Файл функций системы измерения каталитической активности

Основные функции

1. `measureLightLevel()`

Назначение: Измерение уровня освещенности с усреднением результатов.

```
float measureLightLevel()
```

Описание:

- Выполняет 10 последовательных измерений датчиком BH1750
- Усредняет результаты для повышения точности
- Использует задержку 100 мс между измерениями
- Возвращает среднее значение освещенности в люксах

Параметры: Нет **Возвращает:** `float` - средний уровень освещенности (люкс)

Использует:

- Датчик BH1750 (I2C)
- Функцию `lightMeter.measurementReady()`
- Функцию `lightMeter.readLightLevel()`

---

2. `writeDataToFile()`

Назначение: Запись данных измерений в файл на SD-карте.

```
void writeDataToFile(int cycleNumber, float lightLevel)
```

Описание:

- Открывает файл для записи в режиме добавления
- Записывает данные в формате: номер цикла, время, уровень освещенности
- