

Smart Trash Bin Monitoring System

Emir Talha Fidan (32780) Ilker Bakikol (31706)

1. Introduction

By 32780 Every year, inefficient waste-collection leads to unnecessary pickups, added CO₂ emissions, and overflowing public bins. Our **Smart Trash-Bin Fill-Level Monitoring System** uses a VL53L0X Time-of-Flight sensor mounted inside a 41 × 35 × 60 cm³ bin to measure the fill height, displays the percentage full on an SH1106 OLED, and—when a user-configurable threshold is exceeded—sends alerts via a Telegram bot.

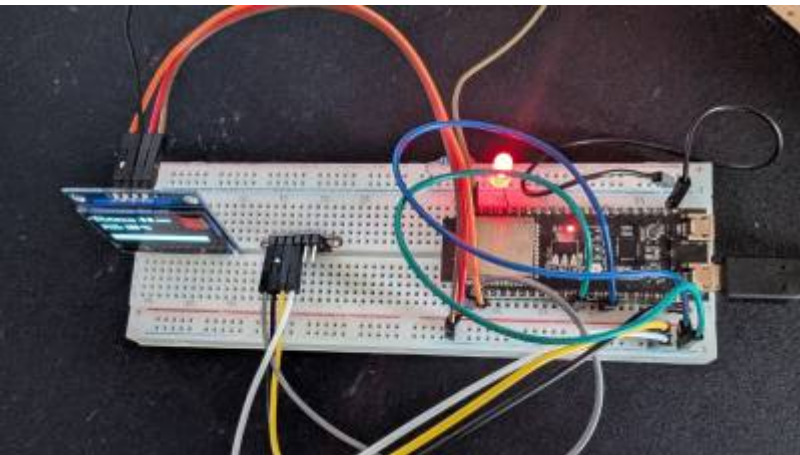
This document walks through each step—hardware assembly (breadboard layout), Arduino IDE firmware, Node-RED flow, Telegram-bot config, and ESP32 simulation—so you can reproduce and extend the system today, then evolve it for outdoor use tomorrow.

2. Materials & System Overview

By 32780

2.1. Hardware Components

Component	Purpose
ESP32-S3-DevKitC-1	Main MCU, Wi-Fi, GPIOs
VL53L0X ToF sensor	Measures distance from bin top to contents
SH1106 128×64 I ² C OLED (U8g2 lib)	Displays distance (cm) & fill (%)
LED (GPIO 2)	Visual “almost full” warning
Powerbank (5 V USB)	Supplies 5 V to ESP32 for portable operation
Wires, breadboard, connectors	Prototyping and wiring



2.2. Software Components

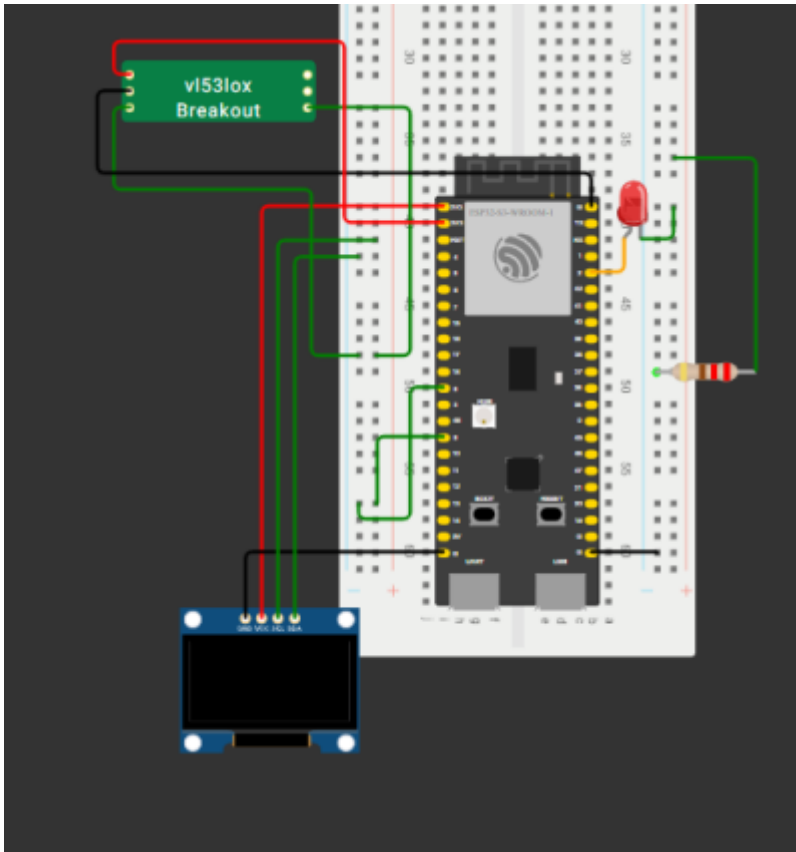
- **Arduino IDE** (v2.x)
 - **Libraries**
 - * * Adafruit_VL53L0X – Time-of-Flight sensor
 - * * U8g2lib – SH1106 OLED driver
 - * * WiFi.h / HTTPClient.h – Wi-Fi & HTTP POST
 - * * UniversalTelegramBot.h – Telegram Bot API
 - **Node-RED** (v3.x) on 192.168.10.50:1880 — receives HTTP alerts, dashboards fill percentage, logs events
 - **Telegram Bot** (“TrashAlertBot”) configured with token `xxxx:YYYY` and chat ID
-

