FALL / WINTER 2022
VOLUME 14, NUMBER 2

Print ISSN: 2152-4157
Online ISSN: 2152-4165

WWW.IJERI.ORG

International Journal of Engineering Research & Innovation

Editor-in-Chief: Mark Rajai, Ph.D.
California State University Northridge

Published by the
International Association of Journals & Conferences

CUTTING EDGE JOURNAL OF RESEARCH AND INNOVATION IN ENGINEERING

www.ijeri.org

Print ISSN: 2152-4157
Online ISSN: 2152-4165

www.iajc.org

INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH AND INNOVATION

ABOUT IJERI:

- IJERI is the second official journal of the International Association of Journals and Conferences (IAJC).
- IJERI is a high-quality, independent journal steered by a distinguished board of directors and supported by an international review board representing many well-known universities, colleges, and corporations in the U.S. and abroad.
- IJERI has an impact factor of **1.58**, placing it among an elite group of most-cited engineering journals worldwide.

OTHER IAJC JOURNALS:

- The International Journal of Modern Engineering (IJME) For more information visit www.ijme.us
- The Technology Interface International Journal (TIIJ) For more information visit www.tiij.org

IJERI SUBMISSIONS:

- Manuscripts should be sent electronically to the manuscript editor, Dr. Philip Weinsier, at philipw@bgsu.edu.

  For submission guidelines visit www.ijeri.org/submissions

TO JOIN THE REVIEW BOARD:

- Contact the chair of the International Review Board, Dr. Philip Weinsier, at philipw@bgsu.edu.

  For more information visit www.ijeri.org/editorial

INDEXING ORGANIZATIONS:

- IJERI is indexed by numerous agencies. For a complete listing, please visit us at www.ijeri.org.

Contact us:

Mark Rajai, Ph.D.

Editor-in-Chief
California State University-Northridge
College of Engineering and Computer Science
Room: JD 4510
Northridge, CA 91330
Office: (818) 677-5003
Email: mrajai@csun.edu

www.tiij.org

www.ijme.us

INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH AND INNOVATION

The INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH AND INNOVATION (IJERI) is an independent and not-for-profit publication, which aims to provide the engineering community with a resource and forum for scholarly expression and reflection.

IJERI is published twice annually (fall and spring issues) and includes peer-reviewed research articles, editorials, and commentary that contribute to our understanding of the issues, problems, and research associated with engineering and related fields. The journal encourages the submission of manuscripts from private, public, and academic sectors. The views expressed are those of the authors and do not necessarily reflect the opinions of the IJERI editors.

EDITORIAL OFFICE:

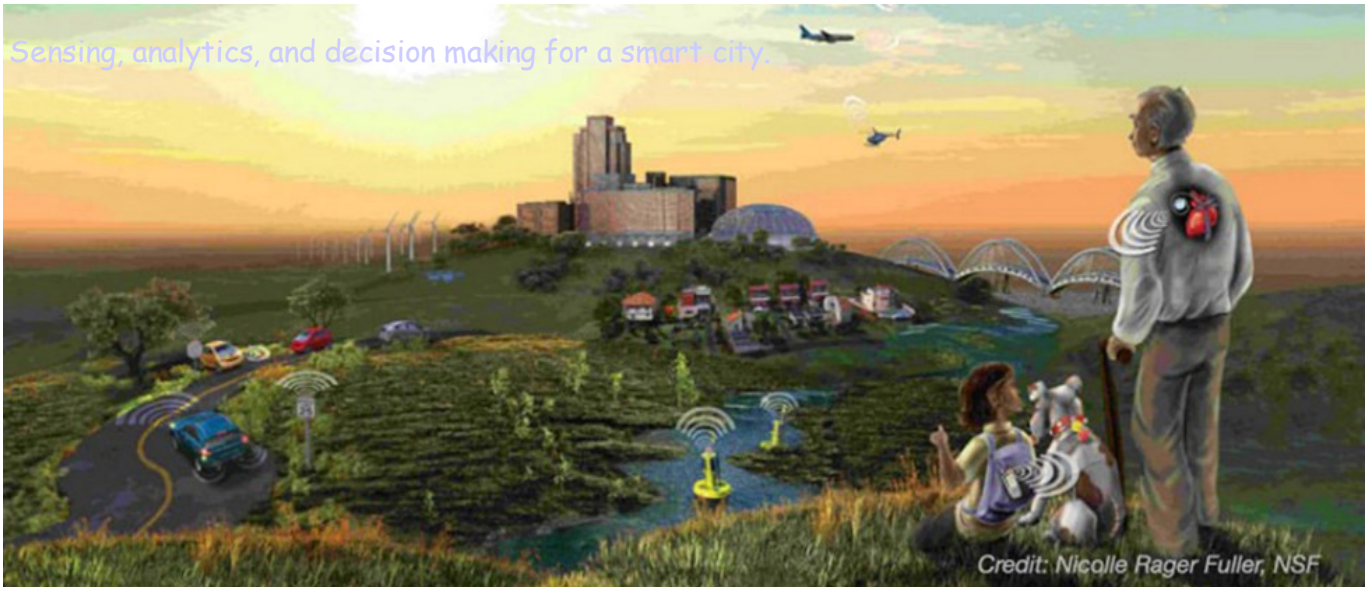
Mark Rajai, Ph.D.
Editor-in-Chief
Office: (818) 677-2167
Email: ijmeeditor@iajc.org
Dept. of Manufacturing Systems Engineering & Management
California State University-Northridge
18111 Nordhoff Street
Northridge, CA 91330-8332

THE INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH AND INNOVATION EDITORS

TABLE OF CONTENTS

In This Issue (p.5)
Security of Cyber-Physical Systems

Philip Weinsier, IJERI Manuscript Editor



If we look at the overarching aspect of the internet, we'll see that it has transformed the way people interact with information. Are our interactions, or searches, via the internet private and secure? Unless you've been living "off the grid," you already know the answer to that. And in that pursuit, countless hours and dollars have been spent trying to make the internet as secure as it can be. Now, though, with the advent of IoT, IIoT, M2M (machine-to-machine), and smart cities, we are realizing the control we have—through the internet—over our physical systems: everything from our microwaves to our alarm systems to our cars to entire smart cities. That is, people can also interact with physical systems via their computers and smartphones. With only the security of the internet to contend with, we needed CYBER security. Adding engineered systems to the mix, we find that we now need cyber PHYSICAL security.

Just as cybersecurity is the state of being protected against the criminal or unauthorized use of electronic data, cyber-physical systems (CPS), often referred to as "smart" systems, are co-engineered with interactive networks that take into account physical systems. CPS combine elements from different scientific theories and engineering disciplines, including cybernetics, embedded systems, distributed control, sensor networks, control theory, and systems engineering. In general, information security threats for CPS can be divided into three layers of threats: physical, information, and application control. If you consider the myriad physical systems we rely on these days, you won't need a leap of faith to understand why they need to be secured from possible hacking: robots, intelligent buildings, implantable medical devices, cars that drive themselves, or planes that automatically fly in a controlled airspace.

In general, there are four physical systems that constitute the essential units of the planet's physical systems: the atmosphere, the biosphere, the hydrosphere, and the lithosphere. But given the state of the art for implantable electronic devices in our bodies, I would offer that there's a fifth physical system. And while there are no doubt countless individuals, companies, and governments currently working to develop cyber-physically secure systems, I also have no doubt that there exist just as many individuals, companies, and governments currently working to break down the security of someone else's CPS. Such individuals—and, I surmise, companies—who previously might at one time have been expert at hacking are oftentimes employed to test the "hardness" of newly developed systems before engaging them.

The U.S. government is investing millions of dollars in the development of CPS. For example, the National Science Foundation (NSF) is working closely with multiple agencies across the federal government, including DHS, S&T, DOT, FHWA, NIH, NIBIB, NCI, NCATS, and NIFA. "Small" projects may receive funding of up to $500,000 (up to 3 years); "medium" projects up to one million dollars; and "frontier" projects between one and seven million dollars (up to 5 years). For more on this subject, please read this issue's feature article on page 2.

Editorial Review Board Members

SECURITY OF CYBER-PHYSICAL SYSTEMS THROUGH DYNAMIC COMPONENT MANAGEMENT

Anton D. Hristozov, Purdue University; Eric T. Matson, Purdue University; Eric Dietz, Purdue University;
Roger Marcus, Purdue University

Abstract

The environment in which cyber-physical systems (CPS) operate can vary in complexity and experiences change constantly. These systems work mostly autonomously, without supervision, and with little or no maintenance. They continuously interact with the environment and with other systems. The expectation is for them to last for long periods, sometimes years or even decades. Their operation is under constantly evolving threats, which present new challenges to designers. The ability to replace software components on the fly can be an effective way to improve software security. Techniques such as redundancy and diversification become feasible, if a general mechanism for software component update is available during run-time.

These goals can become possible if issues such as component interfaces, state, authentication, and dynamic management can be handled practically. In this current study, the authors focused on the feasibility, mechanisms, and limitations of dynamic management of software components to improve security. The focus of the experiments was on software architectures, such as the PX4 autopilot. Still, they can be applied to various CPSs, such as edge devices, smart sensors, and even autonomous robots. The use case explores the approach's feasibility for unmanned aerial vehicles (UAV)s. From this study, the authors present an analysis of the effects of such techniques on the overall mission of the system.

Introduction: CPS Operation in the Presence of Attacks

Like any other cyber system, CPSs are prone to different attacks. Some of these can be from their network connections, and some from attacks on sensors, controllers, and actuators (Wang, Song, Jing, Yang, Guan & Sun, 2019). Another source of disruption can come from hardware or software failures, due to reliability issues or design and implementation mistakes. The software complexity in today's CPSs is so great that it is impossible to guarantee that a system will be implemented with no defects (Henkel et al., 2011). The reality is that a certain number of defects per line of code (LOC) will most likely continue to persist in the future. These sources of disruption need to be addressed independently of their reason. A genuinely resilient system should adapt and continue operating in the presence of attacks, failures, or both. Truly resilient systems can go through the following phases when experiencing disruptions (Fitzgerald, 2016):

- Avoiding the disruption
- Surviving the disruption
- Recovering from the disruption

Any system that can handle resilience must be designed to cope with these three phases. Metrics such as how long a system can avoid disruption and survive it and how long it takes to recover can further quantify the readiness for operation in harsh conditions. Another factor to consider is what kind of attacks can be handled. It is hard to claim the ability to detect and neutralize any possible attack; thus, a different approach is needed to make security possible in practical scenarios. The attacker is always trying to use weaknesses in existing architectures, which becomes more achievable if they use well-known models and operate in a predefined fashion. One way to make things significantly more complex for the attacker from a security point of view is to introduce uncertainty in time and space.

Diversity in time can be achieved by running certain portions of the software at different points instead of constantly. Diversity in space can be achieved by running different software types instead of one application or portions of an application. This approach can make the attacker's job more difficult or downright impossible. With the increased sophistication of attacks and the knowledge that attackers have, this is a promising direction for protecting systems left to work in different and complex environments. Such environments can be in urban and war zones, where potential sophisticated attackers can be abundant.

Class of Systems

The CPS is a reasonably large classification of very diverse types of systems. In this current study, the authors narrowed the scope to systems run on unmanned aerial vehicles (UAV) or unmanned ground vehicles (UGV). A further narrowing of the scope was done to include autopilot systems such as Ardupilot and PX4. The techniques that were explored, though, apply to any other autopilot system and even to the robotic operating system (ROS) (Lauer, Amy, Fabre, Roy, Excoffon & Stoicescu, 2016). These systems have common subsystems and components that are prone to the same vulnerabilities. The vulnerabilities come from the components that have interfaces to the outside world and are essential for the control of robotic vehicles. In this current study, the selection to use the PX4 autopilot software for experiments was driven by several crucial characteristics that this system had; notably:

The PX4 has a modular architecture that can be extended to illustrate approaches needed in this study.

- There is a convenient infrastructure to run experiments through simulators, such as JMavsim and Gazebo. The benefits are that the same code that is run on the UAV is run in simulation, and disturbances in the flight performance can be seen with different types of UAV types used in the simulator.
- There is a possible integration with MAVSDK, which allows for developing mission applications that can control drones through the Mavlink protocol. This allows for complex missions to be run in order to test the autopilot software without the need for real UAV equipment and a flight site.
- The PX4 is a prevalent platform, and any studies that reveal vulnerabilities and countermeasures are of interest to the community of researchers and practitioners in the industry.
- Since this platform has high complexity, any techniques that can be deployed successfully here can be used in other platforms with similar or better success.

The PX4 has modules that are essentially software components, and each one runs in a separate thread or in a work queue (Meier, Honegger & Pollefeys, 2015). Work queues are mechanisms that use a common thread for several modules to share by sequentially running their Run() methods periodically. The work queues present further challenges from the perspective of security, because of the increased complexity and difficulty in manipulating individual modules. Work queues make timing guarantees and predictability even harder and introduce dependencies of modules and their main thread functions. The approach used in this current study for software components and their dynamic management was irrelevant to whether or not they would run in a separate thread, work queue, or even in separate processes, as is often the case in ROS systems (Malavolta, Lewis, Schmerl, Lago & Garlan, 2020).

Figure 1 shows the test setup used in this study. It used a high-level mission-control application that would send the mission commands through the Mavlink protocol. The autopilot software was connected to a simulator, which could be Gazebo or Jmavsim. In addition to the autopilot software, a new system was added to take care of the dynamic management of software components. The logs were used for analysis of the flight characteristics to do a comprehensive analysis of the results.

Literature Review

There are some studies that discussed dynamic system reconfiguration and run-time techniques in general. Many of these studies focused on some of the aspects of implementing dynamic changes, but they often did not comprehensively look at all problems, especially the state, authentication, and timing. In addition, those studies did not typically focus on dynamic behavior with respect to improving security, as was the case for this current study. Dynamic reconfiguration may be done for purposes of maintenance, repair, upgrade, and, most frequently, functional changes. Dynamic reconfiguration, with the main goal of improving security and safety, was the main contribution of this study. Nevertheless, this study builds on what is already available in the literature and the authors propose a different angle of the techniques used and the use cases.

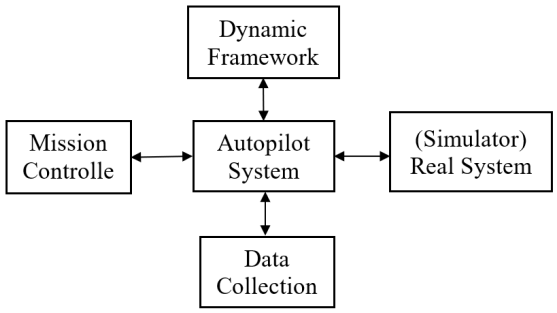


Figure 1. Test setup.

There is a significant number of papers that focused on reconfigurable manufacturing systems (Bortolini, Galizia & Mora, 2018). These papers focused on the challenges related to Industry 4.0 and the ability to quickly reconfigure a manufacturing application or any other large-scale system in order to provide new functionality. These studies considered concepts such as availability and reliability, but security and safety were generally not explored in depth. Some of the studies discussed self-healing and self-management systems (Shin, Cho & Oh, 2018) and the use of machine learning to optimize their operation. The proposed techniques focused on how systems adapt to abnormal situations using self-adaptation techniques.

A general approach to dynamic reconfiguration was discussed by (Saadi, Oussalah, Hammal & Henni, 2018), where the notion of a reconfiguration manager was introduced. This component relies on events that trigger actions, and based on a reconfiguration policy, a new configuration was chosen and implemented. The architecture was represented in the form of a graph. The authors pointed to the challenges of verifying and validating the possible configurations. This opinion was also shared in many other papers, as the possible combinations could make guarantees for architecture stability difficult to provide. An important technique described in the literature was the moving target defense (MTD) technique, which created diversity in the memory space and in the instruction set used in order to create obstacles for the attacker (Potteiger, Zhang & Koutsoukos, 2020). MTD changes properties in the system as it runs so that it can minimize the chances of success of an adversary trying to succeed in devising ways to reverse engineer a working system. This is a form of a proactive approach toward a large number of attacks. The most frequent attacks that the techniques help against are in the category of memory attacks, which can include code injection and code reuse.

Some works proposed a dynamic reconfiguration scheme based on representing the system through graphs. Each node represents a module or component, and each edge is a connection between two components (Pavlenko, Zegzhda & Poltavtseva, 2019). The authors defined routes through the graph, representing a cyber-physical system, and choose different reconfigurations based on information about which node was being compromised by an attack. Choosing a reconfiguration dynamically can maintain the stability of the system. This assumes, though, that there is a mechanism for detecting attacks on any of the nodes. One of the approaches in systems that could do self-adaptation and reconfiguration was that such systems could provide redundancy of important components.

This approach improved fault tolerance and reliability in general. The author suggested that certifying a dynamically reconfigurable system is very hard (Isakovic, 2022). The idea of such reinforced components was used in this current study, too, where the scope was narrowed to individual components in order to make it easier to certify and implement a solution. The majority of the encountered sources dealt with high-level techniques and did not have a case study that implemented a working model that proves that a certain technique can work in practice. As a consequence, they did not focus on a particular class of systems and rarely provided concrete test results. The authors of this current study, however, focused on a specific class of systems, an attack model, and a practical approach to improving security against a large class of attacks. A real-world example of a UAV autopilot was also proposed as well as test data for analysis to help derive relevant conclusions.

Attack Model

In this paper, the authors detail the mechanisms of launching two types of attacks on a running instance of a PX4 autopilot. These attacks can be targeted and can affect specific components of the autopilot software. Attacks can be quite different, but their classification was not the purpose of this paper (Yaacoub, Noura, Salman & Chehab, 2020). The assumptions for the attacks that were made include:

- The attacker can get into the OS where PX4 runs and can gain privileged access.
- The attacker is familiar with the architecture of the running autopilot software, for example, PX4 or Ardupilot.
- The attack will rely on well-established techniques that work in a Linux environment.
- The attack will take advantage of the fact that the software is written in C++ and, therefore, it inherits its vulnerabilities, such as buffer overflow, for example.

These assumptions are relevant to a large percentage of drones used today. Many of these drones are running some version of Linux and autopilot software. In addition, security is not always a deep concern for many UAVs, because they are still considered experimental and standards are still being established. Therefore, widespread attacks such as buffer overflow and /proc file system memory attacks are a real possibility (Nayak, Hibler, Johnson & Eide, 2017).

Attacks can be launched through many different methods. The actual type of attack may use a way to corrupt the memory of the running autopilot process. The authors of this current study used a method of defense that aimed to create a moving target for the attacker so that they would have a very dynamic and much harder environment to attack. Most attackers expect a well-known static environment with a defined memory layout. Figure 2 shows the mechanism of a buffer overflow attack as a popular method for getting to the memory of a running process. The attacker plants a custom socket server that can be contacted later through a client process. It can exploit a buffer overflow attack to get to a component in the autopilot software and compromise it.

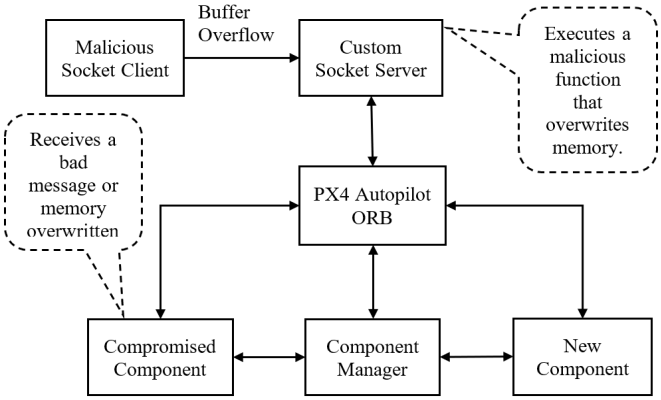


Figure 2. Attack setup.

Dynamic Component Management Approaches

In order to create protection, the choice for the current authors was to replace a working component with another component that had the same functionality. By running each component for a short period of time, the functionality of the autopilot was maintained. The focus was on one of the software components that was central to the autopilot system in order to prove that the approach worked. The technique can be applied to more than one component with some penalty of increased CPU and memory usage. The goal of this experiment was to be able to preserve the functionality and timing, as expected for the normal operation of the UAV.

Dynamic Component Replacement

Component replacement while the system is running with a new component that can continue functioning, at least as

well as the replaced component, is a complex undertaking that requires careful analysis. This technique is an effective countermeasure against the types of attacks described in the previous section. Figure 3 shows a depiction of this operation. This can happen by designing a new component manager module. Component A, as shown in the figure, is a running component that needs to be replaced with component B during run-time. By assuming components A and B have compatible interfaces, they can be switched by unloading one of them and loading the other.

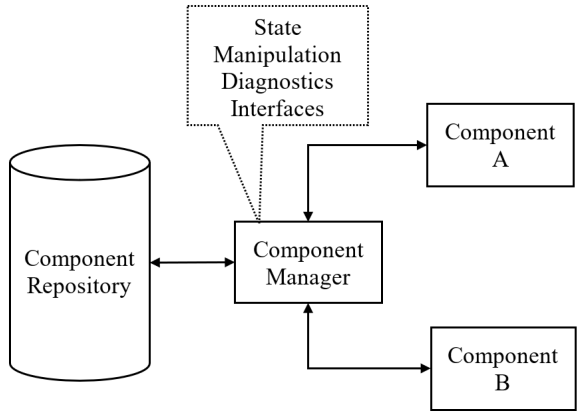


Figure 3. Dynamic component replacement.

The component manager needs to take care of the following concerns:

- Ensure that the interfaces of the new component are compatible with the interfaces of the replaced component.
- Provide authentication so that there is some assurance that the new component is not bogus and is coming from a trustworthy source.
- Make sure that the new component is brought up in a state that matches the state of the replaced component.
- The operation does not have a significant effect on the dynamics of the system.

Figure 4 illustrated the sequence of how a replacement can happen. There is a time when the new component is loaded into memory but does not run right away. This is useful so that the new component can have its state updated after which the old component can be stopped and unloaded; only then does the new component start running in its place. If the system dynamics permit such an operation, the system will not experience any significant effects. This also depends on the component complexity and its state and timing characteristics. Dynamic component replacement can be helpful in situations where there is need for an upgrade, repair of a component, or a switch in the operation of components, due to environmental changes or security requirements. One such situation is related to an ongoing attack on the component, which makes its operation incorrect and dangerous. Another situation is when a component is updated with a newer and improved component that fixes specific vulnerabilities and capabilities of the existing one. In both situations, replacing an existing component has to be done according to an algorithm described in detail later.

