Project Report

Pick and Place Robot

Members:-

Narendra Pal (Mechanical Engineering)

Ansari MohammadDaud (Mechanical Engineering)

Jyoti Pal (Electronics and Communication Engineering)

Satyamkumar Ramsundar Prajapati (Robotics and Automation Engineering)

Deepak Premshankar Yadav (Robotics and Automation Engineering)

Abrar Zafar (Robotics and Automation Engineering)

Objective:

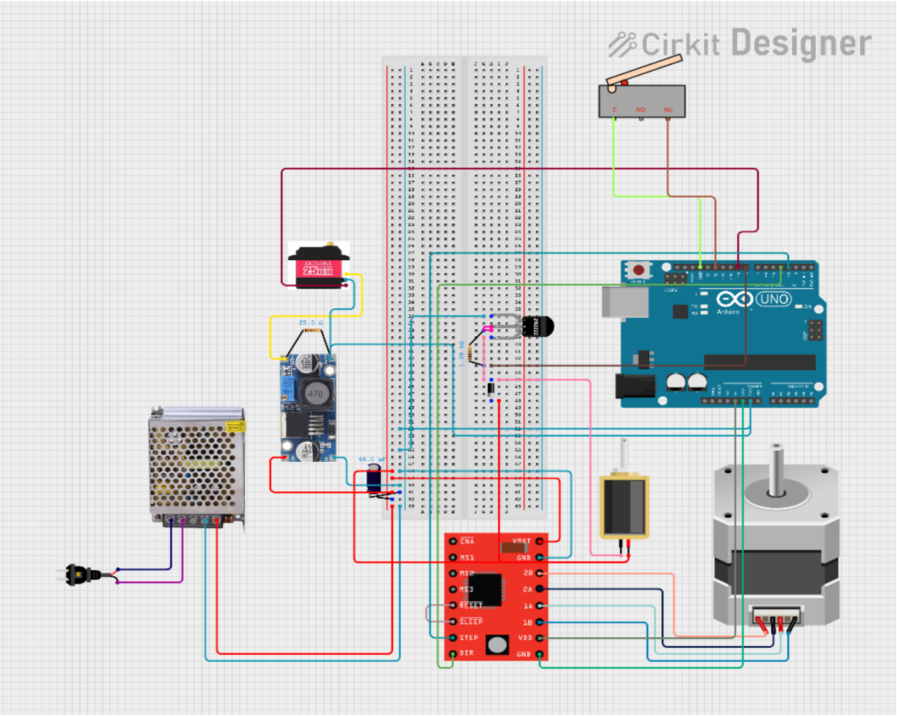
Design a Pick and Place robot with micro-electromagnet as a gripper to pick and place small metal cubes in a loop

Components Used:

Electronic Components:

- Arduino UNO
- LM2596 DC to DC Buck Converter
- A4988 Stepper Motor Driver
- 60Kg RDS5160 Servo Motor
- 42HD4045-02 Stepper Motor
- 12V Micro-Electromagnet
- 1N5399 Schottky Diode
- 2N222A NPN Transistor
- 1k and 25 Ohm Resistor
- Limit Switch
- 12V 5A SMPS
- 45 Micro-Farad Capacitor
- Jumper Wires
- 6 Pin JST PH Connector

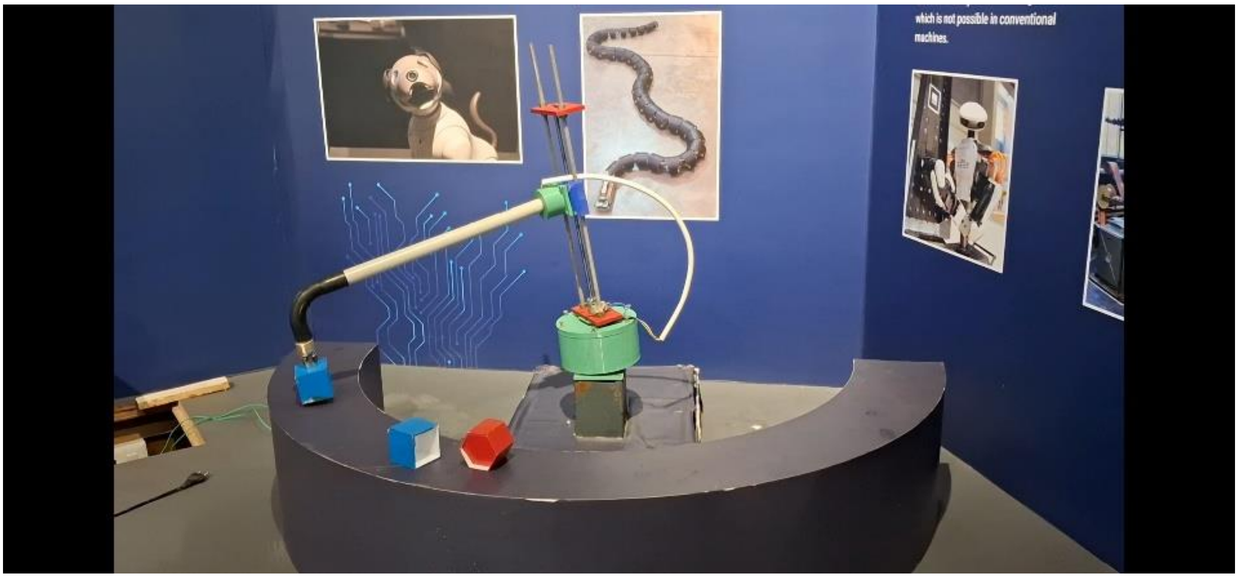
Circuit Diagram:-



Mechanical Components:

- M8 Lead Screw (x3)
- Flexible Shaft Connector
- M6 20mm Head Screw (x18)
- M3 20mm Countershink Screw (x4)
- M8 Hex-Nut (x8)
- M2 20mm Lead Screw (x5)
- M2 10mm Lead Screw (x2)
- Slider Bearing (inner D: 8mm) (x1)
- 3D Printed Parts
- PVC Pipe

Assembled Robot:-



Arduino Code:-

```
#include <Servo.h>


#define REFERENCE 55

#define WORKING_LENGTH 20

#define PICKUP_DELAY 2500

#define RELEASE_DELAY 2000


//define pins

const int electroMagnetPin = 12;

const int dirPin = 4;

const int stepPin = 3;

const int stepsPerRevolution = 200;

const int buttonPin = 8;

const int servoPin = 9;


bool start = true;


// creating base servo object

Servo robotBase;


// defining EndEffect class

class EndEffector {

private:

  int pin;


public:

  EndEffector(int pin)

    : pin(pin) {}


  void setup() {

    pinMode(pin, OUTPUT);
  }

  void ON() {
    digitalWrite(pin, HIGH);
  }

  void OFF() {
    digitalWrite(pin, LOW);
  }
};

//creting an EndEffector object
EndEffector Magnet(electroMagnetPin);

// function for moving servo to any Angle
void toAngle(int x) {
  int pos = robotBase.read();
  if (pos > x) {
    int count = 0;
    while (pos != x) {
      robotBase.write(pos);

      if (count < 5) {
        delay(20);
      } else {
        delay(15);
      }
      pos--;
      count++;
    }
  } else if (pos < x) {
    int count = 0;
    while (pos != x) {
      robotBase.write(pos);
      if (count < 5) {
        delay(20);
      } else {
        delay(15);
      }
      pos++;
      count++;
    }
  }
}

// function to move to the bottom most position of working length
void toBottom() {
  digitalWrite(dirPin, HIGH);
  for (int i = 0; i < WORKING_LENGTH; i++) {
    for (int x = 0; x < stepsPerRevolution; x++) {
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(500);
    }
  }
}

// function to move to the top most position of working length
void toTop() {
  digitalWrite(dirPin, LOW);
  for (int i = 0; i < WORKING_LENGTH; i++) {
    for (int x = 0; x < stepsPerRevolution; x++) {
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(500);
    }
  }
}

// Debounce function: returns true when stable press is detected
bool waitForDebouncedPress() {
  while (true) {
    if (digitalRead(buttonPin) == LOW) {
      delay(20);  // debounce delay
      if (digitalRead(buttonPin) == LOW) {
        // Wait for release to prevent repeated triggering
        while (digitalRead(buttonPin) == LOW)
          ;
        return true;
      }
    }
  }
}

// Homing function
void home() {
  if (digitalRead(buttonPin) != LOW) {
    int i = 0;
    digitalWrite(dirPin, HIGH);
    while (digitalRead(buttonPin) == HIGH) {
      for (int x = 0; x < stepsPerRevolution; x++) {
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(500);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(500);
      }
    }
  }
}

// Function for reaching to reference point
void reference() {
  digitalWrite(dirPin, LOW);
  for (int i = 0; i < REFERENCE; i++) {
    for (int x = 0; x < stepsPerRevolution; x++) {
      digitalWrite(stepPin, HIGH);
      delayMicroseconds(500);
      digitalWrite(stepPin, LOW);
      delayMicroseconds(500);
    }
  }
  delay(1000);
}




void setup() {
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);  // Use internal pull-up
  robotBase.attach(servoPin);
  robotBase.write(0);
  Magnet.setup();
  Serial.begin(9600);
```

```
}


void loop() {
  if(start){
    if (waitForDebouncedPress()) {
    delay(300);
    home();
    reference();
    start = false;
  }
  }
  else{
  toAngle(10);
  toBottom();
  Magnet.ON();
  delay(PICKUP_DELAY);
  toTop();
  toAngle(80);
  toBottom();
  Magnet.OFF();
  delay(RELEASE_DELAY);
  toTop();
  toAngle(40);
  toBottom();
  Magnet.ON();
  delay(PICKUP_DELAY);
  toTop();
  toAngle(120);
  toBottom();
  Magnet.OFF();
  delay(RELEASE_DELAY);
  toTop();
```

```
toAngle(90);

toBottom();

Magnet.ON();

delay(PICKUP_DELAY);

toTop();

toAngle(170);

toBottom();

Magnet.OFF();

delay(RELEASE_DELAY);

toTop();

toBottom();

Magnet.ON();

delay(PICKUP_DELAY);

toTop();

toAngle(90);

toBottom();

Magnet.OFF();

delay(RELEASE_DELAY);

toTop();

toAngle(120);

toBottom();

Magnet.ON();

delay(PICKUP_DELAY);

toTop();

toAngle(40);

toBottom();

Magnet.OFF();

delay(RELEASE_DELAY);

toTop();

toAngle(80);

toBottom();

Magnet.ON();
```

```
  delay(PICKUP_DELAY);
  toTop();
  toAngle(10);
  toBottom();
  Magnet.OFF();
  delay(RELEASE_DELAY);
  toTop();
  toAngle(0);
  while(digitalRead(buttonPin)==HIGH); // Remove this line if want to keep it in loop
}



}
```