COMP0216 Systems Engineering

Inverted Pendulum Project Report

Team 3
Aayan Islam – 24102520
Xavier Parker – 24077390
Peter Neville – 24014343
Helitha Cooray – 24161798

March 2026

Contents

1 Introduction

1.1 Project Statement

Our goal is to design, simulate and construct an autonomous system that is capable of balancing an inverted pendulum, either while being stationary, or in motion. The build cycle we plan to follow for the next 9 weeks will be such that it adheres to the **Model-Based Systems Engineering (MBSE) V- Model**. The final system, simply put, will consist of two parts: the cart and the pendulum. The cart will always be kept on a flat surface and will be battery powered. The primary objective of our system is to achieve a stable equilibrium ($\theta = 0$) for the pendulum while controlling the cart's position.

1.2 Evaluation Criteria

Evaluation A: Disturbance Rejection (Robustness)

The system must maintain pendulum stability while the cart is stationary under external disturbances. Robustness is evaluated using small and large taps applied to the pendulum mass and a shove to the cart chassis. Performance is measured by settling time and the ability to prevent pendulum failure.

Evaluation B: Deep Fall Recovery

The system must recover the pendulum from an initial resting angle on the V-Holder and stabilize it in the upright position.

Evaluation C: Sprint & Stop Race

The cart must travel 2 m from rest and stop within a specified tolerance while maintaining pendulum stability. Performance is evaluated using a weighted score based on travel time and final stability.

Based on the evaluation criteria, we are able to adopt two approaches to completing the project:

1. **Prioritising Stability**: Focusing on the pendulum stability over limiting the cart motion would be ideal for Evaluations A and B, as we understand that lifting the pendulum from a low angle would require a robust control system and significant torque. Therefore, taking this approach would result in our mechanical design focusing on the actuators and a stable cart to maximize our recovery angle and resist external factors that might result in disturbances.
2. **Prioritising the Sprint**: Our target in this approach, if taken, will be to minimize the time the cart takes to complete the 2 meter sprint (Evaluation C). The design would possibly focus on lightweight materials and motors with a higher RPM to provide better acceleration. The control algorithm implemented would be aggressive to maintain the pendulum stability at a higher cart speed.

While we are aware that both of these can be attempted simultaneously, doing so could potentially hinder progress in tackling work related to all the evaluations. Due to the similarity in evaluations A and B, we decided to group them as seen above. We have

decided to pursue the first approach, as we think stability is a foundational requirement for all the tests. We believe this risk-averse strategy ensures compliance well, especially with the "Deep Fall" recovery requirement, which we identify as the highest technical risk due to the non-linear dynamics involved in swinging up from low angles.

1.3 Characteristics

- The system should be able to operate safely in the lab
- The whole build must be sustainable in material choice and use
- Must be autonomous
- The desired outcome must be achievable multiple times (good repeatability)

1.4 Functional Requirements

- The pendulum must stabilise **and** remain in that stable position
- The system must generate real-time data
- We should be able to vary the initial angle of the pendulum

1.5 Constraints

- Cart motion must be linear
- Sprint distance must be $2 \pm 0.1m$
- The pendulum must be free-moving
- Pendulum length should be within $0.6m - 1m$
- There must be a $50g$ mass as the payload

2 Project Planning

2.1 Morphological Chart

Sub-Function	Option 1	Option 2	Option 3	Option 4
Cart Mobility	**Four wheels**	Two tracks	Two wheels, two tracks	Single large omni-wheel
Pendulum Pivot	**Central mounted vertical ball bearing**			
Pendulum Length (cm)	**60**	75	100	
Adjustable V-Holder	grooves on main body	**Folding Mechanism**	Adjustable hinges	
Measuring angle (θ)	**Rotary Encoder**	Gyroscope	IMU	Potentiometer
Position Sensing	IMU	**Motor encoders**		
Chassis Material	Bamboo board	Plywood board	**Acrylic board**	PLA
Control Models	**PID**	**LQR**	MPC	Pole Placement
Microcontroller	**Arduino Uno R4**	ESP32	Arduino Mega	**Arduino Giga**

Table 1: Morphological Chart with Selected Design Path Highlighted

Our morphological chart systematically explores the design space across nine sub-functions, with each row representing the viable options that we considered for a certain design parameter. The **highlighted** options are the final design decisions, that we justified with a trade-off analysis against the evaluation criteria.

2.2 Mind Map

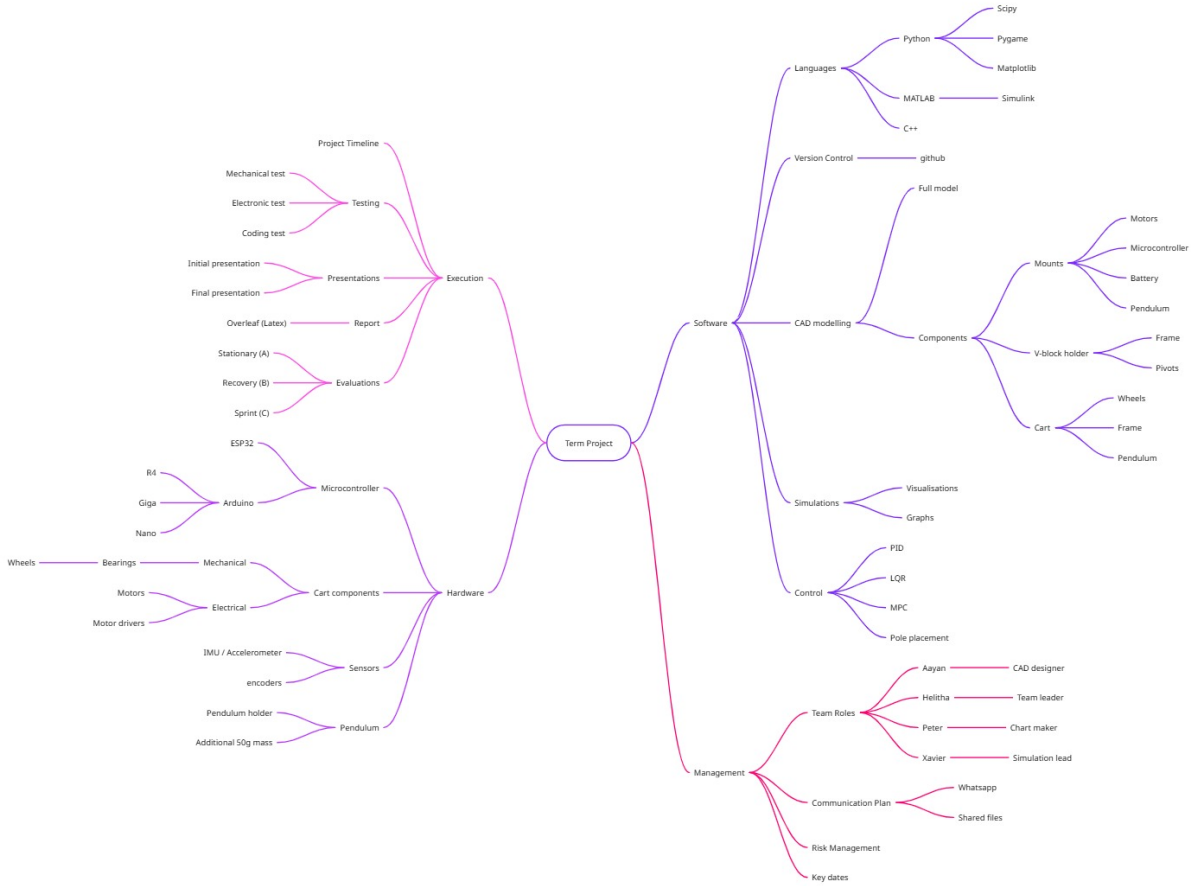


Figure 1: Project Mind Map

Our mind map decomposes the system into its primary domains: mechanical design, electronics, software, and project management . It served as the team's initial brainstorming artefact before more formal decomposition into the WBS, as seen in the following section.

2.3 Work-breakdown structure (WBS)

The Work-Breakdown Structure breaks down the project into 12 activities (A through L), each with deigned immediate predecessors that enforce a logical build order, similar to that of the **MBSE-V** Model. The structure reflects the following strategy: requirements capture (A) must precede all technical work; dynamic modelling (B) and simulation work (C, D) are completed before any physical fabrication begins; and controller implementation (I) is gated on both software readiness (D) and hardware availability.

ID	Activity	Immediate Predecessor
A	Requirements & System Specification	-
B	Dynamic Modelling	A
C	Simulation Environment Development	B
D	Controller Design (PID/LQR)	B, C
E	CAD Design	A
F	Electronics Architecture Design	A
G	Preliminary Presentation	B, C, D
H	Prototype Fabrication	E, F
I	Controller Implementation (Embedded)	D, H
J	System Integration	I
K	Testing and Tuning	J
L	Final Report and Demo	K

2.4 Risk-breakdown structure (RBS)

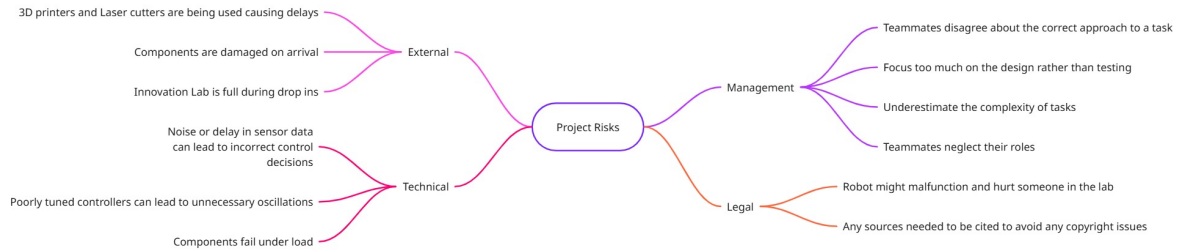


Figure 2: Risk Breakdown Structure

To understand all the threats to the project, the team has developed a RBS shown in figure 2. This diagram highlights the potential failures through four distinct domains. By decomposing risks into these specific categories, the team was able to identify not only standard engineering challenges, such as 'sensor noise' or 'components failing under load', but also logistical and compliance threats. This structured approach allows the team to pinpoint high-probability risks early and allocate slack time in the schedule to mitigate them.

2.5 A-B-M estimates, TE, and sigma

We gathered three-point **PERT** estimates for each activity listed in the WBS, considering **optimistic (a)**, **most likely (m)**, and **pessimistic (b)** scenarios for each task. Expected times are computed as

$$T_E = (a + 4m + b)/6$$

and variances as

$$\sigma^2 = [(b - a)/6]^2$$

The largest variance belongs to the **Activity K** (Testing and Tuning, $\sigma^2 = 1.361$) as we understand the inherent unpredictability of hardware debugging and controller tuning,

Activity	a	m	b	Expected Time	Variance
A	1	2	3	2.000	0.111
B	2	4	6	4.000	0.444
C	3	5	7	5.000	0.444
D	2	4	6	4.000	0.444
E	2	3	5	3.167	0.250
F	3	5	7	5.000	0.444
G	2	4	6	4.000	0.444
H	3	5	8	5.167	0.694
I	3	5	8	5.167	0.694
J	2	4	6	4.000	0.444
K	3	6	10	6.167	1.361
L	2	4	6	4.000	0.444

which is consistent with what we identified as a high–impact risk in the **FMEA**, which can be found further down. We used these estimates in the network analysis and probability of completion calculation.

2.6 Network

We used an **Activity-on-Node (AON)** network diagram to map all the WBS activities and there precedence relationships. We computed the Early Start (ES), Early Finish (EF), Late Start (LS), and Late Finish (LF) for each node, using forward and backward passes. Activities with zero float are classified as critical. The network reveals significant parallel branches that feed into Activity J (System Integration)

Act.	T_E	ES	EF	LS	LF	Slack	Critical?
A	2.000	0	2.000	0	2.000	0	Yes
B	4.000	2.000	6.000	2.000	6.000	0	Yes
C	5.000	6.000	11.000	6.000	11.000	0	Yes
D	4.000	11.000	15.000	11.000	15.000	0	Yes
E	3.167	2.000	5.167	14.000	17.167	12.000	-
F	5.000	2.000	7.000	13.167	18.167	11.167	-
G	4.000	15.000	19.000	-	-	-	milestone
H	5.167	7.000	12.167	17.167	22.334	10.167	-
I	5.167	15.000	20.167	15.000	20.167	0	Yes
J	4.000	20.167	24.167	20.167	24.167	0	Yes
K	6.167	24.167	30.334	24.167	30.334	0	Yes
L	4.000	30.334	34.334	30.334	34.334	0	Yes

Table 2: Detailed Project Schedule with Early/Late Start and Finish Times

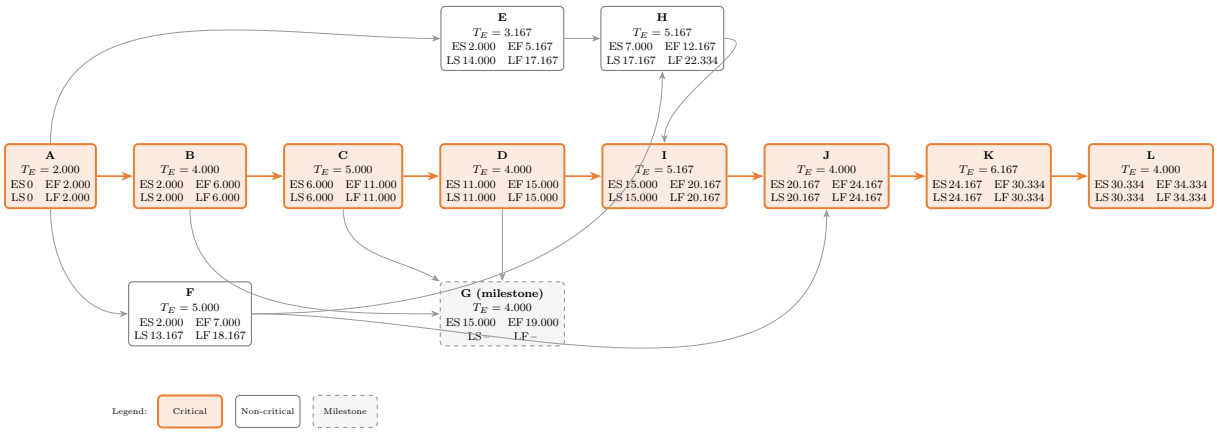


Figure 3: Activity-on-Node (AON) network diagram. Orange nodes and edges form the critical path $\mathrm{A} \to \mathrm{B} \to \mathrm{C} \to \mathrm{D} \to \mathrm{I} \to \mathrm{J} \to \mathrm{K} \to \mathrm{L}$ ($\mathrm{CT} = 34.334$ days). Each node shows activity ID, expected duration T_E, and the ES/EF/LS/LF values from the PERT analysis. The dashed node G is a milestone with no successors on the path to completion.

2.7 Slack

The slack (or float) for each activity in the WBS is computed as

$$S = LS - ES = LF - EF$$

Activities on the critical path have zero slack, meaning any delay in said activities would directly delay the project completion time.

Activity	T_E	ES	EF	LS	LF	Slack
A	2.000	0.000	2.000	0.000	2.000	**0.000**
B	4.000	2.000	6.000	2.000	6.000	**0.000**
C	5.000	6.000	11.000	6.000	11.000	**0.000**
D	4.000	11.000	15.000	11.000	15.000	**0.000**
E	3.167	2.000	5.167	14.000	17.167	12.000
F	5.000	2.000	7.000	13.167	18.167	11.167
G	4.000	15.000	19.000	-	-	milestone
H	5.167	7.000	12.167	17.167	22.334	10.167
I	5.167	15.000	20.167	15.000	20.167	**0.000**
J	4.000	20.167	24.167	20.167	24.167	**0.000**
K	6.167	24.167	30.334	24.167	30.334	**0.000**
L	4.000	30.334	34.334	30.334	34.334	**0.000**

Table 3: Activity Slack and Criticality Analysis

The non-critical activities each have a substantial slack, which means they have some considerable flexibility in scheduling. This was exploited in implementation. Activity E (CAD) and Activity F (electronics) were progressed in parallel during the early weeks without risk to the project timeline. Fabrication of our prototypes (Activity H) similarly has over 10 days of slack, but it is contingent on F not being delayed beyond its LS time.

