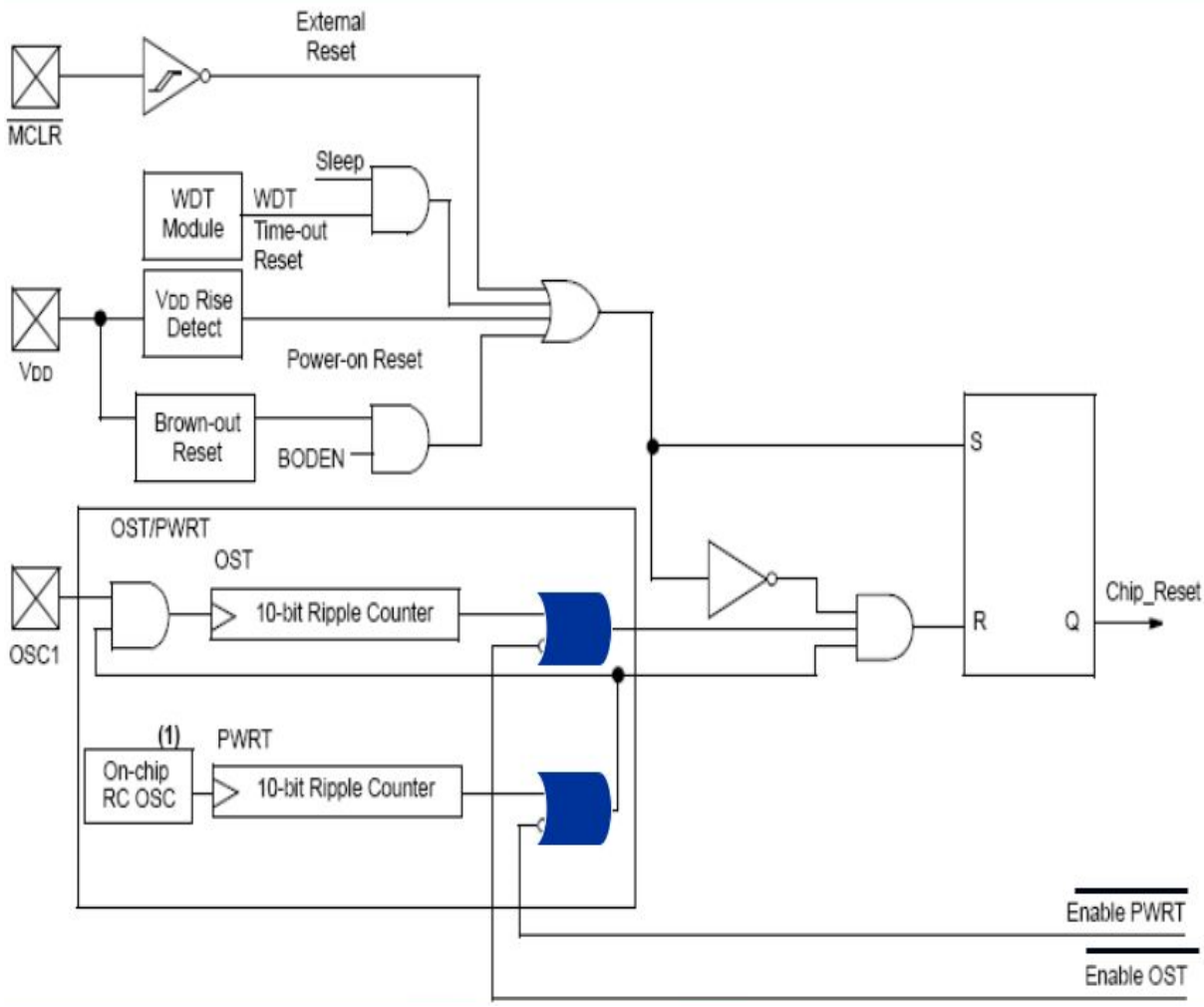

Figure 3.5: Schematic showing basic block diagram of on-chip reset circuit of PIC18F4550 (www.microchip.com, 15th January 2016).

In order to detect and ignore small pulses in the reset path the MCLR has a noise filter.

This pin can be disabled through the software where the programmer disables the MCLRE configuration bit to allow the pin become a digital input throughout the operation. The power-on reset (POR) is also used in which whenever power is switched on, the VDD rises to achieve its maximum voltage depending on the manufacturer's specification. In the course of rising, there is a certain threshold level it reaches when a Reset pulse is produced on the chip. This pulse makes the controller to routinely start in its set states once the VDD is satisfactory enough to run. To employ this method of reset, one ties MCLR pin to VDD through a pull-up resistor (4.7kΩ). This way any POR delay from external RC components is eliminated. When the device is set to start its normal operation, the device operating parameters (voltage, frequency, temperature etc.) should be met to make it work. The device remains in Reset mode if these conditions are not met.

3.2.3 Interrupts

An interrupt is a kind of hardware or software notification conversed to the microcontroller of which when received, the microcontroller momentarily halt or pause its activities and responds to it depending on the priority accorded. Whenever an interrupt is received, the microcontroller first finalizes the activity at hand and then starts the execution of an Interrupt Service Routine (ISR). ISR is an algorithm that instructs the microcontroller on what to do when the interrupt occurs. After the controller completes the execution of ISR, it was scheduled to attend before the interrupt was received.

The PIC18F4550 has numerous sources of interrupts which are assigned either a high or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h as shown in figure 3.6. Each interrupt source needs a Flag bit that indicate the occurrence of an interrupt, Enable bit to allow the program to the execute the interrupt when the Flag bit is set and Priority bit to select either high or low priority.

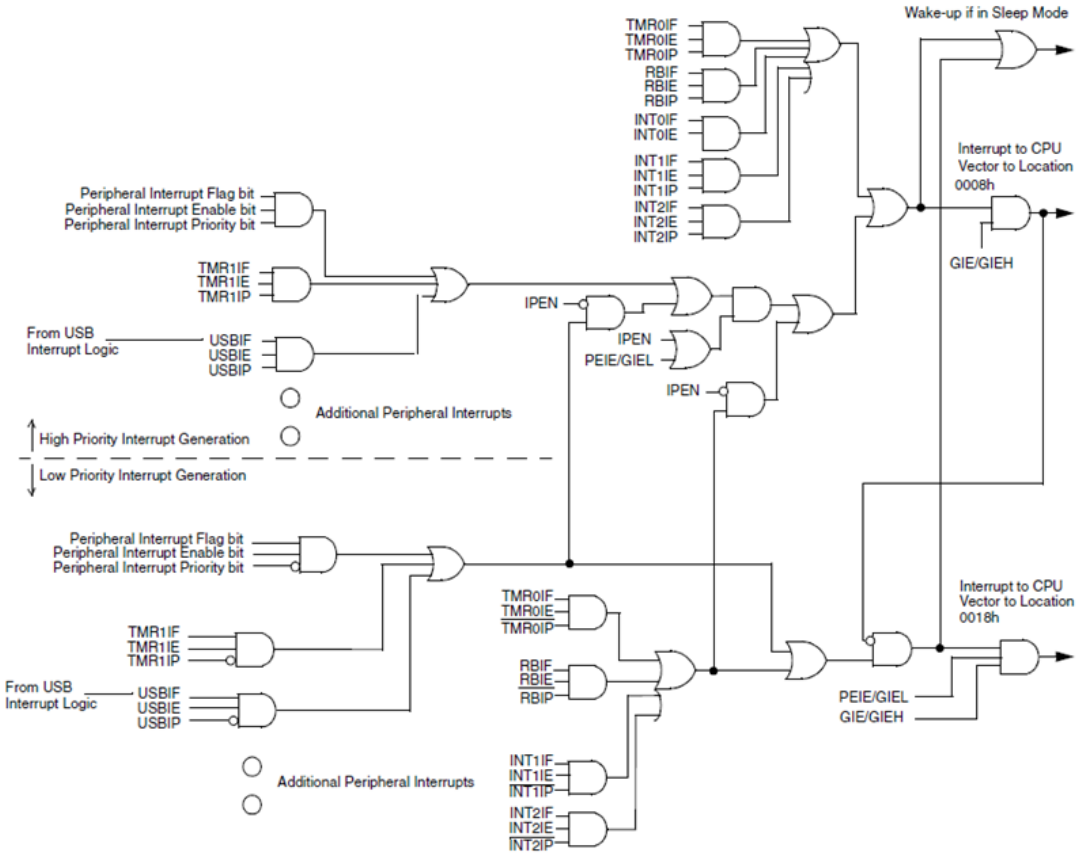

Figure 3.6: Schematic showing interrupt logic diagram of PIC18F4550 (www.microchip.com, 15th January 2016).

3.2.4 Input/Output ports

Depending on the features enabled, PIC18F4550 has five Input/Output (I/O) ports that are available for use. Most of the pins in the microcontroller are internally structured in such a way that they not only act as general I/O ports but carry out some other functions as directed by the peripheral features on the controller. To use them as I/O ports, the programmer has to specify the data direction by use of the TRIS register, to read the voltage levels on the pins with the help of the PORT register as well as write to the pins using the LAT register.

3.2.5 Timer modules

There are four timers in PIC18F4550;

- Timer0 - in both 8-bit and 16-bit modes
- Timer1 - 16-bit counter
- Timer2 - 8-bit counter
- Timer3 - 16-bit counter

The internal speed of the PIC18F4550 is 48 MHZ. One instruction cycle takes four clock cycles i.e. the timer can be chosen to increment at each internal instruction cycle (Fosc/4). The controller takes the internal/external clock timing and divides it by four. The timer has programmable prescaler used for decreasing the timer frequency again. This is a binary ripple-counter that is put before the actual timer. The ripple counter

simply counts the clock source and provides outputs of divide by 2,4,8,16,32 and so on. One selects the divide ratio as the prescaler value and the timer in question sees a lower frequency of the input. It slows down for timer incrementing instead of incrementing each clock cycle it can be adjusted to every 2^{nd}, 4^{th}, 8^{th} etc. The postscaler on the other hand slows the rate of the interrupt generation or watchdog time reset from a counter by dividing it down. It is used to extend the time period that a timer can generate. If one set postscaler to four for example you will get four times the time period the timer can generate.

3.2.6 Enhanced Universal Synchronous Asynchronous Receive Transmitter

The Enhanced Universal Synchronous Asynchronous Receive Transmitter **(**EUSART) module, that is also referred to as a serial communication interface (SCI) is a serial I/O module. It is able to connect to the peripheral devices available such as Cathode Ray Tube (CRT) terminal and PC if configured as a full-duplex asynchronous. The pins of the EUSART are implemented on PORTC as RC6 for transmission and RC7 for receiving.

3.2.7 Oscillator configuration

An oscillator is a standalone clock generating circuit as shown in figure 3.7. There are two built-in oscillators namely internal and external oscillator. To install an external oscillator on to the PIC18F4550, it is interfaced to the OSC1 and OSC2 pins. The word

external comes as a result of relying on an external circuitry to provide the clock signal and frequency stabilization such as a standalone oscillator, quartz crystal, ceramic resonator or resistor-capacitor circuit. There are two different clock signals that are generated by the internal oscillator that are available for use as the microcontroller's clock source. The internal oscillator may eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins if the USB peripheral is not used. Programmers use two configuration registers noted as CONFIG1L for lower bits and CONFIG1H for higher bits as well as two control registers noted as OSCON and OSCTUNE.

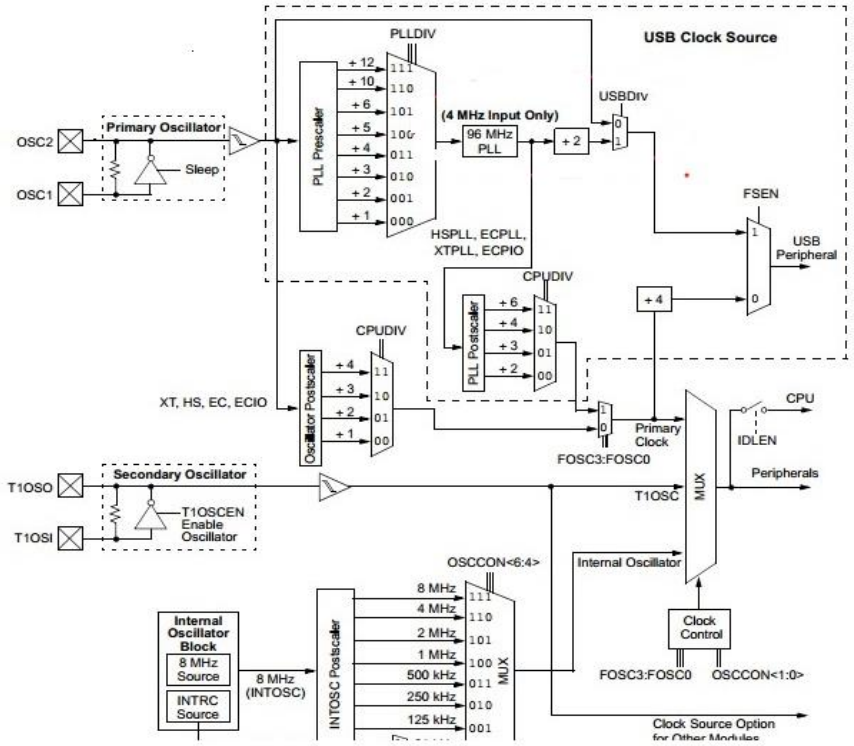

Figure 3.7: Schematic showing oscillator circuit of PIC18F4550 (www.microchip.com, 15th January 2016).

In order to select the mode of operation by the oscillator and the options available for the USB prescaler/postscaler, the programmer makes it possible by the use of configuration registers. The programmer sets the configuration bits for the device during the programming process which remains so unless reprogrammed at a later date. The active mode of the device which primarily controls the clock switching in power-managed modes is selected using the OSCON register while the selection of the frequency clock source that drives special features as well as trimming the INTRC frequency source is done by the OSCTUNE register.

3.3 Servo motor

A servo motor is a special DC motor which has an output shaft (servo arm) interfaced with the motor through a series of gears and a feedback electronic circuit in charge of controlling the position of the shaft as shown in figure 3.8.

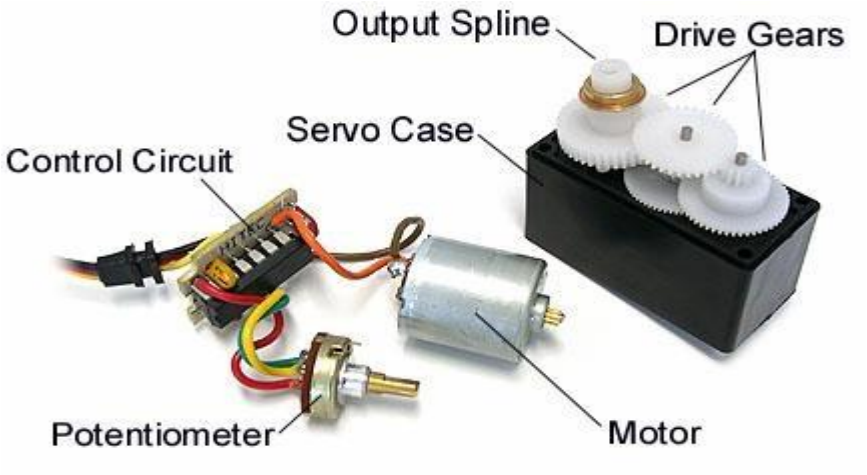

Figure 3.8: Schematic showing the internal parts of servo motor (www.embedded-lab.com, 10th December 2015).