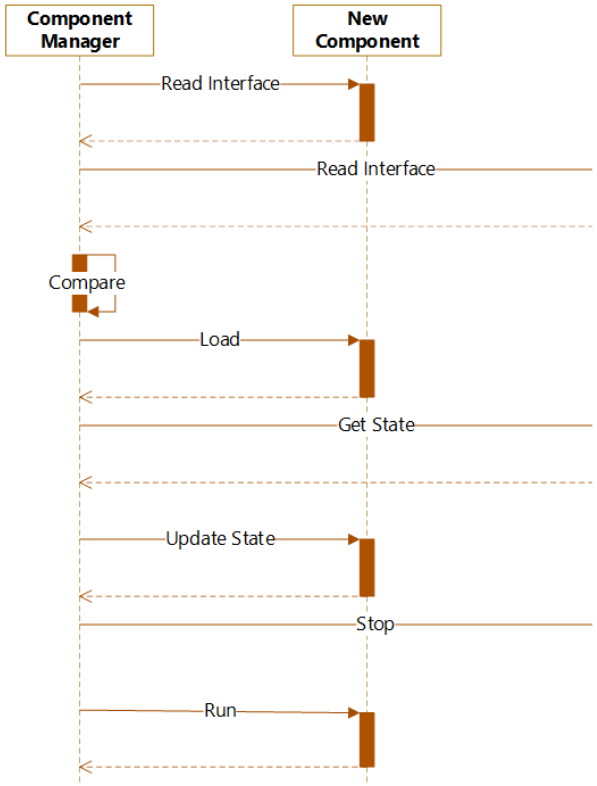


Figure 4. Sequence diagram of component replacement.

Alternating Component Execution at Run-Time

When the number of attacks is large, and the system is executing in real-time, using a proactive approach against potential attacks is better than detecting and remediating in real-time. One such technique assumes that there can be two components with the same functionality but not necessarily the same implementation. If these components are run in separate threads or other execution units, such as processes or tasks, they can be restarted periodically, and their execution can alternate. This guarantees that there is always a running component that can handle the expected functionality. The definitive advantage of the approach followed is that it creates a moving target for the attacker by constantly swapping threads that have different thread identifiers, IDs, and memory footprints, since threads are created and destroyed periodically. The approach takes advantage of the principle of code diversity and time and space randomization. To further improve security, refinements such as randomly changing the names of the threads and the interval of swapping increase the difficulty of a feasible attack.

Ultimately, the alternated threads can have completely different implementations but can perform the same tasks. An example is a traditional PID controller and an LQR controller being swapped repeatedly. This is particularly true for components that have no state; for example, a PID controller. This operation is at the expense of some overhead in time and CPU resources, although the approach can be applied only for the most vulnerable threads of the autopilot software. The diagram of Figure 5 shows the timing of the exchange, where the following time periods are detailed:

T_{load} = time to load a new component
T_{ready} = time to reach a running state
T_{swap} = time to swap the components
T_{unload} = time to unload a component
T_{run} = time to run

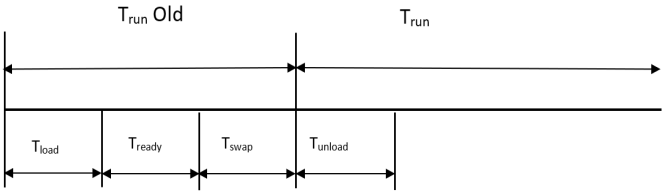


Figure 5. Timeline for component replacement.

Challenges with Dynamic Component Management

There are numerous challenges while implementing a dynamic component replacement. These challenges are based on complexity, security, timing, and dependencies on the specific software architecture. Each component can be represented through the interfaces it supports. Components with the same interfaces can be replaced independently of their implementation, if they comply with the architecture and require certain services to be present. Some of the issues discussed in this section can be implemented during design time. This includes the analysis of component interfaces and the preparation of the components for their run-time authentication. State handling and effects on the architecture are concerns that need to be handled during run-time.

Component Interfaces

The representation of interfaces that can be shared among components throughout the architecture cannot happen without a unique approach. A promising way to address this issue is to use or design a domain specific language (DSL) that can capture the interfaces and which includes the contracts that the component abides by (Holthusen, Quinton, Schaefer, Schlatow & Wegner, 2016). Having a DSL allows the use of automatic tools for integrity and run-time monitoring and performing component rejuvenation and state restoration. The DSL can capture three crucial things that can help in many different directions, when designing software architectures with smart software components:

- The interfaces
- The contracts that are represented by the interfaces
- The state of the component
- The parameters that are used for configuration

Creating a DSL can be done through tools such as Antlr (Stockmann, Laux & Bodden, 2019) or Xtext, or any other tool for the generation of custom parsers. This is going to produce a custom parser of language grammar in general that can also be used to do code generation (Parr & Quong, 1995). A DSL can have high-level constructs that unambiguously define the interface of components. Parsing all component descriptions can produce C++ code that can be used at run-time along with the existing code that components already have. This approach allows for the retrofitting of existing deployments of software components and giving new properties to them. A possible prototype can be developed using the PX4 autopilot software, as it has all the characteristics of a multi-component system with well-defined components (Meier et al., 2015).

A simpler approach to using a DSL is to use a component description language in a structured format such as JSON format. This can arguably be regarded as a form of a DSL, just without the specifics in grammar. The goal of either approach is to express the component's interfaces in a very descriptive and unambiguous way. This could allow comparisons of components and operations such as loading, reloading, destroying, and switching between components to occur easily at run-time. The interface definition of a component associated with the code that uses it can be considered an adapter attached to an existing component for the purposes described thus far.

Authentication

Loading software components dynamically introduces the risk of loading a malicious or tampered component that can jeopardize the regular work of the systems. The first attack based on buffer overflow described earlier uses this vulnerability. Since there is no authentication on what kind of component can be loaded with the existing dyn command in PX4, anyone who has access to the PX4 shell can load an arbitrary module. A module can be loaded even without access to the shell by using the available API that PX4 provides, leading to even more subversive attacks. These issues require a mechanism to be developed that can ensure that components can be trusted, and approaches such as component authentication and attestation come into play. Architectures that use attestation servers and key management have been proposed to handle this (Van, 2017). A simple approach is to use a shared key between the component manager and the build systems that create the components and attach a calculated hash, which is a sequence of bytes to the component that can be verified only by someone who owns the secret key. This approach is simple, and it works, provided the secret key is guarded.

An HMAC approach was presented by Beri and Mishra (2019), where a keyed hash algorithm was used. It allowed for data integrity and proof of origin, which was what was needed in that case. Guarding the shared key is essential to ensuring that there is a secure solution. The technique described in that paper is practical and may be used as a basis for its implementation. The overhead of checking the HMAC also needs to be considered, but since it is something that is done only one time while the component is loaded, it is practical. An adequately long HMAC, such as SHA512, can provide better security at the expense of some computational overhead. This may not be a big hurdle, though, since the HMAC will need to be calculated once the component is compiled and then recalculated again just before a decision is made if the component is authentic and can be loaded and run.

Timing Considerations

Effects on timing are another concern when replacing components (Knight & Strunk, 2004) during run-time. Some components can take a long time to reach a steady state, since they need to collect data based on sensor measurements. One such component is the extended kalman filter (EKF) used in various autopilots and other control systems. Such components can affect the overall stability of the mission, and their timing and inertia need to be considered. Another valuable consideration that needs to be mentioned is how long it takes for a component to be restarted and the effect on the system's overall stability. This includes the time for loading and initializing a component and updating its state. If a new component is loaded that replaces a compromised or vulnerable and obsolete component, there is a need to consider the time when the new component will be ready to take over. Figure 5 shows these timing dependencies.

The components that were used in this current study were an attitude controller and a position controller. Another interesting part of the system was the GPS coordinate handling. This was important, since many attacks are launched through GPS spoofing and their effects can lead to controlling the vehicle's mission. The EKF is also of interest, as its importance in the system is very high. It presents a real challenge for dynamic component management, because of the complexity described above and its six instances in a running PX4 autopilot process. Therefore, the focus was on the attitude and position controller's components, leaving the EKF observer for future explorations.

State Handling

Handling the state in a system is a complex task, since the system state is general, while the local state is specific to individual components. The state can be modified at system initialization and during run-time (Kapova, Buhnova, Martens, Happe & Reussner, 2010). There is a connection between the internal state of each component and the overall state and behavior of the system. Expressing the internal state of a component can help make this connection. Many existing systems use static methods for state evaluation, although the state changes and can be different at different times. This happens because many components change their internal state as the system continues to operate (Lauer, Amy, Fabre, Roy, Excoffon & Stoicescu, 2018). Some have very complex states and some have no state or a very minimal state. Recovering this internal state of a component that has been replaced or refreshed is a major challenge, due to the fact that the effects this state can have on the system can be detrimental. Representing the state in a universal way and providing facilities, part of the component, so that it can be recovered is possible, but it is far from trivial. The difficulty comes from the fact that the state of a component can be constantly changing and the change can be relatively fast or gradual. In addition, components can have multiple parallel threads of execution, making things even more complicated. This can mean that the state can be distributed among threads that are part of one component.

Software components retain states in variables with different types; in many cases, they are complex objects and structures. In order to be able to save the state of each variable independently of its type, a technique called serialization needs to be used (Grochowski, Breiter & Nowak, 2019). This technique takes the variables of an object and converts them to bits that can be stored on a disk. Deserialization is the opposite and converts the bits from the disk to values that can populate the variable of an object. Some languages support serialization, but C++ does not appear to do so in the standard libraries; serialization, though, can be accomplished through additional libraries. This is relevant since the typical autopilot software is written in C or C++. Thankfully, some useful libraries are appropriate for tackling the task of state management of objects that belong to components.

Effects on the System

In this current study, the authors' approach was to compare the performance of the original system and a modified system that experienced dynamic component updates while the system was running through a complex mission. Dynamic component updates can affect the system's operation, so a feasibility study needed to be performed. The use case section attempted to determine if this approach was practical for a modern software architecture like the one used for the PX4 autopilot. Figure 1 shows the setup used for this assessment. Dynamic component replacement would be more accessible when a system is executing a simple mission or, in the case of a UAV, is simply hovering. Many works consider simple cases, such as hovering operations that are not practical scenarios for real UAV systems (Arroyo, Tarek, Kobayashi, Yang & Sethumadhavan, 2019). Some factors worth mentioning that determine the feasibility of the dynamic management approach include:

- The complexity of the component, which includes state and timing dependencies.
- The moment in the mission when the dynamic operation occurs.
- The speed of the CPU, memory, and overall hardware of the UAV.
- The software architecture of the autopilot.

The Concept of Root of Trust and Component Management

Figure 3 shows the vision that there is a specialized module called a component manager, which is the central component that allows for any dynamic scheme to be put in place. The most straightforward approach is to implement it in user space, as was done for the prototyping in this current study. However, a more thoughtful approach was needed. The component manager needed to be protected from potential attacks, and therefore it needed to be in a different place where the root of trust was. Candidates for this were the kernel space of the OS that was used or, even better, in a hypervisor that could control the software components from a secure implementation. Both approaches are discussed next as a potential for future experiments.

Figure 6 shows a general approach to the software application running in user space, as is the usual case for most systems. The kernel space implementation of the component manager can be as simple as dynamically swapping components to maintaining a graph of component interactions and refreshing different components when they do not operate. This allows for the smarts about component dependencies and their state to be kept in the kernel space. This guarantees survivable components in user space, because of the dynamic control from the kernel space.

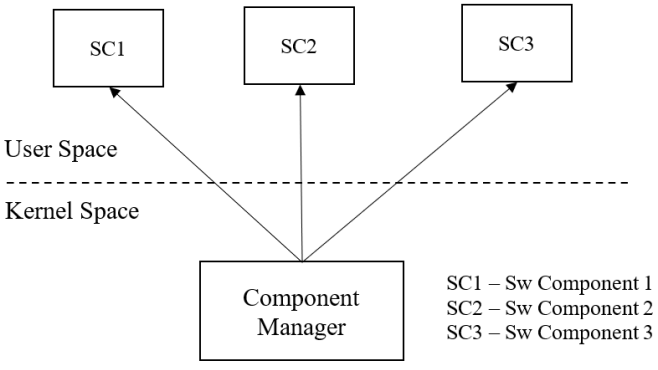


Figure 6. Kernel and user space implementation.

Kernel Space Implementation

The component manager can be implemented as a kernel module that controls the user space components from kernel space. This approach protects the component manager and the dynamic component management scheme from attackers who have access only to the user space. Access to kernel space requires super-user privileges and is harder to penetrate for most attackers. There is some complexity in implementing the component management in kernel space and the interaction with the user space, but it is still possible in a rich environment such as Linux. In addition, it is uncommon to have a kernel module that controls user-space threads and this approach is somewhat unfamiliar to traditional architectures known to attackers.

Hypervisor Implementation

Another robust approach is to use an implementation in a hypervisor so that the execution of the component manager is at an even higher level of protection (Vasudevan, Chaki, Maniatis, Jia & Datta, 2016). Hypervisors can run at the highest privileged level, and their code cannot be accessed from the OS, even from its kernel space. This approach also has the advantage that a solution can be part of a well-designed hypervisor's verified and minimal codebase. It is a more complicated way to implement the solution but it protects against attacks, since the component manager cannot be compromised. Thus, it can continue to manage the user space and keep it resilient through a dynamic refresh of components in the presence of persistent attacks.

Use Case: PX4 Autopilot and Dynamic Component Management

The experiments in this current study were based on running a PX4 in a simulation environment with Jmavsim or Gazebo. Jmavsim was included with the distribution of PX4 and is written in Java. It is a simpler simulator that allows modifications, although it is not a physical simulator. Gazebo is a physical simulator that can be extended through plugins and provides a flexible architecture, including the possibility for creating simulation worlds. Both were adequate for the current experiments. MAVSDK was used as a tool that could generate complex missions for the simulated UAV to execute. MAVSDK is flexible and communicates through the Mavlink protocol in order to send and receive commands from the vehicle. During the flight dynamic component replacement and alternative component execution were performed. In the first case, the process was to go through the replacement of a running component, as depicted in Figure 4. The authors' objective was not to disturb the mission too much and to be able to complete it while implementing the dynamic scheme. A quantitative analysis of how much the flight parameters had been affected was done based on an analysis of the flight logs.

Authentication Implementation

Authentication is a two-step process, where the component that needs to be authenticated is prepared during compilation and verified at a time before it is run. A strong algorithm is needed to ensure that it cannot be easily broken.

The technique that is both practical and secure is HMAC. The build system uses a shared key, and the same key is used during run-time. Key generation and storage are other critical concerns. Key length and the HMAC algorithm itself determine the strength of the solution. SHA512 can be used to generate an HMAC on the contents of the module and use the same key to compare the generated and calculated HMAC before the module is loaded. The computational overhead is not a significant concern, since this is done only once at run-time and once during compilation.

For testing during this current study, the cryptocpp library was used, which has a good arsenal of algorithms for cryptography. Cryptocpp was selected, since it is open source and has different algorithms that can be called from the C++ code in the PX4 autopilot. The selected algorithm was SHA512, based on its safety. It is rather straightforward to add a post-processing task during compilation and create an HMAC. When the module is loaded, the software follows the same procedure of calculating an HMAC on the contents of the module and loading it only if the HMAC is the same. This is constantly emphasized with a shared key approach, although the techniques for accomplishing this are well outside the scope of this current study. Using a shared key, 32 bytes in length, was the approach that was selected, although the handling and storage of the shared key were not part of the study.

State Recovery

Each software component in the PX4 typically has a main class with class variables that are used to preserve the state during the operation of the component. For state saving and recovery of a running component, the bitsery C++ library was used, which allows for serialization of variables with different and complex types. Bitsery allows for variables with arbitrary types to be effectively serialized to disk or RAM and deserialized when needed. This technique allows for the state of one component to be saved to disk or memory and then transferred to a newly loaded component. Tests from this current study allowed the authors to conclude that this was feasible for the attitude and position controller components. This proved that the approach worked, both from a timing and complexity perspective in a real-world scenario.

Dynamic Component Management

The component manager was the central piece of the experiments. It allowed for loading and unloading components at different rates. The pxh module in the PX4 allowed for easier control through a programmatic interface. An improvement of the existing dynamic loading mechanism was made in order to add authentication capabilities and interface checking before the component could be used. The component manager was implemented as a PX4 application thread running in user space but, as discussed previously, future implementation can move it to kernel space.

Test Results

Several tests were performed to estimate the feasibility of such a dynamic scheme with continuous component updates. There were two major classes of experiments. The first class of experiments was to replace an existing component dynamically just once. This included handling authentication and state recovery before the new component was run. The purpose of this test setup was to prove that a component could successfully be reloaded while the system operated and could still provide smooth operation. The experiments showed that this is possible, and the effects of authentication and state handling did not perturb the mission, as well as swapping components in mid-flight. The illustrated scenario was to have a component under attack and, after replacing it with a new component in order to erase the effects of the attack, resume normal operation. The previously described attack methods were used to launch an attack and affect the operation of the original component.

The choice was to perform an attack on the original attitude controller that was part of the PX4. This controller subscribes to several topics and calculates the vehicle rates and the attitude setpoint based on the data it receives from the messages. The attack, in this study, used the /proc file system method to overwrite these variables constantly as the attitude controller worked. The assumption was that the attacker was familiar with the offsets for these variables in the component's memory. Once the new instance of the attitude controller was loaded, the compromised component was unloaded from memory and the effects of the attack were erased.

Initially, both experiments were done for a hovering drone to prove that the solution was working. Finally, a full mission application was run while alternating modules and continuously replacing components dynamically. The mission was successful. The tested dynamic schemes were able to effectively counter an isolated attack against the chosen vulnerable and crucial attitude controller component. The same approach could be implemented for the rate and position controller, the EKF, and any other component that might be under attack and is crucial to the safety of the mission. The UAV mission was preserved in all cases with normal, alternating, and dynamic component replacement. Figure 7 (compared to Figure 8) shows that the most significant deviation in the mission was seen during the continuous alternation of the position controller.

The authentication of a dynamic component was the most CPU-intensive and slowed operation. The time needed to calculate and HMAC was measured with the SHA512 algorithm having a 32-byte long shared key and a compiled component with a length of a little more than 6MB. The time was equal to 26.85 milliseconds on a modern laptop. Since this was done only once, the time was adequate for these experiments and for most CPSs in use today.

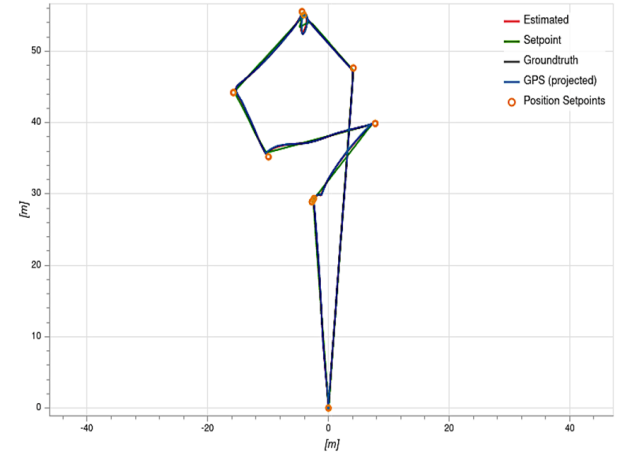


Figure 7. Original mission plot.

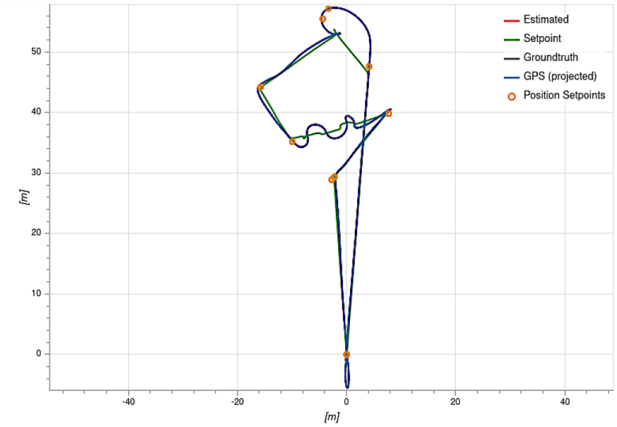


Figure 8. Alternating mission plot.

key:5465737448616 36B31323334353637384861636B546573743837363534333231
plain text length 6066896
hmac:7DCC4B68C65478C60CF19C31B61F535B2D9EB26384539FC5172866F012183 1096D828AFFF28560A9656BEBE2953D1279E77FEEFED155EC3C34352263985B54CF
Time difference = 25986[µs] + 25986585[ns]

This experiment did not require a state update, since the components were given enough time to come up to speed before switching, and the attitude controller did not have a very complex state. The time to save the state for the mc_att_control component was an experiment for completeness so that one could have an idea in case the method for state recovery needed to be used in future experiments. The time to save the state was much longer than the time to restore it; still, though, at 97 microseconds, this is negligible and allows for implementations even in restricted CPU environments.

Time to save state = 97[µs] and 97346[ns]
Time to restore state = 8[µs] and 8966[ns]

The second class of experiments was more ambitious and included having two components with identical or similar functionality, alternatively being in control. A second version of the position controller was created during this scenario and had the original and the alternative position controller change as frequently as two times per second. Each component's instance ran for that period, then it was destroyed and a new instance spawned in its place, ready to run. Figure 5 shows that, during this time, the new controller ran and then was unloaded, allowing the cycle continues. This scenario did not perform authentication, since the two controllers were loaded at the initialization of the software. There was state recovery, since the position controller component had a PID loop and other flight-related state variables. While changing the instances in control were changed, the mission was completed successfully.

