

# BTstudio, a web tool for programming robots with Behavior Trees



*Asociación de Robótica e Inteligencia Artificial JdeRobot*

*<https://jderobot.github.io>*

*JoseMaria Cañas, Javier Izquierdo, Oscar Martínez*

- Introduction
- Using BT Studio
- How has it been developed?
- Experimental validation
- Conclusions

# Introduction

## JdeRobot org

- Develops open source sw in Robotics and AI
- Started in 2018, 20+ members
- Projects:  
RoboticsAcademy, Unibotics, BTstudio, VisualCircuit...
- Activities:  
Google Summer of Code (2015,2017-2024), internships
- <https://jderobot.github.io>, YouTube, LinkedIn, Twitter



## Making Behavior Trees more accessible for Robotics applications

- Development trend of using **Behavior Trees** in Robotics applications, fairly complex ones beyond simple reactive ones. Similar to HFSM.
- Goal: **to facilitate the quick deployment of BT-based robotics applications within ROS2**
- Inspired on already established tools: [Groot](#) and [Groot2](#).
- Built on top of [py\\_trees](#) for better compatibility.
- Provides a similar experience to [BehaviorTrees.CPP](#) 3.8 but for Python.

## BTstudio

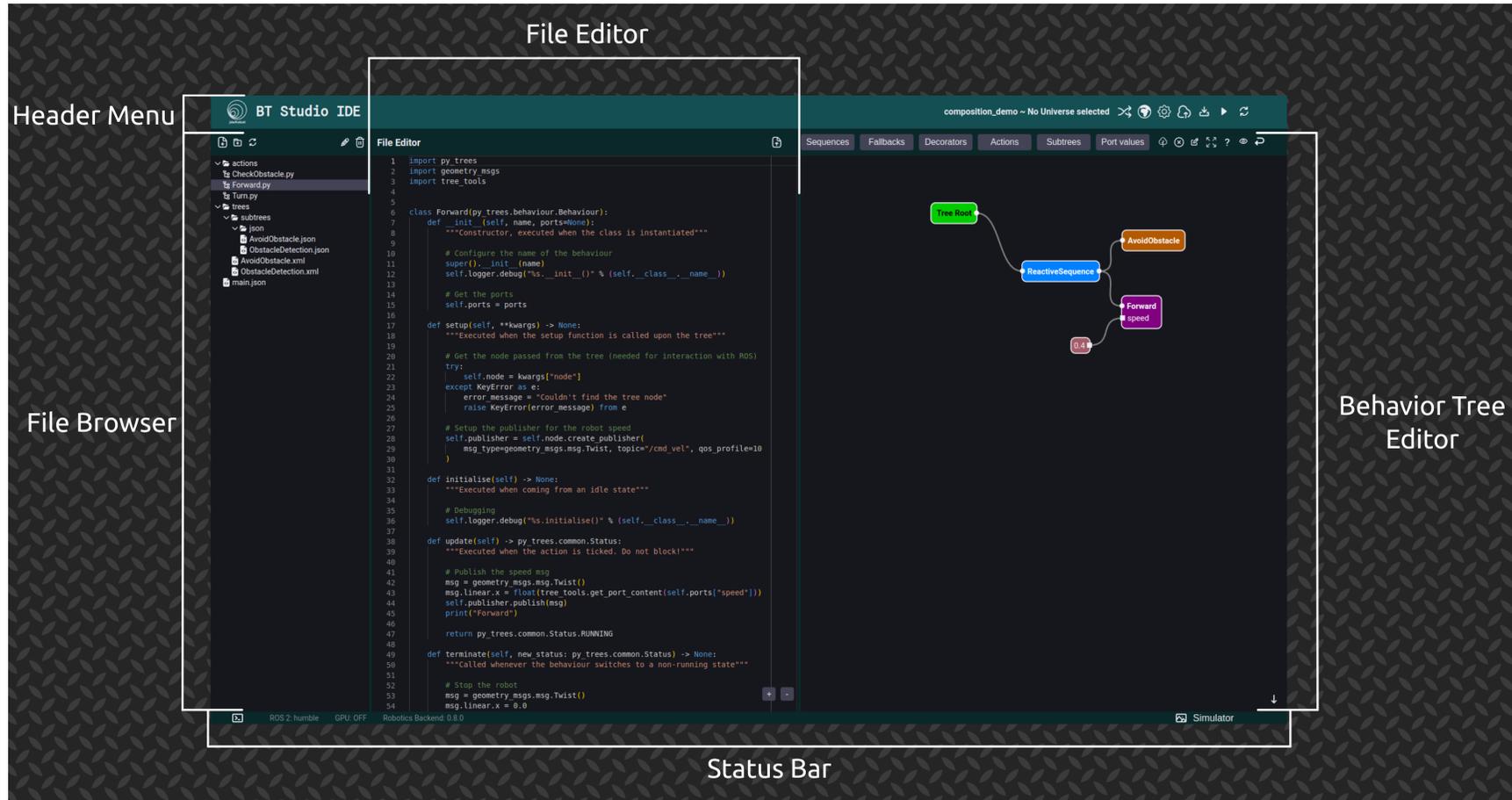
- Web based
- Python language
- Develop applications for ROS2 Humble
- Reuse of behavior trees and modification in a graphical interface
- Free and open-source
- Execution in a dockerized environment (*Robotics Backend*)
- (optional) Streamlines the process of creating a ROS2 package