The feedback mechanism of the control circuit in the servo motor is in such a way that it has a transducer that provides information on the instantaneous positioning of the output shaft. Attaching a potentiometer to the output shaft or somewhere in the gear train as shown in figure 3.9 has made this possible. The work of the control electronics is to compare the feedback signal (that has the current position of the shaft) from the potentiometer with the control input signal (that has the information of the desired position of the shaft), and in case of any difference detected between the actual and desired values (which is known as an error signal), it is amplified and used to drive the DC motor in a direction needed to decrease or eradicate the error.

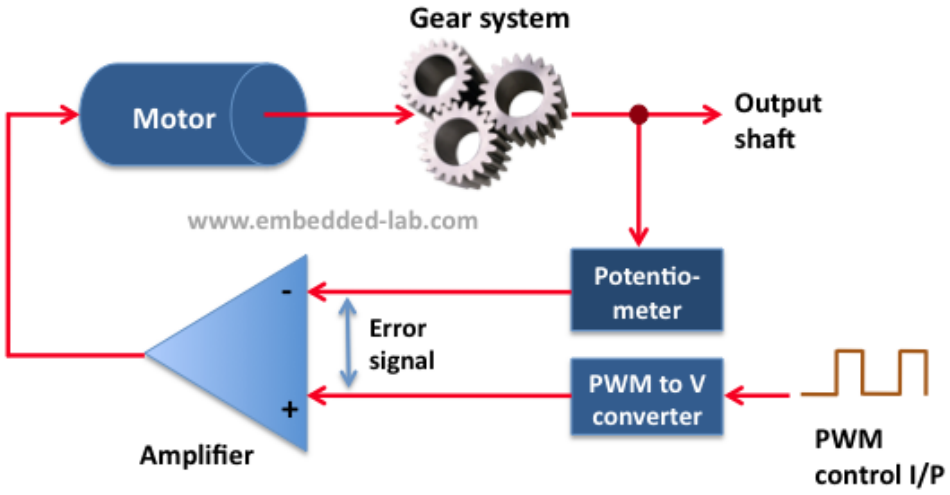

Figure 3.9: Schematic showing the principle operation of servo motor (www.embedded-lab.com, 10th December 2015).

The error is zero when the output shaft gets to the desired position. To control the positioning of a servo motor, a Pulse Width Modulated (PWM) signal that is generally of frequency 50 Hz is fed on to its control input wire. This pulse should repeatedly occur every 20 ms. The width of the pulse or rather the time the pulse is HIGH

compared to when its LOW, determines the angular position of the output shaft. The 'on time', also described in terms of the percentage duty cycle, is the percentage of time a digital signal is on over an interval or period of time as shown on figure 3.10.

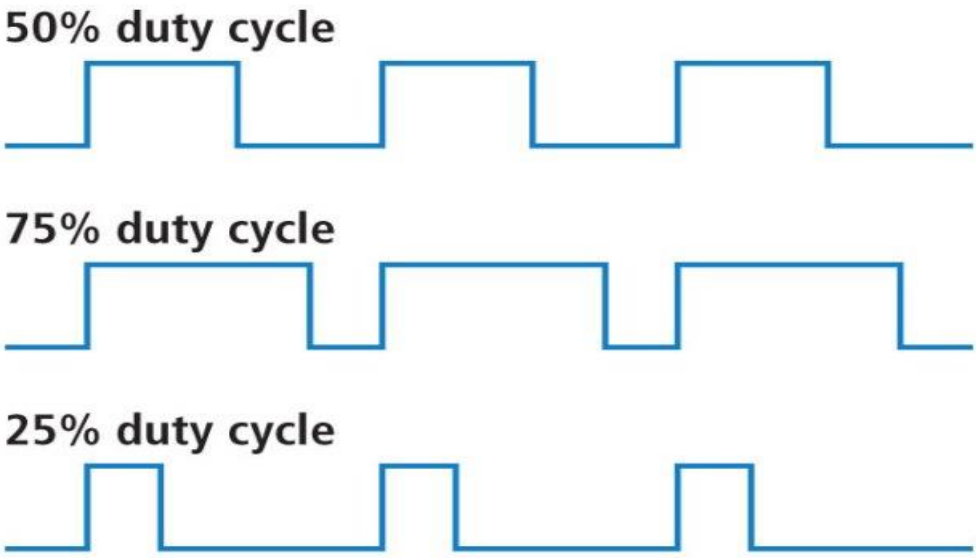

Figure 3.10: Schematic showing the PWM signal to control servo motor (www.embedded-lab.com, 10th December 2015).

The PWM signal provided at the control input wire of the servo is converted appropriately by the electronic circuit to a proportional output voltage which is apparently compared with the feedback voltage from the potentiometer. Most of the analog servo motors operate comfortably with a pulse width ranging from 1.0 to 2.0 ms. Though most of the analog servo motors rotate through 180°, there are those that can rotate through a full 360° or more. Servos are widely used as the moving joints in robotic arms, applied in radio controlled (RC) toys like in RC cars for their steering mechanisms and RC boats to control the rudder due to their precise angular positioning.

A hobby servo motor allows for three connections, two of which provide it with power and the third one with the control signals. Currently servo motors have been controlled using microcontrollers by sending PWM signals on the control wire. The length of the pulse will determine how far the motor turns. The pulses required depend on the specifications of the manufacturer of the servo.

3.4 Liquid Crystal Display (LCD)

16 X 2 character LCD shown in figure 3.11 is one of the simplest display modules used in electronics and control science projects and products. 16 characters are displayed in its 2 rows with each of it in a 5x8 or 5x10-dot matrix. These display modules uses controllers that are compliant with HD44780. It has two inputs to give power V_{CC} and GND. Voltage at V_{EE} determines the contrast of the display. A 10 kΩ potentiometer whose fixed ends are connected to V_{CC}, GND and variable end is connected to V_{EE} can be used to adjust contrast. Data and command information are the two possible instructions a microcontroller sends to operate this display module. Data represent the ASCII value (8bits) of the character to be displayed and commands determines the other operations such as initialization, clearing the screen, setting cursor position to be displayed. Data and commands are sent through the same data lines, which are multiplexed using the Register Select (RS) input. When it is HIGH, LCD takes it as data to be displayed and when it is LOW, LCD takes as a command. Data strobe is given using Enable (E) input. When the E is HIGH, LCD takes it as a valid data or command. The input signal Read or Write (R/W) determines whether data is written to or read from the LCD.

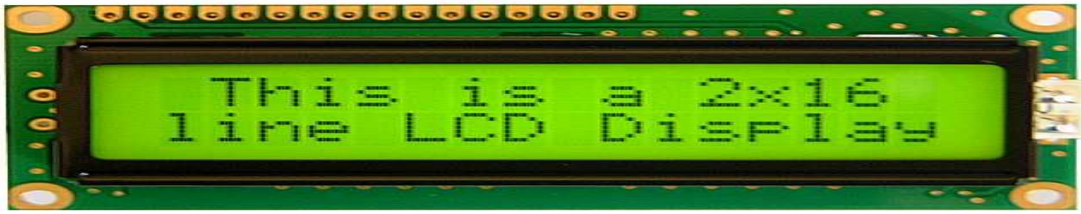

Figure 3.11: Schematic showing a 2x16 LCD (www.embedded-lab.com, 10th December 2015).

The interface between this LCD and microcontroller can be 8 bit or 4 bit and the difference between them is in how the data or commands are send to LCD. In the 8 bit mode, 8-bit data and commands are sent through the data lines D0-D7 and data strobe is given through E input of the LCD. The 4 bit mode uses only 4 data lines. In this, 8bit data and commands are split into two parts (4 bits each) and are sent sequentially through data lines D4-D7 with its own data strobe through E input. The idea of 4 bit communication is introduced to save pins of a microcontroller.

3.5 Pull-up resistor

When a program reads the state of a pin configured as an input pin that is not connected to anything, it is difficult to tell if the pin is HIGH or LOW. This state of an input pin is referred to as floating. To solve this unknown state a pull-up or pull-down resistor will ensure that the pin is in either a HIGH or LOW state while also using a low amount of current. Both pull-up and pull-down resistor operate using the same concepts except

that the pull-up resistor is connected to the V_{CC} and the pull-down resistor is connected to GND. Pull-up resistors are often used with buttons and switches. With a pull-up resistor shown in figure 3.12, the input pin will read a HIGH state when the button is not pressed. This means that a small amount of current is flowing between V_{CC} and the input pin, thus it reads close to V_{CC}. When the button is pressed, it connects the input pin directly to GND. The current flows through the resistor to GND thus the input pin reads a LOW state. If the resistor were not there, the button would connect V_{CC} to GND causing a short circuit which is undesirable.

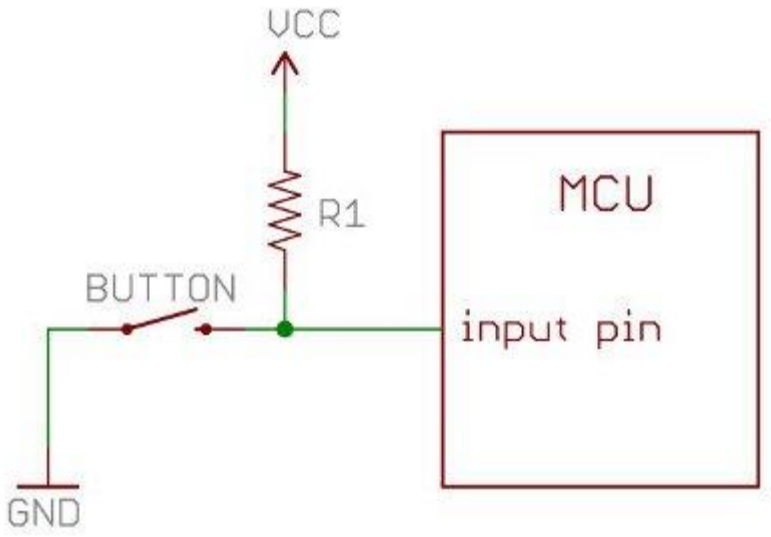

Figure 3.12: Schematic showing a pull-up resistor circuit connected to a push button (www.sparkfun.com, 20th February 2016).

The value of a pull-up resistor needs to be chosen to satisfy two conditions, namely;

1. When the button is pressed, the input pin is pulled LOW. The value of resistor R1 controls how much current flows from V_{CC} through the button and then to GND.

2. When the button is not pressed the input pin is pulled HIGH. The value of resistor R1 controls the voltage on the input pin.

From the Ohm's law:

$V = IR$ ……. (I)

Referring to the Figure 3.12, Ohm's law now is;

$V_{CC} = I_{R1}R1$ ……. (II)

Rearranging the equation (II) to make R1 the subject of the formula;

$R1 = V_{CC}/I_{R1}$…… (III)

3.6 PICkit™3 programmer/debugger

The PICkit™3 programmer/debugger shown in figure 3.13 is a simple, low cost in-circuit debugger that is controlled by a PC running MikroC Pro IDE software on a windows platform. It is a debugger system used for hardware and software development of Microchip PIC microcontrollers and dsPIC Digital Signal Controllers (DSCs) that are based on In-Circuit Serial Programming™ (ICSP™) and enhanced in-circuit serial programming 2-wire serial interfaces.

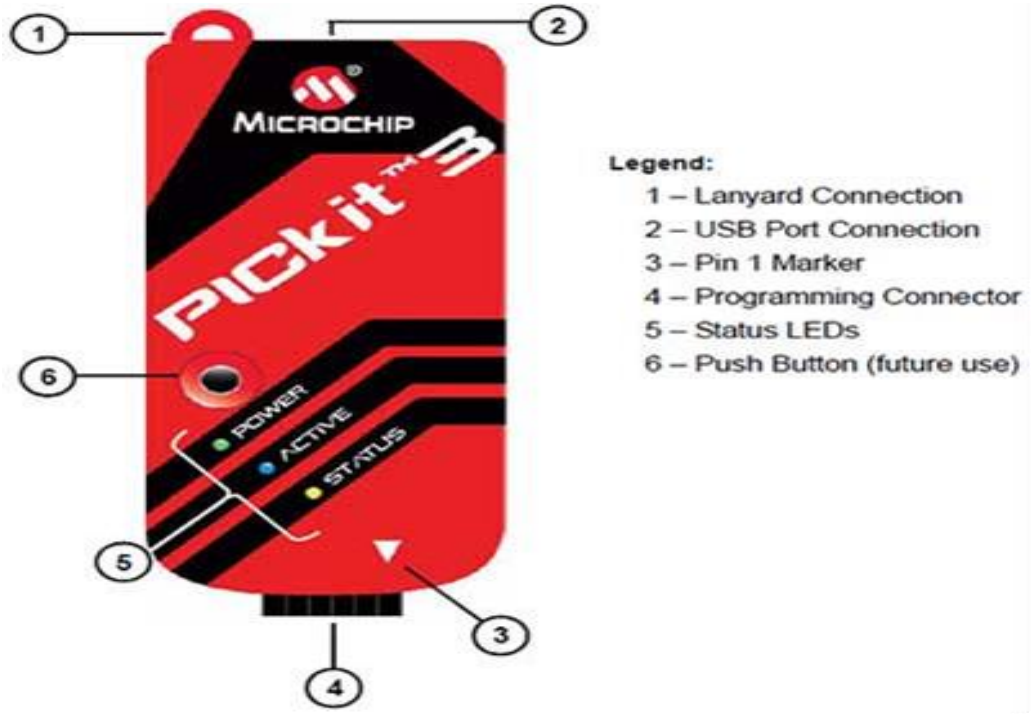

Figure 3.13: Schematic showing PICkit™3 (www.microchip.com, 15th January 2016).

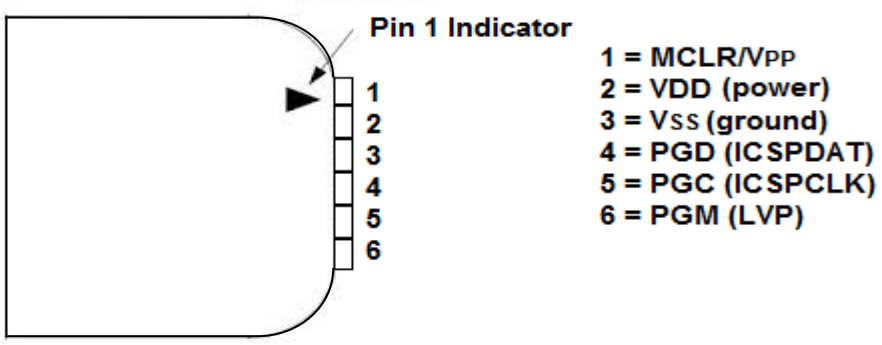

Figure 3.14: Schematic showing the programming connector of PICkit™3 programmer (www.microchip.com, 15th January 2016).

The programming connector shown in figure 3.14 is a 6-pin header on the PICkit™3 programmer that connects to the target device.

3.7 LM2596 Adjustable step down regulator

LM2596 regulator shown in figure 3.15 is a monolithic integrated circuit ideally suited for easy and convenient design of a step-down switching regulator (buck converter). It is capable of driving a 3.0A load with excellent line and load regulation. This device is available in adjustable output version and it is internally compensated to minimize the number of external components to simplify the power supply design. It operates at a switching frequency of 150 kHz thus allowing smaller sized filter components than what would be needed with lower frequency switching regulator. Since LM2596 converter is a switch-mode power supply, its efficiency is significantly higher in comparison with popular 3-terminal linear regulator, especially with higher input voltages.

