HardwareX 11 (2022) e00283

Contents lists available at ScienceDirect

HardwareX

journal homepage: www.elsevier.com/locate/ohx

ADO: An open digital end-to-end tank based aquaculture platform

Ioana Suciu [a,*], Guillem Boquet [a], Pere Tuset-Peiró [a], Xavier Vilajosana [a]

[a] Wireless Networks (WiNe) Research Lab, Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC), Spain

ARTICLE INFO

Article history:

Keywords:
Aquaculture
Fisheries
End-to-end solution
Open source
Digitization platform
Water sensors

ABSTRACT

The ADO project proposes the development of an IoT solution that allows the digitization of the aquaculture sector, developing the basic elements (data acquisition systems, data storage system and visualization platform) in open source format. Hence, ADO makes it easier for small and medium-sized producers to obtain success stories with limited technology background and smaller economic investments. In this article we provide a comprehensive description of the platform building blocks, including the hardware elements, integration procedures and structure, and operation details of the back-end infrastructure.

Specifications table:

Hardware name	*ADO*
Subject area	*Environmental, Planetary and Agricultural and Fisheries Sciences*
Hardware type	*Field measurements and sensors*
Closest commercial analog	*The Waka water monitoring system created by RiverWatch: https://riverwatchsolutions.com/solution/the-waka-hardware*
Open source licence	*ADO software uses BSD 3-clause licence. ADO hardware uses CERN OHL v2 Open Hardware Licence.*
Cost of hardware	*570€* – including 10 sensors
Source file repository	*https://doi.org/10.17632/cvdc3gsjsg.1* *ADO*

1. System in context

Aquaculture is an industrial activity that focuses on the cultivation of either freshwater or saltwater aquatic organisms that have commercial value (i.e., crustaceans, molluscs, algae and plants). Although aquaculture has become more technified over the past decades, it is still far from being as digitalized as other agri-food industries in the world. For instance, the food industry has incorporated digital technologies, such as robotics, big data and artificial intelligence, among others, to meet the chaining requirements of both suppliers and customers, as well as the volatility of the markets. One of the main reasons behind the lack of digitalization of aquaculture production is the highly specialized requirements and the low volumes of

* Corresponnding author.
E-mail address: isuciu0@uoc.edu (I. Suciu).

https://doi.org/10.1016/j.ohx.2022.e00283
2468-0672/

a niche sector, leading to high product costs that make the adoption of such digital technologies economically viable only for large-scale factories. For example, a data acquisition system for a turbidity sensor can cost around €4,500 (€3,000 for the datalogger[1] and €1,500 for the sensor[2]).

In contrast, a solution developed with open-source tools and the integration of existing general purpose hardware and software platforms could cost less than €600 while including different sensors. However, there are cost-performance trade-offs when using a cheaper alternative. The technical specifications of the sensors used in ADO are listed in Table 1. With the information provided by this table, a user can know if ADO is a suitable solution for its own application or if its application requires more specialized sensors.

In the literature, there are multiple examples of relative cheap approaches for aquaculture and water quality monitoring, which we treat in detail in Section 2. In this article we present ADO, an end-to-end, open and low-cost system aimed at supporting the digitization of the aquaculture sector. From the best of our knowledge, this is the most complete open-source work in the related field, fully describing its building blocks, with step by step procedures to setup the system. ADO is composed of three main elements: hardware platform, communication technologies/protocols, and a back-end system. First, the hardware platform is based on a custom PCB that allows to integrate off-the-shelf sensors and other electronic and computing components, and runs a software that is responsible for the data acquisition and processing. Second, the communication technologies and protocols are responsible for the transmission of the data from the hardware platform to the back-end system over the Internet. Finally, the back-end system, which can be hosted on-premises or on the cloud, is used to manage the devices, store the generated data, visualize it and define operational alarms.

The design philosophy behind the ADO project is twofold. Regarding the hardware platform, communication technologies/protocols and the back-end system, we have built upon open-source projects/frameworks and standardized technologies/protocols, thus achieving low hardware and software cost, while maximizing inter-operability, availability and support. Regarding the users, we have designed the system following a plug-and-play philosophy that aims to simplify the device commissioning, setup process and daily operation. This end-to-end system approach is the key to ensure that the aquaculture sector can benefit from the open-source solution, limiting the requirements imposed to the sector to adopt a digital solution.

Overall, the resulting end-to-end system is a low-cost, plug-and-play and real-time asset management and visualization framework customized to the aquaculture sector that is ready for production environments. Moreover, the resulting system is in line with the needs identified by the European Maritime and Fisheries Fund (EMFF), and it aims to encourage the collection and harmonization of available data, avoiding duplication and responding to the need for interoperability with other technologies in the sector or even in related sectors. Still, while the ADO software applies to any aquaculture vertical, the casing design proposed in this article is mainly centered on tank-based aquaculture and any adaptation to some other environment should be done while reconsidering the ingress protection of the device and its power mode.

The validation of the ADO platform has been conducted during the spring, summer and autumn of 2021 and has consisted in a 8-months pilot deployment in an aquaculture educational center located in the Delta del Ebre, Tarragona, Spain. The deployment of the system enabled to validate the simplicity of its deployment, validation of the sensor calibration procedures, obtain feedback from the users of the system and validate the entire data pipeline, from the sensor to the back-end server and visualization.

2. Related work

The work of Parra et al. in [3] presents a water monitoring system for fish farms, where data from the nodes is sent with WiFi to an access point that has Ethernet connection. The nodes can monitor parameters such as: illumination, water level, oil level, temperature, turbidity, conductivity and humidity. Moreover, there is a feed fall detector, presence sensors and fish behavior sensors. For the behaviour of the fish, they use light dependent resistors to monitor the swimming velocities and its depth, strategy that works only when the halogens are turned on, and not at night. Their work focuses on the calibration of the various sensors they use and the total cost of a node is less than €90. However, the resulting node is only a proof of concept and it cannot be deployed as it is, as the sensors are basic electrical components with no water protection. There is no schematic and no performance evaluation compared to off-the-shelf sensors.

Saha et al. present a water quality monitoring system [4] with low-cost off-the-shelf elements and pH, electrical conductivity, temperature and water colour sensors. They use Raspberry Pi 3 and Arduino Uno for building the node, and the provider for the sensors is DFRobot. The data from the sensors is sent with WiFi and stored locally on the Raspberry Pi in a MySQL database, and it is extracted by a web page (via an Android app) in order to be displayed in numeric format. Their approach stays at proof-of-concept level, without the option for remote access, multiple node management or complex graphics or alarms. Even if not applied to tank based aquaculture, but to monitor the water quality in Mekong Delta, we mention the work of Danh et al. [5], that proposes an end-to-end aquaculture monitoring system. The data from the nodes is transmitted in the 433 MHz band with LoRa to a control unit, published to a server and accessible with a mobile app. The monitored parameters are dissolved oxygen, temperature, salinity and pH and the entire system was tested in a 5 node deployment. The information on their hardware and software is not released and, thus, it cannot be reproduced.

Chandanapalli et al. present the implementation of a low cost water monitoring system [6]. However, the authors do not provide information regarding the obtained costs, nor access to the PCB design. Their system has four sensors attached, to

monitor the water and atmospheric temperature, humidity and pH, and data is sent using the GSM standard. In [7], Qayyum et al. present a solution based on the use of off the shelf elements to build a monitoring system for tank based aquaculture. Their solution relies on MoleNet, a hardware based on the Arduino platform, a 433 MHz RFC69 radio transceiver, pH, DO, salinity and temperature sensors. Data is sent periodically by the Arduino to a Raspberry Pi server. The information on the total cost of the system, the PCB and the code is not released. Finally, the work of Encinas et al. focuses on providing a proof of concept and a prototype of a distributed IoT system to monitor the water quality by measuring the temperature, pH and dissolved oxygen [8]. Their solution relies on Arduino and ZigBee modules for sensors nodes, which send data to a computer, further stored in a MySQL database and published to a web app as numerical values.

