DESIGN AND IMPLEMENTATION OF A HYBRID PICK AND DROP ROBOTIC ARM

BY

OKOBRU GENESIS STANLEY	**ENG1805106**
IKOGHO EMMANUEL OKEOGHENE	**ENG1805071**
ALONGE EMMANUEL	**ENG1805037**
EDOSOMWAN MITCHELL	**ENG1805050**
OKANGBUAN OYAKHIRE	**ENG1805104**
OKECHUKWU JULIET	**ENG1805105**

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

OF THE AWARD OF BACHELOR OF ENGINEERING (B.ENG) DEGREE

IN

THE DEPARTMENT OF ELECTRICAL/ELECTRONIC ENGINEERING, FACULTY

OF ENGINEERING, UNIVERSITY OF BENIN, BENIN CITY, NIGERIA.

APRIL, 2024

CERTIFICATION

This is to certify that this project work was carried out by:

OKOBRU GENESIS STANLEY	**ENG1805106**
IKOGHO EMMANUEL OKEOGHENE	**ENG1805071**
ALONGE EMMANUEL	**ENG1805037**
EDOSOMWAN MITCHELL	**ENG1805050**
OKANGBUAN OYAKHIRE	**ENG1805104**
OKECHUKWU JULIET	**ENG1805105**

of the department of Electrical/Electronic Engineering, University of Benin, Edo state, Nigeria.

_____	_____
_____	_____
Signature and Date	Signature and Date
ENGR. DR. L.E. OMOZE	PROF. K.O. OGBEIDE
Project Supervisor	Head of Department

DEDICATION

This project is dedicated to God for his mercy and grace, being with us for the entirety of our years of study in the University of Benin. We also dedicate this project to our parents, family, and loved ones for their care, love and support.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our valued project supervisor, ENGR. DR. L.E. OMOZE, for her invaluable guidance and time spent advising us when we most needed it. She additionally evaluated the project and made the necessary modifications. We would also like to express our gratitude to Prof. K.O. Ogbeide, Head of the Department of Electrical and Electronic Engineering, for making sure we had everything we needed to finish the project.

We would especially like to thank our parents and guardians for their support and encouragement during our time at the university; without them, we could not have progressed as far as we have. We also express our gratitude to the academic and non-academic staff of the Department of Electrical and Electronic Engineering for making our stay here successful.

To our friends who has helped in one way or another during our time of need, we say thank you, for your love and support.

ABSTRACT

The Hybrid Pick and Drop Robotic Arm project presents a versatile robotic arm system capable of both manual and automatic operation. The robotic arm is designed with the flexibility to be controlled either by a user through manual inputs or autonomously through programmed commands. This hybrid functionality allows the robotic arm to adapt to various scenarios and tasks, offering a blend of precision and user control.

The project utilizes a combination of sensors, including an infrared (IR) proximity sensor and an IR receiver, to enable efficient object detection and remote-control capabilities. The Atmega382p microcontroller serves as the central processing unit, coordinating the actions of the robotic arm based on sensor inputs and user commands. The inclusion of a DC-DC buck converter ensures efficient power management for the system, optimizing its performance in both manual and automatic modes.

Through this project, the concept of a hybrid robotic arm is demonstrated, showcasing its potential applications in industries requiring a mix of manual dexterity and automated precision. The project's design and implementation offer insights into the development of adaptable robotic systems, highlighting the importance of flexibility and user-friendliness in robotics technology.

Keywords: Hybrid, Robotic Arm, Manual Control, Automatic Control, Object Detection, Remote Control.

LIST OF FIGURES

LIST OF TABLES

Table of Contents

CHAPTER ONE

1.1 - Introduction

In the world of technology, robotic devices stand as the intersection of human creativity and machine capabilities. Defined as programmable machines capable of autonomous or semi-autonomous task execution, these devices mirror human actions with precision and efficiency. This project delves into the multifaceted world of robotic devices, exploring their evolution from simple automated machines to sophisticated smart devices.

Robotic devices, superior smart machines, have evolved significantly, finding applications in diverse fields such as manufacturing, healthcare, and exploration. Packed with advanced technology like sensors and computer smarts, these machines excel at accurate and precise task execution. They enhance processes in factories, assist in delicate surgeries, and venture into hazardous environments for information gathering, mitigating risks to human lives.

A robotic arm, mimicking the functionality of a human arm, comprises segments connected by joints for multi-directional movements. Equipped with an end effector, such as a gripper, it manipulates objects, relying on actuators and sensors for movement and feedback. George Devol, Jr.'s pioneering work in the 1950s marked a shift from robotic concepts in science fiction to tangible innovation. Robot arms, surpassing human arm capabilities, offer versatile movement and multifunctional tools, with some being mobile and autonomous to minimize errors and fatigue in repetitive tasks

Notable advantages of robotics include;

1. **Precision and Accuracy:**

   Robotic devices excel in executing tasks with unparalleled precision and accuracy, surpassing human capabilities. This attribute proves invaluable in industries requiring meticulous operations, such as manufacturing and healthcare.

2. **Enhanced Productivity:**

   The integration of robotic devices leads to a substantial boost in productivity. Their ability to work tirelessly, without the constraints of fatigue, results in continuous and efficient operation, streamlining processes across various sectors.

3. **Versatility and Adaptability:**

   These machines exhibit remarkable versatility, adapting to diverse tasks and environments. From assembly lines in manufacturing to complex surgical procedures in healthcare, robotic devices showcase adaptability that transcends traditional operational constraints.

4. **Safety in Hazardous Environments:**

   Robotic devices are designed to operate in environments hazardous to humans. This capability proves indispensable in scenarios involving toxic substances, extreme temperatures, or physically demanding conditions, ensuring increased safety for workers.

5. **Consistency in Performance:**

   One of the defining features of robotic devices is their consistent performance. Tasks are executed with unwavering precision and uniformity, mitigating variations that may arise from human factors.

6. **Remote Operation and Accessibility:**

   The potential for remote operation expands the reach of robotic devices. This feature allows tasks to be executed in locations that may be challenging or dangerous for humans to access directly, increasing accessibility and operational scope.

7. **Reduction in Manual Labour:**

   Robotic devices contribute to the reduction of manual labour, particularly in repetitive or strenuous tasks. This not only addresses workforce concerns but also allows human workers to focus on more intricate and cognitive aspects of their roles

1.2 - Statement of the problem.

In contemporary industries, challenges such as labour shortages, time inefficiencies, And the need for consistent precision have become increasingly prevalent. The robotic arm is positioned to address these issues by providing an automated solution that enhances productivity, accuracy, and overall operational efficiency**.**

1.3 - Aim:

To design and implement a hybrid robotic arm with pick and drop functionality.

1.4 - Objectives:

The following are the specific objectives of the project. To:

1. To design a robotic arm
2. To develop necessary software for the proper functioning of the microcontroller
3. To construct the robotic arm
4. To optimize the robotic arm system
5. To design a user-friendly interface for manual control, enabling seamless interaction and operation by human operators.

1.5 - Methodology:

The methodology to be adopted for the study includes;

1. Review previous work on robotic arm designs from articles, research papers, and other literature.
2. Develop software using the Arduino Programming Language for the Arduino Uno microcontroller, which interfaces with the various sensors and other components
3. Choosing components by researching and identifying appropriate materials for the project.
4. Designing a prototype of the robotic arm, ensuring it is reliable, safe and accurate through machine learning.
5. Testing the prototype and refinement.

1.6 - Expected results:

Upon successful implementation of the project focused on designing and constructing a robotic device (arm) with pick and drop functionality, the following outcomes are anticipated:

1. Precise Pick and Drop Operations: The robotic device is expected to showcase accuracy and precision in picking up items and placing them in designated locations based on their shapes and sizes.
2. Item Sorting Capability: The robotic arm is anticipated to effectively sort items based on predetermined criteria, demonstrating its ability to recognize and categorize objects according to their shapes and sizes.
3. Efficient Movement: The robotic device should show smooth movement capabilities, allowing it to navigate its environment seamlessly while carrying out pick and drop tasks.
4. Adaptability to Varied Objects: The robotic arm is expected to demonstrate versatility by effectively handling a diverse range of items, showcasing its ability to adapt to different shapes, sizes, and materials.
5. User Interface Integration: The project aims to include a user-friendly interface for controlling and programming the robotic device, allowing users to easily specify sorting criteria and coordinate pick and drop actions.

Chapter Two

2.1 Literature Review

The article "Smart Automated Robot Changing Tires using Ultrasonic Sensors" was written by Abdulrahman Alkandari, et al., 2023. The aim of the article is to design and develop a robotic system that can change tires on vehicles, which aims to address common issues faced by drivers such as tire punctures or bends in rims. The methodology used in the article involves the use of ultrasonic sensors and a motor to remove and install tires. The results and conclusions of the article suggest that the developed system has potential benefits such as reducing risks and dangers associated with changing tires, improving efficiency and effectiveness, and increasing driver safety. However, there may be limitations to the system such as cost and stability requirements, which may need to be addressed through ongoing development and improvement. Overall, the article provides valuable insights into the potential applications of robotics and automation in the field of mechatronics and mechanics, and highlights the importance of addressing common challenges faced by drivers and mechanics in the automotive industry.

In the paper by (Olawale et al. 2007) A microcontroller-based robotic arm that could pick and release small objects through magnetizing and demagnetizing was formed. 8051 microcontroller was interfaced with three stepper motors, each at the base, shoulder, and wrist of the anthropomorphic robot design. Magnetic mechanism was used for picking and releasing small objects, Arduino microcontroller and LabVIEW interface panel were utilized for control and precision. It was limited to small objects due to the 8051 microcontroller and the magnetic-based mechanism.

The article titled "Design and implementation of Arduino based robotic arm" was written by Hussein Mohammed Ali, 2022. The aim of the article is to design and implement an Arduino-based robotic arm that can be controlled using a mobile application. The methodology used in the article involves designing and building the robotic arm using Arduino, servo motors, and other electronic components. The results show that the robotic arm can be controlled using the mobile application and can perform various tasks such as picking and placing objects. The conclusion drawn from the study is that the Arduino-based robotic arm is an effective tool for performing various tasks. However, the article does not mention any limitations.

In their 2011 paper, (Elfasakhany et al.) proposed a low-cost design for a four Degree of freedom (DOF) robotic arm for less complex tasks, such as light material handling. This was to be made using thin acrylic sheets as lightweight and affordable material for manufacturing the robotic arm, whereas the Arduino microcontroller and LabVIEW software was interfaced with servo motors at each joint to perform inverse kinematics control. The maximum payload of the developed robotic arm was 50g with normal current consumption, which restricted its usage for industrial scenarios with higher payload requirements. This was Successfully designed and developed although further limited by its 4 DOF making it unsuitable for applications requiring higher number of Degree of Freedom.

The authors of the article "Design and Implementation of a Wireless Gesture
Controlled Robotic Arm with Vision" are Love Aggarwal, Varnika Gaur, and Puneet Verma, 2013. The aim of the article is to design and implement a wireless gesture-controlled robotic arm

with vision. The methodology used in the article involves using an RF module for communication between the robotic arm, platform, and user's hand and leg gestures, as well as an accelerometer for measuring gravitational force and real-time video streaming through an IP-based Android application. The result of the study is a successful implementation of the robotic arm that can pick up and place objects at desired locations based on user hand and leg gestures. The conclusion drawn from the study is that the use of wireless gesture control can improve the efficiency and accuracy of robotic arm movements. However, the limitations of the article include the limited range of the RF module and the use of only two axes of the accelerometer.

In the paper by (Hrutwik Dabhade eat al. 2022) Design of an automated system for detecting the color of objects and placing them in specific locations using a robotic arm was proposed. The system employs an Arduino Nano to control a robotic arm, driven by servo motors, and a conveyor belt equipped with sensors. The color of objects is detected using a TCS3200 Color Sensor. The process involves light intensity to frequency conversion for color detection. The Conveyor Belt starts and stops automatically with the help of sensors, facilitating the transportation of objects from the start to the end position. The system's accuracy could be impacted in conditions of high light reflection, leading to a decrease in performance.

The article "Design and Implementation of a Robotic Arm Based on Haptic Technology" was written by Rama Krishna, et al., 2012. The aim of the article is to present a design and implementation of a robotic arm based on haptic technology using an Arduino board. The methodology used in the article involves the use of potentiometers and servos attached to the

body of the robotic arm, with the potentiometers converting mechanical motion into electrical motion and the Arduino board processing the signals received from the potentiometers and converting them into digital pulses that are then sent to the servomotors. The result of the study is the successful design and implementation of a robotic arm based on haptic technology using an Arduino board. The conclusion drawn from the study is that the use of haptic technology in robotics can greatly enhance the user's experience and improve the accuracy and precision of the robotic arm's movements. However, the limitations of the article include the lack of detailed information on the specific applications and potential uses of the robotic arm, as well as the need for further research and development to improve the design and functionality of the arm.

The research paper by (Giannoccaro et al. 2013) outlines a robotic arm that can sort different objects using Radio Frequency Identification (RFID) tag. The proposed system sort objects based on an integrated RFID and webcam to capture the position and identity of an object, solve the inverse kinematics for joint configuration and then transmit the information to each of the five servo motors to achieve the desired end-effector position. This is practically restricted by the events of having objects of the same size and shape to be sorted using the webcam and assigning of identification tags to each object.

In the Paper by (Zain Ali get al. 2023), Development of a low-cost 5-DOF robotic arm for material handling and sorting in small manufacturing industries in Pakistan with a payload of 1kg was proposed. Used aluminum for lightweight, SOLIDWORKS for geometry, ANSYS for Finite Element Analysis (FEA), and Arduino for control were employed. The Finite Element

Analysis (FEA) was conducted for structural analysis. Applications requiring Higher Degree of Freedom were limitations to this design.

From the above literature reviews, below is a table of survey for different proposed robotic arm systems.

Table 2.1 – Survey for different proposed robotic arm systems.

S/N	TYPE	ADVANTAGES	LIMITATIONS
1	Micro-controller Operated Pick and Drop Robotic Arm.	Precise control, Affordable design, Low-cost.	Limited to small objects and payload.
2	Color Sorting Based Pick and Drop Robotic Arm.	Arduino UNO control and TCS3200 sensor for color detection allows for large scale arrange of objects which is Alot of strain on humans	Accuracy can be affected by environmental factors, such as light intensity. And sometimes require expensive image processing hardware.
3	Degree of Freedom (DOF) Operated Pick and Drop Robotic Arm	Low cost and light weight design.	Limited payload, and unsuitable for higher DOF applications.
4	Radio Frequency Identification (RFID) - Based Pick and Drop Robotic Arm	RFID and webcam integration, Sorts objects based on RFID tags.	Limited to objects with same size and shape.
5	Smart Automated Robot Changing Tires using Ultrasonic Sensors	Risk Reduction, Enhanced Efficiency and Effectiveness, Increased Driver Safety.	It is expensive, Requires a very stable surface.
6	Wireless Gesture Controlled Robotic Arm with Vision	Wireless Gesture Control, Real-Time Video Streaming, Vision Integration	Limited RF Module Range, limited to two axes of the Accelerometer

2.2 Review of Components

A component is a distinguishable part of a larger project or structure. Typically, a component carries out a single, well-defined task or a group of related tasks. The selection and examination of the components needed to build a Hybrid Pick and Drop Robotic Arm will be the main topic of this section.

Below is a list of components used in the project:

1. Atmega382p microcontroller
2. Infrared (IR) proximity sensor
3. TSOP1338 IR receiver
4. AVR programmer
5. Crystal Oscillators
6. Metallic robotic arm frame and Servo Motors
7. DC-DC buck converter
8. RF Module

2.2.1 Atmega382p microcontroller

The Atmega382p as shown in figure 2.1 and figure 2.2 below is a powerful and versatile microcontroller well-suited for various applications. Its ease of use, affordability, and extensive support make it a popular choice for beginners and experienced developers alike. It is an 8-bit microcontroller that belongs to the megaAVR family. Developed by Atmel (later acquired by Microchip Technology in 2016).

The ATmega328P microcontroller is distinguished by the "P" signifying its incorporation of Atmel's picoPower technology for low-power applications. It offers versatility for various power supply designs, operating over a broad voltage range of 1.8V to 5.5V. It is noteworthy for providing an adaptable Brownout Detector (BOD) with configurable settings that let developers adjust power monitoring thresholds to suit their project needs. Moreover, a wide operating temperature range of the ATmega328P guarantees dependable functioning in a variety of environmental settings. With 2KB of SRAM for data storage, 1KB of EEPROM for non-volatile data storage, and 32KB of Flash memory for program storage, it offers a sizable memory capacity for a variety of applications.

The ATmega328, on the other hand, is an older version and does not have the "picoPower" designation. While it is comparable to the ATmega328P, there are some small changes. The voltage range may be less vast, and the Brownout Detector settings may provide fewer customizing choices. The operating temperature range could be slightly reduced, though specific specifics may differ depending on the version. Similar to the ATmega328P, the ATmega328 has 32KB of Flash memory, 2KB of SRAM, and 1KB of EEPROM, providing enough memory capacity for a variety of applications.