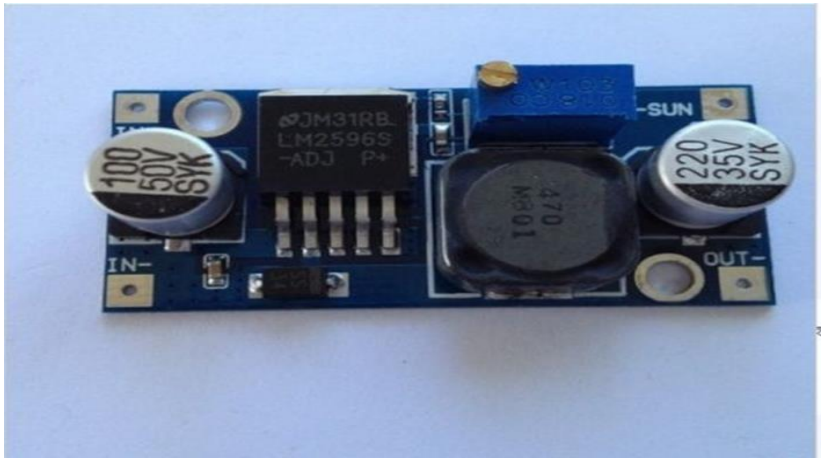

Figure 3.15: Picture of the LM2596 adjustable voltage regulator (www.microchip.com accessed on 13th July 2016).

CHAPTER FOUR

RESEARCH METHODOLOGY

4.1 Introduction

This chapter shows step by step activities that were carried out in the quest to design and develop the 5DOF robotic arm. PIC18F4550 microcontroller was used as the main control unit under the instructions given through the software developed in C programming language. The circuit was developed in three stages which were accompanied by the codes to control them at each stage. The Proteus-ISIS7 professional software was installed in the PC for schematic circuit design and simulation of the codes developed. The procedures outlined below are categorized into the hardware and software designs.

4.2 Hardware design

The hardware designing took four major aspects that included the system design as shown in figure 4.1, circuit designs and implementation, the designing of the arm as shown in figure 4.8 and construction of the robotic arm. All the circuits were designed using the Proteus-ISIS7 professional software for schematic circuit design and software simulation. Actual interface was done to come up with the circuit for the control of the servo motors.

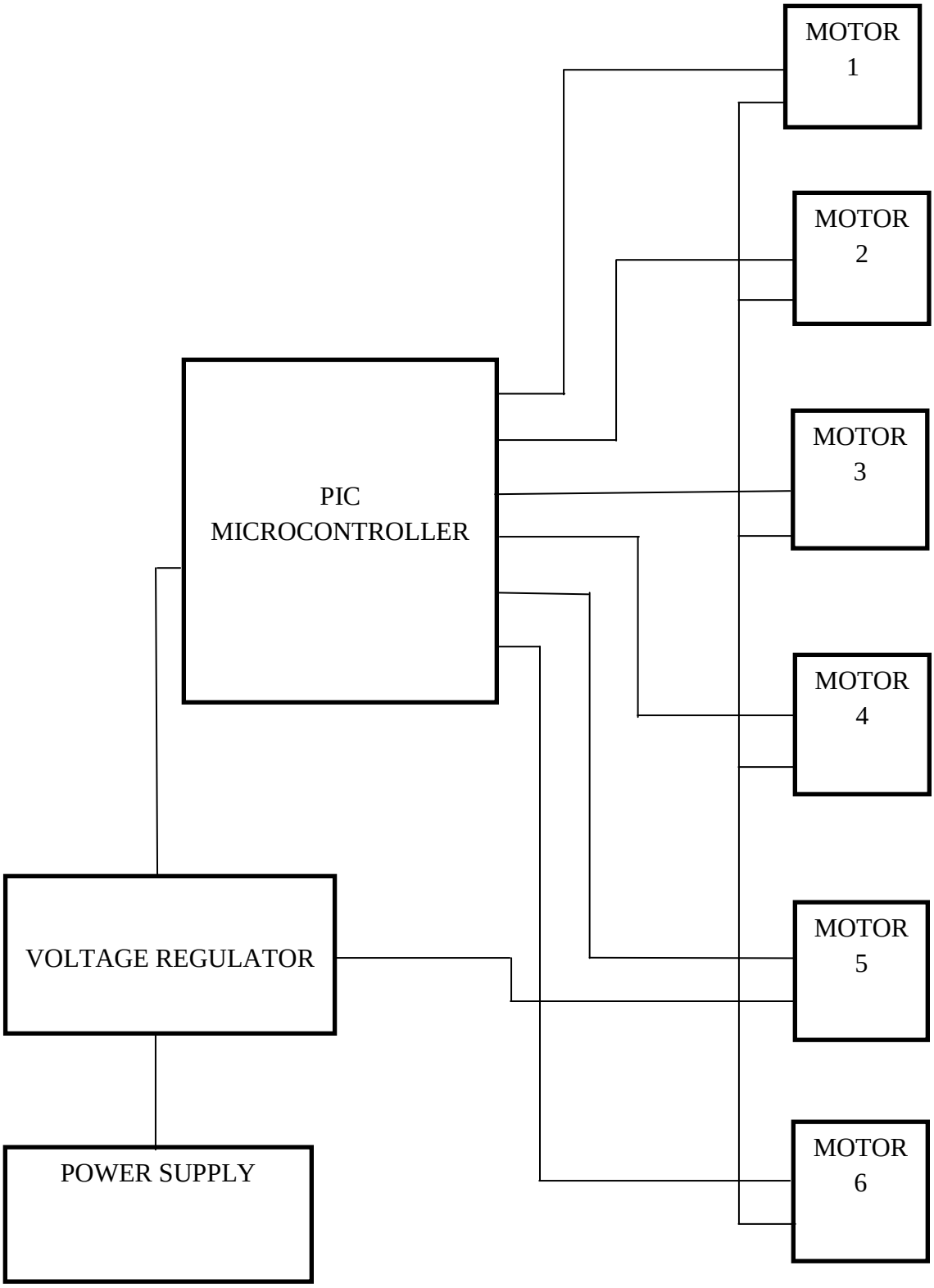

Figure 4.1: Block diagram of the hardware

4.2.1 Power supply

The servo motor needed 6.5V voltage and 3.0A minimum DC individually while the microcontroller needed 5.0V voltage and 2.0A minimum DC. The project's power supply was designed using a 240V to 12V step down transformer, bridge rectifier and two LM2596 adjustable voltage regulators as shown on figure 4.2. The stepdown transformer was connected to the main power supply through s 3-pin plug socket and its 12V output to the 12V AC to DC bridge rectifier. The 12V DC from the output of the bridge rectifier was fed to two separate LM2596 adjustable voltage regulators, one to supply 6.5V to the servo motors and the other to supply 5.0V to the microcontroller. The two GND from the LM2596 regulators were connected together.

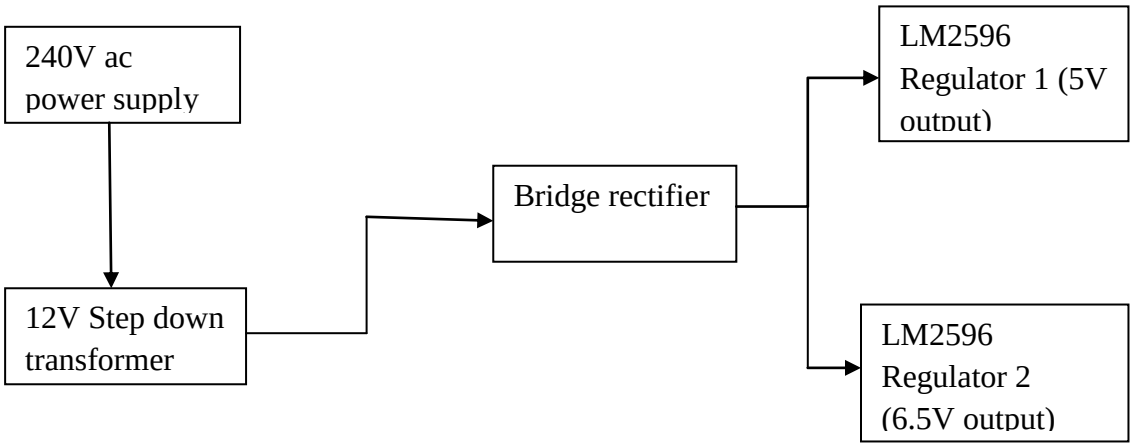

Figure 4.2: Block diagram of the power supply

In order to minimize interference from the motors while the microcontroller initializes, a relay input was interfaced to RA2 pin of the microcontroller. Its output was connected to the motor's power supply in order to switch on the supply through the software by the microcontroller after the initialization process was complete.

4.2.2 Control circuits

In the control of one servo motor, a circuit shown in figure 4.3 was designed using PIC18F4550 microcontroller, white breadboard to allow correction, jumpers, wires, 8MHz crystal, two 10kΩ resistor, two 22pF capacitors and a digital push button.

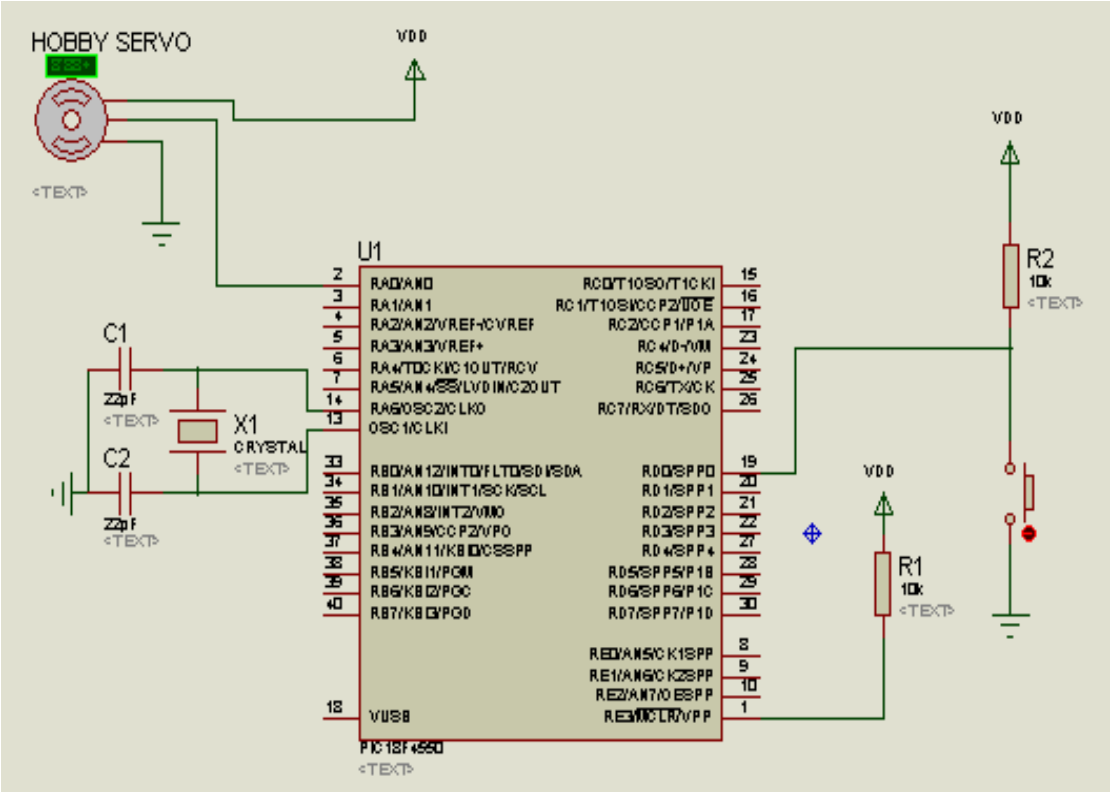

Figure 4.3: Schematic showing the circuit diagram to control one servo motor.

The breadboard was used as the working space. The MCLR pin of the microcontroller was connected to +5.0V through a 10kΩ resistor. The two V_{DD} pins were connected to +5.0V and the GND pins to 0V. To provide a more stable and accurate clock signal, the 8MHz crystal was connected to OSC1 and OSC2 pins and grounded through two parallel 22pF capacitors. The PICkit™3 programmer was connected to the PC through a

USB and to the PIC microcontroller with its pins as shown on the table 4.1. The input digital push button was connected to pin RD0 of the microcontroller and to +5.0V through a 10kΩ pull-up resistor. The control wire of the servo motor was connected to pin RA0 of the microcontroller for PWM signal, power wire to +6.5V and the GND wire to GND. The circuit to control different motors was developed from the circuit to control one motor as shown in figure 4.4.

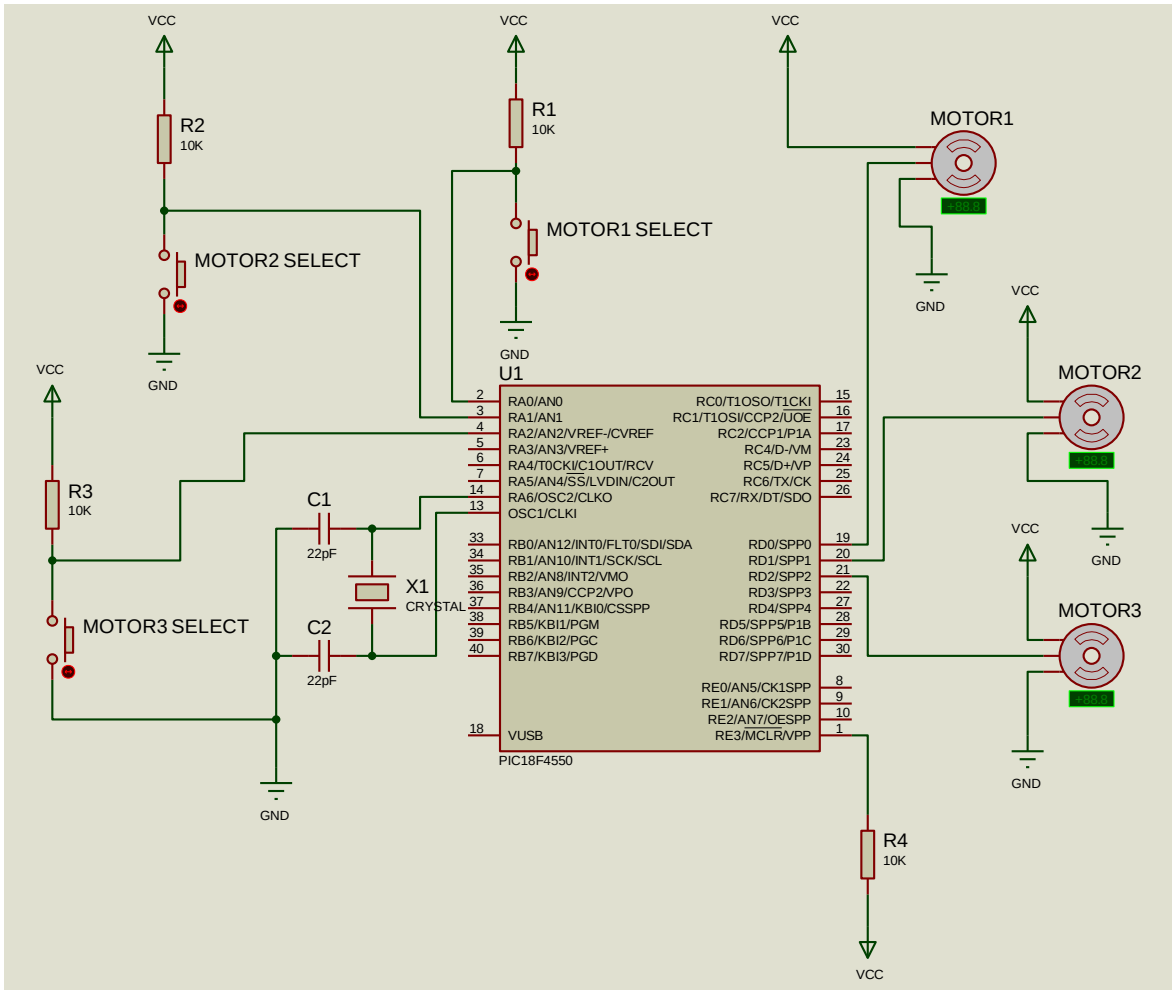

Figure 4.4: Schematic showing the circuit diagram to control different servo motors. The clock signal circuit and MCLR reset pin remained the same. Three pins; RD0, RD1 and RD2 were connected to control lines of servomotors 1, 2 and 3,

respectively. The servo power lines were connected to +6.5V power supply and their ground lines to GND. To select which motor to rotate, push buttons were connected to pins RA0, RA1 and RA2 through 10kΩ resistors for motors 1, 2 and 3, respectively.

