# Systems Engineering: Inverted Pendulum

Sofiya Flenova, Tara Kasayapanand, Thun Sahacharoen, Ted Tang

January 2026

# 1 Introduction

**Abstract**

This report documents the complete design, simulation and implementation of a mobile inverted pendulum system using Model-Based Systems Engineering (MBSE) methodologies. Following the Engineering V-model, we developed a self-contained robotic cart capable of autonomously balancing a pendulum in the upright position while performing dynamic manoeuvres. Key achievements include successful disturbance rejection, partial recovery from initial tilts and 2 meter sprint execution. The project will demonstrate rigorous application of systems engineering principles from requirements analysis through simulation, prototyping and validation, with comprehensive benchmarking of two control strategies.

# Contents

## 1.1 Mission Definition

The mission of this project is to design, develop and demonstrate a self-contained, mobile inverted pendulum system that autonomously balances a passive pendulum rod in an upright position while performing dynamic manoeuvrers. This involves creating a robust control system using Model-Based Systems Engineering methodologies, with a battery-powered, autonomous platform capable of rejecting external disturbances, recovering from significant initial tilts of 15° or greater and executing rapid, precise sprints across a 2-meter distance.

Our system must meet strict mechanical constraints, particularly minimal pivot friction ($\zeta < 0.01$), while quantitatively comparing at least two control strategies. Success is measured through three competitive evaluations: disturbance rejection (settling time after external forces), deep fall recovery (maximum recoverable starting angle) and sprint performance (time to complete 2 meters with $\pm 10$ cm stopping precision).

## 1.2 Our Approach: Stability-First Development

Our mission adopts a **stability-first development strategy**, prioritising robust upright balancing and disturbance rejection as foundational requirements before advancing to dynamic maneuvers. This approach is driven by multiple considerations: risk mitigation ensures the pendulum never drops during development, protecting hardware from damage; incremental progression creates a reliable platform for methodical performance gains; V-Model alignment verifies core requirements before adding complexity; and competition readiness ensures all evaluation metrics are met through test-driven iteration.

Our development philosophy follows four guiding principles: (1) establish reliable upright balancing as

the non-negotiable baseline, (2) add complexity only when previous functionality is verified, (3) validate all designs in simulation before physical implementation and (4) use data-driven decisions to guide control strategy selection.

### 1.3 Bill of Materials

| Component Name | Qty | Purpose |
|---|---|---|
| **Mechanical Bill** | | |
| Pololu 25D mm Metal Gearmotor Bracket Pair | 4 | Mounting motor to chassis |
| Pololu Aluminum Scooter Wheel Adapter for 4mm Shaft | 4 | Connecting motor to wheel |
| 12mm Hex Wheel Adapter for 4mm Shaft | 4 | Connecting motor to wheel |
| Scooter/Skate Wheel 70×25mm | 4 | Wheel |
| Ball bearing 6mm I.D, 19mm O.D | 1 | Ensure pendulum has lowest friction possible |
| Ball bearing 4mm I.D, 12mm O.D | 2 | Rotary mechanism for V-shaped holder |
| Bevel gear 4mm I.D, 30mm O.D | 1 | Rotary mechanism for V-shaped holder |
| Aluminium Corner Bracket | 2 | V-shaped holder component |
| Rack | 2 | V-shaped holder component |
| Wooden dowel 10mm D, 600mm length | 1 | Pendulum |
| Wooden dowel 6mm D, 42mm length | 1 | Perpendicular rod |
| 5mm thick Plywood | 1 | Base |
| 50g weight | 1 | Payload on the pendulum |
| M4 nut | 1 | Mounting payload onto pendulum |
| M4 screw | 1 | Mounting payload onto pendulum |
| M2.5 screw | 8 | Mounting MCU and elevated base to chassis |
| M2.5 nut | 8 | Mounting MCU and elevated base to chassis |
| M3 screw | 18 | Attaching motor to bracket and MCU to chassis |
| M3 nut | 18 | Attaching motor to bracket to chassis |
| **Electrical Bill** | | |
| 34:1 Metal Gearmotor 25Dx63L mm MP 12V with 48 CPR Encoder | 4 | Individual motor control |
| Motoron M3S550 Triple Motor Controller Shield | 1 | Motor shield for controlling 4 motors |
| Rotary Encoder Increment 1000CPR 6mm | 1 | For control and tracking of inverted pendulum |
| Arduino Uno R4 WiFi Board | 1 | Microcontroller |
| Button | 1 | Start/Stop Mechanism |

Table 1: Bill of Materials

## 2 Project Management and Planning

### 2.1 PM Plan

#### 2.1.1 Team Roles & Responsibilities

While all the team members will be partaking actively in every aspect of the project, the team will still be organised around specialised domains with Tara leading mechanical design, Ted managing control systems and simulation, Thun overseeing electronics and hardware and Sofiya acting as a project manager.

This distribution aligns individual expertise with project requirements while maintaining collaborative oversight across all technical areas. All members contribute collectively to weekly progress reviews, risk management, system integration and final presentation preparation.

### 2.1.2 Communication Plan

Our communication strategy balances structured weekly meetings with flexible daily coordination through WhatsApp for real-time updates. Formal two-hour weekly sessions review accomplishments, address challenges, allocate tasks and assess risks, with minutes documented in Google Drive. Daily WhatsApp communication handles quick status updates and urgent technical questions. Google Drive provides centralised document management with organised folders for project management, design documents, simulation code, CAD models, test data, reports and presentations. External communication includes weekly progress summaries to the instructor and scheduled consultations with teaching assistants and lab staff as needed. The weekly to-do lists are handled in our Teams channel, particularly on the Microsoft Planner App. All code developed is committed to our team's GitHub repository, ensuring it is immediately accessible and synchronised for every member.

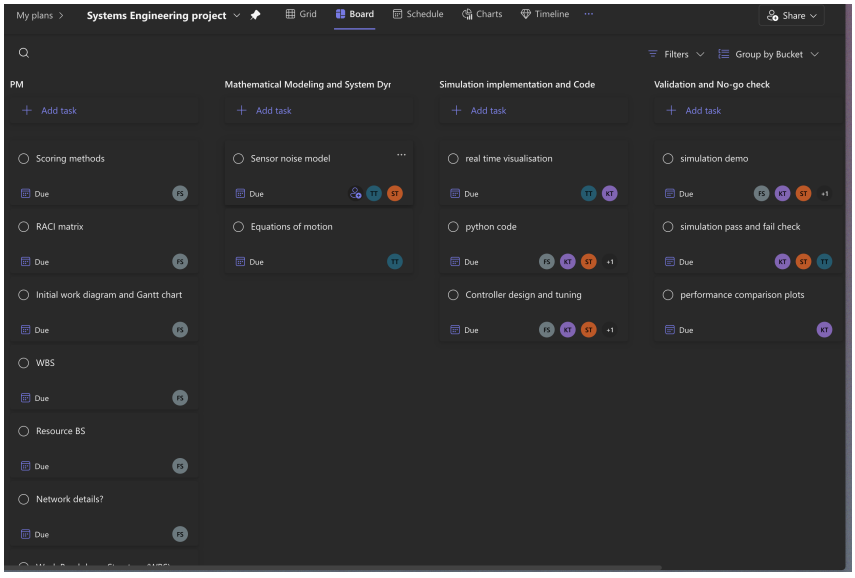


Figure 1: Work Breakdown Structure diagram (Created in Miro)

### 2.1.3 Tool Stack & Version Control Setup

Our technology stack uses Python for primary simulation, C++ for microcontroller firmware, SolidWorks for mechanical design, CATIA Magic for SysML modeling, KiCad for circuit design and LaTeX for final documentation. Version control is managed through GitHub with a structured branch system: `main` for stable code and feature-specific branches for new development. All changes require peer review before merging, with descriptive commit messages and regular daily commits during active development phases. Experimental data and simulation results are timestamped and linked to specific code versions

for traceability.

#### 2.1.4 Approach

We employ a hybrid project management approach that combines the structured framework of Waterfall with iterative, sprint-driven elements of Agile. This allows us to maintain rigorous requirements traceability and phase reviews (MCR, SRR, PDR, CDR, etc.) while also practising rapid prototyping, continuous testing and adaptivity.

### 2.2 Identification of Key Stakeholders

- **Dr. Sajad Saeedi** – Course Professor & Project Supervisor
- **Teaching Assistants:** Rehan Agrawal, Muhammad Maaz, Yusuf Adekola
- **Lab Manager:** Ollie Collier
- **Lab Technicians:** Alex Charitonidis, Matt Madden
- **Lab Support:** Ben Barwise, Isabella Barbaro
- **Team Members:** Tara Kasayapanand, Ted Tang, Thun Sahacharoen, Sofiya Flenova

| Stakeholder | Primary Interest | Communication |
|---|---|---|
| Dr. Saeedi | Educational outcomes, MBSE methodology | Weekly updates, formal submissions |
| TAs | Technical implementation, lab support | Daily lab interaction, technical queries |
| Lab Manager | Resource allocation, safety | Scheduled bookings, formal requests |
| Lab Technicians | Equipment operation, fabrication | As-needed technical consultations |
| Team Members | Project success, collaboration | Continuous internal coordination |

- **Academic Stakeholders:** Adhere to deadlines, demonstrate learning outcomes
- **Technical Support:** Book resources in advance, follow safety protocols
- **Internal Team:** Regular meetings, clear role definitions, transparent communication

## 2.3 Mindmap

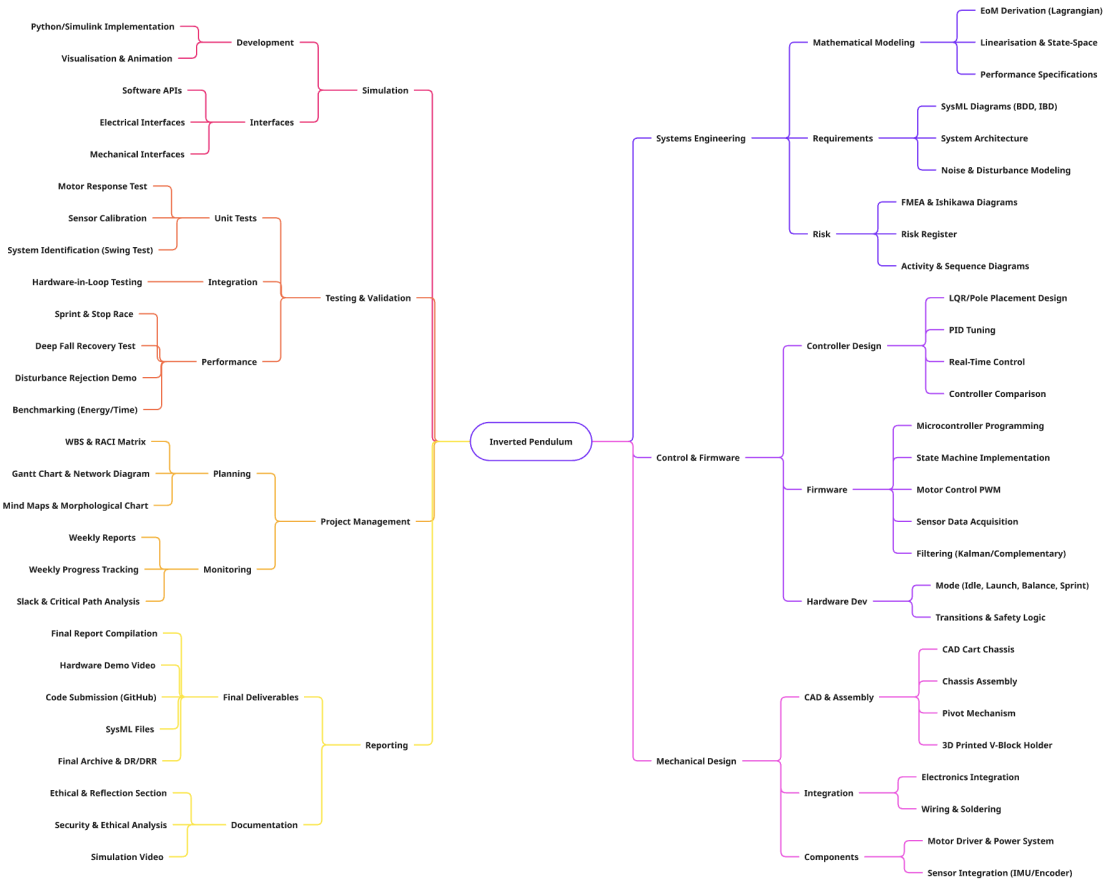


Figure 2: Project mindmap (Created in Miro)

## 2.4 Morphological Chart: System requirements

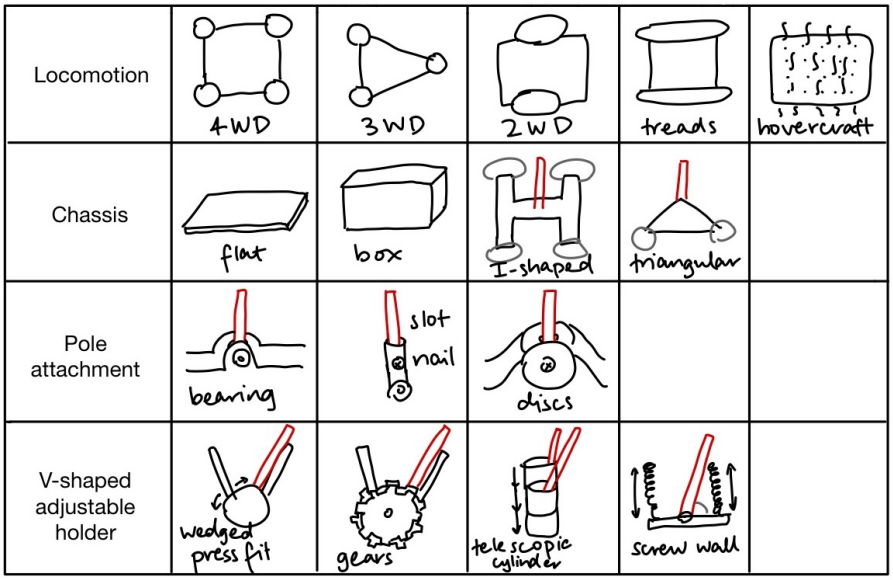


Figure 3: Morphological Chart for locomotion, chassis, pole attachment, v-shaped holder designs

## 2.5 Requirements diagram

Table 2: Inverted Pendulum System

| **Product Characteristics** | **Functional Requirements** | **Constraints** | **Performance Metrics** |
|---|---|---|---|
| **Reliable** | • Cart moves freely on flat surface<br>• Autonomous (battery-powered)<br>• Operates full demo without failure | • Self-contained (no tethers)<br>• Battery fits in chassis<br>• Survives multiple test cycles | • MTBF $>$ 50 test cycles<br>• Demo success rate $\geq$ 90% |
| **Safe** | • Upright balancing ($\theta = 0°$)<br>• Emergency stop capability<br>• No sharp edges/exposed wiring | • Tip mass: 50g (disk)<br>• Rod: cylindrical, Ø12mm<br>• Battery voltage within limits | • Zero pendulum drops<br>• Emergency stop $<$ 1s |
| **Fast** | • Pendulum rotates freely<br>• Quick acceleration for sprint<br>• Rapid disturbance response | • Damping $\zeta < 0.01$<br>• Motor torque-speed limits<br>• Microcontroller speed | • $P = \frac{0.01-\zeta}{0.01}$<br>• Sprint time (2m) [s]<br>• Update rate $\geq$ 100Hz<br>• Max acceleration [m/s$^2$] |
| **Responsive** | • V-fixture: adjustable angles<br>• Reacts to taps/shoves<br>• Smooth mode transitions | • 50g tip mass payload<br>• Cart weight $\leq$ motor capacity<br>• 3D printed holder durable | • Settling time after disturbance [s]<br>• Max recoverable angle $\theta_{\max}$ [°]<br>• Mode transition $<$ 2s |
| **Precise** | • Sprint to finish line within bounds<br>• Minimal oscillation balance<br>• Accurate position control | • Pole: 60–100 cm<br>• Encoder resolution limits<br>• Sensor noise affects precision | • Position accuracy 10 cm<br>• Steady-state error $< 1°$<br>• Tracking error $<$ 5 cm |
| **Accurate** | • Disturbance rejection<br>• Accurate angle measurement<br>• Consistent performance | • Minimal friction pivot<br>• IMU/encoder calibration | • Settling time $<$ 3s<br>• Overshoot $< 5°$<br>• Accuracy: angle $\pm 0.5°$, pos $\pm$1cm |
| **Energy Efficient** | • Optimise control effort | • Limited battery capacity<br>• Motor efficiency varies | • Battery life $>$ 30 min |
| **Robust** | • Handles sensor noise<br>• Works in different conditions<br>• Tolerates component variability | • Use lab-provided sensors/actuators | • Degradation $<$ 10% over 20 runs<br>• SNR $>$ 20dB |
| **Modular** | • Easy assembly/disassembly<br>• Replaceable components<br>• Configurable for tests | • Minimised settling time/overshoot | • Assembly $<$ 15 min<br>• Component swap $<$ 5 min<br>• Tools required <= 3 |

## 2.6 WBS

Work Breakdown Structure has been created in two different formats: as a diagram and as a table. The table format contains additional information about predecessors and resource allocation for each task. For convenience, name abbreviations have been utilised: SF-Sofiya Flenova, TK-Tara Kasayapanand, TT-Ted Tang, TS-Thun Sahacharoen.

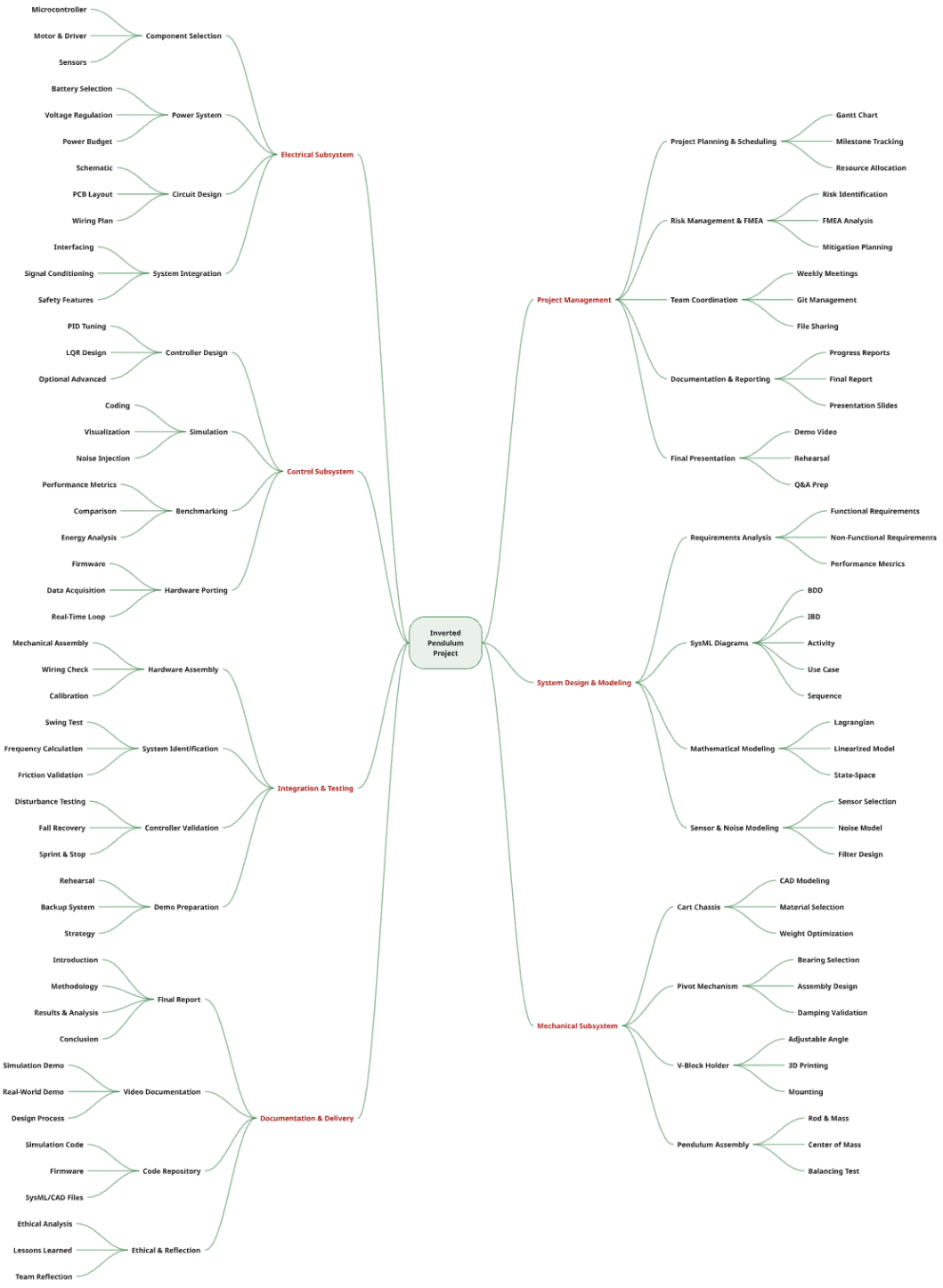


Figure 4: Work Breakdown Structure diagram (Created in Miro)

