Intelligent Fish feeding through Integration of ENabling technologies and Circular principle

Grant Agreement (GA) No: 818036

Report on final Smart feeding AI

Version : 2

Date: 31/10/2022

Document type:	**Report**
Dissemination level:	**Public**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 818036

Project data

Project Title:	Intelligent Fish feeding through Integration of ENabling technologies and Circular principle
Project Grant Agreement (GA) No:	818036
Project Acronym:	iFishIENCi
Duration:	48 months, 1 November 2018 – 31 October 2022
Type of action:	Innovation Action

Deliverable Administration and Summary

Status:	Final	**Due:**	31/10/2021	**Date:**	
Author (s)	Christian Jensen (OXY), Franck Le Gall (EGM), Dimitri Trotignon (BIO), Ahmed Abid (EGM), Joseph A. De Prisco (ABT), Sergei Budaev (UiB)				
Reviewer	Dominique Durand, Elisa Ravagnan				
WP	2	**Deliverable Nr.**	**10**	**Relative Nr.**	2.4
Comments					

Document change history

Version	Date	Author	Description
0.1	04/08/2021		Creation
0.2	24/10/2021		Intermediate version, for review
1.0	28/10/2021		Intermediate version, Final
1.1	19/11/2021		Intermediate version, review fixes
2.0	31/10/2022		Final review

Disclaimer:

Suggested reference to this deliverable: Deliverable number, Deliverable title (Year of publication), Intelligent Fish feeding through Integration of ENabling technologies and Circular principle (iFishIENCi) Horizon 2020 project under Grant Agreement (GA) No: 818036

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 818036

Table of content

List of figures

List of tables

Abbreviations

ABT	AquaBiotech
ADMS	Automated decision-making system
AI	Artifical intelligence
DO	Dissolved oxygen
DT	Digital twin
FCR	Feed conversion rate
HCMR	Hellenic Centre for Marine Research
NN	Neural Network
PWM	Pulse-Width Modulating
TGP	Total gas pressure
CEFACT	The United Nations Centre for Trade Facilitation and Electronic Business

1 Summary

One of the goals of the iFishIENCi project is to bring to market the iFishIENCi Biology Online Steering System (iBOSS) that significantly improves production control and management for all fish aquaculture systems. Within iBOSS, smart functions can be implemented to take advantage of data from multiple sources to give better insight for the farmer, optimize production, better production planning, etc.

The **Report on final Smart feeding AI** aims to implement one such smart function: iBOSS Smart Feeding. By using an AI approach, it is expected appropriate correlations between fish behaviour, feeding regime and water quality can be found, and used for controlling feeder and achieve optimal feeding operation. While a lot of data is available, it has not yet been explored how to concatenate this into one model for optimizing feeding efficiency. Thus, an approach to integrate multiple sources of feeding efficiency quantification in to one, is proposed.

In the following sections, the current state of development is described.

Here, data collection (Task 1.4), new sensor implementation (Task 1.2 and 1.4) and live fish behaviour observations (Task 1.4 and WP3), contributes to developing Artificial Intelligence (AI) algorithms to optimize feed utilization. Furthermore, the testing and validation of the AI algorithms are described.

What is Smart Feeding? To understand the approaches described, a clearer definition of Smart Feeding is necessary. The iBOSS Smart Feeding aims at maximizing feed utilization while minimizing environmental impacts, by optimizing the efficiency of the fish exposure to feed in relation to fish state (e.g., satiety, stress), behaviour, environmental conditions and specific species characteristics, in order to maximize growth and reduce feed loss. Multiple AI models are being developed, each adding its own interpretation of the feeding efficiency. All interpretations are integrated in an automated decision-making system (ADMS) to give the feeder signal to either increase or decrease feeding. The changes by the ADMS can be reflected in quantity, frequency or both.

Experiments and data collection carried out at two pilot sites (ABT-Malta and HCMR-Crete pilot sites) have been mainly used to develop AI-based steering algorithm. The ABT site is a RAS system while the HCMR site has both an open cage setup and a RAS system. These systems were chosen because of their existing equipment and the complementarity of these two sites with both open and closed/RAS systems? At both sites a set of standard water quality sensors has been connected to the cloud, a camera system selected for the specific site has been installed and, at HCMR, an echo sounder has been mounted. Data from these systems will be available for the modelling of the AI algorithms. A detailed description of the HCMR data model and sensor deployment was given in the deliverable D2.3, and is therefore left out of this report.

2 RAS system at ABT pilot-site

2.1 Detailed description of the currently measured parameters at ABT

Below, a description of the ABT-Malta experimental site is presented. The system is primarily monitored by OxyGuard equipment. A OxyGuard Pacific is installed as a data collection and control unit along with several water quality probes. See Figure 1 for diagram of the water flow and probes installed.

Waterflow description (whole RAS diagram)

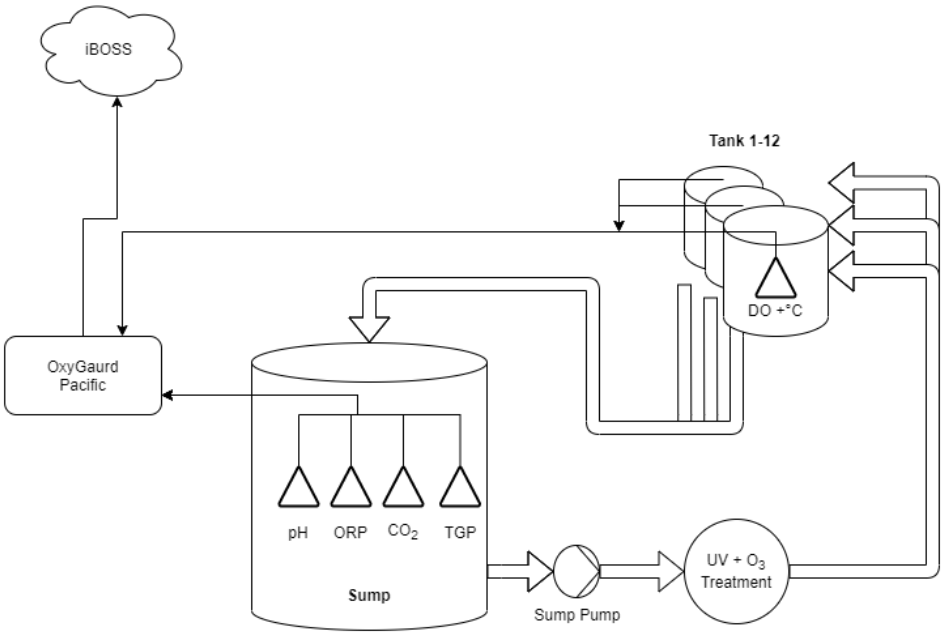


Figure 1: Diagram of sensor configuration in the experimental recirculating system

Automatically monitored parameters

Table 1: Parameters measured by sensor probes and automatically recorded

Measurement	Placement
Dissolved oxygen (mg/l or %saturation)	1 per tank
Temperature (°C)	1 per tank
Redox/ORP (mV)	1 in the sump
CO2 (mg/l or %saturation)	1 in the sump
TGP (% saturation)	1 in the sump

Online continuous measurements from probes, collected by the OxyGuard Pacific, are pushed to the iBOSS cloud every 15 minutes. The data from the probes (Figure 2) are currently used only to ensure that the parameters stay within user defined limits; cloud-based storage and interpretation of these

data will provide a new level of insight and understanding of how each parameter can affect other parameters, feeding efficiency and the systems general health.

In Figure XX, below, the PC visual monitoring program (named VIGO) is shown. It visualizes the entire facility's equipment and assists the operator/farmer in control and monitoring.

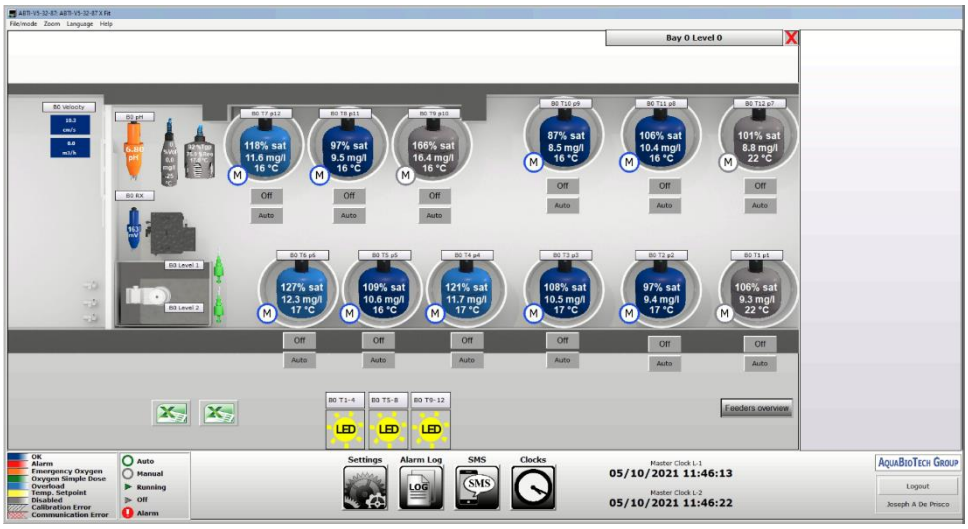


Figure 2: VIGO representation of probes and tanks in experimental system

Camera setup

High-definition cameras mounted above tanks (overhead) with live video feed continually accessible through a network interfaced HIKVision Network Video Recorder (NVR). The overhead observation of fish using video camera technology presents a unique set of challenges in terms of system optimization. Any specific aquaculture system deploying overhead camera technology possess unique characteristics requiring unique configurations in terms of camera and lighting, some of the parameters considered in iFishIENCi for both camera (Table 3) and stock containment (Table 2) are shown in the tables below:

Table 2: Exemplary parameters for fish containment used in iFishIENCi experimentation

Parameter	**Values**
Tank depth (mm)	650
Clarity of water (NTU)	<50
Stocking density (kg/m^3)	65
Mean mass of fish (g)	1000
Tank diameter (mm)	1500

Table 3: Exemplary parameters for camera setup used in iFishIENCi experimentation

Parameter	**Values**
Working distance (mm)	1200

Lens angle (°)	120
Imaging size (mm)	2
Camera resolution (MP)	5
Light intensity (lumens)	500 @ surface of containment
Lighting position	Directly above containment center
Relative angle of camera to center of containment surface (°)	45

Figure 3 shows a still image of rainbow trout captured using an IMENCO HD overhead camera mounted according to the specifications in Table 3 and observing a containment described by Table 2. The field of view is not sufficient to capture the entire containment, yet the image is clear and the resolution good. For the purposes of iFishIENCi, the camera systems were retro-fitted and trade-offs had to be found between the working distance of the camera, the lighting, and the relative position of the camera. It is suggested that, in systems where overhead cameras are installed, specific attention should be given to the objective of the observation and the requirements of the system in terms of camera and containment configuration.

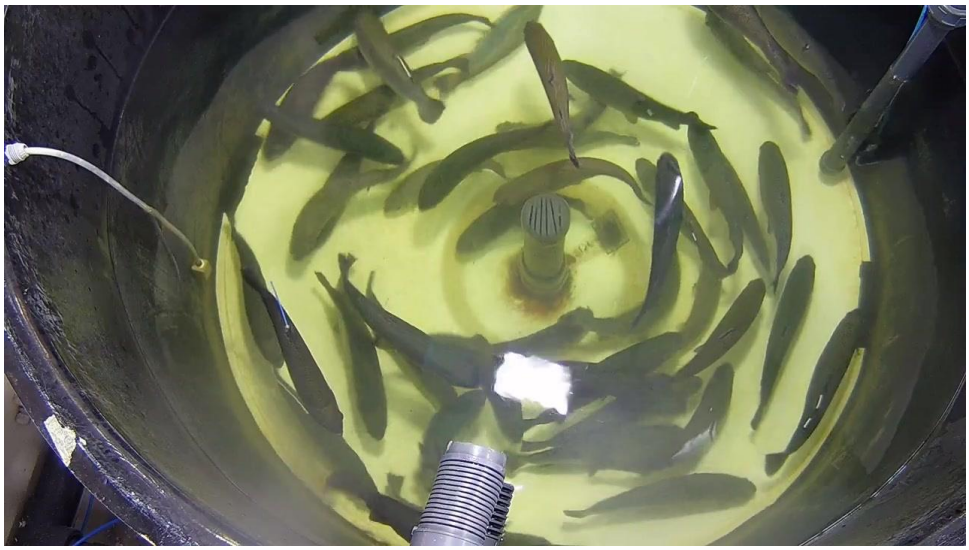


Figure 3: Snapshot of rainbow trout (O. mykiss) in culture tank of recirculating system

Feeder setup

Arvotec T-Drum 2000 automatic feeders distribute feed on a routine basis to tanks according to a set time schedule (feeding pattern). The feeders are connected to the OxyGuard Commander Feeder Controller and controlled through the OxyGuard monitoring system (see section 5 for details on the feeder connection).

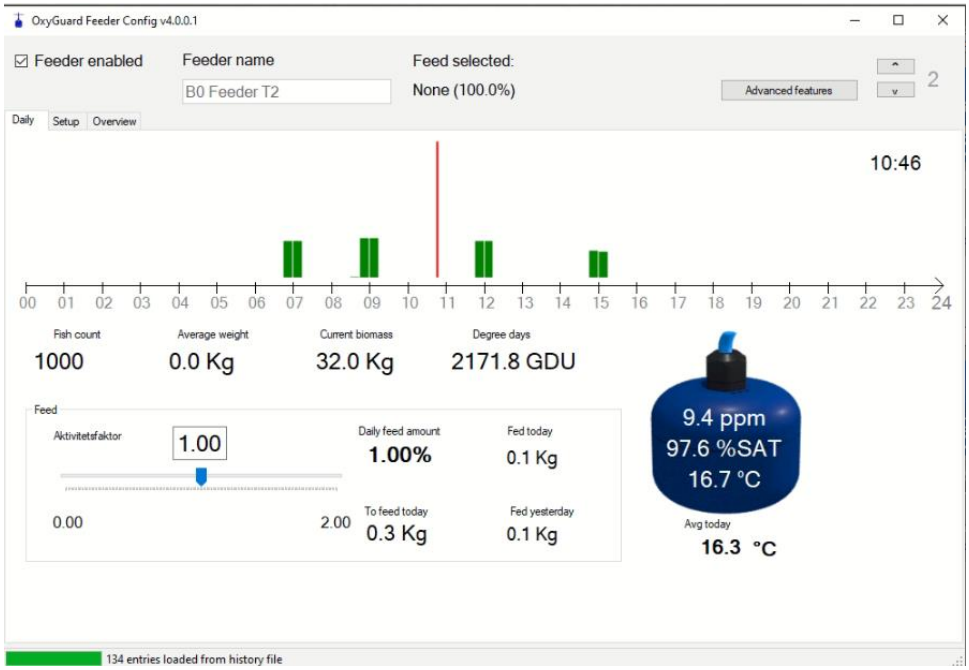


Figure 4: Vigo representation of feeder configuration

For some trails performed within the project, e.g., satiation trials, the controller is setup in a "bypass" mode. This involves a separate relay switch, used to activate and deactivate the feeder and overwriting the feeder control system completely. This is necessary because manual control of the feeders is needed to be able to feed the fish to satiety.

