Great, I'll prepare a concise scientific-style writeup in English that mirrors the structure and length of the PDF examples you provided, and is based on your smart bin monitoring project with ESP32, VL53L0X, and OLED display. I'll let you know as soon as it's ready.

Smart Trash Bin Monitoring System

Introduction

Efficient waste management often requires knowing when a trash bin is nearing capacity to prevent overflow and optimize collection schedules. This project implements a smart trash bin fill-level monitoring system using an ESP32-S3-DevKitC-1 microcontroller and a Time-of-Flight (ToF) distance sensor. The system continuously measures the distance from the bin's lid to the top of the trash and calculates the fill percentage of a waste bin (internal dimensions \~41×35×60 cm). An OLED display provides real-time readouts of the distance and fill level, and an onboard LED gives a visual alert when the bin is almost full. Additionally, the device connects via Wi-Fi to send periodic updates and warnings to a Node-RED dashboard and a Telegram channel, enabling remote monitoring. This report details the design and results of the system, including hardware/software implementation, operational results, code logic, and a brief discussion of limitations and improvements.

Materials and Methods

Hardware Components: The project's hardware centers on the Espressif ESP32-S3-DevKitC-1 board, which serves as the main microcontroller. This board offers ample processing power and built-in Wi-Fi for network connectivity, making it ideal for IoT applications. Key components include:

* *ESP32-S3-DevKitC-1 Microcontroller:* Handles sensor interfacing, data processing, and Wi-Fi connectivity. The board's onboard RGB LED (on GPIO48) is repurposed as a fill-level warning indicator. * *Time-of-Flight Distance Sensor (VL53L0X or VL53L1X):* Short-range LiDAR module used to measure the distance from the bin's lid to the trash surface. The VL53L0X can measure distances up to \~2 m, while the VL53L1X extends to \~4 m – both easily covering the \~0.6 m bin height. The sensor is mounted at the top of the bin pointing downward. It communicates via I²C to provide high-resolution distance readings in millimeters. * *SH1106 OLED Display (128×64 pixels, I²C):* Used to show the current distance reading (in cm) and the calculated fill percentage on a compact screen. This provides immediate visual feedback to users on site. * *Bin and Casing:* A standard trash bin (\~41 cm × 35 cm base, 60 cm tall) is used as the container. The electronics are mounted such that the ToF sensor protrudes inside the lid, and the OLED is visible externally for easy reading. The ESP32 board and sensor are powered via USB or a 5 V supply.

Wiring and Assembly: The circuit is straightforward since most components use the I²C bus. The VL53L0X/VL53L1X sensor and the SH1106 OLED display are both connected to the ESP32's I²C pins (SDA and SCL) using the Wire library for communication. Power (3.3 V) and ground lines are shared among the ESP32 and the I²C peripherals. The ESP32's onboard LED is used for alerts (no extra wiring needed for an external LED). During assembly, care was taken to firmly attach the sensor at the bin's top interior, ensuring it faces downwards without obstruction. The OLED is mounted on the outside or the lid for visibility. The ESP32 board connects to a Wi-Fi network for data transmission but requires no additional wiring for this (it uses its internal antenna).

Software and Libraries: The firmware is written in C++ using the Arduino framework. Development and testing were done via the Arduino IDE. Several standard libraries facilitate the functionality:

* *Wire.h:* I²C communication library used to interface with the ToF sensor and OLED display. * *Adafruit_VL53L0X.h:* Driver for the ST VL53L0X ToF sensor (also compatible with VL53L1X with

minor modifications) to initialize the sensor and read distance values. * *U8g2lib.h:* U8g2 graphics library for driving the SH1106 OLED display. It provides functions to render text and graphics on the screen over I²C. * *WiFi.h:* ESP32 Wi-Fi library to connect the device to a local wireless network (providing internet/LAN connectivity for IoT features). * *HTTPClient.h:* Used to send HTTP requests to a backend (in this case, a Node-RED endpoint). The system uses this to *POST* or *GET* the bin status to a Node-RED server, which can log data or trigger further actions. * *UniversalTelegramBot.h:* Library for interacting with the Telegram Bot API. This enables the ESP32 to send messages or alerts to a predefined Telegram chat (via an HTTP(S) request to Telegram's servers). For example, the system uses this to push a warning message when the bin exceeds a fill threshold. * (Other core Arduino libraries like \<ArduinoJson.h> may be used for formatting data payloads, and \<ESP32HTTPClient.h> is typically included via HTTPClient on ESP32.)

