ROBOTIC ARM CONTROL WITH ARDUINO

A Project submitted in partial fulfillment of the
Requirements for the Award of Degree of
Bachelor of Science in Electrical and Electronic Engineering

Submitted By
MD. Mobarak Ali
ID: 143-33-2170
Tiasha Mesbah Noireet
ID: 143-33-2207
Tamanna Tasnim
ID: 143-33-2230

Supervised By
Tasmia Baten Dina
Senior Lecture, Faculty of Engineering
Department of EEE

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
FACULTY OF ENGINEERING

DAFFODIL INTERNATIONAL UNIVERSITY

November 2018

TO

OUR BELOVED PARENTS

&

HONOURABLE FACULTY MEMBERS

Certification

This is to certify that this project entitled "ROBOTIC ARM CONTROL WITH ARDUINO" is completed by the following students under my direct supervision also this work has been carried out by them in the Department of Electrical and Electronic Engineering under the Faculty of Engineering of Daffodil International University in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering. The presentation of the work was held on November 2018

Signature of the candidate

____
Name: Md. Mobarak Ali
ID: 143-33-2170

____
Name: Tiasha Mezbah Noireet
ID: 143-33-2207

____
Name: Tamanna Tasnim
ID: 143-33-2230

Signature of the supervisor

____
Mrs. Tasmia Baten Dina
Senior Lecturer
Department of EEE
Daffodil International University

ACKNOWLEDGEMENT

At first we want to thank the almighty Allah from the bottom of our heart. We would like to express our sincere gratitude to our honorable supervisor Senior Lecturer Mrs. Tasmia Baten Dina, Department of Electrical and Electronic Engineering, DIU. She encourage us in every moments. We are thankful to her for her continuous encouragement, kind co-operation, and scholastic guidance all along the project work. He has always been extremely generous with his time, knowledge and ideas and allowed us great freedom in this research. We also want to convey our thankfulness to Prof. Dr. Md Shamsul Alam Head, Department of Electrical and Electronic Engineering for his help, support and constant encouragement. We express our humble gratitude to all teachers of Department of Electrical and Electronic Engineering for their support in numerous ways throughout this project work. We are also grateful to the authors whose valuable research papers and books we have considered as reference in this project paper. Apart from that, we would like to thank our entire friends for sharing knowledge; information and helping us in making this project a success. Also thanks for lending us some tools and equipment. Finally we would like to thank our parents who have given us tremendous inspirations and supports. Without their mental and financial supports, we would not able to complete our project.

Authors

ABSTRACT

In our project, we designed the robotic arm to work with an outside user or to perform predetermined commands. The most developed field of robotic arms is the industry and medicine sector. That’s why we designed our project to make it useful for the industry and the medicine sectors. In our project, the robotic arm can grab round shape objects, pick objects and can place it to the desired position. It also has a record mood. In recording mood, it can perform the recorded moves or recorded angles again and again until we manually stop it. While doing this, the robot control is using Arduino Nano microcontroller.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

LED	Light emitting Diode
VCC	Voltage Common Collector
AC	Alternating Current
DC	Direct Current

CHAPTER 1

INTRODUCTION

1.1 Introduction

In our project, we research about the mechanism and software operations of the robotic arm. We designed the arm to pick and place round shape materials, and to perform the commands previously determined by the user. First, we determined what will be the functions of the robot arm and what movements it could make. The robotic arm can grab round shape objects, pick objects and can place it to the desired position. That's how the programming is. It also has a record mood. In recording mood, it can perform the recorded moves or recorded angles again and again until we manually stop it. This project is also designated for doing manual tasks. The servomotor is preferred in order to be able to perform these operations properly since the motor to be selected must operate accurately and should be at high torque. We build the robotic arm with 3 servo motors so the robotic arm can move in 3 axis direction. We use an Arduino Nano microcontroller and write the program code in C language.

1.2 Problem Statement

In our project, we develop the design, mechanism and programming section of the three finger robotic arm. We build the body structure with aluminium metal sheet. The metal body makes the arm stronger than the previous three finger robotic arms. We also improve the programming section. Now it can pick a round shape object then place it very accurately in a position can the movement and can do the same thing until we manually stop it. We use better servo motors to make the movements smooth. To get a better grip we use tie cable for making the fingers. The previous three finger robotic arm cannot lift much weight for the lacking of body strength. In our project we try to fulfill the requirements to make the robotic arm more useful.

1.3 Objects

This robotic arm is designed to perform predetermined commands which is picking and place objects. The robotic arm can grab round shape objects, pick objects and can place it to the desired position. That's how the programming is. It also has a record mood. In recording mood, it can perform the recorded moves or recorded angles again and again until we manually stop it. The servomotor is preferred in order to be able to perform these operations properly since the motor to be selected must operate precisely and must be at high torque. We build the robotic arm with 3 servo motors so the robotic arm can move in 3 axis direction. We use an Arduino Nano microcontroller and write the program code in C language. A Buck converter used for step down the input voltage. And 4 potentiometers for controlling the robot arm. A 5V power supply is also preferred for the robot to work.

1.4 Advantages

- Able to work in the risky sections
- Both manual and auto mood are available.
- It can lift weight up to 200 gms
- Having record mood it can do the same thing again and again
- It Can move all 4 axis
- designed to make repetitive movements

1.5 Disadvantages

- Complexity of the mechanism.
- Can't lift heavy weights (more than 200 grm).
- Only can lift round shaped materials.

1.6 Methodology

There are many kinds of robotic arms exists on the world of robotics such as two fingers, three fingers, five fingers etc. But we choose to design a microcontroller based three finger robotic arm. We program both manual and record mood. Having that record mood it can follow the same command again and again. This robotic arm purposely designed for medical and industrial sectors. It can pick and place round shaped objects under 200 grams.

1.7 Organization of the Report

This project report has six chapters in total. The first chapter describes an idea about our project “Robotic arm control with arduino”, Brief description of the project, problem statement, objects, advantages, disadvantages and methodology. The second chapter about system review, block diagram, circuit diagram, list of components. The third chapter is about component description, cost analysis of our system. The fourth chapter contains software analysis & program explanation. Then fifth chapter describes result & discussion properly. Finally, chapter six gives the concluding remarks and suggestion for the future works.

CHAPTER 2

SYSTEM REVIEWS

2.1 Introduction

This robotic arm is mainly build for picking and placing objects. The whole systems aim is to build a robotic arm with 3 servo motors, a dc motor, a microcontroller and some variable resistors. We build this robotic arm to make it useful for the industry and medical sectors.

2.2 General Block Diagram

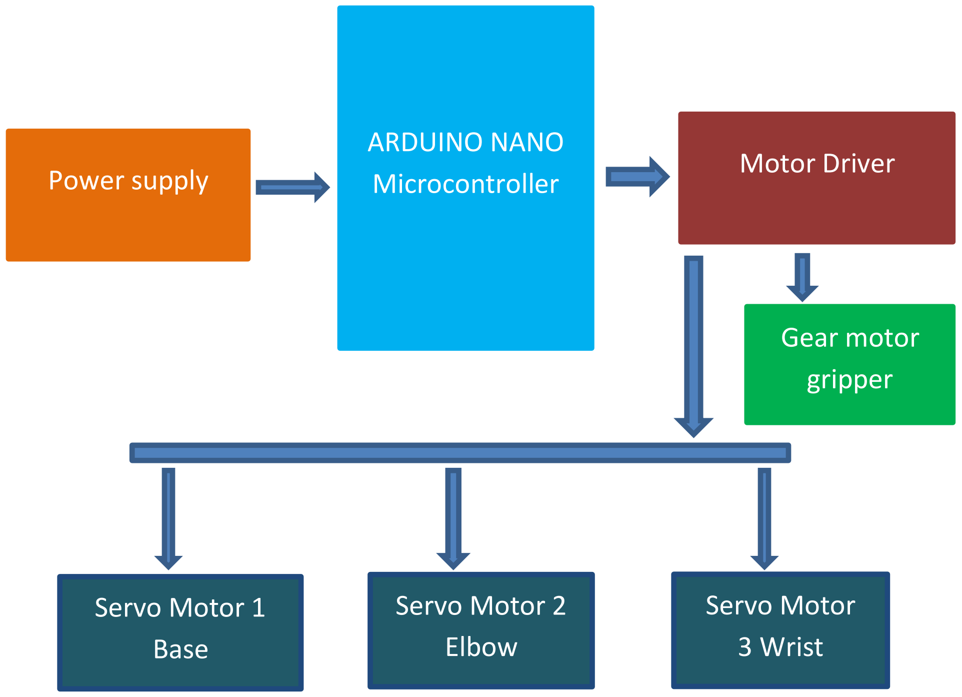


Fig 2.1: General Block Diagram

2.2.1 Block Diagram Description

In our system block is connected with multifunction equipment's. We mainly connect it with all the servos, LED's and gear motors. All servos are connected with arduino nano and as a power supply we connect a 11.1v battery with some variable resistances it also connects with the arduino nano. That's why the connecting process is called multifunctional process. Servo 1, servo 2 and servo 3 is connected with arduino nano also with the variable resistances that we show in our block diagram figure.