Due to their prominent placement on well-known boards such as the Arduino Uno, both microcontrollers have played crucial roles in the Arduino development platform. The decision between the ATmega328P and ATmega328 depends on the particular needs of the project. For low-power applications and projects with varying power supply requirements, the "P" version is recommended.

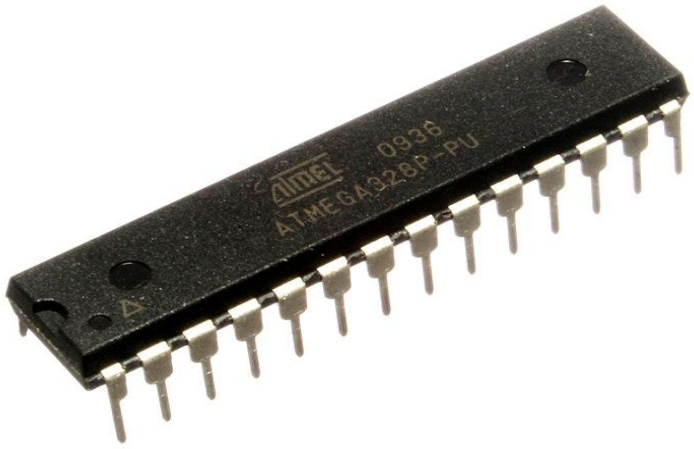


Figure 2.1 - An image of the ATMega328P Microcontroller (https://en.wikipedia.org/wiki/ATmega328)

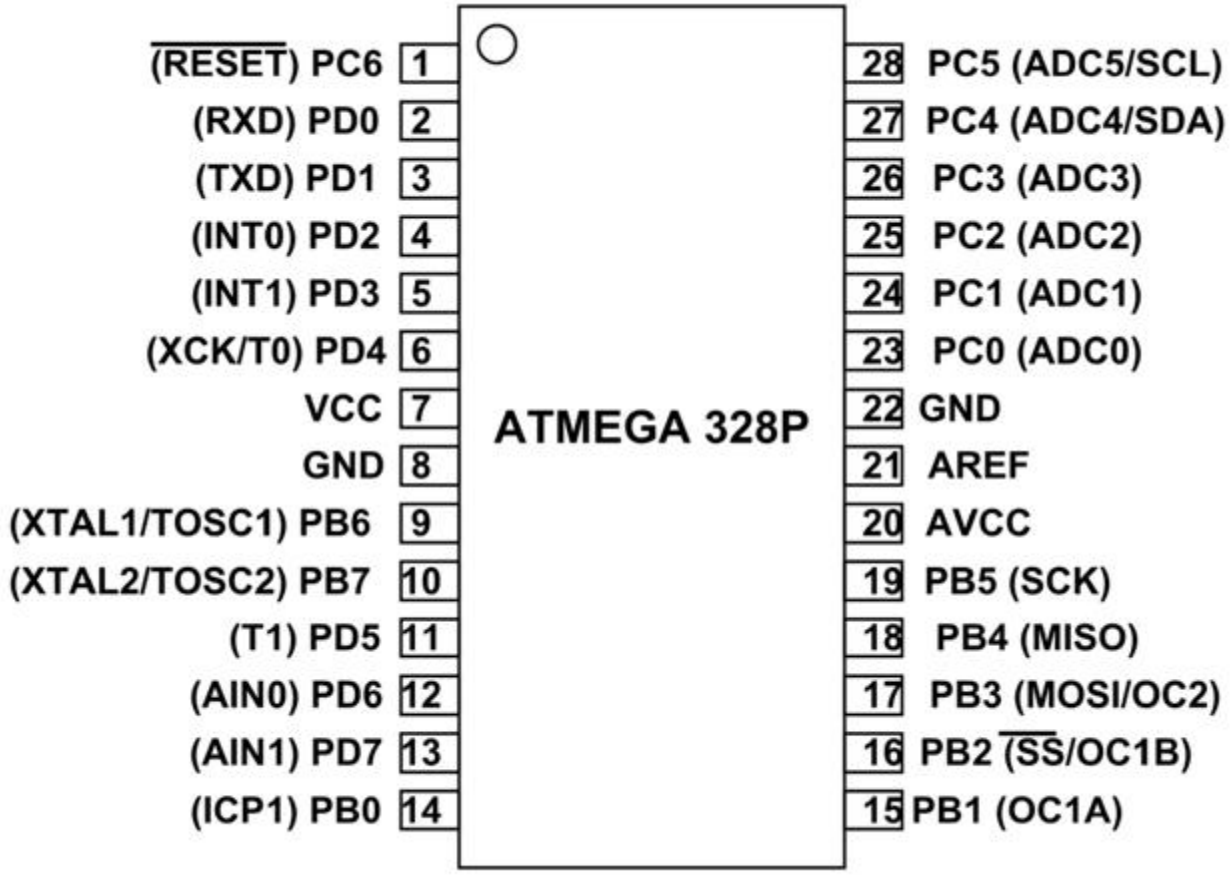


Figure 2.2 - Diagram of the ATMega328P pinout (https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet)

ATMega328 Pinout Configuration

ATMEGA328P is a 28-pin chip as shown in pin diagram above. Many pins of the chip here have more than one function. We will describe the functions of each pin in the table below.

Table 2.2 – Pin numbers of ATMega328P pinout and Special Functions (*https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet*)

Pin No.	Pin name	Description	Secondary Function
1	PC6 (RESET)	Pin6 of PORTC	Pin by default is used as RESET pin. PC6 can only be used as I/O pin when RSTDISBL Fuse is programmed.

2	PD0 (RXD)	Pin0 of PORTD	RXD (Data Input Pin for USART) USART Serial Communication Interface [Can be used for programming]
3	PD1 (TXD)	Pin1 of PORTD	TXD (Data Output Pin for USART) USART Serial Communication Interface [Can be used for programming] INT2(External Interrupt 2 Input)
4	PD2 (INT0)	Pin2 of PORTD	External Interrupt source 0
5	PD3 (INT1/OC2B)	Pin3 of PORTD	External Interrupt source1 OC2B(PWM - Timer/Counter2 Output Compare Match B Output)
6	PD4 (XCK/T0)	Pin4 of PORTD	T0(Timer0 External Counter Input) XCK (USART External Clock I/O)
7	VCC		Connected to positive voltage
8	GND		Connected to ground
9	PB6 (XTAL1/TOSC1)	Pin6 of PORTB	XTAL1 (Chip Clock Oscillator pin 1 or External clock input) TOSC1 (Timer Oscillator pin 1)
10	PB7 (XTAL2/TOSC2)	Pin7 of PORTB	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 2)
11	PD5 (T1/OC0B)	Pin5 of PORTD	T1(Timer1 External Counter Input) OC0B(PWM - Timer/Counter0 Output Compare Match B Output)

12	PD6 (AIN0/OC0A)	Pin6 of PORTD	AIN0(Analog Comparator Positive I/P) OC0A(PWM - Timer/Counter0 Output Compare Match A Output)
13	PD7 (AIN1)	Pin7 of PORTD	AIN1(Analog Comparator Negative I/P)
14	PB0 (ICP1/CLKO)	Pin0 of PORTB	ICP1(Timer/Counter1 Input Capture Pin) CLKO (Divided System Clock. The divided system clock can be output on the PB0 pin)
15	PB1 (OC1A)	Pin1 of PORTB	OC1A (Timer/Counter1 Output Compare Match A Output)
16	PB2 (SS/OC1B)	Pin2 of PORTB	SS (SPI Slave Select Input). This pin is low when controller acts as slave. [Serial Peripheral Interface (SPI) for programming] OC1B (Timer/Counter1 Output Compare Match B Output)
17	PB3 (MOSI/OC2A)	Pin3 of PORTB	MOSI (Master Output Slave Input). When controller acts as slave, the data is received by this pin. [Serial Peripheral Interface (SPI) for programming] OC2 (Timer/Counter2 Output Compare Match Output)
18	PB4 (MISO)	Pin4 of PORTB	MISO (Master Input Slave Output). When controller acts as slave, the data is sent to master by this controller through this pin. [Serial Peripheral Interface (SPI) for programming]
19	PB5 (SCK)	Pin5 of PORTB	SCK (SPI Bus Serial Clock). This is the

			clock shared between this controller and other system for accurate data transfer. [Serial Peripheral Interface (SPI) for programming]
20	AVCC		Power for Internal ADC Converter
21	AREF		Analog Reference Pin for ADC
22	GND		GROUND
23	PC0 (ADC0)	Pin0 of PORTC	ADC0 (ADC Input Channel 0)
24	PC1 (ADC1)	Pin1 of PORTC	ADC1 (ADC Input Channel 1)
25	PC2 (ADC2)	Pin2 of PORTC	ADC2 (ADC Input Channel 2)
26	PC3 (ADC3)	Pin3 of PORTC	ADC3 (ADC Input Channel 3)
27	PC4 (ADC4/SDA)	Pin4 of PORTC	ADC4 (ADC Input Channel 4) SDA (Two-wire Serial Bus Data Input/output Line)
28	PC5 (ADC5/SCL)	Pin5 of PORTC	ADC5 (ADC Input Channel 5) SCL (Two-wire Serial Bus Clock Line)

Below is a table showing the simplified features of the ATMega328P

Table 2.3 – Features of ATMega328P (*https://components101.com/microcontrollers/atmega328p-pinout-features-datasheet*)

ATMEGA328P – Simplified Features	
CPU	8-bit AVR
Number of Pins	28

Operating Voltage (V)	+1.8 V TO +5.5V
Number of programmable I/O lines	23
Communication Interface	Master/Slave SPI Serial Interface(17,18,19 PINS) [Can be used for programming this controller] Programmable Serial USART(2,3 PINS) [Can be used for programming this controller] Two-wire Serial Interface(27,28 PINS)[Can be used to connect peripheral devices like Servos, sensors and memory devices]
JTAG Interface	Not available
ADC Module	6channels, 10-bit resolution ADC
Timer Module	Two 8-bit counters with Separate Prescaler and compare mode, One 16-bit counter with Separate Prescaler, compare mode and capture mode.
Analog Comparators	1(12,13 PINS)
DAC Module	Nil
PWM channels	6

External Oscillator	0-4MHz @ 1.8V to 5.5V 0-10MHz @ 2.7V to 5.5V 0-20MHz @ 4.5V to 5.5V
Internal Oscillator	8MHz Calibrated Internal Oscillator
Program Memory Type	Flash
Program Memory or Flash memory	32Kbytes[10000 write/erase cycles]
CPU Speed	1MIPS for 1MHz
RAM	2Kbytes Internal SRAM
EEPROM	1Kbytes EEPROM
Watchdog Timer	Programmable Watchdog Timer with Separate On-chip Oscillator
Program Lock	Yes
Power Save Modes	Six Modes[Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby]
Operating Temperature	-40°C to +105°C(+105 being absolute maximum, -40 being absolute minimum)

2.2.2 Infrared (IR) proximity sensor

A proximity sensor that uses infrared radiation to determine whether an object is within its sensing range is called an infrared (IR) sensor. Numerous industries, including consumer electronics, robotics, and automation, employ this kind of sensor extensively.

Infrared (IR) sensors as presented in figure 2.3 below are electrical parts made to identify specific radiation wavelengths, such as heat. Devices with infrared sensors can detect their surroundings and adjust their behavior accordingly. They function by picking up infrared radiation that are released by an individual or thing that comes into contact with the sensing region of an IR sensor-equipped equipment.

After processing, this data is used for a variety of tasks, such as detecting nearby objects, figuring out whether something is moving in its immediate environment, or even guiding robots toward a desired location using detection distance measurement data supplied by the infrared signal detected at each point along its trajectory path within visual range.

Here are the key aspects of an IR proximity sensor:

1. **Principle of Operation:**
    - The idea behind how infrared proximity sensors function is that they pick up infrared radiation that an item emits or reflects. Usually, these sensors are made up of a photodiode or phototransistor and an infrared LED (Light Emitting Diode).

2. **Emitter and Receiver Pair:**

    - The emitter of the sensor emits infrared light, and the radiation that is reflected or emitted is detected by the receiver. The quantity of infrared light that is reflected or emitted varies when an item is within the sensor's field of view.

3. **Working Modes:**

    There are two primary types of operation for infrared proximity sensors: transmissive and reflecting.

    - In the reflective mode, the sensor picks up infrared light reflected from a nearby object. The emitter and receiver are contained in the same device.
    - In the transmissive mode, an object is identified when it breaks the infrared beam between the emitter and receiver, which are located at a distance from one another.

4. **Distance Sensing:**

    - Depending on the particular sensor model, infrared proximity sensors work well for short- to moderate-range distance sensing, which is normally between a few centimeters and a few meters.

5. **Applications:**

    - IR proximity sensors are frequently used in robotics obstacle detection, consumer electronics touchless switches, proximity-activated interfaces, and automated hand sanitizers. They are also used for object sorting and detection in industrial automation.

6. **Adjustable Sensitivity:**
    - A lot of infrared proximity sensors provide an adjustable sensitivity feature that lets users customize the sensor's response to meet the unique needs of their application.

7. **Advantages:**
    - Infrared proximity sensors are renowned for their dependability, affordability, and adaptability. They are appropriate for both indoor and outdoor applications since they perform well in a range of lighting conditions.

8. **Limitations:**
    - The effectiveness of infrared proximity sensors can be impacted by surface reflection and interference from ambient light. Furthermore, in comparison to other sensing technologies like LiDAR or ultrasonic, their range is restricted.

9. **Integration with Microcontrollers:**
    - IR proximity sensors can be seamlessly integrated into electronic projects since they interface with microcontrollers like Arduino and Raspberry Pi with ease.

In summary, infrared proximity sensors are essential for presence and object recognition and find use in a variety of commercial and household electronic products. Their versatility, dependability, and ease of use make them a well-liked option for proximity sensing applications.

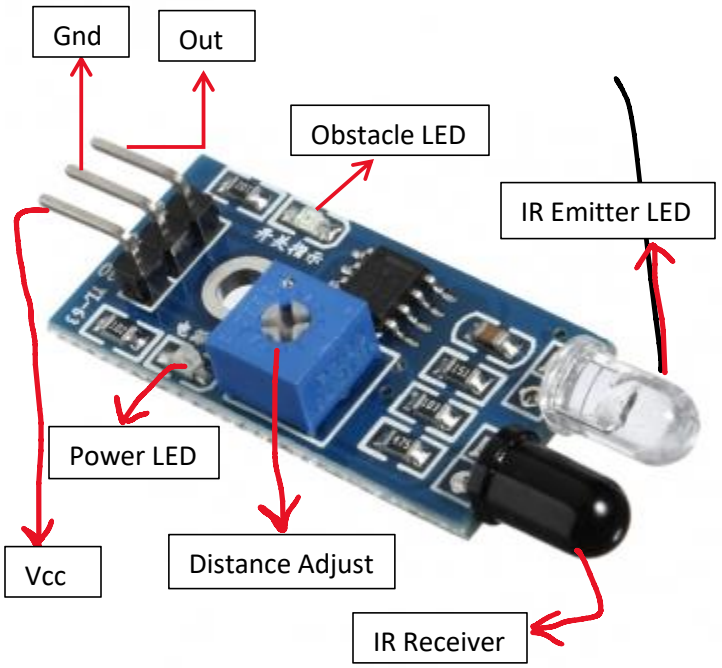


Figure 2.3 - Image of an Infrared (IR) proximity sensor (https://hub360.com.ng/product/duplicate/)

The Infrared (IR) proximity sensor is further analyzed by the table below:

Table 2.4 – Description of Infrared (IR) proximity sensor (*https://www.sunrom.com/p/infrared-sensor-board*)

Pin, Control Indicator	**Description**
Vcc	3.3 to 5V dc supply input
Gnd	Ground input
Out	Output that goes low when obstacle is in range

Power LED	Illuminates when power is applied
Obstacle LED	Illuminates when obstacle is detected
Distance Adjust	Adjust detection distance. CCW decreases distance. CW increases distance.
IR Emitter	Infrared emitter LED
IR Receiver	Infrared receiver that receives signal transmitted by infrared emitter.

2.2.3 TSOP1338 IR Receiver

An IR receiver, sometimes referred to as an infrared receiver module, is an electronic gadget that detects and decodes infrared (IR) signals sent by an IR remote control or other IR emitter. Although these impulses are undetectable to the human eye, an infrared receiver as shown in figure 2.4 below can detect them and transform them into electrical signals that are understandable by electronics.

The TSOP1338 is an infrared (IR) receiver module frequently used in electrical projects to receive modulated infrared (IR) signals. It is specifically tuned to a carrier frequency of approximately 38 kHz, which is commonly used in infrared remote controllers. It works by demodulating incoming infrared signals, extracting data from the carrier frequency, and producing a digital output signal, which is usually low when there is no signal and high when there is a valid infrared signal. consisting of three pins: OUT (output), GND (ground), and VCC (power supply). It is adaptable and can be used for things like communicating with other devices or receiving signals from a remote control.

