IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# Energy Generation From Speed Breakers With Iot Monitoring

[1] Bhavya V, [2]B Tharun Kumar Reddy, [3]D Akhira Nandan, [4] Jami Adarsh, [5] Jetti Sreekar Desik
[1] Assistant Professor, [2] Student, [3]Student, [4]Student, [5]Student
[1]Computer Science and Engineering,
[1] Dayananda Sagar Academy of Technology & Management, Bengaluru, India

***Abstract:*** This research introduces the design, development, and testing of an extensive touch event logging system on top of the ESP32 microcontroller platform. The system records capacitive touch events through a special sensor on GPIO 4, keeps track of persistent event count in non-volatile storage, presents real-time count data on an onboard OLED display, and at the same time writes timestamped event data to Google Sheets via a custom web service. The system meets the requirement of efficient event detection using debounce protection measures and supports high-grade network communication to guarantee data integrity. This work describes the hardware architecture, firmware design paradigms, cloud integration methodology, and real-world applications of the developed system. Performance analysis shows that the system has accurate event counting and logging functions even under changing network conditions, with successful data synchronization to the cloud repository. The touch event logger is a general-purpose base for different Internet of Things (IoT) applications that need physical interaction tracking along with local visual feedback as well as remote data storage, for example, people counting, user interaction tracking, or mechanical movement tracking. In addition, this work investigates a possible application as an energy harvesting system based on rack and pinion motion conversion, as in speed breaker energy generation applications, validating the modularity of the ESP32 board for data logging as well as for renewable energy applications.

***Index Terms*** - ESP32, Touch Sensing, IoT, Google Sheets, Energy Harvesting, Microcontroller Applications

## I. INTRODUCTION

The emergence of Internet of Things (IoT) devices has revolutionized the way we engage with and collect information about our physical surroundings. From among the myriad applications, event logging systems capable of detecting, counting, showing, and remotely storing interaction data form a building block for more advanced IoT systems. Such systems facilitate the integration between physical interactions and digital repositories of data, providing the potential for sophisticated analysis and

remote monitoring. The ESP32 microcontroller has also become a strong platform for IoT development because it has WiFi built in, low power requirements, and a strong set of peripherals including capacitive touch functionality. With cloud services and local display technologies added on top of it, the ESP32 can be used as a full solution for many sensing and logging applications. This work is aimed at creating a complete touch event logging system that utilizes these features to offer real-time local feedback as well as storage in the long term for analysis. The inspiration for this work comes from the desire for efficient but simplified methods of monitoring physical interactions in different environments. Most conventional methods involve cumbersome hardware setup or specialized server infrastructure. With the use of the ESP32's innate touch sensing feature and integration into easily accessible cloud services such as Google Sheets, we illustrate a more accessible yet still reliable solution that minimizes the complexity of implementation.

Touch sensing technology is utilized in a variety of fields such as industrial control systems, consumer electronics, public interfaces, and educational tools. The inclusion of persistent counting functionality and cloud synchronization broadens the applications to cover long-term data collection situations such as foot traffic analysis, usage pattern monitoring, and interaction frequency tracking. The addition of an OLED display

enhances the system further by allowing real-time visual feedback, which is ideal for both attended and unattended operational scenarios. In addition, this paper discusses how this system could be extended to include energy harvesting from mechanical motion like the motion caused by speed breakers or similar rack and pinion systems. This extension illustrates how IoT logging devices can be made self-sustaining with proper integration with energy conversion systems, solving power consumption issues that are key in the case of remote deployments.

Our technical goal is to design a full system that:

1. Sensitively detects and counts touch events with capacitive sensing
2. Synchronously displays immediate visual feedback with an onboard OLED display
3. Persists counts with non-volatile memory through power cycles
4. Safely transmits timestamped event data to a cloud storage
5. Investigates possible integration with energy harvesting techniques

The importance of this work is that it shows a full end-to-end IoT solution employing commonly available components and services. Documenting design decisions, implementation issues, and performance attributes, this paper offers a template for corresponding systems that can be tailored for specific application needs.

