ДОДАТОК А

Програмна частина системи сповіщення про стан пристрою дозування пластичних матеріалів

```
#include <time.h> // Стандартна бібліотека Arduino для роботи із часом
#include <WiFi.h> // Стандартна бібліотека EPS для WiFi модуля
#include <WiFiClientSecure.h> // Додаткова бібліотека для створення об'єкту беспечного WiFi з'єднання
#include <UniversalTelegramBot.h> // Бібліотека для телеграм бота
#include "HX711.h" // Бібліотека для роботи з АЦП тензодатчика маси
//#include <Servo.h> // Бібліотека для роботи із сервоприводом SERVO
#include <Adafruit_GFX.h> // Графічна бібліотека для дисплею
#include <Adafruit_SSD1306.h> // Бібліотека для роботи із дисплеєм

#define default_chat_id // PASTE
std::vector<String> authorized_users = {String(default_chat_id)}; // Додаємо стандартного користувача
const String PASSWORD = "your_password"; // Пароль для авторизації в боті

// Дані для підключення до WiFi
#define WIFI_SSID "" // PASTE
#define WIFI_PASSWORD "" // PASTE

//#define WIFI_SSID "Wokwi-GUEST"
//#define WIFI_PASSWORD ""

#define BOT_TOKEN "" // Токен Telegram бота // PASTE
String chat_id = String(default_chat_id); // Стандартний користувач за замовченням

const unsigned long bot_interval = 2000; // Інтервал для перевірки повідомлень
```

```
unsigned long bot_lasttime; // Час останньої перевірки повідомлень
WiFiClientSecure secured_client; // Створення об'єкту беспечного з'єднання із інтернетом (необхідно для роботи із телеграм API)
UniversalTelegramBot bot(BOT_TOKEN, secured_client); // Створення об'єкту бот
time_t now; // Змінна для поточного часу

// Логічні флаги
bool change_minimal_status = false;
bool change_nominal_status = false;
bool change_interval_status = false;
bool settings_status = false;
bool test_flag = false;

// Визначення дефініцій для світлодіода
#define RED_PIN 32
#define GREEN_PIN 33
#define BLUE_PIN 25
#define BLACK_COLOR 0
#define RED_COLOR 1
#define GREEN_COLOR 2
const byte BLUE_COLOR = 3;
#define YELOW_COLOR 4
#define WHITE_COLOR 5

//Servo myServo; SERVO
#define SRV 15
#define SERVO_FREQUENCY 50  // Частота PWM (50 Гц)
#define SERVO_RESOLUTION 16 // Розширення PWM (16 біт)
bool servo_status = false; // Флаг для сервомотора
```

```
HX711 scale; // Створюємо об'єкт scale для датчика маси
#define DT 16 // Вказуємо номер виводу, до якого підключений вивід DT датчика
#define SCK 17 // Вказуємо номер виводу, до якого підключений вивід SCK датчика
#define max_weight 1250.0 // Визначення максимальної маси
const float calibration_factor = 1942.02; // Вводимо калібрувальний коефіцієнт
float global_weight; // Глобальна змінна для маси
unsigned int plastic_minimal = 130; // Змінна мінімальної маси
unsigned int plastic_nominal = 230; // Змінна номінальної маси
unsigned int measure_interval = 10000; // Змінна інтервалу для автоматичного вимірювання маси
unsigned long measure_lasttime; // Останній час вимірювання маси

// Параметри та дефініції для кнопки
#define BTN 18
bool button = 0;
bool button_status = 0;
unsigned long debounce_lasttime = 0; // Время последнего изменения состояния кнопки

// Ініціалізація дисплею
#define SCREEN_WIDTH 128 // Ширина у пікселях
#define SCREEN_HEIGHT 32 // Висота у пікселях
Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1); // Створення об'єкту дисплею

// Функція вимірення маси
float measureWeight() {
```

```
  float units = 0; // Задаємо змінну для вимірів в грамах та обнулюємо її для накопичення значень
  for (int i = 0; i < 2; i++) { // Усереднюємо показання, зчитуючи значення датчика 2 разів
    units += scale.get_units(); // Сумуємо показання 2 вимірів
  }
  units = (units / 2) - 780.98; // Усереднюємо показання, поділивши суму значень на 2
  // float ounces = units * 0.035274; // Переводимо масу з грам в унції
  oled.clearDisplay(); // Очищення дисплею
  oled.setCursor(1,1); // Повернення курсору на початок дисплею
  oled.setTextSize(1); oled.println("Measured weight is:\n"); // Перша строка
  oled.setTextSize(2.5); oled.println(String(units)); // Друга строка
  oled.display();

  // Перевірка на максимальну масу
  if (units > max_weight) {
      String text_bufer = "Маса більше максимальної " + String(max_weight) + " грам)!\r\n";
      text_bufer = text_bufer + "Це може не гативно вплинути на срок служби пристрою та точність вимірювань!\r\n";
    text_bufer = text_bufer + "Будь ласка, знизьте кількість пластику до меньше ніж " + String(max_weight) + ".";
    Serial.println(text_bufer);
    oled.clearDisplay(); // Очищення дисплею
    oled.setCursor(1,1); // Повернення курсору на початок дисплею
    oled.setTextSize(1); oled.println("Measured weight is:\n"); // Перша строка
    oled.setTextSize(2.5); oled.println("TO HIGH!"); // Друга строка
    oled.display();
    bot.sendMessage(chat_id, text_bufer, "");
```

```
    unsigned int led_lasttime = millis();
    bool led_status = 0;
    while (units > max_weight) {
      if (millis() - led_lasttime >= 250) {
        if (led_status == 1) {
