FOSSEE Winter Internship Report

On

Development and Control of Multi-DOF Robotic Systems

Submitted by

Shivani Sarangi

1st Year B.Tech Student, Department of Electronics and Communication Engineering

SRM Institute of Science and Technology, Kattankulathur

Under the Guidance of

Prof. Jayendran Venkateswaran

Department of IEOR

Indian Institute of Technology Bombay

Mentors:
FOSSEE Team

February 10, 2026

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and guided me throughout the course of my internship and the successful completion of this project. Their encouragement and guidance were instrumental in shaping my learning experience.

I express my sincere thanks to **Prof. Jayendran Venkateswaran**, IEOR Department, IIT Bombay, for overseeing the technical aspects of the work and for his continuous guidance and encouragement.

I am also grateful to the mentors from the department, including the M.Tech students, for their valuable technical support and constructive feedback throughout the project.

I extend my gratitude to **Prof. Kannan M. Moudgalya**, Principal Investigator of the FOSSEE project, Department of Chemical Engineering, IIT Bombay, for enabling this internship opportunity.

I am especially thankful to my FOSSEE guides, **Prof. Sumanto Kar** and **Mr. Varad Patil**, for coordinating all official and administrative aspects of the internship and for ensuring smooth communication during its course.

I also thank the FOSSEE Managers **Ms. Usha Viswanathan** and **Ms. Vineeta Parmar** and their entire team for their consistent support.

I gratefully acknowledge the support of the National Mission on Education through Information and Communication Technology (ICT), Ministry of Education (MoE), Government of India, for facilitating this project.

Finally, I express my sincere gratitude to my college, the Department of Electronics and Communication Engineering, and the Principal for their continued support throughout my studies.

Contents

1 Introduction **4**
- 1.1 National Mission on Education through ICT 4
  - 1.1.1 ICT Initiatives of MoE 4
- 1.2 FOSSEE Project 5
- 1.3 eSim Software 5

2 Internship Task 1: 6-Potentiometer Shield with Teach and Play **6**
- 2.1 Problem Statement 6
- 2.2 Tasks Done 6
  - 2.2.1 Study and Requirement Understanding 6
  - 2.2.2 System Design and Planning 7
  - 2.2.3 Sensor Interfacing and Input Handling 7
  - 2.2.4 Record and Replay Logic 7
- 2.3 Code Explanation 8
- 2.4 Learning Outcomes 8

3 Internship Task 2: Wii Nunchuck I2C Control Interface **9**
- 3.1 Introduction 9
- 3.2 High-Level System Overview 9
- 3.3 Hardware Architecture 9
- 3.4 Data Protocol and Decoding 11
- 3.5 Control Logic Implementation 11
  - 3.5.1 Joystick Mapping 11
  - 3.5.2 Accelerometer Mapping 11
- 3.6 Testing and Calibration 12
- 3.7 Conclusion 12

4 Internship Task 3: Single Potentiometer Multiplexed Control **13**
- 4.1 Overview 13
- 4.2 System Architecture 13

4.3 State Machine Logic 13
4.4 Key Engineering Insight 15

5 Internship Task 4: IR-Sensor Based Platform Automation 16
5.1 Introduction 16
5.2 System Overview 16
5.3 Hardware Configuration 17
5.4 Automated Logic Workflow 17
5.5 Sensor Verification Logic 18
5.6 Conclusion 18

6 Learning Outcomes 19

7 Conclusion 20

Chapter 1

Introduction

1.1 National Mission on Education through ICT

The National Mission on Education through ICT (NMEICT) is a scheme under the Department of Higher Education, Ministry of Education, Government of India. It aims to leverage the potential of ICT to enhance teaching and learning in Higher Education Institutions in an anytime-anywhere mode. The mission aligns with the three cardinal principles of the Education Policy: access, equity, and quality.

NMEICT seeks to bridge the digital divide by empowering learners and teachers in urban and rural areas, fostering inclusivity in the knowledge economy. Key focus areas include the development of e-learning pedagogies, virtual laboratories, and online testing platforms.