After programming, the ESP32-S3 board was powered and connected to Wi-Fi. A simple Node-RED flow was set up on a computer/server to receive HTTP POST data from the ESP32 (containing distance or fill level info) and display or log it. A Telegram bot was created and its credentials (bot token and chat ID) were configured in the firmware to allow the device to send Telegram messages.

Results

Once deployed, the smart bin monitoring system performed continuous measurement and successfully provided both local and remote feedback on the bin's status. *Distance and Fill Readout:* The OLED display updates in real-time with the current distance from the sensor to the trash. For instance, if the trash is 45 cm below the lid, the display might show "Distance: 45.0 cm" on one line. The next line shows the calculated fill percentage, e.g. "Fill: 25%". The fill percentage is computed based on the bin's 60 cm height (e.g. fill% = (60 cm − measured_distance) / 60 cm × 100%). As trash accumulates and the measured distance decreases, the displayed percentage increases. This provides an immediate intuitive indication of how full the bin is.

Visual and Remote Alerts: A key feature is the automatic alert when the bin is nearly full. The system defines "nearly full" as ≥80% capacity (i.e. the trash is within roughly 12 cm of the lid). When this threshold is reached or exceeded, the ESP32's onboard LED turns on as a *visual alert. In testing, when the bin was filled above the 80% mark, the LED illuminated clearly (in the case of an RGB LED, it can be set to a bright color like red). Simultaneously, the system sends a **warning message* over Wi-Fi. The Node-RED endpoint receives a JSON payload (for example: {"bin":"Kitchen","fill":85,"status":"Nearly Full"}), which can be displayed on a dashboard gauge and logged with a timestamp. The Telegram bot sends a notification to the configured channel or user, for example: "⚠ Alert: The kitchen trash bin is 85% full. Please empty it soon." These alerts were verified during demonstrations – the Telegram message arrived within seconds of the bin crossing the threshold, and Node-RED successfully updated its interface. Regular status updates (e.g. periodic messages or Node-RED updates even when the fill level is below the threshold) were also enabled, allowing remote monitoring of the bin level over time. If the sensor ever encounters a read error (for instance, no object detected or an I²C read failure), the system handles it by displaying an error message on the OLED (e.g. "Sensor error") and skipping that reading; an error flag can also be sent to Node-RED/Telegram so the user knows the data might be momentarily unavailable.

Overall, the system proved capable of accurately monitoring the bin's fill level. In a simple test, placing objects inside to different heights yielded the expected distance readings and corresponding fill percentages. The 80% full LED trigger worked as intended, and remote notifications were successfully delivered, demonstrating the viability of the design.

Code Summary

The firmware follows a typical Arduino structure with setup and loop functions, utilizing the aforementioned libraries to implement the logic. The pseudocode of the system is outlined below:

1. *Initialization (setup):* The ESP32 initializes serial communication for debugging and configures the I²C bus. The ToF sensor is started (e.g., using lox.begin() and possibly set to continuous ranging mode). The SH1106 display is initialized via U8g2 and a splash or initial message may be shown. Wi-Fi connectivity is then established using the stored SSID/password, and upon success an HTTP client and Telegram bot client are prepared. If any initialization fails (sensor not found, Wi-Fi not connected), the system prints error messages and may indicate this on the OLED (e.g. “WiFi Error” or “Sensor Fail”). 2. *Main Loop (continuous operation):* The microcontroller continuously measures the distance to the trash. In each cycle, a distance reading is obtained from the VL53L0X sensor (e.g., distance = sensor.readRange()). The code checks for invalid readings (the sensor returns 0 or a timeout if it fails to measure). If a reading is valid, the fill percentage is calculated using the bin’s known height. 3. *Display Update:* The OLED is refreshed with the new values each cycle. The code clears the display buffer and prints the distance (in centimeters, possibly averaged or filtered for stability) and the fill percentage. This update occurs at a reasonable interval (for example, a few times per second or every few seconds) to keep the display current without flickering. 4. *LED Alert Logic:* The program compares the fill percentage against the predefined threshold (80%). If fill_percent >= 80% and the LED is not already on, the onboard LED is activated (e.g., set to HIGH or in case of an RGB LED, set to a solid color). If the fill drops below the threshold (e.g., after emptying the bin) the LED is turned off. This hysteresis prevents flicker around the threshold boundary. 5. *Networking – Node-RED Update:* The loop periodically (e.g. every fixed interval like 30 seconds, or each significant change) sends an HTTP POST request to the Node-RED server with the latest data. This is done using the HTTPClient library: the bin’s status (distance, fill%, maybe a timestamp) is formatted (often as a JSON string) and posted to a known URL endpoint where Node-RED is listening. The code checks the HTTP response to ensure the data was received. 6. *Networking – Telegram Notifications:* Using the UniversalTelegramBot library, the code sends a Telegram message when certain events occur. Specifically, when the fill crosses the 80% threshold (or reaches 100%), a one-time alert message is sent to notify that the bin is nearly full and should be emptied. Additionally, the system can be configured to send periodic Telegram updates (for example, a daily summary or when the bin gets emptied). The bot communication involves sending an HTTPS request via WiFi to Telegram’s API with the bot token and chat ID. 7. *Error Handling and Loop Delay:* If the distance sensor returns an error (e.g., no valid reading), the code can retry a few times or output an error state for that cycle, ensuring the rest of the loop continues. A short delay or timer is used at the end of each loop iteration to regulate the measurement frequency and avoid flooding the network with too many requests. In practice, readings and updates might be done, for example, once every few seconds for local display and perhaps every minute for remote updates, which is sufficient for a trash bin scenario.

This logical structure ensures that the device remains responsive and up-to-date, providing both local feedback via display/LED and remote feedback via network. The code is modular, with separate functions handling sensor reading, display output, and network communication, making it easier to maintain or expand.

Discussion

This smart bin monitoring system demonstrates a practical Internet-of-Things solution for waste management, but there are some limitations and opportunities for improvement. One limitation is that the ToF sensor measures distance at a single point directly below it. If trash is unevenly distributed (e.g. piled higher on one side away from the sensor), the measured distance might not reflect the true highest fill level. In such cases, the bin could be slightly fuller than reported. A possible improvement would be to use multiple sensors or a scanning mechanism (like a servo-mounted sensor) to sample distances at various points, giving a more holistic measurement of fill

level. Another limitation is the reliance on Wi-Fi connectivity: if the network is down or the device loses connection, remote updates and alerts will fail. Implementing reconnection logic or an offline indicator (and perhaps storing data to send later) could mitigate this. Additionally, the system currently runs on continuous power; if battery operation is desired, power-saving techniques (like deep sleep between measurements) could be implemented to prolong battery life.

There is also room for feature enhancements. For example, adding a weight sensor at the bottom of the bin could provide another metric for fill level (by weight) to complement the distance measurement. The OLED display, while useful, could be supplemented with an audible buzzer for a local alarm when the bin is full, which would be hard to miss. On the software side, the Node-RED integration could be expanded — for instance, to send email notifications or log historical fill data for analysis. The Telegram bot could be made interactive, allowing a user to query the current bin status on-demand.

Despite these limitations, the impact of the project is significant for its scale. In a broader context, such a system can reduce the need for manual checks of trash bins and enable more efficient waste collection scheduling. Facilities management in smart buildings or campuses could deploy these sensors on multiple bins and monitor them centrally, ensuring bins are emptied right when needed and never overflow. This not only improves cleanliness and hygiene but can also optimize labor and collection routes, contributing to smarter urban infrastructure. In conclusion, the project achieved its primary goal of creating a functional bin fill-level monitoring device, and it illustrates how low-cost sensors and Wi-Fi connectivity can be leveraged to solve everyday problems in an intelligent way.