IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 April 2023, accepted 18 May 2023, date of publication 26 May 2023, date of current version 6 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3280414

RESEARCH ARTICLE

Discovering Hazards in IoT Architectures: A Safety Analysis Approach for Medical Use Cases

FRYAD KHALID M. RASHID[1,2], OSMAN SHARIF OSMAN[3,4], ETHAN TRAVIS MCGEE[5], AND HAIDER RAAD[6]

[1]Technical College of Engineering, Sulaimani Polytechnic University, Sulaymaniyah 46001, Iraq
[2]Department of Computer Engineering, Komar University of Science and Technology, Sulaymaniyah 46001, Iraq
[3]College of Administration and Economics, University of Sulaimani, Sulaymaniyah, Iraq
[4]Department of Computer Science, Komar University of Science and Technology, Sulaymaniyah, Iraq
[5]Department of Computer Science, Bob Jones University, Greenville, SC 29614, USA
[6]Department of Physics, Xavier University, Cincinnati, OH 45207, USA

Corresponding author: Fryad Khalid M. Rashid (fryad.rashid@komar.edu.iq)

ABSTRACT Internet of Things (IoT) systems are becoming increasingly safety-critical as the "Things" become an integral part of everyday life and are given control over life-sustaining processes. As such, these products will need safety-aware analysis during the software development life cycle to ensure they operate successfully without harming users. The overall objective of this study is to construct an approach for conducting safety analysis on the IoT systems in the design phase of the Software Development Life Cycle. The increasing complexity of the IoT raises concerns with respect to properly assuring IoT safety, since more interaction among components and tighter coupling may result in increased logical errors, posing new safety risks. To show the effect of these problems, we have analyzed several medical systems using our proposed methodology. In this study, we present a methodology to implement IoT systems which takes into account errors and potential hazards at design time. To increase the adoptability of our approach, we use standardized languages/model to represent errors. The suggested approach's viability is substantiated by analyzing diverse medical use cases, wherein different types of software faults have occurred. These faults have resulted in harm or potentially hazardous situations in medical IoT products. The results of our study show that tracing errors via our method leads to the discovery of hazards in IoT architectures without requiring specialized domain knowledge. The results also are validated based on the traceability criteria. By providing a new hazard analysis method based on early design knowledge and validating early in the Software Development Life Cycle, we discover more hazards and safety constraints to ensure the success of safety critical IoT systems.

INDEX TERMS IoT system architecture, medical IoT faults, IoT safety analysis, safety constraints validation.

I. INTRODUCTION

The Internet of Things (IoT) has expanded consistently over the past decade. It is not uncommon today to find homes which have internet connected sprinklers, lights, thermostats or garage door openers [11]. In addition to expansion into the home, IoT functionality has been increasingly used in safety-critical devices like pacemakers and connected inhalers [17]. Safety analysis, a critical component of safety-critical device development, has historically been relegated to devices which were already recognized to be large, long-lived and expensive - aircraft, medical devices, vehicles and stationary medical equipment. Most IoT devices do not fit this description.

The IoT brings new classes of devices that are small and inexpensive, but because of their proximity to humans, the devices have a greater possibility of inflicting harm. One common means of increasing system safety is to isolate the

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

system, which goes against the ethos of IoT. The ''Things'' in the IoT are networked together specifically to solve complex data analysis problems and share a common infrastructure. These complex interactions require correspondingly complex architectures that are expensive to verify.

It is anticipated by researchers and practitioners that there will be a significant increase in the number of internet-connected devices over the next few decades. However, the growing complexity of the Internet of Things (IoT) poses a concern for ensuring IoT safety. The rising interaction among components and tighter coupling increases the likelihood of logical failures, leading to new safety hazards.

Hence, the primary aim of this study is to offer a practical technique to examine safety issues in safety-critical IoT systems. Furthermore, many recent hazard analysis methods, which serve as a core methodology for safety analysis, do not use error ontologies. This amplifies the difficulty of using such methods to analyze systems that do not have specialized training, thereby necessitating the development of a new approach.

In this study, we illustrate our technique, demonstrate it and well defined, using a pacemaker example to build an IoT architecture for the device, to design a safety architecture for it and to identify safety constraints. All steps and analysis are performed in the Open Source Architectural Tool Environment (OSATE) [15] and the Architecture Analysis & Design Language (AADL) [14].

The aims of this study are to:

- investigate a practical approach in analyzing complex, internet-connected systems.
- build a relationship between interdisciplinary research domains, especially IoT and Safety Engineering.
- provide a new safety analysis method with simpler guidelines, well defined processes and a lack of specialized training.

Furthermore, our primary goals in the paper is to establish a safety analysis method for critical IoT systems to address the following challenges:

- How to discover faults which were introduced early in the design phase of the Software Development Life Cycle (SDLC)?
  Our study begins by analyzing faults present in a pacemaker possessing IoT functionality. Our analysis is primarily focused on the device's architecture, so any faults discovered were introduced prior to development, and are the result of poor design, not implementation. To this end, we have first performed a systematic search for safety violations that occurred in medical devices possessing IoT functionality. We use the results of this search to demonstrate that IoT devices do have the capability of inflicting harm on users.
- How emergent behavior (unplanned behavior) can affect the functionality of a medical IoT system?
  We then analyze the IoT-enabled pacemaker from a safety perspective. Due to the number of potential communication paths as well as the number of devices communicating, it is possible for emergent behavior (unplanned behavior) to occur due to components communicating in unforeseen ways. This emergent functionality can drastically affect the safety of IoT enabled devices. From a safety development perspective, such faults are usually introduced in the early blueprint of the software as a result of mistakes such as values beyond boundaries, inconsistency between requirements, or timing mismatches.
- How to identify missed safety constraints for existing medical IoT device?
  Our approach allows us to recognize missed safety requirements for existing IoT system examples based on error behavior and state analysis. It helps safety analysts to consider them for an IoT component in a feedback control loop design which may lead to inadequate control. Such analysis allows for the discovery of faults as well as their source.

Finally, the mentioned challenges have motivated us to make the following scientific contributions:

- We report on real multi-case study of medical IoT devices such as pacemakers, providing insights of what might have been missed in the specification and design of the device.
- We build an IoT architecture for the pacemaker, which explains the phenomena of bubbling up errors from component level to system level.
- We discuss our findings and present the capabilities of the proposed approach. The method is validated based on specific criteria and evaluated based on our claims.

Our primary contributions lead to address the following research questions (RQ)s about criticality of the things in the IoT systems:

- RQ1 - How can component failures be analyzed to demonstrate their affect other components during the safety analysis process?
- RQ2 - During hazard analysis, how can dysfunctional component behaviors be identified?

The paper is structured as follows: The related work is discussed in section II. Background information is discussed in section III. An overview of our method is given in section IV. The validation of the method is described in V. The evaluation of the method is shown in section VI. The capabilities of the method are discussed in section VII. Finally, conclusions and future works described in section VIII.

II. RELATED WORK

In this section, we present our previous works related to an architecture for a typical wearable device and we also describe other works related to the study. Also, we expand the related work discussion to cover the existing research efforts for devising safety-critical IoT systems by ensuring their safeness.

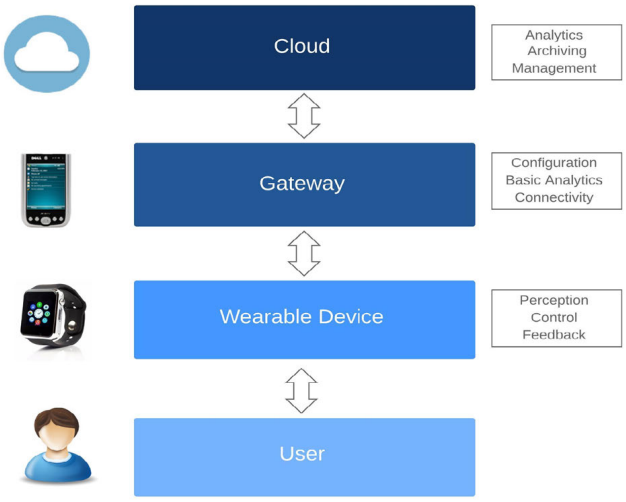


FIGURE 1. General wearable device architecture [33].

In [33], our prior works, the authors presented an architecture for a typical wearable device depicted in Figure1. While sophisticated wearable devices can be designed using standard IoT architectures described in the literature, the limited computational power and size constraints of wearables necessitate the use of a gateway, such as a smartphone, for configuration and processing of perceived data through a dedicated application. The relevant processed data is then translated into a graphical representation using the gateway. Basic processing involves wireless transmission of perceived data using a low-power transceiver, such as Bluetooth Low Energy (BLE). Cloud-based data processing provides more thorough and insightful analysis. The processed data is then transmitted back to the wearable device for feedback, while a copy is archived at the data center. Such communication requires network connectivity, which is provided by the gateway. It should be noted that wearables may be simpler or more complex than depicted in Figure 1. For instance, a wearable navigation system for professional hikers connects directly to a Global Positioning System, bypassing the gateway layer. Conversely, early fitness trackers relied solely on a smartphone for processing and feedback.

It is important to note that wearables can be simpler or more complex than what is shown in the Figure 1. A wearable navigation system used by professional hikers, for example, connects directly to a Global Positioning System, bypassing the gateway layer. In contrast, early fitness trackers relied only on a smart phone for processing and feedback [33].