2.8 CP, CT

The critical path, as shown in the AON diagram is determined by identifying the longest path that would result in the completion of the project. From the forward and backward pass results, the activities that form the critical path are:

$$\text{CP} : A \to B \to C \to D \to I \to J \to K \to L$$

The **Critical Time** is the sum of expected durations along this path:

$$CT = T_E(A) + T_E(B) + T_E(C) + T_E(D) + T_E(I) + T_E(J) + T_E(K) + T_E(L)$$

$$CT = 2.000 + 4.000 + 5.000 + 4.000 + 5.167 + 4.000 + 6.167 + 4.000 = \mathbf{34.334} \text{ days}$$

2.9 Gantt Chart

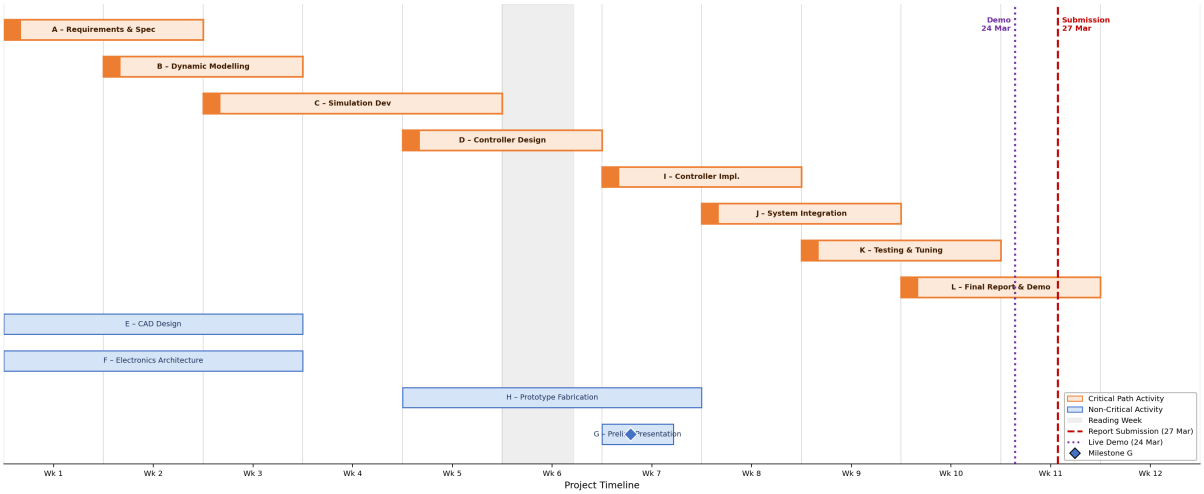


Figure 4: Gantt Chart

2.10 Probability of Completion

Using the PERT methodology, the probability of completing the project by a target data T_D is estimated via a normal approximation over the critical path. Only the variance of critical activities contribute to the schedule uncertainty:

$$\sigma_{CP}^2 = \sigma_A^2 + \sigma_B^2 + \sigma_C^2 + \sigma_D^2 + \sigma_I^2 + \sigma_J^2 + \sigma_K^2 + \sigma_L^2$$

$$= 0.111 + 0.444 + 0.444 + 0.444 + 0.694 + 0.444 + 1.361 + 0.444 = \mathbf{4.386} \text{ days}^{\mathbf{2}}$$

$$\sigma_{CP} = \sqrt{4.386} \approx 2.094 \text{ days}$$

The probability of completing by a target deadline T_D is then:

$$P(T \leq T_D) = \Phi\left(\frac{T_D - CT}{\sigma_{CP}}\right) = \Phi\left(\frac{T_D - 34.334}{2.094}\right)$$

Target T_D (days)	Z-score	$P(T \leq T_D)$
34.334 (= CT, no buffer)	0.00	50.0%
36.0 (1.7 day buffer)	0.80	78.8%
37.0 (2.7 day buffer)	1.27	89.8%
39.0 (4.7 day buffer)	2.23	98.7%
40.0 (5.7 day buffer)	2.71	99.7%

Table 4: Probability of project completion for selected target deadlines

2.11 RACI matrix

Stage	Task	Helitha	Xavier	Aayan	Peter	Sajad	Lab
	Requirements & System Spec.	A	R	R	R	C	I
	General mind map	A/R	R	R	R	I	I
	WBS/RBS	A	I	R	I	I	I
	Gantt chart design	A	C	I	R	I	I
	RACI design	A/R	I	I	R	I	I
	Morphological chart	A/R	R/C	R/C	R/C	I	I
	Derive system equations	A/R	R	C	I	I	I
	Simulation Env. Development	A/R	R	C	I	I	I
	Controller Design (PID/LQR)	A/R	R	C	I	I	I
	Concept & mechanical design	A	I	R	R	I	I
	CAD Design	C	C	R	R	I	I
	Electronics Architecture	C	R	R	C	I	I
	Preliminary Presentation	A	R	R	R	C	I
	Prototype Fabrication	A	C	R	R	I	C
	Controller Impl. (Embedded)	A/R	R	C	C	I	I
	Full Bill of Materials	A	C	C	C	I	I
	Integrated Schematics	C	R	R	C	I	I
	Control flow design	A/R	R	C	C	I	I
	Control sequence diagram	A	R	I	I	I	I
	System Integration	A	R	R	R	I	C
	Testing and Tuning	A	R	R	R	C	C
	Final Report and Demo	A	R	R	R	C	I

Table 5: Project RACI Responsibility Matrix (R = Responsible, A = Accountable, C = Consulted, I = Informed)

2.12 PLC

Week	Phase	Review Gate	Primary Deliverable
1	Pre-Phase A: Concept Studies	MCR	Morphological chart, mind map, mission strategy
2	Phase A: Concept & Tech Dev	SRR/MDR	Requirements, WBS/RBS, PERT network, Python simulation
3	Phase B: Preliminary Design	PDR	CAD models, V-block design, preliminary schematics
4	Phase C: Final Design & Fab	CDR	SysML diagrams, final BoM, complete CAD, CDR presentation
5	Phase C: System Integration	SIR	Assembled chassis, swing test (damping ratio $\zeta < 0.01$)
6	Reading Week	–	Report drafting, simulation refinement
7	Phase D: System Assembly & Test	ORR	Integrated system ready for evaluation
8–9	Phase E: Ops & Sustainment	–	Disturbance rejection demo, controller tuning
10	Phase E: Final Evaluation	–	Live demo: Evaluations A, B, C
11	Phase F: Closeout	DR/DRR	Final report, decommissioning

Table 6: Project Life Cycle Phase Mapping and Milestones

The V-Model structure gives us bidirectional traceability. Every verification activity in phases D–F traces back to a specific requirment defined in phase A. For instance, the damping ratio pass/fail gate at SIR (week 5) verifies the damping constraint on the pivot mechanism. Similarly, the three final evaluations can each be traced to the functional requirements of the system. This prevents scope creep and made our build effort remain aligned with the relevant deliverables throughout the project

3 Risk Management

3.1 Ishikawa Diagram

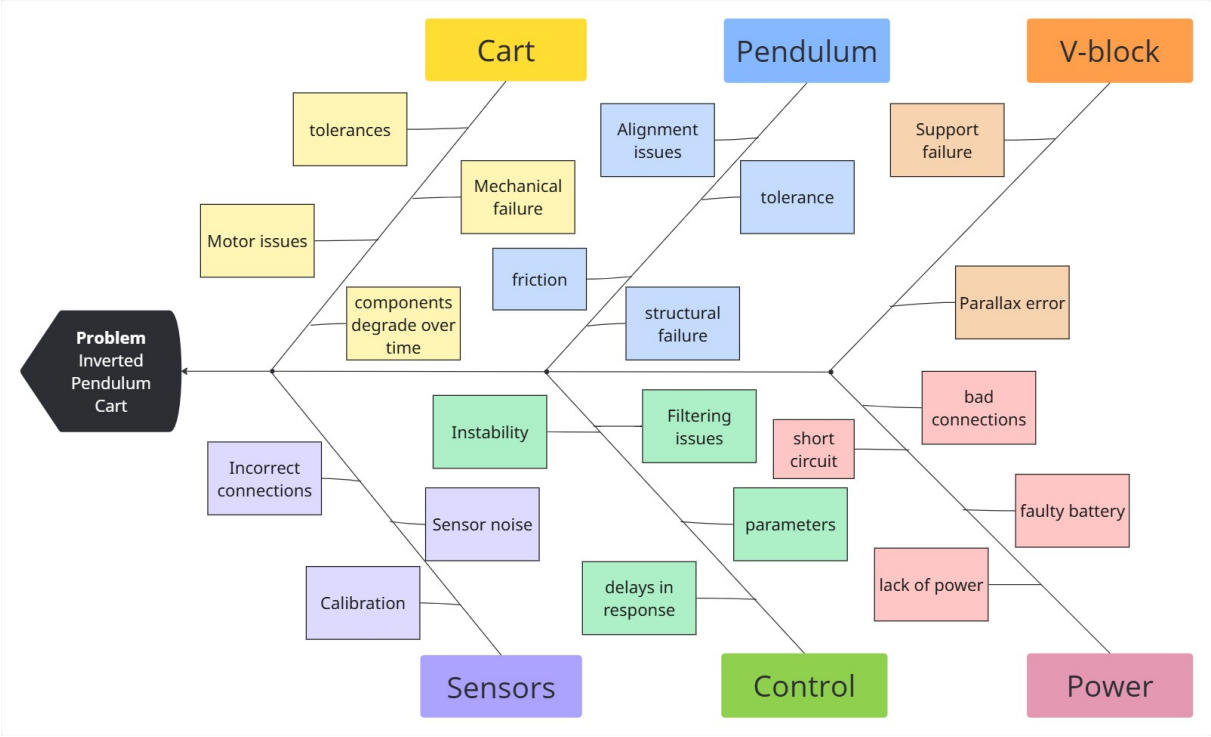


Figure 5: Fishbone diagram

3.2 FMEA

Risk (Failure Mode)	S	L	D	RPN	Causes
Incorrect sensor readings	8	6	4	192	Sensor faults, poor processing
Faulty motors/motor drivers	9	3	3	81	Burning components, manufacturing faults
Slow microcontroller	7	3	6	126	Internal clock speed
Overheating	8	4	2	64	Overusing components, poor connections
Loose wires	6	9	2	108	Mismanagement
Poor soldering	8	5	4	160	Loose attachments, board organisation
Incorrect parameter tuning	6	7	7	294	Simulation inconsistency, design errors
Tight schedule	6	9	7	378	Deadline mismanagement
Part breakage	5	7	1	35	Structural faults, part overuse
Battery charge empty	9	3	1	27	Power overuse, poor conservation

Note: S = Severity, L = Likelihood, D = Detectability (1–10 scale). $RPN = S \times L \times D$.

Table 7: Failure Mode and Effects Analysis (FMEA)

3.3 Risk Matrix

Table 8: Project Risk Matrix

Prob. \ Impact	Low	Medium	High
High	• Underestimate task complexity	• Drop-in sessions are full	• Sensor noise and drifting
Medium	• Disagreement on design choices • Component tolerance incorrect	• Components fail during testing • Incorrect tuning parameters • Weak solders on boards	• Battery connected incorrectly
Low	• Microcontroller fails • Wires disconnect during test • Components overheat	• Components short circuit • Incorrect connections	• Lack of sufficient testing time

3.4 Redundancy and Compensatory Measures

Table 9: Redundancy and Compensatory Measures

Redundancy Type	Compensatory Measures
Sensor Redundancy	Optical encoder tested multiple times to ensure accurate readings
Motor Redundancy	4 motors to ensure that sufficient torque can be provided instantly based off the change in the angle/ change in encoder value
Thermal Redundancy	Parts spaced to prevent overheating; no electrical parts encased to allow sufficient airflow and passive cooling

3.5 Power Interest Grid

Table 10: Stakeholder Power-Interest Grid

Power \ Interest	Low Interest	High Interest
High Power	**Keep Satisfied** • Innovation lab staff	**Closely Manage** • Sajad (Module lead) • Teammates
Low Power	**Monitor** • Other groups	**Keep Informed** • Teaching assistants

3.6 Security

Table 11: Security Analysis: Threats, Vulnerabilities, and Mitigations

Component	Threats & Impact	Vulnerabilities	Mitigations
I^2C Communications (SDA & SCL)	**Data Sniffing & Injection:** Unauthorized access to read system communications or write malicious data. This could lead to a loss of control over the cart.	The I^2C protocol is unencrypted by default. The SDA and SCL connections are physically exposed on the board, making them easily accessible.	**Physical Security:** Restrict physical access to the connections using an enclosure. **Encryption:** Ensure data packets are encrypted or validated before execution by the motor drivers.
Power System (Battery)	**Short Circuits & Thermal Runaway:** Electrical shorts causing critical damage to the battery. This can lead to severe safety hazards, including explosions or fire.	Battery terminals and power distribution connections are exposed, increasing the likelihood of accidental shorts from dropped conductive tools or loose wires.	**Physical Protection:** Ensure all power connections and the battery itself are securely protected within an insulated enclosure. Use heat shrink on all exposed terminals.
Wi-Fi Communications (Arduino Giga)	**Interception & Hijacking:** Attackers could intercept remote tuning data or spoof commands to maliciously enable/disable the cart. This could result in unpredictable behavior or physical hazards.	Wi-Fi signals are broadcasted over the air. Using an open network, weak encryption, or lacking authentication on the web interface leaves the system highly vulnerable to remote access.	**Network Security:** Enforce the use of strong encryption protocols (WPA2/WPA3) for the local network connection. **Authentication:** Implement a secure password or token requirement before the system accepts any remote tuning or state-change commands.

3.7 Applied Lean Method

To optimize our workflow and maximize value, the team implemented Lean methodologies (Kaizen), specifically focusing on visualizing work and minimizing waste through the TIMWOODS framework.

Visualizing Work and Communication
While traditional software teams often use formal Kanban boards, the team adopted Lean visualization principles using a combination of Miro and WhatsApp. Miro served as the macro-level visualization tool, housing a comprehensive project structure that clearly mapped out every team member's assigned tasks. This ensured total transparency and prevented duplicated effort. WhatsApp was utilized for agile, daily coordination, allowing the team to outline immediate objectives and manage remote work such as CAD modelling and control algorithm programming effectively.

Waste Reduction (TIMWOODS)
The TIMWOODS framework was actively applied to the fabrication and assembly processes to eliminate inefficiencies:

- **Defects:** To prevent manufacturing faulty products, each part was meticulously checked and incorporated appropriate tolerances prior to 3D printing or laser cutting.
- **Waiting:** To eliminate team bottlenecks and prevent machines from sitting idle, 3D prints were strategically scheduled either overnight or concurrently with other active tasks.
- **Motion:** Minimized unnecessary motion such as team members searching for tools by fully utilizing the Innovation Lab's highly organised, labelled, and sorted component storage system, allowing for rapid and efficient part retrieval.
- **Overproduction:** To avoid manufacturing items before they were truly needed, only the components required for the current prototyping phase were fabricated. Reprints or re-cuts were only used for major structural changes; minor adjustments were performed manually by hand to conserve materials and time.

3.8 Commitment Assessment

Table 12: Commitment Assessment Matrix (X = Current, O = Desired)

Level of Commitment	Teammates	Sajad (Module Lead)	Innovation Lab Staff	Teaching Assistants
Strongly Support	XO	O		
Support			O	
Neutral		X	X	XO
Opposed				
Strongly Opposed				

4 System Requirement Specification

4.1 Dynamics and Kinematics Requirements Consistency

The dynamics of the system are governed by two linearised equations of motion that we will be deriving in the following section. To verify consistency, we substituted the physical parameters of the built system into these equations and checked against the functional requirements and constraints that we identified in the beginning.

Parameter	Symbol	Value
Cart mass (total)	M	1.381 kg
Pendulum mass (rod + disk)	m	0.168 kg
Pendulum length to CoM	l	0.60 m
Moment of inertia (point mass)	I	ml^2
Gravitational acceleration	g	9.81 m/s^2

Table 13: System Dynamic Parameters and Measured Values

Since the payload is modelled as a point mass at the top of the pendulum, the moment of inertia will be $I = ml^2$, which means the equation for rotation becomes:

$$(ml^2 + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x}$$

$$2ml^2\ddot{\phi} - mgl\phi = ml\ddot{x}$$

Dividing through by ml:

$$2l\ddot{\phi} - g\phi = \ddot{x}$$

We identified this to be significant as the pendulum mass m cancels entirely from the rotational dynamics under the point-mass assumption. The angular behaviour of the system is therefore governed by the pendulum length l and gravitational acceleration g, independently of how heavy the rod and payload are. This simplified out controller design considerably, while not accounting for reduced performance, as gain tuning was done accordingly, and will not have to be drastically changed or redone at all if the payload mass changes.

The open-loop natural frequency of the linearised pendulum about the upright equilibrium is:

$$\omega_n = \sqrt{\frac{g}{2l}} = \sqrt{\frac{9.81}{2 \times 0.60}} \simeq 2,86 rad/s$$

This corresponds to an open loop time constant of $\tau = 1/\omega_n \simeq 0.35\,s$. Our LQR control loop for instance, runs at 100 Hz, which is a loop period of 10 ms so is approximately 35 times faster than the open loop time constant. This satisfies the Nyquist sampling criterion with significant margin, ensuring the controller can observe and correct perturbations well before they grow.