2.3 Circuit Diagram

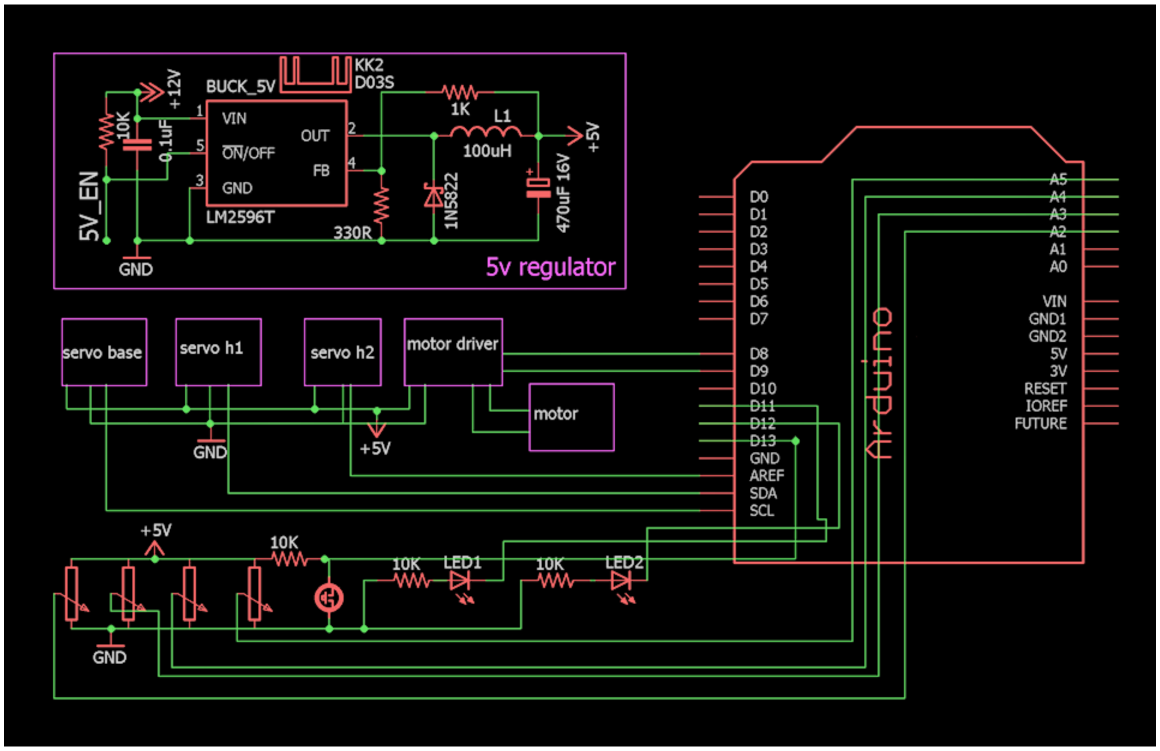


Fig 2.2: Circuit Diagram

2.3.1 Working Process

After connecting with power supply the robotic arm activated and then it can operate both manual and record mood. To operate it in manual mood we have 4 potentiometers. 1st one for controlling the base servo motor, 2nd one for servo 2 to control the elbow movements, 3rd one for servo 3 to control the wrist and the 4th one for the gear motor to control the fingers of the robotic arm.

To operate it in record mood, first we have to hold the push button until the red LED turns on. After the red LED turns on we can give 1st command by changing the angle or making any movements through the potentiometers. Then we press the push button and the green LED will be turn on, now we have to return the robotic arm to its initial position or initial angel. Now we have to press and hold the push button until the red and green LED's started to blink. When the two LED's stared to blink we have to release the push button. Then the robotic arm will repeatedly follow the command until we stop it manually. To stop the record mood we have to press and hold the push button, then the robotic arm will go to the initial position and stops.

2.4 List of Components used in Circuit

Table 2.1: List of components used in circuit

No	Component Name	Quantity	Used
01	Servo Motor MG995	**03**	For controlling Axis
02	Gear Motor	**02**	To control the arm fingers.
03	Arduino Nano V3.0	**02**	For controlling
04	Buck Converter LM2596	**04**	To step down the input voltage
05	Aluminium Sheet	**02**	To build the structure
06	Varo Boaard	**01**	To build Circuit board
07	Tiger lipo battery 1100mAh	**02**	To supply power
08	Battery connector	**02**	To connect the battery
09	Motor Driver L293D	**02**	To controlled amounts of resistance into electrical circuits

10	Push Button	**01**	To giving command
11	Female Header connector	**02**	To connect header connector
12	Resistor 10k, 330k	**02**	To controlled amounts of resistance into electrical circuits
13	Glue gun Stick	**01**	To attached the joints and devices with body structure.
14	LED	**02**	To detect signal
15	Cable Tie	**02**	To build the finger
16	Potentiometer 10k	**01**	To regulated voltage

CHAPTER 3
HARDWARE AND COMPONENT DESCRIPTION

3.1 Introduction

In our project we research and implement the information about the mechanical and software of the robotic arm. This robotic arm is designed to perform predetermined commands which is picking and place objects. The robotic arm can grab round shape objects, pick objects and can place it to the desired position. That's how the programming is. It also has a record mood. In recording mood, it can perform the recorded moves or recorded angles again and again until we manually stop it. The servomotor is preferred in order to be able to perform these operations properly since the motor to be selected must operate precisely and must be at high torque. We build the robotic arm with 3 servo motors so the robotic arm can move in 3 axis direction. We use an Arduino Nano microcontroller and write the program code in C language. A Buck converter used for step down the input voltage. And 4 potentiometers for controlling the robot arm. A 5V power supply is also preferred for the robot to work.

3.2 Arduino Nano V3.0

The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P (Arduino Nano3.0). It has more or less the same functionality of the Arduino Uno, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

The Arduino Software (IDE) is used to program Arduino Nano. The Arduino Software is an Integrated Development Environment that is common to all Arduino boards and runs both online and offline.

3.2.1 Specification

The detailed specification of the Arduino Nano board is as follows:

- Microcontroller ATmega328
- Architecture : AVR
- Operating Voltage (logic level): 5 V
- Input Voltage (Recommended): 7-12 V

- Input Voltage (limits): 6-20 V
- Digital I/O Pins : 14 (of which 6 provide PWM Output)
- Analog Input Pins: 8
- DC Current per I/O Pin: 40 mA
- Flash Memory 32 KB (ATmega328) of which 2 KB used by boot loader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz
- Power Consumption : 19 milliAmps
- PBC size : 18 x 45 mm
- Weight : 7 gms

3.2.2 Arduino Nano Pinout

This article discusses about the technical specs most importantly the pinout and functions of each and every pin in the Arduino Nano board.

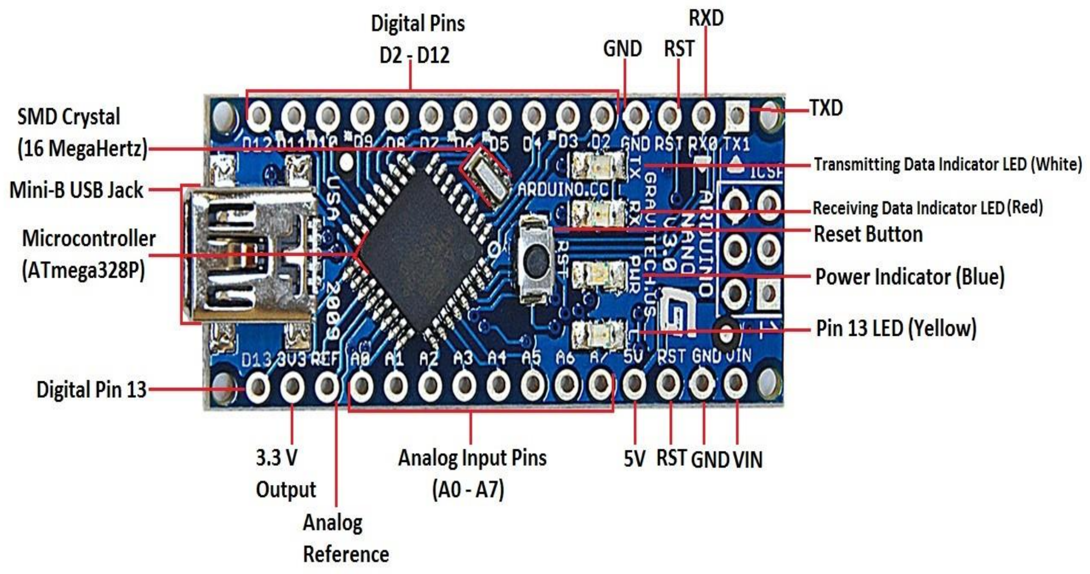


Fig 3.1: Arduino Nano Pinout

Digital I/O, PWM - 14 Pins
For Analog Functions - 9 Pins
Power - 7 Pins
SPI (Apart from Digital I/O Section) - 3 Pins
Reset - 3 Pins
Total=36 pins

3.2. 3 Arduino Nano – Pin Description

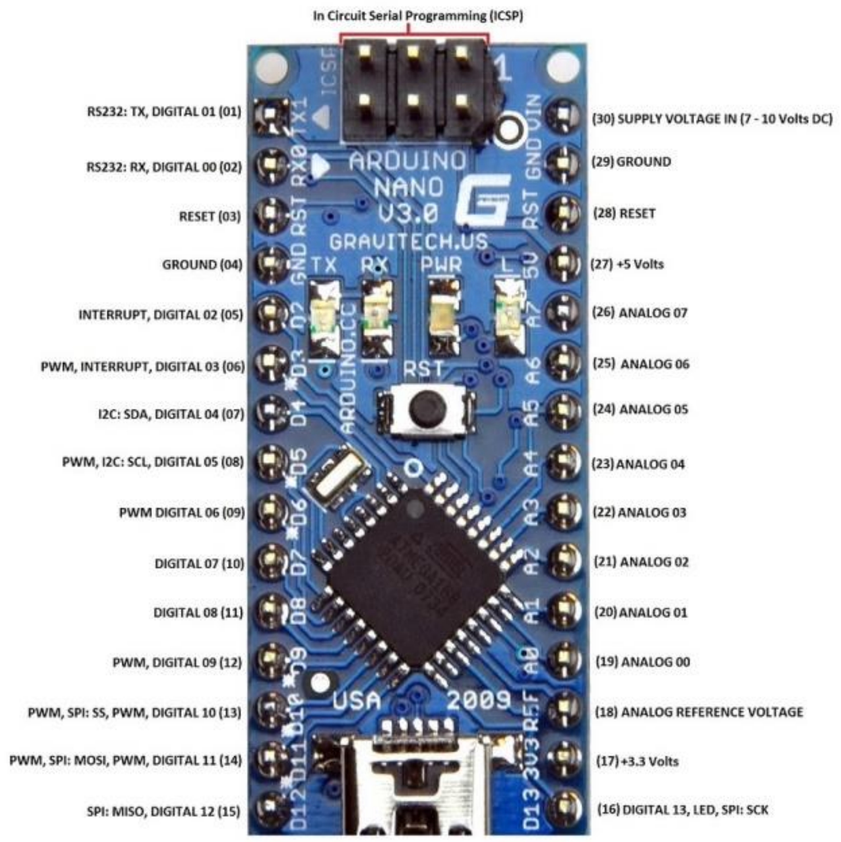

In Circuit Serial Programming (ICSP)
RS232: TX, DIGITAL 01 (01)
RS232: RX, DIGITAL 00 (02)
RESET (03)
GROUND (04)
INTERRUPT, DIGITAL 02 (05)
PWM, INTERRUPT, DIGITAL 03 (06)
I2C: SDA, DIGITAL 04 (07)
PWM, I2C: SCL, DIGITAL 05 (08)
PWM DIGITAL 06 (09)
DIGITAL 07 (10)
DIGITAL 08 (11)
PWM, DIGITAL 09 (12)
PWM, SPI: SS, PWM, DIGITAL 10 (13)
PWM, SPI: MOSI, PWM, DIGITAL 11 (14)
SPI: MISO, DIGITAL 12 (15)
(30) SUPPLY VOLTAGE IN (7 - 10 Volts DC)
(29) GROUND
(28) RESET
(27) +5 Volts
(26) ANALOG 07
(25) ANALOG 06
(24) ANALOG 05
(23) ANALOG 04
(22) ANALOG 03
(21) ANALOG 02
(20) ANALOG 01
(19) ANALOG 00
(18) ANALOG REFERENCE VOLTAGE
(17) +3.3 Volts
(16) DIGITAL 13, LED, SPI: SCK
ARDUINO NANO V3.0
GRAVITECH.US
USA 2009