In [7], similar to our study, the authors aim to heighten awareness regarding safety concerns in the realm of the IoT. This study agrees that IoT devices encounter challenges such as reliability and availability. For instance, in some cases, the failure of medical devices has posed a danger to users' lives. In our research, our primary concern is to raise awareness about the safety of medical IoT devices by minimizing the impact of faults. With the widespread integration of sensors, controllers, actuators, and physical objects that are enhanced with computing capabilities and communication technologies, the IoT is infiltrating every aspect of our daily lives. This further highlights the need for increased awareness and safety analysis in the IoT.

In [26], the classification of IoT hazards by researchers has resulted in two distinct types. The first type is directly linked to the IoT device, encompassing hazards such as overheating, shock, sonic hazards, and others. The introduction of IoT devices as new products onto the internet may lead to an increase in the frequency of such hazards. For instance, an electronic oven that can be remotely turned on and off has the potential to become hazardous if it receives an incorrect command. On the other hand, the second type of hazard is indirectly related to the IoT device and its operation, such as sensor failure due to severe weather conditions. Our methodology differs from others as we utilize a pre-existing set of generic error categories derived from analyses across multiple domains instead of creating our own categories.

In [25], the researchers were able to identify potential IoT risks (i.e., hazards for mission-critical applications) during the third stage of system development. This allows for the implementation of mitigating measures before accidents, losses, or frustrations can occur. Additionally, the article aims to evaluate the effectiveness of current hazard or risk analysis techniques in the upcoming era of the Internet of Things. To achieve this goal, the researchers consulted IoT experts to conduct an analysis of hazards, vulnerabilities, and risks in IoT. In contrast, our approach aims to reduce the dependence on experts and enable safety analysis to be conducted earlier in the software development life cycle. By adopting our approach, we seek to minimize the need for expert involvement while enabling a more efficient and effective approach to safety analysis.

In [27], the researchers have devised an IoT architectural analysis methodology aimed at identifying defects during the early stages of the design process. They have also created a design decision support system to facilitate security assessments across various scenarios. Furthermore, the researchers have identified the medical sector as one of the most susceptible fields to safety and security risks. Their investigations have led them to conclude that IoT component detection techniques are insufficient, as not every component may pose a risk during the early design phase. Thus, alternative approaches must be employed to ensure that IoT devices are safe and secure from the outset.

In [28], the researchers put forward an IoT risk analysis methodology that leverages the STPA (System Theoretic Process Analysis) standard to evaluate the safety and security concerns of medical IoT devices, such as insulin pumps. They demonstrated that while functional safety cannot prevent accidents caused by security threats, it is possible to detect them. Their findings revealed that the probability of accidents occurring can be reduced through countermeasures enacted by the decision maker, which our methodology can identify. Overall, the proposed IoT risk analysis methodology enables the identification of potential countermeasures that

can reduce the likelihood of accidents occurring, thereby improving the safety and security of medical IoT devices.

The doctoral dissertation [29] highlights the need for new safety analysis approaches to address the complexity of IoT and cyber-physical systems. Additionally, the dissertation emphasizes the importance of developing new resiliency methodologies to ensure operational continuity in the face of failures. The author proposes Artificial Intelligence (AI) as a potential solution for enhancing the safety and reliability of these systems.

In another doctoral dissertation [30] highlighted the importance of traceability as a pre-condition for valid safety requirements. The researcher identified three crucial elements for validating safety constraints: identifying hazards, analyzing hazards, and tracing software safety requirements. Our methodology offers a means to achieve this traceability, allowing for more comprehensive validation of safety requirements.

In [44], the study proposed a real-time hybrid vision and IoT-based system for monitoring the use of safety hooks at risky elevations in construction sites to prevent fall from height accidents. The system integrates vision and IoT sensors, a web-based management platform, and a backend cloud server storage system. It aims to automate multiple workers' safety monitoring in real-time, provide communication and visualization, and record workers' behavior to facilitate mitigative planning by safety engineers. We employ a unique methodology in which we develop an IoT framework for the pacemaker, a medical IoT device, with the aim of identifying any overlooked safety limitations during the initial design stage.

In [45], the author mentioned that while security concerns are the primary focus of the Internet of Things (IoT), device safety is also an important concern that is often overlooked. The IoT devices are composed of systems of systems, which can lead to security breaches and device misbehavior. Ensuring device safety is important to prevent harm to the environment in which the device operates, such as a wearable medical device causing harm to a patient or a vehicle causing an accident due to software malfunction. Examples of IoT devices that can cause harm include wearable medical devices, vehicles, thermostats, and other smart home devices. While our methodology enables us to identify and address potential hazards, it does not specifically address any security concerns in our proposed approach.

In [46], authors in the article discuss the risks associated with coal mining due to hazardous working conditions, such as high temperatures, humidity, and emissions. To address these challenges, the authors propose an IoT-based system called IoT-DSICS that uses sensor networks and control systems to monitor and control the environment in underground coal mines. The system is designed to improve safety and reduce accidents. The feasibility study shows that the system can be successfully implemented using existing communication and tracking infrastructure. Our methodology stands apart because our IoT framework is designed to track errors and understand how they propagate from one component to another.

III. FOUNDATION

In this section, we provide the necessary background concerning IoT in healthcare domains and system safety domains to understand the concept of our work.

A. IoT HEALTHCARE DOMAIN

Much research over the last 10 years has focused on creating technological advancements in the medical industry. Specifically, the Internet of Things (IoT) has showed potential for connecting a variety of medical devices, sensors, and healthcare specialists in order to provide high-quality medical services even to remote areas. This progression in healthcare IoT technology has lead to improved treatment, reduced costs, and enhanced accessibility services [31].

Designing wearable devices is not enough; a complete ecosystem is required. Sensors in a body area network seamlessly synchronize data with cloud services via the IoT infrastructure. Figure 2 depicts the architectural features commonly required in healthcare IoT systems (Health-IoT). Three essential core components make up the architecture: 1) body area sensor network 2) Internet-connected gateways 3) big data support or cloud environments. Through this platform, several applications deliver services to various stakeholders in the system. Caregivers, family members, and authorized parties can access data collected by sensors attached to users at any time from any place, allowing them to check the user's vital signs [32].

1) PACEMAKER

In this section, we provide background on a modern IoT pacemaker. A pacemaker is a medical device that is implanted beneath the skin and attached to the heart to assist in the regulation of irregular cardiac rhythms. Essentially, pacemaker systems consist of three important components which are the Pulse Generator (PG), Device Controller-Monitor (DCM) and Leads. The specification of each component and how it works have been described in detail in [16].

Pacemakers have reduced in size and longevity since their inception, and current technical improvements have permitted additional digital capabilities, particularly when it comes to transmitting data from the patient's body to access points such as a Bluetooth-equipped smartphone. A modern pacemaker can collect data on patients and send it to the hospital or doctors via IoT collection systems (such as WiFi or Bluetooth). This data includes information about the patient's health such as how often the pacemaker is triggering and under what circumstances. Also, patients with mobility issues can benefit from pacemakers that transmit data over the Internet as it means fewer trips to hospitals or doctor's offices [2]. In spite of the developments of the pacemaker, pacemakers can inflict harm on their users. To demonstrate this, we have

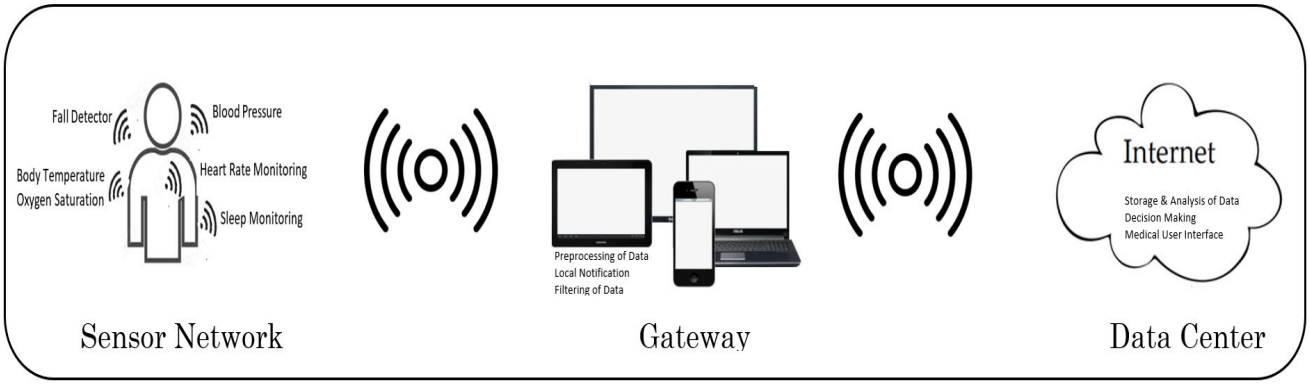


FIGURE 2. IoT-based health monitoring system [32].

collected evidence as shown in the mini cases discussed in section IV-A.

B. SYSTEM SAFETY DOMAIN

In this section, we provide background on error ontologies which are guides to help users consider potential errors in architecture fault modeling:

1) OPEN SOURCE PLATFORMS

IoT experts can use modeling languages to improve the safety of the systems they study or build. We guide the readers to be familiar with the Architecture Analysis and Design Language (AADL) [13], Error Model Annex (EMV2) [14], Open Source Architectural Tool Environment (OSATE) [15], and their implementation. AADL and OSATE are used because of their existing support for our method via various plugins or extensions as well as its presence in the Safety Analysis community.