This test explored the elements of software diversity and redundancy as methods for improving reliability and security. The goal was to prevent an attack in the first place. Alternating two threads, and performing the same task when possible while their memory footprint changed dynamically, was one strategy for increasing the difficulty of devising a successful attack. Since it is not hard to introduce randomness in the time that each thread runs, this can further make the solution more resistant. The tests with the alternate implementation of the attitude controller showed no change or degradation in the flight quality, since there was no significant state that the attitude controller contained.

Alternating the controller was possible, although some effects were due to timing and state recovery. Overall, the mission was successful, though the authors noticed that the flight time had increased from 2 minutes and 20 seconds to 3 minutes and 20 seconds. The mission trajectory showed some deviations, although not very significant ones. All flight parameters were affected compared to the pristine case, where there were no controller changes during the flight, but the approach was feasible for a relatively complex and dynamic mission which was completed successfully during the simulation.

The results were analyzed through the flight review tool as part of the PX4 ecosystem. Figures 9 and 10 show that the local position deviated slightly from the estimated position during the test, where the position controller was swapped continuously. Figures 11 and 12 indicate that the actuator outputs also showed larger swings during alternative execution. There was also some increase in the vibrations of the UAV. Finally, Figures 13 and 14 show that there was a slight increase in CPU and memory usage. These results were expected as an aggressive restart of alternate versions of the position controller was performed, which was one of the most important controllers in the PX4. Overall, the results were very promising, because the mission was completed with some precision and time penalties, but the important result was that it finished successfully.

This result was pertinent to safety-critical missions, where safety is more important than efficiency. This is particularly true for military operations, where the loss of life and equipment is more important than time to complete the mission or some other mission criteria.

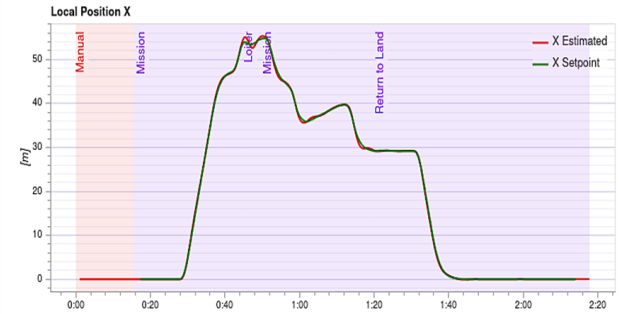


Figure 9. Local X position normal.

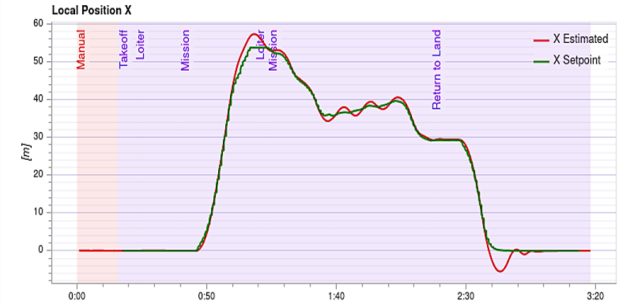


Figure 10. Local X position alternate.

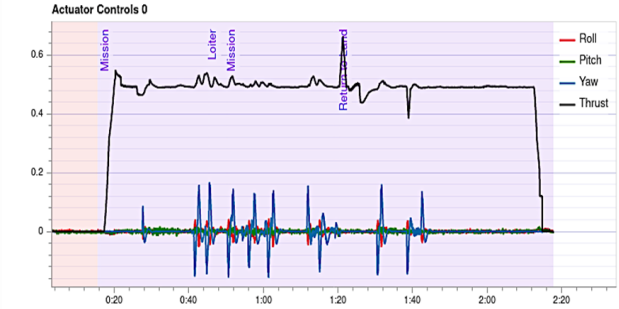


Figure 11. Actuator outputs normal.

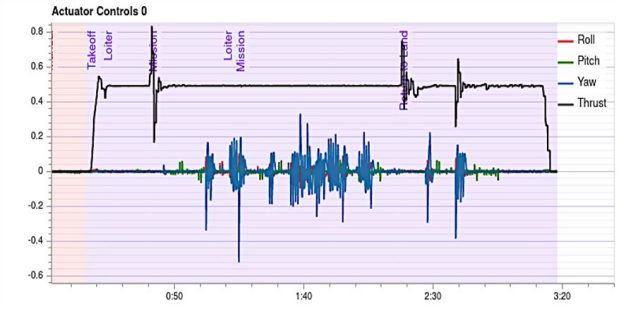


Figure 12. Actuator outputs alternate.

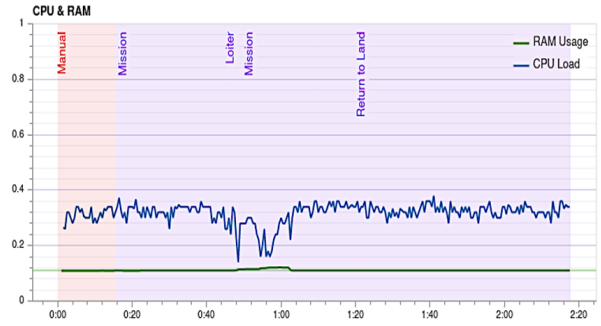


Figure 13. CPU and RAM normal run.

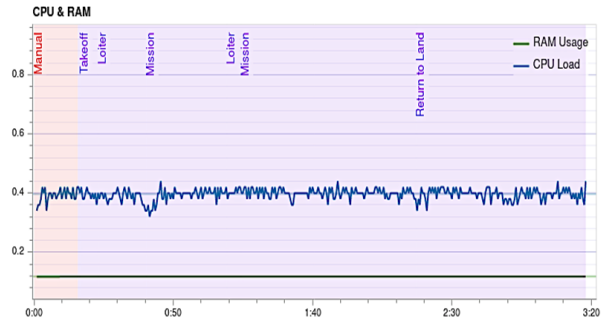


Figure 14. CPU and RAM alternate run.

Conclusions

Through this current study, the authors aimed to explore the possibility of improving the security of existing systems by retrofitting them with schemes allowing dynamic component management. For the study, modern software architectures, such as the PX4, were explored, although the results are applicable to other similar systems. The running behavior of the modified autopilot with dynamic component management proved that such an approach is not only possible but holds much promise for the ever-increasing threats in the quickly changing world of deployed UAVs. The results from these experiments showed minimal effects on mission execution and remarkable resilience against a specific class of widespread attacks. This happened independently of the fact that alternate versions of the position controller were continuously reloaded. Further studies will be needed to continue exploring the possibilities in this study. Hopefully, these results and recommendations will affect architectural decisions of current and future systems regarding their resistance to malicious attacks and failures.

References

Arroyo, M., Tarek, M., Kobayashi, H., Yang, J., & Sethumadhavan, S. (2019). *YOLO: frequently resetting cyber-physical systems for security* (Vol. 11009). International Society for Optics and Photonics SPIE. https://doi.org/https://doi.org/10.1117/12.2518909

Beri, P. S., & Mishra, A. (2019). Dynamic Software Component Authentication for Autonomous Systems using Slack space. Third International Conference on Trends in Electronics and Informatics (ICOEI).

Bortolini, M., Galizia, F. G., & Mora, C. (2018). Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems*, *49*, 93-106. https://doi.org/https://doi.org/10.1016/j.jmsy.2018.09.005

Fitzgerald, J. S., & Larsen, J. I. (2016). Resilience Profiling in the Model-Based Design of Cyber-Physical Systems. N. Plat, & Battle, N., *The 14th Overture Workshop: Towards Analytical Tool Chains*, Cyprus, Greece

Grochowski, K., Breiter, M., & Nowak, R. M. (2019). Serialization in Object-Oriented Programming Languages. S. Keshav, E. Pakize, & K. Seifedine (Eds.), *Introduction to Data Science and Machine Learning* (pp. Ch. 12). IntechOpen. https://doi.org/10.5772/intechopen.86917

Henkel, J., Bauer, L., Becker, J., Bringmann, O., Brinkschulte, U., Chakraborty, S. . . . Wunderlich, H. J. (2011). Design and architectures for dependable embedded systems. 2011 Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS).

Holthusen, S., Quinton, S., Schaefer, I., Schlatow, J., & Wegner, M. (2016). Using Multi-Viewpoint Contracts for Negotiation of Embedded Software Updates. *Electronic Proceedings in Theoretical Computer Science*, *208*, 31-45. https://doi.org/10.4204/EPTCS.208.3

Isakovic, H. (2022). Towards Dependable CPS/IoT Ecosystem. (Doctoral dissertation, Technische Universitat Wien). *reposiTUm* (p.155).

Kapova, L., Buhnova, B., Martens, A., Happe, J., & Reussner, R. (2010). *State dependence in performance evaluation of component-based software systems* Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering, San Jose, California, USA. https://doi.org/10.1145/1712605.1712613

Knight, J. C., & Strunk, E. A. (2004). *Achieving Critical System Survivability Through Software Architectures. Architecting Dependable Systems II*. Berlin, Heidelberg: Springer

Lauer, M., Amy, M., Fabre, J., Roy, M., Excoffon, W., & Stoicescu, M. (2016). Engineering Adaptive Fault-Tolerance Mechanisms for Resilient Computing on ROS. 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE).

Lauer, M., Amy, M., Fabre, J. C., Roy, M., Excoffon, W., & Stoicescu, M. (2018). Resilient computing on ROS using adaptive fault tolerance. *Journal of Software: Evolution and Process*, *30*(3), e1917.

Malavolta, I., Lewis, G., Schmerl, B., Lago, P., & Garlan, D. (2020). How do you Architect your Robots? State of the Practice and Guidelines for ROS-based Systems. 2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).

Meier, L., Honegger, D., & Pollefeys, M. (2015). PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. 2015 IEEE International Conference on Robotics and Automation (ICRA).

Nayak, P., Hibler, M., Johnson, D., & Eide, E. (2017). A Wingman for Virtual Appliances. International Conference on Runtime Verification, Seattle, WA, USA.

Parr, T. J., & Quong, R. W. (1995). ANTLR: a predicated-LL(k) parser generator. *Software. Practice and Experience*, *25*(7), 789-810. https://doi.org/10.1002/spe.4380250705

Pavlenko, E., Zegzhda, D., & Poltavtseva, M. (2019). Ensuring the sustainability of cyberphysical systems based on dynamic reconfiguration. 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS).

Potteiger, B., Zhang, Z., & Koutsoukos, X. (2020). Integrated moving target defense and control reconfiguration for securing Cyber-Physical systems. *Microprocessors and Microsystems*, *73*, 102954. https://doi.org/https://doi.org/10.1016/j.micpro.2019.102954

Saadi, A., Oussalah, M. C., Hammal, Y., & Henni, A. (2018). An approach for the dynamic reconfiguration of software architecture. 2018 International Conference on Applied Smart Systems (ICASS).

Shin, H.-J., Cho, K.-W., & Oh, C.-H. (2018). SVM-Based Dynamic Reconfiguration CPS for Manufacturing System in Industry 4.0. *Wireless Communications and Mobile Computing*, *2018*, 5795037. https://doi.org/10.1155/2018/5795037

Stockmann, L., Laux, S., & Bodden, E. (2019). Architectural Runtime Verification. 2019 IEEE International Conference on Software Architecture Companion (ICSA-C).

Van, J. (2017). VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks. *ACSAC '17* Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL .

Vasudevan, A., Chaki, S., Maniatis, P., Jia, L., & Datta, A. (2016). {überSpark}: Enforcing Verifiable Object Abstractions for Automated Compositional Security Analysis of a Hypervisor. 25th USENIX Security Symposium (USENIX Security 16).

Wang, R., Song, H., Jing, Y., Yang, K., Guan, Y., & Sun, J. (2019). A Sensor Attack Detection Method in Intelligent Vehicle with Multiple Sensors. 2019 IEEE International Conference on Industrial Internet (ICII).

Yaacoub, J.-P., Noura, H., Salman, O., & Chehab, A. (2020). Security analysis of drones systems: Attacks, limitations, and recommendations. *Internet of Things*, *11*, 100218. https://doi.org/https://doi.org/10.1016/j.iot.2020.100218

Biographies

ANTON D. HRISTOZOV graduated from the University of Pittsburgh with a Masters in Telecommunications and Information Science. He is currently pursuing a doctoral degree in technology at Purdue University, while working as a research engineer. His research interests include embedded systems, Internet of Things, smart sensors, and real-time systems. He is a Linux enthusiast and enjoys working with cyber physical systems, which use sensors and convert energy in different forms. Smart energy and the use of artificial intelligence is a new and interesting direction, given the recent advances in use of machine learning in cyber physical systems. Mr. Hristozov may be reached at ahristoz@purdue.edu

ERIC T. MATSON is a professor and University Faculty Scholar in the Department of Computer and Information Technology at Purdue University, West Lafayette. He is the Director of the of the Korean Software Square at Purdue and the co-founder of the M2M Lab at Purdue University, which performs research in the areas of multi-agent systems, cooperative robotics, and wireless communication. The application areas are focused on safety and security robotics systems. His research has generated high levels of external funding, approximately 190 published, refereed articles, numerous MS and PhD students, and practical, demonstrable results for a number of sponsors. Prof. Matson has a PhD in Computer Science and Engineering from the University of Cincinnati, MBA in Operations Management from Ohio State University and BS and MSE degrees in Computer Science from Kansas State University. Dr. Matson may be reached at ematson@purdue.edu

J. ERIC DIETZ is the Director of the Purdue Homeland Security Institute and Professor of Computer and Information Technology. Eric also directs the Purdue Military Research Institute, which seeks to support military officers research who will maintain a technology superiority in national defense and security programs. Dr. Dietz's research interests include measurement and modeling of cyberphysical security systems, energy security, and heath response to disaster. Eric earned his BS and MS from Rose-Hulman Institute of Technology and PhD in Chemical Engineering from Purdue University. Dr. Dietz may be reached at jedietz@purdue.edu

MARCUS ROGERS is a professor and Assistant Dean for Cybersecurity Initiatives at Purdue Polytechnic. He earned his BA, MA and PhD in Forensic Psychology from the University of Manitoba. Dr. Rogers interests include digital forensic analysis, applied behavioral analysis, and digital evidence. Dr. Rogers may be reached at rogersmk@purdue.edu

STUDYING THE EFFECTS OF GEOGRIDS ON FLEXIBLE PAVEMENTS USING LARGE-SCALE LABORATORY MODELLING

Tamer M. Breakah, Ball State University; Maram Saudy, The American University in Cairo;
Safwan Khedr, The American University in Cairo; Omar Elkadi, The American University in Cairo;
Mai Ahmed, The American University in Cairo; Mennatallah Abdelhamid, The American University in Cairo;
Sarah Hussain, The American University in Cairo

Abstract

Geogrids are used in the reinforcement of unbound layers. One application of geogrids is in the reinforcement of flexible pavement. In this study, the authors prepared a large-scale tank to study the change in performance of a full pavement structure when geogrids were incorporated. The geogrid mesh was placed at the interface between the base and asphalt concrete layers. The tank dimensions were designed such that its boundaries would not interfere with the soil stresses. Dynamic loading was applied on the samples and the stresses in the subgrade and base layers were measured using preinstalled pressure gauges. A statistical analysis was done to compare the performance of both conditions. The results showed that rutting in the case of the unreinforced sample was significantly higher than for the reinforced sample. The presence of the geogrids also reduced the stresses induced in the subgrade and base layers at all measured points, except under the point of load application where stress increased.

Introduction

Rutting in asphalt pavements is a major concern, especially in countries with hot climates. Rutting is caused by deformation in the pavement layers and/or the subgrade. Geogrids are used to help reduce rutting in the pavements. Geogrids are polymers made as tensile ribs that are connected in parallel with openings between them through which the aggregate particles can penetrate causing a stronger interlocking effect. They are classified as a type of geosynthetic reinforcement system (Abu-Farsakh, Hanandeh, Mohammad & Chen, 2016). Geogrids can decrease the rutting potential of the asphalt concrete layers in flexible pavements (Gu, Luo, Luo, Lytton, Hajj & Siddharthan, 2016). This may be attributed to the mechanism of the geogrids' interaction with the pavement layers (Moghaddas-Nejad & Small, 1999). Geogrids help in spreading the load over a larger area, thereby reducing the stresses in the soil. They also reduce stresses by absorbing shear stress (Zornberg, 2017). Geogrids can be manufactured with integral junctions, which themselves are manufactured by extruding and orienting sheets of polyolefins (polyethylene or polypropylene). These types of geogrids are often called extruded or integral geogrids. Geogrids may also be manufactured from multifilament polyester yarns that are joined at the crossover points using a knitting or weaving process and then encased in a polymer-based, plasticized coating.

These types of geogrids are often called woven or flexible geogrids. A third type, a welded geogrid, as the name implies, is manufactured by welding polymeric strips together at their crossover points. All of these manufacturing techniques allow geogrids to be oriented such that the principal strength is in one direction, called uniaxial geogrids, or in both directions (but not necessarily the same), called biaxial geogrids. Geogrid reinforcement is utilized as a part of flexible paved roadways in two noteworthy application zones: base reinforcement and subgrade stabilization. In base reinforcement applications, the geogrids are placed within or at the bottom or top of unbound layers of a flexible pavement system; this improves the load-carrying capacity of the pavement under increased traffic. The use of geogrids has two main benefits for flexible paved roads:

- Reduction of the base course and subbase thicknesses and fewer asphalt layers.
- Improvement in road performance by increasing road serviceability and reliability.

Although the use of geogrids in flexible pavement has been studied over the last two decades, there is still limited data showing the optimum location for the use of reinforcement and the effect on stress distribution within the layers. There is, nonetheless, a consensus that the use of geogrids enhances the performance of flexible pavements. The use of geogrid reinforcement is also one of the most cost-effective ways to strengthen the structural capacity of the pavement and extend its service life (Abu-Farsakh et. al, 2016; Ibrahim, El-Badawy, Ibrahim, Gabr & Azam, 2017). When compared to pavement without geogrid, pavement with geogrid has a significantly longer life. Experimental results revealed that integrating geogrid into the asphalt pavement layer greatly reduces rutting and, hence, contributes to pavement service-life extension. (Susanto, Yang & Duc, 2021). In a study using triaxial and biaxial geogrids, the use of geogrids reduced the maximum deformation from one inch in the nonreinforced case to 0.25 and 0.5 inches in the triaxial and biaxial geogrid cases, respectively (Alimohammadi, Schaefer, Zheng, White & Zheng, 2021). A major aspect that is analyzed in the use of geogrids in flexible pavements is the location of the geogrid placement.

The importance of the position of the geogrids in the cross section comes from its effect on the stresses developed in the road's cross section (Mittal & Shukla, 2020). In addition, it is important to identify the effect of geogrids on the stresses created within the soil. The way geogrid reinforcement works is that the aggregate particles penetrate through the geogrids' openings during the compaction process resulting in the creation of a strong interlock and, therefore, drastically reducing the lateral movements of the unbound base material or mixed particles. This interlock increases resistance to rutting development such that stresses are transferred by tensile forces leading to an enhancement in pavement performance (Abu-Farsakh & Chen, 2011). Haas, Walls, and Carroll (1988) studied the effect of geogrid reinforcement mechanisms in flexible pavements when applied in the granular base layer.

The study included an analysis of the stresses, strains, and deflections within the layers. The authors of that study concluded that the use of geogrids has benefits in the reduction of rutting by altering the distribution of the stresses within the pavement. The appropriate design and selection of the location of the geogrids are of high importance. The recommended location for the geogrids, in the case of a thin base, is at the interface between the base and subgrade. With thicker bases, the authors recommended the middle portion of the base as the optimum location for the geogrid reinforcement. The authors noted that there is no benefit in using the geogrids in a compression zone. Reyes and Kohler (2006) used an accelerated testing scheme to study the effect of geogrid reinforcement location. The study was composed of four sections: a control section and three sections with geogrids at different locations.

The results showed that failure was due to fatigue in the asphalt concrete layer and that the section with the best performance was the section that had the geogrids placed within the granular base. The authors of that study also concluded that, if two layers of geogrids were used, the locations should be on top of the subgrade and in the base. In another study (Ibrahim et al., 2017), the authors investigated the use of geogrid reinforcement in flexible pavement applications. In that study, the authors used geogrids to develop five testing sections in the laboratory. The steel container used for the large-scale model was 1.0m in length, 0.35m in width, and 0.55m in height. The layers in the model were composed of a 30 cm thick clay subgrade, a 15 cm granular base layer, and a 5 cm flexible pavement layer.

The variable in that study was the location of the geogrids and the loading, which was done using a static plate. The results showed that the optimum location for the reinforcement was directly below the asphalt layer, while the tensile strain at the bottom of the AC layer was reduced to its lowest level, resulting in the longest fatigue life. The second recommended location for geogrid placement was at 33-50 percent of the base layer's height from the bottom.

In other studies, a 3D finite element analysis showed that the use of geogrids in unbound layers reduced the longitudinal and transverse deformations and that the use of a single geogrid mesh at the upper third of the layer improved the performance (Jasim, Fattah, Al-Saadi & Abbas, 2020). The authors of that study concluded that the addition of a geogrid mesh at the interface between the subgrade and the base could improve structural stability. Zornberg, Gupta, Prozzi, and Goehl (2008) used geogrid reinforcement in projects with highly expansive soils and showed that the in-service performance resulted in the reduction of problems associated with expansive soils when using geogrid reinforcement.

Pereira, Pitanga, da Silva, E Almeida, and Lunz (2021) found that the insertion of the geogrid in contact with the asphalt layer provided better interlocking, reduced deformation, and enhanced mechanical strength and stiffness. This was due to the adhesion between the geogrids and the hot mix asphalt. Moayedi, Kazemian, Prasad, and Huat (2009) found that moving the geogrid closer to the top of the pavement increased tensile stress absorption. The highest values were obtained when the geogrids were placed between the asphalt and base layers. In a study by Mattar et al. (2022), the authors placed the geogrids in the treated base course in varying locations and found that the optimum location is at the middle of the binder course.

Banerjee, Srivastava, Manna, and Shahu (2022) showed that the use of geogrids also enhances the service life of the pavement, while Sharbaf and Ghafoori (2021) used numerical modelling to show that the introduction of geogrids can enhance the service life ratio of the pavement by 1.27-1.67 times. The authors also reported an increase in structural capacity by a factor of 1.5-7.5. Limited research has been conducted on base reinforcement applications, thus the scope of this current study was to explore and test the effects of reinforcing the interface between the base and the asphalt concrete layers in a flexible pavement section using biaxial geogrids. In this study, the authors analyzed the effect on rutting and the stresses induced in the base and subgrade layers.