Fig 3.2: Arduino Nano – Pin Description

Table 3.1: Arduino Nano – Pin Description (Pins 1 to 30)

Arduino Nano Pin	Pin Name	Type	Function
1	D1/TX	I/O	Digital I/O Pin

			Serial TX Pin
2	D0/RX	I/O	Digital I/O Pin Serial RX Pin
3	RESET	Input	Reset (Active Low)
4	GND	Power	Supply Ground
5	D2	I/O	Digital I/O Pin
6	D3	I/O	Digital I/O Pin
7	D4	I/O	Digital I/O Pin
8	D5	I/O	Digital I/O Pin
9	D6	I/O	Digital I/O Pin
10	D7	I/O	Digital I/O Pin
11	D8	I/O	Digital I/O Pin
12	D9	I/O	Digital I/O Pin
13	D10	I/O	Digital I/O Pin
14	D11	I/O	Digital I/O Pin
15	D12	I/O	Digital I/O Pin
16	D13	I/O	Digital I/O Pin
17	3V3	Output	+3.3V Output (from FTDI)
18	AREF	Input	ADC reference
19	A0	Input	Analog Input Channel 0
20	A1	Input	Analog Input Channel 1
21	A2	Input	Analog Input Channel 2
22	A3	Input	Analog Input Channel 3
23	A4	Input	Analog Input Channel 4
24	A5	Input	Analog Input Channel 5
25	A6	Input	Analog Input Channel 6
26	A7	Input	Analog Input Channel 7
27	+5V	Output or Input	+5V Output (From On-board Regulator) or +5V (Input from External Power Supply
28	RESET	Input	Reset (Active Low)
29	GND	Power	Supply Ground
30	VIN	Power	Supply voltage

Table 3.2: ICSP Pins

Arduino Nano ICSP Pin Name	Type	Function
MISO	Input or Output	Master In Slave Out
Vcc	Output	Supply Voltage
SCK	Output	Clock from Master to Slave
MOSI	Output or Input	Master Out Slave In
RST	Input	Reset (Active Low)
GND	Power	Supply Ground

3.2.4 Arduino Nano Digital Pins

Pins - 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, and 16

3.2. 5 Serial Communication Pins

Pins - 1, 2

1 - RX and 2 – TX

These two pins RX- receive and TX- transmit are used for TTL serial data communication. The pins RX and TX are connected to the corresponding pins of the USB-to-TTL Serial chip.

3.2. 6 PWM Pins

Pins - 6, 8, 9, 12, 13, and 14

3.2.7 External Interrupts

Pins - 5, 6

3.2.8 SPI Pins

Pins - 13, 14, 15, and 16

3.2.9 LED

Pin – 16

If you remember your first Arduino code, blinking LED, then you'll definitely came across this Pin16. The pin 16 is being connected to the blinking LED on the board.

3.2.10 Arduino Nano Analog Pins

Pins - 18, 19, 20, 21, 22, 23, 24, 25, and 26

3.2.11 I2C

Pins 23, 24 as A4 and A5

3.2.12 AREF

Pin 18
As mentioned already the AREF- Analog Reference pin is used as a reference voltage for analog input for the ADC conversion.

3.2.13 Reset

Pin 28

Reset pins in Arduino are active low pins which mean if we make this pins value as low i.e., 0v, it will reset the controller. Usually it used to be connected with switches to use as reset button.

3.2.14 ICSP

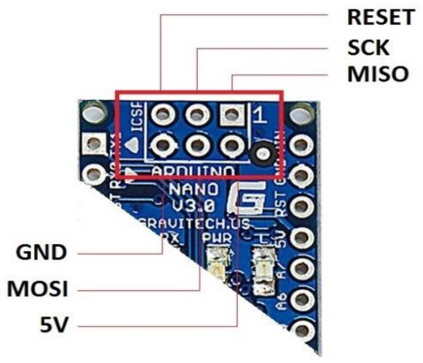


Fig 3.3: ICSP

We can use one Arduino to program another Arduino using this ICSP.

Table 3.3: One Arduino to program another Arduino using ICSP

Arduino as ISP	ATMega328
Vcc/5V	Vcc
GND	GND
MOSI/D11	D11
MISO/D12	D12
SCK/D13	D13
D10	Reset

3.2.15 Reset

Pins 3, 28 and 5 in ICSP

3.2.16 Power

Pins 4, 17, 27, 28, 30 and 2 & 6 in ICSP

3.3 Servo Motor

MG995 is a high-speed standard servo motor which uses PWM signals to control its angular position. Its functions are similar to the commonly used SG90 servo motor. It is sturdier as it uses metallic gears, compared to the SG90 which uses plastic components. It has an operating voltage range of 4.8 V to 7.2 V and has an operating speed of 0.2 s per 60 degrees.

Although the MG995 claims to have an angular range of only 120 degrees, practically it can cover an angular range of about 180 degrees without reaching its angular limit. A 50 Hz PWM signal is generally used to operate the MG995 and the angular position of the servo motor depends on the duty cycle of the PWM control signal. It uses a position sensor to continuously sense the current position, compares it to the required position specified by the PWM signal and makes an angular displacement if required.

3.3.1 Servo Wiring

It has 3 wires:

- Black for ground
- Orange for signal and
- Red for power

3.3.2 Specifications

Weight: 55g

Dimension: 40.7×19.7×42.9mm

Stall torque: 9.4kg/cm (4.8v); 11kg/cm (6v)

Operating speed: 0.20sec/60degree (4.8v); 0.16sec/60degree (6.0v)

Operating voltage: 4.8~ 6.6v

Gear Type: Metal gear

Temperature range: 0- 55deg

Servo wire length: 32cm

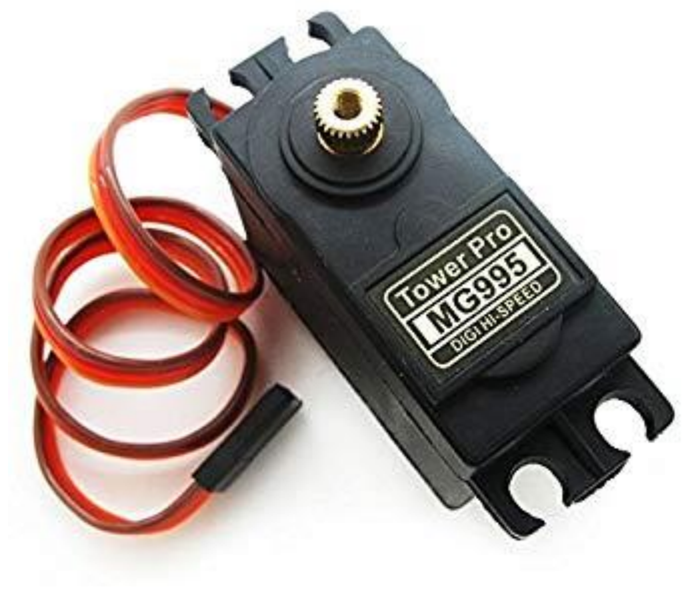


Fig 3.4: Servo motor

3.4 Gear motor

The dc motors are most easy to control .One dc motor will require only two dc signals for its operation , if we want to change the direction then we just need to change the polarity of the power across it . We can vary speed by varying the voltage across the motor by making use of gears.

The dc motor does not have enough torque to derive a robot directly by connecting wheels to it, gears increases the torque at the expense of the speed.

Fig 3.5: Gear Motor

3.5 Motor Driver

L293D is a dual H-bridge motor driver integrated circuit (IC). Motor drivers act as current amplifiers since they take a low-current control signal and provide a higher-current signal. This higher current signal is used to drive the motors. L293D contains two inbuilt H-bridge driver circuits. In its common mode of operation, two DC motors can be driven simultaneously, both in forward and reverse direction.

Fig 3.6: Motor Driver

3.5.1 Pin Diagram

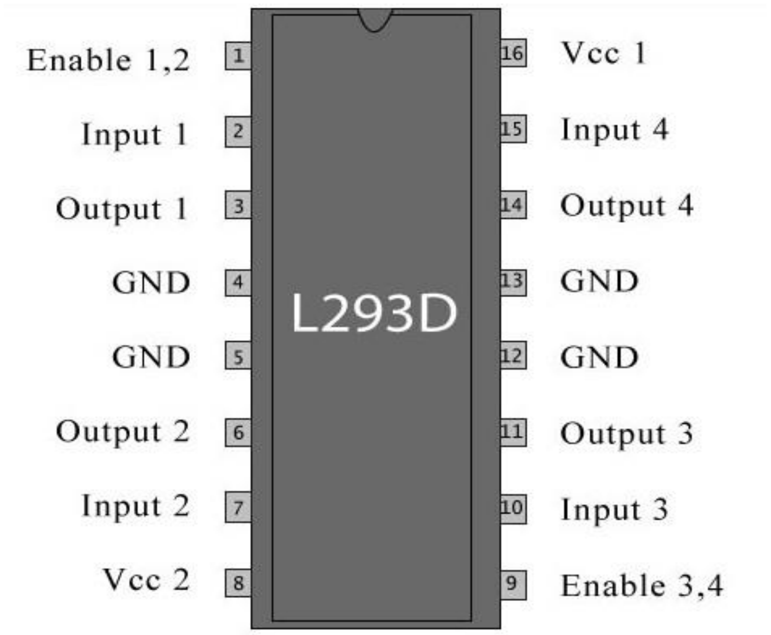


Fig 3.7: Pin diagram

3.5.2 Features

- Wide Supply-Voltage Range: 4.5 V to 36 V.
- Separate Input-Logic Supply.
- Internal ESD Protection.
- High-Noise-Immunity Inputs.
- Output Current 1A per Channel (600 mA for. L293D)
- Peak Output Current 2A per Channel (1.2 A for. L293D)
- Output Clamp Diodes for Inductive Transient. Suppression (L293D)

3.5.3 Pin Description

Table 3.4: Motor Driver Pin Description

Pin No.	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground

5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	Vcc $_2$
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	Vcc $_1$

3.6 Buck converter

The buck converter is a very simple type of DC-DC converter that produces an output voltage that is less than its input. The buck converter is so named because the inductor always “bucks” or acts against the input voltage.