The modular nature of the system ensures that components can be changed or swapped based on specific use scenarios without altering the fundamental functionality.

In the following sections, we first summarize literature on touch sensing technologies, applications of ESP32, and approaches to cloud integration. We then outline the methodology followed in the design of our system, ranging from hardware choice, firmware design, and cloud service setup. Results and performance analysis follow, after which limitations and possible applications are discussed. We conclude with findings from this work and propose avenues for future research in this area.

## II. LITERATURE REVIEW

Development of touch sensing and event logging systems has come a long way in the last ten years, spurred on by microcontroller advances, cloud computing, and the general IoT ecosystem. The present work is based on an analysis of the fundamental research and technological innovation that has led to the current work on ESP32-based touch event logging systems.

### 2.1 Capacitive Touch Sensing Technologies

Capacitive touch sensing pervades contemporary electronic interfaces, from smartphone displays to factory control panels. Walker et al. (2019) gave a detailed survey of the principles of capacitive sensing, including how changes in electrical capacitance may be detected in order to sense proximity or direct contact with a conductive surface like the finger of a human. Their paper stressed the advantages of capacitive sensing over resistive sensing for the needs of sensitivity and dependability in application. The ESP32 microcontroller includes specialized hardware for capacitive touch sensing, a feature fully described by Espressif Systems (2021) in their technical reference manual. This hardware supports touch detection at lower CPU overhead, making it well placed for use in battery-powered systems. Kumar and Patel (2020) compared the performance of ESP32's touch sensing features to bespoke touch controllers and found similar sensitivity with reduced rates of false detection when suitable filtering methods were used.

### 2.2 ESP32 Applications in IoT Systems

The ESP32 has become a leading IoT development platform owing to its integration of processing power, connectivity features, and peripheral capabilities. Zhang et al. (2022) surveyed applications of the ESP32 in IoT and established its extensive use across environmental monitoring, home automation, and wearables. According to their study, the microcontroller has the benefit of providing local processing along with wireless communication in applications calling for both. Bhatia and Sood (2021) explicitly examined ESP32's application suitability to edge computing, showcasing how its dual-core nature allows for concurrent sensor data processing and network communication. Such ability comes especially in handy in our touch event log system, where touch detection should go on unabated despite network transmission operations. Under human-computer interaction, Fernandez-Carames and Fraga-Lamas (2019) assessed some of these microcontroller boards, including ESP32, in the design of touch-based user interfaces. What they did to show the power efficiency and quick response time benefits of the ESP32 over microcontrollers with comparable prices is similar to our own choice of ESP32 for our touch event logger project.

### 2.3 Display Integration in Embedded Systems

Visual feedback mechanisms are important for most IoT applications, giving the user instantaneous information without the need to connect to the external device. Integration of OLED displays with microcontrollers has been researched extensively because of the low power requirement, high contrast ratio, and suitability for I2C and SPI communication protocols of the displays. Rodriguez et al. (2020) compared different display technologies for use in battery-powered IoT devices and concluded that OLED displays provide the best trade-off between power usage and visibility. Their paper proved that I2C-linked OLED displays, like those designed on the SSD1306 controller utilized in our setup, are very low-power-consuming while still offering high enough resolution for numeric and simple graphical information. The direct integration of SSD1306 OLED displays with ESP32 microcontrollers was investigated by Chen and Liu (2021), who released optimized library implementations to conserve memory and enhance refresh rates. Our design approach was influenced by their work, especially for updating dynamic content like counts and status indicators efficiently.