## Features

- **Edition mode** and **Execution mode**
- Behavior Tree **actions** are created in a *text editor* and a *file navigator*
- Behavior Tree **structure** is defined using a *visual editor*
- Integrated *execution viewer* and *execution BT monitor* for dockerized execution, VNC based
- Integrated *text console*
- Both on real robots and on simulated robots (Gazebo, Webots...)
- Manage multiple projects and multiple universes (worlds + robots)

## User Interface in Edition Mode

- *File Navigator + Text editor + BT visual editor*

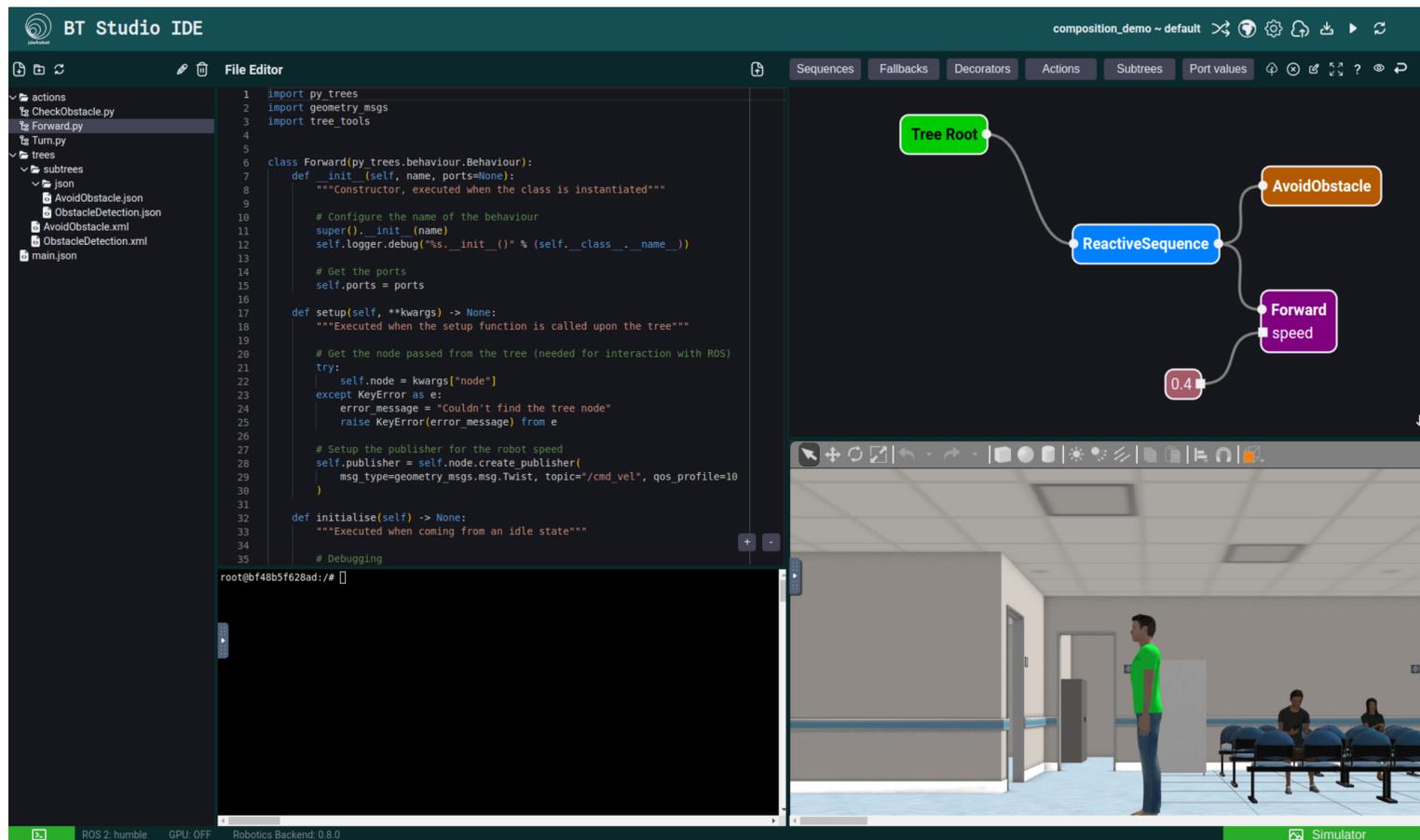


The screenshot shows the BT Studio IDE interface in Edition Mode. The interface is divided into several sections:

- Header Menu:** Located at the top left, it contains icons for file operations and the application name "BT Studio IDE".
- File Browser:** Located on the left side, it displays a tree view of files and folders, including "actions", "trees", "subtrees", "json", "AvoidObstacle.json", "ObstacleDetection.json", "AvoidObstacle.xml", "ObstacleDetection.xml", and "main.json".
- File Editor:** The central area displays Python code for a "Forward" behavior tree node. The code includes imports, class definitions, and methods for initialization, update, and termination.
- Behavior Tree Editor:** Located on the right side, it shows a visual representation of the behavior tree. The tree starts with a "Tree Root" node, which branches into a "ReactiveSequence" node. This node further branches into "AvoidObstacle" and "Forward speed" nodes. A "0.0" node is also visible at the bottom of the tree.
- Status Bar:** Located at the bottom, it displays the current environment: "ROS 2: humble", "GPU OFF", "Robotics Backend: 0.8.0", and "Simulator".

## User Interface in Execution Mode

- *Text editor + Visual BT monitor + Console + Execution viewer*
- Fast feedback to the developer

