ARGoS-based Air Resistance — User Manual

Roi Sela

October 2025

Contents

1 Introduction

This project extends ARGoS with a simple but effective wind model and aerodynamic "blocking" (wake) between robots. It lets you demonstrate how a robot aligned behind another can make upwind progress even when an unshielded robot cannot. The examples mirror the style of the ARGoS gallery for fast onboarding.

What you get

- **Global wind**: a 2D wind vector (angle in degrees, magnitude in **cm/s**) configured once per experiment.
- **Post-step impulses**: wind and drive are applied as impulses after collisions, so the dynamics engine stays authoritative.
- **Blocking / wake model**: upwind robots reduce the effective wind on downwind neighbors with a smooth lateral (Gaussian-like) and downwind falloff.
- **On-screen wind arrow**: a red arrow shows direction and scales with magnitude in the Qt-OpenGL visualization.
- **Units**: controller `velocity` and wind `magnitude` are both in **cm/s**.

2 Installation & Build

Prerequisites

ARGoS (3.0.0-beta or newer) and standard build tools (`cmake`, `make`, a C++ compiler).

Get the sources

Clone the project from GitHub (GitHub link below):

$$GitHub : ARGoS - basedAirResistance$$

The repository contains the library code under `src/` and ready-to-run example configs under `examples/`.

Build & install the libraries

```
mkdir -p build && cd build
cmake -DCMAKE_BUILD_TYPE=Release ../src
make
sudo make install
```

If ARGoS cannot find the plugins, reconfigure with an explicit prefix:

```
cmake -DCMAKE_BUILD_TYPE=Release ../src -DCMAKE_INSTALL_PREFIX=/usr
```

Build the examples

```
cd examples
mkdir -p build && cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
make
```

3 Running

From the `examples` directory:

```
argos3 -c <config-file>.txt
```

Use the Qt-OpenGL visualization to see the wind arrow and RAB rays.

4 Configuration

Global wind is set once per experiment:

```
<configuration>
  <air_resistance angle_deg="0" magnitude="15.0"/>
</configuration>
```

Each robot controller declares a target speed and uses RAB for wake logic:

```
<controllers>
  <air_resistance_controller id="airbot"
      library="build/lib/controllers/air_resistance/libair_resistance">
    <actuators>
      <differential_steering implementation="default"/>
```

```
      <range_and_bearing implementation="default"/>
    </actuators>
    <sensors>
      <positioning implementation="default"/>
      <range_and_bearing implementation="medium" medium="rab" show_rays="true"/>
    </sensors>
    <params velocity="15.0"/>
  </air_resistance_controller>
</controllers>
```

5 Examples

Each example follows the ARGoS gallery style: *what it shows*, *how to run*, *what to observe*, and a figure.

5.1 Blocked vs Unblocked (3 e-puck2)

Shows: A leader acts as an upwind blocker. A follower directly behind is shielded and moves upwind; a sideways follower is not fully shielded and struggles.
Run

```
argos3 -c airResistance_blocked_vs_unblocked.txt
```

Observe With wind magnitude equal to controller velocity, the unshielded robot is still while the shielded one advances.

