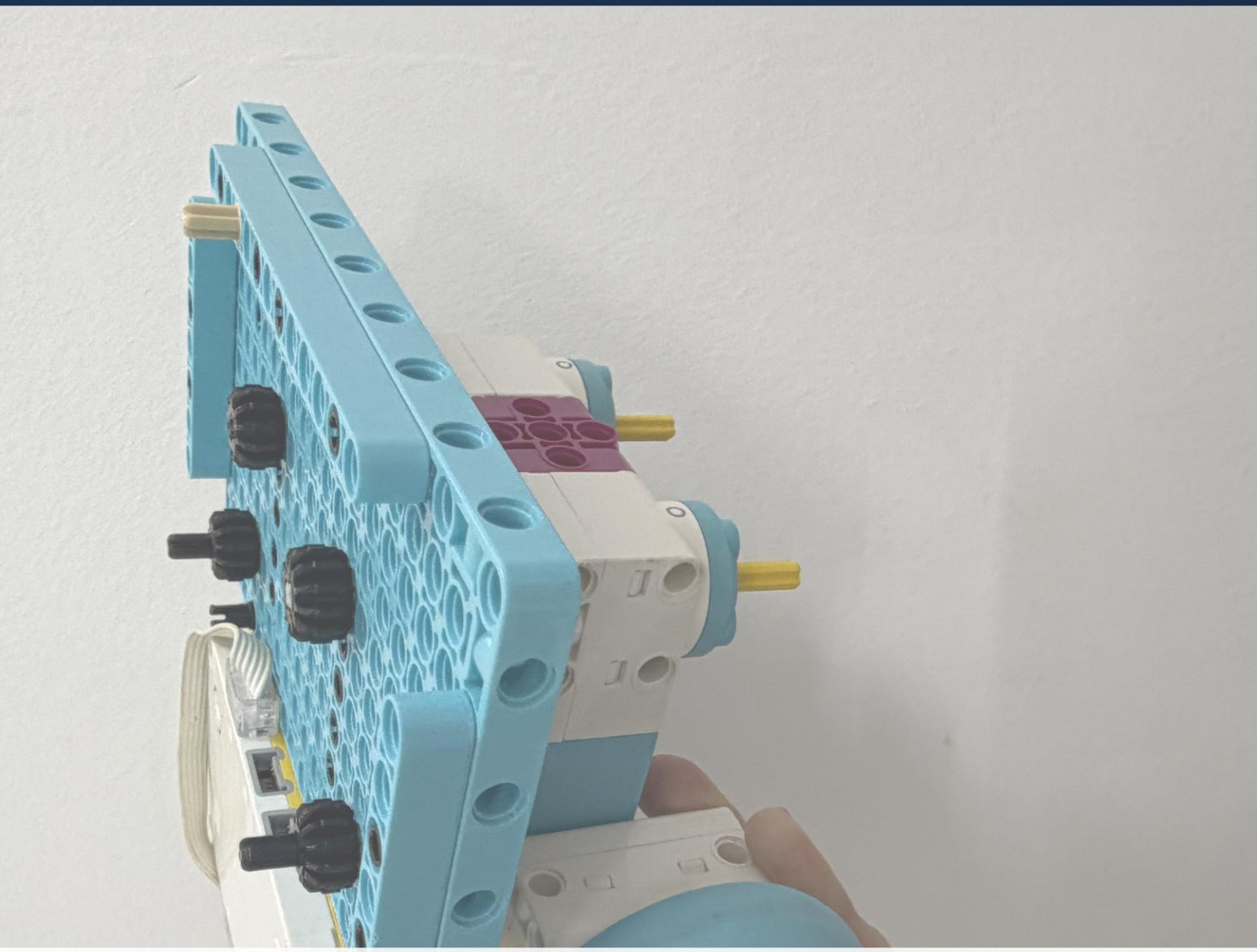


# Robot Design
## & Mission Strategy

Technical book for judges, table display, and team presentation

Run A | Run B | Run C | Run D | Run E | Run G

JUDGE QUICK SUMMARY

# What we engineered this season

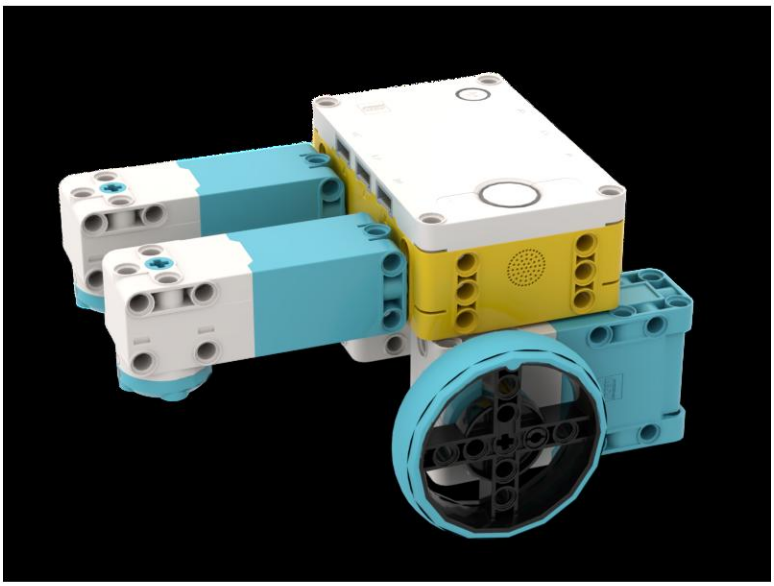
Our design goal was to make a robot the whole team could align, repair, explain, and trust under competition pressure.