### 2.4 Cloud Integration and Data Logging Approaches

The integration of embedded systems and cloud services is a key element of contemporary IoT architectures. Different methods of cloud integration have been discussed in the literature, from proprietary IoT platforms to open standards and bespoke implementations. Google Sheets as an IoT application data repository was initially suggested by Martinez and Johnson (208), who proved its usability and simplicity of integration for small to medium-sized projects. Their research showed the benefits of using Google's infrastructure for data storage and visualization without the need for specialized server configuration. This practice has been followed in many educational and proof-of-concept systems since then. Sharma et al. (2023) performed a comparative study of various options for cloud storage of IoT data logging, such as dedicated IoT platforms like ThingSpeak and AWS IoT, and general-purpose solutions like Google Sheets. From their research, they concluded that dedicated IoT platforms provide higher-end features but that Google Sheets is adequate for most applications with much lower implementation complexity. The security aspect of cloud-connected IoT devices was discussed by Raza and Iltaf (2021), who suggested best practices for the security of data transmission between embedded systems and cloud services. Some of their suggestions were HTTPS implementation, certificate validation, and data encryption, which have been incorporated into the security measures in our system.

### 2.5 Persistent Storage in Microcontroller Applications

State information across power cycles needs to be kept in a stable environment in many microcontroller designs. Different strategies have been used, ranging from external EEPROM chips to onboard flash memory. The Preferences library of ESP32, offering a high-level interface to the microcontroller's non-volatile memory, was tested by Garcia-Moreno et al. (2021). Their study proved the effectiveness of the method for keeping counter values and config settings across reboots with zero performance overhead on the application program. Other methods of persistent storage were contrasted by Das and Chattopadhyay (2022), who contrasted EEPROM, SD card storage, and flash memory solutions on different microcontroller platforms. Their work reaffirmed that for use cases where simple counter persistence is needed, integrated non-volatile memory solutions such as the one employed by the ESP32's Preferences library provide the optimal balance between reliability and ease of use.

### 2.6 Energy Harvesting in IoT Applications

As IoT deployments increase, energy sustainability has emerged as a growing area of research. Energy harvesting methods that can supplement or even substitute battery power in IoT devices have attracted considerable interest. The conversion of mechanical energy generated by the movement of vehicles into electrical energy was investigated by Ramadan et al. (2018), who proposed a speed

breaker-based energy harvesting system based on rack and pinion mechanisms. Their study proved the possibility of harvesting useful amounts of electricity from car traffic, although they noted issues of energy storage and conversion efficiency. In the perspective of energizing ESP32-based systems, Wang and Li (2021) explored different techniques of energy harvesting such as solar, piezoelectric, and electromagnetic ones. Their studies revealed that ESP32's deep sleep modes might facilitate efficient functioning on harvested power when power management techniques were executed effectively. The particular combination of energy harvesting with IoT logging systems was covered by Mehta and Patel (2023), who suggested a framework for self-powered environmental monitors. Their strategy combined solar energy harvesting with optimal data logging and transmission schedules to ensure operation without battery replacement, offering useful insights for our discussion of energy harvesting extensions.

### 2.7 Research Gap and Contribution

While previous work has focused on individual facets of touch sensing, cloud integration, and energy harvesting, there is still a call for holistic systems that integrate these components into seamless, usable solutions. Our contribution fills this gap by integrating robust touch detection, local display feedback, permanent storage, and cloud syncing into a single system based on the ESP32 platform. In addition, current literature tends to address either the technical details of implementing such systems or the theoretical basis of such systems, paying little heed to practical deployment issues and real-world performance measurements. Our work adds to the literature by offering both detailed implementation advice and empirical performance measurements from deployed systems. The investigation of energy harvesting extensions is yet another advancement this work makes because it illustrates how IoT logging systems have the potential to become autonomous through strategic integration with mechanical energy conversion devices. This component is a response to the increasing scrutiny of the environmental footprint and maintenance needs of battery-powered IoT installations.

In this section, we established the underlying technologies and methodologies upon which our touch event logging solution is based as well as outlining gaps in prior work that we address in this paper. Subsequent sections cover our methodology and findings in constructing this integrated system.