1.1.1 ICT Initiatives of MoE

The Ministry of Education (MoE) has launched several ICT initiatives aimed at students, researchers, and institutions. The table below summarizes the key details:

Table 1.1: Summary of ICT Initiatives by the Ministry of Education

No.	Resource	For Students/Researchers	For Institutions
1	SWAYAM	Earn credit via online courses	Develop and host courses; accept credits
2	National Digital Library	Access e-content in multiple disciplines	List e-content; form NDL Clubs
3	e-PG Pathshala	Access free books and e-content	Host e-books
4	Shodhganga	Access Indian research theses	List institutional theses
5	e-Yantra	Hands-on embedded systems training	Create e-Yantra labs with IIT Bombay
6	FOSSEE	Volunteer for open-source software	Run labs with open-source software
7	Virtual Labs	Perform online experiments	Develop curriculum-based experiments
8	SAMARTH ERP	Manage student lifecycle digitally	Enable institutional e-governance

1.2 FOSSEE Project

The FOSSEE (Free/Libre and Open Source Software for Education) project promotes the use of FLOSS tools in academia and research. It is part of the National Mission on Education through Information and Communication Technology (NMEICT), Ministry of Education (MoE), Government of India. Key activities include Textbook Companions, Lab Migration, and specialized activities to promote niche software tools.

1.3 eSim Software

eSim is an open-source EDA tool for circuit design, simulation, analysis, and PCB design. It is an integrated tool built using KiCad, Ngspice, and Python. It allows users to create circuit schematics, perform simulations, and design PCB layouts, making it an essential tool for electronics engineering.

Chapter 2

Internship Task 1: 6-Potentiometer Shield with Teach and Play

2.1 Problem Statement

In the field of robotics, calibrating multi-degree-of-freedom (DOF) arms often requires complex inverse kinematics or expensive teleoperation rigs. This project aimed to develop a low-cost, intuitive "Teach and Play" system where a human operator could manually guide the robot using a physical replica controller, record the movements, and have the robot autonomously replay the sequence with high fidelity.

2.2 Tasks Done

2.2.1 Study and Requirement Understanding

- Studied the working principle of servo motors and PWM signal generation.
- Identified the need for a custom hardware interface to manage 6 analog inputs simultaneously without signal noise.
- Analyzed the memory constraints of the Arduino microcontroller for storing large arrays of position data.

2.2.2 System Design and Planning

The system was designed around a Master-Slave architecture. The "Master" controller consists of a custom-designed shield housing six 10kΩ potentiometers, each corresponding to a specific joint of the robotic arm (Base, Shoulder, Elbow, Wrist Pitch, Wrist Roll, Gripper). The "Slave" is the robotic arm itself, actuated by MG996R high-torque servos.

