Enabling the Ocean Internet of Things with Renewable Marine Energy

Mathew B. R. Topper, Nicolas A. M. M. Jarnoux, Ronan Costello, Cian Murtagh, Simone Giorgi and Ben Kennedy

Abstract—**Marine renewable energy can play an integral role in reducing the cost and complexity of collecting data from the oceans and enhancing its exploitation. The processing and transmission of data at sea may be modelled as an 'internet of things' (IoT) application. The 'thing' is any offshore device which can collect and process data in situ, while the 'internet' represents the medium for transmitting data. IoT is differentiated from other internet-based communication paradigms by the constraints on the system. These constraints are typically limited energy budgets and computing power, intermittent and low-bandwidth connectivity, and limited physical access. While land-based IoT applications are already well served by a wide selection of hardware and software components, the needs of offshore IoT applications are not well served. Utilizing marine energy can advance the adoption of IoT at sea by providing in-situ energy generation, whilst also benefitting from the same technology. The Sustainable Energy Authority of Ireland funded 'BlueBox' project aims to overcome barriers to entry for applying IoT technologies to offshore sensing by developing Ocean IoT (OIoT) hardware and software solutions. Features of the BlueBox system include modular offshore-focussed hardware, no-code configuration and control of peripherals (such as sensors and actuators), duplex transmission of data using multiple media, serverless cloud server architecture, and an edge computing framework. This paper presents an overview of BlueBox, tank tests for system validation of a prototype wave-energy powered ocean-observing platform, and a discussion of future applications of the technology.**

Index Terms—**Blue Economy, Internet of Things, Edge Computing, Renewable Marine Energy, BlueBox™**

I. Introduction

As communication technology advances, contemporary computing models, typically deployed in onshore applications, can be applied to offshore assets. These models enable computation and data storage to be transferred from remote devices to the cloud (i.e. onshore, managed servers). The Internet of Things (IoT) connects devices to the cloud via the internet. Examples of IoT devices can include anything from domestic refrigerators to industrial robots. A distinguishing feature of IoT versus classical internet communications is that IoT devices can be constrained by computing power and communication bandwidth. As such, lightweight communication protocols and advanced computing paradigms have been developed to overcome these limitations.

Edge computing is a distributed computing paradigm for sharing data storage and processing needs between the equipment used to generate and act upon data (such as an autonomous sensing platform) and the cloud. According to [1] the advantages of edge computing include:

1) Reduced communications bandwidth requirement: by processing data on board the distributed system, it is possible to reduce the transmission of non-essential data.
2) Increased response speed: by eliminating the transmission latency to the cloud and back, rapid control actions can be made with on-board computations.
3) Increased reliability: by having control and data processing systems on board, the distributed system becomes robust to network interruptions.

Traditionally, an offshore sensor, such as a wave buoy, would process and store all of its data, transmitting a limited amount, if possible, and storing the rest for physical collection later. Advances in communication technology allow offshore assets to access the internet more easily, but the degree of connectivity can still be limited and variable. Edge computing therefore provides an excellent paradigm to allow offshore assets to utilize cloud infrastructure [2]. By utilizing edge computing, transmission and energy budgets can be optimized [3], and large scale or computationally expensive applications, such as big data analysis [2] or artificial intelligence (AI) [4], can be offloaded to the cloud. DARPA's 'Ocean of Things' demonstrated the capabilities of edge computing in drifter buoys developed for the programme. Sampled data was scanned onboard for anomalous events, which were then transmitted to the cloud by satellite [5]. Data from thousands of these low-cost buoys, working in unison, was collated and analysed in the cloud.

Leveraging IoT brings several other benefits for offshore applications:

Part of a special issue for EWTEC 2023. Original version published in EWTEC 2023 proceedings at https://doi.org/10.36688/ewtec-2023-466.

Manuscript submitted 9 January 2025; Accepted 5 February 2025. Published 31 May 2025.

This article has been subject to single blind peer review by a minimum of two reviewers.

This work was supported by the Sustainable Energy Authority of Ireland (SEAI) under grant 21/RDD/733'

M. B. R. Topper is with Data Only Greater, Maynooth, Ireland (e-mail: mathew.topper@dataonlygreater.com).

N. A. M. M. Jarnoux, R. Costello, S. Giorgi and B. Kennedy are with Ocean Wave Venture Ltd, Cork, Ireland (e-mails: nicolas.jarnoux@wave-venture.com, ronan@wave-venture.com, simone.giorgi@wave-venture.com and ben@wave-venture.com).

C. Murtagh is with Sea Power Ltd., Enniscrone, Ireland (e-mail: cian.murtagh@seapower.ie).

Digital Object Identifier: https://doi.org/10.36688/imej.8.139-147

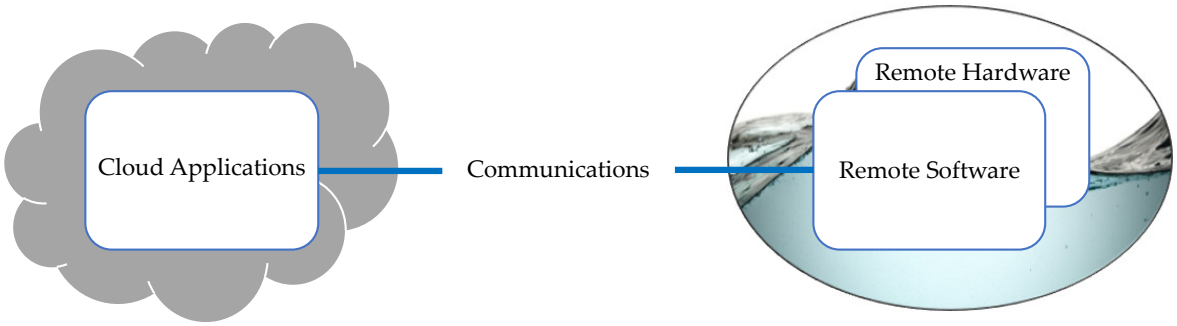

Fig. 1. The BlueBox system design is divided into the remote device (hardware and software) and the cloud applications that interact with the device. The remote device and the cloud share responsibility for the communications design.

