**Self Learning for MIME Capstone Design**

**April 19th, 2026**
**Team C603**
**AIAA Student Competition Rocket Team (SCRT) - Roll Control**

# Arduino PWM Control for Servo Actuation

**Oliver**
**Braitsch**

School of Mechanical, Industrial, and Manufacturing Engineering
(MIME)

Oregon State University

# 1.0 Introduction and Skill Selection

For the self-learning assignment, I chose to learn about Arduino based servo control using Pulse Width Modulation (PWM). I chose this skill because my capstone project involves a servo-actuated system for roll control, and understanding how servos are controlled is important for system integration, even from a mechanical design perspective.

This skill develops several key attributes of engineering design, including:

- Systems integration between mechanical and electrical components
- Understanding of actuator behavior and control inputs
- Ability to interpret embedded system logic
- Improved design decision-making based on actuator constraints

To learn this skill, I completed approximately 4-6 hours of self-directed training through LinkedIn Learning courses and supplementary online resources. Specifically, I completed the following courses:

- Learning Arduino: Foundations
- Learning Arduino: Interfacing with Analog Devices
- Learning Arduino: Interfacing with Hardware

These courses provided a good introduction to Arduino systems, including how microcontrollers interact with external components, how signals are generated and interpreted, and how hardware interfaces with the embedded code. In addition to these courses, I reviewed Arduino documentation and instructional youtube videos to reinforce key concepts related to PWM and servo control.

PWM works by sending a periodic signal where the pulse width determines the angular position of the servo. For most standard servos, pulse widths corresponding to approximately 1 ms to 2 ms are mapped to angular positions between 0 and 180 degrees. Understanding this relationship is critical for designing systems that require both precise and repeatable motion. As evidence of my self learning training, I have included LinkedIn Learning completion certificates for the courses listed above in Figure 1.


**Figure 1.** LinkedIn Learning course completion certificates for Arduino training modules

## 2.0 Application of Skill and Results

To further my learning, I applied the concepts I learned by developing and testing a simulated closed loop control system using Tinkercadan, an online Arduino environment. This allowed me to implement control logic in a way that reflects how a real  system would operate.

In this simulation, a potentiometer was used to represent roll error input, analogous to data that would be obtained from our flight sensors such as an IMU. The Arduino processes this input and mapped it to a corresponding trim tab deflection, which is then executed by the servo motor. This creates a simplified representation of a control loop consisting of a sensor, controller, and actuator. The simulation setup, Arduino code, and corresponding output are shown in Figures 2, 3, and 4, respectively.

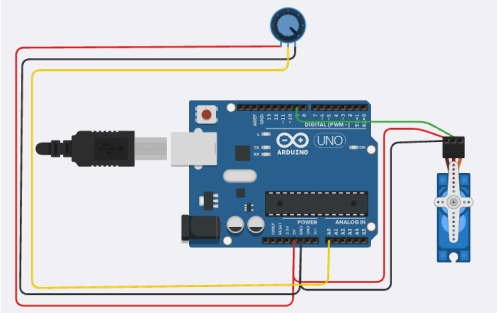

**Figure 2.** Tinkercad simulation setup including Arduino, potentiometer input, and servo actuator.

```
#include <Servo.h>

Servo trimTabServo;

int potPin = A0;
int rollError;
int trimCommand;
int servoAngle;

// Store previous values for change detection
int prevTrimCommand = 999;   // initialize to impossible value

void setup() {
  trimTabServo.attach(9);
  Serial.begin(9600);
}

void loop() {
  rollError = analogRead(potPin);

  // Convert roll input to percentage (-100% to +100%)
  float rollPercent = map(rollError, 0, 1023, -100, 100);

  // Map to trim tab deflection (-15° to +15°)
  trimCommand = map(rollError, 0, 1023, -15, 15);

  // Deadband (ignore small disturbances)
  if (abs(trimCommand) < 2) {
    trimCommand = 0;
  }

  // Convert to servo angle (centered at 90°)
  servoAngle = trimCommand + 90;

  trimTabServo.write(servoAngle);

  // Only print when trim command changes
  if (trimCommand != prevTrimCommand) {
    Serial.print("Roll Error (%): ");
    Serial.print(rollPercent);
    Serial.print(" | Trim Deflection (deg): ");
    Serial.print(trimCommand);
    Serial.print(" | Servo Angle: ");
    Serial.println(servoAngle);

    prevTrimCommand = trimCommand;
  }

  delay(50);
}
```

**Figure 3.** Arduino code implementing PWM-based closed-loop control of trim tab deflection.

```
Roll Error (%): 100.00 | Trim Deflection (deg): 15 | Servo Angle: 105
Roll Error (%): 23.00 | Trim Deflection (deg): 3 | Servo Angle: 93
Roll Error (%): 4.00 | Trim Deflection (deg): 0 | Servo Angle: 90
Roll Error (%): -100.00 | Trim Deflection (deg): -15 | Servo Angle: 75
```

**Figure 4.** Serial monitor output displaying roll error percentage and corresponding trim tab deflection.

This implementation provided several important insights into system behavior:

- Closed-Loop Control Representation:
    - The system mimics a real control loop where sensor input (roll error) is processed into a corrective actuator response (trim tab deflection).
- Normalized Input Interpretation:
    - Representing roll error as a percentage (-100% to +100%) improves interpretability and aligns with how real sensor data is processed in aerospace control systems.
- Efficient Data Logging:
    - Output is printed only when a change in trim command is detected, demonstrating event based data handling.
- Controlled Deflection Range:
    - Limiting trim tab motion to ±15 degrees ensures realistic actuator behavior within our mechanical limits.
- System Integration:
    - This simulation improves understanding of how mechanical components (trim tabs) and control systems (embedded logic) interact.

Although no physical prototype was built,  this simulation based model supports the design process by reducing uncertainty and improving confidence in the actuation concept. It also provides a foundation for future implementation of sensor feedback and real time control in a physical system. This demonstrates how embedded control concepts can be integrated into mechanical system design, reinforcing the importance of interdisciplinary engineering in aerospace systems.

## References


[1] Arduino, “Servo Library Documentation,” 2026. [Online]. Available: https://www.arduino.cc/reference/en/libraries/servo/

[2] LinkedIn Learning, “Learning Arduino: Foundations,” 2023.

[3] LinkedIn Learning, “Learning Arduino: Interfacing with Analog Devices,” 2024.

[4] LinkedIn Learning, “Learning Arduino: Interfacing with Hardware,” 2024.

[5] Afrotechmods, “What is PWM? Pulse Width Modulation tutorial!,” YouTube, 2008. [Online]. Available: https://www.youtube.com/watch?v=YmPziPfaByw

[6] The Engineering Mindset, “How to Control a Servo With an Arduino,” YouTube, 2022. [Online]. Available: https://www.youtube.com/watch?v=QbgTl6VSA9Y

[7] P. McWhorter, “Arduino Tutorial 33: Understanding How to Control Servos with a Joystick,” YouTube, 2019. [Online]. Available: https://www.youtube.com/watch?v=KCdoRDDLWgo