- Stable base: lower, cleaner, and easier to inspect than earlier concepts.
- Mission strategy: grouped nearby missions into short runs to reduce risk.
- Attachment system: active + active + passive mechanisms for strong multi-mission tools.
- Programming: Pybricks with gyro-enabled DriveBase, encoder distance, and tuned PID.
- Competition readiness: diagnostics for gyro drift, motor sync, turn accuracy, and battery voltage.

ROBOT EVOLUTION

# From concept to reliable competition base

The robot moved from broad concept work toward a compact base, cleaner motor layout, and attachment access that teammates could understand quickly.

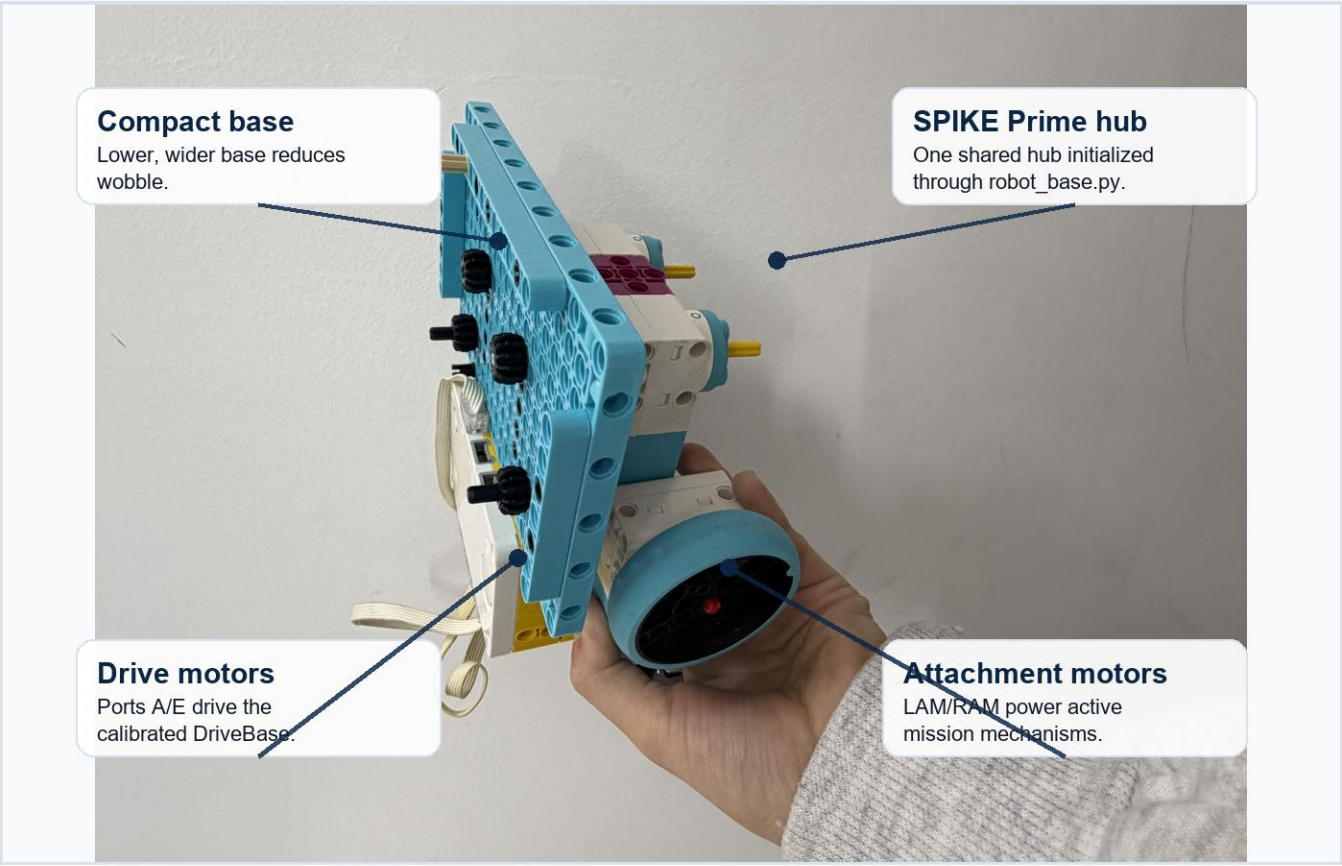


- Earlier ideas helped us decide what was too large, fragile, or hard to reset.
- The final base favors repeatable alignment and easy attachment access.

CURRENT ROBOT

# Anatomy of the final base

The robot anatomy overlay connects the physical build to the code architecture and attachment system.




- Drive motors are initialized once in the shared base file.
- Attachment motors are separated into left and right mechanisms.
- The base is designed around repeatable launch alignment and fast swaps.

MISSION STRATEGY

# Grouped runs reduce risk

Colored paths show how each mission group moves from launch to targets.

## Mission Run Map

Each colored path shows one planned run group on the Unearthed field.