2.2 iBOSS data model at ABT

The ABT data model on the iBOSS follows the NGSI-LD Data Model[1] principles and reflects main existing components and measured parameters at ABT.

The NGSI-LD data model is mainly based on Entities. An entity in NGSI-LD is an informational representation of something that is supposed to exist in the real system to be modelled, physically or conceptually. *Entities* in NGSI-LD may have NGSI-LD *Properties* and *Relationships* relating entities to each other's.

The NGSI-LD ABT data model contains five types of entities: Fish Containments, Pumping sump, Device, Feeder and Feed, each one detailed in the subsections below.

2.2.1 Fish Containment Entities

In the ABT NGSI-LD data model there will be 12 Fish Containments entities representing the 12 tanks. Each Fish Containment entity contains the following NGSI-LD attributes:

Table 4: Fish Containment Entity Model Attributes

Attributes	Type/value	Sub-Attributes	Comments
id	urn:ngsi-ld:FishContainment:ABT:BAY01		Used as a unique identifier of the entity.
type	FishContainment		
temperature	Property	observedAt	Time in ISO 8601 format.
		value	The value of the property
		unitCode	Unit code of the property in CEFACT format. "CEL" for Celsius.
		observedBy	the Relationship used to link the observed property to the Device entity which delivered the value.
		refAlert	Used to link the
dissolvedOxygen	Property ps: Multi-Instance Property. This property will contain 2 instances dedicated for dissolved oxygen values measured in M1 (mg/l) and in P1 (percent)	observedAt	Time in ISO 8601 format.
		value	The value of the property.
		unitCode	Unit code of the property in CEFACT format. P1/M1

[1] NGSI-LD-Spec - https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.04.02_60/gs_CIM009v010402p.pdf

		observedBy	The Relationship used to relate the observed property to the Device entity which delivered the value.
		datasetId	The identifier of the instance.
refSump	Relationship	object	The Relationship used to model the relation between a Tank and the Sump entity.
feedingOperation	Property	observedAt	Time in ISO 8601 format.
		value	The value of the property. It is used to model the quantity of feed.
		unitCode	Unit code of the quantity of feed.
		startsAt	A sub-property to model the starting date and time of the feeding operation.
		endsAt	A sub-property to model the ending date and time of the feeding operation.
		refFeeder	A Relationship to link the feeding operation to its executer. The Feeder may be processed by a Farmer or Automatically.
		refFeed	A Relationship to link the feeding operation the nature of delivered feed.
context	https://raw.githubusercontent.com/easy-global-market/ngsild-api-data-models/master/aquac/jsonld-contexts/aquac-compound.jsonld		A json-LD specification in order to define the URI of used attributes

Table 5: Example of Fisch Containment Entity in NGSI-LD


```
{    "id": "urn:ngsi-ld:FishContainment:ABT:BAY01",
    "type": "FishContainment",
    "refSump": {
                "type": "Relationship",
                 "object": "urn:ngsi-ld:Sump:ABT"
     },
    "dissolvedOxygen": [

        {
            "type": "Property",
            "datasetId": "urn:ngsi-ld:Dataset:dissolvedOxygen:M1:2937",
            "refAlert": {
                "type": "Relationship",
                "object": "urn:ngsi-ld:Alert:DO:M101"
            },
            "value": 5.4,
            "observedAt": "2021-08-31T11:31:29Z",
            "unitCode": "M1",
            "observedBy": {
                "type": "Relationship",
                "object": "urn:ngsi-ld:Device:OxyguardPacific01"
            }
        },
        {
            "type": "Property",
            "datasetId": "urn:ngsi-ld:Dataset:dissolvedOxygen:P1:2937",
            "value": 80,
            "observedAt": "2021-06-26T19:32:52Z",
            "unitCode": "P1",
            "observedBy": {
                "type": "Relationship",
                "object": "urn:ngsi-ld:Device:OxyguardPacific01"
            }
        }
    ],
    "temperature": {
        "type": "Property",
        "datasetId": "urn:ngsi-ld:Dataset:temperature:CEL:2937",
        "value": 15.2,
        "observedAt": "2021-06-26T19:32:52Z",
        "unitCode": "CEL",
        "observedBy": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:Device:OxyguardPacific01"
        }
    },
    "@context": ["https://raw.githubusercontent.com/easy-global-market/ngsild-
api-data-models/master/aquac/jsonld-contexts/aquac-compound.jsonld"]
}
```

2.2.2 Sump Entity

The NGSI-LD Sump entity is used to model the Sump installed in at the ABT. The Sump entity contains the following attributes

Table 6: Sump Entity Model Attributes

Attributes	Type/value	Sub-Attributes	Comments
id	urn:ngsi-ld:Sump:ABT		Used as a unique identifier of the entity.
type	Sump		
redoxPotential	Property	observedAt unitCode (2Z) observedBy	
totalGazPressure		observedAt unitCode (Pa) observedBy	
co2	Property	observedAt, unitCode (59), observedBy	
pH	Property	observedAt, unitCode (Q30), observedBy	
refFishContainment	Relationship ps: Multi-Instance Relationship.	object datasetId	The Relationship used to model the link between a Sump and all fish containments at ABT.
context	https://raw.githubusercontent.com/easy-global-market/ngsild-api-data-models/master/aquac/jsonld-contexts/aquac-compound.jsonld		

Table 7: Example of Sump Entity in NGSI-LD


```
{
    "id": "urn:ngsi-ld:Sump:ABT",
    "type": "Sump",
    "redoxPotential": {
        "type": "Property",
        "value": 8.21,
        "observedAt": "2020-09-12T18:09:23Z",
        "unitCode": "2Z",
        "observedBy": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:Device:OxyGuardPacific:ABT:01"
        }
    },
    "totalGazPressure": {
        "type": "Property",
        "value": 20,
        "observedAt": "2020-09-12T18:09:23Z",
        "unitCode": "Pa",
        "observedBy": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:Device:OxyGuardPacific:ABT:01"
        }
    },
    "co2": {
        "type": "Property",
        "value": 8.21,
        "observedAt": "2020-09-12T18:09:23Z",
        "unitCode": "59",
        "observedBy": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:Device:OxyGuardPacific:ABT:01"
        }
    },
    "pH": {
        "type": "Property",
        "value": 8.21,
        "observedAt": "2020-09-12T18:09:23Z",
        "unitCode": "Q30",
        "observedBy": {
            "type": "Relationship",
            "object": "urn:ngsi-ld:Device:OxyGuardPacific:ABT:01"
        }
    },
    "@context": [
        "https://raw.githubusercontent.com/easy-global-market/ngsild-api-data-
models/master/aquac/jsonld-contexts/aquac-compound.jsonld"
    ]
}
```

2.2.3 Device Entities

The NGSI-LD Device entity is used to model the Oxyguard Pacific installed at the Sump and Fish containments. The NGSI-LD model of the Oxyguard Pacific Device Entity will follow the Device model of Smart Data Models of Fiware[2].

Table 8: Device Data Model Attributes

Attributes	Type/value	Sub-Attributes	Comments
id	urn:ngsi-ld:Device:OxyguardPacific01		Used as a unique identifier of the entity.
type	Device		
connectsTo	Relationship		A Relationship to link the Device to its emplacement.
			All Other Properties/Relationships may be picked from the Fiware Smart Data models specification: https://github.com/smart-data-models/dataModel.Device/blob/master/Device/doc/spec.md
context	https://raw.githubusercontent.com/easy-global-market/ngsild-api-data-models/master/aquac/jsonld-contexts/aquac-compound.jsonld		

The automatic feeder installed on fish Containments is also modelled as a Device NGSI-LD Entity and follows the Device Model of Fiware Smart Data model specification.

2.2.4 Feed Entities

Feed Entity will contain details about the distributed food. All Feed types presented at ABT will be modelled as NGSI-LD entities to know the type of distributed food in a feeding operation.

[2] https://github.com/smart-data-models/dataModel.Device

2.2.5 Actuation Supports in NGSI-LD

Following the needs of monitoring the feeder to adapt the quantity of food for fishes, the iBOSS should provide data models and mechanisms that ensure the control of the feeder.

Actuators are defined as things (mainly devices) that can change their state or execute actions. The application interacts with an actuator through the actuator's Digital Twin System.

The NGSI-LD data model is an entity-based data model in which properties describe the status of parameters of the entity. Executing the feeder will request updating some properties (via the NGSI-LD API) and then certainly technical solutions that ensure transmitting the status change to the feeder to be executed.

There is currently no explicitly and precisely defined support for actuation in the NGSI-LD API. Thus, a best-practice NGSI-LD API and data model for interaction between application and actuators is proposed.

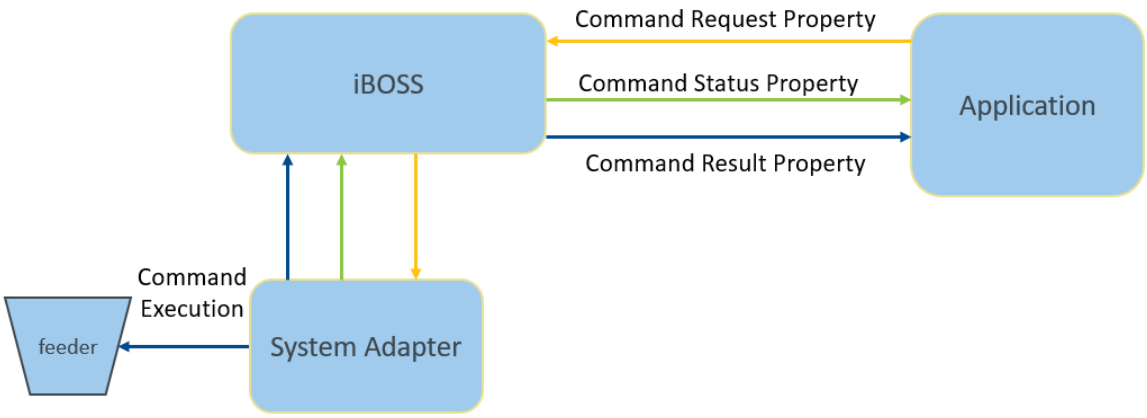


Figure 5: Actuation Supports in iBOSS

Commands are sent to the iBOSS, using the standard NGSI-LD API and a suggested convention for representing them. In turn, feedback about command execution is received by the application, both as continuous status updates and/or a final command result.

More specifically, the component that handles direct communication with the actuator is the System Adapter: it uses an actuator-specific protocol to control the actuator and get responses and updates from it, i.e., from the real system. On the other hand, the System Adapter is able to use the NGSI-LD API to receive NGSI-LD command requests from the NGSI-LD Entity and send back command status and result to it. The NGSI-LD Entity is responsible for handling direct communication with the application.

2.2.6 Summary

The NGSI-LD data model of the ABT in the iBOSS will serve as a Digital Twin model of the ABT workflow. The NGSI-LD model is evolutive and is continuously updated. The iBOSS NGSI-LD API proposes several operations for evolving the model and updating its values keeping the iBOSS model up to date and exploiting data for new processes such as predictions and decision making (see section 4.3).

The iBOSS offers a visual extension (based on Grafana[3] tool) to follow temporal evolution of collected data. Below some screenshots (Figure 6, Figure 7 and Figure 8) issued from the visualization tool of the iBOSS with the data from real time data collected from ABT fish containments and sump.

Figure 6: Sump Data pH, redox potential and CO_2

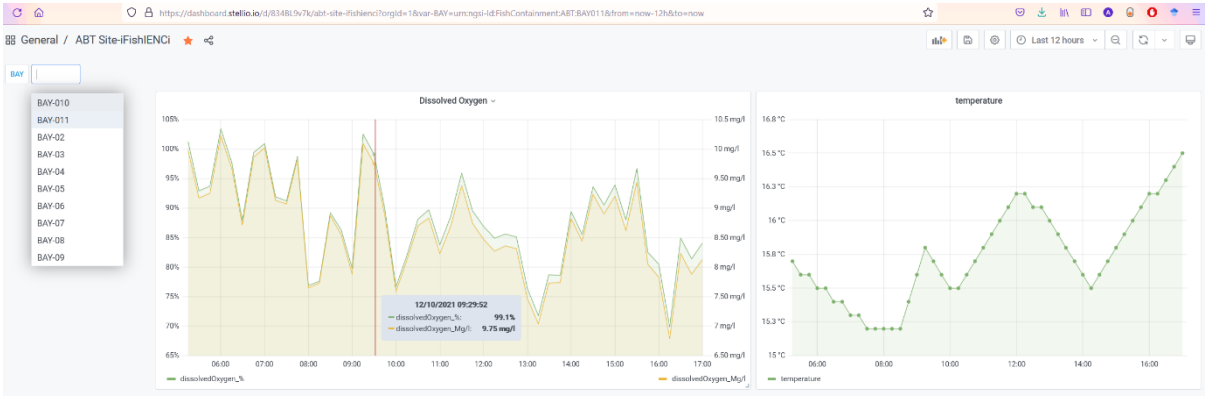


Figure 7: Dissolved Oxygen in Percent and mg/l and Temperature of Fish Containment BAY0-11

[3] Grafana - https://grafana.com/

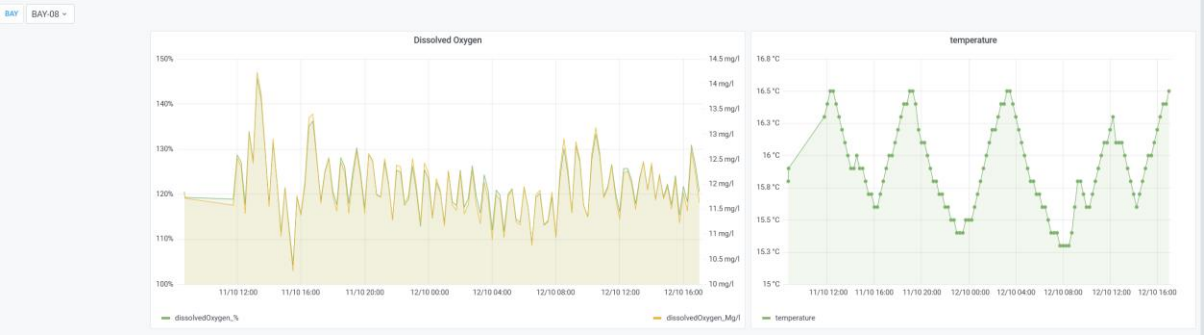


Figure 8: Dissolved Oxygen in Percent and mg/l and Temperature of Fish Containment BAY0-8