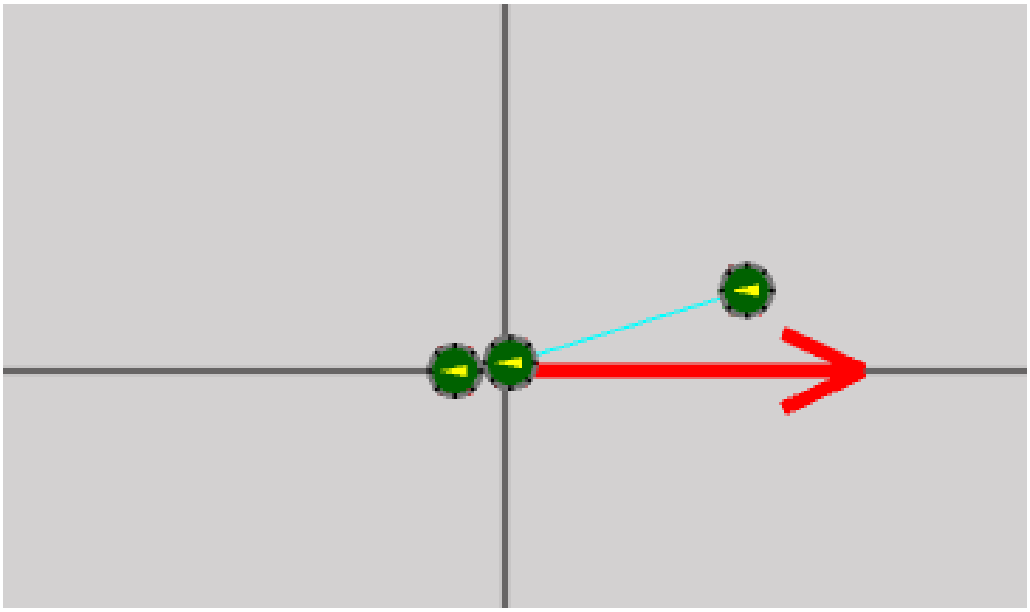

Figure 1: Blocked vs Unblocked. The follower in the wake progresses upwind; the offset follower does not.

5.2 Two Robots — No Blocking (parallel columns)

Shows: Two e-puck2 separated laterally so their wakes do not overlap; both feel the full wind.
Run

```
argos3 -c airResistance_two_no_block.txt
```

Observe Same behavior for both; if wind = drive speed, both are still, if wind > drive speed, both robots will go down-wind etc..

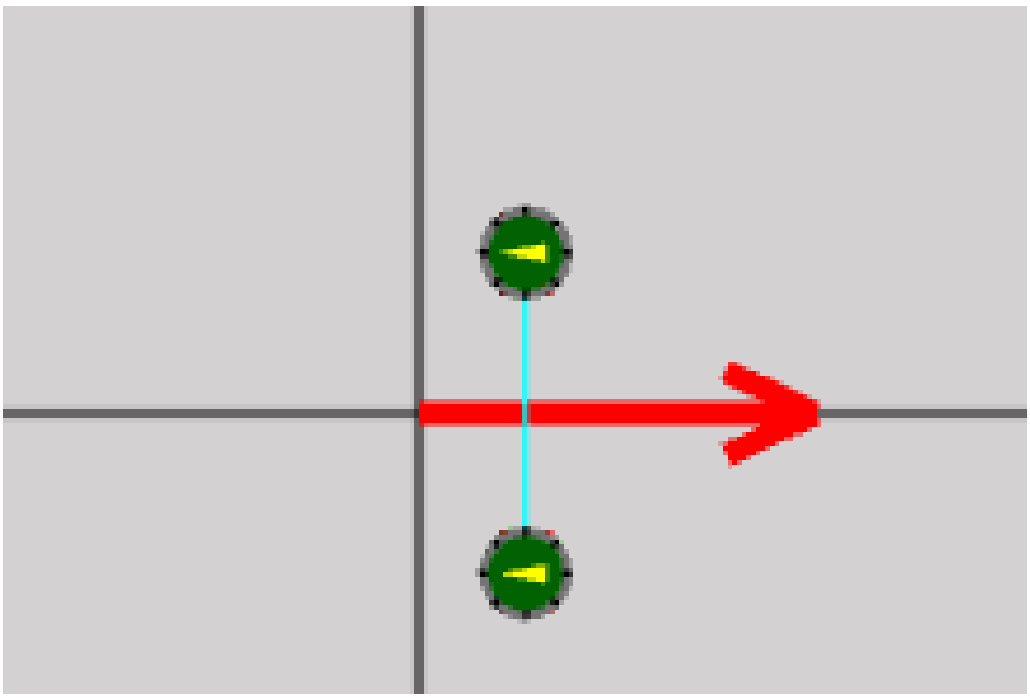

Figure 2: Two robots in parallel columns: no wake overlap, both exposed to wind, because wind = drive speed both are still.

5.3 Two Robots with a Blocker (1 leader, 2 followers)

Shows: A leader upwind; two followers slightly spread in Y but still within the wake core. Both are partially shielded.
Run

```
argos3 -c airResistance_two_with_block.txt
```

Observe Both followers advance upwind more than an unshielded unit (see previous example); small lateral offsets still benefit if inside the wake.

