ACTUATORS, DRIVES, AND CONTROL COMPONENTS

LABORATORY EXPERIMENT 3

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. Schematic Diagram

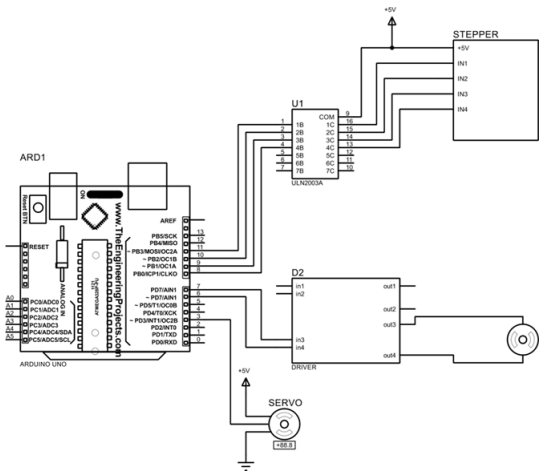

Fig. 1. Prototype Schematic

VI. Results

The stepper motor has been configured to rotate at a speed of approximately one full revolution every 15 seconds, equating to a frequency of about 0.067 Hz. This controlled rotational speed enables precise, timed movement suitable for tasks requiring consistent angular positioning or slow mechanical actuation.

In parallel, the DC motor has been set up with three distinct speed modes, managed through PWM control using analogWrite values of 55, 130, and 255. These values correspond to low, medium, and maximum speed levels, allowing for

flexible adjustment depending on the desired performance or task conditions.

Additionally, the servo motor has been programmed to perform a continuous sweeping motion, moving back and forth from 0 degrees to 180 degrees during each iteration of the loop. This configuration provides a full range of motion, which is useful for simulating scanning, directional control, or interactive mechanisms in robotic applications.

VII. Prototype Development

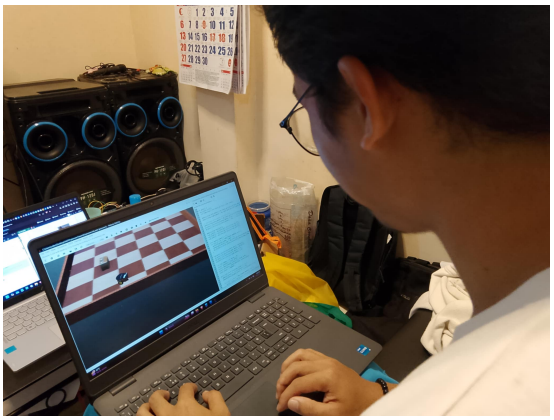

Fig. 2. Webots Simulation

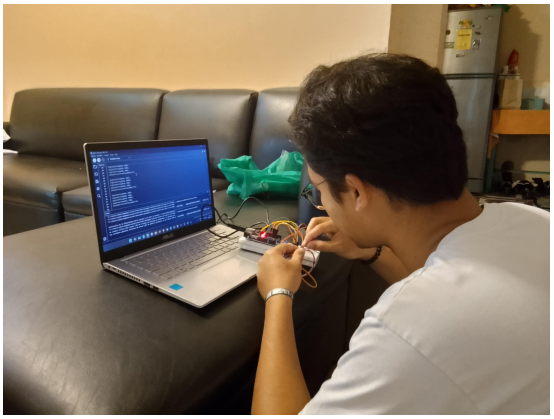

Fig. 3. Speed Control, Stepper, Servo Motor Testing

VIII. 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, 130, 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.

IX. 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. Each motor was effectively driven by the microcontroller, showcasing fundamental techniques in motor control within embedded systems.

The DC motor exhibited distinct speed variations in response to changes in the PWM duty cycle, clearly illustrating how pulse width modulation can be used to regulate motor speed. This provided a hands-on understanding of how analogWrite values correspond to different performance levels in real-time operation.

Meanwhile, the stepper motor maintained a consistent, low-frequency rotational motion, ideal for applications requiring precise step-based movement. The servo motor also operated as expected, reliably performing full-range position sweeps from 0° to 180°, confirming accurate and responsive position control. Together, these results underscored the versatility and effectiveness of PWM-based motor control in basic robotics projects.

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);

}
```