To fundamentally understand a piece of part of our work, we need to describe some crucial AADL terms to explain error behavior analysis, Figure 8 and hazard recorded for the particular component, Figure 7 as follows [6]:

- **Error Behavior**: Demonstrates how specific system components behave incorrectly.
- **States**: Shows the component's normal and abnormal states in terms of its states.
- **Events**: A error appears as an event.
- **Transitions**: Illustrates how error events cause the state of the component to transition from its normal state to an abnormal one.
- **Hazard Properties**: Hazards are defined in the component specification according to severity and priority.

2) ERROR ONTOLOGY

The AADL Error Annex [13], describes how to represent error types in a hierarchical structure to aid in safety analysis. It also introduces the concept of error classifications, which are used to categorize the various types of faults that can arise during analysis.

The error type is a category label that is used to identify error propagations from one component to another, internal error occurrences of the component itself, component behavior when errors occur, and how errors flow through the component from input port to output port. The label of the error is used as the condition declaration for identifications [10].

The model of Error Annex also permits the description of error behavior within an architectural component. This behavior is described in the format of a state machine, but it permits errors to be traced to their source. It also allows analysts to determine what downstream effects an error in one module might have on others [20].

Error types can be used by stakeholders to explain how components might fail and to link those failures to error events. For example, a sensor failure transmits an incorrect reading (value error) because of overheating, the sensor misses a reading (item omission) because of radiation, or the sensor fails to provide readings (service omission) because of low power [5].

The error ontology classifies errors into six major error types, [5] and [10]:

1) **Service Errors** - Represents errors that occur during the delivery of a service. Omission errors, which indicate that no service was given, and commission errors, which indicate that unintended service was provided, are examples of service errors.
2) **Value Errors** - Errors that are connected to the service value are represented here. Incorrect values, a value outside or outside of the expected range, or a stuck value, the same value being sent repeatedly, are all examples of value errors.
3) **Timing Errors** - Errors relating to service scheduling are represented here. Early or late item delivery and delayed services are examples of timing errors.
4) **Replication Errors** - Errors relating to the delivery of replicated services are represented here. Similar replicated programs, for example, could be delivered at different times.

5) **Concurrency Errors** - Errors related to concurrent system actions, like executing two tasks at the same time. Also, race condition and mutual exclusion errors can be classified as concurrent errors in the memory.
6) **Login Control Errors** - Authentication and authorization errors are examples of errors related to the application of access control systems.

3) ARCHITECTURE FAULT MODELING

Architecture fault modeling and analysis supports automated evaluation of quality attributes for dependable systems such as safety, reliability, and security. The architecture faulting modeling supported by the AADL error model annex enables annotation of an architecture model with fault occurrence, malfunction, and fault propagation behavior to resolve dependability issues in safety critical systems [10]. This approach has three levels of abstraction [6]:

1) **Component Behavior Analysis** - This level supports performing safety analysis with regards to probabilistic reliability as well as availability analysis. Also, this level shows the status of a component in the failure mode when it gets a fault. For example, incoming error propagation and internal faults lead to changing the status of the component from operational mode to failure mode. Furthermore, the component has ability to recover itself from the failure mode.
2) **Compositional Abstraction Analysis** - Combines the various fault models of each component into one global fault model for the entire system. This level permits performing safety analysis of the entire system as a whole rather than each individual component.
3) **Error propagation Analysis** - Error propagation effects on the operational environment of the system by propagating the error among components within a system. This level supports hazard identification and error impact analysis.

C. SYSTEM SAFETY DOMAIN FOR MEDICAL IoT

In this section, we provide state of the art to examine the current research on system safety development for medical IoT systems:

System safety is a crucial area of research for medical IoT devices to ensure patient safety and data privacy. In recent years, researchers have made significant progress in developing system safety approaches for medical IoT devices, as evidenced by recent studies. One recent study proposed a systematic framework for identifying and mitigating safety risks in medical IoT devices [34]. The framework utilized a combination of hazard analysis, risk assessment, and safety management approaches to identify potential hazards and mitigate them before causing harm to patients.

Similarly, a study proposed a machine learning-based approach to enhance the reliability and safety of medical IoT devices [35]. The proposed approach used a combination of anomaly detection, data pre-processing, and prediction algorithms to detect potential failures and take appropriate actions to prevent them.

Furthermore, a recent review of system safety methods for medical IoT devices identified several challenges, including the lack of standardization, interoperability issues, and the complexity of integrating multiple devices [36]. The review also suggested that a comprehensive approach that considers both hardware and software aspects of the devices is necessary to ensure system safety in medical IoT.

Several frameworks and methodologies have been proposed for developing system safety for medical IoT systems. For instance, [37] proposed a hazard analysis and risk assessment framework that can be used to identify and mitigate potential safety hazards in medical IoT systems. The framework includes four stages: hazard identification, hazard analysis, risk assessment, and risk mitigation.

In addition to safety analysis, several studies have proposed using formal methods for system safety development in Medical IoT systems. For instance, [38] proposed using model-based formal methods to ensure the safety of Medical IoT systems. The authors used the Timed Abstract State Machine (TASM) method to model the behavior of Medical IoT systems and verify safety properties. They demonstrated the effectiveness of their approach by applying it to a case study of a Medical IoT system.

Proposed a framework for Medical IoT system safety development, which includes safety management, safety requirements, safety analysis, and safety verification. The authors emphasized the importance of safety management in Medical IoT systems to ensure that safety requirements are met during system development. They also suggested using safety analysis techniques such as Hazard Analysis and Critical Control Points (HACCP) to identify potential safety hazards [39].

Another recent study proposed a novel approach for system safety management in medical IoT devices using a combination of Failure Modes and Effects Analysis (FMEA) and fault tree analysis (FTA) techniques [40]. The study showed that this approach could effectively identify potential system failures and hazards and mitigate them before causing harm to patients.

Similarly, [41] proposed a safety analysis approach for Medical IoT systems, which includes hazard identification, hazard analysis, risk assessment, and risk management. The authors used Fault Tree Analysis (FTA) and Failure Mode and Effects Analysis (FMEA) to identify potential safety hazards in Medical IoT systems. They also suggested using a risk matrix to assess the severity and likelihood of safety hazards.

Several studies have highlighted the need for a system safety approach for medical IoT systems. For example, [42] emphasized the importance of addressing the unique safety challenges of medical IoT systems, such as the integration of different technologies, the heterogeneity of system components, and the potential for cyber attacks.

Furthermore, [43] proposed a safety testing approach for Medical IoT systems, which includes functional testing,

performance testing, and security testing. The authors emphasized the importance of testing in ensuring the safety of Medical IoT systems. They suggested using automated testing tools to increase the efficiency and effectiveness of testing.

Overall, these studies and reviews demonstrate the importance of developing robust system safety approaches for medical IoT devices to ensure the safety and well-being of patients. Medical IoT systems have the potential to revolutionize healthcare by providing personalized and real-time care to patients. However, ensuring the safety of Medical IoT systems is crucial due to the critical nature of healthcare. This literature review examined the current research on system safety development for Medical IoT systems. The studies reviewed proposed various approaches, including safety management, safety analysis, formal methods, and testing, to ensure the safety of Medical IoT systems. These approaches can guide the development of safe and reliable Medical IoT systems, ultimately improving patient care.

IV. METHODOLOGY

In this section, we have established a practical top-down safety analysis approach to determine safety issues in IoT systems. Our method makes use of both AADL and OSATE [15]. It is carried out utilizing the concept of general system specification and initial architecture design. The architecture representation aids stakeholders in recognizing hazards and errors as interactions among components are considered. Following the identification of the hazards, the related errors are annotated with details about the hazards, and this information is built into the system architecture as system development begins. The steps of our process are:

1) Gather Evidence
2) Build Architecture
3) Design Safety Architecture
4) Identify Safety Constraints
5) Identify Quality Attributes

The authors have presented an inter-relationship diagram as shown in Figure 3 that illustrates the connections between the various steps of our proposed method. Furthermore, we have demonstrated how our methodology relates to the input of the problem statement and the output of the solution for the problem.

As shown in Figure 3, we have clarified that the complexity of the IoT creates a challenge in ensuring safety, as the increased interaction and tight coupling among components can result in logical errors that pose new safety risks. To demonstrate the impact of these issues, we proposed a methodology for examining medical IoT systems, such as pacemakers. The first step is for the analyst to gather information about faults that have occurred in IoT products, leading to harm or potentially harmful situations. The second step involves building a feedback control loop system architecture for the identified IoT system, based on its essential components: sensors, controllers, actuators, and Internet connectivity. This feedback control loop architecture enables analysts to observe the interaction among components and diagnose abnormal components that could cause faults in the system. It also generates a model that supports further analysis, such as identifying the source of the error, how the error propagated, and the destination of the error. The third step is for stakeholders to design a safety architecture for the system, using a system modeling language to record the identified errors in the model that could lead to hazards. In this step, we utilized the Architecture Analysis and Design Language (AADL) [13] and the Error Model Annex (EMV2) [13], which are supported by the Open Source Architectural Tool Environment (OSATE) [15] to build architecture models and develop error state and behavior analysis. Finally, in the fourth step, stakeholders identify suitable safety requirements that mitigate the identified hazards. This step provides developers with the necessary knowledge to prevent identified hazards from occurring by explicitly handling the errors. At the end, the proposed method allows for the identification of missed safety constraints in IoT products before construction.

The subsequent subsections will cover each step in greater detail.

A. GATHER EVIDENCE (PACEMAKER–CASE STUDIES)

In this step, significant stakeholders gather information about faults that have happened in IoT products (both similar and non-similar) which have led to harm or potentially harmful situations. This step depends on the use of error ontologies and is supplemented by stakeholder experience in systems such as medical devices and smart vehicles. This step helps the stakeholders / analysts to identify what kind of errors to mitigate against. To implement this step, we have systematically collected several studies detailing errors in pacemakers:

1) **Mini-Case (Pacing Failure)**: A patient in the emergency room (80 years old) whose pacemaker was unable to regulate his atrial channel. Four years prior to the incident, the patient had a pacemaker implanted. Interrogation of the pacemaker revealed pacing thresholds were set too high [3].
2) **Mini-Case (Mode Failure)**: Photon therapy was used to treat a 76 year old patient who had inoperable cancer. The pacemaker's dysfunction was most likely caused by electromagnetic interference during radiotherapy according to the findings. The error message suggested that there was a problem with the software. Further investigation found that the device's DDD mode had switched to a bipolar AAI mode [1].
3) **Mini-Case (Error Message)**: A 77 year old man was undergoing aortic valve and coronary artery bypass surgery. His pacemaker failed to discharge when it turned on, instead displaying an error message on the pacemaker's monitor. Intensive-care staff were not able to access the program of the pacemaker [9].
4) **Mini-Case (Software Error)**: Due to the risk of a software error, Medtronic is recalling its dual chamber Implantable Pulse Generators. Patients and clinicians

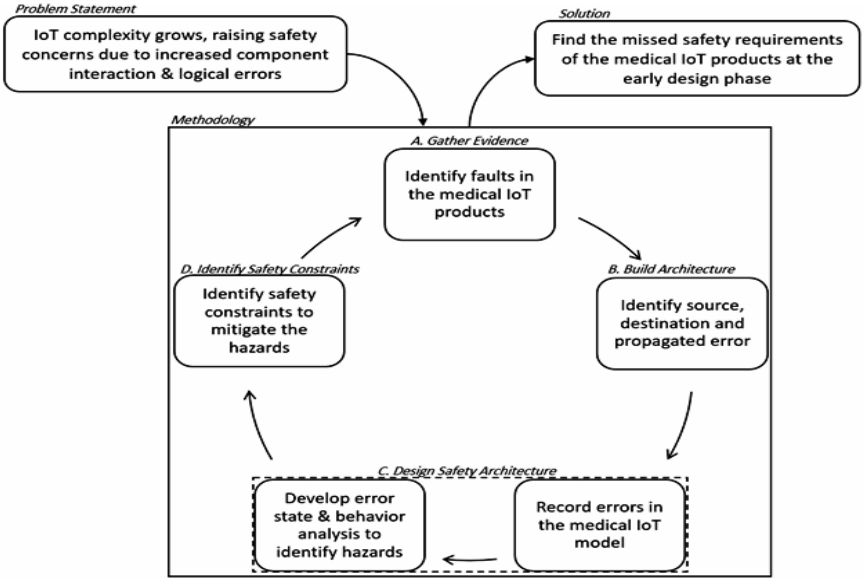


FIGURE 3. Inter-relationship among steps of the proposed method.

have no way of knowing whether or when a software malfunction will occur. Patients can experience symptoms such as light headedness as a result of a lack of pacing [18].

5) **Mini-Case (Data Corruption)**: A dual chamber pacemaker programmed in DDDR mode was given to an 85 year old man who needed a pacemaker due to high-grade atrioventricular block. The system's cybersecurity update prompt appeared during routine pacemaker interrogation and the update was initiated. A delay occurred before the resumption of pacing during the cybersecurity upgrade phase, and it was linked to the patient feeling lightheaded. An effort to export the interrogation data after the software upgrade failed due to an error caused by missing or corrupt data [8].
6) **Mini-Case (Explants Pacemaker)**: Boston Scientific Product Performance published a report concerning failed return rate for an explanted pacemaker. Figure 4 shows the recorded percentage of failures rate of the pacemaker that were explanted and then returned to Boston Scientific for different product therapy forms. The failure rate of the pacemakers has increased from 58% since 2006 to 69% since 2016. Failures here represent both hardware and software failures [12].

B. BUILD ARCHITECTURE

In this step, system architects and analysts build a feedback control loop system architecture for the identified IoT system based on the important components of the system: sensors, controllers, actuators and Internet connectivity. In addition, the feedback control loop architecture allows the analysts to see the interaction among components and how they work with each other. For example, the analysts want to know what will happen if the sensor does not send the correct value to the controller. This step allows the architects / analysts to diagnose abnormal components in the loop which could lead to faults in the system. It also yields a model supporting further analysis such as identifying source of the error, how the error propagated, and destination of the error.

Below, we have developed a feedback control loop architecture for the an IoT enabled pacemaker. Pacemaker systems consist of three important components which are the Pulse Generator (PG), Device Controller-Monitor (DCM), and Leads (Sensors). The specification of each component and how it works have been described in detail in [16]. Additionally, we have extended the system by adding two additional components which are the ''Thing'' and the ''Internet''. We have connected all of the components together in the form of an architectural diagram as shown in Figure 5. We describe the connections as follows:

First, the pulse generator (PG) consists of the sensing unit, the controlling unit, and the pacing unit. In the sensing unit, we have an accelerometer which is used to read electrical values from the heart and dispatch them to the controller. In the controlling unit, accelerometer values are used to generate a pacing signal which is sent to the pacing unit. Each pacing signal will be sent to the pacing unit as a command. In addition, based on the doctor's decision, the technician will configure the device into a specific mode such as AAT, VVT, AOO, AAI,...etc. Each mode has a status and will be monitored remotely through the Internet. In the pacing unit, an actuator, called Rate-Adaptive-Pacing, will get the command from the controller to send electrical pulses to the heart. These pulses will be a result of the computations done in the controlling unit. The electrical pulses are provided to the heart via the leads.

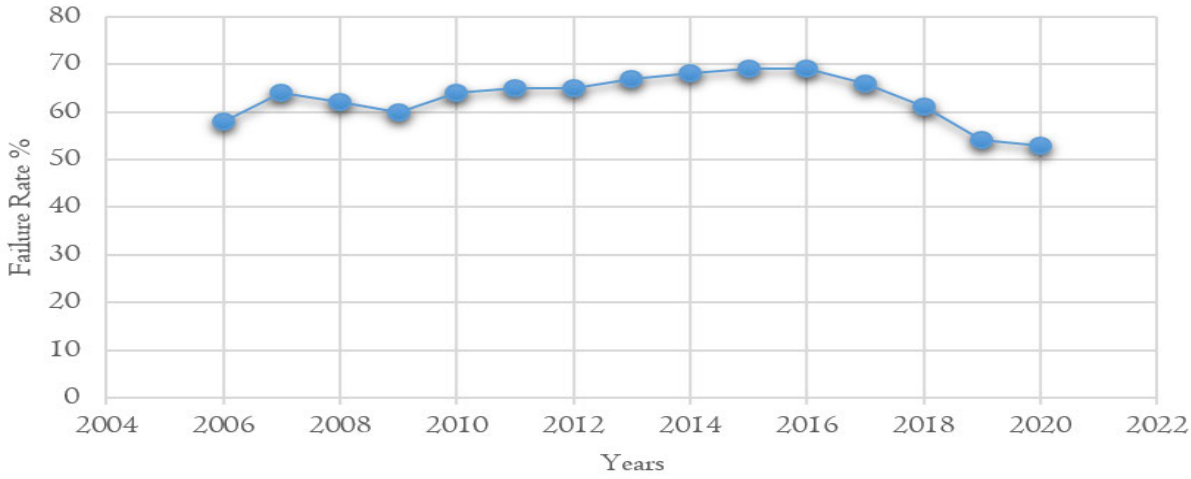


FIGURE 4. Failure rate of explanted pacemaker [12].

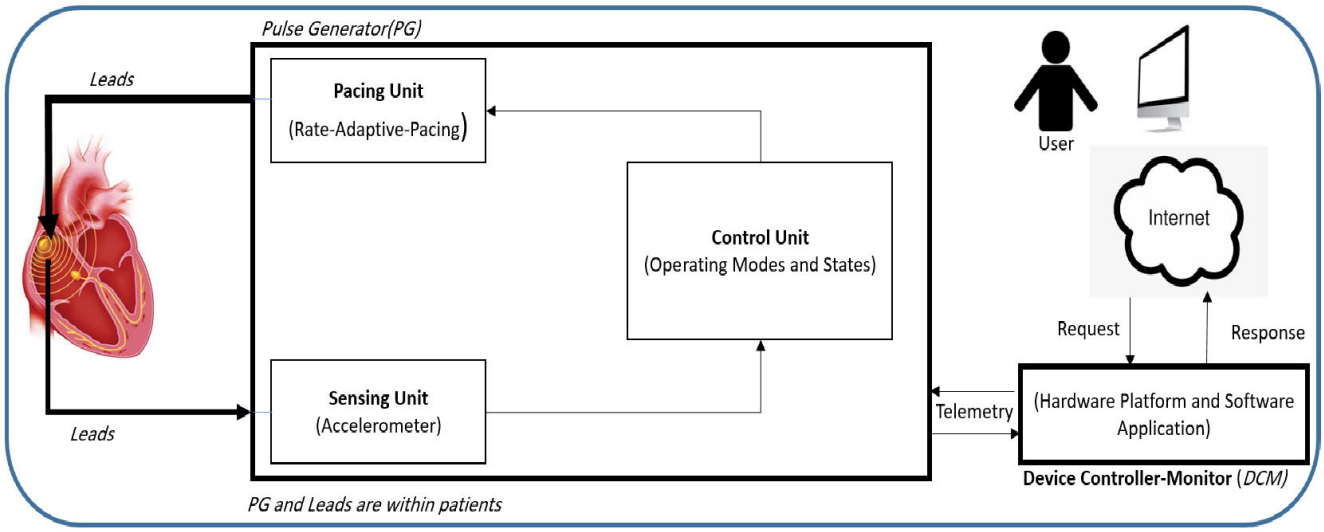


FIGURE 5. Pacemaker IoT architecture.

