Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Самарский государственный технический университет»
(ФГБОУ ВО «СамГТУ»)

Институт «Автоматики и инженерных технологий»

Разработка люксметра на базе Arduino UNO с использованием FreeRTOS

Лабораторная работа №5, отчёт

Выполнили студенты

3 курса, 3-ИАИТ-110 группы

Беляков Даниил Андреевич
Питьев Дмитрий Артёмович
Чалый Антон Викторович

Самара, 2025 г.

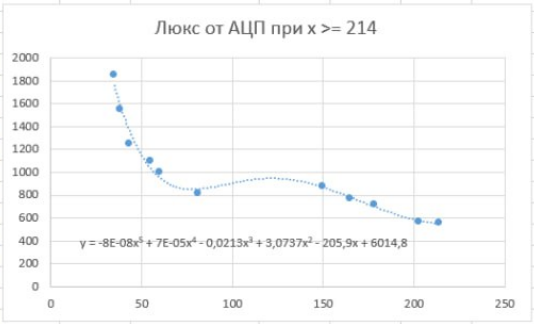
Цель работы – Цель работы: научиться измерять освещённость с помощью фоторезистора (LDR), провести экспериментальное измерение его характеристик, реализовать интерполяцию экспериментальных данных, анализ спектра сигнала, разработать программу на Arduino UNO с использованием FreeRTOS, реализующей три независимые задачи и обмен данными через очередь FreeRTOS, вывод значений на I2C LCD 16x2 и в последовательный порт, а также получение временной метки от часов реального времени DS1302.

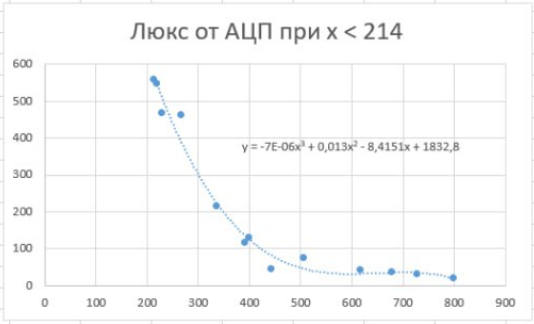
Задание 1:

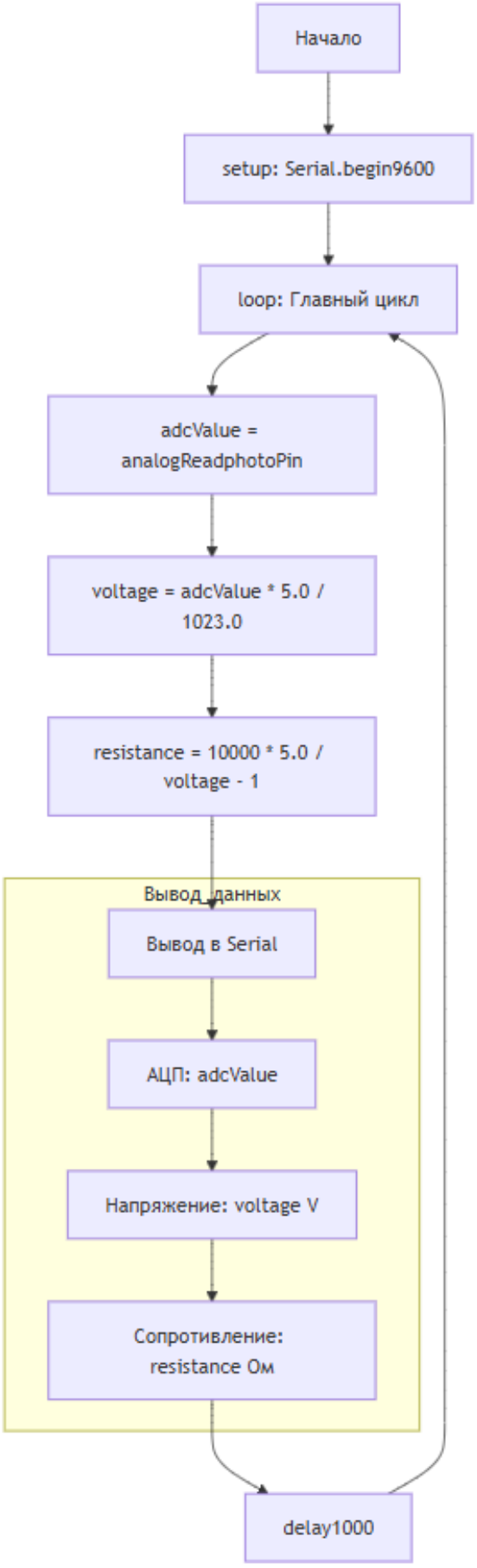
Скетч без FreeRTOS для считывания показаний АЦП при различных уровнях освещенности и вывода информации в терминал

```
int photoPin = A0;     // Пин подключения фоторезистора
int adcValue;          // Сырое значение АЦП (0-1023)
float voltage;         // Напряжение (0-5V)
int resistance;        // Сопротивление фоторезистора

void setup() {
  Serial.begin(9600);
}

void loop() {
  // Сырое значение АЦП
  adcValue = analogRead(photoPin);

  // Расчет напряжения
  voltage = adcValue * (5.0 / 1023.0);

  // Расчет сопротивления фоторезистора
  resistance = 10000 * (5.0 / voltage - 1); // для резистора 10 Ом

  Serial.print("АЦП: ");
  Serial.print(adcValue);
  Serial.print(" | Напряжение: ");
  Serial.print(voltage);
  Serial.print("V | Сопротивление: ");
  Serial.print(resistance);
  Serial.println(" Ом");

  delay(1000);
}
```

