Human Factors in Design, Engineering, and Computing, Vol. 159, 2024, 2335–2344
https://doi.org/10.54941/ahfe1005803

Self-Adapting, Integrated Apparel Robot for Responsive Wearable Comfort Utility

Minji Lee[1] **and Eric T. Matson**[2]

[1]Technology, Purdue University, West Lafayette, IN 47906, USA
[2]Computer and Information Technology, Purdue University, West Lafayette, IN 47906, USA

ABSTRACT

This research focuses on developing a prototype apparel system that integrates intelligent autonomous agents, human-based sensors, wireless networks, a mobile app, and a small-scale zipper robot. The goal is to create a practical assistive device that dynamically adjusts to user needs, particularly for the elderly and those with self-care challenges. Unlike existing wearable technologies that are overly technical, this system prioritizes usability and adaptability. It enables autonomous control of zipper speed and direction based on user profiles, enhancing comfort and functionality. Initial testing demonstrated effective communication between the zipper robot and mobile app, with adaptive adjustments and manual override features. The prototype serves as a proof-of-concept for future intelligent wearable devices, aiming to improve independence and quality of life.

Keywords: Robotics, Wearable technology, Intelligent agents, Assistive devices, Autonomous control

INTRODUCTION

The rapid development of wearable technology has transitioned from a conceptual vision to a reality, offering stylized products for the general public and those with assistive needs. Key drivers include lower electronic costs, widespread mobile access, and growing demand for fitness-enhancing devices. For instance, Meta's Ray-Ban glasses (Waisberg et al., 2024) blend fashion and tech, showcasing the potential of wearables (Lee et al., 2016). These innovations not only improve daily life but hold promise for aiding individuals with disabilities (Baig et al., 2019).

According to the CDC (Centers for Disease Control and Prevention, 2016) and U.S. Census Bureau (U.S. Census Bureau, 2016), self-care tasks like dressing are common challenges for those with disabilities, particularly the elderly. Zipping garments, especially those with back zippers, can be difficult for these individuals. However, current designs lack intelligent control and adaptability, making it difficult to change modes or stop mid-operation.

This research is an extension of the work presented in the paper (Lee, 2021), which focused on the development of an intelligent self-adapting apparel system to improve comfort utility. In this paper, we expand on that foundational work by focusing on a smarter, user-friendly zipper robot using a multi-agent system. Powered by an ESP8266 microcontroller (Mesquita et al., 2018), which supports Wi-Fi and real-time communication, the system enables intuitive control via a mobile app. The Message Queuing Telemetry Transport (MQTT) protocol (Yassein et al., 2017) ensures efficient communication, allowing users to easily zip or unzip garments (Atmoko Riantini & Hasin, 2017). The system provides real-time feedback, adjusting zipper speed and direction based on user input, enhancing usability for people with physical disabilities. This approach has the potential to positively impact independent living for people with physical disabilities by providing them with greater autonomy in performing daily tasks, ultimately improving physical comfort and quality of life.

A single-agent system involves a single agent that typically operates autonomously within its environment to complete various tasks. This agent has its own goals, makes decisions, and takes actions based on its perception of the environment (Shoham & Leyton-Brown, 2008). For individuals with physical disabilities and the elderly, controlling a zipper robot for tasks such as zipping and unzipping garments can be particularly challenging, especially when zippers are located in hard-to-reach areas like the back. Existing solutions often lack intelligent control, making it difficult for users to adjust the zipper mid-operation. To address this, the purpose of this study is to design and develop an intelligent zipper control system that enables users to manage zipper devices on clothing, such as shirts or dresses, via a mobile application. This system will incorporate user preferences, allowing for personalized control over the speed, direction, and stopping points of the zipper for optimal comfort and usability. Lightweight networking protocols and ESP8266 modules, widely used in the IoT industry, will be integrated to ensure fast and efficient communication between the zipper robot and the control app. Through seamless control offered by the mobile app, users can adjust the zipper's behavior in real time, enhancing both independence and ease of self-care.

Two specific questions are to be addressed in this work: (1) How can an intelligent agent system be designed to optimize the control and adaptability of a zipper robot for individuals with physical disabilities, enhancing both functionality and user experience? (2) What are the key factors in developing a stable and adaptive robot agent system for autonomous control of zippers, enhancing usability for individuals with physical disabilities?