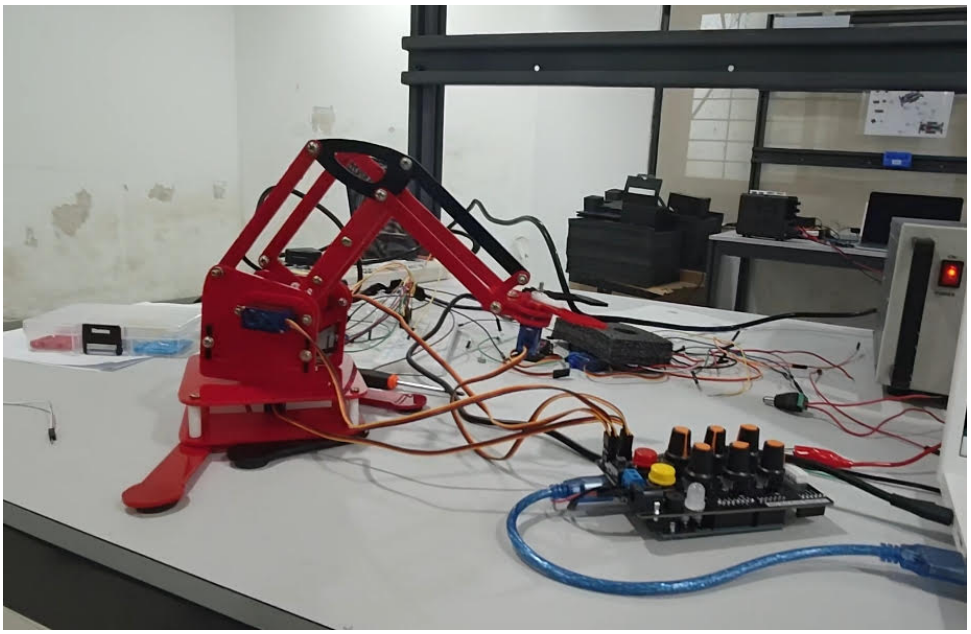

Figure 2.1: System architecture showing the Master (Potentiometer Shield) and Slave (Robotic Arm) configuration.

2.2.3 Sensor Interfacing and Input Handling

Each potentiometer forms a voltage divider circuit. The analog voltage (0-5V) is read by the Arduino's 10-bit ADC, returning a value between 0 and 1023. A mapping function was implemented to convert this range into the 0-180 degree range required by the servo library.

2.2.4 Record and Replay Logic

To implement the "Teach" functionality, a dynamic array data structure was used. Upon pressing a "Record" button, the system captures the instantaneous state of all six servos and appends it to the memory buffer. For the "Play" functionality, the system iterates through this buffer, writing the stored angles to the servos with a user-defined delay to ensure smooth mechanical transition and prevent current spikes.

2.3 Code Explanation

The firmware is written in Embedded C++. It utilizes the standard Servo library to manage the PWM pulses. The main loop checks the state of the mode-select switch. If in 'Teach' mode, it continuously reads analog ports A0 through A5. These values are mapped using the standard map function. If the 'Record' button is high, these mapped values are stored in a multi-dimensional array. In 'Play' mode, a for-loop iterates through the saved index, executing the servo write commands sequentially.

Listing 2.1: Teach and Play Logic for Servo Recording

```
#include <Servo.h>

// Arrays to store recorded positions
int recorded_positions[100][6];
int record_index = 0;

void loop() {
  if (digitalRead(RECORD_BUTTON) == HIGH) {
    // Capture state of all 6 servos
    for(int i=0; i<6; i++) {
        recorded_positions[record_index][i] = analogRead(A0 + i);
    }
    record_index++;
    delay(100); // Debounce delay
  }
}
```

2.4 Learning Outcomes

- Gained deep understanding of ADC resolution and signal mapping.
- Learned to manage dynamic memory allocation for storing gesture sequences.
- Developed skills in designing custom PCB shields using KiCad (FLOSS) to reduce wiring clutter.
- Understood the importance of power isolation when driving multiple high-torque motors.

Chapter 3

Internship Task 2: Wii Nunchuck I2C Control Interface

3.1 Introduction

This task involved integrating a consumer-grade game controller, the Wii Nunchuck, to control the robotic arm. Unlike simple potentiometers, the Nunchuck uses the I2C communication protocol and contains a 3-axis accelerometer and a 2-axis joystick. This project aimed to decode these digital signals to create a "Natural Mapping" interface, where tilting the controller physically tilts the robot.

3.2 High-Level System Overview

The architecture consists of three layers:

1. **Input Layer:** The Wii Nunchuck transmitting data packets via I2C.
2. **Processing Layer:** An Arduino decoding the 6-byte payload.
3. **Actuation Layer:** The robotic arm responding to calculated Pitch and Roll values.

3.3 Hardware Architecture

The Wii Nunchuck communicates over the I2C bus (Inter-Integrated Circuit). It acts as a slave device with the address 0x52. The connection

requires two lines: SDA (Data) and SCL (Clock), connected to the Arduino's analog pins A4 and A5 respectively.

Figure 3.1: Wiring diagram connecting the Wii Nunchuck via I2C (SDA/SCL) to the Arduino.