This section reviewed the most relevant, prototype-centered publications in the literature. We identify a general architecture, with the same building blocks: sensor node, server or local data storage and web page/app for data display. The variations are in terms of communication protocol, user interface design, number of sensors or final price. Our work contributes with a solution that supports a number of maximum 10 sensors that can be even replaced with other electrically compatible sensors to better fit a specific application. The software is already built to support multi-tenancy, secure connections, built in dashboards, data download possibility and real time control and calibration of the sensors. This paper provides a complete setup guide, with information regarding the final price and specifications. The resulting node is ready to be deployed and the back-end system is currently available at https://ado.wine-lab.org/ to support early adopters.

3. System description

As introduced earlier, ADO is an end-to-end system intended for monitoring and controlling water quality, allowing to take knowledge-based decisions for quality aquaculture. The ADO project is composed of three different subsystems: hardware platform, communication technologies/protocols, and back-end system. In this section, we describe these subsystems.

3.1. Hardware platform

The hardware platform is responsible for the acquisition and the pre-processing of the data from the sensors. It is divided into the electronics components and the firmware operation.

3.1.1. Electronic components

The PCB of the ADO node is depicted in Fig. 1. The main components of the electronics are:

- a maximum of 10 sensors. The supported sensor destination (e.g., ambient temperature), operating voltage (e.g., 3.3 V or 5 V) and communication protocol (e.g., I2C, 1-Wire, analog, digital) are shown in the block diagram of the hardware (Fig. 2).
- an Arduino MKR series (e.g., MRK1000, MKR1100 or MKR1300) in charge of collecting the sensor data, with a number of 7 analog input pins (8/10/12 bit) and 8 digital pins.
- a Raspberry Pi (RPI), in charge of controlling the Arduino unit: scheduling data collection for individual sensors, managing sensor calibration and publishing data securely to the server.

As it can be observed, all these elements are connected to a single control PCB (Fig. 1), which is in charge of interconnecting the devices, providing the required connectors to the actual sensors and offering a flexible power source control. The actual pin mapping between the main components is represented in Fig. 3.

The ADO node can be powered by a 12 V battery or directly from a 220VAC power outlet via a 12 V, 60 W AC/DC converter. The power source can be selected by using a switch. Regardless of the origin, the 12VDC input voltage is later converted to 5 V and 3.3 V to be used by the sensors, embedded systems, and the remaining electronic modules of the board. The sensors are all Arduino-compatible and come with their own integration boards, which are screwed in the sensor placeholders of the ADO PCB. The sensors are connected to the power supply and to the Arduino input pins (i.e., analog, digital and I2C). For the analog sensors, Arduino has built-in analog-to-digital converters (ADCs), so that there was no need to add ADC modules to the ADO PCB. Table 1 provides a summary of the technical specifications[1] of the sensors used in ADO.

3.1.2. Firmware operation

As stated in the previous subsection, the electronics include two main functional subsystems. On one hand, we have an embedded microcontroller unit integrated into an Arduino module. This unit has the responsibility to handle the configuration and sampling of the different sensors and to serialize the information to the node main CPU upon a request. On the other hand, the RPI module is responsible for handling the sampling period and requesting data to the Arduino module according to a given configuration. It is also responsible for the connection to the network, interacting with the back-end server and processing the configuration requests from the end-user.

[1] The main supplier for the sensors is DFRobot: www.dfrobot.com.

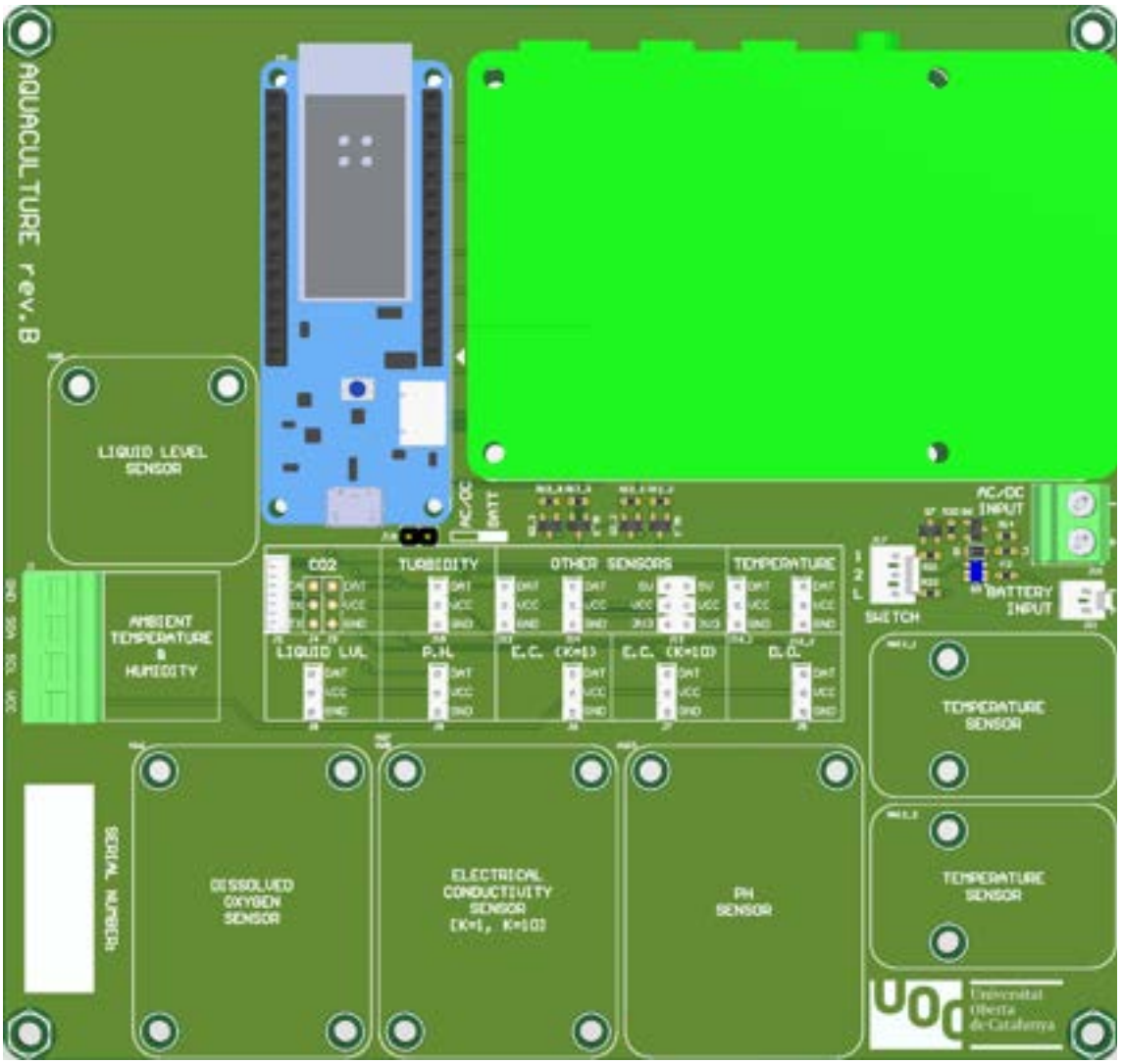

Fig. 1. 3D PCB of the ADO node.