Constraint consistency check:

- **Pendulum Length:** The choice of $l = 0.60\,\text{m}$ satisfies the design constraint $0.6\,\text{m} \leq l \leq 1.0\,\text{m}$. By placing the system at the lower bound, we maximise ω_n, making the open-loop instability faster. This was an intentional design choice to stress-test the LQR controller's responsiveness.
- **Cart Geometry:** Cart motion is constrained to a single horizontal axis by the four-wheel chassis geometry. The linearised cart equation assumes purely linear motion; the rigid acrylic chassis and parallel wheel alignment are used to mitigate yaw or lateral drift that would otherwise introduce unmodelled coupling terms.
- **Payload Dynamics:** The 50 g payload disk is attached at the tip of the rod, which is consistent with the point-mass assumption used to simplify the moment of inertia $I = ml^2$. With a disk diameter of 31.58 mm relative to the 600 mm rod, the distributed mass approximation remains valid to within a few percent.
- **Sprint Constraints:** For Evaluation C, the sprint target is set to 2.0 m in the LQR position reference. A tolerance of $\pm 0.1\,\text{m}$ is enforced in the benchmarking code, which is directly traceable to the $2 \pm 0.1\,\text{m}$ constraint specified in the project requirements.

4.2 Logical Requirements Consistency

The logical requirements we have set ensures that the behavioural specification of the system does not contradict, overlap or leave gaps in the functional requirements we have defined earlier. This is verified through two lenses: a formal analysis of the logical architecture, and a direct traceability mapping from each requirement to its implementation.

Logical Architecture The cart operates as a discrete-event reactive system, responding to sensor measurements and serial operator commands to transition between defined operating modes. The logical architecture decomposes into three functional layers:

- **Sensing Layer:** The `SensorSubsystem` continuously samples the pendulum angle θ via the AS22 optical encoder and the cart position x via the Pololu 25D motor encoder. Both are sampled at 10 kHz through the `mbed::Ticker` interrupt service routine. A 5-sample moving average filter is applied to both signals before they are consumed by the control layer, effectively attenuating high-frequency encoder noise without introducing significant phase lag at the 100 Hz control frequency.
- **Decision Layer:** The `ControllerSubsystem` evaluates the current system state and filtered sensor inputs at 100 Hz to determine the appropriate control action. This layer is solely responsible for state transitions, guard condition evaluation, and execution of the LQR control law. By centralising the logic, the design ensures that no control logic exists outside this subsystem, maintaining strict architectural modularity.
- **Actuation Layer:** The `ActuatorSubsystem` translates the scalar control signal u into motor speed commands. These commands are distributed identically to all four drive motors via I^2C through dual Motoron drivers. This configuration ensures the signal path from controller output to physical motor torque is entirely deterministic and remains synchronous with the 100 Hz control loop.

4.3 Mealy Machine

Table 14: Mealy machine state transition table

Current State	Input / Guard Condition	Next State	Output u
IDLE	`GO` command received	JERK	u_{kick} (750 units, 200 ms)
IDLE	No `GO` command	IDLE	$u_{off} = 0$
JERK	$t_{jerk} < t_{kick}$	JERK	u_{kick}
JERK	$t_{jerk} \geq t_{kick}$	COAST	$u_{off} = 0$
COAST	$t_{coast} < 40\,\text{ms}$	COAST	$u_{off} = 0$
COAST	$t_{coast} \geq 40\,\text{ms}$ and $\lvert\theta\rvert < 0.35\,\text{rad}$	BALANCING	u_{LQR}
COAST	$t_{coast} \geq 40\,\text{ms}$ and $\lvert\theta\rvert \geq 0.35\,\text{rad}$	JERK	u_{kick} (retry)
BALANCING	$\lvert\theta\rvert \leq 0.60\,\text{rad}$	BALANCING	$u_{LQR} = -\mathbf{Ke}$
BALANCING	$\lvert\theta\rvert > 0.60\,\text{rad}$ **or** `STOP`	FALLEN	$u_{off} = 0$
JERK / COAST	$t_{out} > 2\,\text{s}$ **or** `STOP`	IDLE	$u_{off} = 0$
FALLEN	any	IDLE	$u_{off} = 0$

COAST is the 40 ms open-loop phase between jerk and LQR capture. $\mathbf{e}$ *is the state error vector* $[\theta - \theta_{des},\ \dot{\theta},\ x - x_{des},\ \dot{x}]^\top$.

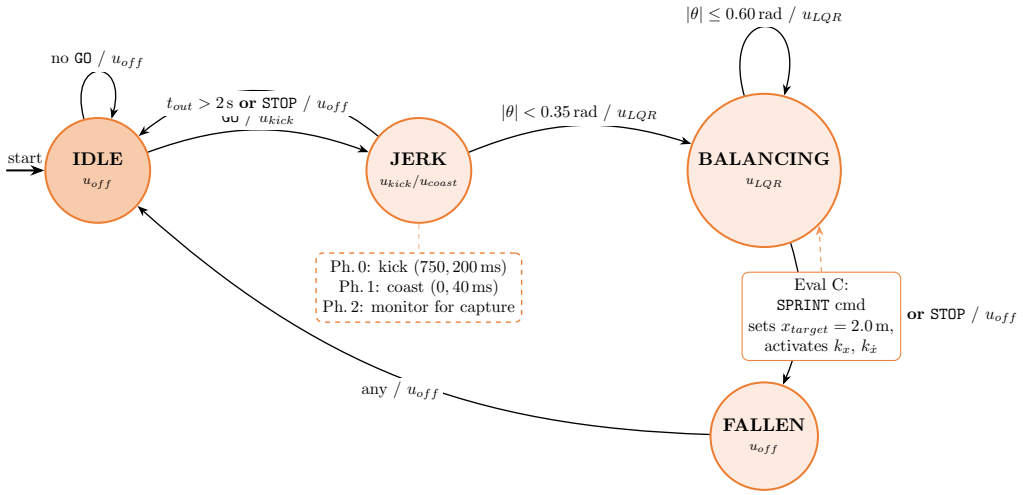


Figure 6: Mealy machine state diagram for the inverted pendulum controller. States are IDLE, JERK, BALANCING, and FALLEN. Transitions show guard conditions and output actions. The JERK sub-phase box details the open-loop swing-up sequence preceding LQR capture.

4.4 Model-Based Systems Engineering Diagrams

4.5 Reliability (Failure Time Metrics)

Reliability was estimated theoretically from the FMEA, using RPN values as proxies for relative failure rate. Each failure mode is modelled as an independent exponential process; only modes with RPN ≥ 100 are included. Mission time is one evaluation run ($T_m = 5\,\text{min} = \frac{1}{12}\,\text{hr}$).