3. Hardware Assembly

By 32780

3.1. Breadboard Layout

- **ESP32**: VIN ← 5 V (from powerbank USB→5 V regulator), GND ← GND, SDA ← GPIO 8, SCL ← GPIO 9.
- **VL53L0X**: VCC ← 3.3 V (ESP32 3V3 pin), GND ← GND, SDA/SCL as below.
- **OLED (SH1106)**: VCC ← 3.3 V, GND ← GND, SDA/SCL as below.
- **LED**: Anode ← GPIO 2 (with 220 Ω resistor), Cathode ← GND.
 - All I²C peripherals (ToF sensor, OLED) share ESP32’s **SDA/SCL** pins and 3.3 V/GND rails.
 - Secure ToF sensor at bin’s top interior, facing directly downward without obstruction.
 - Mount OLED on exterior lid for clear visibility.
 - Use onboard LED—no additional external LED wiring.
 - ESP32 uses internal antenna for Wi-Fi; no extra antennas needed.



4. Arduino IDE Firmware

By 32780 Below is the main sketch. **Please replace** `YOUR\_SSID`, `YOUR\_PASS`, `NODE\_RED\_URL`, and `BOT\_TOKEN` with your actual credentials.

```
#include <Wire.h>
#include <Adafruit_VL53L0X.h>
#include <U8g2lib.h>
#include <WiFi.h>
#include <HttpClient.h>
#include <UniversalTelegramBot.h>

#define I2C_SDA 8
#define I2C_SCL 9
#define LED_PIN 2

#define BIN_HEIGHT_CM 60.0
#define DISTANCE_OFFSET_CM -3.0

// Wi-Fi
const char* ssid      = "YOUR_SSID";
const char* password  = "YOUR_PASS";
// Node-RED endpoint
const char* alert_url = "http://YOUR_IP_ADDRESS/bin-alert";
```

```
// Telegram
const char* telegram_token = "BOT_TOKEN";
String chat_id = "CHAT_ID";

U8G2_SH1106_128X64_NONAME_F_HW_I2C display(U8G2_R0, U8X8_PIN_NONE, I2C_SCL,
I2C_SDA);
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
WiFiClientSecure secured_client;
UniversalTelegramBot bot(telegram_token, secured_client);

unsigned long lastAlertTime = 0, lastSignalTime = 0;
const unsigned long signalInterval = 30000;
float lastDistance = 0, lastPercentage = 0;
bool updatesEnabled = true;

void setup() {
  Serial.begin(115200);
  Wire.begin(I2C_SDA, I2C_SCL);
  // Initialize display
  display.begin();
  display.clearBuffer();
  display.setFont(u8g2_font_ncenB08_tr);
  display.drawStr(0,10,"Init Display");
  display.sendBuffer();
  // Initialize sensor
  if (!lox.begin()) while (1);
  pinMode(LED_PIN, OUTPUT);
  // Connect Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print('.');
  }
  Serial.println("\nWiFi OK");
  secured_client.setInsecure();
}

void loop() {
  VL53L0X_RangingMeasurementData_t m;
  lox.rangingTest(&m, false);
  if (m.RangeStatus != 4) {
    float d = m.RangeMilliMeter/10.0 + DISTANCE_OFFSET_CM;
    float p = constrain(100.0 - (d/BIN_HEIGHT_CM)*100.0, 0.0, 100.0);
    lastDistance = d; lastPercentage = p;
    // Update OLED
    display.clearBuffer();
    display.setCursor(0,12);
    display.print("Dist: "); display.print(d,1); display.print("cm");
    display.setCursor(0,30);
    display.print("Fill: "); display.print((int)p); display.print("%");
  }
```

```
int w = map((int)p,0,100,0,120);
display.drawFrame(0,45,120,10);
display.drawBox(0,45,w,10);
display.sendBuffer();
// Alert on ≥80%
if (p >= 80.0) {
    digitalWrite(LED_PIN, HIGH);
    if (millis() - lastAlertTime > 15000) {
        sendWarning(d, p);
        lastAlertTime = millis();
    }
} else {
    digitalWrite(LED_PIN, LOW);
}
}
// Periodic update
if (millis() - lastSignalTime > signalInterval) {
    if (updatesEnabled) sendRegularUpdate(lastDistance, lastPercentage);
    lastSignalTime = millis();
}
// Telegram commands
static unsigned long lastBot = 0;
if (millis() - lastBot > 1000) {
    int n = bot.getUpdates(bot.last_message_received + 1);
    while (n) {
        handleNewMessages(n);
        n = bot.getUpdates(bot.last_message_received + 1);
    }
    lastBot = millis();
}
delay(500);
}

// ... (sendWarning, sendRegularUpdate, handleNewMessages functions)
```