The RPI runs a GNU/Linux distribution and executes a set of micro-services implemented in Python that handle the user-interaction, configuration and authentication methods. The RPI is responsible to flash the Arduino when booted for the first time and takes care of checking and processing the firmware updates upon any hardware reset. Fig. 4 is a sequence diagram illustrating the firmware operation described earlier.

3.2. Communication technologies/protocols

The ADO board integrates different communication interfaces. In addition and thanks to its modular design, it enables to integrate communication interfaces as required by the end user.

3.2.1. Communication technologies

By default, the basic hardware configuration of the ADO board offers two main communication interfaces, namely: Ethernet and Wi-Fi. The MKR Arduino module selected by the presented ADO version (e.g., MKR 1000) features a Wi-Fi interface, but it can simply be replaced by a LTE board as they are pin compatible if a user is required to use cellular connectivity instead.

The Ethernet interface is used for the configuration and bootstrapping phase of the devices. It is reasonable to assume that near the ponds the users will not have access to Ethernet sockets, and therefore they will be using WiFi or another wireless connectivity interface once deployed. The Ethernet interface however is needed during the setup phase as described above.

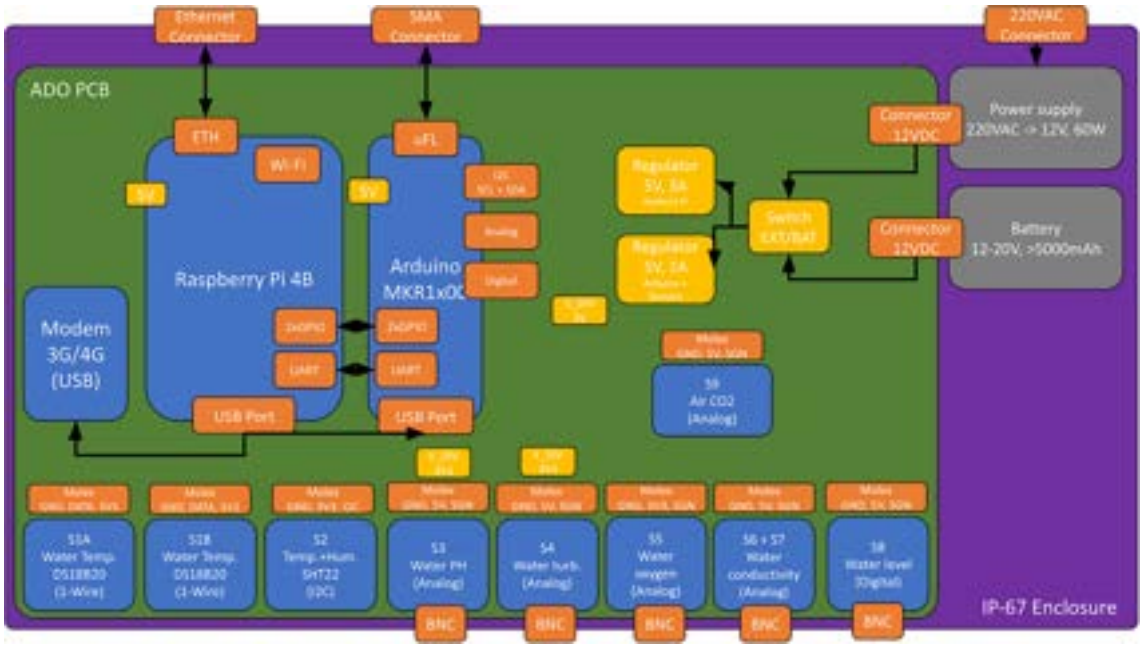

Fig. 2. Block diagram of the ADO node.

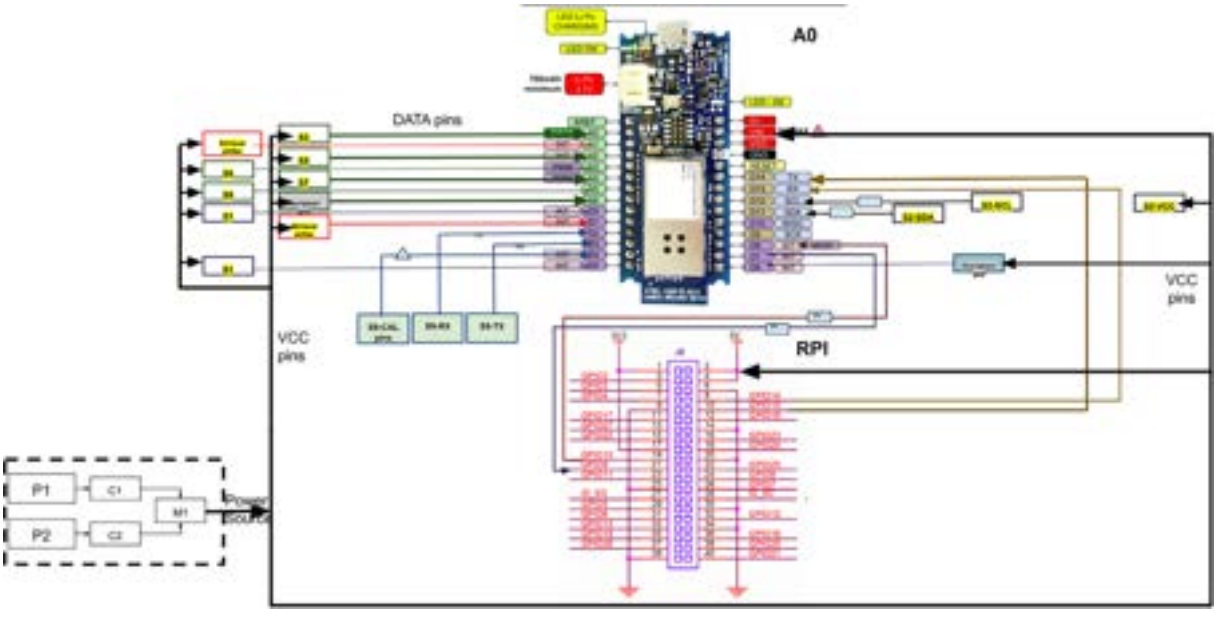

Fig. 3. Pin mapping between the components of the ADO node: the symbols are defined in the Bill of Materials (Section .5).

During the setup process, the user is prompted to configure the WiFi interface and connect to the existing network. The WiFi interface is then used during the lifetime operation of the device.

3.2.2. Communication protocols

As described earlier, each ADO node periodically sends sensor data to the back-end. The communication protocol used for sending this data is MQTT (Message Queue Telemetry Transport) [9]. MQTT has been selected as it is extremely lightweight and extensively used in a wide variety of industries (i.e., automotive, manufacturing, telecommunications, oil and gas, etc.) worldwide. It is based on a publish/subscribe mechanism, in which the node publishes messages containing sensor data. There are entities (other ADO nodes, clients) that can subscribe to specific topics. There is a central entity, a broker (ADO

Table 1
Technical specification of the ADO sensors.

