Home Automation Using Augmented Reality (HAAR)

Ali A. El-Moursy1 · Fadi N. Sibai2 · Jahanzeb Rehman1 · Omar M. Gouda1 · Abdelrahman T. Gaber1 · Ahmed M. Khedr3

Accepted: 14 November 2021 / Published online: 20 January 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
The development of Smart Home Controllers has seen rapid growth in recent years, especially for smart devices, that can utilize the Internet of Things (IoT). However, a large portion of the household devices and appliances already in use, are not IoT enabled, and therefore, requires their default control mechanisms for the devices to operate. This paper proposes a smart appliance controller that uses Augmented Reality, MQTT, and other up-to-date platforms to control aftermarket home appliances in the most efficient manner. The proposed work integrates mobile AR with IoT, to control household appliances, with the help of infrared (IR) signals. The characteristics of the system are evaluated through a series of tests and performance measures. The results of the test highlight the quick response time of MQTT for the implementation of a Home Automation System, when compared to the request–reply protocol: CoAP (4 times as fast).

Keywords Augmented reality · IoT · Infrared · MQTT · Smart home controller

* Ahmed M. Khedr
akhedr@sharjah.ac.ae

Ali A. El-Moursy
aelmoursy@sharjah.ac.ae

Fadi N. Sibai
fsibai@pmu.edu.sa

Jahanzeb Rehman
jrehman@sharjah.ac.ae

Omar M. Gouda
U19104867@sharjah.ac.ae

Abdelrahman T. Gaber
U15101080@sharjah.ac.ae

1 Department of Computer Engineering, University of Sharjah, Sharjah, UAE
2 College of Computer Engineering and Science, Prince Mohammad Bin Fahd University, Al-Khobar, Saudi Arabia
3 Department of Computer Science, College of Computing and Informatics, University of Sharjah, Sharjah 27272, UAE
1 Introduction

The IoTs has become a cornerstone of automation systems development. An estimate of 127 new devices are being connected to the internet every second [1], which highlights the humongous growth of IoT. Subsequently, IoT has also made its way into normal households in the form of home automation. Coined in 1999 by Kevin Ashton [2], the IoT is a heterogeneous network of smart devices or objects, capable of communication and computation. Figure 1 showcases a standard IOT architecture, created by combining prolific data collection, data analysis, and the unified framework of cloud computing. IoT aims at improving productivity and living standards through smart devices interaction [3]. IoT traditionally consists of a number of wireless sensor networks (WSN) and devices with wireless connectivity integrated. While a WSN is created by connecting smart sensing devices that can communicate with each other, the complementary devices, however, are built by combining components that are capable of computation and communication [4].

Smart Home is an IoT domain, which is a network of physical devices that provide electronic, sensor, software, and network connectivity within a home. This network of physical devices has to be installed in accordance to the system’s requirements, and requires wiring along with additional hardware. Within the context of home automation, IoT enables simple, everyday tasks to be automated. Smart devices connected to a network can be turned on and off when commanded, or can be configured to operate at given times in the form of a routine [5]. Figure 2 illustrates a standard smart home architecture. These routines can either be programmed on the devices themselves individually, or through the means of a controller that allows the user to program multiple devices simultaneously through a single terminal. In a more advanced mode of operation, the appliances can also be controlled remotely using artificial intelligence, incorporated for the purposes of energy saving and safety [6]. Home Automation Systems (HAS), provide surveillance of home security systems via WSN, and monitoring of power consumption.

A notable issue, however, is the implementation of HAS in the home itself. A discussion by [8], highlights the methods of implementing HAS in homes, and make them smart. The methods discussed include, retrofitting the existing homes, converting properties according to a buyer’s requirement, or purpose-built a new home. The latter two methods are the most
commonly used ones, as they allow for certain features, such as high-capacity wiring, to be installed during construction. However, due to the high installation costs of smart home technologies, they are mostly installed in luxury-oriented housings, and rarely in economic accommodations. If a customer decides to retrofit their home with HAS, a typical option is to add off-the-shelf products, with networking features, such as a controller. Manufacturers, however, gives customers different options, and not all the products are compatible with each other. With all the open protocols available, manufacturers find it difficult to agree upon a single universal standard for smart home devices [9]. Amidst the attempts to make devices communicate with each other, concerns such as integration of “non-smart” devices, and user experience, are often considered secondary.

"Non-smart" devices in this context can be described as appliances that are manufactured, to be controlled via IR based communication. Examples of such appliances can be television sets, set top boxes, air conditioning units, pedestal fans, humidifiers, air purifiers etc. These appliances have no networking capabilities and usually cannot be controlled by conventional HAS.

Augmented Reality Remote Control (ARRC) introduces a HAS solution that overcomes this challenge, of integrating non-networking home appliances. Conventionally, IR signals are used to transmit the control signals to the appliance from the traditional remote control. They have been used in industrial, scientific and medical applications, since William Herschel’s discovery in 1800, and in short-range wireless communication. Different IR transmission protocols like ITT, NEC, Sharp, Philips RC-5, Sony SIRC, are used to prevent surrounding IR from interfering with remote control signals. The receivers distinguish between protocols based on the carrier wave of the signal [10]. Compounded by the development of IoT technology, a growing number of IoT devices now support remote IR access over the internet. These characteristics of IR transmitter technology is used by the ARRC system to control home appliances that do not have embedded network connectivity or smart capabilities.

Augmented Reality (AR) technology refers to the incorporation of virtual elements when viewing physical environments, with the aim of real time mixed reality implementation [11].
AR technology aims to enhance user experiences by the means of improving interactions with environments when it comes to education, control, entertainment etc. Mobile based AR is an area of development which has become increasingly relevant, because of manufacturers producing high performance mobile phones. AR applications for mobile purposes can be categorized into three categories: location-based AR applications, image recognition-based AR applications, and vision-based AR applications. Location-based AR uses GPS or other telemetry sensors of a device. Image recognition-based or marker based AR provide additional information reliant on recognition results. Vision-based AR integrates augmented results with a real world visual stream [12]. The ARRC system makes use of marker based AR for the implementation of its software application. This paper introduces ARRC, that makes use of IoT for interconnection of the devices involved.

This paper discusses the design and implementation of a smart home automation framework which uses mobile AR, publish–subscribe (pub–sub) based protocol for IoT devices, and IR, in order to control home appliances with no embedded network or smart connectivity. Moreover, it provides a justification for using pub–sub based protocols instead of request–reply based protocols, for the implementation of a HAS. This justification is presented in the form of a quantitative comparison between two protocols using the two different interaction models. The comparisons indicate the significant disparities in the latency between the two protocols, by conducting tests in multiple scenarios.