Second, the Device Controller-Monitor (DCM) is the hardware platform and software of the device. It is used to configure the device based on the physician's description via bidirectional telemetry access ports. In addition, the DCM is used to monitor the functionality of the device and monitor the patient's heart remotely via the Internet. At the same time, the user can update the software components of the device through the Internet by sending a request command and getting the response back from the device remotely.

Finally, the feedback control loop system architecture of the pacemaker allows us to identify abnormal components which have taken error values in the loop and lead to the generation of errors for the pacemaker. Based on this, we have connected the previous step IV-A to the current step to show the errors for each real mini case study enabling us to know that where the error is located.

The result of the current step has shown in the table 1. For the first mini-case study, the affected component is the pacing unit but the error originally propagated from the control unit. For the second mini-case study, the control unit made a wrong decision by changing the mode of operation because of incorrect sensing values. For the third mini-case study, the control unit has an error propagation for itself which led to an inactive status. For the fourth mini-case study, the source of the error is the control unit which sent the incorrect command. For the fifth mini-case study, the device controller monitor has error propagation for itself which led to failure to update. The final case study shows that the errors have different sources which led to multiple different effects.

C. DESIGN SAFETY ARCHITECTURE FOR THE IOT APPLICATION

In this step, stakeholders design a safety architecture for the system using one of the system modeling languages to record the identified errors in the model which could lead to hazards.

TABLE 1. Results of IoT architecture for pacemaker.

Mini Case	Source of Error	Error Propagation	Destination of Error
1	Control Unit	no pacing or high pacing	Pacing Unit
2	Sensing Unit	shifted operational mode	Control Unit
3	Control Unit	inactivate status	Control Unit
4	Control Unit	missing of pacing	Pacing Unit
5	Device Controller-Monitor	missing data	Device Controller-Monitor
6	Different Original Units	Different types of errors	Explanted Pacemaker

In this step, we have used the Architecture Analysis and Design Language (AADL) [13] and the Error Model Annex (EMV2) [10] which are supported by the Open Source Architectural Tool Environment (OSATE) [15] to build architecture models, develop error state / behavior analysis.

So, the error state analysis is the process of identifying potential error states in a system, modeling the behavior of the system in each error state, and developing strategies to handle errors. The goal of error state analysis is to ensure that the system is resilient and can continue to function correctly even in the face of unexpected events. This involves developing models that describe the behavior of the system in each potential error state, and developing strategies to handle errors, based on the error state models.

In addition, the error behavior analysis, on the other hand, is the process of analyzing the behavior of a system when errors occur. This involves developing models that describe the behavior of the system when errors are detected, and defining recovery actions to ensure that the system can continue to function correctly even when errors occur. Error Behavior Analysis is concerned with how the system behaves in response to errors, and how it can recover from errors to return to a safe state.

Both error state analysis and error behavior analysis are important techniques for managing system errors in AADL. By identifying potential error states and modeling the behavior of the system in each error state, designers can develop strategies to mitigate the effects of errors and ensure that the system is resilient. By analyzing the behavior of the system when errors occur, designers can develop recovery actions to ensure that the system can recover from errors and return to a safe state.

To better present this step, we have developed error and behavioral analyses for the first mini case study 1.

1) ERROR STATE ANALYSIS

We have analyzed the first mini case study to determine what caused the harm inflicted on the patient. As shown in Figure 6, two types of errors happened in the pacing unit which are *"No Data"* and *"Bad Data"*. The *"No Data"* error came from failed pacing and *"Bad Data""* error came from high or low pacing.

In the context of normal operation, the pacing unit should work correctly, but it does not because it gets *"No Data"* from the controller which leads to the pacing unit changing from normal pacing to failed pacing. The pacing unit can return to normal if the *"Recover1"* event runs. It depends on the design of the pulse generator (PG) whether that event can fire with respect to the type of the error. Also, the designers should specify that how long it takes to run the recovery event.

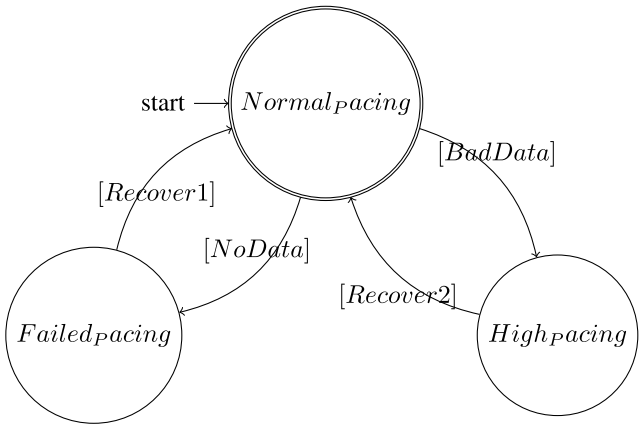


FIGURE 6. Error state analysis- results of first mini case study.

In addition, should the pacing unit have gotten *"Bad Data"* from the controlling unit, the state of the pacing unit would change from normal pacing to high pacing. Furthermore, the pacing unit can get back to normal if the *"Recover2"* event is run. The error state analysis tells us that both errors have led to harm potentially being inflicted on the patient because no pacing was able to be provided.

2) ERROR BEHAVIOR ANALYSIS

In the early design phase of the pacemaker development life cycle, we have provided the error behavior analysis as shown in Figure 7. It can be used as a standard error pattern and reused across the architecture model. In the pacing unit, we have provided the AADL representation to analyze the behavior by recording the states, events, and transitions. In the states, we have recorded normal pacing (initial state), failed pacing, and high pacing. In the events, we have recorded *"No Data"* and *"Bad Data"* as the error events. In addition, *"Recover1"* and *"Recover2"* have been recorded as recovery events. In the transitions, we have represented failure1 as a transition that will happen if the normal pacing state changes to failed pacing. Also, we have represented recovery1 as a transition that will happen if the failed pacing state changes to normal pacing. We have also represented failure2 as a transition that will happen if the normal pacing state changes to high pacing. The final transition is recovery2

```
error behavior Pacing_Unit_Three_States
use types First_Mini_Case;
states
    Normal_Pacing: initial state;
        --pacing unit is operating
        normally
    Failed_Pacing:state; --pacing unit
        do not have output
    High_Pacing:state; --pacing unit is
        computing and outputting bad
        values
events
    NoData:error event {NoData} if "it
        happens in pacing unit if no data
        calculated";
    BadData: error event {BadData} if
        "it happens in pacing unit if bad
        values have calculated"
    Recover1: error event {Recover1} if
        "repairs being made for No Data";
    Recover2: error event {Recover2} if
        "repairs being made for Bad Data";
transitions
    Failure1: Normal_Pacing -[No_Data]
        -> Failed_Pacing;
    Recovery1: Failed_Pacing -
        [Recover1] -> Normal_Pacing
    Failure2: Normal_Pacing -[Bad_Data]
        -> High_Pacing;
    Recovery2: High_Pacing -[Recover2]
        -> Normal_Pacing;
end behavior;
```

FIGURE 7. Error behavior analysis - results of first mini case study.

```
EMV2::hazards =>
(
  [
    CrossReference => "Hazard:H1";
    Description => "Pulse generator has
        no pacing to the heart";
    Severity => 1;
    CrossReference => "Hazard:H2";
    Description => "Pulse generator has
        high pacing to the heart";
    Severity => 1;
  ]
) applies to pacing_unit;
```

FIGURE 8. Hazard recorded in the pacing unit - results for the first mini case study.

which happens when state of the pacing unit changes from high pacing to normal pacing.

As shown in Figure 8, we have applied the hazards identified in the error model (EMV2) to the pacing unit. For example, the hazard (H1) will happen if the pacing unit does not provide pacing to the heart. Additionally, the hazard (H2) will happen if the pacing unit provides high pacing to the heart. So, it can be concluded that the errors identified are sources of these hazards. These two hazards have severity level of 1, or can induce severe harm.

D. IDENTIFY SAFETY CONSTRAINTS

In the final step, stakeholders find suitable safety requirements that mitigate the identified hazards. This step helps developers to know what steps must be taken to prevent the identified hazards from occurring by knowing which errors must be explicitly handled. This step allow us to find safety constraints for the system before it is constructed.

Since we have identified hazards that could lead to potential harm being inflicted on the patient, we now need to prescribe mitigations to prevent the hazards from occurring. These mitigations will be in the form of safety requirements. For the first mini case study, we have provided safety constraints to address the hazards of the pacing unit as shown in table 2.