          ledProcessing(RED_COLOR);
        }
        if (led_status == 0) {
          ledProcessing(BLACK_COLOR);
        }
        led_status = !led_status;
        led_lasttime = millis();
            Serial.println("Поточна маса все ще більше за " + String(max_weight) + " грам!\r\nЗнизьте масу пластику!\r\n");
      }
      for (int i = 0; i < 5; i++) { // Усереднюємо показання, зчитуючи значення датчика 5 разів
        units += scale.get_units(); // Сумуємо показання 5 вимірів
      }
      units = (units / 5) - 779.98;
    }
    ledProcessing(GREEN_COLOR);
  }
  return units;
}

// Функція висипання пластику
void plasticDispense(String(chat_id)) {
  String text_bufer = "Маса пластику недостатня (" + String(global_weight) + " грам)! Ініціалізація висипання пластику у " + formatTime(now) + "...";
```

```
Serial.println(text_bufer);
oled.clearDisplay(); // Очищення дисплею
oled.setCursor(1,1); // Повернення курсору на початок дисплею
oled.setTextSize(1); oled.println("Initialising of\n"); // Перша строка
oled.setTextSize(2.5); oled.println("Dispense!"); // Друга строка
oled.display();
ledProcessing(YELOW_COLOR);
bot.sendMessage(chat_id, text_bufer, "");
//myServo.write(179);

// Плавне відкриття сервопривіда
for (int angle = 0; angle <= 45; angle++) {
  int duty = map(angle, 0, 90, 5000, 8192); // Перевведення кута в значення duty
  ledcWrite(SRV, duty);
  delay(10); // Затримка для плавного повороту
}

servo_status = 1;
now = initialiseTime();

// Цикл продовжується, поки маса меньше мінімальної
while (true) {
  Serial.println("Маса все ще недостатня! " + String(global_weight) + " грам.");
  global_weight = measureWeight(); // Вимірюємо масу
  if (global_weight >= (plastic_nominal - 20 /*Упередження*/)) {
    break; // Якщо маса досягла мінімальної, то виходимо з циклу
  }
}

//myServo.write(0); // SERVO
```

```
  // Плавне закриття кришки
  for (int angle = 45; angle >= 0; angle--) {
    int duty = map(angle, 0, 90, 5000, 8192);
    ledcWrite(SRV, duty);
    delay(10);
  }

  servo_status = 0;
  ledProcessing(GREEN_COLOR);
      text_bufer = "Маса більше номінальної (" + String(plastic_nominal) + ")\r\nВиміряна маса: " + String(global_weight) + " грам. Статус OK.";
  Serial.println(text_bufer); // Выводим статус на монитор последовательного порта
  bot.sendMessage(chat_id, text_bufer, "");
}

// Функція перевірки чисельних значень
bool validateText(String text) {
  // Перевірка на порожню строку
  if (text.length() == 0) {
    return false;
  }
  // Перевірка на від'ємне значение
  if (text.charAt(0) == '-') {
    return false;
  }
  // Перевірка на чисельне значення
  for (int i = 0; i < text.length(); i++) {
    if (!isdigit(text.charAt(i))) {
      return false;
```

```
    }
  }
  int value = text.toInt(); // Перетворення String у int
  // Перевірка на значення більше одного
  if (value < 1) {
    return false;
  }
  return true;
}

// Функція перевірки нових повідомлень
void handleNewMessages(int numNewMessages) {
  Serial.print("Отримано нові повідомлення у " + formatTime(now)+ ": ");
  Serial.println(numNewMessages);
  String keyboardJson;

  // Основний цикл перевірки
  for (int i = 0; i < numNewMessages; i++) {
    chat_id = bot.messages[i].chat_id;
    String text = bot.messages[i].text;
    String from_name = bot.messages[i].from_name;
    if (from_name == "") from_name = "Гість";
    now = initialiseTime();
    Serial.println("Повідомлення від " + from_name + ": \"" + text + "\"");
    String text_bufer;

    // Перевірка на авторизацію користувача
          if (std::find(authorized_users.begin(), authorized_users.end(), chat_id) == authorized_users.end()) {
      // Користувач не авторизований
```

```
    if (text == PASSWORD) { // Перевірка пароля
      authorized_users.push_back(chat_id);
      bot.sendMessage(chat_id, "Ви успішно авторизовані!", "");
    }
    else {
       bot.sendMessage(chat_id, "Будь ласка, введіть пароль для авторизації.", ""); // Повідомлення про необхідність авторизації
    }
    continue; // Пропускаємо подальшу обробку повідомлення
  }

  // Якщо /start відправка стартового повідомлення
   if (text == "/start" && !change_nominal_status && !change_minimal_status && !change_interval_status && !settings_status) {
    text_bufer = "Вітаємо вас у боті Plastic Dispenser Bot, " + from_name + ".\r\n";
    text_bufer += "Це приклади команд, які наразі доступні:\r\n\r\n";
    text_bufer += "/measure : отримання поточної маси у повідомленні;\r\n";
    text_bufer += "/dispense : примусова ініціалізація висипання пластику;\r\n";
    text_bufer += "/settings : змінити налаштування пристрою.\r\n";
    keyboardJson = keyboardJson = "[[\"/measure\", \"/dispense\"], [\"/settings\"]]";
    bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
    Serial.println("У Telegram відправлено стартове повідомлення.");
  }

  // Якщо /measure вимірення маси
   if (text == "/measure" && !change_nominal_status && !change_minimal_status && !change_interval_status && !settings_status) {