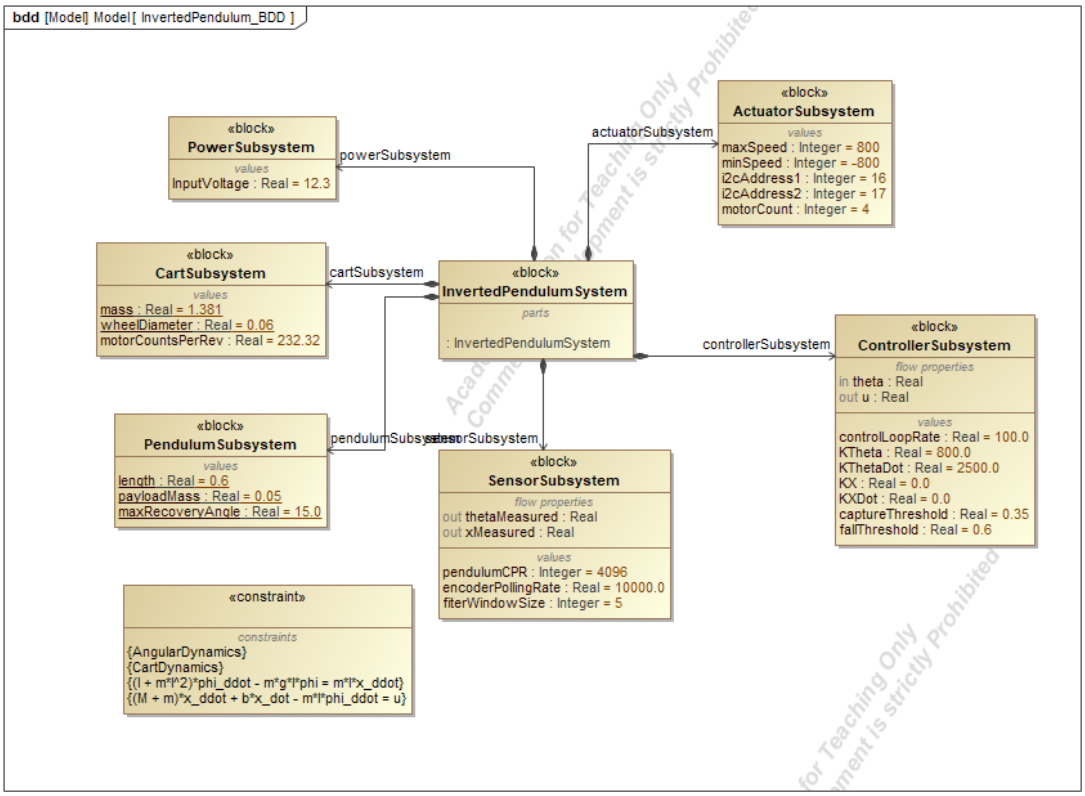


Figure 7: Block Definition Diagram showing system decomposition

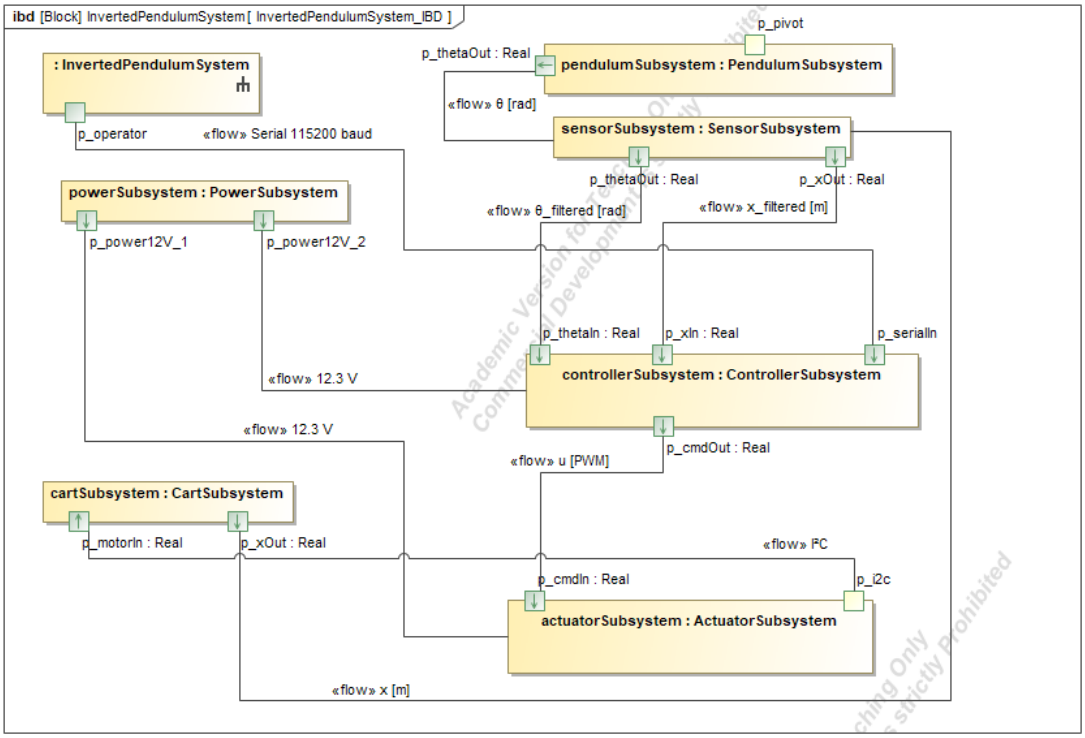


Figure 8: Internal Block Diagram showing signal flow

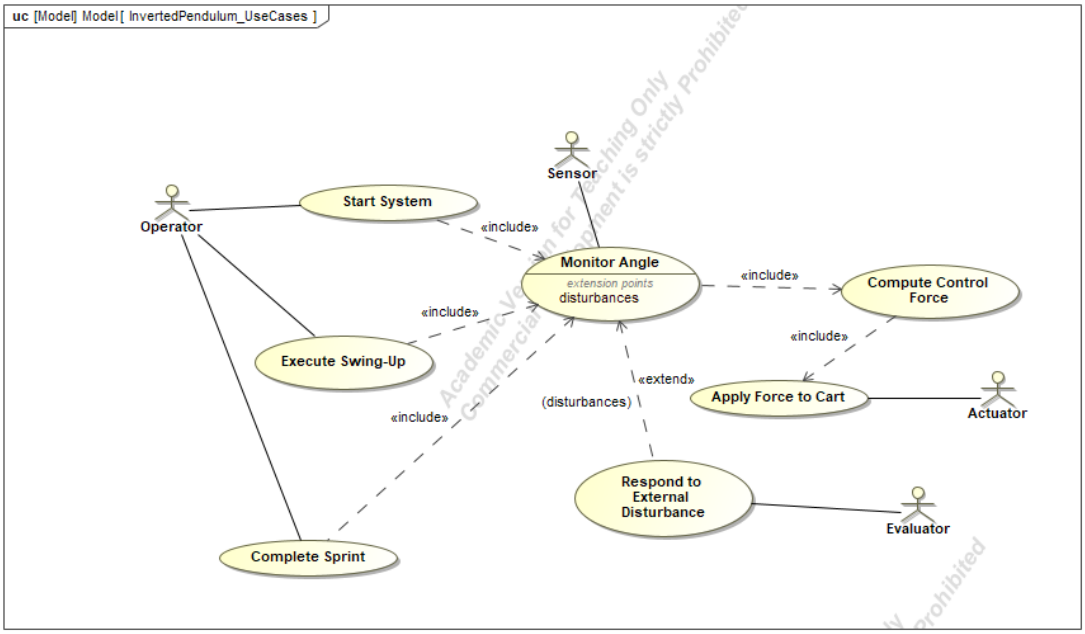


Figure 9: Use Case Diagram showing actor interactions

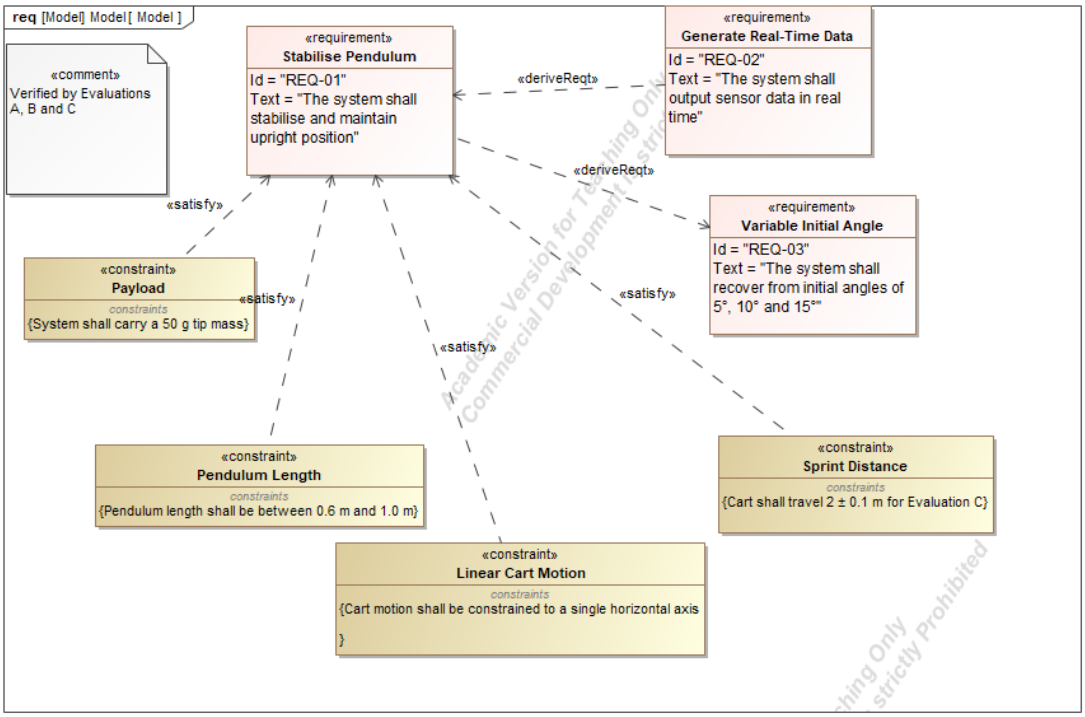


Figure 10: Requirements Diagram showing constraints and traceability

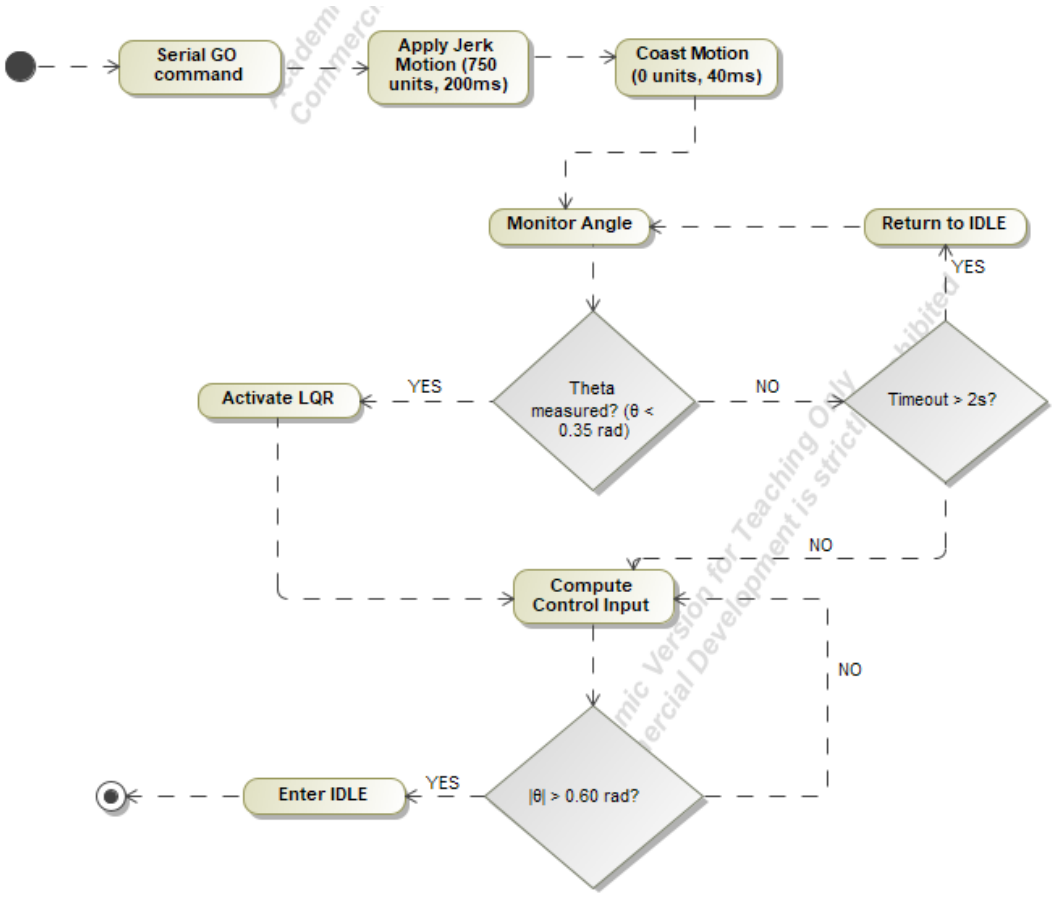


Figure 11: Activity Diagram showing swing-up control flow

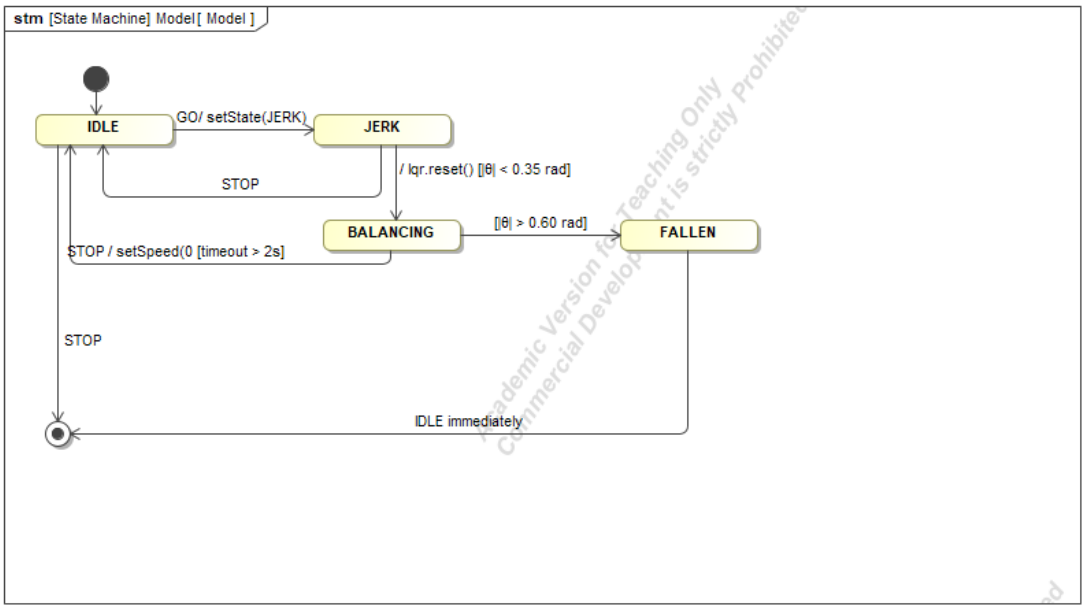


Figure 12: State Machine Diagram showing IDLE, JERK, BALANCING, and FALLEN states

Individual failure rates are proportioned from a baseline $\lambda_{sys,base} = 0.1\,\text{hr}^{-1}$ (MTTF $= 10\,\text{hr}$, conservative for a built embedded system):

Table 15: Per-failure-mode reliability parameters (RPN ≥ 100)

Failure Mode	RPN	w_i	$\lambda_i\,(\text{hr}^{-1})$
Tight schedule	378	0.272	0.0272
Incorrect parameter tuning	294	0.212	0.0212
Incorrect sensor readings	192	0.138	0.0138
Poor soldering	160	0.115	0.0115
Slow microcontroller	126	0.091	0.0091
Loose wires	108	0.078	0.0078

The system failure rate, MTTF, reliability, and availability are:

$$\lambda_{sys} = \sum_i \lambda_i = 0.0906\,\text{hr}^{-1} \qquad \text{MTTF} = \frac{1}{\lambda_{sys}} \approx 11.04\,\text{hr}$$

$$R(T_m) = e^{-\lambda_{sys} \cdot T_m} = e^{-0.0906/12} \approx \mathbf{99.25}\%$$

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} = \frac{11.04}{11.04 + \frac{1}{6}} \approx \mathbf{98.51}\%$$

where MTTR $= 10\,\text{min}$ is a conservative estimate based on observed fault resolution during testing (loose wires and parameter retuning were consistently resolved within 5 minutes).

The dominant risk is tight schedule management (RPN $= 378$), a project-level rather than in-run hardware failure. Excluding it gives $\lambda_{hw} = 0.0634\,\text{hr}^{-1}$, MTTF $\approx 15.77\,\text{hr}$, and $R(T_m) \approx 99.47\%$. Both estimates confirm the system is sufficiently reliable for short evaluation runs, with the primary risk lying in pre-run preparation, mainly soldering quality and parameter tuning, rather than in-run hardware failure.

5 Modelling and Simulation

5.1 Equations of Motion

5.1.1 Setup

We assume our cart has mass M with linear motion, with a pendulum with total mass (rod + payload) m attached to it. The pendulum length l to the centre of mass and let's consider the moment of inertia I.

- $x(t)$: Horizontal position of the cart.
- $\theta(t)$: Angle of the pendulum measured from the vertical **down** position.

- F (or u): Force applied to the cart.
- b: Coefficient of friction on the cart.
- N: Horizontal reaction force at the pivot.
- P: Vertical reaction force at the pivot.

5.1.2 Cart Equation of Motion

Summing the horizontal forces on the cart, we get:

$$\sum F_x = M\ddot{x} \tag{1}$$

The forces on the cart are the input force u, the opposing friction $b\dot{x}$, and the horizontal reaction force N exerted by the pendulum rod on the cart (opposing motion).

$$u - b\dot{x} - N = M\ddot{x} \tag{2}$$

This is our first key relationship. However, we do not know N yet. We analyse the motion of the pendulum's centre of mass:

$$x_p = x + l\sin\theta \tag{3}$$
$$y_p = -l\cos\theta \tag{4}$$

To find the forces, we need the acceleration of this centre of mass. We differentiate with respect to x:

first derivative (velocity):

$$\dot{x}_p = \dot{x} + l\dot{\theta}\cos\theta \tag{5}$$
$$\dot{y}_p = l\dot{\theta}\sin\theta \tag{6}$$

second derivative (acceleration):

$$\ddot{x}_p = \ddot{x} + l\ddot{\theta}\cos\theta - l\dot{\theta}^2\sin\theta \tag{7}$$
$$\ddot{y}_p = l\ddot{\theta}\sin\theta + l\dot{\theta}^2\cos\theta \tag{8}$$

We then apply Newton's second law of motion to the pendulum in the horizontal direction.

$$N = m\ddot{x}_p \tag{9}$$

Substitute the equation $\ddot{x}_p$ for into this:

$$N = m(\ddot{x} + l\ddot{\theta}\cos\theta - l\dot{\theta}^2\sin\theta) \tag{10}$$

5.1.3 Combining the cart and pendulum equations

We now eliminate the unknown reaction force N by substituting that equation back into the cart equation.

$$M\ddot{x} + b\dot{x} + N = u$$
$$M\ddot{x} + b\dot{x} + [m(\ddot{x} + l\ddot{\theta}\cos\theta - l\dot{\theta}^2\sin\theta)] = u$$

Grouping the $\ddot{x}$ terms:

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = u \tag{11}$$

the first non linear equation

5.1.4 The rotation equation

We sum the forces perpendicular to the rod/ moments about the center of mass. Let's label that as τ

$$\sum \tau = I\ddot{\theta} \tag{12}$$

From geometry, the lever arm components result in:

$$\tau = -N(l\cos\theta) - P(l\sin\theta) \tag{13}$$

Sum vertical forces on the pendulum:

$$P - mg = m\ddot{y}_p \implies P = m(l\ddot{\theta}\sin\theta + l\dot{\theta}^2\cos\theta) + mg$$

Summing Moments about the Centre.

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta}$$

We substitute N and P (vertical force balance: $P - mg = m\ddot{y}_p$).

$$P\sin\theta + N\cos\theta - mg\sin\theta = ma_\perp$$

then we derive the final rotational equation directly from the standard moment balance:

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

Moving everything to the left side in this equation:

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \tag{14}$$

$$(I + ml^2)\ddot{\theta} - mgl\sin\theta + ml\ddot{x}\cos\theta = 0 \tag{15}$$

5.1.5 Linearisation

The equations we have are nonlinear due to the trig terms. To design a linear controller, we must linearise about the unstable equilibrium point.

Variable Transformation: Let $\theta = \pi + \phi$, where ϕ is a small deviation from the vertical up position.

$$\dot{\theta} = \dot{\phi}, \quad \ddot{\theta} = \ddot{\phi}$$

Small Angle Approximations: For small ϕ:

1. $\cos\theta = \cos(\pi + \phi) = -\cos\phi \approx -1$
2. $\sin\theta = \sin(\pi + \phi) = -\sin\phi \approx -\phi$
3. $\dot{\theta}^2 = \dot{\phi}^2 \approx 0$ (Second order small term is negligible)

Linearising the cart eqn Start with:

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

Substitute the approximations:

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\phi}(-1) - ml(0)(-\phi) = u$$

Simplify:

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \tag{16}$$

Linearising the pendulum eqn start with:

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

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

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

Rearrange:

$$(I + ml^2)\ddot{\phi} - mgl\phi - ml\ddot{x} = 0 \tag{17}$$

Final Linearised System We now have a system of two linear ordinary differential equations:

1. **Cart Dynamics:**
$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u$$
2. **Pendulum Dynamics:**
$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x}$$

5.1.6 State Space Model

Once we had the linearised equations, we then turned them into state space matrices, following the principle $\dot{\vec{x}} = \mathbf{A}\vec{x} + \mathbf{B}u$

State Vector: $\begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$

Matrix A:

$$\frac{1}{\alpha}\begin{bmatrix} 0 & \alpha & 0 & 0 \\ 0 & -b(I+ml^2) & m^2l^2g & 0 \\ 0 & 0 & 0 & \alpha \\ 0 & -mlb & mlg(M+m) & 0 \end{bmatrix} \tag{18}$$

where $\alpha = \frac{(I(M+m)+Mml^2}{ml}$

Matrix B:

$$\frac{1}{\alpha}\begin{bmatrix} 0 \\ I+ml^2 \\ 0 \\ ml \end{bmatrix} \tag{19}$$

5.2 Controller Design and Implementation

We went with two different controllers, PID and LQR, we chose these two controllers as they would be sufficient and would be easier to implement than other controllers such as MPC, providing us more time to tune the controller and quicker to debug when the controller breaks. In addition, objective C had a slightly different PID algorithm because we found PID. At each time step the time between control inputs is recorded and then the derivative from the previous term is used and the integral is updated using the rectangle rule.

5.2.1 PID

The first controller, PID, differed slightly from a standard PID response. Our one had the usual three terms (proportional, integral, derivative) for both position (x) and the angle (θ), however they were arranged in a cascading manner.

The first step was to calculate the errors, the readings are passed through the moving window average filter for the current position and angle. Calculate the derivatives of x & θ using the change in value over the time step. We calculate the integral term using the rectangle rule. The integral errors are constrained to not go over 10 to avoid integral drift and taking over the control input.

$$\theta_{des} = kp_x * error_x + ki_x * integral_x + kd_x * \dot{x} \tag{20}$$

$$error_\theta = \theta - \theta_{des} \tag{21}$$

$$input = kp_\theta * error_\theta + ki_\theta * integral_\theta + kd_\theta * \dot{\theta} \tag{22}$$

The cascaded approach allowed the PID to have an understanding of where the robot needs to be to balance the pendulum. These equations run every loop. We used the Ziegler-Nichols method [5] to tune the PID.

5.2.2 LQR

The second controller we decided to use was an LQR controller. Using the system model from earlier and by choosing a Q matrix and R constant, we could feed those into the Riccati equations [3] to obtain an optimum K for each state variable, stored in the vector $\vec{k}$ to produce the control input, following the equation below:

$$u = \begin{bmatrix} \theta - \theta_{des} \\ \dot{\theta} - \dot{\theta_{des}} \\ x - x_{des} \\ \dot{x} - \dot{x}_{des} \end{bmatrix} \cdot \vec{k}$$

(23)

5.3 PID Trajectory

When trying to tune the PID controller in the simulation, we were finding it very difficult to get the robot to stop on 2 metres, we could get it to slide infinitely, but not stop and balance at 2 metres. So we used a slight variation of the PID controller, by creating a trajectory for the pendulum to follow. We decided on a smooth S-curve path following the cubic polynomial $3s^2 - 2s^3$ (where s is a normalised unit of time), this polynomial was chosen because it ensures a velocity of 0 at both the start and end. Furthermore, when $s = 0.5$ (the halfway point) the velocity will be a maximum. A normal PID can be quite jerky, this trajectory ensures a smoother response.