- IoT enables 'over the air' (OTA) updates, allowing software to be updated in situ without requiring removal and re-installation, which require expensive marine operations [6].
- Using the cloud to store data reduces the storage requirements of the offshore device, reduces the risk of data loss, and allows advanced analytics [7].
- Data from external sources can be integrated (e.g. satellite data, weather forecasts), expanding the situational awareness of a system [8].

The duration and capabilities of systems that utilize IoT and edge computing offshore have, traditionally, been limited by fixed energy budgets (e.g. primary battery capacity). Utilizing in-situ power generation from marine renewables reduces this limitation, increasing duration and enabling advanced computing problems (such as 'big data') to be readily applied in the marine environment [2]. In [9] the opportunity for marine energy experts and the ocean observation community to improve the capabilities of autonomous sensing platforms is recognized.

Several authors have noted the importance of standardized systems for enabling wide-ranging adaptation of ocean IoT systems (e.g. [9], [10]). At present there is no standardized IoT platform that addresses the control, data processing and communications needs of ocean data systems. Currently, developers must assemble their own systems from low-level components and develop large and broad software infrastructures.

The aim of the BlueBox project is to harness the power of distributed computing to advance the use of data science in the ocean. Developing a dedicated offshore Internet of Things platform can reduce costs and enhance the value of data collected. Additionally, by enabling on-board renewable energy power generation, BlueBox will increase the feasibility of applying advanced computational models in offshore applications. BlueBox is a two-year project funded by the Sustainable Energy Authority of Ireland (SEAI), which will complete in May 2024. As a publicly funded project, BlueBox will endeavour to release outputs as open-source wherever possible.

Fig. 1 represents an overview of the major components of the BlueBox system. The remote components consist of hardware designed for offshore use and a software system to enable data collection, energy management, control of the host platform, and communication with the cloud system. The cloud system is responsible for storing and analysing the data collected from the remote system, observing the system status and enabling users to issue control commands. The technologies used to communicate between the two systems form the final component.

The Methods section of this article describes the components illustrated in Fig. 1 in greater detail. Subsequently, the Testing section will describe tank tests planned to validate the system. Some further work enabled by the BlueBox system is described in the Future Vision section, followed finally by the conclusions.

II. Methods

A. Communications

Communications in offshore environments is challenging due to limited connectivity and, potentially, scarce energy resources. Conversely, in test environments, high-bandwidth connections and ample energy resources may be readily available. A hierarchy of technologies can be applied to provide networking access ranging from cabled connections, Wi-Fi or Bluetooth (when a high-speed link is nearby), to cellular networks, very-high-frequency (VHF) or ultra-high-frequency (UHF) radio (such as LoRaWAN), and satellite communications for longer-distance communications [11]. Selection of the most appropriate medium is a balance of cost, availability, bandwidth, and energy budget. BlueBox offers a variety of communication technologies for different use cases.

Fig. 2 illustrates the different media that BlueBox can utilize. When available, BlueBox will communicate through the internet, using an Ethernet connection first, followed by Wi-Fi and then through cellular networks. If a cellular network cannot be found, BlueBox will communicate with the cloud using the Iridium satellite network [12].

When the internet is available, BlueBox will use the MQTT protocol [13] to send and receive messages. MQTT (originally an initialism of MQ Telemetry Transport) is a lightweight client-server protocol, where clients publish and subscribe to named 'topics'. Arbitrary data can be sent by a publisher, which is then delivered to any number of subscribers. It contains features designed for intermittency of connections, such as a 'Quality of Service' (QoS) flag, that allows

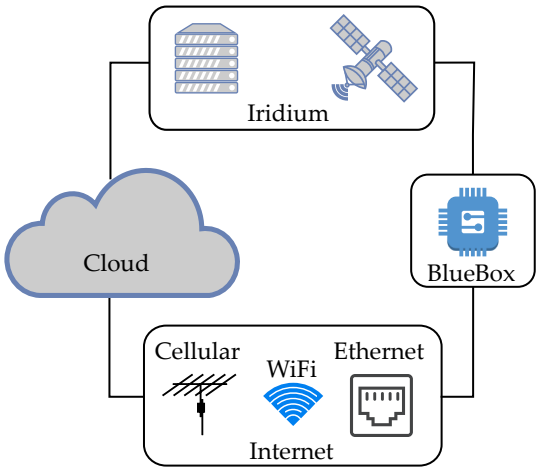

Fig. 2. Communication technologies used to connect the BlueBox hardware to the cloud

messages to be sent without verification (QoS 0) to a guarantee that the message has been received once and only once (QoS 2). Clients can also form 'persistent' connections, where the server will queue messages to be sent if the client is disconnected; if a connection is not persistent, the messages are discarded. Clients may also pre-publish a 'last-will' message that is sent if an unexpected disconnect occurs.

Due to the high monetary cost of transmission using satellite communications, minimizing the size of the message payload is essential. Even when transmitting over the internet, smaller-payload messages are less likely to be corrupted in transit. BlueBox achieves payload reduction through both a technological and operational approach.