Experimental Program

In this current study, the authors prepared a large-scale laboratory model in order to study the effects of geogrid placement on pavement structure. Two full-scale pavement structures were tested: a control section and a geogrid-reinforced section. The geogrids were placed between the asphalt concrete layer and the base layer. The selection of this location was based on the findings from the literature review. The properties of the material used in constructing each layer of the large-scale model were characterized. The materials used in this research project were sampled during the construction of a street in the new administrative capital of Egypt—the properties of each layer are presented here.

All of the tests were carried out in accordance with ASTM guidelines. The subgrade soil used in the experiment was sand. Table 1 presents the properties of the sand. Table 2 shows that the base layer used in this study was an aggregate mix.

Table 1. Subgrade layer properties.

Property	Result
Specific Gravity	2.70
Optimum Water Content (%)	7.60
$\gamma_{dry\ max}$ (kN/m^3)	17.76
CBR (%)	26.00
Modulus of Elasticity (E), (MPa)	15.80
Cohesion (C)	0
Angle of Internal Friction (ϕ^o)	30.00
Unit Weight in Full Scale Exp Model, (kN/m^3)	18.00

Table 2. Base layer properties.

Property	Result
Specific Gravity	2.56
CBR (%)	142.70
Modulus of Elasticity (E) (MPa)	22.00
Angle of Internal Friction (ϕ^o)	40.00
Unit Weight in Full Scale Exp Model, (kN/m^3)	20 .00

The asphalt concrete used in this project was a premixed asphalt concrete that was sampled during the paving of a city street in the New Administrative Capital of Egypt. The asphalt concrete was tested and asphalt extraction was done to identify the asphalt content. Table 3 presents the asphalt concrete properties. Table 4 presents the gradation of aggregate used in the asphalt concrete mix. After preparing a full -scale model, two samples were cored from each test setup to calculate the actual air voids in the compacted asphalt concrete layer. These air voids were found to be 13.7%.

Table 3. Asphalt concrete layer properties.

Property	Result
Flow (2.5mm)	10.80
Stability (kN)	1.80
Asphalt Content (%)	5.66
Maximum Specific Gravity G_{mb}	2.10
Theoretical Maximum Specific Gravity (G_{mm})	2.41
Unit Weight in Full Scale Exp Model, (kN/m^3)	22.00

Table 4. Asphalt concrete gradation.

Sieve Size	Percentage Passing Cumulative (%)
1”	100.0
¾”	97.8
3/8”	55.0
#4	32.7
#8	23.1
#20	10.8
#50	7.2
#100	3.9
#200	1.8

The geogrids used were polypropylene biaxial geogrids. The aperture dimension was 40 mm x 27 mm and the nominal ultimate tensile strength was 30kN/m, which was less than the value tested in the lab. The ultimate tensile strength was found in the lab to be 40.9 kN/m and the modulus of elasticity was 161.5 MPa. Table 5 presents the details of the geogrid properties.

Table 5. Geogrid properties.

Property	Value	Unit
Aperture dimensions	40x27	mm
Minimum rib thickness (MD)	2.1	mm
Minimum rib thickness (XMD)	1.5	mm
Nominal ultimate tensile strength	30.0	kN/m
Junction efficiency	93	%
Flexural stiffness	2,750,000	mg-cm
Aperture stability	0.75	m-N/deg
Resistance to Installation damage	90-95*	%
Resistance to long term degradation	100	%
Resistance to UV degradation	100	%

*based on soil type

Test Design: Full-Scale Test

The study’s major purpose was to develop a realistic test model that would be large enough to minimize the effects of the boundary conditions on the stresses induced during the test, while avoiding the production of an unnecessarily large model. To do this, MIDAS was utilized to run a simulation using a crude finite element model to optimize the size of the large-scale model used in the experiment. The results revealed that a model with the dimensions of two meters high, two meters wide, and one meter deep would

meet the stated criterion. Further details about the simulation used in designing the full-size model can be found in a previous study by Ahmed et al. (2018).

The Egyptian Code of Practice (ECP) for low-volume traffic was used to design the thicknesses of the different layers (Housing and Building Research Center, 2008). The Egyptian code of practice is based on the American Association for State Highway and Transportation Officials (AASHTO, 1995) design methodology. The design suggested a typical section including 10 cm of asphalt concrete and 25 cm of untreated base. This remaining 65 cm were filled with soil that represented the subgrade soil. Figure 1 shows the pavement cross section. To accomplish the appropriate degree of compaction, the weights of the soil were prepared and compacted to a pre-marked depth, according to their unit weight. A plate compactor was used to compact the material. Each material layer was subdivided into sublayers to ensure proper and homogeneous compaction. The strains created inside the sample at various points were measured using pressure gauges. In addition, two linear variable displacement transducers (LVDTs) were used to measure surface rutting. Figure 2 shows the test setup. The laboratory temperature was maintained at 21.5° C, which was the mean annual air temperature (MAAT) in the project location (more on this later).

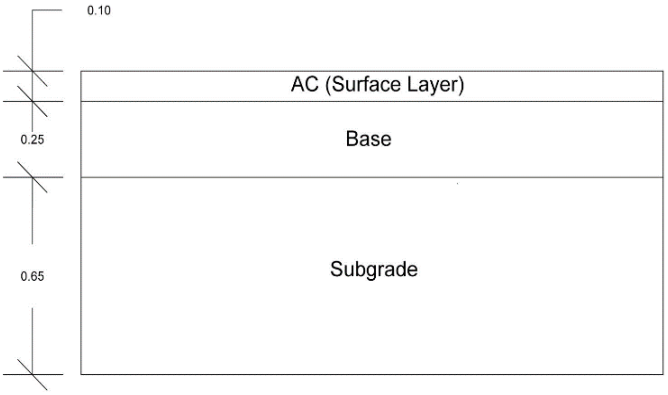


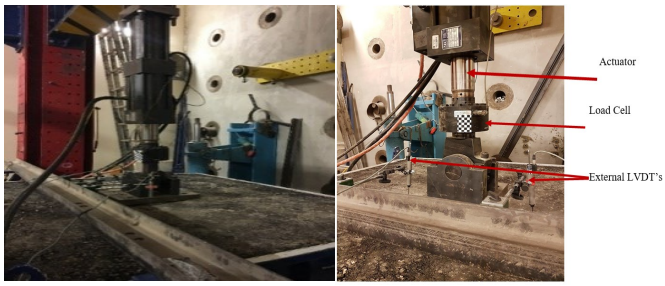
Figure 1. Pavement cross section (all dimensions in meters).

Loading Pattern

Figure 3 shows the load pattern applied in the test. This load pattern was selected to represent slow traffic, which is reflective of traffic intersections, toll stations and slow-speed roads. This was selected because slow traffic has the tendency to cause the highest rutting potential for asphalt pavement, thereby accelerating damage. The duration of the loading cycle was one second, which consisted of

0.2 second of a constant load of 0.5 tons
0.3 second of loading from 0.5 ton to 10 tons
0.3 second of unloading from 10 tons to 0.5 tons
0.2 second of a constant load of 0.5 tons

The loading cycle was applied on the sample for 30,000 cycles.



a) Actuator and LVDT.

b) Complete test setup.

Figure 2. Full-scale test setup.

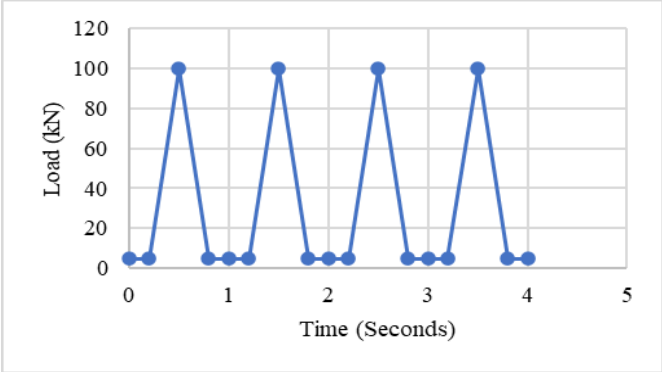


Figure 3. Load pattern.

The applied maximum load of 10 tons (98kN) was converted to equivalent single axle loads (ESALs) using the ESAL equivalency table presented in the Egyptian Code of Practice (Housing and Building Research Center, 2008; AASHTO, 1995). The single tire pressure of 98kN was equivalent to an axle load of 196kN. Using the equivalency table, each load repetition would have had damage equivalent to 36 ESALs. The test was continued up to 30,000 cycles, which means that the maximum ESALs applied throughout the test was equivalent to 1,080,000 for each of the test setups. The new Egyptian administrative capital is located east of Cairo, Egypt, where the mean annual air temperature (MAAT) is 21.54°C. In this area, the pavement is never exposed to freeze/thaw cycles. The weather is generally dry in that area, with an average annual rainfall of 48 mm. During the experiment, the laboratory temperature was set to the MAAT in order to ensure that the pavement would be exposed to the average temperature in the field.

According to previous research, a loading area size of 0.31 x 0.20m was used to simulate the tire loading imprint, which was selected based on previous research (Breakah & Williams, 2015). A steel plate with these dimensions was attached to the machine actuator. Furthermore, rubber was placed in the interface between the plate and the asphalt layer to provide a realistic contact surface. Figure 4 shows the locations of the ten pressure gauges that were inserted inside the samples. These pressure gauges were used to study the stress differences in stress distribution between in different locations and across the two test setups (i.e., with and without geogrids). The pressure gauges were labelled P1 to P10. All of the gauges were placed vertically except P1 and P8, which were placed laterally in order to measure lateral pressures.

Results

The response of the sample due to the applied loading followed the expected pattern. Figure 5 shows a sample of the deformation versus cycle. The deformation had resilient and residual components. The residual component was used to calculate rutting accumulation with cycles. Significant rutting occurred in both samples. The control section's total rutting deformation was 7.3 cm, while the reinforced section's was 5.2 cm. Figure 6 presents the shape of the surface of the asphalt layer at the end of the test. When the results of the two models were compared, it was discovered that the sample with geogrids began to show considerable rutting at cycle number 10,000 (360,000 ESALs). The sample without the geogrid, on the other hand, began to acquire considerable rutting after about 1000 cycles (36,000 ESALs).

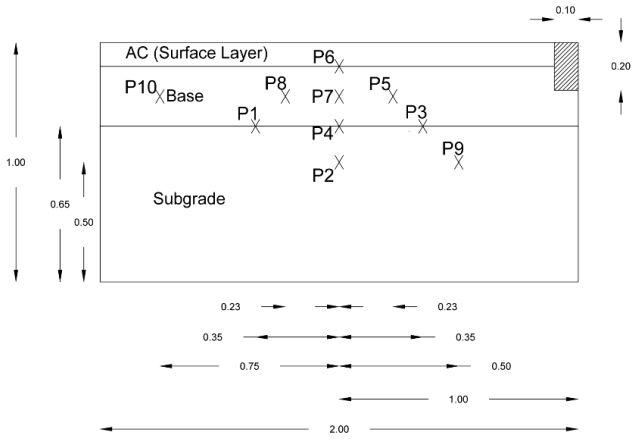


Figure 4. Pressure gauge locations (Elevation).

After removing the asphalt layer and inspecting the model without geogrids, an evident deformation in the base layer was discovered. This could have been due to the extensive rutting that occurred, which caused the stresses to be carried by the base layer rather than the asphalt layer. The asphalt layer and the geogrids, on the other hand, carried all the stresses in the sample with geogrids, as evidenced by the absence of deformation in the base layer after the asphalt layer was removed. This showed that the use of geogrids supported the asphalt layer and led to better stress distribution in the granular layers.

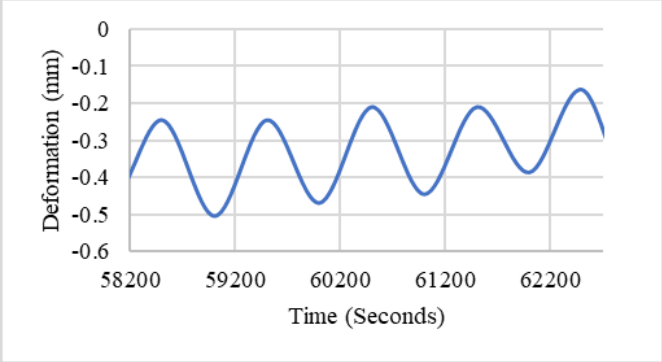


Figure 5. Graph of deformation versus time.

a) Without geogrids.

b) With geogrids.

Figure 6. Sample surface at the end of the test.

Residual Stresses in the Unbound Layers

The results from the ten pressure gauges installed in the different locations were analyzed. Figure 7 presents a sample of the results. The trend shown in this figure was followed by all the sensors placed under the actuator. These four sensors, P2, P4, P6, and P7, which were located directly under the applied load, showed an increase in the stresses after geogrid placement. The remaining sensors showed reduction in the stresses. To better analyze the trends, only the first 1000 cycles will be discussed in order to provide better visualization. All of the pressure gauges placed at the center of the sample (under the point of load application), namely P2, P4, P6, and P7, followed the same trend. In these locations, the stresses developed in the case of geogrid reinforcement were higher than the control sample. Figure 8 shows the results from gauge P2, which can be used to illustrate the trend.

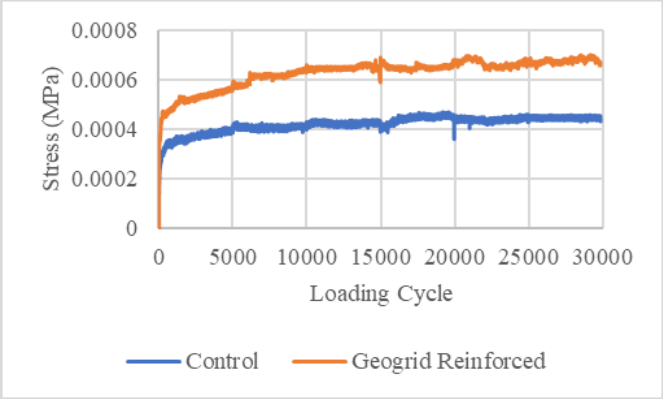


Figure 7. Sample results for the residual stresses.

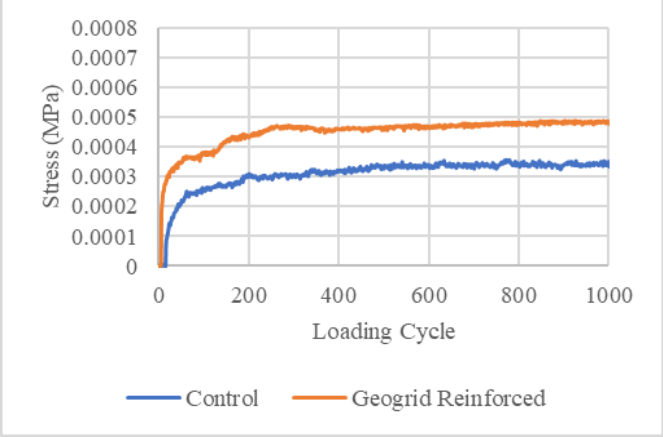


Figure 8. Results of gauge P2.

Gauges P1 and P3 were placed at the interface between the subgrade and base layers. They were also placed at the same distance from the applied load. Figures 9 and 10 illustrate that, during the first few cycles, slightly higher stresses were developed in the case of geogrid reinforcement. Subsequently, there was a significant increase in the stresses developed by the control sample. It can also be seen that the lateral stresses developed were about ten times higher than the vertical stresses; this could have been due to the confinement in the lateral direction.

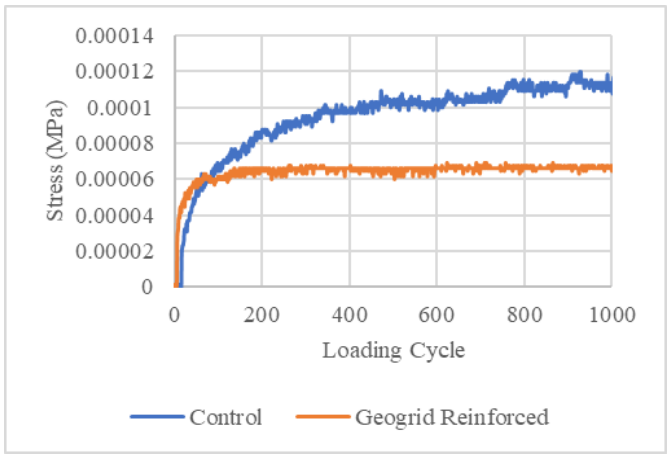


Figure 9. Results of gauge P1, measuring lateral stresses at the subgrade/base interface.

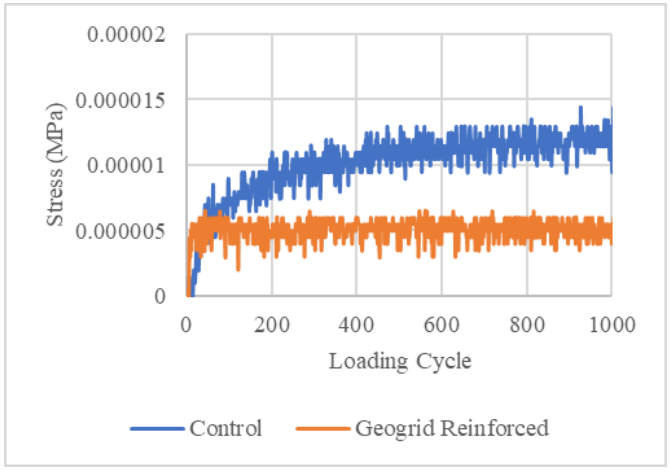


Figure 10. Results of gauge P3, measuring vertical stresses at the subgrade/base interface.

Figures 11 and 12 display the results of all the pressure gauges for the control and the geogrid reinforced samples, respectively. The highest value of stress was found at gauge P6, in the case of geogrid reinforcement. This sensor was the one placed at the interface of the asphalt concrete and base at the center of the sample. The increase in stress could have been due to the redistribution of stresses in the sample due to the presence of geogrids. In order to further study the change caused by the use of geogrid reinforcement, the average change in stress due to the presence of geogrids was analyzed. From the results presented in Table 6, it can be concluded that the pressure gauges placed at the center of the model were the ones that were subjected to an increase in stresses with the introduction of geogrid

reinforcement. It can be also seen that the further the sensor is placed from the center of the model, the higher the reduction in stresses that occur as a result of the introduction of geogrid reinforcement.

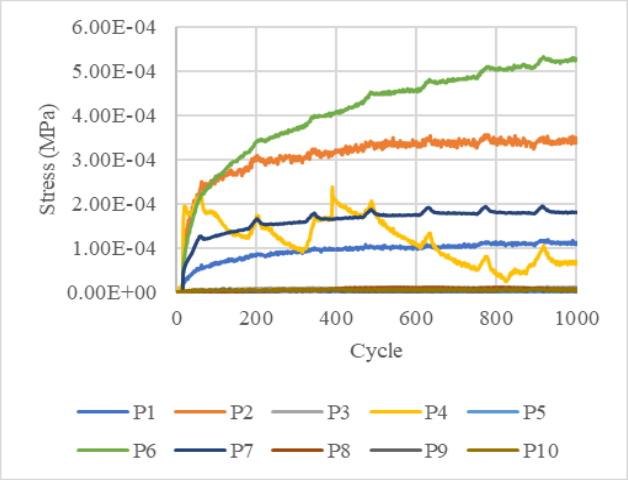


Figure 11. Results of all gauges placed in the control sample.

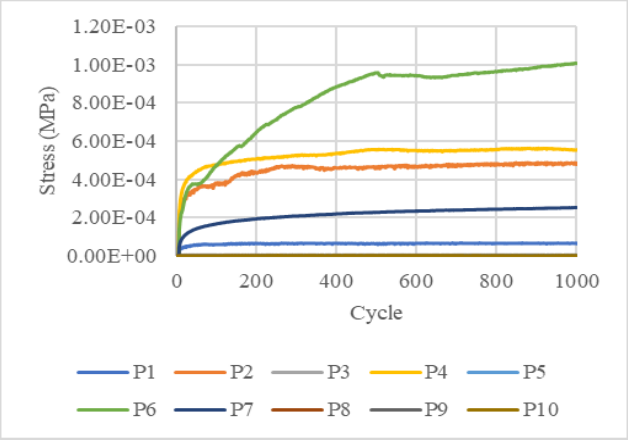


Figure 12. Results of all gauges placed in the geogrid reinforced sample.

Design Considerations

Based on the experimental work done in this study, there are some design considerations that need to be discussed. According to the literature, the position of geogrids has a major effect on their performance. In this current study, only one position was studied and the results showed a major enhancement in the rutting performance of the pavement, but the other locations need to be tested. The stress in the soil increased immediately under the point of load application, but the general stresses outside of this location were reduced. Geogrids can work effectively with inclined slopes of base, because they are used for soil stabilization in several geotechnical applications. The effect of changing the load intensity needs to be analyzed, but it is expected that, with the change in the thickness of the pavement layer with the increased load, geogrid-reinforced pavement performance will remain superior to that of non-reinforced pavements, as shown in this study and literature. Finally, due to the size of the openings of the geogrids mesh, the use of geogrids is not expected to have any effect on the drainage behavior of the pavement.

Table 6. Average change in stresses.

Pressure Gauge	Change in Stress Due to Geogrid	Type of Change
P1	60.7%	Decrease
P2	50.5%	Increase
P3	51.7%	Decrease
P4	96.0%	Increase
P5	10.0%	Decrease
P6	50.9%	Increase
P7	32.7%	Increase
P8	77.2%	Decrease
P9	92.6%	Decrease
P10	48.0%	Decrease

Conclusions

From the results and analysis of this study, it can be concluded that the use of biaxial geogrids placed in the interface between the asphalt concrete layer and the base layer is effective in reducing the rutting potential of flexible pavements. Based on the conditions presented from this current study, the following can be concluded:

- Less deformation occurred in the base layer when geogrids were used.
- The use of geogrids delayed the occurrence of rutting in the studied pavement structure.
- The stresses in the subgrade and base layers directly under the point of load application were higher with the use of geogrids.
- Moving away from the load center, the stresses in the soil were reduced by the introduction of geogrids.
- The size of the stress bulb generated in the soil due to the loading was reduced with the introduction of geogrids and the intensity of the stress at the center of the bulb was increased.
- The authors recommend that different locations for the geogrid placement, different subgrades, different base layer strength, and different asphalt mixtures be studied.