E. IDENTIFY QUALITY ATTRIBUTES

In this section, authors identified some quality attributes for the evaluation of safety requirements in the pacemaker IoT architecture:

1) **Reliability**: The system should be reliable, meaning it should perform its intended function without failure or errors. This is important for safety-critical IoT systems to prevent criticality of the things. To solve the criticality of the things, authors worked on the reliability quality attribute by developing a method to find the missed/incorrect requirements of the medical IoT product at the early design phase. For that purpose, we have focused on gather information about faults that have happened in medical IoT device such as pacemaker, which have led to harm or potentially harmful situations. We found that most of the failures happened based on the component and system level which led explant the pacemaker. As we know that, based on the reliability engineering principles, failure is the result of an error and the error comes from fault. So, now the source of the failure which is fault. Through our methodology, we found faults that have been made at the early design phase.
2) **Safety**: Safety should be a primary concern throughout the system's lifecycle, from design to operation and maintenance. The authors worked on the safety quality attribute by building feedback control loop architecture to identify the path of the error (source-path-destination) and provide the safety constraints to mitigate it. For example, for the first mini-case study, the affected component is the pacing unit but the error originally propagated from the control unit. The patient gets hurt because of the high or no pacing error. So, the error may or may not be the source of the harm (i.e hazard). Now, the hazards are mitigated by providing safety requirements at the early phase.

To evaluate the safety requirements for the first mini-case study, we provide the following Figure 9 to make sure that the hazards have been mitigated by the safety constraints. By considering these quality attributes, we can ensure that the system meets its safety requirements and is reliable and safe.

TABLE 2. Safety constraints identification - results of first mini case study.

Hazards (H)	Safety Constraints (SC)
H1:Pacing unit does not provide pacing.	SC1: Pacing should be provided at all times if it is commanded.
H2:Pacing unit does provide high pacing.	SC2: Pacing level must be provided at the commanded level.

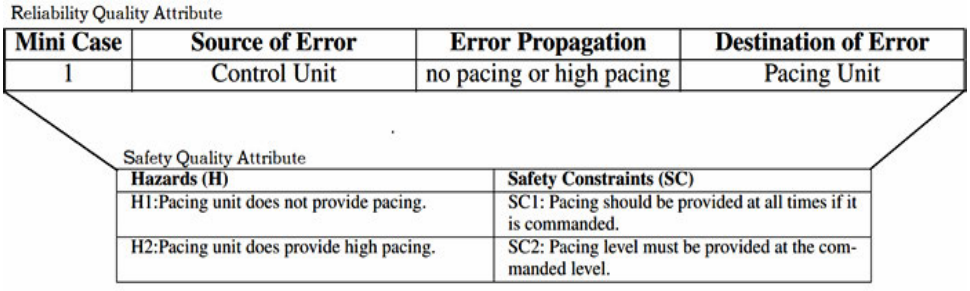


FIGURE 9. Evaluation of safety requirements – first mini case study.

Regarding the difficulty in evaluating the technical perspective of safety analysis methods, one of the main challenges is the complexity of the systems, which often involves multiple interconnected devices and systems with varying levels of autonomy and interdependence. Additionally, lack of standardization and consistency in safety analysis methods and tools as a challenge, which can make it difficult to compare and evaluate different approaches. Addressing these challenges requires a systematic and rigorous approach to safety analysis, along with appropriate expertise and resources to effectively evaluate the technical perspective of safety analysis methods in IoT systems.

V. METHOD VALIDATION

In this section, we present an architecture that is built upon traceability criteria. The purpose of this architecture is to validate the effectiveness of our proposed method. It focuses on identifying errors, hazards, and safety constraints. Furthermore, we need to consider whether our model is applicable to a wider range of IoT devices or if it is limited only to medical IoT devices.

The proposed method is suitable for application to other IoT devices based on their level of safety criticality. For instance, during the COVID-19 pandemic, IoT systems have played a crucial role in monitoring the health of patients. Consider a scenario where a medical IoT device inaccurately measures body temperature, pulse rate, and/or oxygen saturation. This could lead to medical professionals making incorrect decisions, such as diagnosing a patient with COVID-19 when they do not actually have the virus. In such a case, the patient may suffer adverse consequences. This example is not limited to COVID-19 monitoring but also extends to other medical IoT monitoring devices, such as those used for glucose monitoring, heart rate monitoring, depression/mood monitoring, asthma inhaler monitoring, and more.

The case studies presented in section IV-A involving pacemakers provide compelling objective evidence for the necessity of a methodology to identify hazards and errors during the design phase of medical IoT devices. Therefore, the proposed method and its validity effectively fulfill this crucial need.

In addition, our approach emphasizes the importance of traceability criteria in validating the results. To achieve this, we have designed an architecture, illustrated in Figure 10, that highlights how errors can propagate from individual IoT components to the overall IoT system. This architecture comprehensively covers all aspects of the results, including error propagation, hazards, and safety constraints.

Traceability is a fundamental principle in software engineering, and it serves as a prerequisite for establishing valid requirements. To achieve this, we have developed an architecture, as depicted in Figure 10, that traces errors from the individual IoT component level to the overall IoT system level. We have followed the longest path for the trace, which is represented by the red line. Please follow the red line as described in detail below:

During the early design phase of safety-critical IoT system development, stakeholders may introduce faults, including missing requirements, missing values, out-of-range values, or values that are above or below the expected range. These faults enter the system through the Sys_{in} port in the form of errors. According to the principles of reliability system engineering, errors are caused by faults and can result in failures.

So, the error enters the Controller through the port of C_{in}. If the Error is recognized in the Controller, it will be handled by the controller itself. But, if it is unrecognized, it leads to changing from normal state to failure. So, the Error propagates through the failure controller to the output port which is C_{out}.

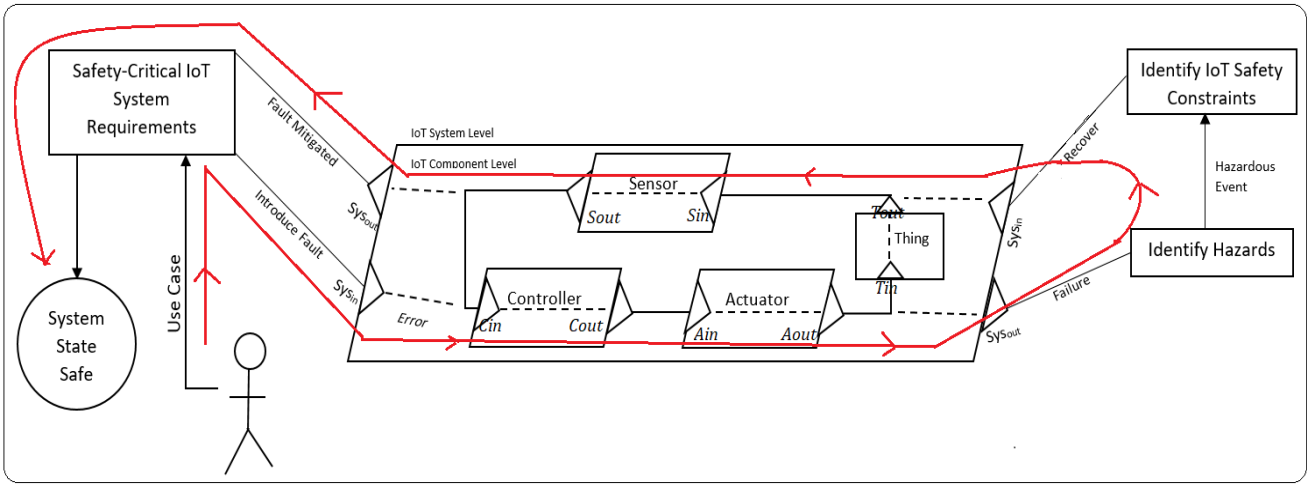


FIGURE 10. Traceability of IoT safety constraints.

The error then travels through the connection line and reaches the actuator. It enters the actuator through the A_{in} port. If the actuator recognizes the error, it will handle it accordingly. However, if the error goes undetected, it can cause the actuator to transition from a normal to a failure state. The error then passes through the actuator and reaches the A_{out} port. Finally, the error continues on to the Sys_{out} port, causing the entire IoT system to transition into a failure state. This occurs because the error propagates from the individual component level to the overall system level.

According to safety engineering principles, failures may or may not pose a potential hazard. Stakeholders must identify any potential hazards during the early design phase by incorporating hazardous events into the state of the components. Additionally, stakeholders must identify safety constraints that can help mitigate these hazards at the system level. For instance, if an error occurs due to an out-of-range value, the system can be designed to force the values to stay within range, or ignore any out-of-range values altogether. In this way, the system only deals with normal values. When a normal value enters the system through the Sys_{in} port, the system recovers from failure and returns to its normal state. This ensures that the system can operate safely and without any hazards.

The normal value is then transmitted to the S_{in} port of the Sensor via the connection line. The Sensor reads and calculates the normal value, sending it to the output port S_{out}. The calculated value is then received by the Controller through the C_{in} port, allowing both the Controller and Actuator to recover.

Tracing errors based on the component level allow us to identify the faults which have been introduced in the requirements of the safety-critical IoT systems. So, the faults and hazards have been mitigated based on system level by identifying safety constraints. The end of the red line trace shows that the IoT system state is safe from the identified hazards or their level has been sufficiently reduced as to be acceptable.

VI. METHOD EVALUATION

In this section, we will evaluate the top-down architectural safety analysis technique that is used for developing life-critical IoT systems. This method enables stakeholders to reduce the impact of errors by developing safety requirements during the development of an IoT architecture model.

We focus on identifying safety requirements for hazards in our method. To do this, we injected the error ontology into an IoT architecture model of a pacemaker. This allowed us to identify unsafe actions through error behavior analysis and to determine the causes of those actions through error state analysis.

So, we evaluate our technique based on the following claims:

Claim 1: The lack of safety analysis methods that address the criticality of ''Things'' in cross-domain areas such as IoT and life-critical systems is a serious problem. This claim is supported by the fact that IoT medical devices have caused harm, and in some cases, this harm was caused by the IoT functionality itself, as demonstrated in SectionIV-A.