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version.

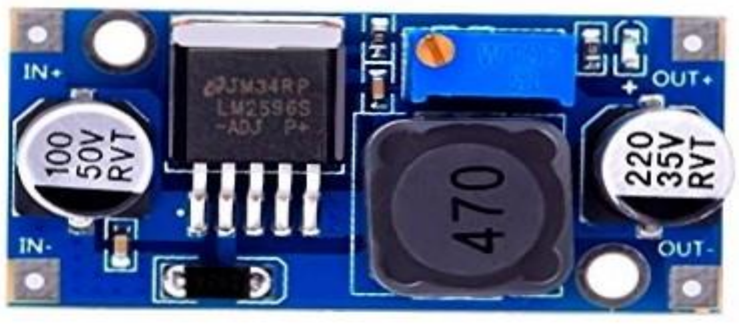


Fig 3.8: Buck converter

3.6.1 Features

- 3.3-V, 5-V, 12-V, and Adjustable Output Versions
- Adjustable Version Output Voltage Range: 1.2-V to 37-V ± 4% Maximum Over Line and Load Conditions
- Available in TO-220 and TO-263 Packages
- 3-A Output Load Current
- Input Voltage Range Up to 40 V
- Requires Only 4 External Components
- Excellent Line and Load Regulation Specifications
- 150-kHz Fixed-Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode, I_Q, Typically 80 µA
- High Efficiency
- Uses Readily Available Standard Inductors
- Thermal Shutdown and Current-Limit Protection

3.7 Aluminium sheet

Aluminium sheet is defined as cold-rolled material over 0.2mm thick but not exceeding 6mm thick. Aluminium is a silvery-white, lightweight metal. It is soft and malleable. Aluminium is used in a huge variety of products including cans, foils, kitchen utensils, window frames, beer kegs and aeroplane parts. This is because of its particular properties.

Fig. 3.9: Aluminium sheet

3.8 Push Button

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed.

3.8.1 Using

The "push-button" has been utilized in calculators, push-button telephones, kitchen appliances, and various other mechanical and electronic devices, home and commercial. In industrial and commercial applications, push buttons can be connected together by a mechanical linkage so that the act of pushing one button causes the other button to be released.

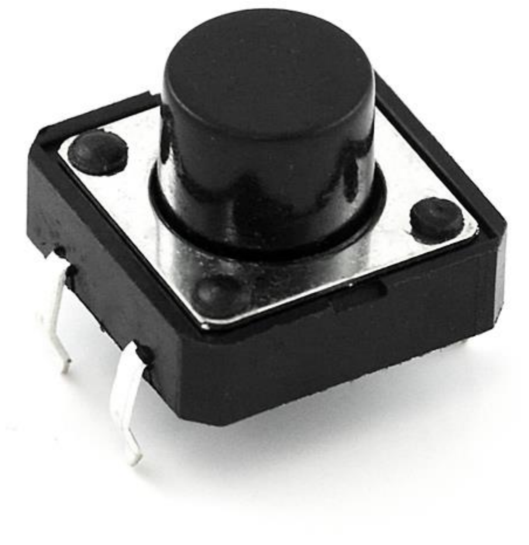


Fig. 3.10: Push button

3.9 Vero Board

Veroboard is a brand of stripboard, a pre-formed circuit board material of copper strips on an insulating bonded paper board which was originated and developed in the early 1960s by the Electronics Department of Vero Precision Engineering Ltd (VPE).

Veroboard) is designed primarily for hard wiring of discrete components, typically in analogue circuits, but is equally useful where a number of common bus or signal lines are required.

3.9.1 Features

- Turn circuit diagrams into functional projects easily
- Transfer your breadboard layouts on to a permanent base
- Specifically designed to make working with DIL or DIP ICs easy
- Simple strip layout makes it easy to build complex circuits
- Copper tracks are isolated from each other and can be cut to stop current flow

3.9.2 Specifications

- Material: Fiberglass FR4
- Surface: Copper PCB
- Hole Diameter: 1.50mm
- Hole Pitch: 2.54mm
- Board Size: 14.5 x 6.5cm
- Board Thickness: 1.20mm

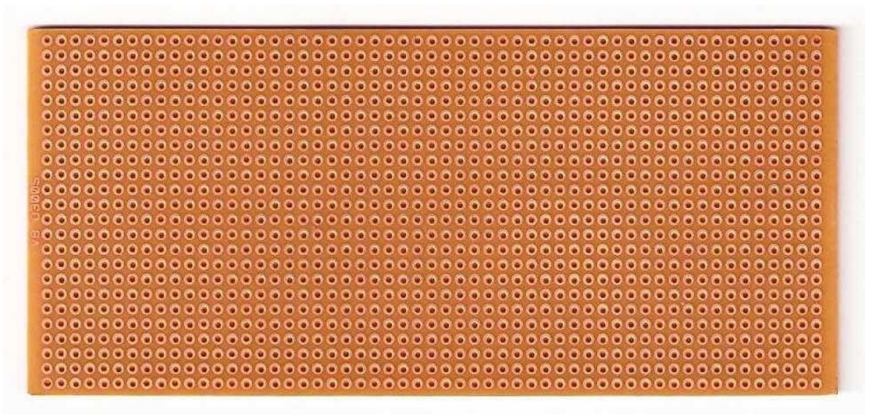


Fig. 3.11: Veroboard

3.10 Potentiometer

A potentiometer is a manually adjustable variable resistor with 3 terminals. Two terminals are connected to both ends of a resistive element, and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element. The position of the wiper determines the output voltage of the potentiometer. The potentiometer essentially functions as a variable voltage divider. The resistive element can be seen as two resistors in series (potentiometer resistance), where the wiper position determines the resistance ratio of the first resistor to the second resistor.
A potentiometer is also commonly known as a potmeter or pot.

3.10.1 Potentiometer Pin Configuration

Table 3.5: Potentiometer Pin Configuration

Pin No.	Pin Name	Description
1	Fixed End	This end is connected to one end of the resistive track
2	Variable End	This end is connected to the wiper, to provide variable voltage
3	Fixed End	This end is connected to another end of the resistive track

3.10.2 Features

- Type: Rotary a.k.a Radio POT
- Available in different resistance values like 500Ω, 1K, 2K, 5K, 10K, 22K, 47K, 50K, 100K, 220K, 470K, 500K, 1 M.
- Power Rating: 0.3W
- Maximum Input Voltage: 200Vdc
- Rotational Life: 2000K cycles

3.10.3 Applications

- Voltage and Current Control Circuits
- Used as volume control knobs in radios
- Tuning or controlling circuits
- Analog input control knobs

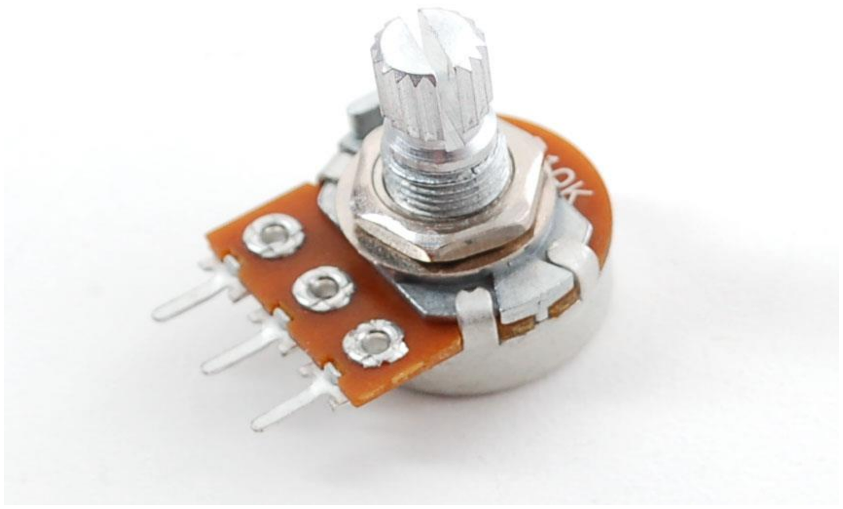


Fig. 3.12: Potentiometer

3.11 Female Header Connector

A range of quality machined PCB sockets in a single row format with 2.54mm (0.1 inch) pitch. Designed to match 0.1" male header's. Easily stackable to making multi row headers.

Can be placed end to end to make longer runs, but the gap between adjacent units will be slightly more than 0.1inch and will need trimming with a sharp knife to maintain 0.1in pitch.

3.11.1 Features

- Pin spacing: 1" (2.54mm)
- Number of pin: 15
- Pin length: 10.5 mm

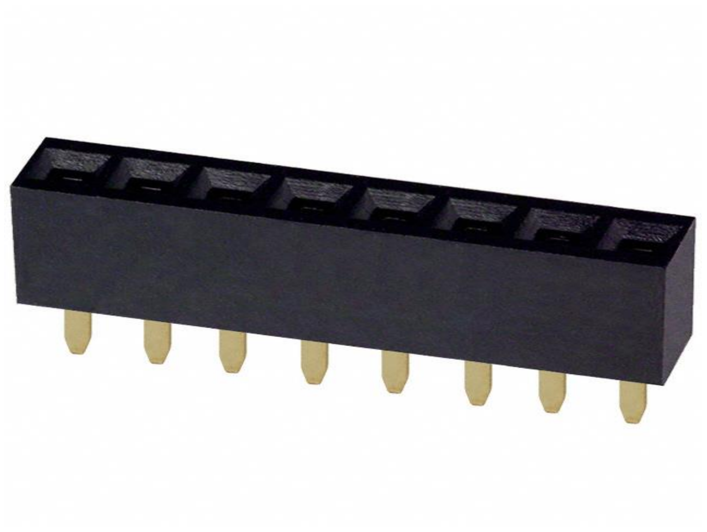


Fig. 3.13: Female Header Connector

3.12 Battery (11.1V)

This 11.1 volt lipo battery is perfect to run any the project. In robotics this kind of batteries are used now. It also has high energy density, platform time consistency is good, safe and reliable.

3.12.1 Specifications

- 11.1V lipo battery
- Perfect for Robotics
- Discharge C-rate 35C

3.12.2 Features

- High energy density
- High working voltage for single battery cells
- Pollution-free
- Long cycle life >500times
- No memory effect
- Capacity, resistance, Voltage, platform time consistency is good
- With short-circuit production function, safe and reliable
- Factory price& High quality
- Good consistency, low self-discharge
- Light weight, small and customized size

- Wide application for electronics and industry
- Environmentally compatible.