Table 4.1: The PICkit™3 programmer interfaced with the microcontroller.

PIN	CONNECTION
1	MCLR
2	V_{DD}
3	GND
4	RB7
5	RB6

A 2 x 16 character LCD was used. It was connected as shown on the table 4.2. Its software was designed to display text on its screen whenever needed to.

Table 4.2: The 2 x 16 LCD interface with the PIC18F4550 microcontroller

LCD PIN	CONNECTION
V_{SS}	GND
V_{DD}	+5.0V
V_{EE}/V_o	POTENTIOMETER
RS	RB4
RW	GND
E	RB5
D0, D1, D2, D3	_
D4	RA0
D5	RA1
D6	RB2
D7	RB3
K	GND
A	V_{CC}

The circuit was designed using the Proteus software as shown in the figure 4.5 and the circuit implemented.

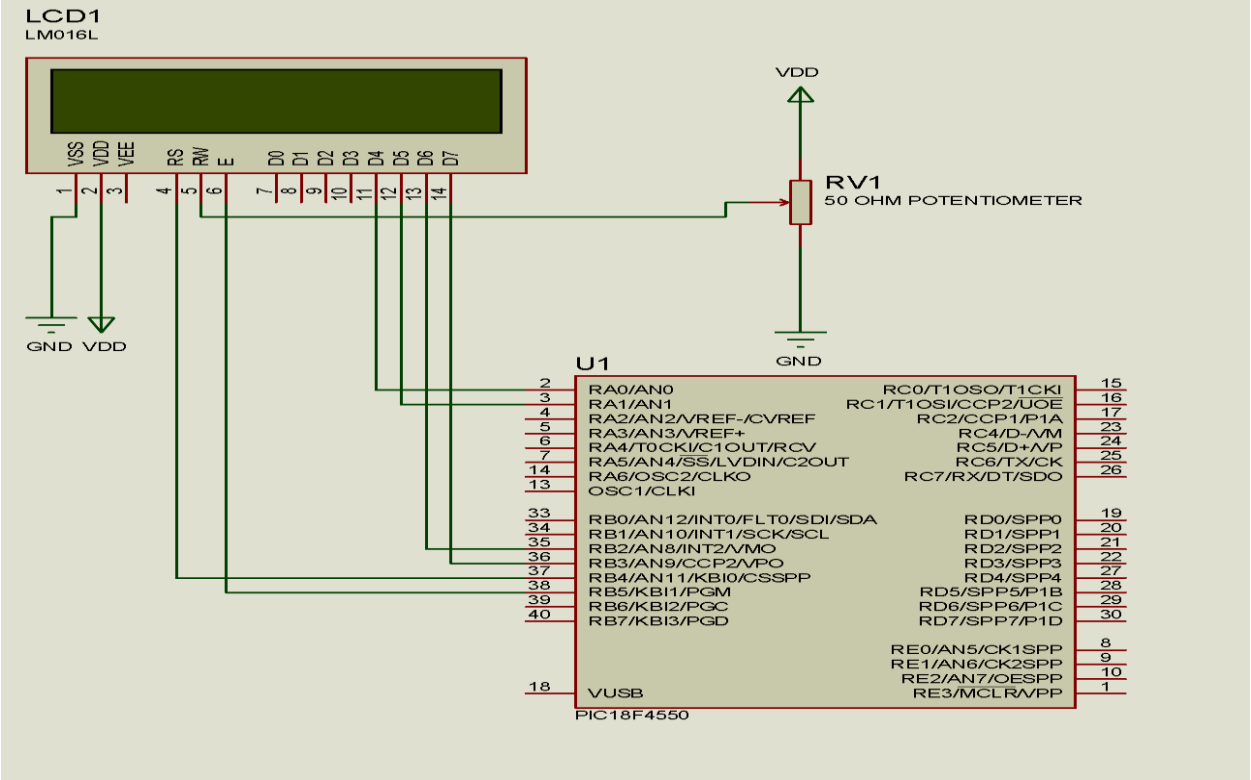

Figure 4.5: Schematic showing the circuit diagram to display onto LCD

Having achieved the control of different motors that were intended to make the various movements of the robotic arm, the final circuit for the control of robotic arm was developed based on the previous circuits as shown in the figure 4.6. The circuit remained as it was for the control of different servomotors apart from the elimination of the push button and two more servomotors were included. A Light Emitting Diode (LED) was connected to pin RA3 through a 290Ω resistor to the ground. This LED was used to debug the software by being toggled i.e. being ON and OFF depending on the number of delays in a specified function. The LED connection was in current sourcing

mode i.e. one leg of the LED was connected to the output port of the microcontroller and the other leg to GND through the current limiting resistor.

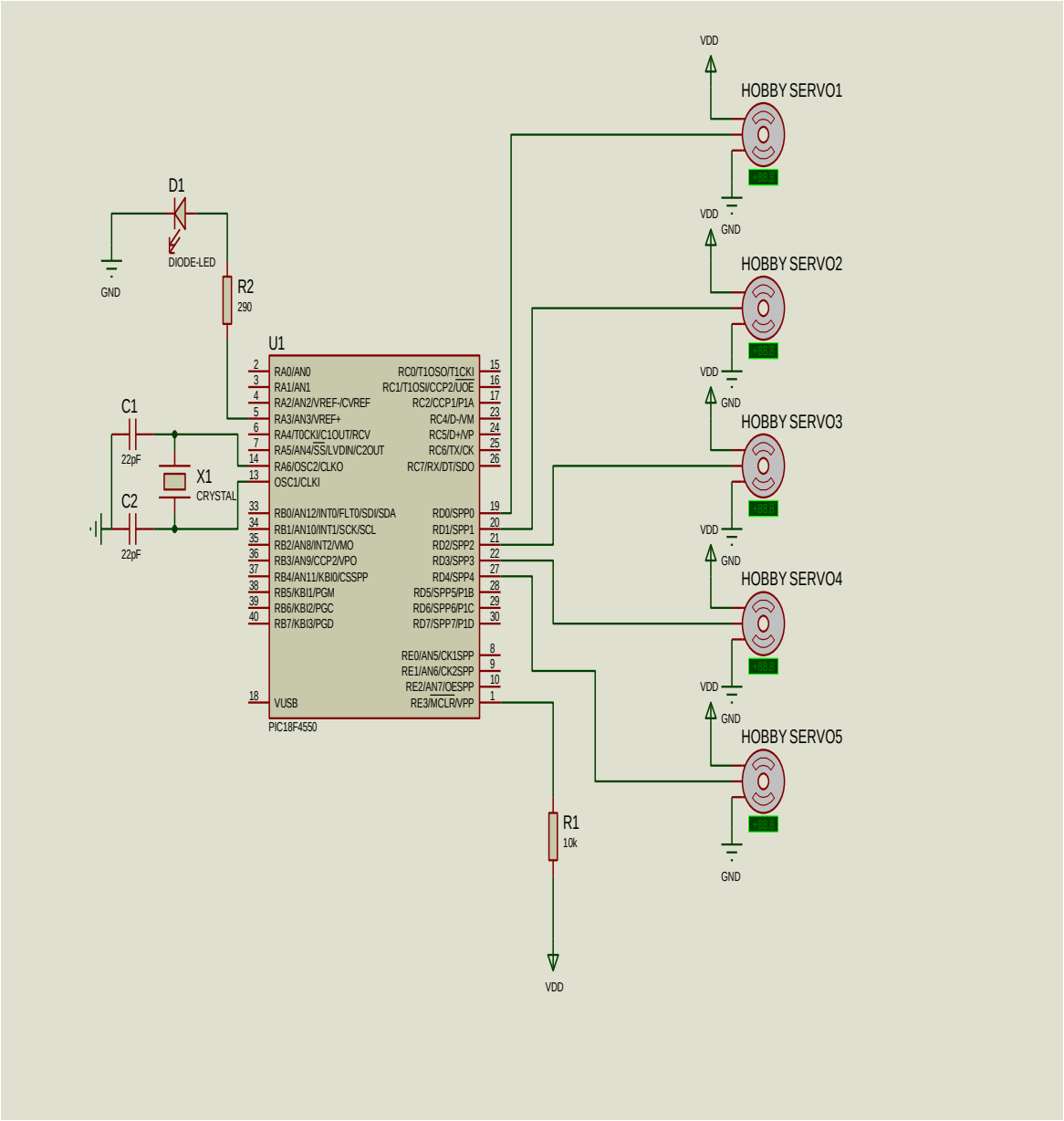

Figure 4.6: Schematic showing the circuit diagram to control robotic arm.

The value of the current limiting resistor R was calculated as follows:

Under normal working conditions, the voltage across an LED is approximately 2V and the current is approximately 10mA. From the Ohm's Law,

$R = V/I$

$R = (5V - 2V)/10mA$

$R = 0.3k\Omega$, the nearest standard resistor was 290Ω.

The robotic arm was designed as shown in figure 4.7 and constructed using five servomotors (four 1501mg type and one sg5010 type), pan and tilt servo brackets, plastic gripping parts, L-shaped servo connector, aluminium metal sheet, plywood, hardwood timber, wires and nails. Three servomotors (type 1501mg) were utilized to construct the base, shoulder and elbow, one (type 1501mg) to make the wrist and the sg5010 type to construct the gripper. Each motor represented a joint for the five degree of freedom robotic arm. The base motor was attached to a timber casing as its stand through a piece of aluminium sheet of metal that was wrapped round the motor and fixed with bolt, nuts and nails. To join one motor to another pan and tilt servo brackets were used and bolts and nuts used to fasten them together. The different parts showed represented motors as follows:

Base – motor 1

Shoulder – motor 2

Elbow – motor 3

Wrist – motor 4

Gripper – motor 5

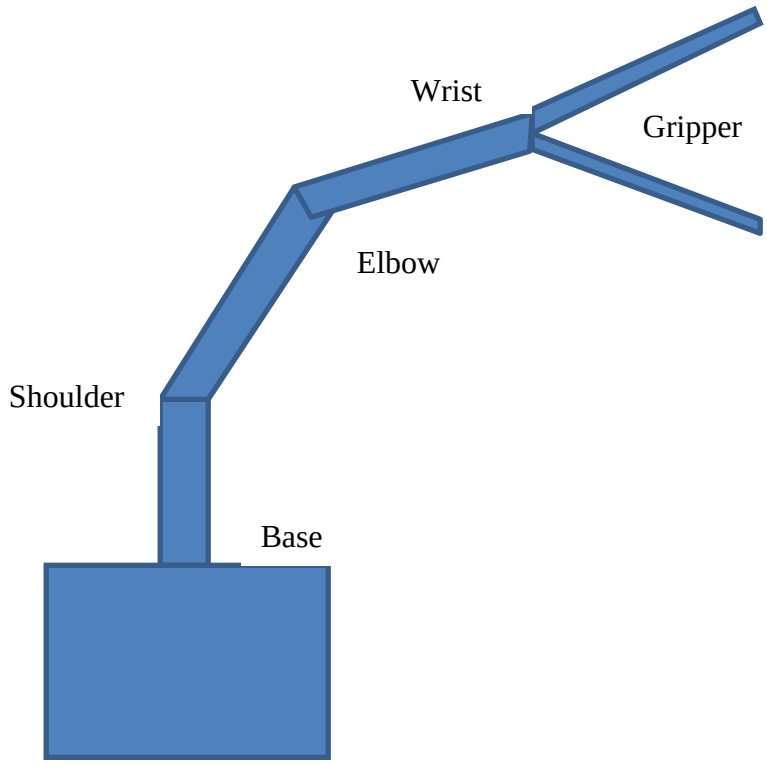

Figure 4.7: Sketch showing the robotic arm design

To attach the wrist motor and the gripper, an L-shaped servo motor connector was used. The gripper arms were joined to the motor through a gear system that would produce the opening and closing effect to the grippers whenever the motor rotated. The robotic arm stand was constructed with hardwood timber and plywood. The base was made square and heavy for the stability of the robotic arm as it moved to avoid shaking or falling. Its body was made with two of its side hardwood timber, the other two side plywood and the top part was made with a handle for portability. Due to the height of the stand each motor wire was added with a similar color wire to the control board and

cello-tape used to attach them together for identification and to the stand to avoid hanging loosely and being a hindrance to the free movement of the robotic arm.

4.2.3 Debugging methods

Different ways of debugging were used in the course of doing the project which included UART reading and writing, LED display, LCD display and current flow indication by the digital multimeter. FTDI adaptor (USB to serial converter module) was used to enable communication between the microcontroller and PC. It was connected to the TX and RX pins of the PIC18F4550 and GND and a code to write to the PC through the mikroC PRO virtual instrument was designed.This was used in different stages of the software development to find out where the codes were working by writing ‘OK’ to the mikroC PRO Virtual instrument or ‘ERROR’ to indicate that the code had failed at a specific point and needed to be checked. Whenever such indications were required, this function was called.

The LCD was used the same way by displaying text and later for communication without using the PC and whenever the serial pins were engaged in other uses. Its function was called immediately after an execution that needed to be debugged. For instance when a motor was selected and given the angle to rotate, the LCD displayed the current motor in use.The LED pin was toggled so as to be ON and OFF depending on the duration of delays in every function called and incase it failed at any point it was used as an indication that a certain part of the software was not working and needed the attention of the programmer. The LED was connected to pin RA3. The pin was toggled as follows;

```
LATA.B3=~LATA.B3;          //pin RA3 toggling
```

The hardware part of the project was also debugged at various stages using the digital multimeter by checking whether the connections were okay. Connecting a microcontroller pin to the GND through the meter terminals could indicate whether that pin was outputting the required voltage. The buzzer part of the meter was used to show wires that were faulty from inside and not allowing current to flow.

4.3 Software design

4.3.1 Introduction

In order to control the rotation of the servo motors and hence the robotic arm, the microcontroller was programmed to give the appropriate instructions to the control wire of the motors through output pins. Software codes were developed at each stage during the design and construction of the robotic arm.

4.3.2 Software installations in the PC

In order to generate the various codes required in the research, the following software were installed in the PC;

- MikroC PRO for PIC version 6.6.1 from Elektronika Software Company, an Integrated Development Environment (IDE) and compiler.
- PICkit™3 programmer software version 3.10 used to import the Hex files from the PC and write them to the PIC microcontroller through the PICkit™3 programming kit and a USB cable.

- Timer calculator software for generating the counts the timers make before generating an interrupt in the production of PWM signal.

