PROJECT PHASE-II REPORT

AUTOMATIC PICK AND PLACE ROBOT

SUBMITTED IN PARTIAL FULFILLMENT OF THE DEGREE OF

Bachelor of Technology

ELECTRONICS AND COMMUNICATION ENGINEERING

ECD 416 : PROJECT PHASE-II

BY

ABHAYAJITH S (PKD19EC002)

KARTHIKEYAN C R (LPKD19EC066)

ROHITH K P (WYD19EC076)

VIVIN K V (LPKD19EC069)

under the guidance of

Dr. BINDU P

Head of the Dept.of Electronics and Communication Engineering

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GOVERNMENT ENGINEERING COLLEGE SREEKRISHNAPURAM

MAY 2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

GOVERNMENT ENGINEERING COLLEGE PALAKKAD

CERTIFICATE

26 May, 2023

Certified that this is a bonafide record of the project report of **ABHAYAJITH (PKD19EC 002), ROHITH KP (WYD19EC076), KARTHIKEYAN CR (LPKD19EC066), VIVIN KV (LPKD19EC069)** for Project Phase II of the project **"AUTOMATIC PICK AND PLACE ROBOT"** in partial fulfillment of the requirements for the B.Tech. Degree in Electronics and Communication Engineering under APJ AbdulKalam Kerala Technological University during the year 2022-2023.

Dr.Bindu.P	**Prof.Rani L**
Project Guide	**Project Coordinator**

Dr.Bindu.P.Palakkal

Head of the Department

Dept. of Electronics and Communication Engineering

ACKNOWLEDGEMENT

First of all we thank the Almighty, for giving us the strength ,courage and knowledge to do this project successfully.

We express our heartfelt gratitude to the Principal Dr. P C Reghu Raj, Govt. Engineering College, Sreekrishnapuram, for his inspiration throughout the project.

We are extremely grateful to thank Dr. Bindu P , Head Of the Department, Electronics and Communication Engineering, Govt. Engineering College, Sreekrishnapuram for providing permission and availing all required facilities for undertaking the project in a systematic way and also who guided us with kind, ordinal, valuable suggestions.

We also appreciate the guidance given by Assoc.Prof.M Anwar Hussain and Asst.Prof.Rani L our project Coordinators, Department of Electronics and Communication Engineering, Govt. Engineering College, Sreekrishnapuram, for providing sincere guidance.

We would also like to express our sincere gratitude to all teaching and non-teaching staffs of Department of Electronics and Communication Engineering, Govt. Engineering College, Sreekrishnapuram, for the sincere directions imparted and the cooperation in connection with the project.

We are also thankful to our parents for the support given in connection with the project. Gratitude may be extended to all well-wishers and our friends who supported us for the project.

ABSTRACT

In recent years the industry and daily routine works are found to be more attracted and implemented through automation via Robots.The Pick and place robots are widely used in manufacturing, warehousing, and logistics to improve efficiency and reduce the need for manual labor. The system is so designed that it can perform pick and place operations in manual and automated manner. It eliminates the human error and human intervention to get more precise work. This robot typically consists of a robotic arm or gripper, and a means of movement, such as wheels or tracks. When activated, the robot uses its robotic arm to pick up the objects and move them to the designated locations.

Contents

List of Figures

Chapter 1

INTRODUCTION

Pick and place robots enable companies to use automated solutions for lifting objects from one location and placing them at other locations. Simple tasks such as lifting objects or moving them do not require a lot of thought processes. Therefore, using human workers on these tasks can be wasteful, as the workforce can be used for other tasks that require higher mental abilities. These repetitive tasks are handled by pick and place robots. One of the key benefits of pick and place robots is their ability to work quickly and accurately, reducing the need for manual labor and increasing efficiency. They are also able to operate continuously without the need for breaks, making them ideal for use in fast-paced production environments. If humans are doing the repeated tasks like relocating the products again and again , there is higher possibilities of error and inefficiency. Also they become tired and bored while doing the repeated tasks. We can eliminate such problems by implementing this rorbot. There are several motivations for implementing this project for manufacturing and logistics operations. These include:

- **Increased efficiency:** Pick and place robots can work continuously without the need for breaks, and they can handle tasks quickly and accurately, reducing the need for manual labor and increasing efficiency.
- **Improved accuracy:** Pick and place robots are able to handle tasks with a high level of accuracy and repeatability, reducing the risk of mistakes and errors that can occur with manual handling.

- **Cost savings:** While the initial cost of purchasing and installing a pick and place robot can be significant, the long-term cost savings and increased productivity that they provide can more than offset this investment.
- **Increased flexibility:** Pick and place robots can be programmed to perform a wide range of tasks and can be easily reconfigured to handle different types of objects, making them highly flexible and adaptable to changing needs.

This robot will contribute towards the automation of industries based on these motivations. The next chapter 2 will discuss about the objective of this project.

Chapter 2

OBJECTIVE

The objective of this project is to design and implement a pick and place robot to pick and move an object from one location to another. The Dimensions of robotic arm is about 16cm (h)x 15cm(l) and Dimensions of base is about 30cm (l)x 15cm (w)x 8cm (h). This project aims to build the robot which has maximum finger opening upto 6cm ,so the robot can pick/place any object of dimension less than 8cm and can carry maximum weight up to 100gm. It should also be able to move upto 10m from one place to another using the supporting wheels. The main goal of this project is to make the robot to do all the actions repeatedly and automatically if once we control it to do so.

Chapter 3

LITERATURE REVIEW

3.1 Design and Implementation of Pick and Place Robotic Arm

The idea of this project is from the paper reference no [1],[2],[3] & [4] which describes how to design and implement a six DOF (Degree of Freedom) pick and place robotic arm using CAD tool and 3D printing the model created from CAD software tool. The reference paper [1] and [2], presents the design and development of 5-Degree of Freedom (DOF) robotic arm which is used for feeding the elderly or specially challenged people.

The papers [3] and [4] gives the ideas about the design and materials that can be used for making the robotic arm. The paper no [3] describes the use of aluminium sheets and paper no [4] describes the use of strong fibers for the making of arm.

3.2 Wireless Mobile Robotic Arm

The robot could be controlled by both a joystick and a mobile app. After further research, a conclusion was made that controlling the robot using the mobile app is more efficient. The reference paper no [4],[5] & [6] gives the idea about to control the robotic arm using a mobile app. The papers [4] talks about the development of a wireless robot arm. A robot that functional to do pick and place operation and be controlled by using wireless PS2 controller. It can move forward, reverse, turn right and left for a specific distance according to the controller specification. The paper [5] and [6] presents the use of mobile application instead of PS2 controller.

3.3 Fabrication of chasis with meccanum wheels

The robotic arm could be mobilized using a chassis coupled with mecanum wheels. The wheels will be powered by stepper motors. From reference paper nos [6] & [7] describes how to control the stepper motors on chassis and the robotic arm by a single micro controller. In normal cases most of the movable pick and place robot comes with just normal tires. But from the paper no [6] & [7] gives the idea of using mecanum wheels Which allows the robot to move in all the directions very easily. These papers presents about the use and working of mecanum wheels.

All of these literatures, provided the knowledge of how to design and implement the proposed robotic arm and chasis. And Also found solutions for the some of the limitations of the existing pick and place robot. The robot will be developed with Arduino board as the controller which makes the robot more cost effective than using other controllers like Raspberry pi. The DOF of the robot will be 6, which makes the robot more flexible in movements. ABS polymer material is decided to select for build the arm because it is most suitable for 3D printing and it is strong enough to hold required objects. The control of the movements of robot will be made by using the mobile application instead of using the joystick or PS2 controller Which makes the easy control of robot. In addition, This project also adds a special feature that is the ability to store the movements and automatically repeat them if once we control the robot to do so.

Next chapter 4 will discuss about design formulation part and specifications of this project.

Chapter 4

DESIGN FORMULATION

4.1 Design Methodology

The designing of robot is be done by using the cad tools. Here in this project solid work 3D model software is used to design the parts of the robot. The parts includes; robotic arm, base of movement and mecanum wheels. Solid Works is a computer-aided design (CAD) software application that is commonly used to create 3D models of mechanical parts, assemblies, and other engineering designs. It is widely used in a variety of industries, including manufacturing, automotive, aerospace, and defense.