Sensor	Measurement Range	Resolution	Accuracy	Temperature Range	Water- proof
S1: Water Temp.	−55 to 125°C	0.1°C	±0.5°C	−55 to 125°C	Yes
S2: Atmospheric Hum and Temp	0 to 100 RH, −40 to 125°C	0.1 RH, 0.1°C	±3% RH, ±0.3°C	−40 to 125°C	Yes
S3: pH	0 to 14	0.01	±0.1	5 to 60°C	Yes
S4: Turbidity	0 to 3000 NTU	1 NTU	N/A	5 to 90°C	Yes, in IP67 box
S5: DO	0 to 20 mg/L	0.01 mg/L	N/A	0 to 40°C	Yes
S6: Conductivity K = 1	0 to 15 ms/cm	0.001 ms/cm	±5%	0 to 40°C	Yes
S7: Conductivity K = 10	10 to 100 ms/cm	0.001 ms/cm	±5%	0 to 40°C	Yes
S8: Water Level	No limit	N/A	±0.5 mm	−25 to 105°C	Yes
S9: CO2	0 to 5000 ppm	1 ppm	±50 ppm	0 to 50°C, 0 to 95% RH	Yes, in IP55 box

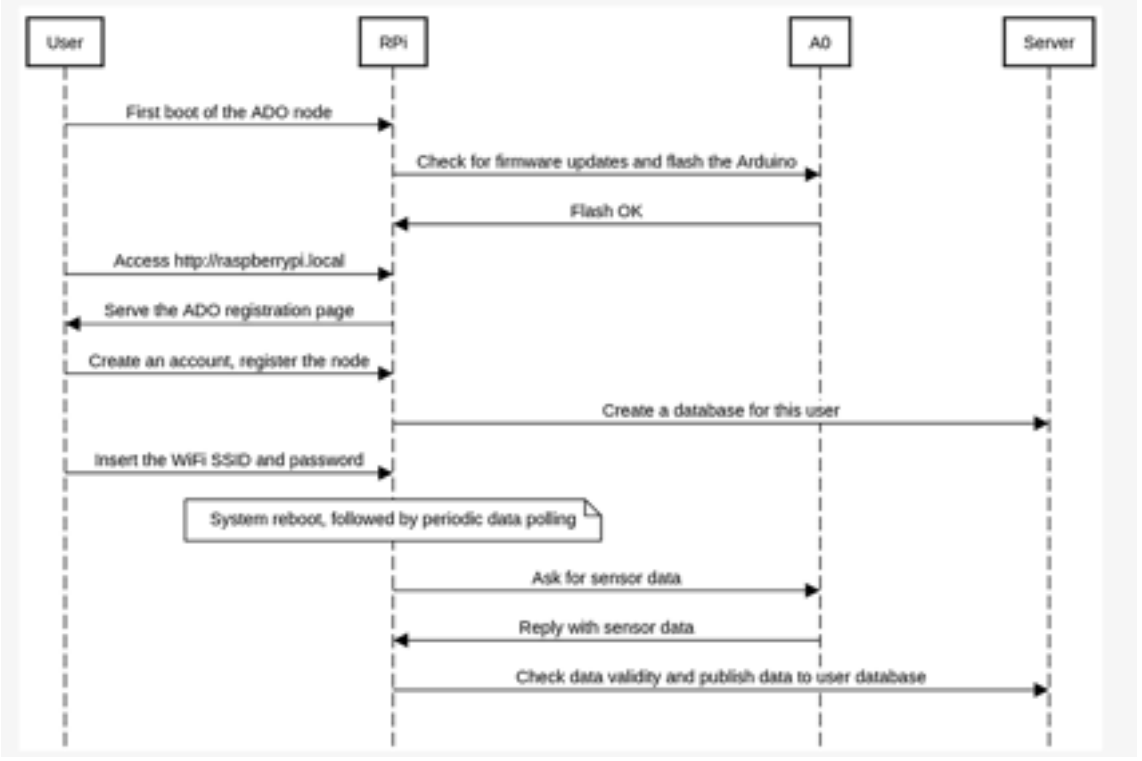

Fig. 4. ADO node: sequence diagram of the firmware operation and the corresponding interactions between the user, RPi, Arduino and back-end server, when registering the node and visualizing data.

server), that filters the received messages and topics, and makes them available to the entities that subscribed to that specific topic. In this way, the messages sent by the publisher will always be available to the subscriber regardless of them running at the same time, or knowing each other's IP address or identities.

More specific, for ADO, when a user registers, it gets assigned a unique channel ID. This ID is being inserted in the topic used by its nodes for publishing data: ''channels/[channel ID]/messages". All the nodes of this user will publish messages to this topic and will be subscribed to the control topic: ''channels/[channel ID]/control". The Graphical User Interface extracts from the user's database the data published in the messaging topic and adds it to the user's dashboards; moreover, it uses the control topic to send control actions from the user to the nodes: calibration, sampling rate, etc. By using these unique topics, the messages intended for one user will never be available to another user. The ADO back-end infrastructure, described in detail in the next section, takes care of instantiating the MQTT broker that handles all the topics.

The messages sent by the ADO nodes in the messaging topic are stored in an Influx database. Each user database contains two measurements: ''control" – storing the commands sent by the user – and ''messages" – storing the sensor readings sent by the nodes. Data stored under both measurements is organized into the following influxdb fields and tags:

- time: timestamp
- channel: user's unique channel ID

- device name: the specific ADO device that sends the data (messaging topic), or which the control is acting on (control topic)
- sensor name: the specific sensor that the reading belongs to (messaging topic), or which the control is acting on (control topic)
- protocol: MQTT
- publisher: unique ID associated with the ADO device
- unit: data measurement unit
- update time: the maximum time before this sensor will provide an updated reading for a measurement
- value: actual sensor reading (messaging topic), or the setting imposed on that sensor (control topic)
- subtopic: two possible values – SR and CAL – particular to the "control" measurement, to differentiate between sampling rate control and calibration control

Data is sent in JSON format, using SenML labels: payload = ["bn": "", "n": sensor_name, "u": unit, "v": value, "t": timestamp]. These labels are used to fill the InfluxDB database with the corresponding values. The information on unique ID's is collected automatically, as each device uses its own ID and key (received upon registration) to log into the MQTT broker. The information on channel ID and the corresponding influx measurement is contained in the MQTT topic used for publishing the JSON data.

3.3. Back-end system

The ADO back-end infrastructure is responsible for two things. First, receiving the data from the ADO nodes and storing the data in a time-series database. Second, providing a set of tools to manage the assets and visualize the data. Regarding the asset management, it includes mechanisms to authenticate the devices and a security framework to enable multiple tenants. Regarding the visualization tools, it includes mechanisms to create alarms based on predefined thresholds.

These functionalities are provided by a framework composed of different building blocks leveraging open-source tools and frameworks that already exist. These components have been integrated together to provide a functional, secure and configurable server infrastructure closing the system orientation of the ADO framework. Fig. 5 presents a schematic diagram of the different components used to integrate the back-end system.

3.3.1. Platform orchestration

The Mainflux framework [10] has been used as a base building block to orchestrate the system. Mainflux is a framework developed in Go and built as a set of microservices that provides device management, data aggregation and data management functionalities. On top of that, it provides connectivity and message routing between different back-end components, event and alarm management, analytics on the core operation and the possibility to connect to different user interfaces.

One important feature of Mainflux is that it is built to support multi-tenancy and enables different organizations and users to run their dashboards and management interfaces in an isolated manner. The ADO platform leverages the security and user management functionalities of Mainflux to both authenticate and secure the devices and the back-end platform.

Fig. 5. ADO back-end building blocks: input and user APIs, user authentication and asset management, data visualization and data storage, and the corresponding tools for building them.

To operate, Mainflux uses two different databases. First, Postgress SQL is a relational database used to store operational and management data, such as information about users, devices and locations. Second, InfluxDB is a time-series database used to store the data received from the sensor nodes.

Message routing between ADO devices and the ADO back-end is handled by a MQTT broker, exposed by the Mainflux framework as a microservice. Finally, the NGINX application server is used as proxy to hide the interfaces of the different micro-services. This ensures that the management and other critical interfaces are not exposed to non-specialized users.