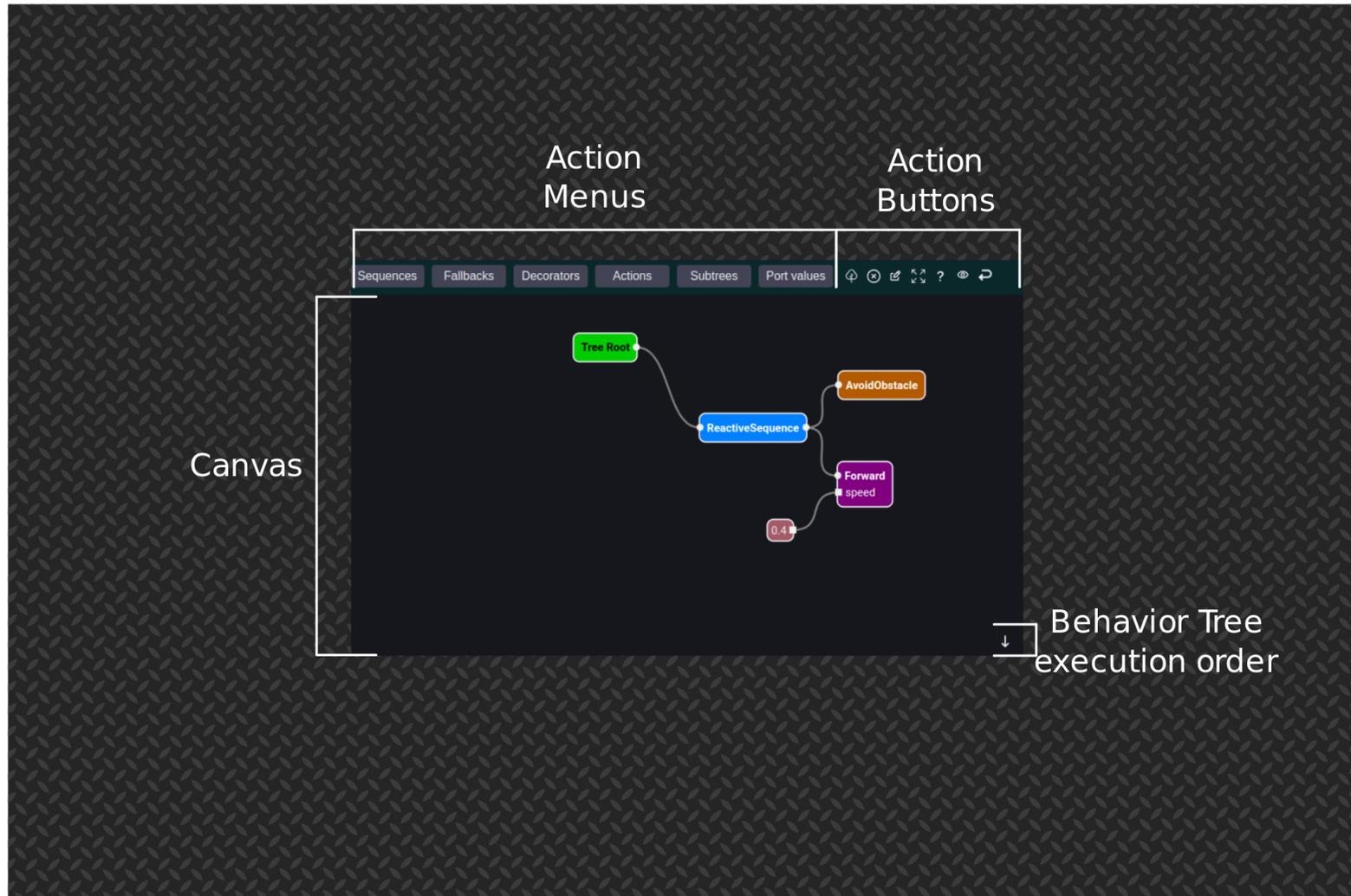


The screenshot displays the BT Studio IDE interface during execution. The main window is titled 'BT Studio IDE' and shows a 'composition\_demo ~ default' environment. The interface is divided into several panels:

- File Editor:** Shows the source code for a Python class named `Forward` (part of `py_trees`). The code includes imports for `py_trees`, `geometry_msgs`, and `tree_tools`. It defines a `Forward` class with methods for `__init__`, `setup`, and `initialise`. The `setup` method sets up a publisher for robot speed.
- Visual BT Monitor:** Displays a Behavior Tree structure. The root node is 'Tree Root', which branches into a 'ReactiveSequence' node. This sequence includes an 'AvoidObstacle' node and a 'Forward speed' node. A '0.4' value is shown near the 'Forward speed' node, likely representing a delay or a parameter.
- Console:** Shows the terminal output for the current session, with the prompt `root@bf48b5f628ad:/#`.
- Simulator:** A 3D rendering of a robot in a virtual environment, showing a hallway with a robot in a green shirt and blue pants, and other people sitting on a bench.

