Quasi-Wireless Capacitive Power Transfer with Secure Data Acquisition for Robotic Systems in Space Infrastructure

T. Burks[§], G. Cathey[§], V. Kholodilo, D. Ulybyshev
Department of Computer Science
Tennessee Technological University
Cookeville, United States of America
tmburks42, glcathey42, vkholodil42, dulybyshev@tntech.edu

M. Pearce, T. Marcrum, M. Coultis, C. Van Neste
Department of Electrical and Computer Engineering
Tennessee Technological University
Cookeville, United States of America
mpearce, tgmarcrum42, mrcoultis42, cvanneste@tntech.edu

B. Northern, M. Gupta
Department of Computer Science
Tennessee Technological University
Cookeville, United States of America
banorthern42, mgupta@tntech.edu

D. Boyd
NASA, Marshall Space Flight Center
Huntsville, United States of America
darren.r.boyd@nasa.gov

***Abstract*—Space exploration is dependent on robotic systems that utilize end effectors to collect samples, probe surfaces, and manipulate objects. These systems can rarely be designed to do all three, forcing engineers to make tradeoffs based on the mission parameters - i.e. should the robotic appendage have a claw, drill, or shovel, and which would be best suited for the mission? Additionally, as more industrial and government entities partake in space exploration, data protection is needed. To address these challenges, we present a first-of-its-kind robotic linkage that has no wiring between the joints. Instead, quasi-wireless capacitive power transfer is used to send energy over the robot's chassis without a return wire. This enables the system to be completely modular through the use of single-contact permanent magnet connections, allowing rapid alterations in joint kinematics and/or the changing of end effectors.**

For collecting sensor data from the robotic arm and to send remote commands to it, we use a Supervisory Control and Data Acquisition (SCADA) system. Data transmission relies on MQTT and OPC UA communication protocols with encryption. The SCADA server logs and archives sensor data and provides the functionality for authorized users to send remote commands from SCADA client(s) to motors. A SCADA client can be any of the web browsers that connect to a server via a secure communication channel using SSL protocol. Furthermore, as an extra data protection mechanism, we inject noise to the sensor data traffic, which obfuscates the timing of sensor data packets and adds confusion about which data packet represents which motor.

***Index Terms*—wireless sensor networks, IoT security, space cybersecurity, SCADA systems**

I. Introduction

The infrastructure used in this paper is a prototype robotic arm with magnetically connected joints, powered by a quasi-wireless power transfer system [1]. This allows a robotic arm system with a lighter weight, greater versatility, and higher robustness. This would be very useful for space missions requiring a robotic arm with easily swapped end effectors or re-configurable kinematics, as it removes the need for contemporary multi-wire harnesses and connectors.

For collecting the sensor data and sending remote control signals, we deployed a Supervisory Control and Data Acquisition (SCADA) system and rely on the Message Queuing Telemetry Transport (MQTT) and the Open Platform Communications Unified Architecture (OPC UA) communication protocols. The SCADA server includes a historian that logs data and supports historical data visualization on a SCADA client. The SCADA client can be accessed in any major web browsers that connect to the SCADA server's Uniform Resource Locator (URL). Client's web browsers can run on smart phones, tablet computers, laptops, personal computers, and other computational devices. A password-based authentication mechanism allows authorized parties to view current sensor data and data archives from the client's web browser in the form of graphs and tables. The functionality to send remote commands to motors is also supported for authorized users. For example, the authorized user can send a remote command from the SCADA client to set an axis of our robot to 30 degrees. Motor angles are collected on an ESP32® board and then delivered in encrypted form via the MQTT protocol to the MQTT/OPC UA gateway that runs on a Raspberry Pi® Single Board Computer. This gateway provides MQTT-to-OPC UA payload conversion in both directions, and handles data encryption and delivery to the SCADA server. The MQTT virtual device shadow is used to provide a reliable point of communication when the ESP32® is not available. The Secure Sockets Layer (SSL) protocol between the SCADA server and client provides data confidentiality and integrity. Furthermore, we inject noise to the sensor data traffic to add an extra protection layer.

