

crazyflie-project/drone

Generated on Tue Apr 20 2021 21:35:32 for crazyflie-project/drone by Doxygen 1.9.2

Tue Apr 20 2021 21:35:32

| | |
|---|-----------|
| 1 Drone - ARGoS3 | 1 |
| 1.1 Configure a container in a Wayland environment (e.g. argos3 is not installed) | 1 |
| 1.2 Launch argos simulation without container (We assume argos3 is installed) | 2 |
| 1.3 [Debug commands] | 2 |
| 1.4 Documentation | 2 |
| 2 Class Documentation | 3 |
| 2.1 brain::Brain Class Reference | 3 |
| 2.1.1 Detailed Description | 3 |
| 2.1.2 Member Function Documentation | 3 |
| 2.2 CameraData Struct Reference | 5 |
| 2.2.1 Detailed Description | 5 |
| 2.3 CCrazyflieController Class Reference | 6 |
| 2.3.1 Member Function Documentation | 6 |
| 2.4 Chn< T > Class Template Reference | 7 |
| 2.5 FlowDeck Class Reference | 7 |
| 2.5.1 Detailed Description | 7 |
| 2.5.2 Member Function Documentation | 7 |
| 2.6 MathUtils Class Reference | 8 |
| 2.7 brain::NextMove Struct Reference | 8 |
| 2.8 Proxy Class Reference | 9 |
| 2.9 RTStatus Class Reference | 9 |
| 2.9.1 Detailed Description | 9 |
| 2.9.2 Constructor & Destructor Documentation | 9 |
| 2.9.3 Member Function Documentation | 10 |
| 2.10 SensorData Struct Reference | 10 |
| 2.10.1 Detailed Description | 11 |
| 2.11 SharedQueue< T > Class Template Reference | 11 |
| 2.12 Vec3 Class Reference | 11 |
| 2.13 Vec4 Class Reference | 12 |
| Index | 13 |

1 Drone - ARGoS3

This repository stores the code that defines the drone behavior in the ARGoS3 simulator.

1.1 Configure a container in a Wayland environment (e.g. argos3 is not installed)

If you want to launch ARGoS separately in a container without using the docker-compose do this :

```
# Create the container
./container.sh build
# Launch a specific arena
./container.sh {1,2,3,4,5}
```

```
# Launch a random arena
./container.sh 0
# Launch the with the last configuration
./container.sh
```

To install Weston use `$ apt-get install -y weston`.

1.2 Launch argos simulation without container (We assume argos3 is installed)

5 arenas are predefined in the simulation. You can specify which one you want to use when launching it :

```
# Launch a specific arena
./launch.sh {1,2,3,4,5}
# Launch a random arena
./launch.sh 0
# Launch the with the last configuration
./launch.sh
```

1.3 [Debug commands]

See <https://gitlab.com/polytechnique-montr-al/inf3995/20211/equipe-203/crazyflie-project.md>

```
{"type":"startMission", "data":{"name": "simulation_1"}}
{"type":"startMission", "data":{"name": "simulation_2"}}
{"type":"returnToBase", "data":{"name": "simulation_1"}}
{"type":"returnToBase", "data":{"name": "simulation_2"}}
{"type":"land", "data":{"name": "simulation_1"}}
{"type":"land", "data":{"name": "simulation_2"}}
```

1.4 Documentation

This project comes with an automatically generated documentation from the code's docstrings. An example is located in `doc/latex/refman.pdf`

To generate the project's documentation :

- Install Doxygen

```
git clone https://github.com/doxygen/doxygen.git
cd doxygen
mkdir build && cd build
cmake -G "Unix Makefiles" ..
make
make install
```
- Install the recommended extension in VSCode (Name: Doxygen Documentation Generator <https://marketplace.visualstudio.com/items?itemName=cschlosser.doxdocgen>)
- Once in a while, run Doxygen. Make sure you are in the root directory of the project (`/simulation`)

```
doxygen doc/doxygen-config
```

The output can be found in the `latex` folders located in `doc`.