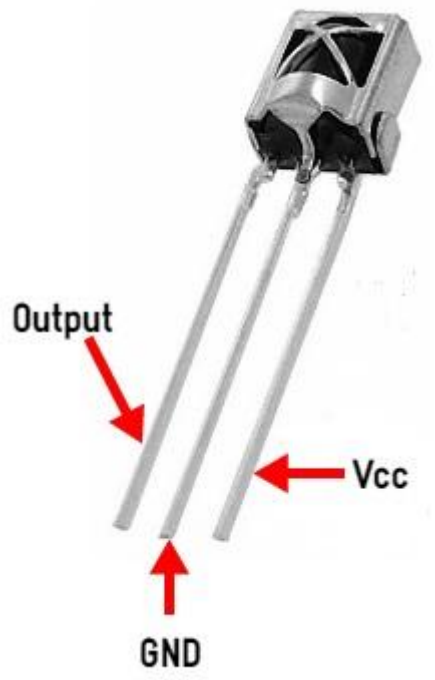


Figure 2.4 - Image of an IR Receiver (https://qxf2.com/blog/arduino-tutorials-for-testers-decoding-of-ir-receiver/)

2.2.4 AVR Programmer

A tool or device used to program Atmel AVR microcontrollers is referred to as an AVR programmer. Widely utilized in a variety of embedded systems, the AVR series of microcontrollers was created by Atmel, which is currently owned by Microchip Technology. To enable the AVR microcontroller to carry out the programmed instructions, compiled code can be loaded into it using an AVR programmer. To program the microcontroller, you must connect the AVR programmer to both the microcontroller and your computer, transfer the produced code, and set up a few parameters.

Types of AVR Programmers

1. **Parallel Programmers:** These AVR programmers as presented in figure 2.5a below uses a parallel Port on a computer for communication. This type is gradually exiting the market due to less availability of parallel ports on modern computers.

Figure 2.5a - Image of a parallel port AVR programmer (http://criticalvelocity.com/item.php?itemid=prog1 /)

2. **Serial Programmers:** As shown in figure 2.5b below, these are currently the most prevalent in contemporary systems making connection to a computer through either a serial port of USB-to-serial converter.

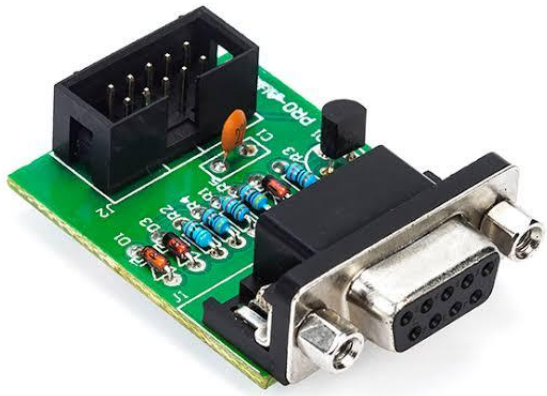


Figure 2.5b - Image of a serial AVR programmer (https://simplemetaldetector.com/programmers/avr-serial-programmer/)

3. **USB AVR Programmers**: These type of AVR programmer as depicted in figure 2.5c below connects directly to a computer through a USB port, eliminating the need for additional adapters. They are a very convenient type of AVR programmer.

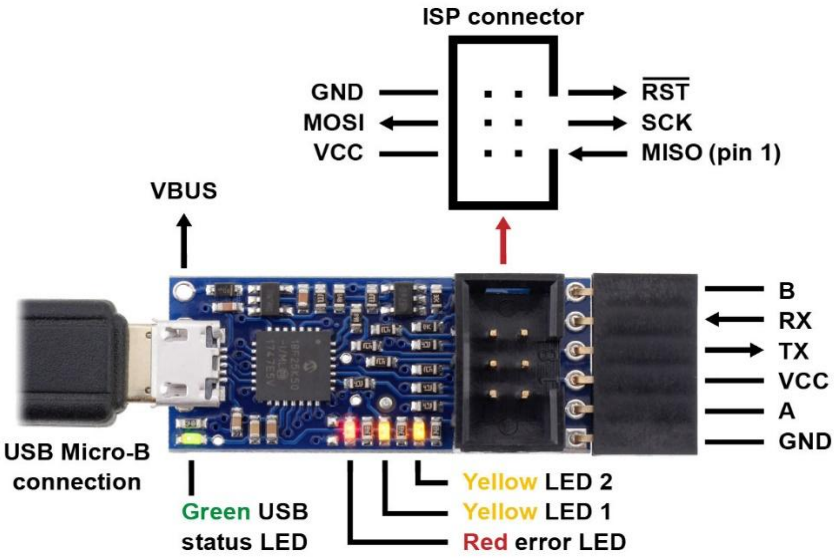


Figure 2.5c - Image of a USB AVR Programmer (https://www.pololu.com/product/3172)

2.2.5 Crystal Oscillators

A crystal oscillator as shown in figure 2.6 below is a circuit that utilizes a piezoelectric crystal—typically a quartz crystal—as a frequency-selective component. As a resonator, the crystal vibrates mechanically, and the oscillation frequency is determined by the vibration frequency of the crystal. Compared to LC or RC oscillators, crystal oscillators have substantially higher frequency stability due to their extremely high Q-factor and superior temperature stability over tuned circuits. They produce the clock signal in quartz clocks and computers, and they steady the frequency of most radio transmitters. With the crystal in place of the tuned circuit, crystal oscillators frequently employ the same circuitry as LC oscillators. Another type of piezoelectric resonator utilized in crystal oscillators that can produce significantly higher frequencies is the surface acoustic wave (SAW) device. They are employed in specific applications, such mobile phones, that call for a high frequency reference.

Working Principle

The working principle of a crystal oscillator is based on the piezoelectric effect. When an alternating voltage is applied to a crystal, it vibrates at its natural resonant frequency. The crystal's mechanical resonance induces an alternating electrical signal, and this process is sustained, resulting in a continuous oscillation with a stable frequency determined by the crystal's physical characteristics.

Various types of the Crystal Oscillator exist, ranging from:

- **TCXO (Temperature Compensated Crystal Oscillator):** which are crystal Oscillators that include temperature compensation circuitry for improved frequency stability.
- **OCXO (Oven-Controlled Crystal Oscillator):** Which Maintains a stable temperature environment to achieve extremely high frequency stability.
- **VCXO (Voltage-Controlled Crystal Oscillator)**: Which allows tuning of frequency through a control voltage, and are often used in network switches, 5G Radio and so on; down to
- **MCXO (Microcomputer Compensated Crystal Oscillator)**: a Crystal Oscillator controlled by a microcontroller.

These are used in various applications such as:

- **Communication Systems:** including radio transceivers, cellular networks, and satellite communication, ensuring precise timing synchronization.
- **Measurement Instruments:** requiring accurate timekeeping, such as oscilloscopes and frequency counters, often employ crystal oscillators.
- **Global Positioning System (GPS):** Crystal oscillators contribute to accurate timekeeping in GPS receivers.
- **Consumer Electronics**: like clocks, watches, and digital cameras, providing accurate timekeeping; and

- **Military and Aerospace:** relying on crystal oscillators for precise timing and synchronization.

Crystal Oscillator Specifications

For a basic Crystal Oscillator, the following specifications are usually given:

Crystal Frequency, Crystal Frequency Tolerance, Operating Temperature Range, Aging, Load Capacitance, Drive Level, Start-up Time, Output Waveform, Aging Rate, RMS phase jitter, Frequency Control, Power Supply Voltage, Output Load, Package Type, and Avalanche Transistor Type (AT-Cut or SC-Cut)

Figure 2.6 - Image of a 16MHz Crystal Oscillator (https://academic-accelerator.com/encyclopedia/crystal-oscillator)

2.2.6 - Metallic Robotic Arm Frame and Servo Motors

The metallic robotic arm frame as presented in figure 2.7 below provides the structural support for the robotic arm. It is usually made up of metallic frame ensuring stability of the system and Servo motors which are actuator devices responsible for precise control of the arm's movement.

Working Mechanism of Servo Motors:

At the core of a servo motor is a small direct current (DC) motor, similar to what you might find in a toy. These motors operate on DC supply sources spinning at high RPM (rotations per minute) but putting out very low torque.

There is a gear arrangement in the servo motor which takes the high speed of the motor and convert it to a slower rotational output speed but with more torque. Which establishes the movements. How far the user wants the servo to rotate, the direction to rotate the shaft to get the precise position are achieved with the servo motor through the aid of a positional sensor on the final gear of the Servo motor interfaced with a small circuit board.

There are three basic types of Servo Motors

1. Positional Rotation Servo - Which rotate at about 180°
2. Continuous Rotation Servo - Rotates in either direction indefinitely
3. Linear Servo - Similar to Positional Rotation Servo but with more gears and are rare to find.

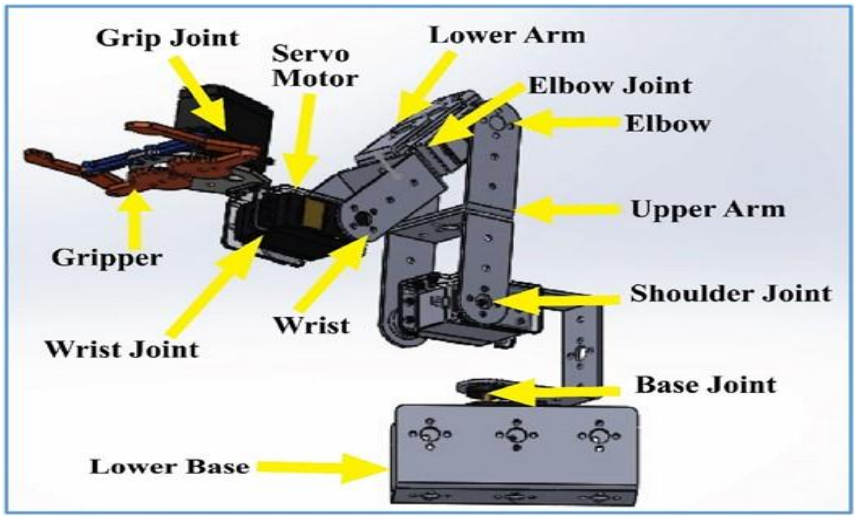


Figure 2.7 - Image of a robotic arm (https://www.sciencedirect.com/science/article/pii/S2590123023004425)

Figure 2.8 - Image of a Servo Motor (https://store.fut-electronics.com/products/micro-servo-motor-1-3kg-cm)

2.2.7 - DC-DC Buck Converter

A DC-DC Buck Converter, also known as a step-down converter, is an electronic circuit that transforms a higher input voltage to a lower output voltage with increased current. It is a type of switching power supply commonly used to efficiently regulate and step-down voltage levels in electronic devices.

The following features are associated with it:

- **Compact Size:** They are often compact and lightweight, making them suitable for applications with space constraints.
- **Fast Transient Response**: They provide rapid response to changes in load conditions, maintaining stable output voltage.
- **Regulation**: They offer voltage regulation, ensuring a consistent output voltage despite variations in input voltage or load.
- **Low Heat Dissipation**: Compared to linear regulators, buck converters generate less heat during operation, contributing to energy efficiency.

- **Cost-Effective**: They are cost effective ways of stepping down voltage.

Working Principle

The DC-DC Buck converter as shown in figure 2.9 and figure 2.10 below works by switching between the ON and OFF states of a semiconductor switch (usually a transistor) with energy storage elements- inductor and capacitor. When it is in the ON state, energy is stored in the inductor, and when it is in the OFF state, the energy is released to the load. By controlling the ON-OFF duration, the output voltage of the device is regulated.

It finds application in

- **Battery-Powered Devices:** such as smartphones, laptops, and portable electronics, etc. to efficiently step down the battery voltage.
- **Voltage Regulation**: They used in various voltage regulation circuits for microcontrollers, FPGAs, and other integrated circuits where a stable voltage is crucial for proper operation.
- **LED Drivers**: to regulate the voltage and current supplied to LEDs.
- **Solar Power Systems:** to step down the voltage generated by solar panels to the required level for charging batteries. And much more.

Figure 2.9 - Image of a DC-DC Buck Converter (https://m.indiamart.com/proddetail/dc-to-dc-buck-converter-22978847897.html?pos=3&pla=n)

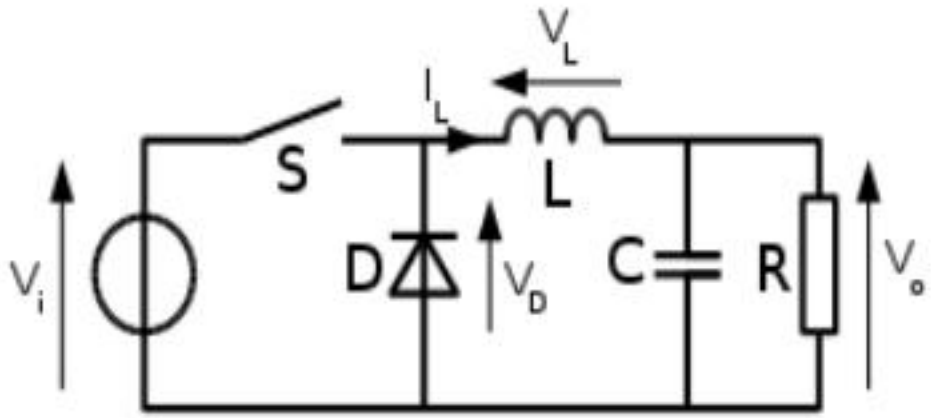


Figure 2.10 - Diagrammatic Representation of DC-DC Buck Converter (https://www.semanticscholar.org/paper/Design-and-implementation-of-MPPT-solar-system-on-Gaga-Errahimi/21ea616973c00316f67b2efd398d9d6434172aa1)

2.2.8 – RF Module

An RF (Radio Frequency) module is an electronic device used to transmit or receive radio signals wirelessly. It converts electrical signals into radio waves and incoming radio signals into electrical signals depending on it's built.

It can be used for various purposes such as

- Remote control

- Wireless data transmission

- Telemetry etc.

In this Project, the Sony Game Controller (PS2 controller) acts as our RF MODULE, functioning as a remote control for the robotics arm. The Image below shows the Sony Game Controller.

Figure 2.11 – Image of the Sony game controller (https://electropeak.com/learn/tutorial-interface-wireless-playstation-ps2-controller-arduino/)

The Sony Game Controller has:

- 12 analog keys sensitive to pressure, (4 keys for direction, 4 operation keys, Cross, Triangle, Circle, and Square, L1, L2, R1, and R2) and;

- 5 digital keys (MODE, START, SELECT, R3, L3) and 2 analog joysticks.

There are 2 motors in the controller that can make vibration because of their imbalance.

The wireless controller works with 2,4GHz frequency and it has a range of 10 meters. It also has an optical indicator for sending and receiving data. This controller needs only 3 AAA batteries for power (in some cases it needs just 2 AAA batteries).

2.3 Software/Programming language used

The software used during the course of this project is Arduino IDE. The IDE and its compiler were important as it aided the smooth running of the written codes in the Atmega328p microcontroller for the Robotic arm manipulation. The Arduino makes use of either C language or C++ language but C language was used because: it enables easy interaction with the low-level hardware components and Robotic arms have memory limitations.

2.3.1 - Introduction to C language

C is a general-purpose programming language, and is used in this project to achieve our aim. It is a small language, with just 32 keywords. It provides "high-level" structured programming constructs such as **statement grouping, decision making,** and **looping**, as well as "low- level" capabilities such as the ability to manipulate bytes and addresses.

Much of the functionality of C is provided by way of software routines called **functions**.

The C programming language consists of the **main ()** function within which the entire code is written. These **functions** are code snippets that ask the computer to process instructions and give the correct output. The basic C program structure consists of **declared variables, constants,** and **functions** that is executed to get the desired output. The C programming language also consists of entities termed as **identifiers** that are nothing but l**abels, arrays, functions, variables, structures** and **unions**.

The language is accompanied by a standard library of functions that provide a collection of commonly used operations.

2.3.2 - Brief History of C language

The C programming language was developed in the Bell Labs of AT&T by an employee called Dennis Ritchie between 1969 and 1973 while working on Unix operating system at Bells Lab in New Jersey. Dennis Richie is known as the founder of C language.

He created this language using ALGOL, BCPL, and B the languages that were used before C was created. He added many powerful features to C in order to overcome the problems of previous languages such as B, BCPL, etc. He then used it to further develop the UNIX operating system. American National Standards Institute (ANSI) in 1983, formed a committee to provide a comprehensive definition to the C language and thus came into existence the new ANSI C language with better features.

2.3.3 - Structure of C language

C is a typed language. Each variable is given a specific type which defines what values it can represent, how its data is stored in memory, and what operations can be performed on it.