| | | Name | Duration | Start | Finish | Resource Initials | Predecessors |
|---|---|---|---|---|---|---|---|
| 1 | | **Project Management** | **57.5 days?** | **12/01/2026, 08:00** | **01/04/2026, 13:00** | **SF** | |
| 2 | | Project Planning & S... | 45 days? | 12/01/2026, 08:00 | 13/03/2026, 17:00 | SF | |
| 3 | | Risk Management & ... | 10 days? | 19/01/2026, 08:00 | 30/01/2026, 17:00 | SF | |
| 4 | | Team Coordination | 45 days? | 19/01/2026, 08:00 | 20/03/2026, 17:00 | SF | |
| 5 | | Documentation & Re... | 55 days? | 12/01/2026, 08:00 | 27/03/2026, 17:00 | SF | |
| 6 | | Final Presentation pr... | 2.5 days? | 30/03/2026, 08:00 | 01/04/2026, 13:00 | SF;TK;TS;TT | 5 |
| 7 | | **System Design & Mo...** | **58 days?** | **19/01/2026, 08:00** | **08/04/2026, 17:00** | **SF;TK;TS;TT** | |
| 8 | | Requirements Analysis | 7 days? | 19/01/2026, 08:00 | 27/01/2026, 17:00 | SF | |
| 9 | | SysML Diagrams | 3.5 days? | 28/01/2026, 08:00 | 02/02/2026, 13:00 | SF;TT | 8;10;11 |
| 10 | | Mathematical Modeli... | 3.5 days? | 19/01/2026, 08:00 | 22/01/2026, 13:00 | TS;TT | |
| 11 | | Sensor & Noise Mod... | 3.5 days? | 22/01/2026, 13:00 | 27/01/2026, 17:00 | TS;TT | 10 |
| 12 | | **Mechanical Subsyst...** | **13.5 days?** | **19/01/2026, 08:00** | **05/02/2026, 13:00** | **TK;TS;TT** | |
| 13 | | Cart Chassis | 7 days? | 19/01/2026, 08:00 | 27/01/2026, 17:00 | TK | |
| 14 | | Pivot Mechanism | 7 days? | 26/01/2026, 08:00 | 03/02/2026, 17:00 | TK;TT | |
| 15 | | V-Block Holder | 3.111 days? | 26/01/2026, 08:00 | 29/01/2026, 08:53 | TS | |
| 16 | | Pendulum Assembly | 3.5 days? | 02/02/2026, 08:00 | 05/02/2026, 13:00 | SF;TK;TS;TT | 15 |
| 17 | | **Electrical design** | **7 days?** | **19/01/2026, 08:00** | **27/01/2026, 17:00** | **SF;TS;TT** | |
| 18 | | Component Selection | 7 days? | 19/01/2026, 08:00 | 27/01/2026, 17:00 | TT | |
| 19 | | Power System | 10 days? | 12/02/2026, 08:00 | 25/02/2026, 17:00 | TK | 18 |
| 20 | | Circuit Design | 30 days? | 26/02/2026, 08:00 | 08/04/2026, 17:00 | TS | 18;19 |
| 21 | | System Integration | 8.75 days? | 28/01/2026, 08:00 | 09/02/2026, 15:00 | SF;TK;TS;TT | 18 |
| 22 | | **Control Subsystem** | **44.833 da...** | **26/01/2026, 08:00** | **27/03/2026, 15:40** | **SF;TS;TT** | |
| 23 | | Controller Design | 7 days? | 26/01/2026, 08:00 | 03/02/2026, 17:00 | SF;TS | |
| 24 | | Simulation | 3.5 days? | 04/02/2026, 08:00 | 09/02/2026, 13:00 | TS;TT | 23 |
| 25 | | Benchmarking | 14 days? | 09/02/2026, 08:00 | 26/02/2026, 17:00 | TK | |
| 26 | | Hardware Porting | 7 days? | 27/02/2026, 08:00 | 09/03/2026, 17:00 | TK;TT | 23;24;25 |
| 27 | | **Integration & Testi...** | **29.5 days?** | **02/02/2026, 08:00** | **13/03/2026, 13:00** | **SF;TK;TS;TT** | |
| 28 | | Hardware Assembly | 7 days? | 02/02/2026, 08:00 | 10/02/2026, 17:00 | SF;TK | |
| 29 | | System Identification | 7 days? | 12/02/2026, 08:00 | 20/02/2026, 17:00 | TK;TT | |
| 30 | | Controller Validation | 7 days? | 16/02/2026, 08:00 | 24/02/2026, 17:00 | TS | |
| 31 | | Demo Preparation | 3.5 days? | 10/03/2026, 08:00 | 13/03/2026, 13:00 | SF;TS | 30 |
| 32 | | **Documentation & ...** | **5.833 day...** | **20/03/2026, 08:00** | **27/03/2026, 15:40** | **SF;TK;TS;TT** | |
| 33 | | Final Report | 1.75 days? | 24/03/2026, 10:40 | 26/03/2026, 08:40 | SF;TK;TS;TT | 34 |
| 34 | | Video Documentati... | 2.333 days? | 20/03/2026, 08:00 | 24/03/2026, 10:40 | SF;TK;TT | |
| 35 | | Code Repository | 3.5 days? | 20/03/2026, 08:00 | 25/03/2026, 13:00 | TK;TS | |
| 36 | | Ethical & Reflection | 1.75 days? | 26/03/2026, 08:40 | 27/03/2026, 15:40 | SF;TK;TS;TT | 33;34;35 |

Systems Engineering

Figure 5: Work Breakdown Structure table (Created in ProjectLibre)

## 2.7 A-B-M estimates, TE, and sigma

To account for uncertainty in project scheduling, three-point estimates were used for key tasks in the inverted pendulum project. For each task, we defined:

- $A$: optimistic duration (best-case scenario),
- $M$: most likely duration (expected case),
- $B$: pessimistic duration (worst-case scenario).

The expected duration for each task was computed using the PERT formula [1]:

$$TE = \frac{A + 4M + B}{6}, \tag{1}$$

and the associated standard deviation was estimated as:

$$\sigma = \frac{B - A}{6}. \tag{2}$$

Table 3 summarises the estimates for key tasks in the project.

Table 3: A-B-M estimates and uncertainty for key project tasks.

| **Task** | $A$ | $B$ | $M$ | $TE$ | $\sigma$ |
|---|---|---|---|---|---|
| EKF Design | 7 | 14 | 10 | 10.2 | 1.2 |
| Controller Tuning | 10 | 20 | 15 | 11.7 | 1.66 |
| Hardware Integration | 14 | 28 | 21 | 21 | 2.33 |
| Testing and Debugging | 7 | 12 | 16 | 11.8 | 1.5 |
| Final Report Writing | 5 | 7 | 14 | 7.8 | 1.5 |

As an example, for the task *[EKF design]*, with

$$A = 7, \quad B = 14, \quad M = 10,$$

the expected duration is

$$TE = 10.2 \text{ hours}, \tag{3}$$

and the standard deviation is

$$\sigma = 1.2 \text{ hours}. \tag{4}$$

The use of A-B-M estimates was particularly important in this project due to the high uncertainty associated with hardware and integration tasks. In particular, delays were observed due to component failures, including encoder faults and multiple motor replacements, which significantly increased the pessimistic time estimates for hardware-related tasks.

In contrast, software development tasks such as EKF implementation and report writing were more predictable, resulting in smaller uncertainty.

## 2.8 Probability of Completion

To evaluate the likelihood of completing the project within a given deadline, a probabilistic schedule analysis was performed using the PERT framework.

The expected project duration was computed by summing the expected times of tasks along the critical path:

$$T_E = \sum TE_i = 696 \text{ hours}. \tag{5}$$

Assuming task durations are independent, the variance of the total project duration is given by:

$$\sigma^2 = \sum \sigma_i^2 = 11.025, \tag{6}$$

and the standard deviation is:

$$\sigma = \sqrt{\sigma^2} = 105. \tag{7}$$

Let $T_D$ denote the project deadline. The probability of completing the project before this deadline is

approximated using the normal distribution:

$$Z = \frac{T_D - T_E}{\sigma}. \tag{8}$$

Substituting the values:

$$Z = \frac{\underline{780} - \underline{696}}{\underline{105}} = \underline{0.8}. \tag{9}$$

Using standard normal distribution tables, this corresponds to a completion probability of approximately:

$$P(T \leq T_D) \approx \underline{80percent}. \tag{10}$$

The results indicate that the probability of completing the project within the planned timeframe is $\underline{0.8}$, reflecting the impact of uncertainty in hardware integration and testing. In particular, tasks involving physical components exhibited higher variance due to failures such as encoder faults and motor replacements, which increased schedule risk. Software development tasks such as EKF implementation and report writing contributed less uncertainty.

## 2.9 Network diagram

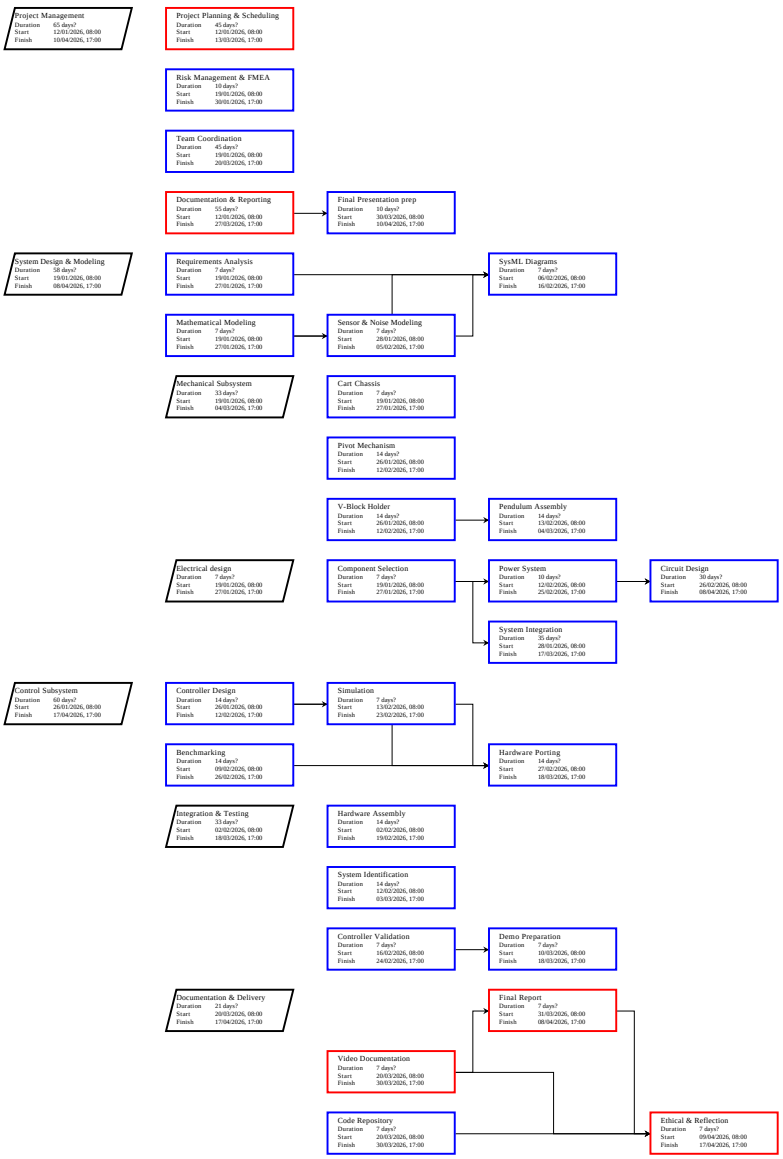


Figure 6: Network Diagram (created in ProjectLibre)

## 2.10 Gantt chart

A simplified Gantt chart, created in ProjectLibre and including resource allocation, has been inserted below.

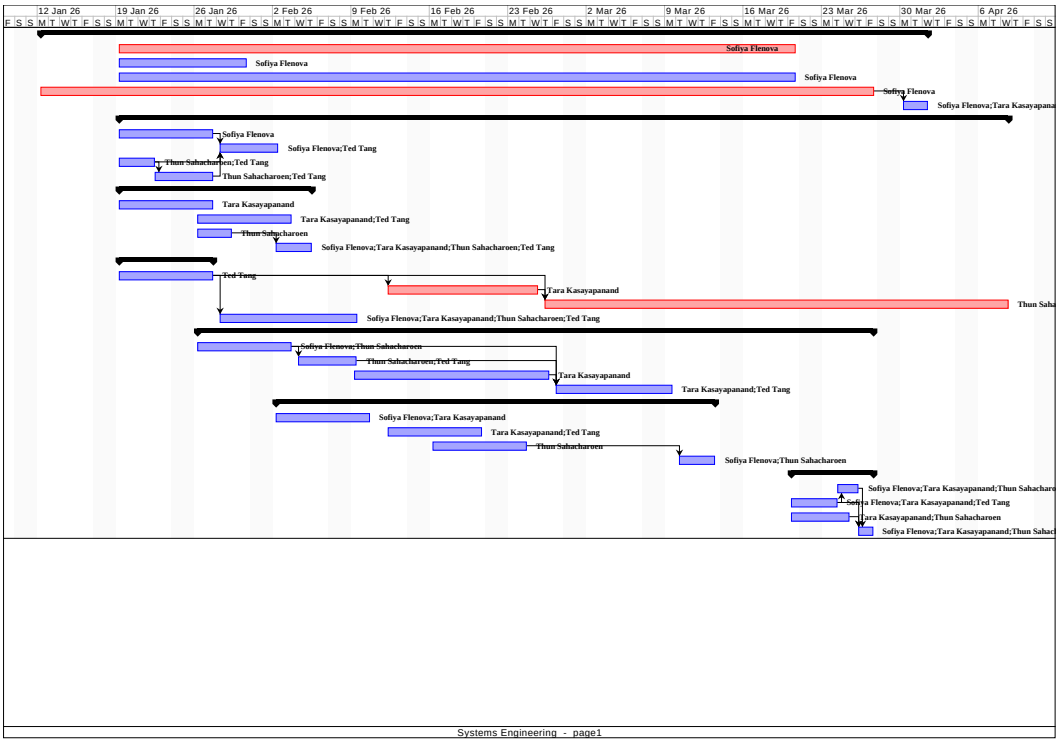


Figure 7: Gantt Chart (created in ProjectLibre)

## 2.11 Critical Path and Crashing Technique

The project schedule is presented in the Gantt chart above, where the Critical Path is highlighted in red. The total duration of the critical path is 58 days, starting on the 12th of January and ending on the day of the report submission, which is 27th of March. The critical path includes the weekends in its structure, since our team has been doing remote work outside of the lab hours. The critical activities in the path include most of the project planning activities, as well as report writing. This path has zero slack and identifies project's minimum completion time. The way in which the project could be shortened, would be crashing tasks in the critical path. That means that the only approach to a crashing technique would be finishing writing the report earlier and completing all the Project Planning tasks earlier than the final deadline, shorter than in 58 days. The crash time is not identified in the diagram above, however, it would be finishing the report before the final demonstration date, which is 17th of March. This would reduce the project duration and the critical path length by 10 days, thus becoming 48 days long.

## 2.12 RACI Matrix

| Task | Tara | Ted | Thun | Sofiya |
|---|---|---|---|---|
| **1.0 Project Management** | | | | |
| 1.1 Project Planning & Scheduling | I | I | I | R |
| 1.2 Risk Management & FMEA | C | C | C | R |
| 1.3 Team Coordination & Communication | C | C | C | R |
| 1.4 Documentation & Reporting | C | C | C | R |
| 1.5 Final Presentation Preparation | R | R | R | R |
| **2.0 System Design & Modeling** | | | | |
| 2.1 Requirements Analysis & Specification | A | C | A | R |
| 2.2 SysML Diagrams | A | A | C | R |
| 2.3 Mathematical Modeling | I | R | R | I |
| 2.4 Sensor & Noise Modeling | I | R | R | I |
| **3.0 Mechanical Subsystem** | | | | |
| 3.1 Cart Chassis Design | R | I | I | I |
| 3.2 Pivot Mechanism Design | R | R | I | I |
| 3.3 V-Block Holder Design | R | R | C | R |
| 3.4 Pendulum Assembly | R | R | R | R |
| **4.0 Electrical Subsystem** | | | | |
| 4.1 Component Selection | R | A | R | A |
| 4.2 Power System Design | C | C | R | C |
| 4.3 Circuit Design & PCB Layout | C | R | C | C |
| 4.4 System Integration | R | R | R | R |
| **5.0 Control Subsystem** | | | | |
| 5.1 Controller Design | I | A | A | R |
| 5.2 Simulation Implementation | I | A | A | R |
| 5.3 Algorithm Comparison & Benchmarking | I | A | A | I |
| 5.4 Code Porting to Hardware | C | A | A | I |
| **6.0 Integration & Testing** | | | | |
| 6.1 Hardware Assembly | R | R | C | I |
| 6.2 System Identification | R | C | R | C |
| 6.3 Controller Tuning & Validation | I | R | I | R |
| 6.4 Final Demo Preparation | R | R | R | R |
| **7.0 Documentation & Delivery** | | | | |
| 7.1 Final Report Compilation | C | C | C | R |
| 7.2 Video Documentation | R | I | I | R |
| 7.3 Code & Model Repository | R | R | I | I |
| 7.4 Ethical & Reflection Section | R | R | R | R |

| Legend | Description |
|---|---|
| R | Responsible (does the work) |
| A | Accountable (approves the work, final decision-maker) |
| C | Consulted (provides input, two-way communication) |
| I | Informed (kept updated, one-way communication) |

## 2.13 PLC

The development of the system followed a Systems Engineering Project Life Cycle (PLC). This approach structures the project into sequential phases, each validated by review milestones to ensure that the system meets requirements before progressing to the next stage.

### 2.13.1 PLC Overview

The project followed the lifecycle shown in Figure 8.

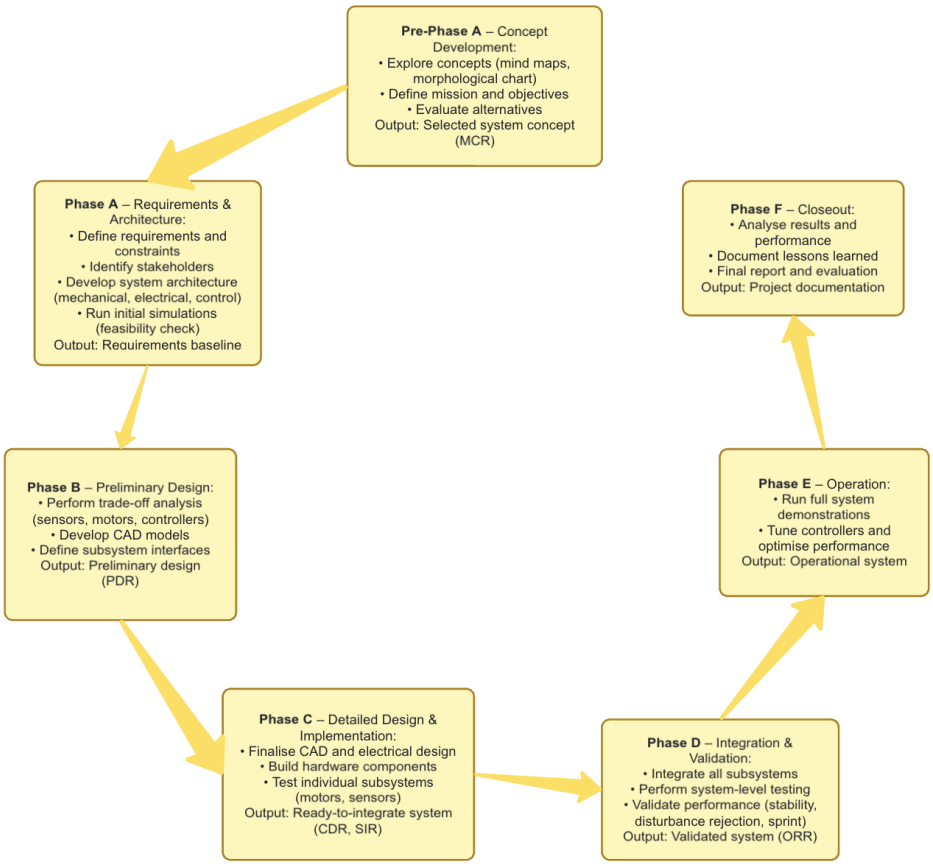


Figure 8: Systems Engineering Project Life Cycle applied to the project

### 2.13.2 Pre-Phase A: Week 1 - Concept Exploration (MCR)

During the concept phase, multiple design approaches were explored to determine the most suitable system architecture. Our team used mind maps and morphological charts to generate and evaluate different solutions. The main project objective was defined. A Mission Concept Review (MCR) was used to select the final system direction.

