SAFE REINFORCMENT LEARNING

MILESTONE PRESENTATION II

TU Berlin, SESE: Entwurf eingebetteter Systeme 2020

Agenda

- Achieved and future goals
- Architecture for automated testing
- Environment and RL agent
- Fake environment
- Quality assurance

Progress Report

Webot/Controller

- ● Supervisor functions for randomised world generation
- ◑ Automated testing
- ◕ PID controller
- ○ Safe communication

Progress Report

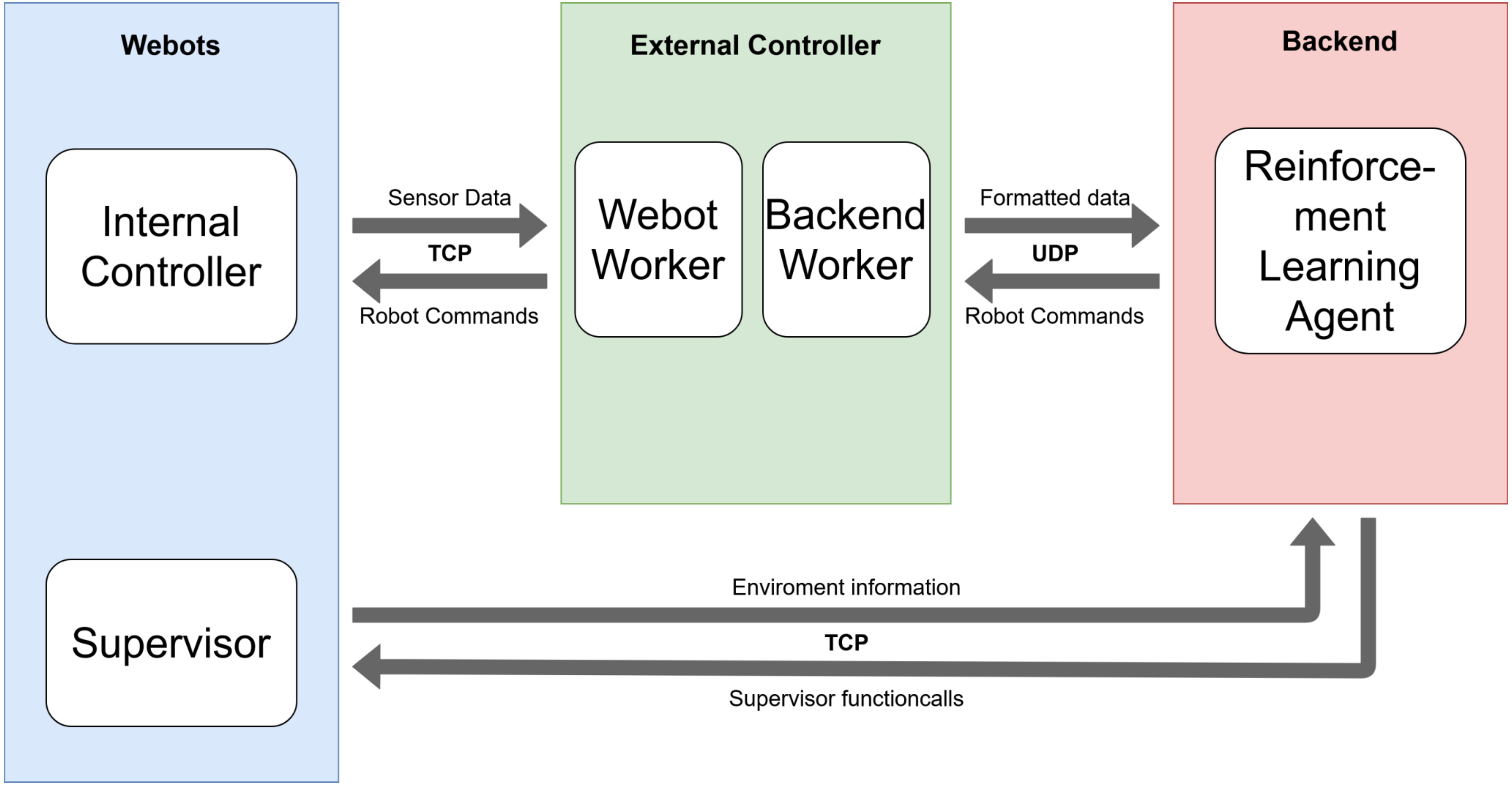
Webot/Controller

● Supervisor functions for randomised world generation
◑ Automated testing
◕ PID controller
○ Safe communication