4.3.3 Software for controlling servo motors

The software to produce the PWM signal needed was designed using interrupts with its flow chart drawn as shown in figure 4.8 to help in the programming. Timer1 (T1) was used to set period at 20 ms and Timer0 (T0) to set the PWM signal. The microcontroller was set to operate at an external clock frequency of 8 MHz. T1 was started to count 20 ms by pressing the push button after which an interrupt occurred due to overflow (overflow occurs when counts reaches 0xFFFF or 65535). The interrupt of T1 was set to start T0 and set control pin RA0 HIGH. T0 was set to develop interrupt after making its full count for the time the pin RA0 was supposed to be HIGH, for instance 1.0 ms, 1.5 ms and 2.0 ms. The interrupt of T0 was set to put the pin RA0 LOW and to disable itself. The time the pin is set HIGH gives the desired PWM signal that control the position of the servo motor shaft.

This method of producing PWM signal was found to work only with three servo motors since there are only four Timers available in PIC18F4550, i.e. T0 for setting the period and T1, T2 and T3 for three control lines.

Due to this limitation of the number of Timers in the PIC18F4550, another method of producing the PWM signal for five servo motors was used. Since a servomotor receives a signal every 20 ms (20000 µs), this time was divided into ON and OFF time so that the control pin to the motor control line is set ON (HIGH) for the duration the signal is

required and with its completion the pin is set OFF (LOW) for the remaining time of the 20000 µs.

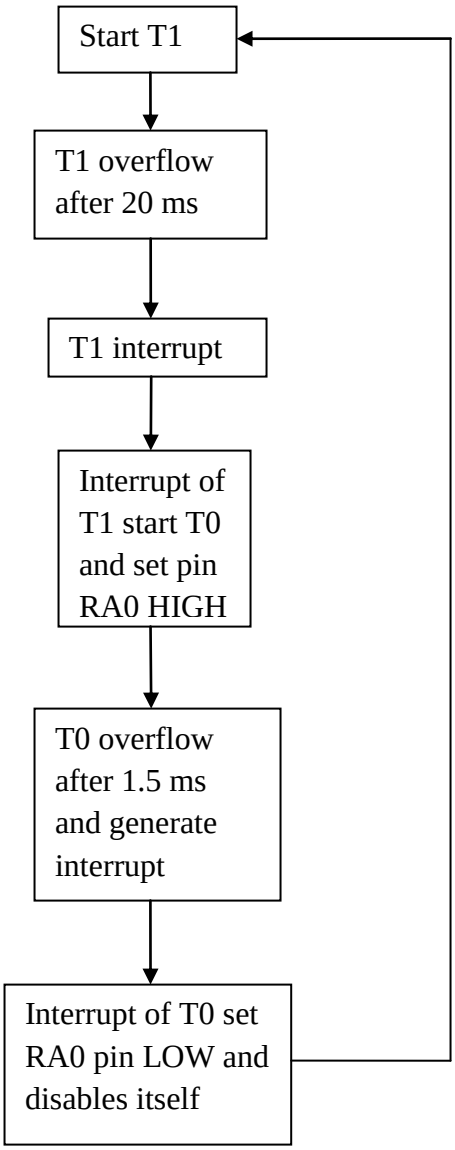

Figure 4.8: Schematic showing the software flow chart to control one servo motor.

4.3.4 Software for controlling the robotic arm

The software to control the motors on the robotic arm was now modified to be able to select a motor number automatically without involving a push button and give it the

desired pulse for the right positioning of the arm as indicated in figure 4.9 and as shown on the software flow chart below;

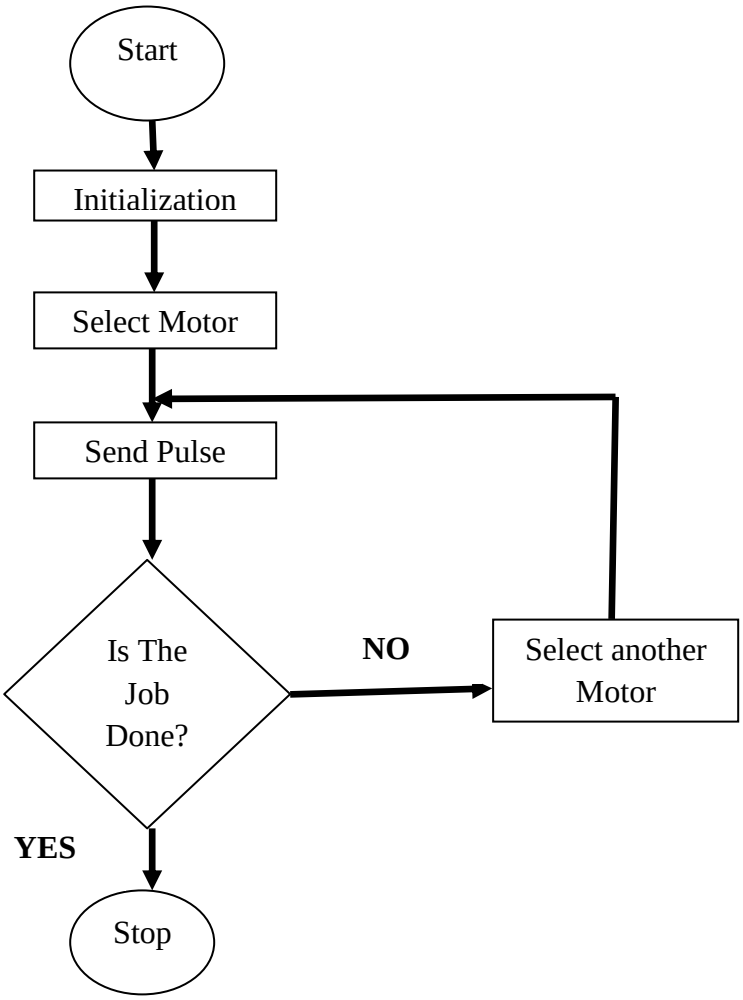

Figure 4.9: Flow chart diagram for the software to control the robotic arm

1. Start – in this step the microcontroller is reset and the first instruction to be executed shall be at address 0001H.

2. Initializing the system.

3. Select a motor.

4. Set the angle and direction of the motor rotation (by sending the appropriate pulse signal).

5. Is the job done?

6. If no, select another motor and repeat step 4.

7. If yes, stop.

The port pins connected to each control line of the servomotors were defined. Switch functions were used to select a motor and give the motor angle. Angles were selected from -90^{o} to $+90^{o}$ with an interval of 10^{o} and each assigned to all the motors so that when a motor is selected, it could move whichever angle given from the range provided. Another switch function was used to combine the motor_select and motor_angle called motor_control.

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1 Introduction

This chapter presents the various data that were obtained in every step of designing and construction of the microcontroller-based five degree of freedom robotic arm using servo motors. Some of the results obtained included the screen shots of the software installed on the PC, the control circuit designs at the various stages of designing and construction of the robotic arm, the software codes involved in controlling the servo motor rotation, angles measured with their respective PWM signal and timer counts and the complete robotic arm. Later in the discussion the data obtained is used to plot a graph that was analyzed to draw conclusions.

5.2 Hardware results

5.2.1 Power supply

The diagram shown in figure 5.1 is the designed power circuit for the system containing 220V to 12V stepdown transformer, two LM2596 adjustable voltage regulators and D4SBA60 bridge rectifier on a white breadboard.

5.2.2 Control circuits

Before connecting the motors to the microcontroller, an external clock signal circuit was designed and the PICkit^{TM}3 module connected as shown in figure 5.2. With the

module connected to the PC through a USB, the microcontroller was ready for programming and the servo motor was connected as shown in figure 5.3.

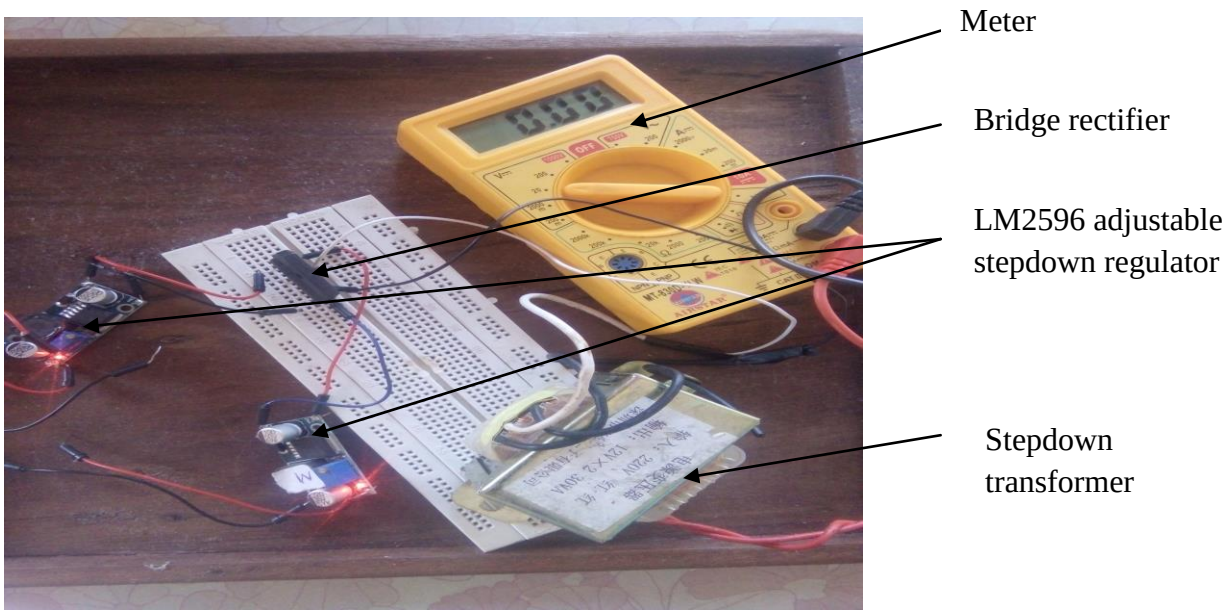

Figure 5.1: Picture showing the power supply section of the circuit

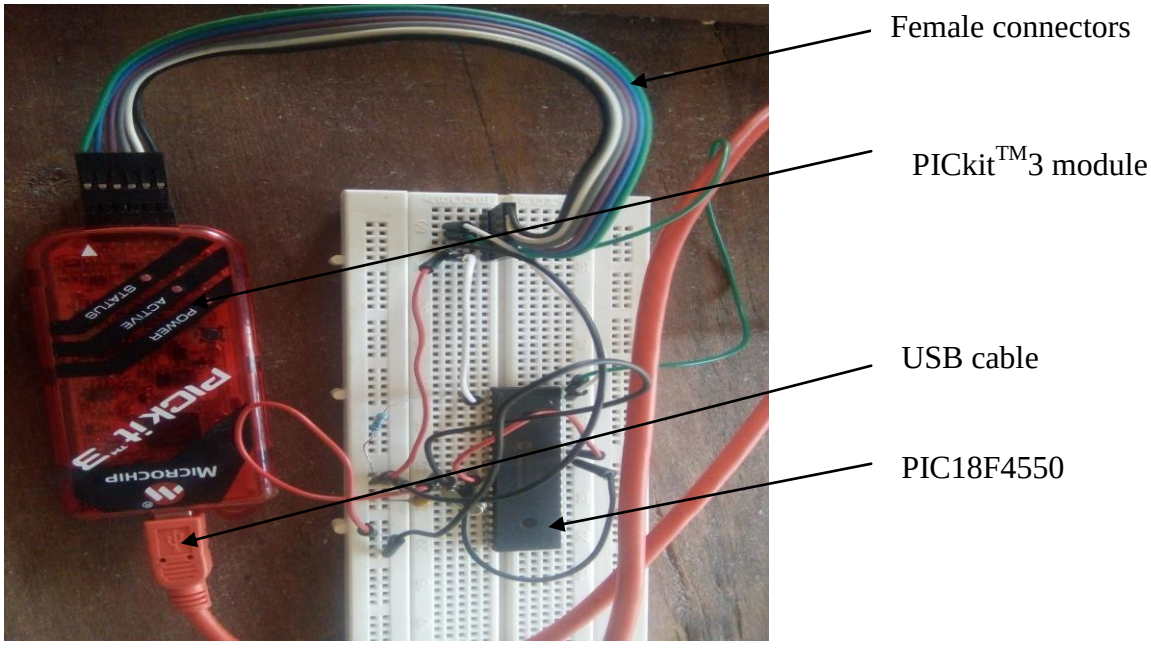

Figure 5.2: Picture showing the PICkit™3 connection to the microcontroller

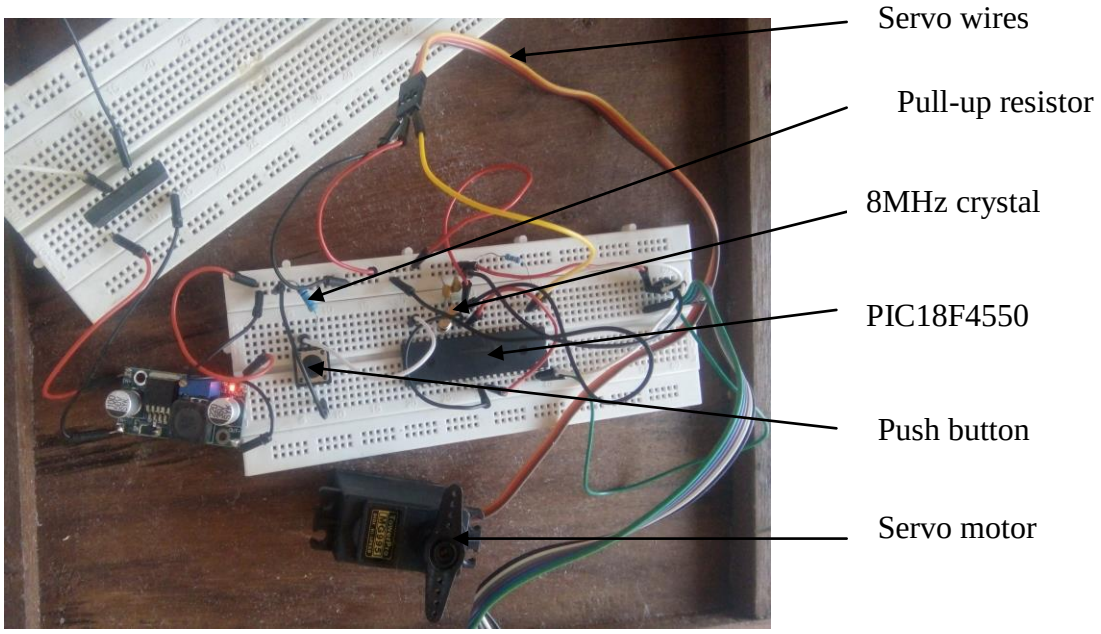

Figure 5.3: Picture showing the control circuit for one servo motor

The control of different motors was done in two stages with the first taking three and the second with the five motors. The figure 5.4 shows the circuit for three motors controlled by the use of buttons that determined which motor to rotate and by what angle.

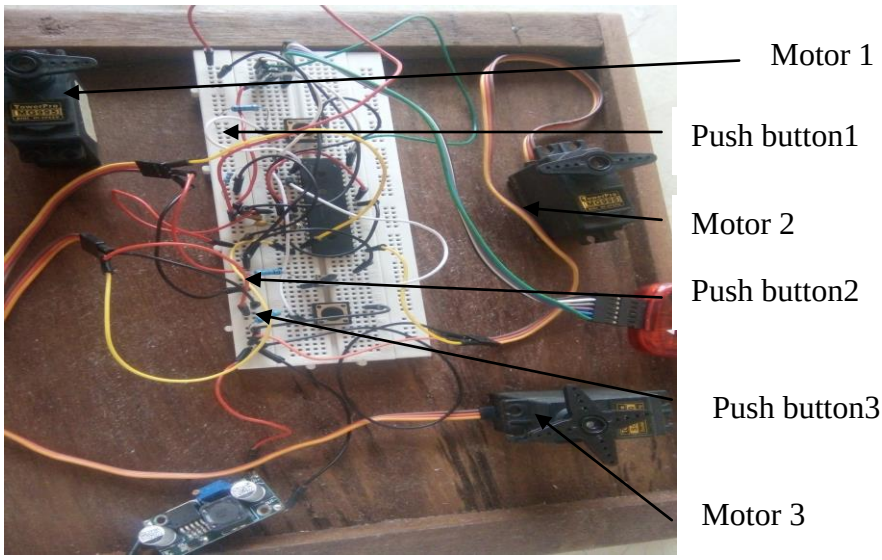

Figure 5.4: Picture showing circuit to control three motors with push buttons

Figure 5.5 shows the circuit for the control of five motors without depending on the buttons but the sequence as indicated on the software.

