crazyflie-project/server

Generated on Wed Apr 21 2021 10:43:35 for crazyflie-project/server by Doxygen 1.9.2

Wed Apr 21 2021 10:43:35

1 Server

This repository stores the code of the master-server, which links the dashboard to the ARGoS3 drones or the real drones.

1.1 Getting started

Please follow steps below to launch the server.

```
# Install python env
sudo apt-get install python3-venv uwsgi-plugin-python3
# Create environment
python3 -m venv .venv
# Activate the environment
. ./.venv/bin/activate
# Update pip just in case
pip install -U pip
# Install wheel package
pip install wheel
# Install dependencies
pip install -r requirements.txt
# Start app in dev mode
python src/server.py
# Start app in production mode
# uwsgi --ini=wsgi.ini
```

1.2 Others scripts

```
# Deactivate environment
deactivate
# Extract dependencies
>requirements.txt pip freeze
# Update dependencies
mv requirements.txt requirements.txt.orig
<requirements.txt.orig >requirements.txt sed 's/==/>=/g'
rm requirements.txt.orig
pip install -U -r requirements.txt
>requirements.txt pip freeze
```

1.3 Docker

```
# build
docker build -t crazyflie-server .
# run
docker run -it --name crazyflie-server -p 3995:3995 -p 5000:5000 crazyflie-server
```

1.4 Documentation generation

This project comes with an automatically generated documentation 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
  ```
- Once in a while, run Doxygen. Make sure you are in the root directory of the project (`/server`)
  ```
  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 :
```
def exitHandler(signal, frame):
    """Function description
        @param signal: signal parameter description
        @param frame: frame parameter description
    """
    import logging
    logging.info('CLOSING SERVER APPLICATION')
    pass
```

2 Namespace Documentation

2.1 drones_set Namespace Reference

Classes

- class DroneSearchReturn
- class DronesSet

2.1.1 Detailed Description

```
dronesSet
robot-handler.py
Static class to manage robots and their states
```

3 Class Documentation

3.1 argos_client.ArgosClient Class Reference

Public Member Functions

- None **__init__** (self)
- None connect (self, WebSocket webSocket)
- def handleCommunications (self)
- def closeClient (self)

Public Attributes

- **socket**
- **connection**

3.1.1 Member Function Documentation

3.1.1.1 closeClient() `def argos_client.ArgosClient.closeClient (`

```
            self )
```

```
Force close the connection. Called by the sigint handler.
```

3.1.1.2 connect() `None argos_client.ArgosClient.connect (`

```
            self,
           WebSocket webSocket )
```

```
Assign the client to the specified websocket and start a thread to
handle the communication.

  @param webSocket: the socket on witch the client is connected.
```

3.1.1.3 handleCommunications() `def argos_client.ArgosClient.handleCommunications (`

```
            self )
```

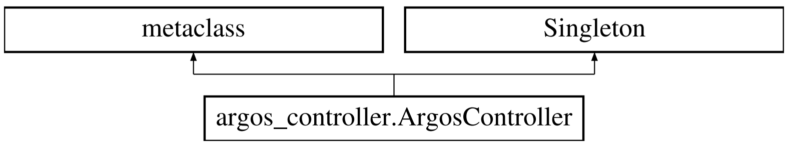
```
Listen on the socket for messages. It closes the client if it
receive an empty string.
```

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

- src/clients/argos_client.py

3.2 argos_controller.ArgosController Class Reference

Inheritance diagram for argos_controller.ArgosController:



Static Public Member Functions

- Thread launch ()
- def launchServer ()
- def stopServer ()
- def handleClient (clientSocket, addr)
- None onClientConnect (ArgosClient client)
- None onClientDisconnect (ArgosClient client)
- None onClientReceivedMessage (ArgosClient client, bytes message)
- None onClientRaisedError (ArgosClient client, Exception error)
- def onControllerReceivedMessage (Message message)
- None sendMessage (Message message)
- def sendMessageToSocket (clientSocket, Message message)
- def startMission (dict initialDronePos, dict offsetDronePos)
- Optional[Mission] **getCurrentMission** ()

Static Public Attributes

- string **TCP_HOST** = '0.0.0.0'
- int **TCP_PORT** = 3995
- int **BUFFER_SIZE** = 20
- int **N_MAX_DRONES** = 10
- **socket**
- bool **running** = True
- **dronesSet** = DronesSet()
- **MissionHandler**
- **ArgosClient**

3.2.1 Member Function Documentation

3.2.1.1 handleClient()

```
def argos_controller.ArgosController.handleClient (
              clientSocket,
              addr )  [static]
```

```
Creates a client witch will listen to the new connection.
Callbacks handlers are set for every event.

  @param clientSocket: the socket of the new connection.
```

3.2.1.2 launch()

```
Thread argos_controller.ArgosController.launch ( )  [static]
```

```
Launches a thread in witch the TCP server will start, and return
that thread.
```

3.2.1.3 launchServer() `def argos_controller.ArgosController.launchServer ( ) [static]`

```
Start the TCP server in non-blocking mode. Tries to accept new
connection as long as the server is running.
```