АЦП	Освещенность
35	1850
38	1550
43	1250
55	1100
60	1000
81	816
150	879
165	768
178	718
203	563
214	558
219	545
230	465
268	460
337	215
392	114
400	130
443	43
506	75
618	40
680	36
729	30
800	20






Начало
setup: Serial.begin9600
loop: Главный цикл
adcValue = analogReadphotoPin
voltage = adcValue * 5.0 / 1023.0
resistance = 10000 * 5.0 / voltage - 1
Вывод данных
Вывод в Serial
АЦП: adcValue
Напряжение: voltage V
Сопротивление: resistance Ом
delay1000

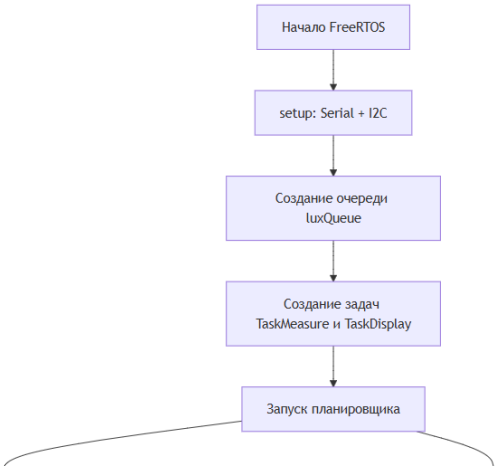
Задание 2: Работа с FreeRTOS

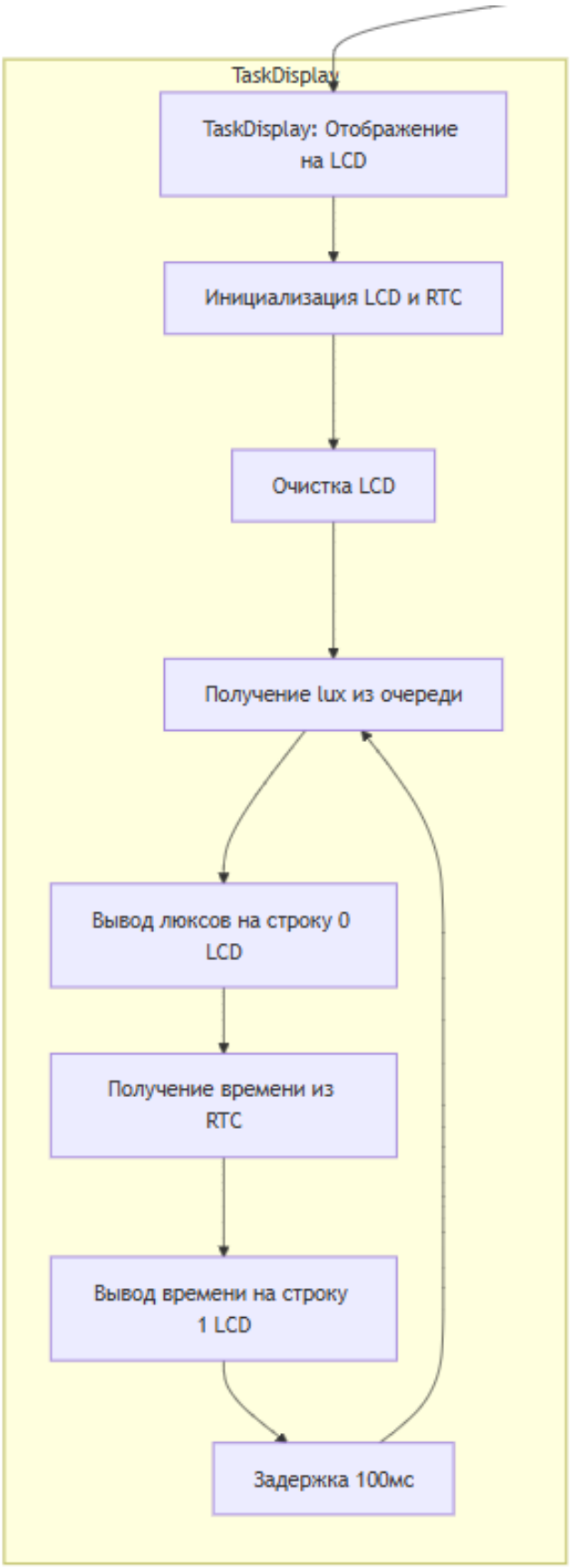
Расчёт значения Люкс исходя из аппроксимации по АЦП и вывод значения Люкс и времени на дисплей