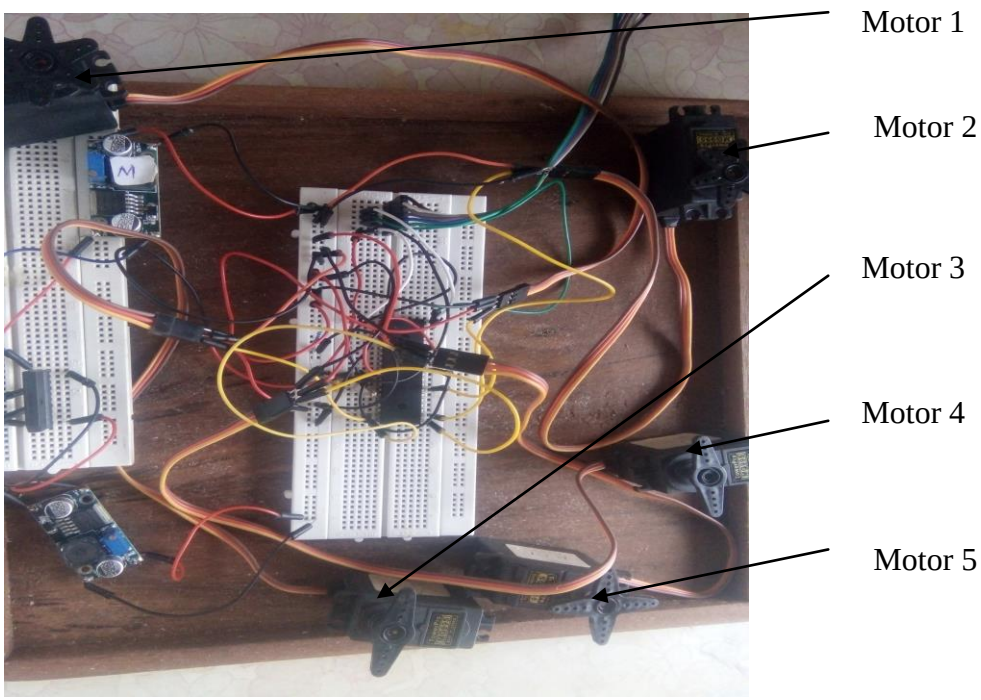

Figure 5.5: Picture showing circuit to control five motors without push buttons

The figure 5.6 shows the circuit to control the robotic arm with the software running without the arm in place. The LCD display the motor running and the LED on the toggled pin ON. The LCD connections were done as specified in the chapter of methodology and as shown in the figure 5.7.

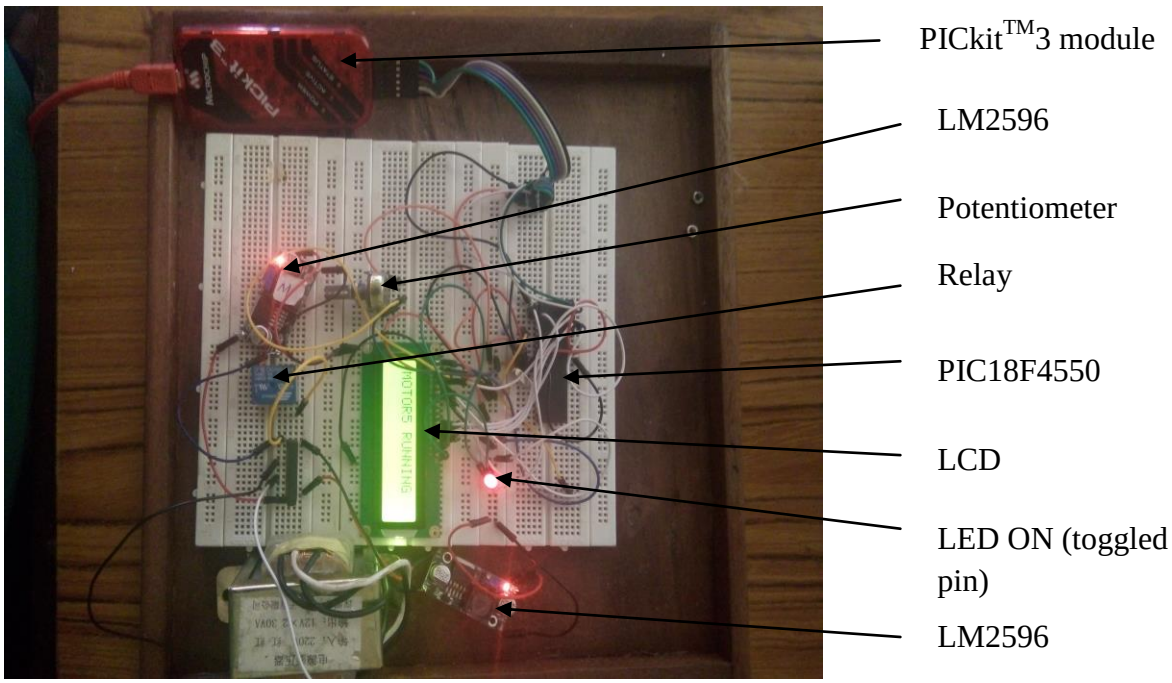

Figure 5.6: Picture of the circuit to control the robotic arm

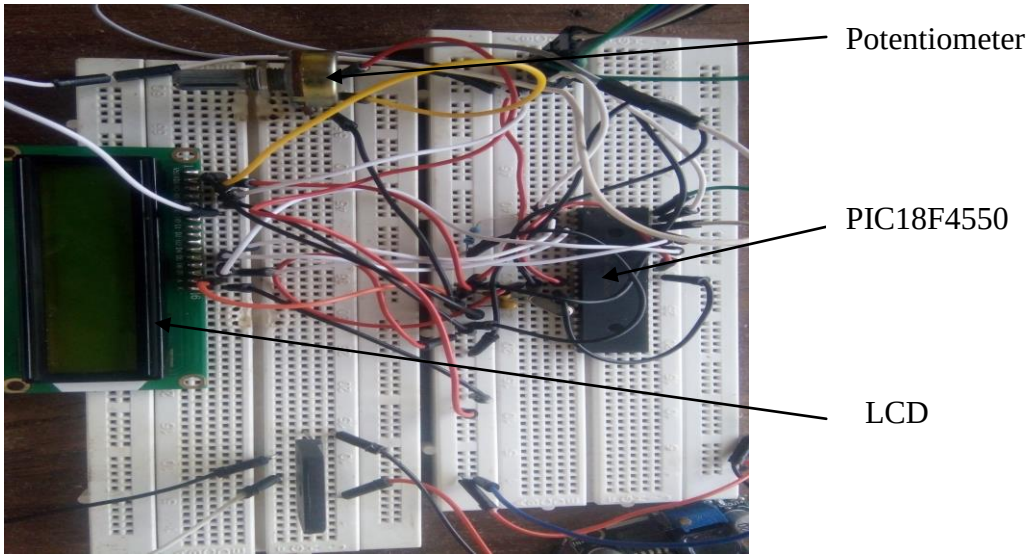

Figure 5.7: Picture of the LCD connections

5.2.3 The robotic arm

The robotic arm was designed and constructed as discussed in the chapter of methodology. The figure 5.8 shows the robotic arm as the complete product.

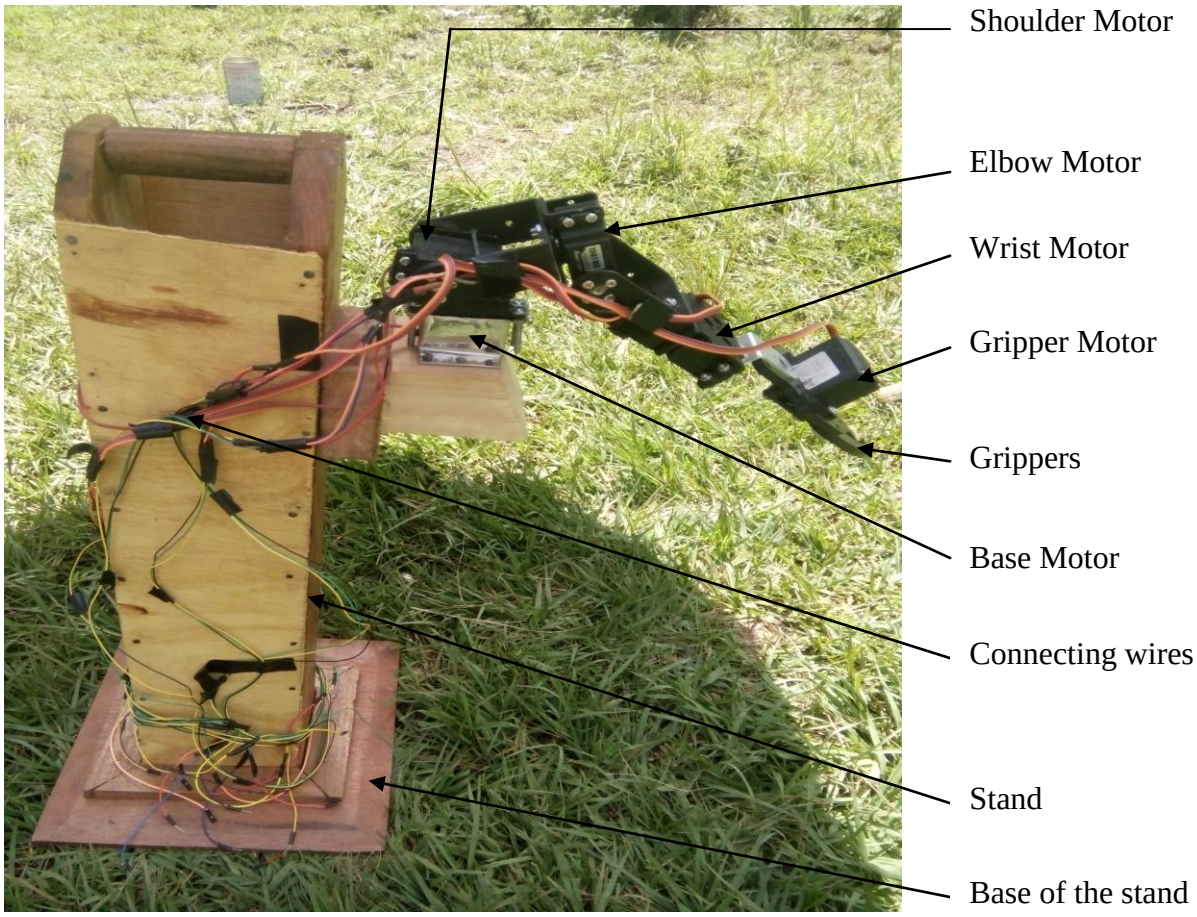

Figure 5.8: Picture of the designed robotic arm

5.3 Software results

5.3.1 Installed software

MikroC PRO, Proteus, timer calculator and PICkit™3 were installed in the PC. The screen shot shown on figure 5.9 shows mikroC PRO and PICkit™3 that were used to

program and upload the software to the microcontroller. Figure 5.10 shows the Proteus software for circuit design and figure 5.11 shows a screen shot of the timer calculator.