To generate a PDF :

```
cd doc/latex
make pdf
```

Example of a docstring :

```
{C++}
/**
 * @brief Updates the status of the drone (battery, position, and speed)
 *
 * @param battery battery percentage
 * @param pos drone position in the simulation
 */
void RTStatus::update(std::float_t battery, const Vec4 &pos);
```

2 Class Documentation

2.1 brain::Brain Class Reference

Brain : the class that decides what the drone should do next depending on where it is, what it sees, and what it receives from the station.

```
#include <Brain.hpp>
```

Public Member Functions

- **Brain** (uint16_t id)
- std::optional< **NextMove** > **computeNextMove** (const **CameraData** *cd, const **SensorData** *sd, const double &battery_level)

Finds and returns the next move the drone should do.
- State **getState** ()

Get the state of the brain.
- void **setState** (State newState)

Set the State.
- bool **isReturningToBase** () const

Checks if the drone is returning to its base.
- void **setInitialPosition** (**Vec4** pos)

Set the Initial Position.

2.1.1 Detailed Description

Brain : the class that decides what the drone should do next depending on where it is, what it sees, and what it receives from the station.

2.1.2 Member Function Documentation

2.1.2.1 computeNextMove() std::optional< **NextMove** > brain::Brain::computeNextMove (
 const **CameraData** * cd,
 const **SensorData** * sd,
 const double & battery_level)

Finds and returns the next move the drone should do.

This function takes in input the different sensors it has (the **FlowDeck** data, the multiranger data, and the battery level), and decides what the drone should do with these input.

7 states are defined :

- idle : do nothing and stay where you are
- take_off : go up until you reach your cruise altitude, then start exploring by going in auto_pilot
- land : go down until you reach the ground, then go idle.

- `auto_pilot` : move forward until you see an obstacle too close on your side or front. If you are in front of an obstacle and you are still exploring, go `exploration_dodge`. If you are in front of an obstacle and you are trying to return to your base, go `return_to_base_dodge`
- `exploration_dodge` : rotate until no obstacle is on your side or front, and return to `auto_pilot`
- `stabilize` : do nothing until you reached the position you are supposed to be.
- `return_to_base` : Rotate to the direction of the base. Then go `auto_pilot` with the flag `returning_to_base_` as `true`
- `return_to_base_dodge` : rotate and move to the side in order to avoid an obstacle on your way to the base. When nothing is in front of you anymore, go `return_to_base`.

`x,y,z` and `yaw` are the current position and orientation of the drone at the time the function is called

Parameters

| | |
|----------------------|--------------------------|
| <i>cd</i> | current flowdeck data |
| <i>sd</i> | current multiranger data |
| <i>battery_level</i> | current battery level |

Returns

`std::optional<NextMove>` struct containing the next instruction. If `nullopt`, that means the last instruction was not finished.

2.1.2.2 `getState()` `State brain::Brain::getState () [inline]`

Get the state of the brain.

Returns

state

2.1.2.3 `isReturningToBase()` `bool brain::Brain::isReturningToBase () const [inline]`

Checks if the drone is returning to its base.

Returns

true if yes
false if no

2.1.2.4 `setInitialPosition()` `void brain::Brain::setInitialPosition (Vec4 pos) [inline]`

Set the Initial Position.

Parameters

| | |
|------------|------------------|
| <i>pos</i> | initial position |
|------------|------------------|

2.1.2.5 setState() `void brain::Brain::setState (State newState) [inline]`

Set the State.

Parameters

| | |
|-----------------|--|
| <i>newState</i> | |
|-----------------|--|

The documentation for this class was generated from the following files:

- include/Brain.hpp
- src/Brain.cpp

2.2 CameraData Struct Reference

CameraData : struct to hold a flowdeck-like outputted data. Inspired from the real flowdeck : <https://github.com/bitcraze/crazyflie-firmware/blob/master/src/deck/drivers/src/zranger.c> https://github.com/bitcraze/crazyflie-firmware/blob/master/src/deck/drivers/src/flowdeck_v1v2.c.

```
#include <CameraData.hpp>
```