3.12.3 Product information

Item: Li-po 3S
Capacity: 1100mAh
Voltage: 11.1V
Continuous Discharge C-rate: 35
Max charging rate: 2C

Fig. 3.14: Battery (11.1V)

3.13 Cable Tie

A cable tie is a non-releasable double locking head. The cable tie secures cable and wires quickly without slipping. The double locking design of the cable tie results in a more rounded shape around the cable bundle.

3.13.1 Specifications

- Length: 8 Inch
- Width: 0.30 Inch
- Color: Black
- Material: Nylon 6/6 - General Purpose
- Min Tensile Strength 120 Lbs
- Weight (Lbs): 0.64

Fig. 3.15: Cable Tie

3.14 Cost Analysis

In this section we will show cost of our project that means cost sheet representation of our project.

3.15.1 Cost Sheet

Table 3.6: Cost sheet

No	Component Name	Quantity	Purchase Price (TK)
01	Servo Motor MG995	**03**	1500/-
02	Gear motor	**01**	500/-
03	ARDUINO NANO V3.0	**01**	365/-
04	Buck converter LM2596	**01**	296/-
05	Aluminium sheet	**01**	1200/-
06	Varo Board	**01**	45/-
07	Tiger LiPo Battery 1100mAh	**01**	1500/-
08	Battery connector	**02**	5/-
09	Motor Driver L293D	**01**	95/-
10	Push Button	**01**	2/-
11	Female Header Connector	**02**	20/-
12	Resistor 10k,220k,330k	**03**	30/-
13	Glue gun stick	**05**	15/-
14	LED	**02**	2/-
15	Cable tie	**03**	15/-
16	Potentiometer 10k	**04**	60/-
	Total		**= 5650/-**

CHAPTER 4

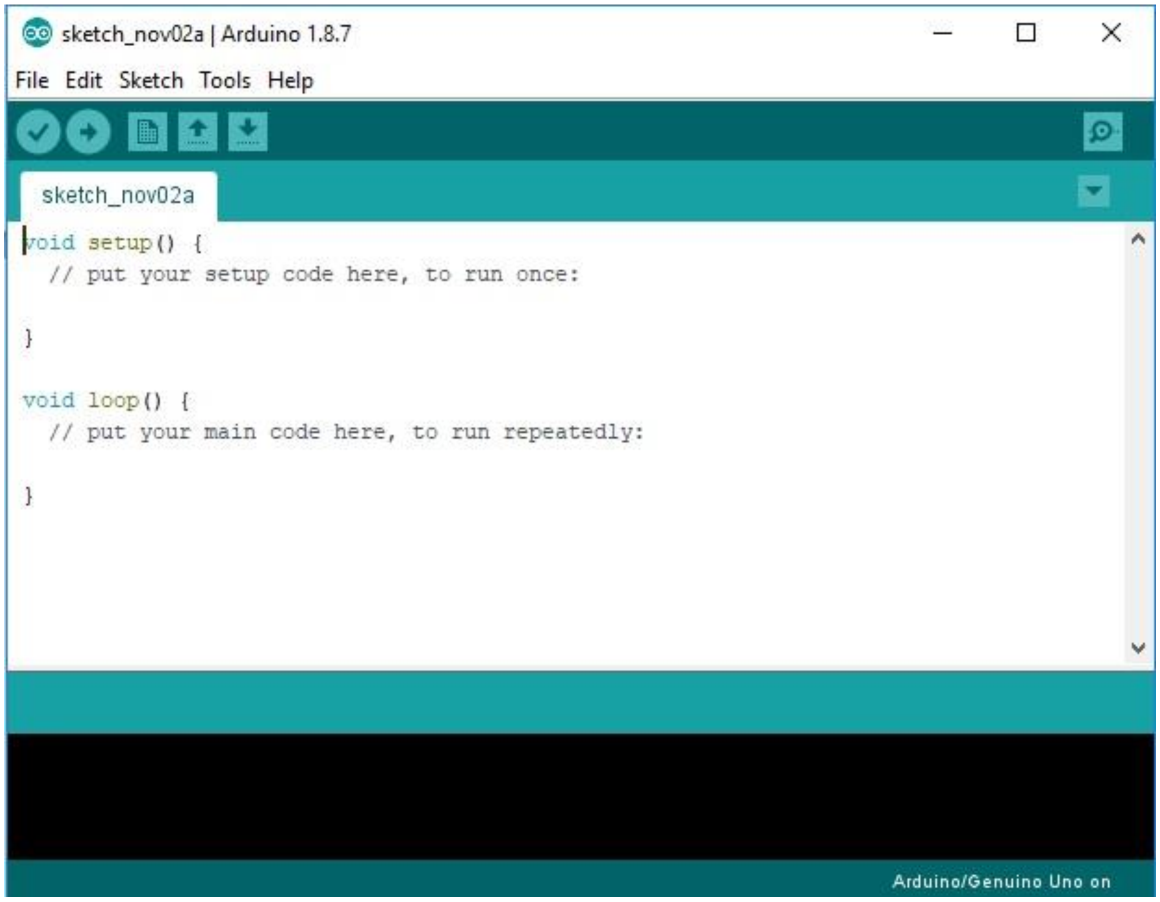
SOFTWARE ANALYSIS

4.1 Introduction

In this chapter we describe the software and the language of the program code. We also explained the program code dumping tools. We add the full code by using the compiled window and we also attached the flow chart of our project.

4.2 Description of our Software Compiler

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. The screen shot of ARDUINO 1.8.7 is shown below;



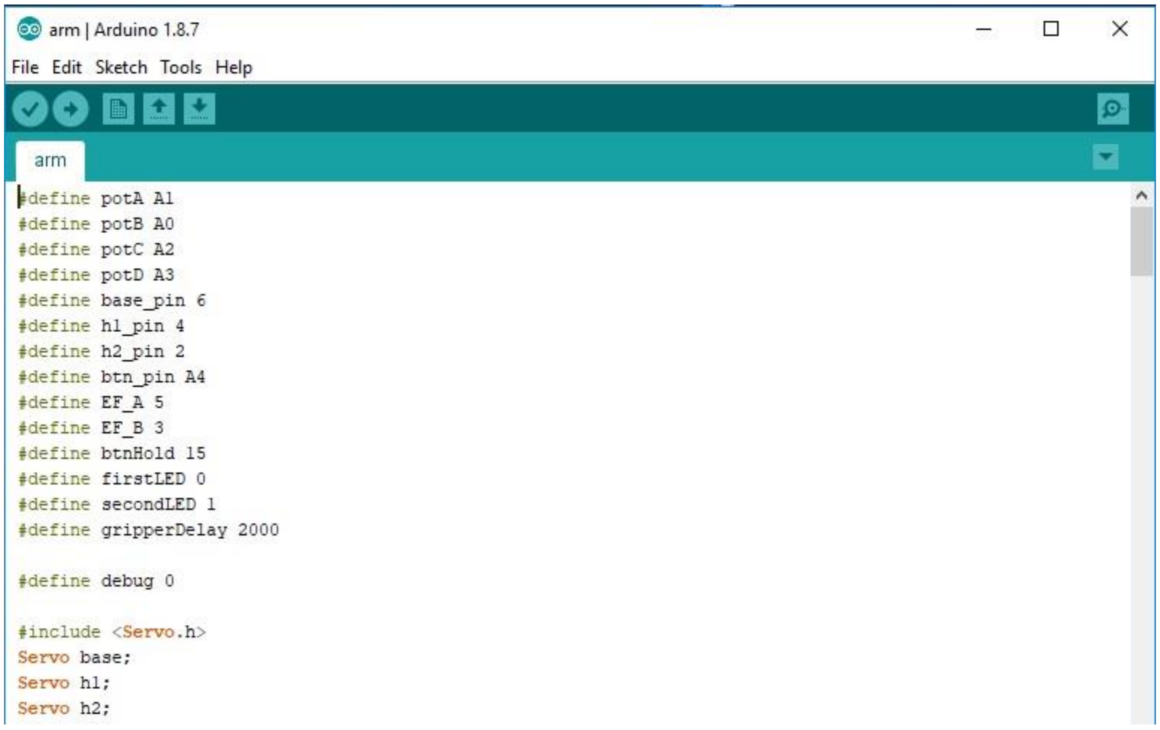
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig. 4.1: Software Platform

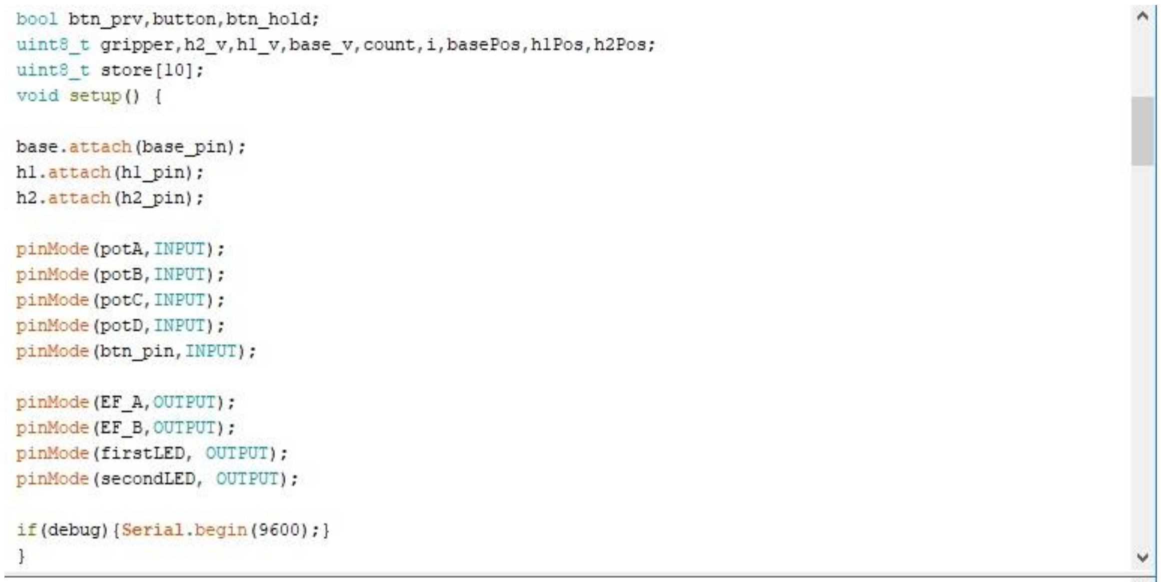
It is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit make files or run programs on a command-line interface. Although building on command-line is possible if required with some third-party tools such as Ion.

