TRIBHUWAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHWOK CAMPUS

A project report on “Wireless Robotic Arm and its real life application”

Prepared By

Santosh Ray (080 BEI 036)

Naman Adhikari (080 BEI 025)

Neetika Neuapane (080 BEI 026)

Pawan Adhikari (080 BEI 027)

Prabin Chandra Gautam (080 BEI 028)

Rison Maharjan (080 BEI 033)

Safal Subedi (080 BEI 034)

Sameer Chaulagain (080 BEI 035)

A COURSE PROJECT SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE IN ELECTRONICS, COMMUNICATION AND INFORMATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KATHMANDU, NEPAL

BAISHAK 10, 2080

Acknowledgement

We would like to extend our heartfelt gratitude to the Electronics Department of Pulchwok Campus, and in particular, I wish to express our sincere appreciation to Mr. Suresh Jha and Mr. Jitendra Kumar Manandhar for their invaluable guidance and support throughout the completion of our electronic lab project.

Their expertise, dedication, and unwavering commitment have been instrumental in the success of this project. Their encouragement and insightful feedback have significantly enriched our learning experience and contributed immensely to the quality of our work.

We would also like to extend our thanks to all the other staff members and lab instructors in the Electronics Department who have contributed their time, knowledge, and resources to make this project possible.

We are truly grateful for the opportunity to learn and grow under their mentorship, and we look forward to applying the skills and knowledge gained from this experience in our future endeavors.

With sincere appreciation,

Santosh Ray

Naman Adhikari

Neetika Neuapane

Pawan Adhikari

Prabin Chandra Gautam

Rison Maharjan

Safal Subedi

Sameer Chaulagain

CONTENTS

INTRODUCTION

In the realm of robotics, integrating advanced functionalities into robotic systems is key to enhancing their utility and versatility. This project focuses on the development of a Wireless Robotic Arm equipped with a record and play feature, adding a new dimension of automation and control to its operation.

OBJECTIVES

The primary objective of this project is to design and implement a wireless robotic arm capable of recording and playing back sequences of movements. This feature aims to automate repetitive tasks, facilitate training and education, and enable precise control in various applications.

REAL-WORLD APPLICATIONS

The wireless robotic arm system developed in this project has various real-world applications, including:

• Manufacturing: Automated assembly lines and robotic workstations.

• Automation: Industrial processes, such as material handling and packaging.

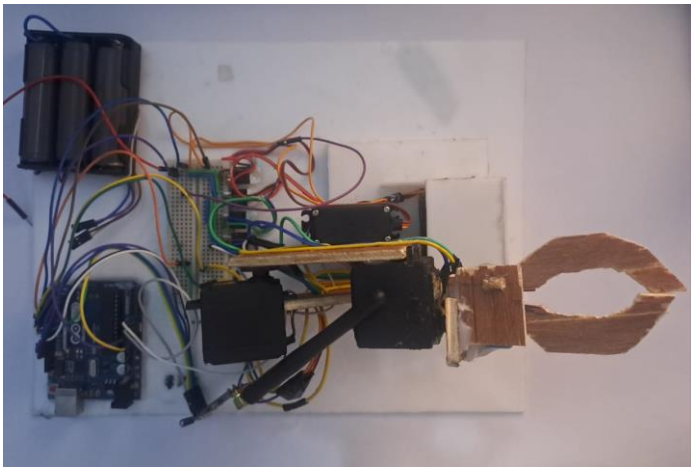
• Research: Laboratory experiments and scientific studies requiring precise manipulation.

• Education: STEM (Science, Technology, Engineering, and Mathematics) learning activities and robotics competitions.

COMPONENTS USE

The robotic arm and the controller comprises the following components, each serving a specific function:

ARM



1. Arduino UNO

- Background: The Arduino UNO is a popular open-source microcontroller board based on the atmega328p microcontroller chip. It was originally designed as an easy-to-use platform for hobbyists, artists, and designers to create interactive projects.

- Function: The Arduino UNO serves as the brain of the robotic arm, controlling its movements and operations. It can be programmed using the Arduino Integrated Development Environment (IDE), allowing users to write and upload code to define the arm's behavior.

2. Servo Motors (MG995R and SG90):

- Background: Servo motors are DC motors integrated with a feedback mechanism that allows for precise control of angular position, velocity, and acceleration. They are commonly used in robotics, RC vehicles, and automation applications.

- Function: In the robotic arm, servo motors actuate the various joints and components, enabling precise and controlled movement. Each servo motor is responsible for rotating a specific part of the arm, such as the base, arms, or gripper.

3. NRF24L01 Radio Module:

- Background: The NRF24L01 is a low-cost, low-power 2.4GHz transceiver module commonly used for wireless communication in Arduino projects. It provides a reliable and efficient way to establish communication between devices over short distances.