### 2.13.3 Phase A: Week 2 - Requirements Definition (SRR)

In this phase, system requirements and constraints were identified and formalised. Stakeholders were considered and requirement diagrams were created to ensure that the system objectives were clearly defined. Initial feasibility simulations were conducted to validate the concept. A System Requirements Review (SRR) confirmed that the proposed architecture could meet the defined goals.

### 2.13.4 Phase B: Week 2-3 - Preliminary Design (PDR)

Phase B focused on developing the preliminary system design and evaluating trade-offs between different technical solutions. The system was decomposed into subsystems including the mechanical structure, sensing components and control system. Initial CAD models of the cart and supporting structure were developed. A Preliminary Design Review (PDR) ensured that the design satisfied system requirements.

### 2.13.5 Phase C: Week 4-5 - Detailed Design and Implementation (CDR)

In the detailed design phase, the final system configuration was developed. Full CAD models, electrical schematics and system architecture diagrams were completed. Hardware components were assembled and individual subsystem testing was conducted to verify functionality. A Critical Design Review (CDR) confirmed readiness for system integration.

### 2.13.6 Phase D: Week 6-7 Integration and Validation (ORR)

During this phase, all subsystems were integrated into a complete working system. Testing was performed to evaluate stability, response to disturbances and overall system performance. The system was validated against the defined requirements. An Operational Readiness Review (ORR) ensured the system was ready for final operation.

### 2.13.7 Phase E–F: Operation and Closeout

Finally, the system was operated under the required test conditions and performance was evaluated. The project concluded with documentation, analysis of results and identification of lessons learned for future system improvements.

# 3 Risk Assessment

## 3.1 Identified technical risks

Table 4: Identified Technical Risks

| Risk ID | Risk Statement | Cause | Effect | L | S | RPN | Mitigation |
|---|---|---|---|---|---|---|---|
| **T-R1** | Excessive pivot friction: Damping ratio may exceed $\zeta < 0.01$ threshold, preventing free swing and compromising stabilisation. | Misalignment of bearings, poor bearing selection, insufficient lubrication, or manufacturing tolerances in 3D-printed V-holder. | System fails "Swing Test" validation; poor disturbance rejection; inability to recover from deep fall angles. | 3 | 4 | **12** | Use dual ball bearings (6mm ID and 4mm ID) as specified in BOM; validate with free decay test before proceeding; design pivot with clearance tolerances; test print with fit verification. |
| **T-R2** | Motor torque saturation: Motors may not provide sufficient torque to recover pendulum from large initial angles ($\geq 15°$), causing saturation and failure to stabilise. | Undersized motors relative to pendulum mass; battery voltage drop under load; incorrect gear ratio selection. | Inability to complete Evaluation B (Deep Fall Recovery); poor sprint performance; possible motor stall or damage. | 4 | 4 | **16** | Simulate torque requirements in Python; use gain scheduling for large-angle recovery; implement anti-windup; select higher torque motors (9.7:1 gear ratio); verify power delivery under load. |
| **T-R3** | Sensor noise and calibration drift: Encoder noise or drift may introduce measurement errors, leading to instability or inaccurate tracking. | Electrical interference, low-resolution encoders (48 CPR), improper mounting, temperature effects, or lack of filtering. | Oscillation around equilibrium; failure to meet precision constraints ($\pm 10$ cm sprint accuracy); poor disturbance rejection. | 3 | 3 | **9** | Use 1000 CPR encoder for pendulum angle; implement Extended Kalman Filter (EKF) as described in Section 3.2.2; sensor fusion via complementary filter; log raw data to identify interference. |

**Note:** L = Likelihood (1-5), S = Severity (1-5), RPN = Risk Priority Number (L × S)

## 3.2 Identified legal risks

Table 5: Identified Legal Risks

| Risk ID | Risk Statement |
|---|---|
| **L-R1** | IP ownership and third-party code usage: Using open-source libraries or code snippets without proper attr |
| **L-R2** | Lab safety and equipment damage liability: Damage to lab equipment or injury to team members |
| **L-R3** | Data protection and recording consent: Recording videos or photos without explic |

**Note:** L = Likelihood (1-5), S = Severity (1-5), RPN = Risk Priority Number (L × S)

## 3.3 Ishikawa Diagram

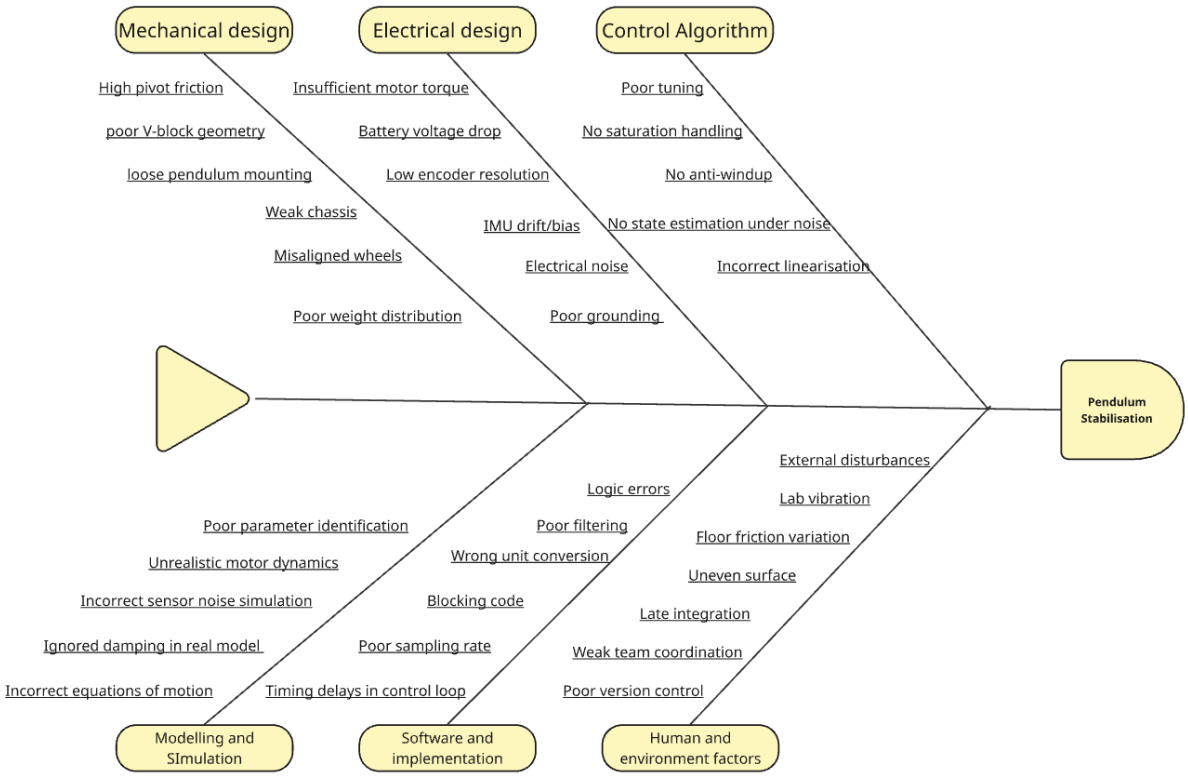


Figure 9: Ishikawa diagram (created on Miro)

## 3.4 FMEA Table

| Item | Failure Mode | Effect | Cause | S | O | D | RPN | Mitigation |
|---|---|---|---|---|---|---|---|---|
| Pendulum Pivot | Excess friction | Fails swing test; poor stabilisation | Misalignment or poor bearing | 9 | 5 | 3 | 135 | Use ball bearings; validate with free decay test |
| Motor Actuator | Torque saturation | Cannot recover from deep fall | Undersized motor | 10 | 4 | 6 | 240 | Simulate torque margin; select higher torque motor |
| IMU Sensor | Drift / bias | Long-term instability | Poor calibration | 8 | 6 | 5 | 240 | Implement calibration and sensor fusion filtering |
| Control Software | Control loop delay | Oscillation or instability | Low sampling rate / blocking code | 9 | 5 | 6 | 270 | Use fixed-time interrupt loop; verify loop frequency |
| Power System | Voltage sag | Reduced motor torque | Battery internal resistance | 7 | 6 | 4 | 168 | Test under load; monitor voltage in real-time |
| 3D Printed V-Holder | Structural failure | Pendulum drops prematurely | Weak infill or layer adhesion | 8 | 4 | 5 | 160 | Use 100% infill; print with stronger orientation |
| Encoder | Signal loss | Erratic angle readings | Loose wiring or electrical noise | 9 | 5 | 4 | 180 | Use shielded cables; secure connectors with strain relief |
| Wheel Traction | Slippage | Cart fails to accelerate as commanded | Low friction surface or worn wheels | 7 | 6 | 3 | 126 | Add rubber gripping; test on different surfaces |
| Pendulum Rod | Bending / flex | Unmodeled dynamics; reduced control accuracy | Insufficient stiffness | 6 | 3 | 4 | 72 | Verify rod straightness; use specified 12mm diameter |

| Item | Failure Mode | Effect | Cause | S | O | D | RPN | Mitigation |
|---|---|---|---|---|---|---|---|---|
| Launch Sequence | Failure to lift off | Pendulum stuck on V-holder | Excessive static friction or weak initial acceleration | 8 | 5 | 5 | 200 | Polish V-holder contact surfaces; tune launch acceleration profile |
| Wire Management | Cable snagging | Restricted cart movement | Loose hanging wires | 5 | 7 | 3 | 105 | Secure wiring to chassis; use cable chains if needed |
| Microcontroller | Overflow / underflow | Control loop crashes | Integer overflow in timing calculations | 9 | 3 | 6 | 162 | Use watchdog timer; implement bounds checking |
| Tip Mass | Detachment | Center of mass shifts; pendulum dynamics change | Loose M4 screw | 6 | 4 | 4 | 96 | Use thread-locking compound; verify tightness before each demo |
| Competition Timing | False trigger | Timer starts/stops incorrectly | Sensor misalignment at finish line | 7 | 5 | 5 | 175 | Use redundant finish detection (optical + manual backup) |

$$RPN = Severity \times Occurrence \times Detection$$

## 3.5 Risk Matrix

| L/H | 1 (Insig) | 2 (Minor) | 3 (Moderate) | 4 (Major) | 5 (Catastrophic) |
|---|---|---|---|---|---|
| **5 (Almost Certain)** | | | **R12** | | |
| **4 (Likely)** | | **R9** | **R4** | **R10** | **R1** |
| **3 (Possible)** | **R7** | **R11** | **R5 R13 R15** | **R3 R8** | **R2** |
| **2 (Unlikely)** | | | **R6 R16** | | |
| **1 (Rare)** | | | **R14 R16** | | |

Table 7: Project Risk Matrix for Inverted Pendulum System

**Technical Risks (1-6):**
- R1: Motor/Battery Failure During Demo
- R2: Inability to Meet "Deep Fall" Recovery Angle
- R3: Excessive Pivot Friction (Damping Ratio > 0.01)
- R4: Sensor Calibration Drift (IMU/Encoder noise)
- R5: Control Loop Timing Jitter
- R6: Wireless Interference with Remote Monitoring

**Technical Risks (7-15):**
- R7: 3D Print Warping/Tolerances (V-holder failure)
- R8: Incorrect System Identification (wrong dynamics model)
- R9: Power Supply Instability (voltage drops under load)
- R10: Motor Saturation (insufficient torque for recovery)
- R11: Encoder Quantisation Error (low resolution)
- R12: Simulation vs. Reality Mismatch (unmodeled dynamics)
- R13: Control Loop Timing Jitter (Duplicate)

**Legal & Compliance Risks (16-19):**
- R14: Radio Frequency Spectrum Non-Compliance (Wireless Interference)
- R15: Laboratory Safety Violation (moving parts hazard)
- R16: Unauthorised Export of Controlled Technology (Export Control / ITAR)

**Legend: Severity:** 1=Insignificant, 2=Minor, 3=Moderate, 4=Major, 5=Catastrophic
**Likelihood:** 1=Rare, 2=Unlikely, 3=Possible, 4=Likely, 5=Almost Certain
**Color Code:** Low Risk, Medium Risk, High Risk

## 3.6 Redundancy and Compensatory Measures

This section outlines strategies implemented to mitigate risks identified in Section 6.3 (Risk Matrix).

| Risk ID | Mitigation Strategy | Type |
|---|---|---|
| R1 | Four motors provide redundancy; if one fails, remaining three can still operate (though asymmetric thrust may require control compensation). Capacitor bank on power rails prevents brownouts. | Redundancy |
| R2/R10 | Gain scheduling for large-angle recovery; anti-windup on PID integral term; S-curve trajectory planning for sprint test to prevent motor saturation. | Compensatory |
| R3 | Dual ball bearings (6mm ID and 4mm ID) at pivot; bevel gear mechanism for smooth rotation; mandatory swing test to verify $\zeta < 0.01$ before proceeding. | Compensatory |
| R4/R11 | Primary: 1000 CPR encoder for angle; Secondary: motor encoders (48 CPR) for cart position. Sensor fusion via complementary filter combines IMU (if available) and encoder data. | Redundancy |
| R5 | Arduino Uno R4 with deterministic control loop; watchdog timer enabled; non-blocking code structure; telemetry/logging runs at lower priority. | Compensatory |
| R6 | Not applicable (no wireless telemetry planned). Offline data logging to serial monitor only. | - |
| R7 | Test prints of V-holder with clearance tolerances; post-processing (sanding, reaming) of pivot holes; PLA annealing if warping observed. | Compensatory |
| R8 | System ID via logarithmic decrement from swing test; cross-validation against theoretical model; iterative parameter refinement in simulation. | Compensatory |
| R9 | Separate power regulation for motors (12V) and Arduino (5V via USB/regulator); decoupling capacitors at motor driver and MCU power pins. | Compensatory |
| R12 | Hardware-in-the-loop validation; PID/LQR gains tuned in simulation first; reduced power during initial real-world tests; gain margins of 6dB minimum. | Compensatory |

Table 8: Risk mitigation strategies mapped to project-specific risks

## 3.7 Power-Interest Grid

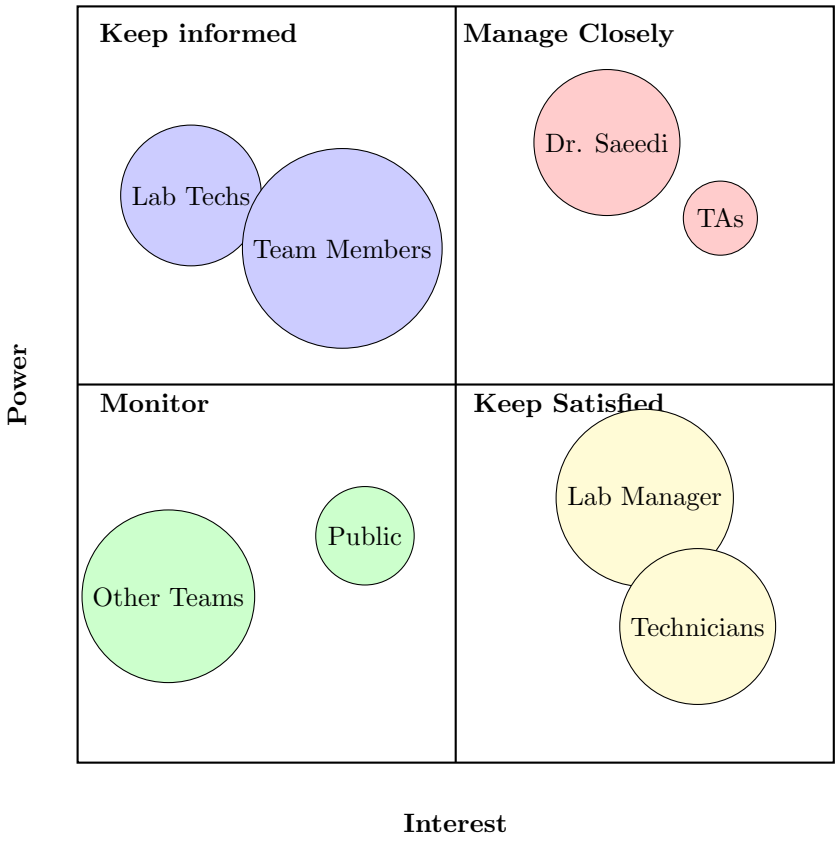


Figure 10: Power-Interest Grid for Stakeholders

## 3.8 Potential human-related issues in design and deployment

Several human-related issues may impact the success of this project. Within the team, communication breakdowns between mechanical, electrical and software sub-teams could lead to integration difficulties, while knowledge silos may create bottlenecks if a key member is unavailable. Task ownership confusion, addressed by the RACI matrix in Section 2.11 and technical disagreements over design choices also present internal challenges. During deployment, operator error during demonstrations, such as incorrect startup sequence or failure to engage emergency stop, poses a risk to both system performance and safety. Fatigue during extended testing sessions may further increase error rates. Externally, misalignment of expectations with instructors or teaching assistants could result in last-minute scope corrections. To mitigate these issues, the team maintains daily check-ins, weekly structured meetings, clear task allocation via Microsoft Planner, cross-training across subsystems and practiced operating procedures for all demonstrations.

## 3.9 Security

Ensuring the security of the inverted pendulum system is important to prevent unauthorised access or accidental damage to the hardware and software. The table below summarises one key security threat, its vulnerability and the mitigation measures.

| Threat | Vulnerability | Mitigation |
|---|---|---|
| Unauthorised code injection via USB during programming | The microcontroller accepts unsigned firmware without authentication during upload | • Physically secure the Arduino during development<br>• Verify the source of all code before uploading<br>• Use a password-protected computer for programming<br>• Disconnect USB when not actively programming |

Table 9: Security threat, vulnerability and mitigation for the inverted pendulum system

## 3.10 Applying Lean Mehtod to Minimise Waste

The primary Lean waste identified was defects, specifically failed 3D prints, incorrect component selection and unstable controller behaviour requiring rework. To address this, a structured root cause analysis approach was adopted for all issues: verifying CAD dimensions for mechanical failures, logging raw data for sensor noise, comparing real behaviour against simulation for controller instability and measuring voltage under load for power issues. This eliminated guesswork and prevented superficial fixes.

Before implementing hardware changes, all controller modifications were validated in Python simulation to ensure stability margins, recovery from large angles ($> 15°$), adequate noise filtering and achievable damping ratio ($\zeta < 0.01$). A test print protocol was implemented for the V-holder: low-resolution rapid prototypes verified fit and clearance tolerances ($\pm 0.2$mm), with high-resolution final prints only after approval, reducing failed prints by approximately 30%.

## 3.11 Commitment Assessment

This section evaluates stakeholder commitment levels to identify potential project risks from insufficient engagement.

### Key Stakeholders

The following stakeholders are critical to project success:

- **Dr. Sajad Saeedi** (Course Professor)
- **Teaching Assistants** (Rehan Agrawal, Muhammad Maaz, Yusuf Adekola)
- **Team Members** (Sofiya Flenova, Tara Kasayapanand, Thun Sahacharoen, Ted Tang)
- **Lab Manager** (Ollie Collier)
- **Lab Technicians** (Alex Charitonidis, Matt Madden, Ben Barwise, Isabella Barbaro)

**Current vs. Required Commitment Assessment**

| Stakeholder | Current Commitment | Required Commitment | Gap | Evidence / Observations |
|---|---|---|---|---|
| Dr. Sajad Saeedi (Professor) | High | High | None | Regular lectures; defined project requirements; available during scheduled review sessions |
| Teaching Assistants | High | High | None | Weekly lab presence; responsive to queries on Moodle; provide feedback on milestone submissions |
| Team Members | Moderate to High | High | **Minor Gap** | Regular attendance at meetings; tasks distributed but occasional coordination delays observed |
| Lab Manager | Moderate | Moderate | None | Provides equipment access; enforces lab safety protocols; available for bookings |
| Lab Technicians | Moderate | Moderate | None | Assist with 3D printing and component sourcing; response time varies based on workload |
| External Evaluators (Demo Day) | Unknown (Future) | High | **Potential Gap** | Commitment level unknown until demonstration day; risk if expectations misaligned |

Table 10: Commitment assessment for key project stakeholders

**Gap Analysis**

The following commitment gaps present potential project risks:

- **Team Coordination:** While individual team members show high commitment, occasional communication delays and overlapping responsibilities create minor inefficiencies. This could impact integration timelines if not addressed.