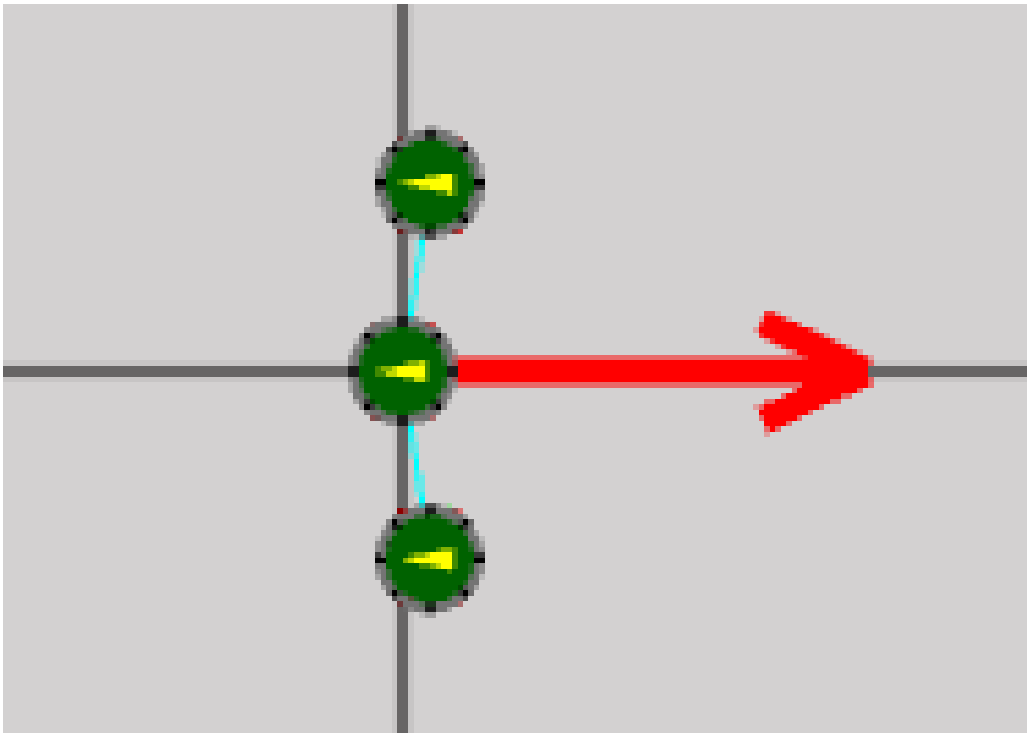

Figure 3: Two followers behind a single blocker; both benefit from partial shielding.

5.4 Three in a Row (wake chaining)

Shows: Three e-puck2 aligned with the wind: leader → follower → follower. The first follower gains the most; the third, farther downwind, gains less due to wake fade.
Run

```
argos3 -c airResistance_three_in_row.txt
```

Observe The first follower progresses fastest; the second follower still benefits but less.

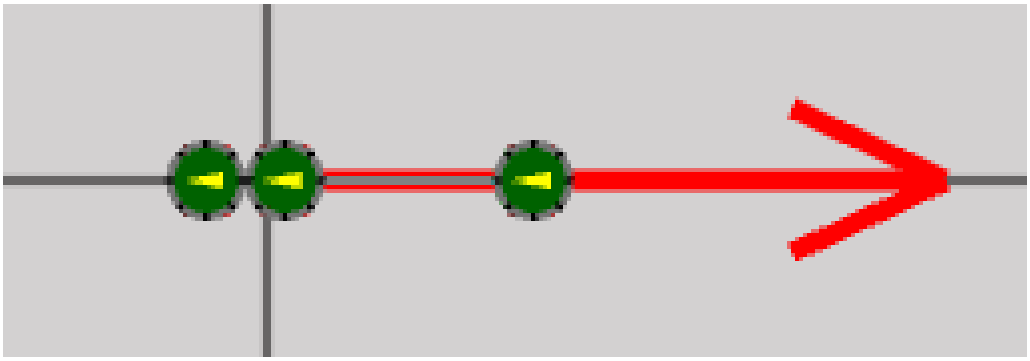

Figure 4: Wake chaining with three robots aligned downwind, Because wind = in this example, the leader does not move and the follower is getting closer to it thereby getting more coverage and going faster, distancing itself from its own follower.