3.2.1.4 onClientConnect() `None argos_controller.ArgosController.onClientConnect ( ArgosClient client ) [static]`

```
Called by a client when it connects to its socket.

  @param client: the client witch called the function.
```

3.2.1.5 onClientDisconnect() `None argos_controller.ArgosController.onClientDisconnect ( ArgosClient client ) [static]`

```
Called by a client when it disconnects from its socket. It send a
message to the dashboards to inform the disconnection only if the
server is running.

  @param client: the client witch called the function.
```

3.2.1.6 onClientRaisedError() `None argos_controller.ArgosController.onClientRaisedError ( ArgosClient client, Exception error ) [static]`

```
Called when the client raises an error while it waits for messages.

  @param error: the exception  raised by the client.
```

3.2.1.7 onClientReceivedMessage() `None argos_controller.ArgosController.onClientReceived↩ Message ( ArgosClient client, bytes message ) [static]`

```
Called by the client when it receives a message. The message is
first decoded from byte to ascii, then parsed as json. It updates the
drone stored status, then sends the message to the dashboards and
the mission handler if it exists.

  @param message: the message in bytes received by the client.
```

3.2.1.8 onControllerReceivedMessage() `def argos_controller.ArgosController.onController↩ ReceivedMessage ( Message message ) [static]`

```
Decide what to do with the given message. It can start a mission
or send the message as is to the clients.

  @param message: the message received by the controller.
```

3.2.1.9 sendMessage() `None argos_controller.ArgosController.sendMessage ( Message message ) [static]`

```
Sends the specified message to the client.

  @param message: the message to send.
```

3.2.1.10 sendMessageToSocket() `def argos_controller.ArgosController.sendMessageToSocket ( clientSocket, Message message ) [static]`

```
Sends the specified message to the specified socket after
converting it from json to string.

  @param clientSocket: the socket to send the message.
  @param message: the message to send.
```

3.2.1.11 startMission() `def argos_controller.ArgosController.startMission ( dict initialDronePos, dict offsetDronePos ) [static]`

```
Start a mission. Order drones to takeoff and initialize a mission
handler.

  @param initialDronePos: the drone position at the moment of the mission creation.
  @param offsetDronePos: the drone position offset given by the dashboard.
```

3.2.1.12 stopServer() `def argos_controller.ArgosController.stopServer ( ) [static]`

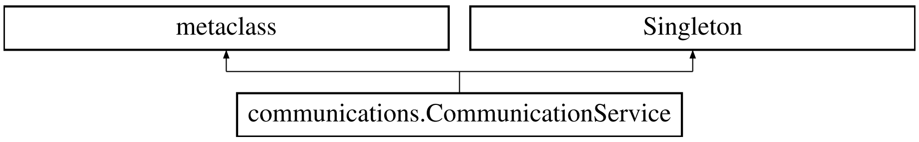
```
Closes all the clients connections, then closes the server.
```

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

- src/controllers/argos_controller.py

3.3 communications.CommunicationService Class Reference

Inheritance diagram for communications.CommunicationService:



Public Member Functions

- def __init__ (self)
- None registerControllers (self, dashboard, argos, crazyradio)
- def sendToDashboardController (self, Message message)
- def sendToArgosController (self, Message message)
- def sendToCrazyradioController (self, Message message)
- List[Drone] getAllDrones (self)
- Optional[Mission] getCurrentMission (self)

Public Attributes

- **dashboardController**
- **argosController**
- **crazyradioController**

3.3.1 Constructor & Destructor Documentation

3.3.1.1 __init__() `def communications.CommunicationService.__init__ ( self )`

```
Initialize the communication service. The controllers are set to
None because their object do not exists yet.
```

3.3.2 Member Function Documentation

3.3.2.1 getAllDrones() `List[Drone] communications.CommunicationService.getAllDrones ( self )`

```
Get all the drone saved across the argos and crazyradio controller.
```

3.3.2.2 getCurrentMission() `Optional[Mission] communications.CommunicationService.getCurrent↩ Mission ( self )`

```
Returns the current active mission.
```

3.3.2.3 registerControllers() `None communications.CommunicationService.registerControllers ( self, dashboard, argos, crazyradio )`

```
Register the controller to the specified values.

  @param dashboard: the object of the dashboard controller.
  @param argos: the object of the argos controller.
  @param crazyradio: the object of the crazyradio controller.
```

3.3.2.4 sendToArgosController() `def communications.CommunicationService.sendToArgosController ( self, Message message )`

```
Send the message to the argos controller.
  @param message: the message to send.
```

3.3.2.5 sendToCrazyradioController() `def communications.CommunicationService.sendToCrazyradio↩ Controller ( self, Message message )`

```
Send the message to the crazyradio controller.
  @param message: the message to send.
```

3.3.2.6 sendToDashboardController() `def communications.CommunicationService.sendToDashboard↩ Controller ( self, Message message )`

```
Send the message to the dashboard controller.
  @param message: the message to send.
```