The structure of C language consists of six important segments which makes it easy to read, modify, document, and understand in a particular format. C program must follow the below-mentioned outline in order to successfully compile and execute. Debugging is easier in a well-structured C program.

These are documentation section, link section, definition section, global declaration section, main() function, and other executable functions of the program (as shown in figure 2.11 below).

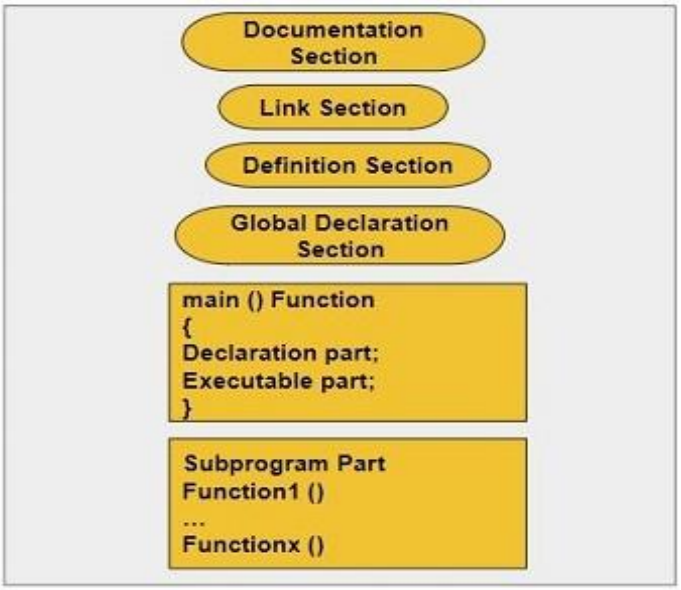


Figure 2.12 – Structure of a C program (https://hktsoft.net/c-programming-language-history-features-structure-and-why-study-it/)

Documentation:

This section consists of the description of the program, the name of the program, and the creation date and time of the program. It is specified at the start of the program in the form of comments. Documentation can be represented as:

// description, name of the program, programmer name, date, time etc.

or

*/**

description, name of the program, programmer name, date, time etc.

**/*

Link Section:

All the header files of the program will be declared in the link section of the program. Header files help us to access other's improved code into our code. A copy of these multiple files is inserted into our program before the process of compilation.

Example;

#include<stdio.h>

#include<math.h>

Definition:

Link/Preprocessors are the programs that process our source code before the process of compilation. There are multiple steps which are involved in the writing and execution of the program. Preprocessor directives start with the '#' symbol. The #define preprocessor is used to create a constant throughout the program. Whenever this name is encountered by the compiler, it is replaced by the actual piece of defined code.

Example; *#define long long ll*

Global Declaration:

The global declaration section contains global variables, function declaration, and static variables. Variables and functions which are declared in this scope can be used anywhere in the program.

Example; *int num = 18;*

Main() Function:

Every C program must have a main function.

The main() function of the program is written in this section. Operations like declaration and execution are performed inside the curly braces of the main program. The return type of the main() function can be int as well as void too. void() main tells the compiler that the program will not return any value. The int main() tells the compiler that the program will return an integer value.

Example;

int main() **or**

void main()

Sub Programs:

User-defined functions are called in this section of the program. The control of the program is shifted to the called function whenever they are called from the main or outside the main() function. These are specified as per the requirements of the programmer.

Example;

```
int sum(int x, int y)
{
   return x+y;
}
```

Below is an example of a C program to find the sum of two numbers: (the comments in C start with /* and are terminated with */. They can span multiple lines and are not nestable).

```
// Documentation
/**
 * file: sum.c
 * author: you
 * description: program to find sum.
*/

// Link
#include <stdio.h>

// Definition
#define X 20

// Global Declaration
int sum(int y);

// Main() Function
```

```
int main(void)
{
  int y = 55;
  printf("Sum: %d", sum(y));
  return 0;
}

// Subprogram
int sum(int y)
{
  return y + X;
}
```

Output= 75

Chapter Three

Methodology

The methodology employed in the development of the Hybrid Pick and Drop Robotic Arm involved a systematic and comprehensive approach to address the outlined objectives. This section describes the entire process in detail. Every phase is elaborated extensively to give a comprehensive picture of the decision-making procedures used to guarantee the project's success. Our method is based on a combination of electronic and mechanical parts as well as creative control algorithms. In order to provide a comprehensive understanding of the development process, ethical issues and teamwork are also explored. This overview aims to provide a roadmap of the methodology section, offering insights into the intricacies of creating a versatile and efficient robotic arm system. Therefore, as shown below; a block diagram illustrating how the various components are being connected to achieve the desired output efficiently.

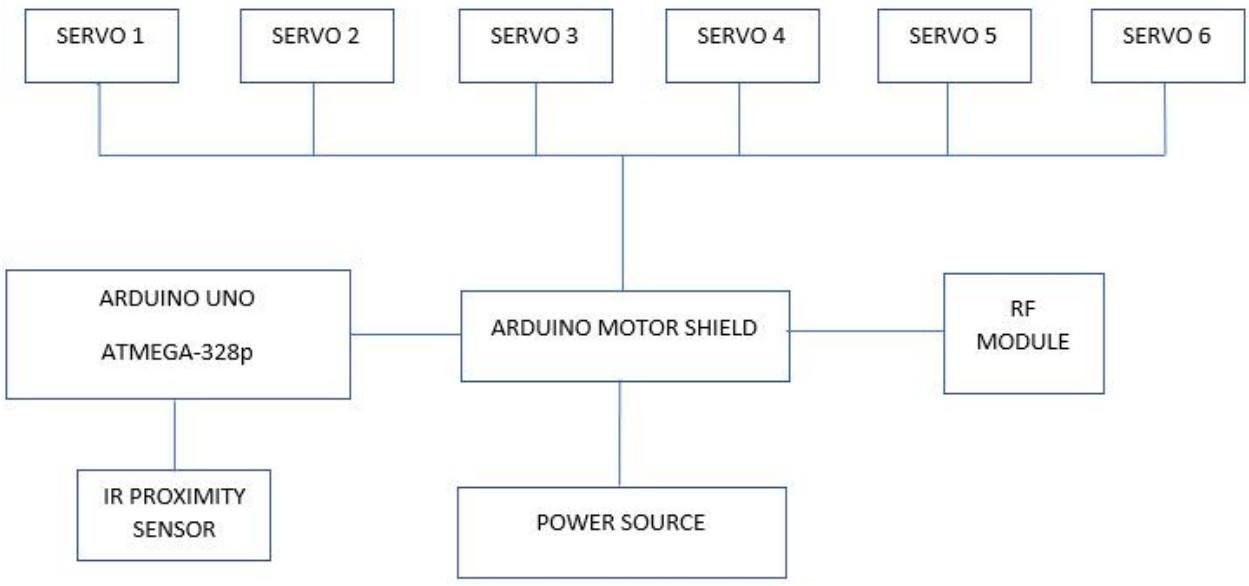


Figure 3.1 – Block diagram showing the components connection

3.1 - Power Source

The efficient and reliable operation of the robotic arm hinges on the availability of a robust power supply. As the primary source of electrical energy, the power source plays a pivotal role in ensuring smooth functionality and precise control of the arm's movements. From powering servo motors for joint articulation to energizing sensors for environment perception, every aspect of the robotic arm's operation is intricately tied to the performance of its power supply system.

The power supply unit is built by the parallel and series arrangement of twelve 3.7 ~ 4.2V Li-ion batteries to supply 11.1V at full charge capacity and 12.6 at maximum charge capacity respectively. Below is the circuit diagram showing the arrangement of the batteries to give the required voltage.

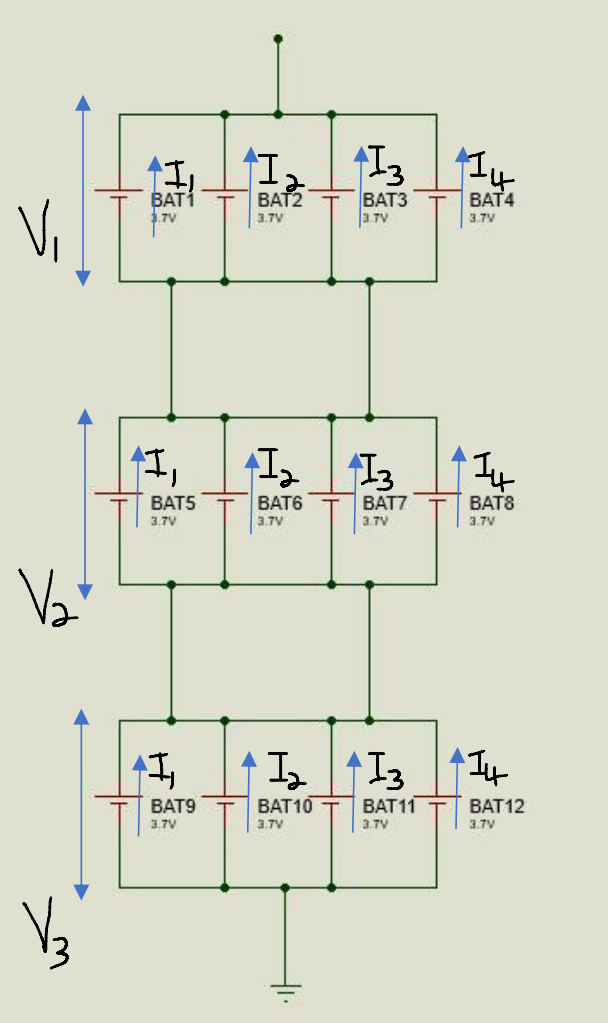


Figure 3.2 – Diagram showing the arrangement of the batteries

Design Consideration

1. Li-ion Battery cells (twelve pieces)
   Voltage: 3.7V
   Capacity: 7800mAh

2. Li-ion Charger
   Output: DC 12.6V ⎓ 10A
3. Servo Motor (six)
   Operating voltage: 4.8 ~ 7.2V
   Operating Current: 500 mA ~ 900mA (6V)

4. Arduino Uno
   Operating voltage: +1.8 V ~ +5.5V
   Operating current: 0.3mA (1.8V)

5. IR Proximity Sensors (three)
   Operating voltage: 3.3 ~ 5V
   Operating current: 5.3mA (3.3V)

Equivalent Voltage of Battery

From figure 3.2 above, we have four battery cells connected in parallel, and in series with two other four battery cells connected in parallel, the equivalent voltage can be calculated as shown below.

Where: $V_1 = V_2 = V_3 = 3.7V$

$$V_{eq} = V_1 + V_2 + V_3$$

$$= 3.7 + 3.7 + 3.7$$

$$= 11.1V$$

Equivalent Current of Battery

Where: $I_1 = I_2 = I_3 = 7800mAh$

$$I_{eq} = I_1 + I_2 + I_3 + I_4$$

$$= 7800 + 7800 + 7800 + 7800$$

$$= 31200mAh$$

Energy Available in The Battery

$$P = V_{eq} \times I_{eq}$$

$$= 11.1 \times 31.2$$

$$= 346.32Wh$$

Charging Capacity

The charging capacity illustrate how much time it takes for the charger to fully charge the battery this can be calculated as shown below.

Since the output for the charger is DC 12.6V ⎓ 10A, then the power can be calculated by

$$P = V \times I$$

$$= 12.6 \times 10$$

$$= 126W$$

$$Time\ to\ charge = \frac{Energy\ available\ in\ battery}{Power\ of\ the\ charger}$$

$$= \frac{346.32}{126}$$

$$= 2.75h\text{rs}$$

$$= 9900sec$$

Energy Consumption

The Hybrid Pick and Drop Robotic Arm's energy consumption is the total amount of electrical energy used by all of its components while they are in use. Understanding the amount of energy that the robotic arm uses is essential to maximizing its effectiveness and lifespan of its power supply.

- **For Servo Motors**

These motors consume electrical energy to generate torque and control the angular position of the robotic arm's joints. Since not all the servo motor can be utilized at the same time, it is assumed that only one is used at a time; therefore:

$$\begin{aligned} Energy\ consumed &= voltage \times current \\ &= 6 \times 500 \times 10^{-3} \\ &= 3W \end{aligned}$$

- **For Arduino Uno**

In the Arduino Uno, energy is consumed during data processing, execution of control algorithms, and communication with peripherals. Energy consumed is also calculated by:

$$\begin{aligned} Energy\ consumed &= voltage \times current \\ &= 1.8 \times 0.3 \times 10^{-3} \\ &= 0.00054W \end{aligned}$$

- **For IR Proximity Sensors**

While the IR proximity sensors are low-power devices, they still consume energy during operation. Continuous monitoring of the environment for object detection contributes to overall energy consumption. In this case the three sensors are operating at the same time.

$$\begin{aligned} Energy\ consumed &= voltage \times current \\ &= 3.3 \times 5.3 \times 10^{-3} \\ &= 0.0175W \end{aligned}$$

Since there are three IR sensors, the energy consumed becomes

$$= 0.0175 \times 8$$

$$= 0.14W$$

3.2 - Microcontroller (Arduino Uno)

This block consists of the Arduino-Uno module. The Arduino-Uno microcontroller board is based on the ATmega328p microcontroller IC. Its specifications have been discussed in chapter two of this work. In this device, the Arduino UNO serves as the microcontroller which is the brain of the device responsible for receiving data from required units and sending out information through another unit.

All other units were connected to the Arduino UNO board through the Arduino motor shield by interfacing. These components were connected to the Arduino UNO board by soldering and the aid of jumper cables and glues where necessary. This process of interfacing different components is shown in their respective figures.

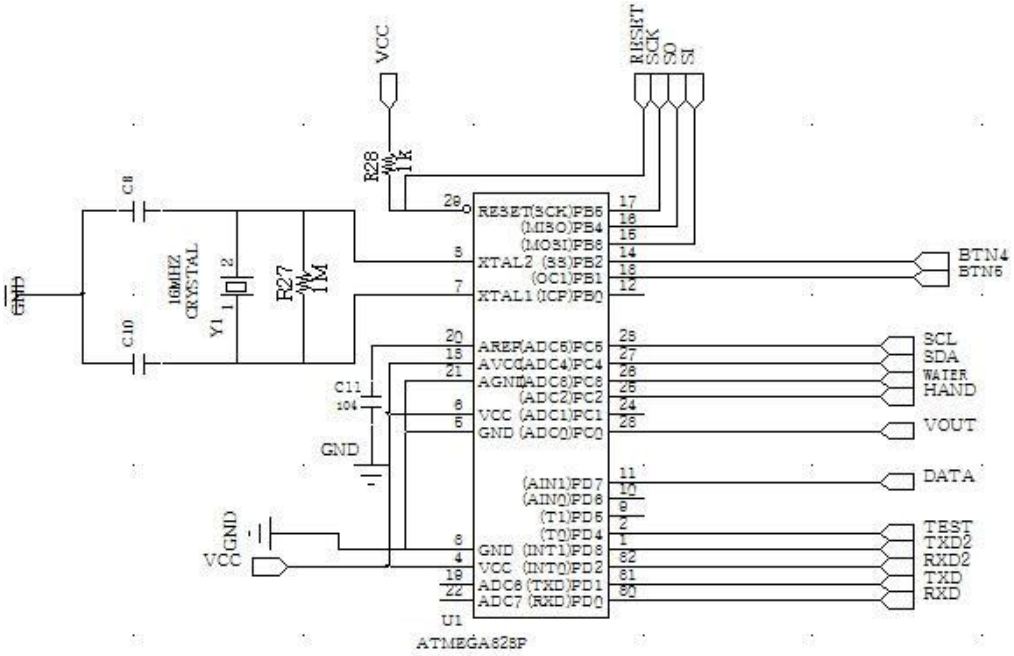


Figure - 3.3 ATmega328p Circuit Diagram (https://www.researchgate.net/figure/ATMEGA328P-Circuit-Diagram_fig2_334329108)

The Arduino-Uno being a programmable controller module is programmed in C Language and this code is compiled into machine language using the Arduino IDE.

The Arduino Uno is connected to the robotic arm's control interface. This typically involves using digital pins on the Arduino to send signals to the robotic arm's motors or actuators.

The program is written in the Arduino IDE (Integrated Development Environment) using C/C++ to control the robotic arm's movements. This program will include commands for the arm to pick up objects, move them, and release them at specific locations.

IR sensors are integrated with the Arduino to provide feedback on the arm's position, object detection, size etc.

Implement control logic in the Arduino program to coordinate the robotic arm's actions. This involve using conditional statements, loops, and functions to ensure precise and accurate movements.

After the program is written, the robotic arm is tested with the Arduino Uno to ensure that it performs pick and drop operations correctly. Calibrations are made to meet the arm’s required movement and adjust the program as needed.

Radio frequency module, the Radio frequency Hardware is connected to Arduino uno through the Arduino motor shield, this helps in the manual control of robotic arm by communicating the RF transmitter with the RF receiver.