5.5 Foot-bot Wake Demo (multi-body)

Shows: The same wake mechanism with foot-bots (multi-body model).
Run

```
argos3 -c airResistance_foot_bot_blocking.txt
```

Observe The downwind foot-bot gains upwind progress compared to an unshielded peer.

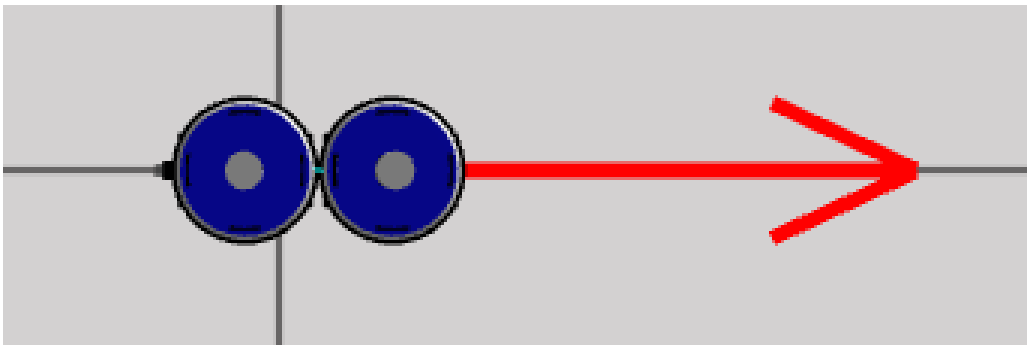

Figure 5: Wake effect with foot-bots (multi-body dynamics).

5.6 Crosswind "Crab" Control (foot-bot)

Shows: Two foot-bots under a crosswind from $+Y$. The *base* controller drifts while going west; the *wind-aware* controller yaws slightly into the wind and tracks west more cleanly.
Run

```
argos3 -c wind_crab_footbot.txt
```

Observe Compare the ground tracks: the wind-aware robot maintains the intended heading with less lateral error.