The Concise Binary Object Representation (CBOR) is a binary-data serialization format that allows the concise transmission of data structured as key–value pairs [14]. When decoded, CBOR messages are similar to JSON (JavaScript Object Notation). In [15], it was shown that CBOR had a 22.4% average reduction in size compared to JSON over a set of test documents, with a maximum recorded reduction of 43.2%. Further size reductions can be achieved from the use of CBOR tags, which allow users to create custom data types. For example, coordinates described as key-value pairs might be represented as follows:

```
{
  "latitude": 47.21473,
  "longitude": -1.5532045
}
```

Using a custom CBOR tag, this can be reduced to an integer identifier and a length 2 array:

```
CBORTag(4000, [47.21473, -1.5532045])
```

Operationally, the sharing of the data schema between the remote device and the cloud allows further reduction in payload size. A no-code configuration is prepared by the BlueBox user prior to the build step for the remote platform software (see sec. II-B2). 'No code', here, means that no programming is required, but configuration files still need to be written, using a given format. For example, a sensor might be configured as shown in Fig. 3.

```
# Identifies sensor metadata
type: sensor
# The name of the sensor
name: accelerometer

driver:
  # The driver type for the sensor
  class: bb::AnalogInput
  # The physical input pins
  pins: [ A0, A1, A2 ]
  # The names of the inputs
  names: [x, y, z]

# Sampling period
period: 100ms

# Expected bounds on the readings
min_expected: [0, 1, 2]
max_expected: [2, 3, 4]

# Enable inclusion of user code
enable_callback: true
```

Fig. 3. Example no-code configuration for an accelerometer

The configuration is processed during the build step for the platform software. During the build step, the user must also be logged onto the cloud system to allow for provisioning. Provisioning identifies the remote software to the cloud and secures the connection between them. Concurrently, the configuration data is shared with the cloud system. The shared state can then eliminate the need to transmit metadata. For instance, for the sensor defined above, an example CBOR payload would simply be:

```
CBORTag (
  VALUE_TAG, # A single record
  {
    TIME_KEY: 1234567890
    ENTITY_KEY: 0 # Auto generated
    VALUES_KEY: [0, 1, 2]
  }
)
```

B. Remote platform

1) Hardware: The remote hardware component of the BlueBox system comprises the embedded controller plus a depth rated enclosure, as seen in Fig. 4. The BlueBox controller is designed to be expandable. Essential hardware is included on a main module, while optional add-on modules provide more specialized functionality where needed.

The system has been designed with prototyping in mind and aims to address user needs in several phases of product development and prototyping, in addition to the final deployment in the sea. The system is designed to be able to address the needs of:

- Bench testing
- Tank testing

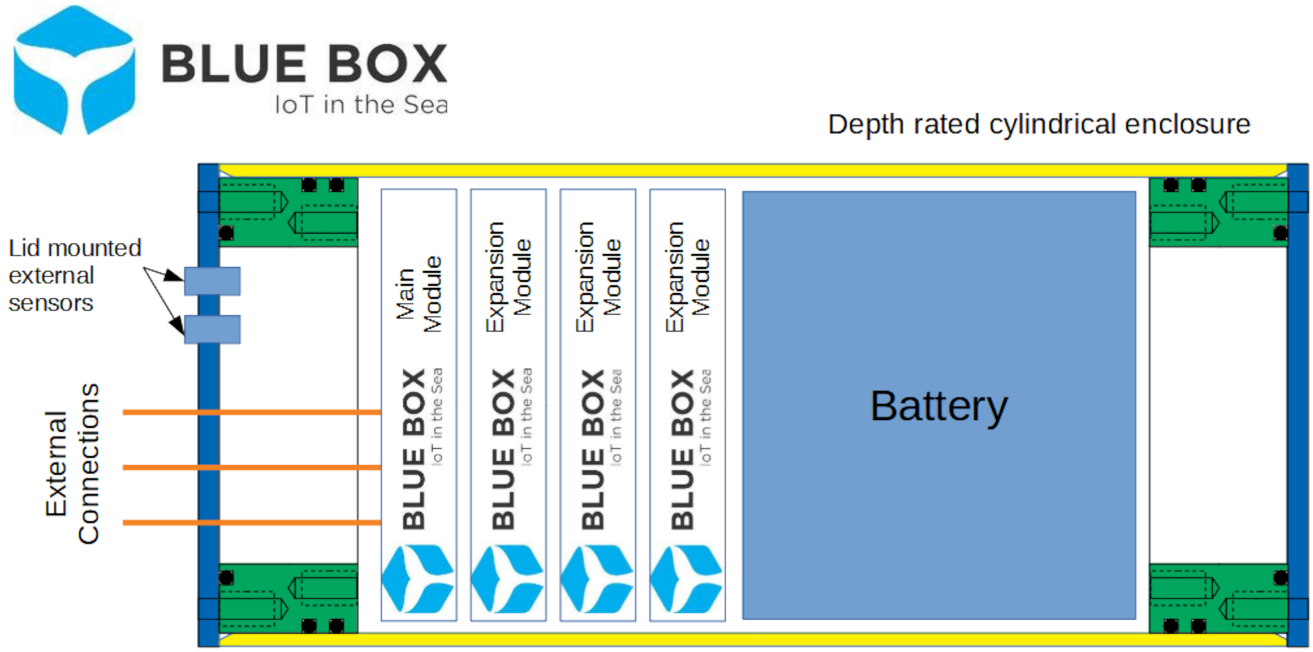

Fig. 4. Hardware layout sketch showing depth rated cylindrical enclosure, BlueBox main module, and several expansion modules.