**Run A: M5 M6 M7 M8**
Start: right launch
Push/lift chain with fast return

**Run B: M9 M10**
Start: right launch
Tip and pan control

**Run C: M1 M2**
Start: left-side route
Pickup geometry

**Run D: M3 M4**
Start: left launch
Precision lift and drop

**Run E: M11 M13**
Start: right launch
Long blue-side transition

**Run G: M15**
Start: left launch
Slow controlled final drop

MISSION MODELS

# What each run is built to solve

Mission model photos are grouped by run so judges can connect strategy to the table.

## Mission model grouping

| | | |
|---|---|---|
| Run A |  M5  M6  M7  M8 | **Start: right launch** Push/lift chain with fast return |
| Run B |  M9  M10 | **Start: right launch** Tip and pan control |
| Run C |  M1  M2 | **Start: left-side route** Pickup geometry |
| Run D |  M3  M4 | **Start: left launch** Precision lift and drop |
| Run E |  M11  M13 | **Start: right launch** Long blue-side transition |
| Run G |  M15 | **Start: left launch** Slow controlled final drop |

RUN PLAN

# Current competition map

The plan favors shorter trips, repeatable alignment, and attachments that solve multiple models before returning home.

| Run | Missions | Launch | Design focus |
|---|---|---|---|
| Run A | M5 M6 M7 M8 | right launch | Push/lift chain with fast return |
| Run B | M9 M10 | right launch | Tip and pan control |
| Run C | M1 M2 | left-side route | Pickup geometry |
| Run D | M3 M4 | left launch | Precision lift and drop |
| Run E | M11 M13 | right launch | Long blue-side transition |
| Run G | M15 | left launch | Slow controlled final drop |