3.4 Data Protocol and Decoding

The Nunchuck sends data in 6-byte chunks. The structure of the data packet is as follows:

- **Byte 0:** Joystick X-axis value.
- **Byte 1:** Joystick Y-axis value.
- **Byte 2-4:** Accelerometer X, Y, and Z axis values (10-bit data split across bytes).
- **Byte 5:** Button states (C and Z buttons).

The raw data is encrypted by default. An initialization handshake sequence was implemented to disable encryption, allowing the microcontroller to read raw sensor data directly.

3.5 Control Logic Implementation

3.5.1 Joystick Mapping

The Joystick X values (0-255) control the Base rotation of the robot. A dead-zone logic was implemented (values 120-136) to prevent drift when the stick is centered.

3.5.2 Accelerometer Mapping

The accelerometer data was passed through a complementary filter to reduce jitter. The calculated tilt angle was then mapped to the Wrist Pitch servo. This allows the operator to control the robot's orientation simply by rotating their wrist.

Listing 3.1: I2C Decoding and Joystick Mapping

```
#include <Wire.h>

void loop() {
  Wire.requestFrom(0x52, 6); // Request 6 bytes from Nunchuck

  while (Wire.available()) {
    // Byte 0 is Joystick X, Byte 1 is Joystick Y
    int joy_x = Wire.read();
    int joy_y = Wire.read();
```

```
    // Map Joystick X (0-255) to Base Servo (0-180)
    if (joy_x > 136 || joy_x < 120) { // Dead-zone check
        int servo_angle = map(joy_x, 0, 255, 0, 180);
        base_servo.write(servo_angle);
    }
  }
}
```

3.6 Testing and Calibration

Calibration was performed by reading the raw values when the controller was placed on a flat surface. These offsets were hard-coded into the firmware to ensure accurate zero-positioning. The system was stress-tested for I2C bus hang-ups, and a timeout reset function was added to recover from communication failures.

3.7 Conclusion

This task demonstrated the power of digital communication protocols over analog control. By using I2C, we reduced the wiring complexity from 6 wires to just 2, while gaining access to sophisticated sensor data like acceleration and digital button states.

Chapter 4

Internship Task 3: Single Potentiometer Multiplexed Control

4.1 Overview

While the 6-potentiometer shield provided full control, it was hardware-intensive. This task explored an optimization strategy: controlling a multi-DOF system using a single high-quality analog input. This "Multiplexed" approach mimics industrial CNC controls where a single jog-wheel controls multiple axes.

4.2 System Architecture

The system consists of:

- **Master Input:** A precision 10k potentiometer.
- **Axis Selector:** A momentary push-button or toggle switch.
- **Indicator:** An LED array or LCD display indicating the active axis.

4.3 State Machine Logic

A Finite State Machine (FSM) was implemented in the firmware.

- **State 0 (Idle):** Potentiometer input is ignored.

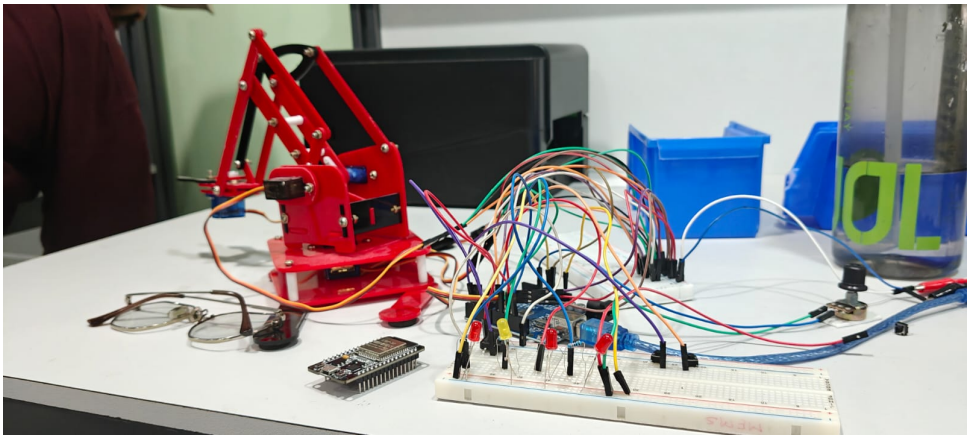

Figure 4.1: Multiplexed control architecture showing the single input controlling multiple axes via a selector switch.