- Calm water testing
- Supervised sea trials
- Deployment at sea

The BlueBox main module includes processing, communications, storage, selected on-board sensors, power management, general-purpose input/outputs (GPIO) pins, and an expansion bus for communications with the add-on modules. The integrated wireless communications hardware includes Wi-Fi, LTE (4G) cellular, and satellite. The integrated wired communications hardware includes USB for hands-on configuration and data transfer and RS485 for applications where a longer umbilical cable is needed. Integrated sensors include:

- GPS/GNSS
- Inertial measurement unit
- External temperature and pressure
- Enclosure internal temperature, pressure, and humidity
- Water ingress detection
- Battery voltage and current

The initial suite of optional expansion modules include:

- Additional analogue and digital inputs
- Load switching
- Brushless DC (BLDC) motor driver
- Generator control and battery charging
- General purpose user modifiable prototyping module

The enclosure is depth rated to maximize the range of possible applications and installation locations. The installation locations considered include:

- Exterior, above water (e.g. on deck, on superstructure)
- Exterior, below water (e.g. fixed to hull exterior, in flooded hull, where enclosure is integral to hull, or stand alone submerged deployment)
- Interior (e.g. below deck, inside hull)

2) Software: The remote platform software is a real-time operating system (RTOS) [16] that is responsible for managing inputs and outputs to or from sensors and actuators, providing automatic control, and managing communications with the cloud. BlueBox also features a no-code configuration system (as seen in the sec. II-A), which generates the embedded code and installs it on the platform hardware.

The BlueBox RTOS is a C++ framework based on the ARM MBED OS (os.mbed.com). The RTOS schedules different threads to poll sensors, record data on local storage, send data over the network, as well as control and manage the state of the device. The system can also be extended with user-defined code (written in C++).

```
# Start a new project
> bluebox init

# Monitor configuration files and
# automatically run the code generation
# step
> bluebox refresh --watch

# Run the code on a test device
> bluebox build --flash

# Deploy the final code on devices
> bluebox build \
  --build-type release \
  --provision-new \
  --flash
```

Fig. 5. Commands for creating and building a BlueBox project, provided by the command line tool

A desktop tool is provided to manage BlueBox projects. It is responsible for code generation by pro-

cessing the configuration provided by the user to generate C++ code. The generated code includes callback functions, where user code can be added. To aid the user to develop the configuration and callbacks, the tool also provides integrations with well-known integrated development environments (IDEs) such as Visual Studio Code. Finally, the tool is used to build the project, deploying the code and interfacing with the cloud system to provision the device.

The project lifecycle can be managed using the BlueBox command line interface, as shown in Fig. 5. For example, using the configuration snippet shown in sec. II-A, the code generation step will produce the file shown in Fig. 6.

```
#include "bluebox.h"

void on_accelerometer_read(
  const Sensor_3f& accelerometer,
  Sensor_3f::ValueArray& values
)
{
    // User code goes here.
    // The callback arguments are:
    // - A reference to the sensor
    //   being read
    // - An array of recorded values
    //   to be manipulated
}
```

Fig. 6. Example callback generated from the configuration shown in Fig. 3

In the file, the user can add custom code to the provided `on_accelerometer_read` callback to update the values recorded by the sensor. Similar callbacks are provided for various aspects of the system; for example, a sensor reading, a change in the state of the device, or a periodic callback to control the device behaviour (e.g. monitoring a propeller throttle or the device GPS location).

3) Generation Control: The BlueBox system is targeted at renewably powered Blue-Economy applications. These applications are generally low power, typically in the 10 W to 1000 W range, and commonly rely on battery power. The BlueBox generation-control-and-power-management module is sized for a maximum power input in the 100 W to 1000 W range, and has been designed to accept DC power inputs from solar panels and micro-wind and -wave generators, and to charge lithium-ion batteries. The module features overvoltage and overcurrent protections on the generator and battery side, and a number of suitable maximum-power-point-tracking (MPPT) algorithms are under development. For novel applications, the MPPT algorithms can be replaced by user written functionality.

C. Cloud

1) Serverless: Cloud computing is a distributed computing model where all or a significant portion of an application's computing and storage occurs on centralized servers, accessed via the internet. Early cloud-hosted applications required software developers to be responsible for every aspect of the application, from network management to application scaling; this often necessitated multidisciplinary teams. The late 2000s saw the invention of a new cloud computing paradigm, known as 'serverless' [17]. Serverless computing breaks an application into its component resources, such as functions, storage, etc., removing the need for application developers to handle networking or application scaling. Additionally, these services are often charged on a pay-as-you-go basis, allowing cost-effective development of new applications.

A disadvantage of the serverless model is that a developer must provision numerous interlinked resources. Additionally, it is important for software developers to avoid creating 'orphaned' services that could be generating unexpected expenses. To help manage the resources of a serverless application, a process known as 'Infrastructure as Code' (IaC) was developed [18]. IaC allows a developer to describe the required resources for their application in code and then deploy and destroy those resources, as desired. Originally, IaC was described in declarative, domain-specific languages. Recent advances allow IaC to be written using the same programming language as the application, greatly reducing the barrier to entry for software developers.

For BlueBox, Pulumi (www.pulumi.com) was chosen as the IaC provider. Pulumi offers IaC using multiple programming languages, that can be used to provision cloud infrastructure from multiple providers (e.g. Amazon Web Services (AWS), Google Cloud Platform, etc.). Choosing Pulumi avoids vendor lock-in from using an infrastructure provider's in-house solution (such as AWS's Cloud Development Kit), which enables the BlueBox project to change or mix cloud providers as required.