- Function: The NRF24L01 radio module enables wireless communication between the controller and the robotic arm. It allows the user to send control signals from the controller to the arm wirelessly, providing freedom of movement and remote operation.

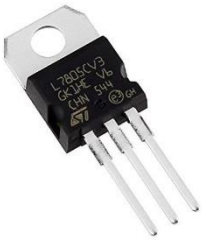
4. 18650 Li-ion Batteries:

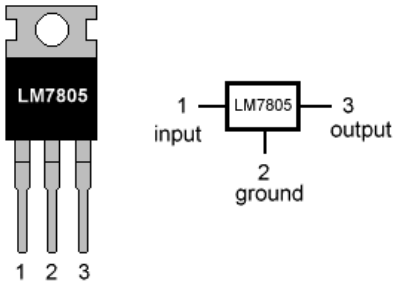
- Background: 18650 Li-ion batteries are rechargeable lithium-ion cells commonly used in portable electronic devices, power banks, and electric vehicles. They are known for their high energy density, long cycle life, and stable voltage output.
- Function: In the robotic arm, 18650 Li-ion batteries provide electrical power to the Arduino UNO, servo motors, and other electronic components. They offer a reliable and portable power source, allowing the arm to operate independently of external power supplies.

5. LM7805 Voltage Regulator:

- Background: The LM7805 is a three-terminal voltage regulator IC that converts an input voltage of 7-35V into a stable 5V output voltage. It is widely used in electronic circuits to provide a regulated power supply to components that require a constant voltage.

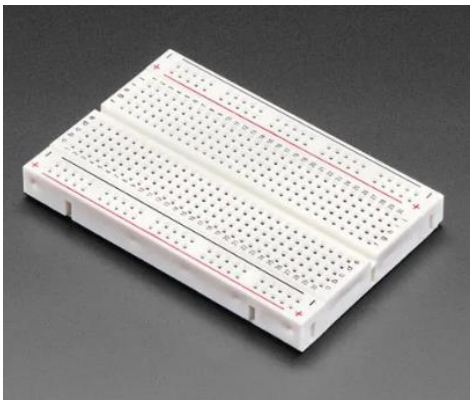
- Function: The LM7805 voltage regulator ensures that the Arduino UNO and servo motors receive a stable 5V power supply from the 18650 Li-ion batteries. It regulates the voltage output, preventing fluctuations and ensuring consistent performance of the electronic components.





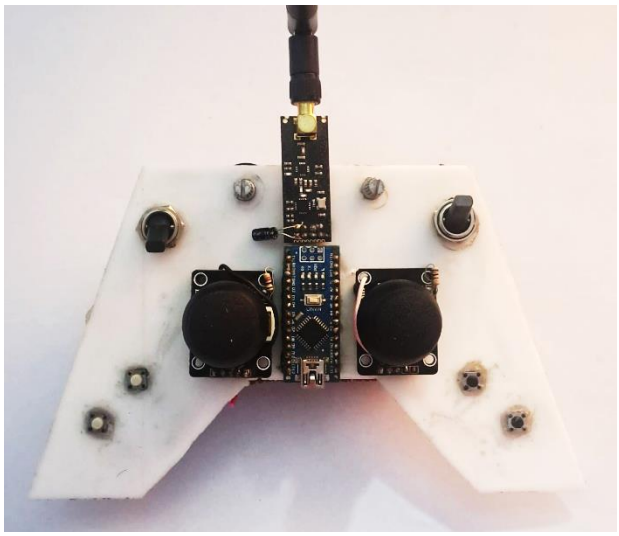
6. Breadboard, Jumper Wires:

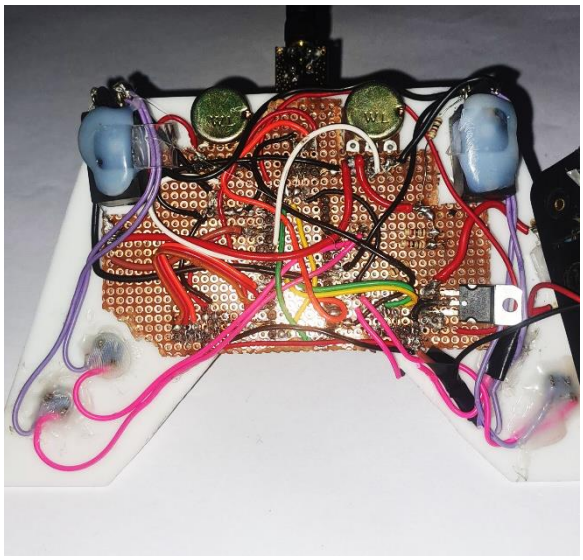
- The breadboard and jumper wires are used to create and organize the circuit connections between the electronic components in the robotic arm and controller. They provide a convenient and flexible platform for prototyping and assembling the circuitry, allowing for easy modifications or expansions as needed during the development process.