We select the duration we would like the trajectory to target over, t_{des} and all the same gains from the PID controller. Below are how the target x, x_{ref} and target $\dot{x}$, $\dot{x}_{ref}$ are calculated which are then fed into the same cascaded PID equations from above. x_{start} will always be 0 as it has a start line and finish line 2 metres apart.

$$s = \frac{t}{t_{des}} \tag{24}$$

$$x_{ref} = x_{start} + (x_{target} - x_{start}) * (3s^2 - 2s^3) \tag{25}$$

$$\dot{x}_{ref} = \frac{(x_{target} - x_{start}) * (6s - 6s^2)}{t_{des}} \tag{26}$$

5.4 Python Implementation Details

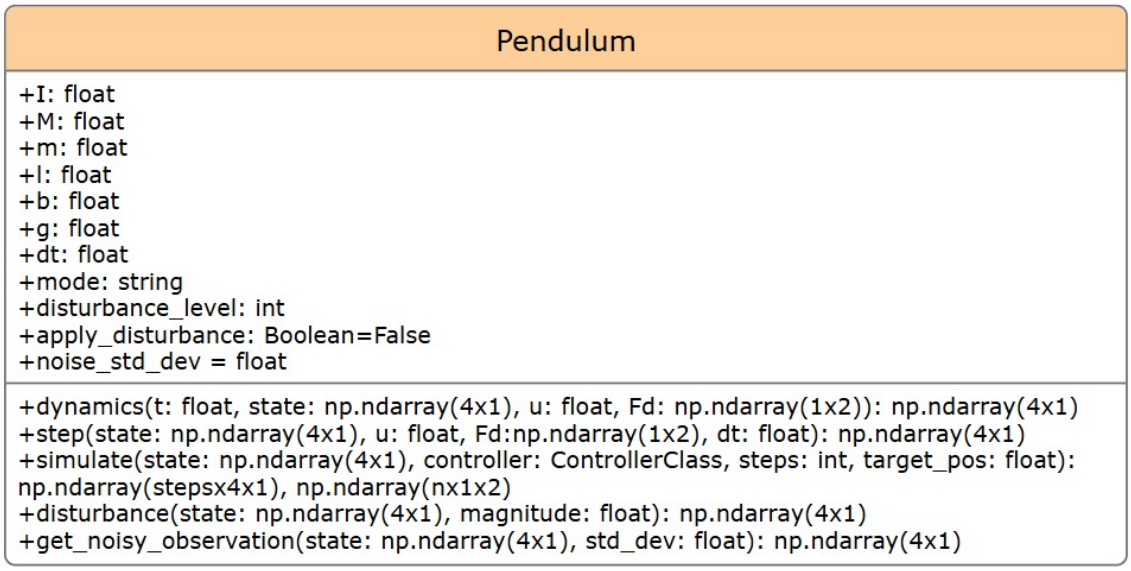


Figure 13: UML diagram for the pendulum

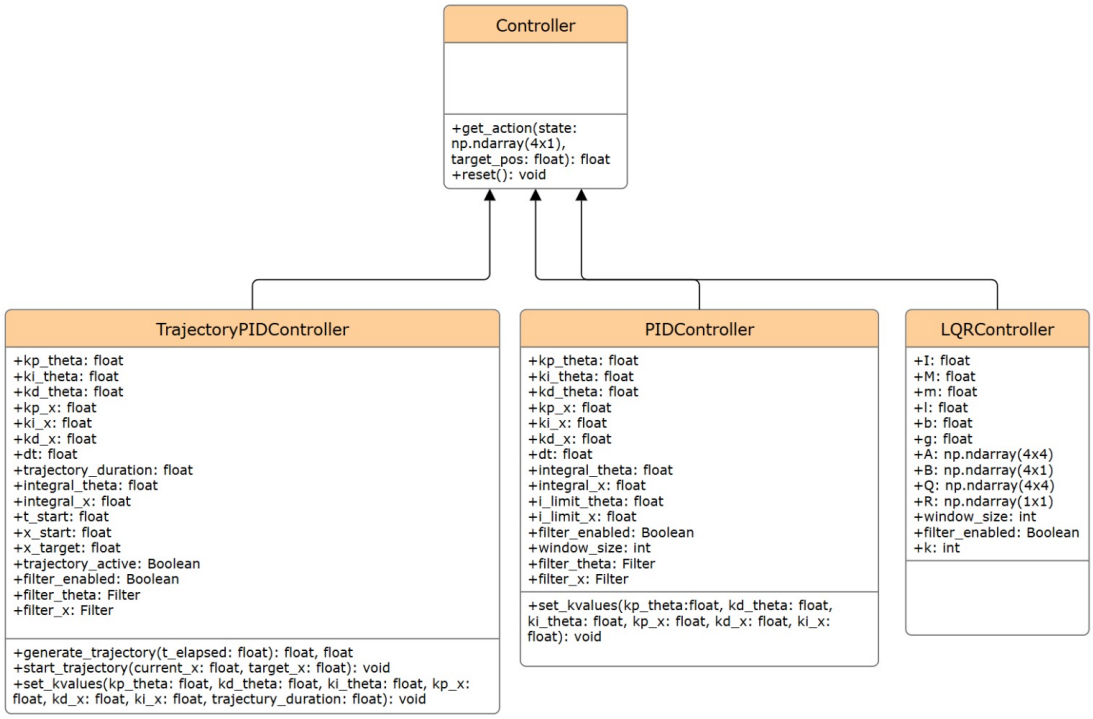


Figure 14: UML diagram for the controllers

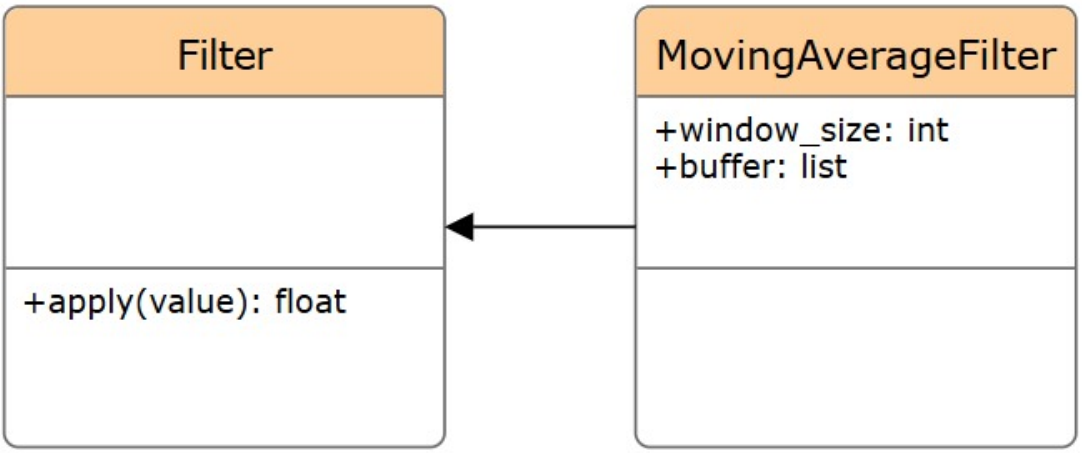
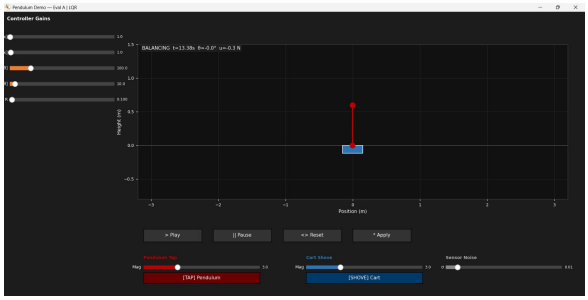


Figure 15: UML diagram for the filter

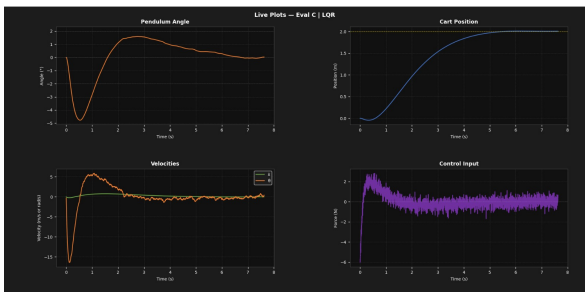
Our python implementation was built with OOP principles in mind. Our UML diagram can be seen in figures 13, 15 and 14. The external libraries we utilised were: `Numpy`, `scipy`, `matplotlib`, `control`, `queue`, `threading` and `abc`.

Using abstract classes for the controllers (Figure 14) and filters (Figure 15), we were able to design multiple controllers that could just be "plug and play" in the software. The abstract methods utilised polymorphism, enabling the simulation to only need to call specific function names which would be consistent across inherited classes. The `get_action function` is what the simulation calls to turn get the response from the controller for the current state, with the `apply` function being the equivalent for the filter classes.

The `pendulum` class (Figure 13) was where the actual simulation and dynamics took place. Firstly, the initialisation took all the system parameters defined earlier and stored them as attributes within the class. The `dynamics` function allowed a forward pass of the dynamics using the equations derived previously. The `step` function integrated the dynamics using a manual Runge-Kutta method[2]. `Disturbance` adds a disturbance to the system, either to the pendulum or to the cart. It simulates a disturbance by adding a random value to the velocity of the respective state that is being affected. This mimics a sudden impulse on the system. To mimic the noise sensors would have and the imperfectness of the simulation, we added noise to the simulation in `get_noisy_observation`. This creates normally distributed noise with mean 0, and crucially is done outside of the dynamics and step updates so that the noise does not mess with the physics, the noise is only applied when the simulation calls for the sensors action. Finally, the most important function, `simulate`, this runs the simulation for a chosen amount of steps, updating the cart's state at each time step, getting the control input and applying disturbances and noise. This function returns the state over every step and a tracker of all disturbances. Both of these are then given to the visualising function to present to the user



(a) Visualisation of the inverted pendulum robot with buttons and sliders to affect the simulation



(b) 4 plots detailing the simulation, from top left to bottom right: pendulum angle, cart position, angle velocity & cart velocity and control input

The visualisation and interactive elements come from the visualisation script. The main function is `visualize_trajectory_interactive`. This runs a simulation with the controller of your choice, gets the trajectories and disturbance log back. Two screens are produced, one with the visualisation of the trajectory (Figure 16a) on an inverted pendulum on a cart, with arrows showing disturbances when they occur. The other screen contains four plots (Figure 16b), all against time: pendulum angle (degrees), pendulum cart position (metres), angle and position velocities (two lines on one plot), and control input. All lines have a circle that traces the path along with the visualisation.
Our visualisation also had a selection of buttons and sliders. The sliders were for: target position (to switch between objective A and C seamlessly), sensor noise and disturbance. Additionally, depending on the controller being used, the gains could be adjusted on the `matplotlib` window using the slider. Finally, there were four buttons: play, pause, reset and apply. Play and pause started and stopped the simulation, apply added the new gains and settings from the sliders and reset reran the simulation with the new settings if you changed them or with the same ones.

A key challenge in the visualisation was maintain a responsive user interface while re-running computationally expensive simulations whenever the user adjusted a parameter. The approach if blocking the animation loop while the simulation recomputed froze the display entirely. To avoid this, the visualisation decouples the UI thread from the simulation thread using Python's `threading` and `queue` modules. When the user clicks *Apply*, the current slider values are placed onto a `queue.Queue` by the main thread. A dedicated background `simulation_worker` thread continuously polls this queue; on receiving a parameter set it recomputes the full trajectory, re-derives the LQR gains via the Riccati equation if necessary, and writes the resulting trajectory and control log back into a shared `sim_data` dictionary protected by a `threading.Lock`. A `threading.Event` flag (`update_ready`) is then set to signal the animation loop. On the next animation frame, this flag is checked, acquires the lock, swaps in the new data, rescales all four plot axes, and resets the playhead to $t = 0$.

5.5 Results and Benchmarking

This is the first of two benchmarking sections, which includes all simulation based performance analysis. We then extend these results to the physical hardware and discuss sim-to-real discrepancies. The two sections are complementary: here, we characterise the

controller behaviour in a repeatable environment, after which these results are validated against the real system.

All simulations use the verified physical parameters that we obtained; ($M = 1.3816\,\text{kg}$, $m = 0.05\,\text{kg}$, $l = 0.60\,\text{m}$, $b = 0.1$), a 4th-order Runge-Kutta integrator at $dt = 1\,\text{ms}$, and Gaussian sensor noise ($\sigma = 0.01\,\text{rad}$) applied to observations before the controller reads them, making it consistent with the encoder noise we identified on hardware. We conducted four experiments: controller comparison, noise sensitivity, filter window sensitivity, and since we found LQR to be the most reliable, we did a cost-matrix sensitivity test.

5.5.1 Experiment 1 Controller Comparison

PID and LQR were evaluated across all three evaluation scenarios. The settling band was defined as $|\theta| < 2°$ held continuously for 500 ms, and RMS metrics were computed post-settling where applicable.

Table 16: Simulation: Evaluation A disturbance rejection results

Controller	Disturbance	Peak $\lvert\theta\rvert$ (deg)	Settling (ms)	RMS θ (deg)	RMS Ctrl (N)
PID	Small tap	1.83	<1	0.280	1.016
PID	Large tap	4.53	286	0.686	2.462
PID	Cart shove	0.14	<1	0.056	0.269
LQR	Small tap	4.10	1455	1.350	1.381
LQR	Large tap	10.14	1945	3.365	3.428
LQR	Cart shove	6.65	1668	2.203	0.897

Disturbances applied at $t = 3\,s$ as impulses: small tap $F_\theta = 800\,N$, large tap $F_\theta = 2000\,N$, cart shove $F_x = 1500\,N$. "<1" = peak never exceeded the 2° band.

Evaluation A - Disturbance Rejection. The cascaded PID controller demonstrates superior disturbance rejection in simulation. For the small tap and cart shove cases, the peak angle (1.83° and 0.14° respectively) never exceeds the 2° settling band. The disturbance is absorbed within a single control period without a measurable transient. For the cart shove specifically, PID's cascade architecture provides a natural advantage: the outer position loop counteracts the cart displacement directly, almost entirely decoupling it from the inner angle loop. The large tap is settled in 286 ms with a peak of 4.53°.

LQR shows higher peaks and substantially longer settling times across all three cases. The cart shove is particularly revealing: LQR produces a 6.65° peak (versus 0.14° for PID) because the full-state feedback law couples cart position correction with angle correction The sudden change in cart velocity triggers a position error that the LQR gain matrix translates into an angle-destabilising corrective force. This is not a failure of LQR as such, but a consequence of the current gain configuration ($k_x = 0$, $k_{\dot{x}} = 0$): with position gains disabled for Evaluations A and B, the LQR is operating without one of its state channels, making it structurally more similar to a pure angle controller. Enabling position gains would reduce the cart shove response at the cost of increased control effort.

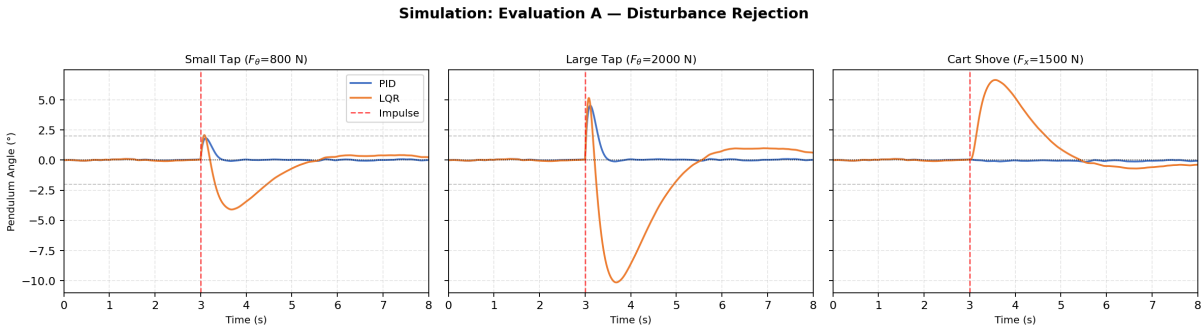


Table 17: Simulation: Evaluation B recovery results

Controller	θ_0 (deg)	Settling (ms)	Peak (deg)	RMS θ (post) (deg)	SS Error (deg)	RMS Ctrl (post) (N)
PID	5	184	5.00	0.159	0.053	0.315
PID	10	250	10.00	0.152	0.075	0.417
PID	15	281	15.00	0.164	0.106	0.501
LQR	5	214	5.00	0.439	0.250	0.221
LQR	10	1234	10.00	0.476	0.279	0.174
LQR	15	1548	15.00	0.468	0.318	0.165

Post-metrics computed from settling point onwards. No disturbances applied.

Evaluation B - Deep Fall Recovery. PID settling times scale approximately linearly with start angle: 184 ms at 5°, 250 ms at 10°, 281 ms at 15° which is around 1.5× increase from 5° to 15°. Post-settling RMS θ is consistently tight (0.15–0.16°) and nearly independent of start angle.

LQR settling times scale nonlinearly: 214 ms at 5°, 1234 ms at 10°, 1548 ms at 15°, roughly a 7.2× increase from 5° to 15°. This nonlinear degradation is the expected consequence of operating further from the linearisation point. The LQR gain matrix was computed for the system linearised at $\phi = 0$; as the initial angle increases, the small-angle approximation $\sin\phi \approx \phi$ introduces growing model error, and the fixed gain matrix becomes increasingly suboptimal. At 5° the approximation error is ≈0.4%, well within the regime where the linear controller performs well. At 15° the error reaches ≈3.4%, which is sufficient to meaningfully slow recovery.

However, LQR uses significantly less control effort post-settling: 0.165-0.221 N versus 0.315–0.501 N for PID — a reduction of approximately **44–48%**. This is a direct consequence of the **R** term in the LQR cost function, which penalises large control signals, and represents a genuine engineering advantage of the optimal control formulation.