Several approaches contribute to these objectives. For example, adaptive and personalized control can be achieved by incorporating user preferences, enabling real-time adjustments to speed, direction, and stopping points. This enables the system to adapt to each user's unique physical abilities and preferences, enhancing comfort and functionality. Also, integration of lightweight protocols such as MQTT ensures efficient communication between the robot and mobile application, even in low-resource environments, contributing to overall system stability. Finally, a user-centric design focusing on ease of use and portability is essential, ensuring that individuals can operate the device with minimal effort through an intuitive app interface and ergonomic hardware design. Together, these technologies form the foundation for creating a stable, intelligent, and adaptable zipper robot system that enhances independent living for people with physical disabilities.

LITERATURE REVIEW

The Robot Operating System (ROS) (Quigley et al., 2009) is a middleware framework that provides comprehensive services including hardware abstraction, device control, and message-passing (Macenski et al., 2022). Its extensive toolset, simulation capabilities, and large ecosystem have made it a standard platform in robotics research and academia (Park Delgado & Choi, 2020). While ROS 1 has limitations regarding real-time performance due to its lack of deterministic scheduling and execution guarantees (Blaß et al., 2021), ROS 2 has significantly improved these aspects through its DDS middleware and real-time capabilities. However, in resource-constrained applications like wearable devices, even ROS 2 may present challenges due to its computational overhead and system requirements. For lightweight applications requiring strict timing constraints and minimal resource usage, alternative approaches using protocols like MQTT or specialized industrial frameworks can offer advantages. This research implements a system based on the MQTT protocol and ESP8266 hardware, optimizing for both energy efficiency and real-time performance in resource-limited contexts. MQTT provides publish-subscribe messaging with minimal overhead, making it suitable for low-bandwidth, power-constrained applications. The ESP8266 microcontroller complements this approach with its dual-core architecture, integrated Wi-Fi connectivity, and sophisticated power management features including various sleep modes, making it particularly effective for battery-operated devices. The combination of MQTT and ESP8266 creates a lightweight system well-suited for specific real-time applications where minimal latency and power efficiency are paramount. While this approach doesn't offer the comprehensive development tools and advanced features of ROS, it excels in focused applications such as wearable assistive devices or portable robotics where resource constraints are a primary consideration.

Cliff (Baharom et al., 2016) is a zipper-assistance robot made of 3D-printed components powered by a 6V DC motor and two 3.7V LiPo batteries. The robot uses magnets to hold its chassis together and gears for smooth zipper motion. While this device is innovative, effective communication between the product and the user is critical for ensuring acceptance of assistive technologies both on an individual level and within society. Personalization is also identified as a key factor in reducing the stigma associated with wearable devices (Baharom et al., 2020). Just as individuals personalize their clothing to express identity and style, wearable robots like Cliff can benefit from customization options, allowing users to select products that align with their personal preferences and sense of identity. To address these limitations, this research prioritizes user-centered design principles, focusing on creating a device that is both technically efficient and adaptable to individual user needs.

IMPLEMENTATION AND METHODOLOGY

An efficient combination of hardware and software plays an important role in maximizing performance. This system architecture, as shown in Figure 1, represents an IoT-based solution where a wearable device is controlled through a central broker using the MQTT protocol. The Wearer interacts with a Device that is connected to the system via a Java Agent using MQTT Paho Client, which facilitates communication with the MQTT broker. The Wireless Router serves as the central communication hub, allowing data transmission between the broker, wearable device, and other components such as the ESP8266. The Raspberry Pi functions as the MQTT Central Broker, responsible for managing all communication between clients, including the wearable device and the ESP8266, which is equipped with a temperature and humidity sensor. The ESP8266 collects temperature and humidity data via the DHT22 and transmits this information to the Raspberry Pi, which acts as a central data processing and distribution node. Three solutions are implemented for subscribing to topics on the Raspberry Pi: (1) locally on the Raspberry Pi, (2) through the Java Paho Client, and (3) using MQTT Dash.