- **State 1 (Base Control):** Potentiometer maps to Base Servo. All other servos hold position.
- **State 2 (Shoulder Control):** Potentiometer maps to Shoulder Servo.
- **State 3 (Gripper Control):** Potentiometer maps to Gripper Servo.

Pressing the selector button triggers a state transition. The critical engineering challenge was "Latching." When switching from Base to Shoulder, the potentiometer physical position might not match the Shoulder's current position. A "Soft Latch" algorithm was written to ignore input until the potentiometer crosses the current servo value, preventing dangerous jumps in movement.

Listing 4.1: Finite State Machine for Axis Selection

```
int current_state = 0; // 0=Idle, 1=Base, 2=Shoulder

void loop() {
  // Check if selector button is pressed
  if (digitalRead(BUTTON_PIN) == HIGH) {
    current_state++;
    if (current_state > 2) current_state = 0;
  }

  // Execute logic based on current state
  switch (current_state) {
    case 1: // Base Control
      base_servo.write(map(analogRead(POT_PIN), 0, 1023, 0, 180));
      break;
    case 2: // Shoulder Control
      shoulder_servo.write(map(analogRead(POT_PIN), 0, 1023, 0, 180));
```

```
      break;
  }
}
```

4.4 Key Engineering Insight

This task highlighted the trade-off between hardware complexity and software complexity. Reducing hardware (1 pot vs 6 pots) significantly increased the complexity of the code (State Machine + Latching Logic) but resulted in a much cheaper and more robust physical product.

Chapter 5

Internship Task 4: IR-Sensor Based Platform Automation

5.1 Introduction

Automation relies on environmental awareness. This project involved automating a model train platform where the train's behavior was dictated not by a user, but by its physical position. This mirrors industrial AGV (Automated Guided Vehicle) systems used in logistics.

5.2 System Overview

The system uses infrared (IR) proximity sensors to detect the presence of the train at specific stations.

- **Station A:** Equipped with IR Sensor 1 and a Red LED.
- **Station B:** Equipped with IR Sensor 2 and a Green LED.
- **Actuator:** L298N Motor Driver controlling the train's DC motor.

5.3 Hardware Configuration

Table 5.1: Hardware Pin Configuration

Component	**Pin Type**	**Arduino Pin**
IR Sensor 1 (Station A)	Digital Input	2
IR Sensor 2 (Station B)	Digital Input	3
Motor Enable (L298N)	PWM Output	9
Motor Input 1	Digital Output	4
Motor Input 2	Digital Output	5

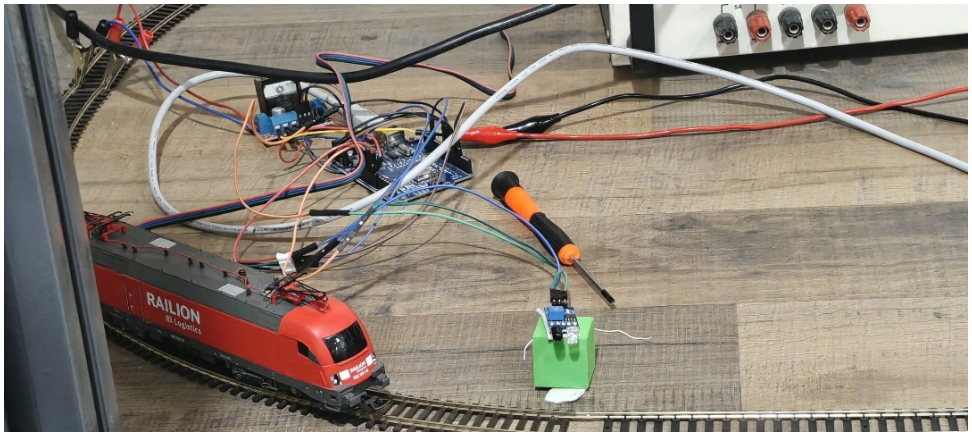

Figure 5.1: Circuit diagram showing IR sensors at Stations A and B connected to the L298N Motor Driver.