    global_weight = measureWeight();
    text_bufer = "Виміряна маса: " + String(global_weight) + " грам.";
    Serial.println(text_bufer);
```

```
    bot.sendMessage(chat_id, text_bufer, "");
   }

   // Якщо /dispense ініціалізація примусового висипання
    if (text == "/dispense" && !change_nominal_status && !change_minimal_status &&
!change_interval_status && !settings_status) {
    text_bufer = "Спроба примусової ініціалізації висипання пластику...";
    Serial.println(text_bufer);
    bot.sendMessage(chat_id, text_bufer, "");

    global_weight = measureWeight();
    if (global_weight < plastic_nominal) {
     plasticDispense(chat_id);
    }
    else {
       text_bufer = "Виміряна маса більше номінальної: " + String(global_weight) + "
грам.";
     Serial.println(text_bufer);
     bot.sendMessage(chat_id, text_bufer, "");
    }
   }

   // Якщо /settings перехід до налаштувань
   if (text == "/settings" && !settings_status) {
    settings_status = 1;
    ledProcessing(BLUE_COLOR);
    text_bufer = "Відкрито меню налаштувань.\r\n\r\n";
        text_bufer += "Якщо ви хочере змінити налаштування - оберіть одну із
наступних команд:\r\n";
```

```
text_bufer += "/measure_interval : для зміни інтервалу автоматичних вимірювань маси (" + String(measure_interval) + ");\r\n";
text_bufer += "/plastic_nominal : для зміни номінального рівня маси пластику, яке буде підтримуватись (" + String(plastic_nominal) + ");\r\n";
text_bufer += "/plastic_minimal : для зміни мінімальної маси пластику, при якій буде виконуватись автоматичне висипання (" + String(plastic_minimal) + ");\r\n";
text_bufer += "/defaults : щоб зкинути налаштування до замовчувань.\r\n\r\n";
text_bufer += "Якщо ви хочете повернутись на початок - натисніть /main .";
oled.clearDisplay(); // Очищення дисплею
oled.setCursor(1,1); // Повернення курсору на початок дисплею
oled.setTextSize(1); oled.println("User opened menu of\n"); // Перша строка
oled.setTextSize(2.5); oled.println("Settings"); // Друга строка
oled.display();
keyboardJson = "[[\"/measure_interval\"], [\"/plastic_nominal\", \"/plastic_minimal\"],[\"/defaults\", \"/main\"]]";
bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
Serial.println("У Telegram відправлено повідомлення про налаштування.");
}

// Вихід із налаштувань
if (text == "/main" && settings_status) {
settings_status = 0;
change_interval_status = 0;
change_minimal_status = 0;
change_nominal_status = 0;
text_bufer = "Пристрій дозування повернуто до стандартного режиму.";
oled.clearDisplay(); // Очищення дисплею
oled.setCursor(1,1); // Повернення курсору на початок дисплею
oled.setTextSize(1); oled.println("Device is back to\n"); // Перша строка
oled.setTextSize(2.5); oled.println("Main mode"); // Друга строка
```

```
    oled.display();
    keyboardJson = keyboardJson = "[[\"/measure\", \"/dispense\"], [\"/settings\"]]";
    bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
    Serial.println(text_bufer);
    ledProcessing(GREEN_COLOR);
  }

  // Зкидання налаштувань до стандартних
  if (text == "/defaults" && settings_status) {
    settings_status = 0;
    change_interval_status = 0;
    change_minimal_status = 0;
    change_nominal_status = 0;
    measure_interval = 10000;
    plastic_minimal = 130;
    plastic_nominal = 230;
        text_bufer = "Налаштування за замовчуванням встановлено та пристрій повернуто до стандартного режиму.";
    oled.clearDisplay(); // Очищення дисплею
    oled.setCursor(1,1); // Повернення курсору на початок дисплею
    oled.setTextSize(1); oled.println("Device is back to\n"); // Перша строка
    oled.setTextSize(2.5); oled.println("Main mode"); // Друга строка
    oled.display();
    keyboardJson = keyboardJson = "[[\"/measure\", \"/dispense\"], [\"/settings\"]]";
    bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
    Serial.println(text_bufer);
    ledProcessing(GREEN_COLOR);
  }

  // Зміна інтервалу опиту датчику
```

```
  if (text == "/measure_interval" && settings_status && !change_interval_status) {
    change_interval_status = 1;
      text_bufer = "Щоб змінити інтервал опиту датчику маси, будь ласка, введіть ціле додатне число більше нуля у мілісекундах.";
    text_bufer += "\r\n\r\n/back : щоб повернутись назад до налаштувань.";
    bot.sendMessage(chat_id, text_bufer, "");
    Serial.println(text_bufer);
  }
  if (text != "/measure_interval" && change_interval_status && text != "/back") {
    if (validateText(text)){
      measure_interval = text.toInt();
      settings_status = 0;
      change_interval_status = 0;
      change_minimal_status = 0;
      change_nominal_status = 0;
                text_bufer = "Інтервал опиту датчику маси змінено на " + String(measure_interval) + " мілісекунд.\r\n";
      text_bufer += "Пристрій повернуто до стандартного режиму.";
      oled.clearDisplay(); // Очищення дисплею
      oled.setCursor(1,1); // Повернення курсору на початок дисплею