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

- src/services/communications.py

3.4 connection.Connection Class Reference

Public Member Functions

- None **__init__** (self)
- None addCallback (self, HandlerType handlerType, Callable callback, ∗args)
- None removeCallback (self, HandlerType handlerType, Callable callback)
- None callAllCallbacks (self, HandlerType handlerType, ∗args)

Public Attributes

- **handlers**

3.4.1 Member Function Documentation

3.4.1.1 addCallback()

```
None connection.Connection.addCallback (
        self,
        HandlerType handlerType,
        Callable callback,
        ∗ args )
```

```
 Add a new handler for the given callback.

  @param handlerType: the type of the handler.
  @param callback: the callback function.
  @param args: the params to give the callback fucntion.
```

3.4.1.2 callAllCallbacks()

```
None connection.Connection.callAllCallbacks (
        self,
        HandlerType handlerType,
        ∗ args )
```

```
Call all the registered callbacks of the same type of handler.

  @param handlerType: the type of handler to call the callbacks.
  @param args: the arguments to give to the callback.
```

3.4.1.3 removeCallback()

```
None connection.Connection.removeCallback (
        self,
        HandlerType handlerType,
        Callable callback )
```

```
Removes the given callback.

  @param handlerType: the type of the handler
  @param callback:
```

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

- src/models/connection.py

3.5 crazyradio_client.CrazyradioClient Class Reference

Public Member Functions

- None **__init__** (self)
- None connect (self, str droneUri)
- None sendMessage (self, Message message)
- None closeClient (self)

Public Attributes

- **uri**
- **connection**
- **queue**

3.5.1 Member Function Documentation

3.5.1.1 closeClient()

```
None crazyradio_client.CrazyradioClient.closeClient (
        self )
```

```
Force close the client connection. Called by the sigint handler
```

3.5.1.2 connect()

```
None crazyradio_client.CrazyradioClient.connect (
        self,
        str droneUri )
```

```
Assign the client to the connection. Add callbacks for the
different events.

  @param droneUri: the drone’s identifier.
```

3.5.1.3 sendMessage()

```
None crazyradio_client.CrazyradioClient.sendMessage (
        self,
        Message message )
```

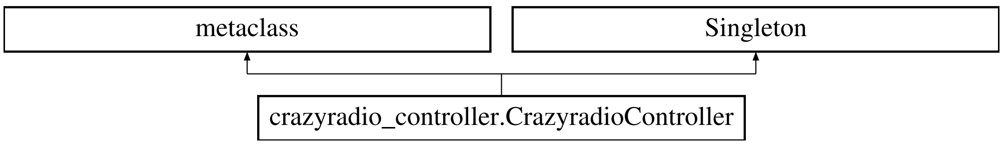
```
Take given message string and sends it as bytes to the appchannel.

  @param message: the message to send
```

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

- src/clients/crazyradio_client.py

3.6 crazyradio_controller.CrazyradioController Class Reference

Inheritance diagram for crazyradio_controller.CrazyradioController:



Static Public Member Functions

- Thread launch ()
- def launchServer ()
- def findNewDrones ()
- bool isDongleConnected ()
- List getAvailableInterfaces ()
- def stopServer ()
- def handleClient (interface)
- None onClientConnect (CrazyradioClient client)
- None onClientDisconnect (CrazyradioClient client)
- None onClientReceivedMessage (CrazyradioClient client, data)
- None onClientRaisedError (CrazyradioClient client, Exception error)
- def onControllerReceivedMessage (Message message)
- None sendMessage (Message message)
- def sendMessageToClient (CrazyradioClient client, Message message)
- Union[Drone, Any] getDroneIdentifier (str uri)
- def startMission (dict initialDronePos, dict offsetDronePos)
- def stopMission ()
- def loadProject (LoadProjectData loadProjectData)
- def loadProjectThread (ProjectType projectType, code=None)
- Optional[Mission] **getCurrentMission** ()

Static Public Attributes

- bool **running** = True
- **dronesSet** = DronesSet()
- **MissionHandler**
- bool **projectCurrentlyLoading** = False
- int **FIRST_DRONE_ADDRESS** = 0xE7E7E7E701
- int **MAX_DRONE_NUMBER** = 2

3.6.1 Member Function Documentation

3.6.1.1 findNewDrones() `def crazyradio_controller.CrazyradioController.findNewDrones ( ) [static]`

```
Try to reconnect to new drones every 5 seconds.
```

3.6.1.2 getAvailableInterfaces() `List crazyradio_controller.CrazyradioController.getAvailable←↩ Interfaces ( ) [static]`

```
Scans for available drone connections and adds them to the list of clients.
```

3.6.1.3 getDroneIdentifier() `Union[Drone, Any] crazyradio_controller.CrazyradioController.get←↩ DroneIdentifier (`
`str uri ) [static]`

```
Find the drone associated with the specified uri. If the drone
name isn't yet registered, the uri is returned.

  @param uri: the uri of the searched drone.
```

3.6.1.4 handleClient() `def crazyradio_controller.CrazyradioController.handleClient (`
`interface ) [static]`

```
Creates a client witch will listen to the new connection.
Callbacks handlers are set for every event.

  @param interface: the interface of the new connection.
```