The robot is design to control with a mobile application. So the required android application can be developed by using the APP INVENTER software. MIT App Inventor is a free, online platform for creating mobile applications (apps) without the need for coding or programming knowledge. To use MIT App Inventor, users start by dragging and dropping "blocks" of code onto a design canvas to build their app. The blocks represent different actions and events that can be triggered within the app, such as displaying text or images, responding to user input, or interacting with the device's sensors or other hardware. The application has various control buttons to control all the movements and actions of robotic arm and base wheels. It will have the buttons like "speed,front,back,right,left,rotate,pick,etc..." By clicking on that buttons the robot will do the respective actions. To connect the robot with this application Wifi module can be used.

In the hardware, arduino mega act as the controller unit. The arm has 6 degree of freedom. For the first 3 axis, the waist, the shoulder and the elbow, we can use the MG996R servos, and for the other 2 axis, the wrist roll and wrist pitch, as well as the gripper the smaller SG90 micro servos can be used. The base of the robot will have 4 mecanum wheels. Each wheel can be attached on a NEMA 17 stepper motor, and knowing the fact that stepper motors can be precisely controlled.

The robotic arm will be made by 3D printing technology.3D printing is a process of creating a physical object from a digital model by building it up layer by layer. It is also known as additive manufacturing. 3D printing can be used to create a wide variety of objects, including prototypes, parts for machines and products, and even finished products. The material used here is ABS (acrylonitrile butadiene styrene). ABS has a relatively low melting point and it strong enough to be using for variety of 3D printing applications. The base platform of this robot is a simple box which will make out of 8mm thick MDF boards. MDF (medium density fiberboard) is a type of engineered wood product made from wood fibers that have been mixed with resin and formed into sheets under high pressure. It is commonly used in a variety of applications, including furniture, cabinetry, and construction. MDF is known for its smooth surface, uniform density, and good machinability, making it a popular choice for applications where a smooth finish is desired. It is also relatively inexpensive compared to other types of wood products.

More Specifications of System

- It is a robot consisting of two parts, the arm and base.
- The robotic arm is used to pick and place the objects.
- Dimensions of robotic arm: 16cm (h)x 15cm(l)
- Dimensions of base : 30cm (l)x 15cm (w)x 8cm (h).
- Maximum finger opening : 6cm
- The distance range up to 10 meters.
- Movements can be remotely controlled by a mobile application (App Inventer).
- It can also do repeated task by storing the path of it's movement and actions.
- The robotic arm is built by 3D printing Using (Acrylonitrile butadiene styrene [ABS] material).
- MDF (Medium Density Fiber) material is used to make the base of the robot.

4.2 Block Diagram

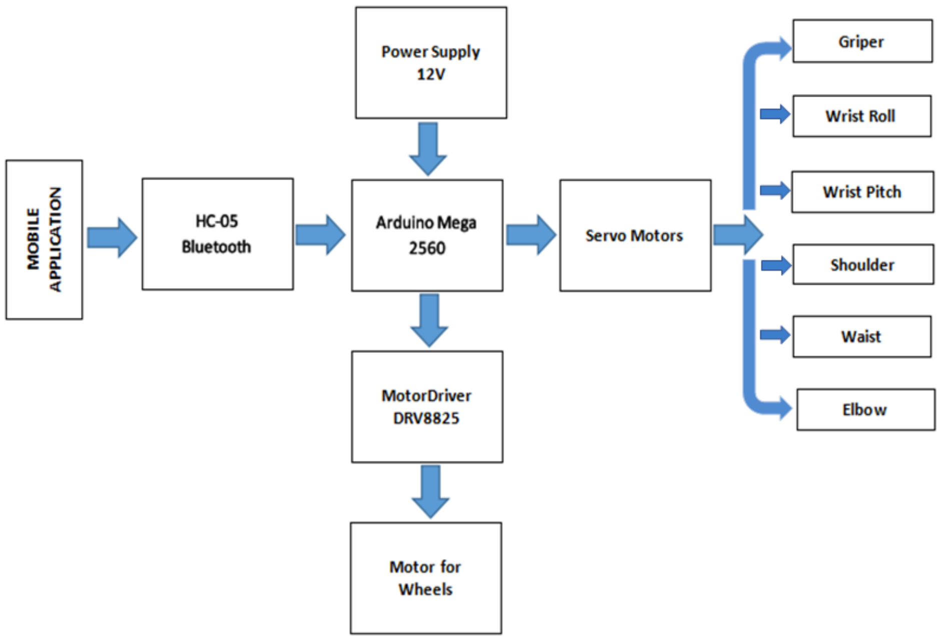

Figure 4.1: Block Diagram

The proposed block diagram of the system is shown in the above figure. It consist of a Arduino mega which act as the controller unit. The Hc-05 Blutooth module is the inputs to the controller. The servo motors and stepper motors are the output of the controller. The stepper motors are for wheels. And servo motors are for arm movements. The entire system is powered by a 12v lithium battery. When ever we tap on the control buttons on the mobile application, the application will communicate with the robot through the bluetooth wirelessly.

4.3 Hardware Requirements

4.3.1 Arduino Mega Board

The Arduino Mega is a microcontroller board based on the ATmega2560 microcontroller. It has 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. The Arduino

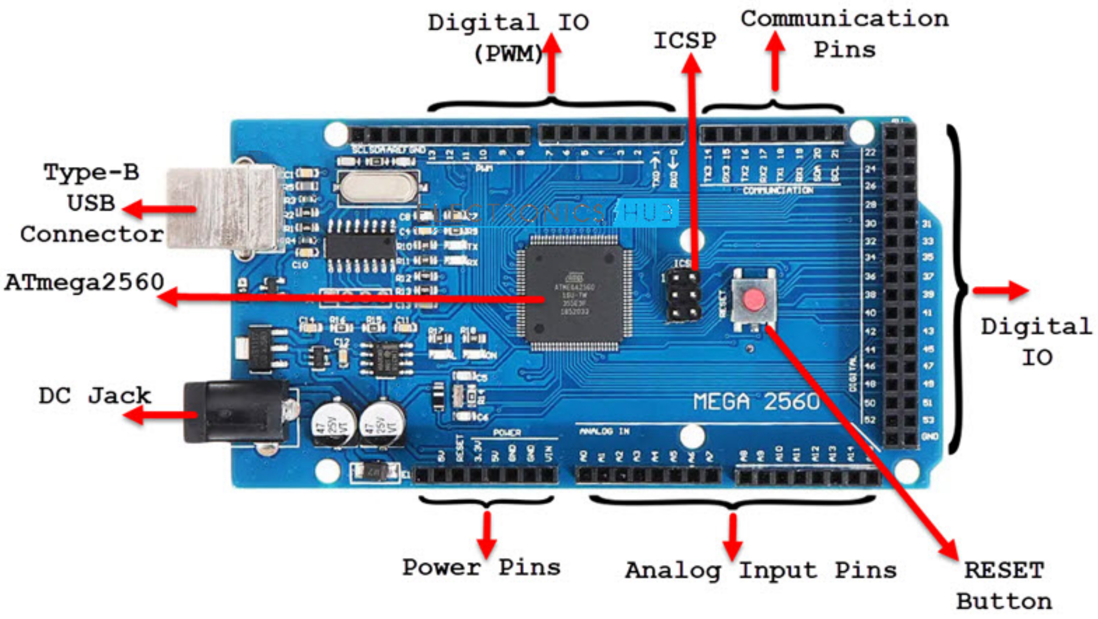

Figure 4.2: Arduino-Mega-Board

Mega is designed to be used with the Arduino software development environment, which makes it easy to write code and upload it to the board. The Arduino Mega is commonly used in projects that require a lot of digital input/output pins, such as controlling many LEDs or servo motors, or projects that require a lot of analog inputs, such as sensing a variety of sensor data.

