**1)DC MOTOR**

A black and silver electric motor

Description automatically generated- Commonly used for a mobile

robot base like differential drive,

omni-wheeled and

mecanum wheeled robot.

- It also usually being used for any

rotational movement that require

high speed or high torque.

**ADVANTAGES:**

1. **Precise Speed Control:** Easily adjustable speed for various applications.
2. **High Starting Torque:** Ideal for tasks requiring strong initial force.
3. **Simplicity and Cost:** Simple design and control, leading to lower initial costs and easier maintenance.
4. **Versatility:** Suitable for a wide range of applications.
5. **Quick Response:** Rapid response to control signals.
6. **High Efficiency:** Particularly efficient at lower speeds.

**DISADVANTAGES:**

1. **Maintenance Needs:** Regular upkeep for brushes and commutator.
2. **Size and Weight:** Larger and heavier compared to AC motors of the same power.
3. **Component Costs:** Brushes and commutator add to overall cost.
4. **Voltage Limitations:** Generally suited for low voltage applications.
5. **Reduced Lifespan:** Shorter lifespan due to mechanical wear.
6. **Sparking Issues:** Not suitable for hazardous environments due to potential sparking.

**WIRING DIAGRAM :**

**A diagram of a circuit board

Description automatically generated**

**SAMPLE CODE :**

// Define the pins

const int dirPin = 8;

const int pwmPin = 9;

void setup() {

// Set the direction pin as an output

pinMode(dirPin, OUTPUT);

// Set the PWM pin as an output

pinMode(pwmPin, OUTPUT);

}

void loop() {

// Turn the motor clockwise

digitalWrite(dirPin, HIGH); // Set direction to clockwise

analogWrite(pwmPin, 255); // Set speed (0-255)

// Wait for 2 seconds

delay(2000);

// Stop the motor

analogWrite(pwmPin, 0); // Set speed to 0

// Wait for 1 second

delay(1000);

// Turn the motor counterclockwise

digitalWrite(dirPin, LOW); // Set direction to counterclockwise

analogWrite(pwmPin, 255); // Set speed (0-255)

// Wait for 2 seconds

delay(2000);

// Stop the motor

analogWrite(pwmPin, 0); // Set speed to 0

// Wait for 1 second

delay(1000);

}

**2) SERVO MOTOR**

A small blue motor with a red wire

Description automatically generated- Commonly used to controlthe joints

of robotic arms,allowing for precise

positioningand movement of the

arm segments.

- It also being used to control the

opening and closing of grippers

or other end effectors on the robot,

enabling it to grasp and manipulate objects.

**ADVANTAGES:**

1. **Precision and Accuracy**: Servo motors provide precise control of angular or linear position, velocity, and acceleration, making them ideal for applications requiring high accuracy.
2. **High Torque**: They can generate high torque at low speeds, which is beneficial for applications needing strong and controlled movement.
3. **Speed Control**: Servo motors offer excellent speed control, enabling smooth and precise motion profiles.
4. **Closed-Loop System**: The built-in feedback system (usually involving encoders) allows for accurate positioning and speed control, as well as error correction.
5. **Efficiency**: They are highly efficient in converting electrical energy into mechanical energy, leading to less heat generation and lower energy consumption.
6. **Versatility**: Available in various sizes and configurations, making them suitable for a wide range of applications from small hobby projects to industrial machinery.

**DISADVANTAGES:**

1. **Cost**: Servo motors are generally more expensive compared to other types of motors, such as stepper motors.
2. **Complexity**: The control systems for servo motors are more complex, requiring specialized knowledge and equipment to implement and maintain.
3. **Maintenance**: They often require more maintenance due to the presence of brushes (in brushed servo motors) and the complexity of the feedback system.
4. **Limited Rotation**: Most servo motors are designed for limited rotation (typically 180 degrees or less), which may not be suitable for applications requiring continuous rotation.
5. **Heat Dissipation**: At high loads, servo motors can generate significant heat, necessitating cooling systems to prevent overheating.