      oled.setTextSize(1); oled.println("Device is back to\n"); // Перша строка
      oled.setTextSize(2.5); oled.println("Main mode"); // Друга строка
      oled.display();
      keyboardJson = keyboardJson = "[[\"/measure\", \"/dispense\"], [\"/settings\"]]";
      bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
      Serial.println(text_bufer);
      ledProcessing(GREEN_COLOR);
    }
    else {
```

```
        text_bufer = "Ви ввели невірне значення, будь ласка, введіть ціле додатне число більше нуля у мілісекундах.";
      text_bufer += "\r\n\r\n/back : щоб повернутись назад до налаштувань.";
      bot.sendMessage(chat_id, text_bufer, "");
      Serial.println(text_bufer);
     }
   }

   // Зміна номінальної маси
   if (text == "/plastic_nominal" && settings_status && !change_nominal_status) {
     change_nominal_status = 1;
      text_bufer = "Щоб змінити номінальну масу, будь ласка, введіть ціле додатне число більше нуля, більше " + String(plastic_minimal) + ", та не більше ніж " + String(max_weight) + " у грамах.";
     text_bufer += "\r\n\r\n/back : щоб повернутись назад до налаштувань.";
     bot.sendMessage(chat_id, text_bufer, "");
     Serial.println(text_bufer);
   }
   if (text != "/plastic_nominal" && change_nominal_status && text != "/back") {
          if (validateText(text) && text.toInt() <= max_weight && text.toInt() > plastic_minimal){
      plastic_nominal = text.toInt();
      settings_status = 0;
      change_interval_status = 0;
      change_minimal_status = 0;
      change_nominal_status = 0;
       text_bufer = "Номінальну масу пластику змінено на " + String(plastic_nominal) + " грам.\r\n";
      text_bufer += "Пристрій повернуто до стандартного режиму.";
      oled.clearDisplay(); // Очищення дисплею
```

```
      oled.setCursor(1,1); // Повернення курсору на початок дисплею
      oled.setTextSize(1); oled.println("Device is back to\n"); // Перша строка
      oled.setTextSize(2.5); oled.println("Main mode"); // Друга строка
      oled.display();
      keyboardJson = keyboardJson = "[[\"/measure\", \"/dispense\"], [\"/settings\"]]";
      bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
      Serial.println(text_bufer);
      ledProcessing(GREEN_COLOR);
    }
    else {
        text_bufer = "Ви ввели невірне значення, будь ласка, введіть ціле додатне число більше нуля, більше " + String(plastic_minimal) + ", та не більше ніж " + String(max_weight) + " у грамах.";
      text_bufer += "\r\n\r\n/back : щоб повернутись назад до налаштувань.";
      bot.sendMessage(chat_id, text_bufer, "");
      Serial.println(text_bufer);
    }
  }

  // Зміна мінімальної маси
  if (text == "/plastic_minimal" && settings_status && !change_minimal_status) {
    change_minimal_status = 1;
      text_bufer = "Щоб змінити мінімальну масу, будь ласка, введіть ціле додатне число більше нуля, менше " + String(plastic_nominal) + " та менше ніж " + String(max_weight) + " у грамах.";
    text_bufer += "\r\n\r\n/back : щоб повернутись назад до налаштувань.";
    bot.sendMessage(chat_id, text_bufer, "");
    Serial.println(text_bufer);
  }
  if (text != "/plastic_minimal" && change_minimal_status && text != "/back") {
```

```
        if (validateText(text) && text.toInt() <= max_weight && text.toInt() < plastic_nominal){
      plastic_minimal = text.toInt();
      settings_status = 0;
      change_interval_status = 0;
      change_minimal_status = 0;
      change_nominal_status = 0;
       text_bufer = "Мінімальну масу пластику змінено на " + String(plastic_minimal) + " грам.\r\n";
      text_bufer += "Пристрій повернуто до стандартного режиму.";
      oled.clearDisplay(); // Очищення дисплею
      oled.setCursor(1,1); // Повернення курсору на початок дисплею
      oled.setTextSize(1); oled.println("Device is back to\n"); // Перша строка
      oled.setTextSize(2.5); oled.println("Main mode"); // Друга строка
      oled.display();
      keyboardJson = keyboardJson = "[[\"/measure\", \"/dispense\"], [\"/settings\"]]";
      bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
      Serial.println(text_bufer);
      ledProcessing(GREEN_COLOR);
     }
     else {
        text_bufer = "Ви ввели невірне значення, введіть ціле додатне число більше нуля, менше " + String(plastic_nominal) + " та менше ніж "+ String(max_weight) + " у грамах.";
      text_bufer += "\r\n\r\n/back : щоб повернутись назад до налаштувань.";
      bot.sendMessage(chat_id, text_bufer, "");
      Serial.println(text_bufer);
     }
   }
```

```
      if (text == "/back" && (change_minimal_status || change_nominal_status || change_interval_status)) {
    change_interval_status = 0;
    change_minimal_status = 0;
    change_nominal_status = 0;
    text_bufer = "Відкрито меню налаштувань.\r\n\r\n";
        text_bufer += "Якщо ви хочере змінити налаштування - оберіть одну із наступних команд:\r\n";
