3006ICT Robotics and Computer Vision

Working as an Engineering Team

Why groups matter, how to divide responsibilities, and how to build one successful system together

A group project is not three individual assignments placed next to each other.

It is one engineering system built by several people.

The team succeeds when the parts work together reliably.

Read this before starting the group project.

Why do engineers work in groups?

Real engineering systems are too broad for one person to do everything well.

Robotics combines perception, software, control, navigation, testing, documentation and integration.

A team allows people to work in parallel while bringing different technical strengths to the same system.

Team contribution	What it means	Benefit
Parallel work	Different technical tasks can be investigated at the same time.	Faster progress
Specialisation	A member can develop deeper expertise in a difficult area.	Better technical quality
Independent checking	Another member can question assumptions and find hidden bugs.	Fewer errors
Integration	Components are connected and tested as one system.	A working complete system
Shared problem solving	Difficult failures are investigated together.	Better recovery
Communication	Design decisions and evidence are recorded and explained.	Maintainability

ENGINEERING LESSON Teamwork is not simply splitting the workload. It should increase the quality, speed, reliability and maintainability of the complete system.

Why this matters for your Webots project

Your project asks the group to develop one autonomous e-puck controller that uses the RGB camera to identify the required target and then navigates safely to the associated observation position. The project is open-ended: your group should choose, justify, integrate and evaluate the technical approach.

Think in this sequence: problem → system design → components → interfaces → integration → testing → evidence → improvement.

DO NOT OPTIMISE THE WRONG THING Accurate vision is not enough if navigation fails. A good planner is not enough if the target is misidentified. Fast motion is not enough if the robot collides. The assessment is a system-level problem.

What does good teamwork look like?

Good teamwork is visible in the way the group works.

A strong team has clear ownership, frequent communication, early integration, shared testing and enough cross-knowledge that no critical part depends on only one person.

Weak pattern	Better engineering practice
“You do vision, I do navigation, they do the report.”	Assign technical ownership, but keep everyone connected to the complete system.
Members work separately for weeks.	Integrate small pieces early and test interfaces repeatedly.

Only the author understands a component.	The owner explains it; another member reviews or tests it.
Problems are hidden until the end.	Record failures early and use them to guide experiments.
Choose an algorithm because it sounds advanced.	Choose methods using evidence, constraints, robustness and integration cost.
Everyone edits everything.	Agree on ownership, interfaces and a consistent integration process.
One person becomes the permanent integrator.	Make integration a team activity and share testing responsibility.

Own a component, but understand the system

Each member should have a primary area of responsibility.

However, primary responsibility does not mean exclusive knowledge.

All group members should be able to explain the complete project, not only their own component.

A GOOD TARGET "I am mainly responsible for this part, but I can explain how it works, why we chose it, what can fail, and how it connects to the rest of the system."

Suggested roles for a 3-person robotics team

The following roles are a practical starting point, not a requirement.

You may organise yourselves differently. What matters is that every critical responsibility is covered and integrated.

Role	Primary responsibility	Questions to answer
A — Perception / Target Identification	Camera processing, target representation, visual matching, confidence and verification.	How do we know this is the requested target? How robust is the decision?
B — Navigation / Planning	Map and pose use, route generation, waypoint selection and route efficiency.	How does the robot decide where to go and reach the correct observation position?
C — Control / Safety / Integration	Wheel control, obstacle avoidance, state transitions, recovery and system integration.	How does the robot move safely? What happens when something fails?

Do not create three isolated mini-projects

The roles create ownership; they should not create silos. Perception should understand what navigation needs. Navigation should understand where visual evidence can be obtained. Control/integration must understand the assumptions of both.

Give every role a secondary responsibility

Primary role	Useful secondary responsibility
Perception	Help with evaluation and failure analysis.
Navigation	Review perception decisions and test route robustness.
Control / Integration	Coordinate system-level testing and architecture documentation.

This creates redundancy: if one member is unavailable, another member can still diagnose and continue the work.

Divide work by interfaces, not just by topics

Many group failures are interface failures. Two components can work perfectly in isolation but fail together because they disagree about inputs, outputs, units, coordinate systems, timing or confidence.

Agree on the system interfaces early

Interface	Agree on this early
Mission input	What target identity is provided and how is it obtained?
Vision → decision	What does perception return: target evidence, candidate station, confidence, or another representation?
Decision → navigation	What information is required before navigation begins?
Navigation → control	Does navigation output a goal, waypoint, heading or trajectory?
Sensors → safety	Which proximity information triggers avoidance or recovery?
Controller states	What states exist, and what causes transitions?
Success condition	How does the controller know it has reached the required observation position?

EXAMPLE Do not wait until the end to discover that one member outputs a pixel coordinate while another expects a station ID. A short interface discussion early can prevent hours of debugging.

Use a simple component contract

For every major component, write: Input → Output → Assumptions → Failure behaviour.

Component	Input	Output	Failure behaviour
Target perception	RGB camera + required target	Target evidence / candidate	Low confidence → continue searching or verify
Route planner	Pose + goal + map	Route / waypoints	No valid route → recover or report
Motion controller	Pose + waypoint	Wheel speeds	Unsafe → stop / avoid / recover
Mission manager	Component results + state	Next state	Unexpected result → safe recovery

A practical team workflow

Do not spend the first half building separate components and the second half trying to connect them. Integration should begin early.

1. **Agree on the problem.** Make sure everyone understands the mission, constraints, success criteria and prohibited shortcuts.
2. **Draw the system.** Create one simple architecture showing perception, decision, navigation, control, safety and stopping.
3. **Assign ownership.** Give each member a primary component and a secondary responsibility.
4. **Define interfaces.** Agree on data passed between components before implementation becomes complicated.