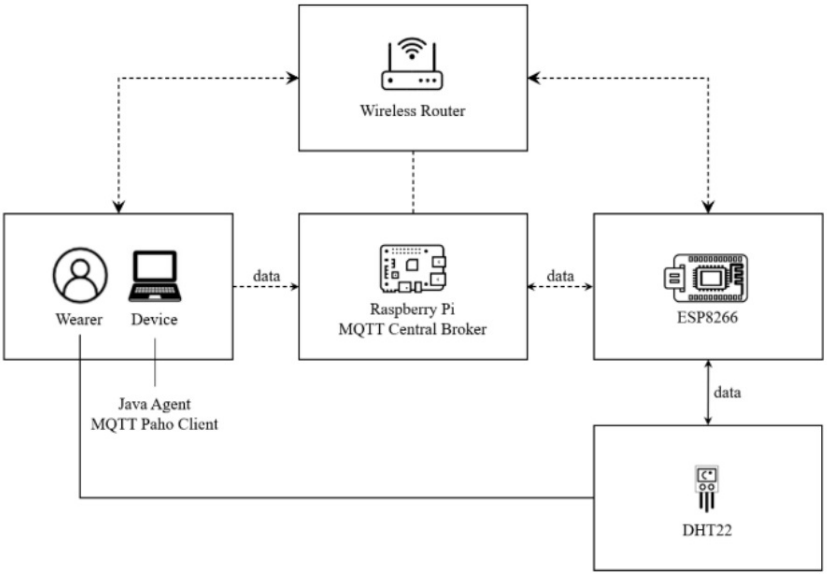


Figure 1: Intelligent zipper robot architecture.

The architecture enables real-time data exchange between the wearable device and environmental sensors, making it particularly useful for dynamic systems where adjustments need to be made based on environmental conditions. The lightweight MQTT protocol is optimized for IoT, ensuring efficient communication with minimal bandwidth, which is critical for resource-constrained devices like the ESP8266. The centralized Operating System (OS) control provided by the Raspberry Pi allows for scalability, enabling the addition of more sensors or devices to enhance the system's functionality. This system could be especially beneficial in applications such as smart garments, where wearable devices adjust automatically based on real-time data, improving user independence and usability.

In local communication, MQTT messages are published and subscribed to specific topics (such as “temp”) to control the zipper robot’s operation based on sensor data. The Java Graphical User Interface (GUI), developed in Eclipse using the PAHO client for MQTT (Selvi et al., 2020), provides a user-friendly interface with four essential features: name entry for user profiles, binary and variable zipper operations, and an autonomous mode that adjusts zipper movement based on environmental data like temperature and humidity. The Java classes, including the SourceAgent and its child classes (BinaryAgent, VariableAgent, and AutonomousAgent), handle the robot’s control in different modes, such as fully zipping/unzipping, sliding to a specified position, or adjusting automatically based on sensor feedback. The MQTT Dash application supports control over the system via an Android device, facilitating communication with various IoT components like ESP8266 and sensors, making it easy to publish and subscribe to MQTT topics to control the zipper robot. Through this integrated system, real-time data is communicated efficiently, allowing for dynamic operation of the wearable device.