References

Abu-Farsakh, M. Y., & Chen, Q. (2011). Evaluation of geogrid base reinforcement in flexible pavement using cyclic plate load testing. *International Journal of Pavement Engineering*, *12*(03), 275-288.

Abu-Farsakh, M., Hanandeh, S., Mohammad, L., & Chen, Q. (2016). Performance of geosynthetic reinforced/ stabilized paved roads built over soft soil under cyclic plate loads. *Geotextiles and Geomembranes*, *44*(6), 845 -853.

Ahmed, M., Abdelhamid, M., Hussain, S., Khedr, S., Breakah, T., Saudy, M. …Abou-Zeid, M. (2018). Geogrid reinforcement in flexible paved roads. *Proceedings of the Canadian Society of Civil Engineers Annual Conference*.

Alimohammadi, H., Schaefer, V. R., Zheng, J., White D. J., Zheng, G., (2021). A State-of-the-art Large-scale Laboratory Approach to Evaluating the Effectiveness of Geogrid Reinforcement in Flexible Pavements. *Geosynthetics Conference 2021,* 618-628.

American Association of State Highway and Transportation Officials (AASHTO). (1995). AASHTO provisional standards. *American Association of State Highway and Transportation Officials. Washington, D.C., USA.*

Banerjee, S., Srivastava, M. V. K., Manna, B., & Shahu, J. T. (2022). A Novel Approach to the Design of Geogrid -Reinforced Flexible Pavements. *International Journal of Geosynthetics and Ground Engineering*, *8*(2), 1-15.

Breakah, T. M., & Williams, R. C. (2015). Stochastic finite element analysis of moisture damage in hot mix asphalt. *Materials and Structures*, *48*(1-2), 93-106.

Gu, F., Luo, X., Luo, R., Lytton, R. L., Hajj, E. Y., & Siddharthan, R. V. (2016). Numerical modeling of geogrid -reinforced flexible pavement and corresponding validation using large-scale tank test. *Construction and Building Materials*, *122*, 214-230.

Haas, R., Walls, J., & Carroll, R. G. (1988). Geogrid reinforcement of granular bases in flexible pavements. *Transportation research record*, *1188*(1), 19-27.

Housing and Building Research Center (HBRC) (2008) Egyptian Code of Practice, Part 6: Structural Design of Roads. *Housing and Building Research Center, Cairo, Egypt*.

Ibrahim, E. M., El-Badawy, S. M., Ibrahim, M. H., Gabr, A., & Azam, A. (2017). Effect of geogrid reinforcement on flexible pavements. *Innovative Infrastructure Solutions*, *2*(1), 54.

Jasim, A. F., Fattah, M. Y., Al-Saadi, I. F., & Abbas, A. S. (2020). Geogrid reinforcement optimal location under different tire contact stress assumptions. *International Journal of Pavement Research and Technology*, 1-9.

Mattar, S., Tammam, M., Mekhail, J., Mahran, O., Soliman, N., Badawy, Y. ...Zayed, Z. (2022). Geogrid Reinforcement in the Treated Base Layer of Flexible Pavements. *Canadian Society of Civil Engineering Annual Conference* (pp. 609-621).

Mittal, A., & Shukla, S. (2020). Effect of Geogrid Reinforcement on Strength, Thickness and Cost of Low-volume Rural Roads. *Jordan Journal of Civil Engineering*, *14*(4).

Moayedi, H., Kazemian, S., Prasad, A., & Huat, B. B. (2009). Effect of geogrid reinforcement location in paved road improvement. *Electronic Journal of Geotechnical Engineering*, *14*, 1-11.

Moghaddas-Nejad, F., & Small, J. C. (1999). Effect of geogrid reinforcement in model track tests on pavements-Closure. *Journal of Transportation Engineering, 125* (3), 267.

Pereira, G. S., Pitanga, H. N., da Silva, T. O., e Almeida, D. C., & Lunz, K. B. (2021). Effect of geosynthetic reinforcement insertion on mechanical properties of hot and cold asphalt mixtures. *International Journal of Pavement Research and Technology*, *14*(4), 496-504.

Reyes, F., & Kohler, E. (2006). Colombian experience with accelerated testing of geogrid-reinforced flexible *pavement. Transportation Research Board 84th Annual Meeting, Washington D.C., USA.*

Sharbaf, M., & Ghafoori, N. (2021). Laboratory evaluation of geogrid-reinforced flexible pavements. *Transportation Engineering*, *4*, 100070.

Susanto, H. A., Yang, S. H., & Duc, M. A. (2021). Performance evaluation of geogrid in flexible pavement using mechanical-empirical design approach. *International Journal of Pavement Research and Technology*, 1-15.

Zornberg, J. G. (2017). Functions and applications of geosynthetics in roadways. *Procedia engineering*, 189, 298-306.

Zornberg, J. G., Gupta, R., Prozzi, J. A., & Goehl, D. (2008, March). Case histories on geogrid reinforced pavements to mitigate problems associated with expansive subgrade soils. In *the First Pan American Geosynthetics Conference & Exhibition, Cancun, Mexico*.

Biographies

TAMER BREAKAH is an Assistant Professor of Construction Management at Ball State University. He earned his BS and MS degrees in construction engineering (2001 and 2006, respectively) from the American University in Cairo, Egypt, and PhD in civil engineering in 2009 from Iowa State University. Dr. Breakah's research interests include construction materials, sustainable materials, and lifecycle costing. Dr. Breakah may be reached at tmbreakah@bsu.edu

MARAM SAUDY is an assistant professor and associate chair in the Construction Engineering Department at The American University in Cairo (AUC). Saudy received her BS and MS degrees from Cairo University in 2001 and 2004, respectively. She earned her PhD in highway and airport engineering from Cairo University in 2008. Her research interests include pavement materials, design, and

sustainability. Dr. Saudy may be reached at maramsaudy@aucegypt.edu

SAFWAN KHEDR is a professor of construction engineering at The American University in Cairo, Egypt. He earned his BS from Cairo University (Civil Engineering) in 1971, and MS and PhD degrees in civil engineering (1974 and 1979, respectively) from The Ohio State University. Dr. Khedr research interests include construction materials, sustainable materials, and geotechnical engineering. Dr. Khedr may be reached at safkhedr@aucegypt.edu

OMAR EL-KADI is an Adjunct Assistant Professor of Geomatics and Geoinformatics at the American University in Cairo, Egypt, where he also operates the structural laboratory. He earned his BS and MS degrees in civil engineering (2004 and 2011, respectively) from the Ain Shames University in Cairo, Egypt, and PhD (Civil Engineering) in 2020 from Cairo University, Egypt. Dr. Kadi's research interests include geomatics, geotechnical, structural engineering, and construction materials, Dr. Kadi may be reached at o.elkadi@aucegypt.edu

MAI AHMED earned her BSc in Construction Management from the American University in Cairo, Egypt, in 2018 and is currently pursuing an MSc in the same field. Ms. Ahmed may be reached at mai_ibrahim@aucegypt.edu

MENNATTALLAH ABDELHAMID is a tender and coordination engineer at the Bureau Egyptien de Conseils Technique (BECT). She earned her BSc in Construction Management from the American University in Cairo, Egypt, in 2018. Ms. Abdelhamid may be reached at mennatallahmohamed@aucegypt.edu

SARAH HUSSAIN is a project management professional with demonstrated experience working with PMOs and large-scale contractors on marquee mega projects. Sarah earned her BSc in Construction Management from the American University in Cairo, Egypt, in 2018 and is currently pursuing an MSc in the same field, while also working as a graduate teaching assistant since 2020. Sarah has had several project managements positions, including PMO Specialist, and more recently, Contracts Administrator. Ms. Hussain may be reached at sarahzaki95@aucegypt.edu

GAS-LIFT, CLOSED-LOOP OPTIMIZATION USING IIOT EDGE TECHNOLOGY

Denise Sherrod, Texas A&M University; Behbood Zoghi, Texas A&M University

Abstract

Gas-lift optimization is the process of determining the optimal injection rate for each well within a network supported by a central compression facility. This process is usually an open-loop process utilizing nodal analysis software. In this paper, the authors present an advanced method that performs the optimization process automatically, utilizing new IIoT (Industrial Internet of Things) technology with edge devices. IIoT technology provides the ability to implement a closed-loop system by collecting data from multiple sources, running an advanced algorithm to determine the optimal injection rates, writing the setpoints to the well controllers, and analyzing the results using real-time data. A project was conducted on a central compression facility with six compressors supporting a network of 12 active gas-lift wells. The wells were monitored for 94 days before and after the implementation of the new solution. Both quantitative and qualitative research methods were used to determine the overall effectiveness of the closed-loop IIoT gas-lift optimization solution. Data collected proved the algorithm to be accurate, and the closed-loop optimization resulted in improved gas-lift optimization, increased production, as well as a 66% improvement in response time to upset conditions. Based on these findings, the authors recommend the implementation of a secure IIoT architecture, expansion of IIoT solutions, and the development and implementation of a full IIoT support lifecycle.

Introduction

When a well is initially drilled, the reservoir pressure is adequate for moving the well's products (oil, water, and gas) to the surface, enabling the well to flow naturally. As production continues, the natural reservoir pressure decreases; consequently, various artificial lift methods need to be employed to maintain production. The gas-lift method, which is often the first method applied, works by injecting compressed gas into the wellbore. The gas decreases the hydrostatic pressure of the column of oil, enabling the lower reservoir pressure to produce the fluids. There is a series of valves that open and close, based on the tubing pressure and casing pressure through which the gas is injected. Each gas-lift well, or often a group of wells, is controlled by a PLC (programmable logic controller), which contains setpoints to control the gas injection rates by adjusting an automated choke on the injection valve.

A production engineer runs various well models to determine the appropriate injection rate for each well. Figure 1 shows that more gas does not necessarily mean increased production.

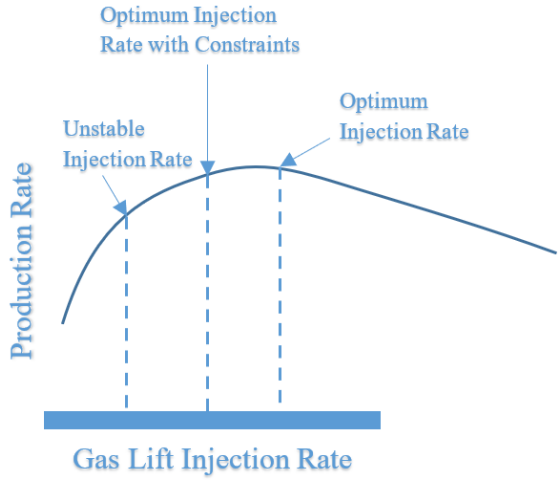


Figure 1. Optimum gas-lift injection rate.

"The quantity of the injected gas [into each well] at any point in time is considered as a critical variable; whereas a high injection rate does not necessarily increase total production and a low injected rate may result in a decrease in production rate" (Miresmaeili, Zoveidavianpoor, Jalilavi, Gerami & Rajabi, 2019). The optimal injection rate changes due to the well's natural decline in production and conditions within the reservoir. Surface compression facilities are used to produce the compressed gas, with each facility being connected to multiple wells, thereby creating a network of wells. The compressors' capacity and availability determine the total amount of gas that can be distributed to the wells in the network. Determining the optimal amount of gas to allocate to each well within a given network is accomplished through network modeling. A production engineer first determines the optimal rate for each well then puts that information into the network model. Figure 2 shows how the network model determines the optimal rate for all wells within the network based on the total amount of gas that can be allocated.

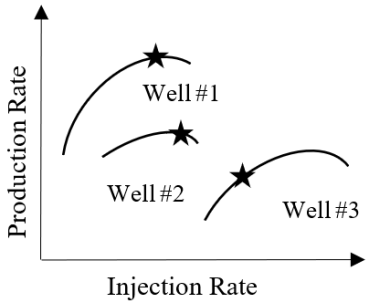


Figure 2. Optimal gas-lift injection rates (marked with "stars") for wells in a network.

This process, referred to as an integrated process model (IPM), is typically performed whenever a new well test is available. The output of the process provides updated injection rates, which are entered remotely as recalibrated setpoints into each well's controller (PLC).

Industrial Internet of Things (IIoT) Technology

Industrial control systems in the oil and gas industry typically consist of PLCs and HMIs (Human Machine Interface). The PLCs are located throughout the field and are programmed using basic logic to control various types of equipment, such as valves and pumps. The HMIs collect the data from the PLCs and provide visualizations and alarm notifications to help the operators manage their wells and equipment. The PLCs and HMIs are usually deployed in an industrial control network, which requires a closed architecture, allowing minimal outside access. Direct connection to other business networks and the internet is strictly prohibited. IIoT is a newer technology that utilizes edge devices in place of PLCs. The edge devices are small, low-cost computers capable of making complex calculations. They can be programmed to perform the same type of control functions as the PLCs but with the added ability to run advanced analytics. Because the edge devices communicate to the cloud, they present security concerns as an inherent function of their design. A new security architecture is required to ensure that IIoT technology meets the security requirements of the industrial control network.

Literature Review
Gas-lift Optimization Research

Numerous studies have been published on the topic of gas-lift optimization. Some publications concentrate on the gas-lift process itself, while others attempt to find the most efficient way to optimize the full integrated network while applying gas-lift capacity constraints. For example, various methods of gas-lift optimization range from a simple single-well nodal analysis to a very complex full-scale integrated modeling approach (Rashid, Bailey & Couët, 2012; Rashid et al., 2012). The limitations of each method are discussed, as well as the data requirements for each process. The integrated model technique is recommended, because it can account for the numerous constraints encountered in an actual production environment. This recommendation validates the decision to use the IPM optimization method in this current project.

A simple gas-lift algorithm was tested by Fandi (2015) to determine its accuracy, when compared to the Equal Slope allocation method established by Kanu, Mach, and Brown (1981) as a reliable allocation method. The advantage of the simple algorithm is that it requires only a few data elements and can be calculated quickly. In contrast, the Equal Slope allocation method requires much more data and involves a lengthy procedure. A simulation using six wells was conducted using both methods. The results showed that the two approaches produced similar results. This study's finding is significant because the algorithm tested is comparable to the one used in this current project. In addition, the algorithm used in this current project included several additional data elements, so it should prove equally, if not more, effective.

A study of a nonlinear predictive model process control (NPMC) method of gas-lift optimization was published in the *Journal of Process Control* (Diehl et al., 2018). The study utilized simulators for both the reservoir and the surface network and focused on analyzing a single well. The results of the study—that is, the ability to stabilize the well—showed an increase in production using a predictive model solution. "The gains verified in this work reached around 45%, which corroborates with the order of magnitude expected in some research papers" (Diehl et al., 2018). The algorithm used in this current project was not as sophisticated as a NPMC, but it incorporated the same concept through real-time optimization and, therefore, was expected to produce similar results.

IoT

IoT technology has been used for years in other industries but is relatively new to the oil and gas industry. Although some papers have been published that describe IoT technology within this industry (Irons-Mclean & Greengrass, 2016), it is not yet recognized as a trusted alternative to legacy automation systems. A systematic review of IoT in the oil and gas industry was published in the JIoT (Wanasinghe, Gosine, James, Mann, De Silva & Warrian, 2020). providing an evaluation of the application, impact, challenges, and opportunities of this technology within the industry. The assessment gathered data from 66 articles dealing with IoT in the oil and gas industry to develop its final report. The report shows that this technology's primary application within the oil and gas industry is for automation and control, personnel tracking, collaboration and digital twin, and supply chain fleet management.

The impact this technology provides is "a paradigm shift in the entire industrial and social status quo" (Wanasinghe et al., 2020). The challenges include cybersecurity issues, the lack of intrinsically safe devices, and interoperability with other systems. This current project has overcome these challenges by implementing an approved security architecture, eliminating the need for intrinsically safe device by placing the edge device away from the wells, and developing a specific user interface for the application. The opportunities offer a means to potentially reduce capital and operations expenditures while advancing technology in health, environment, and safety. The report quotes Mahdavi (2017) as saying, "with the right IoT infrastructure, we can achieve integrated management of the reservoir, well, and surface

facilities, enabling end-to-end optimization with a much more economic value proposition compared to the status quo." The benefits of using an edge device to implement a closed-loop solution are discussed in a case study that included the analysis of 41 process variables associated with 11 manipulated variables through a feedback loop (Luo, Zhao & Yin, 2018). In that study, the proposed edge architecture was applied to the TE benchmark process, which was described in a study by Bathelt, Ricker & Jelali (2015). The study concluded that an IoT architecture functions well in a large-scale industrial process used for monitoring and controlling. It showed that this architecture eliminates online processing and reduces communication data usage. This study also validated the use of edge technology versus cloud computing.

Project Overview

Determining the optimal injection rate for a gas-lift well is critical to maximizing the well's production. It is particularly important when upset conditions occur, such as the loss of a compressor. When a compressor failure occurs, the total amount of available gas that can be distributed to the wells decreases. If the well setpoints are not updated, the wells closest to the facility will consume all the gas, causing the more distant wells to stop producing. This scenario can cause significant production losses, especially if the wells farthest from the compression facility are higher producing wells. Consequently, when upset conditions occur, models are run to determine the appropriate injection rate for each well, accounting for the decrease in available gas, and priority is given to the better wells. In an open-loop process, this is a manual process that, subsequently, causes a delay in network optimization. Operating when the network is not fully optimized results in lower production volumes and decreased revenue. In this current project, the authors explored the capabilities of utilizing new IIoT technology with edge devices to demonstrate the benefits of a closed-loop gas-lift optimization system. The solution tested the ability to run an algorithm on an edge device that calculated the optimal injection rate for each well. A closed-loop system was created by developing a scenario in which the edge device would automatically download recalibrated setpoints to the well controllers.

The findings from Kanu et al. (1981) were significant because the algorithm tested was comparable to the one used in this current project. In addition, the algorithm used in the project included several additional data elements, so it should prove equally, if not more, effective. Real-time data from the wells, compression facility, and production network were provided as inputs to the algorithm. The overall project objectives were:

1. Implement an advanced algorithm that would automatically determine the optimum allocation of lift gas for a network of wells.
2. Automatically send the optimum gas-lift setpoints to the well controllers.
3. Demonstrate the benefits of a closed-loop optimization engine.
4. Prove the capabilities of IIoT technology.
5. Establish a secure IIoT architecture within an industrial control network.

The project was conducted on a central compression facility with six compressors supporting a network of 12 active gas-lift wells. The wells were monitored for 94 days before and after the implementation of the new solution. The site selected for this project offered a reduced risk compared to other sites in the area, because excess compression was available at this site. Producing more gas than is needed decreases the impact of a compressor outage. The injection setpoints would not vary widely under normal operations with excess lift gas; therefore, error deviations would be less impactful.

The benefits of this project include reduced workforce requirements, increased production volumes, and the opportunity to test the applicability of IIoT technology within an oil and gas industrial control network. Currently, engineers use various commercial applications to perform gas-lift network optimization, which requires intensive resources to maintain configurations and run the well models. The closed-loop solution automatically calculates the optimal setpoints and downloads the setpoints to the well controllers without human intervention. This programmed approach reduced the workforce requirements needed to perform gas-lift optimization. In addition, this advanced control resulted in faster field optimization and increased production by efficiently utilizing the existing gas injection capacity and quickly adjusting to disturbances. Implementing this solution on an edge device also proved the viability of IIoT technology, which expands the system's ability to perform real-time advanced process control.

Limited research has been done on true closed-loop gas-lift optimization, although it has been recommended over open-loop control (Jing, Errouissi, Al-Durra & Boiko, 2015). One study, conducted by Ni, Ren, and Mao (2012), demonstrated the results of a closed-loop solution using a sequenced-based automation controller. A dynamical simulator was used to illustrate the general gas-lift principles, and a transient multiphase pipe flow simulator was used to represent the network model. "Around 20-40% of production loss is observed due to gas-lift instability for typical well settings in our simulations" (Ni et al., 2012), The study showed that the closed-loop solution increased production due the ability to stabilize the wells.

Methodology

Both quantitative and qualitative research methods were used to determine the overall effectiveness of the closed-loop IIoT gas-lift optimization solution. Case study qualitative research methods were used to examine the changes in manpower requirements between the current manual

process and the new closed-loop solution. Participants included the production engineers that supported the field in which the solution was deployed. The participants were asked to document the amount of time they spent performing the network optimization prior to implementing the new closed-loop solution. They were asked to document the same data after the implementation. Descriptive quantitative research methods were conducted using real-time data from the wells and compression facility. Quantitative data were used to achieve the following objectives.

1. Validate the accuracy of the algorithm's computations.
2. Measure the overall effect of a closed-loop gas-lift optimization solution.
3. Test the security of the IIoT architecture.

A generic edge device was used to automatically calculate the optimal injection rate for each well. Various containers were deployed on the edge device to perform the needed functions. The containers were developed in an Azure IoT hub in the cloud and downloaded to the edge device. A Modbus container acquired the real-time data from PLCs. A Python container ran the algorithm. A SQL container stored the inputs and outputs.

A data historian was used to capture all the research data during the project's implementation phase. User interfaces were developed to visualize and analyze the data. The IIoT edge device communicated directly with the PLCs to obtain real-time data through a fiber-optic network. Well test data, used to calculate the well models, were sent from the business network to an Azure database in the cloud and then to the edge device through a cellular connection. The edge device calculated and downloaded new injection rates for each well every 15 minutes. Each time the algorithm ran, the inputs and outputs were logged in an SQL database in the edge device for onsite analysis and were collected by a data historian over an Ethernet IP-Radio communication network directly from the PLCs. Data obtained before the implementation of the closed-loop system were compared to the same data after the implementation. The data sets used in the analysis included the following.

- Before Closed-Loop Gas-lift Optimization – Data from October 3, 2020, to January 4, 2021.
- After Closed-Loop Gas-lift Optimization – Data from January 5, 2021, to April 8, 2021.