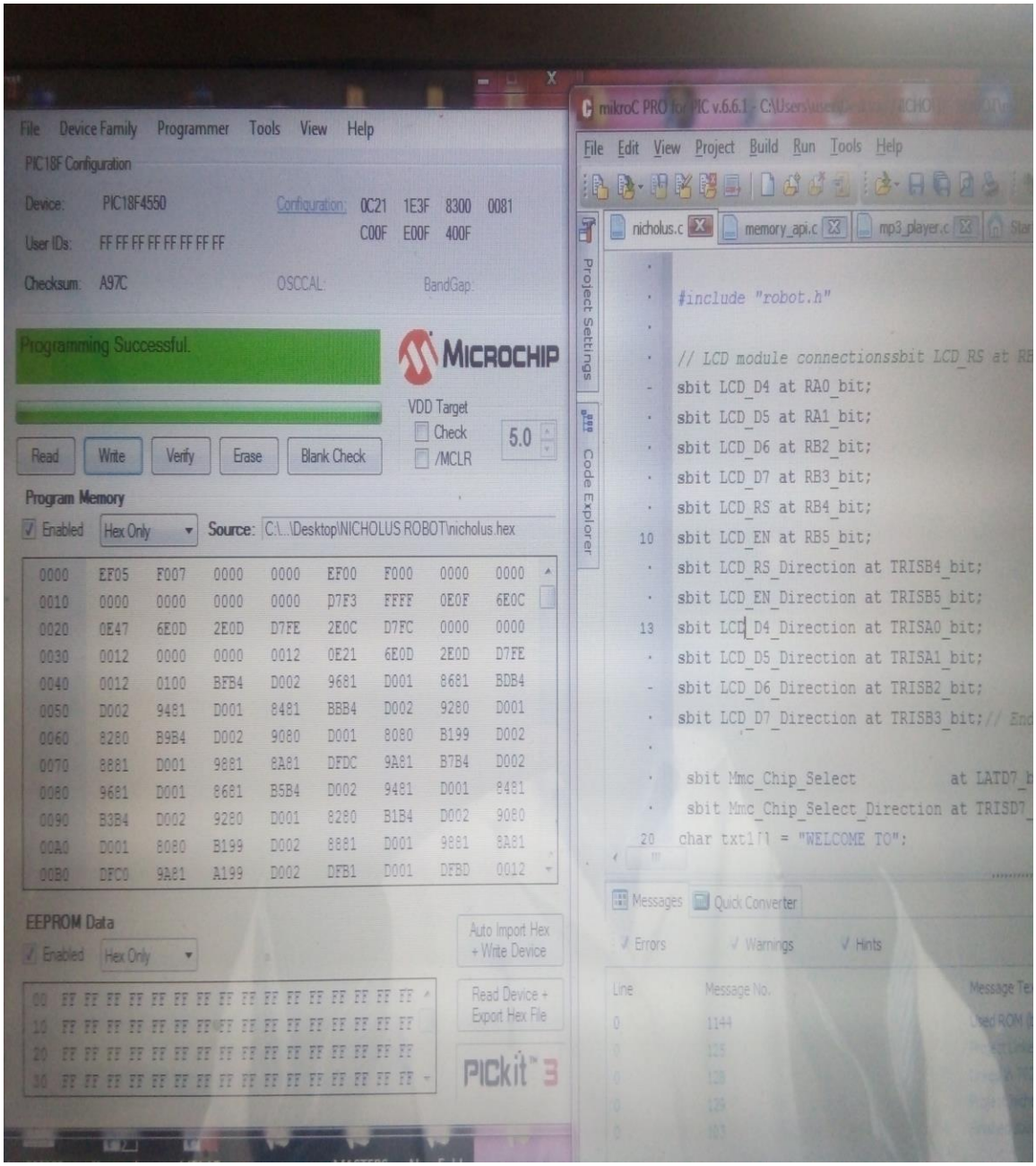

Figure 5.9: screen shot showing the mikroC PRO and PICkit™3 software on the PC

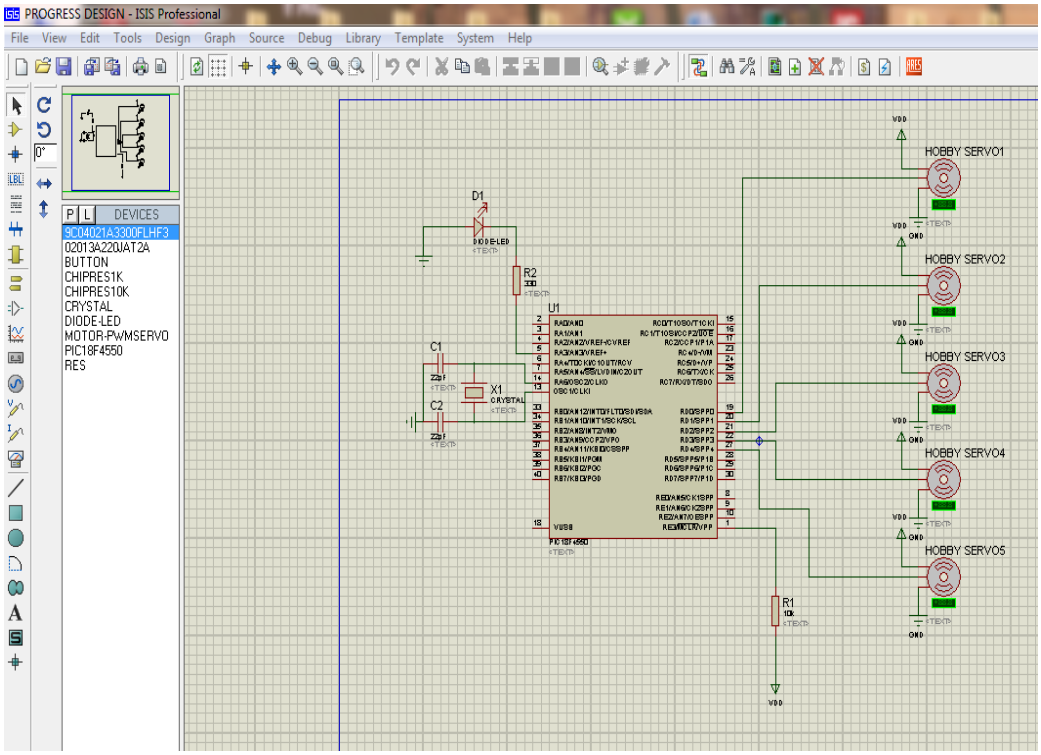

Figure 5.10: Screen shot showing the Proteus software for designing circuits

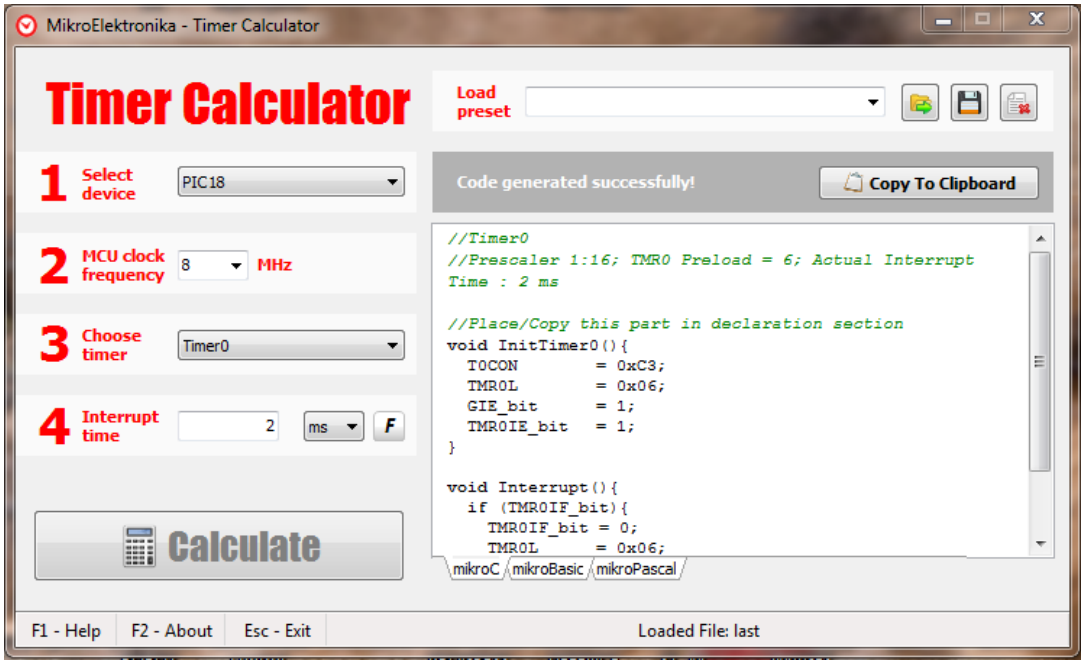

Figure 5.11: Screen shot showing the timer calculator software

5.3.2 Software for controlling servo motors

The software to control one servomotor was designed as shown in appendix B. The codes shown in appendix C were used to control the rotation of motors without using interrupt. Out of the 20000 µs period the control line has an ON signal for only 1000 µs and OFF signal for the remaining 19000 µs as follows:

```
PORTD.B0 = 1;          //pin RD0 set HIGH

Delay_us(1000);        //Time the pin was HIGH

PORTD.B0 = 0;          //pin RD0 set LOW

Delay_us(19000);       //Time the pin was LOW
```

To select and instruct one motor out of the five, an "IF-else" conditional function was used and input pins with push button connected to 10kΩ resistor to the +5.0V were used. Pressing the push button on pin RA0 as given above selected the servomotor on pin RD0 and gave it a pulse of 1000 µs time while releasing it gave the servomotor a pulse of 1500 µs time. When the circuit was powered all the motors rotated to the specified position by the "else" part of the function and then wait for their buttons to be pressed. To display text on the LCD, the codes shown in appendix D were used and the function called in the main program whenever it was necessary. The texts were specified at whatever point this function was called in the program. To debug the system, the codes shown in appendix E were used for the communication between the microcontroller and the PC hence displaying errors in the software at different stages of

development. This was meant to indicate areas where the codes were working with errors. Whenever such indications were required, this function was called.

5.3.3 Software for controlling the movement of the robotic arm

The software to control five motors was modified to suit the control of the robotic arm. The port pins connected to each control line of the servomotors were defined. Switch functions were used to select a motor and give the motor angle. Another switch function called motor_control was used to combine the motor_select and motor_angle. These functions are shown in the appendix F. The main function for controlling the movement of the robotic arm is shown in the appendix G.

5.4 Data obtained

5.4.1 Timer0 counts

In the production of the PWM signal for driving the servomotors using interrupts, the Timer0 was loaded with different values of where to start counts for particular interrupt time needed. A timer calculator from the MikroElektronika Company was used in calculating the time interval. The actual interrupt time was obtained after depleting the counts i.e. counting from the preload value to 0xFFFF or 65535. The angles moved by the servomotors were measured and the values obtained were shown in the table 5.1.

Table 5.1: Showing the angles moved by the servo motors against the PWM signal supplied (interrupt method).

Actual	Prescaler	TMR0	T0CON	TMR0H	TMR0L	Angle
1000	1:8	6	0xC2	_	0x06	0
1055	1:1	63426	0x88	0xF7	0xC2	10
1110	1:1	63316	0x88	0xF7	0x54	20
1165	1:1	63206	0x88	0xF6	0xE6	30
1220	1:1	630966	0x88	0xF6	0x78	40
1280	1:1	62976	0x88	0xF6	0x00	50
1330	1:1	62876	0x88	0xF5	0x9C	60
1390	1:1	62756	0x88	0xF5	0x24	70
1445	1:1	62646	0x88	0xF4	0xB6	80
1500	1:1	62536	0x88	0xF4	0x48	90
1555	1:1	62426	0x88	0xF3	0xD4	100
1610	1:1	62316	0x88	0xF3	0x6C	110
1670	1:1	62196	0x88	0xF2	0xF4	120
1720	1:16	41	0xC3	_	0x29	130
1780	1:1	61976	0x88	0xF2	0x18	140
1830	1:1	61876	0x88	0xF1	0xB4	150
1890	1:1	61756	0x88	0xF1	0x3C	160
1945	1:1	61646	0x88	0xF0	0xCE	170
2000	1:16	6	0xC3	_	0x06	180

5.4.2 ON and OFF PWM signal production method

The other method of producing PWM signal through distribution of the 20000 µs into ON time and OFF time was used and the values in the table 5.2 were obtained. For each ON time, the angle moved by the servomotor was measured. A graph of ON time against the angle was plotted and analyzed on figure 5.12.

Table 5.2: Showing the angles moved by the servo motors against the PWM signal supplied (ON and OFF PWM signal production method).

ANGLE OF ROTATION (DEGREES)	**PULSE ON TIME** µS	**PULSE OFF TIME** µS	**PERIOD** µS
0	1000	19000	20000
10	1055	18945	20000
20	1110	18890	20000
30	1165	18835	20000
40	1220	18780	20000
50	1280	18720	20000
60	1330	18670	20000
70	1390	18610	20000
80	1445	18555	20000
90	1500	18500	20000
100	1555	18445	20000
110	1610	18390	20000
120	1670	18330	20000
130	1720	18280	20000
140	1780	18220	20000
150	1830	18170	20000
160	1890	18110	20000
170	1945	18055	20000
180	2000	18000	20000

A graph of ON time against the angle was plotted. The graph shows a linear trend which indicates that the angle of rotation is directly proportional to the time the pulse is on.

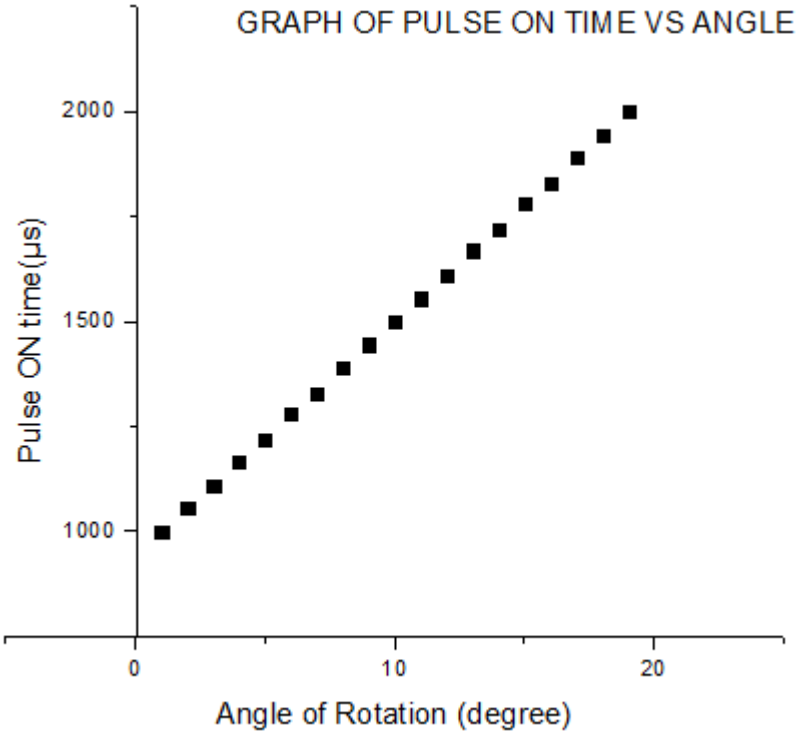

Figure 5.12: Graph of pulse ON time against angle of rotation

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

A PIC microcontroller-based five degree of freedom robotic arm using servo motors was designed, constructed and implemented. The control circuit for the robotic arm was designed in three phases starting with the one to control one servo motor, then to control different servo motors and lastly to control the arm.

In the control of one servo motor, a circuit was designed using PIC18F4550 microcontroller, white breadboard, jumpers, wires, 8 MHz crystal, two 10 kΩ resistor, two 22pF capacitors and a digital push button. The power supply to the microcontroller and servo motors was designed using a stepdown transformer, bridge rectifier and two LM2596 adjustable voltage regulators.

The software to produce the PWM signal needed was designed by use of C programming language on MikroC PRO for PIC Integrated Development Environment and compiler, version 6.6.1 from Elektronika Software Company. The software Hex file was imported from the PC to the microcontroller using PICkit^{TM}3 programming software through the PICkit^{TM}3 programming kit and a USB cable. Interrupt method was used where Timer1 (T1) was used to set period at 20 ms and Timer0 (T0) to set the PWM signal. Due to the limitation of the number of Timers in the PIC18F4550, another method of producing the PWM signal for five servo motors was used. Since a servomotor receives a signal every 20 ms (20000 μs), this time was divided into ON

and OFF time so that the control pin to the motor control line is set ON (HIGH) for the duration the signal is required and with its completion the pin is set OFF (LOW) for the remaining time of the 20000 μs.

The circuit designed was used to send the appropriate signals to different servo motors to produce the required angles of rotation. The circuit and the software were modified at different stages depending on the number of motors and various angles needed.

The robotic arm was designed and constructed using five servo motors, pan and tilt servo brackets, plastic gripping parts, L-shaped servo connector, aluminium metal sheet, plywood, hardwood timber, wires and nails.

The software was further modified to control the movement of the robotic arm such that it could rotate in both clockwise and anticlockwise direction from the base motor, move up and down from the shoulder and elbow motors, rotate with its wrist motor and lastly open and close the gripper motor for picking and placing an object of specified mass.

Through the software, the robotic arm move by itself once the power is put on. It repetitively picks, lifts and drop objects of specific mass from one place to another without the influence or interference by the operator. This was an attempt to complement the operations of the land movers available that require the operator to remain throughout the entire working time.

6.2 Recommendations

a) Investigate the effect of using motors of higher torque on the strength of the robotic arm.

b) Investigate the effect of using sensors in providing a feedback mechanism for the robotic arm's self-evaluation.

c) Investigate the effect of sensors on the intelligence of the robotic arm.

d) Investigate the effect of incorporating an external memory (EEPROM) for saving each motor angle for check-up during initialization for the safety of the load in case of unexpected shutdown.

e) Investigate the effect of incorporating fingers on the arm instead of the grippers used.

REFERENCES

Aakash, K. S. (2012). Gesture Activated Robotic Arm, *International Journal of Scientific and Research Publications*, **2:** 1-6.

Abdul, K. N. L., Ahmad, S. A., Che, A. S. and Ishak, A. T. (2012). Development of Adjustable Gripper for Robotic Picking and Placing Operation: *International Journal on Smart Sensing and Intelligent Systems*, **4:** 1019-1043.

Aishwarya, D., Mayuresh, M. and Virushali, D. (2016). Haptic Arm Robot: *International Journal of Advanced Research in Electronics and Communication Engineering*, **5:** 1438-1440.

Ajayi, A., Awodele, O. and Jegede, O. (2007). Development of a Microcontroller Based Robotic Arm, *Journal of Proceedings Of the 2007 Computer Science and IT Education Conference*, **1:** 549-557.

Alan, L., Ryan, M. and Vuk, Z. (2006). Report on Project, Monitor and Control of an Excavator Robot: Department of Electrical and Computer Engineering, QUEEN'S University, Kingston, Canada.

Amarashid, P., Arriffuddin, J., Muhammad, S. S. and Vijaya, K. R. V. (2009). Development of a Single DOF Robot Arm Using PIC Microcontroller: *Journal of Malaysian Technical Universities Conference on Engineering and Technology* **1:** 36-38.