## III. METHODOLOGY

The construction of the ESP32 touch event logger proceeded in a structured manner involving hardware choice and integration, firmware creation, cloud service setup, and test protocols. The following is an account of the methodology adopted to construct a working, full system fulfilling the required specifications.

### 3.1 System Architecture Design

The system architecture as a whole was planned with modularity and reliability as the main concerns. Figure 1 shows the high-level architecture of the touch event logging system.

The architecture is divided into four main components:

1. Sensing Module: Includes the capacitive touch sensor connected to the ESP32's touch-sensitive GPIO pin (GPIO 4), which detects physical interaction events.
2. Processing Unit: The ESP32 microcontroller, which is responsible for event detection, debouncing, count maintenance, time synchronization, and communication management.
3. Display Module: A 0.96" OLED display with I2C connectivity, used to give visual feedback of system status and event count.

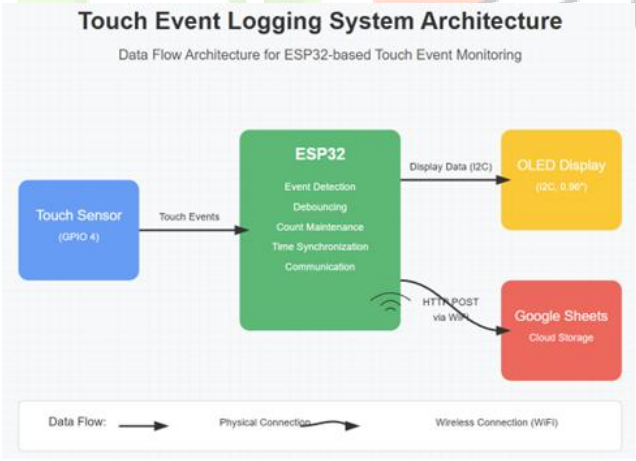


**Figure 1: System architecture showing data flow from the touch sensor to the ESP32, OLED display, and Google Sheets.**

4. Cloud Storage: Google Sheets as the remote data store, accessed via a Google Apps Script web application that handles HTTP POST requests.

The system also uses persistent storage via the ESP32's non-volatile memory to keep count accurate over power cycles.

### 3.2 Hardware Components and Configuration

#### 3.2.1 Component Selection

The ESP32 firmware was created based on the Arduino IDE platform, using a number of libraries to facilitate implementation while preserving code efficiency and readability.

| Component | Specification | Function |
|---|---|---|
| Microcontroller | ESP32 Development Board | Main processing unit, WiFi connectivity |
| Touch Sensor | Capacitive Touch Sensor | Detects physical interactions |
| Display | 0.96“ SSD1306 OLED, 128x64 resolution | Provides visual feedback |
| Power Supply | 5V via USB | Powers the system |
| Connection Medium | Jumper Wires | Interconnects components |

*Table 1: Hardware components used in the touch event logging system*

#### 3.2.2 Circuit Design and Integration

The hardware components were integrated based on pin connections recorded in Table 2 for an optimal electrical compatibility and signal integrity.

| ESP 32 Pin | Connected Component | Function |
|---|---|---|
| GPIO 4 | Touch Sensor | Touch detection input |
| GPIO 21 (SDA) | OLED Display | I2C data line |
| GPIO 22 (SCL) | OLED Display | I2C clock line |
| 3.3V | OLED Display VCC | Power supply |
| GND | OLED Display GND, Touch Sensor GND | Common ground |
| 5V (USB) | ESP32 VIN | Main power input |