5. **Build small prototypes.** Prove each component with simple, measurable tests.
6. **Integrate early.** Connect real components as soon as possible, even if they are initially imperfect.
7. **Test as a team.** Run controlled experiments, record failures, compare alternatives and improve.
8. **Prepare evidence.** Keep useful screenshots, measurements, failure cases and design decisions as you go.

A useful team meeting

Question	What to report
What did I complete?	Concrete outcomes, not a long activity list.
What evidence do I have?	A test result, screenshot, metric or reproducible observation.
What failed?	What happened, under what conditions and what was learned.
What do I need from the team?	A decision, interface change, test, review or debugging help.
What will I do next?	A small task with a clear expected outcome.

KEEP MEETINGS TECHNICAL AND SHORT The goal is not to prove everyone has been busy. The goal is to keep the engineering system moving forward.

Integration: where projects are often won or lost

Integration makes independently developed components behave as one coherent system. In robotics, perception, planning, control and sensing interact continuously.

Integrate in layers

1. **Motion first.** Make sure the robot can move, turn, stop and recover safely.
2. **Perception next.** Verify the camera pipeline produces useful evidence under realistic views.
3. **Navigation next.** Test movement to a known goal before asking the system to discover the goal.
4. **Safety next.** Make obstacle avoidance work without destroying useful progress.
5. **Mission logic.** Connect search → identify → navigate → arrive → stop, then repeatedly test the full loop.

Debug one boundary at a time

Observed failure	First question
Wrong station	Was the target identified incorrectly, or was navigation wrong?
Correct station but wrong final position	Is the goal coordinate/frame correct? Is final control accurate enough?
Collision	Was detection too late, or was the reaction too slow?
Oscillation	Is control too aggressive, or is perception noisy?
Robot gets stuck	Is the route blocked, is recovery missing, or is the state machine unable to progress?
Full system unstable	Which component first produced an incorrect output? Test it separately.

Responsibility does not mean “my code, my marks”

Professional engineering separates individual responsibility from system responsibility.

You should know who owns a task, but the team owns the final system.

Situation	Professional response
Your component fails	Tell the team early, show evidence and propose a debugging experiment.
Another member's component fails	Help isolate the cause rather than treating it as someone else's problem.
Two approaches compete	Define a test and compare them using the same criteria.
You disagree	Record assumptions, run an experiment and decide using evidence.
One member is overloaded	Rebalance work before the deadline becomes critical.
A shortcut looks attractive	Check it against the assessment rules and engineering objective first.

Use evidence to resolve disagreements

Instead of “method A is better”, define what better means: target accuracy, false detections, completion time, collision rate, robustness across starts, or integration complexity. Then test both under comparable conditions.

A STRONG ENGINEERING DISCUSSION “Our hypothesis is X. We tested it under Y conditions. We observed Z. Therefore we will use A rather than B.”

8. Make the group efficient

Use a shared task board

- **To do —** not started.
- **Doing —** currently being implemented or tested.
- **Blocked —** cannot progress without a decision, input or fix.
- **Done —** tested and supported by evidence.

Make tasks small enough to have a clear finish. “Improve navigation” is too broad. “Test the waypoint controller at three distances and record final-position error” is much better.

Keep a decision and failure log

Date	Decision / failure	Evidence	Impact
	Example: choose approach X	Test result / robustness	Changes perception interface
	Example: adjust safety threshold	Collision test	Changes control behaviour
	Example: change search order	Completion-time comparison	Changes mission manager

Protect the final integration period

Do not leave integration, robustness testing, report evidence and presentation preparation until the final days. The final period should improve reliability, not reveal that the components cannot work together.

GOOD MILESTONE Get at least one end-to-end run working well before the project is “nearly finished”. Then spend the remaining time on robustness, not basic integration.

Make a team agreement on Day 1

Before serious implementation begins, agree on the following and keep the answers somewhere everyone can access.

Question	Our team agreement
Primary owner of each technical component?	
Secondary/backup person for each component?	
Shared system architecture?	
Interfaces between components?	
How will changes be tested before integration?	
How will decisions and failures be recorded?	
When will full end-to-end tests start?	
How will disagreements be handled?	
How will every member learn the complete system?	

What a strong engineering team should be able to demonstrate

Area	Your team should be able to explain…
Problem	What the robot must do and what information is known or unknown.
Architecture	How perception, decision, navigation, control, safety and stopping interact.
Design choices	Why the selected methods are appropriate.
Integration	How components exchange information and handle uncertainty.
Evaluation	What was tested, under what conditions and what results show.
Failure analysis	What failed, why, and what was changed.
Team contribution	Who led which work and how members collaborated and reviewed one another.

Complete system	How the whole controller works—not only one member's code.

The professional mindset

Do not measure success by how much code each person writes. Measure it by whether your team builds a reliable, understandable, testable and defensible system within the available time.

REMEMBER Divide the work. Share the knowledge. Integrate early. Test with evidence. Own the complete system.

Quick checklist before you start

☐ We all understand the project goal and success criteria.
☐ We have agreed on primary and secondary responsibilities.
☐ We have one shared system architecture.
☐ We know what each component takes as input and produces as output.
☐ We have a plan for early integration.
☐ We have a simple way to record tests, failures and decisions.
☐ We will test the complete controller, not only isolated components.
☐ We will use evidence to compare technical alternatives.
☐ Every member will understand the complete system.
☐ We have reserved time for robustness testing, the report and the presentation.

Next: Part 3 — Group Project Guide