ATTACHMENT ITERATION

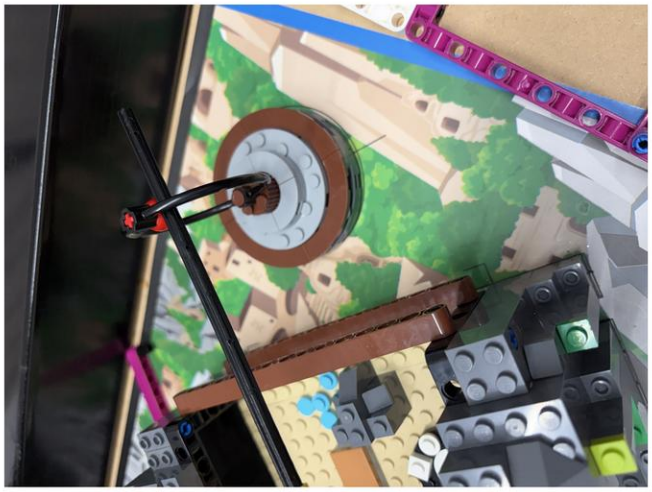
# Run A: versions 1-2

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run A Attachment Iteration

Vertical evidence page 1: versions 1-2

### 1. Prototype



**Evidence photo v1**

### 2. Improve



**Evidence photo v2**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

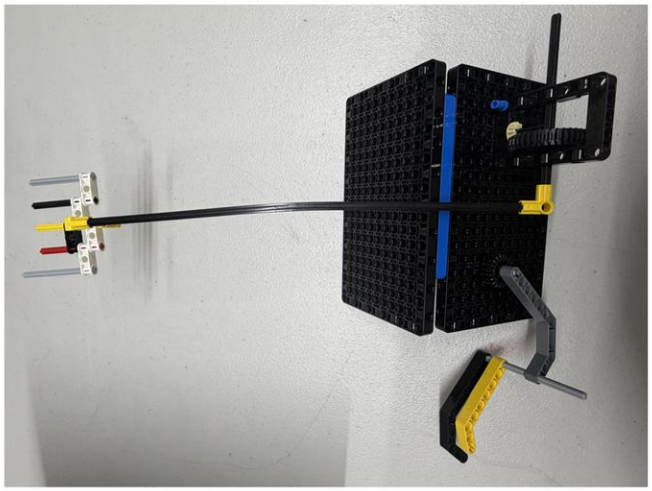
# Run A: versions 3-4

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run A Attachment Iteration

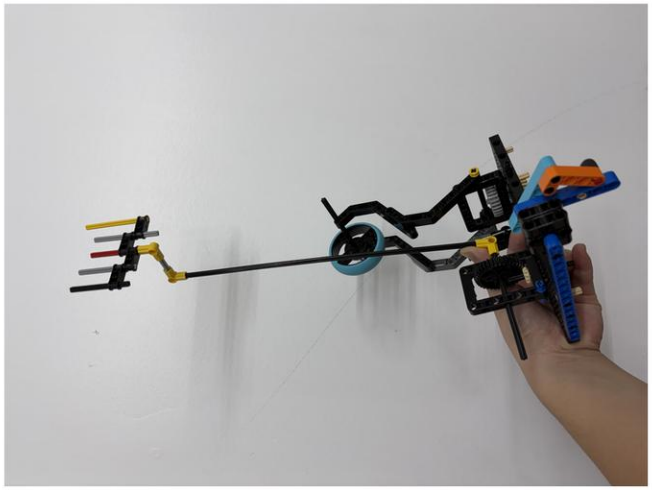
Vertical evidence page 2: versions 3-4

### 3. Refine



**Evidence photo v3**

### 4. Final



**Evidence photo v4**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

# Run B: versions 1-2

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run B Attachment Iteration

Vertical evidence page 1: versions 1-2

### 1. Prototype



**Evidence photo v1**

### 2. Improve



**Evidence photo v2**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

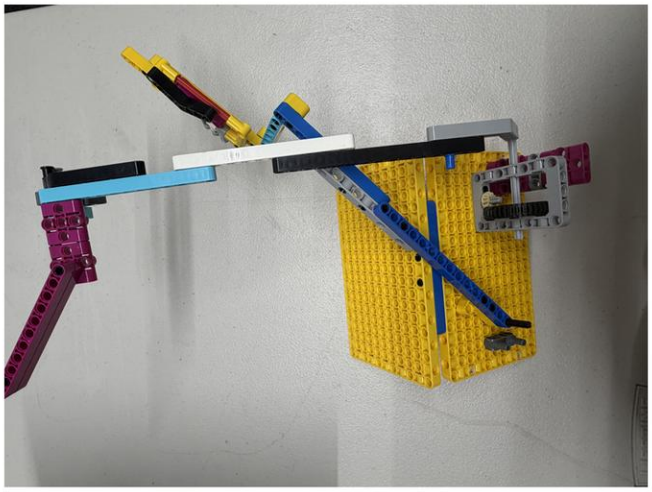
# Run B: versions 3-4

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run B Attachment Iteration

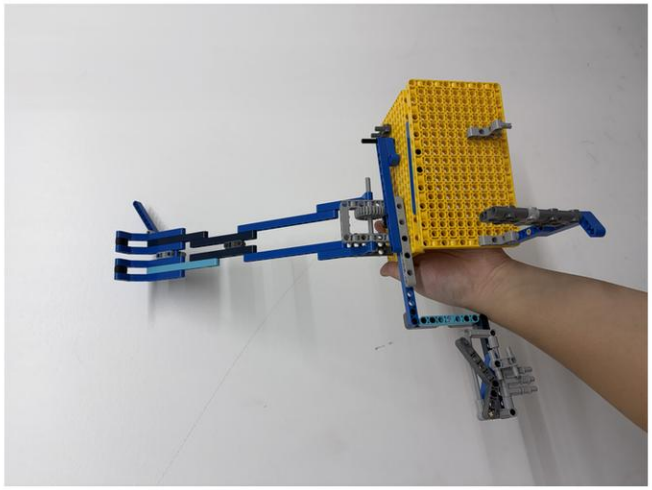
Vertical evidence page 2: versions 3-4

### 3. Refine



**Evidence photo v3**

### 4. Final



**Evidence photo v4**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

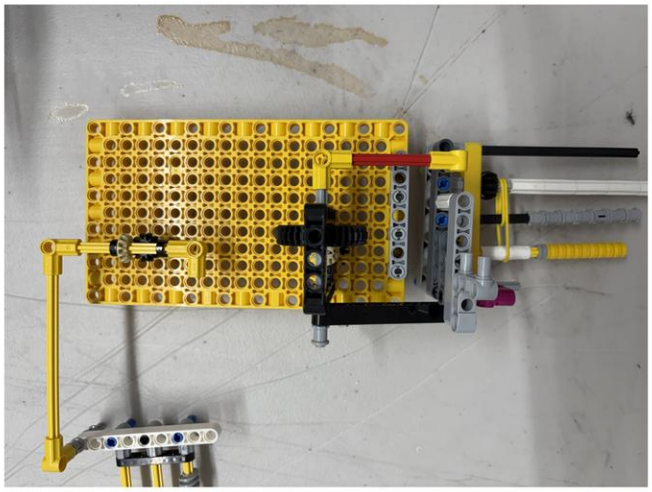