3.6.1.5 isDongleConnected() `bool crazyradio_controller.CrazyradioController.isDongleConnected ( ) [static]`

```
Check if the dongle is connected.
```

3.6.1.6 launch() `Thread crazyradio_controller.CrazyradioController.launch ( ) [static]`

```
Launches a thread in witch the crazyradio controller will start,
and return that thread.
```

3.6.1.7 launchServer() `def crazyradio_controller.CrazyradioController.launchServer ( ) [static]`

```
Start the server. Scans for the dongle and wait until its is
connected. Scans for interface (drones) then create a client for each
interface found.
```

3.6.1.8 loadProject() `def crazyradio_controller.CrazyradioController.loadProject ( LoadProjectData loadProjectData ) [static]`

```
 Starts a thread to load the given project.

  @param loadProjectData: the data of the project to load
```

3.6.1.9 loadProjectThread() `def crazyradio_controller.CrazyradioController.loadProjectThread ( ProjectType projectType, code = None ) [static]`

```
Stops the drone connection to flash it.

  @param projectType:
  @param code:
```

3.6.1.10 onClientConnect() `None crazyradio_controller.CrazyradioController.onClientConnect ( CrazyradioClient client ) [static]`

```
Called by a client when it connects to its interface.

  @param client: the client witch called the function.
```

3.6.1.11 onClientDisconnect() `None crazyradio_controller.CrazyradioController.onClientDisconnect ( CrazyradioClient client ) [static]`

```
Called by a client when it disconnects from its interface. It send
a message to the dashboards to inform the disconnection only if the
server is running.

  @param client: the client witch called the function.
```

3.6.1.12 onClientRaisedError() `None crazyradio_controller.CrazyradioController.onClientRaised↩ Error ( CrazyradioClient client, Exception error ) [static]`

```
Called when a client raises an error while it waits for messages.

  @param client: the client witch called the function.
  @param error: the exception  raised by the client.
```

3.6.1.13 onClientReceivedMessage() `None crazyradio_controller.CrazyradioController.onClient←↩ ReceivedMessage (`

```
            CrazyradioClient client,
             data )  [static]
```

```
Called by a client when it receives a message. The message is first decoded then
it updates the drone stored status. Finnally it sends the message
to the dashboards and the mission controller.

  @param client: the client witch called the function.
  @param data: the data of the message in bytes received by the client.
```

3.6.1.14 onControllerReceivedMessage() `def crazyradio_controller.CrazyradioController.on←↩ ControllerReceivedMessage (`

```
            Message message )  [static]
```

```
Decide what to do with the given message. It can start a mission
or send the message as is to the clients.

  @param message: the message received.
```

3.6.1.15 sendMessage() `None crazyradio_controller.CrazyradioController.sendMessage (`

```
            Message message )  [static]
```

```
Sends the specified message to the correct drone. doesn't sends
anything if the requested drone doesn't exist.

  @param message: the message to send.
```

3.6.1.16 sendMessageToClient() `def crazyradio_controller.CrazyradioController.sendMessageTo←↩ Client (`

```
            CrazyradioClient client,
            Message message )  [static]
```

```
Sends the specified message to the specified client.

  @param client: the client to send the message.
  @param message: the message to send.
```

3.6.1.17 startMission() `def crazyradio_controller.CrazyradioController.startMission ( dict initialDronePos, dict offsetDronePos ) [static]`

```
Start a mission. Order drones to takeoff and initialize a mission
handler.

  @param initialDronePos: the drone position at the moment of the mission creation.
  @param offsetDronePos: the drone position offset given by the dashboard.
```

3.6.1.18 stopMission() `def crazyradio_controller.CrazyradioController.stopMission ( ) [static]`

```
Stops the current mission and send to the client th emessage to do so.
```

3.6.1.19 stopServer() `def crazyradio_controller.CrazyradioController.stopServer ( ) [static]`

```
Close the controller by closing every client.
```

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

- src/controllers/crazyradio_controller.py

3.7 dashboard_client.DashboardClient Class Reference

Public Member Functions

- None **__init__** (self)
- None connect (self, socket)
- None handleCommunications (self)
- def closeClient (self)

Public Attributes

- **thread**
- **connection**

3.7.1 Member Function Documentation

3.7.1.1 closeClient() `def dashboard_client.DashboardClient.closeClient ( self )`

```
Force close the connection. Called by the sigint handler.
```

3.7.1.2 connect() `None dashboard_client.DashboardClient.connect ( self, socket )`

```
Assigning the client to the specified socket and start a thread to
handle the communication.

  @param socket: the socket on witch the client is connected.
```

3.7.1.3 handleCommunications() `None dashboard_client.DashboardClient.handleCommunications ( self )`

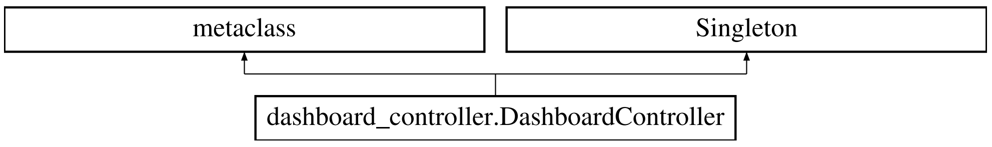
```
Listen for message on the socket while the connection is active.
```