The status bar at the bottom indicates the system configuration: ROS 2: humble, GPU: OFF, and Robotics Backend: 0.8.0.

## Visual Behavior Tree Editor



- Intuitive and reactive editor
- Customizable colors for each action
- Customizable order of BT execution (bottom-to-top or top-to-bottom)
- Support for Subtrees and composition
- Everything you need for developing BT based applications

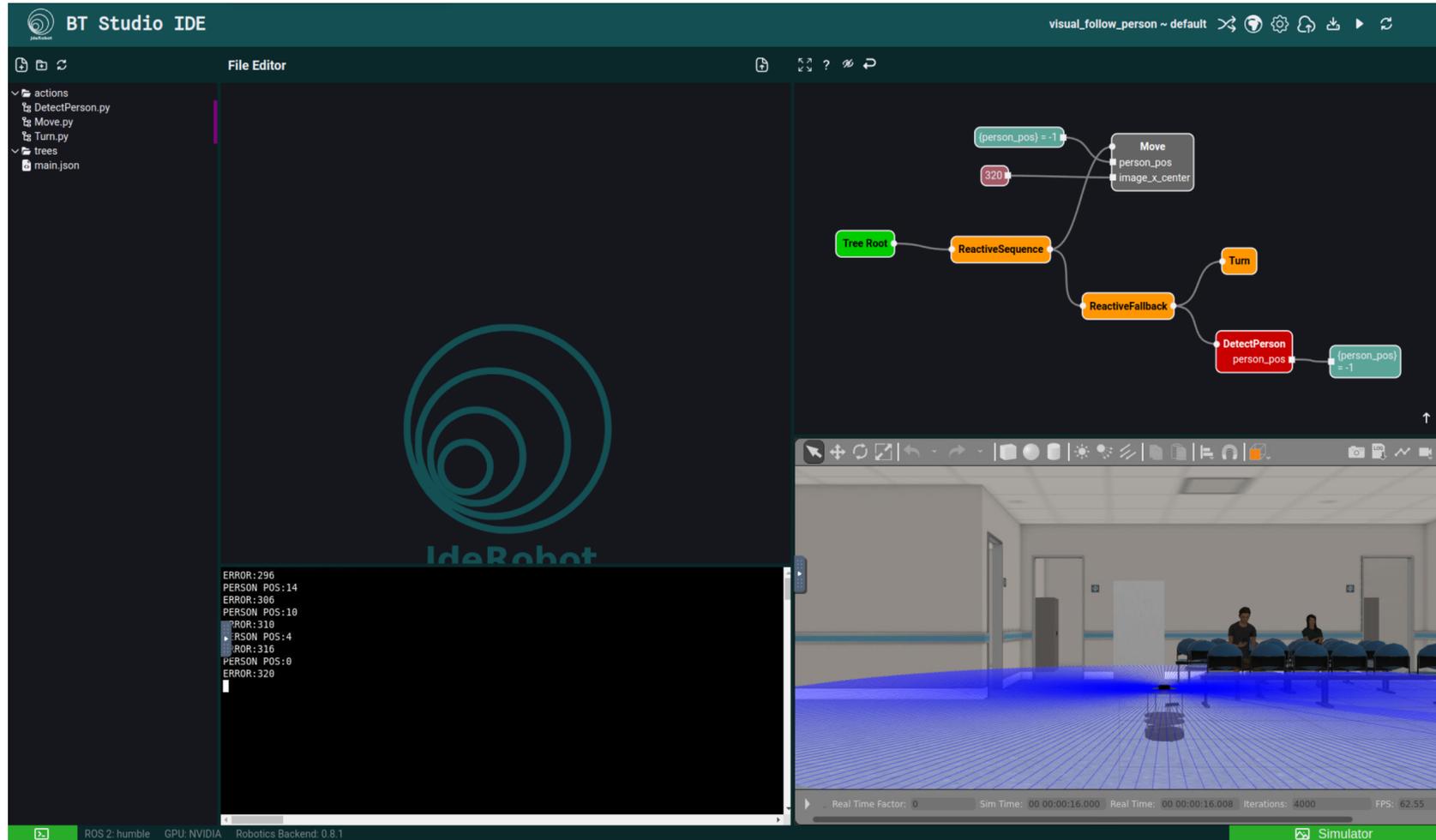
## Subtrees

- BT composition
- Speed up the development of complex robotics applications
- Reusing of existing BTs for common functionality
- Library