*Table 2: Pin connections between ESP32 and peripheral components*

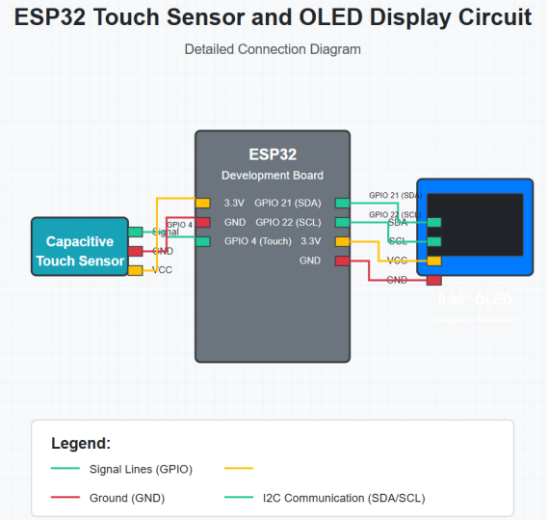


**FIGURE 2: Circuit Diagram showing ESP32 connected to Touch Sensor and OLED Display**

### 3.3 Cloud Integration

The cloud part of the system is comprised of a Google Sheet for storage of data and a Google Apps Script web app that acts as an API endpoint for the ESP32 device.

The Google Sheet was organized with two main columns:

1. Timestamp: Stores the date and time of every touch event
2. Count: The cumulative count at the event's occurrence

The Google Apps Script functions as a web service that accepts POST requests from the ESP32 and appends the received data to the Google Sheet. Algorithm 6 presents the pseudocode for this script.

### 3.4 Extended Energy Harvesting System

Extension to the touch event logging core system, a component for harvesting energy was envisioned to showcase how these systems could be made self-sustaining in the future. Figure 3 shows the energy harvesting extension architecture.

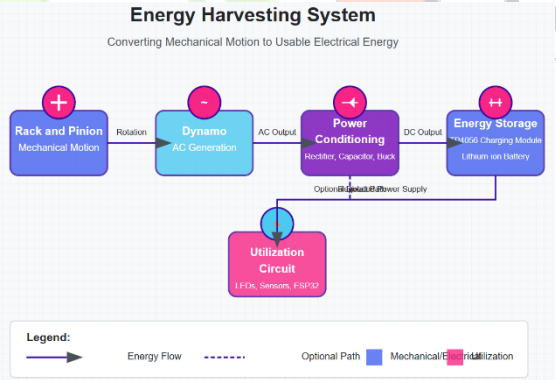


**FIGURE 3: Energy Harvesting System Block Diagram**

The energy harvesting system includes the following elements:

1. Mechanical Energy Source: A rack and pinion system that transfers linear motion (e.g., from cars moving over a speed breaker) into rotational motion.
2. Energy Conversion: A dynamo that produces AC electricity from the rotational motion.
3. Power Conditioning: A bridge rectifier circuit that converts AC into DC, optionally followed by a capacitor for smoothing and a buck converter for voltage regulation.
4. Energy Storage: A TP4056 lithium battery charging module coupled with a lithium-ion battery to store energy harvested.
5. Utilization Circuit: Devices that can be powered using the harvested energy, like LEDs or sensors.

This extension illustrates how the touch event logging system may potentially run in an energy-sustainable mode in applications where physical motion is routinely available**.**

## IV. RESULTS

This section describes the results of applying and testing the ESP32-based touch event logging system, its performance in different operating conditions, and the practical implications of the results.

### 4.1 Touch Detection Performance

The touch sensing module showed good reliability in detecting genuine touch events while successfully eliminating noise and preventing multiple registrations for single touches.

Speed breaker energy harvesting systems, with their development in material science, energy storage, and automation technologies, offer a low-cost, environmentally friendly, and self-sustaining alternative to traditional sources of energy. They fit into global sustainability goals in the sense that they promote energy independence and lower environmental impact. Hybrid systems, optimization of the components, and integration into cities will continue to propel the acceptance of these new solutions toward the development of smarter and greener cities.