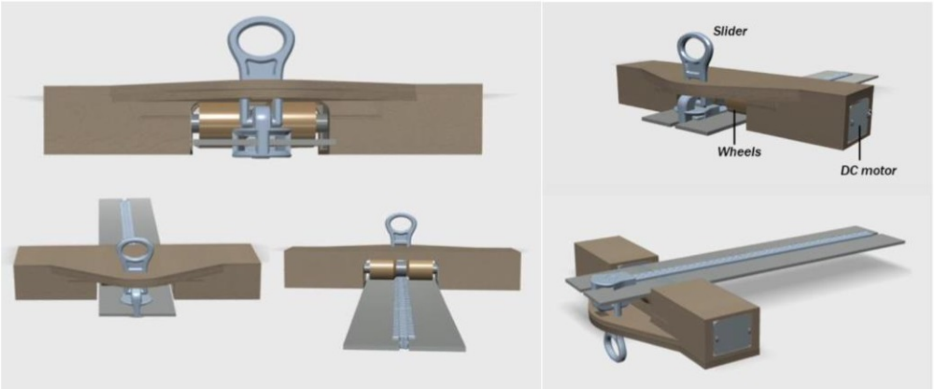
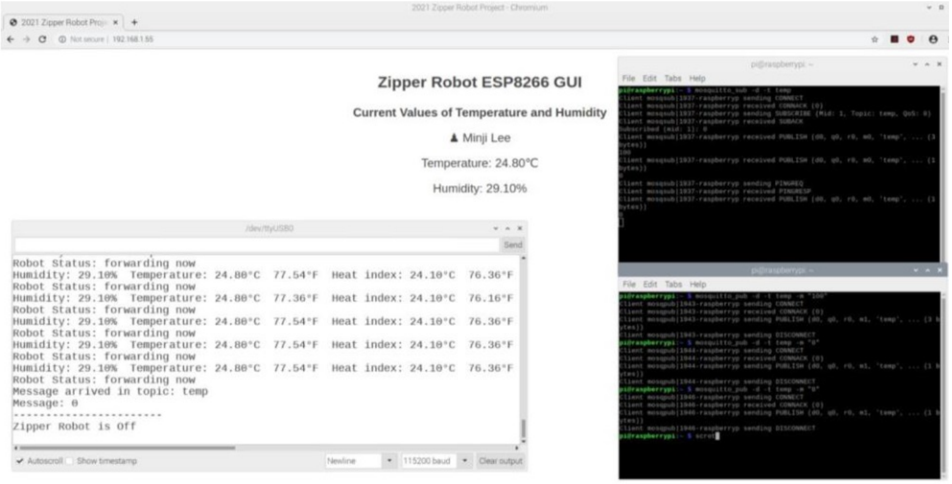


Figure 2: Prototype of 3d view of zipper robot design.

Figure 2 illustrates multiple views of a 3D-designed robotic zipper mechanism prototype. The design is a structure with integrated mechanical components including a slider mechanism and wheels for smooth movement along the zipper track. A distinctive ring-shaped pull handle is incorporated at the top of each view, providing an option for manual operation if needed. The top right view specifically highlights key components such as the slider, wheels, and a mounting position for a DC motor. The bottom right view demonstrates the extended rail system that ensures stable linear motion along the zipper path. This design will accommodate two DC motors and an ESP8266 microcontroller to be mounted on the top section, enabling automated control and movement of the zipper mechanism.

Figure 3: Core components and simulation.

Figure 3 demonstrates the practical implementation of the zipper robot system in a real environment. The left image shows the core electronic components: an ESP8266 microcontroller board mounted on a breadboard, connected to a relay module (shown in blue) for motor control. The right image displays the complete setup in action, featuring a black garment with an integrated zipper robot. The prototype is shown connected to a laptop, for programming and interface purposes, while the electronic components are wired to control the zipper movement. Red wires are visible running along the garment, connecting the control system to the zipper mechanism, demonstrating a working proof-of-concept for the automated zipper system. This system allows for real-world testing of the automated zipper functionality in an actual garment context, bridging the gap between theoretical design and practical application.