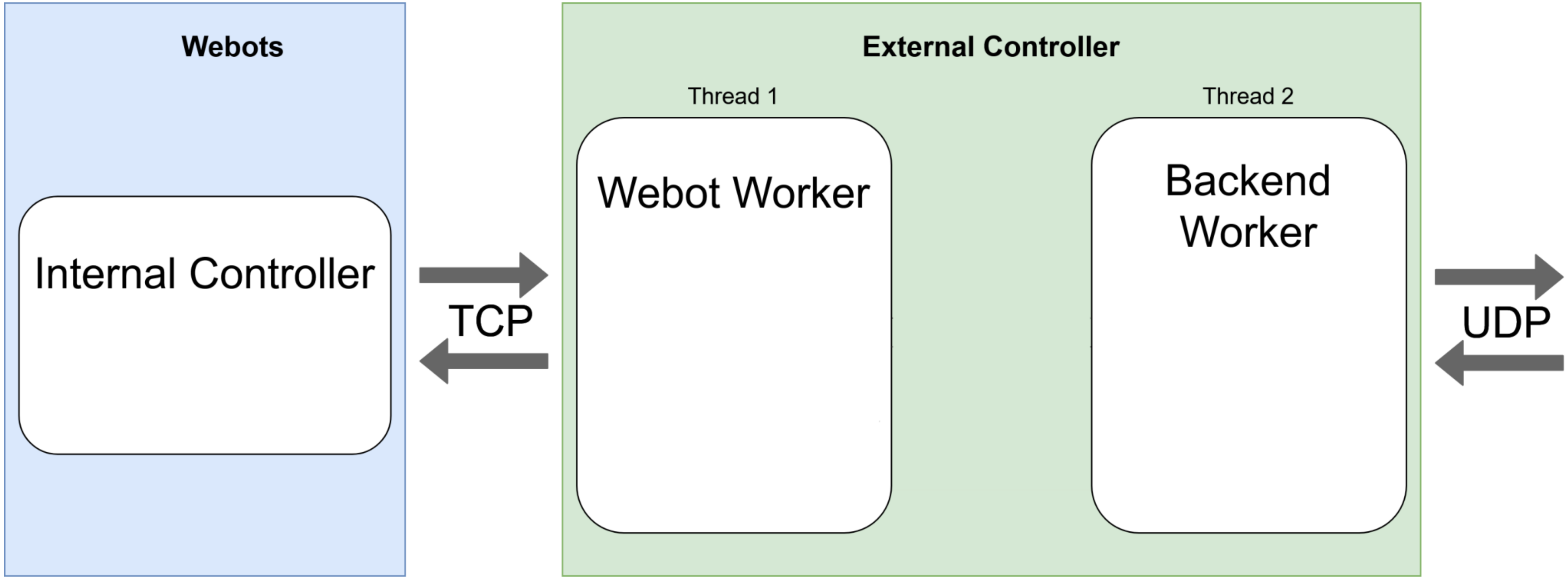
Backend

● Implementation of fake environment
◕ Connect backend to supervisor to controll training runs
◑ Training with algorithms in Stable Baselines to test reward functions