2) Primary architecture: Following review of the available cloud providers, and any dedicated IoT services offered, AWS was chosen as the primary provider for BlueBox. AWS has an established managed MQTT broker (called 'IoT Core'), AWS offers an (almost) entirely pay-as-you-go cost model, and AWS has enough features to adapt to almost any workload (e.g. web-hosting, AI, etc.). For simplicity, BlueBox aims to use a single cloud provider as much as possible, but for certain aspects (such as specialized databases) other cloud service providers need to be used.

In IoT nomenclature, a 'tenant' represents a logical separation of data processing and storage. The 'silo' tenancy model duplicates all infrastructure (i.e. databases, etc.) per tenant, while the 'pooled' model shares infrastructure between tenants. BlueBox uses a mixed tenancy model, using siloed infrastructure for data processing and storage and pooled services for communications (such as the MQTT broker) and the frontend web-application.

Fig. 7 illustrates a simplified version of the AWS architecture for data input, storage, and output, for a tenant. Adding incoming data to an object storage

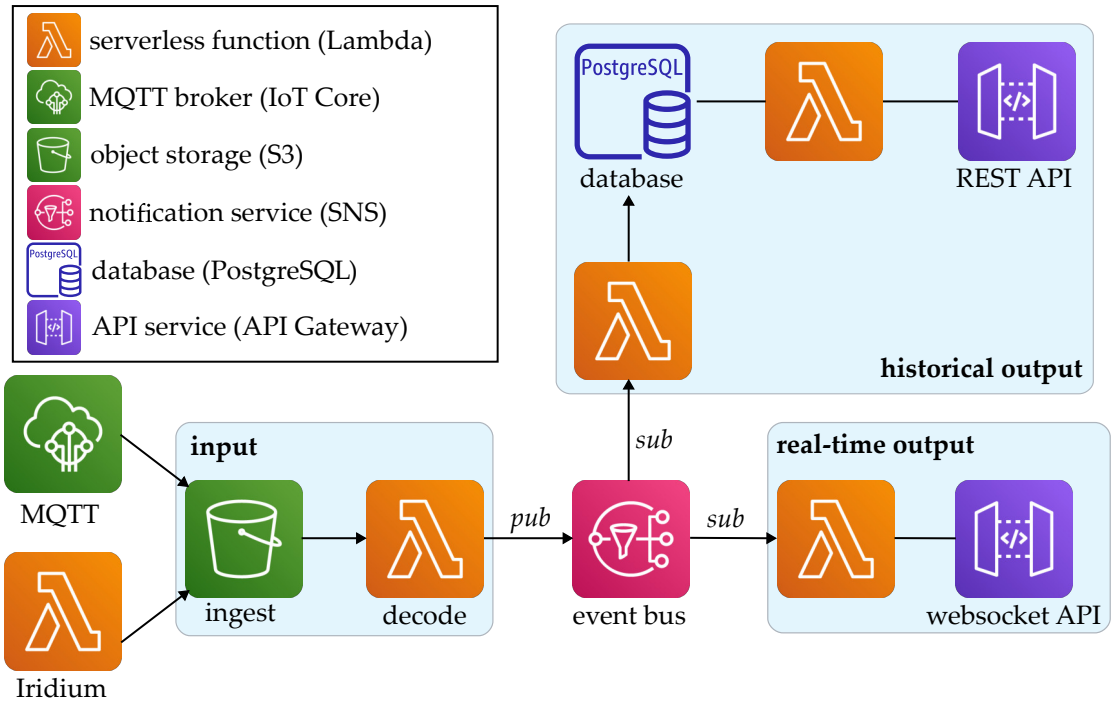

Fig. 7. Simplified illustration of the cloud resources that receive, decode, store and transmit tenant data. The key in the top left shows the service type for each symbol, with the service name in parentheses. 'Pub' and 'sub' refer to publishers-of and subscribers-to events transmitted through the event bus.

service (S3 bucket) triggers the processing pipeline. Fig. 7 shows data being added by the IoT Core MQTT broker and a serverless function receiving Iridium Short Burst Data (SBD) messages. When data is added to the bucket, a serverless function is triggered which will decode the message depending on its format. The serverless function then publishes the data (now in a homogenous format) to a notification service (SNS) used as an event bus; there can be multiple subscribers to the notification service. Fig. 7 shows two subscribed serverless functions, one for handling real-time data and another for handling data for historical analysis.

The serverless function for real-time output sends data to clients via a WebSocket application programming interface (API). WebSocket is a full-duplex (i.e. allows simultaneous 2-way communication) web technology [19] used for persistent real-time connections.

The serverless function responsible for the historical analysis adds the data received from the event bus to a database. The *ideal* requirements for a database used to store user-generated data are as follows:

1) Time series compatible, preferably offering server-side aggregation
2) Geographical data compatible, preferably with server-side spatial functions
3) Flexible data model, which can accept data in any structure
4) Flexible queries, that allow searching across arbitrary fields
5) Reasonable storage limits
6) Low cost
7) Serverless deployment, allowing scaling in storage and computational requirements

No single serverless database is available that can fulfil all of the above requirements, and BlueBox may need to support different types of databases based on the needs of particular users. A selection of databases that meet the requirements for storing time and geographical data, and allow flexible queries, are discussed below.

Timescale (www.timescale.com), is an extended PostgreSQL database, optimized for time-series data. AWS Aurora Serverless (aws.amazon.com/rds/aurora) offers unmodified PostgreSQL but has the ability to autoscale computational requirements, which Timescale does not. Any PostgreSQL database must have its data model defined in advance, in a tabular format, which can be challenging for complex data structures.