          text_bufer += "/measure_interval : для зміни інтервалу автоматичних вимірювань маси (" + String(measure_interval) + ");\r\n";
     text_bufer += "/plastic_nominal : для зміни номінального рівня маси пластику, яке буде підтримуватись (" + String(plastic_nominal) + ");\r\n";
     text_bufer += "/plastic_minimal : для зміни мінімальної маси пластику, при якій буде виконуватись автоматичне висипання (" + String(plastic_minimal) + ");\r\n";
    text_bufer += "/defaults : щоб зкинути налаштування до замовчувань.\r\n\r\n";
    text_bufer += "Якщо ви хочете повернутись на початок - натисніть /main .";
    keyboardJson = keyboardJson = "[[\"/measure\", \"/dispense\"], [\"/settings\"]]";
    bot.sendMessageWithReplyKeyboard(chat_id, text_bufer, "", keyboardJson, true);
    Serial.println("У Telegram відправлено повідомлення про налаштування.");
   }

   // Деавторизація користувача
   if (text == "/logout") {
    unauthorizeUser(chat_id);
    bot.sendMessage(chat_id, "Ви вийшли з системи.", "");
   }
  }
}

// Функція отримання часу
```

```
time_t initialiseTime() {
  configTime(2 * 3600, 1*3600, "pool.ntp.org"); // Визначення часу UTC через NTP, встановлення часової зони
 time_t nowBufer = time(nullptr); // Отримання поточного часу
 unsigned int led_lasttime;
 bool led_status;
 while (nowBufer < 60* 60 * 24 * 365 * 30) {
  if (millis() - led_lasttime >= 250) {
   led_lasttime = millis();
   if (led_status == 1) {
    ledProcessing(YELOW_COLOR);
   }
   if (led_status == 0) {
    ledProcessing(BLACK_COLOR);
   }
   led_status = !led_status;
   Serial.print(".");
   nowBufer = time(nullptr);
  }
 }
 return nowBufer;
}

// Функція форматування часу
String formatTime(time_t rawTime) {
 struct tm * timeinfo = localtime(&rawTime);
 char buffer[20];
 strftime(buffer, sizeof(buffer), "%H:%M:%S %m.%d.%Y", timeinfo);
 return String(buffer);
}
```

```
// Функція обробки натискання кнопки
void buttonProcessing() {
  if (digitalRead(BTN) != button) {
    debounce_lasttime = millis(); // Зкидання часу дебаунса
    button = digitalRead(BTN); // Оновлення стану кнопки
  }

  if ((millis() - debounce_lasttime) > 50) {
    if (!button && !button_status) {
      button_status = 1;
      Serial.println("Кнопку натиснуто!");
      unsigned int button_time = millis();
      unsigned int button_bufer;
      bool sleep_status = 0;
      while (!button){
        button_bufer = millis() - button_time;
        if (digitalRead(BTN) != button) {
          debounce_lasttime = millis(); // Зкидання часу дебаунса
          button = digitalRead(BTN); // Оновлення стану кнопки
        }
        if ((button_bufer >= 10000) && !sleep_status) {
          sleep_status = 1;
          Serial.println("Тут повинна бути спроба входу в сплячий режим...");
          /*Serial.println("Entering light sleep...");
          // Налаштування кнопки для пробудження ESP32
          delay (2000);
              esp_sleep_enable_ext0_wakeup(GPIO_NUM_18, 0); // Пробудження при
натисканні кнопки (логічний нуль)
```

```
        // Вхід в режим легкого сну
        esp_light_sleep_start();

        // Пробудження після натискання кнопки
        Serial.println("Wakeup from light sleep");
        */
        sleep_status = 1;
       }
      }
          Serial.println("Кнопка була натиснута " + String(button_bufer / 1000) + " секунд.");
    }

    if (button && button_status) {
      button_status = 0;
      String text_bufer;
       text_bufer = "Кнопку відпущено!\r\nСпроба примусової ініціалізації висипання пластику...";
      Serial.println(text_bufer);
      bot.sendMessage(chat_id, text_bufer, "");
      global_weight = measureWeight();
      if (global_weight < plastic_nominal) {
        plasticDispense(chat_id);
      }
      else {
         text_bufer = "Виміряна маса більше номінальної: " + String(global_weight) + " грам.";
        Serial.println(text_bufer);
        bot.sendMessage(chat_id, text_bufer, "");
      }
```

```
    }
  }
}

// Функція для роботи зі світлодіодом
void ledProcessing (byte color) {
  switch (color)
  {
  case BLACK_COLOR:
    analogWrite(RED_PIN, 0);
    analogWrite(GREEN_PIN, 0);
    analogWrite(BLUE_PIN, 0);
    break;

  case RED_COLOR:
    analogWrite(RED_PIN, 255);
    analogWrite(GREEN_PIN, 0);
    analogWrite(BLUE_PIN, 0);
    break;

  case GREEN_COLOR:
    analogWrite(RED_PIN, 0);
    analogWrite(GREEN_PIN, 255);
    analogWrite(BLUE_PIN, 0);
    break;

  case 3:
    analogWrite(RED_PIN, 0);
    analogWrite(GREEN_PIN, 0);
    analogWrite(BLUE_PIN, 255);
```

```
    break;

  case WHITE_COLOR:
    analogWrite(RED_PIN, 255);
    analogWrite(GREEN_PIN, 255);
    analogWrite(BLUE_PIN, 255);
    break;

  case YELOW_COLOR:
    analogWrite(RED_PIN, 255);
    analogWrite(GREEN_PIN, 255);
    analogWrite(BLUE_PIN, 0);
    break;

  default:
    analogWrite(RED_PIN, 255);
    analogWrite(GREEN_PIN, 255);
    analogWrite(BLUE_PIN, 255);
    break;
  }
}

// Функція деавторизації користувача
void unauthorizeUser(String chat_id) {