## Execution monitoring of the Behavior Tree

Monitor the execution status of the behavior tree from inside the docked execution

- Real time updates
- Move through the subtrees seamlessly
- Also displays the content of blackboard tags



The screenshot displays the BT Studio IDE interface. The top bar shows the application name "BT Studio IDE" and the current project name "visual\_follow\_person ~ default". The left sidebar contains a file explorer with folders for "actions", "trees", and "main.json", and files for "DetectPerson.py", "Move.py", and "Turn.py". The main workspace is divided into three sections: a top-right section for the Behavior Tree editor, a bottom-right section for a 3D simulator, and a bottom-left section for a terminal window.

The Behavior Tree editor shows a "Tree Root" node connected to a "ReactiveSequence" node. The "ReactiveSequence" node has two children: a "ReactiveFallback" node and a "Move" node. The "ReactiveFallback" node has two children: a "Turn" node and a "DetectPerson" node. The "DetectPerson" node has a child node with the condition "{person\_pos} = -1". The "Move" node has three children: a node with the condition "{person\_pos} = -1", a node with the value "320", and a node with the value "image\_x\_center".

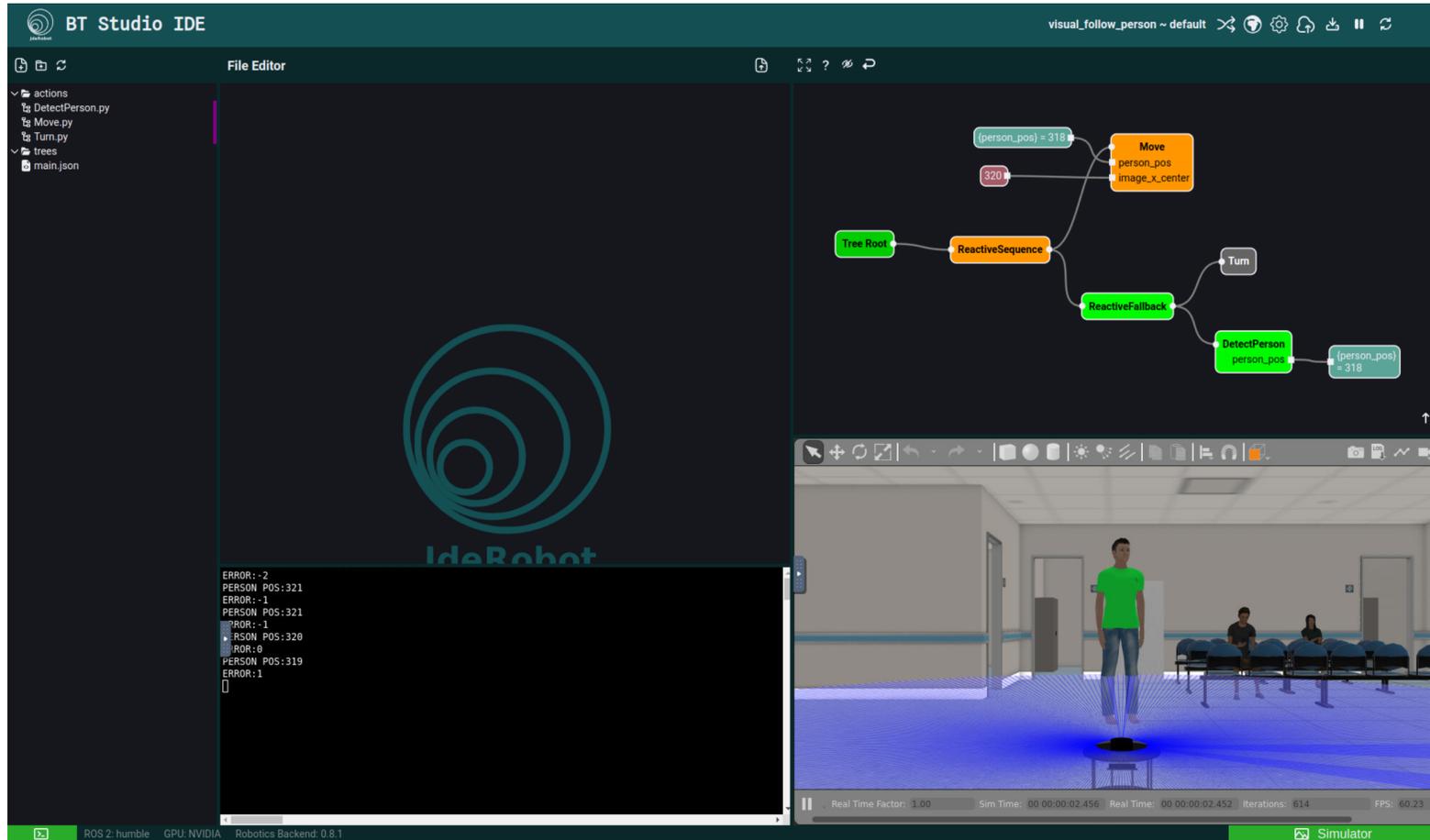
The 3D simulator shows a robot in a virtual environment with a blue grid floor and a white wall. The robot is positioned in the center of the grid. The simulator interface includes a toolbar with various tools and a status bar at the bottom showing "Real Time Factor: 0", "Sim Time: 00:00:00:16.000", "Real Time: 00:00:00:16.008", "Iterations: 4000", and "FPS: 62.55".