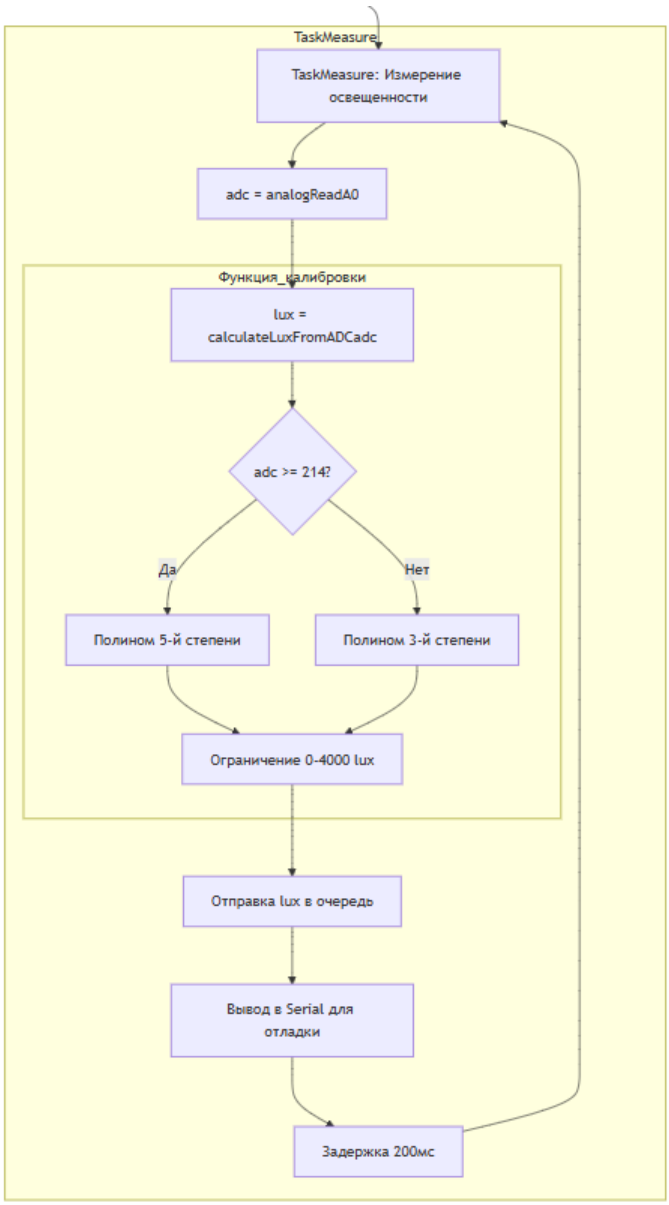
```
#include <Arduino_FreeRTOS.h>
#include <queue.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DS1302.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
const int RTC_RST = 8, RTC_DAT = 7, RTC_CLK = 6;
DS1302 rtc(RTC_RST, RTC_DAT, RTC_CLK);
QueueHandle_t luxQueue;

float calculateLuxFromADC(int adcValue) {
  float x = (float)adcValue; // Преобразуем в float для расчетов
  float y;

  // Кусочно-заданная полиномиальная функция
  if (x >= 214) {
    // y = -8E-08*x^5 + 7E-05*x^4 - 0.0213*x^3 + 3.0737*x^2 - 205.9*x +
       6014.8
    y = -8e-08 * pow(x, 5) +
         7e-05 * pow(x, 4) -
         0.0213 * pow(x, 3) +
         3.0737 * pow(x, 2) -
         205.9 * x +
         6014.8;
  } else {
    // y = -7E-06*x^3 + 0.013*x^2 - 8.4151*x + 1832.8
    y = -7e-06 * pow(x, 3) +
         0.013 * pow(x, 2) -
         8.4151 * x +
         1832.8;
  }

  // Защита от некорректных значений
  if (y < 0) y = 0;
  if (y > 4000) y = 4000;

  return y;
}

void TaskMeasure(void *pv) {
  (void) pv;
  float lux;

  for (;;) {
    int adc = analogRead(A0);
    lux = calculateLuxFromADC(adc);

    // Отправляем в очередь
```

```
    if (xQueueSend(luxQueue, &lux, 10 / portTICK_PERIOD_MS) != pdPASS) {
      // Если очередь полна, пропускаем измерение
    }

    // Отладка в Serial
    Serial.print("ADC: ");
    Serial.print(adc);
    Serial.print(" -> Lux: ");
    Serial.println(lux, 1);

    vTaskDelay(200 / portTICK_PERIOD_MS);
  }
}

void TaskDisplay(void *pv) {
  (void) pv;
  float lux = 0;

  // Инициализация LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.print("Luxmeter RTOS");

  // Инициализация RTC
  rtc.halt(false);
  rtc.writeProtect(false);
  // При первом запуске - снимаем комментарий для инициализации времени
  // Потом - комментируем обратно
  //Time t(2025, 12, 5, 15, 48, 0, Time::kMonday);
  //rtc.time(t);

  vTaskDelay(1000 / portTICK_PERIOD_MS);
  lcd.clear();

  for (;;) {
    if (xQueueReceive(luxQueue, &lux, 100 / portTICK_PERIOD_MS) == pdPASS)
       {
      // Строка 1: Освещенность
      lcd.setCursor(0, 0);
      lcd.print("Lux: ");
      lcd.print(lux, 0);
      lcd.print(" lx  ");

      // Строка 2: Время от RTC
      Time now = rtc.time();
      lcd.setCursor(0, 1);
      if (now.hr < 10) lcd.print("0");
      lcd.print(now.hr);
      lcd.print(":");
      if (now.min < 10) lcd.print("0");
      lcd.print(now.min);
      lcd.print(":");
```

```
      if (now.sec < 10) lcd.print("0");
      lcd.print(now.sec);
      lcd.print("    ");
    }

    vTaskDelay(100 / portTICK_PERIOD_MS);
  }
}

void setup() {
  Serial.begin(9600);
  Wire.begin();

  luxQueue = xQueueCreate(3, sizeof(float));

  xTaskCreate(TaskMeasure, "Measure", 80, NULL, 1, NULL);
  xTaskCreate(TaskDisplay, "Display", 150, NULL, 1, NULL);

  vTaskStartScheduler();
}

void loop() {}
```