  auto it = std::find(authorized_users.begin(), authorized_users.end(), chat_id);
  if (it != authorized_users.end()) {
    authorized_users.erase(it);
  }
}
```

```
void setup() {
  Serial.begin(115200);

  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
  ledProcessing(RED_COLOR);

  //myServo.attach(SRV); // SERVO
  //myServo.write(0); // SERVO
   ledcAttach(SRV, SERVO_FREQUENCY, SERVO_RESOLUTION); // Ініціалізація сервомотора
  ledcWrite(SRV, 5000); // Встановлення сервомотору на початкове положення

  // Ініціалізація дисплею
  if (!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("failed to start SSD1306 OLED"));
    while (1);
  }
     delay(2000); // Затримка для уникнення помилок із дисплеєм при його ініціалізації
  oled.clearDisplay(); // Очищення дисплею
  oled.setTextColor(WHITE); // Колір дисплею
  oled.setCursor(1, 1); // Встановлення курсору на початку екрану
  oled.setTextSize(1); oled.println("Device is\nInitialising...");
  oled.display();
  Serial.println("Дисплей ініціалізовано успішно.");

  pinMode(BTN, INPUT_PULLUP); // Ініціалізуємо порт для кнопки
  scale.begin(DT, SCK); // Ініціалізуємо роботу з датчиком маси
```

```
    scale.set_scale(); // Виконуємо вимірювання значення без калібрувального коефіцієнта
  /*scale.tare(); // Скидаємо значення маси на датчику на 0*/
  scale.set_scale(calibration_factor); // Встановлюємо калібрувальний коефіцієнт
    Serial.println("Контрольне вимірювання маси 1: " + String(measureWeight()) + "грам.");
    Serial.println("Контрольне вимірювання маси 2: " + String(measureWeight()) + "грам.");
  global_weight = measureWeight();
     Serial.println("Контрольне вимірювання маси 3: " + String(global_weight) + "грам.");

  oled.clearDisplay(); // Очищення дисплею
  oled.setTextColor(WHITE); // Колір дисплею
  oled.setCursor(1, 1); // Встановлення курсору на початку екрану
  oled.setTextSize(1); oled.println("Device is connecting\n");
  oled.setTextSize(2); oled.println("to WiFi..."); // set text
  oled.display();

  // Спроба підключення до Wi-Fi мережі
  Serial.print("Підключення до Wi-Fi мережі із SSID ");
  Serial.println(/*WIFI_SSID*/); // Розкоментувати
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
       secured_client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Додавання корінного сертифікату для api.telegram.org
  unsigned int led_lasttime;
  bool led_status = 1;
  while (WiFi.status() != WL_CONNECTED) {
    if (millis() - led_lasttime >= 250) {
      if (led_status == 1) {
```

```
      ledProcessing(YELOW_COLOR);
    }
    if (led_status == 0) {
      ledProcessing(BLACK_COLOR);
    }
    led_status = !led_status;
    led_lasttime = millis();
    Serial.print(".");
  }
 }
 Serial.println();
 Serial.print("WiFi підключено. IP адреса: ");
 Serial.println(/*WiFi.localIP()*/); // Розкоментувати
 Serial.println();

 // Отримання поточного часу
 Serial.println("Отримання часу із серверу: ");
 now = initialiseTime();
 Serial.println();
 Serial.println("Отримано час " + formatTime(now));
 Serial.println();
  String text_bufer = "Пристрій в мережі і успішно ініціалізований!\r\nКонтрольна маса " + String(global_weight) + " грам." + "\r\nВикористайте команду /start або іншу для початку.";
 Serial.println(text_bufer);
 oled.clearDisplay(); // Очищення дисплею
 oled.setCursor(1, 1); // Встановлення курсору на початку екрану
 oled.setTextSize(1); oled.println("Device is\n");
 oled.setTextSize(2); oled.println("Started!");
 oled.display();
```

```
  bot.sendMessage(chat_id, text_bufer, "");
  delay(100);
  measure_lasttime = millis();
  bot_lasttime = millis();
  Serial.println("Початок циклу:\r\n");
  ledProcessing(GREEN_COLOR);
}

void loop() {

  // Обробка кнопки
   if (!change_nominal_status && !change_minimal_status && !change_interval_status
&& !settings_status)
  {
    buttonProcessing();
  }

  // Перевірка на нові повідомлення
  if ((millis() - bot_lasttime >= bot_interval) && !servo_status) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while (numNewMessages) {
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    bot_lasttime = millis();
  }

  // Обробка кнопки
```

```
  if (!change_nominal_status && !change_minimal_status && !change_interval_status && !settings_status)
  {
   buttonProcessing();
  }

  // Функція автоматичного вимірювання маси та автовисипання при мінімальній масі пластику
  if ((millis() - measure_lasttime >= measure_interval) && !change_nominal_status && !change_minimal_status && !change_interval_status && !settings_status){
   global_weight = measureWeight();
   if (global_weight < plastic_minimal){
    plasticDispense(String(chat_id));
   }
   else {
        Serial.println("Автоматично вимірюяна маса: " + String(global_weight) + " грам.");
   }
   measure_lasttime = millis();
  }

  // Обробка кнопки
  if (!change_nominal_status && !change_minimal_status && !change_interval_status && !settings_status)
  {
   buttonProcessing();
  }
}
```