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

- src/clients/dashboard_client.py

3.8 dashboard_controller.DashboardController Class Reference

Inheritance diagram for dashboard_controller.DashboardController:



Static Public Member Functions

- Thread launch ()
- def launchServer ()
- def stopServer ()
- def handleClient (WebSocket webSocket)
- None onClientConnect (DashboardClient client)
- None onClientDisconnect (DashboardClient client)
- None onClientReceivedMessage (DashboardClient client, str message)
- None onClientRaisedError (DashboardClient client, Exception error)
- None sendAllRobotsStatus (socket)
- None sendAllMissions (socket)
- def onControllerReceivedMessage (Message message)
- None sendMessageToSocket (socket, Message message)

Static Public Attributes

- **app** = Flask(__name__)
- **sockets** = Sockets(app)
- string **HOST** = '0.0.0.0'
- int **SERVER_PORT** = 5000
- **webSocketServer** = None

3.8.1 Member Function Documentation

3.8.1.1 handleClient() `def dashboard_controller.DashboardController.handleClient ( WebSocket webSocket ) [static]`

```
Creates a client witch will listen to the new connection.
Callbacks handlers are set for every event.

  @param webSocket: the socket of the new connection.
```

3.8.1.2 launch() `Thread dashboard_controller.DashboardController.launch ( ) [static]`

```
Launches a thread in witch the webSocket server will start,
and return that thread.
```

3.8.1.3 launchServer() `def dashboard_controller.DashboardController.launchServer ( ) [static]`

```
Start the websocket server.
```

3.8.1.4 onClientConnect() `None dashboard_controller.DashboardController.onClientConnect ( DashboardClient client ) [static]`

```
Called by a client when it connects to its socket.

  @param client: the client witch called the function.
```

3.8.1.5 onClientDisconnect() `None dashboard_controller.DashboardController.onClientDisconnect ( DashboardClient client ) [static]`

```
Called by a client when it disconnects from its socket.

  @param client: the client witch called the function.
```

3.8.1.6 onClientRaisedError() `None dashboard_controller.DashboardController.onClientRaisedError ( DashboardClient client, Exception error ) [static]`

```
Called when a client raises an error while it waits for messages.

  @param client: the client witch called the function.
  @param error: the exception  raised by the client.
```

3.8.1.7 onClientReceivedMessage() `None dashboard_controller.DashboardController.onClient←ReceivedMessage ( DashboardClient client, str message ) [static]`

```
Called by a client when it receives a message. The message is
parsed as json, then sent to all aof the drones (physical and
simulated).

  @param client: the client witch called the function.
  @param message: the message received by the client.
```

3.8.1.8 onControllerReceivedMessage() `def dashboard_controller.DashboardController.onController←ReceivedMessage ( Message message ) [static]`

```
Sends the given messages to all the clients.

  @param message: the message received by the controller.
```

3.8.1.9 sendAllMissions() `None dashboard_controller.DashboardController.sendAllMissions ( socket ) [static]`

```
Send all the saved mission in the database to the socket.

  @param socket: the socket to send the info to.
```

3.8.1.10 sendAllRobotsStatus() `None dashboard_controller.DashboardController.sendAllRobots←Status ( socket ) [static]`

```
Send all the information of every saved drones to the socket.

  @param socket: the socket to send the info to.
```

3.8.1.11 sendMessageToSocket() `None dashboard_controller.DashboardController.sendMessageTo←Socket ( socket, Message message ) [static]`

```
Sends the specified message to the specified socket after
converting it from json to string.

  @param socket: the socket to send the message.
  @param message: the message to send.
```

3.8.1.12 stopServer() `def dashboard_controller.DashboardController.stopServer ( ) [static]`

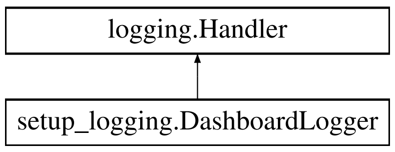
```
Closes all the clients connections, then closes the server.
```

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

- src/controllers/dashboard_controller.py

3.9 setup_logging.DashboardLogger Class Reference

Inheritance diagram for setup_logging.DashboardLogger:



Public Member Functions

- None **emit** (self, logging.LogRecord record)

Static Public Member Functions

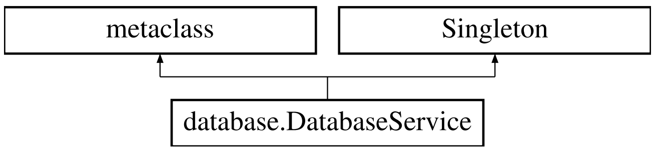
- LogType **getDashBoardLevelName** (logging.LogRecord record)

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

- src/utils/setup_logging.py

3.10 database.DatabaseService Class Reference

Inheritance diagram for database.DatabaseService:



Static Public Member Functions

- List[Mission] getAllMissions ()
- def saveMission (str missionId, Mission mission)

Static Public Attributes

- string **MISSIONS_TABLE_NAME** = 'missions'
- **db** = TinyDB('data/db.json')