- **Lab Technician Availability:** Technicians are committed but shared across multiple projects. Response times for 3D printing support or component requests may vary, potentially delaying mechanical prototyping.
- **Demonstration Day Evaluators:** External evaluators' commitment level and specific assessment criteria are unknown until the final demo. This introduces uncertainty in performance expectations.

**Actions to Maintain and Increase Commitment**

To address identified gaps and sustain stakeholder engagement:

- **Team Coordination:**
  - Daily WhatsApp check-ins for quick updates
  - Weekly structured meetings with written agendas and minutes
  - Clear task allocation using Microsoft Planner with assigned owners and deadlines
  - Shared GitHub repository with documented contribution guidelines
- **Supervisor and TA Engagement:**
  - Prepare concise progress updates for each review milestone
  - Seek technical guidance proactively rather than waiting for issues to escalate
  - Document feedback received and track resolution
- **Demonstration Preparation:**
  - Clarify evaluation criteria early through TAs and project brief
  - Conduct mock demonstrations to simulate evaluator expectations
  - Prepare backup plans for each evaluation scenario

The project faces low commitment-related risk. The primary risk area is internal team coordination, which is being actively managed through structured communication and task tracking.

# 4 System Requirement Specification

## 4.1 Dynamics and kinematics requirements consistency

The simulation and control software implement a single coherent kinematic and dynamic description.

The controlled state is

$$\mathbf{z} = [\, x,\ \dot{x},\ \theta,\ \dot{\theta}\,]^{\mathsf{T}},$$

with $x$ the cart position (m) along the ground, positive to the right; $\theta$ the pendulum angle (rad) measured from the upright vertical, with $\theta = 0$ at the inverted equilibrium and positive $\theta$ defined clockwise when viewed along the axis of rotation (Section 5.1.1–5.1.3). The simulation initial condition used in `main.py` is $x(0) = 0$, $\dot{x}(0) = 0$, $\theta(0) = 15°$, $\dot{\theta}(0) = 0$, which exercises recovery from a non-zero tilt consistent with the mission scenario.

The plant model uses lumped parameters $M$, $m$, $l$, $I$, $g$ and viscous damping coefficients $b_x$ (cart) and $b_\theta$ (pendulum), with default values $M = 1.0kg$, $m = 0.2kg$, $l = 0.3m$, $I = 0.006kg\,m^2$, $g = 9.81m/s^2$, $b_x = 0.1N\,s/m$, $b_\theta = 0.05N\,m\,s$. The linearised matrices used for LQR match the nonlinear integrator's parameter set.

The horizontal control force applied to the cart is saturated in software as $|u| \leq u_{\max}$ with $u_{\max} = 20N$, which bounds the simulated actuator effort.

The estimator assumes measurements of cart position $x$ and pendulum angle $\theta$ only, i.e. the output map $\mathbf{H}$ with structure [1 0 0 0] and [0 0 1 0]. In closed-loop demos, Gaussian noise with $\sigma_x = 0.01m$ and $\sigma_\theta = 0.05rad$ is applied to those two components (`Sensor` in `demos.py`), matching the noise levels used in Section 5.2.1 *[adjust cross-ref if your section numbering differs].* The same state ordering, angle convention, parameter set, force limit and $(x, \theta)$ measurement assumption are used across `dynamics.py`, `controllers.py`, `kalman_filter.py` and `demos.py`; the only deliberate variation is alternative LQR weights or sprint references in specific benchmark scripts, which do not change the underlying kinematic or dynamic definitions above.

## 4.2 Logical Requirements Consistency

To check that the system requirements are logically consistent, we follow the formal logic-based approach introduced in the lecture, where requirements are first expressed as logical propositions and then examined to determine whether there exists at least one combination of states and inputs for which they can all be true simultaneously. A set of requirements is considered consistent if it does not contain any contradiction.

For the inverted pendulum system, the key behavioural requirements can be represented using the following propositions:

- $p$: the start button is pressed,
- $q$: the pendulum angle satisfies $|\theta| \leq \theta_{\text{launch}}$,
- $r$: the pendulum angle satisfies $|\theta| \leq 5°$,
- $s$: a sprint command is issued,
- $t$: the cart has reached the 2 m target,
- $f$: the pendulum angle exceeds the safe limit, i.e. $|\theta| > \theta_{\max}$.

Using these propositions, the main behavioural requirements of the controller can be written as:

$$R_1 : (p \wedge q) \Rightarrow \textsc{Launch}, \tag{11}$$

$$R_2 : r \Rightarrow \textsc{Balance}, \tag{12}$$

$$R_3 : (\textsc{Balance} \wedge s) \Rightarrow \textsc{Sprint}, \tag{13}$$

$$R_4 : (\textsc{Sprint} \wedge t \wedge r) \Rightarrow \textsc{Balance}, \tag{14}$$

$$R_5 : f \Rightarrow \textsc{Fault}. \tag{15}$$

These statements encode the intended system behaviour: the system may only enter Launch when the start condition is satisfied, may only enter Balance when the pendulum is sufficiently upright, may only begin sprinting from the balanced state, and must always transition to Fault whenever the safe tilt bound is violated.

The consistency check is performed by examining whether any requirement directly forbids another. In this case, no contradiction arises. For example, $R_3$ requires sprinting to occur only when the system is already in Balance, while $R_5$ overrides all other modes whenever the fault condition $f$ is true. This does not create a contradiction, because the two rules apply under different logical conditions. Similarly, the launch, balance, and sprint requirements are sequentially compatible: satisfying $R_1$ can lead to a state in which $R_2$ becomes true, and satisfying $R_2$ enables the condition in $R_3$.

Therefore, the requirement set is logically consistent, since there exists at least one valid sequence of inputs and states in which all requirements can be satisfied without conflict.

### 4.3 Mealy Machine

Following the formal methods framework covered during the lecture, the control logic of the inverted pendulum system is specified as a Mealy machine, an extension of the finite state machine (FSM) in which outputs depend on both the current state and the current input, rather than on state alone. The machine is defined as $M = \{S, I, T, \Sigma, \delta, \Gamma\}$, where:

- $S = \{\textsc{Idle},\ \textsc{Launch},\ \textsc{Balance},\ \textsc{Sprint},\ \textsc{Fault}\}$ is the finite, nonempty set of states;
- $I = \textsc{Idle}$ is the initial state;
- $T = \{\textsc{Fault}\}$ is the set of terminal states;
- $\Sigma = \{\text{startPressed},\ |\theta| \leq \theta_{\text{launch}},\ |\theta| \leq 5^\circ,\ |\theta| > \theta_{\max},\ \text{sprintCmd},\ x \geq 2\,\text{m},\ \text{reset}\}$ is the finite alphabet of input events;
- $\delta : S \times \Sigma \to S$ is the state transition function;

- $\Gamma = \{\text{motorsOff, initController, enableClosedLoop, setSprintRef, resumeBalance, clearFaults}\}$ is the set of possible outputs.

The transition function $\delta$ and its associated outputs are summarised in Table 11. In IDLE, the system polls the start button; upon receiving a launch trigger with $|\theta| \leq \theta_{\text{launch}}$, it transitions to LAUNCH and initialises the controller. Once the pendulum satisfies $|\theta| \leq 5°$, the system transitions to BALANCE and enables closed-loop control. A sprint command received in BALANCE causes a transition to SPRINT, where a 2 m position reference is set. Upon reaching the target, the system returns to BALANCE. From any state, exceeding $\theta_{\max}$ triggers an immediate transition to FAULT, where all motor outputs are set to zero, which is the only terminal state of the machine.

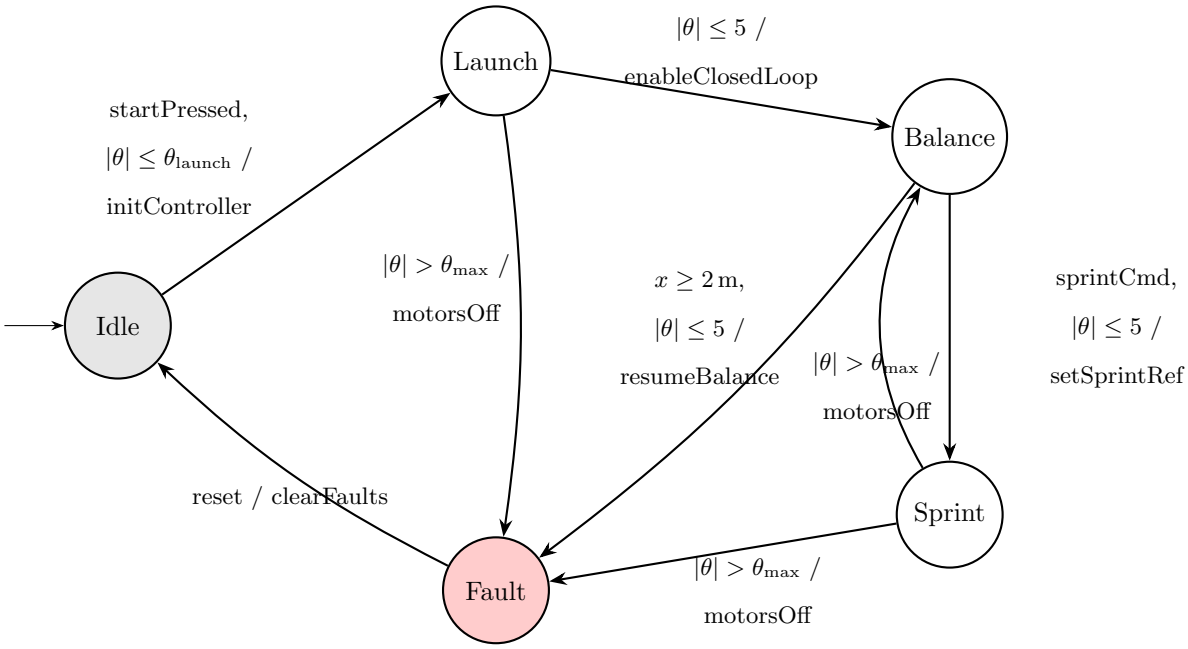


Table 11: Mealy machine transition function $\delta$ for the inverted pendulum system.

| **Current State** | **Input** $\Sigma$ | **Next State** | **Output** $\Gamma$ |
|---|---|---|---|
| IDLE | startPressed, $\lvert\theta\rvert \leq \theta_{\text{launch}}$ | LAUNCH | initController |
| LAUNCH | $\lvert\theta\rvert \leq 5°$ | BALANCE | enableClosedLoop |
| LAUNCH | $\lvert\theta\rvert > \theta_{\max}$ | FAULT | motorsOff |
| BALANCE | sprintCmd, $\lvert\theta\rvert \leq 5°$ | SPRINT | setSprintRef |
| BALANCE | $\lvert\theta\rvert > \theta_{\max}$ | FAULT | motorsOff |
| SPRINT | $x \geq 2\,\text{m}$, $\lvert\theta\rvert \leq 5°$ | BALANCE | resumeBalance |
| SPRINT | $\lvert\theta\rvert > \theta_{\max}$ | FAULT | motorsOff |
| FAULT | reset | IDLE | clearFaults |

## 4.4 Model-based System Engineering

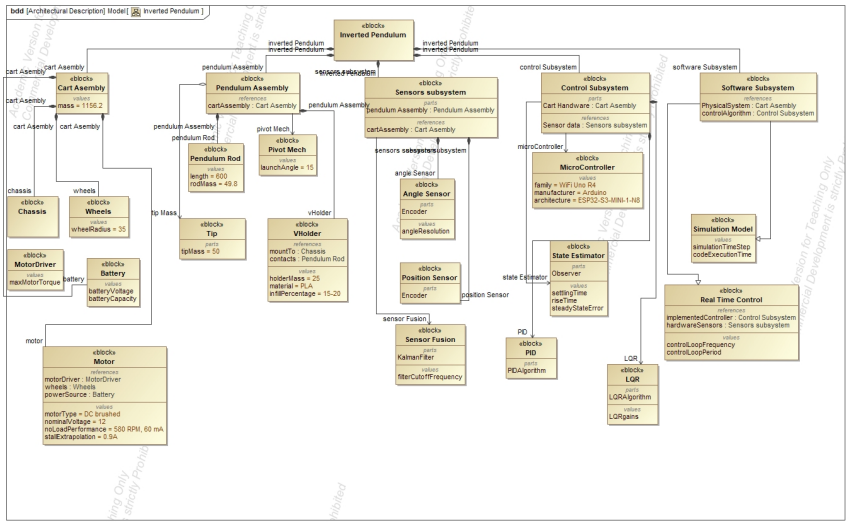


Figure 11: BDD diagram - Inverted Pendulum. Created in Catia Magic.

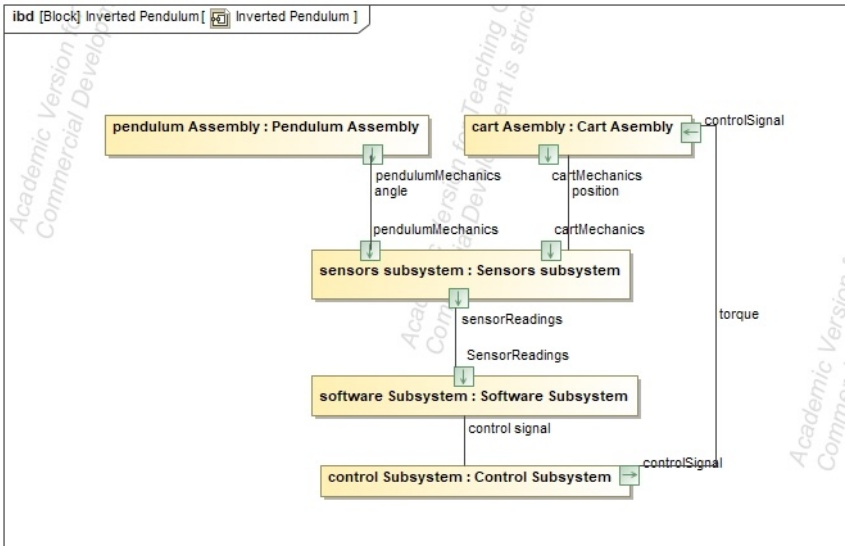


Figure 12: IBD diagram - Inverted Pendulum. Created in Catia Magic.

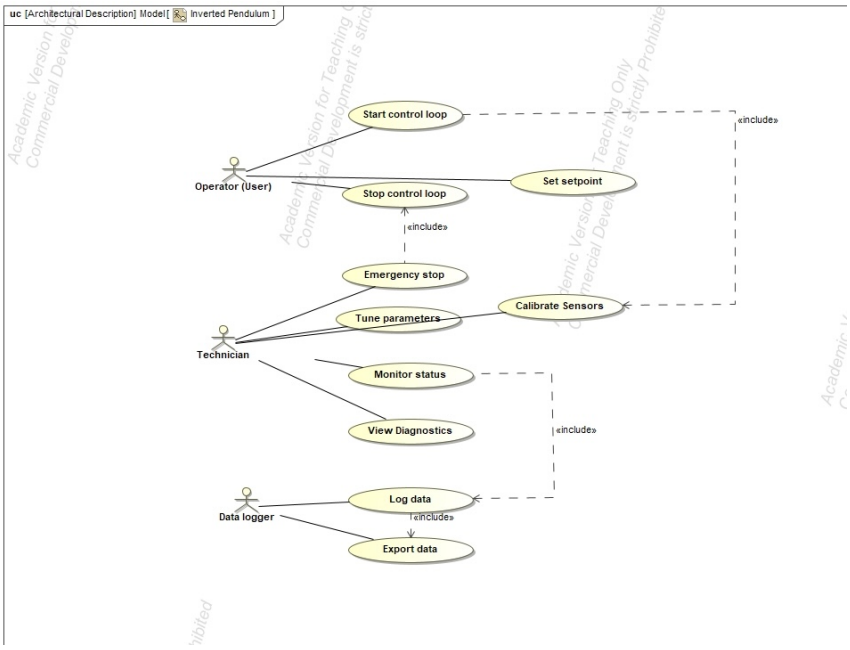


Figure 13: UC diagram - Inverted Pendulum. Created in Catia Magic.

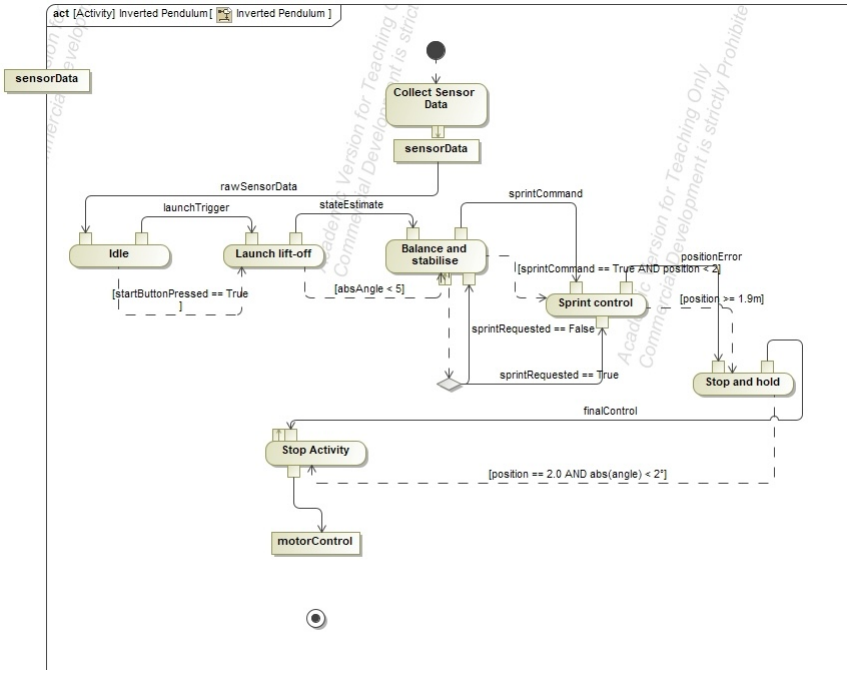


Figure 14: ACT diagram - Inverted Pendulum. Created in Catia Magic.

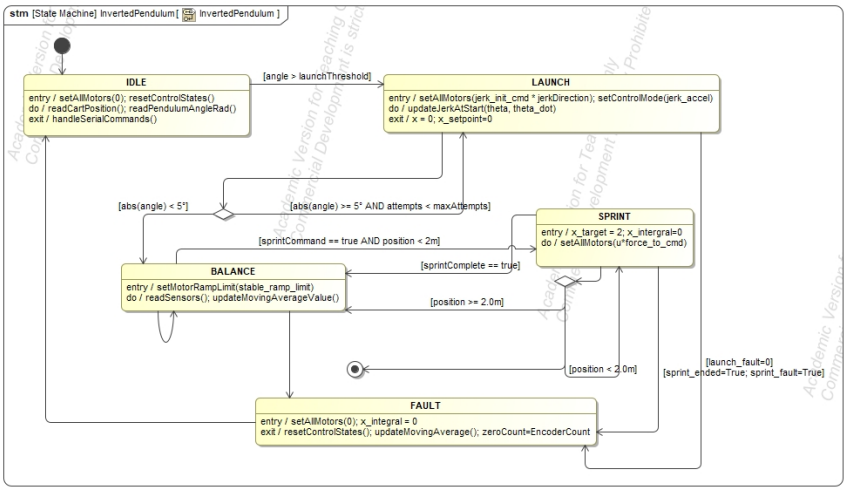


Figure 15: STM diagram - Inverted Pendulum. Created in Catia Magic.

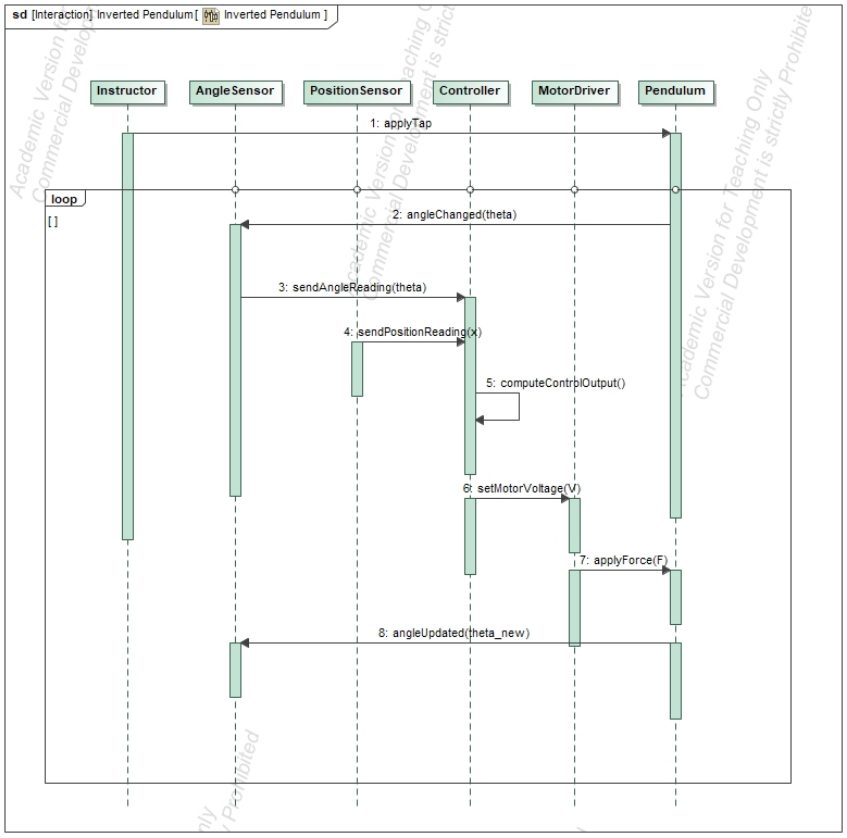


Figure 16: SD diagram - Inverted Pendulum. Created in Catia Magic.