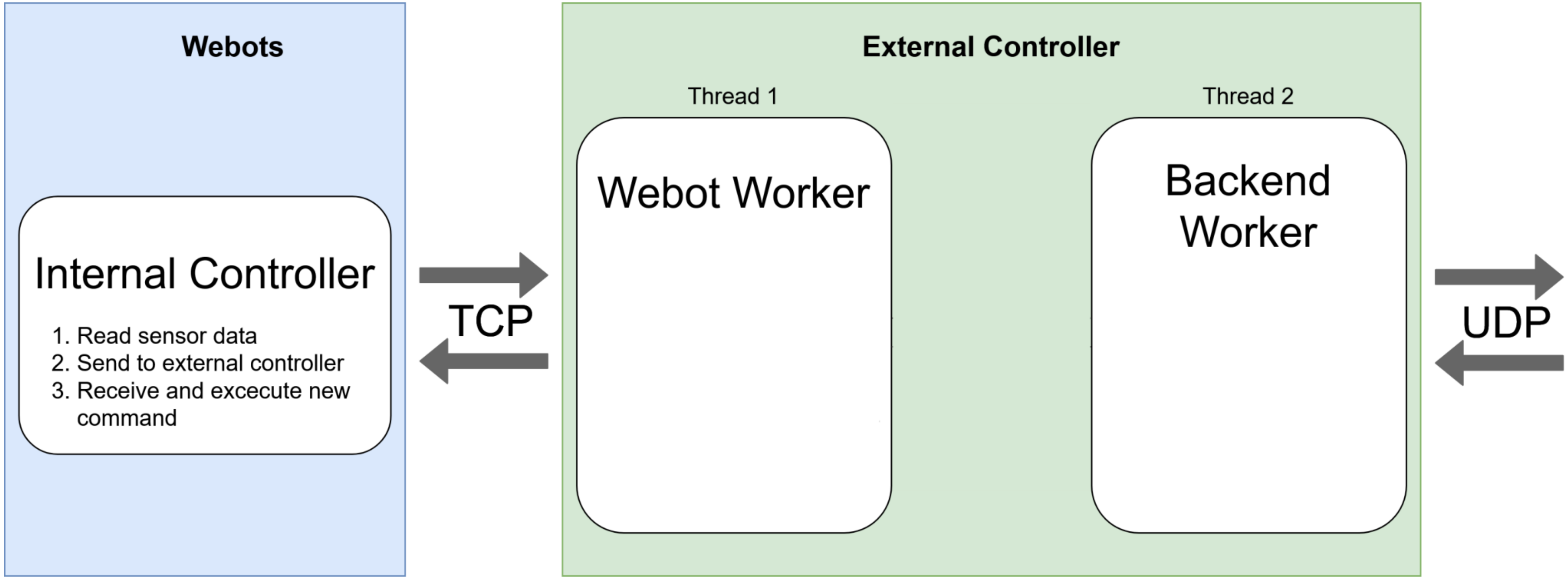
Architecture Overview



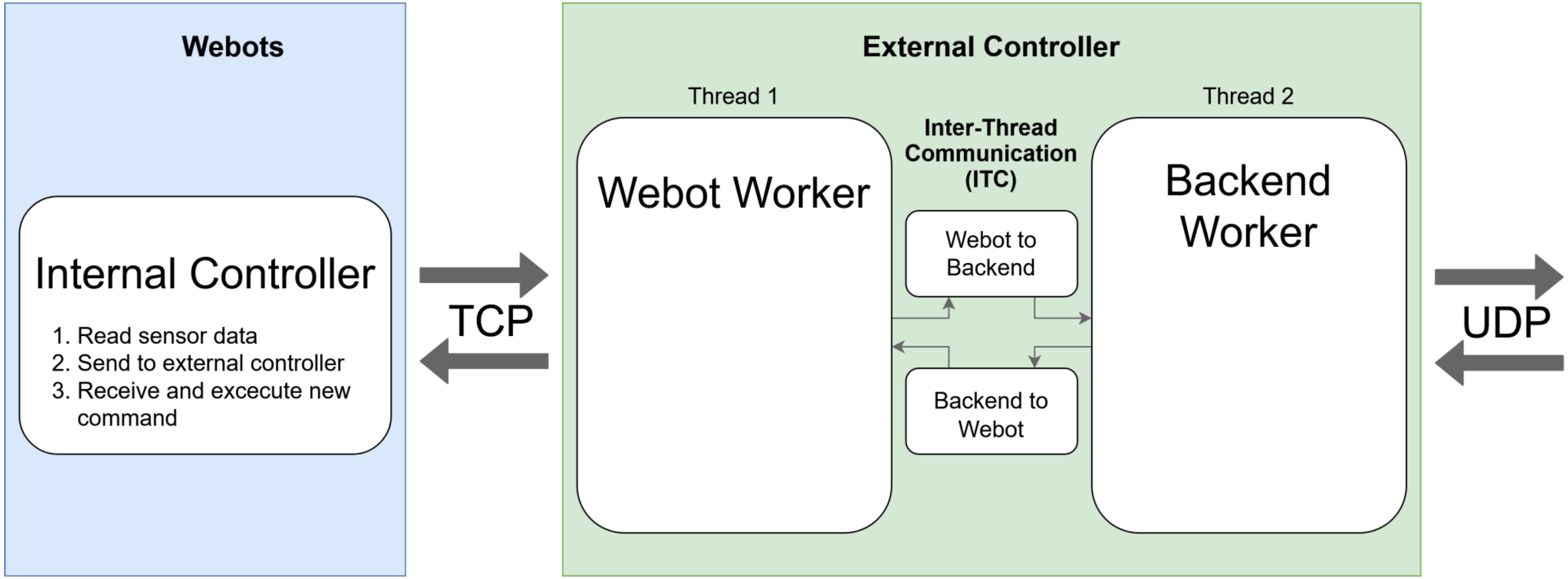
Webot-External Controller Communication (1)



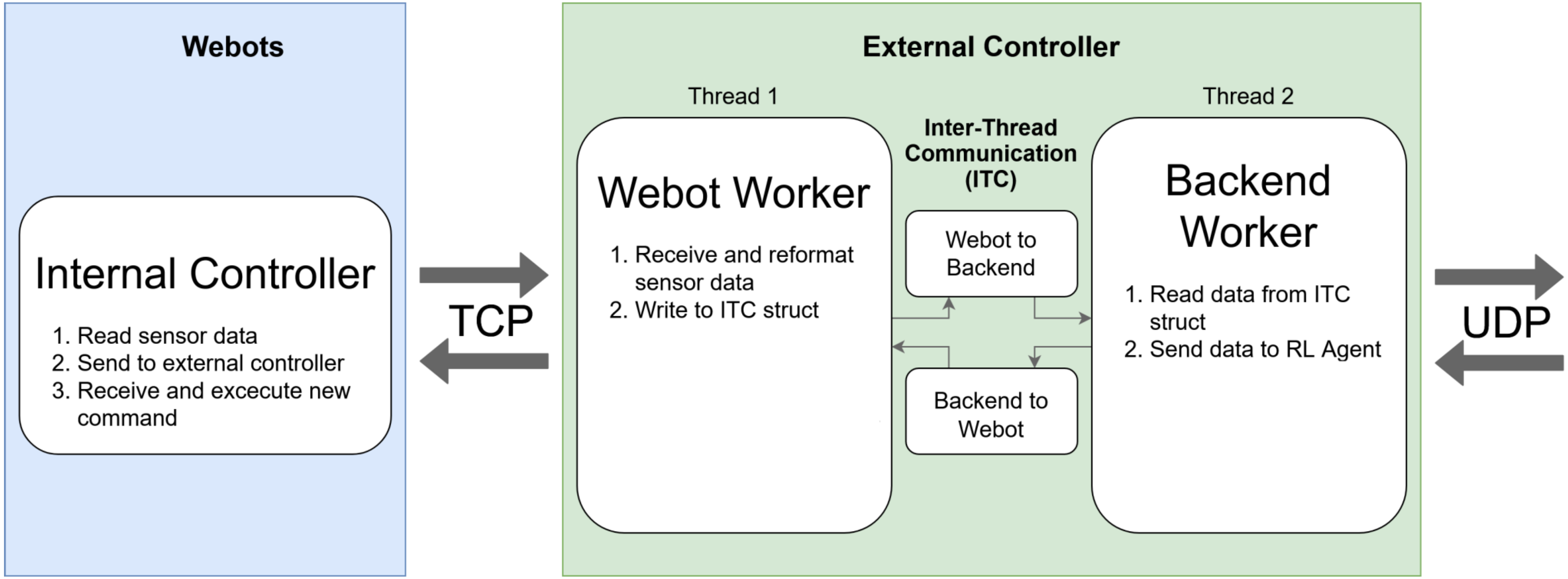
Webot-External Controller Communication (2)



