Žiga Čampa

DC motor, clutch and brake control

Final seminar for the course Mechatronic Actuators

Mentors: Primož Podržaj, Tomaž Požrl

Ljubljana, 2025

Contents

List of Figures

1 Description

1.1 Use for product

As part of the final seminar in the subject of mechatronic actuators, I made a drive train control consisting of a DC motor, a clutch and a brake. The brake ensures that the drive axis remains in its position, even when the motor is not working. The clutch in this system ensures an easier start for the motor and protects it from excessive load. Due to easier operation and cheaper production, I will control the drive train for my own needs via four buttons with relays. For demonstration purposes at the faculty, the control was made via an Arduino Uno microcontroller and a power part consisting of relays, motor drivers, buttons and a potentiometer.

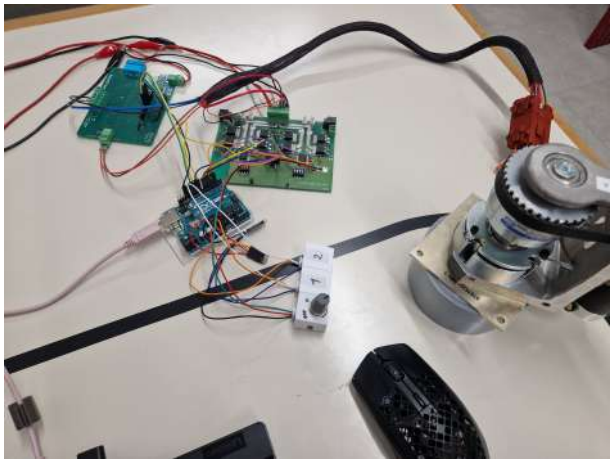

Figure 1: Drive train

1.2 Similar products

Similar compositions like this are used mainly in two cases:
-Electrical cranes
Electric lifts use asynchronous motors that are controlled in an open loop via buttons. They still use powerful AC brakes to hold the position..
-Robotic axis
Each joint in a 6-axis robot starts as a motor connected to a brake. Torque is then transferred through reductors to the individual joint. The two assemblies differ primarily in the type of motor (DC vs. asynchronous) and the type of control (open-loop vs. closed-loop servo control)

Figure 2: Robot drive Source: https://www.kuka.com/

2 Components description

2.1 DC motor

Manufacturer: BALDOR
Type: 509402 C
Voltage: 24 V DC
Current: 1.7 A
speed: 6.7 rpm
Reductor: 25.01:1
Magnet type: Permanent

The motor has 4 connections that can power 2 windings. The rear winding is used when we need fine position adjustment, and the front winding is used when we want a more responsive drive.

Figure 3: Motor BALDOR

2.2 Clutch

Manufacturer: Chain Tail
Type: ME10S4AC
Voltage: 24 V DC
El. power: 10 W
Max torque: 3.6 Nm

Figure 4: Clutch Chain Tail Source: https://www.chaintail.com/

2.3 Brake

Manufacturer: Chain Tail
Type: ALS0S2CV
Voltage: 24 V DC
El. power: 10 W
Max torque: 2 Nm

Figure 5: Brake Chain Tail Source: https://www.chaintail.com/

2.4 Microcontroler

Manufacturer: Arduino
Type: Uno Rev3 SMD
Microcontroler: ATmega328P
Voltage: 5V
Digital I/O pins: 14
Analog input pins: 6
PWM pins: 6
Clock: 16MHz

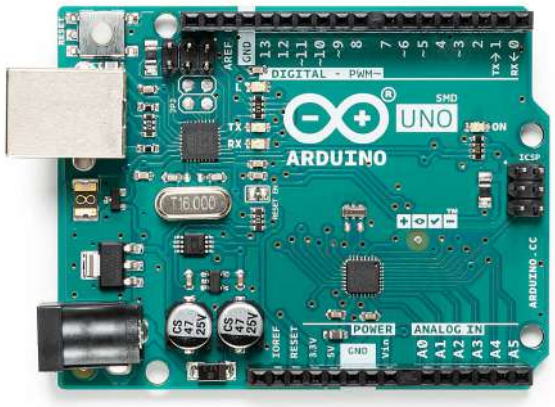

Figure 6: Microcontroler Arduino Uno Rev3 SMD Source: https://www.arduino.cc/

3 Assembly process

Because this drivetrain was given to me by a friend, the first step was to identify the specifications of its components. The motor in this drivetrain is quite unusual — it has four connectors, which turned out to be for two separate windings. The windings had different resistances and produced different speeds when connected to the same power source.

