ROBOTIC DUEL

Excitino

A Arduino based departmental activity

ELE531: Embedded Systems and IOT Fundamentals

VELAN E
224015, B.Sc CME
`velan.e@cme.christuniversity.in`

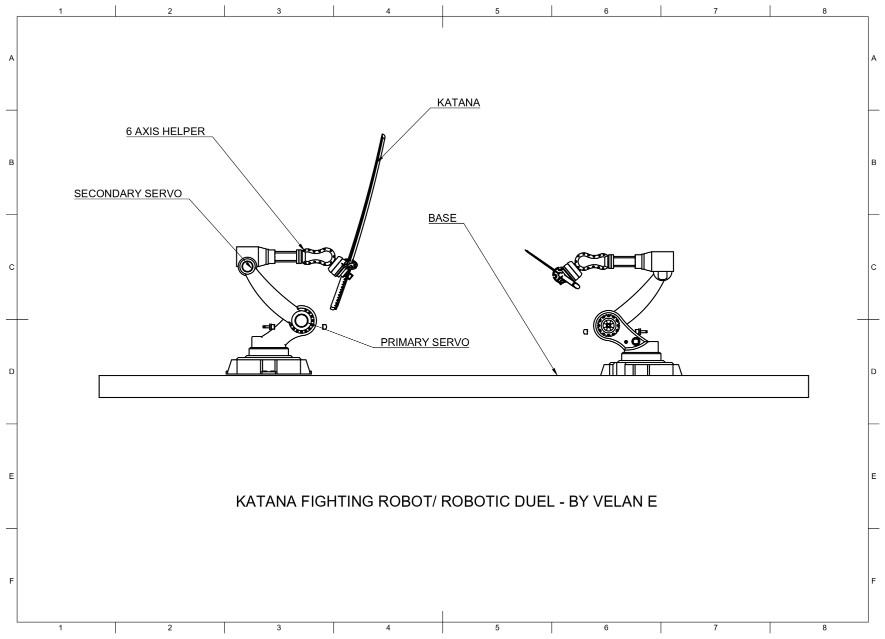
Under Supervision of Dr. Mukund N Naragund

Department of Physics and Electronics
September 2024

Abstract

This paper presents the design, development, and implementation of a 'robotic duel' project made for the Departmental activity "EXCITINO" which focuses on Arduino projects. This is a unique and innovative concept for educational and entertainment purposes. The project features two autonomous robots equipped with various sensors, actuators, and an Arduino microcontroller for decision-making and control. The robots are designed to engage in simulated combat scenarios, utilizing their sensors to detect opponent movement. The robots are equipped with mechanisms for attacking and defending, ensuring a safe and engaging duel experience. Additionally, it provides a unique and exciting form of entertainment for audiences of all ages.

INTRODUCTION



The "Robotic Duel" project is an exciting venture into the world of robotics, concentrating on the creation and deployment of combat-ready robotic arms. By utilizing an **Arduino Uno** as the main microcontroller, this project employs various servo motors to enable intricate movements and precise control over the robotic elements. In robotics, effective motion control is essential for performing tasks with precision and agility. These robotic arms are crafted to be as lightweight as possible, balancing the entire weight on sensitive contacts while imitating martial arts movements, allowing them to participate in simulated duels. This project requires a robust system for managing the servos, ensuring smooth and fast actions throughout the process.

1 Block Diagram

1.1 Aim

The aim of this project is primarily for leisure and showcase purposes and does not have immediate practical applications. However, in real-world scenarios, similar systems are used in industrial robotic modules.

1.2 Description

This project uses an **Arduino Uno microcontroller** to control servo motors. The servo motors are connected to Arduino's digital pins, and an external power supply provides sufficient power. A **servo motor** is a precise actuator that controls position, velocity, and acceleration. It has a motor, a sensor for feedback, and a controller.