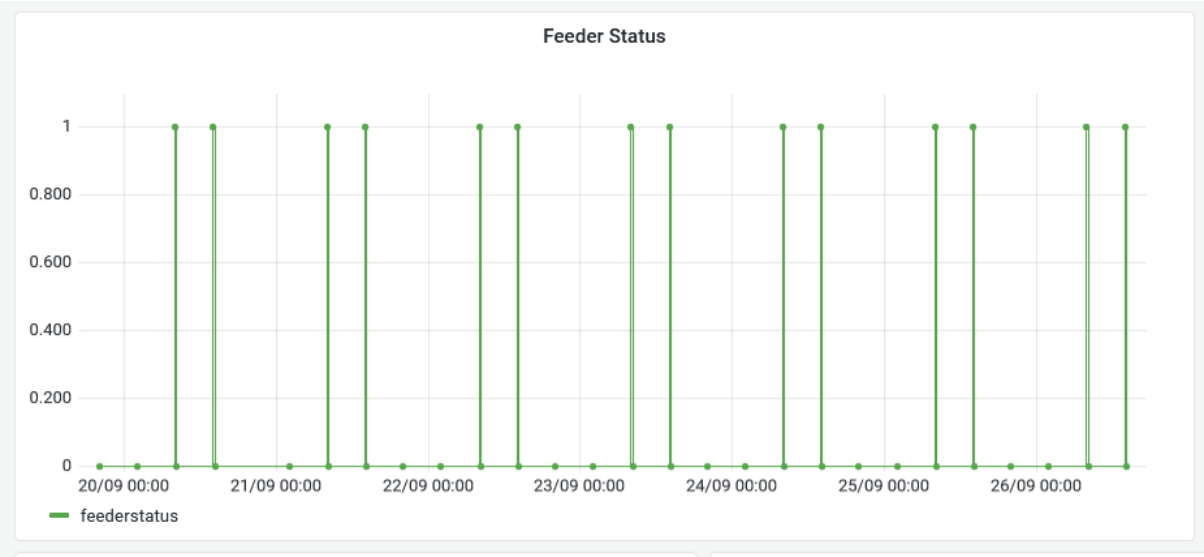


Figure 9: Feeder Status: Feeding executed 2 times per day at 8h and 14h for 15 min each one

3 Model development and integration

3.1 Different approaches have been developed, each providing a specific assessment of the performance of the feeding. For iBOSS to be relevant for different types of systems (e.g., RAS, open cage), dedicated approaches are considered for each situation. The objective is to assess the value of these systems for the smart feeding algorithms of iBOSS. Camera-based behaviour analysis in ABT

The goal of the work done on ABT cages is to measure fish behavior based on camera footage, in order to later be able to estimate fish satiety based on this behavior. The pipeline consists of 4 steps: detecting the fish, tracking them, computing behavior metrics, and sending those metrics to the iBOSS.

3.1.1 Data

There were two rounds of satiety trials done at ABT tanks, providing data rich in volume and diversity to train detection and behavior models. The first round of trials varied the duration of starvation of the fish prior to feeding (24h, 6h, 3h starvation). Each individual feeding pulse timestamp was labelled manually on the video, and Figure 10 represents a simplified view of the fish activity throughout the feedings.

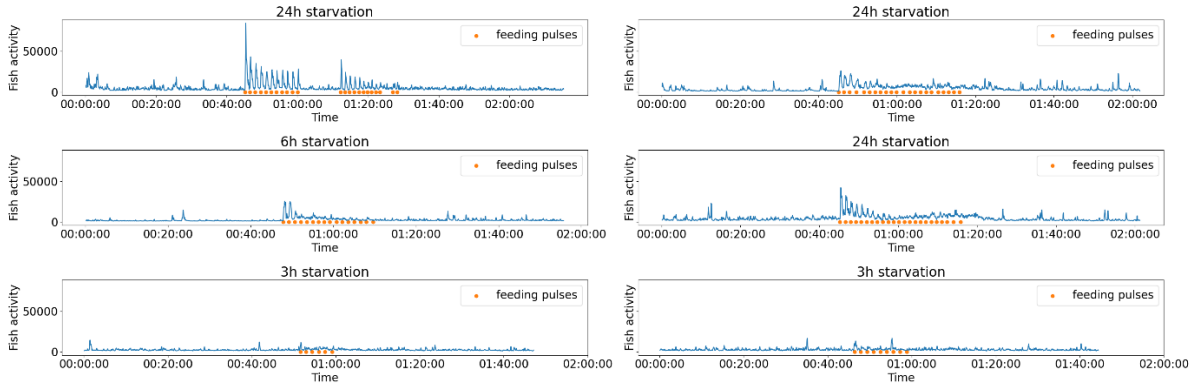


Figure 10: Fish activity and feeding pulses during the first trials

The second round of trials varied the amount of feed provided to the fish (minimum feeding, recommended feeding, maximum feeding).

These videos were used primarily to train the fish detector model, so that it would be capable of detecting the fish in various scenarios, and not just at rest. At a later time, they will allow the development of a fish behaviour classifier based on the metrics measured on the video, thanks to the detection and tracking.

3.1.2 Detection

Fish detection was achieved using a deep object detection model implemented in Python, YOLOv5[4]. For calibration, a light pretrained model was used (coined YOLOv5s), and retrained on 148 images from ABT tank videos, manually labelled in the BiOceanOr labelling software. The model was evaluated on 20 other images, labelled likewise. Each image contained between 13 and 44 fish (this number varied through different experiments and fish sizes, but also with hidden fish impossible to label on certain frames), so there were in total 3741 training fish instances and 456 validation instances. The images were chosen to represent the diversity of behaviors, such as resting, rotating, or rushing to eat. The labels are bounding boxes: a rectangle surrounding the fish as closely as possible. Figure 11 represents an example of detections on one frame.

To evaluate the model on validation data, the default parameters of YOLOv5 were used for the confidence threshold (which activates a detection) and the intersection-over-union threshold (used to measure if a detected box is "sufficiently similar" to the actual box). With these, we reached a precision of 82% and a recall of 87% on the validation dataset ("precision" is the percentage of model detections that correspond to actual boxes, and "recall" is the percentage of real boxes that were detected).

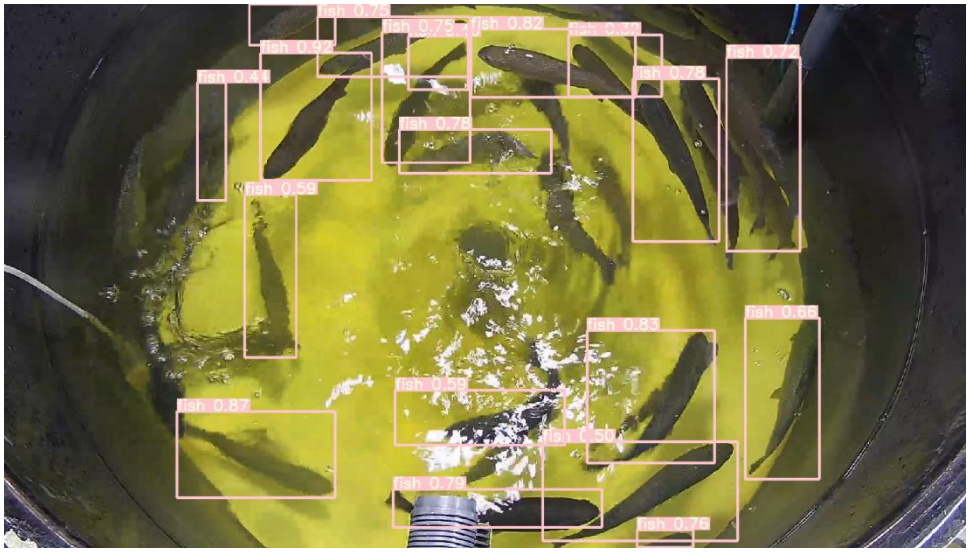


Figure 11: Fish detection results from trained model

3.1.3 Tracking

Fish tracking was done using Norfair[5], a Python library which implements a tracker using Kalman filters, pluggable on top of any object detector, like YOLOv5. With the fish moving quite fast, the minimal

[4] https://github.com/ultralytics/yolov5
[5] https://github.com/tryolabs/norfair

frame rate to track the fish without loss of information was 12.5 frames per second (which is every other frame of a 25-fps video). For this task, the centroids of the bounding boxes were tracked. In the future, detecting and tracking body parts like the head and the tail of the fish may greatly improve tracking, as fish crossing at a 90° angle could not be mixed up. The performance of the tracker was evaluated visually, lacking resources to manually label fish trajectories. Figure 12 is a representation of the tracking, where each individual fish speed is represented as an arrow.

Figure 12: Fish tracking example

3.1.4 Computing metrics

Based on the trajectories of the fish in the tank, several metrics were computed to identify their behavior. The choice of these metrics was made by representativity (if it gives a useful information about the movement of the fish) and performance (if it works). Metrics chosen were:

- **Median speed and the two quartiles of speed:** each individual fish speed is measured, but that would be too much information. The distribution of the speeds was evaluated with the median and quartiles, meaning the three thresholds such that, respectively, 25%, 50% and 75% of the fish go slower than it (using the maximum value to detect a single attack was not reliable). At the start of a feeding, typically, all fish started moving faster, but there are moments when a fraction of the fish were at rest, so there will be a larger gap between the 25% quartile and the 75% quartile.

- **Rotation: A**n ad-hoc algorithm was developed (based on the physical notion of curl, see Figure 13) which measures if the fish are moving in a circle together in the cage. It is negative if the fish are moving clockwise, and positive if they are moving counterclockwise. This is meant to detect the typical behaviour after feeding, when the fish start moving together in a circle.
- **Convergence:** A second ad-hoc algorithm was developed (based on the physical notion of divergence, see Figure 13) which measures if the fish are moving together towards a single point in the cage. That is typical of the start of a feeding, when the fish rush towards the food.
- **Detections:** finally, a safety metric was kept, which measures how many fish are being detected. This gives a confidence level on the other metrics. Typically, the more the fish are agitated, the fewer fish are detected. Indirectly, we can therefore also measure the fish activity through the number of detections.

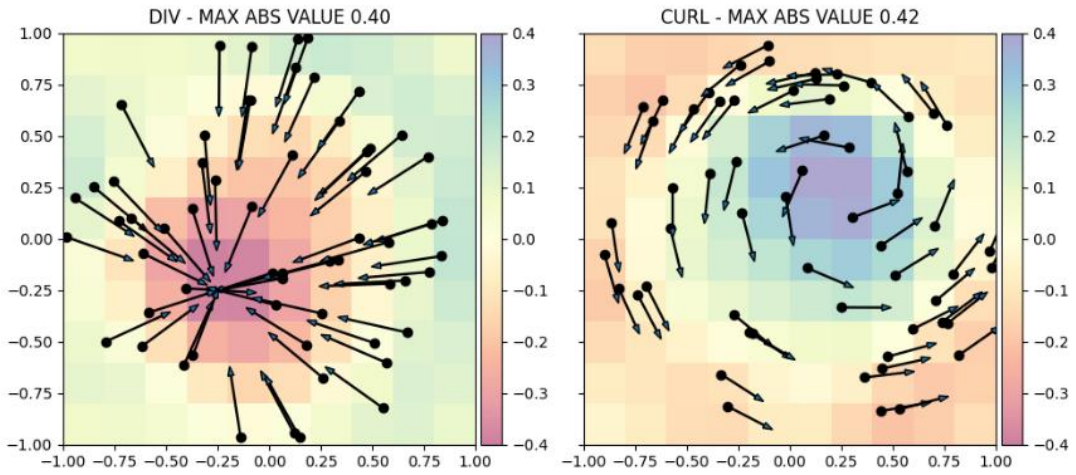


Figure 13: Illustration of convergence ("DIV") and rotation ("CURL") metrics based on fish speeds

Fish were tracked, and the metrics were computed, at 12.5 fps. However, in order to save storage space and without significant loss of information, we measured that the metrics could be averaged and sent every 8 frames, which gave a sending rate of 1.5625 fps (one sending every 0.64s). Beyond that, the sudden bursts of activity characteristic of aggressions and feedings were lost. In the future, we may want to adapt the sending frequency to fish activity, as there are long periods of relative rest.

Figure 14 represents an example of scenario, where the fish were initially rotating (clockwise, rotation is negative) the first ten seconds, probably because they had eaten already, then food was dropped in the tank, so the fish rush towards it (convergence is high, and the number of detections dropped), but quickly the fish started rotating again, they were not very hungry.

Figure 15 is another example, where the fish were at rest in the beginning (low speed), probably starved, then food was dropped, and they all started moving much faster, with a peak in convergence, and the number of detections dropped and remained low because this time, they were very agitated, very hungry.