CONTROLLER

- Background: The custom controller is designed to provide intuitive and ergonomic control over the robotic arm. It typically consists of input devices such as joysticks, potentiometers, switches, and buttons, integrated with a microcontroller Arduino Nano for processing user inputs.



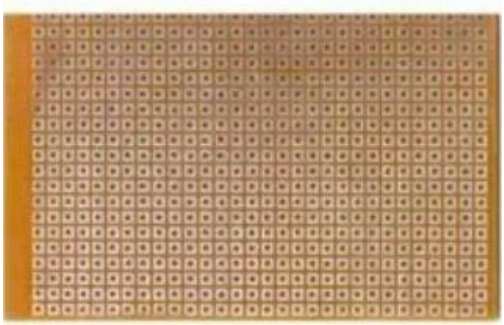


- Function: The custom controller allows the user to interact with the robotic arm, sending commands and controlling its movements. It provides a user-friendly interface for adjusting parameters, initiating actions, and switching between operating modes.

The controller contains the following Components

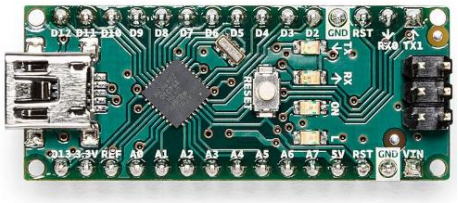
1. Zero PCB (Perfboard):

- The controller employs a zero PCB, also known as a perfboard, to construct a custom circuit layout for mounting and interconnecting electronic components and to create a permanent and compact circuit layout for the electronic components.
- Unlike a printed circuit board (PCB), a perfboard consists of perforated holes arranged in a grid pattern, allowing components to be soldered onto the board manually.
- The perfboard serves as a versatile platform for assembling circuits by hand, providing flexibility in component placement and circuit design.
- By soldering the components onto the board, it ensures secure connections, reduces the risk of loose connections or short circuits, and provides a professional finish to the controller assembly.

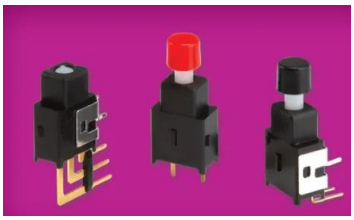


2. Arduino Nano:

- Serving as the brain of the controller, the Arduino Nano is a compact microcontroller board based on the ATmega328P chip.
- Mounted on the perfboard, the Arduino Nano facilitates the control and communication functions of the controller, enabling interaction with the robotic arm system.



3. Potentiometers, Joystick Modules, Toggle Switches, and Push Buttons:
   - These input devices enable user interaction with the controller, allowing for the transmission of commands to the robotic arm.
   - Potentiometers and joystick modules provide analog input signals corresponding to arm movements and configurations.
   - Toggle switches and push buttons offer digital input signals for mode selection, function activation, and other user commands.
   - Soldered onto the perfboard, these input components form the user interface of the controller, facilitating intuitive control over the robotic arm's operations.



4. NRF24L01 Radio Module:
   - Facilitating wireless communication between the controller and the robotic arm, the NRF24L01 2.4GHz radio module enables bidirectional data transmission.
   - Integrated into the perfboard design, the radio module allows the controller to send control commands to the robotic arm and receive status updates or acknowledgment signals in return.

5. 18650 Li-Ion Batteries and LM7805 Voltage Regulator:

- Powering the controller's operation, 18650 Li-Ion batteries provide a portable and rechargeable energy source.
- The LM7805 voltage regulator ensures a stable 5V supply to the Arduino Nano and other components, regulating the voltage output from the batteries to prevent fluctuations or overvoltage conditions.

6. Potentiometers and Joystick Modules:
    - Potentiometers and joystick modules provide analog input signals corresponding to arm movements and configurations.
7. Toggle Switches and Push Buttons:
    - Toggle switches and push buttons are used to activate or deactivate specific functions or modes of operation in the robotic arm. They provide digital input signals to the Arduino Nano, allowing the user to toggle between different operating modes, initiate specific actions, or adjust parameters dynamically.

SYSTEM OVERVIEW

The robotic arm designed and developed consists of several interconnected components that work together seamlessly to enable precise control and manipulation of the arm's movements. Here's an overview of the system and the flow of operations:

The robotic arm comprises five servo motors, namely one for rotation, two for arm movement, one for gripper rotation, and one for gripping action. The servos used are MG995R for base rotation, arm movement and gripper rotation whereas SG90 is used for gripping action.