NoSQL databases can handle complex data structures, as they do not require a schema to be defined in advance. MongoDB Atlas Serverless (www.mongodb.com/atlas) is a popular NoSQL database service, but it is limited to 1 TB of storage capacity. Work is currently ongoing to determine which database offers the best compromise solution to the requirements of BlueBox.

A second serverless function is used to read the database and serve a REST API. A REST API uses HTTP methods (GET, POST, etc.) to exchange data with clients from an endpoint (URL) that describes the resource (e.g. example.com/tenant/device/sensor). Unlike WebSocket, REST APIs do not provide a persistent connection, so are more suited to querying historical data.

BlueBox provides a frontend (i.e. a website) to manage tenants, users and devices, and interact with the APIs shown in Fig. 7. The frontend is served from AWS using a Content Delivery Network (CDN) which duplicates files across multiple locations. This helps to maintain a consistent user experience regardless of the location from which the site is accessed.

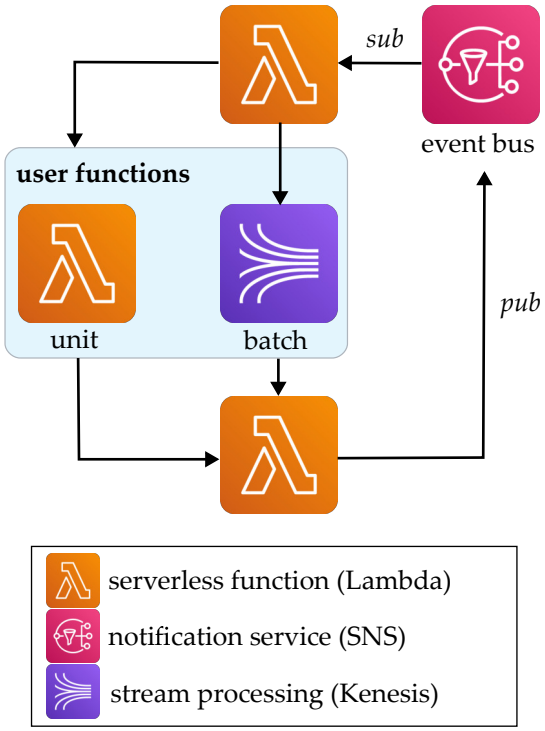

Fig. 8. User functions can be triggered from messages sent to the event bus. The key shows the service type for each symbol, with the service name in parentheses. 'Pub' and 'sub' refer to publishers-of and subscribers-to events transmitted through the event bus.

3) User functions: Another feature of the BlueBox cloud system is the ability for users to apply arbitrary functions to their data. Fig. 8 shows how these functions can be hooked into the event bus in the primary architecture (Fig. 7). The functions can either be applied to a single parameter as it is sampled (unit) or to many samples and parameters in unison (batch) by using stream processing.

Stream processing is a suite of tools for operating on large volumes of data without requiring separate temporary storage. Multiple inputs can be reduced by sampling or aggregation, reducing storage requirements and allowing more complex metrics to be generated.

User functions can be authored in any language supported by AWS Lambda functions. Once the functions have processed the data, the results will be sent back to the event bus.

D. Security

Security is vitally important for a system reliant on the publicly-facing internet, such as BlueBox. 'Data protection by design and by default' is also a requirement of the General Data Protection Regulation (GDPR) under EU law [20]. When secure, data should be:

- inaccessible to unauthorized users (confidentiality)
- protected from interception or modification (integrity)
- transmittable without impedance (availability)

In the cloud, BlueBox is secured using the zero-trust model. Essentially, zero trust implies that all resources are inaccessible by default and that access is explicitly granted only when required. AWS denies access to all resources by default, and specialized 'policies' must be created to allow access. This can be done dynamically, allowing access for time-limited periods only, while a user is authenticated.

When communicating between the cloud and the remote device, BlueBox sends data using encrypted protocols when available. For example, IoT Core uses TLS (transport layer security) to secure MQTT connections. To avoid excessive computational requirements for the remote device, the payloads of messages are not encrypted. If BlueBox data were sent without an encrypted protocol, therefore, messages could be intercepted and read. Although this is not possible for the current BlueBox architecture, a future user may require an insecure communication medium, such as radio.

Even if a user chooses to send data without encryption, it remains important to avoid spoofing of messages (i.e. preventing an attacker pretending to be the user of the system). To ensure that messages have originated from a trusted source, a digital 'signature' is attached to each message. A signature is created by cryptographically signing a 'hash' (a reduction of the payload via a known algorithm) with a private key. A public key can then be used to decrypt the signature and verify it (by comparing the unencrypted hashes).

III. Testing

A generic Blue-Economy application has been chosen as a test bed for the BlueBox system. Fig. 9 is a schematic for a generic autonomous underwater vehicle which will be used for system validation tests. The BlueBox system will be used to control the vehicle thrusters and variable buoyancy, to accept commands and send data to the cloud system, to monitor the internal status of the system, and to collect environmental data. The BlueBox enclosure used for these tests will be rated to 20 m and is integrated with an underwater vehicle. (This is an example of an application where the BlueBox enclosure is integral to the device hull.) In future, a 200 m rated version of the BlueBox enclosure will be available. The BlueBox enclosure is fitted with wet-mate connectors for connection to the external components.

The actuators included in the vehicle are two pumps that are used to control the variable buoyancy system and three thrusters (main forward thruster and vertical and horizontal bow thrusters). These actuators are all controlled via a BLDC expansion module. The test vehicle is a sensor-carrying platform and can, in principle, be adapted to include a wide range of relevant sensors. For the tests, conductivity, temperature, and depth (CTD) measurements will be made. In addition to the built-in sensors mentioned in sec. II-B1, the test vehicle will also include a forward speed sensor. Planned testing of high-level user commands includes basic navigation and depth control.