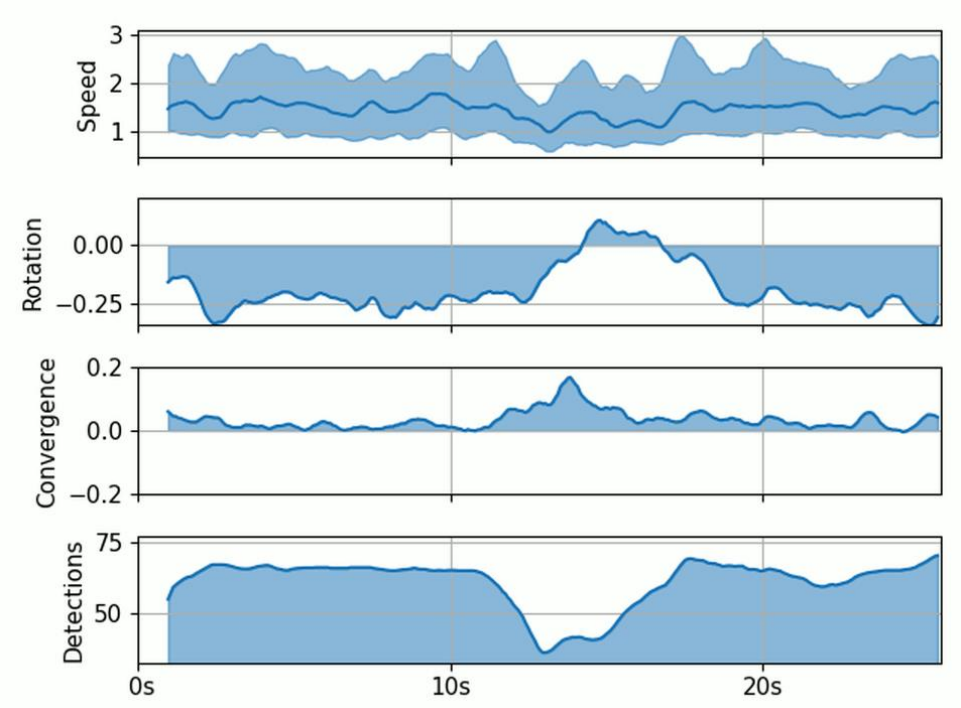


Figure 14: Metrics during fish feeding – rotation then convergence

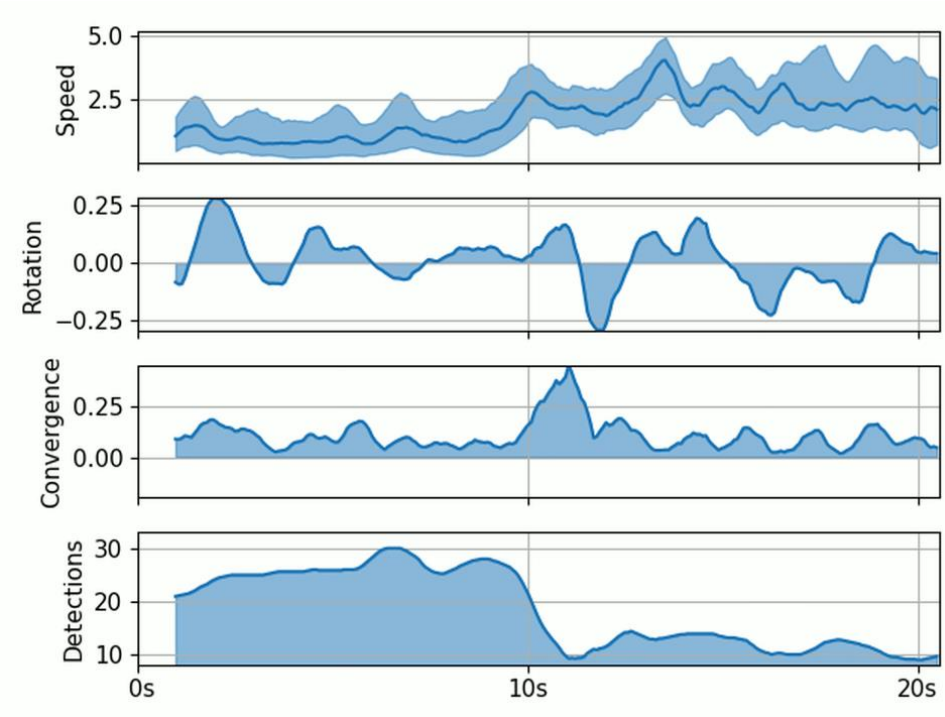


Figure 15: Metrics during fish feeding – rest then convergence

3.1.5 Pipeline activation

The video processing pipeline can run all day long, but the most interesting phases are around feedings. In order to reduce storage space, we only start the processing 10 minutes before the feeding starts, and end it 30 minutes after feeding has stopped. In order to do that, we read the video stream continuously with a ten-minute buffer, and when the feeder starts, it sends a MQTT message through the Raspberry PI, which triggers the recording and the processing, starting at the top of the buffer.

3.1.6 Computational challenges and solutions

The detection and tracking algorithms are computationally heavy. The detection can be significantly improved by using a GPU, but the tracking only works on CPU. Therefore, without a very powerful computer, the processing cannot be done in real time for now. The solution implemented is to run the processing offline: during the feeding, a first process records the video stream. As soon as the recording has ended, it enters a second, parallel process, which can deliver the metrics in its own time (meanwhile, another recording can start in parallel). This does not allow real time feeder control, but can provide insight to adapt the next feeding, based on the fish reaction to the present feeding.

3.2 Camera based behavior analysis in HCMR

Automated routines that can track European seabass (*Dicentrarchus labrax*) (i.e., extract fish trajectories of a short time length) in recirculating aquaculture system (RAS) and sea cages using single cameras were developed by HCMR. They use a combination of methodologies from Computer Vision and Artificial Intelligence, using models such as the YOLOv5 and Deepsort. The models are able to extract different individual or group swimming performance metrics, such as fish speed, direction, fish turning events and group polarization. The extracted feature parameters can be used for the detection of variations in swimming behaviour and we provide application examples of the developed methodology using different husbandry scenarios. Specifically, daily (morning and afternoon) variations in swimming speed are calculated and presented for both rearing systems. In addition, variations of the group direction in the sea cages, daily differences in the vertical distribution of the fish in tanks and, specifically, their presence close to the tank surface are also presented. Furthermore, based on the group-level motion changes of the fish, the models can detect when feeding starts and ends in the sea cages.

In a RAS system, a group of 50 E. seabass juveniles (weight $W = 200\ g$ and total length $L = 25.1 \pm 0.94\ cm$) was kept in a cylindroconical tank of 2 m^3 volume and 1.5 m diameter at a thermoregulated marine RAS under typical rearing conditions for the species and the group was monitored using network cameras capturing at 6 fps for a period of two months (May and June 2019), from 08:00 to 19:30. In open cages, a group of E. seabass fish of 220 ± 30 g body weight at a stocking density of 5.2 kg m^{-3}cage farm of HCMR. A submerged network camera (Fyssalis$4\ \mathrm{m}$$15\ \mathrm{minutes}$.

The routine (based on OPENCV/Python; OPENCV 4.0[6], Python 3.8[7]) that automatically tracks the fish in tanks and sea cages for a short time consists of the following steps: 1) the pre-processing step, where the images are denoised and prepared for analysis; 2) the fish detection step, where the background is subtracted (using a simple threshold and object filtering in an iterative way for the open cages, the GMM/KNN background subtractor from OPENCV library for RAS) and the objects are filtered out to select the most appropriate ones; and 3) the association step, where each detected object is associated with an object or a predicted centroid position.

The speed is calculated as the Euclidean distance between two consecutive frames divided by the estimated fish length to give the normalized speed values. The direction of the fish was calculated as the angle between the velocity vector (starting from the centroid position at the previous time frame and ending at the centroid position at the current time frame) and the horizontal positive x-axis of the

[6] https://opencv.org/
[7] https://www.python.org/

image. To estimate the preference for the tank surface, the sum of the foreground (white) pixels in the image is divided by its total number of pixels. This value shows the normalized area in the image that is covered by the fish. Low values indicate that most fish remained at the bottom of the tank, while high values indicated a preference of the group to be on the surface.

Four different measures were used to evaluate the presented methodology: the precision in fish detection (the total number of correctly detected positions divided by the total number of estimated positions), the precision in speed estimation (the mean and standard deviation of the difference between the manually and automatically extracted speed), the error of the length estimation (the median difference between the manual and the automatically estimated fish length and the deviation was the inter-quantile range). Fish from five videos were manually tracked to compare the result with the automatically extracted trajectories.

To test the proposed methodology for its sensitivity in detecting behavioural changes in European seabass, we estimated the swimming speed, the direction, and the preference for the tank surface at different times of the day and for different human presence/absence and feeding scenarios. To detect significant (significance level: $\alpha = 0.05$) changes between morning and afternoon speeds in both RAS and sea cage systems, repeated measures ANOVA tests were applied (AnovaRM StatsModels Python library). ANOVA tests were applied after verifying that the assumptions of the test were met (normality was tested using the Shapiro-Wilk test).

Monitoring feeding and abnormal behaviours

A group of E. seabass fish of 200 g body weight was reared in a circular polyester cage (40 m diameter, 9 m depth) located at the pilot scale cage farm of HCMR at Souda bay, Crete (certified as an aquaculture facility from the national veterinary authority; code GR94FISH0001). A submerged network camera (Fyssalis V3.1) capturing at 10 fps was used for monitoring and video recording during daylight hours in April 2022. The camera was positioned at 4 m depth using a gyroscopic gimbal stabilizer to ensure it pointed upwards. Manual feeding was performed once daily, between 08:00 to 10:00. A machine learning model for tracking people (DeepSort[8]) was trained and adapted to track fish individually (using OPENCV/Python) and extract their speed and direction. In addition, computer vision techniques that can detect feeding events and group polarized movement were incorporated into the model.

[8] https://github.com/nwojke/deep_sort

3.2.1 Results

Monitoring swimming behaviour

In RAS, the percent of the detected objects in time decreases, with 30–60% of the initially detected fish remaining detected and tracked after seven frames. This shows that in each frame, there is at least one or two tracked objects. The centroid detection precision, i.e., the total number of correctly detected positions divided by the total number of estimated positions, is 0.85 ± 0.11. The length estimation error is $46.07 \pm 17.29\ pxls$ or for cages and 0.60 ± 0.46 if normalized, i.e. if the error is divided by the total length in pixels. The algorithm estimates a constant fish length. The precision of the speed estimation is 0.045 ± 0.03 bd/sec. However, the methodology tends to slightly overestimate speed values. In cages, the total number of detected objects in the first frame varied between 30 and 50 individuals. The percent of the detected objects in time decreased with 20% of the initially detected fish remaining after 15 frames. This indicates that, on average, a minimum of three individuals were tracked at all time frames. The centroid detection precision is 0.89 ± 0.07. The algorithm tends to underestimate the length of the very large fish, i.e., the fish that are very close to the camera. The error of length estimation is $25.8 \pm 27.1\ pxls$ and the normalized was 0.43 ± 0.46. The precision of the speed estimation is 0.015 ± 0.009 bd/sec. Again, the algorithm captures the slight temporal changes but overestimates the speed magnitude.

Fish speed showed a significant decrease between morning and afternoon, from 0.48±0,09 to 0.27±0,09 Bd per frame in RAS (Sunday: $F-statistic\ 0.11,\ P-value = 0.77$; Monday: $F-statistic = 10.91,\ P-value = 0.05$; Wednesday: $F-statistic = 13.79, P-value = 0.03$, Figure 7). On Sundays, the speed remained at the same level in the morning and the afternoon (0,45±0,15 and 0.38±0.15 Bd per frame, respectively). In contrast, during normal feeding days, the speed in the morning (0,46±0,05 Bd per frame) was at the same level as on Sunday (Morning: $F-statistic\ 0.32,\ P-value = 0.74$; Afternoon: $F-statistic\ 0.29,\ P-value = 0.76$) but decreased significantly in the afternoon (0,31±0,03 bd/frame). On Sundays, the fish show a constant preference to be at the surface. In contrast, during normal days, the preference varies, with the fish avoiding the surface until late in the afternoon, a period associated with a human presence.

In cages, fish show a similar pattern in the daily speed variation. During the morning they have a significantly higher speed (1.31±0,60 Bd per frame) that decreases in the afternoon (0,83±0,26 Bd per frame), (F-statistic: 20.58; P-value < 0.001). The methodology can detect differences in direction. As fish move freely in an asynchronized way their directions are widely spread between 0 and 360 degrees, while, when fish move in synchrony the distribution of the swimming directions narrows, and all angles are around the main directional component of the group motion.

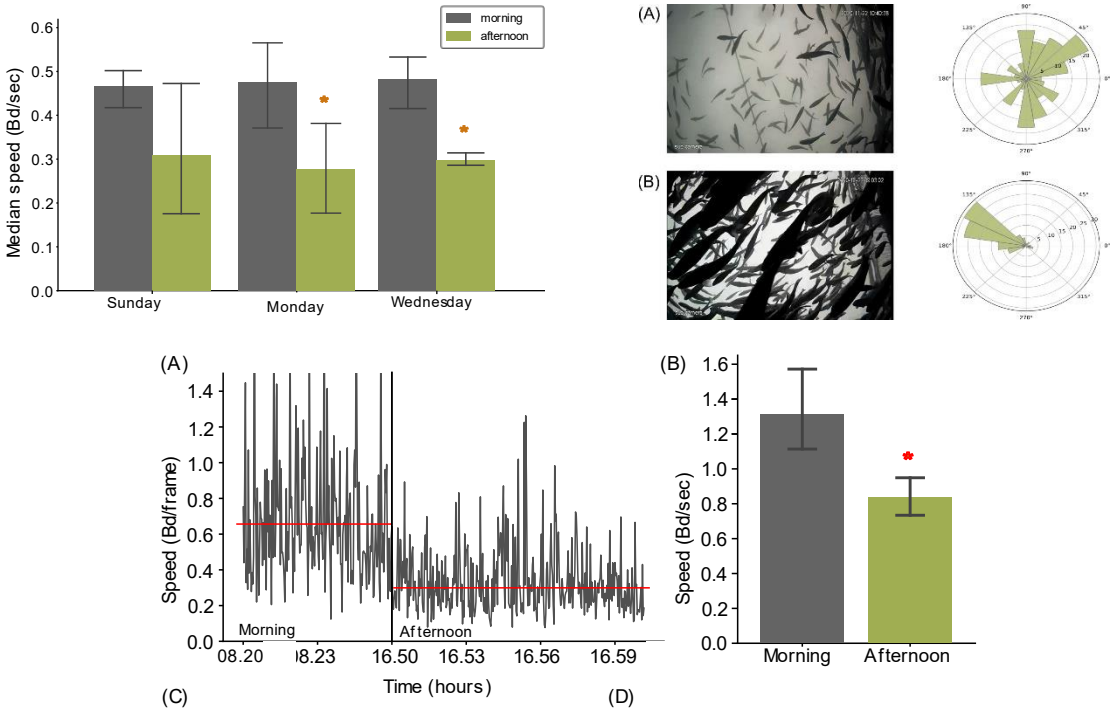


Figure 16: Fishes median speed evolution over time.

Monitoring feeding and abnormal behaviours

The system is capable of distinguishing between three different movement patterns related to feeding (Figure 17). The polarized motion seen before feeding is realized (Figure 17 left), motion that resembles feeding behaviour but lasts for a limited time period and could be an indicator of feeding or other warning situation (Figure 17 middle) and the feeding event where the fish shoal swirls around the feed (Figure 17 right).

Figure 17: Automatic detection of three different movement states

In addition, based on the group density changes seen during the feeding, the system is capable of monitoring the evolution of the feeding (the parameter is called feeding ratio) and detect when the fish group's density is decreasing.

Figure 18 shows two examples of how this parameter evolves during feeding and decreases significantly when fish are satiated. This decrease is supposed to be correlated with the satiation levels of the group, something that will be confirmed in future analysis.

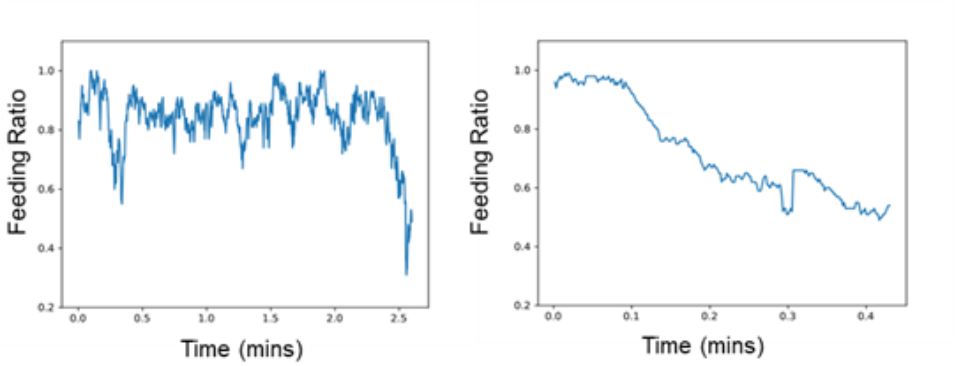


Figure 18: The evolution of feeding ratio in time.