[§]Both authors contributed equally and are considered co-first authors.

The novelty of our proposed solution is as follows:

- Quasi-wireless power transfer system for a robotic arm, which provides a greater versatility, lighter weight, and higher robustness.
- Secure and scalable sensor data collection and remote control system for a robotic arm. Our solution works with wireless sensors and relies on MQTT and OPC UA communication protocols with an extra security mechanism.

II. Related Work

Robotic arms are common today, and are particularly seen in manufacturing and in remote manipulation tasks, such as NASA's Perseverance rover [2].

Wireless power techniques involving Inductive Power Transfer and Capacitive Power Transfer to power robotic arms have been explored to some degree [3, 4, 5, 6]. However, quasi-wireless capacitive techniques as discussed in this paper are a new idea for powering robotic arms.

Commercially available robotic arms are available from companies like FANUC®, which provide multi-axis robotic arms. FANUC® uses their own Teach Pendant (TP), R-30*i*B Plus® or R-30*i*B Mate Plus® controller, with an *i*pendant® for programming and data management. From a mechatronics standpoint, they use full copper wiring to power each axis through a servo motor. Our solution, in contrast, relies on wireless power transfer. The interface channel, depending on the FANUC® product, ranges from Serial via RS-232 and TelNet, to Ethernet via File Transfer Protocol, or high speed USB 2.0 with a Memory Card [7]. They can either be programmed through the TP or remotely using the KAREL programming language. For data acquisition and sending commands, FANUC® supports DeviceNet Protocol [8] for remote read and write access to String, Position, and Numeric Register objects.

Compared to other protocols, for example, the Modbus TCP® [9] communication protocol, the MQTT protocol fits better for IoT systems with limited energy resources. MQTT is an open, lightweight, easy to implement publish/subscribe transport protocol. These characteristics make it ideal for IoT devices where a small footprint is required and network bandwidth is limited. The protocol runs over Transmission Control Protocol/Internet Protocol (TCP/IP) or other network protocols that provide ordered, lossless, and bi-directional connections. MQTT includes the following features:

1) "Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
2) A messaging transport that is agnostic to the content of the payload.
3) Three qualities of service for message delivery:
   a) "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
   b) "At least once", where messages are assured to arrive but duplicates can occur.
   c) "Exactly once", where messages are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied." [10]

TABLE I. MQTT vs. OPC UA vs. Modbus TCP® Protocols

Protocol	MQTT	OPC UA	Modbus TCP®
Architecture	Publish-Subscribe	Client-Server	Master-Slave
Error Checking	Cannot check errors in the data payload	Yes	Yes
Quality of Service (QoS) Guarantees	Guarantees message delivery to the broker, but not to the subscriber	Guarantees message delivery if working on top of TCP/IP	Serial Modbus® protocol does not provide delivery guarantees, Modbus TCP® does
Access Control	Password-based access control to the broker	Can specify access to any tag for users	No access control embedded in the protocol
Data Sending Mode	Any time (publish-subscribe model)	Client can read and write data from/to server at any time	On-request sent by a master to a slave

According to the MQTT specification, the protocol itself does not offer a lot of security mechanisms. It only offers password-based access control. More advanced security mechanisms can be implemented using other protocols such as Transport Layer Security [10]. Since all communication happens through the MQTT broker, there is no point to point communication, so clients only need to know the broker's IP address. This makes it easier to change the network configuration on-the-fly. Most of the connection error handling is done by the TCP protocol. However, MQTT offers a 'keep alive' feature which improves connection failure handling.

Based on information from Table I and [11], the combination of MQTT and OPC UA is a good fit for our system, which needs low power consumption and a fine-grained access control. OPC UA protocol also scales well for large IoT systems and is supported by many SCADA systems.