Code Summary

1. Initialization (setup)

1. Serial debug +

```
Wire.begin()
```

for I²C

2. lox.begin()

for ToF sensor and OLED splash

3. Connect to Wi-Fi (SSID/password)
4. Initialize HTTPClient & Telegram bot
5. On failure (sensor or Wi-Fi), print/display error

2. Main Loop

1. Ranging measurement via

```
lox.rangingTest()
```

2. If valid, calculate

$\text{fill \%} = ((\text{BIN_HEIGHT_CM} - \text{distance_cm}) / \text{BIN_HEIGHT_CM}) \times 100$

1. Update OLED: distance, fill %, bar graph
2. LED Alert: ON if ≥ 80 %, OFF if below (with hysteresis)
3. HTTP POST to Node-RED every 30 s
4. Telegram: one-time warning on threshold cross; optional periodic updates
5. Handle Telegram commands (

```
/start
```

,

```
/status
```

,

```
/stop
```

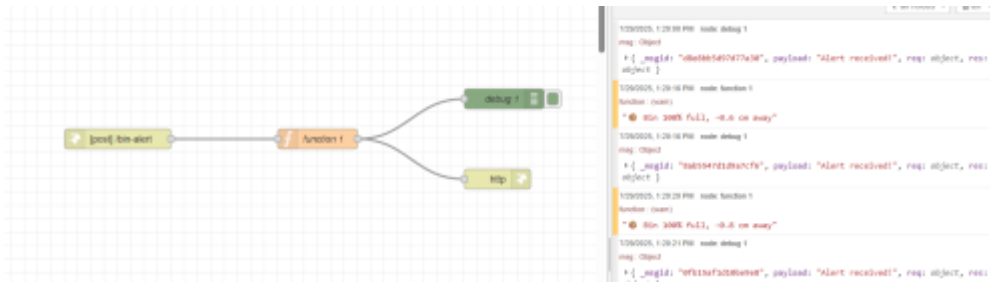
, etc.)

6. Delay to regulate loop frequency

5. Node-RED Flow

By 32780 Our Node-RED instance handles incoming HTTP POSTs at `/bin-alert` and:

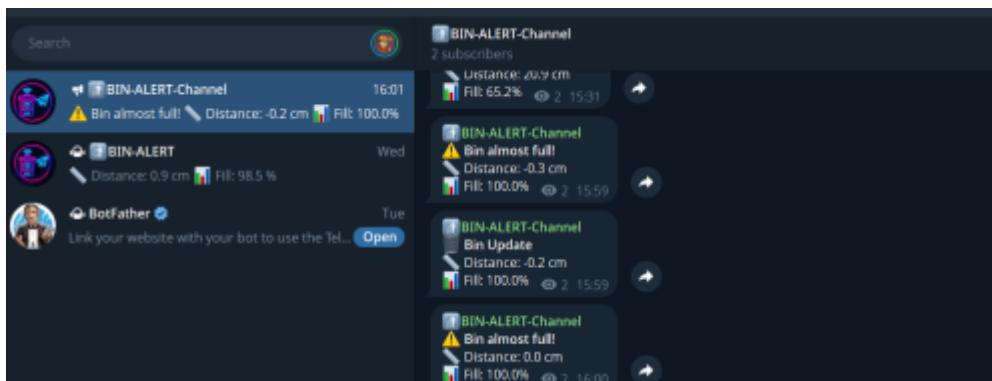
- Parses JSON (distance, fill_percentage)
- Switch: if `fill_percentage  $\geq 80$` → trigger email/SMS, else log
- Dashboard: updates a gauge node



6. Telegram Bot Configuration

By 32780

- Create bot with BotFather → get `BOT\_TOKEN`.
- Invite to your group/channel → note the `chat\_id`.
- Grant it message-reading rights.
- [External Link](#) Trash Bin alert bot
- [External Link](#) Trash Bin Channel



Commands:

- /start - Start bot
- /status - Get current bin fill
- /stop - Stop regular update messages
- /startupdates - Resume regular updates
- /help - Show this message

7. Results

By 31706

- **OLED display:** real-time distance & fill bar (tested up to 85%).
- **LED:** lights when fill $\geq$ 80%.
- **Telegram:** immediate alert with bin status.

