⚙️ Wiring Table (simplified for OLED I²C)

Component	Pin on Module	Connect to ESP32	Notes
OLED Display (SSD1306)	VCC	3.3V	0.96" OLED supports 3.3V
	GND	GND	Common ground
	SDA	GPIO 21	I²C SDA
	SCL	GPIO 22	I²C SCL
MFRC522 RFID	VCC	3.3V	Must not use 5V
	GND	GND	Common ground
	SDA (SS)	GPIO 5	Slave select
	SCK	GPIO 18	SPI clock
	MOSI	GPIO 23	SPI data out
	MISO	GPIO 19	SPI data in
	RST	GPIO 17	Reset
HC-SR04 Ultrasonic	VCC	5V	
	GND	GND	
	TRIG	GPIO 13	
	ECHO	GPIO 12 (via voltage divider)	1.8k + 3.3k divider
Buzzer (active)	+	GPIO 25	via resistor or transistor
	–	GND	
LED Green	+	GPIO 26	via 220Ω
LED Red	+	GPIO 27	via 220Ω

```
/* Smart Incentivized Dustbin - OLED Version
 * ESP32 + MFRC522 + HC-SR04 + OLED 0.96" (I2C SSD1306)
 * Offline demo with local persistence using LittleFS
 */

#include <Wire.h>
#include <SPI.h>
#include <MFRC522.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <LittleFS.h>
#include <ArduinoJson.h>

// ---------------- PIN CONFIG ----------------
#define RST_PIN 17
#define SS_PIN 5

#define TRIG_PIN 13
#define ECHO_PIN 12 // via voltage divider

#define BUZZER_PIN 25
#define LED_GREEN 26
#define LED_RED 27

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// ---------------- RFID OBJECT ----------------
MFRC522 mfrc522(SS_PIN, RST_PIN);

// ---------------- STUDENT DATABASE ----------------
struct Student {
  String uid;
  String name;
  int credits;
};

Student students[] = {
  {"04A1B2C3", "VIRAJ", 10},
  {"03D4E5F6", "GARGI", 8},
  {"027A9B8C", "TANVI", 12},
  {"09112233", "AMAN", 5},
  {"00AA11BB", "PRIYA", 7}
};
const int STUDENT_COUNT = sizeof(students)/sizeof(students[0]);
const char *DB_FILE = "/credits.json";

// ---------------- BIN SETTINGS ----------------
const int BIN_HEIGHT_CM = 30;
```

```
int lastFillPercent = 0;
const int SIGNIFICANT_THRESHOLD_PERCENT = 8;

// ---------------- HELPERS ----------------
long readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH, 30000);
  if (duration == 0) return -1;
  long dist = duration * 0.034 / 2;
  return dist;
}

int findStudent(String uid) {
  uid.toUpperCase();
  for (int i = 0; i < STUDENT_COUNT; i++) {
    if (students[i].uid.equalsIgnoreCase(uid)) return i;
  }
  return -1;
}

// ---------------- FILE I/O ----------------
void saveCredits() {
  DynamicJsonDocument doc(1024);
  for (int i=0;i<STUDENT_COUNT;i++){
    doc[students[i].uid] = students[i].credits;
  }
  File f = LittleFS.open(DB_FILE, "w");
  serializeJson(doc, f);
  f.close();
}

void loadCredits() {
  if (!LittleFS.exists(DB_FILE)) { saveCredits(); return; }
  File f = LittleFS.open(DB_FILE, "r");
  DynamicJsonDocument doc(1024);
  deserializeJson(doc, f);
  for (int i=0;i<STUDENT_COUNT;i++){
    if (doc.containsKey(students[i].uid))
      students[i].credits = doc[students[i].uid];
  }
  f.close();
}

// ---------------- DISPLAY HELPERS ----------------
void showMessage(String l1, String l2 = "", String l3 = "") {
  display.clearDisplay();
  display.setTextSize(1);
```

```
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 16);
  display.println(l1);
  if (l2 != "") display.println(l2);
  if (l3 != "") display.println(l3);
  display.display();
}

void animateProcessing() {
  for (int i=0;i<4;i++){
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(20, 20);