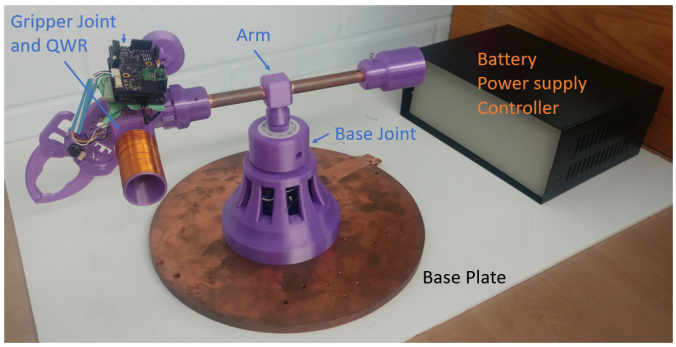
III. Core Design

A. Robotic Infrastructure

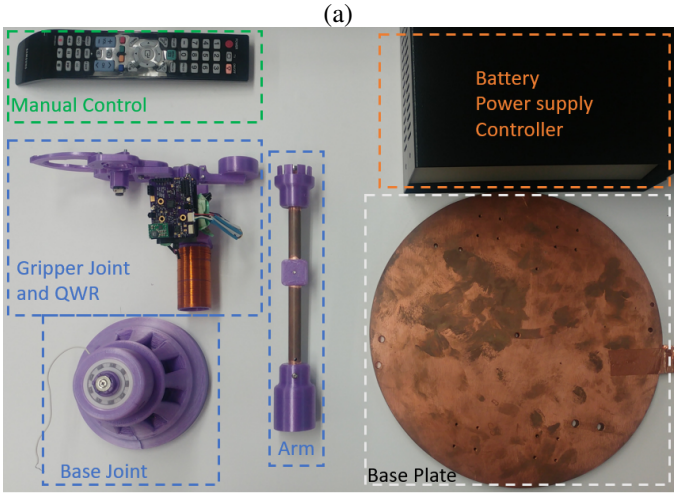
The overall arm system consists of five components, shown in Fig. 1(a), with a block diagram shown in Fig. 2.

The secure communication system installed on the ESP32® controls the signaling electronics. Bi-directional Infrared (IR) signaling of plaintext data is used to communicate with each of the arm joints, similar to that found in television remote controls. Motor angles are collected on the ESP32® board and then delivered in encrypted form to the Raspberry Pi®. Future work includes implementing space grade processors throughout the arm. Data workflow details are explained in the subsection III-B.

To power the arm, the inverter produces a High Frequency alternating current that operates at a frequency close to the



(a)



(b)

Fig. 1: Photo of prototype arm (a) assembled and (b) disassembled

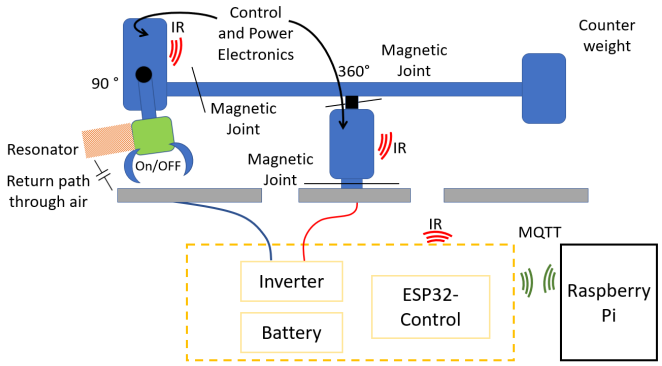


Fig. 2: Block diagram of prototype robotic arm infrastructure

6.78 MHz ISM band, and is attached to a connecting plate that consists of an outer ring and an inner plate. The outer ring simulates the overall vehicle chassis and is connected to the negative output of the inverter, while the inner plate simulates the arm connection point (which must be insulated from the chassis) and is connected to the positive output of the inverter. The robotic arm base attaches magnetically to the inner plate, allowing current to flow in the metallic arm. The power returns to the power supply capacitively through the outer ring. The quarter wave resonator helps amplify the voltage to provide a return path for the current. Thus, only one conductive connection is needed to transfer power throughout the whole arm, at an efficiency of greater than 50 % [1, 12].