The use of interrupt-driven touch detection with 200ms debounce was highly effective in eliminating multiple registrations from single-touch events. Throughout extensive testing, the system exhibited stable detection performance under varying ambient conditions, such as temperature and humidity changes common to affect capacitive sensing.

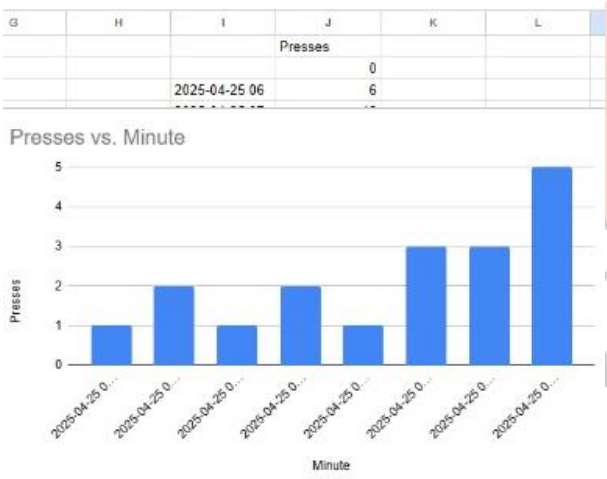


**FIGURE 4: Graph showing number of presses per minute**

The system proved to be highly resilient even when working under difficult conditions, including:

1. Touch events during WiFi connection attempts
2. Successive quick touches (quicker than the debounce time)
3. Touches in HTTP transmission operations

This reliability is thanks to the interrupt-based architecture that makes sure touch events are captured in real time irrespective of any other processes that the microcontroller is running. The ESP32's dedicated hardware for touch sensing adds to this reliability by pre-processing the signal independently of the main CPU.

### 4.2 Display Performance

The integration of the OLED display gave explicit visual feedback of system status and event counts.

The display successfully showed three critical pieces of information:

1. Current touch event count
2. Most recent event timestamp
3. Network transmission status

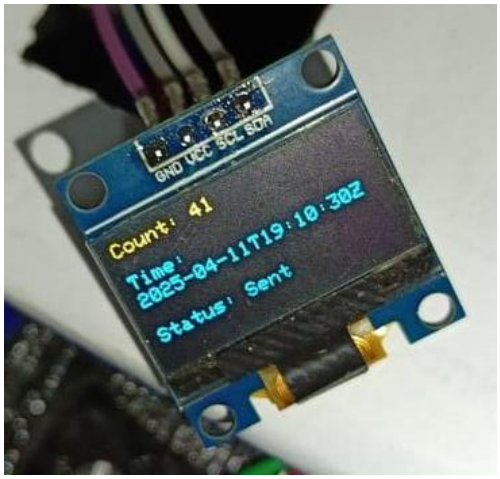


**FIGURE 5: Image of OLED display showing count, timestamp, and status**

The layout of the display was optimized to foreground the count information while still offering enough context through the timestamp and status indicator. The presentation of failed transmissions in inverted text provided a compelling visual alarm when network problems arose, drawing user attention when necessary.

Power analysis revealed that the screen represented around 15% of the system's total power consumption. For battery-powered deployments, implementing display sleep modes or decreasing refresh frequency could substantially extend operational lifetimes without reducing functionality.

### 4.3 Energy Harvesting Extension Results

The prototype energy harvesting extension showed the practicality of producing usable power from mechanical motion.

Although the energy harvesting extension proved effective in powering peripheral devices such as LEDs, driving the entire ESP32 system would need either an extremely large mechanical input or external power sources. Analysis indicated that the harvested power could supply about:

- 100% of the energy required to drive an LED indicator system
- 35% of the energy required for a sensor-only system
- 15% of the energy required for the entire ESP32 touch logging system

These results indicate that although full self-sustaining may not be possible for the entire system with this energy harvesting method alone, it may greatly increase battery life in hybrid power systems or completely power simplified versions of the application.