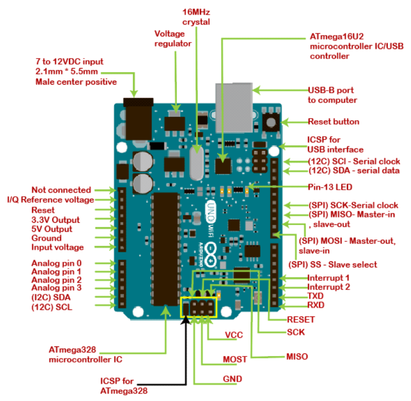


Figure 1: Arduino Uno Board

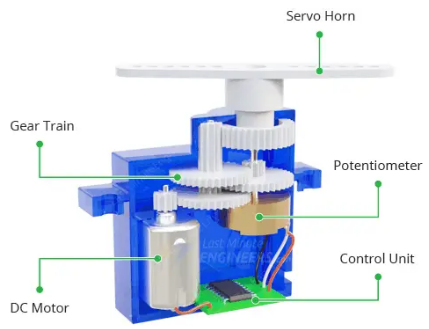


Figure 2: Servo Motor

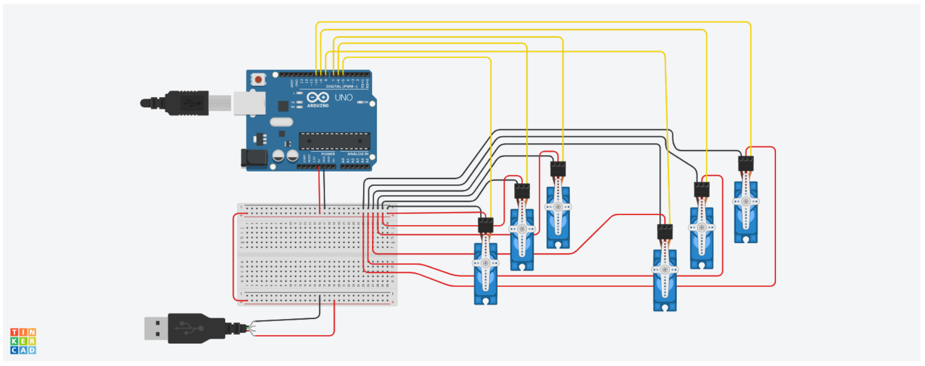


Figure 3: Circuit diagram

2 Circuit diagram

Components used: **Arduino UNO R3, Servo motor, Breadboard, External power supply**.

3 Working Description

The circuit consists of an Arduino Uno connected to one or more servo motors, allowing for precise control of the motors' positions through PWM signals.

The servo motor's power wire (red) is connected to the 5V pin on the Arduino.
The ground wire to the GND pin.
And the control wire to a PWM-enabled digital pin.

The Arduino produces PWM signals on designated pins linked to the servo motors, where the pulse width dictates the angle of the motor shaft: generally, a pulse width of 1ms aligns the servo at 0 degrees, 1.5ms at 90 degrees, and 2ms at 180 degrees. The Servo library is utilized to program the Arduino, enabling it to transmit the correct PWM signals to the motors. For instance, a simple Arduino sketch can rotate the servo to specific angles with pauses in between movements. When using multiple servos, it is crucial to confirm that the Arduino can provide enough power; otherwise, an external power source may be necessary for the motors.