The arm itself consists of three components (shown in blue in Fig. 1) that can be magnetically connected and disconnected, a base joint, a connecting arm, and an end effector. The base joint connects to the base plate and contains power and control electronics, and a motor which can spin the arm. The connecting arm is completely passive and consists of a copper pipe with a counterweight at one end, and a magnetic connection at the other end. The magnetic connection allows the end effector to be easily attached and detached. The end effector contains two motors, one for rotating the gripper, and one for opening and closing the gripper. It also contains the quarter wave resonator which allows the power to return back to the power supply.

B. Supervisory Control and Data Acquisition

The architecture of our data acquisition system is illustrated in Fig. 3. Motor angles are collected on an ESP32® board and then delivered in encrypted form, using MQTT protocol, to the MQTT/OPC UA gateway through the MQTT broker and the virtual device shadow, which run on a Raspberry Pi®. We use the wireless communication standard 802.11N with WPA2-PSK security mechanism for network communication between the ESP32® and Raspberry Pi®. On the Raspberry Pi®, the MQTT/OPC UA gateway converts the MQTT payload to an OPC UA payload. This gateway also handles the data encryption/decryption and data delivery to the "Ignition"® SCADA server. This server is used to archive sensor data, and to send remote commands from SCADA client(s) to motors via the OPC UA protocol. As a SCADA system, we selected Ignition® Maker Edition, which is a special edition of "Ignition"® that is free for non-commercial and educational use [13]. The center of the MQTT functionality lies in the virtual device shadow, or digital twin, shown in Fig. 4. The virtual device maintains a record of last received device state, and provides a layer of abstraction between layers in the architecture. It propagates desired states from OPC UA to the ESP32®, which, in turn, sends control signals to the arm via infrared Light-Emitting Diode (LED). Communication with the virtual device is modelled after AWS MQTT defined topics [14] for clarity and standard conformity. An additional topic has been added to return device connectivity status. This is done to inform the virtual device shadow when to send desired state signals to the ESP32®, and to inform "Ignition"® when the device is disconnected. Desired states are pushed to the virtual device via the '/update' topic, then compared to the last received reported state from the physical device, and designated as a delta if they differ. If the ESP32® is connected, these delta states are then sent to the device via the '/update/delta' topic to be resolved. Once the arm has moved to match the desired state, the ESP32® reports the movement as the new state, and the delta is removed from the virtual device shadow. At all times, the virtual device shadow accurately reflects the last known state of the device, its connectivity status, as well as the desired state of the arm. The virtual device

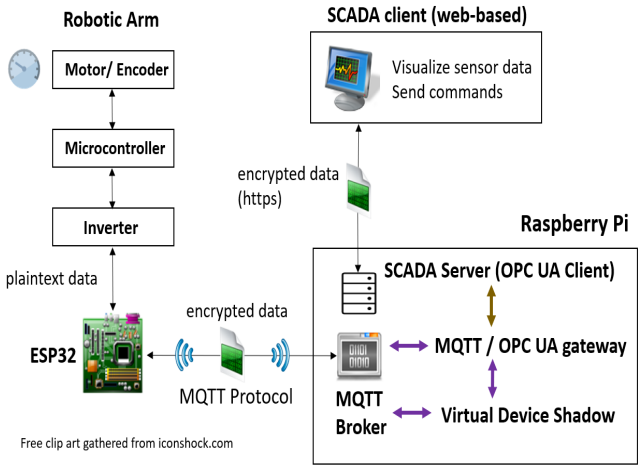


Fig. 3: Data Acquisition System Architecture

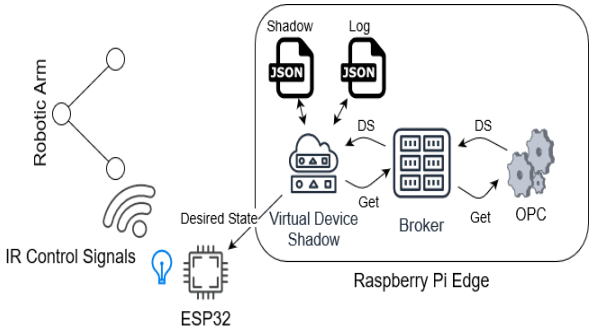


Fig. 4: MQTT Implementation

also logs device reported states, incoming desired states, and device connectivity status as they are received. Communication between the ESP32® and the arm incorporates infrared LED to enable wireless propagation of control signals. Each joint of the arm is individually addressable using different control signals generated by the ESP32® and sent by the LED. The power level supplied to the motors is also variable and adjustable using this communication method. These control signals are hexadecimal codes encoded and transmitted via the NEC protocol [15].