Webot-External Controller Communication (3)



Webot-External Controller Communication (4)



Webots

Internal Controller

1. Read sensor data
2. Send to external controller
3. Receive and excecute new command

TCP

External Controller

Thread 1

Webot Worker

1. Receive and reformat sensor data
2. Write to ITC struct

Inter-Thread Communication (ITC)

Webot to Backend

Backend to Webot

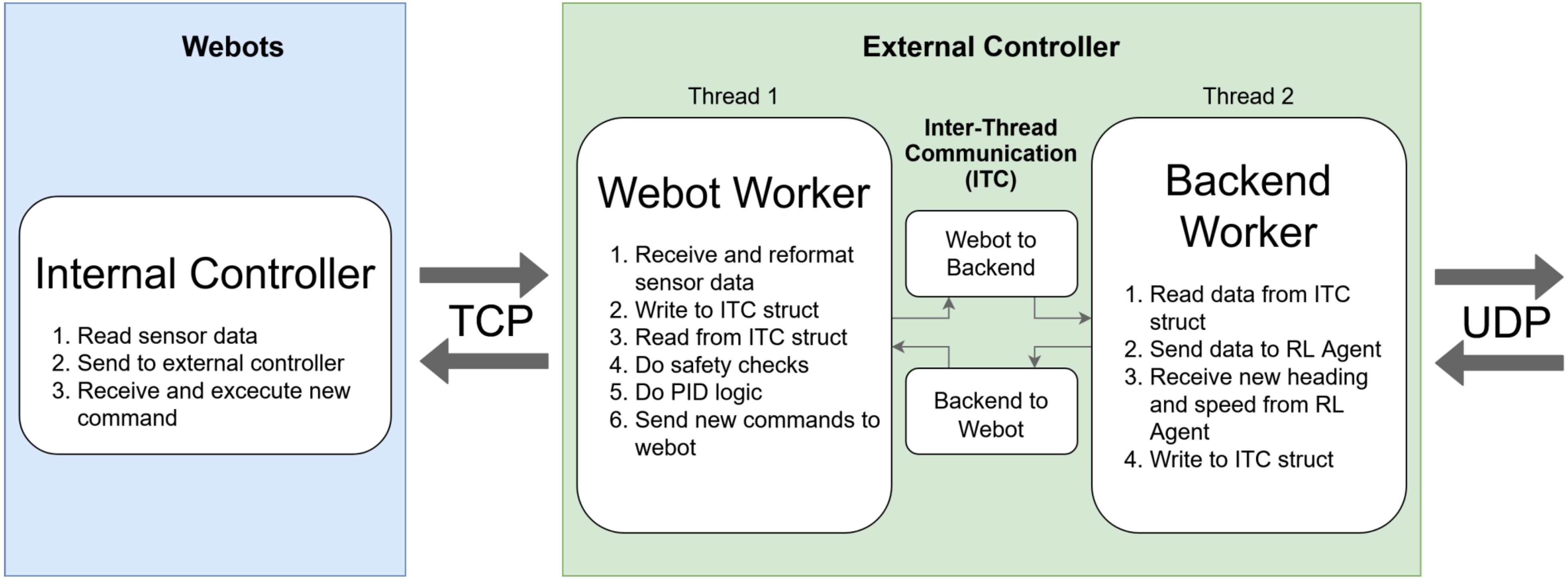
Thread 2

Backend Worker

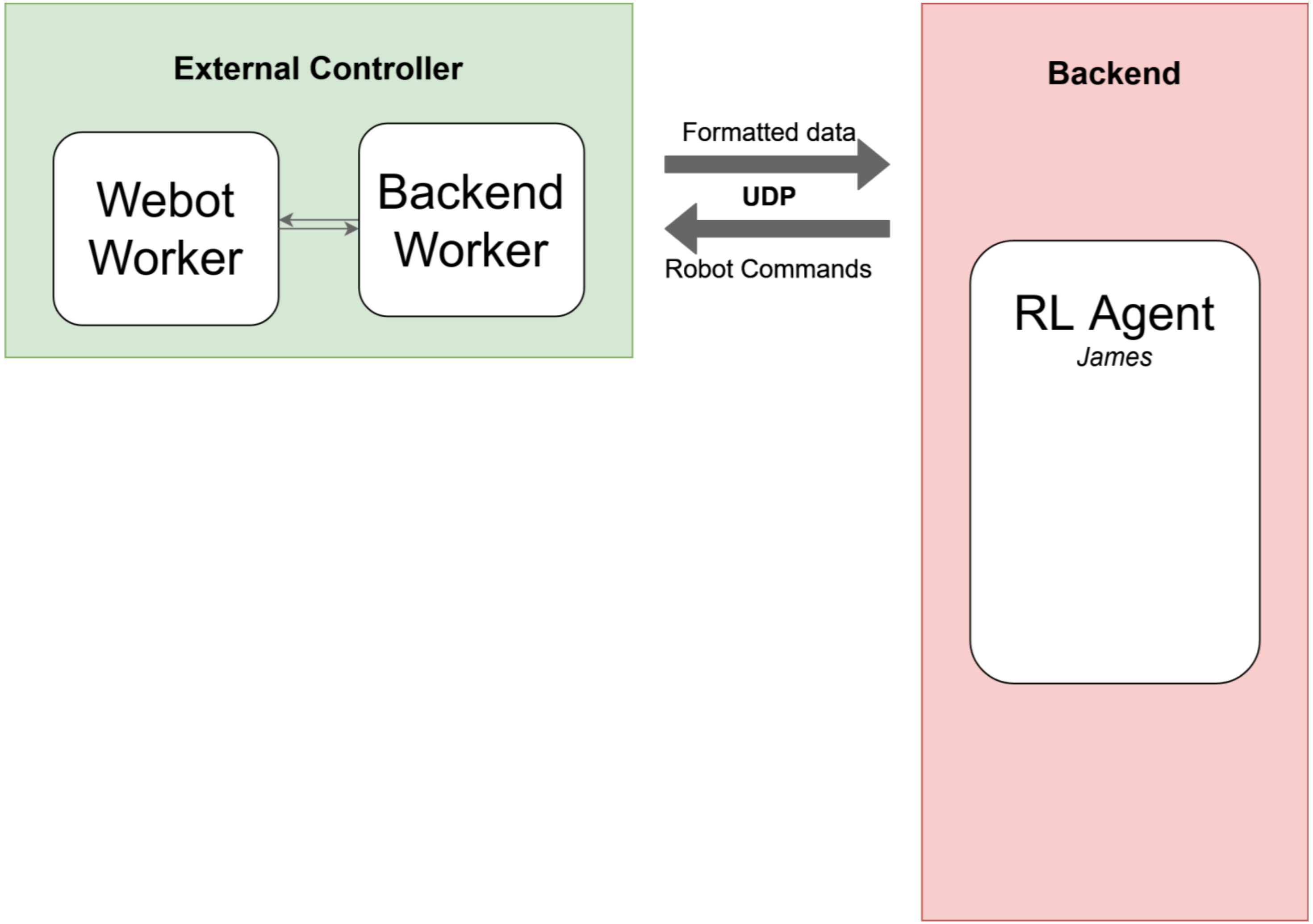
1. Read data from ITC struct
2. Send data to RL Agent

UDP

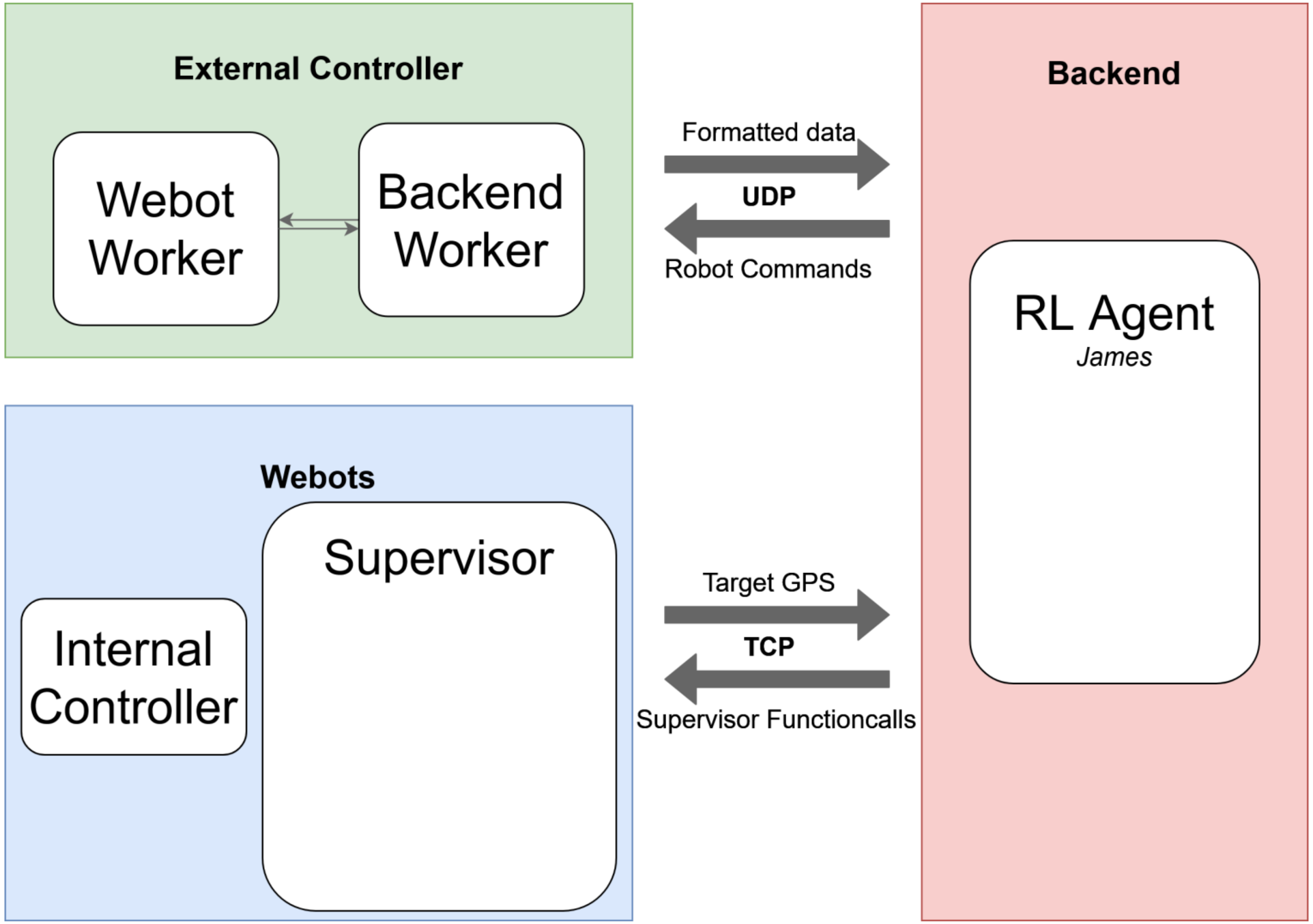
Webot-External Controller Communication (5)