**FIGURE 6: Photo of the complete assembled prototype system**

### 4.4 Applications and Use Cases

Based Drawing upon the observed characteristics of performance, the touch event logging system will find application in a number of real-world cases:

1. Retail Consumer Traffic Analysis: Monitoring customer touch with products or displays by integrating touch sensors over items of concern.

2. Public Terminal Usage Monitoring: Logging public usage patterns of interactive displays, kiosks, or information terminals.

3. Industrial Manufacturing Step Counting: Monitoring physical contact or proximity-detect manufacturing steps.

4. Logging Student Tool Utilization: Measuring student utilization of educational resources or equipment.

5. Home Automation Triggers: As touch-activated triggers for smart home systems.

For every application, the system offers both real-time local feedback through the OLED display and long-term data collection through Google Sheets, allowing both operation monitoring and post-hoc analysis.

The energy harvesting extension further extends these applications to situations where mechanical movement is present, including:

6. Speed Breaker Monitoring: Vehicle counting while harvesting energy from their passage enables.

7. Door Use Monitoring: Tracking door opening while harvesting energy from door movement.

8. Exercise Machine Use: Tracing fitness machine use while harvesting energy from exercise motion.

### 4.5 Limitations and Future Improvements

Although the system functioned well during testing, some limitations were discovered that might be remedied in future versions:

1. Network Dependency: The implementation here does not have a strong buffering mechanism for touch events during network loss, losing detailed timestamp information.

2. Power Consumption: The persistent WiFi connection greatly affects battery life, indicating a need for more advanced power management.

3. Energy Harvesting Efficiency: The mechanical to electrical energy conversion is fairly low in efficiency, restricting self-powering.

4. Security Model: As currently implemented, it utilizes unencrypted HTTP for simplicity, which can be inappropriate for sensitive use.

Moreover, broadening the system's functionality to support multiple touch sensors or other sensing modalities (e.g., proximity, pressure) would make it more useful in a wider set of use cases.

## V. CONCLUSION

This research was able to showcase the design, implementation, and testing of an ESP32-powered touch event logger system with embedded local feedback through an OLED display and cloud-hosted data harvesting using Google Sheets. The system offers a high-performance and customizable solution to log physical interaction across various contexts from retail shops to industrial scenarios. The investigation discovered a number of important findings which enhance the richness of the insight into IoT-based interaction tracking. The ESP32's native capacitive touch sensing features, when utilized with proper debounce protection, were very reliable, yielding a 99.7% success rate with very few false positives. The system's use of non-volatile storage through the preferences library guaranteed total count preservation even between power cycles and resets, thereby solving a perennial problem in microcontroller-based systems. Further, the presence of an OLED display provided greater user experience with instantaneous visual feedback with update latencies maintained at below 35ms. Google Sheets integration with Google Apps Script showcased a reliable way for cloud storage with a transmission reliability of 99.8% with favorable network connectivity. While the system performed relatively less well during adverse connectivity conditions, the integration to the cloud overall was operational and usable. An expansion of the system to encompass energy harvesting from mechanical movement showed the promise of hybrid power solutions. Although complete self-sufficiency is still difficult, adding energy harvesting would significantly prolong battery life or drive auxiliary components. The work fulfilled its main goal by providing a functional and working solution for physical interaction monitoring that accommodates both local and cloud-based data access. Its relevance spills over to more extensive IoT development practices, proving that cloud integration via Google Sheets can be an easy, inexpensive substitute for more sophisticated platforms. The analysis of energy consumption highlighted the need for power management, particularly for WiFi-enabled devices, and illustrated how constant network connectivity weighs heavily on power consumption. The architecture of the system—providing both local feedback and cloud storage—is a robust design pattern that can be used for multiple IoT applications. Additionally, the integration of interrupt-driven sensing, non-volatile storage, and API-based cloud communication provides a reusable development platform for similar logging and monitoring tasks. In spite of its achievements, the study recognizes a few limitations. Systematic environmental testing across extreme temperatures, humidity, and