Virtual device shadow is installed on the Raspberry Pi®. It keeps communication between the Raspberry Pi® and the ESP32® alive. If the connection between them is lost, the virtual device shadow notifies the MQTT/OPC UA gateway. When the gateway receives this notification, it sets a 'communication_error' register to 'true' to trigger an alarm which is visible on the SCADA client's screen. This allows an operator to take necessary actions to fix the communication failure. When the connection is restored, the virtual device shadow sends the last desired state of the motors to the ESP32®. The virtual device shadow eliminates the need to implement complicated logic for communication failure recovery in the MQTT/OPC UA gateway, making the system easier to debug. The gateway is used to convert the payload received via MQTT protocol to OPC UA tags. This conversion works in both ways, as shown in Fig. 3. The gateway supports an event-based model which makes it efficient. The gateway subscribes to certain topics to receive messages from the virtual device shadow. When the message is received, it is decoded and mapped to OPC UA tags based on the configuration defined in a gateway configuration file. The gateway also subscribes to OPC UA tags which can be changed from the SCADA system. When any tag is changed, the gateway creates a MQTT payload based on the format defined in the gateway configuration file. This payload is then transferred to the virtual device shadow, which propagates it to the ESP32®.

We implemented encryption between the MQTT/OPC UA gateway running on the Raspberry Pi® and the OPC UA client that is built into the "Ignition"® SCADA server. It provides additional data protection if the gateway and the SCADA server run on different devices. "Ignition"® supports and recommends using the standard security policies that have been established by the OPC Foundation to secure OPC UA traffic [16]. The security policy we chose to use is Basic256Sha256 with the 'sign and encrypt' security mode. The Basic256Sha256 security policy is designed for systems that require high levels of security. This policy currently does not have an end date set by the OPC Foundation and NIST [17]. This means that there are currently no computationally feasible ways known to break the encryption algorithms used, so the only way to read the data is to have correct certificates and keys established by the Public Key Infrastructure. In "Ignition"®, a certificate has to be manually declared as trusted for each OPC UA connection. This means that "Ignition"® operators have to approve any OPC UA device that is trying to connect to the system, which prevents attackers from establishing a connection with the OPC UA client. The OPC UA protocol is used in our system for the following reasons:

1) It provides data delivery guarantees and flexible access control.
2) It supports remote commands that can be sent from a SCADA client.
3) Lots of modern SCADA systems support OPC UA.
4) It scales for big systems and allows easy additions of new devices.

For added security between the MQTT/OPC UA gateway and the OPC UA client that runs in "Ignition"®, 100 fake tags are randomly generated and sent to the OPC UA client from the gateway. These tags are updated at random time intervals between one and five seconds. The value assigned to the tags follows a discrete uniform distribution. The rotary motors in our system have a resolution of 5 degrees. This means there are 72 possible angles for motor 1: 0, 5, 10, 15, ..., 350, and 355 degrees. The probability for each angle is equal to 1/72 or 0.014. For motor 2 there are 19 possible angles: 0, 5, 10, ..., 85, 90. The probability for each angle is 1/19 or 0.053. In order to not confuse the operators of the SCADA system, these fake values are not visualized on the operating screens in the SCADA client. Fake values are also not stored in the historian on the SCADA server. However, if any attackers eavesdrop the communication channel, and are theoretically able to break our AES-256 encryption scheme and see sensor

data in plaintext, they will see the fake tags and the real tags, and must determine which are the tags representing the data read from motors. Fake data packets add confusion about which tag represents which motor, obfuscate the timing of sensor data packets, and thus, add an extra data protection layer against cyber attacks that rely on timing.