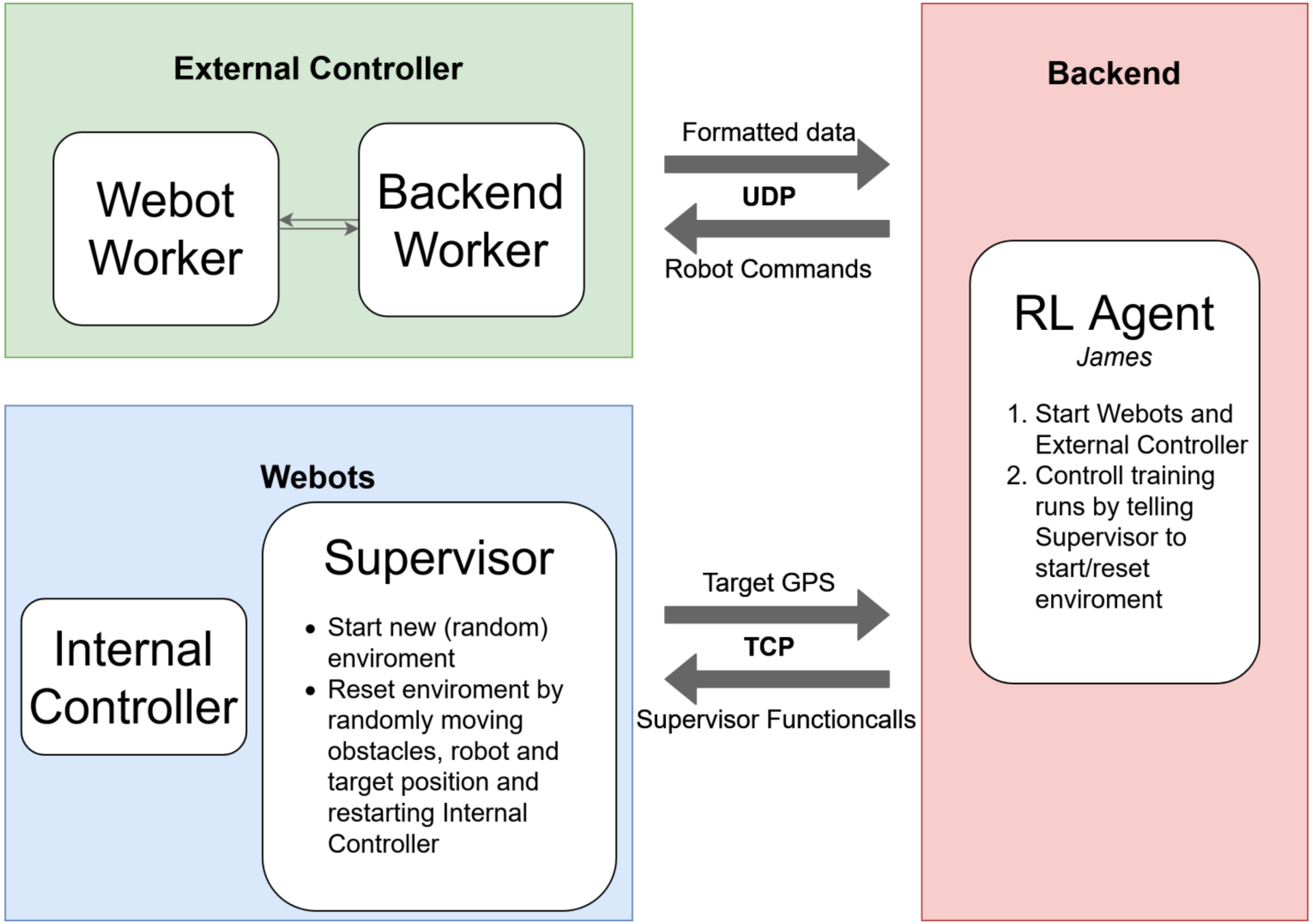
External Controller – Backend Communication (1)



External Controller – Backend Communication (2)



External Controller – Backend Communication (3)