5.4 Automated Logic Workflow

The control logic follows a strict "Stop-Wait-Go" protocol:

1. **Detection:** When IR Sensor 1 goes LOW (Active), the system detects "Arrival at Station A."

2. **Stop:** The Motor Driver pins are set to LOW, cutting power to the track. The train coasts to a halt.

3. **Process:** A delay timer (simulating passenger loading) is initiated. The Station LED blinks.

4. **Departure:** After the timer expires, the Motor Driver is re-engaged, and the train accelerates towards Station B.

5.5 Sensor Verification Logic

To prevent false triggers (e.g., a hand passing over the track), a "Debounce" logic was verified. The sensor must remain active for at least 200ms for the signal to be considered a valid train arrival. This is a crucial concept in industrial automation to filter out transient noise.

Listing 5.1: Automated Stop-Wait-Go Logic

```
void loop() {
  // Check if train has arrived at Station A
  if (digitalRead(IR_SENSOR_A) == LOW) {
    delay(200); // Debounce check
    if (digitalRead(IR_SENSOR_A) == LOW) {
        stopTrain();
        digitalWrite(LED_A, HIGH);
        delay(5000); // Wait for 5 seconds
        startTrain();
    }
  }
}
```

5.6 Conclusion

This project bridged the gap between manual robotics and fully autonomous systems. It demonstrated how simple binary sensors (IR) can be combined with state-machine logic to create complex, reliable behaviors without human intervention.

Chapter 6

Learning Outcomes

- **Embedded Systems Mastery:** Gained extensive experience in programming Arduino and ESP32 microcontrollers, specifically in handling interrupts, timers, and PWM generation.
- **Protocol Proficiency:** Developed a deep understanding of communication protocols, specifically I2C for sensor integration and UART for debugging.
- **Open Source Hardware Design:** Learned to design and fabricate custom PCB shields using KiCad and eSim, adhering to FOSSEE's FLOSS principles.
- **Control Theory:** Applied kinematic concepts and state-machine logic to solve real-world actuation problems.
- **System Integration:** Understood the complexity of integrating diverse components (sensors, motors, controllers) into a unified, functional system.
- **Debugging Resilience:** Developed a systematic approach to troubleshooting, isolating issues between hardware connections and software logic errors.

Chapter 7

Conclusion

The FOSSEE Winter Internship provided a rigorous, hands-on exposure to industrial engineering concepts through the integration of software systems and embedded hardware. The internship progressed from foundational manual control systems, such as the 6-potentiometer robotic arm, to complex digital communication tasks involving I2C protocols and accelerometer mapping.

The exploration of diverse control strategies—from the "Teach and Play" memory arrays to the "State Machine" logic of the single-potentiometer controller—highlighted the various approaches to solving engineering problems. The final integration of sensor-based automation in the Model Train project reinforced the importance of environmental feedback in autonomous systems.

Overall, the internship strengthened technical skills in embedded C, Open Source PCB design (KiCad/eSim), and hardware-software co-design, while instilling a strong appreciation for pragmatic engineering decisions, fail-safe design, and industrial realism.

Bibliography

[1] Ministry of Education, Government of India, National Mission on Education through Information and Communication Technology (NME-ICT). Official Website: https://www.nmeict.ac.in

[2] FOSSEE Project, IIT Bombay, Free/Libre and Open Source Software for Education (FOSSEE). https://fossee.in

[3] eSim Project, FOSSEE. https://esim.fossee.in

[4] Arduino Documentation. https://docs.arduino.cc

[5] Nintendo Wii Nunchuck Communication Protocol, Wiibrew.org.

[6] KiCad EDA Documentation. https://docs.kicad.org