The SCADA server includes a historian that allows data archiving to a database, and visualization of collected sensor data in a SCADA client. Using historian, we visualize historical data from the different motors in the client in the form of graphs and tables. On-demand data reading, i.e. only when the sensor value changes, significantly increases the capacity of the archive and saves the network bandwidth. A SCADA client can be any of the web browsers that connect to a SCADA server's URL. A password-based authentication mechanism allows authorized users in the SCADA client to view current and historical sensor data, as well as to send remote commands to motors. For example, the command to change the motor 1 angle from 0 to 60 degrees can be issued by an authorized user. The SSL protocol is used between "Ignition"® SCADA server and SCADA client, as illustrated in Fig. 3, to provide data protection in transit. In our prototype, we use a self-signed SSL certificate for https communication. However, in a production environment, a SSL certificate from a trusted Certificate Authority should be deployed on the SCADA server. When a client sends a data request to the SCADA server, the server eventually sends back the SSL certificate which will be trusted by the client's web browser.

IV. Evaluation

To evaluate the data acquisition performance, in the first experiment we collected Round-Trip Time (RTT) by measuring the time starting from sending a remote command from the "Ignition"® SCADA server, and receiving the reported sensor value sent by the ESP32® to the SCADA server. RTT is the sum of times spent for remote command transfer from the SCADA server to ESP32®, OPC UA to MQTT conversion, and response reading and delivery back to the SCADA server from the ESP32® board via wireless communication channel. Wireless network delays are included in the RTT. This test excluded the robotic arm to ensure we were only evaluating the data acquisition and control components of the project.

We wrote a script in the Python programming language that changes the desired value tag of a motor every second. The historian in "Ignition"® writes the new desired value of the tag to the database, saving the value and the current system time. Once the desired value of the tag is changed, the command propagates from the "Ignition"® SCADA server to the MQTT/OPC UA gateway, then to the ESP32®. The ESP32® returns that value as the reported value tag. Whenever the reported value tag is updated on the ESP32®, it propagates through the system to the SCADA server. Once the SCADA server receives the new reported value tag, the historian writes it to the database, again storing the value and current system time. We subtracted the timestamp of the desired tag from the timestamp of the reported tag to determine the overall time it took for the system to send a command and receive the response from the ESP32®. Each iteration, one tag of four bytes, or one integer value, is updated. Below is the description of the system configuration where the first experiment was conducted:

Hardware: ESP32-WROVER®, Raspberry Pi®[1] Model 3B+
Raspberry Pi® CPU: Broadcom BCM2837B0 quad-core A53 (ARMv8™[2]) 64-bit @ 1.4GHz;
Raspberry Pi® RAM: 1GB LPDDR2 SDRAM;
Raspberry Pi® Network Interface: Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi;
Raspberry Pi® OS: Raspberry Pi® OS Lite, May 7, 2021
Software: Ignition Maker Edition™ version 8.1.1; Ignition Designer™ version 1.1.1; Eclipse Mosquitto™ (MQTT broker) version 2.0.11

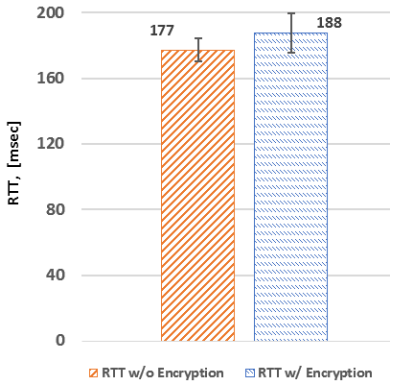


Fig. 5: Sensor Data Round-Trip Time (RTT)

We computed median and mean RTT values for 100 measurements. Desired values can only be changed once in 10 milliseconds. The left bar in Fig. 5 represents the mean value of RTT without encryption between the OPC UA server and client. The right bar represents the mean value of RTT when encryption was enabled. The 95% confidence interval for RTT without encryption is 7 milliseconds, the 95% confidence interval for RTT with encryption is 12 milliseconds. Data encryption adds 6% overhead to the RTT. Disabling encryption improved the average RTT by 11 milliseconds. The median RTT values with and without encryption are 184 and 173 milliseconds respectively.