Claim 2: Our method and its complements offer improved validation of existing hazard analysis approaches, such as [21], [22], [23], and [25], by providing multiple layers of evaluation. By identifying requirements, designing and developing safety architecture, and applying state and behavioral analyses for errors early in the design phase, we offer a comprehensive approach that goes beyond traditional hazard analysis. This can be demonstrated by our safety analysis of a pacemaker architecture in sections IV-C and IV-D.

Claim 3: One benefit of our proposed safety analysis method for IoT systems is that it can be carried out before the implementation stage, allowing for early detection and mitigation of potential hazards. This is in contrast to many existing methods that delay safety evaluation until the system is partially implemented. As evidence, we demonstrate in Section IV-B how our approach was applied to analyze the safety of a pacemaker architecture prior to implementation.

Our method addresses all three claims and reduces the requirement for expert domain knowledge by utilizing existing safety analysis tools from other fields. We leverage the error ontologies and state analysis tools from the AADL community to identify safety requirements, design safety architecture, and analyze errors at an early stage. This approach eliminates the need to delay safety evaluation until the system is partially actualized, as many methods require. Additionally, AADL is an open-source tool that is freely available and supported by an active and growing community.

VII. DISCUSSION

In this section, we review the capabilities of our proposed safety analysis method:

- Our method aids in the elimination of risky situations by allowing for complex analysis of inter-connected systems, either at scale or of individual components.
- Our method can detect possible internal faults in main components of the feedback-control loop system design, reducing the risk of residual hazards in IoT systems which use that pattern.
- Based on a two-way communication format(i.e propagating an error from a component to another such as from controller to actuator) between components and back-tracing for the error, our method will display the effects of unsafe interactions on the entire system.
- Our approach aids IoT organizations and businesses in reducing their reliance on human experts in recognizing hazardous circumstances for their IoT system through the use of Error Ontologies.
- For current IoT systems, our method may help to uncover previously unknown safety hazards.

A. CHALLENGES IN METHOD VALIDATION

The challenges presented in this section are primarily related to ensuring the effectiveness and applicability of the proposed method for identifying errors, hazards, and safety constraints in IoT devices, particularly those used in medical settings.

- One of the significant challenges is determining whether the proposed approach is limited to medical IoT devices or can be applied to other types of IoT devices based on their level of safety criticality. To address this, stakeholders must evaluate the safety-criticality of the IoT devices and determine the appropriate level of traceability required for identifying errors and hazards.
- Another challenge is ensuring that the proposed method considers all aspects of the IoT system, including individual components and their interconnections. The proposed architecture aims to address this challenge by tracing errors from the individual component level to the overall IoT system level, which helps stakeholders to identify potential hazards and safety constraints.
- Ensuring traceability and validation of the proposed method is another challenge. Traceability is essential for establishing valid requirements, and stakeholders must ensure that the architecture effectively traces errors and identifies potential hazards and safety constraints. Validation involves demonstrating that the proposed method is effective in identifying errors and hazards and reducing their potential impact on the IoT system's safety.

B. CHALLENGES IN METHOD EVALUATION

The challenges in the evaluation of the top-down architectural safety analysis technique for developing life-critical IoT systems may include:

- Complexity of IoT systems: IoT systems are highly complex and often involve numerous interconnected devices, sensors, and networks. The complexity of these systems can make it challenging to identify potential hazards and safety requirements, as well as design and implement safety architecture.
- Lack of standards: There is currently a lack of standardization in IoT safety analysis methods and tools, making it difficult for stakeholders to adopt consistent and effective approaches to safety analysis.
- Dynamic nature of IoT systems: IoT systems are highly dynamic and can change rapidly over time, making it difficult to ensure that safety requirements and architecture remain effective throughout the system's life cycle.
- Difficulty in predicting all possible scenarios: Despite the use of error ontologies and state analysis tools, it may still be challenging to predict all potential error scenarios that could lead to safety hazards in an IoT system.

VIII. CONCLUSION AND FUTURE WORK

Our proposed hazard analysis method, based on early design knowledge, allows for the discovery of additional hazards and identification of safety constraints to mitigate those hazards. This guidance helps stakeholders understand how IoT components can produce errors, how errors can be propagated among components, and how users of the systems may be harmed.

Our research found that tracing errors lead to the discovery of additional hazards in medical IoT device architectures. Errors in IoT systems may or may not result in a hazard, it depends on the cause and system under consideration. Our method's core idea is that an error analysis of the IoT domain yields a list of potential hazards that other IoT systems should consider.

Finally, the IoT experts and safety engineers can use this approach to efficiently and effectively identify hazards in device interactions and ensure critical IoT systems have appropriate measures against known and emerging hazards.

To further validate our method, the proposed approach will undergo implementation and comparative analysis with existing methods. Subsequently, it will be thoroughly tested on widely used systems to showcase the advantages and effectiveness of the suggested safety analysis approach.

To extend of our method, we have a plan to analyze the impact of battery usage and deterioration on IoT device

TABLE 3. List of abbreviations.

Symbols	Stands for
H	Hazards
SC	Safety Constraints
Sys_{in}	System input
Sys_{out}	System output
C_{in}	Controller input
C_{out}	Controller output
A_{in}	Actuator input
A_{out}	Actuator output
S_{in}	Sensor input
S_{out}	Sensor output
T_{in}	Things input
T_{out}	Things output

performance and error identification in a safety-critical system. This could include studying how battery capacity and condition affect sensor readings and communication reliability, as well as identifying potential issues such as false positives and false negatives that may arise due to battery-related problems.

APPENDIX

In this section, the authors have provided a comprehensive list of symbols utilized in the validation section V.

Competing Interests: The authors whose names are listed above certify that they have NO involvement in any organization or entity with any financial interest (such as educational grants; participation in speakers' bureaus; membership, employment, consultancies, or other equity interest), in the subject matter or materials discussed in this manuscript. This statement is signed by all the authors to indicate agreement that the above information is true and correct. (Authors' photos, names, emails are our signature)!

ACKNOWLEDGMENT

The author would like to thank Xavier University for covering the publication fee of the paper.

REFERENCES

[1] A. Zweng, R. Schuster, R. Hawlicek, and H. S. Weber, "Life-threatening pacemaker dysfunction associated with therapeutic radiation: A case report," *Angiol. J.*, vol. 60, no. 4, pp. 509–512, Aug. 2009.

[2] A. Chacko and T. Hayajneh, "Security and privacy issues with IoT in healthcare," *J. Endorsed Trans. Pervasive Health Technol.*, vol. 4, no. 14, pp. 1–7, 2018.

[3] B. Obszański, Ł. Tułecki, A. Kutarski, and A. Kleinrok, "Lightning-induced pacing system malfunction: A case report," *Eur. Heart J.-Case Rep.*, vol. 3, no. 2, pp. 1–3, Jun. 2019.

[4] C. Chen and S. Helal, "A device-centric approach to a safer Internet of Things," in *Proc. NoME-IoT*, New York, NY, USA, 2011, pp. 1–6.

[5] F. Peter, "Architecture-led safety analysis of the joint multi-role (JMR) joint common architecture (JCA) demonstration system," Carnegie Mellon Univ./Softw. Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU-SEI-2015-SR-032, Dec. 2015.

[6] P. Feiler and D. Gluch, *Model-Based Engineering With AADL: An Introduction to the SAE Architecture Analysis and Design Language*. Upper Saddle River, NJ, USA: Addison-Wesley, 2013.

[7] I. Silva, R. Leandro, D. Macedo, and L. A. Guedes, "A dependability evaluation tool for the Internet of Things," *Comput. Electr. Eng.*, vol. 39, no. 7, pp. 2005–2018, Oct. 2013.

[8] Z. J. Lee, J. M. Henrich, P. Bibby, K. S. Mulpuru, A. P. Friedman, Y.-M. Cha, and K. Srivathsan, "Pacemaker firmware update and interrogation malfunction," *Heart Rhythm J.*, vol. 5, no. 4, pp. P213–P216, 2019.

[9] B. Kleinman, M. Baumann, and C. Andrus, "Faulty design resulting in temporary pacemaker failure," *Chest*, vol. 120, no. 2, pp. 684–685, Aug. 2001.

[10] P. H. Feiler et al., "Architecture fault modeling and analysis with the error model annex, version 2," Carnegie Mellon Univ./Softw. Eng. Inst., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2016-TR-009, Jun. 2016.

[11] Postscapes. (2021). *IoT Home Guide | 2019 Overview of the Best Connected Home Products*. Accessed: May 14, 2021. [Online]. Available: https://www.postscapes.com/internet-of-things-award/connected-homeproducts/

[12] R. Russie, O. Hedrich, S. Zurn, and J. Litzau, *Rhythm Management Product Performance Report*, 1st ed. Boston, MA, USA: Scientific Advancing for Life, 2020.

[13] *Architecture Analysis and Design Language (AADL)*, Standard SAE AS5506B, 2012. [Online]. Available: https://www.sae.org/standards/content/as5506b/

[14] *SAE Architecture Analysis and Design Language (AADL) Annex Volume 1: Annex A: ARINC653 Annex, Annex C: Code Generation Annex, Annex E: Error Model Annex*, SAE Standard SAE AS5506/1A, 2015. [Online]. Available: https://www.sae.org/standards/content/as5506/1a/

[15] Software Engineering Institute. (2021). *Open Source AADL Tool Environment (OSATE), Version 2.9*, o https://osate.org/

[16] Software Quality Research Lab, McMaster University, Boston Scientific Copyright. (2007). *Pacemaker System Specification*. [Online]. Available: https://cutt.ly/2vaQ0ah

[17] K. Tarakji, S. Zweibel, A. Seiler, P. Roberts, N. Shaik, J. Silverstein, A. Patwala, S. Mittal, G. Molon, G. Augello, A. Porfilio, K. Holloman, N. Varma, S. Sears, and M. Turakhia, "P577Early experience with the first pacemakers to directly connect with smart devices for remote monitoring," *Eur. Heart J.*, vol. 40, no. 1, pp. 0188–747, Oct. 2019.