The Arduino Software (IDE) comes with a C/C++library called "Wiring" (from the project of the same name), which makes many common input/output operations much easier. Arduino Software (IDE) programs are written in C/C++, although users only need define two functions to make a runnable program:

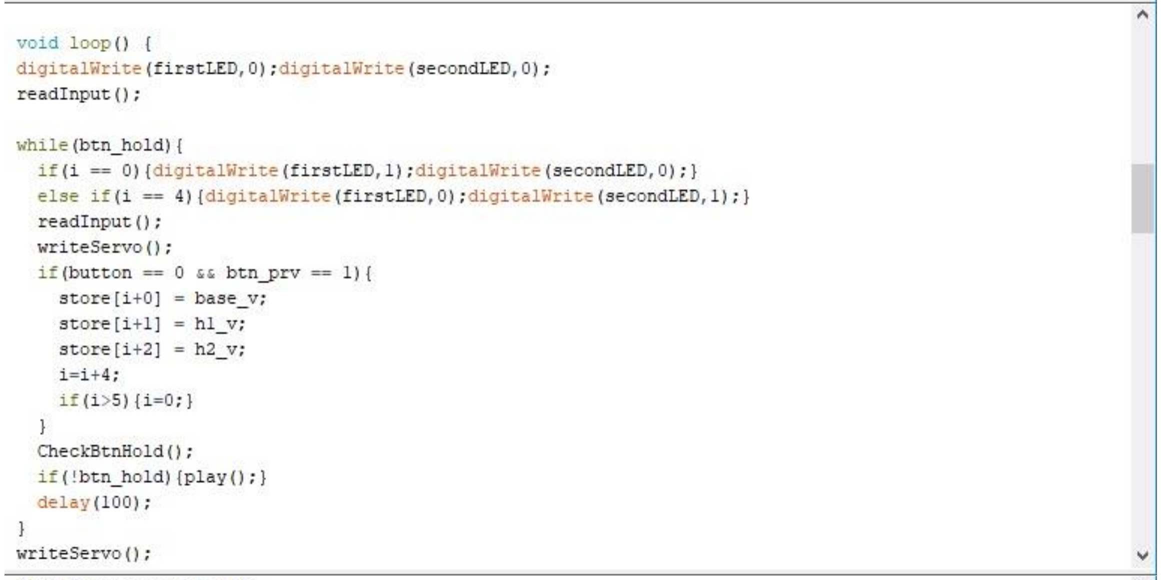
4.3 The compiled window of my code is shown below



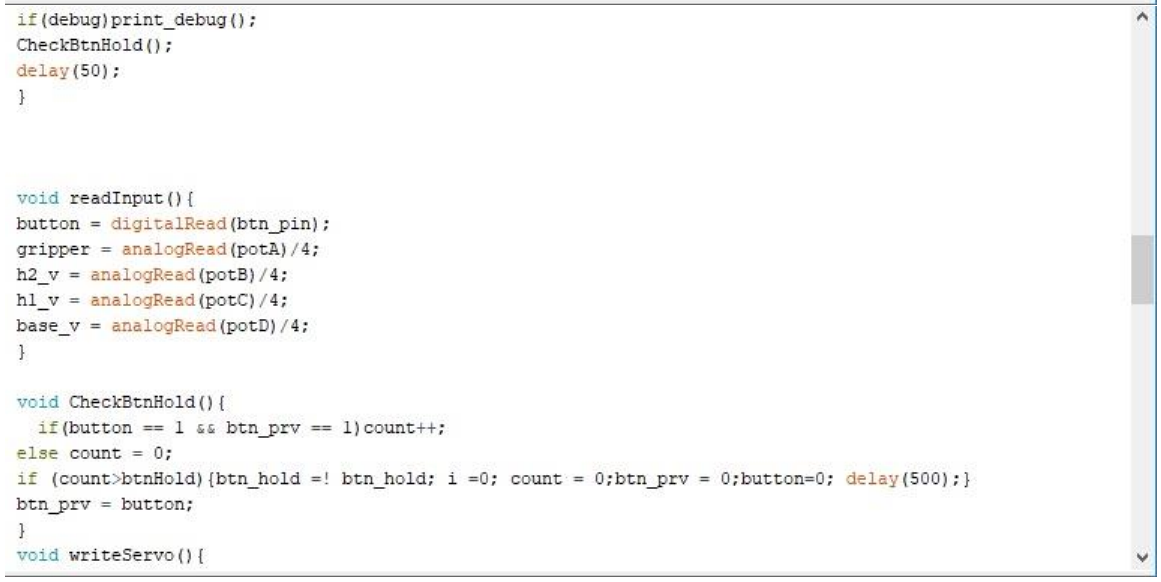
```
#define potA A1
#define potB A0
#define potC A2
#define potD A3
#define base_pin 6
#define h1_pin 4
#define h2_pin 2
#define btn_pin A4
#define EF_A 5
#define EF_B 3
#define btnHold 15
#define firstLED 0
#define secondLED 1
#define gripperDelay 2000

#define debug 0

#include <Servo.h>
Servo base;
Servo h1;
Servo h2;
```



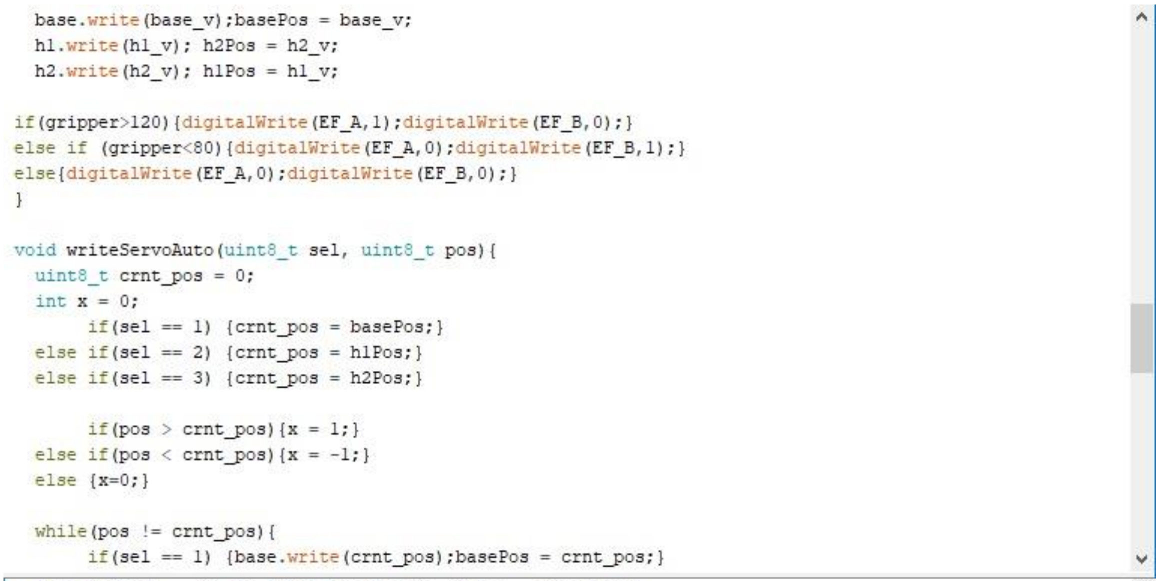
```
bool btn_prv,button,btn_hold;
uint8_t gripper,h2_v,h1_v,base_v,count,i,basePos,h1Pos,h2Pos;
uint8_t store[10];
void setup() {

base.attach(base_pin);
h1.attach(h1_pin);
h2.attach(h2_pin);

pinMode(potA,INPUT);
pinMode(potB,INPUT);
pinMode(potC,INPUT);
pinMode(potD,INPUT);
pinMode(btn_pin,INPUT);

pinMode(EF_A,OUTPUT);
pinMode(EF_B,OUTPUT);
pinMode(firstLED, OUTPUT);
pinMode(secondLED, OUTPUT);

if(debug){Serial.begin(9600);}
}
```



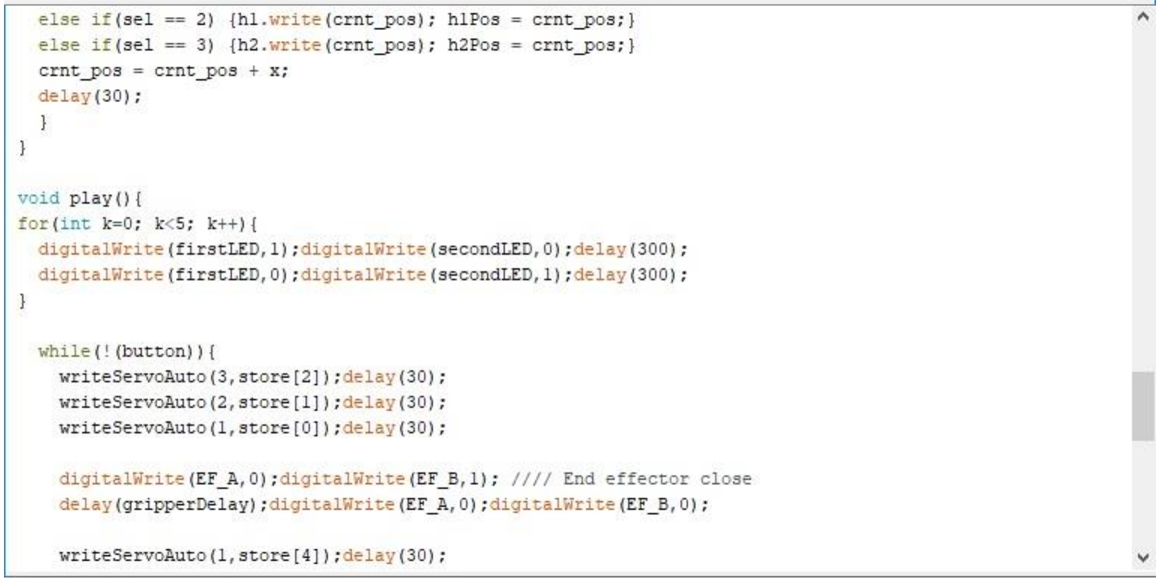
```
void loop() {
digitalWrite(firstLED,0);digitalWrite(secondLED,0);
readInput();

while(btn_hold){
  if(i == 0){digitalWrite(firstLED,1);digitalWrite(secondLED,0);}
  else if(i == 4){digitalWrite(firstLED,0);digitalWrite(secondLED,1);}
  readInput();
  writeServo();
  if(button == 0 && btn_prv == 1){
    store[i+0] = base_v;
    store[i+1] = h1_v;
    store[i+2] = h2_v;
    i=i+4;
    if(i>5){i=0;}
  }
  CheckBtnHold();
  if(!btn_hold){play();}
  delay(100);
}
writeServo();
```