In addition to integrating Mainflux, the ADO back-end has a custom Python service that is responsible to customize this application for the aquaculture vertical and to allow the user to control its devices remotely. This control is done via the Grafana user interface and dashboards, as explained in detail in Section 3.3.2), and allows users to identify its devices, turn the sensors on/off and control their sampling rate, calibrate them following on-screen instructions, and set thresholds to determine when alarms should be triggered.

3.3.2. Visualization framework

The visualization framework is built using Grafana [11]. Grafana is an open-source dashboard and visualization system that is well-known and widely adopted.

Within the ADO back-end, Grafana has been integrated into Mainflux as a microservice and is used as the entry point for the non-administrative user, allowing to access sensor data and configuring sensor devices, as explained earlier. Sensor data is obtained directly from the InfluxDB database, communication is handled by MQTT, and access is provided through the NGINX application server.

Several Grafana templates have been developed to simplify the use of the platform, defining the proper visualization dashboards organized by tenant, device and sensor type. We have extended the Grafana dashboard to enable the end users to configure the sampling rate of the ADO nodes. With a simple combo-box, the users can change the sampling rate of the ADO devices, as well as start or stop them. More specifically, the user has access to the following visualization dashboards:

1. "All devices" (Fig. 6b,c): this dashboard offers the visualization of aggregated sensor data from all the registered devices, available at one glance. This is useful for quickly checking the status of a parameter in all the locations that have devices installed. For example, checking the current temperature in all the pools. By selecting the temperature sensor from the drop-down menu, two types of information are displayed: i) the bar plot is showing the average temperature value for each device. The number of bars is equal to the number of user registered devices that have that type of sensor connected. ii) the time series plot is showing the evolution in time of the temperature value, for each device, on the same plot, but with different colors. The time range can be selected by the user to any interval since the registration of the devices. Fig. 6b) shows the aggregated temperature data for the case of a user that has two devices equipped with temperature sensors: there are two bars corresponding to each device, showing the average temperature value, and two traces in the time series plot, showing the evolution of this parameter for both devices. In comparison, Fig. 6c) shows the dissolved oxygen data, but the user has only one device equipped with this type of sensor, so only one wide bar is showing, and one green trace.
2. "Calibration": this dashboard is to be used if the user has at least one of the following sensors: C02, Dissolved Oxygen, Conductivity and/or pH. It offers written instructions for correctly calibrating each of these sensors, according to their technical specifications. Calibration is recommended to be repeated once a month, but this dependents on the environment where the sensors are used and their usage frequency. We created a collection of YouTube videos[2] that describe not only the calibration process, but also the other visualization options of the ADO graphical interface.
3. "Device configuration" (Fig. 6d,f): this dashboard is used as per device basis, to look deeply into the data that a certain device provides. From the drop-down menu, the device of interest can be selected. There are three types of information displayed: i) overall device health (Fig. 6d), showing the most recent value recorded by each sensor of the selected device, in green if the sensor is currently working and in red if that sensor is currently turned off; ii) time series data for each of the sensors of the selected device (Fig. 6d), – this data can be downloaded in various formats: CSV, text, for a specific time range; iii) the current sampling rate value for each sensor of the selected device, with options to change it to any value from the drop-down menu, or to turn the sensor off (Fig. 6f); This control is sent to the device in real time, using MQTT messages.
4. "System alerts" (Fig. 6e): these dashboards has two sections. The top section shows a history of the alerts that the ADO system triggered, aggregated for all the devices and all sensors. Alerts are displayed in red if they are critical and an alarm was triggered, in orange if the event is still evaluated or there is no data to be evaluated, and in green if the sensor values are within the specified ranges. The alert mentions the device name and the sensor value that triggered it. The bottom section of this dashboard allows the user to select each sensor and to set the "healthy" range of values. Any values that would be outside that range will trigger an alarm, that consists in a new red entry in the alarm history, a new red entry on the GUI homepage and an email notifying about the registered value, the device in question and optionally, a screen capture from the GUI.

[2] ADO tutorial: https://www.youtube.com/playlist?list=PLcyHQsMeofzA-NNKaqbpQZJie6e59i0zi

Fig. 6. Graphical user interface of ADO deployed in the pilot case: a) ADO login screen; b,c) "All devices" dashboard; d),f) "Device configuration" dashboard; e) "System alerts" dashboard.

The user has the possibility to create its own dashboards and manipulate the collected data from its sensors, to obtain any other useful information for its personal application.

4. Design files

The complete list of the hardware design files and software projects is summarized in the following table. These files are available at https://doi.org/10.17632/cvdc3gsjsg.1 and contain the necessary information to replicate the hardware PCB, the firmware code that has to be installed on the hardware, and the software code for launching the server. A description of each of the listed design files follows:

- Schematic_Prints.pdf: PDF file containing the schematic describing the interconnections between components of the ADO hardware.
- Board.step: the 3D model of the ADO PCB.
- BOM.xlsx: Excel file containing the Bill of Materials for replicating the ADO node.
- Gerber.zip: compressed file containing the Gerber files for building the ADO PCB and assembling the ADO Node.
- NC Drill.zip: compressed file containing the information for drilling the holes in the PCB.
- Pick Place.zip: compressed file containing the pick and place instructions for the assembly of the board.
- ADO-Arduino.zip: compressed file containing the source code and the libraries to program the Arduino module. This code is automatically installed by the RPi unit and the user does not have to deal with it. The Arduino module collects data from sensors and sends it to the RPi module upon request.

- ADO-RPi.zip: compressed file containing the source code for the RPi. The installation guide is provided in Listing 1. This code handles the Arduino firmware installation, schedules sensor data collection and securely publishes the data to the server.
- ADO-backend.zip: compressed file containing the source code for the ADO server. It is build on top of a fork of the Mainflux [10] Github project. The developed extensions enable its simple deployment and integration to the ADO nodes (see Section 3.3.1).

Design filename	File type	Open source licence	Location of the file
Schematic_Prints.pdf	*PDF*	*CERN OHL v2*	*PCB*
Board.step	*STEP file (.step)*	*CERN OHL v2*	*PCB*
BOM.xlsx	*Excel file (.xlsx)*	*CERN OHL v2*	*PCB*
Gerber.zip	*Compressed file (.zip)*	*CERN OHL v2*	*PCB*
NC Drill.zip	*Compressed file (.zip)*	*CERN OHL v2*	*PCB*
Pick Place.zip	*Compressed file (.zip)*	*CERN OHL v2*	*PCB*
ADO-Arduino.zip	*Compressed file (.zip)*	*BSD 3-clause*	*Firmware*
ADO-RPi.zip	*Compressed file (.zip)*	*BSD 3-clause*	*Firmware*
ADO-backend.zip	*Compressed file (.zip)*	*BSD 3-clause*	*Server*

5. Bill of materials

The ADO node can be built using the list of materials provided in the following table. The main supplier for the sensors is DFRobot (https://www.dfrobot.com). Note that from this list of sensors, a user may choose those that are more adequate for their setting, without needing to connect all the listed sensors to the node.