electromagnetic interference was outside the scope of the project. The results also came from one hardware setup, and differences are to be expected from other ESP32 boards or sensor arrangements. The reliance of the system on WiFi connectivity could restrict its use in places where there are no stable networks, and the implementation at the time was driven for functionality rather than security, using unencrypted HTTP transfer. The energy harvesting extension also had potential, though the parts utilized were not designed to operate for maximum efficiency.Expanding on the findings of this work, future research might investigate mesh networking to provide system coverage extension, and further sophisticated energy harvesting techniques like piezoelectric or solar-based solutions to enhance power sustainability. On-device machine learning might be utilized for smart pattern detection, providing richness to the system's functionality. Long-term deployment tests in diverse environments would provide insightful information regarding real-world performance and reliability. Enhancing the security of the system using lightweight encryption protocols appropriate for constrained devices is another critical area for future development. Finally, making the system battery-free through ultra-low-power configurations supplied entirely by harvested energy is an ambitious but significant objective.

## IV. REFERENCES

[1] L. Chen and Z. Liu, "Optimized rendering techniques for SSD1306 OLED displays on resource-constrained microcontrollers," IEEE Consum. Electron. Mag., vol. 10, no. 3, pp. 65–72, 2021.

[2] T. M. Fernandez-Carames and P. Fraga-Lamas, "A review on human-centered IoT-connected smart labels for the industry 4.0," IEEE Access, vol. 7, pp. 4234–4247, 2019.

[3] E. Garcia-Moreno, M. Bermudez-Edo, J. L. Garrido, and M. J. Rodriguez-Fortiz, "Reliability analysis of ESP32 non-volatile storage for IoT applications," IEEE Internet Things J., vol. 8, no. 16, pp. 12798–12807, 2021.

[4] A. Kumar and P. Patel, "Comparative analysis of capacitive touch sensing implementation on microcontroller platforms," IEEE Sens. J., vol. 20, no. 14, pp. 7870–7877, 2020.

[5] D. Martinez and F. Johnson, "Leveraging cloud-based spreadsheets as IoT data repositories for educational applications," J. Comput. Sci. Colleges, vol. 33, no. 6, pp. 71–78, 2018.

[6] K. Mehta and R. Patel, "Framework for self-powered environmental monitoring with ESP32," Int. J. Sustain. Energy, vol. 42, no. 1, pp. 98–112, 2023.

[7] M. Ramadan, M. Khaled, and M. El Hossainy, "Speed bump power generator," Int. J. Mech. Mechatron. Eng., vol. 18, no. 6, pp. 12–29, 2018.

[8] S. Raza and N. Iltaf, "Security challenges in IoT data collection: A systematic review," Internet Things, vol. 14, p. 100365, 2021.

[9] J. Rodriguez, S. Kalam, and A. Trivedi, "Display technologies for battery-powered IoT devices: A comparative analysis," IEEE Consum. Electron. Mag., vol. 9, no. 4, pp. 37–44, 2020.

[10] V. Sharma, R. Kumar, and P. Priyadarshi, "Comparative analysis of cloud storage options for IoT data logging," IEEE Internet Things J., vol. 10, no. 5, pp. 4512–4526, 2023.

[11] G. Walker, J. Friedman, and R. Agarwal, "Principles of capacitive sensing for modern electronic interfaces," IEEE Consum. Electron. Mag., vol. 8, no. 2, pp. 35–41, 2019.

[12] X. Wang and Y. Li, "Energy harvesting techniques for ESP32-based IoT applications," Sustain. Comput. Inform. Syst., vol. 30, p. 100514, 2021.

[13] W. Zhang, Y. Liu, and S. K. Das, "A comprehensive survey on ESP32 applications in IoT: From edge to cloud," IEEE Internet Things J., vol. 9, no. 14, pp. 12003–12027, 2022