The closed-loop system was implemented on Jan 4, 2021; therefore, the production data from Jan 5, 2021, represented the system in full operation.

Project Results

During the test phase of the project, it was important to validate the new algorithm's accuracy. To confirm that the injection rate setpoints generated by the algorithm were valid, the program was run in test mode for several weeks. During this test mode, the algorithm ran every hour using real-time data obtained from the PLCs and well models. The algorithm ran exactly as it would in production, except the injection rate setpoints were written to an SQL database rather than to the well PLCs. The production engineers continued to generate the well and network models manually, producing the recalibrated setpoints. The production engineers compared the manually generated setpoints to those generated by the new algorithm. Initially, slight variations were observed, which necessitated adjustments to improve the algorithm's accuracy. Once the accuracy was validated to the production engineers' confidence level, the new closed-loop optimization solution was put into production.

One of the primary objectives of the closed-loop solution was to automatically detect and respond to compressor down events. This objective aimed to detect compressor outages, calculate new optimized injection rates for each well based on the reduced gas capacity, and automatically download the recalibrated setpoints to each well. An upset condition was defined as a rate or pressure PV (Present Value) of zero for any of the compressors in the central compression facility. The condition had to be detected in two concurrent runs of the algorithm to be considered valid. This constraint was implemented to eliminate bad data. A user interface was developed to help the production engineers verify compressor events and verify that recalibrated setpoints were being downloaded to the wells. Numerous compressor down events occurred during the project and recalibrated setpoints were downloaded to the wells during this same timeframe. This result validated the algorithm's ability to detect and respond to compressor down events.

To achieve improved optimization in a gas-lift system, proper utilization of the available gas is crucial. In Figure 7, the amount of available gas for injection is represented by the purple line. The total gas being injected is represented by the green line.

Figure 4. Gas-lift utilization graph.

Before the closed-loop system was implemented, the available gas was ~21,540 mcf and the total gas being injected was ~16,150 mcf, resulting in a 79% utilization rate. After the closed-loop system was in production, the total available gas was ~20,133 mcf and the total gas being injected was ~18,494, resulting in a 92% utilization rate. From this project, the authors confirmed that automatically adjusting the injection rate setpoints through a closed-loop system improved overall gas utilization.

Another metric used to determine the closed-loop system's success was the response time to upset conditions such as compressor outages. During an outage, the amount of available gas is constrained, thereby decreasing production until the recalibrated setpoints are downloaded to the well PLCs. For purposes of this analysis, response time was defined as the difference between the time the compressor down event was captured in the data historian and the time all of the recalibrated setpoints were downloaded. The response time for the manual process, before the closed-loop system was implemented, was 59 minutes. The response time with the automated solution was 20 minutes. This result proved that the closed-loop optimization resulted in a 66% improvement in response time.

The closed-loop system was designed to provide a much less manpower-intensive solution than the previous manual processes. To calculate the time difference, customer interviews were conducted. The production engineers were asked to provide time estimates of their process before and after the new solution was implemented. Figure 5 shows the four steps that made up the manual process.

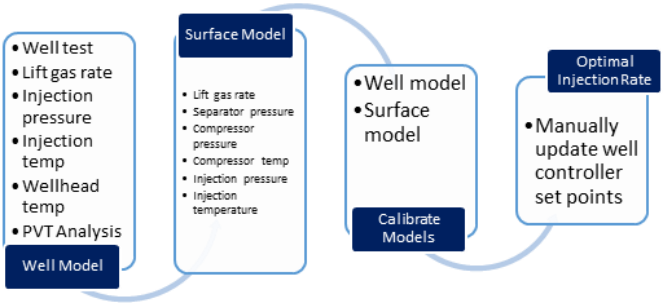
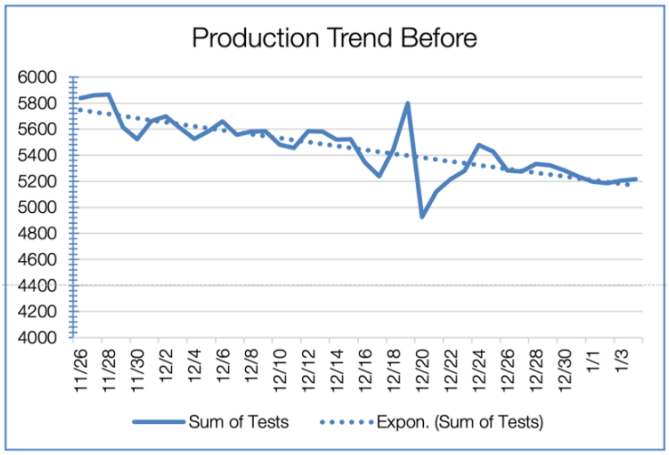


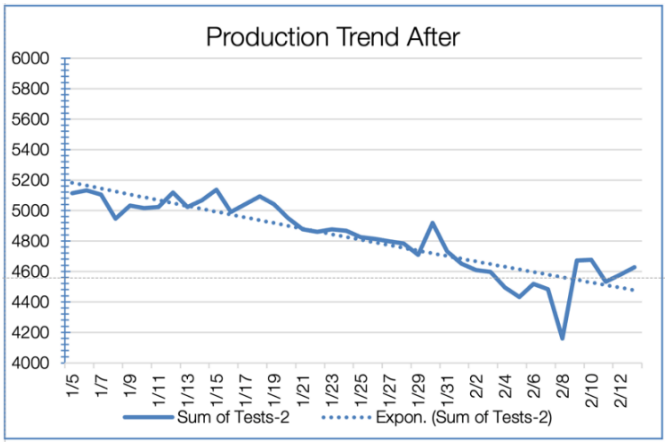
Figure 5. Manual gas-lift optimization process.

The closed-loop solution performed all the steps automatically, so the only time spent was for data validation. The overall time savings, for the 12 wells included in the project, equaled 34 hours per month. An increase in overall production determined the ultimate success of the closed-loop system. The original estimate was an increase of between 0.5% and 1%. The data used to analyze this information was the sum of the well tests (oil measurement) for the wells included in this project. Figures 6(a-b) show that the data were analyzed in two time periods: before the closed-loop system was implemented and after it was implemented.

The difference in the trend before the system was implemented was calculated by subtracting the ending oil volume, 5217 BOPD (barrel of oil per day) from the beginning oil volume, 6610 BOPD. The result was a -1392 BOPD difference, which equaled a 21% decrease for this period. The same method was used for the trend after the system was implemented, subtracting the ending oil volume of 4085 BOPD from the beginning oil volume of 5113 BOPD. The result was a -1027 BOPD difference, which equaled a 20% decrease for this period. Subtracting the ending decrease from the beginning decrease showed a resulting increase of 1%. Figure 7 combines all of this information into a single graph for easier comparison.



a) Before the closed-loop system was implemented.



b) After the closed-loop system was implemented.

Figure 6. Comparison of production volumes.

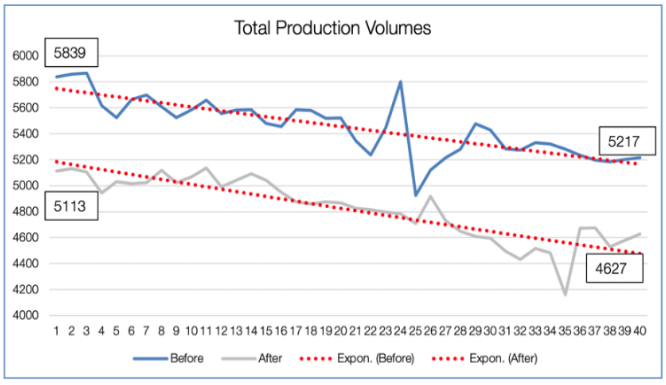


Figure 7. Total production comparison.

In order to determine the actual impact of the gas-lift optimization solution, the expected production trend for the wells had to be examined. The wells included in this project were horizontal shale wells that had a high rate of decline. Figure 8 shows the production trend plotted for all twelve wells (in green) against the expected decline curve for wells in this reservoir (in black). The system used to capture this data only contained production information for 365 days, but the wells in this study were between 450 and 550 days old, as indicated by the area in Figure 8 enclosed by an oval.

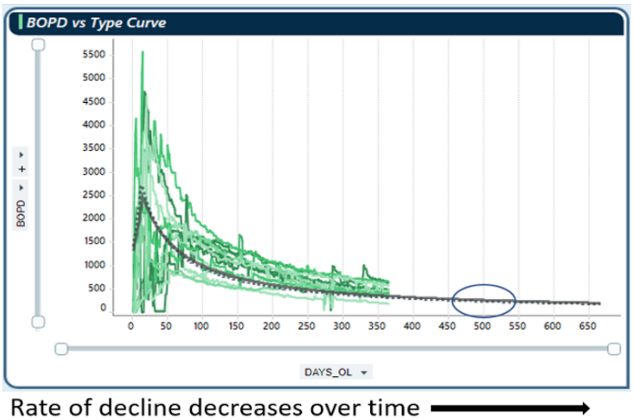


Figure 8. Rate of decline.

As the information shows, the rate of decline decreased throughout the life of the well. Based on analysis of this data, the expected decline should have been 0.5% higher during the period after the solution was implemented, compared to the same period before. The results of the study showed a 1% change; therefore, 0.5% could be contributed to the implementation of the closed-loop solution. The primary concern for implementing an IIoT solution within a process control network was the system's security. The process control network's security architecture requires separation from the business network and complete isolation from the internet. Introducing a device that communicates directly to the cloud broke this security model.

Figure 9 shows that an IIoT architecture was initially developed using an IoT private cloud that communicated to the business network, the industrial control network, and the edge device. The use of a private cloud provided increased security over a public cloud; however, it did not meet the security requirements that the three networks be completely separate. Figure 10 shows that, in order to achieve complete separation, an IoT DMZ was implemented. To test the security of this architecture, penetration tests into the IIoT device were performed. The penetration testing was performed with an open-source application called Ubuntu. The following functions were run with this application.

1. Execute vulnerabilities assessment tasks.
2. Identify online devices in a network.
3. Collect information of targeted devices
4. Expose the attacks against targeted devices.

The results of the tests confirmed that there were no vulnerabilities or exposures to the IIoT edge device.

Conclusions and Recommendations

This project achieved all the stated objectives. It offered a unique opportunity to concurrently test several value propositions during normal production operations with minimal risks. The IIoT technology provided a platform to test the new gas-lift algorithm within a closed-loop system, thus supplying a continuous process for automatic injection rate calculations and updates. The data showed that the algorithm resulted in improved gas-lift optimization and increased production, which is significant beyond this project for several reasons.

1. This project was conducted using 12 wells that were toward the bottom of their decline curve. The results of this project could be amplified by implementation on newly drilled wells with high production volumes and in fields with larger gas-lift networks.
2. Many field operation processes in the oil and gas industry are still manual. Optimization can be achieved by expanding IIoT closed-loop solutions to these processes.
3. The IIoT platform delivers technology that can be used to implement other solutions quickly.

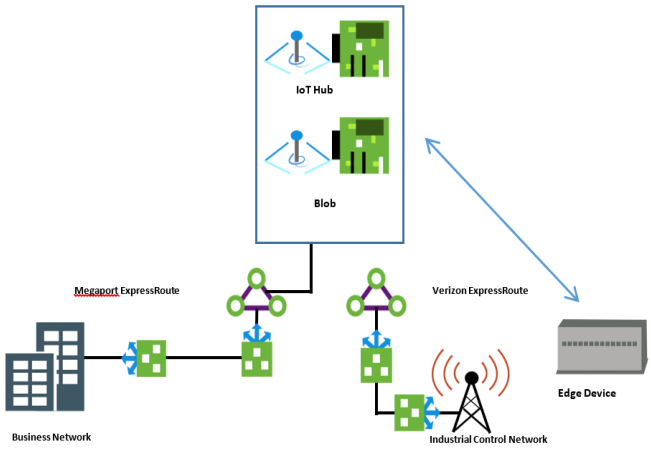


Figure 9. IIoT initial architecture.

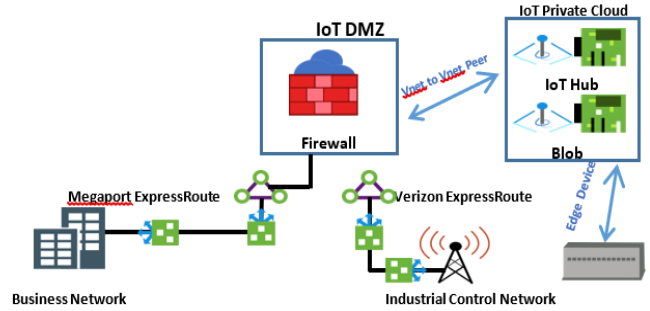


Figure 10. IIoT final architecture.

Recommendations

The most critical component in implementing IIoT technology is the full implementation of a secure IIoT architecture. The environment should be approved by the corporate cybersecurity team. The final architecture needs to include the following components.

1. Device certification provides a method for certification of newly installed devices.
2. Azure Active Directory for Identity Access Management provides a means to ensure that only authenticated users have access.
3. Production Azure environment with backup and DR solutions provides a secure and reliable cloud architecture to support IIoT devices deployed in the field.
4. Cybersecurity testing and audits ensures that the IIoT does not introduce new security threats.
5. Implement an edge device management solution to provide capabilities for monitoring the health of deployed edge devices.
6. Network segmentation extends the corporate security architecture to the cloud environment.

Another recommendation includes the continued development of IIoT solutions. This project proved that IIoT technology is a viable solution for industrial control in the oil and gas industry. Gas-lift optimization is only one application. This technology can be applied to numerous other situations. One operational area that could be greatly impacted is the deployment of such a solution at central production facilities. An IIoT device could be programmed to control the amount of fluid in and out of the facility, which would optimize operations. Wells could be slowed down to manage facility constraints, which would eliminate large-scale shutdowns. Another option for an IIoT implementation is the replacement of well controllers. Many artificial lift methods require expensive proprietary controllers, which could be replaced with lower-cost edge devices.

The final recommendation is to develop and implement a full IIoT support lifecycle. IIoT technology presents unique challenges, because it requires very tight integration between Operation Technology (OT) and Information Technology (IT). Many other factors must be considered, such as cybersecurity, network architecture, data management, field technician support, etc. Processes need to be defined, and roles and responsibilities need to be assigned before the deployment of the edge devices can be fully supported. The following support functions need to be considered.

- Procurement
- Device commissioning
- Device Monitoring and Maintenance
- User Interface (UI) Development
- New Technology

References

Bathelt, A., Ricker, N. L., & Jelali, M. (2015). Revision of the Tennessee Eastman process model. *IFAC-PapersOnLine*, *48*(8), 309-314. doi: 10.1016/j.ifacol.2015.08.199.

Cahill, J. (2016, February 18). Maximizing Production through Optimized Gas Lifting. https://www.emersonautomationexperts.com/2016/industry/oil-gas/maximizing-production-optimized-gas-lifting/

Diehl, F. C., Almeida, C. S., Anzai, T. K., Gerevini, G., Neto, S. S., Von Meien, O. F. ...Trierweiler, J. O. (2018). Oil production increase in unstable gas-lift systems through nonlinear model predictive control. *Journal of Process Control*, *69*, 58-69. doi: 10.1016/j.jprocont.2018.07.009.

Epiclift.com. (n.d.). How It Works. https://epiclift.com/how-it-works/

Fandi, R. A. (2015, September). A simple-effective analytical model for multi-well gas-lift allocation optimization 2015IFEDC. *International Field Exploration and Development Conference (IFEDC 2015),* 1-7. IET. doi: 10.1049/cp.2015.0579

Irons-Mclean, R., & Greengrass, J. (2016, September). The Internet of Things IoT and security: A practical strategy to secure an OT and IT integrated process control domain. *SPE Intelligent Energy International Conference and Exhibition*. OnePetro. https://www.onepetro.org/conference-paper/SPE-181002-M. DOI: 10.2118/181002-MS

Jing, S., Errouissi, R., Al-Durra, A., & Boiko, I. (2015, September). A data-driven subspace predictive controller design for artificial gas-lift process. *IEEE Conference on Control Applications (CCA)*, 1179-1184. IEEE. doi: 10.1109/CCA.2015.7320772

Kanu, E. P., Mach, J., & Brown, K. E. (1981). Economic approach to oil production and gas allocation in continuous gas lift (includes associated papers 10858 and 10865). *Journal of Petroleum Technology*, *33*(10), 1887-1892.

Luo, H., Zhao, H., & Yin, S. (2018). Data-driven design of fog-computing-aided process monitoring system for large-scale industrial processes. *IEEE Transactions on Industrial Informatics*, *14*(10), 4631-4641. doi: 10.1109/tii.2018.2843124

Mahdavi, M. (2017). Guest editorial: Ushering in a new era of oilfield innovation with the Internet of Things. *Journal of Petroleum Technology*, *69*(07), 14-15. doi: 10.2118/0717-0014-JPT

Miresmaeili, S. O. H., Zoveidavianpoor, M., Jalilavi, M., Gerami, S., & Rajabi, A. (2019). An improved optimization method in gas allocation for continuous flow gas-lift system. *Journal of Petroleum Science and Engineering*, *172*, 819-830. doi: 10.1016/j.petrol.2018.08.076

Ni, J., Ren, Z., & Mao, F. (2012, October). Improved feedback control system of unstable gas-lift wells. *5th International Conference on BioMedical Engineering*

and Informatics (pp. 1240-1244). IEEE. doi: 10.1109/BMEI.2012.6512943

Rashid, K., Bailey, W., & Couët, B. (2012). A survey of methods for gas-lift optimization. *Modelling and Simulation in Engineering*, *2012*. doi: https://doi.org/10.1155/2012/516807

Wanasinghe, T. R., Gosine, R. G., James, L. A., Mann, G. K., De Silva, O., & Warrian, P. J. (2020). The internet of things in the oil and gas industry: a systematic review. *IEEE Internet of Things Journal*, *7*(9), 8654-8673. doi: 10.1109/JIOT.2020.2995617

Biographies

DENISE SHERROD is the Chief of Automation at Occidental Petroleum Corporation in Texas. She graduated from The University of Texas Permian Basin with a Bachelor of Business Administration and Management degree, and from Texas A&M University with a Master's in Engineering Technical Management. Ms. Sherrod may be reached at Denise_Sherrod@oxy.com

BEN ZOGHI is the Victor H. Thompson chair and Professor of Electronics System Engineering Technology at Texas A&M. He directs the RFID/Sensor Lab and the new online professional Master of Engineering in Technical Management program. A member of the Texas A&M University faculty for 30 years, he has distinguished himself as a teacher, writer, and researcher, and has been honored for his teaching excellence by the College and the Texas A&M University Association of Former Students. Ben's academic and professional degrees are from Texas A&M (PhD), The Ohio State University (MSEE), and Seattle University (BSEE). Dr. Zoghi may be reached at zoghi@tamu.edu

DEVELOPMENT OF MULTI-MOBILE ROBOT SYSTEMS FOR ROBOTICS CLASSROOM LEARNING

Wutthigrai Boonsuk, Eastern Illinois University

Abstract

Demand for multi-mobile robot systems has grown substantially for manufacturing production and supply chain logistics. Amazon warehouses, for example, have more than 200,000 mobile robots moving millions of different products for their picking and packing processes, since fast delivery has become Amazon's key competitiveness against other retailers. Thus, having a workforce well-versed in the technology behind multiple mobile robot control systems is essential in many industries. While commercial systems are available on the market, multi-mobile robot control systems are too complex and costly for classroom learning. In this study, an ESP-SQL Bot system was developed specifically for classroom settings. This system was intended to be simple to construct and control at a reasonable cost. The system employed an SQL database server as a command center that stored a sequence of instructions for controlling robots. An ESP8266 module on each robot was used as a cost-effective microcontroller for retrieving instructions from the database server over a Wi-Fi network and utilized the instructions to operate the robot. A visualization tool, such as ROS Visualization or RViz, was used for tracking multiple mobile robots in a virtual space. The system developed in this study can be used to assist educators, including engineering-technology and technology instructors, in creating such a system for classroom teaching, while also allowing students to develop more complex control systems.

Introduction

A mobile robot is a robot equipped with motorized wheels or legs that allow it to move in a given environment. This type of robot has been utilized in various applications to aid human operations ranging from industrial processes, surveillance, and exploration to household cleaning tasks. The key advantages of a mobile robot are the unit's flexibility and mobility. Mobile robots can also be equipped with a variety of sensors to detect or monitor their surroundings. A group of mobile robots is often utilized to perform predefined tasks, because multiple robots yield better efficiency than a single robot. For instance, the amount of time to complete warehouse operation tasks—such as transferring materials from one area to another—can be reduced dramatically by increasing the number of robots in the process, leading to an improvement in production cycle time (an actual throughput). Amazon, one of the largest online e-commerce companies, has implemented a warehouse mobile robot system for order picking in its distribution centers to help workers process more than 10,000 daily online orders (Li & Liu, 2016).

With such mobile robot systems, the overall efficiency of Amazon's warehouse has improved by at least 3.5 times over its traditional picking process (Li & Liu, 2016). Due to its benefits and wide utilization, the multiple mobile robot is an interesting and exciting subject for students in applied engineering and technology-related fields. Providing the opportunity to learn this technology in the classroom not only helps students develop essential robotics skills, but also supports their attention to and engagement with classroom lessons. However, teaching the technology behind multiple mobile robot systems can be challenging, due to the cost of commercial robots and the complexity of their control systems. Thus, the author or this study aimed to develop a simpler and more comprehensible approach that could be easily adopted for teaching this subject to students in robotics classrooms. The proposed ESP-SQL Bot system focused on the following considerations: (a) simple construction and maintenance, (b) a simple control system, (c) expandability, and (d) reasonable cost. The proposed system in this study employed an ESP8266 Wi-Fi module as a microcontroller for an individual robot. Each robot received a series of instructions from a database server that was broadcast over a Wi-Fi network.