The terminal window in the bottom-left corner displays the following error messages:

```
ERROR:296  
PERSON POS:14  
ERROR:306  
PERSON POS:10  
ERROR:310  
PERSON POS:4  
ERROR:316  
PERSON POS:0  
ERROR:320
```

## Dockerized Execution

- Use multiple universes seamlessly
- Ready to use universes provided by the Robotics Backend
- Use your own custom universes
- Control the flow of execution: Run, Pause or Restart



The screenshot displays the BT Studio IDE interface. On the left, a file editor shows a project structure with folders for 'actions' (containing DetectPerson.py, Move.py, Turn.py) and 'trees' (containing main.json). The main workspace is divided into three sections: a large central area with the 'IdeRobot' logo, a terminal window at the bottom left showing error logs, and a 3D simulation window at the bottom right. The simulation shows a robot in a virtual hallway environment. The Behavior Tree (BT) diagram in the top right includes a 'Tree Root' node connected to a 'ReactiveSequence' node. This sequence branches into a 'Move' node (with parameters person\_pos and image\_x\_center) and a 'ReactiveFallback' node. The 'ReactiveFallback' node contains a 'Turn' node and a 'DetectPerson' node (with parameter person\_pos). The terminal window displays the following error messages:

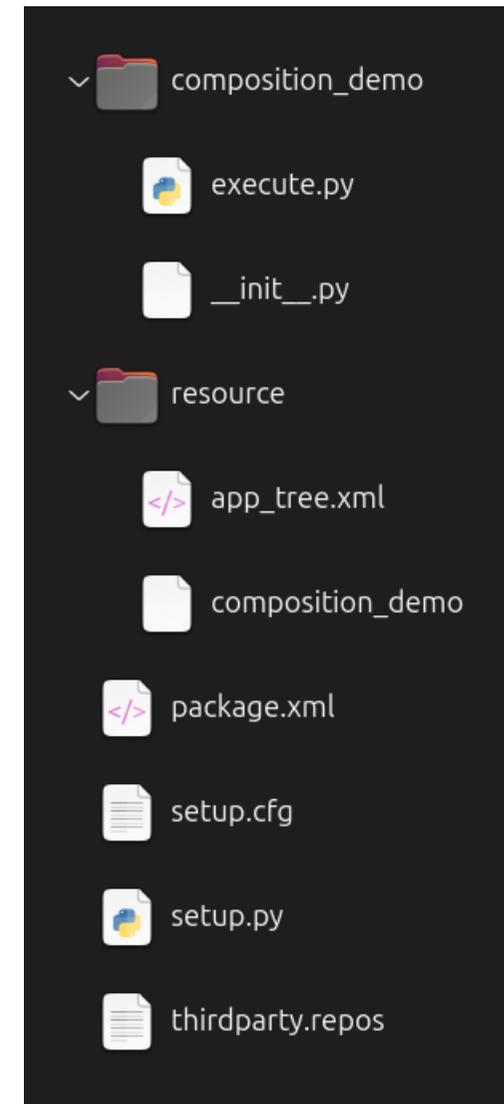
```
ERROR: -2  
PERSON POS:321  
ERROR: -1  
PERSON POS:321  
ERROR: -1  
PERSON POS:320  
ERROR: 0  
PERSON POS:319  
ERROR: 1
```

The simulation window at the bottom right shows a 3D rendering of a robot in a virtual hallway. The robot is a small black circle with a blue glow. The hallway has blue walls and a blue carpet. There are other people in the background. The simulation status bar at the bottom indicates: Real Time Factor: 1.00, Sim Time: 00:00:00:02.456, Real Time: 00:00:00:02.452, Iterations: 614, FPS: 60.23. The 'Simulator' button is visible in the bottom right corner.

## Local Application Package

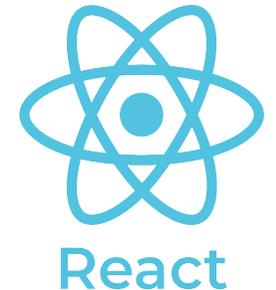
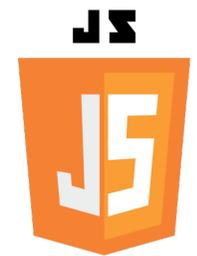
- ROS2 Humble or Jazzy is needed