The Arduino Mega board has a total of 54 digital input/output pins, among which 15 can be used for PWM (Pulse Width Modulation) output. Here's a breakdown of the pins on an Arduino Mega board:

- Digital Pins:

  Digital Pins 0 to 13: These pins are labeled with numbers and can be used for digital input or output. Digital Pins 14 to 21: These additional pins are labeled A0 to A7 and can be used as either digital input or analog input pins. Digital Pins 22 to 53: These are additional digital input/output pins labeled with numbers.

- PWM Pins:

  Digital Pins 2 to 13: These pins can be used for analog output using Pulse Width Modulation (PWM) technique. Digital Pins 44 to 46: These pins, labeled with , can also be used for PWM output.

- Analog Pins:

  Analog Inputs A0 to A15: These pins can be used for analog input. However, on the Arduino Mega board, only pins A0 to A7 are labeled on the board itself.

- Special Function Pins:

  TX0 and RX0: These are the hardware serial communication pins for Serial Port 0 (Serial0) used for communication with external devices.
  TX1 and RX1: These are the hardware serial communication pins for Serial Port 1 (Serial1) used for communication with external devices.
  TX2 and RX2: These are the hardware serial communication pins for Serial Port 2 (Serial2) used for communication with external devices.
  TX3 and RX3: These are the hardware serial communication pins for Serial Port 3 (Serial3) used for communication with external devices.

- Power and Ground Pins:

  VIN: This pin is used to supply an external voltage to the board (typically between 7V and 12V).
  5V: This pin provides a regulated 5V output.

3.3V: This pin provides a regulated 3.3V output.

GND: These pins are the ground connections.

4.3.2 Stepper Motor

A stepper motor is an electric motor that rotates in discrete steps, or "steps," rather than continuously, as a regular electric motor does. Stepper motors are used in a wide variety of applications which requires precise positioning or movement.Stepper motors are controlled by sending electrical pulses to the motor's windings, which causes the motor to rotate a specific number of steps. These are highly precise and can be accurately controlled, that why decided to use the stepper motors.

Figure 4.3: Stepper Motor

4.3.3 DRV8825 Stepper Driver

The DRV8825 is a stepper motor driver that is commonly used to control NEMA 17 stepper motors. It is made by Texas Instruments and is known for its high current capability, low heat generation, and precise current control.The DRV8825 is a bipolar stepper motor driver, which means that it can control stepper motors that have two coils (also known as "four-wire" motors). It operates by applying current to the coils of the stepper motor in a specific sequence, which causes the motor to rotate in precise increments (steps).

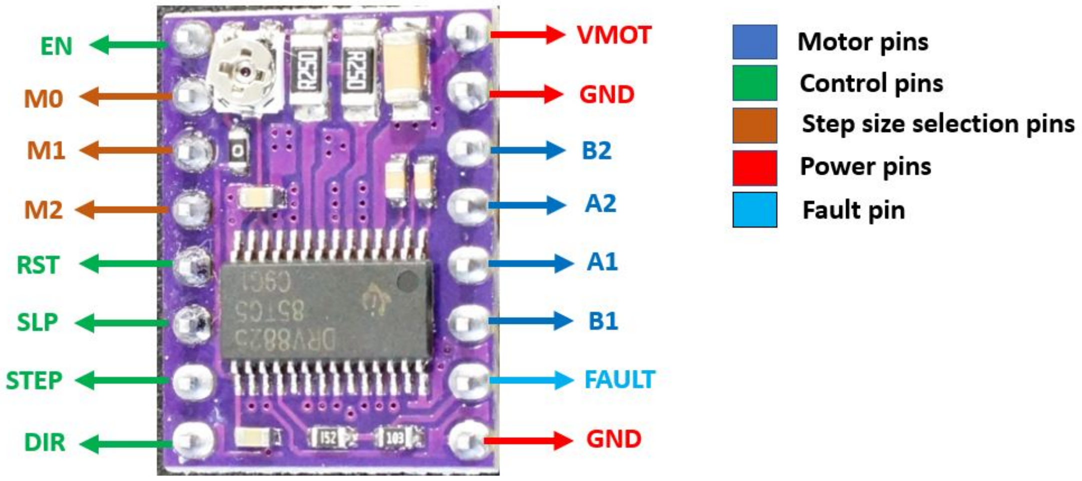

Figure 4.4: DRV8825-Driver

The DRV8825 has a number of features that make it easy to use, including an on-board voltage regulator, over-temperature protection, and an adjustable current limit. It can be controlled using pulse-width modulation (PWM), which allows the user to adjust the speed and torque of the motor by modulating the width of the pulse applied to the driver.

It has several pins with specific functions. Here's a description of the DRV8825 driver pinout:

1. VMOT: This pin is used to connect the motor power supply voltage. It typically accepts a voltage range of 8.2V to 45V. Make sure to provide the appropriate voltage for your stepper motor.

2. GND: The GND pin is the ground reference for the motor power supply. Connect it to the ground of your circuit or power supply.

3. VDD: This pin is used to provide logic power supply voltage for internal logic circuitry. Connect it to a 3.3V or 5V power supply.

4. EN (Enable): The EN pin is used to enable or disable the motor driver. When this pin is set HIGH (logic 1), the driver is enabled, and the motor

can run. When it is set LOW (logic 0), the driver is disabled, and the motor stops. You can connect this pin directly to a digital output pin of a microcontroller or use it in combination with a pulldown resistor to set the default state.

5. STEP: The STEP pin is used to control the step pulse signal for the stepper motor. Each transition from LOW to HIGH (or HIGH to LOW) on this pin generates a step movement. Connect this pin to a digital output pin of a microcontroller or any other signal source that will control the stepping motion.

6. DIR (Direction): The DIR pin determines the direction of the motor rotation. Setting this pin HIGH or LOW determines the direction of movement. Connect this pin to a digital output pin of a microcontroller or any other signal source that controls the motor direction.

7. M0, M1, M2: These three pins are used to configure the microstepping resolution of the driver. By setting these pins to specific logic levels (HIGH or LOW), you can select full-step, half-step, quarter-step, or higher microstepping resolutions, depending on the driver's capabilities. The specific logic levels required for each microstepping mode can vary, so refer to the datasheet or documentation for your driver to set the desired resolution.

8. SLEEP: The SLEEP pin is used to put the driver into a low-power sleep mode when set HIGH. It is an optional pin, and if not used, it can be left unconnected or tied to GND to keep the driver in an enabled state.

9. RESET: The RESET pin is used to reset the driver to its default settings when pulsed HIGH. It is an optional pin, and if not used, it can be left unconnected or tied to GND to keep the driver in its current configuration.

4.3.4 Servo Motor

This is a type of motor that is commonly used in robotics and other motion control applications.A servo motor is a type of motor that is designed to rotate to a specific position and hold that position. It consists of a motor, a gear train, and a control circuit. The control circuit receives a signal from a controller (such as an Arduino board) and uses it to rotate the motor to the desired position.

Figure 4.5: Servo Motor

The position of the motor is controlled by the duration of the control signal, with longer durations corresponding to greater rotation. The servo motors are used in project to controll the robotic arm movemenents.

4.3.5 HC-05 Blutooth Module

The HC-05 module is a Bluetooth communication module that enables wireless connectivity between electronic devices. With a typical range of around 10 meters, it allows devices to communicate without the need for physical connections. The module can function in both master and slave modes, facilitating bidirectional communication between devices. It enables the control of robot through mobile application in the project by sending commands to the micro controller wirelessly. It is providing a cost-effective solution for adding wireless capabilities and enhancing connectivity between devices.

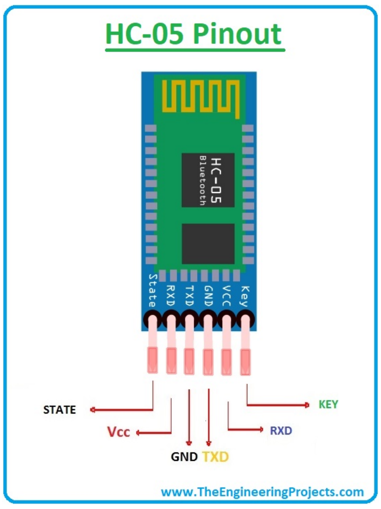

Figure 4.6: HC-05 Blutooth Module

The HC-05 Bluetooth module typically has six pins, each serving a specific purpose. Here's a description of the pinout for the HC-05 module:

1. VCC: This pin is used to provide power to the module. It typically requires a 3.3V power supply, although some modules can tolerate 5V. Make sure to connect it to the appropriate voltage source.