ДОДАТОК Б

Апробація результатів наукових досліджень

Міністерство освіти і науки України

NURE

Харківський національний університет радіоелектроніки

ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2024

(Випуск 1)

[електронне видання]

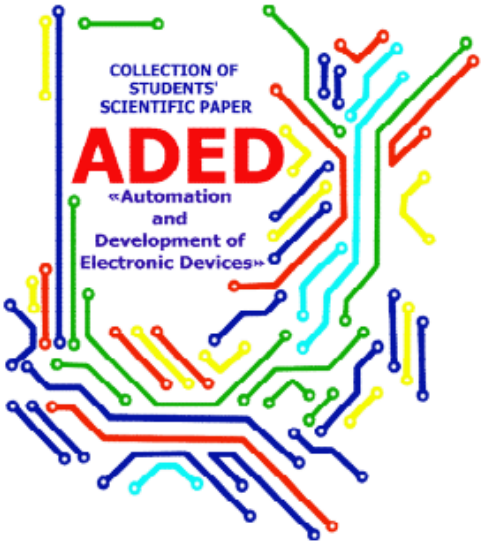
http://nure.ua/department/kafedra-komp-yuterno-integrovanih-tehnologiy-avtomatizatsiyi-ta-mehatroniki-kitam

http://itez.zntu.edu.ua/

http://kafea.kdu.edu.ua

Харків 2024

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(КІТАР)



ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2024

(Випуск 1)

[електронне видання]

Харків 2024

Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2024) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2024. – Вип. 1. – 207с.