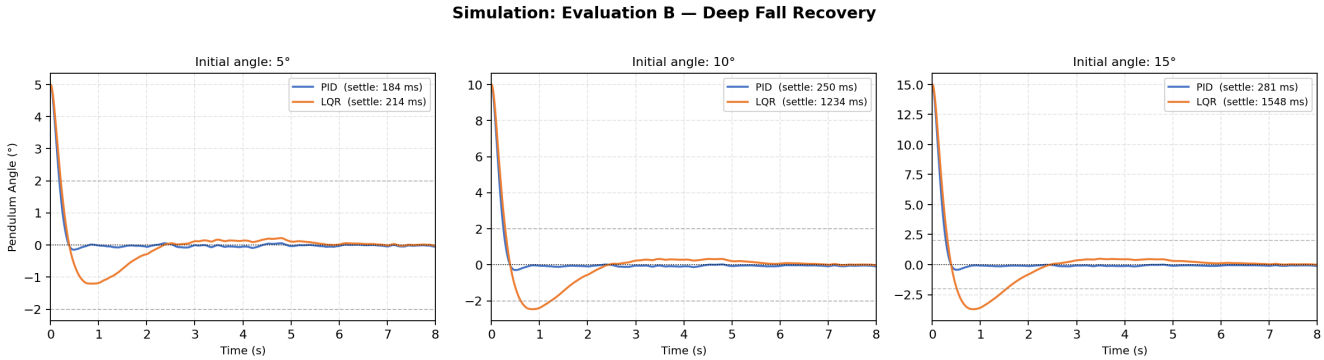


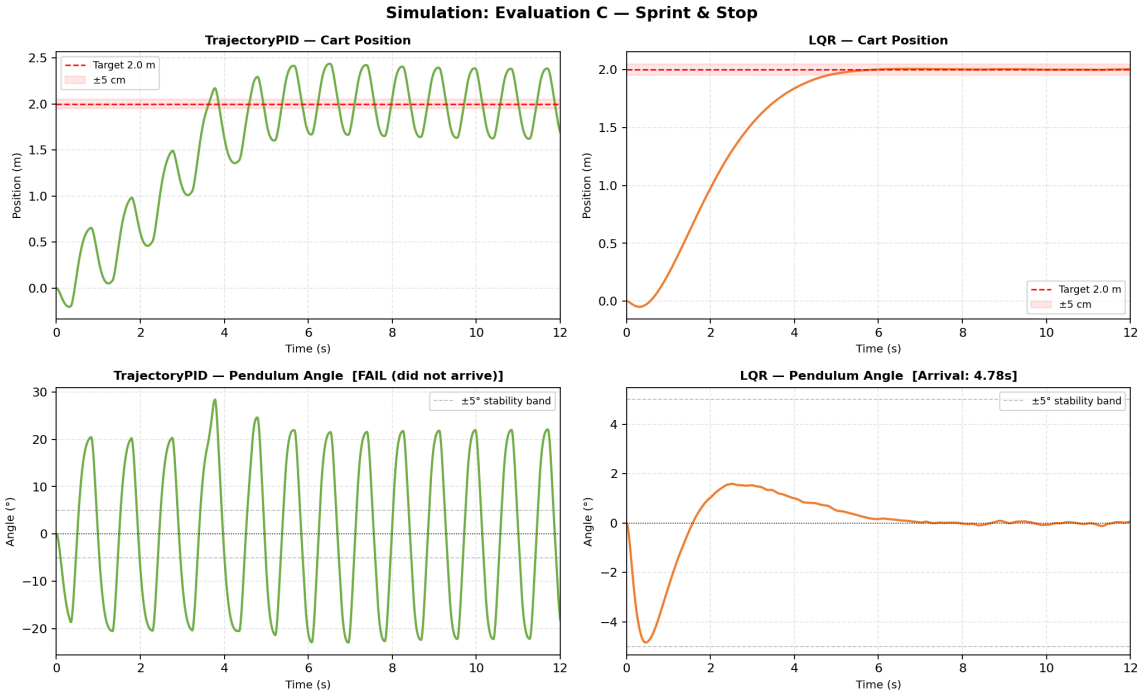
Table 18: Simulation: Evaluation C sprint results

Controller	Arrival (s)	Final x (m)	Pos. error (cm)	Final θ (deg)	RMS θ (deg)	Max θ (deg)	Result
Traj. PID	—	1.69	−30.8	−18.1	15.79	28.40	**FAIL**
LQR	4.78	1.95	−5.0	0.70	2.03	4.84	**PASS**

Arrival: $|x - 2.0| < 5$ *cm and* $|\theta| < 5°$ *held for 500 ms.*

Evaluation C - Sprint & Stop. LQR successfully completes the sprint in 4.78 s, stopping within 5.0 cm of the target with a final angle of 0.70°. The RMS θ of 2.03° and maximum of 4.84° confirm that the pendulum remained well within the linear operating region throughout.

The Trajectory PID controller fails to complete the sprint, reaching only 1.69 m before becoming unstable (peak 28.40°). The root cause is a specific software design limitation: the controller contains an early-exit condition (`if abs(x - target) < 0.1: return 0.0`) intended to suppress overshoot, but this halts motor output prematurely when the cart is within 10 cm of the target, at which point the pendulum has residual angular momentum that the suddenly-inactive controller cannot arrest. The trajectory planning mechanism itself (cubic S-curve over 5 s) is structurally sound - the cart accelerates smoothly to ∼1.69 m before the instability - and the failure would be resolved by replacing the hard cutoff with a smooth deceleration law.



5.5.2 Experiment 2 - Noise Sensitivity

Both controllers were run on the Evaluation B 10° recovery task with five sensor noise levels spanning zero (ideal) to 10× the hardware-estimated value ($\sigma = 0.10$ rad).

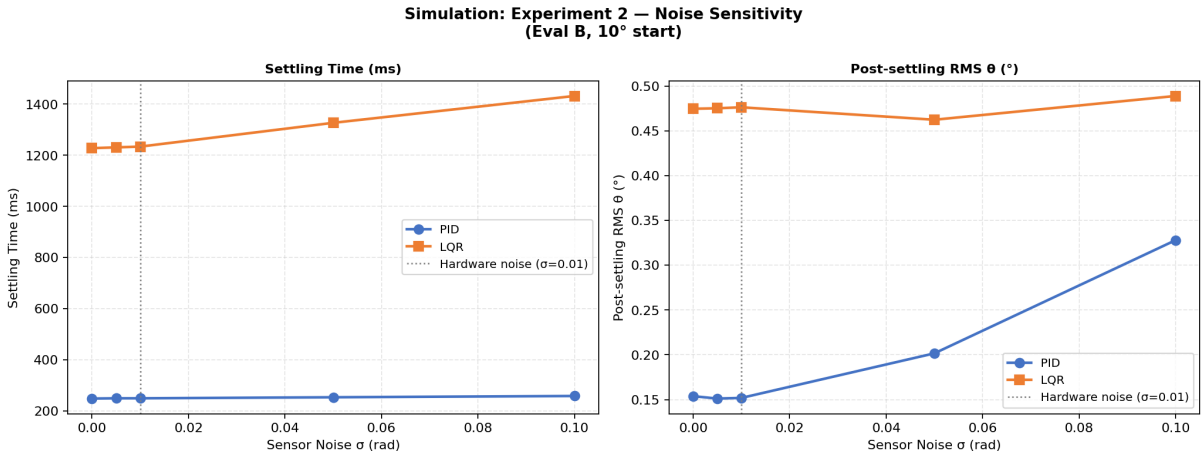
Table 19: Simulation: Noise sensitivity (Eval B, 10° start)

Controller	Noise σ (rad)	Settling (ms)	RMS θ (deg)	SS Error (deg)
PID	0.000	249	0.154	0.084
PID	0.005	250	0.151	0.079
PID	0.010	250	0.152	0.075
PID	0.050	254	0.202	0.135
PID	0.100	259	0.328	0.254
LQR	0.000	1228	0.475	0.284
LQR	0.005	1231	0.475	0.281
LQR	0.010	1234	0.476	0.279
LQR	0.050	1327	0.462	0.283
LQR	0.100	1432	0.489	0.333

Hardware noise level estimated at $\sigma \approx 0.01$ rad from encoder resolution and observed steady-state oscillation.

Both controllers are robust across the hardware-realistic noise range (σ = 0–0.01 rad): settling time and RMS θ change by less than 1% across this range for both PID and LQR. This validates the noise model used in simulation and confirms that the moving average filter is effective at the hardware noise level.

At σ = 0.05 rad (5× hardware noise) performance begins to degrade noticeably: PID steady-state error increases from 0.075° to 0.135° (+80%), and LQR settling time increases from 1234 ms to 1327 ms (+7.5%). At σ = 0.10 rad both controllers show meaningful degradation. This establishes the noise robustness margin of the current filter design: the system can tolerate approximately 5× the expected hardware noise before performance degrades significantly, providing a comfortable safety margin.



5.5.3 Experiment 3 - Moving Average Filter Window Size

The effect of filter window size on control performance was characterised by running Evaluation B at 10° with five window sizes from 1 (unfiltered) to 50 (hardware setting).

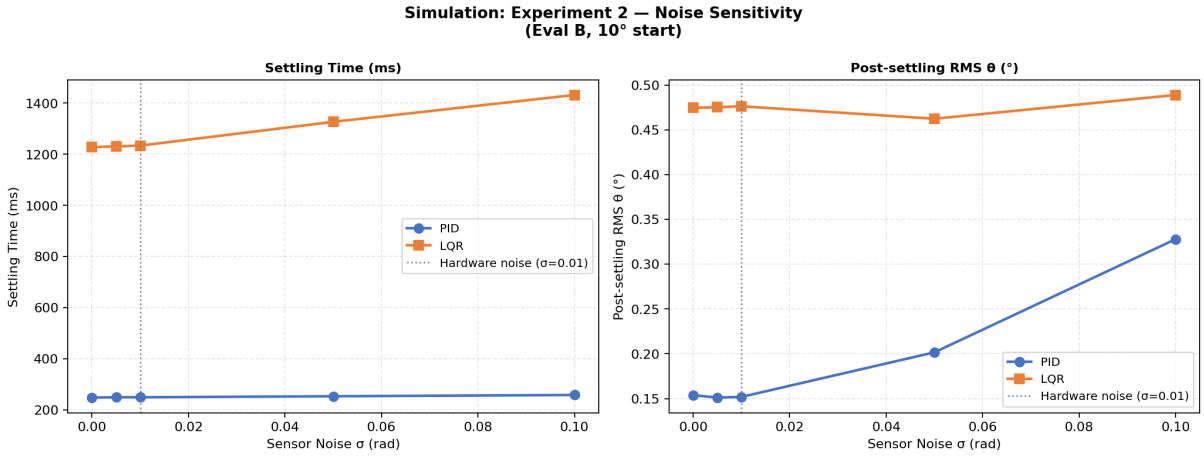
Table 20: Simulation: Filter window size sensitivity (Eval B, 10°, $\sigma = 0.01$ rad)

Controller	Window	Settling (ms)	RMS θ (deg)	RMS Ctrl (N)
PID	1	283	0.171	1.252
PID	5	280	0.168	0.610
PID	10	276	0.166	0.482
PID	20	269	0.160	0.410
PID	50	250	0.152	0.417
LQR	1	1369	0.461	0.820
LQR	5	1358	0.462	0.376
LQR	10	1345	0.463	0.283
LQR	20	1320	0.464	0.221
LQR	50	1234	0.476	0.174

The most significant effect of window size is on **RMS control effort**. For LQR, increasing the window from 1 to 50 reduces RMS control effort from 0.820 N to 0.174 N which is roughly a 79% reduction because the filter suppresses high-frequency noise before it enters the derivative computation, preventing the controller from issuing rapid corrective impulses in response to sensor artefacts.

For PID, the effect on settling time is more pronounced: window=50 settles in 250 ms versus 283 ms for window=1. This reflects the derivative term's sensitivity to noise larger windows produce a smoother $\dot{\theta}$ estimate, allowing the k_d term to provide effective damping rather than noise amplification.

The window=5 to window=10 range represents the best trade-off between noise rejection and phase lag (the averaging filter introduces a delay of approximately window/2 $\times$ dt, which is 2.5 ms at window=5 and 25 ms at window=50). The hardware setting of window=50 was chosen conservatively to prioritise smooth motor commands, at the cost of a 25 ms additional lag — well within the control bandwidth.



5.5.4 Experiment 4 - LQR Cost Matrix Sensitivity

The sensitivity of LQR performance to the angle penalty $Q_{[2,2]}$ was explored by varying it from 10 to 500 while keeping all other cost weights fixed. This directly controls the resulting feedback gain k_θ.

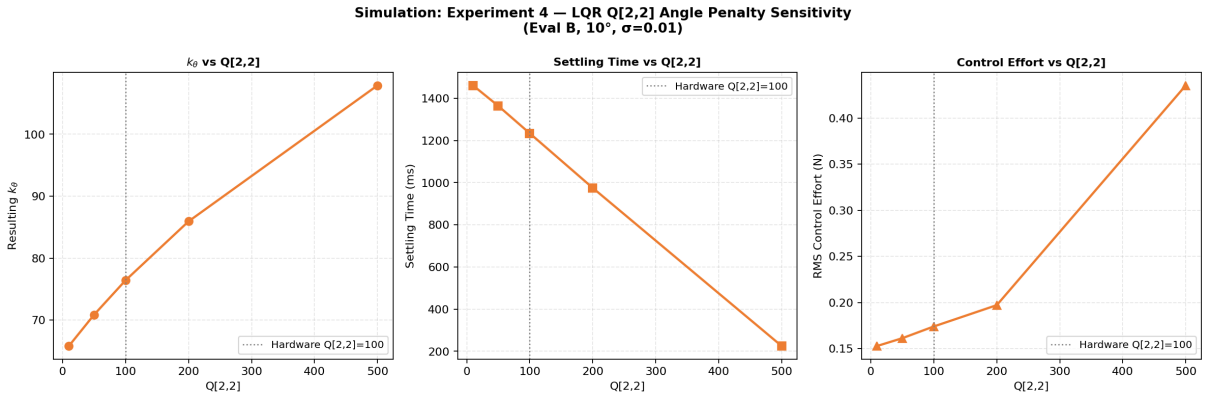
Table 21: Simulation: LQR $Q_{[2,2]}$ sensitivity (Eval B, 10°, $\sigma = 0.01$ rad)

$Q_{[2,2]}$	k_θ	Settling (ms)	RMS θ (deg)	RMS Ctrl (N)
10	65.82	1461	0.447	0.153
50	70.86	1364	0.451	0.161
100	76.42	1234	0.476	0.174
200	85.93	975	0.530	0.197
500	107.87	225	0.558	0.436

$Q_{[2,2]} = 100$ is the value used in hardware ($k_\theta = 76.42$, perturbed to 700 for hardware deployment).

Increasing $Q_{[2,2]}$ monotonically reduces settling time at the cost of increased control effort and slightly higher post-settling RMS θ (because more aggressive corrections introduce more closed-loop noise). At $Q_{[2,2]} = 500$ ($k_\theta = 107.87$), settling time reaches 225 ms, which is competitive with PID, but at the cost of a 2.5× increase in control effort relative to $Q_{[2,2]} = 100$.

The hardware LQR uses $k_\theta = 700$ which is significantly higher than the $k_\theta = 76.42$ computed by the Riccati solver for $Q_{[2,2]} = 100$. This perturbation was applied empirically during hardware tuning to compensate for the motor deadband, structural compliance, and the reduced effective gain caused by the +200 unit motor bias in `forceToMotorSpeed`. The $Q_{[2,2]}$ sensitivity analysis confirms that this perturbation is in the right direction: higher k_θ does improve settling time, and the simulation predicts this should come with modest control effort increases, consistent with what was observed on hardware.

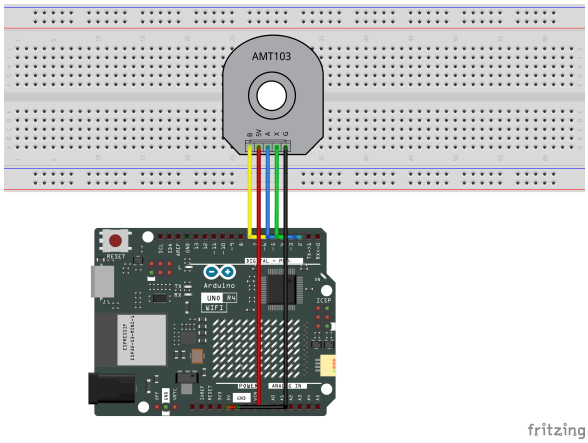


6 Prototyping

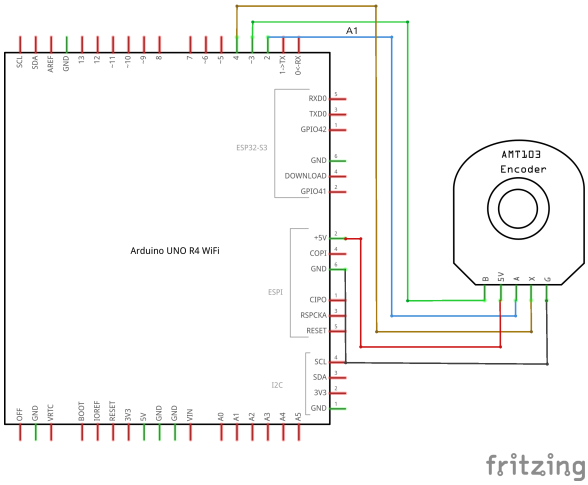
6.1 Electrical Diagram (Schematics)

The MCU we initially used was an Arduino Uno R4 WiFi before switching to the Arduino Giga R1 WiFi, as the Uno did not have adequate processing capability to run the code

for our controller models. Each subsystem is connected to the MCU but has no direct connection to other subsystems. The subsystem diagrams are presented in isolation with the MCU to better understand how they connect.