In the second experiment, to test the performance of the OPC UA protocol, we wrote a script in Python that sends data from the OPC UA server to the OPC UA client 100 times. The OPC UA client and server were also written in Python using an open-source library "FreeOpcUa" [20]. We used timers available in Python to output the timestamp once data had been sent by the OPC UA server and received by the OPC UA client. Subtracting the server timestamp from

[1]Raspberry Pi is a trademark of the Raspberry Pi Foundation [18]

[2]AMBA, Arm, Arm ServerReady, Arm SystemReady, Arm7, Arm7TDMI, Arm9, Arm11, ... are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. [19]

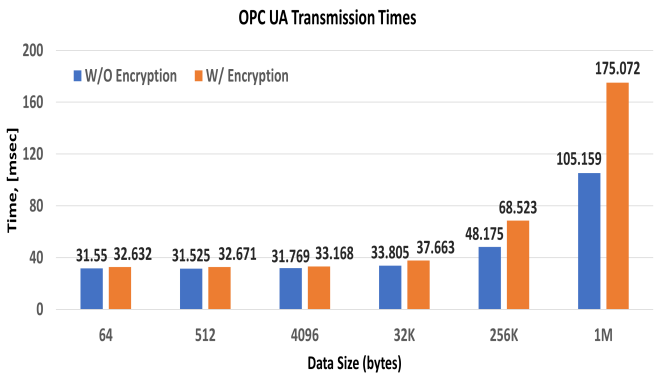


Fig. 6: Data Transmission Times via OPC UA protocol

the client timestamp gave us the time it took to transfer the data via OPC UA protocol. We then took the median value of the data transmission times. To remove the variable of network speed and delays, the OPC UA client and server were both running on the same Raspberry Pi 3®. The hardware specification of the Raspberry Pi® is similar to what we used in the first experiment. We first tested the data transmission time without encryption, then we performed another test where we timed how long it took to encrypt and decrypt the data. The encryption algorithm used was AES-CBC with a 256 bits key. In Fig. 6, the left bars represent the median time it took to transmit data without encryption and decryption. The right bars represent the total of the median values of the encryption, transmission, and decryption times. We varied transmitted data sizes from 64 bytes to 1 megabyte. The Y-axis represents the data transmission time in milliseconds. For transmitting 64 bytes of data, encryption adds 3% overhead. For transmitting 32 kilobytes of data, encryption adds 11% overhead, and for 1 megabyte of data, encryption adds 66% overhead.

V. Conclusion

A prototype robotic arm using magnetically connected joints and powered quasi-wirelessly has been shown to be remotely controllable. Secure control of robotic arms with reconfigurable joints allows for missions with great versatility and light weight, and should be given serious consideration. The combination of the MQTT and OPC UA communication protocols is a good match for our data acquisition system with wireless sensors that are powered via a wireless power transfer channel. Our solution provides sensor data protection in transit. The extra security mechanism based on data injection protects sensor data from cyber attacks that rely on timing and adds confusion about which data packet represents which motor of the robotic arm. Data encryption adds 6% overhead to the total Round-Trip Time for sensor data acquisition.

Acknowledgments

We thank NASA Marshall Space Flight Center for funding this research project and providing guidance. We thank Center for Energy Systems Research (CESR) and Cybersecurity Education Research and Outreach Center (CEROC) at Tennessee Technological University for providing resources.