**WIRING DIAGRAM:**

**A circuit board with wires

Description automatically generated**

**SAMPLE CODE :**

#include <Servo.h>

Servo myservo; // create servo object to control a servo

// twelve servo objects can be created on most boards

int pos = 0; // variable to store the servo position

void setup() {

myservo.attach(9); // attaches the servo on pin 9 to the servo object

}

void loop() {

for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

// in steps of 1 degree

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

myservo.write(pos); // tell servo to go to position in variable 'pos'

delay(15); // waits 15ms for the servo to reach the position

}

}

**3) DYNAMIXEL SERVO**

A small black wheel with a black wheel

Description automatically generated**- Robotic Arms:** Dynamixel servos are frequently used in the joints of robotic arms to provide smooth and precise movements.

**- Humanoid Robots:** In humanoid robots, these servos are used in the joints of limbs, including elbows, knees, shoulders, and hips, to mimic human motion.

- **Quadrupeds and Hexapods**: Dynamixel servos are used in the legs of these robots to provide locomotion capabilities.

**- Claws and Grippers:** For industrial robots, Dynamixel servos are used in the actuation of claws and grippers to handle objects.

**ADVANTAGES:**

1. **High Precision and Control :** Real-time feedback on position, speed, temperature, and load. High-resolution control for accurate and smooth movements.
2. **Ease of Integration :** Daisy-chain capability simplifies wiring and control architecture. Compatibility with various communication protocols (e.g., TTL, RS485).
3. **Programmability :** Customizable parameters such as speed, torque, and position limits.
4. **Durability and Reliability :** Robust design ensures longevity and reliability in various conditions.
5. **Versatility :** Wide range of models suitable for different applications, from small hobby projects to large industrial robots.

**DISADVANTAGES:**

1. **Cost :** Generally more expensive than standard servos, which can be a limiting factor for budget-conscious projects.
2. **Complexity :** Higher learning curve requiring advanced programming skills, challenging for beginners.
3. **Power Requirements :** Higher power consumption compared to standard servos, requiring robust power management.
4. **Maintenance :** Potential for overheating under heavy loads, necessitating proper cooling and maintenance.

**WIRING DIAGRAM:**

**A circuit board with wires

Description automatically generated**

**SAMPLE CODE:**

#include <SoftHalfDuplexSerial.h>

#include <DynamixelAx.h>

softHalfDuplexSerial port(8);  // data pin 8

dxlAx dxlCom(&port);

String \_readString;  // Input string from serial monitor

bool \_strComplete = false;

int \_id = 1;  // Default Dynamixel servo ID

void printServoId(String msg);

void printDxlResult();

void printDxlError(unsigned short dxlError);

void setup() {

  // Open serial communications and wait for port to open (PC communication)

  Serial.begin(57600);

  while (!Serial) {

    ;  // wait for serial port to connect. Needed for native USB port only

  }

  Serial.println("Starting COM!");

  dxlCom.begin(57600);

}

/////////////////////////////////////////////////////////////////////////////////////

void loop() {

  while (Serial.available()) {

    char inputChar = Serial.read();  //gets one byte from serial buffer

    \_readString += inputChar;        //makes the string readString

    if (inputChar == '\n')

      \_strComplete = true;

  }

  if (\_strComplete) {

    \_strComplete = false;

    if (\_readString.startsWith("ID")) {

      \_readString.remove(0, 2);

      \_id = \_readString.toInt();  //convert readString into a number

      printServoId("Communicating with");

      Serial.println(\_id);

    } else if (\_readString.startsWith("ping")) {

      printServoId("Ping");

      dxlCom.ping(\_id);

      printDxlResult();

    } else if (\_readString.startsWith("action")) {

      dxlCom.isRegistered(\_id);

      while (!dxlCom.dxlDataReady())

        ;  // waiting the answer of servo

      printDxlError(dxlCom.readDxlError());

      if (dxlCom.readDxlResult())  // if it is registred

      {

        printServoId("Execute reg command in");

        while (dxlCom.isBusy())