Public Attributes

- `std::float_t` **delta_x**
- `std::float_t` **delta_y**
- `std::float_t` **z**
- `std::float_t` **yaw**

2.2.1 Detailed Description

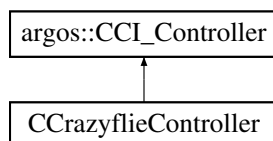
CameraData : struct to hold a flowdeck-like outputted data. Inspired from the real flowdeck : <https://github.com/bitcraze/crazyflie-firmware/blob/master/src/deck/drivers/src/zranger.c> https://github.com/bitcraze/crazyflie-firmware/blob/master/src/deck/drivers/src/flowdeck_v1v2.c.

The documentation for this struct was generated from the following file:

- include/CameraData.hpp

2.3 CCrazyflyController Class Reference

Inheritance diagram for CCrazyflyController:



Public Member Functions

- [CCrazyflyController](#) ()
Constructor.
- [~CCrazyflyController](#) () override=default
Destructor.
- void [Init](#) (argos::TConfigurationNode &) override
This function initializes the controller.
- void [ControlStep](#) () override
This function is called once every time step. The length of the time step is set in the XML file.
- void [Reset](#) () override
This function resets the controller to its state right after the [Init\(\)](#). It is called when you press the reset button in the GUI. In this example controller there is no need for resetting anything, so the function could have been omitted. It's here just for completeness.
- void [Destroy](#) () override
Called to cleanup what done by [Init\(\)](#) when the experiment finishes. In this example controller there is no need for clean anything up, so the function could have been omitted. It's here just for completeness.

2.3.1 Member Function Documentation

2.3.1.1 Init() `void CCrazyflyController::Init (argos::TConfigurationNode &) [inline], [override]`

This function initializes the controller.

Parameters

| | |
|---------------------|---|
| <code>t_node</code> | points to the <parameters> section in the XML file in the <controllers><crazyfly_sensing_controller> section. |
|---------------------|---|

The documentation for this class was generated from the following file:

- src/main.cpp

2.4 Chn< T > Class Template Reference

Public Member Functions

- void **drain** ()
- bool **is_open** ()
- bool **send** (T &&val)
- std::optional< T > **recv** (bool wait=false)

The documentation for this class was generated from the following file:

- include/Chn.hpp

2.5 FlowDeck Class Reference

FlowDeck : class that mimicks the behavior of a real crazyflie flowdeck. It saves the initial position and returns a **CameraData** struct, containing the delta_x and delta_y relative to the initial (takeoff) position.

```
#include <FlowDeck.hpp>
```

Public Member Functions

- void **init** (const argos::CVector3 &init_position)
Saves the takeoff position in case the drone does not takeoff at (0,0) in ARGoS. Called only when the simulation is launching.
- **CameraData** **getInitPositionDelta** (const argos::CVector3 &position, const argos::CQuaternion &orientation)
*Get the **CameraData** containing the relative movement done by the drone.*

2.5.1 Detailed Description

FlowDeck : class that mimicks the behavior of a real crazyflie flowdeck. It saves the initial position and returns a **CameraData** struct, containing the delta_x and delta_y relative to the initial (takeoff) position.

2.5.2 Member Function Documentation

2.5.2.1 getInitPositionDelta() **CameraData** FlowDeck::getInitPositionDelta (
const argos::CVector3 & position,
const argos::CQuaternion & orientation) [inline]

Get the **CameraData** containing the relative movement done by the drone.

Parameters

| | |
|--------------------|--|
| <i>position</i> | real absolute position gotten from ARGoS directly |
| <i>orientation</i> | real absolute orientation gotten from ARGoS directly |

Returns

[CameraData](#)

2.5.2.2 init() `void FlowDeck::init (`
`const argos::CVector3 & init_position) [inline]`

Saves the takeoff position in case the drone does not takeoff at (0,0) in ARGoS. Called only when the simulation is launching.