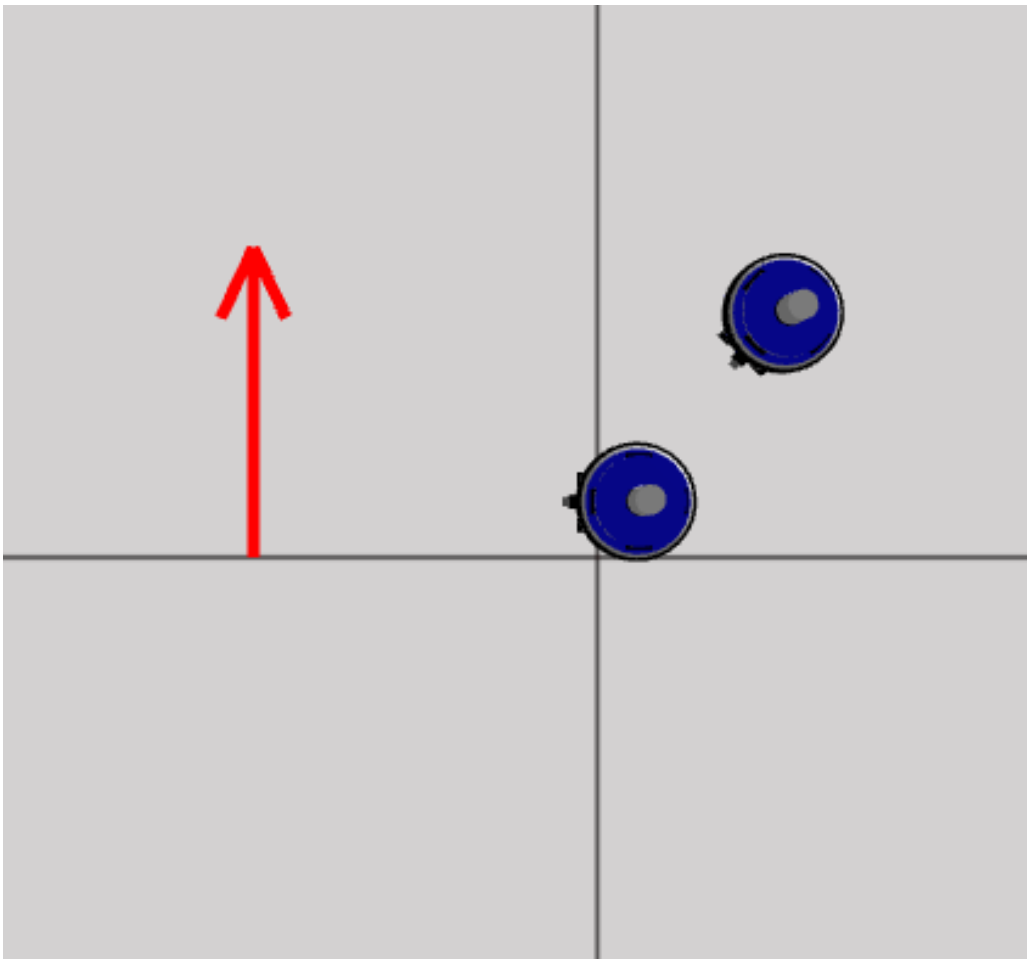

Figure 6: Crosswind compensation: wind-aware controller crabs into the wind.

This example also goes to show how one can inherit from the AirResistance base class and modify/extend its methods.

6 How it Works (Brief)

- **Reading config:** wind angle/magnitude are parsed once from `<configuration><air_resistance ... />`.
- **Broadcasting radius:** each robot encodes an effective body radius in a RAB byte; neighbors use it to evaluate wake overlap.
- **Blocking model:** upwind neighbors reduce effective wind via (i) lateral Gaussian coverage across columns and (ii) downwind fade; an upwind gate prevents side-by-side false positives.
- **Impulse application:** per-tick, wind and drive impulses are accumulated and applied post physics step at the body COM.

7 Tuning Tips

- Set `magnitude` $\approx$ `velocity` to make differences obvious (unshielded $\approx$ stationary; shielded moves upwind).
- Increase `magnitude` and align robots within $\sim$1–2 body radii laterally to emphasize wake effects.
- Place followers at increasing downwind distances to visualize wake fade.

8 Troubleshooting

- **No wind arrow:** ensure Qt-OpenGL visualization and that the user-functions library is loaded.
- **Robot not moving:** if wind $\approx$ drive and the robot is unshielded, this can be expected; adjust either value.
- **Plugins not found:** rebuild with `-DCMAKE_INSTALL_PREFIX=/usr` or update your ARGoS plugin path.

Appendix A: Example Summary Table

Example	File	Robots	Wind
Blocked vs Unblocked	`airResistance_blocked_vs_unblocked.txt`	3 × e-puck2	$+X$, 15 cm/s
Two robots — no blocking	`airResistance_two_no_block.txt`	2 × e-puck2	$+X$, 10 cm/s
Two robots with a blocker	`airResistance_two_with_block.txt`	3 × e-puck2	$+X$, 10 cm/s
Three in a row	`airResistance_three_in_row.txt`	3 × e-puck2	$+X$, 15 cm/s
Foot-bot wake demo	`airResistance_foot_bot_blocking.txt`	2 × foot-bot	$+X$, 25 cm/s
Crosswind crab control	`wind_crab_footbot.txt`	2 × foot-bot	$+Y$, 15 cm/s

Appendix B: XML Snippets (Copy/Paste)

Global wind

```
<configuration>
  <air_resistance angle_deg="0" magnitude="15.0"/>
</configuration>
```

Controller block

```
<controllers>
  <air_resistance_controller id="airbot"
      library="build/lib/controllers/air_resistance/libair_resistance">
    <actuators>
      <differential_steering implementation="default"/>
      <range_and_bearing implementation="default"/>
    </actuators>
    <sensors>
      <positioning implementation="default"/>
      <range_and_bearing implementation="medium" medium="rab" show_rays="true"/>
    </sensors>
    <params velocity="15.0"/>
  </air_resistance_controller>
</controllers>
```