Furthermore, the system is also capable to detect variations in the activity of the fish. Extraction of the group speed for two consecutive days showed significant variations of the fish speed during the day, suggesting that the fish start being more active during and after feeding and less active in the afternoon. More specifically, the group's speed increased from 0.2 bd/s to 1.1 bd/s during the feeding window (i.e., from 08:00 to 10:00, Figure 19), and it remained at maximum levels until 12:00, two hours after feeding. Then it started decreasing until it reached a minimum speed of 0.2 bd/s at midnight. This pattern is repeated also in the second day.

Figure 19: Variations in speed for two consecutive days in April 2022.

3.3 Echosounder deployed in HCMR

Monitoring the fish behaviour in a cage is essential for optimizing feeding control. Several methods have been used, including video monitoring and echosounders. Echosounders provide information on the distribution of fish in a cage a parameter related with the feeding behaviour as shown for salmon by e.g., F. Oppedal, T. Dempster, L. H. Stien, 2011[9]. As analogous behaviour is expected for the E. seabass, a SIMRAD EK 15 single beam system has been installed in the pilot scale cage farm of the HCMR and data related to the vertical distribution of the fish is collected. In Figure 20Figure 20 a typical dataset is shown.

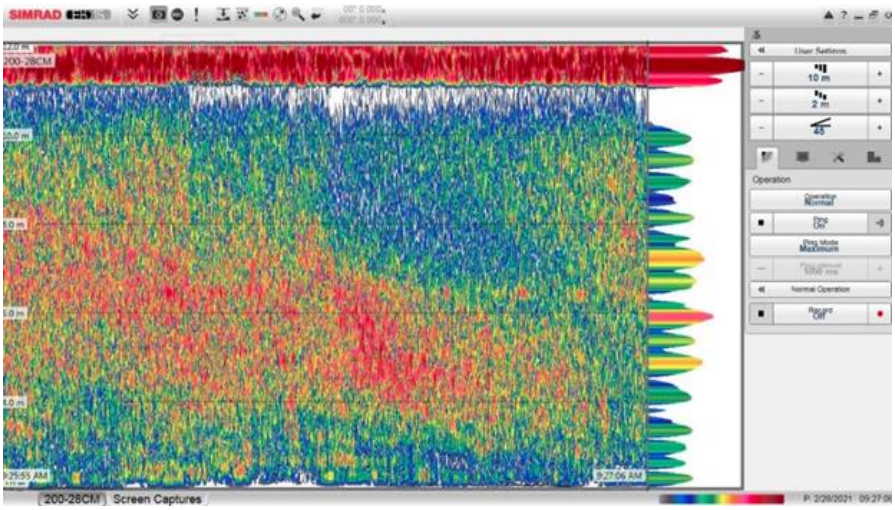


Figure 20. Vertical distribution of a E. seabass group in a cage

HCMR and NORCE are currently working on an application that will allow the online availability of the data (iBOSS cloud) and for its integration to the iBOSS. **Error! Reference source not found.** shows the echosounder data from iBOSS dashbords.

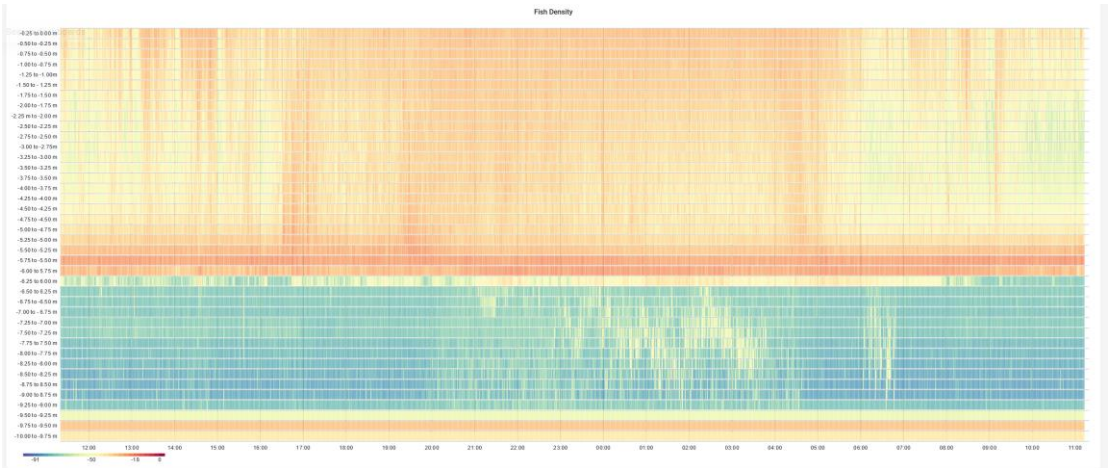


Figure 21 : Echosounder data from iBOSS dashboards

[9] Environmental drivers of Atlantic salmon behaviour in sea-cages: A review. Aquaculture 311: 1-18)

3.4 FishMet

FishMet is a digital simulator of salmon feeding, fish behaviour, physiology, and metabolism, which consists of equations that define feed intake, stomach filling, digestion, absorption, retention of nutrients, dynamic energy budget and growth. It will be developed to include feeding behaviour and waste production. The FishMet model include new knowledge about physiological factors that affect appetite and feed intake. Intestinal passage and postabsorptive processing are principal elements. A central part of the model focuses on the changes that occur due to variations in feed type, feeding amount and frequency as well as how environmental factors including temperature, oxygen availability and stress (e.g., handling) will affect feeding behaviour and processes leading to growth. The software engineering work, realising the FishMet, aims to produce a highly modular and customizable system that can run as a separate application and be embedded into the larger decision-support and intelligent fish farm control system iBOSS (see Figure 22). The FishMet development is decribed in details in D1.4.

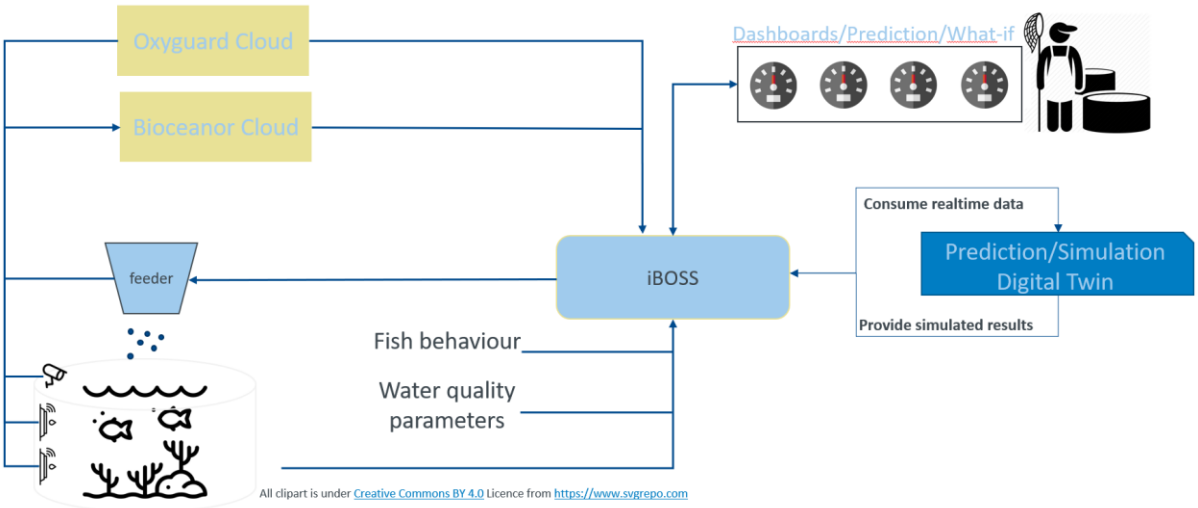


Figure 22: Overall schematic of digital twin integration in a fish farming system

The fish digital twin model accepts the basic parameters of the fish biology as input, such as body size, energetic characteristics. These parameters are based on experiments and published parameters of fish physiology. Environmental inputs include the physical characteristics of the feed, feeding schedule, ambient temperature and oxygen level. The model's outputs are fish appetite, food intake, feed conversion efficiency, waste production and growth. The autonomous agent model also allows to collect additional outputs, such as activity patterns, and predict the internal state of the fish, e.g., motivation and stress level, responses to unexpected change of the feed, stochastic environmental perturbations etc. The modular organization of the model allows to simulate a single fish or a large group in an individual-based model. Water quality, environmental data, fish physiology and behaviour data are collected and made available to the FishMet model in real time through the iBOSS.

3.4.1 Digital Twin integration

The DT integration on the iBOSS is based on a layered data model approach (see Figure 23). The models of these layers and their data models are detailed in the sub-sections below.

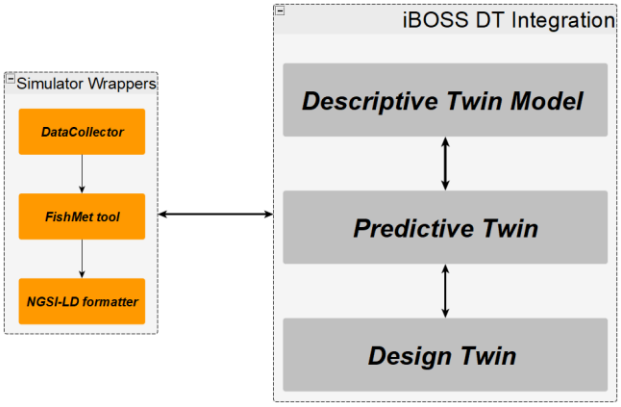


Figure 23: DT Integration

3.4.1.1 Descriptive Twin Model

The descriptive twin model consists of modelling the current use case. This layer will follow the iBOSS data model of ABT site detailed in section 2.2. It models the set of Fish containments, installed devices their properties and relationships.

To enhance the FishMet DT integration, the Descriptive twin model will also contain two new types of entities: Fish Population and Specie used to perfectly define the cultured fishes in fish containments. The data models of these entities are detailed below:

- **Species Entities** models the scientific definition of a species that may be cultured in the fish containment. For a correct FishMet DT integration, the Property names of a species in the iBOSS corresponds to the requested ones by the FishMet: absorption_ratio, stomach_capacity, midgut_volume, ingestion_delay, water_uptake, water_uptake_a, water_uptake_r, digestion_delay, midgut_maxdur, transport_pattern_t, transport_pattern_r, appetite_factor_a, appetite_factor_r, midgut_michaelis_r_max, midgut_michaelis_k.

  A species in the iBOSS may be related to the Fish Population entity via the relationship “hasFishPopulation”.
- **Fish Population** Entity models the current cultured fishes in a fish Containment. Attributes of this entity in Table 9.

Table 9: Fish Population Attributes

Attributes	Type/value	Sub-Attributes	Comments
id	urn:ngsi-ld:FishPopulation:01		Used as a unique identifier of the entity.
type	FishPopulation		
culturedIn	Relationship	observedAt	A Relationship to link the current fish population to the correspondent fish containment entity, with the date of realization.
refSpecie	Relationship		A Relationship to link the current fish population to the correspondent fish species entity.
initialNumber	Property		The initial number of cultured fishes with date
fishRemoved	Property	unitCode observedAt	The number of removed fishes (dead), with date of observation and unit code.
body_masse	Property		The average body mass of the selected fish population
context	https://raw.githubusercontent.com/easy-global-market/ngsild-api-data-models/master/aquac/jsonld-contexts/aquac-compound.jsonld		

The overall graphical model of the Descriptive Twin Layer with an example of one (1) fish containment is depicted in Figure 24.

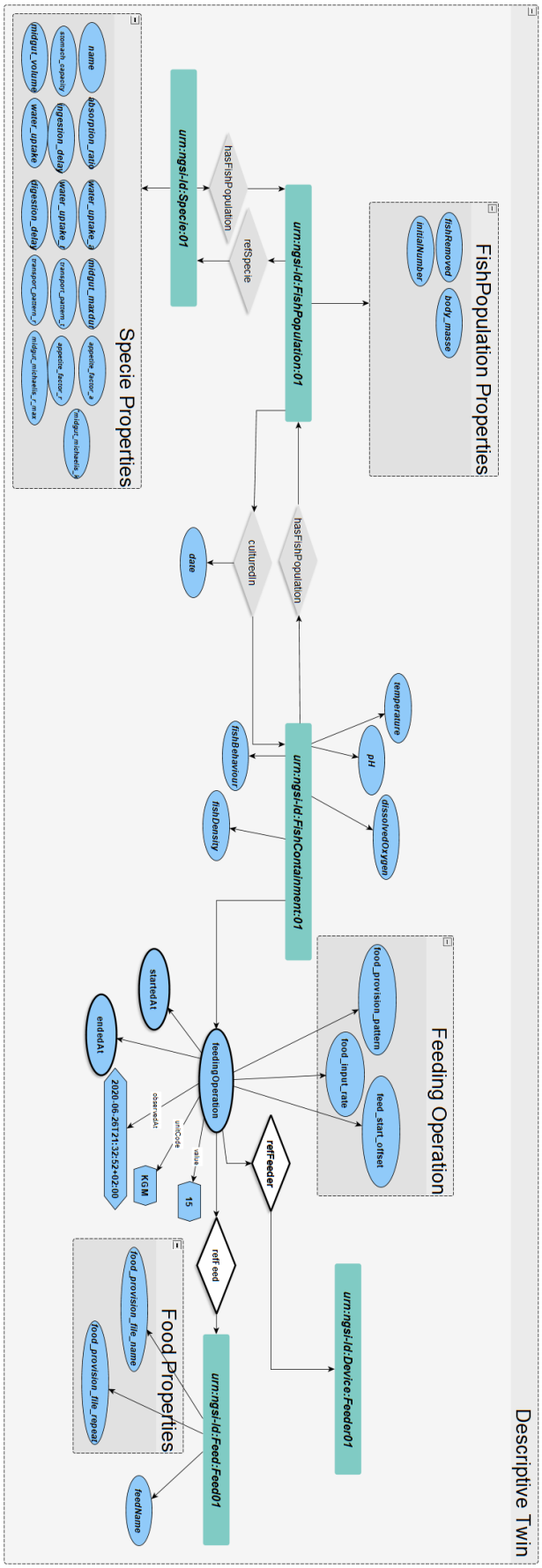


Figure 24: Descriptive Twin Data Model

3.4.1.2 Predictive Twin

When the FishMet DT is executed for prediction, a new NGSI-LD entity of type (FishMet) is created on the iBOSS. Details of the execution of FishMet are modelled as properties: configuration_version, run_model_hours, interface_graphical, daytime_hours and rate_interval.

The FishMet entity is related to the correspondent entities fishPopulation and FishContainment (of Descriptive Twin level) via Relationships “refFishPopulation” and “refFishContainement”. All these Relationships and properties will be exploited by the simulation wrapper (see section 3.4.1.4) which will collect all these data for the execution of the FishMet.

The model outputs (prediction) are stored in a new entity of type Fish Population with a “prediction” tag. It is related to the correspondent Fish Population (of the Descriptive Twin Level) via the relationship “hasInputFishPopulation”. The FishMet entity is related to the result entity via the Relationship “refPredictedFishpPopulation”.

The overall graphical model of the Predictive Twin Layer depicted in Figure 25.