At the faculty lab, the lab assistant provided me with two motor drivers, an Arduino, and all the necessary wiring. The first H-bridge driver is used to power the motor. This is necessary because the maximum output voltage from the Arduino board is 5 V, while the motor operates at 24 V. The H-bridge driver is a board with two H-bridges, each built using eight transistors. These two H-bridges were used to power the stronger and weaker windings of the motor. The 24 V power was supplied by a lab power source.

The second motor driver is used to power the clutch and brake. This board is connected to both 24 V and 12 V power sources. It features one H-bridge, also built using transistors. To protect the board from voltage spikes generated by the actuators, diodes were installed.

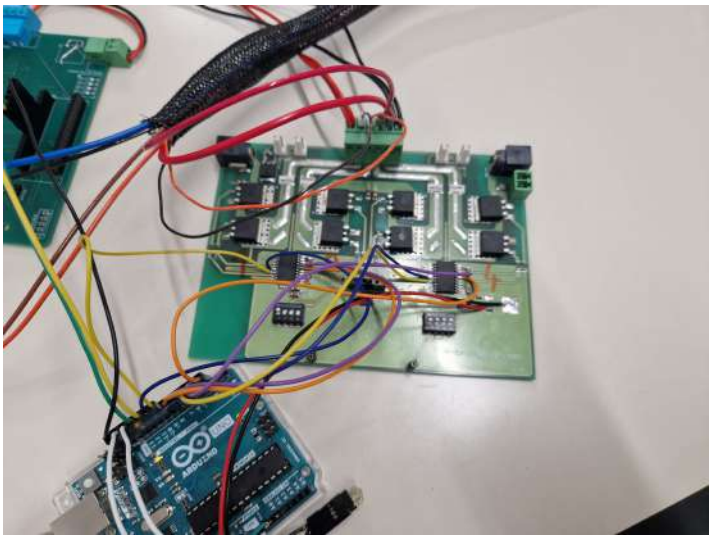

Figure 7: H-bridge driver

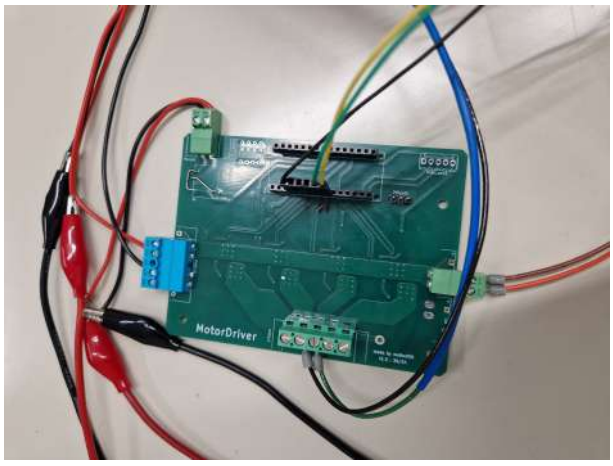

Figure 8: Motor driver

The final step in the assembly process was wiring the buttons and the potentiometer. All components are powered from the Arduino's 5 V output. The buttons are wired in a normally closed (NC) configuration.

4 Microcontroler code

The program controlling the drivetrain is based on a finite state machine. It consists of five states:

Start

MotorLeft1

MotorLeft2

MotorRight1

MotorRight2

The Start state is the initial state after boot-up. In this state, all actuators are powered off. From the Start state, the program can transition to either MotorLeft1 or MotorRight1, depending on whether button 1 or button 2 is pressed, respectively. A button press is detected when the Arduino input pin reads 0 V.

In the MotorLeft1 and MotorRight1 states, the low-power windings of the motor are activated, causing the drivetrain to spin in the specified direction for 250 milliseconds. During this phase, part of the drivetrain rotates without any mechanical load. After this period, the program either returns to the Start state or transitions to MotorLeft2 or MotorRight2, depending on the input conditions.