The objective of our ARRC system is to make a paradigm shift by eliminating the need to have networking capabilities in home appliances. Instead, to make use of the networking capabilities already present in our smartphones along with a single device that can be easily controlled, updated and can control a majority of traditional home appliances. This unique and efficient solution reduces the incompatibility of the communication protocol and the cost of manufacturing.

The main contributions of this system can be summarized as follows:

- Marker based AR control for home appliances.
- Upgrade old and non-network connected appliances to be controlled through the internet.
- A single low cost device designed to control multiple appliances.
- Publish–Subscribe based interaction model for low-latency device communication.
- An adaptable system to work with different control schemes including but not limited to IR, Radio Frequency and/or Bluetooth.

The smart home controllers can be utilized in a variety of applications, including wireless sensor networks, traffic flow prediction, and weather forecasting [13–17]. Table 1 summarizes the abbreviations for the key terms used in the paper.

The remainder of this paper is organized as follows: Sect. 2 presents the proposed system design, Sect. 3 discusses the related work. Possible alternatives for pub–sub protocols are discussed in Sect. 4. Section 5 discusses the implementation and functionalities of the developed ARRC system. Section 6 provides insight into the experimental results with Sect. 7 providing the conclusion.
This section describes the basic function of the system, system components, system flow, and a comprehensive explanation of each component, along with how they communicate. The majority of the processing is performed via the Smartphone App (SA). This reduces the processing load of the microcontroller (MC). By focusing on the user experience, and ease of use and accessibility, the SA allows for a more streamlined method of operation. The use of an IoT-based wireless communication network gives scalability and ease of implementation for the system. The components are designed using the latest iteration of their respective hardware and software platforms, thus ensuring improved performance efficiency and provision of features, when compared to existing solutions.

### 2.1 System Components

The proposed system overview, highlighting the communication exchange between components is illustrated in Fig. 3. The components are:

**Smartphone Application** Designed with the idea of replacing remote controls, the smartphone application allows users to use their smartphones as controllers, in order to control appliances in their home.

**Database** A storage medium intended for IR control codes, which allows the SA to download and the user to upload IR control codes at will.
Wireless Communication Network A robust and low latency communication network based on IoT, in order to facilitate efficient machine-to-machine communication between the components.

Appliance Module A single device designed to control multiple home appliances. Equipped with the necessary communication modules required, the device is able to receive control commands from the SA and forward them to appliances.

2.2 Flow of Operation

Using the mobile application, the user initiates system flow, identifies the appliance module in the room and registers a command through the mobile application. The selected command is collected by the application from the database and sent to the appliance module through a network. The appliance module (AM) is responsible for forwarding the command to the appliances through an IR transmitter. The operational flow is illustrated in Fig. 4.

2.3 Component Operation

This section addresses the operational roles of the major components in the ARRC system and the components interaction with each other.

2.3.1 Smartphone Application

The SA plays a vital role in the system. It is responsible for detecting and identifying AMs, providing a user interface, retrieving commands from the online database and forwarding them to the AM. By using the onboard camera on the smartphone, the application can locate any AM within the room. This is done by placing specially designed marker tags on the AM. The tags have discernible features that allow the application to control multiple appliances. Upon identification, an overlay is generated on the screen, that the user can interact with a touch. The command is then retrieved from the online database containing stored appliance commands, and forwarded to the AM. The SA is also responsible for
Fig. 4 Operational flow of ARRC system
setting up the AM, and monitoring its battery status. The battery level of AM is sent periodically to the SA through the network. Figure 5 highlights the flow of the SA.

2.3.2 Database

The Database serves as a storage for the commands used by the system. The reason for storing the commands on a separate platform is to alleviate any storage limitations on both the smartphone and the AM. The user can upload whatever set of commands they want to use, while also having the option to allow other users to add more commands to the database.

Fig. 5 Operational flow of smartphone application
2.3.3 Wireless Communication Network

For device to device communication, a wireless communication network is required. A duplex connection between the SA and AM is to be established, as both the components need to exchange information with each other. Factors such as range, response time, ease of implementation and device pairing, are considered when choosing a wireless communication network. For the implementation of the ARRC, a server based exchange is selected. Both the smartphone and AM will connect to a server via Wi-Fi in order to exchange information.

2.3.4 Appliance Module

The function of the AM is to act as a communication hub between the SA and the appliances. For appliances using IR based control signals, it converts the commands received from the SA into the appropriate form and transmits them. Designed to run off a portable power supply, it periodically relays the battery level to the SA. AM is equipped with the required transceiver in order to connect to the wireless network, along with the transmitters needed to communicate with the appliances. Any processing required by the AM is handled by a MC. Figure 6 illustrates the flow of operation of the AM.

![Operational flow of appliance module](image-url)
3 Related Work

Numerous studies have been conducted in the past, about the design and implementation of HAS. They are discussed in this section.

In [18], the authors implemented a HAS that relies on a single board computer-based server. The sensors are connected directly to the server, and the server communicates sensor information to the cloud. The user uses a SA, which allows the system to be monitored and controlled by a smartphone if the user is within range of the device. Range limitation is due to the short coverage of Bluetooth used in the design. However, the system did allow for commands to be sent to the server via the cloud. Although the system allows integration with other smart home appliances, it cannot communicate with appliances that have no connectivity capabilities. Our ARRC system can overcome communication range limitations by using a wireless communication network that allows for increased range and scalability at a lower cost.

In [19], the authors presented an embedded smart home control system based on General Packet Radio Services (GPRS) and Zigbee. The system uses a smartphone to act as a gateway to control devices which are connected to the controller via Zigbee. The controller is connected to a GPRS module in order to communicate with a smartphone, and a Zigbee module to communicate with the devices. The system takes advantage of the low power consumption, low cost, and low latency characteristics of Zigbee. However, GPRS is not ideal for HAS implementation given its low data rates, slower speeds, and high latency. Our system solves this issue by using a more cost effective, and lower latency method of communication.

In [20], the authors utilize a Wi-Fi enabled MC to control electrical appliances via relay switches. The system comprises of a user application, a server, a database, and a switching element. The server acts as an exchange point between the relays and database, responsible for relaying switch state information. In order to control the relays, the user application is used to change values on the database. The system is successful in integrating appliances with no connectivity features into its automation scheme. However, the control offered by the system is limited to switching the devices on and off. The ARRC system solves the issue of limited control by using IR transmitters in order to provide a broader range of control over home appliances.