Designator	Component	Number	Cost per unit currency	Total cost	Source of materials
S1	*Gravity: Waterproof DS18B20 Sensor Kit KIT0021*	*2*	*€6.60*	*€13.20*	*https://www.dfrobot.com/product-1354.html*
S2	*SHT20 I2C Temperature & Humidity Sensor (Waterproof Probe) SEN0227*	*1*	*€19.85*	*€19.85*	*https://www.dfrobot.com/product-1636.html*
S3	*Gravity: Analog pH Sensor/Meter Kit V2 SEN0161-v2*	*1*	*€35.00*	*€35.00*	*https://www.dfrobot.com/product-1782.html*
S4	*Gravity: Analog Turbidity Sensor For Arduino SEN0189*	*1*	*€8.70*	*€8.70*	*https://www.dfrobot.com/product-1394.html*
S5	*Gravity: Analog Dissolved Oxygen Sensor/ Meter Kit For Arduino SEN0237-A*	*1*	*€149.00*	*€149.00*	*https://www.dfrobot.com/product-1628.html*
S6	*Gravity: Analog Electrical Conductivity Sensor/ Meter V2 (K = 1) DFR0300*	*1*	*€52.00*	*€52.00*	*https://www.dfrobot.com/product-1123.html*
S7	*Gravity: Analog Electrical Conductivity Sensor/ Meter V2 (K = 10) DFR0300-H*	*1*	*€61.66*	*€61.66*	*https://www.dfrobot.com/product-1797.html*
S8	*Gravity: Photoelectric Water/ Liquid Level Sensor For Arduino SEN0205*	*1*	*€6.09*	*€6.09*	*https://www.dfrobot.com/product-1470.html*
S9	*Gravity: Analog Infrared CO2 Sensor For Arduino (0–5000 ppm) SEN0219*	*1*	*€51.20*	*€51.20*	*https://www.dfrobot.com/product-1549.html*
A0	*Arduino MKR1000*	*1*	*€32.50*	*€32.50*	*https://store-usa.arduino.cc*
RPI	*Raspberry Pi 4 Model B*	*1*	*€64.95*	*€64.95*	*https://www.tiendatec.es*
IP67 box	*Europa Components IP67 box, ID: PB526219*	*1*	*€15.97*	*€15.97*	*https://es.farnell.com*
IP55 box	*WISKA IP55 box, ID: 10109426*	*1*	*€0.97*	*€0.97*	*https://es.farnell.com*
P1	*Keenso AC/DC converter, 5A, 12 V, ID: Keensorosx8hy3id*	*1*	*€18.89*	*€18.89*	*https://www.amazon.es*
SD card	*SanDisk 16 GB*	*1*	*€5.66*	*€5.66*	*https://www.amazon.es*
PCB	*ADO PCB*	*1*	*€36.62*	*€36.62*	*Circuitos Impresos 2CI, S.L.*

6. Build instructions

The ADO node can be built by assembling the different components into the main board. Fig. 7 presents a visual summary of different steps during the assembly process. The main PCB and the power source are the first two elements to be placed (2). They should be screwed on top of a support plate (1) that later has to be inserted as the base of the IP67 box (3). The box should be prepared by adding the power button and the AC IN connector (4). After that, the PCB can be inserted in the box and the RPI (with the SD card containing the firmware image inserted, as instructed in Section 7.1) and Arduino modules can be connected in their respective placeholders (5). For each of the sensors that need to be connected into the main board, the specific auxiliary board should be plugged into the respective placeholder. Then sensor cables need to be passed through the cable holes in front of the box (4, 6 and 7). Once the sensors are connected in their respective auxiliary boards, these boards need to be connected to the main board using the flat wires (8). All the needed information regarding where the flat wires have to be connected is printed on the PCB, as depicted in Fig. 1. The last step is to close the box, insert the Ethernet cable and power the node by plugging in the power cable and pressing the power button (9). The device will start sending sensor data to the back-end once the device has been configured, the server has been successfully deployed, and the user has registered with an account, as described in Section 7.

7. Operation instructions

This section presents the operation of the ADO hardware platform and back-end system. In addition to the following summary, we have created a step-by-step video guide that describes how to create an account, register an ADO node and how to use the Graphical User Interface, available on YouTube[3].

7.1. Hardware platform

In order to install the firmware, first we need a SD card plugged into the Raspberry Pi module with the GNU/Linux operating system and then we need to install an Arduino code uploader (i.e., Arduino CLI). In our case we have used Raspberry Pi OS Lite[4], but any other arm64-based GNU/Linux distribution should work. The Raspberry Pi module is in charge of flashing the Arduino code, which makes having installed[5] the Arduino CLI a prerequisite. After that, plug in the Ethernet cable into the ADO board, make sure you have Internet access, and follow the instructions in Listing 1 to complete the setup process.

In order to test the installation, open a new tab in a browser in your personal computer and access the following link: http://raspberrypi.local. Please notice that the name "raspberrypi" in the previous link can be changed[6] to access the devices using any other name. For example, if the hostname is set to "ado" the device can be accessed using http://ado.local.

After accessing the URL (i.e., http://raspberrypi.local) a warning message may appear because the web server certificates are self-signed. If that is the case, please click on "Advanced" and choose "Proceed" to continue with the log-in process. After that a login/registration page should appear and, after registration, the nodes will start sending their data to http://ado.wine-lab.org.

Listing 1: Installing the ADO firmware

```
$ sudo apt-get update
$ sudo apt-get install git, sqlite3
$ cd/opt
$ sudo mkdir Raspberry
$ cd Raspberry
$ git clone https://github.com/wine-uoc/ADO-RPi.git
$ cd ADO-RPi
$ git checkout DEMO
$ pip3 install -r requirements.txt
$ sudo apt-get install supervisor
$ sudo cp config/ado.conf/etc/supervisor/conf.d
$ sudo service supervisor start
```

[3] https://www.youtube.com/playlist?list=PLcyHQsMeofzA-NNKaqbpQZJie6e59i0zi
[4] For the detailed installation instructions please check reference [12].
[5] For installing Arduino CLI please follow this guide: https://arduino.github.io/arduino-cli/0.19/installation/.
[6] To change the system hostname please follow the guide in reference [13]

Fig. 7. Assembly process sketch.

For the more experienced users that want to replicate this installation for multiple nodes, an image of this first installation can be created. With such approach the image can be copied to any number of SD cards, that can be directly plugged into other nodes, without the need to repeat the steps from Listing 1.

7.2. Back-end system

The ADO back-end system is also based on the GNU/Linux operating system and can be hosted on premises or in the cloud. The server minimum requirements are 2 CPUs, 4 GBytes of RAM, and 40 GBytes of disk, although these parameters may vary depending on the concurrent number of connected users and ADO nodes reporting data simultaneously. For the cloud service we have used Amazon AWS and for the operating system we have used Ubuntu Server 18.04.4 LTS. However, other cloud providers and GNU/Linux distributions are known to be fully compatible with the ADO back-end system.

In addition to the hardware and operating system requirements, the ADO back-end system requires Docker (version 19.03) and Docker Compose (version 1.24.1). Since the application server is built following a micro-services architecture and containerized for an easy and scalable deployment, we use Docker and Docker Compose to pack everything together and provide a simple launch procedure to run the entire framework.

In order to execute the ADO back-end in a server the user first needs to install Docker (Listing 2) and Docker compose (Listing 3). Then, a TLS certificate has to be generated[7], allowing nodes and users to connect to the server securely. After that, the code has to be downloaded from Github and a few configurations have to be done to insert the user's personal information. The last step is to launch everything with just one 'make run' command (Listing 4).

[7] For example, it can be obtained by following this guide: https://letsencrypt.org/getting-started/.