The controller, designed to resemble a standard PS4 controller, is equipped with two joystick modules, two 10k potentiometers, two toggle switches, and four push buttons. These components provide analog and digital input data to an Arduino Nano microcontroller, which reads the input values and transmits it to the arm through NRF24L01.

Communication between the robotic arm and the controller is established using NRF24L01 radio modules, enabling wireless control of the arm's movements.

FLOW OF OPERATIONS

- The user interacts with the controller, adjusting the joystick modules, potentiometers, toggle switches, and push buttons to send control signals to the Arduino Nano.
- The Arduino Nano reads the analog and digital input values from the controller and constructs data packets containing the control commands for the robotic arm.
- These data packets are transmitted wirelessly via the NRF24L01 radio modules to the NRF24L01 modules connected to the Arduino Uno in the robotic arm.
- Upon receiving the control commands, the Arduino Uno processes the data packets and translates them into corresponding movements of the servo motors.
- Each servo motor adjusts its position according to the control commands received, resulting in the desired movements of the robotic arm, including rotation, arm movement, gripper rotation, and gripping action.
- The robotic arm executes the commanded movements, allowing users to control its actions remotely and manipulate objects with precision and accuracy.

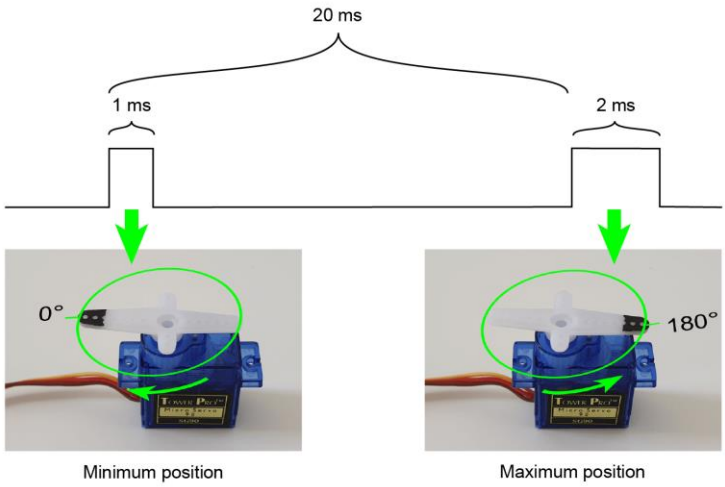
WORKING MECHANISM

1. Controlling servo motors with Arduino

Arduino microcontrollers can easily control servo motors using a technique called Pulse Width Modulation (PWM). Here's how it works:

A. PWM Signal:

- Servo motors require a specific type of control signal called a PWM signal to determine their position.
- Arduino boards have dedicated PWM pins that can output a PWM signal. PWM is a method used to generate analog-like signals using digital pins by rapidly switching the output pin between HIGH and LOW states.
- The PWM signal has a fixed frequency (usually around 490 Hz on Arduino boards), but the duty cycle (the percentage of time the signal is HIGH within each period) can be varied to control the servo position.
- The PWM signal for a servo typically has a frequency of around 50 Hz, with a pulse width ranging from 1 millisecond (ms) to 2 ms.
- The position of the servo is determined by the width of the PWM pulse: 1 ms corresponds to one extreme position, 1.5 ms corresponds to the center position, and 2 ms corresponds to the other extreme position.



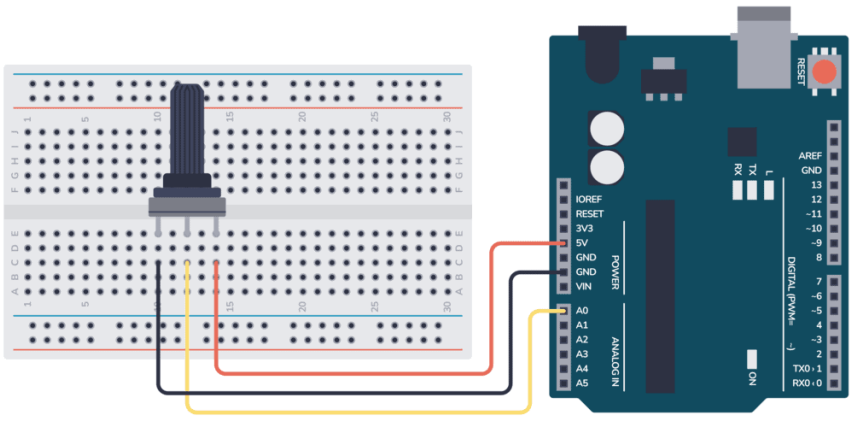
B. Arduino Servo Library:

- Arduino provides a built-in Servo library that simplifies the task of controlling servo motors.
- This library allows you to attach servo objects to specific pins, set their positions using angles or microseconds, and smoothly move them between positions.