In [21], the proposed system is able to control appliances that require IR based control protocols and accommodate appliances without networking features. This is accomplished by interfacing an IR transceiver with a MC. The system is then able to control appliances that use IR based control protocols. Set up is done by first connecting the MC to the Wi-Fi access point (AP) at home, and then using a smartphone to connect to the same AP, in order to control the system. The system is effective in controlling IR dependent appliances, but it relies on IR control code libraries that have been pre-installed on the MC, and manually entering the codes if needed. This causes an inconvenience as the user needs to access the MC physically in order to update IR protocols. A solution to this drawback, is implemented in the form of a database by the ARRC. This allows the user to directly update any necessary device control codes without physically accessing the MC. Moreover, the system isn’t secure, since it allows anyone connected to the Wi-Fi AP at home to access the MC and control devices. The ARRC overcomes this obstacle by limiting access to the AM. A user ID and password which is printed on the AM itself, requires any new user to physically access the AM in order to control it. Furthermore, in order for commands to be communicated back and forth between the mobile phone and the MC, HTTP is used.
A request–reply protocol such as HTTP tends to have higher delays, which aren’t suitable for such applications. The ARRC overcomes this hindrance by using a pub–sub protocol known as Message Queue Telemetry Transport Protocol (MQTT) to transmit commands from the smartphone to the AM. The use of this protocol allows for less delay and quicker response times.

In [22], the authors implement a system that operates by incorporating data collected by sensors into a HAS, and utilizes an IR transceiver to learn IR control commands of the appliances, and obtaining codes from a pre-installed library. Control of the system operates on two modes, the first one being a web based remote control layout on a smartphone, and the second one uses the sensor data in order to control appliances. If the motion sensors detect a user entering their home, it can switch on the necessary appliances such as air conditioning, and keep it on while the user is present. This can be expanded further by using sensors to check if a person is present in a specific room and turn devices on and off accordingly. By allowing sensors to communicate with each other and operate devices accordingly, the system introduces improvements in terms of energy saving. Old appliances with no sensory apparatus and network features, need not be replaced, and can be a part of the HAS. However, the system requires IR control protocols to be manually pre-installed and updated, requiring physical access to the MC. Moreover, it needs a cumbersome hardware design, that requires a separate device to interface with the sensors, and a MC to facilitate the IR transmitter and receiver. These issues can be resolved by implementing a single device solution that can be placed in the operating environment and be controlled via a smartphone, a solution offered by the ARRC system. Moreover, the system uses a very simplistic and rudimentary user interface, in the form of a webpage. This can be limiting because the more devices the user wants to access, the more menus they have to scroll through. The ARRC overcomes this, by using a point and control approach. The user simply points the smartphone camera at an appliance they wish to control and a command menu is generated on the screen in real time.

In [23], the proposed system is able to control appliances that are connected to the relay attached to a MC. This is accomplished by interfacing a bluetooth module with a master MC that send a signal to the relay connected MC. Furthermore, a number of buttons are connected to the master MC to control different appliances. A speaker is also connected to give an audio feedback to the user after a command is sent. The system stores all the commands on a SD card attached to the MC. Although the system is extremely cost effective but the its functionalities are very limited with ON/Off commands only. The system does not add networking functionality to the appliances and needs to be in the user’s hands in order to be used. ARRC system solves these issues by controlling the appliances using a SA connected to the internet and stores all the commands in an online database.

In [24], the authors propose a system that utilizes a pub–sub interaction model. By programming Wi-Fi enabled MCs to communicate via a pub–sub protocol, the system is able to establish remote monitoring and control over devices. However, for the purpose of collecting and forwarding information to the MCs, a fully sized computer is used. This can be considered a lack of efficiency in terms of system design, due to wastage of computing resources. These flaws can be countered by delegating the computation tasks required for the exchange of information, to a cloud-based server or even a single board computer. The information exchange of the ARRC system is handled by a cloud-based server.

In [25], the presented system uses a single board computer, to act as an exchange point, in order to implement a pub–sub interaction model. The single board computer also acts as a control platform to command Wi-Fi enabled MCs. The MCs are connected to sensors and actuators. The system is capable of establishing a wireless network of nodes. The highlight
of the system is its ability to implement the pub–sub model for HAS over a local area network in a cost-effective manner. Although the HAS is capable of achieving a pub–sub model locally, there is no discussion as to whether or not the latency differs when compared to a cloud-based server, or in comparison with other pub–sub based protocols. The tests conducted for the ARRC system explore the possible implementations of a pub–sub model in order to evaluate the limitations of the system.

A different yet innovative approach by [26], combines AR, with IoT based home automation. The system consists of a database implemented on a cloud-based server, and a single board computer which facilitates device to device communication. A SA is responsible for image detection, tracking and generating graphics. Furthermore, MCs with relay switches are used for switching appliances. A database is used to hold switch state values of appliances, which could be changed by the user. Virtual buttons are generated on the smartphone screen for user interaction, and a change is triggered in the database. The changes in the databases are communicated to the MCs. By using virtual buttons over appliances in a real-world environment, they are able to switch appliances on and off, which is the only extent of control offered by the system. Moreover, the software application used to generate the virtual button, is heavily dependent on the target image of the appliances. If the target image was not visible clearly to the smartphone, the virtual buttons would not be generated, making the system unusable. A summary of all the HAS discussed in this section is illustrated in Table 1.

Unlike the aforementioned system which uses vision-based AR, the ARRC system makes use of marker-based AR with the help of a proprietary marker. This eliminates the need for having the object be in perfect lighting conditions for an overlay to be generated, and thus reduces time taken for a user to control the necessary devices. Moreover, instead of using multiple MCs in order to control appliances, the ARRC systems uses a single device equipped with an IR transmitter in order to control a variety of appliances. Using an IR transmitter not only reduces the required components, but also increases the amount of control that a user has over an appliance. The need for a single board computer is replaced by a cloud-based server interacting directly with a device based on a pub–sub interaction model. The above-mentioned systems do not contribute to the optimization of the user application in their architecture, and rely heavily on the MC or microprocessor on the device or actuator end. They also do not make effective use of the platforms used in their systems, so their architecture does not guarantee the full achievable throughput.