## 4.5 Reliability

Reliability is defined as the probability that the system operates correctly without failure over a given time interval. For the inverted pendulum system, a failure is defined as any run in which the controller enters the FAULT state, the pendulum exceeds the safety limit $|\theta| > \theta_{\max}$, or the system is unable to complete the required task safely.

A probabilistic measure of reliability can be written as

$$R(t) = P(T > t), \tag{16}$$

where $T$ is the time to failure. Based on our experiments, the empirical reliability over a task duration of $t = 30\,\mathrm{s}$ is

$$\hat{R}(t) = \frac{N_{\text{success}}}{N_{\text{total}}} = \frac{23}{35} = 0.657. \tag{17}$$

Here, $N_{\text{success}}$ is the number of successful runs completed without entering the FAULT state, and $N_{\text{total}}$ is the total number of runs tested. The Mean Time To Failure (MTTF) test is run over 9 failed tests, where the failure point is clear and obvious. It is estimated as

$$\text{MTTF} = \frac{\sum_{i=1}^{N_f} T_i}{N_f} = \frac{13 + 7 + 3 + 7 + 9 + 19 + 11 + 3 + 29}{9} = \frac{102}{9} = 11.\overline{2}\ \text{s}. \tag{18}$$

where $T_i$ is the duration until failure for the $i$th failed run, and $N_f$ is the number of failed runs observed. If no failures occurred during testing, then the experimental result only supports the statement

$$\text{MTTF} > 50\ \text{s}, \tag{19}$$

where the bound is given by the longest failure-free observation period.

For repairable systems, the Mean Time Between Failures (MTBF) is defined as

$$\text{MTBF} = \frac{T_{\text{op}}}{N_f} = \frac{3 + 13 + 9 + 6 + 7 + 19 + 11 + 29 + 7}{9} = \frac{104}{9} = 11.\overline{5}\ \text{s}. \tag{20}$$

where $T_{\text{op}}$ is the total accumulated operating time across all runs. In our case,

$$T_{\text{op}} = \underline{9} \times \underline{30} = \underline{270}\ \text{s}, \tag{21}$$

corresponding to $\underline{9}$ runs of duration $\underline{30}\,\mathrm{s}$ each. Since explicit repair or recovery times are not modelled in this system, the mean time between failures is approximately equal to the mean time to failure.

In addition to probabilistic measures, heuristic reliability indicators were also used. The estimator and controller achieved a position RMSE of $\underline{0.11}\,\mathrm{m}$, a yaw RMSE of $\underline{5.2}^\circ$, and a final position error of $\underline{0.14}\,\mathrm{m}$ on the most representative task. These values indicate that the system remained stable and accurate throughout operation, with no evidence of estimator divergence or unsafe control behaviour.

In addition to run-based reliability metrics, system-level reliability was evaluated based on hardware performance over the full development period.

The system was tested over approximately 5–6 weeks of active lab work, with an average of 6 hours

per week, giving a total operating time of

$$T_{\text{op}} \approx 33 \text{ h}. \tag{22}$$

During this period, a total of $N_f = 5$ hardware failures were observed, including two encoder failures and three motor replacements.

For a repairable system, the Mean Time Between Failures (MTBF) is defined as

$$\text{MTBF} = \frac{T_{\text{op}}}{N_f} = \frac{33}{5} = 6.6 \text{ h}. \tag{23}$$

Since the system was repaired after each failure, MTBF is the most appropriate reliability measure. An approximate Mean Time To Failure (MTTF) can also be reported as

$$\text{MTTF} \approx 6.6 \text{ h}, \tag{24}$$

assuming similar failure behaviour between repairs.

These results indicate that the system operated reliably for several hours between failures, although overall platform reliability was limited by hardware durability.

# 5 Modelling and Simulation

## 5.1 Equations of Motion

### 5.1.1 Generalised Coordinates

Our system dynamics can be described using this generalised coordinates:

$$\mathbf{q} = \begin{bmatrix} x \\ \theta \end{bmatrix}$$

where $x$ represents the horizontal distance that the cart moved along the ground and the rightwards direction is defined to be positive. $\theta$ represents the angular displacement of the pendulum with respect to the vertical upright direction. To be more specific, the upright equilibrium corresponds to $\theta = 0$. Positive values of $\theta$ is clockwise rotation of the pendulum from the vertical upright position.

### 5.1.2 Physical Parameters

The physical parameters that we used in our dynamic modeling for the mobile inverted pendulum system are defined as follows:

| Symbol | Description |
|---|---|
| $M$ | Mass of the cart |
| $m$ | Total mass of the pendulum rod including the tip disk |
| $l$ | Distance from the pivot point to the center of mass of the pendulum |
| $h$ | Vertical distance from the cart center of mass to the pendulum pivot point |
| $I$ | Moment of inertia of the pendulum about its center of mass |
| $g$ | Gravitational acceleration |
| $F$ | Horizontal input force applied to the cart |

Table 12: Physical parameters of the mobile inverted pendulum system

### 5.1.3 State Vector and Kinematic Relationships

The system state vector is defined as

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} \end{bmatrix}^T$$

where $x$ and $\dot{x}$ denote the horizontal position and velocity of the cart and $\theta$ and $\dot{\theta}$ represent the angular displacement and angular velocity of the pendulum respectively.

Throughout the project, the cart will be moving on the horizontal ground. So we assume the center of mass of the cart is given by:

$$\mathbf{p}_c = \begin{bmatrix} x \\ 0 \end{bmatrix}$$

And since vertical motion of the cart is neglected, the velocity of the cart center of mass is:

$$\dot{\mathbf{p}}_c = \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix}.$$

The pendulum is attached to the cart at a pivot point located a fixed vertical distance $h$ above the cart's center of mass. The position of the pendulum pivot is therefore given by

$$\mathbf{p}_{pivot} = \begin{bmatrix} x \\ h \end{bmatrix}.$$

The position of the pendulum center of mass relative to the inertial frame is expressed as

$$\mathbf{p}_p = \begin{bmatrix} x + l\sin\theta \\ h + l\cos\theta \end{bmatrix}.$$

where $l$ denotes the distance from the pivot point to the center of mass of the pendulum.

The velocity of the pendulum center of mass can be figured out by differentiating the position with respect

to time:

$$\dot{\mathbf{p}}_p = \begin{bmatrix} \dot{x} + l\dot{\theta}\cos\theta \\ -l\dot{\theta}\sin\theta \end{bmatrix}.$$

Apart from translational motion,the pendulum also undergoes rotational motion about its center of mass, which is characterised by the angular velocity $\dot{\theta}$.

### 5.1.4 Kinetic and Potential Energy

Total kinetic and potential energy of the system will be figured out in this section, ready for the future Lagrangian formulaion of the equations of motion.

The total kinetic energy consists of the translational kinetic energy of the cart, the translational kinetic energy of the pendulum center of mass and the rotational kinetic energy of the pendulum about its center of mass.

The cart moves only in the horizontal direction. Its kinetic energy is

$$T_c = \frac{1}{2}M\dot{x}^2.$$

The squared magnitude of the translation velocity of the pendulum is

$$\|\dot{\mathbf{p}}_p\|^2 = (\dot{x} + l\dot{\theta}\cos\theta)^2 + (l\dot{\theta}\sin\theta)^2.$$

Thus, the translational kinetic energy of the pendulum is

$$T_{p,\text{trans}} = \frac{1}{2}m\left[(\dot{x} + l\dot{\theta}\cos\theta)^2 + l^2\dot{\theta}^2\sin^2\theta\right].$$

In addition to translation, the pendulum rotates about its center of mass with angular velocity $\dot{\theta}$. The corresponding rotational kinetic energy is

$$T_{p,\text{rot}} = \frac{1}{2}I\dot{\theta}^2.$$

The total kinetic energy of the system is therefore

$$T = T_c + T_{p,\text{trans}} + T_{p,\text{rot}}.$$

The cart does not contribute to the gravitational potential energy since its motion is purely horizontal.

The vertical position of the pendulum center of mass is

$$y_p = h + l\cos\theta,$$

whicha leads to the gravitational potential energy

$$V = mg(h + l\cos\theta).$$

As $h$ is a constant offset, it does not affect the equations of motion and may be omitted without loss of generality. The potential energy is therefore written as

$$V = mgl\cos\theta.$$

### 5.1.5 Equations of Motion

Based on the total kinetic and potential energy expressions that we calculated in the previous section, the equation of the motion of the system can therefore be obtained using the Euler–Lagrange formulation [2].

The Lagrangian of the system is defined as

$$\mathcal{L}(q, \dot{q}) = T - V,$$

The generalised coordinates that we stated before are:

$$q = \begin{bmatrix} x & \theta \end{bmatrix}^T.$$

Ignoring any disturbances for now, an external horizontal force $F$ exerted by the motor is applied to the cart, while no external torque is directly applied to the pendulum. So the generalised forces associated with $x$ and $\theta$ are given by

$$Q_x = F, \qquad Q_\theta = 0.$$

Applying the Euler–Lagrange equation

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = Q_i,$$

for equations of motion.

After simplification, the equations of motion can be expressed as

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F,$$

$$(I+ml^2)\ddot{\theta} + ml\ddot{x}\cos\theta - mgl\sin\theta = 0.$$

These equations together describe the full non-linear dynamics of the system and form the basis for future linearisation, simulation and controller design.

### 5.1.6 Damping Addition

In practical, inverted pendulum system's energy dissipation arises due to friction in the cart motion and rotational resistance at the pendulum pivot. In order to capture these effects, viscous damping are added to the equation of motion.

A linear damping force acting on the cart is modelled as

$$F_d = -b\dot{x},$$

where $b$ is the cart damping coefficient.

Similarly, a viscous rotational damping torque at the pendulum pivot is given by

$$\tau_d = -c\dot{\theta},$$

where $c$ is the rotational damping coefficient.

These damping effects are incorporated into the generalised forces as:

$$Q_x = F - b\dot{x}, \qquad Q_\theta = -c\dot{\theta}.$$

After introducing damping, our final equation of motion can be expressed as:

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F - b\dot{x}$$

$$(I+ml^2)\ddot{\theta} + ml\ddot{x}\cos\theta - mgl\sin\theta = -c\dot{\theta}$$

We include these terms in our equation of motion to improve the realism of the model so that it helps us

get more accurate parameters in the simulation, making actual real system easier to tune and implement.

### 5.1.7 Linearisation Around the Upright Equilibrium

In order to work for simulation and controller analysis, our non-linear dynamics of the system should be switched into linear one. Therefore the equation is further computed by linearising it about the upright equilibrium of the pendulum. The equilibrium point represents that the pendulum is balanced vertically upright with the cart at rest, where:

$$\theta = 0, \qquad \dot{\theta} = 0, \qquad \dot{x} = 0.$$

Assuming small angular deviations about this equilibrium, we can therefore apply small-angle approximations to the previous equation model:

$$\sin\theta \approx \theta, \qquad \cos\theta \approx 1, \qquad \dot{\theta}^2 \approx 0.$$

Applying these approximations to the previous non-linear equations of motion, the following linearised differential equations can be computed

$$(M+m)\ddot{x} + ml\ddot{\theta} = F - b\dot{x}$$

$$(I+ml^2)\ddot{\theta} + ml\ddot{x} - mgl\theta = -c\dot{\theta}$$

### 5.1.8 State-Space Representation

The above linearised equations form a set of coupled second-order differential equations. Instead, in order to express the system in state-space form, these equations are solved for the accelerations $\ddot{x}$ and $\ddot{\theta}$.

The equations in matrix form can be written in:

$$\begin{bmatrix} M+m & ml \\ ml & I+ml^2 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} F - b\dot{x} \\ mgl\theta - c\dot{\theta} \end{bmatrix}$$

the determinant of the mass matrix can be defined as

$$\Delta = (M+m)(I+ml^2) - (ml)^2 = I(M+m) + Mml^2.$$

Solving it for the yields of accelerations:

$$\ddot{x} = \frac{I + ml^2}{\Delta}(F - b\dot{x}) + \frac{m^2 g l^2}{\Delta}\theta - \frac{mlc}{\Delta}\dot{\theta}$$

$$\ddot{\theta} = \frac{ml}{\Delta}(F - b\dot{x}) + \frac{(M+m)mgl}{\Delta}\theta - \frac{(M+m)c}{\Delta}\dot{\theta}$$

Recalling the state vector that we defined:

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} \end{bmatrix}^T,$$

and further define the control input as the horizontal force applied to the cart, normally acted by motors:

$$u = F,$$

the linearised system can be written in standard state-space form:

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu,$$

where the system matrix $A$ and input matrix $B$ are given by

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{b(I+ml^2)}{\Delta} & \frac{m^2 g l^2}{\Delta} & -\frac{cml}{\Delta} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{bml}{\Delta} & \frac{(M+m)mgl}{\Delta} & -\frac{c(M+m)}{\Delta} \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ \frac{I+ml^2}{\Delta} \\ 0 \\ \frac{ml}{\Delta} \end{bmatrix}.$$

## 5.2 Sensor Noise Model

To simulate realistic measurements for the mobile inverted pendulum, we modelled the cart position and pendulum angle sensors as being subject to Gaussian noise. Let the true system state be

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix},$$

where $x$ is the cart position, $\theta$ is the pendulum angle and the dots denote derivatives with respect to time.

### 5.2.1 Noise Injection

The sensor measurements are assumed to be noisy observations of the cart position $x$ and pendulum angle $\theta$:

$$y = \begin{bmatrix} x_{\text{measured}} \\ \theta_{\text{measured}} \end{bmatrix} = \begin{bmatrix} x \\ \theta \end{bmatrix} + \begin{bmatrix} \eta_x \\ \eta_\theta \end{bmatrix},$$

where $\eta_x \sim \mathcal{N}(0, \sigma_x^2)$ and $\eta_\theta \sim \mathcal{N}(0, \sigma_\theta^2)$ represent zero-mean Gaussian noise for position and angle, respectively. In our simulation, the standard deviations were chosen as

$$\sigma_x = 0.01 \text{ m}, \quad \sigma_\theta = 0.05 \text{ rad}.$$

This noise model was implemented in Python using a `Sensor` class, where the method `add_noise()` injects Gaussian perturbations into the true state:

```
noisy_measurement = sensor.add_noise(true_state[[0,2]])
```

Here, `true_state[[0,2]]` corresponds to the cart position and pendulum angle.

### 5.2.2 Filtering Method

To reduce the impact of sensor noise on control, an Extended Kalman Filter (EKF) [3] was employed. The EKF uses the noisy measurements $y$ to estimate the full state vector $\mathbf{x}$, including the unmeasured velocities $\dot{x}$ and $\dot{\theta}$. The prediction step uses the nonlinear dynamics of the system [4]:

$$\dot{\mathbf{x}} = f(\mathbf{x}, u),$$

and the update step corrects the predicted state using the noisy measurement $y$ and the linearised measurement model:

$$\mathbf{x}_{\text{est}}^{+} = \mathbf{x}_{\text{est}}^{-} + K\left(y - H\mathbf{x}_{\text{est}}^{-}\right),$$

where $K$ is the Kalman gain and $H = [\,1\,0\,0\,0;\,0\,0\,1\,0\,]$ maps the state vector to the measurement space.

The discrete-time implementation in `kalman_filter.py` evaluates the Jacobian of the continuous-time dynamics at the current estimate for the covariance prediction, applies $\mathbf{H}$ above to position and angle only, and wraps the angle innovation to $(-\pi, \pi]$ to avoid discontinuities.

This approach allows the controller to operate on filtered estimates of the state rather than directly on noisy measurements.

## 5.3 Controller Design and Tuning description

### 5.3.1 PID Controller

The first controller selected to stabilise the inverted pendulum is the Proportional–Integral–Derivative (PID) controller. This approach was chosen due to its simplicity, ease of implementation, and straightforward tuning process. The PID controller also provides a useful baseline for comparison with more advanced control strategies.

For this project, two PID-based control implementations are required because two key variables must be considered including the wheel position and the pendulum angle which were both obtained from an optical encoder. A single PID controller is only sufficient for upright angle stabilisation, but when the cart is required to move to and settle at a specified position, an additional controller for position regulation is necessary.

To simply stabilise the pendulum angle around the vertical position, a regular PID controller was implemented with the error defined as

$$e(t) = \theta_{ref} - \theta$$

where $\theta_{ref} = 0$ corresponding to the vertical position, so the control signal is given by

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt}$$

However, the derivative term is sensitive to noise due to numerical differentiation. Although the Kalman filter provides smoothed state estimates, residual noise can still affect the derivative calculation. Therefore, a low pass filter is applied to the derivative term to reduce high frequency noise, at the cost of introducing a small amount of phase lag.

To achieve both pendulum stabilisation and cart position control, a cascaded PID controller was implemented. This approach separates the control problem into two interconnected loops, an outer loop responsible for cart position and an inner loop responsible for pendulum angle stabilisation.

The outer loop regulates the cart position $x$ and generates a desired pendulum angle $\theta_{ref}$. This is implemented using a proportional-derivative (PD) controller in the form:

$$\theta_{ref} = K_{px}(\mathbf{x}_{ref} - \mathbf{x}) + K_{dx}\dot{\mathbf{x}}$$

This controller works by tilting the pendulum in the direction of motion. To ensure stability, the reference angle is constrained within a small range. This prevents excessive tilting that could lead to system instability or actuator saturation. Additionally, a smooth reference trajectory is used instead of a step input. A cubic smoothstep gradually transitions the position from 0 m to 2 m over a fixed duration. This reduces aggressive transients and improves tracking performance.

The resulting $\theta_{ref}$ was then used as the reference input for the inner-loop PID controller, replacing the fixed vertical reference used in the basic stabilisation case. In this way, the inner controller continued to stabilise the pendulum angle, while the outer loop ensured that the cart moved towards the desired position.

### 5.3.2 LQR Controller

In order to stabilize the inverted pendulum at the upright equilibrium position, the second controller we chose to use is a Linear Quadratic Regulator (LQR) controller.We designed it based on the linearized system dynamics.

LQR is chosen because it can optimally minimize a quadratic cost function to balance both of state errors and control effort. It provides closed-loop stability for the linearized system. The stability is guaranteed. Control inputs are smooth without any abrupt changes, which makes it particularly suitable for this stabilizing pendulum project since it requires precise and efficient control.

Recall the continuous-time state-space model linearized at the equilibrium is

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu,$$

where $\mathbf{x} = [x, \dot{x}, \theta, \dot{\theta}]^\top$ is the state vector and $u$ is the control input force applied to the cart.

The LQR controller we built aims to minimize the quadratic cost function:

$$J = \int_0^\infty \left(\mathbf{x}^\top Q\mathbf{x} + u^\top Ru\right) dt,$$

where $Q$ and $R$ are positive semi-definite and positive definite weighting matrices respectively. These two matrices can balance the trade-off between state regulation and control effort.

The state feedback control law of LQR controller is:

$$u = -K\mathbf{x},$$

where the gain matrix $K$ can be computed by solving the continuous-time Algebraic Riccati Equation (CARE):

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0,$$

and

$$K = R^{-1}B^\top P.$$

The controller gain $K$ was automatically computed using `scipy.linalg.solve_continuous_are` in Python. The controller receives the estimated full state $\hat{\mathbf{x}}$ from an Extended Kalman Filter to reduce the impact of sensor noise. The control input at each time step is computed as

$$u = -K(\hat{\mathbf{x}} - \mathbf{x}_{ref}),$$

where $\mathbf{x}_{ref}$ is the desired reference state.

In this work, the default weighting matrices used in the Python simulation (`LQRController`) are

$$\mathbf{Q} = \text{diag}(10,\ 1,\ 200,\ 5), \qquad \mathbf{R} = [2].$$

The weighting matrices $Q$ and $R$ were tuned iteratively to make sure that the system has a fast stabilization and also smooth control effort. Higher weights were assigned to the pendulum angle and the rate to stabilize. Moderate weights were assigned to the cart position and velocity to prevent movement with excessive oscillations. The effort weight $R$ was chosen to prevent the aggressive inputs which might potentially damage our system. We fine tuned these parameters by comparing simulation results and adjusted to ensure that the performance is robust even in the presence of sensor noise and disturbances that existed in the environment.