# Run C: versions 1-2

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run C Attachment Iteration

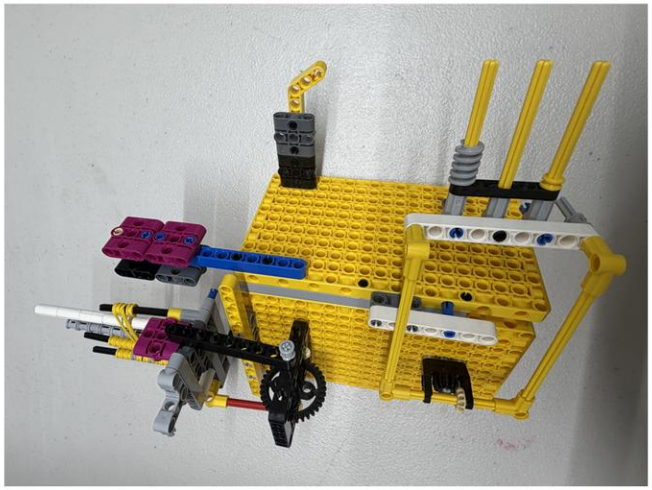
Vertical evidence page 1: versions 1-2

### 1. Prototype



**Evidence photo v1**

### 2. Improve



**Evidence photo v2**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

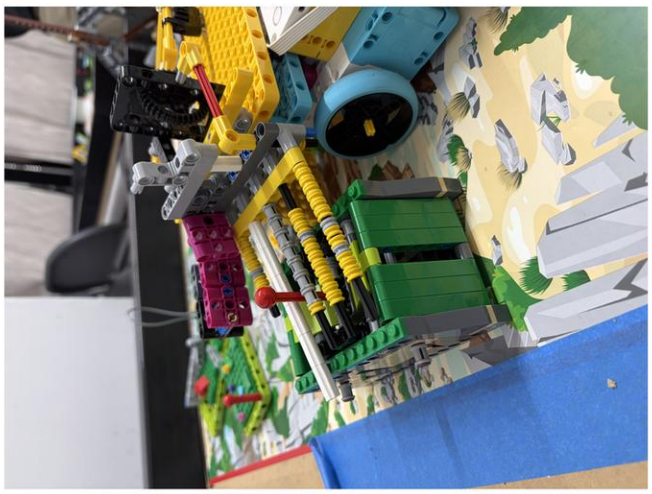
# Run C: versions 3-4

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run C Attachment Iteration

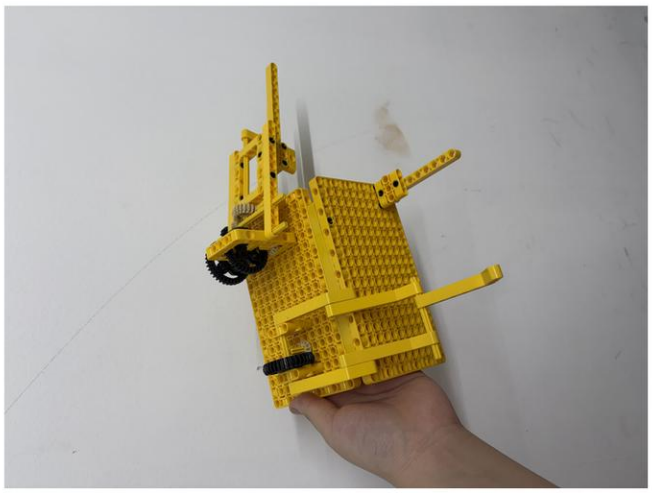
Vertical evidence page 2: versions 3-4

### 3. Refine



**Evidence photo v3**

### 4. Final



**Evidence photo v4**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

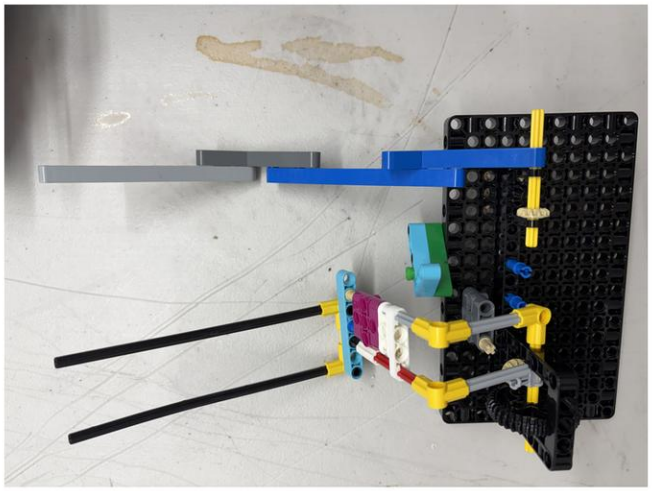
# Run D: versions 1-2

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run D Attachment Iteration

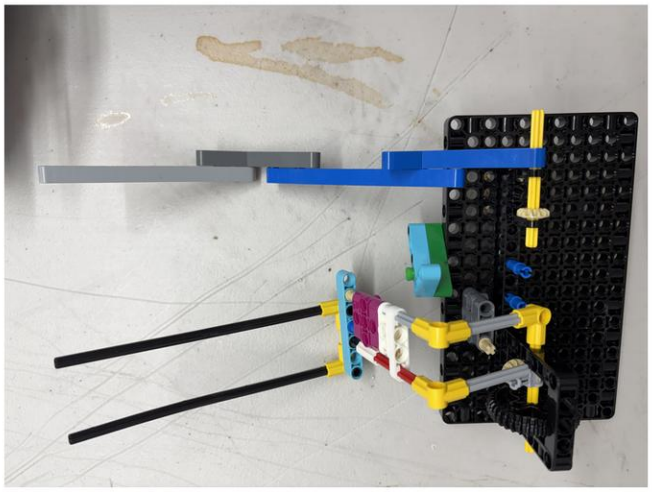
Vertical evidence page 1: versions 1-2

### 1. Prototype



**Evidence photo v1**

### 2. Improve



**Evidence photo v2**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

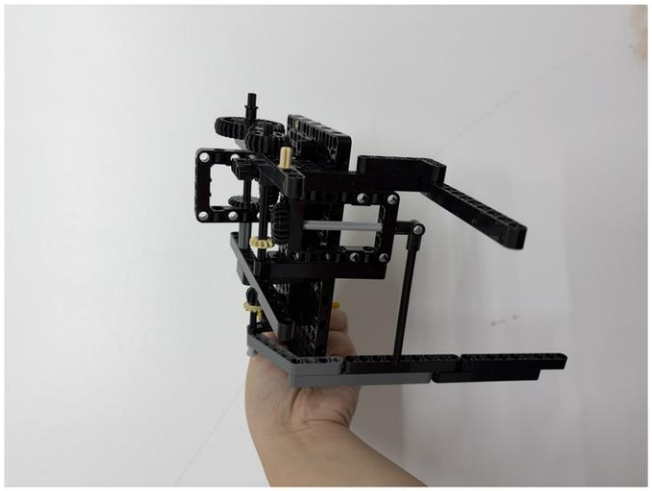
# Run D: versions 3-4

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run D Attachment Iteration

Vertical evidence page 2: versions 3-4

### 3. Refine



**Evidence photo v3**

### 4. Final



**Evidence photo v4**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

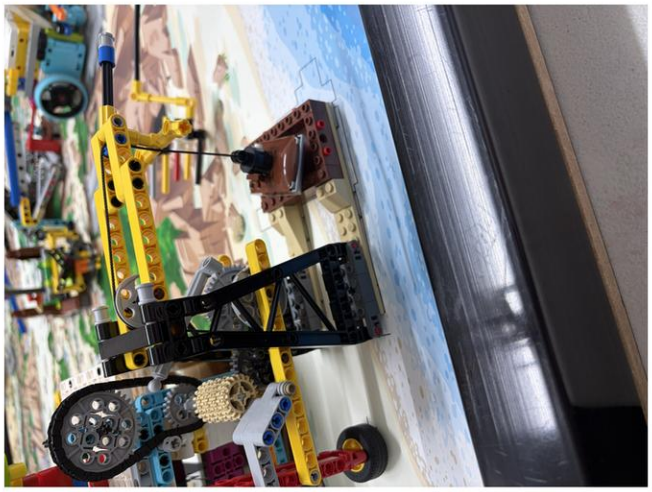