3.10.1 Member Function Documentation

3.10.1.1 getAllMissions() `List[Mission] database.DatabaseService.getAllMissions ( ) [static]`

```
Return a list of saved mission, sorted by their timestamp.
```

3.10.1.2 saveMission() `def database.DatabaseService.saveMission (`

```
str missionId,
Mission mission ) [static]
```

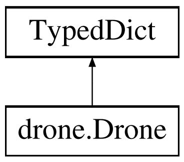
```
Save the specified mission into the database. Update the value of the mission id if it already exists.

  @param missionId: the str id of the mission to save/update
  @param mission: the data of the mission to save
```

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

- src/services/database.py

3.11 drone.Drone Class Reference

Inheritance diagram for drone.Drone:

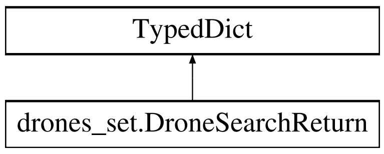


The documentation for this class was generated from the following file:

- src/models/drone.py

3.12 drones_set.DroneSearchReturn Class Reference

Inheritance diagram for drones_set.DroneSearchReturn:



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

- src/services/drones_set.py

3.13 drones_set.DronesSet Class Reference

Public Member Functions

- None **__init__** (self)
- dict getDrones (self)
- Union[None, Drone] getDrone (self, Any key)
- Union[None, DroneSearchReturn] findDroneByName (self, str name)
- None setDrone (self, Any key, Drone drone)
- None removeDrone (self, Any key)

3.13.1 Member Function Documentation

3.13.1.1 findDroneByName() `Union[None, DroneSearchReturn] drones_set.DronesSet.findDroneBy↩ Name ( self, str name )`

```
Search the saved drone with the given name. Return None if it
doesn't find any match.

  @param name: the str name of the searched drone.
```

3.13.1.2 getDrone() `Union[None, Drone] drones_set.DronesSet.getDrone ( self, Any key )`

```
Return the drone identified by the given key.
  @param key: the key witch identifies the drone.
```

3.13.1.3 getDrones() `dict drones_set.DronesSet.getDrones ( self )`

```
Return a deep copy of the drones.
```

3.13.1.4 removeDrone() `None drones_set.DronesSet.removeDrone ( self, Any key )`

```
Revome the drone associatede with the given key.
  @param key: the key that identifies the drone to remove.
```

3.13.1.5 setDrone() `None drones_set.DronesSet.setDrone ( self, Any key, Drone drone )`

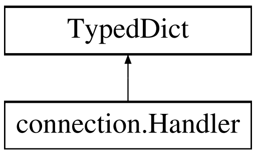
```
Add the given drone to the saved drones.

@param key: the key witch identifies the drone.
@param drone: the drone data to save.
```

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

- src/services/drones_set.py

3.14 connection.Handler Class Reference

Inheritance diagram for connection.Handler:

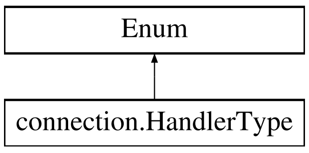


The documentation for this class was generated from the following file:

- src/models/connection.py

3.15 connection.HandlerType Class Reference

Inheritance diagram for connection.HandlerType:



Static Public Attributes

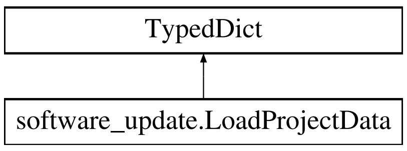
- string **connection** = 'connection'
- string **disconnection** = 'disconnection'
- string **message** = 'message'
- string **error** = 'error'

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

- src/models/connection.py

3.16 software_update.LoadProjectData Class Reference

Inheritance diagram for software_update.LoadProjectData:

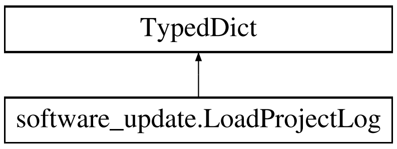


The documentation for this class was generated from the following file:

- src/models/software_update.py

3.17 software_update.LoadProjectLog Class Reference

Inheritance diagram for software_update.LoadProjectLog:

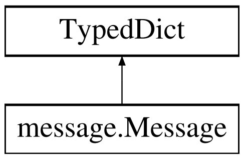


The documentation for this class was generated from the following file:

- src/models/software_update.py

3.18 message.Message Class Reference

Inheritance diagram for message.Message:

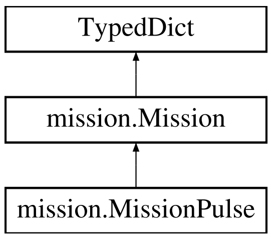


The documentation for this class was generated from the following file:

- src/models/message.py

3.19 mission.Mission Class Reference

Inheritance diagram for mission.Mission:



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

- src/models/mission.py

3.20 mission_handler.MissionHandler Class Reference

Public Member Functions

- def __init__ (self, DronesSet dronesSet, MissionType missionType, dict initialDronePos, dict offsetDronePos, Callable[[Message], None] sendMessageCallable)
- def onReceivedPositionAndRange (self, str droneName, Vec2 position, float yaw, List[int] ranges)
- bool checkPointValidity (self, Tuple[float, float] point)
- def handlePositionAndBorders (self, str droneName, Vec2 position, List[Vec2] points)
- def assignPointsToShapes (self)
- def recursiveAddPointToShape (self, List[MissionPoint] missionPoints, List[MissionPoint] pointsToAdd, List[Vec2] currentShape)
- bool checkMissionEnd (self)
- def endMission (self)
- def stopMission (self)

Public Attributes

- **maxRange**
- **initialDronePos**
- **offsetDronePos**
- **dronesSet**
- **sendMessageCallable**
- **mission**
- **kdtree**

Static Public Attributes

- float **MAX_DISTANCE** = 0.21
- float **MIN_POINTS_DIST** = 0.002
- int **TAKE_OFF_DELAY** = 20
- int **CRAZYRADIO_MAX_RANGE** = 500
- int **ARGOS_MAX_RANGE** = 65530
- float **CRAZYRADIO_SCALE** = 0.001
- float **ARGOS_SCALE** = 0.01

3.20.1 Constructor & Destructor Documentation

3.20.1.1 __init__()

```
def mission_handler.MissionHandler.__init__ (
     self,
    DronesSet dronesSet,
    MissionType missionType,
    dict initialDronePos,
    dict offsetDronePos,
    Callable[[Message], None] sendMessageCallable )
```

```
Initialize the mission handler. Reject the mission if the droneSet
is empty. Gives random colors to the drones ins the droneSet. Save
the newly created mission object and saves it in the database.

  @param dronesSet: the set of drones participating in the mission.
  @param missionType: The type of the mission: real or fake. Fake is
  for demo purposes only. @param sendMessageCallable: the function to
  call to send mission pulses.
```

3.20.2 Member Function Documentation

3.20.2.1 assignPointsToShapes() `def mission_handler.MissionHandler.assignPointsToShapes (`

```
            self )
```

```
Goes over all the point found during the mission and try to
regroup them into shapes. Then add the created shape to the current
mission.
```

3.20.2.2 checkMissionEnd() `bool mission_handler.MissionHandler.checkMissionEnd (`

```
            self )
```

```
Check if every drone is landed to end the mission.
```

3.20.2.3 checkPointValidity() `bool mission_handler.MissionHandler.checkPointValidity (`

```
            self,
           Tuple[float, float] point )
```

```
Check if the given point isn't too close to an already existing point.

 param point: the point to check.
```

3.20.2.4 endMission() `def mission_handler.MissionHandler.endMission (`

```
            self )
```

```
End the mission. Save the 'done' status to the database and inform
the dashboards.
```

3.20.2.5 handlePositionAndBorders() `def mission_handler.MissionHandler.handlePositionAndBorders (`

```
            self,
           str droneName,
           Vec2 position,
           List[Vec2] points )
```

```
Add the new position of the drone as well as the position of the border it found to the mission.
Saves the updated mission to the database.
  @param droneName: the name of the drone witch sent the informations.
  @param position: the 2D position of the drone.
  @param points: a list of 2D points
```

3.20.2.6 onReceivedPositionAndRange()

```
def mission_handler.MissionHandler.onReceivedPosition↩
AndRange (
              self,
             str droneName,
             Vec2 position,
             float yaw,
             List[int] ranges )
```

```
Calculate the point indicated by the given ranges and orientation.
 Translate to coordinates from the received axis to the dashboard axis.

  @param droneName: the name of the drone witch sent the informations.
  @param position: the 2D position of the drone.
  @param yaw: the angle of the drone in radiant.
  @param ranges: the list of ranges (front, left, back, right)
```

3.20.2.7 recursiveAddPointToShape()

```
def mission_handler.MissionHandler.recursiveAddPointTo↩
Shape (
              self,
             List[MissionPoint] missionPoints,
             List[MissionPoint] pointsToAdd,
             List[Vec2] currentShape )
```

```
Add the points to the current shape, then find the nearest point
from the given points, sort them by distance, and call this method
with the newly found points.

  @param missionPoints: the points witch are not in a shape yet.
  @param currentShape: the current shape in witch we add points
  @param pointsToAdd: a list of the points found in the previous
    iteration.
  @param currentShape: the current shape in witch the
    pointsToAdd will be added.
```

3.20.2.8 stopMission()

```
def mission_handler.MissionHandler.stopMission (
              self )
```

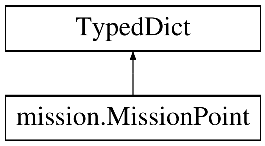
```
Force stop the mission and register it as failed.
```

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

- src/services/mission_handler.py

3.21 mission.MissionPoint Class Reference

Inheritance diagram for mission.MissionPoint:

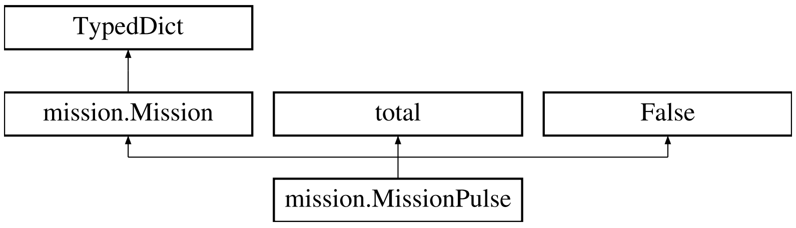


The documentation for this class was generated from the following file:

- src/models/mission.py

3.22 mission.MissionPulse Class Reference

Inheritance diagram for mission.MissionPulse:

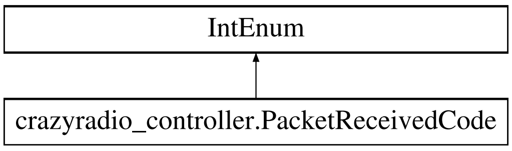


The documentation for this class was generated from the following file:

- src/models/mission.py

3.23 crazyradio_controller.PacketReceivedCode Class Reference

Inheritance diagram for crazyradio_controller.PacketReceivedCode:



Static Public Attributes

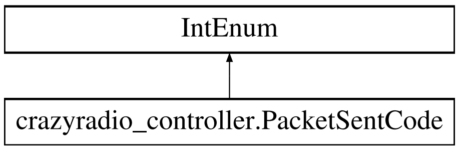
- int **BATTERY** = 0
- int **SPEED** = 1
- int **POSITION_AND_SENSORS** = 2
- int **OTHERS** = 3

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

- src/controllers/crazyradio_controller.py

3.24 crazyradio_controller.PacketSentCode Class Reference

Inheritance diagram for crazyradio_controller.PacketSentCode:



Static Public Attributes

- int **START_MISSION** = 0
- int **END_MISSION** = 1
- int **RETURN_TO_BASE** = 2
- int **TAKE_OFF** = 3
- int **LANDING** = 4
- int **LED_ON** = 5
- int **LED_OFF** = 6

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

- src/controllers/crazyradio_controller.py

3.25 project_loader.ProjectLoader Class Reference

Public Member Functions

- def **__init__** (self, pathlib.Path cf2_bin=cwd/ 'out'/ 'cf2.bin')
- bool setup (self, ProjectType projectType, Optional[str] code)
- def flash (self, List[str] clinks)

Static Public Member Functions

- def createContainer (ProjectType projectType)
- def getContainer (ProjectType projectType)

Public Attributes

- **bin**

Static Public Attributes

- **client** = docker.from_env()
- **cwd** = pathlib.Path.cwd()
- string **firmwarePath** = cwd / '..' / 'firmware'
- string **outPath** = cwd / 'out'
- dictionary **volumes**
- string **containerImage** = 'firmware_image'
- string **containerName** = 'firmware_cload'
- list **targets** = [cflib.bootloader.Target('cf2', 'stm32', 'fw')]
- string **sandboxSrc** = firmwarePath / 'projects' / 'sandbox' / 'src' / 'main.cpp'

3.25.1 Member Function Documentation

3.25.1.1 createContainer() `def project_loader.ProjectLoader.createContainer ( ProjectType` *projectType* `) [static]`

```
Create a docker container for the given project type.

  @param projectType: the type of the project to create a container for.
```

3.25.1.2 flash() `def project_loader.ProjectLoader.flash (` *self,* `List[str]` *clinks* `)`

```
Try to flash the drone with the compiled code.

  @param clinks: the compilated code.
```

3.25.1.3 getContainer() `def project_loader.ProjectLoader.getContainer ( ProjectType` *projectType* `) [static]`

```
Returns the container with the same given project type, or create a new one.

  @param projectType: the project type of the container searched.
```

3.25.1.4 setup() `bool project_loader.ProjectLoader.setup (` *self,* `ProjectType` *projectType,* `Optional[str]` *code* `)`

```
Run and try to build the code given in a container for the given project type.

  @param projectType: the type of the project to execute.
  @param code:the code to compile.
```

3.25.2 Member Data Documentation

3.25.2.1 volumes `dictionary project_loader.ProjectLoader.volumes [static]`

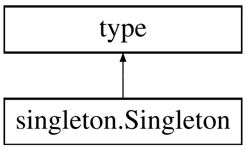
Initial value:
```
= {
        # str(cwd / '..' / '.git'): {
        #    'bind': '/.git',
        #    'mode': 'ro',
        # },
        # str(firmwarePath): {
        #    'bind': '/firmware',
        #    'mode': 'ro',
        # },
        # str(outPath): {
        #    'bind': '/out',
        #    'mode': 'rw',
        # },
    }
```

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

- src/services/project_loader.py

3.26 singleton.Singleton Class Reference

Inheritance diagram for singleton.Singleton:



Public Member Functions

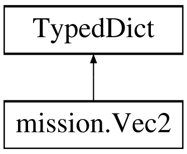
- def **__call__** (cls, ∗args, ∗∗kwargs)

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

- src/metaclasses/singleton.py

3.27 mission.Vec2 Class Reference

Inheritance diagram for mission.Vec2:



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

- src/models/mission.py

Index