## 5.4 Python Implementation Details

The simulation and benchmarking code is implemented in Python 3 using `numpy` for numerical linear algebra, `scipy.linalg.solve_continuous_are` for the continuous-time algebraic Riccati equation (CARE) in LQR design, and `matplotlib` for static plots and interactive animations. The codebase is split into small modules: plant dynamics, controllers, estimation, scenario scripts and visualisation.

**Plant model and integration:** File `dynamics.py` implements the nonlinear equations by solving the $2 \times 2$ mass matrix for $(\ddot{x}, \ddot{\theta})$ at each state, with viscous damping $b_x$ on the cart and $b_\theta$ on the pendulum. The linearised model $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$ is built analytically for controller and filter linearisations. Time stepping uses a fixed-step fourth-order Runge–Kutta (`rk4_step`) routine inside `simulate` for open-loop trajectories; closed-loop scripts advance the true plant with RK4 while the estimator runs at the same sample period.

**Controllers:** `controllers.py` defines a discrete-time `PID_controller` acting on the pendulum angle error, with a filtered derivative (first-order low-pass on the differenced error, parameter $N$) to limit noise amplification, plus proportional–integral terms. The `LQRController` loads $(\mathbf{A}, \mathbf{B})$ from `linearised_state_space`, solves the CARE for $\mathbf{P}$ and forms $\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^\top\mathbf{P}$, then applies $u = -\mathbf{K}(\hat{\mathbf{x}} - \mathbf{x}_{\text{ref}})$ with symmetric saturation $|u| \le u_{\max}$.

**Sensors and state estimation:** `kalman_filter.py` provides a `Sensor` class that adds zero-mean Gaussian noise to measured cart position and pendulum angle (standard deviations configurable; defaults align with the noise levels used in Section 5.2). The `KalmanFilter` implements an extended Kalman filter (EKF): the predict step integrates the nonlinear dynamics with Euler steps, builds a Jacobian $\mathbf{F}$ of the continuous-time dynamics (with a numerical fallback for the angular row where needed), uses $\mathbf{F}_d \approx \mathbf{I} + \mathbf{F}\Delta t$ for covariance propagation, and wraps the angle to $(-\pi, \pi]$ after prediction and update. The measurement model is $\mathbf{H} = [\mathbf{e}_1^\top; \mathbf{e}_3^\top]$ (position and angle only); process noise $\mathbf{Q}$ and measurement noise $\mathbf{R}$ are diagonal [5]. A simple first-order low-pass helper is also available for raw measurements.

**Simulation scenarios and benchmarking:** `demos.py` runs the main closed-loop experiments: open-loop comparison of nonlinear versus linearised dynamics (`run_linear_vs_nonlinear_demo`); Kalman-only filtering on the nonlinear plant (`run_kalman_demo`); and regulation or disturbance tests with PID or LQR, each using noisy outputs, EKF state estimates and RK4 for the truth model. The "move 2 m" sprint uses an outer position loop for PID (generating a bounded angle reference) and, for LQR, a smooth reference trajectory for $x$ and $\dot{x}$ based on a smoothstep profile over a chosen move time, with separate LQR weight matrices tuned for tracking. Impulsive disturbances can be injected over the first part of the horizon. Settling metrics (e.g. time within a position band for a minimum duration) are computed in helper functions for comparison with Table 13 metrics.

**Visualisation and interactive tuning:** `visualisation.py` produces multi-panel animations comparing true and filtered states, time histories of position, angle, velocities and control effort, and optional controls to switch between PID and LQR or apply large or small impulsive forces to either the cart chasis or the pendulum. `animate_realtime_control` (invoked from `main.py`) runs a longer horizon with sliders

for PID gains and for diagonal entries of $\mathbf{Q}$ and $R$, recomputing LQR gains on change, plus disturbance and reset actions for informal tuning as well as the simulated sensor noise.

**Entry point:** `main.py` centralises physical parameters $(M, m, l, I, g, b_x, b_\theta)$ and initial conditions, then launches either batch demos or the real-time animation, keeping a single parameter dictionary consistent across dynamics, control and estimation.

## 5.5 Benchmarking and Results

After implementing both the PID and LQR controllers in simulation with Python, here are the results, starting with the 2 meter sprints. The LQR controller shown in Fig. 17 demonstrates a fast and smooth response. The cart accelerates rapidly toward the 2 m target, reaching it in approximately 5.52 seconds. There is a small overshoot (peaking slightly above 2 m), followed by a well-damped settling behavior. The system stabilizes close to the desired position with minimal oscillations. In contrast, the PID controller in Fig. 18 exhibits a slower and more oscillatory response. The cart gradually approaches the target position, taking around 8.16 to settle around 2 m. There are also noticeable oscillations throughout the trajectory, indicating that the controller is continuously correcting errors but not damping them as effectively as LQR.

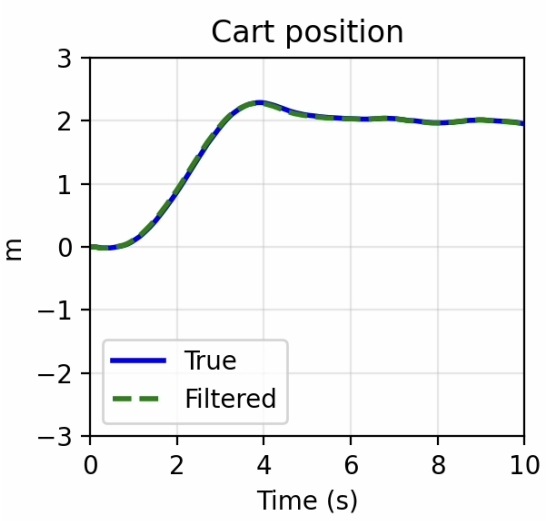


Figure 17: LQR Sprint Position Plot

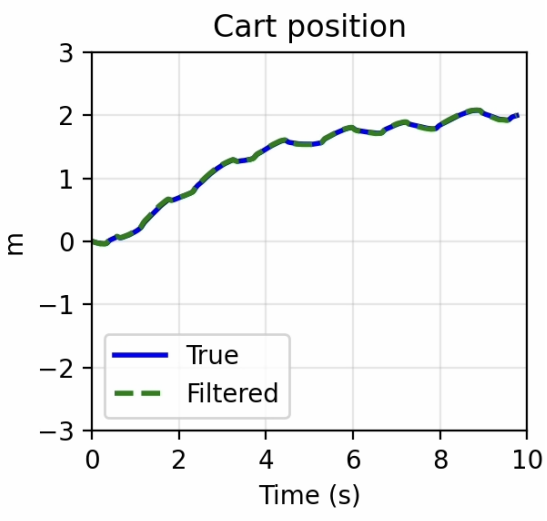


Figure 18: PID Sprint Position Plot

As for starting from an initial angle, the LQR controller (Fig. 19) is able to successfully recover when initialized at 52°. The system quickly counteracts the large initial deviation, exhibiting a brief transient with some oscillations before rapidly converging to the upright equilibrium. Similarly, the PID controller (Fig. 20) is also able to recover from a comparable initial condition of 50°. The response is fast and stable, with the system quickly reducing the initial angle and settling near the upright position with minimal steady-state error. However, unlike the LQR controller, the cart position continues to increase

during the recovery process.

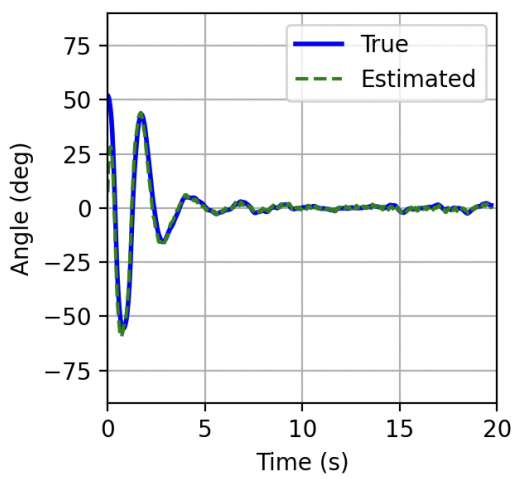


Figure 19: LQR Starting from 52 Degrees

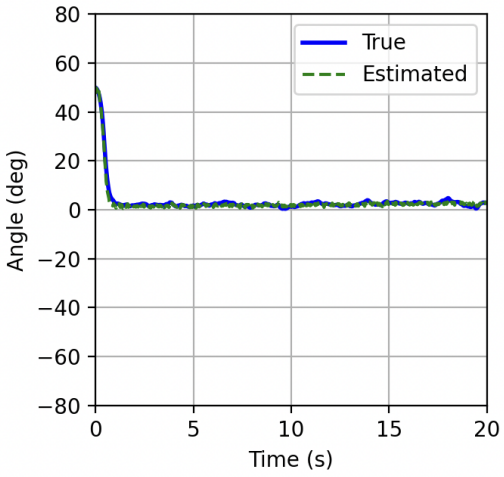


Figure 20: PID Starting from 50 Degrees

To large evaluate disturbance rejection applied to the pendulum, a large external force of 0.5 N was applied to the pendulum for 0.2 seconds, as shown in Figs. 21 and 22. The LQR controller (Fig. 21) demonstrates a stable response to the disturbance, with the angle deviating briefly before returning to the upright position. The transient response includes small oscillations, which are quickly damped, and the system settles back to equilibrium within a few seconds. The PID controller (Fig. 22) also maintains stability under the applied disturbance. The angle deviation is relatively small, and the system quickly returns to the upright position with minimal oscillation. However, the cart position does not settle and continues to drift over time, indicating that the controller does not regulate translational motion. This behaviour highlights a limitation of the angle-only PID approach, where stabilizing the pendulum is achieved at the expense of uncontrolled cart movement.

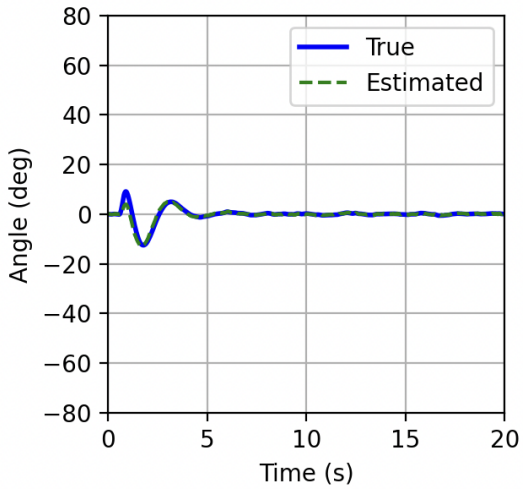


Figure 21: LQR adding Large Disturbance

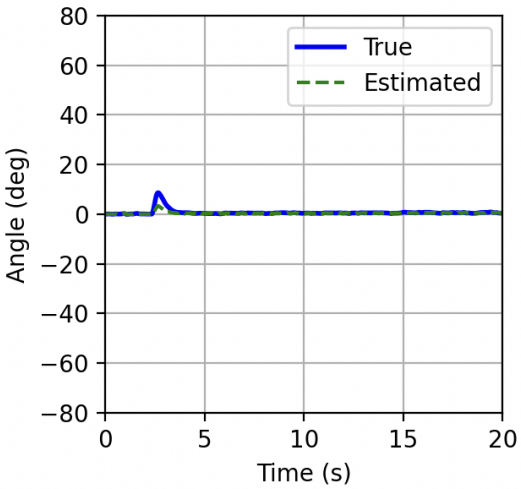


Figure 22: PID adding Large Disturbance

To quantitatively compare the performance of the PID and LQR controllers, a benchmarking framework

was established based on the three primary mission objectives: disturbance rejection, deep fall recovery, and sprint precision. Table 13 summarises the key performance metrics for each controller.

Table 13: Simulation Benchmarking Results for PID and LQR Controllers

| **Metric** | **PID Controller** | **LQR Controller** |
|---|---|---|
| Settling Time (Small Disturbance to Pendulum) [s] | 1.40 | 4.58 |
| Settling Time (Large Disturbance to Pendulum) [s] | 2.12 | 4.52 |
| Settling Time (Large Disturbance to Chasis) [s] | 1.14 | 3.68 |
| Max Recovery Angle [deg] | 50 | 52 |
| Sprint Recovery Time (2 m) [s] | 8.16 | 5.52 |
| Final Position Error [cm] | 0.05 | 0.05 |

The LQR controller, with its optimal state-feedback mechanism, is demonstrate superior disturbance rejection and stability margins compared to the PID controller. Conversely, the PID controller, with its simpler structure, offered faster tuning for the sprint task. However, simulation results show that the PID controller is more susceptible to oscillations, particularly under disturbance or during its movement to $x_{ref}$. This is primarily due to the limited ability of PID control to account for the coupled and nonlinear dynamics of the system. The final selection of the primary control strategy for the physical prototype will be informed by these benchmarking results, prioritising stability and robustness.

# 6 Prototyping

## 6.1 Mechanical Design

The design of the system is a basic 4-wheel drive, chosen for its simplicity. The structure is seen below in Figures 23 and 24.

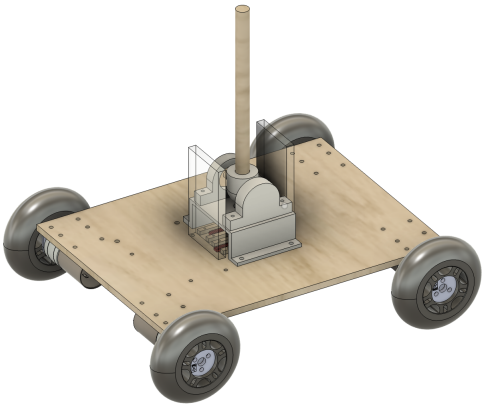

Figure 23: CAD Assembly

Figure 24: Final System

The base is a 170x230x5mm piece of plywood, laser cut with the necessary holes for motor and pendulum mounting. The overall design is kept as simple as possible as the cart only needs to move forwards and backwards. On centre of the robot rests an elevated platform for the pendulum base and v-shaped holder components. A 50g weight is attached to the top of the wooden dowel as seen in Figure 24 through a countersunk screw that connects the mass to the pole.

For horizontal cart actuation we use the Pololu 34:1 Metal Gearmotor 25Dx63L mm MP 12 V with integrated 48 CPR encoder, which provides a stall torque of approximately $\tau_{\text{stall}} = 4.7\,\text{kgcm}$ ($\sim$0.4609 Nm) at 12V while drawing $I_{\text{stall}} = 1.8\text{A}$. Based on a total moving mass of $m_{\text{tot}} = 1845\,\text{g}$ and a target cart acceleration of $a_{\text{req}} = 4\,\text{m/s}^2$, the required horizontal force is

$$F_{\text{req}} = m_{\text{tot}} a_{\text{req}} + F_{\text{friction}} = 7.88\,\text{Nm}$$

where $F_{\text{friction}}$ is a conservative constant of 0.5N. For a drive wheel of radius $r = 0.07\,\text{m}$, this corresponds to a motor shaft torque of

$$\tau_{\text{req}} = F_{\text{req}} r = 0.552\,\text{Nm}$$

In our design, $\tau_{\text{req}}$ is below $\tau_{\text{stall}}$, giving a safety margin of approximately 1.2$\times$ on the available torque.

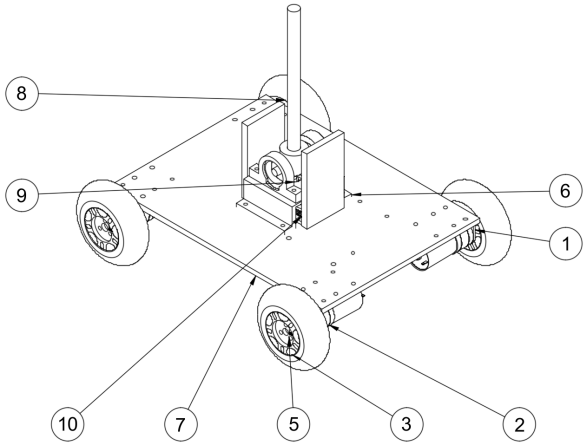


Figure 25: Sketch of Assembly

| Item No. | Quantity | Part |
|---|---|---|
| 1 | 4 | Gearmotor Bracket |
| 2 | 4 | Gearmotor |
| 3 | 4 | Wheel |
| 4* | 4 | Inner Wheel Adapter |
| 5 | 4 | Outer Wheel Adapter |
| 6 | 1 | Elevated Base Assembly |
| 7 | 1 | Base Plate |
| 8 | 1 | Pendulum |
| 9 | 1 | Bevel Gear |
| 10 | 2 | Rack |

Table 14: Parts List

* Indicates parts not indicated in diagram for clarity

The pendulum itself sits inside a 3D printed holder, suspended by a second perpendicular rod that is 6mm in diameter which is screwed through the pole. This second rod is attached to a bearing on one end and the optical encoder on the other to ensure minimum friction. This setup is raised to allow for enough clearance for the pole and its holder to swing, and additionally for the v-shaped holder design.

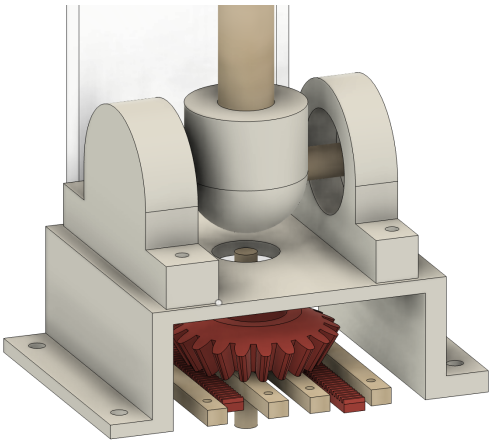

Figure 26: V-Shaped Holder CAD

Referencing Figure 26, an acrylic wall is affixed to the end of each rack, acting as a barrier to stop the pendulum. To adjust the lateral distance of these barriers, a central bevel gear is used to transfer rotary motion to linear across the racks. The bevel gear is actuated manually from underneath the chassis with a wooden rod that is attached to the gear and a bearing. Additionally, two aluminium braces were appended to the bottom of the barriers for extra support and for more absorption of impact from a falling pendulum.

The components on the chassis are placed such that mass distribution would be as even as possible, making the centre of mass of the robot at the bottom of the pendulum. A symmetrical design is adopted, with the encoder and the bearing opposite each other, and similarly with the v-shaped holder. The underside of the chassis also attempts to balance the mass distribution with the battery attached to the front

and the MCU and PCBs on the back end. This evenness allows the dynamics of the robot to be more symmetric and predictable, which is integral for optimal control. Along the horizontal plane, the same amount of control effort should produce similar behaviour when accelerating forwards and backwards.

The swing test is completed for system identification. The cart and pendulum are hung vertically down to perform a "free decay" test.

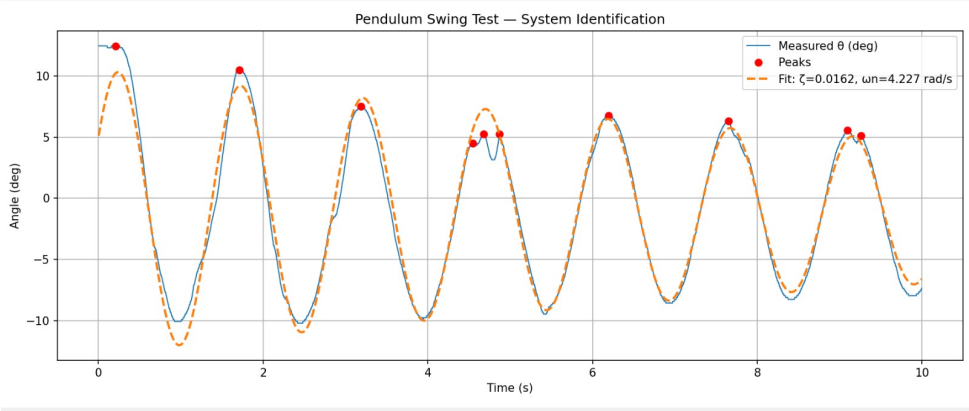


Figure 27: Swing Test

As seen in Figure 27, the damping ratio $\zeta$ is 0.016, obtained from a logarithmic decrement of $\delta \approx 0.099$ over 7 periods. The blue line is the noisy measured angle which we fitted directly to a second-order underdamped model shown in orange. This indicates non-negligible friction in the system, not meeting the passing criteria of $\zeta \leq 0.01$. However, we believe this result is not reflective of the system's friction, but rather the physical conditions of the swing test. The robot was manually held in the air, as such, a higher $\zeta$ value may have occurred due to human error in empirical noise during the experiment. This also explains the minor oscillations seen on the fourth and seventh peak. Unfortunately, we did not have the opportunity to retry the swing test in better conditions like placing the robot upside down between two surfaces. We acknowledge that this result may have upstream effects, particularly with the LQR controller.