# Run E: versions 1-2

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run E Attachment Iteration

Vertical evidence page 1: versions 1-2

### 1. Prototype



**Evidence photo v1**

### 2. Improve



**Evidence photo v2**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT ITERATION

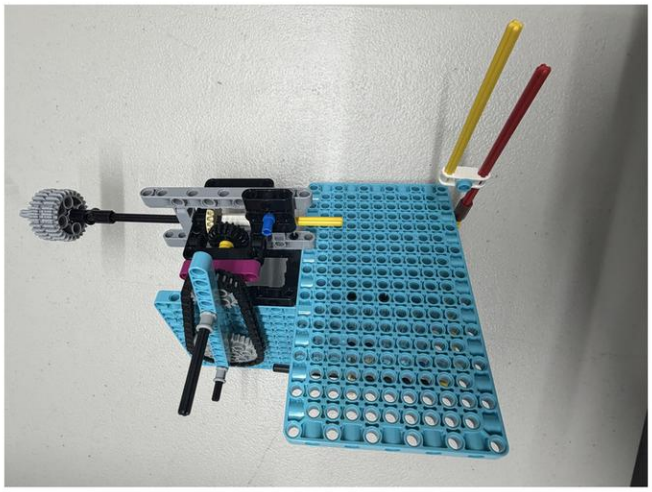
# Run E: versions 3-4

**Vertical run iteration page: fewer photos per sheet, much larger evidence images.**

## Run E Attachment Iteration

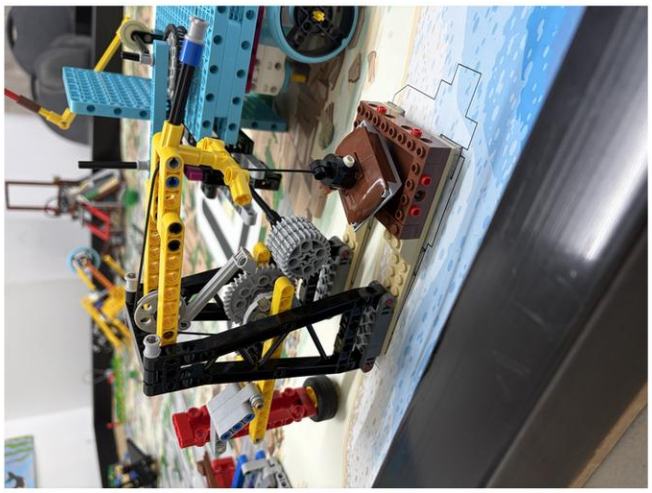
Vertical evidence page 2: versions 3-4

### 3. Refine



**Evidence photo v3**

### 4. Final



**Evidence photo v4**

**Design rule: change one main variable at a time so the team can tell what improved reliability.**

ATTACHMENT SYSTEM

# Two active actions plus one passive action

Our strongest attachments combined powered control with simple passive motion. The active mechanisms gave precision, while passive geometry made setup and release faster.

| Design rule | Why it helped |
| --- | --- |
| Active attachment | Precise movement for lifts, drops, and controlled pushes. |
| Active attachment | Second powered action lets one run solve more than one model. |
| Passive attachment | No motor timing needed, so the action is faster and simpler. |
| Color coding | Teammates can grab the right run tool instantly. |