Overall, the Arduino Uno serves as the brain of the operation, sending commands to the robotic arm based on programmed instructions and sensor feedback, enabling it to perform pick and drop operations effectively. Below is the flowchart showing how the system function both automatically and manually.

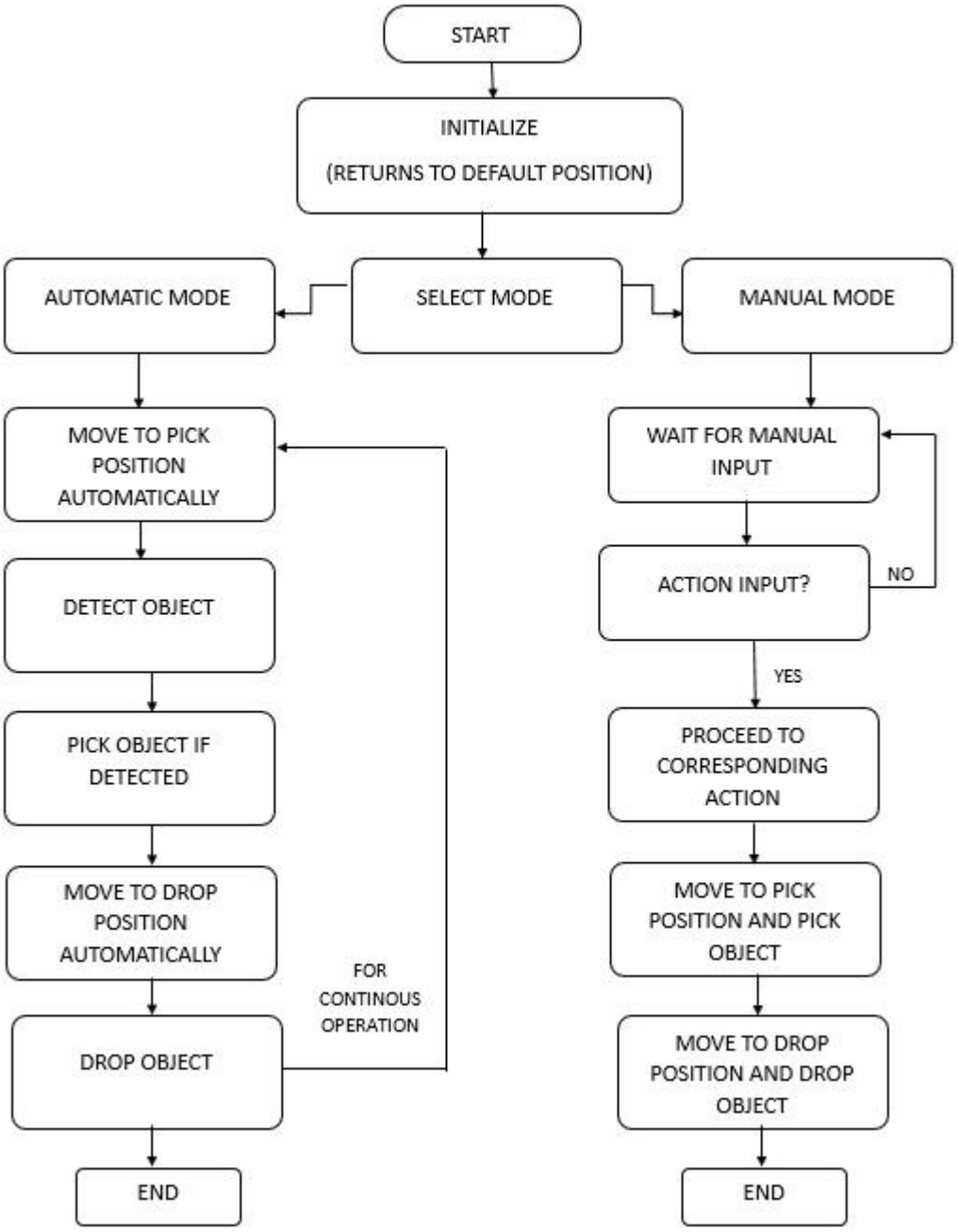


Figure 3.4 - Flow diagram of the system

Instruction Cycle of An Arduino Uno

The instruction cycle of an Arduino Uno, which uses the ATmega328P microcontroller, can be calculated using its clock frequency and the number of clock cycles required for each instruction. The Arduino Uno typically operates at a clock frequency of 16 MHz

The instruction cycle time (T) can be calculated using the formula:

$$T = \frac{1}{f}$$

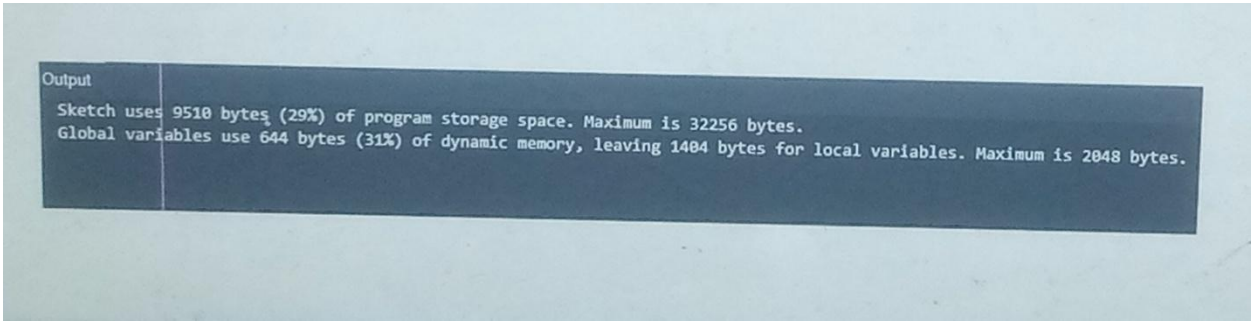
where f, is the clock frequency.

For the ATmega328P microcontroller, most instructions require one clock cycle to execute, except for branch instructions, which require two cycles if the branch is not taken and three cycles if it is taken.

So, if we assume a clock frequency of 16 MHz:

$$T = \frac{1}{16 \times 10^6} = 62.5\ nanoseconds$$

This means each instruction cycle takes approximately 62.5 nanoseconds to complete on the Arduino Uno.



Arduino instruction Efficiency;

$$\frac{Maximum\ bytes - used\ byte}{Maximum\ bytes} \times 100\%$$

$$\frac{32256 - 9510}{32256} \times 100\% = 71\%$$

This means the instruction efficiency of the Arduino uno program is 71%.

3.3 - Servo Motor

Numerous motors are presently being manufactured, each with its own set of advantages and ideal applications. When it comes to robotic arms, the choice between a servo motor and a stepper motor is often a crucial decision. Servo motors are favoured for their precise control and feedback mechanisms, making them ideal for tasks requiring accurate positioning and speed control. In comparison, stepper motors excel in applications where movement needs to occur in discrete steps, offering simplicity and cost-effectiveness but with slightly less precision than servo motors.

Design considerations and specifications:

The design considerations are hinged on the following factors;

Weight: the weight has to be lightweight and also robust enough to handle the weight of the robotic arm and the objects it will lift, the maximum weight to be lifted is 20kg. Choosing the right motor will ensures optimal performance and efficiency. A Servo motor weighing 60g was chosen.

Rotating angle: for the sake of the design, the servo chosen can rotate approximately 120 degrees (60 in each direction).

Operating Voltage (4.8 V to 7.2 V): This specification indicates the range of voltages over which the servo motor can operate. The motor is powered within this range to ensure optimal performance and avoid damage.

Operating speed (0.17 s/60º at 4.8 V, 0.13 s/60º at 6 V): Operating speed indicates how quickly the servo motor can rotate to a specific angle. It is measured in seconds per 60 degrees of rotation and is essential for determining the speed and responsiveness of the robotic arm's movements.

Running Current (500 mA): Running current is the amount of current the servo motor draws during normal operation. This specification helps to determine the power requirements of the motor and select an appropriate power supply

Stall Current (2.5 A at 6 V): Stall current is the maximum current drawn by the motor when the shaft is prevented from rotating (stalled). It's important to ensure that the power supply can deliver this current without voltage drop or damage.

Stall Torque: This refers to the maximum torque output of the servo motor when the shaft is prevented from rotating. It is an important parameter for determining the motor's ability to lift heavy loads or overcome resistance in your robotic arm's movements. The stall torque specifications: 9.4 kgf·cm at 4.8 V, 12 kgf·cm at 6 V

Standby Current (5mA at 6.8V): This specification indicates the range of temperatures within which the servo motor can safely operate. It's important to operate the motor within this range to prevent overheating or performance issues.

The electrical input power can then be given as;

$$P(input) = V(supply) \times I(standby)$$

$$P = 6.8V \times 5mA$$

$$P = 0.034W$$

The mechanical output power can be given as;

$$P(output) = Angular\ speed \times stall\ torque$$

$$P = 0.14 \times 12$$

$$P = 1.68W$$

The Servo motors, which are six in number are placed in strategic position for the easy maneuver of the robotic arm, providing 6 degrees of freedom. Below is a figure showing the positions of each servo motor on the robotic arm.

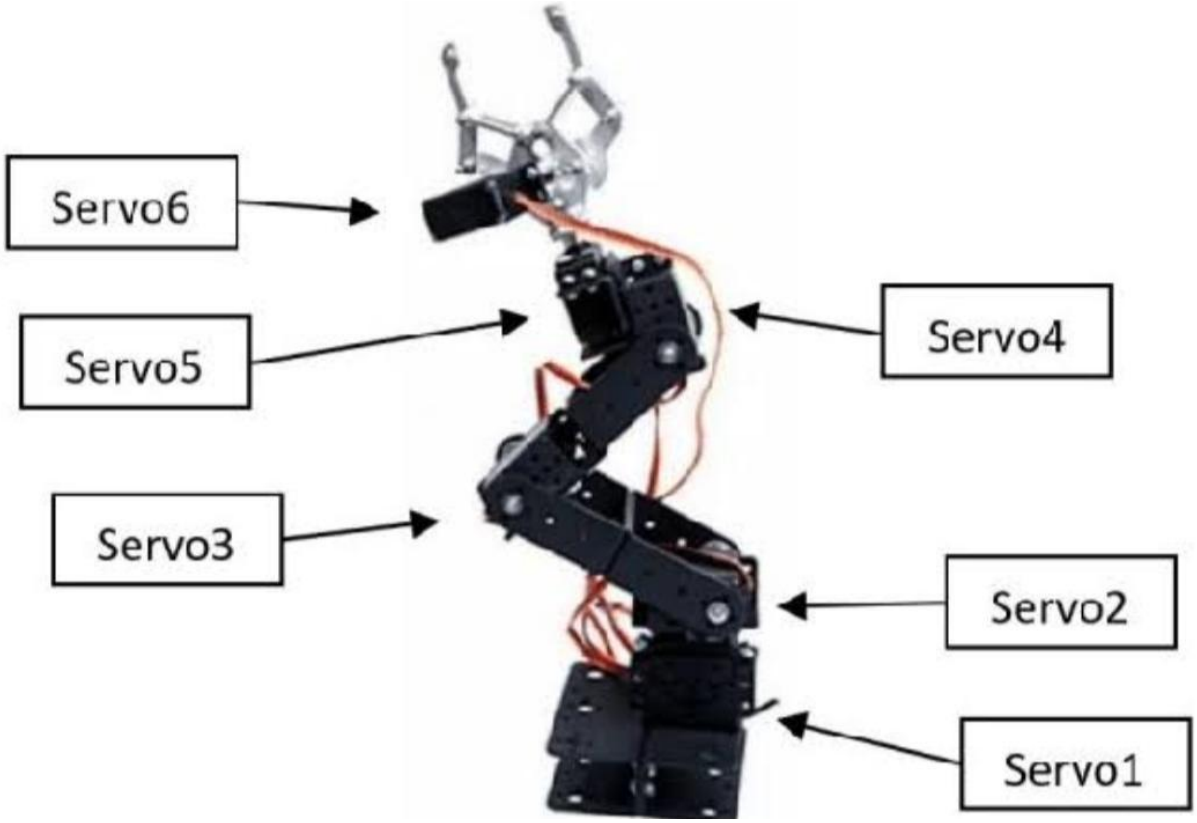


Figure 3.5 – Image showing the positions of the servo motor (https://www.researchgate.net/figure/Installation-of-motor-servo-on-arm-frame_fig2_342697611)

3.4 – IR Proximity Sensor

For an infrared sensor, its two main components are the emitter and the detector. An infrared photodiode serves as the detector, and an infrared LED serves as the emitter. The emitter emits infrared light at a wavelength that the photodiode can detect. As the photodiode is exposed to infrared light, its resistance varies, causing the output voltage to vary proportionately.

Infrared sensors come in two varieties: passive and active. We are using an active infrared sensor, which is made up of an emitter and a detector. The LED is the emitter in this arrangement, and the photodiode is the detector. The passive infrared sensor, on the other hand, measures the radiant heat in its environment to identify the presence of objects; it lacks an emitter or detector.

The emitter emits infrared light to power the IR sensor. The detector receives this light after it reflects off of objects. The sensor can determine whether or not there are things nearby by detecting the reflected infrared light. The Circuit diagram for the IR proximity sensor is shown below.

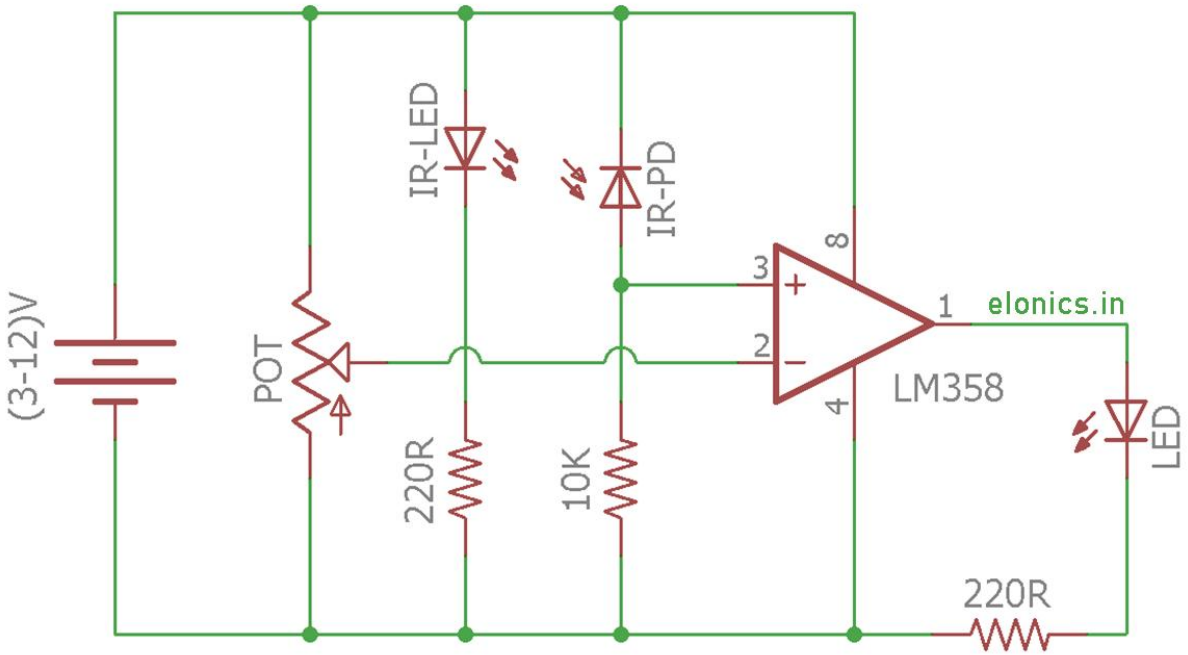


Figure 3.6 - Circuit diagram for IR proximity sensor (https://www.instructables.com/How-to-Make-a-Proximity-Sensor/)

Analysis

Let us discuss infrared radiation. The rate at which radiation is released, the impact on the object's surface, and your return should all be proportionate to the distance covered. The equation below describes the relationship between time, distance, and speed:

$$speed = \frac{distance}{time}$$

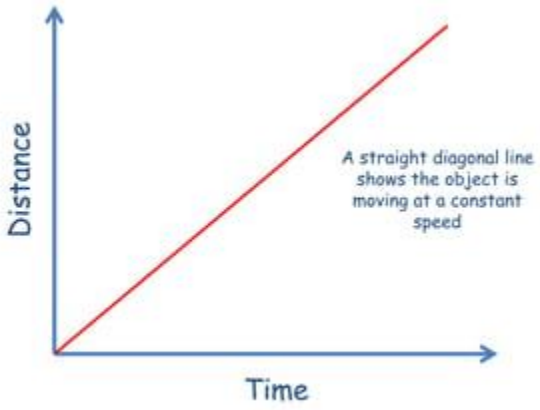


Figure 3.7 - Distance/time graph diagram. (https://shorturl.at/inDS7)

The speed of light is defined as $3\ x\ 10^8$m/s. This speed implies that the relationship between time and distance must match the speed of light.