External Controller

Webot Worker

Backend Worker

Formatted data

UDP

Robot Commands

Backend

RL Agent

James

1. Start Webots and External Controller
2. Controll training runs by telling Supervisor to start/reset enviroment

Webots

Internal Controller

Supervisor

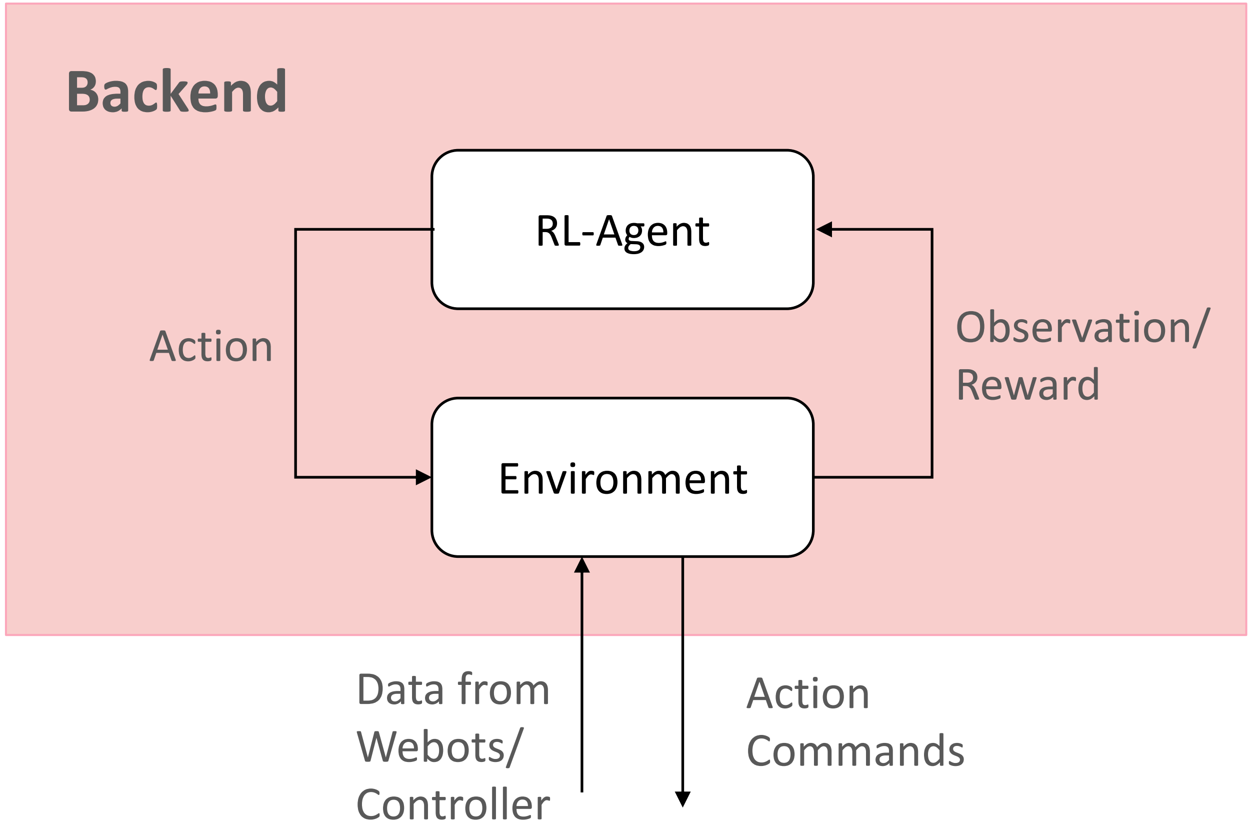
- Start new (random) enviroment
- Reset enviroment by randomly moving obstacles, robot and target position and restarting Internal Controller

Target GPS

TCP

Supervisor Functioncalls

Backend



Backend

RL-Agent

Action

Observation/
Reward

Environment

Data from
Webots/
Controller

Action
Commands

Environment

Why is environment necessary?

- to use the RL baselines with custom environments, they need to follow the gym interface

What is the environment?

- connection between non-backend part and RL agent, which means wrapped the data from webots/controller into standard form for openAI algorithms
- inherits from OpenAI Gym Class
- implement the necessary methods, such as init(), step(), reset(), etc.

Environment

Current environment

- reset function, used to create a new environment for training
- observation function, uses the states from webots to setup an observation to be fed to RL agent
- action space, includes speed commends and direction commends
- reward class, includes several reward functions
- step function, gets the current state from the external controller and sends action back

Environment

Current environment

- reset function, used to create a new environment for training
- observation function, uses the states from webots to setup an observation to be fed to RL agent
- action space, includes speed commends and direction commends
- reward class, includes several reward functions
- step function, gets the current state from the external controller and sends action back

Goals

- test reward function and optimize it
- complete automated training
- add more info to observation (optional)

RL Agent

How to implement?

- Stable Baselines

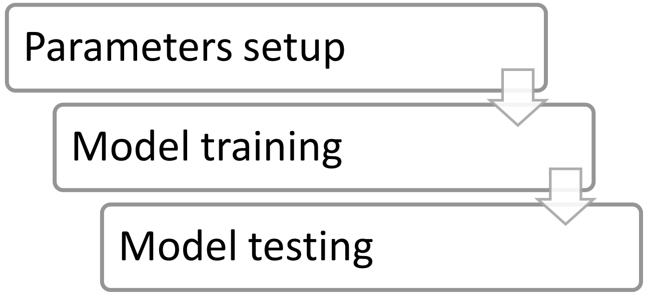
What is Stable Baselines

- a set of improved implementations of Reinforcement Learning (RL) algorithms based on OpenAI Baselines

Why use it?