4 Publish and Subscribe Protocols

With the massive amount of information transaction occurring in IoT-based systems, a significant challenge faced by designers is the choice of an appropriate communication protocol. This decision is made by factoring in specifics such as, the type of available wireless communication channel, the limitations of the devices on the network, and the type of data exchanged [27]. When running on devices with computing capabilities, the machine to machine (M2M) protocol or application layer protocol, can differ based on their interaction models. The common interaction models are request–reply and pub–sub. The request–reply interaction model is one of the most commonly used interaction models in communications. A client may request information from a server, which processes it upon receipt of the request message, and returns a response message. This sort of knowledge is centrally shared and regulated. The most popular protocols which are based on the request–reply
model are REST HTTP and CoAP. Figure 7 displays the Request–Reply interaction model for both HTTP and CoAP.

An alternative to the traditional request–reply model is the pub–sub interaction model. An example of this interaction model is made up of three members: broker, publisher, and subscriber. The client acting as a subscriber does not need to request information from the server. Instead of sending a request, the client only needs to subscribe to specific events (topics), that are interested in receiving messages from the client. The client subscribes to the central point in the architecture, which is the broker. The broker is responsible for filtering all incoming messages and forwarding them to the clients and publishers based on their subscription [28]. The publisher is the third member involved, and is responsible for providing information. When an event pertaining to a specific topic takes place, publisher submits the event to the broker who forwards the data on the requested topic to the subscriber. Figure 8 displays the pub–sub interaction model for both MQTT, Data Distribution Service (DDS) and Advanced Message Queueing Protocol (AMQP).

This interaction model can be fruitful for IoT applications, that require different devices of varying computational ability, to communicate with each other. Multiple protocol options using the pub–sub interaction models can be considered for the ARRC system, discussed in the following sub-sections.

4.1 Constrained Application Protocol (CoAP)

Constrained Application Protocol (CoAP) is designed to be used in constrained devices with limited processing capabilities, the CoAP was designed by the Constrained RESTful Environments, a working group of Internet Engineering Task Force (IETF) [29]. Its distinguishing feature is the implementation of the request–response interaction in constrained environments. Although originally intended to be used for a request–reply model, CoAP has an optional feature that allows clients to receive changes from the requested resource on the server. This enables clients to receive notifications, when a state in the resource undergoes a change. This interaction mimics the behavior of a pub–sub model, and allows the protocol to be an option for the implementation of the ARRC system. CoAP runs over the User Datagram Protocol, which is less reliable in comparison to Transmission Control Protocol. It also does not receive responses to messages in a synchronous manner, another factor in its unreliability. In a situation where commands for multiple appliances needs to be processed simultaneously, and in the order of transmission, CoAP is not the ideal protocol for a HAS design.

4.2 Data Distribution Service

DDS is described as a real-time data-centric interoperability standard by the Object Management Group (OMG) [30], DDS operates on the pub–sub interaction model. Based on decentralized and peer-to-peer communication, DDS is a unique pub–sub protocol which does not depend on a broker component. This allows publishers and subscribers to communicate with each other as peers, resulting in an asynchronous exchange of data. This can be beneficial, since the absence of a broker limits the probability of system failure. It possesses a built-in discovery protocol, which allows for subscribers to discover which publishers are present, the type of information they wish to acquire, and the desired quality of service in exchange [31]. However, for a system like the ARRC, DDS is not the ideal protocol. Primarily because, the decentralized exchange model does not fit the system
design. Although the absence of a broker can be tolerated, the ARRC system is designed to facilitate communication through a broker.

### 4.3 Advanced Message Queueing Protocol

OASIS defines AMQP as an open standard protocol [32] that follows the pub–sub interaction model, and is designed to allow interoperability between an extensive variety of multiple systems and applications. This interoperability is independent of the internal design of the system. Due to this feature, platforms that can be programmed using different languages can communicate with each other [33]. Previous versions of AMQP, consists of a broker with two main entities. While one entity is responsible for publishing messages, the other performs routing of messages into more appropriate queues. If there is more than one subscriber for a message, the broker duplicates the message and forwards the copy. A message remains in the queue, until it is received by a subscriber. The more recent versions are capable of following a peer-to-peer model, without the need for a broker. This support for varying topologies [34], makes AMQP a suitable alternative. However, it requires a lot of processing capabilities from the platform it runs on, along with memory requirements. A heavy protocol is not suitable for the design of the ARRC, as it relies on a device with memory constraints.

### 4.4 Extensible Messaging and Presence Protocol (XMPP)

XMPP was originally intended to be used for instant messaging and message exchanges between applications. XMPP is a text-based protocol relying on Extensible Markup Language (XML) [35]. It is capable of both client–server and pub–sub interactions [36], while operating using transmission control protocol (TCP). In IoT environments, it is designed to allow clients to exchange messages in real time. XMPP uses XML streams to exchange data between clients and servers in the form of XML stanzas (semantic structured data unit). The stanzas are of three types: `<presence/>`, `<message/>` and `<iq/>`(info/query), with all three having distinct roles. The presence stanza is responsible for subscription among clients, allowing for notification of client presence to the entities. The message stanza is used to send data between entities. The iq stanza, pairs the senders of a message, to its appropriate recipients. A disadvantage of using XMPP, however, is in networks with bandwidth constraints, due to its size. Moreover, the absence of reliable QoS guarantees and inefficient binary encoding are deemed impractical for use over constrained wireless networks used in IoT environments. These factors make it unsuitable for the implementation of the ARRC system.

### 4.5 Message Queue Telemetry Transport Protocol

Released by IBM, and later adopted for IoT applications by the OASIS [37], MQTT is a lightweight messaging protocol which follows the pub–sub interaction model. Its lightweight feature makes it suitable for devices with memory and power constraints, and for low bandwidth networks with high latency levels. MQTT is constructed from multiple components which consist of a server, clients, and data. In the context of MQTT, the server is usually referred to as the broker. MQTT is a protocol for persistent asynchronous message queuing. Persistent indicates that the messages transmitted by the sender
are stored in the queue until the receiver retrieves them [38]. Clients can either subscribe to or publish data through the broker, and the duty of the broker is to perform intermediary functions, such as exchanging data between clients. Information is routed through “Topics”, with a name and multiple levels being associated to each topic, the character “/” as separators in a topic tree, and other special characters such as # and + to match multiple levels in a topic [39]. Another feature of MQTT, is that the server buffers all the messages if the client is offline and delivers them when the session is enabled. A disadvantage, however, is that MQTT does not offer encryption, which is considered a problem from a security perspective. If the user wants to incorporate some level of encryption, it simply adds overhead. Brokers offer authentication techniques, requiring clients to provide a username and password before validating a connection. All these features make it a feasible solution for the implementation of the ARRC system.