References

[1] C. W. V. Neste, J. E. Hawk, A. Phani, J. a. J. Backs, R. Hull, T. Abraham, S. J. Glassford, A. K. Pickering, and T. Thundat, "Single-contact transmission for the quasi-wireless delivery of power over large surfaces," *Wireless Power Transfer*, vol. 1, no. 2, pp. 75–82, Sep. 2014, publisher: Cambridge University Press. [Online]. Available: https://www.cambridge.org/core/journals/wireless-power-transfer/article/singlecontact-transmission-for-the-quasiwireless-delivery-of-power-over-large-surfaces/70F11E56CE4DE43147D661FDBF2C7DA6

[2] "Mars 2020 Perseverance Rover - NASA Mars." [Online]. Available: https://mars.nasa.gov/mars2020/

[3] S. Ditze, A. Endruschat, T. Schriefer, A. Rosskopf, and T. Heckel, "Inductive power transfer system with a rotary transformer for contactless energy transfer on rotating applications," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 1622–1625, iSSN: 2379-447X.

[4] S. Kikuchi, T. Sakata, E. Takahashi, and H. Kanno, "Development of wireless power transfer system for robot arm with rotary and linear movement," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Jul. 2016, pp. 1616–1621.

[5] C. Zhang, D. Lin, and S. Y. R. Hui, "Ball-Joint Wireless Power Transfer Systems," *IEEE Transactions on Power Electronics*, vol. 33, no. 1, pp. 65–72, Jan. 2018. [Online]. Available: http://ieeexplore.ieee.org/document/7918527/

[6] A. Sarin, D. Abbot, S. Revzen, and A.-T. Avestruz, "Bidirectional Capacitive Wireless Power Transfer for Energy Balancing in Modular Robots," in *2020 IEEE Applied Power Electronics Conference and Exposition (APEC)*, Mar. 2020, pp. 852–859, iSSN: 2470-6647.

[7] "Fanuc robotics america corporation, fanuc robotics system r-30ia and r-30ib controller karel reference manual," accessed: June 11, 2021. [Online]. Available: https://icdn.tradew.com/file/201606/1569362/pdf/7066350.pdf

[8] "Devicenet™ unplugged – a view 'under the hood' for end users," accessed: July 1, 2021. [Online]. Available: https://www.rtautomation.com/technologies/devicenet/

[9] "Modbus organization, modbus protocol," 2021, accessed: January 08, 2021. [Online]. Available: www.modbus.org/specs.php

[10] "Oasis open, mqtt version 5.0." 2019, accessed: June 10, 2021. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

[11] F. C. D. S. SILVA, "Modbus tcp and its client-server model and mqtt and its publish-subscribe model: when to use each of them and what are their advantages and disadvantages?" 2019, accessed: June 12, 2021. [Online]. Available: https://www.novusautomation.com/site/default.asp?Idioma=1&TroncoID=053663&SecaoID=0&SubsecaoID=0&Template=../artigosnoticias/user_exibir.asp&ID=939463

[12] J. Dean, M. Coultis, and C. Van Neste, "Wireless sensor node powered by unipolar resonant capacitive power transfer," in *2021 IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (WoW)*, 2021, pp. 1–5.

[13] "Inductive automation, maker edition," accessed: June 13, 2021. [Online]. Available: https://docs.inductiveautomation.com/display/DOC81/Maker+Edition

[14] "Aws, device shadow mqtt topics," accessed: June 12, 2021. [Online]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/device-shadow-mqtt.html

[15] "Altium, nec infrared transmission protocol," 2017, accessed: June 12, 2021. [Online]. Available: https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol

[16] "Inductive automation, ignition security hardening guide," 2020, accessed: June 10, 2021. [Online]. Available: www.inductiveautomation.com/resources/article/ignition-security-hardening-guide

[17] "Opc foundation, opc ua basic256sha256 security policy," 2021, accessed: June 10, 2021. [Online]. Available: https://reference.opcfoundation.org/Core/docs/Part7/6.6.165/

[18] "Raspberry pi trademark rules and brand guidelines," accessed: June 12, 2021. [Online]. Available: https://www.raspberrypi.org/trademark-rules/

[19] "Arm trademarks," accessed: June 12, 2021. [Online]. Available: https://www.arm.com/company/policies/trademarks

[20] "Free opc-ua library," accessed: June 11, 2021. [Online]. Available: https://github.com/FreeOpcUa