The ESP8266 module was an inexpensive microcontroller that integrated TCP/IP protocol for accessing a Wi-Fi network. Similar to an Arduino pin interface (Mehta, 2015; Parihar, 2019), the ESP8266 module has GPIO (general-purpose input/output) pins that can communicate with or control other electronic devices, such as LEDs, sensors, motor controllers, and displays. The ESP8266 has played a significant role in expanding the Internet of Things (IoT) platform in various applications, including robotics. There have been several notable examples of implementing ESP8266 or IoT platforms in robotics research. Zaini, Zaini, Latip, and Hamzah (2016) presented an IoT platform concept by equipping an RC car with a Raspberry Pi, a single-board computer, that was remotely driven by a user through a Wi-Fi network. The control interface was developed in a web browser that allowed the user to operate the car while viewing the live video feed from the camera attached to the car.

Another example of IoT-based surveillance robots is the InterBot 1.0 (Nayyar, Puri, Nguyen & Le, 2018). This robot utilized the ESP8266 module to transfer real-time data from sensors such as humidity, temperature, gas, and accelerometer over the internet. The data were then plotted and analyzed on an IoT web-based service, ThingSpeak.com (ThingSpeak for IoT Projects, n.d.). Similarly, an IoT robot designed for landmine detection used an Arduino microcontroller to capture sensor information and an ESP8266 to

enable a control interface on a smartphone through a cloud server platform (Ashokkumar & Thirumurugan, 2018). In addition, Uddin (2020) proposed a low-cost Wi-Fi robot to teach students about the robotics system and IoT platform. This robot utilized an ESP8266 to establish communication between the robot and a smartphone through a Wi-Fi network. Users were able to control the robot manually using a smartphone application. The author of this current study was inspired by these IoT robots. The proposed ESP-SQL Bot was equipped with an ESP8266 module for controlling the robot through a Wi-Fi network. However, manual control using an IoT platform such as web browsers or smartphone interfaces was inefficient for operating multiple robots simultaneously. Thus, different control methods were required for multi-mobile robot systems.

Control methods for multi-mobile robots can be classified into local and global approaches (Issa & Rashid, 2019). The local approach utilizes proximity data from sensors such as radars, lidars, sonars, ultrasonic sensors, or cameras. While this approach may not need to specify a target destination, it involves sensing the surrounding environment and developing short-term paths for specific purposes, including generating maps, avoiding obstacles, or maintaining group formations. On the other hand, the global approach requires either known start and end locations or a complete map of the environment for path planning to reach the target. Several researchers did studies that combined local and global approaches for multi-mobile robot systems. For instance, Gao, Xin, Cheng, Liu, and Li (2018) proposed a multi-mobile robot navigation system for logistics using a simultaneous localization and mapping (SLAM) algorithm to create a 2D map of the environment. Then, Anytime Repairing A* (ARA*) and dynamic window approach (DWA) algorithms were used to find an optimal path from the starting location to the target destination.

Islami, Nurmaini, and Satria (2022) also proposed a multi-mobile robot control system using particle swarm optimization (PSO) and fuzzy logic algorithms for seeking target location and performing obstacle avoidance. Other path planning algorithms for controlling multi-mobile robots—such as genetic algorithm (GA), ant colony optimization (ACO), and simulated annealing (SA)—were reviewed by Zhang, Lin, and Chen (2018), Rubio, Valero, and Llopis-Albert (2019), and Zghair and Al-Araji (2021). These existing algorithms may be suitable for advanced topics in robotics, but they can be too complex for demonstration in the classroom. For this current study, the author employed only the global approach by assuming that the map and locations of the robots were analogous to the real-world scenario, where the robots are used in industrial warehouses or manufacturing facilities. The proposed system utilized an SQL database server to store and distribute the sequence of instructions for multiple robot operations. In this paper, the author describes the design of the ESP-SQL Bot—including hardware specifications for classroom implementation—the framework of the control system for the multi-mobile robot using the SQL database server, the implementation of the robot operating system (ROS) and a visualization tool named Rviz (ROS Visualization) for tracking the positions of robots in an operating environment, and provides examples of adopting the ESP-SQL Bot for classroom learning.

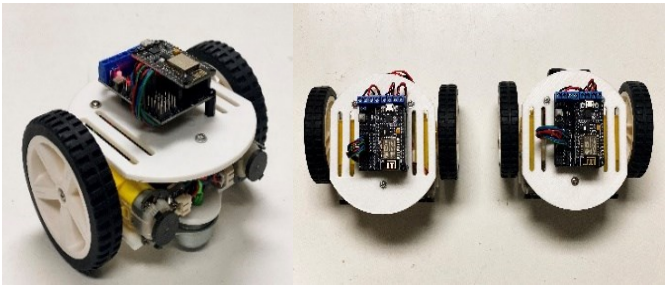
Robot Design

The design of the ESP-SQL Bot aimed at simplicity and maintainability. Table 1 lists the basic components and their approximate costs.

Table 1. Robot components and costs.

Component	Cost (in US dollars)
ESP8266 (NodeMCU ESP-12)	$7
Motor shield	$6
Three-wheel robot platform	$8
Micro-geared motors with encoders	$16
Batteries (rechargeable Li-ion 18650)	$15
Total:	$52

Figure 1 shows an assembly platform that consists of two 3D-printed plates, two rubber wheels, and a metal wheel caster. The ESP8266 module was aligned and inserted into a socket on the motor shield attached to the robot's top plate. Two micro-geared motors were connected to the shield. The encoders of the motors were connected to the I/O pins of the shield. Two rechargeable Lithium-ion 18650 batteries were used to supply 7.4 DCV to the shield. All components were easy to access, repair, and replace, if needed. Figure 2 illustrates the framework of the control system. The relational database, MySQL, was a database server serving as a command center that stored a series of instructions to control each robot. Users could add to and manipulate the instructions in the database. The "deploy" table in Figure 3 shows how the instructions were stored in a tabular format consisting of columns and rows.



(a) Robot assembly. (b) Multi-robots.

Figure 1. Design of the ESP-SQL Bot.

Control System

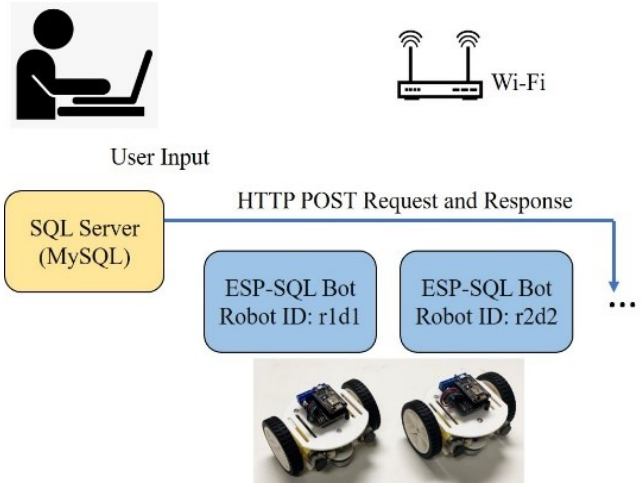


Figure 2. Framework of the ESP-SQL Bot system.

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default
1	id	int(11)			No	None
2	robot_id	varchar(10)	utf8mb4_general_ci		No	None
3	start	datetime(3)			Yes	NULL
4	end	datetime(3)			Yes	NULL
5	linear_vel	float			Yes	NULL
6	angular_vel	float			Yes	NULL

(a) Deploy table structure.

id	robot_id	start	end	linear_vel	angular_vel
627	r1d1	2022-09-22 08:35:33.743	2022-09-22 08:35:34.876	0	-1
628	r1d1	2022-09-22 08:35:34.876	2022-09-22 08:35:35.376	0	0
629	r1d1	2022-09-22 08:35:35.376	2022-09-22 08:35:40.563	0.1	0
630	r1d1	2022-09-22 08:35:40.563	2022-09-22 08:35:41.063	0	0
631	r2d2	2022-09-22 08:35:25.905	2022-09-22 08:35:27.467	0	1
632	r2d2	2022-09-22 08:35:27.467	2022-09-22 08:35:27.967	0	0
633	r2d2	2022-09-22 08:35:27.967	2022-09-22 08:35:35.824	0.1	0

(b) Deploy table data.

Figure 3. Deploy table in MySQL database that contains the robot ID, start time, end time, linear velocity, and angular velocity columns.

Each record contains the robot ID, start time, end time, linear velocity (in meters per second), and angular velocity (in radians per second). The robot retrieved the instruction records using a hypertext transfer protocol (HTTP) request. The HTTP is a transfer protocol for requesting and retrieving information from the server (Tukade & Banakar, 2018). This protocol was chosen for the current study because of its simplicity. However, this protocol may cause a delay in transactions. The delay scale depends on several factors, including payloads, the number of devices (robots), and system bandwidth. The impact of delay can be significant for applications requiring large payloads or a vast number of devices. Thus, other protocols, such as message queuing telemetry transport protocol (MQTT), constrained application protocol (CoAP), and advanced message queuing protocol (AMQP) might be better alternatives (Naik, 2017; Al-Masri, Kalyanam, Batts, Kim, Singh, Vo & Yan, 2020). Fortunately, the impact of delay using HTTP in this current study was minimal, due to small payloads and the number of devices given sufficient bandwidth.

The robot code run on the ESP8266 module consisted of three main functions: (1) establishing a Wi-Fi connection, (2) retrieving instructions from the database, and (3) executing instructions to control the robot. The ESP8266 module had a built-in Wi-Fi feature that could connect to a 2.4 GHz network. Figure 4 shows how the first part of the code allowed the module to establish a connection with the network. Once the connection was established, a different IP address was assigned to each robot.

```
const char* ssid = "ESPRouter";
const char* password = "pa$$w0rd";
String get_host = "http://192.168.1.37";
WiFiServer server(80);  // port 80 for server connection
WiFiClient client;

void setup()
{
  Serial.begin(115200); //initialize serial communication
  setupWiFi();
  setupMotor();
}

void setupWiFi()
{
  WiFi.begin(ssid, password); // establish Wi-Fi connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.println(WiFi.localIP());
  server.begin();
}
```

Figure 4. Code for establishing a Wi-Fi connection for the ESP8266 module.

The second part of the robot code allowed the ESP8266 module to retrieve instruction records from the MySQL database using an HTTP request. The code would send the request to the database server with the robot ID for verification every 50 milliseconds. The server then responded to the request with information for that specific robot ID. Figure 5 illustrated the code on the ESP8266 module (client) and the PHP code on the server. In the robot code, the robot ID was used for authentication when sending the request over Wi-Fi to the SQL server. The server then used this ID in the query to acquire information that would be sent back to the robot. Supposedly, the current time was within the start time and end time in the Deploy table. The robot would receive the

response that contained linear and angular velocities. Otherwise, the response would have been a string that contained either "No command" or "Error" text.

```
void deploy(String robot_id, String robot_name)
{
  WiFiClient client = server.available();
  HTTPClient http;
  String url = get_host+"/multirobot/deploy.php?robot_id="+robot_id;
  http.begin(url);
  int httpCode = http.GET();
  String payload = "";

  if (httpCode > 0) {
    payload = http.getString(); // retriving data from server
    if (payload != "No data") {
      String list[5];
      String temp_str = "";
      int index = 0;
      int str_length = payload.length();
      for (int i=0;i<=str_length;i++) {
         if (payload.charAt(i) == ',' || i == str_length) {
            list[index] = temp_str;
            temp_str = "";
            i++;
            index++;
         }
         temp_str += payload.charAt(i);
      }
      linear_velocity = list[0].toFloat();      // retrieved linear velocity
      angular_velocity = list[1].toFloat();     // retrieved angular velocity
      velocity_calculation(linear_velocity, angular_velocity, left_pwm_val, right_pwm_val);
      turnWheel(left_dir, left_pwm, left_pwm_val);      // move left wheel
      turnWheel(right_dir, right_pwm, right_pwm_val);   // move right wheel
    }
    else {
      stopWheel(); // stop both wheels
    }
  }
  else
    Serial.println("No response");

  http.end();
  delay(50); // send request every 50 ms
}
```

Robot ID

(a) Client code (robot).

```
<?php
    include_once('connect.php');
    $robot_id    =   $_GET["robot_id"];
    $sql = "SELECT linear_vel, angular_vel FROM deploy
            WHERE robot_id='$robot_id' AND now(3) >= start AND now(3) <= end";
    $result = $conn->query($sql);
    if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc())
        {
            echo $row['linear_vel'];
            echo ",";
            echo $row['angular_vel'];
        }
    }
    elseif ($result->num_rows == 0) {
        echo "No command";
    }
    else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }
    $conn->close();
?>
```

Robot ID

(b) Server code (MySQL).

Figure 5. Code for the HTTP request on the ESP8266 module and the PHP code on the MySQL server.

The ESP-SQL Bot became idle if it did not receive linear and angular velocities from the server. The movement of the robot was controlled by these velocities. When linear velocity was positive, the robot moved forward. On the other hand, if linear velocity was negative, the robot moved backward. Angular velocity controlled the rotation of the robot. Figure 6 demonstrates how a positive angular velocity turned the robot clockwise, and negative angular velocity turned it counterclockwise. Although the robot could be controlled using a combination of linear and angular velocities, in this study, the movement of the robot was simplified by turning the robot with angular velocity first and then moving the robot with linear velocity. For example, to move the robot in a 45-degree northeast direction, the robot would first turn counterclockwise 45 degrees (northeast) and then move forward in a straight line.

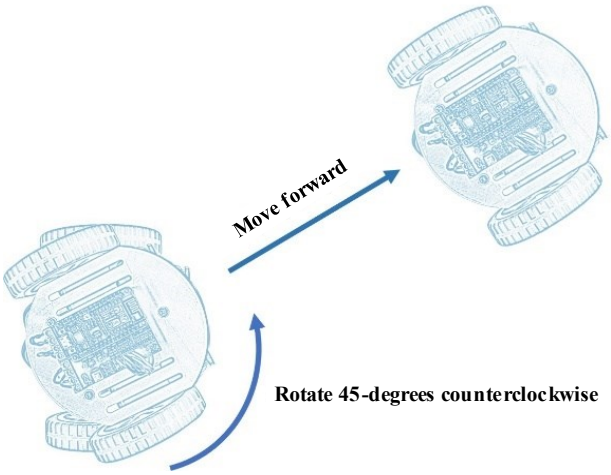


Figure 6. Robot movement using linear velocity and angular velocity.

A proportional-integral-derivative (PID) controller was used to maintain the speed of the robot. The method is comparable to the cruise control on a car, where constant speed is maintained even though the car goes up or down hills. Once the robot received the initial linear and angular velocities (v_i and ω_i) from the server, these velocities would be used in the code to compute the initial linear and angular speeds (S_i and $S\omega_i$) for the robot (see Equations 1 and 2). Then, the initial speeds of left and right motors (S_{LMi} and S_{RMi}) were calculated using Equations 3 and 4.

$$S_i = \left(v_i * 60\right) / \left(2\pi * r_{wheel}\right) \tag{1}$$

$$S\omega_i = \left(\omega_i * d_{wheel} * 60\right) / \left(4\pi * r_{wheel}\right) \tag{2}$$

$$S_{LMe} = S_i - S\omega_i \tag{3}$$

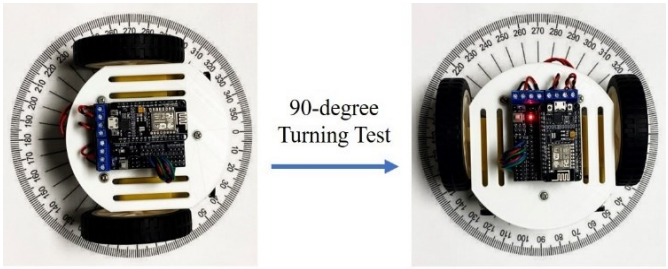
$$S_{RMe} = S_i + S\omega_i \tag{4}$$

where, S_i and $S\omega_i$ are the initial linear and angular speeds [in m/s and rad/s, respectively], v_i and ω_i are the initial linear and angular velocities [in m/s and rad/s, respectively], and S_{LMi} and S_{RMi} are the initial speeds for the left and right motors [in m/s]. Equation 5 is the PID equation implemented in the code. The library in C++ for the PID calculation is available in GitHub repository (Jimeno, 2016).

$$PID(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \tag{5}$$

where, K_p, K_i, and K_d are the constant coefficients for the proportional, integral, and derivative terms, and $e(t)$ is the error over time.

The coefficients K_p, K_i, and K_d are non-negative constant values that were manually calibrated for specific motors. Figure 7 shows the calibration test. Because the robot movement was simplified, linear velocity was zero when the robot turned clockwise or counterclockwise. For the same reason, angular velocity was zero when the robot moved forward or backward. In the 90-degree turning test, the angular velocity was set to 1 rad/s for 1.57 seconds. In the 30-cm distance test, the linear velocity was set to 0.1 m/s for 3 seconds.



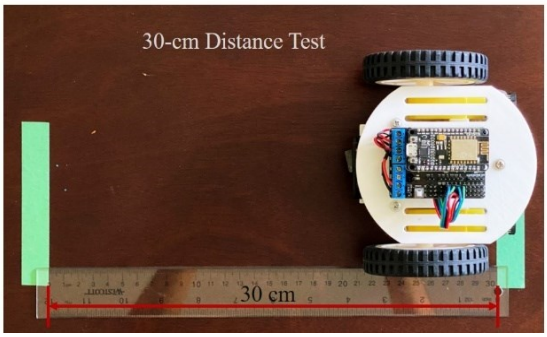


Figure 7. Distance and turning tests during the manual calibration process.

For the PID setup process of the ESP-SQL Bot, the coefficient constants had the values of $K_p = 2.5$, $K_i = 6.2$, and $K_d = 0.3$ were used. The error over time, $e(t)$, was calculated by finding the difference between the desired speeds (initial speeds (S_i , $S\omega_i$) and the measured speeds from the encoders attached to the robot's motors. As a result, the PID library calculated the pulse-width modulation (PWM) values that were used to control the robot's left and right motors. Figure 8 illustrates this code implementation.

Visualization

Visualization allows tracking the robots in the working space. In a multi-robot system, it is necessary to track the robots' movement, which can help with developing a schedule and debugging the control system. In this study, a robot operating system (ROS) framework and its visualization tool, RViz, were implemented for visualization purposes (ROS, n.d.). ROS is a de-facto open-source framework that provides a variety of libraries and tools to help developers create robotic applications. ROS has been used by many universities and companies for various applications, such as mapping, visualization, path planning, and building autonomous robots. The main function of ROS is to provide communication between computers and robots using ROS nodes. Each node is independent and interacts with other nodes by sending and receiving messages, which can contain data, commands, or other types of information. ROS topic is used to define the types of messages that will be sent to other nodes. When the node transmits data, it publishes the topic name and message type to be sent. Other nodes can receive messages by subscribing to the topic. ROS uses an odometry system to provide the estimated position of a robot based on its velocities. The velocity information can be acquired from different sources, including an inertial measurement unit (IMU), light detection and ranging sensor (LiDAR), and wheel encoders. In this study, the velocity information was only obtained from encoders attached to the motors of the ESP-SQL Bot. Figure 9 shows how to utilize this information for tracking the robot, as ROS requires publishing data to *nav_msgs/Odometry* and *geometry_msgs/TransformStamped* messages.

```
#include <PID.h>        // include PID library
#include <Encoder.h>    // include Encoder library
...

// Set PID constants
double K_P = 2.5;
double K_I = 6.2;
double K_D = 0.3;

// Initialize PID for each motor
PID left_pid(-1023.0, 1023.0, K_P, K_I, K_D);
PID right_pid(-1023.0, 1023.0, K_P, K_I, K_D);

// Initialize Encoders
Encoder left_encoder(left_encoderA, left_encoderB, count_per_rev);
Encoder right_encoder(right_encoderA, right_encoderB, count_per_rev);
...

void velocity_calculation(float linear_velocity_x, float angular_velocity_z,
int &left_pwm, int &right_pwm)
{
  float left_speed_encode = 0.0, right_speed_encode = 0.0;
  float left_speed = 0.0, right_speed = 0.0;
  float left_velocity = 0.0, right_velocity = 0.0;
  float dt, dx, dy;

  // Compute estimated speed of each motor from initial velocities
  float tan_velocity = 0.0, x_rpm = 0.0, tan_rpm = 0.0;
  tan_velocity = angular_velocity_z * (wheel_distance / 2);
  x_rpm = (linear_velocity_x / (2.0 * pi * wheel_radius)) * 60;
  tan_rpm = (tan_velocity / (2.0 * pi * wheel_radius)) * 60;
  left_speed = x_rpm - tan_rpm;
  right_speed = x_rpm + tan_rpm;

  // Compute actual speed from each encoder
  left_speed_real = left_encoder.getRPM();
  right_speed_real = right_encoder.getRPM();

  // Compute PWM value for controlling each motor by comparing
     the estimated speed to the actual speed
  left_pwm = (int)left_pid.compute(left_speed, left_speed_encode);
  right_pwm = (int)right_pid.compute(right_speed, right_speed_encode);

  ...
}
```

Figure 8. Code for PID implementation.