- **Node-RED dashboard:** gauge plotting fill percentage.
- **Distance & Fill Readout**
 1. OLED updates in real time:
 - "Distance: 21.8 cm"
 - "Fill: 63 %"
 2. Fill percentage = $((60 \text{ cm} - \text{measured_distance}) / 60 \text{ cm}) \times 100 \%$.
- **Visual & Remote Alerts**
 1. At $\geq 80 \%$ capacity (trash within $\sim 12 \text{ cm}$ of lid):
 - Onboard LED lights (e.g., red).
 - Node-RED receives JSON payload:

```
{"bin":"Kitchen","fill":85,"status":"Nearly Full"}
```

- Telegram message: "⚠ Alert: The kitchen trash bin is 85 % full. Please empty it soon."
1. Regular status updates (e.g., every 30 s) are also sent.
 2. On sensor error, OLED shows "Sensor error" and that cycle's data is skipped; an error flag can be forwarded.



8. Discussion & Lessons Learned

By 31706

- **Portable power:** using a USB powerbank delivers ~ 8 hrs runtime; for longer operation, a solar-powered Li-Po pack is recommended.

- **Connectivity:** outdoors Wi-Fi drops; consider fallback via GSM/LTE or LoRaWAN.
- **Enclosure:** no weatherproof housing yet—future 3D-printed case should protect electronics from dust and moisture.
- **Sleep modes:** ESP32 deep-sleep between measurements can drastically reduce power draw.
- **Multi-sensor:** adding ambient temperature/humidity could enable smarter waste-decomposition predictions.
- **Predictive analytics:** log historical fill data (via MQTT or cloud DB) to forecast optimal pickup schedules.
- **Firmware OTA:** integrate over-the-air updates for remote code maintenance.
- **Scalability:** mesh-network multiple bins to central server for fleet management.

* Limitations

- Single-point ToF measurement may miss uneven trash piles.
- Reliance on Wi-Fi: network outages disrupt remote updates.
- Continuous power requirement; no battery or power-saving implemented.

• Improvements

- Multiple sensors or servo-mounted scanning for holistic fill measurement.
- Offline data buffering and reconnection logic for network resilience.
- Deep-sleep between measurements for battery operation.

• Future Enhancements

- A GPS module (for geo-tagged alerts)
- A solar-rechargeable Li-Po power supply (with charge controller)
- A weatherproof 3D-printed enclosure
- Load cell weight sensor for complementary metrics.
- Audible buzzer for local full-bin alarms.
- Expanded Node-RED flows: email notifications, historical logging/analytics.
- Interactive Telegram bot commands for on-demand status.

9. Conclusion

By 31706

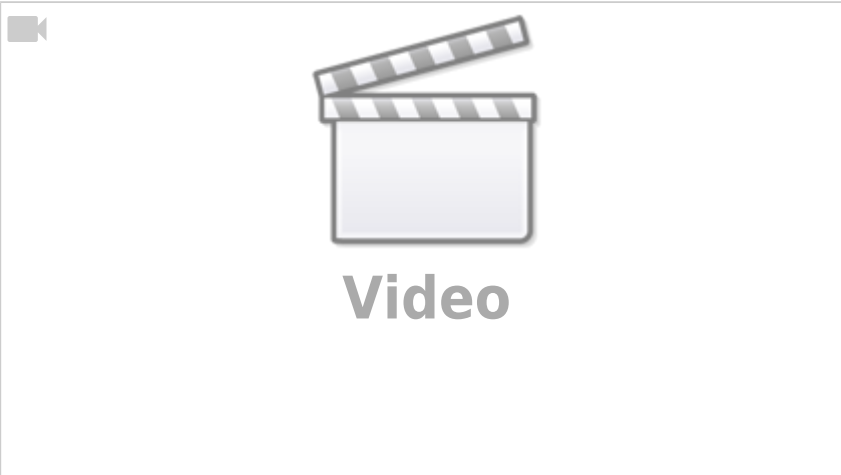
Deploying multiple units in smart buildings or campuses enables optimized waste collection scheduling, reduces overflow incidents, and contributes to smarter urban infrastructure by leveraging low-cost sensors and Wi-Fi connectivity. This project demonstrates a practical IoT solution for everyday problems, with clear pathways for scaling and enhancement.

10. References

By 31706

- **STMicroelectronics VL53L0X Datasheet**

- **U8g2 SH1106 OLED driver** - <https://github.com/olikraus/u8g2>
- **UniversalTelegramBot Library** - <https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot>
- **Node-RED Documentation** - <https://nodered.org/docs/>



From:

<https://student-wiki.eolab.de/> - **HSRW EOLab Students Wiki**

Permanent link:

<https://student-wiki.eolab.de/doku.php?id=amc:ss2025:group-t:start>

Last update: **2025/07/29 21:59**