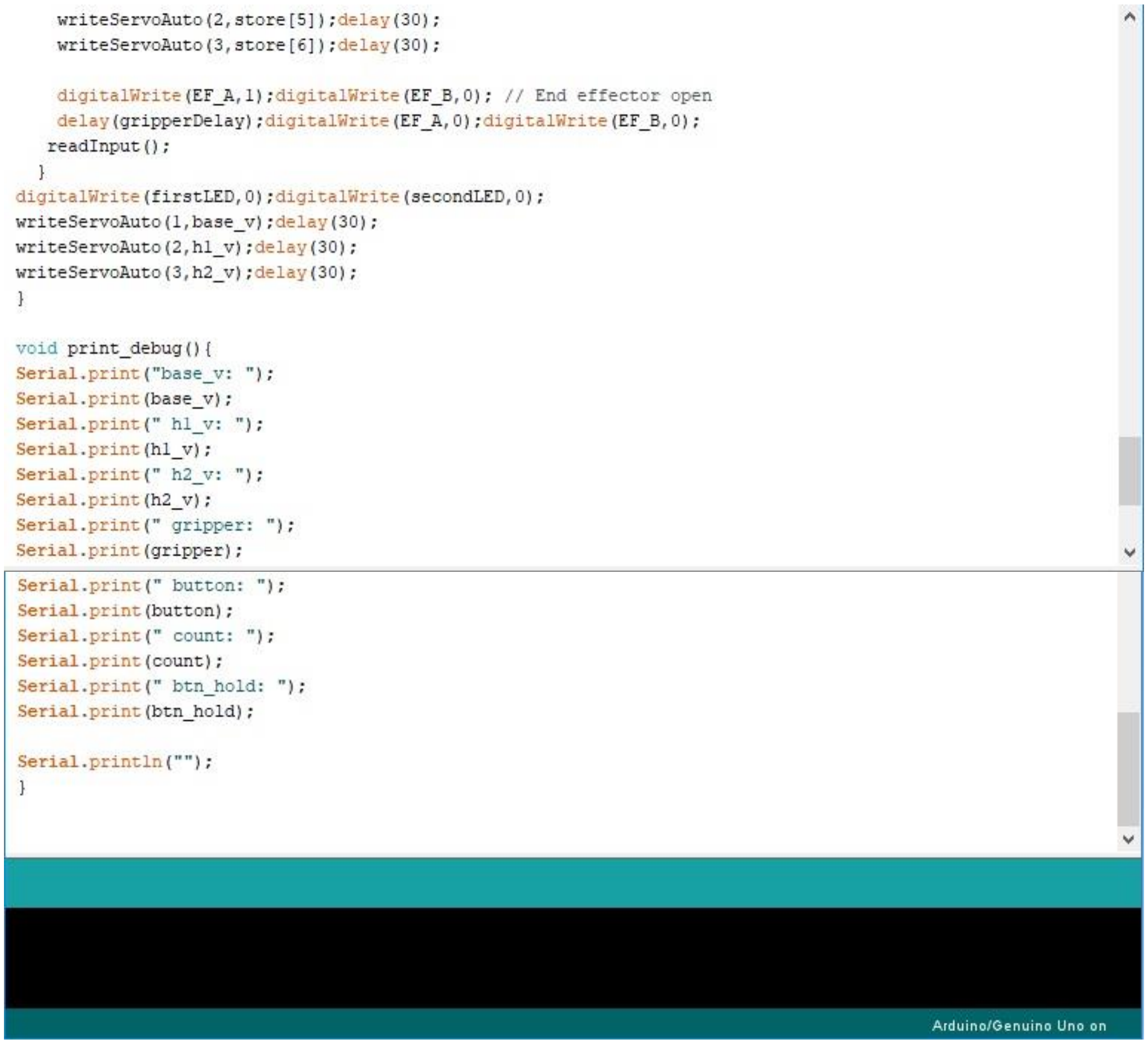
```
if(debug)print_debug();
CheckBtnHold();
delay(50);
}



void readInput(){
button = digitalRead(btn_pin);
gripper = analogRead(potA)/4;
h2_v = analogRead(potB)/4;
h1_v = analogRead(potC)/4;
base_v = analogRead(potD)/4;
}

void CheckBtnHold(){
  if(button == 1 && btn_prv == 1)count++;
else count = 0;
if (count>btnHold){btn_hold =! btn_hold; i =0; count = 0;btn_prv = 0;button=0; delay(500);}
btn_prv = button;
}
void writeServo(){
```



```
  base.write(base_v);basePos = base_v;
  h1.write(h1_v); h2Pos = h2_v;
  h2.write(h2_v); h1Pos = h1_v;

if(gripper>120){digitalWrite(EF_A,1);digitalWrite(EF_B,0);}
else if (gripper<80){digitalWrite(EF_A,0);digitalWrite(EF_B,1);}
else{digitalWrite(EF_A,0);digitalWrite(EF_B,0);}
}

void writeServoAuto(uint8_t sel, uint8_t pos){
  uint8_t crnt_pos = 0;
  int x = 0;
       if(sel == 1) {crnt_pos = basePos;}
  else if(sel == 2) {crnt_pos = h1Pos;}
  else if(sel == 3) {crnt_pos = h2Pos;}

       if(pos > crnt_pos){x = 1;}
  else if(pos < crnt_pos){x = -1;}
  else {x=0;}

  while(pos != crnt_pos){
       if(sel == 1) {base.write(crnt_pos);basePos = crnt_pos;}
```



```
  else if(sel == 2) {h1.write(crnt_pos); h1Pos = crnt_pos;}
  else if(sel == 3) {h2.write(crnt_pos); h2Pos = crnt_pos;}
  crnt_pos = crnt_pos + x;
  delay(30);
  }
}

void play(){
for(int k=0; k<5; k++){
  digitalWrite(firstLED,1);digitalWrite(secondLED,0);delay(300);
  digitalWrite(firstLED,0);digitalWrite(secondLED,1);delay(300);
}

  while(!(button)){
    writeServoAuto(3,store[2]);delay(30);
    writeServoAuto(2,store[1]);delay(30);
    writeServoAuto(1,store[0]);delay(30);

    digitalWrite(EF_A,0);digitalWrite(EF_B,1); //// End effector close
    delay(gripperDelay);digitalWrite(EF_A,0);digitalWrite(EF_B,0);

    writeServoAuto(1,store[4]);delay(30);
```



```
    writeServoAuto(2,store[5]);delay(30);
    writeServoAuto(3,store[6]);delay(30);

    digitalWrite(EF_A,1);digitalWrite(EF_B,0); // End effector open
    delay(gripperDelay);digitalWrite(EF_A,0);digitalWrite(EF_B,0);
   readInput();
  }
digitalWrite(firstLED,0);digitalWrite(secondLED,0);
writeServoAuto(1,base_v);delay(30);
writeServoAuto(2,h1_v);delay(30);
writeServoAuto(3,h2_v);delay(30);
}

void print_debug(){
Serial.print("base_v: ");
Serial.print(base_v);
Serial.print(" h1_v: ");
Serial.print(h1_v);
Serial.print(" h2_v: ");
Serial.print(h2_v);
Serial.print(" gripper: ");
Serial.print(gripper);
Serial.print(" button: ");
Serial.print(button);
Serial.print(" count: ");
Serial.print(count);
Serial.print(" btn_hold: ");
Serial.print(btn_hold);

Serial.println("");
}
```

4.4 Flow Chart of Diagram

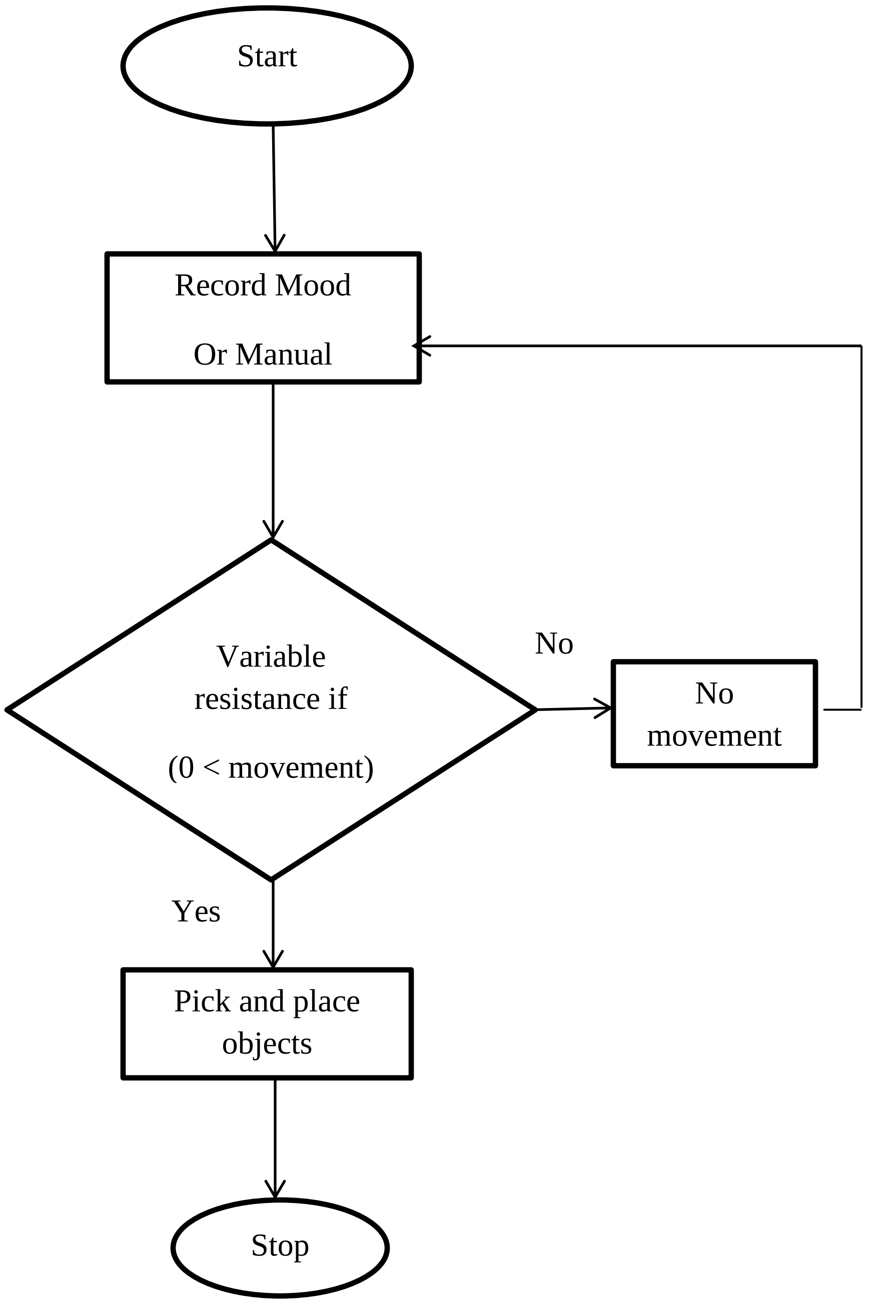


Fig. 4.2: Flow chart of our system.