Parameters

| | |
|----------------------|---|
| <i>init_position</i> | real absolute position gotten from ARGoS directly |
|----------------------|---|

The documentation for this class was generated from the following file:

- include/FlowDeck.hpp

2.6 MathUtils Class Reference

Static Public Member Functions

- static float **computeDirectionToBase** (const [Vec4](#) &pos, const [Vec4](#) &init_pos)
- static void **wrapToPi** (float *val)

Static Public Attributes

- static constexpr float **PI_f** = 3.14159265F

The documentation for this class was generated from the following files:

- include/MathUtils.hpp
- src/MathUtils.cpp

2.7 brain::NextMove Struct Reference

Public Attributes

- [Vec4](#) **coords**
- bool **relative**
- float_t **yaw**

The documentation for this struct was generated from the following file:

- include/Brain.hpp

2.8 Proxy Class Reference

Public Member Functions

- **Proxy** (std::string name)
- auto **next_cmd** ()
- void **send** (std::string &&msg)

Static Public Member Functions

- static void **recv_cb** (gen_buf_t &&msg)

The documentation for this class was generated from the following files:

- include/Proxy.hpp
- src/Proxy.cpp

2.9 RTStatus Class Reference

RTStatus (RealTimeStatus): stores the real time state (position, orientation, speed, battery...) of a drone.

```
#include <RTStatus.hpp>
```

Public Member Functions

- **RTStatus** (std::string name)
*Construct a new **RTStatus::RTStatus** object.*
- std::string **encode** ()
Encodes the current status of the drone as json.
- const std::string & **get_name** () const
- void **update** (std::float_t battery, const **Vec4** &pos, const float_t &yaw, const **SensorData** &sd, const brain::↔ State &brain_state, const bool &brain_returning_to_base)
*Updates the status of the drone (battery, position, orientation, speed, brain state). Sets the **RTStatus::state_↔** appropriately depending on the brain state.*
- bool **isFlying** () const
- void **setPosition** (**Vec4** pos)
- void **setBattery** (std::float_t battery)
- void **enable** ()
Set the drone as flying.
- void **disable** ()
Set the drone as not flying.

2.9.1 Detailed Description

RTStatus (RealTimeStatus): stores the real time state (position, orientation, speed, battery...) of a drone.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 RTStatus() **RTStatus::RTStatus** (
std::string name) [explicit]

Construct a new **RTStatus::RTStatus** object.

Parameters

| | |
|-------------|----------------------------|
| <i>name</i> | name to identify the drone |
|-------------|----------------------------|

2.9.3 Member Function Documentation

2.9.3.1 encode() `std::string RTStatus::encode ()`

Encodes the current status of the drone as json.

Returns

`std::string` serialized pulse object

2.9.3.2 update() `void RTStatus::update (` `std::float_t battery,` `const Vec4 & pos,` `const float_t & yaw,` `const SensorData & sd,` `const brain::State & brain_state,` `const bool & brain_returning_to_base)`

Updates the status of the drone (battery, position, orientation, speed, brain state). Sets the `RTStatus::state_` appropriately depending on the brain state.

Parameters

| | |
|--------------------------------|---|
| <i>battery</i> | battery level |
| <i>pos</i> | absolute position gotten from the flowdeck |
| <i>yaw</i> | absolute orientation gotten from the flowdeck |
| <i>sd</i> | multiranger sensors data |
| <i>brain_state</i> | current state of the brain class |
| <i>brain_returning_to_base</i> | boolean to indicate if the drone is currently returning to its base |

The documentation for this class was generated from the following files:

- `include/RTStatus.hpp`
- `src/RTStatus.cpp`

2.10 SensorData Struct Reference

SensorData : struct to store the multiranger sensors. The value is between 0-255 if the wall detected is reasonably close, and 65534 if it's too far away.

```
#include <SensorData.hpp>
```

Public Attributes

- std::uint16_t **front**
- std::uint16_t **left**
- std::uint16_t **back**
- std::uint16_t **right**
- std::uint16_t **up**

2.10.1 Detailed Description

SensorData : struct to store the multiranger sensors. The value is between 0-255 if the wall detected is reasonably close, and 65534 if it's too far away.