Zipper Robot ESP8266 GUI

Current Values of Temperature and Humidity

♟ Minji Lee

Temperature: 24.80°C

Humidity: 29.10%

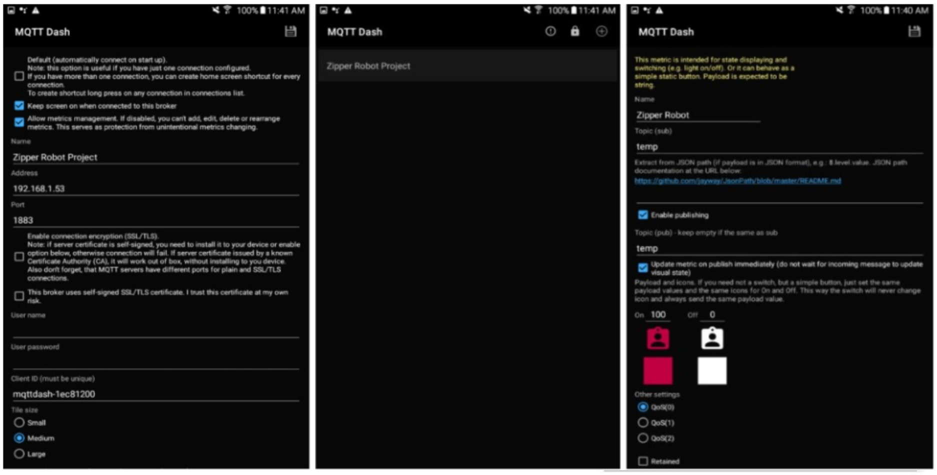
```
Robot Status: forwarding now
Humidity: 29.10%  Temperature: 24.80°C  77.54°F  Heat index: 24.10°C  76.36°F
Robot Status: forwarding now
Humidity: 29.10%  Temperature: 24.80°C  77.36°F  Heat index: 24.10°C  76.16°F
Robot Status: forwarding now
Humidity: 29.10%  Temperature: 24.80°C  77.54°F  Heat index: 24.10°C  76.36°F
Robot Status: forwarding now
Humidity: 29.10%  Temperature: 24.80°C  77.54°F  Heat index: 24.10°C  76.36°F
Robot Status: forwarding now
Humidity: 29.10%  Temperature: 24.80°C  77.54°F  Heat index: 24.10°C  76.36°F
Robot Status: forwarding now
Message arrived in topic: temp
Message: 0
-----------------------
Zipper Robot is Off
```

```
pi@raspberrypi:~ $ mosquitto_sub -d -t temp
Client mosqsub|1937-raspberryp sending CONNECT
Client mosqsub|1937-raspberryp received CONNACK (0)
Client mosqsub|1937-raspberryp sending SUBSCRIBE (Mid: 1, Topic: temp, QoS: 0)
Client mosqsub|1937-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub|1937-raspberryp received PUBLISH (d0, q0, r0, m0, 'temp', ... (3
bytes))
100
Client mosqsub|1937-raspberryp received PUBLISH (d0, q0, r0, m0, 'temp', ... (1
bytes))
0
Client mosqsub|1937-raspberryp sending PINGREQ
Client mosqsub|1937-raspberryp received PINGRESP
Client mosqsub|1937-raspberryp received PUBLISH (d0, q0, r0, m0, 'temp', ... (1
bytes))
0
```

```
pi@raspberrypi:~ $ mosquitto_pub -d -t temp -m "100"
Client mosqpub|1943-raspberryp sending CONNECT
Client mosqpub|1943-raspberryp received CONNACK (0)
Client mosqpub|1943-raspberryp sending PUBLISH (d0, q0, r0, m1, 'temp', ... (3 b
ytes))
Client mosqpub|1943-raspberryp sending DISCONNECT
pi@raspberrypi:~ $ mosquitto_pub -d -t temp -m "0"
Client mosqpub|1944-raspberryp sending CONNECT
Client mosqpub|1944-raspberryp received CONNACK (0)
Client mosqpub|1944-raspberryp sending PUBLISH (d0, q0, r0, m1, 'temp', ... (1 b
ytes))
Client mosqpub|1944-raspberryp sending DISCONNECT
pi@raspberrypi:~ $ mosquitto_pub -d -t temp -m "0"
Client mosqpub|1946-raspberryp sending CONNECT
Client mosqpub|1946-raspberryp received CONNACK (0)
Client mosqpub|1946-raspberryp sending PUBLISH (d0, q0, r0, m1, 'temp', ... (1 b
ytes))
Client mosqpub|1946-raspberryp sending DISCONNECT
pi@raspberrypi:~ $ scre
```

Figure 4: Real time data on web server.

Figure 4 figure displays real-time data on the web server hosted on the Raspberry Pi. The GUI shows the current temperature and humidity values, along with the operational status of the zipper robot. Additionally, terminal windows illustrate MQTT communication, where messages are being published and subscribed to, allowing the zipper robot to receive commands based on the temperature and humidity data.



☐ Default (automatically connect on start up).
Note: this option is useful if you have just one connection configured.
If you have more than one connection, you can create home screen shortcut for every connection.
To create shortcut long press on any connection in connections list.

☑ Keep screen on when connected to this broker

☑ Allow metrics management. If disabled, you can't add, edit, delete or rearrange metrics. This serves as protection from unintentional metrics changing.

Name: Zipper Robot Project

Address: 192.168.1.53

Port: 1883

☐ Enable connection encryption (SSL/TLS).
Note: if server certificate is self-signed, you need to install it to your device or enable option below, otherwise connection will fail. If server certificate issued by a known Certificate Authority (CA), it will work out of box, without installing to you device.
Also don't forget, that MQTT servers have different ports for plain and SSL/TLS connections.