Amey, P. R., Bhagwat, S. D., Prateek, R. N. and Vipul, J. G. (2013). Robotics Arm Control Using Haptic Technology: *International Journal of Latest Research in Science and Technology*, **2:** 98-102.

Angeliki, B., Bin, H., Bryan, B., Jaron, O., Jianjun, M. and Shuying, Z. (2016). Scientific Report on Noninvasive Electroencephalogram Based Control of a Robotic Arm For Reach and Grasp Tasks: University of Minnesota, College of Science and Engineering and The Medical School.

Arian, F. (2014). Thesis on Design, Implementation and Control of a Robotic Arm Using PIC16F877A Microcontroller: Eastern Mediterranean University, Gazimagusa, North Cyprus.

Arid, K., Mahesh, V. and Puran, S. (2013). Design of a Robotic Arm with Gripper and End Effector for Spot Welding: *Universal Journal of Amity school of Engineering and Technology* **3:** 92-97.

Ashutosh, P. and Rajiv, R. (2013). Thesis on Robotic Arm Control Through Human Arm Movement Using Accelerometers: Electronics and Instrumentation Engineering Department; National Institute Of Technology, Rourkela.

Babul, S. K. I., Mohd, A. K. Y. and Reza, E. S. (2012). Project Report on Wireless Mobile Robotic Arm: *International Symposium on Robotics and Intelligent Sensors, Procedia Engineering* **41:** 1072-1078

Ball, S. R. (2002). Embedded Microprocessor Systems: Butterworth - Heineman Ltd, 219

Benachiba, C., Haidari, A. M. A. and Zahir, M. (2013). Software Interfacing Of Servo Motor with Microcontroller: *Journal of Electrical Systems* **9:** 84-99.

Bhanu, P. S. B., Gokul, S. A., Rama, K., Sastry, A. S. C. S. and Sowmya, B. G. (2012). Design and Implementation of a Robotic Arm Based on Haptic Technology, *International Journal of Engineering Research and Applications* **2:** 3098-3103.

Dirman, H., Mohamed, F. Z. and Yousef, M. A. (2013). Wall Follower Autonomous Robot Development Applying Fuzzy Incremental Controller: *Journal of Intelligent Control and Automation*, **4:** 18-25.

Dylan, S., Jason, W. and Nehran, H. R. (2004). Research Paper on Computer Controlled Robot Arm: Faculty of Engineering, Summer 2004 ELEC499A.

Eega, S., Jaimes, A., Jamshidi, M., Matthew, A. J., Nagothu, K. and Shaneyfelt, T. (2008). Applications and Prototype For System of Systems Swarm Robotics: *Journal of Proceedings of 2008 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, Piscataway, N. J., 2049-2055.

Florentinus, B. S. and Ricky, F. A. (2016). Report on Project on Robot Arm Controlled by Muscle Tension Based on Electromyography and PIC18F4550: Department of Electrical Engineering, Unika Soegijapranata, Semarang, Indonesia.

Haptic_technology http://en.Wikipedia.org/wiki/, 20th March 2016.

Hung, C. P., Hong, J. S., Jia, W. C., Chiu, B. M. and Wei, G. L. (2009). PIC-Based Multi- Channel PWM Signal Generation Method and Application to Motion Control of Six Feet Robot Joy: *International Journal of Circuits, Systems and Signal Processing*, **3:** 73- 81.

ISO Standard 8373: 1994, Manipulating Industrial Robots- Vocabulary.

Kabuka, R. M. (1988). Microcontroller Based Architecture for Control of a Six Joints Robot Arm: *Journal of IEEE Transactions on Industrial Electronics* **35:** 217- 221.

Khairul, A. B. R. (2009). Report on Pick and Place Robot (Robotic Arm): Faculty of Electrical Engineering: Technical University of Malaysia, Melaka.

Mathia, K. (2010). Robotics for Electronics Manufacturing: Principles and Applications In Clearroom Automation, UK: Cambridge University.

MicroEngineering Labs (2008). PicBasic Pro Compiler: MicroEngineering Labs Inc.

Midhun, M., Mithun, S., Muhammed, F., Mohammed, J. N. K., Neetha, J. and Safwan, C. N. (2015). Wireless Control of Pick and Place Robotic Arm Using an Android Application: *International Journal of Advanced Research in Electrical, Electronics And Institution Engineering* **4:** 2410-2416.

Muir, F. P. And Neuman, P. C. (1985). Pulse width Modulation Control of Brushless Dc Motor for Robotic Applications: *Journal of IEEE Transactions on Industrial Electronics* **32:** 222- 229.

Paulo, E. M. (2005). How to Interface a Microchip PIC MCU with a Hobby R/ C Servo: http://www.merloti.com/EngHome/computing/software.htm, 20th December 2015, Interfacing a microchip.

PIC18F4550 data sheet: www.microchip.com. 15th January 2016.

Pull-up resistor: www.sparkfun.com, 20th February 2016.

Ramaiah, P. S., Salyanarayana, G. V. and Venkateswara, R. (2011). A Microcontroller Based Four Fingered Robotic Hand: *International Journal of Artificial Intelligence and Applications)* **2:** 111-115.

Reshamwala, A. and Singh R. (2015). A Review on Robot Arm Using Haptic Technology: *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, **4:** 2204-2211.

Robert, S. L. (2011). Report on AL5A Robotic Arm Project; Web-Based Control with Spatial Awareness and Intuitive Manipulator, Senior Software Engineering project: Florida Gulf Coast University; Fort Myers, Florida.

Savas, S. and Yalchin, I. (2013). Microcontroller-Based Robotics and SCADA Experiments: *Journal of IEEE Transactions on Education*, **56:** 424-429.

Servo motors: www.embedded-lab.com, 10th December 2015.

Siti, H. B. I. (2011). Research Report on Design of Robotic Arm Controller Using MATLAB: Technical University of Malaysia, Melaka.

Timothy, W. (2007). Designing Embedded Systems with PIC Microcontrollers: Principles and Applications: Published By Elsevier Ltd; UK.

Vajnberger, V. (2011). Remote Control of Robotic Arm with Five DOF: MIPRO Electronics.

APPENDICES

Appendix A: PIC18F4550 data sheet

MICROCHIP **PIC18F2455/2550/4455/4550**

28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μA typical
- Sleep mode currents down to 0.1 μA typical
- Timer1 Oscillator: 1.1 μA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide Operating Voltage Range (2.0V to 5.5V)

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I²C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Appendix B: Software for controlling one servo motor using interrupt

```
void InitTimer1(){
   T1CON        = 0x01;
  TMR1IF_bit        = 0;         //clear T1 interrupt flag bit
  TMR1H        = 0x63;         //set T1 higher bit value
  TMR1L        = 0xC0;         //set T1 lower bit value
  TMR1IE_bit        = 1;        //set T1 interrupt enable bit
  INTCON        = 0xC0;
}
void Interrupt(){
  if (TMR1IF_bit){{
    TMR1IF_bit = 0;
    TMR1H        = 0x63;
    TMR1L        = 0xC0;
   if(portd.B0==0){
  LATA.B0=1;            //set control pin high
h_reg=0x00;             //2.0ms pulse width
```

```
l_reg=0x51;

freq_rf=0xC2;

 InitTimer0();          //start T0

}
```

Appendix C: Software for controlling one servo motor without using interrupt

```
If (LATA.B0==0)          //input pin pressed (grounded)
{ PORTD.B0=1;            //control pin goes HIGH
Delay_us(1000);          //time the control pin stays HIGH
PORTD.B0=0;              //control pin goes LOW
Delay_us(19000);         //time the control pin stays LOW
}
Else
{ PORTD.B0=1;            //control pin goes HIGH
Delay_us(1500);          //time the control pin stayed HIGH
PORTD.B0=0;              //control pin went LOW
Delay_us(18500);         //time the control pin stayed LOW
}
```

Appendix D: Software for displaying text on the LCD display

```
//declarations

sbit LCD_D4 at RA0_bit;

sbit LCD_D5 at RA1_bit;

sbit LCD_D6 at RB2_bit;

sbit LCD_D7 at RB3_bit;

sbit LCD_RS at RB4_bit;

sbit LCD_EN at RB5_bit;

sbit LCD_RS_Direction at TRISB4_bit;

sbit LCD_EN_Direction at TRISB5_bit;

sbit LCD_D4_Direction at TRISA0_bit;

sbit LCD_D5_Direction at TRISA1_bit;

sbit LCD_D6_Direction at TRISB2_bit;

sbit LCD_D7_Direction at TRISB3_bit;

//displaying text

  Lcd_Init();                                  // Initialize LCD

  Lcd_Cmd(_LCD_CLEAR);                  // Clear display
```

```
Lcd_Cmd(_LCD_CURSOR_OFF);          // Cursor off

 Lcd_data-output (1,1, txt1);                //display text1 on starting at column1, row1

 Delay_ms(2000);
```

Appendix E: Software for debugging the system

```
void console_log(char *data_uart){
  UART1_Write_Text(data_uart);
   UART1_Write(0X0D);
  UART1_Write(0x0A);}
//function for error display
if(check_errors!=0){                    //returns error
       console_log("ERROR");          //write the word ERROR onto the PC
    }
else {                                  //returns no error
     console_log("OK");                //write the word OK onto the PC
    }
```

Appendix F: Switch functions to select motors and give appropriate angles

```
//defining ports
sbit mt1 at LATD0_bit;
sbit mt2 at LATD1_bit;
sbit mt3 at LATD2_bit;
 sbit mt4 at LATD3_bit;
 sbit mt5 at LATD4_bit;
//selecting motors
void motor_select(int motor_number){
switch(motor_number){
   case 1:{
         port_pin_holder=1;          //motor number 1
      break;
    }
    case 2:{
        port_pin_holder=2;            //motor number 2
      break;
```

```
}
case 3:{
     port_pin_holder=3;              //motor number 3
   break;
}
case 4:{
        port_pin_holder=4;           //motor number 4
  break;
}
case 5:{
         port_pin_holder=5;          //motor number 5
        break;
}
default:{
 port_pin_holder=1;                  //motor number 1
break;
}
```

```
        }

       }

     //assigning angles to motors
void motor_angle (int motor_angle) {
  switch(motor_angle){
    case 0:{                        //angle 0 equivalent to -90o
      unsigned int i;               //declaring variable i as unsigned integer
  for(i=0;i<50;i++)
  {
  if( port_pin_holder==1){     //angle 0 assigned to motor number 1
    mt1 = 1;                    //pin assigned to motor 1 becomes HIGH
    Delay_us(1000);             //duration motor 1 receive the pulse
    mt1 = 0;                    //pin assigned to motor 1 becomes LOW
    Delay_us(19000);            // duration motor 1 receive no pulse
 }
   if( port_pin_holder==2)     //angle 0 assigned to motor number 2
{
```

```
   mt2 = 1;                    //pin assigned to motor 2 becomes HIGH
   Delay_us(1000);             //duration motor 2 receive the pulse
   mt2 = 0;                    //pin assigned to motor 2 becomes LOW
   Delay_us(19000);            // duration motor 2 receive no pulse
 }
//selecting a motor and assigning it an angle to rotate
void motor_control(int motor_select,int motor_angle){
motor_select(motor_select);
motor_angle (motor_angle);}
```

Example:

```
motor_control(1,10);                    //motor1 selected to move to 10^o position
Delay_ms(1000);
motor_control(2,150);                   //motor2 selected to move to 150^o position
Delay_ms(1000);
motor_control(4,70);                    //motor4 selected to move to 70^o position
Delay_ms(1000);
motor_control(3,140);                   //motor3 selected to move to 140^o position
```

```
Delay_ms(1000);

motor_control(5,90);                    //motor5 selected to move to 90^o position

Delay_ms(1000);
```

Appendix G: Software showing the main function for controlling the movement of the robotic arm

```
/* Program Name: ROBOTIC ARM CONTROL

Name of Author: NICHOLUS K. NDWIGA

KENYATTA UNIVERSITY

PHYSICS DEPARTMENT

August 2016

……*/

/* this piece of code shows the main function to control the movement of the robotic arm through the servo motors. The code help in supplying the correct PWM signal to servo motors and displaying what is happening on an LCD display as well as toggling the pin

……..*/


#include "robot.h"

char txt1[] = "WELCOME TO";

char txt2[] = "NICHOLUS ROBOT";

char txt8[] = "LOADING OK";

char txt9[] = "ROBOT LOADING";
```

```
char i;

// LCD module connections

sbit LCD_RS at RB4_bit;

sbit LCD_EN at RB5_bit;

sbit LCD_D4 at RA0_bit;

sbit LCD_D5 at RA1_bit;

sbit LCD_D6 at RB2_bit;

sbit LCD_D7 at RB3_bit;

sbit LCD_RS at RB4_bit;

sbit LCD_EN at RB5_bit;

sbit LCD_RS_Direction at TRISB4_bit;

sbit LCD_EN_Direction at TRISB5_bit;

sbit LCD_D4_Direction at TRISA0_bit;

sbit LCD_D5_Direction at TRISA1_bit;

sbit LCD_D6_Direction at TRISB2_bit;

sbit LCD_D7_Direction at TRISB3_bit;// End LCD module connections

void main() {
```

```
ADCON1=0b00001111;                  //configure pins as digital I/O

UART1_Init(9600);                   // Initialize UART module at 9600 bps

Delay_ms(100);

TRISA.B2=0;                         //relay pin set as an output

Lcd_Init();

Delay_ms(100);

Lcd_Cmd(_LCD_CLEAR);                // Clear display

Lcd_Cmd(_LCD_CURSOR_OFF);           // Cursor off

Lcd_Out(1,1, txt1);

Delay_ms(2000);

Lcd_Cmd(_LCD_CLEAR);

Lcd_Out(1,1, txt2);

Delay_ms(2000);

Lcd_Cmd(_LCD_CLEAR);

Lcd_Out(1,1, txt9);

Delay_ms(2000);

Lcd_Cmd(_LCD_CLEAR);
```

```
Lcd_Out(1,1, txt8);

Delay_ms(2000);

TRISA.B3=0;                          //LED RUNTIME

LATA.B3=1;                           //toggled pin high

Delay_ms(2000);

LATA.B2=1;                           // output to the relay

Delay_ms(2000);

while(1){                            //endless loop

LATA.B3=~LATA.B3;                    //toggling the pin

motor_control(1,0);                  //motor 1 selected to rotate to angle 0 position

Delay_ms(2000);

motor_control(4,150);                //motor 4 selected to rotate to angle 150 position

Delay_ms(2000);

motor_control(5,120);                //motor 5 selected to rotate to angle 120 position

Delay_ms(2000);

motor_control(4,0);                  //motor 4 selected to rotate to angle 0 position

Delay_ms(2000);
```

```
motor_control(3,120);          //motor 3 selected to rotate to angle 120 position

motor_control(2,0);            //motor 2 selected to rotate to angle 0 position

Delay_ms(2000);

motor_control(1,120);          //motor 1 selected to rotate to angle 120 position

Delay_ms(2000);

motor_control(3,0);            //motor 3 selected to rotate to angle 0 position

Delay_ms(2000);

motor_control(4,150);          //motor 4 selected to rotate to angle 150 position

Delay_ms(2000);

motor_control(5,70);           //motor 5 selected to rotate to angle 70 position

Delay_ms(2000);

motor_control(5,120);          //motor 5 selected to rotate to angle 120 position

Delay_ms(2000);

motor_control(4,0);            //motor 4 selected to rotate to angle 0 position

Delay_ms(2000);

}

}
```