- A testing environment is provided with the Webots simulator and a tree execution visualizer as third-party repos.
- Compile and run the app using the executor provided
- The actions and behavior tree are merged into a single xml source file.

- `app_tree.xml`: behavior tree and source code
- `execute.py`: launcher for the application
- The rest is the same as a basic ROS2 package



# Developers: How does it work?

- Web technologies
  - backend: Django
  - frontend: React, HTML5, CSS
- Robotics technologies
  - ROS2
  - Based around py\_trees
- DevOps technologies
  - Docker



## Action Structure

- The structure is the same as py\_trees actions

```
1 class Action(py_trees.behaviour.Behaviour):
2     def __init__(self, name, ports=None):
3         """Constructor, executed when the class is instantiated"""
4
5     def setup(self, **kwargs) -> None:
6         """Executed when the setup function is called upon the tree"""
7
8     def initialise(self) -> None:
9         """Executed when coming from an idle state"""
10
11    def update(self) -> py_trees.common.Status:
12        """Executed when the action is ticked. Do not block!"""
13
14    def terminate(self, new_status: py_trees.common.Status) -> None:
15        """Called whenever the behaviour switches to a non-running state"""
```

## Translation process

- Translating from the user code and the diagram is done in the backend
- The 2 parts are combined into a xml single file divided into 2 sections: the BehaviorTree and the Code
- In the BehaviorTree section resides the Behavior Tree and is the same as what is generated by Groot2.
- The code section is used instead of external files for containing each action source code

# Working demos

## Follow Person Demo

- Video

## Bump & Go Demo

- Video

## Receptionist Demo

- Video

## Conclusions

- BTstudio facilitates the quick deployment of behavior tree-based robotics applications within ROS2
- Integrated in Unibotics robot programming website
  
- Present BTstudio to the open source community and ROS community
- Develop more relevant robotics applications and subtree library
- Jump to BehaviorTree.CPP 4.X