## 6.2 Electrical Design

Table 15: Hardware Interface and Wiring Details of the Inverted Pendulum System

| **Module** | **Signal** | **Arduino Pin / Shield Port** | **Signal Type / Level** | **Notes** |
|---|---|---|---|---|
| Pendulum Encoder | Channel A | D3 | Digital Interrupt (TTL) | AS22 Kit Encoder A channel, 5 V logic, measures pendulum angle $\theta$ |
| Pendulum Encoder | Channel B | D5 | Digital Interrupt (TTL) | AS22 Kit Encoder B channel, 5 V logic, quadrature phase |
| Pendulum Encoder | Index (optional) | — | Digital (TTL) | Optional homing pulse |
| Cart Encoder | Channel A | D2 | Digital Interrupt (TTL) | Pololu 4882 motor encoder A channel, 48 CPR, measures cart position $x$ |
| Cart Encoder | Channel B | D4 | Digital (TTL) | Pololu 4882 motor encoder B channel, 48 CPR, quadrature phase |
| Motor Driver Shield | $I^2C$ SDA | A4 | $I^2C$ (3.3–5 V) | Motoron M3S550 control interface |
| Motor Driver Shield | $I^2C$ SCL | A5 | $I^2C$ (3.3–5 V) | Motoron M3S550 control interface |
| Motor Driver Shield | Motor Supply | VIN | Power (1.8–22 V) | External motor power supply |
| Motor Driver Shield | GND | GND | Ground | Shared ground with Arduino |
| Motor Driver Shield | Motor Output 2 | Left Wheel Pair | DC Output | Two left wheels connected in parallel |
| Motor Driver Shield | Motor Output 3 | Right Wheel Pair | DC Output | Two right wheels connected in parallel |

### 6.2.1 Hardware Selection and System Components Justification

The inverted pendulum system relies on carefully chosen hardware components to achieve reliable closed-loop control, fast response, and high-precision sensing. Each component was selected based on its performance, interface compatibility, and suitability for prototyping.

**Microcontroller Selection: Arduino UNO R4** The Arduino UNO R4 was selected as the central microcontroller for its combination of high processing performance, interface flexibility, and compatibility with existing shields. The R4 features an ARM Cortex-M4 processor running at 48 MHz, providing significantly higher computational throughput compared to the classic Arduino UNO R3, which uses an 8-bit ATmega328P at 16 MHz. The increased clock speed enables precise timing of encoder interrupts,

high-frequency control loops, and real-time calculation of control laws without introducing latency that could destabilize the inverted pendulum. In addition, Arduino UNO R4 provides wifi function to help design kill switch wirelessly. This helps improve levels of security for our system.

Compared with the Arduino Giga, which offers even higher processing power and additional peripherals, the UNO R4 provides sufficient computational resources for the current system while maintaining smaller form factor, lower power consumption, and full compatibility with standard Arduino shields such as the Pololu M3S550 motor driver. The R4's multiple external interrupt-capable pins are particularly advantageous for handling independent quadrature encoder channels for both the pendulum and cart wheels. Additionally, the R4 maintains the standard Arduino IDE and peripheral library support, simplifying software development and rapid prototyping.

In summary, the UNO R4 strikes an optimal balance between processing capability, interface availability, power efficiency, and prototyping convenience, making it an ideal choice for precise, low-latency control of the inverted pendulum system.

**Pendulum Encoder: Broadcom AS22 Optical Encoder** The AS22 rotary optical encoder was selected for its high resolution (1024 counts per revolution) and TTL logic output, providing precise angular position feedback of the pendulum shaft. Its digital quadrature channels support fast, interrupt-driven sampling on the Arduino, minimizing latency in the measurement of rapid pendulum motions. The encoder's compact size and straightforward wiring make it suitable for direct integration with the pendulum assembly.

**Cart Encoder: Pololu 4882 Quadrature Encoder** The cart position is measured by the Pololu 4882 encoder mounted on the wheel shaft, which delivers 48 counts per motor shaft revolution. This encoder was selected for its reliability, robustness under mechanical stress, and ease of integration with Arduino digital pins. The quadrature output allows both position and direction detection, providing high-fidelity real-time feedback for the cart motion.

**Motor Driver: Pololu M3S550 Motor Shield** The Pololu M3S550 motor driver shield was chosen for its ability to drive multiple brushed DC motors in parallel with independent output channels, supporting the two-by-two wheel configuration of the cart. It is controlled via an $I^2C$ interface at 3.3–5 V logic levels, allowing precise motor voltage regulation from the Arduino. The shield supports an external supply up to 24 V and high continuous current, ensuring sufficient torque for fast cart maneuvers, while the shared ground with the Arduino preserves signal integrity.

**Motors: Brushed DC Motors** Four small brushed DC pololu low power motors were selected for their simplicity, ease of control, and sufficient torque to move the cart while stabilizing the pendulum. They are arranged in a two-by-two configuration to distribute load and improve traction. The motors are driven in parallel pairs to simplify wiring and ensure balanced actuation forces between left and right wheels.

**Design Considerations** All components were chosen to minimize latency, maximize measurement accuracy, and ensure robust actuation. Encoder signals are routed with short, shielded wiring to reduce noise, and power traces are sized for peak currents. Decoupling capacitors are used to suppress voltage spikes, and shared grounding ensures consistent reference for both digital and power signals. This combination of component selection and careful system integration guarantees stable, low-latency closed-loop control suitable for experimental prototyping and rapid iteration.

### 6.2.2 Overview and System Topology

The inverted pendulum system is composed of two main functional blocks: the sensing subsystem and the actuation subsystem. The sensing subsystem captures the pendulum angular position $\theta$ and the cart linear position $x$ through high-resolution quadrature encoders, while the actuation subsystem applies control forces via four brushed DC motors arranged in a two-by-two wheel configuration. All subsystems are integrated with the Arduino UNO R4, which implements closed-loop feedback control. Figure 28 provides a complete schematic, showing all signal routing, power distribution, and grounding.

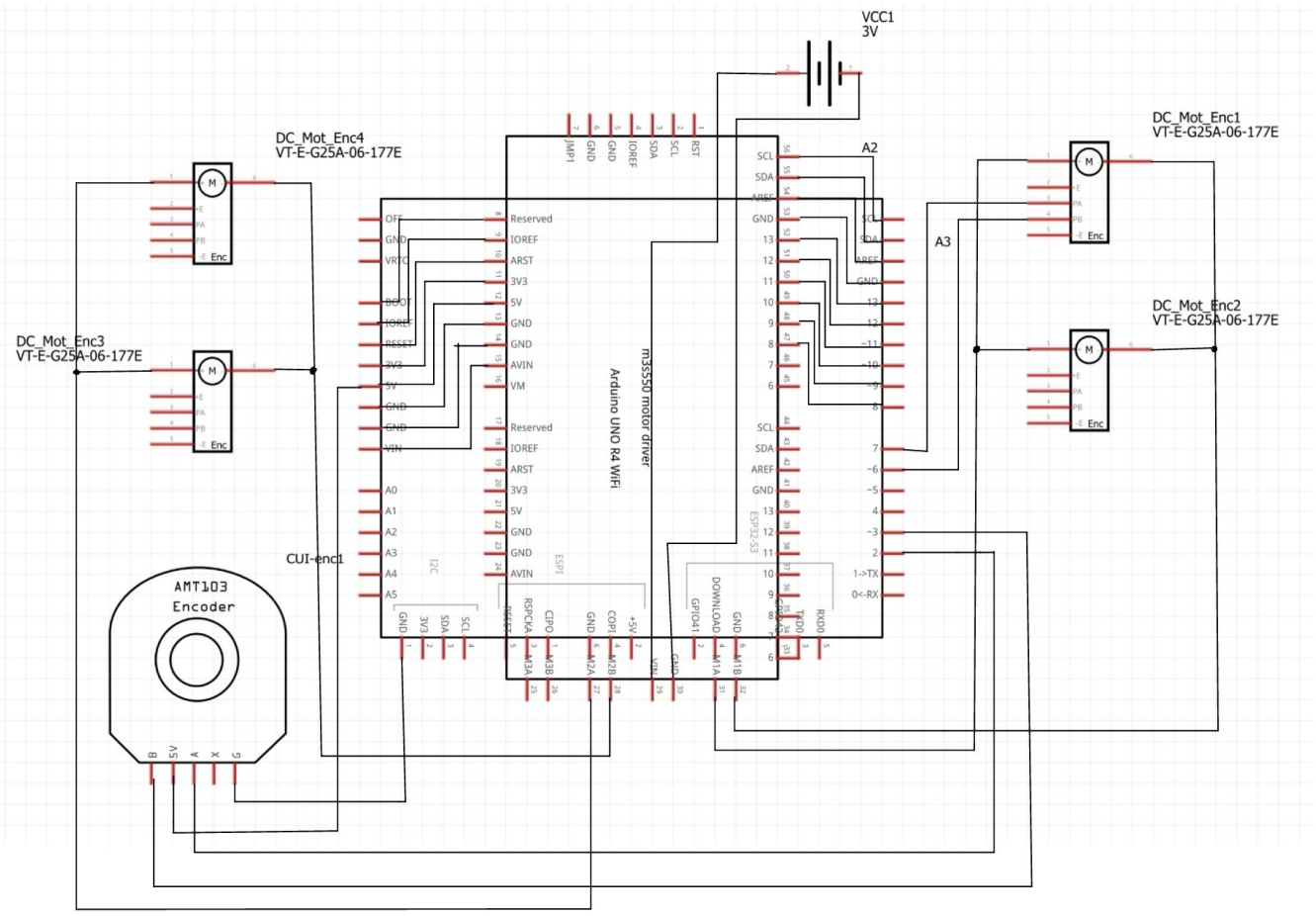


Figure 28: Electrical Schematics for the System

### 6.2.3 State Sensing

The system relies on two high-resolution quadrature encoders to obtain the full state vector necessary for feedback control.

A rotary optical encoder mounted on the pendulum shaft provides angular position $\theta$ with a resolution of 1024 counts per revolution. The encoder outputs two digital quadrature channels (A and B) and an optional index pulse, operating at 5 V TTL logic levels. Channels A and B are connected to Arduino digital interrupt pins D3 and D5, allowing accurate and low-latency detection of angular changes. Interrupt-driven reading ensures precise capture of fast pendulum perturbations, which is critical for closed-loop stabilization.

A wheel-mounted quadrature encoder measures the cart position $x$ through pins D2 and D4, with D2 configured as an external interrupt. The encoder provides 48 counts per motor shaft revolution, with effective wheel resolution determined by the gear ratio. This configuration enables high-frequency sampling of cart motion, ensuring real-time position and velocity feedback for the control loops.

### 6.2.4 Actuation Subsystem

Four brushed DC motors provide the actuation necessary for pendulum stabilization through cart motion. The left and right wheel pairs are electrically connected in parallel, with left wheels connected to Motor Output 2 and right wheels to Motor Output 3 of the Pololu M3S550 motor driver shield. The driver receives an external 12 V supply and shares a common ground with the Arduino to maintain a consistent voltage reference. Control commands are transmitted over the $I^2C$ interface at 3.3–5 V logic levels, allowing precise voltage regulation for motor torque control.

### 6.2.5 Signal and Power Integrity

The electrical design emphasizes minimizing propagation delays and preserving signal fidelity. All encoder signals are routed through short, shielded paths to the Arduino to reduce noise susceptibility. Decoupling capacitors are installed on power rails to suppress voltage spikes, and current traces are dimensioned to accommodate peak motor currents without significant voltage drop. High-current motor switching is a potential source of electromagnetic interference (EMI); mitigation measures include twisted pair wiring, shielding, and proper grounding.

### 6.2.6 Timing and Latency Analysis

Accurate timing of sensor measurements and control commands is critical for closed-loop stabilization. The estimated delays along key signal paths are summarized in Table 16.

Table 16: Estimated Signal Path Delays in the Inverted Pendulum System

| **Signal Path** | **Estimated Delay** |
|---|---|
| Optical Encoder $\theta \rightarrow$ Arduino D3/D5 | $\sim 5\,\mu$s |
| Motor Encoder $x \rightarrow$ Arduino D2/D4 | $\sim 5\,\mu$s |
| Arduino Uno R4$\rightarrow$ Motor Driver ($I^2C$) | $\sim 50\,\mu$s |

These delays are within acceptable limits for the control loop frequency of 200–500 Hz, ensuring stable and responsive pendulum actuation.

### 6.2.7 Safety and Reliability Considerations

To enhance system robustness, several measures are implemented:

- Shared ground between Arduino and motor driver to avoid potential differences and signal errors.
- Short, shielded signal routing for encoder channels to minimize EMI impact from motor switching.
- Power supply decoupling and sizing to prevent voltage drops during high-torque maneuvers.
- Optional fuses or PTC resistors on motor power lines for overcurrent protection.

### 6.2.8 Scalability and Modularity

The design allows future expansion, including:

- Additional sensors interfaced via spare Arduino pins or $I^2C$ channels.
- Motor replacement or upgrading by using different Motoron shield outputs.
- Modularity in encoder selection, allowing higher-resolution or alternative types without major redesign.

## 6.3 Control and Software Implementation

### 6.3.1 Motor Driver Test

The Pololu M3S550 triple output motor shield was tested independently to verify the correct direction control and PWM modulation. Each output channel was driven using open-loop PWM signals from the MCU and monitored for the motor response. During the test, the driver can respond consistently to directional commands across all three outputs. There is no observable instability or unintended coupling between channels

**Test Setup:** Each motor channel was driven using open-loop PWM commands in the full range of ([-800, 800]), corresponding to the driver's maximum duty cycle limits. A DC motor with an attached encoder was connected to each output. Supply voltage and current were monitored via a regulated 12V power supply, while output voltages were measured using a multimeter.

**Performance Metrics:** The evaluation focused on:

- PWM-to-speed response and linearity
- Direction switching behaviour
- Channel isolation under single-channel actuation

**Results:** A monotonic and approximately linear relationship between PWM input and measured rotational speed was observed across the full control range.

The driver produced stable bidirectional control, with correct motor rotation observed for both positive and negative PWM inputs. Direction switching was consistently achieved with a fast response time.

Under operation at maximum command (PWM = 800), the supply current reached approximately 0.75–0.8A at 12V. Voltage measurements at the motor terminals indicated an effective output of approximately

5–6,V per channel during operation, consistent with PWM modulation and load sharing characteristics.

To assess channel isolation, a single motor was actuated while the remaining channels were held at zero input. No measurable voltage fluctuation (within ±0.1,V) or unintended motion was detected in inactive channels, indicating negligible cross-channel interference.

Overall, the motor driver demonstrates stable, repeatable, and well-isolated performance, suitable for reliable multi-actuator control in the system.

### 6.3.2 Motor Test

Each 9.7:1 DC motor was tested using the validated motor shield to assess its behavior. Step changes in PWM input were applied to evaluate both the directional response and the start-up characteristics. The motors illustrated smooth rotation in both directions without stalling or excessive vibration. This demonstrated reliable actuator performance.

**Test Setup:** Step inputs in PWM command were applied over the range ([-800, 800]) to excite the motor dynamics. Rotational speed was measured using an encoder, while supply current and voltage were monitored via a 12V power supply.

**Performance Metrics:** The following characteristics were assessed:

- Start-up response and minimum actuation threshold(dead-zone)
- Speed response to step PWM inputs
- Directional symmetry and smoothness of rotation

**Results:** The motors exhibited consistent and smooth bidirectional rotation across the tested PWM range. A minimum actuation threshold was observed at approximately ($PWM \approx 80$), below which no sustained motion occurred due to static friction.This helps us to set the dead-zone in latest codes for real system. Making system more robust.

Step response analysis showed that the motors reached steady-state speed within approximately 80–120ms, with no significant overshoot or oscillatory behaviour. The steady-state speed increased monotonically with PWM input, indicating predictable actuator behaviour.

Under nominal operating conditions, current draw remained below 0.8A, consistent with expected motor characteristics. No excessive vibration or irregular motion was observed.

Overall, the motors demonstrate stable, responsive, and repeatable performance suitable for closed-loop control implementation.

### 6.3.3 Wheel Encoder Test

The wheel encoder was evaluated to verify its counting accuracy, directional consistency, and suitability for distance estimation.

**Test Setup:** The encoder was interfaced using an interrupt-based decoding scheme. The motor was driven at a constant low speed (PWM = 400) to ensure stable motion. Encoder counts were recorded through the serial interface at regular intervals while the robot travelled a measured distance of 1m. Both forward and reverse motions were tested.

**Performance Metrics:**

The evaluation focused on:

- Count accumulation over a known travel distance
- Directional consistency (sign of counts)
- Measurement smoothness and absence of count loss

```
Encoder count: 22
Encoder count: 100
Encoder count: 395
Encoder count: 866
Encoder count: 1475
Encoder count: 2001
Encoder count: 2394
Encoder count: 2667
Encoder count: -99
Encoder count: -263
Encoder count: -627
Encoder count: -937
Encoder count: -1356
Encoder count: -1399
Encoder count: -2030
Encoder count: -2649
```

Figure 29: Serial Output During Testing Wheel Encoder

**Results:** During forward motion, the encoder count increased monotonically from zero to approximately 2667 counts over a travel distance of 1m, corresponding to an effective resolution of approximately 2667 counts per metre. When the direction of rotation was reversed, the count decreased consistently, confirming correct quadrature phase interpretation.

The recorded count sequence showed smooth and continuous increments with no abrupt jumps or missing pulses, indicating reliable interrupt handling and signal integrity. Minor fluctuations in incremental step size were observed due to non-uniform motor speed, but no systematic error was detected.

Based on these results, the encoder provides stable and consistent measurements suitable for feedback control. Given the consistent behaviour observed, a single encoder is considered sufficient for wheel motion estimation in the system.

### 6.3.4 Optical Encoder Test

The optical encoder measuring the pendulum angle was tested by manually perturbing the pendulum around the upright position. It measured the angle signal which provide a response to angular displacement. It can also provide sufficient resolution for feedback control.

**Test Setup:** The pendulum was manually perturbed around the upright equilibrium position, and encoder counts were recorded through the serial interface. Angular displacements were estimated using known reference positions, including a 90° rotation from the vertical.

**Performance Metrics:** The evaluation focused on:

- Count variation with respect to angular displacement
- Resolution and sensitivity near the upright position
- Measurement smoothness and repeatability

```
Encoder count: -386
Encoder count: -384
Encoder count: -355
Encoder count: -280
Encoder count: -193
Encoder count: -135
Encoder count: -104
Encoder count: -21
Encoder count: 46
Encoder count: 107
Encoder count: 207
Encoder count: 264
Encoder count: 305
Encoder count: 342
Encoder count: 358
Encoder count: 366
Encoder count: 368
Encoder count: 373
Encoder count: 371
Encoder count: 328
```

Figure 30: Serial Output During Testing Optical Encoder

**Results:** A monotonic relationship between angular displacement and encoder counts was observed. A rotation of approximately 90° corresponded to an encoder change of roughly 386 counts, yielding an

effective resolution of about 4.29 counts per degree.

The encoder output varied smoothly with angular motion, with no observable discontinuities or missed counts. Near the upright position, small perturbations produced measurable count changes, indicating sufficient sensitivity for stabilization tasks.

Repeated manual tests showed consistent count responses for similar angular displacements, suggesting good repeatability. Overall, the encoder provides reliable and sufficiently high-resolution measurements suitable for pendulum angle feedback in control applications.

### 6.3.5 PID Control Algorithms

To stabilize the inverted pendulum while controlling its cart position, a two-layer cascade PID control structure is implemented. The design separates fast-angle stabilization (inner loop) from slower position tracking (outer loop), allowing each loop to operate on its suitable timescale.

The outer loop regulates the horizontal position of the cart. The position error is calculated as

$$x_{\text{err}} = x_{\text{target}} - x,$$

and fed into a PID controller to generate a desired pendulum angle:

$$\theta_{\text{target}} = K_p^x x_{\text{err}} + K_i^x \int x_{\text{err}}\, dt + K_d^x \frac{dx_{\text{err}}}{dt}.$$

To ensure safe operation, the requested lean angle is constrained to a maximum value $\theta_{\max}$, preventing excessive pendulum tilt.

The inner loop stabilizes the pendulum around the upright position. The angle error is defined as

$$\theta_{\text{err}} = \theta_{\text{target}} - \theta,$$

where $\theta$ is the current pendulum angle measured by the rotary encoder. The PID controller then computes the motor command:

$$u = -(K_p^\theta \theta_{\text{err}} + K_i^\theta \int \theta_{\text{err}}\, dt + K_d^\theta \frac{d\theta_{\text{err}}}{dt}),$$

with anti-windup and output saturation applied to limit the control signal within the motor's safe operating range. Both left and right motors receive the same command to maintain balance.