4 Arduino program

```
#include <Servo.h>

Servo pr1, sr1, r1;

int pr1Pin = 5;
int sr1Pin = 6;
int r1Pin = 7;

void respect() {
  r1.write(0);
  delay(250);
  pr1.write(120);
  delay(500);
  pr1.write(30);
  delay(250);
  r1.write(90);
  delay(250);
  pr1.write(30);}

void idle() {
  pr1.write(30);
  sr1.write(25);
  r1.write(90);}

void verticalslash() {
  sr1.write(90);
  pr1.write(90);
  r1.write(180);
  delay(400);
  r1.write(0);}

void horislash() {
  r1.write(90);
  pr1.write(90);
  sr1.write(180);
  delay(400);
  sr1.write(0);}

void sslash() {
  r1.write(90);
  pr1.write(90);
  for (int i = 0; i <= 180; i++) {
    sr1.write(i);
    r1.write(i);
    delay(5);}
  delay(400);
  for (int i = 180; i >= 2; i--) {
    sr1.write(i);
    r1.write(i);
    delay(5);}}

void stab() {
  r1.write(90);
  pr1.write(0);
  sr1.write(90);
  for (int i = 0; i <= 180; i++) {
    pr1.write(i);
    sr1.write(90 + i);
    delay(5);}}

void protect() {
  pr1.write(0);
  sr1.write(90);
  r1.write(180);
  delay(750);}

void setup() {
  pr1.attach(pr1Pin);
  sr1.attach(sr1Pin);
  r1.attach(r1Pin);

  Serial.begin(9600);}

void loop() {
  if (Serial.available() > 0) {
    int choice = Serial.parseInt();
    switch (choice) {
      case 1: verticalslash(); break;
      case 2: horislash(); break;
      case 3: sslash(); break;
      case 4: stab(); break;
      case 5: protect(); break;
      case 6: lowslash(); break;
      case 7: respect(); break;
      default: Serial.println("Invalid␣choice!");}}}
```

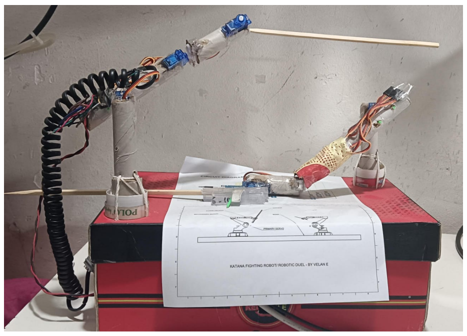


Figure 4: Image of the project-The right side robotic arm got defeated in the fight

The Servo Library is a great library for controlling servo motors. In this article, you will find two easy examples that can be used by any Arduino board.

RESULT AND DISCUSSION

The project observations showed that servo motors effectively responded to control commands via the serial interface, leading to precise and coordinated movements of the robotic arms, with reliable performance and minimal delays.

The observed movements are:

Respect, Idle, Vertical slash, Horizontal slash, Slash, Stab and Protect

It had some variations in servo angles, indicating areas for improvement. User interaction via the serial interface was intuitive, allowing easy action selection and feedback. The modular code design facilitated the integration of new movements, enhancing the robotic arms' capabilities. Results showed that the Arduino Uno with servo motors effectively created responsive robotic arms for complex movements. Future enhancements could involve better sensor integration, like ultrasonic sensors. Overall, the project offered valuable experience in programming, electronics, and robotics, emphasizing collaboration and troubleshooting skills.

CONCLUSION

The Arduino-controlled Katana Fighting Robotic Arms project showcased a responsive robotic system that executed precise movements using servo motors. It offered valuable programming and robotics experience, identified improvement areas like sensor integration, and highlighted the technology's potential in education and competition.

References

[1] O'Reilly. (n.d.). *Introduction to Inverse Kinematics.* Retrieved from https://www.oreilly.com/library/view/learning-robotics/9781782168179/9cb2de38-6b7b-4f23-bc7a-91e2958492cd.xhtml

[2] YouTube. (n.d.). *Katana Movements Tutorial for Arduino.* Retrieved from https://youtu.be/MQ-c$_e - Zc3U?si = OQlC8XGHf1TeQe1B$

[3] Yoshida, K. (2004). *The Samurai Sword: Spirit * Strategy * Techniques.* ISBN-10: 0804837511.

AUTHOR'S PROFILE

Author:
Velan E
2240153, B.Sc CME
CHRIST (Deemed to be University)
Bengaluru, India.
`velan.e@cme.christuniversity.in`
`velane929@gmail.com`