          ;  // waiting the status return delay time

        dxlCom.action(\_id);

        printDxlResult();

      } else {

        printServoId("No reg command in");

        Serial.println();

      }

    } else if (\_readString.startsWith("reboot")) {

      printServoId("Reboot (not supported by MX)");

      dxlCom.reboot(\_id);

      printDxlResult();

    } else if (\_readString.startsWith("model")) {

      printServoId("Model number of");

      dxlCom.readModelNumber(\_id);

      printDxlResult();

    } else if (\_readString.startsWith("firmware")) {

      printServoId("Firmware number of");

      dxlCom.readFirmware(\_id);

      printDxlResult();

    } else if (\_readString.startsWith("setID")) {

      \_readString.remove(0, 5);

      int newID = \_readString.toInt();  //convert readString into a number

      printServoId("Setting");

      Serial.print(\_id);

      Serial.print(" to ");

      Serial.print(newID);

      Serial.print(" : ");

      dxlCom.setId(\_id, newID);

      printDxlResult();

    } else if (\_readString.startsWith("led")) {

      \_readString.remove(0, 3);

      bool Status = \_readString.toInt();  //convert readString into a number

      printServoId("Changing led status of ");

      dxlCom.setLedEnable(\_id, Status);

      printDxlResult();

    } else if (\_readString.startsWith("move")) {

      \_readString.remove(0, 4);

      unsigned short Position = \_readString.toInt();  //convert readString into a number

      printServoId("Moving ");

      dxlCom.setGoalPosition(\_id, Position);

      printDxlResult();

      bool isMoving = true;

      while (isMoving) {

        unsigned short error = DXL\_ERR\_SUCCESS;

        while (dxlCom.isBusy())

          ;  // waiting the status return delay time

        dxlCom.readPresentPosition(\_id);

        Serial.print("Pos : ");

        while (!dxlCom.dxlDataReady())

          ;  // waiting the answer of servo

        error = dxlCom.readDxlError();

        if (error != DXL\_ERR\_SUCCESS)  // readDxlResult should always be called before readDxlData

          printDxlError(error);

        Serial.println(dxlCom.readDxlResult());

        while (dxlCom.isBusy())

          ;  // waiting the status return delay time (for testing if it is moving)

        dxlCom.isMoving(\_id);

        while (!dxlCom.dxlDataReady())

          ;  // waiting the answer of servo

        error = dxlCom.readDxlError();

        if (error != DXL\_ERR\_SUCCESS)  // readDxlResult should always be called before readDxlData

          printDxlError(error);

        isMoving = dxlCom.readDxlResult();

      }

    } else if (\_readString.startsWith("speed")) {

      \_readString.remove(0, 5);

      unsigned short Speed = \_readString.toInt();  //convert readString into a number

      printServoId("Set speed of ");

      dxlCom.setMovingSpeed(\_id, Speed);

      printDxlResult();

    } else if (\_readString.startsWith("torque")) {

      \_readString.remove(0, 6);

      unsigned short torque = \_readString.toInt();  //convert readString into a number

      printServoId("Set torque of ");

      dxlCom.setTorqueLimit(\_id, torque);

      printDxlResult();

    } else if (\_readString.startsWith("voltage")) {

      printServoId("Voltage (to be divided by 10) of");

      dxlCom.readVoltage(\_id);

      printDxlResult();

    } else if (\_readString.startsWith("temperature")) {

      printServoId("Temperature of");

      dxlCom.readTemperature(\_id);

      printDxlResult();

    } else if (\_readString.startsWith("regmove")) {

      \_readString.remove(0, 7);

      unsigned short Position = \_readString.toInt();  //convert readString into a number

      printServoId("Write command (type 'action' to execute) in REG register of ");

      dxlCom.sendDxlRegData(\_id, DXL\_ADD\_GOAL\_POSITION, (const byte\*)&Position, 2);

      printDxlResult();

    }

    \_readString = "";  //empty for next input

  }

}

/////////////////////////////////////////////////////////////////////////////////////

void printDxlResult() {

  while (!dxlCom.dxlDataReady())