The MotorLeft2 and MotorRight2 states activate the high-power windings and deactivate the low-power windings. In addition, the clutch and brake are also engaged in these states. The controller remains in either MotorLeft2 or MotorRight2 as long as the corresponding button is held down. Once the button is released, the program transitions back to the Start state.

4.1 Arduino program

```
// Input declarations
#define button1 13
#define button2 12
#define potentiometer A0

// Output declarations
#define M1_HS1 8
#define M1_LS1 9
#define M1_HS2 7
#define M1_LS2 6
#define M2_HS1 4
#define M2_LS1 5
#define M2_HS2 2
#define M2_LS2 3
#define BRAKE 11
#define CLUTCH 10

// Variables
int potentiometerValue;
int pwmValue;
```

```
21
22 void setup() {
23   // Input settings
24   pinMode(button1, INPUT);
25   pinMode(button2, INPUT);
26
27   // Output settings
28   pinMode(CLUTCH, OUTPUT);
29   pinMode(BRAKE, OUTPUT);
30   pinMode(M1_HS1, OUTPUT);
31   pinMode(M1_LS1, OUTPUT);
32   pinMode(M1_HS2, OUTPUT);
33   pinMode(M1_LS2, OUTPUT);
34   pinMode(M2_HS1, OUTPUT);
35   pinMode(M2_LS1, OUTPUT);
36   pinMode(M2_HS2, OUTPUT);
37   pinMode(M2_LS2, OUTPUT);
38 }
39
40 // Naming states for use in switch-case
41 enum State {
42     Start,
43     MotorLeft1,
44     MotorLeft2,
45     MotorRight1,
46     MotorRight2,
47 };
48
49 // Function definitions
50
51 void State_Start() {
52   digitalWrite(CLUTCH, LOW);
53   digitalWrite(BRAKE, LOW);
54   digitalWrite(M2_HS1, LOW);
55   digitalWrite(M2_LS1, LOW);
56   digitalWrite(M2_HS2, LOW);
57   digitalWrite(M2_LS2, LOW);
58   digitalWrite(M1_HS1, LOW);
59   digitalWrite(M1_LS1, LOW);
60   digitalWrite(M1_HS2, LOW);
61   digitalWrite(M1_LS2, LOW);
62 }
63
64 void State_MotorLeft1() {
65   potentiometerValue = analogRead(potentiometer);
66   pwmValue = map(potentiometerValue, 0, 1023, 0, 255);
67
68   analogWrite(M1_HS1, 255);
69   analogWrite(M1_LS1, 0);
70   analogWrite(M1_HS2, 0);
71   analogWrite(M1_LS2, pwmValue);
72   delay(250);
73 }
74
75 void State_MotorRight1() {
76   potentiometerValue = analogRead(potentiometer);
77   pwmValue = map(potentiometerValue, 0, 1023, 0, 255);
78
79   analogWrite(M1_HS1, 0);
80   analogWrite(M1_LS1, pwmValue);
81   analogWrite(M1_HS2, 255);
```

```
  analogWrite(M1_LS2, 0);
  delay(250);
}

void State_MotorLeft2() {
  digitalWrite(CLUTCH, HIGH);
  digitalWrite(BRAKE, HIGH);

  potentiometerValue = analogRead(potentiometer);
  pwmValue = map(potentiometerValue, 0, 1023, 0, 255);

  analogWrite(M2_HS1, 0);
  analogWrite(M2_LS1, pwmValue);
  analogWrite(M2_HS2, 255);
  analogWrite(M2_LS2, 0);
  delay(200);
}

void State_MotorRight2() {
  digitalWrite(CLUTCH, HIGH);
  digitalWrite(BRAKE, HIGH);

  potentiometerValue = analogRead(potentiometer);
  pwmValue = map(potentiometerValue, 0, 1023, 0, 255);

  analogWrite(M2_HS1, 255);
  analogWrite(M2_LS1, 0);
  analogWrite(M2_HS2, 0);
  analogWrite(M2_LS2, pwmValue);
  delay(200);
}

// Start in the Start state
State currentState = Start;

void loop() {
    switch (currentState) {
        case Start:
            State_Start();
            if (digitalRead(button1) == LOW) {
              currentState = MotorLeft1;
            } else if (digitalRead(button2) == LOW) {
              currentState = MotorRight1;
            }
            break;

        case MotorLeft1:
            State_MotorLeft1();
            if (digitalRead(button1) == HIGH) {
              currentState = Start;
            } else {
              currentState = MotorLeft2;
            }
            break;

        case MotorLeft2:
            State_MotorLeft2();
            if (digitalRead(button1) == HIGH)
              currentState = Start;
            break;