Size Evaluation based on the steps:

We use three infrared sensors, one for each servo motor. The length, width, and height of the object placed on the board are represented by the three directions these infrared sensors go across on the board: X, Y, and Z. Each infrared sensor moves in steps as a result of the servo motors controlling its movement.

The IR sensors begin at the default location, or origin, on the X-axis, for example, when there is an object on the board. Until they can no longer detect the object, they proceed in small stages. The steps left on the board are then counted at that point. The relative length of the object is given by this count.

For instance, let's say there are a total of five steps on the X-axis. If the object only occupies step two of the five steps on the X-axis, the IR sensor will detect its disappearance after it travels past step two. Consequently, the object's length is determined to be two steps long. The width and height can be ascertained using the same procedure, which also yields the object's minimum measurable size.

Volume = length x width x height.

Thus, the volume will be determined as follows, for instance, if the sensors measure an object's length, width, and height as five steps, four steps, and one step, respectively.

Volume = 5 x 4 x 1 = 20 step cubed.

We have large, medium, and small sizes for sorting. Thus, for every size, there is a volume threshold value.

Size classification:

We will use the formula;

$$Average\ volume\ = \frac{Length \times Width \times Height}{5}$$

to determine the overall volume or average volume for each object in order to maintain uniformity. This figure will serve as our basis for comparing objects, as one object may have more length or height than others. Thus, it would not be appropriate to compare simply using the volume. Therefore, we use the average volume as a benchmark, guaranteeing that the object's size in relation to others is taken into account equally.

By dividing the volume by 5, which represents the total of 5 steps, it is possible to clearly classify the size of objects as big, medium, or small using three IR sensors, each of which represents a direction in the x, y or z axis.

3.5 - Radio Frequency Module (RF Module)

An RF (Radio Frequency) module is an electronic device used to transmit or receive radio signals wirelessly. It converts electrical signals into radio waves and incoming radio signals into electrical signals depending on it's built.

It can be used for various purposes such as

- Remote control

- Wireless data transmission

- Telemetry etc.

In this Project, the Sony Game Controller (PS2 controller) acts as our RF MODULE, functioning as a remote control for the robotics arm.

How To Connect The Sony Game Controller With Arduino

The Sony Game Controller has a receiver which is connected to the Arduino. The image of the receiver is shown in figure 3.7 below.

This receiver has 9 pins with which it is connected to the Arduino:

1. Data: master line for sending data to slave (MOSI)

2. Command: slave line for sending data to master (MISO)

3. Vibration: vibration motors supply; 7.2 volts to 9 volts

4. Ground: circuits ground

5. VCC: circuits supply; 3.3 volts

6. Attention: CS or Chip Select pin for calling slave and preparing the connection

7. Clock: equivalent to SCK pin for clock

8. No Connection: useless

9. Acknowledge: acknowledge signal from the controller to PS2 receiver

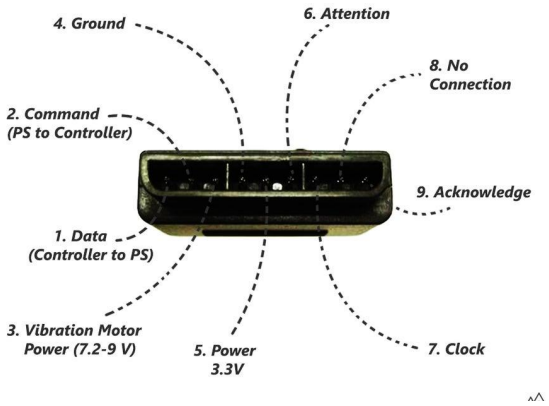


Figure 3.8 - Image of the control receiver (https://forum.arduino.cc/t/using-dualshock-4-to-control-arduino-uno/984289)

Interfacing Ps2 Controller and Arduino:

To use the PS2 controller, we had to introduce the controller's key to Arduino and choose proper functions for each key based on our project via coding. The coding is done aided by the PS2X library.

The most practical functions in this library are:

ps2x.config_gamepad (clock, command, attention, data, Pressures? Rumble?);

function sets the controller pin and sensitivity to pressure and vibration of the motors. If you want keys insensitive to pressure, or motors don't have vibration, set Pressures, and Rumble as false. This function returns value for error.

ready ();

This function determines the detected controller type. 0 means the controller is not detected correctly, 1 means DualShock controller detection, and 2 means Guitar Hero controller detection.

read gamepad (Boolean motor1, byte motor2);

This function start reading the status of the keys when the state of the vibration of the motor is determined. (Motor 2 is the bigger one.)

Button (but type);

This function returns 1 when the specific key in function argument is pressed. In DualShock controller keys are named as the attached table.

Analog (but type);

This function returns analog keys value.

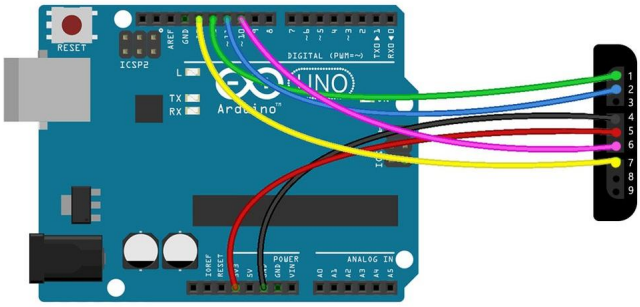


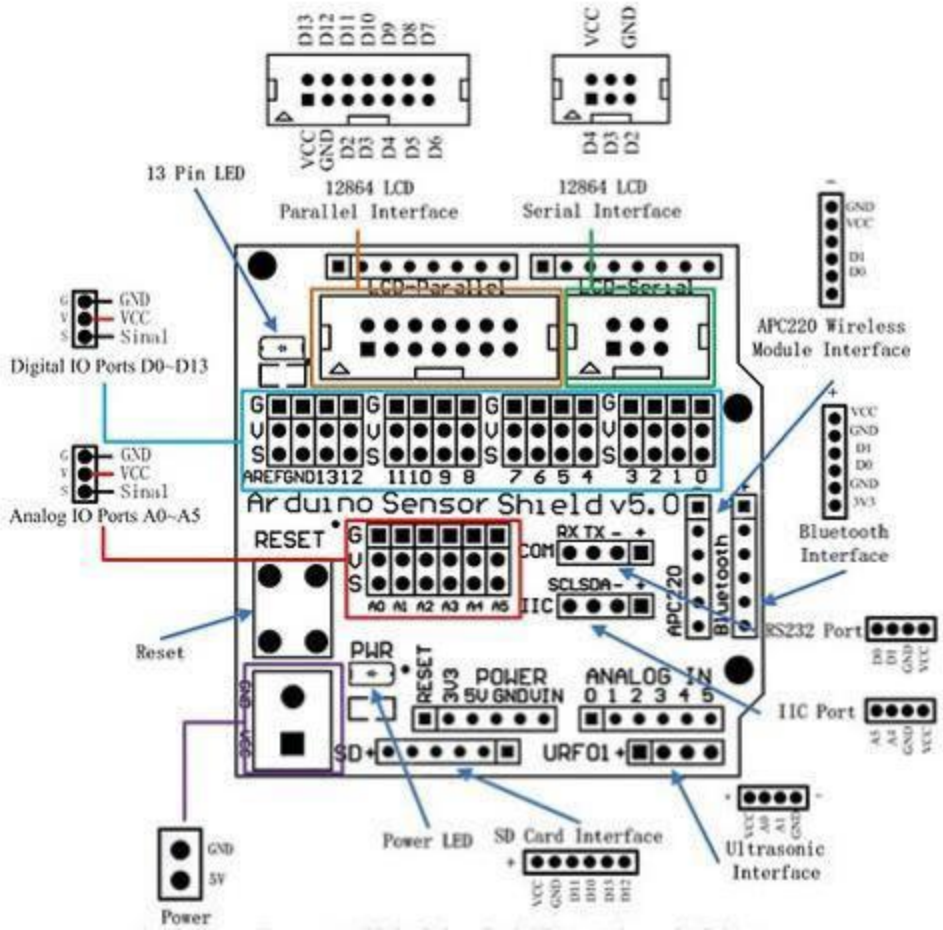
Figure 3.9 - Circuit Diagram of The PS2 Controller (https://electropeak.com/learn/tutorial-interface-wireless-playstation-ps2-controller-arduino/)

3.6 - Arduino Motor Shield

The Arduino motor shield is physically connected to the Arduino Uno board via header pins. This connection allows the shield to receive commands and data from the Arduino Uno and send control signals to the connected components. The Arduino Motor Shield allows for easy control of the motor direction and speed using an Arduino. By enabling the user to directly address Arduino pins, it also grants the ability to power a motor using a separate power supply of up to 12 volts.

Program is written in the Arduino IDE that defines the logic and behavior of the robotic arm for pick and drop operations. This program includes instructions for motor control, sensor readings, decision-making algorithms, and communication protocols.

Within the program, the appropriate Arduino libraries and functions is used to send control signals to the motor shield. For example, The user can use functions like `analogWrite()` for PWM control of motor speed, `digitalWrite()` for direction control, and other custom functions



or libraries for advanced motor control features.

Figure 3.10 – Functional diagram of Arduino motor shield (https://www.researchgate.net/figure/Arduino-sensor-shield-v5-functional-diagram_fig4_319960262)

Apart from motor control and sensor integration, the Arduino Uno, with the motor shield, can also interface with the Radio frequency module to control the direction of the servo motors. The Arduino motor shield plays a crucial role in controlling the motors of a robotic arm for pick and drop operations. The Arduino motor shield communicates with the components of the robotic arm, such as motors and sensors, primarily through the Arduino Uno board to which it is connected.

3.7 – Complete Circuit Diagram

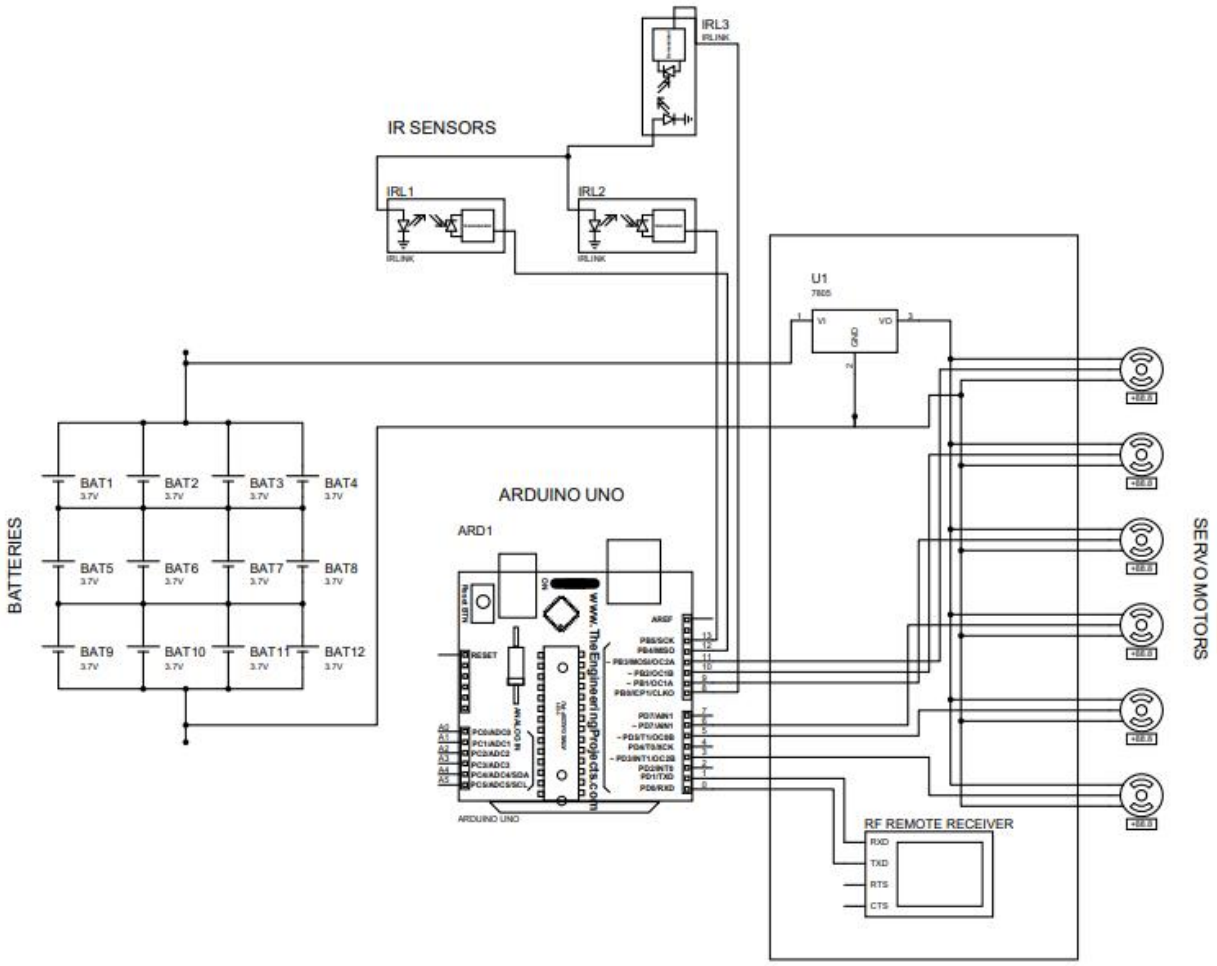


Figure 3.11 – Circuit schematic for the hybrid pick and drop robotic arm

3.8 Working Principle

In this project, we are using 6 servo motors each of which can be controlled by PWM signals from the microcontroller.

Here, we have chosen two reference positions. First reference position is the place from where the arm has to pick the object (pickup position) and second reference position is the place where the robot has to place the object (drop-off position). After the supply of power, the IR proximity

sensor begins to operate, checking to see if there are any object within its range, if an object is detected, it relays it to the microcontroller.

First, the microcontroller signals the motor-1 at the base to make the rotation of the arm to the desired direction, moving the robotic arm from its initial rest position. Then the signal from microcontroller is given to the 2nd motor at the shoulder so that it can make up and down movement. Next motor-3 which is situated at the elbow is activated to also perform an up and down movement. Motor 4 and 5 which are situated at the wrist with are also activated respectively, with motor-4 to perform the up and down movement and motor-5 to perform a rotational movement. All of these movements are done in order to bring the gripper to the precise position of the object. Then, motor-6 is activated so that the gripper holds the object. After grasping the object, the microcontroller calculates a new trajectory from the pickup position to the drop-off position. The microcontroller then sends signals to the servo motors to move the arm along this trajectory, upon reaching the drop-off position, the microcontroller sends a signal to deactivate the servo motor at the gripper, allowing it to release the object gently. After this, the arm may then return to its initial rest position or assume another desired task.

Chapter Four

Construction, Testing and Results

This chapter provides an overview of the various aspects of our project, including the construction process, the extensive testing procedures, the analysis of the results obtained, the adherence to safety protocols, and a variety of tools used during the project's lifetime.

4.1 – Construction

The construction stage is the part where the actual structures or parts are constructed or assembled in accordance with the plans and specifications that were established during the planning and design stages.