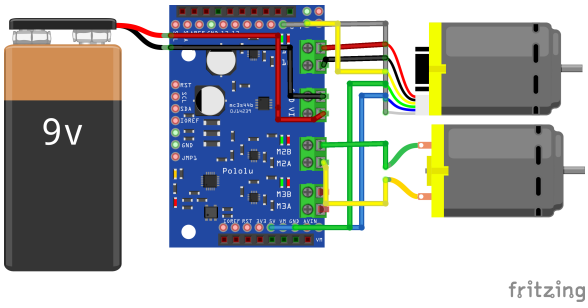


(a) Diagram of the optical encoder subsystem

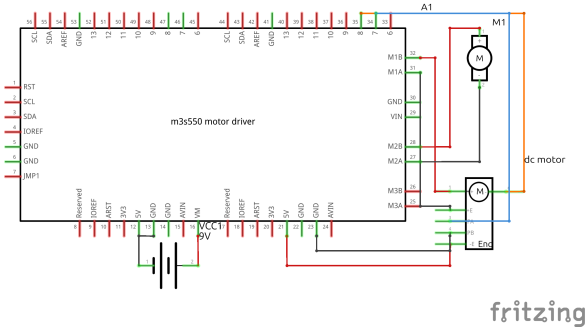


(b) Schematic diagram of the optical encoder subsystem

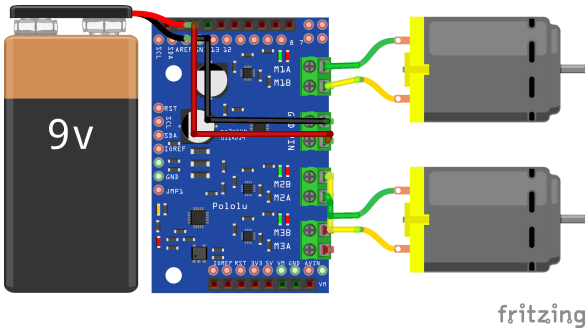
For the 4-motor configuration, we decided to address the motors such that they are paired as the front and back motors, with one of the motors having their encoder con



(a) Diagram for the pair of motors with a configured encoder



(b) Schematic diagram for the pair of motors with a configured encoder



(a) Diagram of the optical encoder subsystem

(b) Schematic diagram of the optical encoder subsystem

6.1.1 Electrical Bill Of Materials

Table 22: Electrical Bill of Materials

Item Description	Quantity
Metal Gearmotor (25Dx63L mm)	4
Arduino Giga R1 WIFI	1
Broadcom AS22 Optical Encoder	1
Motoron M3S550 Dual Motor Driver	2
Breadboard (Includes terminals)	1
Jumper Wire Assortment (Various lengths)	Many

6.2 Mechanical (CAD)

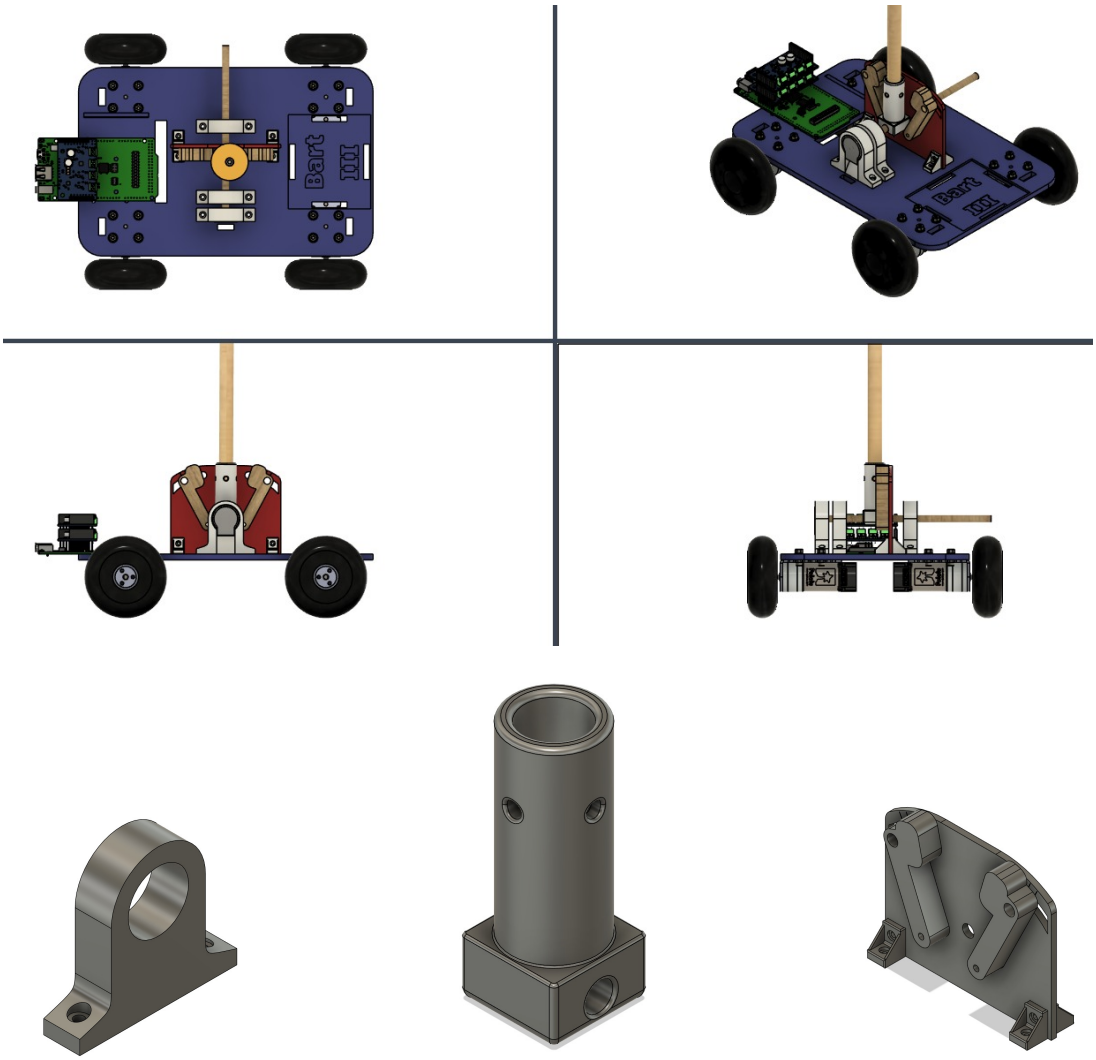
The mechanical design of the cart was developed over multiple iterations of design and implementation. The following figures show the final design of the cart itself as well as the main parts for the inverted pendulum cart.

Bearing Mounts: The bearing mounts were developed to house the bearings to allow them to rotate with the pendulum

Pendulum Mount: The pendulum mount was developed to hold the pendulum rod. It was designed so that it could be secured in place using a nut and bolt. Additionally, a dowel could be inserted at the bottom to connect with the bearings. This decision was made as printing the rods along with the pendulum mount itself lead to deformations and warping which lead to uneven rotations causing friction to not be uniform and minimal.

Motor Mounts: It was decided that prefabricated motor mounts were to be used for the cart. This meant that less time was spent on designing the mounts for the motors and instead could be focused towards other aspects of the cart

V-Block: The v-block was designed so that the pendulum can rest at 3 different angles (5°, 10°, 15°). The main objective of the v-block was for evaluation B, where the cart attempts to stabilise the pendulum from a preset angle that is not 0°. A second purpose of this v-block was to also act as a blocker to prevent the pendulum from falling onto the Arduino Giga or the battery which could potentially cause damage to those components.



(a) Main bearing mounts (b) Pendulum attachment (c) V-block component

Figure 20: Design and fabricated components of the inverted pendulum cart: (top) CAD rendering of the cart system, (a) 3D printed bearing mounts, (b) laser-cut pendulum mount, and (c) fabricated V-block component.

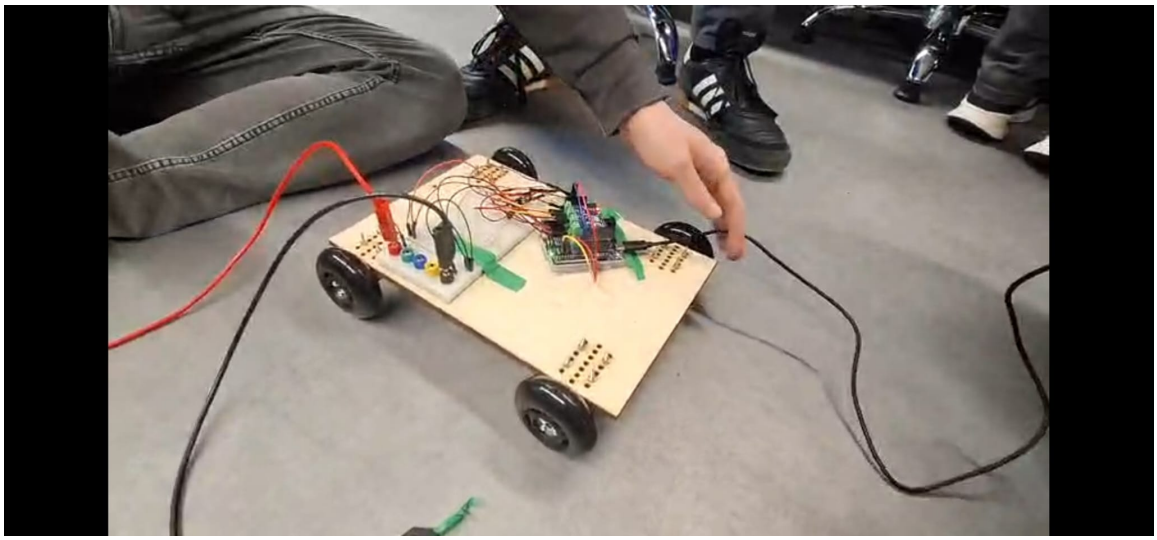
6.2.1 Iterative Development

Iteration 1: This model was a basic wooden board with holes to mount the motors. This iteration was used to test the motors as well as see how large other components needed to be. Testing with this model involved powering the motors and seeing the cart move in basic directions such as forwards and backwards.

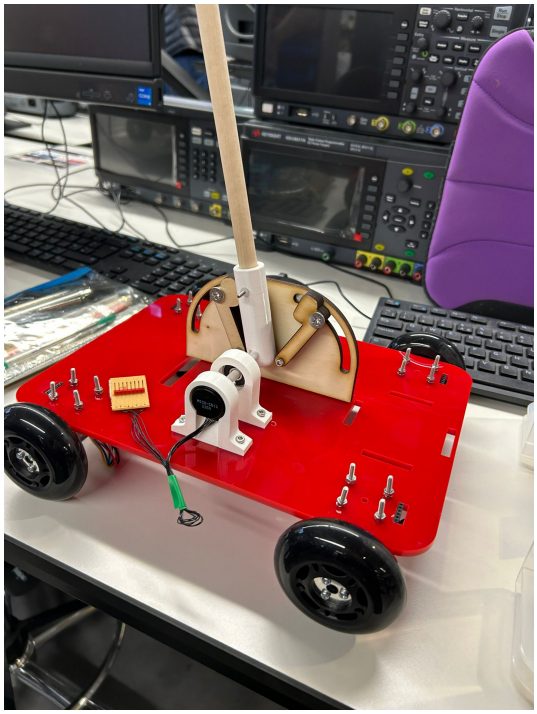
Iteration 2: This model added more mounting holes as well as the first versions of the pendulum mount and v-block. From here, it was found that the components caused a variety of issues. The pendulum mount beams were 3D printed which made them weak to load. This caused them to not rotate equally when inserted into the bearing which introduced unnecessary friction. The v-block size was also reduced to help save material as well as reduce the overall size of the cart.

Iteration 3: The final iteration implemented the smaller chassis design as well as the new components. This version of the cart also had all the electronics connected to it

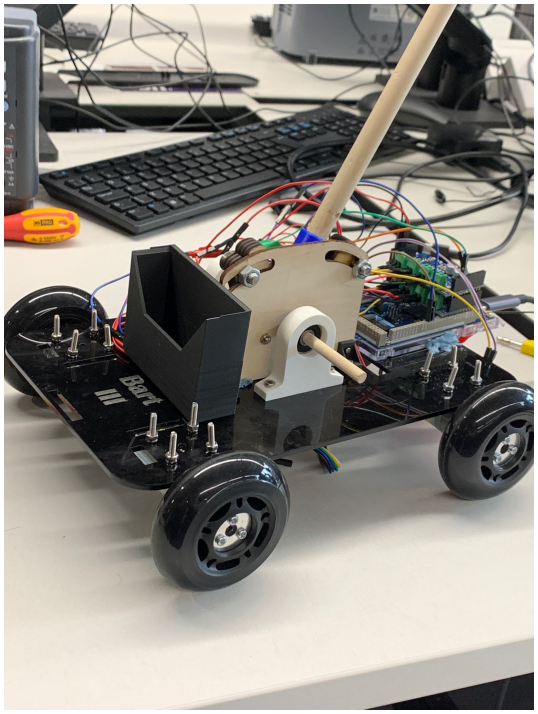
becoming the main model for testing. A wooden dowel was used as the pivot rather than the 3D printed dowel to help reduce the bending of the beam. This was further reduced by decreasing the distance between the two bearings.



Cart Iteration 1



Cart Iteration 2



Final Prototype

Figure 21: Development and testing phases of the inverted pendulum cart

6.2.2 Mechanical Bill Of Materials

Table 23: Bill of Materials (BOM) for the Inverted Pendulum Cart

Item Description	Quantity
M3 Bolts and Nuts	24
M2 Bolts and Nuts	8
M2.5 Bolt and Nut	1
M6 Bolts and Nuts	2
M4 Phillips Screw	1
3mm Acrylic Sheet	1
Scooter Wheels (70x25 mm)	4
Motor Mounts	4
Pendulum Mounts (3D Printed)	2
Encoder Mount (3D Printed)	1
Pendulum Holder (3D Printed)	1
Dowel (6mm diameter, 15mm length)	1
Pendulum Rod (12mm diameter, 60mm length)	1
50g Mass	1

6.3 Experimental Identification of Pivot Damping Parameters

To ensure the assembly meets the low-friction requirement ($\zeta < 0.01$), a free-decay test was performed. With the cart mass $M = 1.381\,\text{kg}$ held stationary ($\ddot{x} = 0$), the pendulum's decay profile was analyzed to calculate both the dimensionless damping ratio and the viscous damping coefficient.

6.3.1 Experimental Observations

The following data were recorded during the free-oscillation test:

- **Initial Displacement:** $\theta_0 = 10^\circ$
- **Final Displacement:** $\theta_n = 5^\circ$ (after $n = 29$ cycles)
- **Test Duration:** $\Delta t = 44\,\text{s}$
- **System Mass (Cart):** $M = 1.381\,\text{kg}$

6.3.2 Logarithmic Decrement and Damping Ratio

The logarithmic decrement δ characterizes the exponential decay of the oscillations:

$$\delta = \frac{1}{n} \ln\left(\frac{\theta_0}{\theta_n}\right) = \frac{1}{29} \ln\left(\frac{10}{5}\right) \approx 0.0239 \tag{27}$$

For an underdamped system where $\zeta \ll 1$, the damping ratio ζ is determined by:

$$\zeta = \frac{\delta}{\sqrt{4\pi^2 + \delta^2}} \approx \frac{\delta}{2\pi} \tag{28}$$

$$\zeta \approx \frac{0.0239}{2\pi} \approx 0.0038 \tag{29}$$

6.3.3 Effect of Mass on Damping Coefficient

While the damping ratio ζ is a dimensionless measure of decay rate, the physical damping coefficient c is a function of the system's inertia. Given the natural frequency $\omega_n = \frac{2\pi n}{\Delta t} \approx 4.14\,\text{rad/s}$, the damping coefficient for the pivot is related to the mass and length (L) of the pendulum:

$$c = 2\zeta\sqrt{Ig} \quad \text{or} \quad c = 2\zeta\omega_n I \tag{30}$$

The calculated $\zeta = 0.0038$ confirms the pivot meets the design requirement of $\zeta < 0.01$.

6.3.4 Compliance and Future Refinements

The calculated damping ratio of $\zeta \approx 0.0038$ is well within the specified design constraint of $\zeta < 0.01$.

6.4 Control Algorithms

When we took readings from the encoder, we found that it was very accurate with minimal noise, therefore we lowered the moving average down to 5 from 25 in the simulation.

All controllers had serial commands, as can be seen in table 24, that could stop the robot, start the robot and change its gains, to minimise valuable lab time spent waiting for code to compile and send.

Table 24: Serial commands we could send our robot

Command	Description
SET	Set gains for controller
STOP	Stop the motors
GAINS	Display the current gains in the serial monitor
RESET	Restart the controller
START	Turn the controller on
RESET_MOTOR	Reset motor encoder
JERK	Set jerk values for objective B

We also found that the control would output motor values that were too low for the motors to run, however these low values were when the pendulum was just tipping, the most crucial time for the control to kick in. Our resolution was to increase the control input enough so the motors always effect the pendulum.