5 Implemented Design

A hardware prototype for the AM is implemented in order to test and analyze the performance capabilities of the system. The hardware prototype is referred to as the ARRC Appliance Unit. A server is chosen to act as a broker for information exchange. The server will be based off a pub–sub model, using MQTT protocol. Certain components of the design will be using cloud based services, due to ease of implementation.

5.1 Software Application

The software application is implemented on an android smartphone. The application uses AR, in order to generate a control overlay, when it recognizes a distinguishable marker. By using a generated overlay only over the appliances to be controlled, the user does not have to go through the arduous process of scrolling through multiple pages, in order to control a specific device. The only instruction needed is to point at the device, and press the command needed using the smartphone touch screen. A description of the application design is given in the following section.

5.1.1 Vuforia

Vuforia is a Software Development Kit (SDK) that integrates and enables applications, with AR capabilities. Vuforia supports native development for both iOS and Android. It enables the development of AR applications in Unity. Therefore, AR applications developed using Vuforia are compatible with a broad range of mobile devices. Marker-based and marker-less recognition are supported by Vuforia. Using computer vision, it recognizes markers, tracks their position in the world, and displays AR elements accordingly.

Vuforia also provides a guide [40] which highlights the design elements required to create markers as shown in the Fig. 9. The recognition is done from a user-defined database. The database can be local or hosted on Vuforia’s cloud-based server. The behavior of the system is coded using C# and Unity IDE.
5.1.2 Mark My ARRC

Markers are used to identify each appliances, and display the corresponding control schemes. Based on Vuforia’s AR marker VuMark, a unique design is created for the ARRC system. The mark shown in Fig. 10 is only identifiable by mobile application.

Certain factors, such as maintaining contrast between elements, to create distinguishable features, are considered for the mark design. The features are highlighted in Fig. 11. These marks can be placed on the ARRC Appliance Unit or can be printed as stickers and placed on the appliance directly. The control panel of the appliance is shown when the mark is scanned. As seen in Fig. 12, when the app scans a mark designed for controlling a television set, the appropriate controls are displayed on the screen.

5.2 Server

For the exchange of information between the application and the AM, a server is implemented. The protocol utilized by the server for the exchange of information is MQTT. The broker for this system is called MQTT Dashboard [41], which is an online broker based on HiveMQ MQTT broker. In the ARRC system, both the AM and user application will serve as clients. They are connected to the internet via a Wi-Fi connection, to access the broker. All the data exchange between the two, is handled by the MQTT broker.

5.3 Appliance Module

A Wi-Fi enabled MC is used in the AM, to carry out computational tasks such as communicating with the smartphone application, decoding command signals, and transmitting command signals, in the form of IR signals. Figure 13 shows a schematic of the AM implemented.

An IR LED is used to send commands as IR signals from the ARRC AM. The format of the IR signal and the protocols for the different appliances are all handled by the software of the system. A 940 nm IR LED with a forward voltage of 1.2 V and a current consumption of 100 mA is used. Due to the current output limitations of the general-purpose input/output (GPIO) pins on the MC, a transistor is used to control the IR LED. Moreover, three different LEDs are connected to the digital pins of the MC which work as battery level indicator.

5.3.1 Module Setup

The MC operates in ‘Dual Mode’ which allows it to function as an AP, and to connect to a separate internet AP. This allows the user to configure and connect the module to a nearby AP; creating a TCP connection for data transfer. The configuration is done via a web page upon connection to the module. All the information regarding APs is stored on the EEPROM of the MC, in case the module is turned off. The setup menu is shown in Fig. 14.

5.3.2 Battery Check

This function allows the user to track the remaining battery level of the AM. The Analog GPIO pin of the MC is used to measure the remaining voltage of the battery bank. The measured voltage value is then converted into percentage using the following formula:
The value ‘1024’ is used, as it is the highest value readable by the MC. This power value is sent to the user application, and is used to indicate the power level on three LEDs, on the exterior of the AM, as shown in Fig. 15.

### 5.3.3 IR Command Decoding

This function is for processing the commands from the application, and transmitting them in the form of IR commands, to the appliances. Figure 16 displays a flowchart of the decoding process, which consists of the following steps:

1. **Fetch**—A message is received by the AM through the MQTT broker, as “Bytes”.
2. **Decode**—The messages are then converted into characters and are assembled as “PXFFFFx99”, “P” being the protocol of the IR Command to be sent (P is replaced with a number between 1 to 5, where each number represents a protocol), “X” and “x” functions as separators to distinguish different parts of the message, “FFFF” is the actual IR command in hexadecimal, and “99” represents the number of bits for the IR command.
3. **Send**—After the signal is decoded, it is then compared with IR protocols pre-installed on the MC, and then transmitted to the appliances.

### 5.4 Command Database

A Realtime NoSQL cloud-hosted database is used, in order to store commands for the appliances. It can synchronize data across all clients in real-time. The data is stored as JSON files. Figure 17 shows the online database of commands for various appliances.

### 5.5 Microcontroller Over Microprocessor

The MC was chosen over a microprocessor as this system cannot utilize the full processing power of the microprocessor, while the MC has the sufficient power to do the required tasks. Moreover, utilizing the on-board storage of the microprocessor to contain the database would have eliminated the need for an online database, but this will create a major limitation to the ARRC AMs. If a user has multiple ARRC AMs, each will store the IR commands for specific appliances and can no longer interact with other appliances. The ARRC AMs would need to coordinate and share the IR codes together in order to be “universal”. Hence, having a unified online database is the best option.

### 6 Discussion

Although the system works as seamlessly as expected, ensuring that it performs exceptionally within the given criteria is necessary. Moreover, testing under different circumstances and alternative features; allows us to discover the best possible configuration for the system. Various tests are conducted, in order to measure parameters of the systems, such as bandwidth, packet size, and delay. The exchange among the AM, the mobile application,
and the broker is facilitated by an AP. A laptop is configured to act as an AP, to capture packets and display them in an unencrypted form.

A total of 5000 commands are sent from the software application to the AM. Table 2 displays the data obtained from the initial test. The average bit-rate is 2394 bits per second, while communicating with a cloud based MQTT broker. The low bandwidth requirement indicates that the system can be implemented in households with weak networking capabilities. A total of 28,657 packets were exchanged which includes 118 re-transmitted packets. The low percentage of packets re-transmitted (0.41%), were all acknowledged by the broker. This allowed all 5000 commands to be communicated. The application is modified to send these commands automatically for the test. For each command, an acknowledgement (ACK) is sent back to the application, the battery level information is communicated from the unit to the broker, and then to the mobile application. There are no ACK packets between the unit and the broker. For every command sent to the broker, an ACK is sent back to the SA. The average time delay for every 10th command sent is shown in Fig. 18.