```

```
143         case MotorRight1:
144             State_MotorRight1();
145             if (digitalRead(button2) == HIGH) {
146               currentState = Start;
147             } else {
148               currentState = MotorRight2;
149             }
150             break;
151
152         case MotorRight2:
153             State_MotorRight2();
154             if (digitalRead(button2) == HIGH)
155                 currentState = Start;
156             break;
157     }
158 }
```

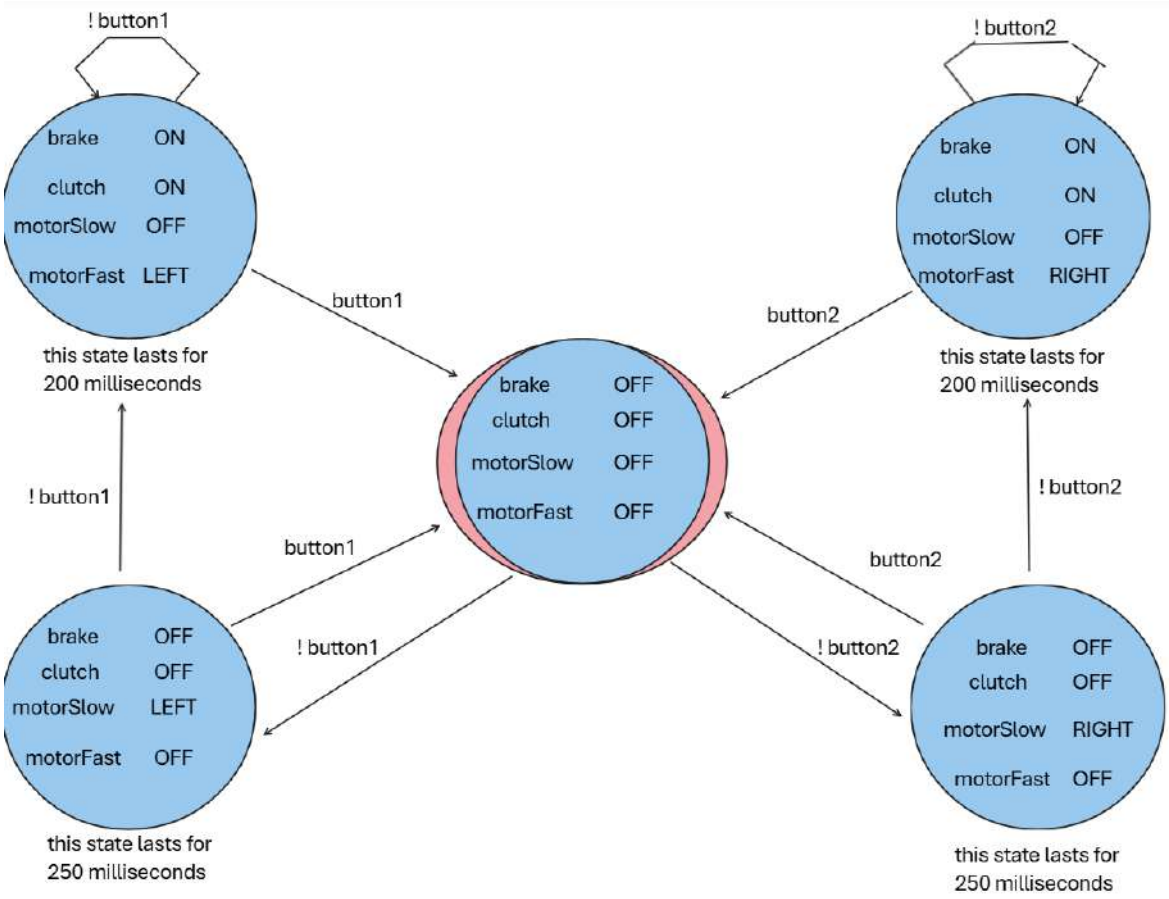

Figure 9: State machine