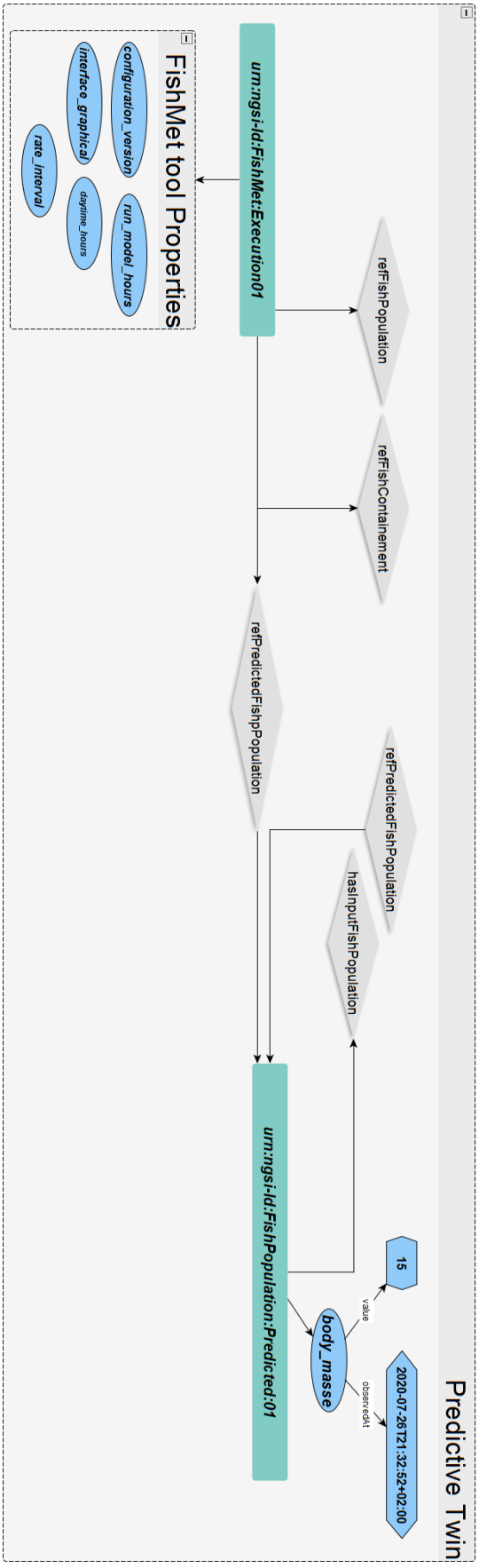


Figure 25: Predictive Twin Data Model

3.4.1.3 Design Twin

The FishMet DT integration in the iBOSS, offers the possibility of running new scenarios on a specific state of the whole system. “What if” scenarios consist of modifying selected inputs of the current fish containment or fish population entities for simulating new conditions and assessing the impact of the fish behaviour.

To enhance such feature, for each new what-if scenario, an NGSI-LD entity of type FishMet is created on the iBOSS. This entity will contain execution parameters of the FishMet as in the predictive twin layer and all the list of desired modified input parameters and related to their target entity. Following the same principles, a simulation result entity will be created to each what-if scenario, in which all output parameters are stored as properties and related to their target entity.

The overall graphical model of the Design Twin Layer depicted in Figure 26.

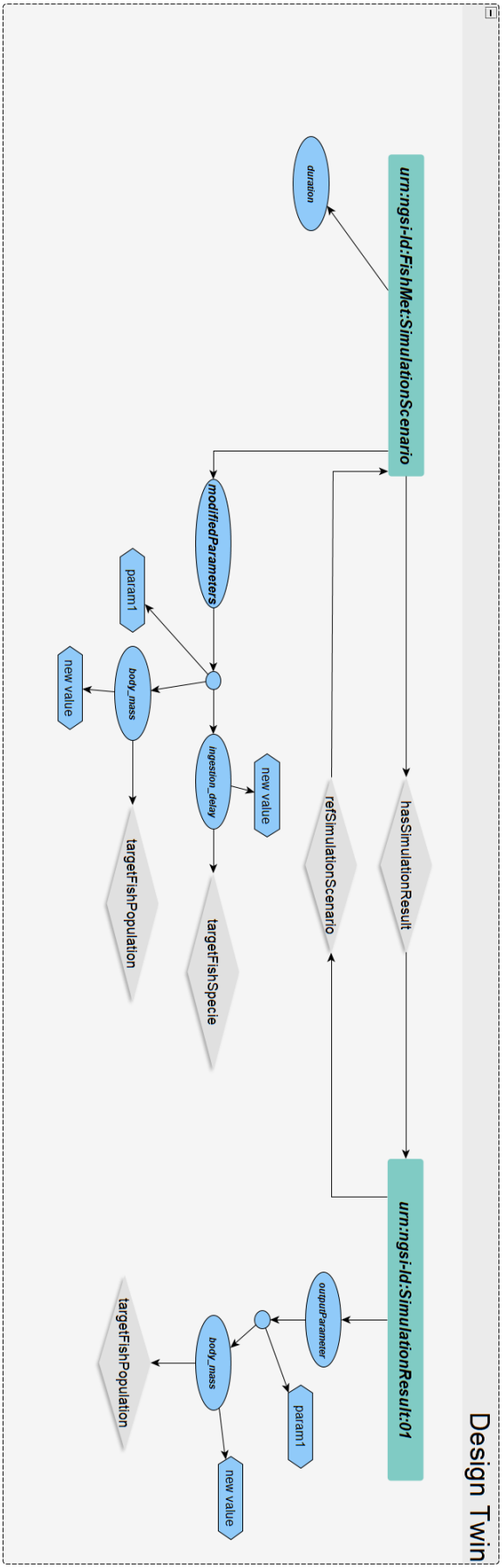


Figure 26: Design Twin Data model

3.4.1.4 *Simulator Wrappers*

The Simulation wrappers have the responsibility to:

- Collect data from the iBOSS
- Format Data for the FishMet tool input
- Execute the FishMet

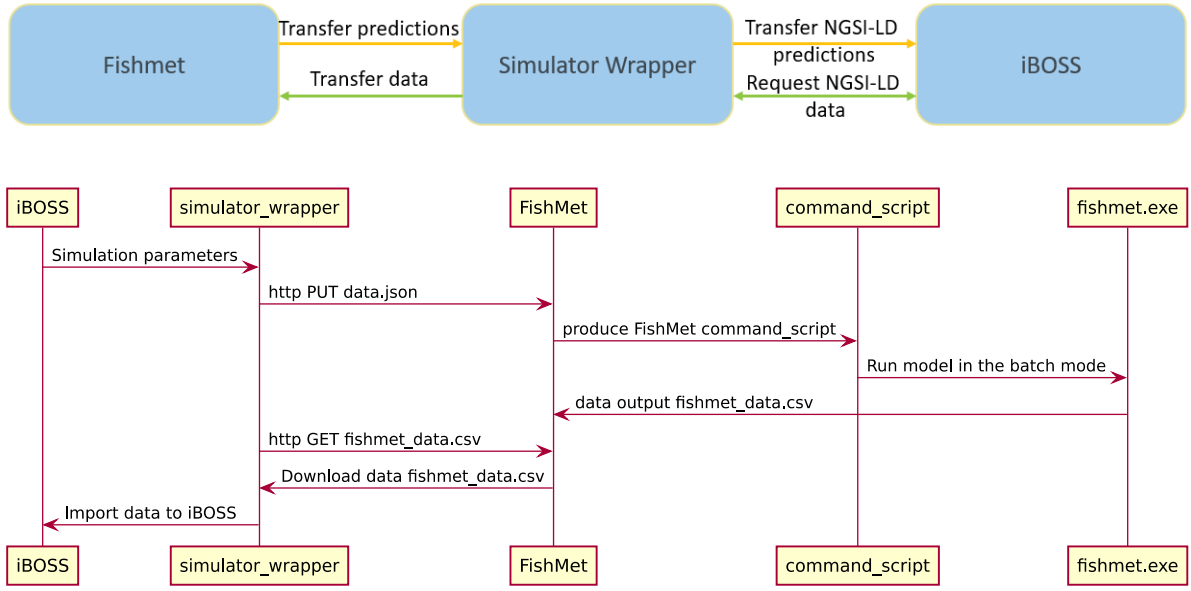


Figure 27: connection between iBOSS and Fishmet

The simulator wrapper connects the FishMet model over the network: to the FishMet computation server that runs the FishMet model software as well as connection and model execution wrapper. All the interactions are initiated on-demand by iBOSS. First, input data (JSON format) is transferred from iBOSS simulator wrapper to the FishMet server via a https request. This triggers the FishMet connector which will (1) generate FishMet command script based on the input parameters (the FishMet command script defines the modelling), (2) run the FishMet executable with the Fishmet command script, the model output is saved to a data share location. Once the simulation is finished and the FishMet output data are available in the data share location, the output data (CSV format) can be downloaded back to iBOSS using the authenticated https request. The system implements error correction (e.g., empty data as a result of failed network transaction). Any FishMet parameters can be encoded and transmitted from iBOSS to FishMet, which means that any possible model scenarios can be simulated. The output data can be obtained over the network in a variety of ways both automated and manual.

4 AI based feeding control

The AI Control of the iBOSS Smart feeding system is based on an automated decision-making system (ADMS) using water quality data, behavioural data and predictive data, as input. An AI algorithm trained for providing the capability of determine a single decision to either reduce feeding, increase feeding, stop feeding or keep on distributing the planned amount of feed. The decision is transferred to the feeder, through the iBOSS interface, and thereby controlling the feeding conditions and optimizing the feeding to obtain the best feed conversion rate (FCR).

The ADMS will be able to adapt and optimize its decision-making continuously, as the changes to the system, resulting from the feeding operation, affects the input data. Effectively creating a feedback loop.

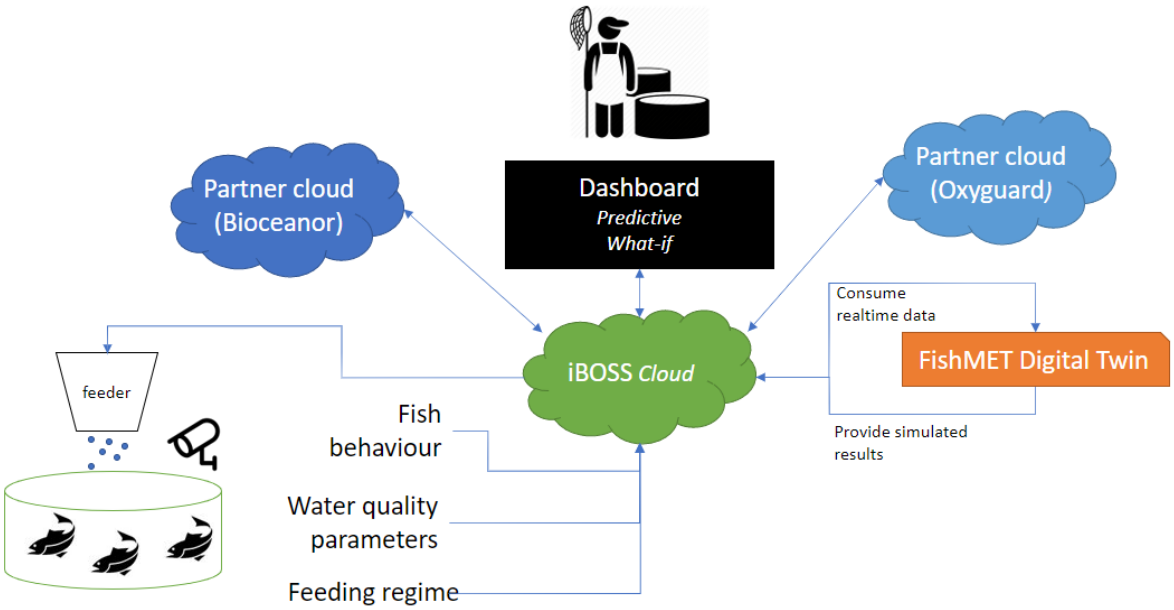


Figure 28: Feedback loop for the ADMS

4.1 Feeding control based on fish behavioural

A camera based (described in section 3.1) or an echo sounder based (described in section 3.3) system can be installed to supply feedback on the behaviour of the fishes. From this behaviour, an indication of the hunger can be extracted and fed in to the ADMS.

Based on the fish monitoring systems developed at the HCMR lab (see section 3.2), we have planned a series of experiments complementary to each other in order to assess movement variations related to feeding and define hunger and satiation levels in European seabass. The results of the analysis of these experiments will be used to estimate satiation/hunger thresholds as inputs to the automated feeders. The experiments are already under implementation and take place at the pilot-scale farm at Souda, Crete.

To better understand the whole spectrum of behaviours related to feeding we will investigate the effect of the feeding mode, rate, duration and quantity on the swimming activity of the European seabass. More specifically, we will examine four different scenarios.

In the first experiment, we will study the effect of the manual and automatic feeding on the swimming behaviour of the European seabass. The fish will be fed manually once daily for a period and then changed to feeding realized once daily by an automatic feeder. The results will help us understand if the human presence/absence affects the swimming performance of the fish during feeding, or the observed behaviour is totally explained by their hunger state and feeding per se.

In a second trial, we will investigate the effect of the feeding frequency on the swimming behaviour of European seabass. For this the fish swimming activity will be record in days with no feeding and will be compared with periods of different feeding rates, i.e., once daily, or twice daily.

In the third experiment, we will investigate the effect of the variation of feeding quantity on the behaviour of the fish. For this, we will test how fish respond under reduced and excess feeding.

Due to the long duration of the trials, we anticipate two problems that we will need to find solutions for. The size of the fish and the temperature of the environment are important factors that partially determine the feed quantity needed by the fish. These factors will change during the experiments. For this reason, by using DEB models developed in the lab, we can predict the difference in feed demand for the different sizes and temperatures and we can incorporate this model-based information in the experiments and the analysis in order to have comparable results.

The results of all the experiments will be combined and analysed in order to define baseline behaviours, estimate hunger and satiation threshold and will then be used as an input to automatically control the start and the end of the feeders.

4.2 Feeding control based on water quality parameters

Some water quality parameters are well known to influence the feed uptake by fish (e.g., temperature, dissolved oxygen), other parameters may be related as well. Even a combination of parameters may lead to suboptimal feed utilization. The water quality parameter-based input data is being collected in the cloud for all trails and are planned for the upcoming WP3 demonstrations.

4.3 Feeding control based on FishMet

The FishMet Digital Twin (described in section 3.4) will supply the ADMS with several prediction of the state of satiety and feeding efficiency, by simulating the fish digestion.

The main part of the development of the FishMet models are part of Task 1.4 and the status of the FishMet model is thoroughly described in deliverable D1.4

The FistMet server integration is detailed in Appendix A

4.4 ADMS integration and feeding optimization

By collecting data from several observations (water quality, behaviour, etc.), in combination with feeding performance data (i.e., FCR), it is possible to determine correlations between all collected parameters and the feeding performance.