- do not have to implement all the algorithms by ourself
- easier standardization/benchmarking

Process



Parameters setup

Model training

Model testing

Fake Environment

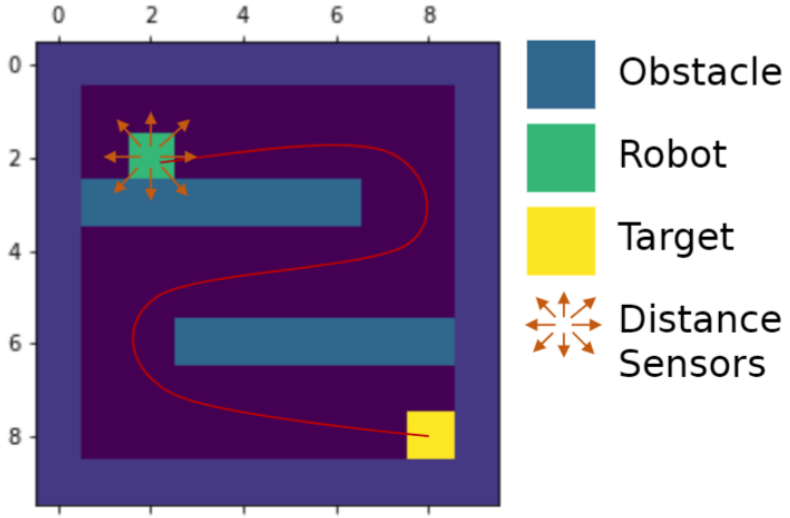
State space

- Current GPS location
- Target GPS location
- Distance sensors
- Touching obstacle

Action space

- 4 directions(N, E, S, W)
- Fixed step size

Reward function



Reward

Time Limit

- number of steps or time available in the webots environment
- number of requests from backend to controller for observation space

Positive Reward

- going closer to the goal
- staying away from obstacles
- entering the goal

Negative Reward

- for each step used to achieve the goal
- crashing or hitting an obstacle
- get too close to an obstacle

Reward

Time Limit

- number of steps or time available in the webots environment
- number of requests from backend to controller for observation space

Positive Reward

- going closer to the goal
- staying away from obstacles
- entering the goal

Negative Reward

- for each step used to achieve the goal
- crashing or hitting an obstacle
- get too close to an obstacle

Reward
reward_time
reward_distance
reward_goal
reward_obstacle
reward_steps

Quality Assurance: Coding Guideline

C/C++ Coding Guide

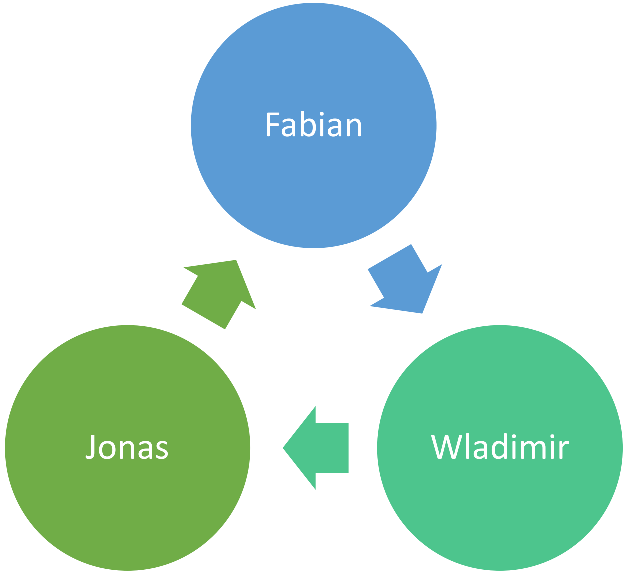
- follow self-defined coding guidelines, includes naming conventions, file structure, etc.

Python Coding Guide

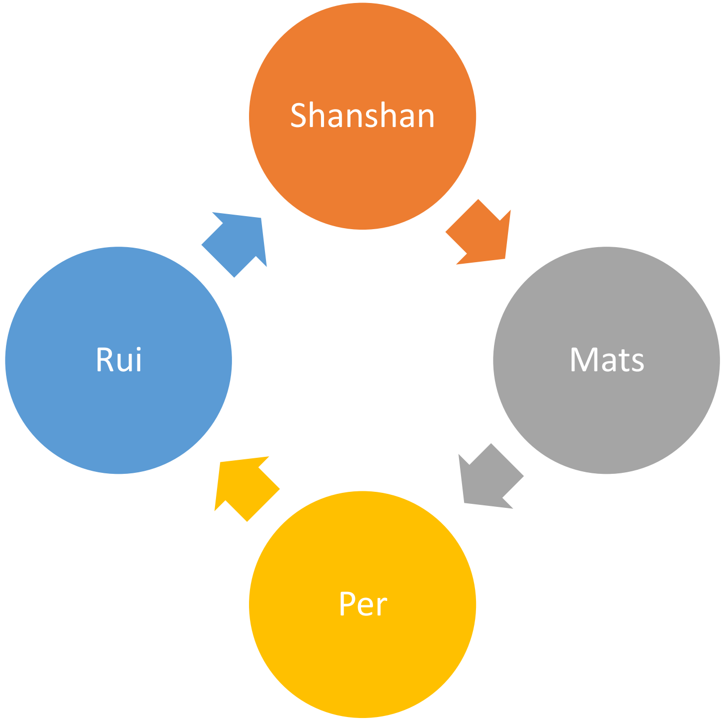
- follow PEP 8 coding guidelines, use Pylint for checking

Quality Assurance: Code Review

Webots/Controller



Backend



Quality Assurance: Automated Testing

Test Framework

- Google Test for C/C++
- Pytest for python

Automated Testing

- Github Action