Collection of Students' Scientific Paper «Automation and Development Of Electronic Devices» ADED-2024 Part 1 (Key infrastructure 2024) - Kharkiv/ The Editorial.: Nevlyudov I.Sh. (head), that all. Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2024. – 207p with.

Рекомендовано рішенням
Науково-технічної ради
Харківського національного
університету радіоелектроніки
протокол №6 від 29.11.2018

Рекомендовано рішенням Вченої ради
факультету Автоматики і комп'ютеризованих технологій
Харківського національного
університету радіоелектроніки
протокол № 10 від 20.05.2024

Збірник містить наукові статті здобувачів першого (бакалаврського), другого (магістерського) рівнів вищої освіти кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР) Харківського національного університету радіоелектроніки, кафедри Інформаційних технологій електронних засобів (ІТЕД) Запорізького національного технічного університету та кафедри Електронних апаратів (ЕА) Кременчуцького національного університету ім. М. Остроградського які навчаються за спеціальностями: 151 Автоматизація та комп'ютерно-інтегровані технології, 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка; 172 Телекомунікації та радіотехніка, 171 Електроніка та 163 Біомедична інженерія. Статті надані в авторській редакції.

ЗМІСТ

УДК 621.67

РОЗРОБКА СИСТЕМИ СПОВІЩЕННЯ ПРО СТАН ПРИСТРОЮ ДОЗУВАННЯ ПЛАСТИЧНИХ МАТЕРІАЛІВ

В. О. Соломатін
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: vitalii.solomatin@nure.ua

***Анотація*:** У сучасному виробництві 3D-друку є постійна потреба в точному та ефективному дозуванні сипучих матеріалів, зокрема гранульованого пластику. У зв'язку з цим виникає необхідність розробки системи, що забезпечує автоматичне дозування та контроль залишкової кількості матеріалу. У даній статті розроблена структурна схема системи сповіщення про стан пристрою дозування пластичних матеріалів та запропоновано апаратна і програмна реалізація. Дану систему може використовувати в промислових або особистих умовах, що робить її актуальною та практичною для широкого кола користувачів.

***Ключові слова*:** 3D-друк, дозування, пластикові гранули, автоматизація, датчик ваги, сповіщення.

DEVELOPMENT OF A WARNING SYSTEM FOR THE STATUS OF A PLASTIC MATERIALS DOSING DEVICE

V. O. Solomatin
Kharkiv National University of Radio Electronics
14, Nauky Ave. 61166, Kharkiv, Ukraine
E-mail: vitalii.solomatin@nure.ua

Abstract: In modern 3D printing production, there is a constant need for accurate and efficient dosing of bulk materials, in particular granular plastic. In this regard, there is a need to develop a system that provides automatic dosing and control of the residual amount of material. In this article, we develop a block diagram of a system for notifying the status of a plastic dosing device and propose hardware and software implementation. This system can be used in industrial or personal settings, which makes it relevant and practical for a wide range of users.

Key words: 3D printing, dosing, plastic granules, automation, weight sensor, notification.

Зростаюча популярність та розвиток 3D-друку вимагають постійного удосконалення технологій та процесів, пов'язаних з друкуванням. Однією з ключових проблем, що виникають у процесі використання 3D-принтерів, є точне та ефективне дозування сипучих матеріалів, зокрема гранульованого пластику. Для вирішення цієї проблеми метою даної наукової роботи є розробка системи сповіщення про стан пристрою дозування пластичних матеріалів. Основною задачею є створення прототипу системи, що базується на використанні сервомотора та датчика ваги, для автоматичного контролю та управління процесом дозування матеріалу та забезпечення оператора вчасною інформацією щодо рівня запасів. Реалізація цієї системи може значно полегшити та покращити процес друкування на 3D-принтерах, що робить дане дослідження актуальним та важливим для сучасної промисловості.

3D-принтер є пристроєм, призначеним для створення тривимірних об'єктів за допомогою нанесення шарів матеріалу один на один. Основним матеріалом для друку на таких принтерах часто є пластик, який може бути у формі філаменту або гранул [1]. Для створення об'єкта на 3D-принтері необхідно матеріал, який постачається до друкарської головки (екструдера), де

ДОДАТОК В

Демонстраційний графічний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК.565100.010 ПЗ	Пояснювальна записка	1	104 с.
			Інші документи		
	2		Пояснювальна записка	1	Ел. варіант
			Додаток А	1	Ел. варіант
			Програмна частина системи		
			сповіщення про стан		
			пристрою дозування		
			пластичних матеріалів		
			Додаток Б	1	Ел. варіант
			Апробація наукових досліджень		
			Додаток В		
			Демонстраційний графічний матеріал	1	12 с.

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮІК.565100.010 ПЗ			
Розроб.		Соломатін В. О.			Розробка системи автоматизації для сповіщення про стан пристрою дозування пластичних матеріалів для 3D-друку	Літера	Аркуш	Аркушів
Перевір.		Сичова О. В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н. П.						
Затв.		Невлюдов І.Ш.						