4.1.1 – Dimensions

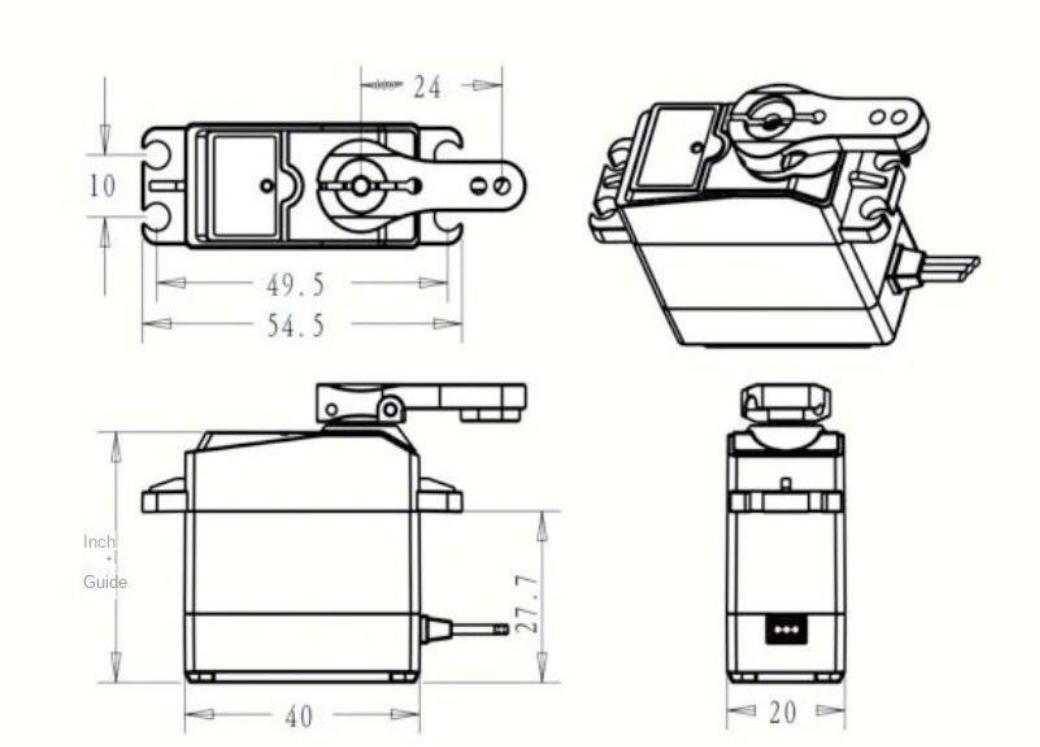


Figure 4.1 – Dimension of the servo motor

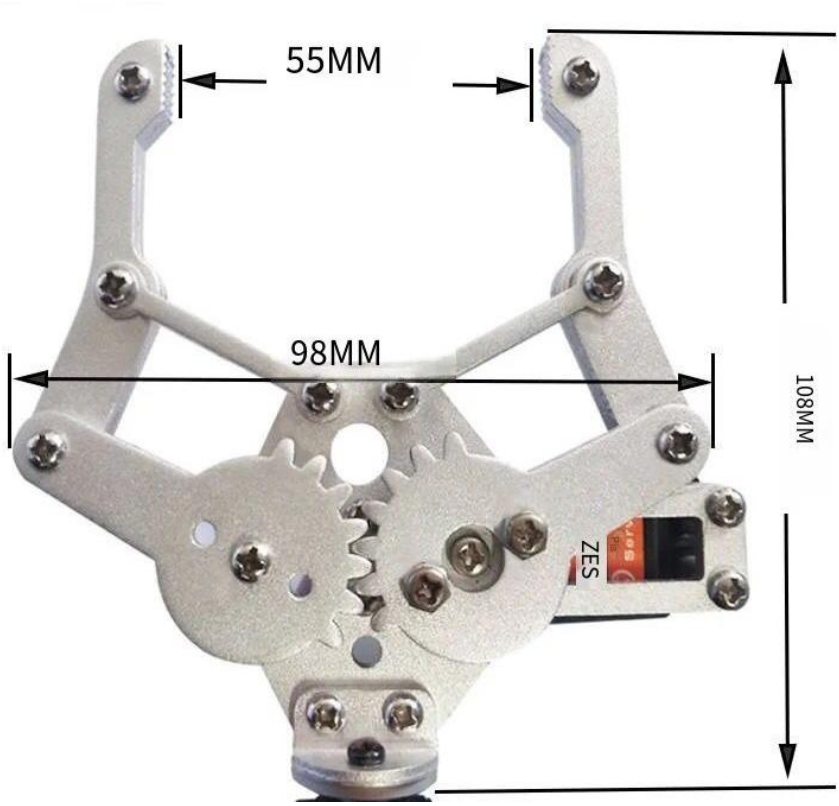


Figure 4.2 – Dimension of the gripper

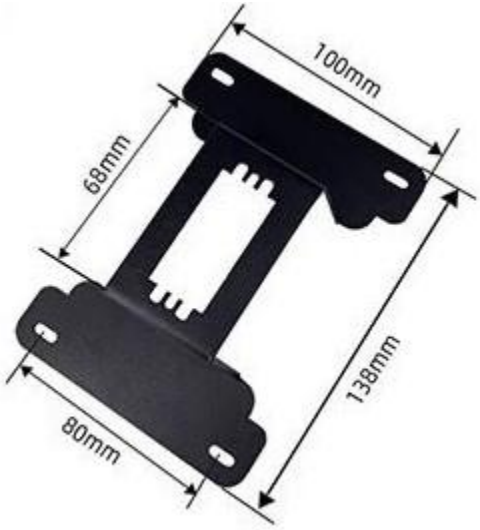


Figure 4.3 – Dimension of the large U-shaped base

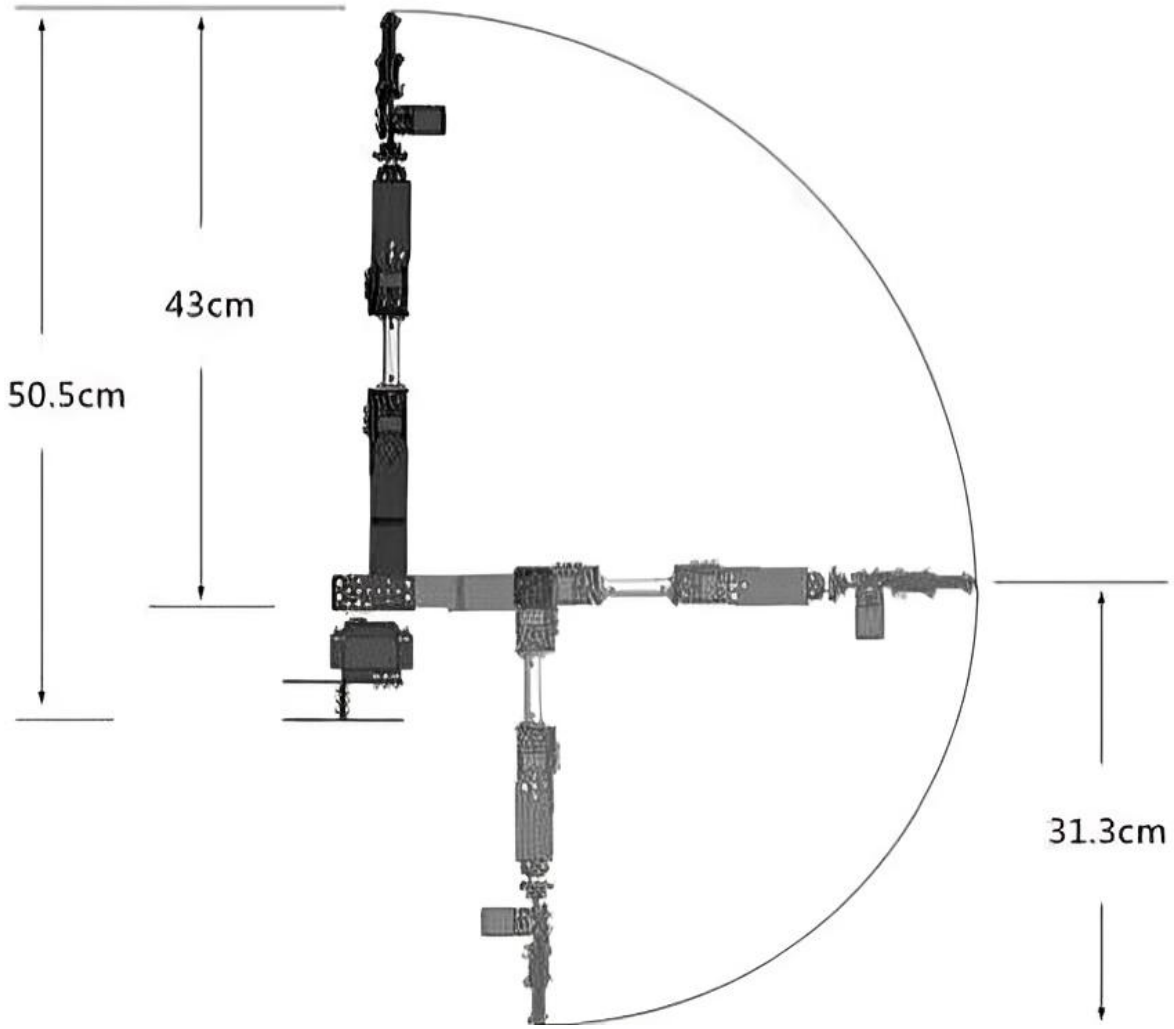


Figure 4.4 – Dimension of metal frame of the robotic arm

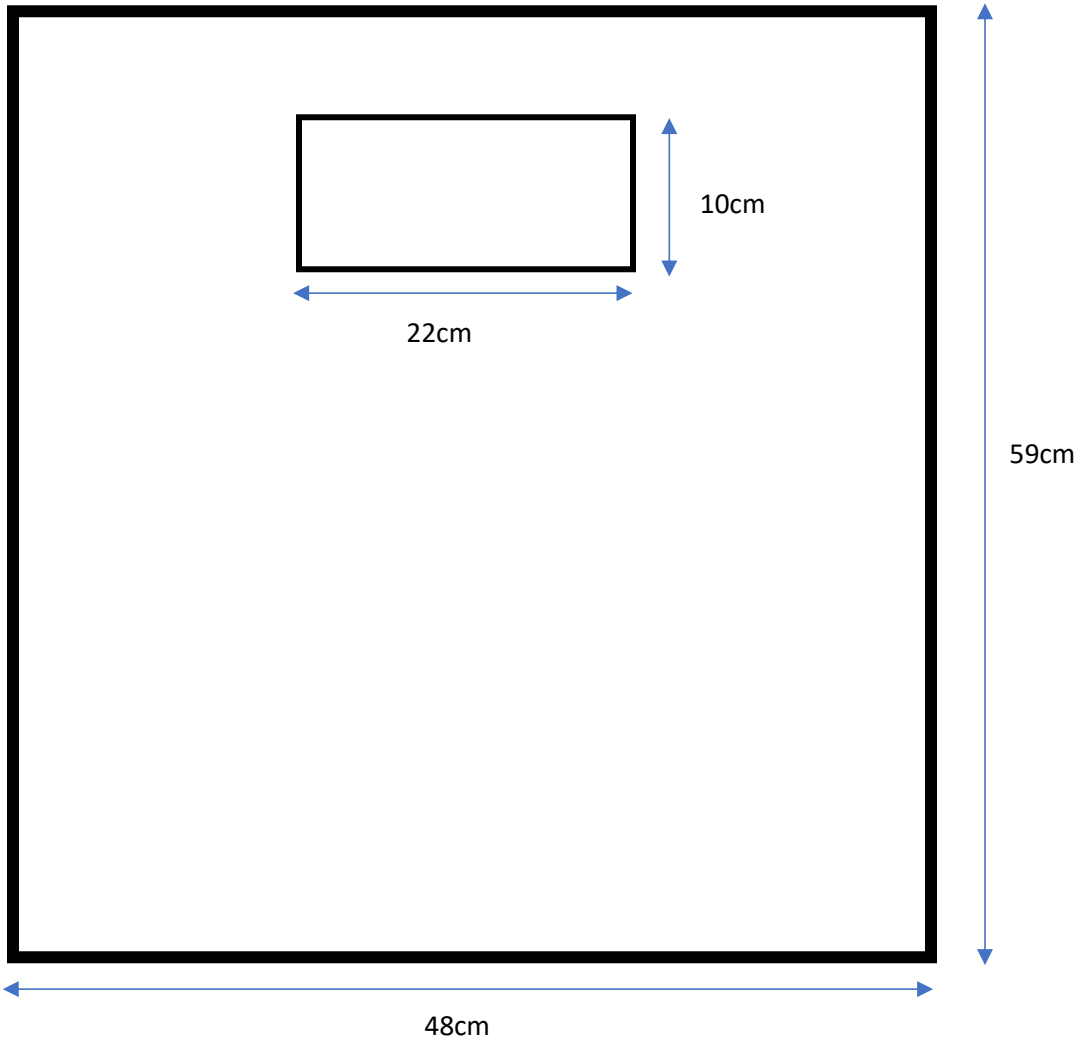


Figure 4.5 – Dimension of the plywood base board

4.1.2 – Procedure for Carrying Out the Construction

Acquisition of Materials: Essential components were procured from various sources, including online stores and local electronics retailers.

Coupling of Robotic Arm:

One servo motor (motor-1) is attached to the U-shaped base by means of screws, as shown in figure 4.6. This provides the rotation of the arm at the base allowing the arm to pivot horizontally.

Another servo motor (motor-2) is then attached to an inverted U-shaped frame which is joined to an upright U-shaped frame. This connection is then attached to the servo motor on the base, as shown in figure 4.7. This joint provides an up and down movement. The arm extension as shown in figure 4.8 with two other frames attached at both ends is then introduced. One end is attached to the upright U-shaped frame via a servo motor (motor-3) as shown in figure 4.9. This joint also provides another up and down movement which extends the reach of the arm. Motor-4 is then attached to the other end of the arm extension and connected to the gripper which has two additional servo motors (motor-5 and motor-6) connected to it as shown in figure 4.10. Motor-5 at its end to provide a rotational movement and motor-6 at its side to open and close the gripper. Figure 4.11 shows the complete coupling of the robotic arm.

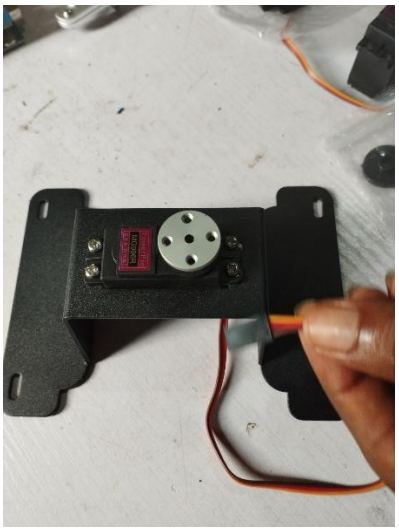


Figure 4.6

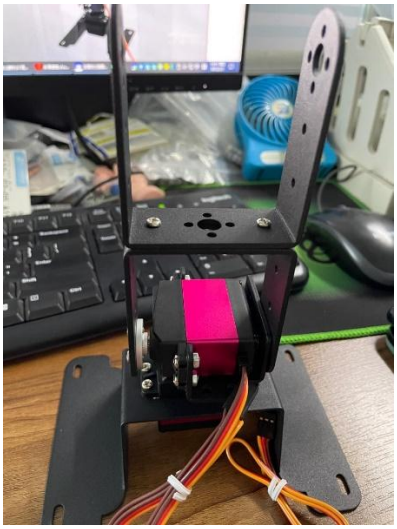


Figure 4.7

Figure 4.8

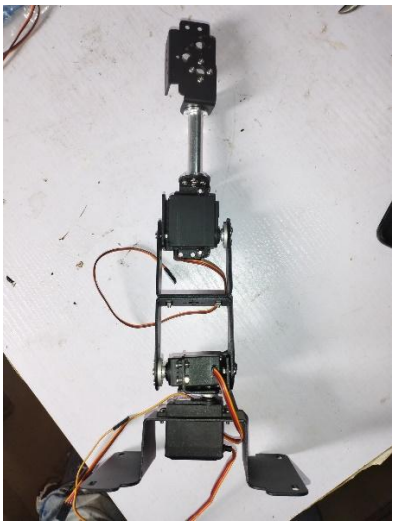


Figure 4.9

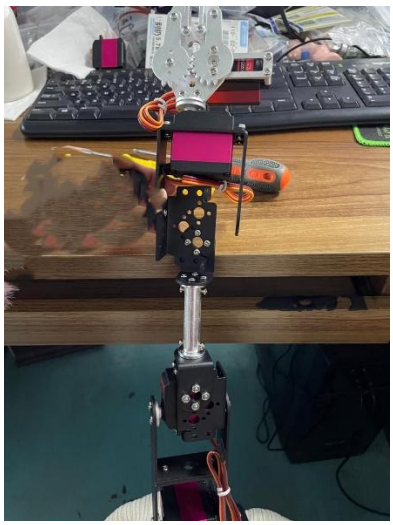


Figure 4.10

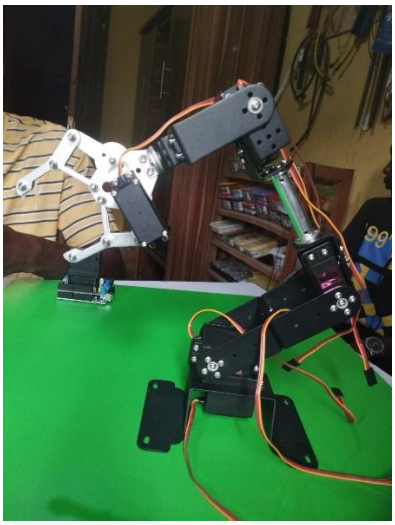


Figure 4.11

Battery Holder Soldering: Following successful coupling of the robotic arm, six sets of battery holder were soldered together. Each to hold a pair of battery to serve as supply. Below is an image showing the soldering pattern.

Figure 4.12 – Image of Battery holder soldering

Cutting of Wooden Board: The wooden board was cut and shaped to the required dimension by the use of a jigsaw machine as shown below.

Figure 4.13 – Image of wooden board being cut to dimension

Drilling of Wooden Board: After shaping of the board, holes were drilled through to allow for easy connection of the wires from the batteries under the board to the robotic arm on top.

Figure 4.14 – Image of hole being drilled on the wooden board

Code Loading: The C program for the project was developed using Arduino

IDE and loaded into the ATmega328p microcontroller.