CHAPTER 5

RESULT AND DISCUSSIONS

5.1 Robotic Arm Movement Coverage

The maximum range for the robotic arm is recorded during the experiment and shown in the figure below. The further pick up point of the robotic arm is 21.0cm, and the maximum angle the robotic arm can reach is 180 degree, with range from 0 to 180 degree.

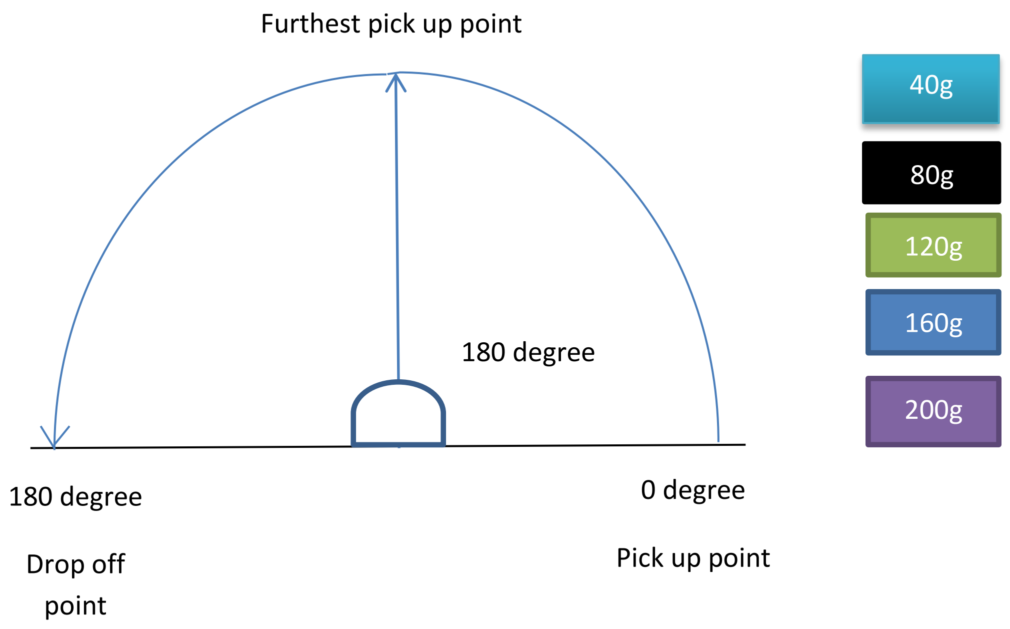


Fig. 5.1: Pick up and Drop off point

5.2 Result

Result of the robotic arm of lifting with different weight is presented in this section. The load to be lifted in this experiment is a sand bag with different weight. The robotic arm is commanded to lift the sand bag and place it to our desired position. We examine the accuracy by using different weight of a sand bag which is in the range of 40 gm’s to 200 gm's, the 40 gm weight is a reference load. From the data obtained, our robotic arm can pick 200 grams as expected result in this project. However, the movement of robotic arm is not smooth when it lifted 200 grams.

A complete lifting cycle can be done at 7.25 seconds and there is an insignificant difference of time taken for various weights.

Table 5.1: Duration of lifting process

Weight (grams)	Duration (second)
No load	7.25
40	7.25
80	7.29
120	7.34
160	7.39
200	7.42

CHAPTER 6

CONCLUSION

6.1 Conclusion

The main object of our project has been achieved by developing the structure and software for the Arduino control robotic arm. From observation, it’s clear now that the robotic arms movement is accurate, precise, easy to control and user friendly. The robotic arm overcome the previous problems, now it can pick and place hazardous objects very smoothly and precisely.

6.2 Future Scope

- In the medicine sector and industrial sector this robotic arm can be helpful for the repetitive tasks
- By some few modifications it can be useful to do hazardous tasks and decreases human disasters.
- The accuracy of picking and placing object will be the main future scope of our project.

REFERENCES

[1]
https://www.researchgate.net/publication/317277584_ROBOT_ARM_CONTROL_WITH_ARDUINO

[2] http://www.circuitstoday.com/arduino-nano-tutorial-pinout-schematics

[3] https://openlabpro.com/guide/mg995-servo-motor-interfacing-with-pic18f4550/

[4] https://www.towerpro.com.tw/product/mg995/

[4] http://www.resistorguide.com/potentiometer/

[5] https://components101.com/potentiometer

[6] http://www.hobbytronics.co.uk/header-female-1-10

[7] https://www.shenzhen2u.com/Long-Legs-Female-Header%20-Single-Row

[8] https://en.wikipedia.org/wiki/Resistor

[9] http://bdspeedytech.com/index.php?route=product/product&product_id=1704

[10]
https://www.researchgate.net/publication/289893486_Design_and_Development_of_a_Mechanism_of_Robotic_Arm_for_Lifting_Part1

[11]
https://www.researchgate.net/publication/317277584_ROBOT_ARM_CONTROL_WITH_ARDUINO

APPENDIX A

```
#define potA A1
#define potB A0
#define potC A2
#define potD A3
#define base_pin 6
#define h1_pin 4
#define h2_pin 2
#define btn_pin A4
#define EF_A 5
#define EF_B 3
#define btnHold 15
#define firstLED 0
#define secondLED 1
#define gripperDelay 2000

#define debug 0

#include <Servo.h>
Servo base;
Servo h1;
Servo h2;

bool btn_prv,button,btn_hold;
uint8_t gripper,h2_v,h1_v,base_v,count,i,basePos,h1Pos,h2Pos;
uint8_t store[10];
void setup() {

base.attach(base_pin);
h1.attach(h1_pin);
h2.attach(h2_pin);

pinMode(potA,INPUT);
pinMode(potB,INPUT);
pinMode(potC,INPUT);
pinMode(potD,INPUT);
pinMode(btn_pin,INPUT);

pinMode(EF_A,OUTPUT);
pinMode(EF_B,OUTPUT);
pinMode(firstLED, OUTPUT);
pinMode(secondLED, OUTPUT);

if(debug){Serial.begin(9600);}
}
```

```
void loop() {
digitalWrite(firstLED,0);digitalWrite(secondLED,0);
readInput();

while(btn_hold){
  if(i == 0){digitalWrite(firstLED,1);digitalWrite(secondLED,0);}
  else if(i == 4){digitalWrite(firstLED,0);digitalWrite(secondLED,1);}
  readInput();
  writeServo();
  if(button == 0 && btn_prv == 1){
    store[i+0] = base_v;
    store[i+1] = h1_v;
    store[i+2] = h2_v;
    i=i+4;
    if(i>5){i=0;}
  }
  CheckBtnHold();
  if(!btn_hold){play();}
  delay(100);
}
writeServo();
if(debug)print_debug();
CheckBtnHold();
delay(50);
}



void readInput(){
button = digitalRead(btn_pin);
gripper = analogRead(potA)/4;
h2_v = analogRead(potB)/4;
h1_v = analogRead(potC)/4;
base_v = analogRead(potD)/4;
}

void CheckBtnHold(){
  if(button == 1 && btn_prv == 1)count++;
else count = 0;
if  (count>btnHold){btn_hold  =!  btn_hold;  i  =0;  count  =  0;btn_prv  =  0;button=0;
delay(500);}
btn_prv = button;
}
void writeServo(){
  base.write(base_v);basePos = base_v;
  h1.write(h1_v); h2Pos = h2_v;
  h2.write(h2_v); h1Pos = h1_v;

if(gripper>120){digitalWrite(EF_A,1);digitalWrite(EF_B,0);}
else if (gripper<80){digitalWrite(EF_A,0);digitalWrite(EF_B,1);}
else{digitalWrite(EF_A,0);digitalWrite(EF_B,0);}
}

void writeServoAuto(uint8_t sel, uint8_t pos){
  uint8_t crnt_pos = 0;
  int x = 0;
       if(sel == 1) {crnt_pos = basePos;}
  else if(sel == 2) {crnt_pos = h1Pos;}
  else if(sel == 3) {crnt_pos = h2Pos;}

       if(pos > crnt_pos){x = 1;}
  else if(pos < crnt_pos){x = -1;}
  else {x=0;}

  while(pos != crnt_pos){
       if(sel == 1) {base.write(crnt_pos);basePos = crnt_pos;}
  else if(sel == 2) {h1.write(crnt_pos); h1Pos = crnt_pos;}
  else if(sel == 3) {h2.write(crnt_pos); h2Pos = crnt_pos;}
  crnt_pos = crnt_pos + x;
  delay(30);
  }
}

void play(){
for(int k=0; k<5; k++){
  digitalWrite(firstLED,1);digitalWrite(secondLED,0);delay(300);
  digitalWrite(firstLED,0);digitalWrite(secondLED,1);delay(300);
}

  while(!(button)){
    writeServoAuto(3,store[2]);delay(30);
    writeServoAuto(2,store[1]);delay(30);
    writeServoAuto(1,store[0]);delay(30);

    digitalWrite(EF_A,0);digitalWrite(EF_B,1); //// End effector close
    delay(gripperDelay);digitalWrite(EF_A,0);digitalWrite(EF_B,0);

    writeServoAuto(1,store[4]);delay(30);
    writeServoAuto(2,store[5]);delay(30);
    writeServoAuto(3,store[6]);delay(30);
    digitalWrite(EF_A,1);digitalWrite(EF_B,0); // End effector open
    delay(gripperDelay);digitalWrite(EF_A,0);digitalWrite(EF_B,0);
   readInput();
  }
digitalWrite(firstLED,0);digitalWrite(secondLED,0);
writeServoAuto(1,base_v);delay(30);
writeServoAuto(2,h1_v);delay(30);
writeServoAuto(3,h2_v);delay(30);
}

void print_debug(){
Serial.print("base_v: ");
Serial.print(base_v);
Serial.print(" h1_v: ");
Serial.print(h1_v);
Serial.print(" h2_v: ");
Serial.print(h2_v);
Serial.print(" gripper: ");
Serial.print(gripper);
Serial.print(" button: ");
Serial.print(button);
Serial.print(" count: ");
Serial.print(count);
Serial.print(" btn_hold: ");
Serial.print(btn_hold);

Serial.println("");
}
```