LABORATORY EXPERIMENT 3

ACTUATORS, DRIVES, AND CONTROL COMPONENTS

Rey Vincent Q. Conde
College of Engineering
Bachelor of Science in Electronics Engineering
Samar State University
rey.conde1206@gmail.com

***Abstract*—This experiment focuses on understanding the types of actuators and control components in robotics and how they interact with microcontrollers. Students will control various actuators through PWM and other control signals, and validate performance both on hardware and in simulation.**

***Index Terms*—Actuators, PWM, DC motor, Servo motor, Stepper motor, Robotics, Webots.**

I. Rationale

This experiment focuses on understanding the types of actuators and control components in robotics and how they interact with microcontrollers. Students will control various actuators through PWM and other control signals.

II. Objectives

- Interface and control at least three actuators (DC motor, servo motor, and stepper motor) using the Arduino, ensuring correct control of speed and direction.
- Use PWM signals to control the speed of a DC motor and observe the change in motor speed across a range from 50% to 100%.
- Simulate the movement of motors in Webots with a success rate of 95% for correct motor movement and speed.

III. Materials and Software

A. Materials

- Arduino Uno
- DC motors
- Servo motors
- Stepper motor
- L298N motor driver
- ULN2003 stepper motor driver
- Breadboard
- Jumper Wires

B. Software

- Arduino IDE
- Webots Simulation Environment

IV. Procedures

1) Set up Arduino to control DC motors, servo motors, and stepper motors.
2) Generate PWM signals to vary motor speed and direction.
3) Write the control code for motor management in Arduino IDE.
4) Simulate the actuators in Webots and check for correct performance.

V. Results

The stepper motor has been configured to rotate at a speed of approximately one full revolution every 15 seconds, which corresponds to a frequency of around 0.067 Hz. For the DC motor, three distinct speed modes have been established, utilizing analogWrite PWM values of 55, 125, and 255 to control its speed levels. Additionally, the servo motor has been programmed to continuously sweep from 0 degrees to 180 degrees during each iteration of the loop, creating a full range of motion back and forth.

VI. Discussion

The stepper motor exhibited precise yet relatively slow movement, completing a full rotation in about 15 seconds (corresponding to 0.067 Hz). This behavior aligns with the expected performance when driven by the ULN2003 driver at the selected stepping rate. The DC motor showed a linear response to changes in the PWM duty cycle: using analogWrite values of 55, 125, and 255, we observed distinct low, medium, and high speed modes. This confirmed that increasing the duty cycle proportionally increased both the motor's torque and rotational speed (RPM). Meanwhile, the servo motor consistently performed a full sweep from $0°$ to $180°$ with each loop iteration, demonstrating accurate position control through standard PWM signal modulation.

VII. Conclusion

This experiment successfully demonstrated the interfacing and control of a stepper motor, a DC motor, and a servo motor using PWM signals generated by an Arduino Uno. The DC motor's different speed modes showed a clear and direct correlation with changes in the PWM duty cycle. The stepper motor maintained a steady, low-frequency rotational motion, while the servo motor reliably performed full-range position sweeps from $0°$ to $180°$, confirming precise position control.

References

[1] STM32f103c6 Documentation, ”STM32f103c6 ”, https://www.st.com/en/microcontrollers-microprocessors/stm32f103c6.html.
[2] Cyberbotics Ltd., ”Webots User Guide”, 2024.
[3] ULN2003 stepper motor driver, https://www.ti.com/product/ULN2003A.

Appendix

Actuators

```
#include <Servo.h>

Servo myservo;
int pin1 = PA4;
int pin2 = PA5;
int pin3 = PA6;
int pin4 = PA7;

int dc_motor_pin1 = PB1;
int dc_motor_pin2 = PB0;

int pinslist[] = {pin1, pin2, pin3, pin4};
int speed_switch_pin = 3;

//stepper controller
void stepper(int ms_delay) {
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(ms_delay);

  digitalWrite(pin1, LOW);
  digitalWrite(pin2, HIGH);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, LOW);
  delay(ms_delay);

  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, HIGH);
  digitalWrite(pin4, LOW);
  delay(ms_delay);

  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  digitalWrite(pin3, LOW);
  digitalWrite(pin4, HIGH);
  delay(ms_delay);
}

// mode 1 = max speed; mode 2 = medium; mode 3
     = slow;
void dc_motor_driver(bool reverse, int
    speed_mode){

  switch(speed_mode){
    case 1:
      if(!reverse){
        digitalWrite(dc_motor_pin1, 0);
        analogWrite(dc_motor_pin2, 255);
      }else{
        digitalWrite(dc_motor_pin2, 0);
        analogWrite(dc_motor_pin1, 255);
      }
    break;
    case 2:
      if(!reverse){
        digitalWrite(dc_motor_pin1, 0);
        analogWrite(dc_motor_pin2, 125);
      }else{
        digitalWrite(dc_motor_pin2, 0);
        analogWrite(dc_motor_pin1, 125);
      }
    break;
    case 3:
      if(!reverse){
        digitalWrite(dc_motor_pin1, 0);
        analogWrite(dc_motor_pin2, 55);
      }else{
        digitalWrite(dc_motor_pin2, 0);
        analogWrite(dc_motor_pin1, 55);
      }
    break;
    default:
    break;

  }

}

void servo_turn(){

  myservo.write(0);
  delay(500);
  myservo.write(180);

}


void setup(){

  for(int i=0; i < sizeof(pinslist)/sizeof(
      pinslist[0]); i++){
    pinMode(pinslist[i], OUTPUT);
    digitalWrite(pinslist[i], 0);
  }
  pinMode(dc_motor_pin1, OUTPUT);
  pinMode(dc_motor_pin2, OUTPUT);
  myservo.attach(9);
  pinMode(speed_switch_pin, INPUT);

}

void loop(){

  if(digitalRead(speed_switch_pin)){

    stepper(300);
    dc_motor_driver(true, 1);
    servo_turn();

  }
  //delay(100);

}
```