    ;  // waiting the answer of servo

  printDxlError(dxlCom.readDxlError());

  Serial.println(dxlCom.readDxlResult());

}

void printServoId(String msg) {

  Serial.print(msg);

  Serial.print(" servo ID ");

  Serial.print(\_id);

  Serial.print(" - ");

}

void printDxlError(unsigned short dxlError) {

  // after any operation error can be retrieve using dx::readDxlResult() (i.e. after read or write operation)

  if (dxlError == DXL\_ERR\_SUCCESS)

    Serial.println("OK");

  else {

    if (dxlError & DXL\_ERR\_VOLTAGE)

      Serial.print("voltage out of range-");

    if (dxlError & DXL\_ERR\_ANGLE)

      Serial.print("angle out of range-");

    if (dxlError & DXL\_ERR\_OVERHEATING)

      Serial.print("overheating-");

    if (dxlError & DXL\_ERR\_RANGE)

      Serial.print("cmd out of range-");

    if (dxlError & DXL\_ERR\_TX\_CHECKSUM)

      Serial.print("Tx CRC invalid-");

    if (dxlError & DXL\_ERR\_OVERLOAD)

      Serial.print("overload-");

    if (dxlError & DXL\_ERR\_INSTRUCTION)

      Serial.print("undefined instruction-");

    if (dxlError & DXL\_ERR\_TX\_FAIL)

      Serial.print("Tx No header-");

    if (dxlError & DXL\_ERR\_RX\_FAIL)

      Serial.print("Rx No header-");

    if (dxlError & DXL\_ERR\_TX\_ERROR)

      Serial.print("Tx error-");

    if (dxlError & DXL\_ERR\_RX\_LENGTH)

      Serial.print("Rx length invalid-");  // Not implemented yet

    if (dxlError & DXL\_ERR\_RX\_TIMEOUT)

      Serial.print("timeout-");

    if (dxlError & DXL\_ERR\_RX\_CORRUPT)

      Serial.print("Rx CRC invalid-");

    if (dxlError & DXL\_ERR\_ID)

      Serial.print("Wrong ID answered-");  // ?? Hardware issue

    Serial.println();

  }

}

A small electric motor with wires

Description automatically generated**4) STEPPER MOTOR**

**- Robotic Arms:** Stepper motors are often used to control the movement of joints in robotic arms, providing precise positioning and repeatability necessary for tasks such as assembly, welding, or painting.

* + - **Humanoid Robots:** In humanoid robots, stepper motors can be used in the joints to mimic human-like movements with a high degree of accuracy.
    - **Linear Actuators:** Frequently used in linear actuators for tasks that require precise linear movement, such as in pick-and-place operations or 3D printing.
    - **XY Tables:** In CNC machines and 3D printers, stepper motors control the movement of the print head or cutting tool along the X and Y axes.

**ADVANTAGES:**

1. **Precision and Accuracy :** Provides precise control over movement without needing feedback systems. Consistently returns to the same position with high accuracy.
2. **Ease of Control :** Can be controlled without complex feedback systems, simplifying the control mechanism. Operates effectively with simple digital control signals.
3. **Reliability :** Typically, stepper motors are brushless, reducing mechanical wear and increasing longevity. Less prone to mechanical failure, resulting in lower maintenance needs.
4. **Holding Torque :** Can hold its position firmly when not moving, making it useful in applications where maintaining a fixed position is critical.
5. **Cost-Effective :** Generally less expensive than servomotors, especially for applications not requiring closed-loop control.

**DISADVANTAGES:**

1. **Efficiency and Heat :** Can consume more power, especially at higher speeds, leading to inefficiency. Heat Generation: Tends to generate significant heat during operation, which can necessitate additional cooling mechanisms.
2. **Torque and Speed Limitations :** Loses torque as the speed increases, which can limit its use in high-speed applications. Generally not suitable for applications requiring very high speeds.
3. **Resonance and Vibration :** Can exhibit resonant frequencies that cause vibrations, potentially affecting performance and requiring dampening techniques. Often noisier than other motor types due to the stepping action.
4. **Control Complexity :** Achieving smoother motion and higher resolution through microstepping can complicate control systems. May require sophisticated driving circuits for optimal performance.