4.1.3 - Tools Used

The construction and assembly process involved the use of various tools and equipment, including:

- Soldering Iron
- Lead
- Screwdriver Set
- Drill Machine
- Jigsaw
- Wire Cutter
- Multimeter
- Desoldering Pump
- Plier
- Allen key

4.2 – Testing

A thorough methodology was used during the project's testing phase to guarantee the system's dependability, functionality, and safety. A number of tests were carried out, each concentrating on different aspects of the project's components. A detailed description of every test may be found below:

4.2.1 – Range Test

The range test involves evaluating the performance of the controller's communication system over varying distances and under different environmental conditions. Below are the tests conducted:

- **Line-of-Sight Performance Test**
  From the test carried out it was confirmed that the range of the controller's signal before it disconnects is about 100ft
- **Non-Line-of-Sight Performance Test**
  After the test it was noticed that obstacles weren't a problem to the RF signal, as the arm could still be controlled even with different obstruction between the RF controller and receiver.

4.2.2 – Sorting Accuracy Test

Sorting accuracy refers to the degree of correctness or precision with which the robotic arm arranges elements in a specified order. Accuracy is typically measured by comparing the actual output to the expected or desired sorted order of the elements as shown in table 4.1 below.

For small objects:

Dimension are: Length = 2cm

Breadth = 2cm

Height = 2cm

For medium objects:

Dimension are: Length = 4cm

Breadth = 4cm

Height = 4cm

For large objects:

Dimension are: Length = 6cm

Breadth = 6cm

Height = 6cm

Size Of Objects	Expected Outcome	Actual Outcome
Small	Small	
Medium	Medium	
Large	Large	

Table 4.1 – Outcome of accuracy test

4.2.4 – Sorting Speed Test

The sorting speed test is a critical evaluation method used to assess the efficiency and performance of the robotic arm. In this test, the time taken by the arm to arrange elements in a specified order is measured and analyzed under various conditions.

The robotic arm took about to pick and drop an object in the required position.

4.3 – Result Analysis

The result analysis of this project aims to evaluate the performance, functionality, and effectiveness of the robotic arm system. This analysis presents findings obtained from various tests and evaluations conducted to assess mechanical performance, control system effectiveness, task performance, safety features, environmental adaptability, system integration, and user satisfaction.

Mechanical Performance: The robotic arm demonstrated a wide range of motion, allowing for flexible positioning and maneuvering. Additionally, the payload capacity of the arm proved sufficient for lifting and carrying objects of various sizes and weights. However, slight deviations in precision were observed during accuracy testing, suggesting the need for fine-tuning in certain areas.

Control System Evaluation: The control system of the robotic arm exhibited commendable responsiveness and coordination. Movements were executed smoothly. The control interface provided intuitive control over the arm's actions, allowing for precise manipulation and task execution.

Task Performance: Tests assessing the task performance of the robotic arm showcased its capability to accurately and efficiently complete pick and drop operations. The arm consistently achieved the desired positioning and orientation when picking up and releasing objects, demonstrating reliability in task execution. Speed and efficiency were notable, with the arm swiftly completing tasks within acceptable timeframes.

4.4 - Bill of Materials

A comprehensive list of components utilized in the project, along with quantities and

estimated prices, is provided in the table below.

Table 4.2 - Bill of Materials (BOM)

S/N	Component/Item	Amount	Price (₦)
1	Metallic robotic arm frame and Servo Motors	1	80,000
2	Atmega382p microcontroller	2	25,000
3	Handheld control pad with transmitter and receiver	1	22,000
4	AVR programmer	1	7,000

5	Crystal Oscillators	2	500
6	Microcontroller Holder	1	850
7	Power Switch	1	2,000
8	DC-DC buck converter	2	3,000
9	IR Led	5	1,000
10	TSOP1338 IR receiver	5	2,500
11	PVC Pipes	4	2,000
12	Plywood base board	1	10,000
13	Black Tape	1	400
14	3s lithium-ion Battery Charger	1	23,000
15	IR proximity sensor	8	12,000
16	Masking Tape	1	500
17	34Wh Lithium-ion Battery	1	15,000
18	Battery Holder	18	7,200
19	Cables, wires and jumpers	100, 50	3,500
20	Miscellaneous	-	60,000
21	Total Cost of Materials	-	277,250
22	Shipping fee + Transport	-	24,000
	TOTAL	-	**301,450**

Chapter Five

Conclusion and Recommendations

5.1 – Conclusion

In conclusion, the development and implementation of the Hybrid Pick and Drop Robotic Arm have been a significant endeavor with several noteworthy outcomes. The project aimed to address the design and implementation of a robotic arm capable of picking and dropping object.

After careful planning, extensive testing, and iterative improvement, the following significant discoveries and accomplishments have emerged:

Achievement of Objectives: The project successfully met its primary objectives of designing and constructing a functional robotic arm capable of hybrid operation. Through careful consideration of design principles and integration of diverse technologies, the robotic arm demonstrates proficiency in both picking and dropping tasks.

Performance and Efficiency: According to test results, the Hybrid Pick and Drop Robotic Arm operates with remarkable resilience, precision, and flexibility across a range of situations. The arm's hybrid design maximizes energy efficiency and improves total task execution efficiency.

Impact and Significance: The Hybrid Pick and Drop Robotic Arm has the potential to solve practical problems and progress the area of robotics, even beyond its current state of development. Its importance in modern automation is highlighted by its capacity to increase productivity, decrease manual labor, and streamline procedures.

Looking ahead, further research and development efforts are warranted to explore additional functionalities, optimize performance, and address emerging challenges.

5.2 – Recommendations

- Integration of Internet of Things (IoT) into the robotic arm system could be done in order to have real time monitoring and control over the arm, allowing operators to remotely monitor and control the robotic arm's performance from anywhere with an internet connection.
- Integration of AI and Machine Learning into the robotic arm's control system could be done to enable adaptive and autonomous operation, unlocking new levels of efficiency and autonomy.

REFERENCES

1. Devol, G. Jr. (1954). "Programmed Article Transfer." Transactions of the Society for Mechanical Engineers, Volume: 76, ISSN: 0096-6981.

2. Alkandari, A., et al. (2023). "Smart Automated Robot Changing Tires using Ultrasonic Sensors." International Journal of Robotics and Automation, ISSN: 1925-7090

3. Olawale, et al. (2007). "Microcontroller-based robotic arm for picking and releasing small objects" IEEE Transactions on Robotics, Volume: 23, Issue: 4, ISSN: 1558-156X.

4. Ali, H. M. (2022) "Design and implementation of Arduino-based robotic arm" -Journal of Mechanical Engineering Research, Volume: 14, Issue: 3, ISSN: 2141-2383.

5. Elfasakhany, et al. (2011). "Low-cost design for a four Degree of freedom (DOF) robotic arm" Robotics and Autonomous Systems, Volume: 59, Issue: 12, ISSN: 0921-8890.

6. Aggarwal, L., Gaur, V., & Verma, P. (2013). "Wireless Gesture Controlled Robotic Arm with Vision" - Journal of Control, Automation and Electrical Systems, Volume: 24, Issue: 4, ISSN: 2195-3880

7. Dabhade, H., et al. (2022). "Automated system for detecting the color of objects and placing them in specific locations using a robotic arm" International Journal of Advanced Manufacturing Technology, Volume: 119, Issue: 1-4, ISSN: 1433-3015

8. Krishna, R., et al. (2012)." Robotic Arm Based on Haptic Technology". IEEE Robotics & Automation Magazine, Volume: 19, Issue: 4, ISSN: 1070-9932.

9. Giannoccaro, et al. (2013). "Robotic arm for sorting different objects using RFID tag" Journal of Intelligent Manufacturing, Volume: 24, Issue: 4, ISSN: 0956-5515.

10. Ali, Z., et al. (2023). "Low-cost 5-DOF robotic arm for material handling and sorting" Automation in Construction, Volume: 120, ISSN: 0926-5805.,

11. researchgate.net, ATMEGA328P CIRCUIT, Content available online at (https://www.researchgate.net/figure/ATMEGA328P-Circuit-Diagram_fig2_334329108)

12. rajguruelectronics.com (2024), Servo Motor - MG996R servo motor (180 DEGREE). Content available online at https://www.rajguruelectronics.com/ProductView?tokDatRef=MTA1MA==&tokenId=NjY=&product=MG996%20SERVO%20MOTOR%20(180%20DEGREE)

13. instructables.com (2024), How to Interface PS2 Wireless Controller With Arduino. Content available online at https://www.instructables.com/How-to-Interface-PS2-Wireless-Controller-W-Arduino/

APPENDIX

CODE FOR ARDUINO

```
#include <Wire.h>
#include "PS2X_lib.h"
#include "Adafruit_MS_PWMServoDriver.h"
#include "QGPMaker_MotorShield.h"

long ARM_MIN[]={10, 10, 40, 10, 0, 0, 0, 0};

long ARM_MAX[]={170, 140, 170, 102, 180, 180, 180, 180};

QGPMaker_MotorShield AFMS = QGPMaker_MotorShield();
PS2X ps2x;
QGPMaker_Servo *Servo0 = AFMS.getServo(0);
QGPMaker_Servo *Servo1 = AFMS.getServo(1);
QGPMaker_Servo *Servo2 = AFMS.getServo(2);
QGPMaker_Servo *Servo3 = AFMS.getServo(3);
QGPMaker_Servo *Servo4 = AFMS.getServo(4);
QGPMaker_Servo *Servo5 = AFMS.getServo(5);
QGPMaker_Servo *Servo6 = AFMS.getServo(6);
QGPMaker_Servo *Servo7 = AFMS.getServo(7);
QGPMaker_DCMotor *DCMotor_2 = AFMS.getMotor(2);
QGPMaker_DCMotor *DCMotor_4 = AFMS.getMotor(4);
QGPMaker_DCMotor *DCMotor_1 = AFMS.getMotor(1);
QGPMaker_DCMotor *DCMotor_3 = AFMS.getMotor(3);
void forward() {
  DCMotor_1->setSpeed(200);
```

```
  DCMotor_1->run(FORWARD);
  DCMotor_2->setSpeed(200);
  DCMotor_2->run(FORWARD);
  DCMotor_3->setSpeed(200);
  DCMotor_3->run(FORWARD);
  DCMotor_4->setSpeed(200);
  DCMotor_4->run(FORWARD);
}

void turnLeft() {
  DCMotor_1->setSpeed(200);
  DCMotor_1->run(BACKWARD);
  DCMotor_2->setSpeed(200);
  DCMotor_2->run(BACKWARD);
  DCMotor_3->setSpeed(200);
  DCMotor_3->run(FORWARD);
  DCMotor_4->setSpeed(200);
  DCMotor_4->run(FORWARD);
}

void turnRight() {
  DCMotor_1->setSpeed(200);
  DCMotor_1->run(FORWARD);
  DCMotor_2->setSpeed(200);
  DCMotor_2->run(FORWARD);
  DCMotor_3->setSpeed(200);
  DCMotor_3->run(BACKWARD);
  DCMotor_4->setSpeed(200);
  DCMotor_4->run(BACKWARD);
```

```
}

void moveLeft() {
  DCMotor_1->setSpeed(200);
  DCMotor_1->run(BACKWARD);
  DCMotor_2->setSpeed(200);
  DCMotor_2->run(FORWARD);
  DCMotor_3->setSpeed(200);
  DCMotor_3->run(BACKWARD);
  DCMotor_4->setSpeed(200);
  DCMotor_4->run(FORWARD);
}

void moveRight() {
  DCMotor_1->setSpeed(200);
  DCMotor_1->run(FORWARD);
  DCMotor_2->setSpeed(200);
  DCMotor_2->run(BACKWARD);
  DCMotor_3->setSpeed(200);
  DCMotor_3->run(FORWARD);
  DCMotor_4->setSpeed(200);
  DCMotor_4->run(BACKWARD);
}

void backward() {
  DCMotor_1->setSpeed(200);
  DCMotor_1->run(BACKWARD);
  DCMotor_2->setSpeed(200);
  DCMotor_2->run(BACKWARD);
```

```
  DCMotor_3->setSpeed(200);
  DCMotor_3->run(BACKWARD);
  DCMotor_4->setSpeed(200);
  DCMotor_4->run(BACKWARD);
}

void stopMoving() {
  DCMotor_1->setSpeed(0);
  DCMotor_1->run(RELEASE);
  DCMotor_2->setSpeed(0);
  DCMotor_2->run(RELEASE);
  DCMotor_3->setSpeed(0);
  DCMotor_3->run(RELEASE);
  DCMotor_4->setSpeed(0);
  DCMotor_4->run(RELEASE);
}



void setup(){
  AFMS.begin(50);

  int error = 0;
  do{
    error = ps2x.config_gamepad(13, 11, 10, 12, true, true);
    if(error == 0){
      break;
    }else{
      delay(100);
    }
```

```
 }while(1);
 for(size_t i = 0; i < 50; i++)
 {
     ps2x.read_gamepad(false, 0);
     delay(10);
 }

 Servo0->writeServo(90);
 Servo1->writeServo(90);
 Servo2->writeServo(90);
 Servo3->writeServo(90);
 Servo4->writeServo(90);
 Servo5->writeServo(90);
 Servo6->writeServo(90);
 Servo7->writeServo(90);

 delay(1500);
Servo1->writeServo((Servo1->readDegrees() - 16)); // shoulder right

delay(1500);
Servo5->writeServo((Servo5->readDegrees() - 10)); // shoulder down

delay(1000);
Servo4->writeServo((Servo4->readDegrees() - 15)); // wrist up

delay(300);
Servo4->writeServo((Servo4->readDegrees() - 10)); // wrist up

delay(300);
```

```
Servo4->writeServo((Servo4->readDegrees() - 10)); // wrist up

delay(300);
Servo4->writeServo((Servo4->readDegrees() - 10)); // wrist up

delay(1500);
Servo5->writeServo((Servo5->readDegrees() + 6)); // shoulder up



delay(1500);
Servo5->writeServo((Servo5->readDegrees() + 6)); // shoulder up

delay(1000);
Servo3->writeServo((Servo3->readDegrees() - 30)); //claw open

delay(1000);
Servo5->writeServo((Servo5->readDegrees() + 8)); // shoulder up

delay(1000);
Servo5->writeServo((Servo5->readDegrees() + 7)); // shoulder up

delay(1000);
Servo3->writeServo((Servo3->readDegrees() + 27)); //claw close

delay(1000);
Servo5->writeServo((Servo5->readDegrees() - 10)); // shoulder up

delay(1000);
Servo5->writeServo((Servo5->readDegrees() - 10)); // shoulder up
```

```
delay(1000);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 13)); // shoulder left

delay(500);
Servo5->writeServo((Servo5->readDegrees() + 10)); // shoulder up

delay(500);
Servo3->writeServo((Servo3->readDegrees() - 30)); //claw open

 Servo0->writeServo(90);
 delay(200);
 Servo1->writeServo(90);
 delay(200);
```

```
 Servo2->writeServo(90);
 delay(200);
 Servo3->writeServo(90);
 delay(200);
 Servo4->writeServo(90);
 delay(200);
 Servo5->writeServo(90);

 Servo6->writeServo(90);
 Servo7->writeServo(90);
}


void Automatic()

{
 delay(1500);
Servo1->writeServo((Servo1->readDegrees() - 16)); // shoulder right

delay(1500);
Servo5->writeServo((Servo5->readDegrees() - 10)); // shoulder down

delay(1000);
Servo4->writeServo((Servo4->readDegrees() - 15)); // wrist up

delay(300);
Servo4->writeServo((Servo4->readDegrees() - 10)); // wrist up

delay(300);
```

```
Servo4->writeServo((Servo4->readDegrees() - 10)); // wrist up

delay(300);
Servo4->writeServo((Servo4->readDegrees() - 10)); // wrist up

delay(1500);
Servo5->writeServo((Servo5->readDegrees() + 6)); // shoulder up



delay(1500);
Servo5->writeServo((Servo5->readDegrees() + 6)); // shoulder up

delay(1000);
Servo3->writeServo((Servo3->readDegrees() - 30)); //claw open

delay(1000);
Servo5->writeServo((Servo5->readDegrees() + 8)); // shoulder up

delay(1000);
Servo5->writeServo((Servo5->readDegrees() + 7)); // shoulder up

delay(1000);
Servo3->writeServo((Servo3->readDegrees() + 27)); //claw close

delay(1000);
Servo5->writeServo((Servo5->readDegrees() - 10)); // shoulder up

delay(1000);
Servo5->writeServo((Servo5->readDegrees() - 10)); // shoulder up
```

```
delay(1000);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 10)); // shoulder left

 delay(500);
Servo1->writeServo((Servo1->readDegrees() + 13)); // shoulder left

delay(500);
Servo5->writeServo((Servo5->readDegrees() + 10)); // shoulder up

delay(500);
Servo3->writeServo((Servo3->readDegrees() - 30)); //claw open
}


/*
/*
```

```
delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}

delay(2000);
Servo4->writeServo((Servo4->readDegrees() - 1));}
```

```
*/


void loop(){


}
```