2. Reading values from potentiometer, joystick module and switches:

A. Potentiometer and Joystick module (Analog Input):

Background: A potentiometer, or pot, is a variable resistor with three terminals: the wiper, the input voltage terminal (usually labeled as VCC), and the ground terminal (GND). As the wiper moves along the resistive track, it changes the resistance between the wiper and one of the other terminals, creating a variable voltage divider. Joystick modules typically consist of two potentiometers, one for the X-axis and one for the Y-axis, which change resistance based on the position of the joystick. Here's how the Arduino reads these values:



i. Analog-to-Digital Conversion (ADC):

- The analog output from the potentiometers of the joystick module is connected to the analog input pins (e.g., A0, A1) of the Arduino.
- The ADC measures the voltage on the analog pin and converts it to a digital value ranging from 0 to 1023, corresponding to voltages between 0 and the reference voltage (usually 5 volts for most Arduino boards).

ii. Reading Analog Input:

- In the Arduino sketch (program), the analogRead() function is used to read the voltage level on the analog input pin connected to the joystick.
- For example, if the joystick's X-axis is connected to analog pin A0 and the Y-axis is connected to A1, the Arduino code would read these values using analogRead(A0) and analogRead(A1) respectively.
- The analogRead() function returns a value between 0 and 1023, representing the analog voltage level at the pin.

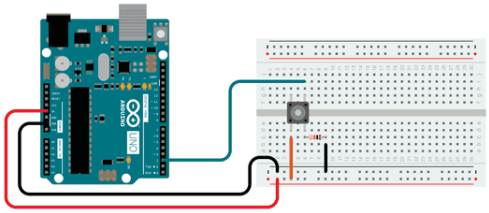
iii. Mapping Values:

- The raw analog readings from the joystick may not directly correspond to the desired range of motion or control for the robotic arm.
- To map these values to a specific range (e.g., 0-180 for servo motor positions), the Arduino sketch typically uses the map() function.
- The map() function scales the input value from one range to another. For example, mapping the raw analog reading (0-1023) to the desired output range (e.g., 0-180) for controlling servo motors.

In summary, the Arduino reads values from a joystick module and potentiometer by measuring the analog voltage levels on its analog input pins, converting them to digital values using ADC, and mapping these values to the desired range for controlling the robotic arm's movements.

B. Switch (Digital Input):

- Background: A switch is a simple electromechanical device that interrupts or completes an electrical circuit when it is toggled between its ON and OFF positions. It typically has two states: OPEN (OFF) and CLOSED (ON).



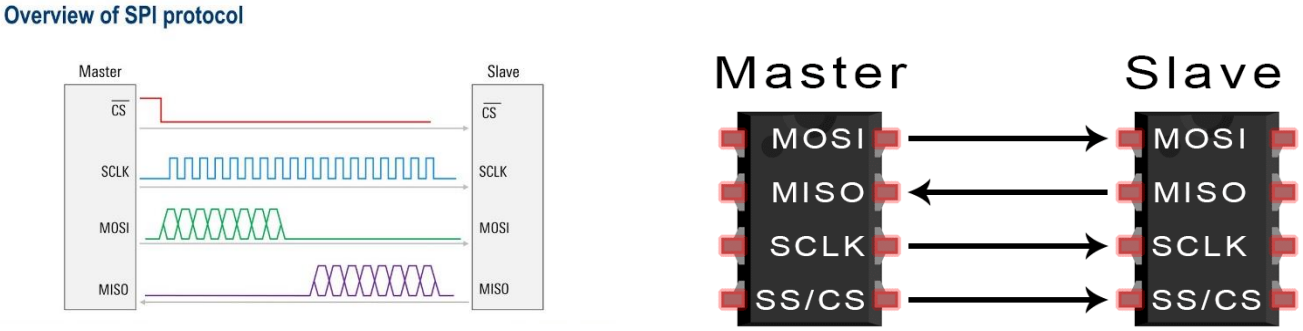
- Digital Data: When connected to a digital pin of a microcontroller, a switch acts as a digital input device. In its default state, it can be either OPEN (not making contact) or CLOSED (making contact). When the switch is pressed or toggled, it changes its state, and the digital pin reads either HIGH (1) or LOW (0) accordingly. In the Arduino sketch (program), the digitalRead() function is used to read the voltage level on the digital input pin.