2. GND: This pin is the ground or 0V reference for the module. Connect it to the ground of your circuit or power supply.

3. TXD: The TXD (Transmit Data) pin is used for serial communication and transmits data from the HC-05 module to another device. Connect this pin to the RX (receive) pin of the device you want to communicate with.

4. RXD: The RXD (Receive Data) pin is also used for serial communication and receives data from another device. Connect this pin to the TX (transmit) pin of the device you want to communicate with.

5. STATE: The STATE pin is an optional pin and is used to indicate the status of the module. It can be used to determine if the module is in a

connected or disconnected state. However, not all HC-05 modules have this pin, and it can be left unconnected in most cases.

6. EN/KEY: The EN or KEY pin is another optional pin. It is used to enter the AT command mode of the HC-05 module for configuration. Connecting this pin to the ground (GND) for a short duration typically puts the module into AT command mode. Again, not all HC-05 modules have this pin, and it can be left unconnected if not needed for configuration

4.3.6 LiPo Battery

A 3S LiPo battery refers to a Lithium Polymer battery pack that consists of three individual cells connected in series. Each cell has a nominal voltage of 3.7 volts, resulting in a total nominal voltage of 11.1 volts for the 3S pack.This battery of 3500 Ah, which act as the complete power supplier for our robot. It is a type of rechargeable battery that are known for their high energy density, which means they can store a significant amount of energy in a relatively compact and lightweight package. This makes them popular for applications where weight and size are important factors.

Figure 4.7: 12V LiPo Battery

4.3.7 LM 350 IC

The LM350 is a positive voltage regulator, which means that it can only be used to regulate a positive input voltage. It has a wide input voltage range, from 3.0 volts to 40 volts, which makes it suitable for use with a variety of power sources. In our system some devices like wify module wich requires less voltage to work. That is why we are using the LM 350 IC. It is capable of delivering a current of up to 1.5 amps, which makes it suitable for use in high-current applications.

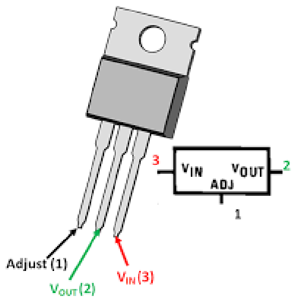

Figure 4.8: LM 350 IC

4.3.8 3D Printer Filament

ABS (Acrylonitrile Butadiene Styrene) is a commonly used 3D printing fila- ment that is known for its strength, durability, and versatility.One of the main advantages of ABS filament is its strength and durability. It is a strong and rigid material that is resistant to impact, making it well-suited for applications that require high strength. It is also resistant to chemicals, making it a good choice for applications that may come into contact with corrosive substance.

Figure 4.9: 3D Printer Filament

4.3.9 MDF Material

MDF (medium-density fiberboard) is a type of engineered wood product that is made from wood fibers that have been combined with resin and other materials and then pressed into sheets. MDF is known for its smooth, uniform surface and its ability to be machined and shaped into a wide variety of products.

Figure 4.10: MDF Sheet

4.4 Software Requirements

4.5 Arduino IDE

The (IDE) is a free software application that allows users to write, upload, and debug code for Arduino boards. It is the official software for programming Arduino boards, and it is the most widely used tool for developing Arduino projects.The Arduino IDE includes a text editor for writing code, a compiler for translating code into machine language that the Arduino board can understand, and a debugger for finding and fixing errors in code. It also includes a library manager that allows users to easily install and use libraries of code that can be used to add functionality to their projects. The Arduino IDE is designed to be easy to use and includes a range of features to help users write and debug their code. It includes a text editor for writing code, a compiler that converts the code into a format that can be uploaded to an Arduino board, and a serial monitor for viewing the output of the code as it runs on the board.

The IDE also includes a library manager that allows users to import and use pre-written code libraries, and a debugger that can help users identify and fix errors in their code.

4.6 Solid Work 3D CAD Software

SolidWorks is a 3D computer-aided design (CAD) software application that is widely used for creating, analyzing, and communicating engineering designs.SolidWorks is a feature-based parametric solid modeling software that allows users to create, modify, and analyze 3D models of mechanical parts, assemblies, and products. It includes a wide range of tools and features that are specifically designed for creating and analyzing mechanical designs, including solid modeling, surface modeling, sheet metal design, and simulation.SolidWorks includes a range of tools and features that allow users to design and visualize products in 3D, create and analyze mechanical assemblies, generate engineering drawings and manufacturing documentation, and perform simulations to test the performance and integrity of designs.

4.7 MIT App Inventor

MIT App Inventor is a web-based platform for building Android mobile apps that was developed by the Massachusetts Institute of Technology (MIT). App Inventor consists of a visual programming environment that allows users to build apps by dragging and dropping blocks of code. Users can create custom user interfaces, connect to the Internet to retrieve data, and add functionality to their apps using a wide range of pre-built blocks. MIT App Inventor is designed to be easy to use, even for people with no programming experience. It includes an emulator, which allows users to test their apps on their computer before uploading them to a device, and a user-friendly interface.