Next, 1000 commands are sent from two separate mobile phones, using the software application, to the ARRC Appliance Unit. The commands sent are for the same operation on the same appliance. The response time for turning on the same appliance, using two different mobile phones is recorded in Fig. 19. Both mobile phones sent the commands to the appliance at the same time. It is noticeable that, when the same operational command for the same appliance is being sent to the AM simultaneously from two mobiles, the time taken for the broker to respond, is almost the same for each smartphone. However, it should be noted that the broker being utilized is cloud-based, so the response time may vary depending on the bandwidth provided by the ISP (Table 3).

In another scenario, 1000 commands using two separate mobile phones using the software application, were sent to the AM. However, this time, the commands were for two different appliances, a TV and an air conditioning unit.

The response time for turning on two different appliances, using two different mobile phones is recorded. Both mobile phones sent the commands at the same time. As shown in Fig. 20 to examine the broker response to two different commands, no significant change from the previous observation. Spikes in the delay occur, simultaneously for both mobile phones. This is because the broker does not give preference to a device when processing requests. Due to this reason, if a substantial delay does occur, due to ISP restrictions or limitations on the broker; both devices experience the same delay.

As a summary, the average value for response delays using one mobile and two mobiles are 137.54 and 157.48 ms respectively. The difference between the highest delay value and the lowest delay value is 7.71 ms (approximately 5%). In both scenarios the delays are well under a fifth of a second. However, there were instances where only one of the devices experienced a substantial delay. Since the broker being utilized is cloud based and free to use, it is also processing requests by the other clients. An overload of requests by multiple devices will lead to substantial delays.

The aforementioned tests were all conducted on an online broker for MQTT. In order to further investigate the responsiveness offered by MQTT, we implemented a broker on a local network by using a Raspberry Pi. The Raspberry Pi would function as the broker, and forward commands and messages between the AM and the smartphone. A thousand commands were sent from the mobile phone to the AM. The time taken for an acknowledgement to be sent back to the phone is calculated. The average time delay for every 2nd command sent is shown in Fig. 21.

The average value for the delay by the broker on a local network was 0.26 ms with the highest value being 2.54 ms. When compared to the response time taken by an online
### Table 2: Summary of HAS from related works

| System Name                          | Functionality/Features | Wireless Interface | Controller | Online Database | Smartphone Application |
|--------------------------------------|------------------------|--------------------|------------|----------------|------------------------|
| Davidovic and Labus [18]             |                        |                    | Raspberry Pi |                |                        |
| Zhang, et al. [19]                   | GPRS/Zigbee            | WiFi               | NodeMCU    |                |                        |
| Singh, et al. [20]                   | WiFi                   | WiFi               | NodeMCU    |                |                        |
| Ramani, et al. [21]                  | WiFi                   | WiFi               | Arduino    |                |                        |
| Lin, et al. [22]                     | WiFi                   | WiFi               | NodeMCU    |                |                        |
| Husain, et al. [23]                  | Bluetooth              | Arduino            | NodeMCU    |                |                        |
| Kodali and Sarathakal [24]           | WiFi                   | WiFi               | NodeMCU    |                |                        |
| Blaft and Patoliya [25]              | WiFi                   | WiFi               | NodeMCU    |                |                        |
| Khamia and M [26]                    | WiFi                   | WiFi               | NodeMCU    |                |                        |
| ARRC                                 |                        |                    |            |                |                        |
broker, the average was 137.54 ms. The broker operating on a local network is 528X quicker than an online broker. A graphical representation of the delay comparison is shown in Fig. 22.

In addition to having lower latency, using a local broker for MQTT implementation also offers reliability. In the case of an online broker, the user has almost no control over the server that is hosting the broker. In the event of the host shutting down, the entire HAS would be inoperable. This problem would occur if the online broker was free to use, or a paid service. Setting up a local broker, would require a one time cost for the additional

Fig. 7 Request–reply model for HTTP and CoAP

Fig. 8 Publish–subscribe model for MQTT, DDS and AMQP
hardware. However, the cost can be overlooked when considering the lower latency and consistent reliability provided by a local broker.

Furthermore, in order to study the difference in response time between two protocols which use a pub–sub interaction model, another test was conducted. CoAP was used instead of MQTT, and instead of a broker, the phone communicates directly with the AM. A 1000 commands were sent to the AM and the time taken for an ACK to be sent back to
the phone is calculated. The average time delay for every 2nd command sent is shown in the Fig. 23.

Figure 24 displays a graphical representation of the comparison between CoAP and MQTT performance on a local network. The average time taken for the AM to respond with an ACK upon receiving a command, is 523 ms. When compared to the average
value by an online MQTT broker (137.54 ms), it is nearly 4× slower than MQTT. When compared to the highest response time value by MQTT on a local broker (2.54 ms), CoAP is nearly 20× slower. The difference in delay is primarily due to the difference in interaction models between the two protocols. CoAP utilizes a request–reply model, which requires that the client send a request to the server every time information needs to be exchanged. In contrast, MQTT makes use of a pub–sub interaction model. This allows the broker to maintain a constant connection with all clients and transmit updated information without the need for continuous requests for every exchange. The aforementioned feature enables the exchange between the client and the server to be quicker when sending information, in comparison to CoAP. Although this reduced delay is negligible when considering the end-user, it has a substantial significance on the system. If the system were to scale up and add more appliances and users, the reduced delay offered by MQTT would be a necessary advantage.
Conclusion

The implementation of IoT based HAS is an ever evolving area of study. Multiple methods, using different platforms while operating on varying parameters are constantly being implemented. These various forms of implementations yield their own respective results.
Fig. 19  Average time delay for every 2nd command for 1000 commands from 2 mobiles

Table 3  Data obtained from packet exchange

|                         |         |
|-------------------------|---------|
| Packets                 | 28,657  |
| Time span (seconds)     | 7327.652|
| Average packet size (Bytes) | 77 |
| Bytes                   | 2193248 |
| Average bytes/s         | 299     |
| Average bits/s          | 2394    |

Fig. 20  Average time delay for every 2nd command for 1000 commands from 2 smartphones for 2 different appliances
In this paper, the design and implementation of an IoT based HAS which uses Marker Based AR is proposed. The system uses IR based control signal in order to control home appliances, and pub–sub based protocols for M2M communication. The system is implemented and tested in real world scenarios. The AR marker is successfully identified and a control overlay is generated. The commands are sent to the broker and successfully forwarded to the appliances. The sensitivity of the broker is tested by sending a high volume of commands, in varying scenarios and measuring the time taken for the broker to respond. The decision of using MQTT is justified by the results of the tests conducted, which showcase quick response time in both online and local networks. The measured time delay for

---

**Fig. 21** Average time delay for every 2nd command for 1000 commands on a local network using MQTT

**Fig. 22** Average time delay for every 2nd command for 1000 commands on a local network (left vertical axis) and an online broker (right vertical axis)
an online broker is acceptable while on a local broker is far quicker in comparison. These tests assist in encouraging the use of MQTT protocol to establish M2M communication for a HAS.