3. NRF24LO1 radio module operation:

The NRF24L01 is a popular 2.4GHz transceiver module commonly used for wireless communication in Arduino projects. It employs a simple and efficient communication protocol that enables reliable data transmission between devices over short distances. Here's an explanation of the communication protocol used between NRF24L01 modules:

1. SPI (Serial Peripheral Interface):

In SPI (Serial Peripheral Interface) communication, MOSI, MISO, SCK, CSN, and CE are essential pins used for data exchange between the master (typically a microcontroller) and one or more slave devices (such as sensors, memory chips, or other microcontrollers). Here's what each pin does:



I. MOSI (Master Out Slave In):
MOSI is an output pin from the master device and an input pin for the slave device(s). The master sends data to the slave(s) using this pin. In other words, it carries data from the master to the slave(s) during SPI communication.

II. MISO (Master In Slave Out):

MISO is an input pin for the master device and an output pin for the slave device(s). The slave(s) send data to the master using this pin. It carries data from the slave(s) to the master during SPI communication.

III. SCK (Serial Clock):

SCK is the clock signal generated by the master to synchronize data transmission between the master and slave(s). It ensures that both devices are using the same clock frequency and timing for data exchange.

IV. CSN (Chip Select Not):

CSN is an active-low chip select signal used to select a specific slave device for communication. When CSN is LOW (active), it indicates that the master is communicating with the selected slave. When CSN is HIGH (inactive), it deselects the slave, allowing other devices to communicate on the SPI bus.

V. CE (Chip Enable):

CE is the chip enable signal used to activate the wireless transceiver module for sending or receiving data. When CE is HIGH, the module is enabled and ready to transmit or receive data. When CE is LOW, the module is disabled, and it enters a low-power standby mode.

2. Communication Protocol:

The NRF24L01 module uses a proprietary communication protocol that operates on the 2.4GHz ISM (Industrial, Scientific, and Medical) band. It employs packet-based communication with features such as data rate selection, channel selection, and automatic retransmission to ensure reliable and efficient data transfer.

The communication protocol typically involves the following steps:

I. Initialization:

- Both the transmitter and receiver modules must be initialized with the same configuration settings, such as data rate, channel frequency, address width, and payload size.
- The initialization process involves configuring the NRF24L01 registers using SPI commands sent from the microcontroller.

II. Addressing:

- Each NRF24L01 module has a unique address (or multiple addresses in the case of multi-node communication) used to identify the transmitter and receiver in the network.

- The transmitter and receiver must use compatible addresses to establish communication. This ensures that the receiver only accepts data from the designated transmitter and ignores signals from other devices.

III. Data Transmission:

- To send data, the transmitter module constructs a data packet containing the payload (the actual data to be transmitted) and the destination address (the address of the receiver).
- The transmitter then sends the data packet wirelessly using the RF (Radio Frequency) transmitter circuitry.
- The receiver module listens for incoming data packets on its configured channel and address. When a packet is received, it extracts the payload and processes the data as needed.

IV. Acknowledgment:

- After receiving a data packet, the receiver module sends an acknowledgment (ACK) packet back to the transmitter to confirm successful reception.
- If the transmitter does not receive an ACK within a specified time frame (timeout period), it assumes that the transmission was unsuccessful and may retransmit the data packet.

V. Error Handling:

- The NRF24L01 module includes features for error detection and correction, such as CRC (Cyclic Redundancy Check) and automatic retransmission.
- CRC ensures the integrity of the transmitted data by appending a checksum to each packet. The receiver verifies the checksum upon receiving the packet and discards any packets with errors.
- Automatic retransmission allows the transmitter to resend data packets if they are not acknowledged by the receiver, increasing the reliability of the communication link.

Overall, the NRF24L01 module provides a robust and efficient wireless communication solution for Arduino projects, enabling reliable data transmission between devices over short distances.

SOURCE CODE

Controller code(Transmitter Code)

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(7, 8); // CE, CSN
const byte address[6] = "00001";
```

```
int tg1=5;
int tg2=6;
int p1=A6;
int p2=A5;
int s1=10;
int s2=9;
int s3=3;
int s4=2;
int j1x=A3;
int j1y=A4;
int j2x=A2;
int j2y=A1;
int j2s=A0;
struct package {
int vp1;
int vp2;
int vj1x;
int vj1y;
int vj2x;
int vj2y;
int vtg1;
int vtg2;
int vj2s;
int vs1;
int vs2;
int vs3;
int vs4;
};
package data;
void setup() {
radio.begin();
radio.openWritingPipe(address);
radio.setPALevel(RF24_PA_MIN);
radio.stopListening();
Serial.begin(9600);
pinMode(tg1,INPUT);
pinMode(tg2,INPUT);
pinMode(p1,INPUT);
pinMode(p2,INPUT);
pinMode(s1,INPUT);
pinMode(s2,INPUT);
pinMode(s3,INPUT);
pinMode(s4,INPUT);
pinMode(j1x,INPUT);
pinMode(j1y,INPUT);
pinMode(j2y,INPUT);
```