[18] U.S. Food and Drug Administration, Medtronic. (2019). *Recalls Dual Chamber Implantable Pulse Generators (IPGs) Due to Possible Circuit Error*. [Online]. Available: https://cutt.ly/wckygFl

[19] J. Bk, *Electronic Safety Systems*. Sausalito, CA, USA: Univ. Science Press, 2004.

[20] J. Delange, *AADL in Practice: Become an Expert of Software Architecture Modeling and Analysis*. Reblochon Development, 2017. [Online]. Available: https://www.amazon.com/AADL-Practice-software-architecture-modeling/dp/0692899642

[21] N. Leveson, "A new accident model for engineering safer systems," *Saf. Sci. J.*, vol. 42, no. 4, pp. 237–270, Apr. 2004.

[22] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA, USA: MIT Press, 2011.

[23] N. Leveson, "An STPA primer: Version 1," MIT Univ., Cambridge, MA, USA, Tech. Rep. 1, 2013. [Online]. Available: https://psas.scripts.mit.edu/home/wp-content/uploads/2013/10/An-STPA-Primer-version-0-4.pdf

[24] J. Thomas and N. Leveson, "Performing hazard analysis on complex, software and human-intensive systems," in *Proc. 29th Int. Syst. Saf. Conf.*, Las Vegas, NV, USA, 2011, pp. 1–9.

[25] J. Thomas and N. Leveson, "A hazard analysis technique for the Internet of Things (IoT) and mobile gregory pope, CSQE," MIT Press, Cambridge, MA, USA, Tech. Rep. LLNL-CONF-727317, 2017. [Online]. Available: http://psas.scripts.mit.edu/home/wp-content/uploads/2017/04/Greg-Pope_STPA-For-IoT-And-Mobile-Software_Paper.pdf

[26] A. Zhu. *Hazards and Safety Requirements of Internet of Things (IoT)*, Accessed: Sep. 25, 2021. [Online]. Available: https://www.intertek.com/blog/2018-09-11-iot/

[27] J. Rauscher and B. Bauer, "Adaptation of architecture analyses: An IoT safety and security flaw assessment approach," in *Proc. 14th Int. Joint Conf. Biomed. Eng. Syst. Technol.*, vol. 4, 2021, pp. 320–327.

[28] T. Hayakawa, R. Sasaki, H. Hayashi, Y. Takahashi, T. Kaneko, and T. Okubo, "Proposal and application of security/safety evaluation method for medical device system that includes IoT," in *Proc. 7th Int. Conf. Netw., Commun. Comput.*, 2018, pp. 157–164.

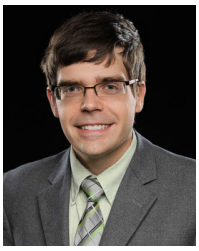
[29] M. E. Laarouchi, "A safety approach for CPS-IoT," Ph.D. dissertation, Institut Polytechnique de Paris, Palaiseau, France, 2020. [Online]. Available: https://tel.archives-ouvertes.fr/tel-03051734

[30] R. Weaver, *The Safety of Software—Constructing and Assuring Arguments*. Seattle, WA, USA: Allen Institute for Artificial Intelligence (AI2), 2003.

[31] B. Pradhan, S. Bhattacharyya, and K. Pal, "IoT-based applications in healthcare devices," *J. Healthcare Eng.*, vol. 2021, p. 18, Mar. 2021, doi: 10.1155/2021/6632599.
[32] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Future Gener. Comput. Syst.*, vol. 78, pp. 641–658, Jan. 2018.
[33] H. Raad, *Fundamentals of IoT and Wearable Technology Design*. Hoboken, NJ, USA: Wiley, 2021. [Online]. Available: https://ieeexplore.ieee.org/book/9296419
[34] Z. Li, Z. Jiang, C. Chen, and Z. Liu, "A systematic framework for identifying and mitigating safety risks in Medical Internet of Things," *IEEE Access*, vol. 9, pp. 108808–108819, 2021.
[35] W. Jiang, L. Huang, X. Chen, and Y. Li, "Machine learning-based system safety research for medical IoT," *IEEE Access*, vol. 9, pp. 22306–22314, 2021.
[36] J. Kim, S. Yoo, and S. Lee, "System safety methods for medical IoT devices: A review," *J. Healthcare Eng.*, vol. 2021, pp. 1–14, Jul. 2021.
[37] H. Chen, Y. Lin, L. Chen, and C. Huang, "Hazard analysis and risk assessment framework for Medical Internet of Things systems," *IEEE Access*, vol. 9, pp. 14517–14531, 2021.
[38] Z. Liu, J. Li, L. Yu, W. Li, and Y. Cui, "A model-based formal method for safety assurance of medical IoT systems," *IEEE Access*, vol. 8, pp. 107234–107245, 2020.
[39] S. Gu, X. Wang, C. Hu, X. Chen, and J. Wu, "A framework for system safety development of medical IoT systems," *IEEE Access*, vol. 8, pp. 129124–129135, 2020.
[40] W. Ma, J. Cao, Z. Jiang, and S. Guo, "A system safety management method of medical IoT devices based on FMEA and fault tree analysis," *Int. J. Environ. Res. Public Health*, vol. 17, no. 24, p. 9349, 2020.
[41] X. Zhou, X. Zheng, J. Chen, and S. Wang, "A safety analysis approach for medical IoT systems," *IEEE Access*, vol. 6, pp. 19729–19739, 2018.
[42] A. Jha, P. Dutta, and J. M. Chatterjee, "A systematic review of safety challenges in Medical Internet of Things (IoT) systems," *IEEE Access*, vol. 9, pp. 29403–29417, 2021.
[43] X. Wang, S. Gu, C. Hu, and J. Wu, "A safety testing approach for medical IoT systems," *IEEE Access*, vol. 8, pp. 96224–96234, 2020.
[44] M. Khan, R. Khalid, S. Anjum, N. Khan, S. Cho, and C. Park, "Tag and IoT based safety hook monitoring for prevention of falls from height," *Autom. Construct.*, vol. 136, Apr. 2022, Art. no. 104153.
[45] J. Zalewski, "IoT safety: State of the art," *IT Prof.*, vol. 21, no. 1, pp. 16–20, 2019.
[46] M. H. Ali, W. K. Al-Azzawi, M. Jaber, S. K. Abd, A. Alkhayyat, and Z. I. Rasool, "Improving coal mine safety with Internet of Things (IoT) based dynamic sensor information control system," *Phys. Chem. Earth, A/B/C*, vol. 128, Dec. 2022, Art. no. 103225.

FRYAD KHALID M. RASHID received the Ph.D. degree in computer science from Clemson University, Clemson, SC, USA. He was doing research with the Strategic Software Engineering Research Group (SSERG), Clemson University. He has used AADL/OSATE for system development in software/system architecture, and software verification and validation. He published some papers in the System Safety Society. His research interests include in developing safety analysis methods for safety-critical systems, the IoT systems, cyber-physical systems, and real-time embedded systems. He is an Official Member of the System Safety Society, USA.

OSMAN SHARIF OSMAN received the B.Sc. degree in geology and the P.G.Dip., M.Sc., M.Phil., and D.Phil. degrees in computer science from The University of Buckingham, U.K. He was a Postdoctoral Research Fellow in U.K., involved in bio-image processing and analysis to assess skin ageing and diseases. He has contributed in the development of novel biological image processing and analysis systems involved for cutaneous researches.



ETHAN TRAVIS MCGEE received the M.S. and Ph.D. degrees from Clemson University, in 2018. After completing his M.S. degree, he joined Bob Jones University (BJU), Greenville, SC, USA, where he is currently an Assistant Professor with the Department of Computer Science. He was with M33 Integrated Solutions/AFS LLC, USA, as a Language and Calculation Engine Developer. He was also a Contractor for several smaller companies in the Greenville area. He started his carrier in academia, and he followed through by joining the University as an Adjunct Professor, in December 2013.

HAIDER RAAD received the M.S. and Ph.D. degrees in systems engineering, specializing in RF telecommunication systems, from the University of Arkansas at Little Rock (UALR), Little Rock, AR, USA, and the M.S. degree in electrical and computer engineering from the New York Institute of Technology. He currently serves as the Director of the Engineering Physics Program and the Wearable Electronics Research Center, Xavier University, Cincinnati, OH, USA. He was previously affiliated with California State University and the University of Arkansas, from 2008 to 2015. He teaches undergraduate and graduate level courses, such as electronic circuits, communication systems, electromagnetics, microcontrollers, and control systems. He is the author of the first textbook on wearable technology and IoT, published by Wiley and IEEE. He also authored several other books on wireless systems and telemedicine, and more than 100 peer-reviewed journal and conference papers on research fields. His research interests include flexible and wearable wireless systems, telemedicine and wireless body area networks, metamaterials, MIMO, and the IoT. He was a recipient of the Center for Population Health Fellowship, the Outstanding Teaching Award, the SSU Research Award, and the AAMI/TEAMS Academic Excellence Award.