    display.print("Processing");
    for (int j=0;j<=i;j++) display.print(".");
    display.display();
    delay(250);
  }
}

void showReward(String name, int reward, int total) {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 10);
  display.println("Congrats " + name + "!");
  display.setCursor(0, 25);
  display.println("Reward: +" + String(reward) + " pts");
  display.setCursor(0, 40);
  display.println("Total: " + String(total) + " pts");
  display.display();
  tone(BUZZER_PIN, 2000, 200);
}

// ---------------- SETUP ----------------
void setup() {
  Serial.begin(115200);
  LittleFS.begin();

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_RED, OUTPUT);

  SPI.begin();
  mfrc522.PCD_Init();

  Wire.begin(21, 22);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("SSD1306 init failed!");
```

```
    while (true);
  }
  display.clearDisplay();
  display.display();

  loadCredits();

  long d = readDistanceCM();
  if (d <= 0 || d > BIN_HEIGHT_CM) lastFillPercent = 0;
  else lastFillPercent = constrain(map(d, 0, BIN_HEIGHT_CM, 100, 0), 0, 100);

  showMessage("Smart Dustbin", "Tap RFID to start");
}

// ---------------- LOOP ----------------
void loop() {
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    String uid = "";
    for (byte i=0;i<mfrc522.uid.size;i++){
      char buf[3]; sprintf(buf, "%02X", mfrc522.uid.uidByte[i]);
      uid += String(buf);
    }
    uid.toUpperCase();
    int idx = findStudent(uid);

    if (idx < 0) {
      showMessage("Unknown Card!", "Access Denied");
      tone(BUZZER_PIN, 1000, 150);
      delay(1500);
      showMessage("Tap RFID to start");
      return;
    }

    digitalWrite(LED_GREEN, HIGH);
    showMessage("Hello, " + students[idx].name,
                "Credits: " + String(students[idx].credits),
                "Bin: " + String(lastFillPercent) + "%");
    tone(BUZZER_PIN, 1500, 100);
    delay(400);

    int initialFill = lastFillPercent;
    int maxDelta = 0;
    unsigned long start = millis();

    while (millis() - start < 5000) {
      long d = readDistanceCM();
      if (d > 0 && d <= BIN_HEIGHT_CM) {
        int now = constrain(map(d, 0, BIN_HEIGHT_CM, 100, 0), 0, 100);
        int delta = now - initialFill;
        if (delta > maxDelta) maxDelta = delta;
        showMessage("Bin Fill: " + String(now) + "%",
```

```
                    "Wait 5s...");
      }
      delay(300);
    }

    if (maxDelta >= SIGNIFICANT_THRESHOLD_PERCENT) {
      animateProcessing();
      int reward = 1 + (maxDelta / 5);
      students[idx].credits += reward;
      saveCredits();
      showReward(students[idx].name, reward, students[idx].credits);
      delay(3000);
    } else {
      showMessage("No significant waste", "No reward given");
      tone(BUZZER_PIN, 800, 200);
      delay(1500);
    }

    digitalWrite(LED_GREEN, LOW);
    mfrc522.PICC_HaltA();
    showMessage("Tap RFID to start");
  }

  static unsigned long lastCheck = 0;
  if (millis() - lastCheck > 5000) {
    long d = readDistanceCM();
    if (d > 0 && d <= BIN_HEIGHT_CM) {
      lastFillPercent = constrain(map(d, 0, BIN_HEIGHT_CM, 100, 0), 0, 100);
      if (lastFillPercent >= 90) digitalWrite(LED_RED, HIGH);
      else digitalWrite(LED_RED, LOW);
    }
    lastCheck = millis();
  }
}
```