TaskDisplay
TaskDisplay: Отображение на LCD
Инициализация LCD и RTC
Очистка LCD
Получение lux из очереди
Вывод люксов на строку 0 LCD
Получение времени из RTC
Вывод времени на строку 1 LCD
Задержка 100мс



Вывод: В ходе работы успешно реализован комплексный проект на базе Arduino UNO, охватывающий ключевые аспекты embedded-систем: от взаимодействия с аналоговыми датчиками до организации многозадачной среды.

Ответы на контрольные вопросы

1. **В чем заключается основное преимущество использования FreeRTOS по сравнению с "зацикленным"(super-loop) управлением в Arduino?**

   Вытесняющая многозадачность с гарантированным временем отклика и безопасной синхронизацией задач. В super-loop любая блокирующая операция парализует всю систему.

2. **Объясните назначение и принцип работы очереди (Queue) в FreeRTOS. Почему для передачи данных между задачами предпочтительнее использовать очереди, а не общие глобальные переменные?**

Очередь — механизм безопасной передачи данных FIFO между задачами и прерываниями. Предпочтительнее из-за автоматической синхронизации, исключения состояния гонки и изоляции данных.

3. **Какую функцию выполняет функция pdMS_TO_TICKS()? Почему при работе с vTaskDelay() нельзя просто передать время в миллисекундах?**

   Преобразует миллисекунды в тики (ticks) планировщика с учётом configTICK_RATE_HZ. vTaskDelay() работает только с тиками для обеспечения портируемости кода между системами с разной частотой тактирования.