☐ This broker uses self-signed SSL/TLS certificate. I trust this certificate at my own risk.

User name

User password

Client ID (must be unique): mqttdash-1ec81200

Tile size
○ Small
● Medium
○ Large

Zipper Robot Project

This metric is intended for state displaying and switching (e.g. light on/off). Or it can behave as a simple static button. Payload is expected to be string.

Name: Zipper Robot

Topic (sub): temp

Extract from JSON path (if payload is in JSON format), e.g.: $.level.value. JSON path documentation at the URL below:
https://github.com/jayway/JsonPath/blob/master/README.md

☑ Enable publishing

Topic (pub) - keep empty if the same as sub: temp

☑ Update metric on publish immediately (do not wait for incoming message to update visual state)

Payload and icons. If you need not a switch, but a simple button, just set the same payload values and the same icons for On and Off. This way the switch will never change icon and always send the same payload value.

On 100 Off 0

Other settings
● QoS(0)
○ QoS(1)
○ QoS(2)

☐ Retained

Figure 5: MQTT Dashboard Setup.

Figure 5 shows the configuration interface for the MQTT Dash application, used for managing and monitoring the Zipper Robot Project via MQTT protocol. The first screen on the left allows the user to set up the connection details, including the project name, the broker's IP address (192.168.1.53), and the port number (1883). The user can also configure SSL/TLS encryption options for secure communication, although the broker in this case uses a self-signed certificate. Additionally, users can specify client identifiers, adjust tile sizes, and keep the screen active during connection. The middle image shows the project listed under Zipper Robot Project on the MQTT Dash dashboard, indicating that the setup has been successfully configured for this project. The final image on the right provides the detailed settings for the Zipper Robot topic (temp), used for monitoring and controlling the robot. The user has set the topic to enable publishing and selected to update metrics immediately upon publishing without waiting for a response. The payload values represent the robot's states, such as "100" for turning on and "0" for turning off, with corresponding icons and colors to visually distinguish between these states. Additionally, the user can adjust other MQTT settings, such as Quality of Service (QoS), and choose whether to retain the last message sent by the broker. This setup demonstrates how the MQTT Dash app is used to manage and control the Zipper Robot by publishing and subscribing to the temp topic, making it easy for users to monitor the robot's status and send control commands from an Android device.

RESULT

The number of experiments for this research were numerous, especially over many prototypes and continuous improvement over a 1 year period. The results listed here are for the final prototype.

Table 1. Sample human systems integration test parameters (Folds et al. 2008).

Status	Sample Size	Angle Side	Length of Zipper (cm)	Amount of Times (avg)	Number of Stops (avg)	Number of Delay (avg)
Unzip	15	-10°	16	13.75	0.6	0.2
Unzip	15	+10°	16	35.20	5.8	3.3
Zip	15	+10°	16	30.10	4.76	2.2
Zip	15	-10°	16	14.95	1.1	0.86

Table 1 exhibits the summary results for this research in a human simulation experiment. With the complexity of the movement of the robot along a zipper, embedded in a light fabric shirt, several common errors are present; robot stopping due to binding and delays due to the physics of the zipper elevation and slippage of the tracks. In the table, the columns shown are the processes of zipping and unzipping with the robot. The final sample size was 15 executions. The angle is the rise or decline for the zipper of 10 degrees. The zipper length is 16 cm, which is a common standard garment zipper. The outcomes are the time to complete the zipping/unzipping process, the average number of stops during the execution, and the number of delays. All of the summary data is shown in the results table.

ANALYSIS

While the operation of the final prototype is not perfect, the behavior in a simulated situation does show good performance and function. Unzipping with a 10 degree decline results in average time of 13.75 seconds, 0.6 average stops and 0.2 average delays. The weight and gravitational effects made this operation effective. Unzipping along a positive 10 degree grade increased the time to 35.2 seconds, 5.8 average number of stops and 3.3 average delays. The decrease in performance characteristics is due to the mechanics of the robot, as the tracks often slip on the cloth surface, on a positive include, but in all cases the robot did complete the task.

Zipping along a 10 degree decline resulted in a 14.95 second average time, with 1.1 average stops and 0.86 average delays. Zipping along a 10 degree incline took 30.10 seconds on average, with 4.76 stops and 2.2 delays.