IV. Future vision

The BlueBox project will establish a platform for the use of IoT technologies in the ocean, supported

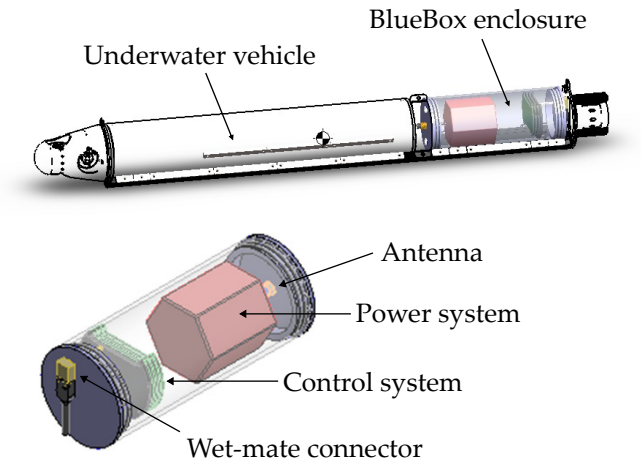

Fig. 9. Schematic of the vehicle to be used in system tests for a wave-energy powered ocean-observing platform. The upper diagram shows the integration of the BlueBox hardware enclosure with the underwater vehicle. The lower diagram shows the layout of the components within the enclosure.

by in-situ energy generation from marine renewables. The scope of the project does not include investigation into specific applications of the technology. This section provides an overview of some potential outcomes from adaption of IoT in the Blue Economy.

While this article has focused on a single system, typically, IoT is concerned with communication between many devices. It follows, therefore, that IoT systems can help enable advanced coordinated control between devices. The advantages of coordinated control in wave energy farms have been recognized in [21]. Coordination of ocean observing platforms offers a paradigm shift in the technology, moving from expensive single platforms to cheaper multi-unit swarms [9]. Coordination of heterogenous systems may present additional benefits, from sensor fusion [9], to energy generation [22], to enhancing bandwidth [23]. A challenge to be overcome for coordination of underwater vehicles is cheap, reliable underwater communications. Recent improvements in quality and reduction in cost of underwater acoustic modems may make them a viable option [24].

Leveraging big data [2] is another benefit of ocean IoT systems. This enables the use of advanced computational models such as predictive analytics [25] and AI [26]. Utilizing AI is projected to generate significant improvements in the autonomy of offshore renewable energy robotics [26] and wave energy control algorithms [27]. Practical implementations of using AI at sea remain scarce [28] due to energy and computational constraints – barriers that IoT systems in combination with marine energy can overcome.

The expanding range and bandwidth of communication technologies for offshore applications, combined with efficient IoT technologies, makes access to live telemetry increasingly practicable. This data can be used to populate digital twins (i.e. live numerical models of offshore assets) which can be used to determine the health of a system in real time. The use of digital twins for offshore renewable energy can reduce scheduled maintenance and predict unexpected failures, reducing costs and increasing availability [29]. The application of digital twins for marine energy applications has been investigated in studies such as [30], [31]; however, practical examples are rare. A standardized IoT platform such as BlueBox will enable wider adoption of digital twins in the marine energy industry.

V. Conclusion

This article presents the findings to date of the Sustainable Energy Authority of Ireland funded 'BlueBox' project. BlueBox is developing a standardized internet of things (IoT) platform for use in offshore applications and can harness marine energy to increase energy budgets. The use of IoT systems in the marine energy industry can also bring several additional benefits, including more reliable data collection, over-the-air control system updates, and access to advanced computation models such as edge computing.

BlueBox consists of hardware designed for use in the offshore environment and which provides straightforward interfacing with common sensors and actuators. The software which operates this hardware accepts no-code configuration for sensors, actuators and system control. The system can also be extended with custom user code. The remote system communicates with an onshore cloud application using IoT technologies. The cloud application is designed using a serverless architecture and is deployed, mainly, to AWS. Communication between the two systems uses a hierarchy of media, from Ethernet to satellite networking. BlueBox will be tested as part of the system validation of a prototype wave-energy powered ocean-observing platform.

BlueBox can enable novel applications of IoT technologies, such as coordinated control of multiple devices. Advanced analytical models and artificial intelligence require large scale data collection, which is streamlined by IoT. Additionally, reliable transmission of live telemetry will accelerate practical application of digital twins for marine energy converters.

References

[1] W. Shi and S. Dustdar, "The Promise of Edge Computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016.
[2] M. Jahanbakht, W. Xiang, L. Hanzo, and M. R. Azghadi, "Internet of Underwater Things and Big Marine Data Analytics – A Comprehensive Survey," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 904–956, 2021.
[3] S. Wang, "Edge Computing: Applications, State-of-the-Art and Challenges," *Advances in Networks*, vol. 7, no. 1, p. 8, Nov. 2019.
[4] D. Mitchell, J. Blanche, O. Zaki, J. Roe, L. Kong, S. Harper, V. Robu, T. Lim, and D. Flynn, "Symbiotic System of Systems Design for Safe and Resilient Autonomous Robotics in Offshore Wind Farms," *IEEE Access*, vol. 9, pp. 141 421–141 452, 2021.
[5] E. Cocker *et al.*, "Low-Cost, Intelligent Drifter Fleet for Large-Scale, Distributed Ocean Observation," in *OCEANS 2022, Hampton Roads*. Hampton Roads, VA, USA: IEEE, Oct. 2022, pp. 1–8.
[6] F. Mahfoudhi, A. K. Sultania, and J. Famaey, "Over-the-Air Firmware Updates for Constrained NB-IoT Devices," *Sensors*, vol. 22, no. 19, p. 7572, Jan. 2022.
[7] K. Gurjit and T. Pradeep, *Handbook of Research on Big Data and the IoT*. IGI Global, Mar. 2019.
[8] T. Snyder and G. Byrd, "The internet of everything," *Computer*, vol. 50, no. 06, pp. 8–9, Jun. 2017.
[9] C. Whitt *et al.*, "Future Vision for Autonomous Ocean Observations," *Frontiers in Marine Science*, vol. 7, 2020.