4. **Объясните физический принцип работы фоторезистора (LDR) и почему его характеристика часто нелинейна.**

   Фоторезистивный эффект в полупроводнике: фотоны генерируют электрон-дырочные пары, что увеличивает концентрацию носителей заряда и снижает сопротивление. Характеристика нелинейна из-за степенного закона: $R_L \approx k/E^\gamma$, где $\gamma \approx 0.5 - 1.0$.

5. **Опишите, как нелинейная характеристика фоторезистора компенсируется на этапе калибровки.**

   Создаётся калибровочная таблица (LUT) или вычисляются коэффициенты аппроксимирующей функции, например, логарифмической: $\ln(R) = A + B \cdot \ln(E)$. После измерения напряжения и вычисления R_L по таблице/формуле определяется освещённость E.

6. **Если бы задача измерения освещенности имела самый низкий приоритет, как бы это сказалось на точности показаний при резком изменении освещенности?**

   Привело бы к увеличению задержки измерения, динамической погрешности и возможному пропуску быстрых изменений освещённости.

7. **Как выбор значения фиксированного резистора Rf в делителе влияет на динамический диапазон и разрешение измерения?**

   Определяет центр диапазона измерений. Оптимально $R_f \approx R_{LDR}$ в середине рабочего диапазона. Неверный выбор снижает чувствительность на краях диапазона.

8. **Какие ограничения по разрешению и диапазону накладывает 10-битный ADC Arduino (0–1023)?**

   Разрешение: $\frac{V_{ref}}{1024}$ (4.9 мВ при 5В). Диапазон: 0-V_{ref} (обычно 0-5В). Требует точного подбора R_f для использования всего диапазона ADC.

9. **Опишите процедуру калибровки: какие шаги, какие источники ошибок нужно учесть и как минимизировать систематическую погрешность.**

   1) Подготовка эталонных источников света; 2) Измерение значений АЦП для каждой точки; 3) Построение калибровочной кривой; 4) Аппроксимация данных. Источники ошибок: нестабильность источников, шум ADC, температурный дрейф. Минимизация: усреднение, термостабилизация, использование прецизионных эталонов.

10. **Как правильно выбрать точки калибровки по диапазону освещённости (например, равномерно по логарифмической шкале или по линейной)? Обоснуйте.**

    Равномерно по логарифмической шкале, так как характеристика LDR обычно логарифмическая. Это обеспечивает равномерную относительную точность во всём широком динамическом диапазоне.

11. **Как оценить воспроизводимость измерений (повторяемость) для каждой точки калибровки? Какие статистические величины использовать?**

    Многократные измерения в одинаковых условиях. Использовать: среднее значение $\bar{x}$, стандартное отклонение σ, коэффициент вариации $C_v = \frac{\sigma}{\bar{x}} \cdot 100\%$.

12. **Как рассчитать абсолютную, относительную и приведенную погрешности измерения для каждой точки; приведите формулы.**

    Абсолютная: $\Delta = x_i - x$; относительная: $\delta = \frac{\Delta}{x} \cdot 100\%$; приведённая: $\gamma = \frac{\Delta}{X_N} \cdot 100\%$, где X_N — предел измерения (например, 1023 для АЦП).

13. **Объясните назначение каждого поля функции xQueueCreate(uxQueueLen uxItemSize) и почему важно правильно выбрать длину очереди.**

    uxQueueLength — максимальное количество элементов в очереди; uxItemSize — размер каждого элемента в байтах. Слишком короткая очередь приводит к потерям данных, слишком длинная — к неэффективному использованию памяти.

14. **Чем отличаются блокирующие (blocking) и неблокирующие (non-blocking) вызовы xQueueSend/Receive, и когда следует использовать второй режим?**

    Блокирующие: задача ждёт, пока операция не станет возможной (очередь освободится или появятся данные). Неблокирующие: функция немедленно возвращает статус (успех или ошибка). Второй режим используют в обработчиках прерываний (ISR) и высокоприоритетных задачах.

15. **Что делает макрос pdMS_TO_TICKS() и почему предпочтительно его использовать при вызовах vTaskDelay()?**

    См. ответ на вопрос 3. Обеспечивает портируемость и корректное преобразование времени независимо от частоты тактирования планировщика.