This cascade architecture improves system stability by allowing the inner loop to react immediately to angular deviations, maintaining balance, while the outer loop adjusts the pendulum angle to drive the cart toward the desired position. Our design effectively decouples fast pendulum stabilization from slower cart position control.

Both PID loops are implemented in discrete-time at a 200 Hz control rate. Encoder counts are converted to physical units (radians for the pendulum, meters for the cart), and derivative terms are computed using discrete differences. The code includes safety limits for pendulum angle and cart position to prevent unsafe operation.

### 6.3.6 LQR Control Algorithm

To stabilize the inverted pendulum and regulate the cart position, a linear-quadratic regulator (LQR) is implemented. This method computes the optimal state feedback control law based on a linearized model of the system, directly mapping system states to motor commands.

The system state vector is defined as

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix},$$

where $x$ is the cart position, $\theta$ the pendulum angle, and the dots denote time derivatives. The LQR control law is a linear feedback:

$$u = -\mathbf{K}\mathbf{x} = -\big(K_1 x + K_2 \dot{x} + K_3 \theta + K_4 \dot{\theta}\big),$$

where $\mathbf{K} = [K_1, K_2, K_3, K_4]$ is the gain vector.

The LQR gains are initially computed from a linearized simulation model by selecting appropriate weighting matrices $Q$ and $R$ in the cost function:

$$J = \int_0^\infty \big(\mathbf{x}^{\mathrm{T}} Q \mathbf{x} + R u^2\big) dt.$$

After simulation, the gains are fine-tuned experimentally on the physical system to account for unmodeled

dynamics and actuator nonlinearities. The final gain vector used in the embedded controller is

$$\mathbf{K} = [-8.236,\ -5.813,\ 55.898,\ 12.367].$$

The controller is implemented in discrete time at a 500 Hz sampling rate ($dt = 0.002$ s). Encoder counts are converted to physical units:

$$\theta = \frac{\text{pendulum counts}}{\text{counts per rad}}, \quad x = \frac{\text{cart counts}}{\text{counts per meter}}.$$

Derivative terms are approximated using finite differences:

$$\dot{\theta} \approx \frac{\theta - \theta_{\text{prev}}}{dt}, \quad \dot{x} \approx \frac{x - x_{\text{prev}}}{dt}.$$

To ensure safe operation, the control signal $u$ is scaled to motor commands with saturation:

$$u_{\text{motor}} = \text{constrain}\big(U_{\text{to\_speed}} \cdot u, -\text{MAX\_MOTOR\_SPEED}, \text{MAX\_MOTOR\_SPEED}\big),$$

and a dead zone is added to overcome static friction. The system includes safety checks on pendulum angle ($|\theta| < \theta_{\text{limit}}$) and cart position ($|x| < x_{\text{limit}}$), stopping the motors if limits are exceeded.

# 7 Benchmarking

## 7.1 Evaluation A: Disturbance Rejection

Table 17 summarises the settling time for each disturbance type under both control algorithms.

Table 17: Disturbance rejection results for both controllers.

| Disturbance | Controller | Settling Time (s) | Drop? (Y/N) |
|---|---|---|---|
| Small tap (pendulum mass) | PID | 3.0 | N |
| | LQR | 2.3 | N |
| Large tap (pendulum mass) | PID | 5.3 | N |
| | LQR | 3.6 | N |
| Shove (cart chassis) | PID | 2.3 | N |
| | LQR | 1.9 | N |

The real-world results for these disturbance conditions are shown in Figure 31.

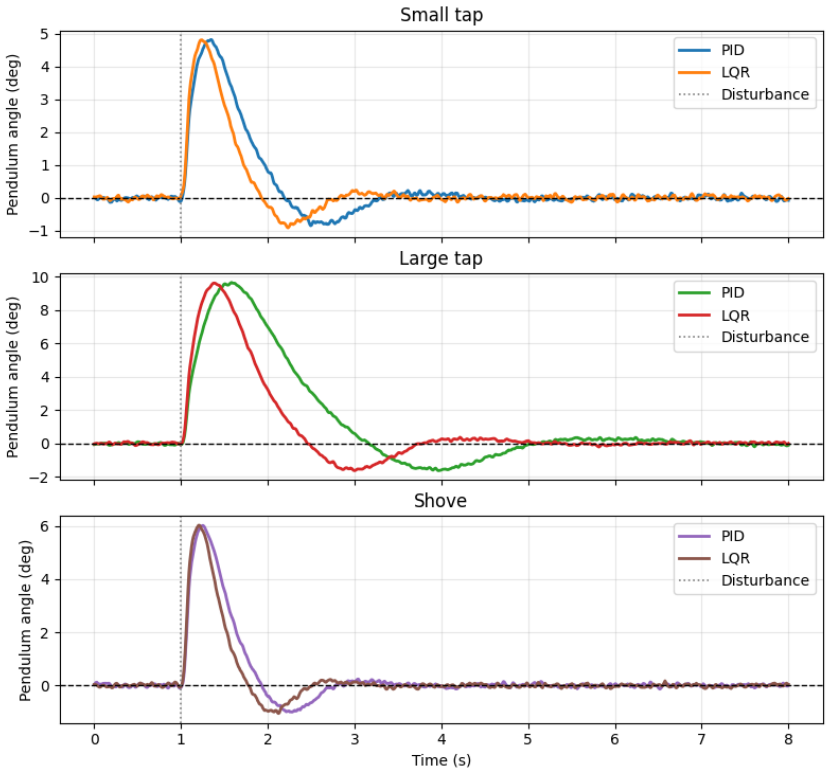


Figure 31: Real-world pendulum angle response under small and large disturbances for both controllers

The disturbance rejection performance of our system was evaluated under three different types of external perturbations: a small tap applied to the pendulum mass, a larger tap on the pendulum and a horizontal shove applied to the cart chassis. For each case, the settling time and the system stability(whether the pendulum fell)were recorded for both PID and LQR controllers.

Overall, both controllers were able to maintain stability across all tested disturbances, with no instances of system failure observed. However, clear differences in performance were observed in the settling time.

For the small tap disturbance, both controllers responded effectively, with the LQR controller achieving a faster settling time(2.3s) compared to hte PID controller(3.0s). This is expected, as the system remains close to the linearised operating point, where both control strategies perform well. However, LQR benefits from optimal state feedback.

As the disturbance magnitude increased(large tap on the pendulum), the difference in performance became more pronounced. The PID controller required significantly more time to stabilise the system(5.3s), while the LQR controller recovered more quickly(3.6s). The behaviour highlights the limitation of PID under larger deviations, where non-linear effects become more significant and tuning trade-offs reduce responsiveness.

What was worthy noticing here was that for the shove disturbance applied to the cart chassis, the settling times(PID:2.3S, LQR:1.9s) were lower than those observed in the large tap case. This can be explained by the nature of the disturbance. While the cart position is directly affected the pendulum angle deviation remains relatively small, allowing the controller to compensate more efficiently. In contrast, direct perturbations to the pendulum introduce larger angular deviation, which are more challenging to correct.

In all scenarios, the LQR controller consistently outperformed the PID controller, demonstrating faster settling time and more efficient recovery. This is because of its full-state feedback design and optimal gain selection, which enable better handling of coupled system dynamics. This also matches the conclusion that we drew in the simulation benchmark.
To conclude, while both controllers are capable of stabilising the system under moderate disturbances, the LQR controller provides superior performance, particularly as disturbance magnitude increases. This makes it a more suitable choice for applications requiring robust and rapid stabilisation of the inverted pendulum system.

## 7.2 Evaluation B: Deep Fall Recovery

The maximum recovery angle was tested incrementally starting from 5° and increasing in 5° steps. Table 18 shows the highest angle from which each controller successfully stabilised the pendulum, in both simulation and the real world.

Table 18: Maximum successful recovery angle for both controllers.

| Controller | Simulation (deg) | Real-world (deg) |
|---|---|---|
| PID | 50 | 15 |
| LQR | 52 | 20 |

Figure 32 shows the pendulum angle over time during a representative recovery attempt at the maximum successful angle.
The maximum successful recovery angle provides an indication of how far the pendulum can deviate from the upright equilibrium position while still being stablised by the controller. As shown in Table 18, the LQR controller achieves a significantly larger recovery angle (52°) compared to the PID controller (35°) in simulation.

This difference highlights the fundamental advantage of LQR control in handling larger deviations from the linearisation point. Since LQR is designed using a full-state feedback approach, it accounts for the coupled dynamics between the cart position and pendulum angle more effectively. As a result, it is capable of generating more appropriate control actions even when the system state moves further away from

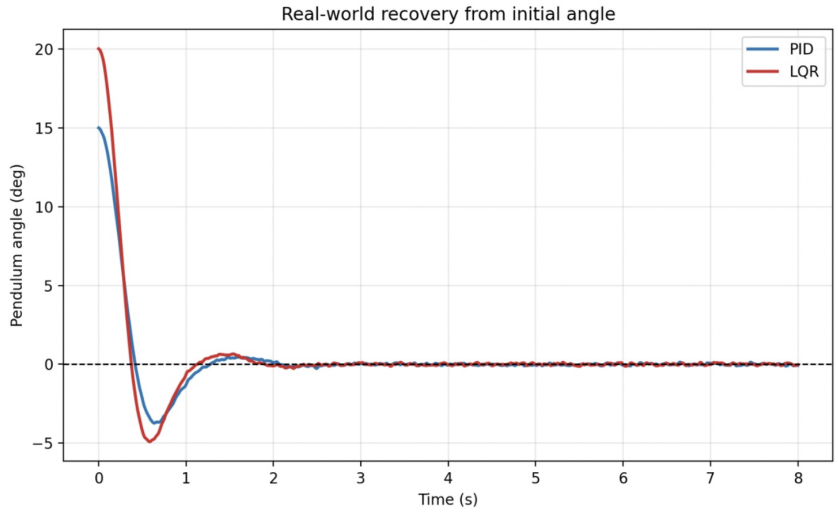


Figure 32: Pendulum angle during deep fall recovery at maximum successful starting angle

equilibrium.

In contrast, the PID controller exhibits a more limited recovery range. This is primarily due to its reliance on error-based feedback, which does not explicitly consider the system's internal dynamics. As the pendulum angle increases, the nonlinear effects become more pronounced, and the PID controller struggles to produce sufficiently strong and coordinated corrective actions, leading to failure beyond approximately 35°.

The real-world results in Figure 32 show a significant reduction in the maximum recovery angle for both controllers compared to simulation. The PID controller is able to recover up to 15° while the LQR achieves a slightly higher value of 20°.

This reduction is expected due to practical limitations such as actuator saturation, sensor noise and insufficient motor acceleration. In practical, the mortor torque is also insufficient to generate the large corrective forces required at higher pendulum angle, leading to early failure in both control strategies.

Although the LQR still outperforms the PID controller, the performance gap is noticeably smaller than in simulation. This can be caused by the fact that LQR is derived from a linearised model based on small angle approximation. As the pendulum deviates further from the upright position, these assumptions become less accurate and reduce the effectiveness of the controller.

Overall. both controllers show limited recovery capability in the real system, highlighting the impact of hardware constraints and non-linear effects.

## 7.3 Evaluation C: Sprint and Stop

Table 19 summarises the completion time and final stopping accuracy for both controllers across two attempts each.

Table 19: Sprint and stop results over two attempts per controller.

| Controller | Attempt | Time (s) | Final Position Error (cm) | Stable? (Y/N) |
|---|---|---|---|---|
| PID | 1 | 8.8 | 6 | Y |
| | 2 | 9.5 | 5 | Y |
| LQR | 1 | 6.6 | 2 | Y |
| | 2 | 7.1 | 2 | Y |

Figure 33 shows the simulated cart position over time during the both attempts for the LQR and PID controller.

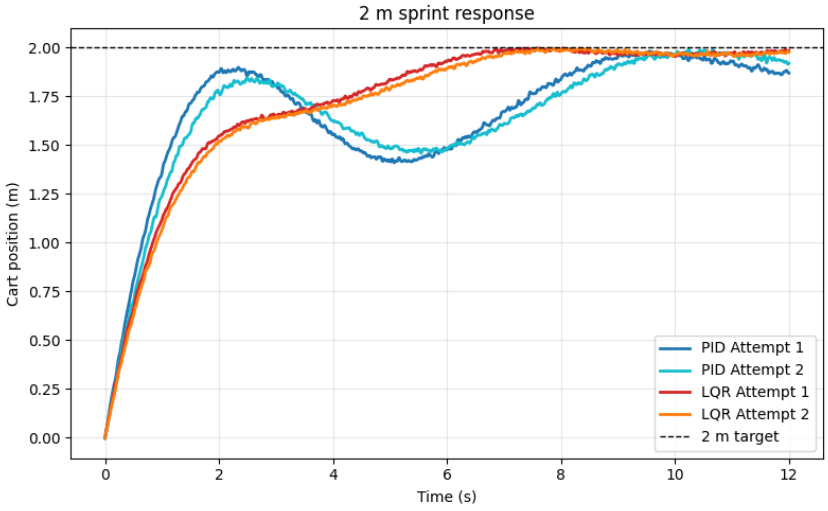


Figure 33: Cart Position for 2 Meter Sprint

The sprint test evaluates the ability of the system to reach a target position efficicently while maintaining stability. The results show that both controllers successfully completed all attempts with no instability or failure observed.

In terms of response speed, the LQR controller consistently outperformed the PID controller. The LQR achieved settling times of 6.6 s and 7.1 s, whereas the PID controller required significantly longer durations of 8.8 s and 9.5 s. This indicates that LQR is more effective at driving the system towards the desired position in a timely manner.

For steady-state accuracy, LQR also outperforms PID where the final position error remained consistently low at approximately 2cm across both attempts. In contrast, PID demonstrated larger residual errors of

5-6cm, indicating less precise convergence to the target position.

Moreover, PID showed slightly greater variability between repeated trials, particularly in settling time. LQR maintained more consistent behaviour. This difference highlights the robustness of LQR.

The overall settling time is relative longer for both controllers in practice, which is likely due to conservative tuning and practical constraints in the real system (e.g. actuator limits and the need to avoid excessive oscillations). As a result, the system prioritises stability and smooth convergence over aggressive responses.

To conclude, while both controllers are capable of completing the sprint task reliably, LQR provides clear advantages in both speed and accuracy, making it a more effective solution for position control in this system.

## 7.4 Evaluation Conclusion

Table 20 provides a summary comparison of both controllers across all three evaluations.

Table 20: Summary comparison of controller performance across all evaluations.

| Metric | PID | LQR | Winner |
|---|---|---|---|
| Settling time – small disturbance (s) | 3.0 | 2.3 | LQR |
| Settling time – large disturbance (s) | 5.3 | 3.6 | LQR |
| Max recovery angle (deg) | 15 | 20 | LQR |
| Sprint time (s) | 9.15 | 6.85 | LQR |
| Final position error (cm) | 5.5 | 2 | LQR |

Table 20 summarises the performance of the PID and LQR controllers across all evaluation metrics, including disturbance rejection, deep fall recovery, and sprint and stop performance. It is evident that the LQR controller consistently outperforms the PID controller in all tested aspects.

In terms of disturbance rejection, the LQR controller achieves faster settling times for both small and large disturbances. While the difference is moderate for small perturbations, it becomes more significant as the disturbance magnitude increases, indicating that LQR handles more challenging conditions more effectively.

The maximum recovery angle further highlights this advantage. Although both controllers experience a reduction in performance in the real system due to practical limitations, the LQR controller is still able to stabilise the pendulum from a larger initial deviation (20°) compared to the PID controller (15°). This

suggests a larger region of attraction and improved robustness to initial disturbances.

For the sprint task, the LQR controller again demonstrates superior performance, achieving a significantly shorter completion time (6.85 s vs 9.15 s) and a lower final position error (2 cm vs 5.5 cm). This indicates not only faster response but also higher steady-state accuracy. In contrast, the PID controller shows slower convergence and larger residual errors, likely due to oscillatory behaviour and less effective handling of coupled system dynamics.

Overall, the results confirm that the LQR controller provides a more efficient and robust control strategy for the inverted pendulum system. Its full-state feedback formulation enables better coordination between cart motion and pendulum stabilisation, resulting in improved performance across all evaluation criteria. While the PID controller remains a viable and simpler alternative, it is clearly outperformed by LQR in both dynamic response and accuracy.

## 7.5 Ethical Considerations

A private company seeks to acquire our inverted pendulum design for use in a weapon system to be sold to a government and deployed in battlefields abroad, with no control retained by the original designers over its use.

Under Kant's duty-based ethics, this is impermissible regardless of outcome: human life must never be treated as a means to an end and contributing to a system designed to cause harm violates that duty unconditionally. According to deontologists, it is one's duty to not hurt others and prevent harm to human life. Using our design for matters of warfare directly infringes upon Kant's principle of following moral duties.

Utilitarianism reaches the same conclusion through different reasoning, the commercial and strategic benefits are outweighed by the risk of widespread casualties and instability, particularly given the complete absence of oversight over deployment. Although, if we consider a situation where other weaponry and systems are already in place for offensive purposes, then one could argue that the best solution according to utilitarianism would be to sell the inverted pendulum design. By giving the design to the government, we may lessen the consequences and casualties. It can be argued that the sacrifice of the opposition will result in minimum suffering possible if, for example, they had a smaller population. The "greater good" in this scenario would be the least number of deaths and casualties, considering that death on some part is inevitable – this would be harming few for the greater good.

Aristotle's virtue ethics similarly can argue both sides as the "golden mean" is subjective. It can reject the transfer, as knowingly enabling harm with no accountability falls far outside the virtuous mean expected of a responsible engineer. It is also a moral decision to not be aiding in the destruction of cities and loss of human life. However, as virtue ethics lacks clear rules, one can argue that the ultimate goal is to save lives, which may have the best chance of success by giving the design away. The virtues for this argument would be that the golden mean balances naïvety and hostility.

Locke's rights-based theory holds that every individual has a fundamental right to life and supplying a weapon over which the designers retain no control offers no protection to those whose rights may be violated by its use. Each of the four theories approach the problem differently, argued in a certain way, they converge on the same conclusion: accepting this acquisition would be ethically unjustifiable. However, both utilitarianism and virtue ethics can also assert that it would be ethical to sell the design for use in defence.

# 8 Conclusion

The inverted pendulum project demonstrated the complete workflow from modelling and simulation through to hardware implementation, control, and experimental validation. Over the course of the project timeline, we implemented swing up and stabilisation controllers, identified key system parameters via swing tests, and iteratively refined both the CAD and the physical prototype to meet the performance requirements.

On the hardware aspect, the mechanical and electrical design evolved significantly as practical issues emerged during integration. In particular, we decided to change the wheels to one of a larger diameter, which allowed the cart to travel at a higher speed with more distance. This improved tracking performance and required minimal updates to the CAD and mounting hardware. We also moved from an initial four motor configuration to a two motor design due to power distribution constraints and current limiting issues when changing from the power pack to the battery, trading some redundancy for a simpler and more reliable drivetrain. It would have been beneficial if we had the time to remove the two redundant motors and replaced it with free-rolling wheels to reduce the unnecessary mass of the robot.

We decided to place most of the electrical components underneath the chassis in order to make the layout compact and have space for the v-shaped holder. However, this also reduced accessibility to the MCU and complicated debugging and re-wiring during testing. This became a large hindrance and is now a lesson on serviceability in future designs. Additionally, the jittering and high-frequency vibrations of the

robot during operation caused several screws to loosen over time, which occasionally affected alignment and introduced extra friction; this motivated the use of locking nuts, thread-locking compounds, and more robust mounting strategies.

From a control and identification perspective, the swing tests showed a lightly damped pendulum but with more friction than the specification required. This influenced our achievable performance and tuning choices. Despite these limitations, the final system was able to stabilise the pendulum using LQR with minimal to none steady-state error, reject step disturbances to both the pendulum top and the cart base, and complete a sprint of desired length. The project as a whole provided hands-on experience in bridging the gap between idealised simulation and models to real hardware, managing iterative redesign under time constraints, and applying MBSE and SysML methods to structure a multidisciplinary robotics system. Looking forward, further work could focus on reducing friction in the pivot, improving cable management and component accessibility, and exploring more advanced control strategies to enhance robustness and disturbance rejection.

# 9 Code

https://github.com/fifictional/SystemsEngineeringProject.git

# References


[1] D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar, "Application of a technique for research and development program evaluation," *Operations Research*, vol. 7, no. 5, pp. 646–669, 1959.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[4] J. L. Crassidis, F. L. Markley, and Y. Cheng, "Survey of nonlinear attitude estimation methods," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 12–28, 2007.

[5] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: Wiley, 2001.