[10] G. Xu, Y. Shi, X. Sun, and W. Shen, "Internet of Things in Marine Environment Monitoring: A Review," *Sensors*, vol. 19, no. 7, p. 1711, Jan. 2019.
[11] F. Vannieuwenborg, S. Verbrugge, and D. Colle, "Choosing IoT-connectivity? A guiding methodology based on functional characteristics and economic considerations," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 5, p. e3308, 2018.
[12] C. Gomez, S. M. Darroudi, H. Naranjo, and J. Paradells, "On the Energy Performance of Iridium Satellite IoT Technology," *Sensors*, vol. 21, no. 21, p. 7235, Jan. 2021.
[13] G. C. Hillar, *MQTT Essentials - A Lightweight IoT Protocol*. Packt Publishing Ltd, Apr. 2017.
[14] C. Bormann and P. E. Hoffman, "Concise Binary Object Representation (CBOR)," Internet Engineering Task Force, Internet standard RFC 8949, Dec. 2020.
[15] J. C. Viotti and M. Kinderkhedia, "A Benchmark of JSON-compatible Binary Serialization Specifications," Jan. 2022.
[16] B. Amos, *Hands-On RTOS with Microcontrollers: Building Real-Time Embedded Systems Using FreeRTOS, STM32 MCUs, and SEGGER Debug Tools*. Packt Publishing Ltd, May 2020.
[17] P. Sbarski, Y. Cui, and A. Nair, *Serverless Architectures on AWS, Second Edition*. Simon and Schuster, Mar. 2022.
[18] B. Campbell, *The Definitive Guide to AWS Infrastructure Automation: Craft Infrastructure-as-Code Solutions*. Berkeley, CA: Apress, 2020.
[19] A. Melnikov and I. Fette, "The WebSocket protocol," Internet Engineering Task Force, Proposed standard RFC 6455, Dec. 2011.
[20] "General Data Protection Regulation (GDPR) – Official Legal Text," https://gdpr-info.eu/.
[21] G. Bacelli, P. Balitsky, and J. V. Ringwood, "Coordinated Control of Arrays of Wave Energy Devices—Benefits Over Independent Control," *IEEE Trans. Sustain. Energy*, vol. 4, no. 4, pp. 1091–1099, Oct. 2013.
[22] B. Polagye, A. Stewart, J. Joslin, P. Murphy, E. Cotter, P. Gibbs, M. Scott, S. Henkel, and S. Matzner, "Final Report: An Intelligent Adaptable Monitoring Package," Univ. of Washington, Tech. Rep. DOE-UW-0006788-1, Apr. 2020.
[23] A. Kabanov and V. Kramar, "Marine Internet of Things Platforms for Interoperability of Marine Robotic Agents: An Overview of Concepts and Architectures," *Journal of Marine Science and Engineering*, vol. 10, no. 9, p. 1279, Sep. 2022.
[24] F. Campagnaro, F. Steinmetz, and B.-C. Renner, "Survey on Low-Cost Underwater Sensor Networks: From Niche Applications to Everyday Use," *Journal of Marine Science and Engineering*, vol. 11, no. 1, p. 125, Jan. 2023.
[25] N. Gorostidi, V. Nava, A. Aristondo, and D. Pardo, "Predictive Maintenance of Floating Offshore Wind Turbine Mooring Lines using Deep Neural Networks," *J. Phys.: Conf. Ser.*, vol. 2257, no. 1, p. 012008, Apr. 2022.
[26] D. Mitchell, J. Blanche, S. Harper, T. Lim, R. Gupta, O. Zaki, W. Tang, V. Robu, S. Watson, and D. Flynn, "A review: Challenges and opportunities for artificial intelligence and robotics in the offshore wind sector," *Energy and AI*, vol. 8, p. 100146, May 2022.
[27] L. Li, Y. Gao, D. Z. Ning, and Z. M. Yuan, "Development of a constraint non-causal wave energy control algorithm based on artificial intelligence," *Renewable and Sustainable Energy Reviews*, vol. 138, p. 110519, Mar. 2021.
[28] R. Kot, "Review of Collision Avoidance and Path Planning Algorithms Used in Autonomous Underwater Vehicles," *Electronics*, vol. 11, no. 15, p. 2301, Jan. 2022.
[29] I. Errandonea, S. Beltrán, and S. Arrizabalaga, "Digital Twin for maintenance: A literature review," *Computers in Industry*, vol. 123, p. 103316, Dec. 2020.
[30] P. Qian, B. Feng, D. Zhang, X. Tian, and Y. Si, "IoT-based approach to condition monitoring of the wave power generation system," *IET Renewable Power Generation*, vol. 13, no. 12, pp. 2207–2214, 2019.
[31] E. Katsidoniotaki, F. Psarommatis, and M. Göteman, "Digital Twin for the Prediction of Extreme Loads on a Wave Energy Conversion System," *Energies*, vol. 15, no. 15, p. 5464, Jan. 2022.