At present, feeding optimization in iBOSS is focusing on fish behaviour as well as the FishMet predictions as the predominate source of information on when to feed and when to stop. As more trails and demonstrations are performed within the project, the more precise the optimization will become.

At the end of the project, it is expected that all parts of the iBOSS smart feeding will work together to predict:

1) When and how much to feed
2) The growth over time
3) When the fish will be fully grown

All giving the fish farmer unparallel insight for cost optimization and planning.

5 Feeder control description

5.1 Feeder control for ABT

At ABT, the feeders are controlled by an OxyGuard Commander Feeder Control system, but the basic features are the same for common feeder control system. The controller has a local computational unit for determining how much feed must be given and when to give it. The controller is configured through the accompanying PC program, that can adjust all the parameters in the controller.

The amount of feed dispensed and the feeding pattern is set by the user to, for example, suit the needs of the specific fish species being fed. The feeding pattern determines how much of the total amount of feed is distributed at each time slots (15min timeslots are used in the OxyGuard Commander Feeder Control). By changing the pattern, the user can choose if the whole amount shall be fed intensely at one or two times a day, or should it be fed in small portions all through the day. The controller then, completely automatically, doses feed, taking both fish growth and the water quality (e.g., DO, temperature) into consideration.

Figure 29: Example from the feeder control configuration program

At the end of the day, the actual amount of feed dispensed during the day, is calculated by the controller by looking at how long the feeder has been turned on and the feed rate (grams of feed per time unit). The value is then used to calculate the growth of the fish from an FCR table. Tomorrows planned feed amount is then calculated taking in to account the newly estimated size of the fish. And the cycle restarts.

The calculations are all made from predetermined lookup tables. A default set of tables are provided by the software package but can be changed by the user, as feed producers may supply tables to specific feed blends. With iBOSS these can be continuously changed and perfected for better accuracy and performance, by looking at many additional parameters being fed back from the actual system.

5.1.1 iBOSS Control

The iBOSS combines information from multiple sources to optimize the amount of feed distributed to the fishes. For this information to take effect on the physical feeders, the iBOSS will need to manipulate the feeders in either direction (feed more or feed less).

In the OxyGuard Commander Feeder Control, an “Activity factor” is designed specifically for the purpose of adjusting the amount of feed in accordance to an intelligent observation of the fish. Increasing the activity factor, will distribute more feed and vice versa. Effectively, changing the active to inactive ratio (or duty cycle) of the feeder (See section 0 for description of Pulse-Width Modulation). Traditionally, this intelligence has been in the form of a human decision, but with the use of AI, specific model and appropriated data collection (sensors, cameras, etc.), the same knowledge has been transferred to the iBOSS Artificial Intelligence system.

The same software that pushes data from the site to the iBOSS, can retrieve data from iBOSS to manipulate the feeder’s activity factor.

5.1.2 Physical connection

The controller operates the feeder using Pulse-Width Modulation (PWM) to turn on/off the power supply of the feeder.

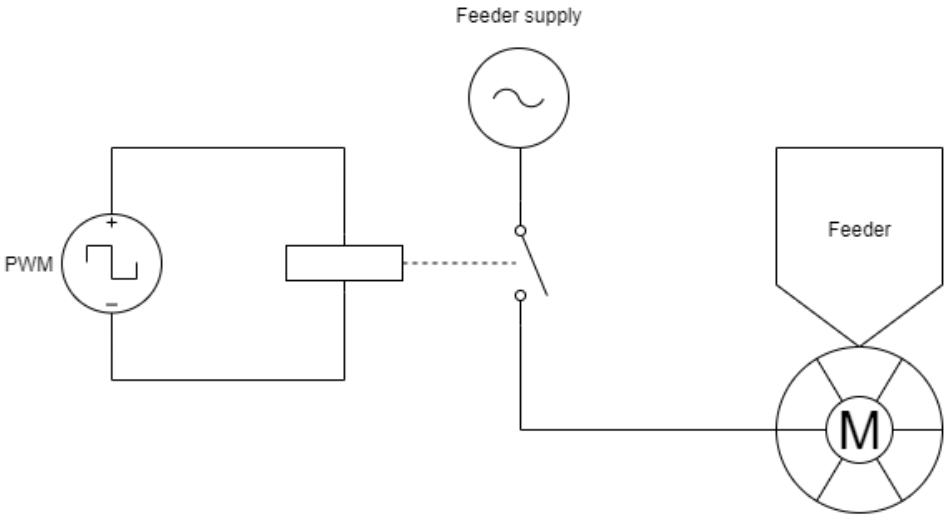


Figure 30: Diagram of feeder connection and control

The feeder will dispense feed only when the relay is engaged. By varying the duty cycle of the signal, the average feed rate is controlled. Higher duty cycles give higher feed rate (grams per minute). The actual feed rate depends on the physical design of the feeder itself.

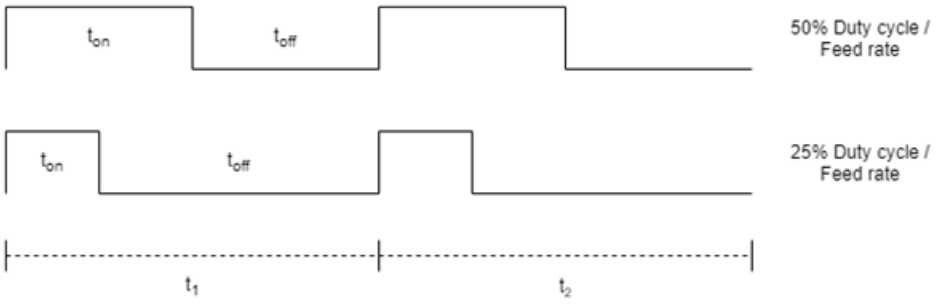


Figure 31: Examples of PWM signal for feeder control

The Feeder Controller calculates the needed duty cycle for distributing the required feed, while maintaining the chosen feeding pattern.

5.2 Feeder Control for HCMR

For the physical feeder, installed at the HCMR pilot site, to be controlled by the iBOSS, a device named System Adapter, is deployed. The System Adapter contains the programs that details when and how much the feeder shall feed. This keeps the feeder running in case of loss of internet connection. The programs are JSON objects in the following format :

```
{
    "startsAt": "feeding start time",
    "endsAt": " feeding end time ",
    "timeInterval":"indicates the frequency of feeding",
    "duration": "indicates the duration of a feeding",
}
```

The System Adapter in the HCMR pilot site is composed of a program listener that listens to new programs in iBOSS. The programs are translated into JSON objects and sends them to the MQTT broker. The Raspberry will then handle the MQTT message and commands the feeder directly, this process is called a downlink process. From the other side (the uplink process), the feeder will inquire the iBOSS about the status of the new program as well as its own status of when feeding started and ended via transmitting messages to the System Adapter.

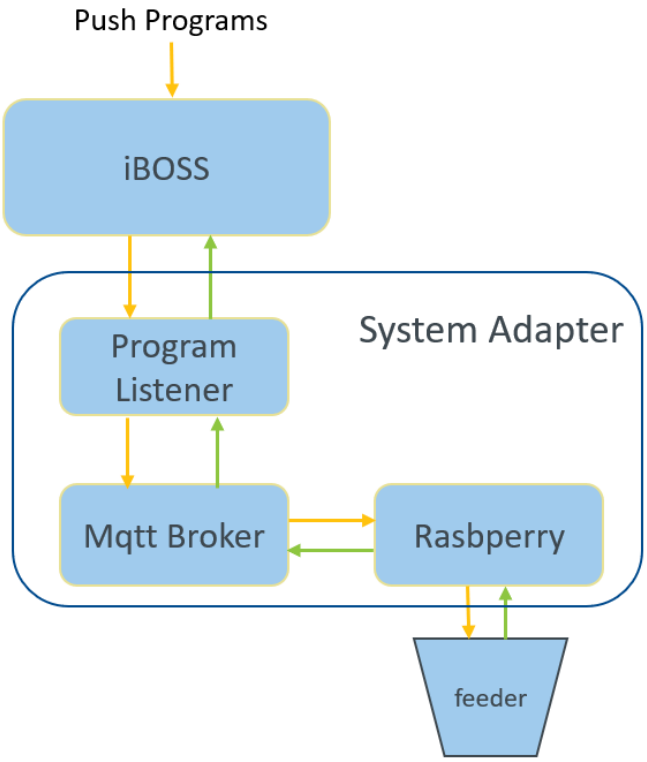


Figure 32: Feeder control in HCMR pilot site

6 Deployment perspective

The project is looking for a replicable and interoperable solution which could be easily deployed on fish farming sites with differing characteristics and equipment. Until recently, fish farming process management systems were running locally with limited connectivity. Taking manual measurements and reporting them into a .CSV file is still common practice. The progresses made with online sensors have opened new opportunities to fish farmers, which now have access to continuous monitoring of their assets. The first wave of connected sensor deployment was made using PLC (Programmable Logic Controller), which were locally collecting and presenting measurements made from the sensors, often, connected to a local computer. The last wave of deployments includes a cloud connectivity where collected data are pushed to the internet. This brings several advantages:

- **Data can be accessed from anywhere**. Sites can be monitored remotely which is of particular interest for e.g., offshore sites, but also semi-closed containment systems. Farmers can check the status and receive alerts from any computer or mobile application connected to the internet.
- **Additional services can be offered**. By connecting the data stream of the site to other information (e.g., weather or production data) new services based on (predictive) modelling can be proposed to the farmer.
- **Data sharing opportunities**. Data from the farm is often seen as private and sensitive data. However, aggregation and anonymisations functions allow cleaning the data from any sensitive information. Sharing such data with other stakeholders (farmers under similar situation, environmental authorities) can enable tackling challenges and issues (e.g., HAB, disease and parasite spreading, etc.) that are common for actors settled in the same area. It also provides additional information to the farmer such as identifying how it perform in comparison with others.

There are however difficulties in deploying such solutions. The main one encountered within the project has been:

- **Security concerns**. Cybersecurity is a growing concern for any business implementing digital solutions. To protect the farms from such a risk, one option is to provide total isolation between the farm local network and the internet. This radical option has however 2 drawbacks:
    - It prevents the farm from benefiting from other externally provided services.
    - It may provide a false feeling of security if not properly implemented, maintained and audited.

- **Internet link reliability**. The internet connectivity may have disconnections, especially in offshore sites. Such disconnections should not harm the process on the farm site thus calling for edge autonomy.

To answer these concerns, there are 2 basic options:

- **Run everything on a closed local network**: in that case all services are deployed on the farm itself. This implies that enough computing power is made available on the farm and configured in a way to ensure required levels of performance and resiliency to power supply disruption, hardware failure, etc. This option however prevents the connection to external clouds and interoperability has often to be achieved at a lower level, such as Modbus.
- **Run mix edge-cloud environment**: in that set-up some processing capabilities are executed on the edge which itself connects to the cloud. This allows managing connection disruption by providing some level of autonomy to the edge node. This autonomy includes in general:
    - Clear definition of default mechanisms, such as running a 'by-default' feeding program when no other information is made available
    - Deploying caching and synchronization capabilities so that data get stored locally during network disruption and get synchronized when connectivity is back to normal.
    - End-to-end security configuration

The latter is the one chosen for iBOSS. An edge component (iBOSS-edge) is in charge of local management of the feeder based on information it can access:

- Latest received feeding program (with a fall-back default program in case no updated feeding program has been received)
- Fish behavioural information available either locally or from the cloud, depending on the place where the video processing gets executed
- Environmental information available from the sensors. Such information is obtained from the cloud API of the sensors' providers but local connection could be obtained by integrating more closely with the locally deployed components (e.g., Oxyguard Pacific)

7 Future directions

In this deliverable, the current state (as of November 2022) of development of the iBOSS Smart Feeding AI has been presented, including descriptions of several approaches to extract information on fish state that will make it possible for iBOSS to make decisions on the feeding intensity. iBOSS will be a useful tool for fish farmers to optimize their feed utilization while improving the FCR. The iBOSS Smart Feeding AI will become self-evolving in the sense that, when the AI model is updated with new data, such as the feeding intensity, the water quality, fish behaviour, growth, etc., the AI will evolve and optimize its models, continuously.

At present state the following equipment is integrated to near or fully expected state:

- Data collection in cloud of all existing and new water quality parameters, following the open NGSI-LD interface
- Cameras deployed and computer vision and AI algorithms are implemented
- Echosounder data is being collected and data handling algorithms are implemented
- Feeder controllers are implemented

The iBOSS will keep evolving through the duration the project and continue after the project ends. In the WP3 demonstration, considerable amount of data is being collected to further improve the AI models in iBOSS. The more data collected in iBOSS, being behavioural, environmental or simulation modelled, the better the Smart Feeding decision-making capability will be. Likewise, the more species and production systems that are connected to iBOSS and using Smart Feeder, the more accurate and comprehensive the product will become.

After the project is complete, the fish farmers will be able to get a valuable tool for reducing cost of feeding and predict when the fish is will be ready for harvest.

8 Appendix A

FishMet Server: iBOSS-FishMet integration

How to use the FishMet server: Brief outline

1. The iboss_connect.sh iBOSS connector script must be running on the FishMet Server. To start, login to the server over ssh with a public key and start the service with this command:
   systemctl --user start fishmet.service
   (provided the systemd user service service has been correctly configured).
2. Model execution is triggered by posting model parameters in JSON format to https://fishmet.uib.no/data/. Example: curl --user 'user:SECRET' -T data.json https://fishmet.uib.no/data/
3. All model output data in CSV format can be downloaded by authenticated http(s) GET request or WebDAV from https://fishmet.uib.no/data/ (e.g. using curl command).

iBOSS Overview

iFishIENCi Biology Online Steering System (iBOSS) is aimed to improve production control and management for all fish aquaculture systems. Within iBOSS, smart function can be implemented to take advantage of data from multiple sources to give better insight for the farmer, optimize production, better production planning, etc. The iBOSS Smart Feeding aims at maximizing feed utilization while minimizing environmental impacts, by optimizing the efficiency of the presentation of feed to the fish in relation to fish state, behaviour, environmental conditions and species to maximize growth and reduce feed loss. Multiple AI models are being developed, each adding its own interpretation of the feeding efficiency. All interpretations are integrated in an automated decision-making system to give the feeder signal to either increase or decrease feeding.

The FishMet digital twin (DT) integration with the iBOSS is based on a layered data model approach. The descriptive twin model consists of modelling the current use case. This layer will follow the iBOSS data model of AquaBioTech Group site. It models the set of Fish containments, installed devices their properties and relationships.

FishMet Server

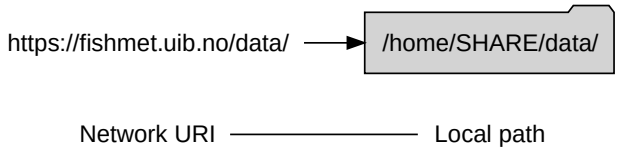
The FishMet server is a cloud-based Linux computational server that makes use of the NREC IaaS infrastructure. FishMet server is running on a Debian Linux OS and includes the following components:

- OpenSSH server for service access, software installation, update, configuration etc.
- Apache web server running WebDAV-based data access over strictly authenticated secure (https) protocol.
- FishMet model source code managed by the Apache Subversion version control system.