While all executions resulted in a successful zipper traverse the 4 tasks did show some difference in operation effectiveness. The declines, whether zipping or unzipping, showed better characteristics, due to gravitational effects on the small robot. The time to decline was less than 50\% of incline and the number of stops and delays was dramatically lower. Given the physical constraints of the operation, it is a logical and expected outcome. Due to that all 60 experiments were driven to completion, that is shows the operation can be successful, even if some of the velocity and delay outcomes vary. This data shows the successful operation and basic validity of the system and prototype.

CONCLUSION

The results from this prototype demonstrate the technology's feasibility and highlight its significant potential to support individuals with physical challenges and other diverse populations. By providing a tool that fosters independence, the system empowers individuals with disabilities to perform daily tasks with greater ease, reducing reliance on external assistance and enhancing overall quality of life. This solution not only addresses immediate accessibility needs but also serves as a foundation for future advancements in wearable assistive technology.

Figure 6: MQTT dashboard setup.

Looking ahead, future development will focus on refining the system through a user-centered design approach. This will involve enhancements in the intuitiveness of zipper functionalities, as illustrated in Figure 6, enabling users to control zipping and unzipping actions seamlessly. Additionally, gathering a broader range of user data will be essential for tailoring default settings to meet individual preferences more precisely. By integrating user feedback and analyzing interaction patterns, the system can be adapted to offer optimal speed, direction, and stopping points for diverse clothing types and user requirements, thereby maximizing usability and comfort. These ongoing improvements will help further extend the system's impact and accessibility, ensuring it remains an invaluable tool for fostering autonomy and self-care among users.

ACKNOWLEDGMENT

The authors would like to acknowledge the support and contributions of everyone involved in this research.

REFERENCES

Atmoko, R. A., Riantini, R., Hasin, M. K. (2017). IoT real-time data acquisition using MQTT protocol. Journal of Physics: Conference Series, vol. 853, no. 1. IOP Publishing.

Baharom, M. Z., et al. (2016). Cliff: the automatized zipper. In: GFC-Global Fashion Conference 2016, Stockholm, Sweden, pp. 20–21.

Baharom, M. Z., et al. (2020). Wearability and usability assessment of Cliff: An automatized zipper. Journal of Industrial and Intelligent Information, 8(1).

Baig, M. M., et al. (2019). A systematic review of wearable sensors and IoT-based monitoring applications for older adults—a focus on aging population and independent living. Journal of Medical Systems, 43, 1–11.

Blaß, T., et al. (2021). A ROS 2 response-time analysis exploiting starvation freedom and execution-time variance. In: IEEE Real-Time Systems Symposium (RTSS), pp. 1–12. IEEE.

Centers for Disease Control and Prevention. (2016). Prevalence of Disabilities and Health Care Access by Disability Status and Type Among Adults — United States, 2016. MMWR.

Lee, J., et al. (2016). Sustainable wearables: Wearable technology for enhancing the quality of human life. Sustainability, 8(5), 466.

Lee, M. (2021). Intelligent Self Adapting Apparel to Adapt Comfort Utility. MS thesis, Purdue University.

Macenski, S., et al. (2022). Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics, 7(66), eabm6074.

Mesquita, J., et al. (2018). Assessing the ESP8266 WiFi module for the Internet of Things. In: 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), vol. 1. IEEE.

Park, J., Delgado, R., Choi, B. W. (2020). Real-time characteristics of ROS 2.0 in multi-agent robot systems: an empirical study. IEEE Access, 8, 154637–154651.

Quigley, M., et al. (2009). ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software, vol. 3, no. 3.2.

Selvi, M., et al. (2020). Energy efficient and secured MQTT protocol using IoT. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 9(4), 11–14.

Shoham, Y., Leyton-Brown, K. (2008). Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, Cambridge.

U. S. Census Bureau. (2016). Disability | American Community Survey, 2016. Retrieved from https://www.census.gov/acs/www/data/data-tables-and-tools/american-community-survey/.

Waisberg, E., et al. (2024). Meta smart glasses—large language models and the future for assistive glasses for individuals with vision impairments. Eye, 38(6), 1036–1038.

Yassein, M. B., et al. (2017). Internet of Things: Survey and open issues of MQTT protocol. In: 2017 International Conference on Engineering & MIS (ICEMIS). IEEE.