4.8 CIRCUIT DIAGRAM AND WORKING

4.8.1 Circuit Diagram

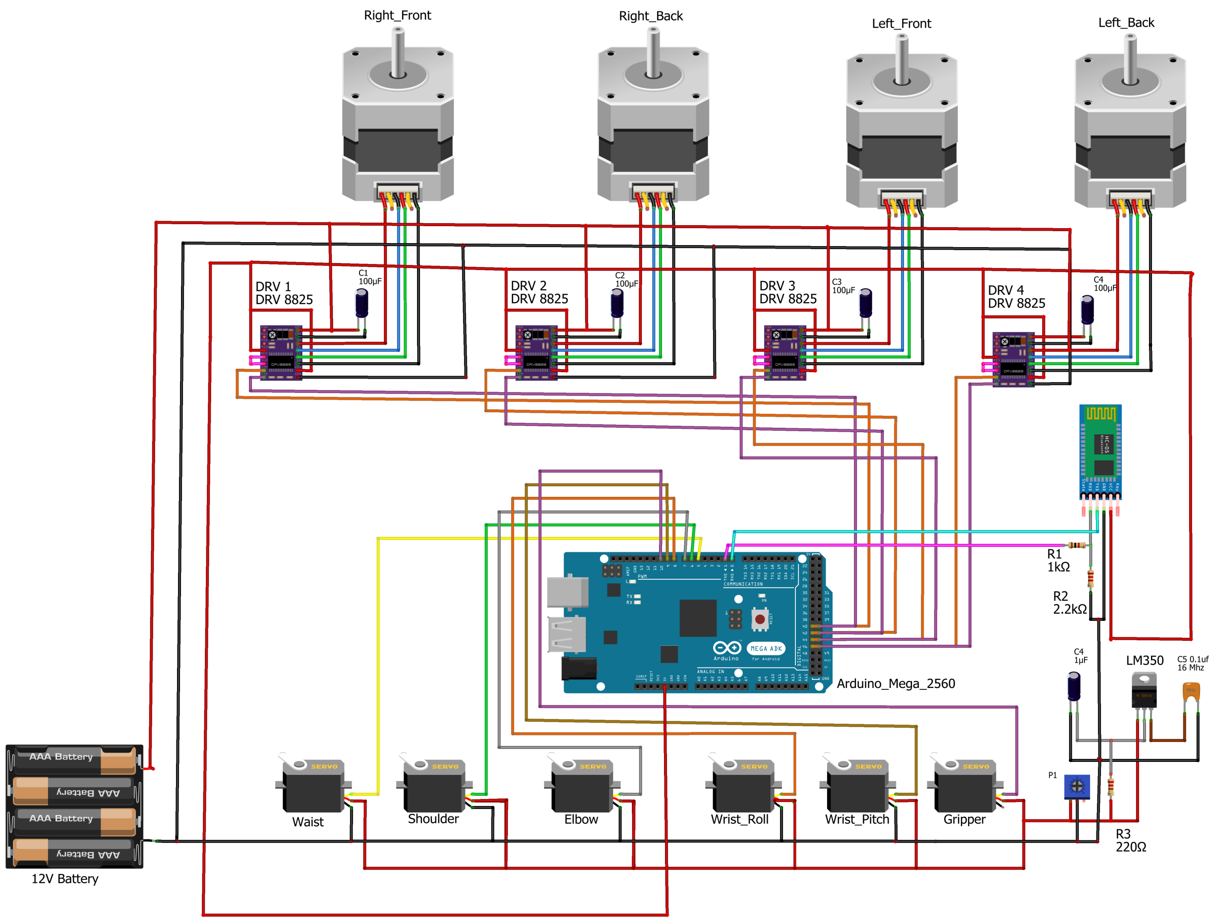

Figure 4.11: Circuit Diagram

4.8.2 Operation

The robot can be operated by using a mobile application wirelessly. For the robotic arm there will be some buttons such as the waist, shoulder, elbow, wrist roll, wrist pitch, and the gripper. These are for different positions of arm. The arm movements can be controlled by clicking on the respective buttons. Also 3 additional buttons are included such as save, run, reset.These buttons are for saving the steps and run them repeatedly and automatically. The reset button provide the option to clear the previously stored actions and store new actions.

For the base wheel movements there are some other buttons provided in the application.It allows the movements in all directions, not only to the back front,front,left and right.

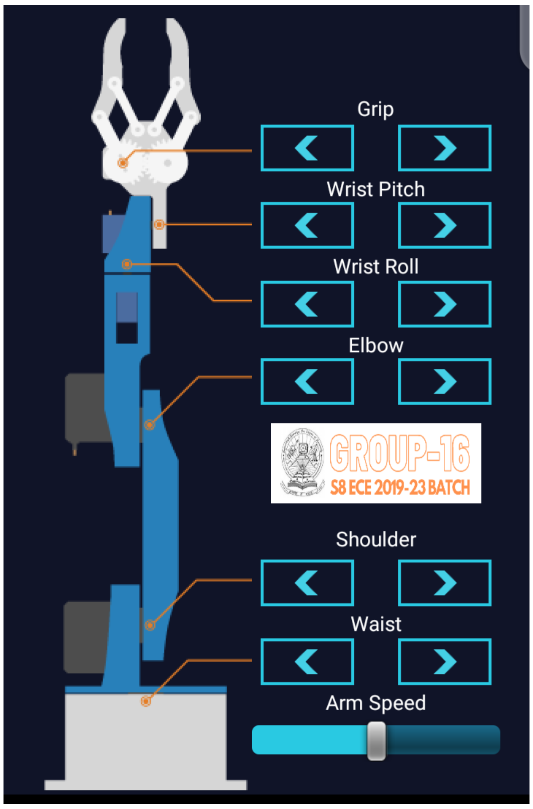

Figure 4.12: APP Interface

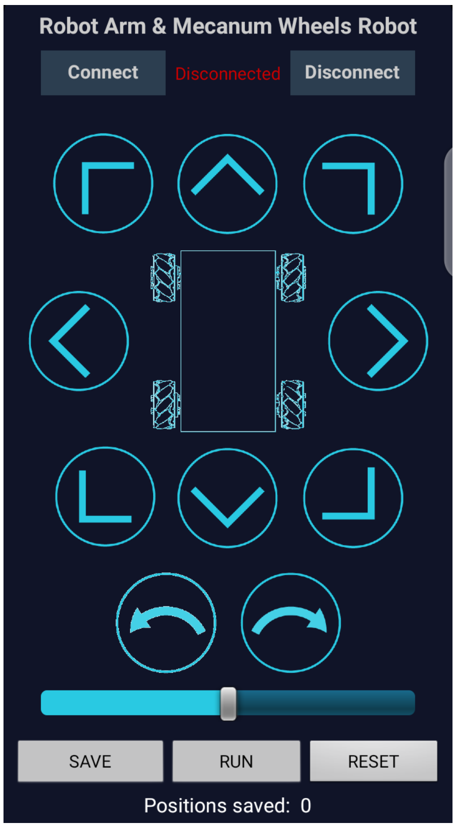

Figure 4.13: APP Interface

4.8.3 3D Models

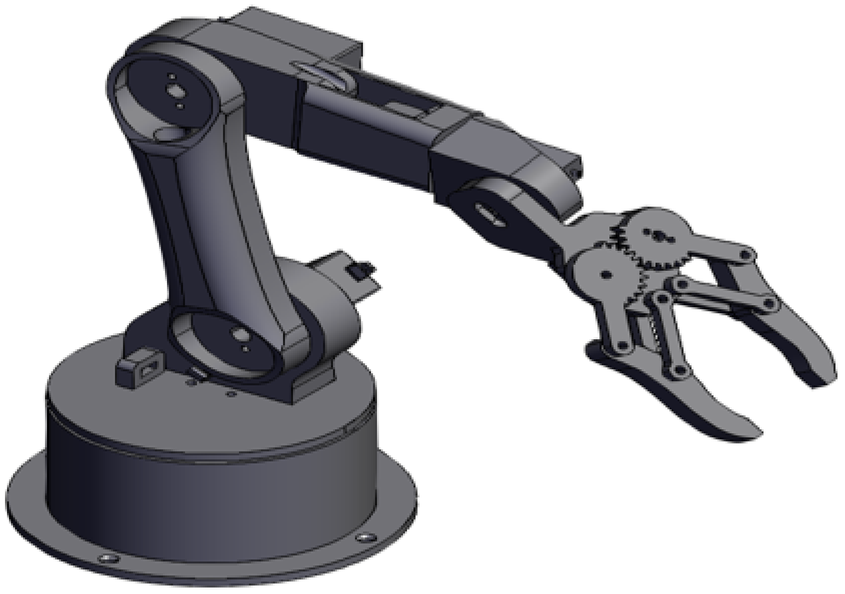

Figure 4.14: 3D Model of Arm

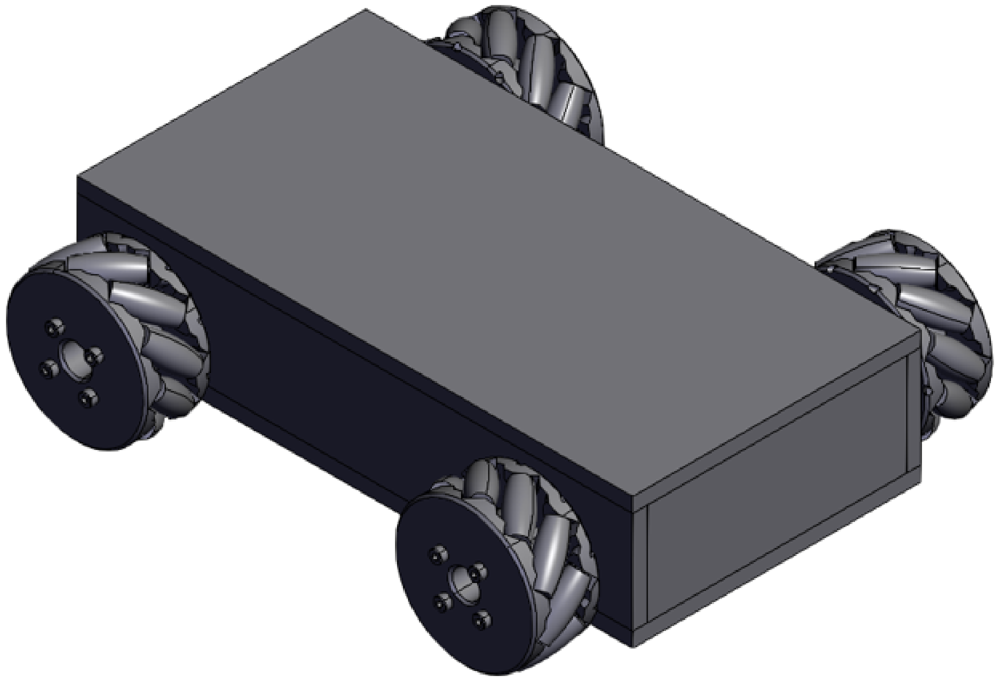

Figure 4.15: 3D Model Base

Chapter 4 comprises block diagram ,components required and how the operations are to be carried out for the proposed system.The next chapter shows the result of our project.