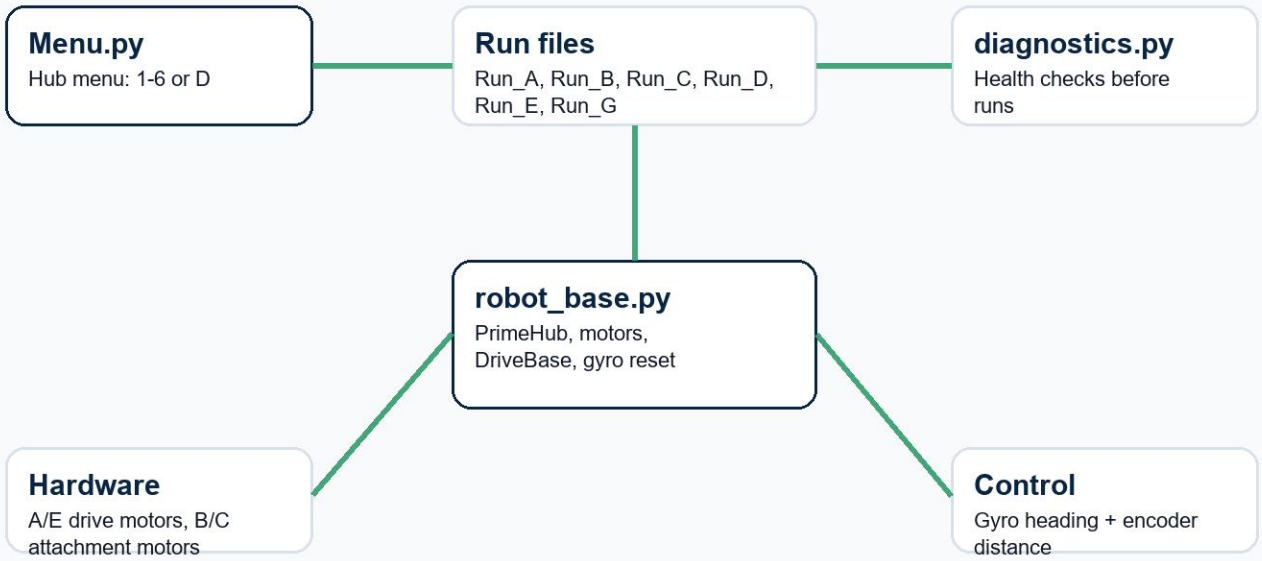
- Run A: all black pieces.
- Run B: blue, grey, and yellow pieces.
- Run C: black and yellow pieces.
- Run D: all black pieces.
- Run E: all blue pieces.

PROGRAMMING

# Pybricks gives us direct control

## Programming architecture

Menu selection imports one run file, while robot_base.py keeps hardware setup consistent.




### robot_base.py excerpt

```
def make_robot(axle_track=AXLE_TRACK_MM):
    """
    Creates and returns a DriveBase configured for this robot.

    Call once at the very top of each run file (before robot.settings):
        robot = make_robot()            # 122 mm — most runs
        robot = make_robot(101.6)       # Run_A_updated
        robot = make_robot(152)         # Run_D
```

The shared factory enables gyro, resets heading safely, and returns a DriveBase ready for each run.

CODE DECISIONS

# Real run files, real calibration choices

### Run_A_updated.py

```
    turn_acceleration=450
)

robot.heading_control.pid(21242, 0, 5310, 4, 8)

# -----------------------------------------------------------

if hub.imu.ready():
    print(robot.heading_control.pid())
```

Run A uses axle_track=101.6 and tuned heading PID because it solves a large mission chain.

### Run_D.py

```
robot = make_robot(axle_track=152)
robot.settings(straight_speed=450, straight_acceleration=200, turn_rate=300, turn_accelera
# robot.heading_control.pid(kp=1000,ki=100,kd=200)

#Run D completes mission 3 and 4.
async def main():
  if hub.imu.ready():
      await robot.straight(-900, then=Stop.HOLD)
```

Run D uses axle_track=152 and slower settings for precision lift/drop work.

MENU AND MATCH FLOW

# One simple selector for competition

**Menu.py**

```
selected = hub_menu("1","2","3","4","5","6","D")
hub.system.set_stop_button(Button.CENTER)

hub.light.on(Color.RED)

if selected == "1":
    import Run_A_updated
elif selected == "2":
    import Run_B
```

The center button is the stop button, the hub light gives status, and D opens diagnostics.

- The menu prevents hunting through files at the table.
- Each selected number imports exactly one run file.
- Diagnostics are available from the same interface as mission runs.

SENSORS AND PID

# Gyro heading plus encoder distance

The robot uses the gyro for heading and motor encoders for distance. Pybricks DriveBase lets us tune straight movement, turns, acceleration, and heading correction.

| Control idea | How the robot uses it |
| --- | --- |
| Gyro | Keeps heading stable during turns and straight driving. |
| Encoders | Measure wheel rotation so distance moves can be repeated. |
| PID | Adjusts motor power based on heading error. |
| Acceleration | Smooth starts and stops reduce slip. |

**94.2% consistency**

Pybricks + PID correction from website testing narrative

DIAGNOSTICS

# Robot health before a run

## Competition readiness dashboard

**Gyro Drift**
Max z-axis drift < 0.5 deg/s and heading stays near zero.

**Motor Sync**
Compare left/right encoder angles after a 2-second run.

**Turn Accuracy**
Four 90-degree turns should finish near 360 degrees.

**Battery Voltage**
Minimum 7400 mV to avoid weak, inconsistent motion.

**Status Lights**
Yellow running, green pass, red fail for quick table checks.

**Why this matters: diagnostics separate code tuning from hardware issues like loose cables, weak motors, low battery, or gyro drift.**

- Diagnostics separate code tuning from hardware problems.
- Battery, cables, weak motors, and gyro drift can be caught before match pressure starts.