Since the data were based on the robot's velocities, the actual velocities were calculated from the encoder speed of the left and right motors using Equations 6-9.

$$v_{LM} = s_{LM} * \left(2\pi * r_{wheel}\right) / 60 \tag{6}$$

$$v_{RM} = s_{RM} * \left(2\pi * r_{wheel}\right) / 60 \tag{7}$$

$$v_{robot} = \left(v_{LM} + v_{RM}\right) / 2 \tag{8}$$

$$\omega_{robot} = \left(v_{LM} + v_{RM}\right) / d_{wheel} \tag{9}$$

where, v_{LM} and v_{RM} are the linear velocities of the left and right motors [in m/s], S_{LM} and S_{RM} are the actual speeds of the left and right motors from the encoders [in m/s], v_{robot} is the linear velocity of the robot from the encoders [in m/s], ω_{robot} is the angular velocity of the robot from the encoders [in rad/s], r_{wheel} is the radius of the wheel [in m], and d_{wheel} is the distance between the left and right wheels [in m].

```
// calculate the odometry values
left_velocity = left_speed_encode * ((2.0 * pi * wheel_radius) / 60.0);
right_velocity = right_speed_encode * ((2.0 * pi * wheel_radius) / 60.0);
linear_velocity_encode = (right_velocity + left_velocity) / 2.0;
angular_velocity_encode = (right_velocity - left_velocity) / wheel_distance;

dt = (float)(millis() - odom_prev_time) * 0.001f;
odom_prev_time = millis();

yaw_est += angular_velocity_real * dt;            // yaw estimated
dx = cos(yaw_est) * linear_velocity_real * dt;    // travelling distance in x
dy = sin(yaw_est) * linear_velocity_real * dt;    // travelling distance in y
geometry_msgs::Quaternion odom_quat = tf::createQuaternionFromYaw(yaw_est);

// publish nav_msgs to nav_msgs::Odometry odom
odom.header.frame_id = "r1/odom";
odom.child_frame_id = "r1/base_link";
odom.pose.pose.position.x += dx;
odom.pose.pose.position.y += dy;
odom.pose.pose.position.z = 0.0;
odom.pose.pose.orientation = odom_quat;
odom.header.stamp = nh.now();
odom.twist.twist.linear.x = linear_velocity_real;
odom.twist.twist.angular.z = angular_velocity_real;
odom.header.stamp = nh.now();
odom_pub.publish(&odom);

// publish geometry_msgs to geometry_msgs::TransformStamped td_odom
tf_odom.header.frame_id = "r1/odom";
tf_odom.child_frame_id = "r1/base_link";
tf_odom.transform.translation.x = odom.pose.pose.position.x;
tf_odom.transform.translation.y = odom.pose.pose.position.y;
tf_odom.transform.translation.z = odom.pose.pose.position.z;
tf_odom.transform.rotation = odom.pose.pose.orientation;
tf_odom.header.stamp = nh.now();
tf_broadcaster.sendTransform(tf_odom);
```

Figure 9. Robot code to publish *nav_msgs* and *geometry_msgs*.

In the robot code, the angular velocity from the encoders was used to compute the robot's heading direction (yaw estimated), and the linear velocity from the encoders was used to compute travelling distance in the *x*-direction (dx) and travelling distance in the *y*-direction (dy). The heading direction and travelling distance in *x*, *y* was updated over time, as the encoder speeds of the motors changed. A 3D model was created in Blender software (Blender, n.d.) to represent the ESP-SQL Bot in a virtual space in the RViz tool. ROS utilizes an XML file called unified robot description format (URDF) for specifying the geometry of robots. Figure 10a illustrates the geometry of the ESP-SQL Bot. Figure 10b shows how the two nodes, espsqlbot_r1 and espsqlbot_r2, were added to the ROS launch file using XML format. This launch file was necessary because it allowed mapping of the robot's geometry in the URDF file and establishing serial communication and transformation information for each robot. ROS visualization (RViz) was a 3D visualization tool that contained plugins for displaying different types of ROS messages, mainly data from sensors such as a camera, IMU, encoder, LiDAR, and laser scanner. Figure 11 shows how, for this study, the robot models were added to the RViz user interface once the launch file that created the espsqlbot nodes was executed. As the physical robot moved in a real environment, the position and orientation of 3D models of the robots in RViz were updated in real time.

```
<?xml version="1.0"?>
<robot name="espsqlbot">
  <link name="map" />

  <joint name="map_joint_r1" type="fixed">
    <parent link="map" />
    <child link="r1/odom" />
    <origin xyz="0 0 0" rpy="0 0 0" />
  </joint>
  <link name="r1/odom">
    <origin xyz="0 0 0" rpy="0 0 0" />
  </link>

  <joint name="map_joint_r2" type="fixed">
    <parent link="map" />
    <child link="r2/odom" />
    <origin xyz="0 0 0" rpy="0 0 0" />
  </joint>
  <link name="r2/odom">
    <origin xyz="0 0 0" rpy="0 0 0" />
  </link>


  <joint name="base_joint_r1" type="fixed">
    <parent link="r1/odom" />
    <child link="r1/base_link" />
    <origin xyz="0 0 0" rpy="0 0 0" />
  </joint>
  <link name="r1/base_link">
    <visual>
      <origin xyz="0 0 0.045" rpy="0 0 1.57" />
      <geometry>
        <mesh filename="package://espbot_description/meshes/espbot.dae" scale="1 1 1" />
      </geometry>
    </visual>
  </link>


  <joint name="base_joint_r2" type="fixed">
    <parent link="r2/odom" />
    <child link="r2/base_link" />
    <origin xyz="0 0 0" rpy="0 0 0" />
  </joint>
  <link name="r2/base_link">
    <visual>
      <origin xyz="0 0 0.045" rpy="0 0 1.57" />
      <geometry>
        <mesh filename="package://espbot_description/meshes/espbot.dae" scale="1 1 1" />
      </geometry>
    </visual>
  </link>

</robot>
```

(a) URDF file.

```
<launch>

  <param name="robot_description" textfile="$(find
  espbot_description)/urdf/espsqlbot.urdf" />

  <node ns="r1" name="espsqlbot_r1" pkg="rosserial_server" type="socket_node">
    <rosparam>
      port: 11411
    </rosparam>
  </node>

  <node ns="r2" name="espsqlbot_r2" pkg="rosserial_server" type="socket_node">
    <rosparam>
      port: 11412
    </rosparam>
  </node>

  <node pkg="tf2_ros" type="static_transform_publisher" name="r1_broadcast"
  args="0 0.5 0 0 0 0 1 map /r1/odom" />

  <node pkg="tf2_ros" type="static_transform_publisher" name="r2_broadcast"
  args="0 -0.5 0 0 0 0 1 map /r2/odom" />

</launch>
```

(b) ROS launch file.

Figure 10. URDF and launch files for the ESP-SQL Bots.

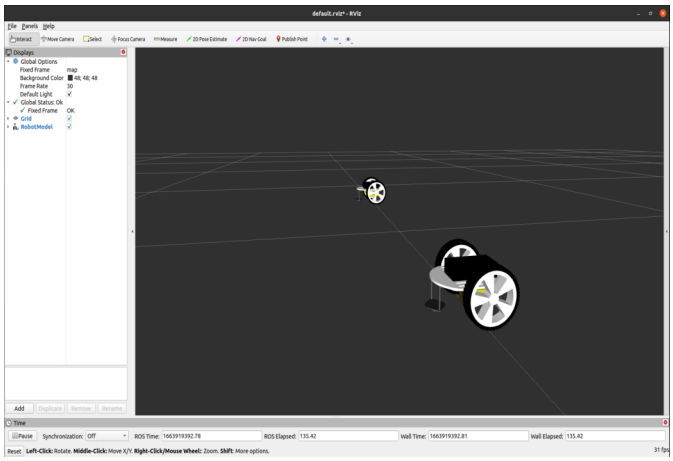


Figure 11. Displaying two ESP-SQL Bots in the RViz user interface.

Classroom Activities

The ESP-SQL Bot system can be used for students' hands-on learning activities in robotics classrooms. Students can participate in several activities through designing and developing the ESP-SQL Bot system, including being involved in the robot designing process, experimenting with tuning PID motor controls, and developing multi-mobile robot tasks and schedules. The first activity that allowed students to engage with the learning experience of using an ESP-SQL Bot system was to develop their own design for the robot. The instructor supplied the main components of the robot, such as two geared motors with encoders, the ESP8266 module and motor shield, and batteries. Students then developed a new design and fabricated it using a 3D printer to make a prototype of the robot. The challenge of this activity was that the new design configuration had to hold and secure all components together and allow the robot to move without any obstructions. This activity helped students gain knowledge of the functions of robot components and optimizing the robot design and the design process. Students were also more motivated and felt more ownership of their robots.

In the second activity, students experimented with tuning a proportional-integral-derivation (PID) control loop for the robot's motors. A PID is a closed control loop using feedback control that has been widely used in industrial control systems. An ROS plug-in called an RQT graph can be used as a manual tuning tool. In the experiment to obtain an optimum setting for the PID control loop, the integral and derivative coefficients (*Ki* and *Kd*) were first set to zero. Students increased the proportional gain coefficients (*Kp*) in small increments (e.g., 0.25 increments) until the motor began oscillating then increased *Kd* until the overshoot was compensated for and reduced to a minimal effect. Too much *Kd* can also cause overshoot. Finally, students increases *Ki* until all errors were eliminated and the motor speed stabilized. Manual PID tuning can be time-consuming and may not lead to the optimal solution; however, students can learn how closed-loop control works and practice through real experimentation that is not just presented as theory.

The last activity after completing the ESP-SQL Bot system was to develop tasks and schedules for the multi-mobile robot system. For instance, students built a small-scale warehouse system in which the robots picked up and dropped off materials to/from multiple workstations. Students came up with logistic solutions, such as predetermined routes and timetables, and then translated those data into instruction records stored in the database, including start time, end time, linear velocity, and angular velocity. Figure 12 illustrates an example of a warehouse system, where Robot1 would travel from WS1 (workstation 1) to WS5 and stop at WS2 and WS3. Robot2 traveled from WS1 to WS5 and made a stop at WS4. Tables 2 and 3 show the movements of Robot1 and Robot2, respectively, including their duration times, linear velocities, and angular velocities. Robot1 first turned counterclockwise 45 degrees and then moved two meters forward to WS2. It then turned clockwise 45 degrees and moved three meters forward to WS3. At WS3, the robot turned clockwise 30 degrees and moved 1.2 meters forward to its destination at WS5. For Robot2, it turned clockwise 10 degrees at WS1 and then moved three meters forward to WS4. At WS4, the robot turned counterclockwise 35 degrees and then moved 2.8 meters forward to WS5. Finally, students put final touches on optimizing workflows and efficiencies of the process.

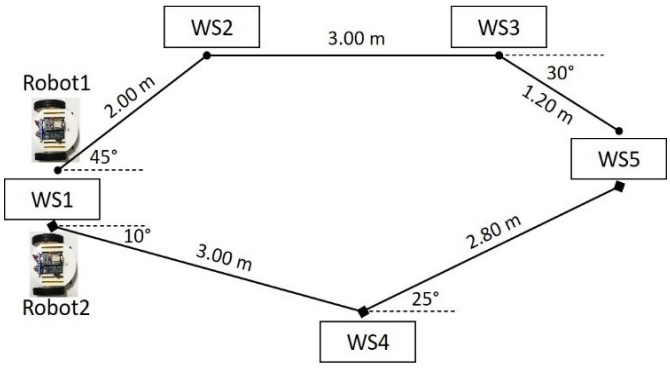


Figure 12. Small-scale warehouse system.

Table 2. Timetable for Robot1.

Movement (Robot1)	Time (sec)	Linear Vel	Angular Vel
Turn 45° (CCW)	1.57	0.0	-1.0
Forward	20.0	0.1	0.0
Turn 45° (CW)	1.57	0.0	1.0
Forward	30.0	0.1	0.0
Turn 30° (CW)	1.05	0.0	1.0
Forward	12.0	0.1	1.0

Table 3. Timetable for Robot2.

Movement (Robot2)	Time (sec)	Linear Vel	Angular Vel
Turn 10° (CW)	0.35	0.0	1.0
Forward	30.0	0.1	0.0
Turn 35° (CCW)	1.22	0.0	-1.0
Forward	28.0	0.1	0.0

Conclusions

A multi-mobile robot system (ESP-SQL Bot) utilizing an SQL database to store and distribute the instructions was developed in this study. In this paper, the author described the design, control systems, and visualization tool of the ESP-SQL Bot system aimed for educational purposes. The author also provided examples of how the system could be employed for teaching robotics in a classroom environment. Future work could include adopting the system into a robotics course and program curriculum. Student evaluation and feedback could be collected and reported for further improvement. Additionally, several improvements could be implemented on the ESP-SQL Bot system. For instance, future work could include developing user interfaces for easy path planning. Users could select start and end points on the map. All necessary data, including start time, end time, linear velocity, and angular velocity, could then automatically be included in the database for controlling the robots. Another potential improvement might be adding a proximity sensor to the robot to avoid obstacles and prevent collisions between robots.

References

Al-Masri, E., Kalyanam, K. R., Batts, J., Kim, J., Singh, S., Vo, T., & Yan, C. (2020). Investigating Messaging Protocols for the Internet of Things (IoT). *IEEE Access, 8*, 94880-94911. Doi: 10.1109/ACCESS.2020.2993363

Ashokkumar, M., & Thirumurugan, T. (2018). Integrated IOT based design and Android operated Multi-purpose Field Surveillance Robot for Military Use. *Advances in Engineering Research, 142*, 236-243.

Blender. (n.d.). Blender. https://www.blender.org

Gao, K., Xin, J., Cheng, H., Liu, D., & Li, J. (2018). Multi-Mobile Robot Autonomous Navigation System for Intelligent Logistics. *Chinese Automation Congress (CAC)*, 2603-2609. doi: 10.1109/CAC.2018.8623343

Islami, A., Nurmaini, S., & Satria, H. (2022). Comparative Analysis Multi-Robot Formation Control Modeling Using Fuzzy Logic Type 2 – Particle Swarm Optimization. *Computer Engineering and Applications, 11*(3), 167-176.

Issa, B. A., & Rashid, A. T. (2019). A Survey of Multi-mobile Robot Formation Control. *International Journal of Computer Applications, 181*(48), 12-16.

Jimeno, J. M. (2016). *Linorobot – PID library*. GitHub repository. https://github.com/linorobot/linorobot/tree/master/teensy/firmware/lib/pid

Li, J., & Liu, H. (2016). Design Optimization of Amazon Robotics. *Automation, Control, and Intelligent Systems, 4*(2), 48-52. doi: 10.11648/j.acis.20160402.17

Mehta, M. (2015). ESP 8266: a breakthrough in wireless sensor networks and internet of things. *International Journal of Electronics and Communication Engineering & Technology (IJECET), 6*(8), 7-11.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *IEEE International Systems Engineering Symposium (ISSE)*, 1-7. doi: 10.1109/SysEng.2017.8088251

Nayyar, A., Puri, V., Nguyen, N. G., & Le, D. N. (2018). Smart Surveillance Robot for Real-Time Monitoring and Control System in Environment and Industrial Applications. *Information Systems Design and Intelligent Applications*, 229-243. https://doi.org/10.1007/978-981-10-7512-4_23

Parihar, Y. S. (2019). Internet of Things and Nodemcu: A review of use of Nodemcu ESP8266 in IoT products. *Journal of Emerging Technologies and Innovative Research (JETIR), 6*(6), 1085-1088.

ROS. (n.d.). ROS – Robot Operating System. https://www.ros.org/

Rubio, F., Valero, F., & Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems, 16*(2). doi:10.1177/1729881419839596

ThingSpeak for IoT Projects. (n.d.). IoT Analytics – ThingSpeak Internet of Things. https://thingspeak.com/

Tukade, T. M., & Banakar, R. (2018). Data transfer protocols in IoT – An overview. *International Journal of Pure Applied Mathematics (IJPAM)*, 118, 121-138.

Uddin, N. (2020). A Development of Low-Cost Wi-Fi Robot for Teaching Aids. *Jurnal Infortel, 12*(2), 60-66.

Zaini, N. A., Zaini, N., Latip, M. F. A., & Hamzah, N. (2016). Remote monitoring system based on a Wi-Fi controlled car using Raspberry Pi. *Proceeding of the IEEE Conference on Systems, Process and Control (ICSPC)*, 224-229.

Zghair, N. A. K., & Al-Araji, A. S. (2021). A one-decade survey of autonomous mobile robot systems. *International Journal of Electrical and Computer Engineering (IJECE), 11*(6), 4891-4906. doi: 10.11591/ijece.v11i6.pp4891-4906

Zhang, H., Lin, W., & Chen, A. (2018). Path Planning for the Mobile Robot: A Review. *Symmetry, 10*(10), 450. https://doi.org/10.3390/sym10100450

Biography

WUTTHIGRAI BOONSUK is an Associate Professor of Engineering Technology at Eastern Illinois University. He earned his master's and doctorate degrees in Industrial and Manufacturing System Engineering from Iowa State University. Dr. Boonsuk also received his second master's degree in human-computer interaction from the same university. Dr. Boonsuk's research interests include manufacturing systems, rapid prototyping, robotics and automation systems, virtual reality, and geographic information system (GIS). Dr. Boonsuk may be reached at wboonsuk@eiu.edu

INSTRUCTIONS FOR AUTHORS: MANUSCRIPT FORMATTING REQUIREMENTS

The INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH AND INNOVATION is an online/print publication designed for Engineering, Engineering Technology, and Industrial Technology professionals. All submissions to this journal, submission of manuscripts, peer-reviews of submitted documents, requested editing changes, notification of acceptance or rejection, and final publication of accepted manuscripts will be handled electronically. The only exception is the submission of separate high-quality image files that are too large to send electronically.

All manuscript submissions must be prepared in Microsoft Word (.doc or .docx) and contain all figures, images and/or pictures embedded where you want them and appropriately captioned. Also included here is a summary of the formatting instructions. You should, however, review the sample Word document on our website (http://ijeri.org/formatting-guidelines) for details on how to correctly format your manuscript. The editorial staff reserves the right to edit and reformat any submitted document in order to meet publication standards of the journal.

The references included in the References section of your manuscript must follow APA-formatting guidelines. In order to help you, the sample Word document also includes numerous examples of how to format a variety of scenarios. Keep in mind that an incorrectly formatted manuscript will be returned to you, a delay that may cause it (if accepted) to be moved to a subsequent issue of the journal.

1. **Word Document Page Setup:** Two columns with ¼" spacing between columns; top of page = ¾"; bottom of page = 1" (from the top of the footer to bottom of page); left margin = ¾"; right margin = ¾".

2. **Paper Title:** Centered at the top of the first page with a 22-point Times New Roman (Bold), small-caps font.

3. **Page Breaks:** Do not use page breaks.

4. **Figures, Tables, and Equations:** All figures, tables, and equations must be placed immediately after the first paragraph in which they are introduced. And, each must be introduced. For example: "Figure 1 shows the operation of supercapacitors." "The speed of light can be determined using Equation 4:"

5. **More on Tables and Figures:** Center table captions above each table; center figure captions below each figure. Use 9-point Times New Roman (TNR) font. Italicize the words for table and figure, as well as their respective numbers; the remaining information in the caption is not italicized and followed by a period—e.g., "*Table 1*. Number of research universities in the state." or "*Figure 5*. Cross-sectional aerial map of the forested area."

6. **Figures with Multiple Images:** If any given figure includes multiple images, do NOT group them; they must be placed individually and have individual minor captions using, "(a)" "(b)" etc. Again, use 9-point TNR.

7. **Equations:** Each equation must be numbered, placed in numerical order within the document, and introduced—as noted in item #4.

8. **Tables, Graphs, and Flowcharts:** All tables, graphs, and flowcharts must be created directly in Word; tables must be enclosed on all sides. The use of color and/or highlighting is acceptable and encouraged, if it provides clarity for the reader.

9. **Textboxes:** Do not use text boxes anywhere in the document. For example, table/figure captions must be regular text and not attached in any way to their tables or images.

10. **Body Fonts:** Use 10-point TNR for body text throughout (1/8" paragraph indention); indent all new paragraphs as per the images shown below; do not use tabs anywhere in the document; 9-point TNR for author names/affiliations under the paper title; 16-point TNR for major section titles; 14-point TNR for minor section titles.

11. **Personal Pronouns:** Do not use personal pronouns (e.g., "we" "our" etc.).

12. **Section Numbering:** Do not use section numbering of any kind.

13. **Headers and Footers:** Do not use either.

14. **References in the Abstract:** Do NOT include any references in the Abstract.

15. **In-Text Referencing:** For the first occurrence of a given reference, list all authors—last names only—up to seven (7); if more than seven, use "et al." after the seventh author. For a second citation of the same reference—assuming that it has three or more authors—add "et al." after the third author. Again, see the *sample Word document* and the *formatting guide for references* for specifics.

16. **More on In-Text References:** If you include a reference on any table, figure, or equation that was not created or originally published by one or more authors on your manuscript, you may not republish it without the expressed, written consent of the publishing author(s). The same holds true for name-brand products.

17. **End-of-Document References Section:** List all references in alphabetical order using the last name of the first author—last name first, followed by a comma and the author's initials. Do not use retrieval dates for websites.

18. **Author Biographies:** Include biographies and current email addresses for each author at the end of the document.

19. **Page Limit:** Manuscripts should not be more than 15 pages (single-spaced, 2-column format, 10-point TNR font).

20. **Page Numbering:** Do not use page numbers.

21. **Publication Charges:** Manuscripts accepted for publication are subject to mandatory publication charges.

22. **Copyright Agreement:** A copyright transfer agreement form must be signed by all authors on a given manuscript and submitted by the corresponding author before that manuscript will be published. Two versions of the form will be sent with your manuscript's acceptance email.

    Only one form is required. Do not submit both forms!

    The form named "paper" must be hand-signed by each author. The other form, "electronic," does not require hand signatures and may be filled out by the corresponding author, as long as he/she receives written permission from all authors to have him/her sign on their behalf.

23. **Submissions:** All manuscripts and required files and forms must be submitted electronically to Dr. Philip D. Weinsier, manuscript editor, at philipw@bgsu.edu.

24. **Published Deadlines:** Manuscripts may be submitted at any time during the year, irrespective of published deadlines, and the editor will automatically have your manuscript reviewed for the next-available issue of the journal. Published deadlines are intended as "target" dates for submitting new manuscripts as well as revised documents. Assuming that all other submission conditions have been met, and that there is space available in the associated issue, your manuscript will be published in that issue if the submission process—including payment of publication fees—has been completed by the posted deadline for that issue.

    Missing a deadline generally only means that your manuscript may be held for a subsequent issue of the journal. However, conditions exist under which a given manuscript may be rejected. Always check with the editor to be sure. Also, if you do not complete the submission process (including all required revisions) within 12 months of the original submission of your manuscript, your manuscript may be rejected or it may have to begin the entire review process anew.

IJERI

The International Journal of Engineering Research & Innovation (IJERI) is the second official journal of the International Association of Journals and Conferences (IAJC). IJERI is a highly-selective, peer-reviewed print journal which publishes top-level work from all areas of engineering research, innovation and entrepreneurship.

IJERI Contact Information

General questions or inquiry about sponsorship of the journal should be directed to:

Mark Rajai, Ph.D.

Founding and Editor-In-Chief

Office: (818) 677-5003

Email: editor@ijeri.org

Department of Manufacturing Systems Engineering & Management

California State University-Northridge

18111 Nordhoff St.

Room: JD3317

Northridge, CA 91330