**WIRING DIAGRAM:**

**A circuit board with wires connected to it

Description automatically generated**

**SAMPLE CODE:**

//Includes the Arduino Stepper Library

#include <Stepper.h>

// Defines the number of steps per rotation

const int stepsPerRevolution = 2038;

// Creates an instance of stepper class

// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence

Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);

void setup() {

// Nothing to do (Stepper Library sets pins as outputs)

}

void loop() {

// Rotate CW slowly at 5 RPM

myStepper.setSpeed(5);

myStepper.step(stepsPerRevolution);

delay(1000);

// Rotate CCW quickly at 10 RPM

myStepper.setSpeed(10);

myStepper.step(-stepsPerRevolution);

delay(1000);

}

**5) BRUSHLESS MOTOR**

1. **An electric motor with wires

   Description automatically generatedActuators**: Used in robotic joints, grippers, and manipulators.
2. **Wheels and Drivetrains**: Drive the wheels or tracks of mobile robots.
3. **Propulsion Systems:** Power propellers in aerial drones.
4. **Conveyor Systems**: Operate conveyor belts in industrial robots.
5. **Servo Motors**: Provide precise control in applications like CNC machines and robotic surgery.
6. **Gimbals and Stabilizers**: Stabilize cameras in drones and handheld devices.
7. **Linear Actuators**: Convert rotational motion into linear motion for precise linear movement.

**ADVANTAGES:**

1. Higher Efficiency: Brushless motors have better efficiency compared to brushed motors, resulting in less energy loss and heat generation.
2. Longer Lifespan: They have fewer mechanical parts (no brushes), which reduces wear and tear and increases durability.
3. Lower Maintenance: The absence of brushes means less frequent maintenance and replacements are required.
4. Higher Performance: Brushless motors provide higher torque-to-weight ratio and can achieve higher speeds.
5. Better Reliability: They offer more consistent performance and reliability under various operating conditions.
6. Reduced Noise: Brushless motors operate more quietly due to the absence of brushes.
7. Precise Control: They offer more precise control of speed and position, which is critical for applications requiring high accuracy.

**DISADVANTAGES:**

1. Higher Cost: Brushless motors are generally more expensive to produce and purchase compared to brushed motors.
2. Complexity: The control systems for brushless motors are more complex, requiring sophisticated electronics and software.
3. Initial Setup: They often need more intricate initial setup and tuning to achieve optimal performance.
4. Electronic Speed Controllers (ESCs): Brushless motors require ESCs, which add to the cost and complexity of the system.
5. Size and Weight: In some applications, the additional components for controlling brushless motors might add to the size and weight of the system.

**WIRING DIAGRAM:**

**A diagram of a circuit board

Description automatically generated**

**SAMPLE CODE:**

#include <Servo.h>

Servo ESC; *// create servo object to control the ESC*

int potValue; *// value from the analog pin*

void setup() {

*// Attach the ESC on pin 9*

ESC.attach(9,1000,2000); *// (pin, min pulse width, max pulse width in microseconds)*

}

void loop() {

potValue = analogRead(A0); *// reads the value of the potentiometer (value between 0 and 1023)*

potValue = map(potValue, 0, 1023, 0, 180); *// scale it to use it with the servo library (value between 0 and 180)*

ESC.write(potValue); *// Send the signal to the ESC*

}

**6) LINEAR ACTUATOR**

1. **A white and red device with wires

   Description automatically generatedGrippers and End Effectors:** Linear actuators are often used to control the opening and closing of robotic grippers or other end effectors that interact with objects.
2. **Lifting Mechanisms:** Robots designed to lift objects, such as warehouse robots or robotic arms, use linear actuators to raise and lower payloads.
3. **Conveyance Systems:** Linear actuators are used in robots that move objects along a conveyor belt or similar systems, ensuring precise placement and movement.
4. **Adjustable Bases:** For robots that need adjustable bases or platforms, linear actuators provide the necessary vertical or horizontal adjustments.