TESTING LOOP

# Rough draft, tune, refine

| Step | What we do | Evidence |
|---|---|---|
| 1. Rough draft | Prove the mission path can work. | Early attachment photos and first run code. |
| 2. Tune + debug | Fix alignment, drift, timing, and attachment contact. | PID values, heading prints, repeated runs. |
| 3. Refine | Make the run competition consistent. | Final photos, color-coded tools, diagnostics. |

- Captain: Isabella.
- Lead coder: Anthony.
- Lead builder: Kyle.
- Red side team: Brian, Owen, Sam.
- Blue side team: Cailey, Jonathan, Julisa, Jasper.

JUDGE REFERENCE

# Questions this book helps us answer

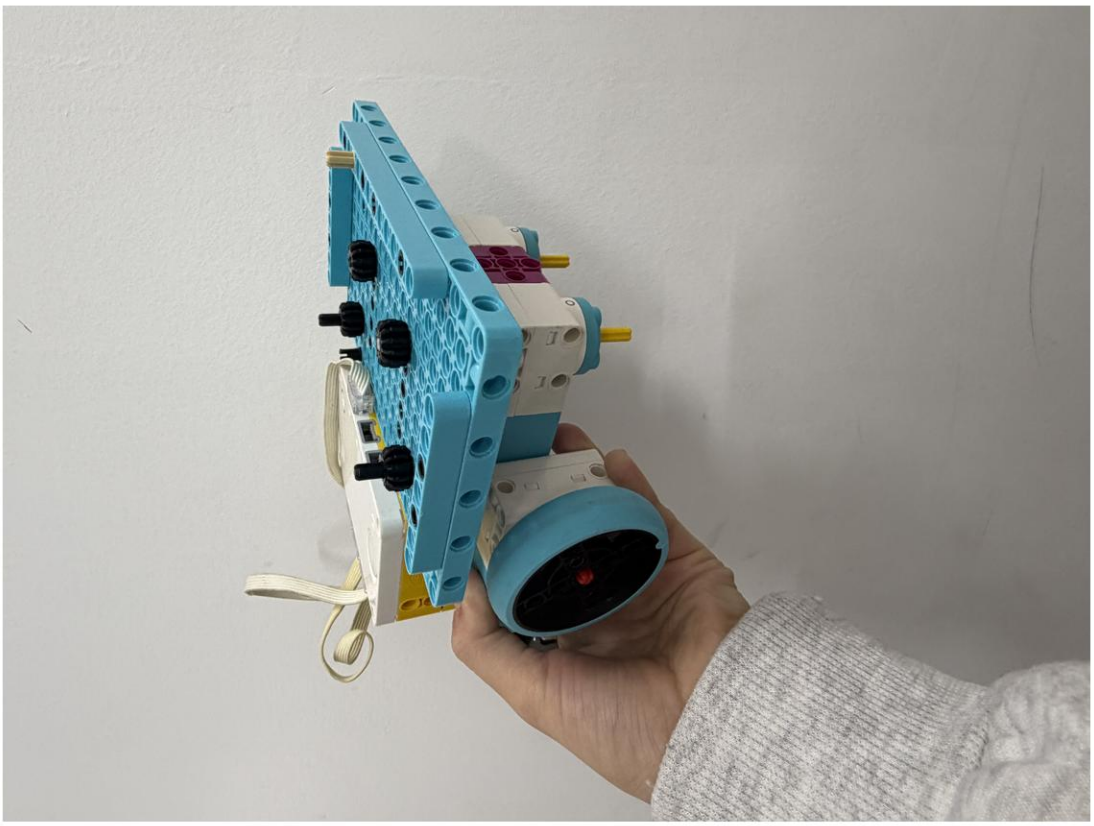
- How did your robot improve from earlier versions?
- Why did you choose these mission groups?
- Which attachments changed the most, and what did each change fix?
- How does your program keep the robot consistent?
- How do diagnostics help you know whether a failure is hardware or software?

| Spec | Current answer |
| --- | --- |
| Platform | LEGO SPIKE Prime with Pybricks |
| Drive | DriveBase using motors on ports A and E |
| Attachment motors | LAM on port B, RAM on port C |
| Wheel diameter | 56 mm |
| Default axle track | 122 mm, with per-run overrides |
| Readiness checks | Gyro drift, motor sync, turn accuracy, battery voltage |

CLOSING

# Our design story in one sentence

**We built a robot system where the physical base, attachments, strategy, code, and diagnostics all support the same goal: repeatable performance under competition pressure.**



APPENDIX

# Calibration notes from robot_base.py

### Calibration guide

```
# STEP 1 — WHEEL DIAMETER (do this first):
#   robot.straight(1000)
#   Measure actual distance traveled.
#   Adjust: new = old * (1000 / actual_distance)
#
# STEP 2 — AXLE TRACK (after wheel diameter is correct):
#   robot.turn(360)
#   Robot should face same direction.
#     Turned too little -> increase AXLE_TRACK_MM
```

These notes document how wheel diameter, axle track, and heading PID are tuned.