```
pinMode(j2s,INPUT);
// Set initial default values
data.vj1x = 512;
data.vj1y = 512;
data.vj2x = 512;
data.vj2y = 512;
data.vp1 = 512;
data.vp2 = 512;
data.vj2s = 1;
data.vtg1 = 1;
data.vtg2 = 1;
data.vs1 = 1;
data.vs2 = 1;
data.vs3 = 1;
data.vs4 = 1;
}
void loop() {
// Read all analog inputs
data.vj1x = analogRead(j1x);
data.vj1y = analogRead(j1y);
data.vj2x = analogRead(j2x);
data.vj2y = analogRead(j2y);
data.vp1 = analogRead(p1);
data.vp2 = analogRead(p2);
// Read all digital inputs
data.vj2s = digitalRead(j2s);
data.vtg1 = digitalRead(tg1);
data.vtg2 = digitalRead(tg2);
data.vs1 = digitalRead(10);
data.vs2 = digitalRead(s2);
data.vs3 = digitalRead(s3);
data.vs4 = digitalRead(s4);
radio.write(&data, sizeof(package));
}
```

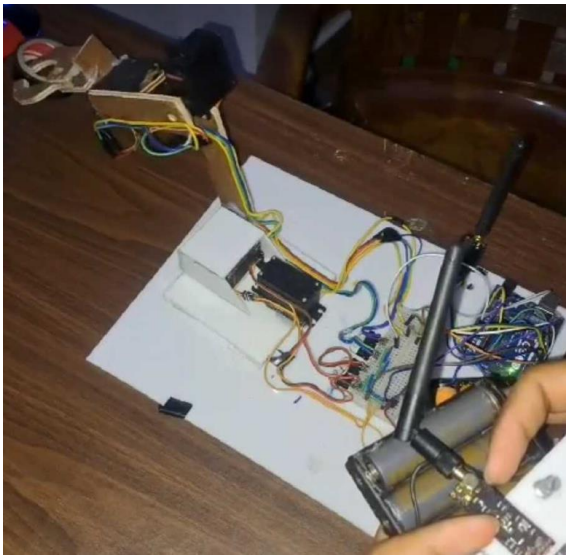
Recover Arm Source Code

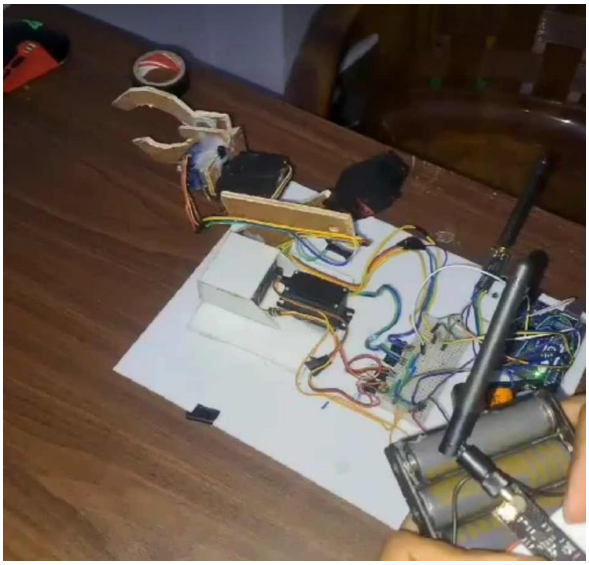
```
#include <Servo.h>
Servo base;
Servo s1;
Servo s2;
Servo s3;
Servo s4;
```

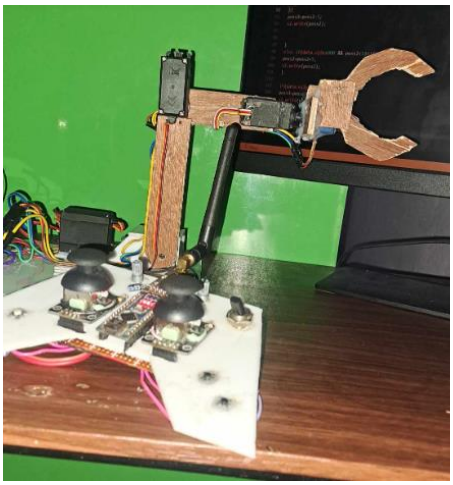
```
int pos = 0;
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(7, 8); // CE, CSN
const byte address[6] = "00001";
struct package {
int vp1;
int vp2;
int vj1x;
int vj1y;
int vj2x;
int vj2y;
int vtg1;
int vtg2;
int vj2s;
int vs1;
int vs2;
int vs3;
int vs4;
};
package data;
int posbase=90;
int poss1=90;
int poss2=90;
int poss3=90;
int poss4=80;
void setup() {
base.attach(2);
s1.attach(3);
s2.attach(4);
s3.attach(5);
s4.attach(6);
Serial.begin(9600);
radio.begin();
radio.openReadingPipe(0, address);
radio.setPALevel(RF24_PA_MIN);
radio.startListening(); // Set the module as receiver
base.write(posbase);
s1.write(poss1);
s2.write(poss2);
s3.write(poss3);
s4.write(poss4);
}
void loop() {
//printdata();
```

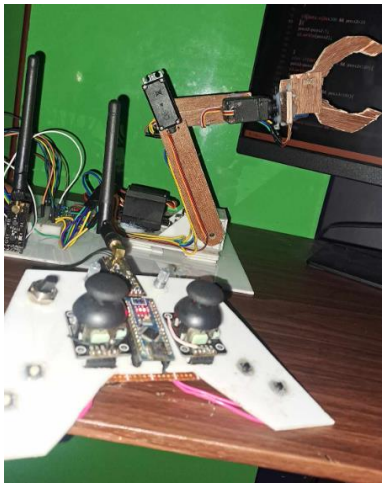
```
if (radio.available()) {
radio.read(&data, sizeof(package));
posbase=map(data.vp1,0,1023,170,10);
base.write(posbase);
poss4=map(data.vp2,0,1023,60,110);
s4.write(poss4);
if(data.vj1x<400 && poss1<180){
poss1=poss1+3;
s1.write(poss1);
}
else if(data.vj1x>600 && poss1>10){
poss1=poss1-3;
s1.write(poss1);
}
if(data.vj2x<300 && poss2>20
){
poss2=poss2-5;
s2.write(poss2);
}
else if(data.vj2x>800 && poss2<180){
poss2=poss2+5;
s2.write(poss2);
}
if(data.vj2y>800 && poss3<180){
poss3=poss3+5;
s3.write(poss3);
}
else if(data.vj2y<200 && poss3>0){
poss3=poss3-5;
s3.write(poss3);
}
}
```

PROJECT PICTURE









CONCLUSION

The Wireless Robotic Arm developed in this project demonstrates the integration of advanced functionalities and wireless control capabilities into a versatile and adaptable robotic system. By leveraging components such as Arduino microcontrollers, servo motors, and NRF24L01 radio modules, the system enables precise and intuitive control over the arm's movements, making it suitable for a wide range of applications in manufacturing, automation, research, and education.