Chapter 5

RESULT

The Automatic Pick and Place Robot project was designed having a robotic arm of dimensions 16cm height and 15cm length placed on a base of 30cm length, 15cm width and 8cm height. The robotic arm has two finger which has a maximum opening of 6cm between them. This robot can be controlled upto a range of 10 meters. This robot picks and places an object weighing about 40 grams between its finger opening using a mobile application.

The robot's arm movements could be controlled by clicking on the respective buttons provided in the mobile application, allowing the user to control the robot's movements. The user could also save and store actions to run them repeatedly and automatically, enabling the robot to perform its actions more efficiently.

The images of the output project are given below. It includes images of application and the product photos.

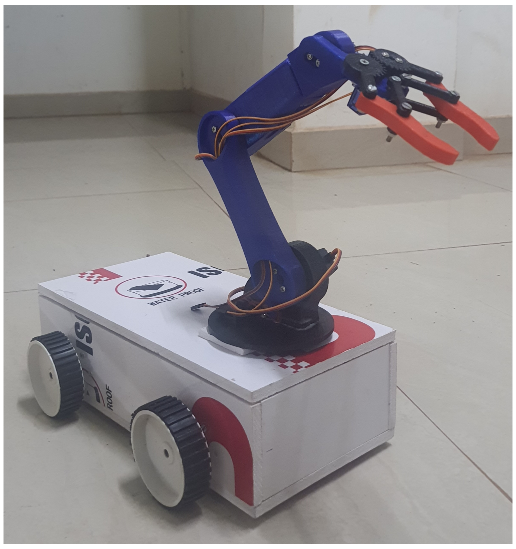

Figure 5.1: Final Product

The application interface for the user is shown below. It has two sections, one for controlling arm and the other for controlling the base wheels. The arm control includes GRIP,PITCH,ROLL,ELBOW,SHOULDER, and WAIST buttons. Also there provided a Button named SPEED to control the speed of arm movements.

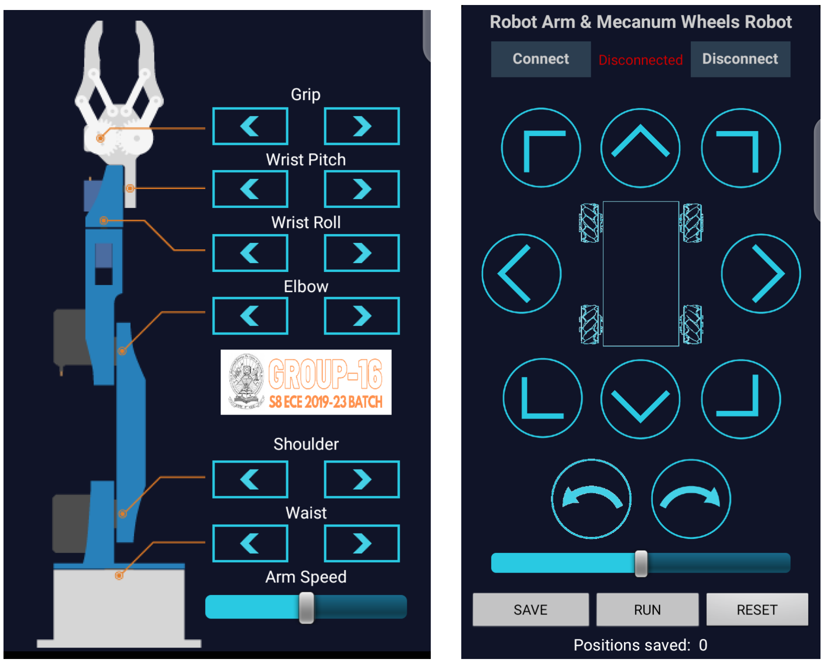

Figure 5.2: Application Interfaces for Arm and Wheel

The following pictures shows the automatic pick and place robot picking and placing a plastic cube of 30gms.

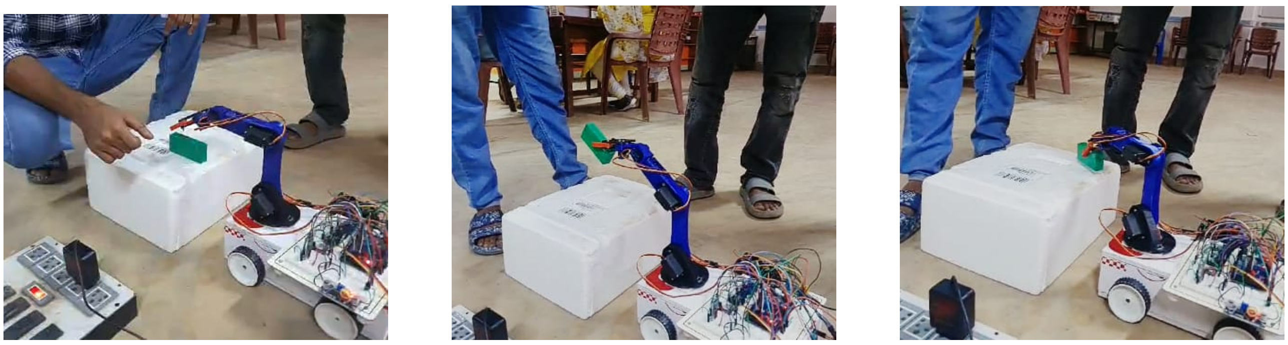

Figure 5.3: Working of Robot

The next chapter will gives the conclusion and future scope of the this project.

Chapter 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The objective of this project was to implement a pick and place robot to pick and place object from one place to another which has been successfully met, but has a limitation that the motor shaft connected to the arm can only lift upto 40grams of weight because all the weight of the arm and the picked object is being lifted by this only one shaft. The robot was designed with a maximum finger opening of 6cm and capable of carrying only small weighted objects. The project aimed to make the robot perform all its actions repeatedly and automatically once it is controlled to do so.

The robot can be operated using a mobile application that has option button to control the different positions of the arm such as waist, shoulder, elbow, wrist roll, wrist pitch, and gripper. The robot's arm movements can be controlled by clicking on the respective option button, and three additional buttons have been included to save the steps, run them repeatedly and automatically and reset the previously stored actions to store new actions.

The robot's base wheel movements can also be controlled by other buttons provided in the application that allows movements in all directions. The project demonstrates the potential of robotic automation in various industries like logistics, and others.

6.2 FUTURE SCOPE

The Automatic Pick and Place Robot project has several future scopes that can be explored to further improve the robot's functionality and performance.

Firstly, the project can be expanded to include more advanced sensors such as cameras or ultrasonic sensors that can enable the robot to detect and pick objects of different shapes and sizes more accurately.

Secondly, the project can be developed to include machine learning algorithms that can enable the robot to learn from its previous actions and improve its performance over time. This can enhance the robot's ability to perform more complex tasks and navigate through unknown environments.

Thirdly, the project can be scaled up to include multiple robots working collaboratively to perform more complex tasks such as assembly line production or logistics operations.

Lastly, the project can be further improved to make it more user-friendly and accessible, with the development of a more intuitive user interface or voice commands for controlling the robot's movements.

BIBLIOGRAPHY

[1] K. Kruthika, B. M. Kiran Kumar and S. Lakshminarayanan, "Design and development of a robotic arm," 2016 International Conference on Circuits, Controls, Communications and Computing (I4C), 2016.

[2] G. H. Lee, Y. J. Lee, and J. H. Kim, "A Fast and Accurate Pick-and-Place Robot for Small Parts". IEEE Transactions on Industrial Electronics, vol. 56, no. 3, pp. 897-906, Mar. 2017.