The FishMet server is accessible by the following URL: https://fishmet.uib.no

FishMet data share

All data exchange between FishMet and iBOSS makes use of the dedicated location on the FishMet server: the FishMet share folder (local filesystem: **/home/SHARE/data/**) that is accessible via http protocol as: https://fishmet.uib.no/data/.



FishMet Server access

To access the server, the user need to be provided with valid credentials.

- ssh access is based strictly on public key authentication. The user then must provide the FishMet server administrator with the ssh public key.
- https access for data exchange is based on password authentication. The user will be given a user name and password by the FishMet server administrator.

The https-based access to the FishMet server is possible vis standard PUT and GET requests that can be easily automated. Additionally, the data in the FishMet share folder can be accessed using any standard web browser or via any standard WebDAV client (including default clients in Microsowt Windows and Mac OSX).

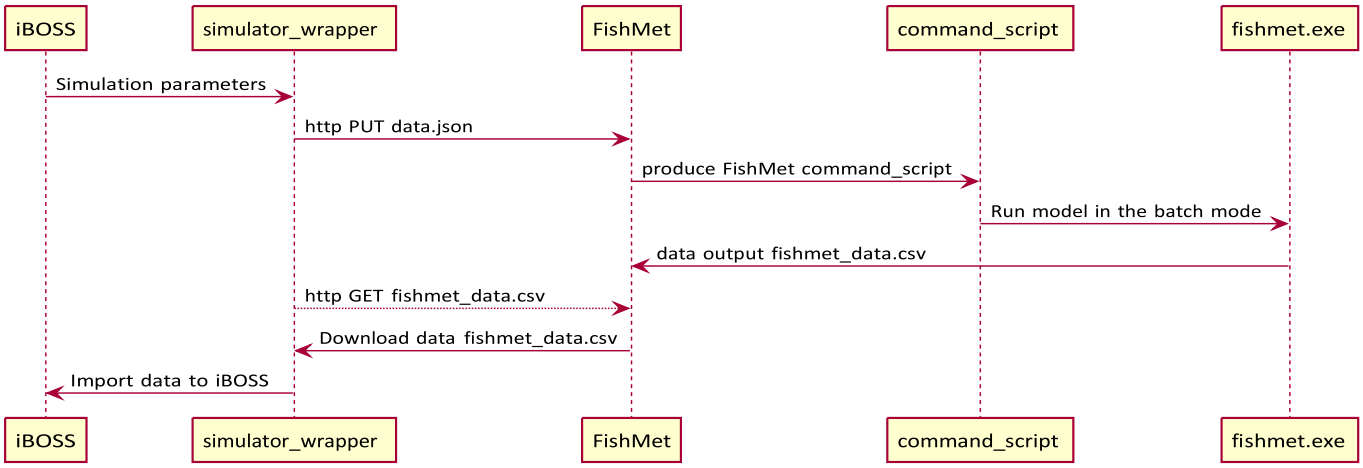
Warning
The default WebDAV client implemented in Microsoft Windows is not fully standard-compliant and may require the following URL to connect the data share folder as a "disk" (Map network drive):
\\fishmet.uib.no@SSL\data\ instead of the standard https://fishmet.uib.no/data/.

FishMet—iBOSS integration

Data produced and consumed by FishMet: FishMet accepts data on the environment conditions, the feeding protocol, various parameters of the fish morphology and physiology. The output of the model is then the pattern of the fish feeding behaviour (e.g., ingesting food items), its internal state (e.g., the level of appetite) and parameters of the gastrointestinal system functioning (e.g., stomach and gut fullness, absorption, evacuation of faeces). There is also approximations for growth, FCR, and oxygen uptake.

Data exchanged with the iBOSS cloud: FishMet can input data on characteristics of the food (e.g., size, gross energy), the feeding protocol, basic biological characteristics of the fish (including group and individual variability) as well as key environmental data (e.g. temperature). Then it will output prediction data on feed intake, activity, individual internal state (e.g., appetite) energetic status and growth.

The flow of the interaction: A common http-based API (application programming interface) was devised for interacting the iBOSS system with the FishMet model implemented on its computational server. This interface includes (1) an input JSON data structure that is sent to the FishMet server as well as (2) the FishMet output data in CSV format. The flow of the interaction between the iBOSS and FishMet is depicted below.



All the interactions are initiated on-demand by iBOSS.

- First, input JSON data file is transferred from iBOSS simulator wrapper to the FishMet server (https PUT request). This triggers generation of the model command script, which will do the following:
  - Get the JSON data from iBOSS as input parameters that will overload the default values from the model configuration file;
  - generate FishMet command script based on these input parameters; the FishMet command script defines the modelling scenario and is executed by the FishMet executable program;
  - run the FishMet executable with the Fishmet command script; when the model is run, output is saved to the appropriate data share location;
  - provide notifications about the triggered model execution, along with the FishMet command script that is generated and further notification when FishMet computation is finished. XMPP protocol is used for this messaging.
- The simulation normally takes a few minutes. Then the data that is generated by the FishMet model is saved into the data share location on the FishMet server.
- Once the simulation is finished and the FishMet output data are available in the FishMet share folder, the output data can be downloaded using the authenticated https GET request to the FishMet server.

See the Implementation section for other details of the above procedure.

Data API

Input JSON data

The input JSON data can provide any of the FishMet model parameters that are defined in the FishMet parameter file.

Basic requirements:

- The data should be posted to the FishMet data share URI: https://fishmet.uib.no/data/.
- The name of the data file is **data.json**.

To encode the following FishMet parameters:

```
food_provision_file_name = "/home/SHARE/data/food_exp2.csv"
food_input_rate = 20.0 temperature = 16.0 body_mass = 100.0
food_item_mass = 0.045
```

One needs to post the following JSON data structure to the server:

```
{
    "food_provision_file_name": "/home/SHARE/data/food_exp2.csv",
    "food_item_mass": 0.045,
    "temperature": 16.0,
    "body_mass": 100.0,
    "food_input_rate": 20.0
}
```

It is easy to encode such data using the following Python code:

```
# Import JSON module import json

# Dictionary data structure that defined the data data = {
        "body_mass": 100.0,
        "temperature": 16.0,
        "food_provision_file_name": "/home/SHARE/data/food_exp2.csv",
        "food_input_rate": 20.0,
        "food_item_mass": 0.045


}

# Show the data on the terminal for manual control json.dumps(data)

# Save the JSON data structure in the output file data.json with open("data.json", "w") as
write_file:
        json.dump(data, write_file)
```

Other (CSV) input data

Several other input data files (usually CSV format) can be used by the FishMet model and need to be therefore transmitted to the FishMet server. This is done in the same way as the data.json, e.g. via the https PUT. For example, an arbitrary feed schedule can be defined by the data file defined by the food_provision_file_name configuration parameter. The path name to these additional data files should refer to the data share location on the server: **/home/SHARE/data/**.

Model execution triggering

HTTP(s) PUT request

To execute the FishMet model with these parameters, the iBOSS simulator wrapper needs to post the above format data.json data file to the FishMet share folder URI: https://fishmet.uib.no/data/.

An example of this command using the simple curl command line program is given below:

```
curl --user 'user:SECRET' -T data.json                    https://fishmet.uib.no/data/
```

Here user:SECRET should be substituted with the appropriate authentication data.

Another example below uses the Python requests library:

```
# Import requests library import requests

# Import getpass library needed to ask password; note that it is
# not mandatory because password can be provided directly in the tuple
# as auth=('user','secret') from getpass
import getpass

# Send PUT request to the FishMet server; note that the password
# will be requested using getpass() method requests.put(
        'https://fishmet.uib.no/data/data.json', auth=('user', getpass()),
        data={'food_provision_file_name': '/home/SHARE/data/food_exp2.csv',
                    'food_item_mass': 0.045,
                    'temperature': 16.0,
                    'body_mass': 100.0,
                    'food_input_rate': 20.0}
)
```

Alternative methods

Because the implementation is based on inotify mechanism, FishMet model execution can be triggered by any change of the data.json file, such as uploading it via the sftp/scp protocol or even local file copying on the server.

An example below shows how to transfer data using the sftp/scp protocol:

```
scp data.json user@fishmet.uib.no:/home/SHARE/data/
```

Warning

There may be one caveat when using the different data transfer methods interchangeably because of the different local users acting on the FishMet server. When the **data.json** data file is transferred via the **http(s)** PUT request, the file is saved on the server as owned by the web server (owner: www-data.www-data). But when the data file is transferred over **sftp/scp**, the file is owned by the local user who logged in for transfer (in the example above, user.user). Therefore a **permission conflict** can occur, so that access to the existing data file may be denied, making it impossible to overwrite the data.json file that belongs to a different user. To solve this, the owner of the data.json file should delete the old data while the iboss_connect.sh is **not running** (otherwise the inotify process will recreate an empty data file data.json). **It is recommended to use the http(s) PUT request mechanism for data transfer.**

Prexisting FishMet command script

It is possible to pass an existing FishMet command script rather than a set of FishMet model parameters. This can be done by providing the file name of the command script in the data.json file as the **fishmet_commands** parameter. But note that this should be the only data in the file. Thus, the data.json file should look like this:

```
{ "fishmet_commands": "/home/SHARE/data/command_script.cmd" }
```

Warning
If the command script file referenced in the `fishmet commands` parameter is not found, the FishMet executable just runs an empty script and does not produce any data or statistics.

To send the command script referenced in the fishmet_commands one can use the same methods as for the data.json data, i.e. HTTP(s) PUT request, such as this:

```
curl --user 'user:SECRET' -T command_script.cmd                    https://fishmet.uib.no/data/
```

Output CSV data

The FishMet output data is a CSV format file that includes all changeable input parameters of the model as well as all output data values as columns. The rows represent separate model runs. The size of the data file can be large.

The initial part of the file reporting model input parameters:

run_model_hours	daytime_hours	temperature	body_mass	stomach_capacity
24	16	16	100	4
24	16	16	100	4

And here is en excerpt from the part of the output data file that reports the output predicted values:

MASS_INGESTED	ABSORP_CUMUL	ENERGY_FOOD	BODY_MASS_STA	TBODY_MASS_END
5.85	0.00	0.00	100.00	98.99
11.79	5.25	120.796	98.99	108.29

Here is an example of downloading the model output using the simple curl command:

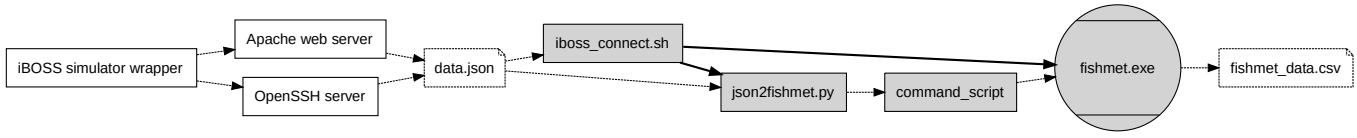
```
curl --user 'user:SECRET' https://fishmet.uib.no/data/fishmet_stats.csv --output fishmet_stats.csv      ←-
```

Implementation

The FishMet—iBOSS integration is implemented in the following components:

- **iboss_connect.sh**: iBOSS connector script. This is a POSIX shell script that runs on the server in an infinite loop. Any modification of the input JSON data is monitored by an inotify-based mechanism that triggers all subsequent actions: generation of the FishMet command script by json2fishmet.py, its execution by the main program executable (fishmet.exe), data output to the data share location and all appropriate notifications.
- **json2fishmet.py**: data import script. This is a Python program that imports the input JSON data and produces the FishMet command script that is executed by the FishMet executable on the server.

The diagram below depicts the call graph and data flow scheme that is implemented in the FishMet—iBOSS integration components. Here bold arrows depict program calls and dashed arrows show data flow directions. FishMet—iBOSS integration components as well as the model executable are filled.



Error correction: these scripts implement elementary error correction for the most typical possible errors, such as empty data.json data file as a result of failed transaction.

Further adaptation: The json2fishmet.py script can be modified to add fixed parameters that cannot be changed from the input data.json data, add commands to produce additional statistics, outputs and plots. All these data are saved into the FishMet share folder. All script changes can be deployed on the FishMet server directly via the Subversion version control or by standard https GET request from the Subversion server.

Defining model root: The environment variable FFA_MODEL_ROOT should be used to set the local folder on the FishMet server where the model executable and the model parameters file are stored. If this variable is not set, the model will run in the current directory.

```
export FFA_MODEL_ROOT=/home/debian/DEV/STOMACH
```

Defining output destination for FishMet executable: The FishMet executable program fishmet.exe that runs on the server must be able to get the input data and save the outputs into the FishMet share folder. This can be done either via the output_dest parameter in the model parameter file

```
output_dest = "/home/SHARE/data/"
```

This parameter can be set using the input data.json data structure:

```
{ ....
    "output_dest": "/home/SHARE/data/", ....
}
```

or by setting the FFA_MODEL_OUTPUT_DEST environment variable on the FishMet server:

```
export FFA_MODEL_OUTPUT_DEST=/home/SHARE/data/
```

The simplest and most reliable method to define the model root directory and the output destination correctly is to code it into the .bashrc or .bash_profile on the server. But this works only when the the iBOSS connector script (iboss_connect.sh) is run manually. If it is run as a service, all environment variables are coded into the fishmet.service file.

Model execution as a service

The iBOSS—FishMet connector script (accepting the data over network and triggering model execution) can be started manually by the user whio has logged in to the Fishmet server by ssh. Then, it is advisable to run the model executable from within the GNU screen or tmux session so the user can log out from the server leaving the executable running.

However, it is advisable to run the iBOSS—FishMet connector script as a systemd user service. A service configuration file **fishmet.service** is a part of the iBOSS—FishMet server software suite.

- The systemd configuration file fishmet.service should be placed into the user's ~/.config/systemd/user/ directory.

The fishmet.service configuration should also define the FFA_MODEL_ROOT environment variable that points to the location on the FishMet server filesystem where the model executable and the

model configuration file are stored. This variable is set in the [Service] section of the fishmet.service file.

To enable the fishmet.service service to start automatically at the server (re)boot, issue the following command:

```
systemctl --user enable fishmet.service
```

To start the fishmet.service service, issue this command:

```
systemctl --user start fishmet.service
```

To stop the fishmet.service service, issue this command:

```
systemctl --user stop fishmet.service
```

Note that starting the service manually does not require enable, but it will not be persistent on reboots in such a case. In many applications, though, it may actually be the desired configuration: enable to run the model as long as it is needed and stop afterwards.

To check the status of the fishmet.service service, do this:

```
systemctl --user status fishmet.service
```

To show the log messages that produced by the fishmet.service service to the system journal, do this:

```
journalctl -e --user-unit=fishmet.service
```

To monitor the log continuously, add -f option (follow).

Conclusion

The iBOSS—FishMet integration is well adapted for flexible on-demand execution. Any kinds of model parameters can be changed and posted from the iBOSS simulator wrapper.