FUTURE ENHANCEMENTS

While the current iteration of the Wireless Robotic Arm fulfills its primary objectives, there are several avenues for future enhancements and improvements:

- Enhanced Control Interface: Develop a user-friendly control interface with graphical elements and interactive features to simplify operation and enhance user experience.
- Autonomous Operation: Implement advanced control algorithms and sensor integration to enable autonomous operation and adaptive behavior in response to changing environmental conditions.
- Expansion and Customization: Explore opportunities for expanding the functionality of the robotic arm, such as adding additional degrees of freedom, integrating sensors for environmental perception, or customizing end-effectors for specific tasks.
- Integration with AI: Investigate the integration of artificial intelligence (AI) techniques, such as machine learning and computer vision, to enable the robotic arm to learn from experience, recognize objects, and adapt its behavior accordingly.

By continually refining and evolving the design of the Wireless Robotic Arm, it is possible to unlock new capabilities and applications, further advancing the field of robotics and automation.

REFERENCES

- Arduino - Home. (n.d.). Arduino. Retrieved December 23, 2023, from https://www.arduino.cc/
- NRF24L01 Wireless Transceiver Module - How to Interface With Arduino. (2022, January 17). Circuit Digest. https://circuitdigest.com/microcontroller-projects/arduino-nrf24l01-wireless-transceiver-module-interfacing-tutorial
- Lee, H. (2021, August 26). How to use NRF24L01 wireless transceiver module with Arduino. The Engineering Projects. https://www.theengineeringprojects.com/2021/08/how-to-use-nrf24l01-wireless-transceiver-module-with-arduino.html
- Singh, J. (2021, December 20). Complete guide for NRF24L01 Transceiver Module With Arduino. Maker Pro. https://maker.pro/arduino/projects/complete-guide-for-nrf24l01-transceiver-module-with-arduino
- Potentiometer - Working, Circuit Diagram, Construction & Types. (2022, January 13). ElProCus. https://www.elprocus.com/potentiometer-working-circuit-diagram-construction-types/
- Joystick Module for Arduino Tutorial (2021, November 18). Arduino Project Hub. https://create.arduino.cc/projecthub/shajeeb/joystick-module-for-arduino-tutorial-b772b2
- Simon Monk. (2013). Programming Arduino: Getting Started with Sketches (p. 179). McGraw-Hill Education.