[3] K. Kim, J. H. Kim, and Y. J. Lee, "A Compact Pick-and-Place Robot with a High-Speed and High-Precision Motion Control System". IEEE Transactions on Industrial Electronics, vol. 59, no. 2, pp. 851-860, Feb. 2018.

[4] Mohd Ashiq Kamaril, Yusuff, Reza Ezucin Samin, Babul Salam, Kader Ibrahim, "Wireless Mobile Robotic Arm" International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012)

[5] Avinash N Bhat, Abhishek Yalnaik, "Design and Fabrication of Pick and Place Robot using Microcontroller" International Journal of Science and Research (IJSR)ISSN: 2319-7064. Volume 10 Issue 11, November 2021

[6] Naga Sudheer Ravela,Sainath Somisetty, "Pick and Place Arm and Robot Movement Control by using Android" Vol. 4, Issue 3, March 2016

[7] T. Bin Mohamed et al., ”Development of mobile robot drive system using mecanum wheels,” 2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEES), 2021

[8] Ch.Shravani, G. Indira, V. Appalaraju ”Arduino Based Color Sorting Machine using TCS3200 Color Sensor” Volume-8, Issue- 6S4, April 2019

APPENDIX

PROGRAM CODE

```
#include <SoftwareSerial.h>

#include <AccelStepper.h>

#include <Servo.h>

Servo servo01;

Servo servo02;

Servo servo03;

Servo servo04;

Servo servo05;

Servo servo06;


// Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)


// Define the stepper motors and the pins the will use

AccelStepper LeftBackWheel(1, 42, 43);

// (Type:driver, STEP, DIR) - Stepper1

AccelStepper LeftFrontWheel(1, 40, 41);  // Stepper2

AccelStepper RightBackWheel(1, 44, 45);  // Stepper3

AccelStepper RightFrontWheel(1, 46, 47); // Stepper4


#define led 14


int wheelSpeed = 1500;


int lbw[50], lfw[50], rbw[50], rfw[50];
```

```
// arrays for storing positions/steps


int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos;
// current position
int servo1PPos, servo2PPos, servo3PPos,
servo4PPos, servo5PPos, servo6PPos;
// previous position
int servo01SP[50], servo02SP[50], servo03SP[50],
servo04SP[50], servo05SP[50], servo06SP[50];
// for storing positions/steps
int speedDelay = 20;
int index = 0;
int dataIn;
int m = 0;


void setup() {
  // Set initial seed values for the steppers
  LeftFrontWheel.setMaxSpeed(3000);
  LeftBackWheel.setMaxSpeed(3000);
  RightFrontWheel.setMaxSpeed(3000);
  RightBackWheel.setMaxSpeed(3000);
  pinMode(led, OUTPUT);
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Serial.begin(9600); // Default baud rate of the Bluetooth module
  Serial.setTimeout(5);
  delay(20);
  Serial.begin(9600);
  // Move robot arm to initial position
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 100;
  servo02.write(servo2PPos);
  servo3PPos = 120;
  servo03.write(servo3PPos);
  servo4PPos = 95;
  servo04.write(servo4PPos);
  servo5PPos = 60;
  servo05.write(servo5PPos);
  servo6PPos = 110;
  servo06.write(servo6PPos);
}

void loop() {
  // Check for incoming data
  if (Serial.available() > 0) {
    dataIn = Serial.read();  // Read the data

    if (dataIn == 0) {
      m = 0;
    }
    if (dataIn == 1) {
```

```
    m = 1;
  }
  if (dataIn == 2) {
    m = 2;
  }
  if (dataIn == 3) {
    m = 3;
  }
  if (dataIn == 4) {
    m = 4;
  }
  if (dataIn == 5) {
    m = 5;
  }
  if (dataIn == 6) {
    m = 6;
  }
  if (dataIn == 7) {
    m = 7;
  }
  if (dataIn == 8) {
    m = 8;
  }
  if (dataIn == 9) {
    m = 9;
  }
  if (dataIn == 10) {
    m = 10;
```

```
}
if (dataIn == 11) {
  m = 11;
}
if (dataIn == 12) {
  m = 12;
}
if (dataIn == 14) {
  m = 14;
}
if (dataIn == 16) {
  m = 16;
}
if (dataIn == 17) {
  m = 17;
}
if (dataIn == 18) {
  m = 18;
}
if (dataIn == 19) {
  m = 19;
}
if (dataIn == 20) {
  m = 20;
}
if (dataIn == 21) {
  m = 21;
}
```

```
if (dataIn == 22) {
  m = 22;
}
if (dataIn == 23) {
  m = 23;
}
if (dataIn == 24) {
  m = 24;
}
if (dataIn == 25) {
  m = 25;
}
if (dataIn == 26) {
  m = 26;
}
if (dataIn == 27) {
  m = 27;
}

// Move the Mecanum wheels platform
if (m == 4) {
  moveSidewaysLeft();
}
if (m == 5) {
  moveSidewaysRight();
}
if (m == 2) {
  moveForward();
}
if (m == 7) {
  moveBackward();
}
if (m == 3) {
  moveRightForward();
}
if (m == 1) {
  moveLeftForward();
}
if (m == 8) {
  moveRightBackward();
}
if (m == 6) {
  moveLeftBackward();
}
if (m == 9) {
  rotateLeft();
}
if (m == 10) {
  rotateRight();
}

if (m == 0) {
  stopMoving();
}

// Mecanum wheels speed
```

```
if (dataIn > 30 & dataIn < 100) {
  wheelSpeed = dataIn * 20;
}


// Move robot arm
// Move servo 1 in positive direction
while (m == 16) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo01.write(servo1PPos);
  servo1PPos++;
  delay(speedDelay);
}
// Move servo 1 in negative direction
while (m == 17) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo01.write(servo1PPos);
  servo1PPos--;
  delay(speedDelay);
}
// Move servo 2
while (m == 19) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
```

```
    servo02.write(servo2PPos);
    servo2PPos++;
    delay(speedDelay);
  }
  while (m == 18) {
    if (Serial.available() > 0) {
      m = Serial.read();
    }
    servo02.write(servo2PPos);
    servo2PPos--;
    delay(speedDelay);
  }
  // Move servo 3
  while (m == 20) {
    if (Serial.available() > 0) {
      m = Serial.read();
    }
    servo03.write(servo3PPos);
    servo3PPos++;
    delay(speedDelay);
  }
  while (m == 21) {
    if (Serial.available() > 0) {
      m = Serial.read();
    }
    servo03.write(servo3PPos);
    servo3PPos--;
    delay(speedDelay);
```

```
}
// Move servo 4
while (m == 23) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo04.write(servo4PPos);
  servo4PPos++;
  delay(speedDelay);
}
while (m == 22) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo04.write(servo4PPos);
  servo4PPos--;
  delay(speedDelay);
}
// Move servo 5
while (m == 25) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo05.write(servo5PPos);
  servo5PPos++;
  delay(speedDelay);
}
while (m == 24) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo05.write(servo5PPos);
  servo5PPos--;
  delay(speedDelay);
}
// Move servo 6
while (m == 26) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo06.write(servo6PPos);
  servo6PPos++;
  delay(speedDelay);
}
while (m == 27) {
  if (Serial.available() > 0) {
    m = Serial.read();
  }
  servo06.write(servo6PPos);
  servo6PPos--;
  delay(speedDelay);
}

// If arm speed slider is changed
if (dataIn > 101 & dataIn < 250) {
  speedDelay = dataIn / 10; // Change servo speed (delay time)
```

```
}

// If button "SAVE" is pressed
if (m == 12) {
  //if it’s initial save, set the steppers position to 0
  if (index == 0) {
    LeftBackWheel.setCurrentPosition(0);
    LeftFrontWheel.setCurrentPosition(0);
    RightBackWheel.setCurrentPosition(0);
    RightFrontWheel.setCurrentPosition(0);
  }
  lbw[index] = LeftBackWheel.currentPosition();