Listing 2: Installing prerequisites: Docker version 19.03.9

```
sudo apt-get update
$ sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg \
  lsb-release
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo gpg --dearmor -o/usr/share/keyrings/docker-archive-keyring.gpg
$ sudo apt-get update
$ sudo apt-get install docker-ce = 5:19.03.9 3--0 ubuntu-focal \
  docker-ce-cli = 5:19.03.9~3--0~ubuntu-focal \
  containerd.io
Test installation by running $ sudo docker run hello-world
```

Listing 3: Installing prerequisites: Docker compose version 1.24.1

```
$ sudo curl -L \
  " https://github.com/docker/compose/releases/download/1.24.1/
  docker-compose-$(uname -s)-$(uname -m)" \
  -o/usr/local/bin/docker-compose
$ sudo chmod + x/usr/local/bin/docker-compose
$ sudo ln -s/usr/local/bin/docker-compose/usr/bin/docker-compose
Test the installation with $ docker-compose --version
```

Listing 4: Steps for launching the ADO server infrastructure

```
  $ git clone https://github.com/wine-uoc/ADO-backend.git
$ cd mainflux
$ sudo nano .env (update here the file with own support email,and with grafana admin username and
  password and save the file)
$ cd docker/ssl/certs (copy here the TLS certificates obtained from Let's Encryptand rename them as
  ca.crt and ca.key)
$ cd ../../..
$ sudo make run
```

It is important to notice that the server DNS used for the TLS certificate should match the one configured for the node in config.py, as shown in Listing 1. In this paper, the ADO node and server are configured to usehttp://ado.wine-lab.org, and the node recognizes the TLS certificates provided for the server by Let's Encrypt (https://letsencrypt.org/).

8. Validation and characterization

The ADO platform has been evaluated in a pilot case carried out in the IEPAAC school in the Ebre's Delta, Catalonia (Fig. 8). The purpose of the pilot is threefold. First, validate the functional operation of the devices and software ecosystem. Second, validate the usability of the platform and adequacy to the different cultures in a typical deployment. Third, receive feedback from aquaculture experts and students to improve the platform.

The pilot has been conducted for a period of 8 months from April 2021 to November 2021, and will continue for the rest of 2021. The IEPAAC is an aquaculture formation school which offers a wide variety of aquaculture scenarios, including larval area, aquariology, seaweed cultivation area, octopus nursery and bivalve ponds among others see Fig. 9.

Regarding the hardware platform, five ADO devices have been deployed during the pilot. Each ADO device has 5–8 sensors depending on the area that has been deployed. This allows to monitor specific parameters of interest for the type of ponds and activity conducted in that area. For example, in the larval area one ADO node has been deployed to monitor parameters such as ambient temperature, water temperature at different depths, dissolved oxygen, turbidity and PH. The

Fig. 8. IEPAAC Aquaculture formation school in Ebre's Delta, with expertise in raising larvae and fattening the different aquaculture species, cultivation of phytoplankton and aquaponic crops.

Fig. 9. Deployment of the ADO system in the Aquaculture Center in the Ebre's Delta. Device testing (top-left and right) and nodes installed in final positions (top-right, bottom-left and right).

price of a node with this specific configuration is €400. The expertise of the professors in the school has been central to determine the sensors to be deployed at each area.

Regarding the back-end system, the server is hosted in the Amazon AWS cloud using an EC2 t2.medium instance (i.e., 2 CPUs, 4 GBytes of RAM) which runs the Ubuntu Server 18.04.4 LTS operating system. To support the pilot, as well as other early adopters, the back-end system is currently available at https://ado.wine-lab.org/.

Fig. 10. Temperature and turbidity data collected from June to October 2021, with a sampling rate of 2 min. Data show daily patterns influenced by the repetitive activities in the pond.

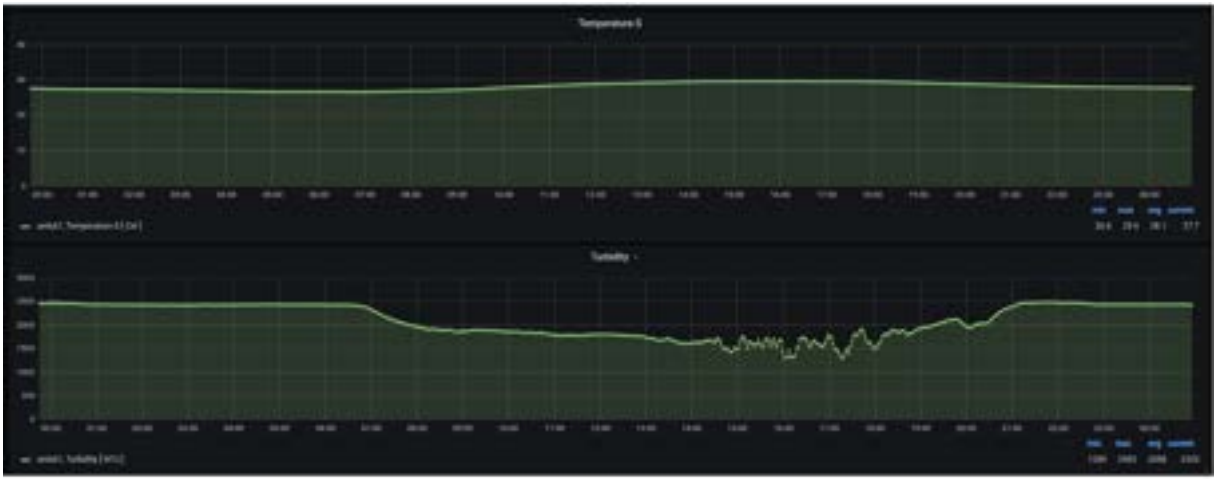

Fig. 11. Temperature and turbidity data collected during a day, with a sampling rate of 2 min. Variations in turbidity levels due to feeding and day-night variation of lighting.

In terms of functional validation, the hardware has been running for the described period. After some initial issues with one PH sensor not being properly calibrated, the system has been smoothly reporting data from the five ADO devices. So far, the system has collected 3.6 million samples from the 5 sensor nodes.

As an example, Fig. 10 and Fig. 11 present some data collected for the temperature and turbidity sensors, for a period of 4 months and 1 day, respectively. In both cases, the sensor sampling rate is 2 min. In addition, Fig. 10 is an overview that indicates that there are daily patterns in the collected data[8], whereas Fig. 11 zooms into this data and shows how the temperature and turbidity values vary during a 24 h window. This data is of importance to the aquaculture experts that can correlate these patterns to the specific activity inside the pond, so that they can adjust the water parameters to obtain the most qualitative environment for the production. For example, they can learn how different water parameters – temperature, light, pH, or others – impact the feeding and reproduction cycles, or how to prevent disease.

Another feature to validate was the alarming system. Normally, for each sensor, the user has to set a range of values that are considered "healthy" for its application; any readings that would be outside that specific range would trigger an alert. More specifically, Fig. 12(top) shows the readings from two temperature sensors, and a user defined range of [10°C–30°C]. The readings are within this range, so no alarm is triggered, and the vertical bars are green. These vertical bars represent the periodic evaluation from the system that checks if the readings are within the specified range. We set the range for this sensor to [0°C–10°C] and now the readings are above this range – Fig. 12(bottom). For this case, the vertical bars

[8] Please notice that the short gaps in the collected data (i.e., Fig. 10) correspond to periods of time when the server has been turned off for maintenance.

Fig. 12. System alerts: (top) sensor readings are within the user-defined range [10°–30°]; (bottom) sensor readings are outside the defined range [0°–10°] and are triggering the Alerting mode.

Fig. 13. System Alerts history. The most recent alert – Atmospheric Temperature sensor – with complete information – device name and sensor value – is displayed on top.

become orange, as the system is evaluating if the sensor values are consistent. As the sensor keeps reading the same values for more than 10 min – user defined – the vertical line changes to red and the system enters the "Alerting" mode. In this mode, notifications show up in the historical section of the dashboard (Fig. 13), mentioning the sensor type, the device, the sensor value and the timestamp of the alert. The user will also be notified by email with the same information that shows up in the historical section.