**Acknowledgements** This research is supported in part by the Distributed and Networked Systems Research Group, University of Sharjah, Operating Grant No. 150410.

**Author Contributions** All authors contributed to the study conception and design. All authors read and approved the final manuscript.
Funding Information  This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Declarations

Conflict of interest  The authors declare that there are no conflicts of interest regarding the publication of this paper. Availability of code, data and material Available upon request.

References

1. Patel, M., Shangkuan, J., & Thomas, C. (2017). What’s new with the internet of things? (2017). https://www.mckinsey.com/industries/semiconductors/our-insights/whats-new-with-the-internet-of-things.
2. Ashton, K., et al. (2009). That ‘internet of things’ thing. RFID Journal, 22(7), 97–114.
3. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 29(7), 1645–1660.
4. Stojkoska, B. L. R., & Trivodaliev, K. V. (2017). A review of internet of things for smart home: Challenges and solutions. Journal of Cleaner Production, 140, 1454–1464.
5. Marti, L. M., Mayor, R. W., Mathias, A. G., & Huang, R. K. (2019). Modification of automated environment behavior based on user routine, US Patent 10,496,050 (Dec 3).
6. Putra, L., Kanigoro, B., et al. (2015). Design and implementation of web based home electrical appliance monitoring, diagnosing, and controlling system. Procedia Computer Science, 59, 34–44.
7. Ferrara, P., Mandal, A. K., Cortesi, A., & Spoto, F. Cross-programming language taint analysis for the IoT ecosystem, Electronic Communications of the EASST 77.
8. Balta-Ozkan, N., Davidson, R., Bicket, M., & Whitmarsh, L. (2013). The development of smart homes market in the UK. Energy, 60, 361–372.
9. Zhou, Z., Zhang, W., Li, S., & Yu, N. (2019). Potential risk of IoT device supporting IR remote control. Computer & Control Engineering Journal, 10(2), 19–32.
10. Khedr, A. M., Aziz, A., & El-Sawy, A. A. (2018). Cluster-tree routing based entropy scheme for data gathering in wireless sensor networks. IEEE Access, 6, 77372–77387.
11. Di Serio, Á., Ibáñez, M. B., & Kloos, C D. (2013). Impact of an augmented reality system on students’ motivation for a visual art course. Computers & Education, 68, 586–596.
12. Jung, J., Ha, J., Lee, S-W., Rojas, F. A., & Yang, H. S. (2012). Efficient mobile AR technology using scalable recognition and tracking based on server–client model. Computers & Graphics, 36(3), 131–139.
13. Osamy, W., Khedr, A. M., & El-Sawy, A. A. (2018). Cluster-tree routing based entropy scheme for data gathering in wireless sensor networks. IEEE Access, 6, 77372–77387.
14. Khedr, A. M. (2008). Learning k-nearest neighbors classifier from distributed data. Computing and Informatics, 27(3), 355–376.
15. Khedr, A. M., & Salim, A. (2008). Decomposable algorithms for nearest neighbor computing. Journal of Parallel and Distributed Computing, 68(7), 902–912.
16. Hanif, S., Khedr, A. M., Al Aghbari, Z., & Agrawal, D. P. (2018). Opportunity to exploit the internet of things for wireless sensor network routing in smart cities. Journal of Sensor and Actuator Networks, 7(4), 46.
17. Omar, D. M., Khedr, A. M., & Agrawal, D. P. (2017). Optimized clustering protocol for balancing energy in wireless sensor networks. International Journal of Communication Networks and Information Security, 9(3), 367–375.
18. Davidovic, B., & Labus, A. (2015). A smart home system based on sensor technology. Facta Universitatis, Series: Electronics and Electrical Engineering, 29(3), 451–460.
19. Zhang, W., Li, G., & Gao, W. (2015). The embedded smart home control system based on GPRS and Zigbee. In MATEC Web of Conferences, Vol. 34, EDP Sciences, p. 04010.
20. Singh, H., Verma, S., Pal, S., & Pandey, K. (2019). A step towards home automation using IoT. In 2019 Twelfth International Conference on Contemporary Computing (IC3). IEEE Xplore. https://doi.org/10.1109/IC3.2019.8844945.
21. Ramesh, N., Kumar, S. T., Vamsi, V., Akarsh, S. Wi-fi controlled universal remote using esp8266.
22. Lin, Y.-W., Lin, Y.-B., Hsiao, C.-Y., & Wang, Y.-Y. (2017). IoTtalk-RC: Sensors as universal remote control for aftermarket home appliances. IEEE Internet of Things Journal, 4(4), 1104–1112.
23. Husain, M. I., Alam, M., Rashed, M. G., Haque, M. E., Rashidul Hasan, M. A., & Das, D. (2019). Bluetooth network based remote controlled home automation system. In 2019 1st International conference on advances in science, engineering and robotics technology (ICASERT), pp. 1–6. https://doi.org/10.1109/ICASERT.2019.8934500.
24. Kodali, R., & Soratkal, S. (2016). MQTT based home automation system using esp8266, pp. 1–5. https://doi.org/10.1109/R10-HTC.2016.7906845.
25. Bhatt, A., & Patoliya, J. (2016). Cost effective digitization of home appliances for home automation with low-power wifi devices. In 2016 2nd International conference on advances in electronics, information, communication and bio-informatics (AEEICB), pp. 643–648. https://doi.org/10.1109/AEEICB.2016.7538368.
26. Khanna, R., & Vanitha, M. (2019). Augmented reality based IoT controller. In 2019 International conference on vision towards emerging trends in communication and networking (ViTECoN), IEEE, pp. 1–5.
27. Dizdarević, J., Carpio, F., Jukan, A., & Masip-Bruin, X. (2019). A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. ACM Computing Surveys (CSUR), 51(6), 1–29.
28. Banavar, G., Chandra, T., Mukherjee, B., Nagaraj Rao, J., Strom, R. E., & Sturman, D. C. (1999). An efficient multicast protocol for content-based publish-subscribe systems. In Proceedings 19th IEEE international conference on distributed computing systems (Cat. No. 99CB37003), IEEE, pp. 262–272.
29. Shelby, Z., Hartke, K., & Bormann, C. (2014). The Constrained Application Protocol (CoAP). Internet engineering task force (IETF) RFC-7252. https://doi.org/10.17487/RFC7252.
30. About the data distribution service specification version 1.4 (Mar 2015). http://www.omg.org/spec/DDS/1.4
31. Corsaro, A. The data distribution service tutorial, Technical Report 4.0.
32. Oasis advanced message queuing protocol (AMQP) version 1.0. http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf.
33. Fernandes, J. L., Lopes, I. C., Rodrigues, J. J., & Ullah, S. (2013). Performance evaluation of restful web services and AMQP protocol. In 2013 Fifth international conference on ubiquitous and future networks (ICUFN), IEEE, pp. 810–815.
34. van der Linden, E., Wallgren, J., Jonsson, P. A latency comparison of IoT protocols in MES.
35. Extensible messaging and presence protocol (XMPP). (2004). Core (Oct https://xmpp.org/rfcs/rfc3920. html.
36. Hornsby, A., & Walsh, R. (2010). From instant messaging to cloud computing, an XMPP review. In IEEE international symposium on consumer electronics (ISCE 2010), IEEE, pp. 1–6.
37. Banks, A., Briggs, E., Borgendale, K., Gupta, R. MQTT version 5.0, OASIS Standard.
38. Swarnamugi, M., & Chinnaiyan, R. (2019). Smart and reliable transportation system based on message queuing telemetry transport protocol. In International conference on intelligent computing and control systems (ICCS), 2019, 918–922. https://doi.org/10.1109/ICCS45141.2019.9065789
39. Nguyen, T. T. L., Nguyen, T. A., Vo, H. K., Luong, H. H., Nguyen, H. T. K., Dao, A. T., & Ha, X. S. (2021). Toward a unique IoT network via single sign-on protocol and message queue. In K. Saeed & J. Dvorsky (Eds.), Computer information systems and industrial management (pp. 270–284). Cham: Springer International Publishing.
40. Vumark design guide (2016). https://library.vuforia.com/articles/Training/VuMark-Design-Guide.
41. MQTT dashboard (2013). http://www.mqtt-dashboard.com/index.html.
42. Introducing JSON (2006). https://www.json.org/json-en.html.