  // save position into the array
  lfw[index] = LeftFrontWheel.currentPosition();
  rbw[index] = RightBackWheel.currentPosition();
  rfw[index] = RightFrontWheel.currentPosition();

  servo01SP[index] = servo1PPos;  // save posi. into array
  servo02SP[index] = servo2PPos;
  servo03SP[index] = servo3PPos;
  servo04SP[index] = servo4PPos;
  servo05SP[index] = servo5PPos;
  servo06SP[index] = servo6PPos;
  index++;                        // Increase the array index
  m = 0;
}

// If button "RUN" is pressed
```

```
    if (m == 14) {
      runSteps();

      // If button "RESET" is pressed
      if (dataIn != 14) {
        stopMoving();
        memset(lbw, 0, sizeof(lbw));
        // Clear the array data to 0
        memset(lfw, 0, sizeof(lfw));
        memset(rbw, 0, sizeof(rbw));
        memset(rfw, 0, sizeof(rfw));
        memset(servo01SP, 0, sizeof(servo01SP));
        // Clear the array data to 0
        memset(servo02SP, 0, sizeof(servo02SP));
        memset(servo03SP, 0, sizeof(servo03SP));
        memset(servo04SP, 0, sizeof(servo04SP));
        memset(servo05SP, 0, sizeof(servo05SP));
        memset(servo06SP, 0, sizeof(servo06SP));
        index = 0;  // Index to 0
      }
    }
  }
  LeftFrontWheel.runSpeed();
  LeftBackWheel.runSpeed();
  RightFrontWheel.runSpeed();
  RightBackWheel.runSpeed();

  // Monitor the battery voltage
  int sensorValue = analogRead(A0);
  float voltage = sensorValue * (5.0 / 1023.00) * 3;
  // Convert the reading values from 5v to suitable 12V
  //Serial.println(voltage);
  // If voltage is below 11V turn on the LED
  if (voltage < 11) {
    digitalWrite(led, HIGH);
  }
  else {
    digitalWrite(led, LOW);
  }
}
void moveForward() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void moveBackward() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(wheelSpeed);
}
void moveSidewaysRight() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void moveSidewaysLeft() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(wheelSpeed);
}
void rotateLeft() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void rotateRight() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(wheelSpeed);
}
void moveRightForward() {
  LeftFrontWheel.setSpeed(wheelSpeed);
  LeftBackWheel.setSpeed(0);
  RightFrontWheel.setSpeed(0);
  RightBackWheel.setSpeed(-wheelSpeed);
}
void moveRightBackward() {
  LeftFrontWheel.setSpeed(0);
  LeftBackWheel.setSpeed(-wheelSpeed);
  RightFrontWheel.setSpeed(wheelSpeed);
  RightBackWheel.setSpeed(0);
}
void moveLeftForward() {
  LeftFrontWheel.setSpeed(0);
  LeftBackWheel.setSpeed(wheelSpeed);
  RightFrontWheel.setSpeed(-wheelSpeed);
  RightBackWheel.setSpeed(0);
}
void moveLeftBackward() {
  LeftFrontWheel.setSpeed(-wheelSpeed);
  LeftBackWheel.setSpeed(0);
  RightFrontWheel.setSpeed(0);
  RightBackWheel.setSpeed(wheelSpeed);
}
void stopMoving() {
  LeftFrontWheel.setSpeed(0);
  LeftBackWheel.setSpeed(0);
  RightFrontWheel.setSpeed(0);
  RightBackWheel.setSpeed(0);
}

// Automatic mode custom function - run the saved steps
void runSteps() {
  while (dataIn != 13) {
  // Run steps until "RESET" button is pressed
    for (int i = 0; i <= index - 2; i++) {
```

```
    // Run through all steps(index)
      if (Serial.available() > 0) {
      // Check for incomding data
        dataIn = Serial.read();
        if ( dataIn == 15) {
        // If button "PAUSE" is pressed
          while (dataIn != 14) {
          // Wait until "RUN" is pressed again
            if (Serial.available() > 0) {
              dataIn = Serial.read();
              if ( dataIn == 13) {
                break;
              }
            }
          }
        }
        // If speed slider is changed
        if (dataIn > 100 & dataIn < 150) {
          speedDelay = dataIn / 10;
          // Change servo speed (delay time)
        }
        // Mecanum wheels speed
        if (dataIn > 30 & dataIn < 100) {
          wheelSpeed = dataIn * 10;
          dataIn = 14;
        }
      }
      LeftFrontWheel.moveTo(lfw[i]);
```

```
LeftFrontWheel.setSpeed(wheelSpeed);
LeftBackWheel.moveTo(lbw[i]);
LeftBackWheel.setSpeed(wheelSpeed);
RightFrontWheel.moveTo(rfw[i]);
RightFrontWheel.setSpeed(-wheelSpeed);
RightBackWheel.moveTo(rbw[i]);
RightBackWheel.setSpeed(-wheelSpeed);

while (LeftBackWheel.currentPosition()
!= lbw[i] & LeftFrontWheel.currentPosition()
!= lfw[i] & RightFrontWheel.currentPosition()
!= rfw[i] & RightBackWheel.currentPosition() != rbw[i]) {
  LeftFrontWheel.runSpeedToPosition();
  LeftBackWheel.runSpeedToPosition();
  RightFrontWheel.runSpeedToPosition();
  RightBackWheel.runSpeedToPosition();
}
// Servo 1
if (servo01SP[i] == servo01SP[i + 1]) {
}
if (servo01SP[i] > servo01SP[i + 1]) {
  for ( int j = servo01SP[i]; j >= servo01SP[i + 1]; j--) {
    servo01.write(j);
    delay(speedDelay);
  }
}
if (servo01SP[i] < servo01SP[i + 1]) {
  for ( int j = servo01SP[i]; j <= servo01SP[i + 1]; j++) {
    servo01.write(j);
    delay(speedDelay);
  }
}

// Servo 2
if (servo02SP[i] == servo02SP[i + 1]) {
}
if (servo02SP[i] > servo02SP[i + 1]) {
  for ( int j = servo02SP[i]; j >= servo02SP[i + 1]; j--) {
    servo02.write(j);
    delay(speedDelay);
  }
}
if (servo02SP[i] < servo02SP[i + 1]) {
  for ( int j = servo02SP[i]; j <= servo02SP[i + 1]; j++) {
    servo02.write(j);
    delay(speedDelay);
  }
}

// Servo 3
if (servo03SP[i] == servo03SP[i + 1]) {
}
if (servo03SP[i] > servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j >= servo03SP[i + 1]; j--) {
    servo03.write(j);
    delay(speedDelay);
  }
}
if (servo03SP[i] < servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j <= servo03SP[i + 1]; j++) {
    servo03.write(j);
    delay(speedDelay);
  }
}

// Servo 4
if (servo04SP[i] == servo04SP[i + 1]) {
}
if (servo04SP[i] > servo04SP[i + 1]) {
  for ( int j = servo04SP[i]; j >= servo04SP[i + 1]; j--) {
    servo04.write(j);
    delay(speedDelay);
  }
}
if (servo04SP[i] < servo04SP[i + 1]) {
  for ( int j = servo04SP[i]; j <= servo04SP[i + 1]; j++) {
    servo04.write(j);
    delay(speedDelay);
  }
}

// Servo 5
if (servo05SP[i] == servo05SP[i + 1]) {
}
```

```
if (servo05SP[i] > servo05SP[i + 1]) {
  for ( int j = servo05SP[i]; j >= servo05SP[i + 1]; j--) {
    servo05.write(j);
    delay(speedDelay);
  }
}
if (servo05SP[i] < servo05SP[i + 1]) {
  for ( int j = servo05SP[i]; j <= servo05SP[i + 1]; j++) {
    servo05.write(j);
    delay(speedDelay);
  }
}

// Servo 6
if (servo06SP[i] == servo06SP[i + 1]) {
}
if (servo06SP[i] > servo06SP[i + 1]) {
  for ( int j = servo06SP[i]; j >= servo06SP[i + 1]; j--) {
    servo06.write(j);
    delay(speedDelay);
if (servo06SP[i] < servo06SP[i + 1]) {
  for ( int j = servo06SP[i]; j <= servo06SP[i + 1]; j++) {
    servo06.write(j);
    delay(speedDelay);
  }
}
```