Regarding the usability of the platform, we have validated it with the academics working at the school during both the development and the pilot phases. During the development phase, we held different meetings to discuss the platform functionality and shape it to the needs of the sector, taking advantage of their expertise. During the pilot phase, we received feedback that has been used to further improve the platform. In the following table, we provide the most relevant items that have been accommodated to meet the scenario requirements.

Issue	Corrective action
Too short cables	The ADO node has to be placed close to the pond or the pool of interest. Yet, the longer the sensor cables, the better for its usability.
Illumination in some pools	There are some pools that have their own illumination. This may cause some issues with the turbidity sensor calibration.
Temperature sensor at different depths	A weight is needed to make sure that the temperature sensor is placed at the required depth.

9. Conclusions

The ADO project develops an open-source IoT solution – data acquisition system, data storage system and visualization platform – that allows the digitization of the aquaculture sector, with a final cost per device under €600. This allows the small and medium-sized producers to obtain success stories with much smaller economic investments. There is a loss in accuracy compared to the professional industrial solutions, but ADO still brings huge benefit because the alternative for the small producers would be manual water sampling, a few times per day. We deployed a pilot in the IEPAAC school in Ebre's Delta, Catalonia, from April 2021 and that will continue for the rest of 2021 and 2022. The pilot deployment validated that the ADO system is working correctly and, most important, that the readings provided by the ADO sensors match those of the manual sensors they were using. With ADO they have automatic sensor readings from five locations, every two minutes (configurable sampling rate). This opens the opportunity of correlating real time sensor values with specific activities inside the pond and obtaining a qualitative environment for the production, based on learning. An example would be learning about the feeding cycles, how feeding impacts reproduction, or how different water parameters impact both reproduction and feeding. Another important learning would be about minimizing the probability of disease.

Finally, the ADO device can be adapted to any aquaculture vertical. However, the casing design proposed in this article focuses on tank-based aquaculture and when adapting it to other vertical, the ingress protection of the device and its powering mode should be reconsidered.

10. Human and animal rights

ADO is a platform to monitor water parameters related to the aquaculture activity. During our work, the sensors have been used in a real environment where different marine species are being taken care of. All the conducted deployments have been carried out with the support and strict supervision of experts at the IEPAAC school, following the code of conduct and rules to interact with animals, avoiding stress situations or harming their habitat.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to acknowledge the team of excellent professors at the IEPAAC center for their patience, support, feedback, and expertise. In addition, this project received funding from the ADO project ARP059 supported by the European Maritime and Fisheries Fund (EMFF), the 2017 SGR 60 project granted by the Generalitat de Catalunya and from the CYTED 520RT0011 AgIoT Research project.

References

[1] InPro 8000 Series Turbidity Sensor, URL: https://www.mt.com/dam/MTPRO/PDF/TD/TD_InPro8000_Series_TurbiditySensor_en_52800246_May15.pdf, online; accessed 22 January 2021..
[2] Accurate medium and high turbidity measurement, URL: https://www.mt.com/int/en/home/products/Process-Analytics/turbidity-meter/medium-to-high-sensor.html, online; accessed 22 January 2021..
[3] L. Parra, S. Sendra, L. Garcia, J. Lloret, Design and deployment of low-cost sensors for monitoring the water quality and fish behavior in aquaculture tanks during the feeding process, Sensors 18 (2018) 750.
[4] S. Saha, R. Rajib, S. Kabir, Iot based automated fish farm aquaculture monitoring system, 2018, pp. 201–206..
[5] Q.D. Luong Vinh, D. Dung, T. Danh, C.-N. Nguyen, Design and deployment of an iot-based water quality monitoring system for aquaculture in mekong delta, Int. J. Mech. Eng. Robot. Res. (2020) 1170–1175.
[6] S. Chandanapalli, Design and deployment of aqua monitoring system using wireless sensor networks and iar-kick, J. Aquacult. Res. Develop. 05 (2014) 01.
[7] K. Qayyum, I. Zaman, A. Förster, H2o sense: a wsn-based monitoring system for fish tanks, SN Appl. Sci. 2 (2020) 1643..
[8] C. Encinas, E. Ruiz, J. Cortez, A. Espinoza, Design and implementation of a distributed iot system for the monitoring of water quality in aquaculture, 2017 Wireless Telecommunications Symposium (WTS) (2017) 1–7.
[9] Mqtt: The standard for iot messaging, [Online], accessed: 08 October 2021. [Online]. Available: URL: https://mqtt.org/..
[10] Mainflux open source iot platform edge computing and consulting services, 2021. [Online]. Available: URL: https://www.mainflux.com/..
[11] Grafana: the open observability platform, 2021. [Online]. Available: URL: https://grafana.com/..
[12] J. Kaplan-Moss, Setting up a headless raspberry pi, [Online], accessed: 08 October 2021. [Online]. Available: URL: https://jacobian.org/2021/jan/22/headless-rpi/..
[13] J. Fitzpatrick, How to change your raspberry pi (or other linux device's) hostname, [Online], accessed: 08 October 2021. [Online]. Available: URL: https://www.howtogeek.com/167195/how-to-change-your-raspberry-pi-or-other-linux-devices-hostname/..

Ioana Suciu received the B.S. degree in electrical engineering from Military Technical Academy, Bucharest, Romania, in 2014, the M.S. degree in wireless communication engineering from CentraleSupelec, Paris, France, the M.S. degree from the Politehnica University of Bucharest, in 2016 and a Magna Cum Laude Ph.D. degree in network engineering in 2020 from the Polytechnic University of Catalonia, Spain, where she was also a Marie-Curie Early Stage Researcher.
From 2014 to 2015, she was a Research Assistant with Military Equipment and Technologies Research Agency in Bucharest. In 2016, she was a Research Assistant Intern with Orange Labs Networks, Paris, and in late 2016 she joined Worldsensing, Barcelona, a global IoT pioneer, as a Ph.D. Researcher, and a Software Engineer, where she was also with the SCAVENGE European Training Network. In 2018, she was a Visiting Research Scholar with the Berkeley Sensor and Actuator Center (BSAC), University of California at Berkeley. In 2019 she joined the Wireless Networks Department of Universitat Oberta de Catalunya, Spain, where she is currently a research assistant and embedded software developer, working on topics related to Internet of Things and wireless communications.

Pere Tuset-Peiro is Associate Professor at the Universitat Oberta de Catalunya (UOC) and Senior Researcher at the Wireless Networks (WiNe) group. He received the B.Sc. and M.Sc. in Telecommunications Engineering from the Universitat Politecnica de Catalunya (UPC) in 2007 and 2011, respectively, and his Ph.D. in Network and Information Technologies from the Universitat Oberta de Catalunya (UOC) in 2015.

Xavier Vilajosana (M'09, SM'15) received his B.Sc. and M.Sc in Computer Science from Universitat Politecnica de Catalunya (UPC) and his Ph.D. in Computer Science from the Universitat Oberta de Catalunya (UOC). He has been a researcher at Orange Labs, HP and UC Berkeley. He is full Professor at UOC.

Guillem Boquet received the degree in telecommunications engineering and the PhD degree in telecommunications engineering from the Universitat Autonoma de Barcelona (UAB) in 2014 and 2021, respectively. From 2012 to 2014, he was a research assistant at the Department of Telecommunications and Systems Engineering (DTSE), UAB. From 2014 to 2016, he worked for several companies such as Everis, Vodafone, Huawei and Catalan government agency CESICAT. Since 2016, he has been with the Wireless Information Networking (WIN) Group, DTSE, UAB, collaborating as an Assistant Professor in teaching courses on analog and digital communications and signals and systems. Since 2020, he is a researcher at the Wireless Networks (WiNe) group of the Universitat Oberta de Catalunya (UOC).