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
Ali A. El-Moursy (Senior Member, IEEE) received the Ph.D. in the area of High-performance Computer Architecture from University of Rochester, Rochester, NY, USA, in 2005. Dr. El-Moursy has worked for Software Solution Group, Intel Corp., CA, USA till early 2007. In 2007, he has joint Electronics Research Institute, Giza, Egypt. His research interest is in high-performance computer architecture, multi-core multi-threaded micro-architecture, power-aware micro-architecture, simulation and modeling of architecture performance and power, workload profiling and characterization, cell programming, high performance computing, parallel computing and Cloud computing. Dr. El-Moursy has also participated with IBM Cairo Technology Development Center, Egypt, as a visitor research scientist for the period from Feb. 2007 till Jan. 2010. In Sep. 2010 Dr. El-Moursy has joint ECE Dep. at University of Sharjah, Sharjah, UAE as an Assistant Prof. In Jan. 2017, Dr. El-Moursy has been promoted to the Associate Prof. Rank.

Dr. Fadi N. Sibai joined Prince Mohammad Bin Fahd University in 2019 as Dean of the College of Computer Engineering and Science. He taught at several Universities in the USA and the Middle East. Between 2011–2019, he worked for Saudi Aramco. Between 2006–2011, Dr. Sibai directed the Computer Systems Design program and IBM Cell Competence Center at the UAE University where he received IBM’s highest research award. Between 1996–2006, he managed programs and teams in performance benchmarks and system validation at Intel Corporation, CA. Between 1990 and 1996, he was an assistant professor of Electrical Engineering at the University of Akron. He authored or co-authored over 200 publications and technical reports and served on the Organizing or Program Committees of over 20 International Conferences. He received the PhD and MS degrees from Texas A&M University, and the BS degree from the University of Texas at Austin, all in Electrical (Computer) Engineering. Dr. Sibai holds PMP, CISSP, CCNA, and CQRM certifications. He is a member of PMI, (ISC)², and Eta Kappa Nu.

Jahanzeb Rehman received the B.Sc. degree in electrical and electronics engineering from the University of Sharjah, UAE, in 2019. He is currently working as a Research assistant with the distributed and networked systems research group at the same institution. His research interests include Wireless Sensor Networks and Internet of Things (IOT) based solutions.
Omar M. Gouda received the Computer Engineering degree in 2019 from the University of Sharjah (UoS), UAE. Since 2020, he has joined the Department of Computing & Informatics as a research assistant while pursuing his master’s degree in computer engineering. He has been the leader in several research projects in several fields. His research interests are in the areas of Internet of things (IoT), IoT security, Wireless Sensor Networks (WSNs), deepfakes, image processing, machine learning and deep learning.

Abdelrahman T. Gaber received his Electrical and Electronics Engineering degree in 2019 from the University of Sharjah. He is currently developing his skills in the industrial field and is pursuing opportunities in the Embedded Systems market. Current research interests are Embedded Systems and Machine Learning.

Ahmed M. Khedr received his B.Sc degree in Mathematics in June 1989 and the M.Sc degree in the area of optimal control in July 1995, both from Zagazig University, Egypt. In July 1999, he received his M.Sc and in March 2003, he received his Ph.D. degrees, both in Computer Science and Engineering, from University of Cincinnati, Ohio, USA. From March 2003 to January 2004, he was a research assistant professor at ECECS Department University of Cincinnati, USA. From January 2004 to May 2009, he worked as an assistant professor at Zagazig University, Egypt. From September 2009 to September 2010 he worked as an associate professor at the Department of Computer Science, College of Computers and Information Systems, Taif University, KSA. Since December 2014, he is a Professor at Zagazig University, Egypt. From September 2010 till December 2019, he worked as an Associate Professor and since January 2020, is a Professor at the Department of Computer Science, College of Computing and Informatics, University of Sharjah, UAE. He was awarded the State Prize of distinction in advanced technology, Sharjah Islamic Bank prize of distinction in research and the University of Sharjah prize of distinction in research, in June 2009, May 2013 and April 2014, respectively. His research interests include Wireless Sensor Networks, Internet of Things, and Distributing Computing.