16. **Как обеспечить атомарный доступ к общим ресурсам (например, к буферу дисплея) при многозадачности в FreeRTOS? Приведите примеры (мьютекс, секция критической области).**

    Использовать мьютекс: xSemaphoreCreateMutex(), xSemaphoreTake(), xSemaphoreGiv
    Или критические секции: taskENTER_CRITICAL(), taskEXIT_CRITICAL() (для очень коротких операций).

17. **Как реализовать таймаут в ожидании отправки в очередь, чтобы задача измерения не блокировала систему при заполненной очереди?**

    Использовать параметр xTicksToWait в xQueueSend: xQueueSend(queue, &data, pdMS_TO_TICKS(100)). При таймауте функция вернёт errQUEUE_FULL.

18. **Опишите, как вы реализуете приоритеты задач в этой системе и что произойдёт, если все три задачи получат одинаковый приоритет.**

    Приоритеты задаются в параметре uxPriority функции xTaskCreate(). При одинаковом приоритете задачи выполняются в режиме round-robin, что может увеличить время отклика на события.

19. **Какие свойства стека задач следует контролировать при отладке (переполнение стека), и как это выявить?**

    Контролировать используемый размер стека с помощью uxTaskGetStackHighWaterM
    Включить configCHECK_FOR_STACK_OVERFLOW для автоматической проверки.

20. **Как отследить и диагностировать потерю данных при передаче via Queue (очередь заполнена) — какие логи и индикации добавить?**

    Проверять возвращаемое значение xQueueSend(). При ошибке errQUEUE_FULL увеличивать счётчик потерь и выводить его значение в Serial или индицировать светодиодом.

21. **Если дисплей периодически не обновляется, какие возможные причины и последовательность проверки?**

    1) Блокировка в очереди; 2) Переполнение стека задачи дисплея; 3) Deadlock; 4) Проблема с аппаратной шиной I2C.

22. **Как замерить реальный интервал срабатывания задачи vTaskMeasureLun (например, 200 ms) и как интерпретировать отклонения?**

    Сохранять время начала выполнения (xTaskGetTickCount()) и вычислять разницу с временем окончания. Отклонения указывают на перегрузку системы или блокировки.

23. **Какие тесты вы проведёте, чтобы проверить реакцию системы на быструю последовательную смену освещения (микросекундные/-миллисекундные вспышки)?**

    Подача импульсов от светодиода с известной длительностью с помощью генератора. Синхронизация и осциллограф для измерения времени реакции системы.

24. **Как влияет использование функции Serial.print внутри часто срабатывающей задачи на планирование FreeRTOS и как это оптимизировать?**

    Serial.print() — блокирующая и медленная операция, вызывает задержки и частые переключения контекста. Оптимизация: буферизация вывода, вынос в отдельную низкоприоритетную задачу, повышение скорости UART.

25. **Как добавить фильтрацию/усреднение (moving average, экспоненциальное сглаживание) для уменьшения шума измерений? Приведите формулы и параметры.**

    Экспоненциальное сглаживание: $y_n = \alpha \cdot x_n + (1 - \alpha) \cdot y_{n-1}$, где $\alpha \in (0, 1)$. Скользящее среднее: $y_n = \frac{1}{N} \sum_{i=0}^{N-1} x_{n-i}$.

26. **Как реализовать адаптивную частоту опроса в зависимости от скорости изменения освещённости (увеличивать частоту при быстром изменении)?**

    Вычислять производную $|\Delta E|$ (скорость изменения), и в зависимости от её величины изменять период задачи: $T = T_{base} \cdot k^{-|\Delta E|}$, где $k > 1$.

27. **Предложите архитектуру для сохранения калибровочных данных в энергонезависимую память (EEPROM) с контролем целостности.**

    Структура данных: заголовок (идентификатор, контрольная сумма CRC), массив калибровочных точек. При загрузке проверять CRC. Реализовать wear leveling для увеличения срока службы EEPROM.

28. **Как добавить сетевой интерфейс (например, ESP8266/ESP32 как модуль) для удалённого логирования показаний и как это повлияет на приоритеты задач?**

Добавить задачу сетевой передачи с высоким приоритетом для своевременной отправки данных. Приоритеты в системе: сетевая задача > задача измерений > задача отображения.

29. **Как реализовать режим автоматического перехода между линейной и логарифмической интерполяцией в зависимости от диапазона ADC?**

    Анализировать коэффициент вариации или диапазон изменения сигнала ADC. При малых изменениях (низкая освещённость) использовать логарифмическую интерполяцию для точности, при больших — линейную для скорости.