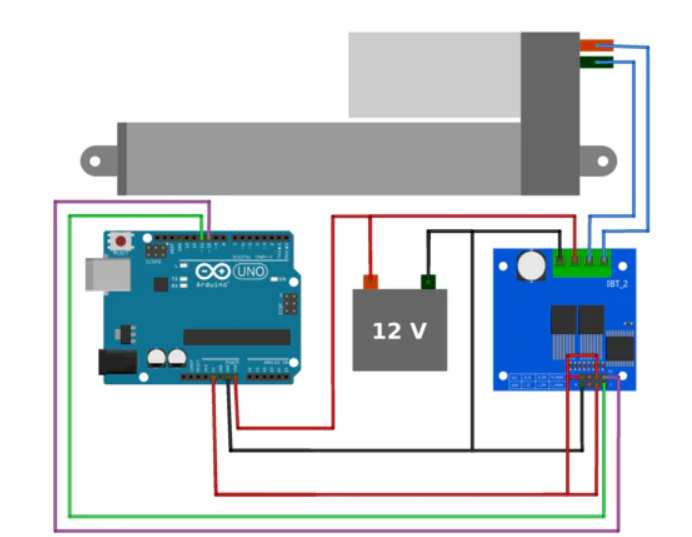
**ADVANTAGES:**

1. **Precision and Control :** Linear actuators provide precise control over linear movement, making them ideal for applications requiring accurate positioning. They offer consistent performance with high repeatability, which is crucial for repetitive tasks.
2. **Force and Load Capacity :** Capable of exerting significant force, making them suitable for lifting and moving heavy loads. Available in various sizes and capacities to match specific applicationrequirements.
3. **Versatility :** Used in diverse applications from industrial machinery to consumer electronics. Can be integrated into various systems and with different types of controllers.
4. **Efficiency and Maintenance :** Many linear actuators, especially electric ones, are energy efficient. Generally, they have fewer moving parts compared to other systems, resulting in lower maintenance needs.
5. **Simple Design and Installation :** Their compact design allows for easy integration into tight spaces. Relatively straightforward to install and set up.

**DISADVANTAGES:**

1. **Speed Limitations :** Typically, linear actuators are slower compared to rotary actuators or other motion systems.
2. **Cost :** High-quality linear actuators can be expensive. Custom or specialized actuators for specific tasks can significantly increase costs.
3. **Complexity and Size :** May require sophisticated control systems and electronics for precise operation. Some actuators can be bulky, limiting their use in compact applications.
4. **Environmental Sensitivity :** Performance can be affected by extreme temperatures or harsh environmental conditions, requiring additional protection or specific models designed for such environments.
5. **Limited Range of Motion :** Linear actuators are limited to a specific range of motion (stroke length), which might not be suitable for all applications.

**WIRING DIAGRAM:**

****

**SAMPLE CODE:**

byte Speed = 0; // Intialize Varaible for the speed of the motor (0-255);

int RPWM = 10; //connect Arduino pin 10 to IBT-2 pin RPWM

int LPWM = 11; //connect Arduino pin 11 to IBT-2 pin LPWM

void setup() {

pinMode(10, OUTPUT); // Configure pin 10 as an Output

pinMode(11, OUTPUT); // Configure pin 11 as an Output

}

void loop() {

// Extend Actuator at Full Speed

Speed = 255;

analogWrite(RPWM, 0);

analogWrite(LPWM, Speed);

delay(2000); // 2 Seconds

// Stop Actuator

analogWrite(RPWM, 0);

analogWrite(LPWM, 0);

delay(2000); // 2 Seconds

// Retract Actuator at Half Speed

Speed = 127;

analogWrite(RPWM, Speed);

analogWrite(LPWM, 0);

delay(2000); // 2 Seconds

// Stop Actuator

analogWrite(RPWM, 0);

analogWrite(LPWM, 0);

delay(2000); // 2 Seconds