6.4.1 PID

The PID controller was programmed into the robot with commands to adjust the proportional, integral, and derivative gains for both the pendulum angle, and the cart position. This design operates through manual gain adjustment due to the physical characteristics of the system making the purely simulated results unreliable. When implemented into the practical system, this controller was found to be too aggressive regardless of tuning attempts to fix the issue, leading to the focus on LQR for practical running of the system.

6.4.2 LQR

The model derived earlier was programmed into the simulation, we took the gains from there that we were happy with and then store those gains in the controller. Crucially, the Arduino is not doing the Ricatti equations, they are performed offline and then inputted to the controller. This design decision was made to simplify the Arduino code on the robot and to help the robot stick closer to the simulation which we know works. However, when transferring gains from the simulation to the robot, we had to perturb the gains slightly to handle the sim to real gap.

6.4.3 Objective B

For objective B, the motors were not powerful enough to raise the pendulum using the same controllers as for objective A without any changes. We came up with an algorithm that runs in two stages: first the robot performs a "jerk" motion, it runs at full power in one direction, and then suddenly switches direction or stops. This sudden jerk forces the pendulum upright for the LQR or PID to then kick in when the pendulum angle is within a threshold.

6.4.4 Objective C

For objective C, we found that the same gains for objective A were not working for objective C when put into the robot. Our solution was to have a gain scheduling algorithm: one set of gains get the cart to the 2 metre line with pendulum control and then the balancing gains take over, which were recycled with only minor perturbations from objective B, with the idea that because B starts at an angle, using gains for roughly the angle it will arrive at will cause the robot to balance.

6.5 Benchmarking and Results

6.5.1 Methodology and Data Collection Status

We scheduled formal benchmarking with the physical system to be done on the 26^{th} of March 2026. However, due to a scheduling miscommunication, we were unable to collect structured performance data during this session and the system was decommissioned as required. As a result, we have written the qualitative data that we gathered during pre-demonstration tuning sessions rather than formally measured metrics. We believe that the qualitative data that we have recorded will remain valid as a performance characterisation of the control algorithms under the verified physical parameters.

The following hardware parameters governed system behaviour throughout testing and the demonstration:

- Supply voltage: 10.8 V (rated 12.3 V), giving a force reduction factor of $V_{\text{ratio}} = 10.8/12.3 = 0.878$
- Motor deadband: ± 200 raw units out of ± 800, equivalent to a ± 2.70 N dead zone in which no cart motion occurs
- Hardware LQR gains: $k_\theta = 700$, $k_{\dot{\theta}} = 2250$, bench-tuned from the Riccati-computed values used in simulation
- Jerk force: 750 units = 10.13 N applied for 200 ms (6.5° case) or 230 ms (10° case), followed by a 40 ms coast phase before LQR handover

6.5.2 Evaluation A - Disturbance Rejection

LQR performance. The LQR controller successfully stabilised the pendulum in the upright position during pre-demonstration tuning sessions. Despite visible cart jitter from rapid motor reversals near the equilibrium point, the controller maintained the pendulum angle within an acceptable range for several seconds before drift-driven failure. The cart drifted laterally over this period, which was a consequence of the angle-only control law ($k_x = k_{\dot{x}} = 0$) having no mechanism to correct cart position. When a light tap was applied to the pendulum mid-run, the LQR recovered and returned the pendulum to near-upright, confirming that the eventual failure is driven by the underlying cart drift rather than sensitivity to disturbances.

PID performance. The PID controller failed to stabilise the pendulum on hardware. Rather than converging to an upright equilibrium, the system oscillated with increasing amplitude before reaching the 20° fall threshold. The PID output, after voltage scaling and deadband clipping, could not deliver the small corrective forces required near equilibrium, and the integral term wound up without producing effective motor response.

6.5.3 Evaluation B - Deep Fall Recovery

LQR swing-up performance. The jerk-to-LQR handover sequence was executed at both V-block angles ($5° - 7°$ and 10°). Following the open-loop jerk phase, the pendulum was raised toward the upright position and the LQR controller activated once $|\theta|$ fell within the 0.35 rad capture threshold. After capture, the system exhibited the same behaviour as Evaluation A: stable balancing for several seconds, followed by drift-driven failure. Both angles were successfully captured and held during pre-demonstration tuning sessions.

PID swing-up performance. The PID controller failed to stabilise after the jerk phase at both starting angles. The controller produced large oscillatory motor commands but could not converge to a stable equilibrium, and the pendulum fell past 20° shortly after the jerk phase completed. The behaviour was consistent across repeated attempts.

6.5.4 Evaluation C – Sprint & Stop

Evaluation C was not completed during the final demonstration session due to a motor mount coming loose under repeated cyclic load. In pre-demonstration tuning sessions, the hardware LQR successfully traversed approximately 1.8–2.0 m while maintaining pendulum balance, with conservative position gains ($k_x = 5.0$, $k_{\dot{x}} = 3.0$) activated via the `SPRINT` serial command.

6.5.5 Comparative Analysis and Failure Discussion

Table 25: Qualitative comparison of LQR vs PID across all evaluations

Metric	LQR	PID
Eval A - stability	Stable for several seconds, limited by cart drift	Falls immediately, oscillates to 20°
Eval A - tap recovery	Recovers, continues balancing	N/A (already unstable)
Eval B - swing-up	Successful capture at both angles	Oscillates, no stable capture
Eval B - post-capture	Same drift pattern as Eval A	Falls within seconds
Eval C - sprint	Partially achieved in tuning (≈2 m)	Not attempted
Hardware deployment	Successful, gains perturbed from simulation	Failed: deadband prevents stabilisation

The LQR controller achieved the primary objectives of Evaluations A and B under observed conditions. The stability window observed in tuning was shorter than anticipated from the nominal simulation, which can be attributed to four compounding hardware factors:

1. **Voltage reduction** (10.8 V vs 12.3 V rated) reduces maximum corrective force by 12.2%, limiting the LQR's authority at larger angles.
2. **Motor deadband** (25% of full range) means small corrective commands produce no motion, allowing slow position error to accumulate without correction.
3. **Structural compliance** of the acrylic chassis and wooden dowel pivot introduces a slow mechanical offset that grows over time and is indistinguishable from a genuine angle error by the angle-only LQR controller.
4. **Absence of position feedback** ($k_x = k_{\dot{x}} = 0$ for Evaluations A and B) means the LQR cannot correct the slow cart drift that accompanies the growing angle offset.

The PID controller's complete failure on hardware despite reasonable simulation performance is explained by the deadband mechanism. The PID integral term winds up in response to persistent small errors, but the resulting large control commands are clipped by the motor saturation limit while the deadband prevents the smaller intermediate corrections from being applied at all. The LQR's simpler angle-only feedback structure is more tolerant of this nonlinearity because it does not accumulate integral state.

Due to the absence of formally collected benchmarking data, a complete quantitative sim-to-real comparison cannot be made for this project. However, the qualitative hardware behaviour is consistent with the simulation predictions: LQR outperforms PID in all evaluated scenarios, the primary failure mode on hardware is cart drift rather than angle instability, and the sim-to-real gap is well-explained by the four hardware factors

identified above. This represents a meaningful V-Model verification outcome: the simulation correctly predicted controller ranking and failure modes, even if formal acceptance criteria could not be evaluated against measured data.

7 Conclusions

Over the last 11 weeks, the team has managed to design, fabricate and program an inverted pendulum cart to achieve the difficult task of stabilising an unstable pendulum. This project has allowed for many skills taught previously as well as the current skills learnt in the module to be applied into the development of the cart.

7.1 Findings

7.1.1 Microcontroller choice and clock speed

We initially designed our system to work with the arduino uno R4 microcontroller due to its simple implementation and low profile. The problem with this decision became obvious once we moved into testing the controller on the system; this issue was that the microcontroller response time was too low for practical implementation of the control script with our design and choice of motors. This lead us to port our system to the more powerful arduino Giga R1, which has both a higher memory, allowing for more complex design, but more notably, has a much higher clock speed than an uno R4 - the uno R4 has a clock speed of 48mHz vs the giga R1 with a speed of 480mHz. This ten times speed increase leads to much more attentive results and easier practical implementation of control design. Our late move did create its own issues due to losing precious time to redesigning some electronic elements for the giga and rewriting the code for the new micro controller.

7.1.2 Part placement

To maximise sustainability, components were made to be as close as possible. The primary area was for the bearing and encoder mounts. This was to reduce the length of the beam to reduce any major bending. However, in doing this, it became extremely difficult to access other parts of the assembly. One such example was the pendulum bolt which was used to secure the dowel in place. Since all the parts were too close together, the bolt could not be pulled out which meant that we could never take the entire pendulum out of the assembly.

7.1.3 Tolerances

During development, many tolerances were off. This leads to parts being either too tight or too loose of a fit. This meant that is was extremely difficult to remove or change certain parts and made servicing components problematic as well.

7.2 Challenges

7.2.1 Vibrations

During the testing of the system, we found that a major obstacle to both mechanical stability and controller performance was the passive vibrations within the system. The primary source of this vibration was through insecure mounting of some components like the pendulum in the pendulum holder, and through the driving motors in the underside of the cart. We found that any simple control motion lead to very high levels of vibration due to the rapid change in cart direction near the equilibrium point. This lead to problems firstly in mechanical stability, most obviously with high stresses on the baseplate and in loosening of nuts on the mounts of the major components. This could have been partially mitigated through the use of washers and nylock nuts for the mountings, which we did implements after the issue was observed, however the issue of base stresses was a continual issue due to a lack of thicker board when the laser cutting was complete and the time commitment of a full disassembly and reassembly being too high to be practical.

7.2.2 Tuning on physical system

A key challenge that we had once we had finished the assembly and electrical implementation of the system was in tuning the LQR parameters in the controller model to allow for suitable control for the tasks. We initially tried to use the numbers that we derived through the simulation via the Riccati equation; this was, however, unsuccessful due to the difficulties in simulating the physical characteristics of the system, particularly in the motors. We also found that the large passive vibration in the system created much higher and less predictable disturbance than simulated. These compounding factors lead to us having to tuning the parameters primarily by hand and through mental mathematical adjustments and visual response to changes in control.

7.2.3 The encoder

The encoder proved to be problematic during the final weeks of development. During testing with the R4, there were no issues with the encoder. However, once we switched to the giga, issues began to arise. The encoder would no longer return values. This meant that the cart could not detect when the pendulum was falling essentially rendering the cart incapable of stabilising the pendulum. Multiple checks were made such as shorts and cable issues. A new encoder was also procured which did not seem to help. The final conclusion to the issue was that the sensor plate was a few micrometers too close to the encoder disk which caused it to be out of focus and unable to make readings. Once the encoder was taken apart and put together again using correct distances, it performed as it had originally.

7.3 Potential Improvements

If we were to revisit this project, there would be a multitude of improvements that would be made to ensure the outcome was met to a higher standard.

Firstly, a more powerful microcontroller than our initial arduino uno R4 would be incorporated from the beginning of the design. This would allow for the physical system to better accommodate the size of the microcontroller and be designed with the dimensions of this

controller in mind when designing the mounting on the base plate. Using a more powerful microcontroller from the beginning would also have been helpful in reducing wasted time with porting the electronics from one board to another and rewriting elements of the script for the new libraries of the Giga.

Furthermore, more time would need to be allocated to tuning and testing. This stage proved to be the biggest setback as many issues and problems came during this time. By allocating more time to this area, the cart would have much better tuned values, which would enable it to stabilise the pendulum much more effectively. Most vitally, the limited time in the tuning lead to rushed iteration of values for the LQR gains as we found too late that the physical system did not match our simulation so we had to manually perturb the values until we reached any level of stability in the system. Allocating more time throughout to testing the cart would have also been very beneficial in finding the mitigating mechanical issues present within the final system, particularly with the securing of mounts to the board and the high stress that the base was put under during the tests due to its low thickness. If we had done continual testing of the system, we could have found this issues in time to fix or mitigate, leaving a clear area for future growth in project timeline design.

8 Appendix

The team utilized a digital workspace to manage tasks and visualize our workflow. The complete project structure and tables can be found here: Link to our Miro Board.

All control algorithms, sensor interrupt routines, and testing scripts developed for this project are available in our repository: **Link to our GitHub Repository**.

Both videos can be found in **this YouTube playlist**

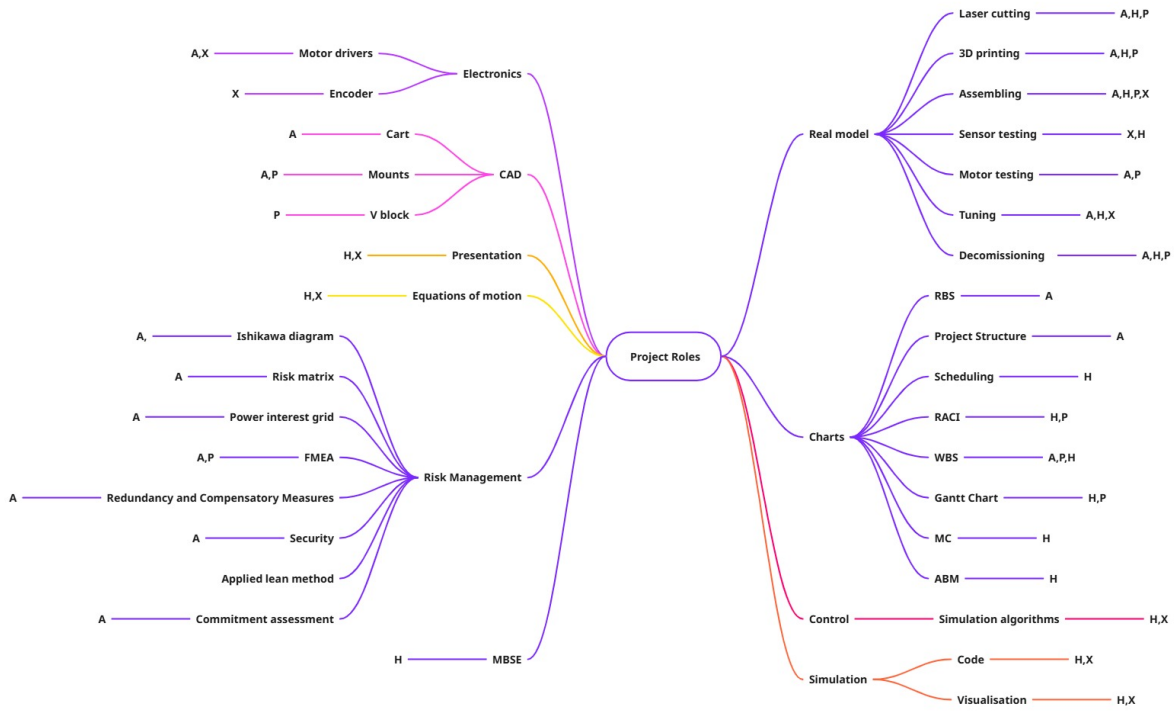


Figure 22: Team roles within the project (A: Aayan, H: Helitha, P: Peter, X: Xavier)

8.1 Ethical Consideration

A hypothetical situation that we have accounted for is that a private company have approached us to acquire our technology to build weaponry for militaries. We will briefly discuss the different ethical viewpoints and what they say we should do. Deontology says we should not sell as one could not universalise a rule of hurting others. Additionally, life is sacred according to deontology and we would be directly aiding in the killing of people. Kant[4] envisioned his rules for everyone to follow and are consequently created with that in mind, leading to in this situation no company should ever have these weapons as no one should be breaking the rules to make or sell them. Contractarianism has the right to life and the right to individual liberty as the two most important ideas; selling this technology to a defence company would directly contradict the right to life; additionally, the weaponry developed could be used to oppress and take away individual's liberty. Following Aristotle's virtue ethics[1], we cannot be a virtuous person and profit from weaponry, this "blood money" could influence future decisions, potentially leading to compounding decisions that a virtuous person should not make, creating more vices than virtues, such as greed and cruelty. Finally, utilitarianism has a more nuanced and complex take, it is not against selling as the technology could be used for defending, potentially saving more lives than it takes away. Furthermore, we could use the money gained from this sale to do good for the world, e.g. a rubbish clearing robot for the oceans, leading to more people being helped overall. However, weaponry normally causes net harm for society and should be avoided in order to maximise benefit for the most people.

References

[1] Rosalind Hursthouse and Glen Pettigrove. "Virtue Ethics". In: *The Stanford Encyclopedia of Philosophy* (2023).

[2] Arieh Iserles. "Runge–Kutta methods". In: *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2008, pp. 33–52.

[3] Riccati. "Animadversiones in aequationes differentiales secundi gradus". In: *Actorum Eruditorum Supplementa* (1724).

[4] Michael Rohlf. "Immanuel Kant". In: *The Stanford Encyclopedia of Philosophy* (2024).

[5] Nichols Ziegler. "Optimum settings for automatic controllers". In: *Trans. ASME* (Nov. 1942), pp. 759–765. DOI: 10.1115/1.4019264.