The documentation for this struct was generated from the following file:

- include/SensorData.hpp

2.11 SharedQueue< T > Class Template Reference**Public Member Functions**

- void **push** (T t)
- void **push** (T &&t)
- std::uint_fast32_t **size** ()
- std::optional< T > **pop** ()

The documentation for this class was generated from the following file:

- include/SharedQueue.hpp

2.12 Vec3 Class Reference**Static Public Member Functions**

- static constexpr **Vec4 add** (const **Vec4** &a, const **Vec4** &b)
- static constexpr void **add** (**Vec4** *a, const **Vec4** &b)
- static constexpr **Vec4 sub** (const **Vec4** &a, const **Vec4** &b)
- static constexpr void **sub** (**Vec4** *a, const **Vec4** &b)
- static constexpr **Vec4 mul** (const **Vec4** &a, const **Vec4** &b)
- static constexpr void **mul** (**Vec4** *a, const **Vec4** &b)
- static constexpr **Vec4 div** (const **Vec4** &a, const **Vec4** &b)
- static constexpr void **div** (**Vec4** *a, const **Vec4** &b)
- static constexpr **Vec4 neg** (const **Vec4** &a)
- static constexpr void **neg** (**Vec4** *a)
- static constexpr std::float_t **sum** (const **Vec4** &a)
- static constexpr std::float_t **norm_sqr** (const **Vec4** &a)
- static constexpr std::float_t **norm** (const **Vec4** &a)
- static constexpr **Vec4 normalize** (const **Vec4** &a)
- static constexpr void **normalize** (**Vec4** *a)
- static constexpr std::float_t **dot** (const **Vec4** &a, const **Vec4** &b)
- static constexpr **Vec4 cross** (const **Vec4** &a, const **Vec4** &b)

The documentation for this class was generated from the following file:

- include/Vec3.hpp

2.13 Vec4 Class Reference

Public Member Functions

- constexpr auto **w** () const
- constexpr auto **x** () const
- constexpr auto **y** () const
- constexpr auto **z** () const
- constexpr **Vec4** (std::float_t r)
- constexpr **Vec4** (std::float_t w, const **Vec4** &v)
- constexpr **Vec4** (std::float_t w, std::float_t r)
- constexpr **Vec4** (std::float_t x, std::float_t y, std::float_t z)
- constexpr **Vec4** (std::float_t w, std::float_t x, std::float_t y, std::float_t z)
- constexpr **Vec4** (const **Vec4** &v)=default
- constexpr **Vec4** (**Vec4** &&v)=default
- constexpr **Vec4** & **operator=** (const **Vec4** &v)=default
- constexpr **Vec4** & **operator=** (**Vec4** &&v)=default
- bool **operator==** (const **Vec4** &v) const

Public Attributes

- std::float_t **v_** [4]

The documentation for this class was generated from the following file:

- include/Vec4.hpp

Index

- brain::Brain, [3](#)
 - computeNextMove, [3](#)
 - getState, [4](#)
 - isReturningToBase, [4](#)
 - setInitialPosition, [4](#)
 - setState, [5](#)
- brain::NextMove, [8](#)
- CameraData, [5](#)
- CCrazyflieController, [6](#)
 - Init, [6](#)
- Chn< T >, [7](#)
- computeNextMove
 - brain::Brain, [3](#)
- encode
 - RTStatus, [10](#)
- FlowDeck, [7](#)
 - getInitPositionDelta, [7](#)
 - init, [8](#)
- getInitPositionDelta
 - FlowDeck, [7](#)
- getState
 - brain::Brain, [4](#)
- Init
 - CCrazyflieController, [6](#)
- init
 - FlowDeck, [8](#)
- isReturningToBase
 - brain::Brain, [4](#)
- MathUtils, [8](#)
- Proxy, [9](#)
- RTStatus, [9](#)
 - encode, [10](#)
 - RTStatus, [9](#)
 - update, [10](#)
- SensorData, [10](#)
- setInitialPosition
 - brain::Brain, [4](#)
- setState
 - brain::Brain, [5](#)
- SharedQueue< T >, [11](#)
- update
 - RTStatus, [10](#)
- Vec3, [11](#)
- Vec4, [12](#)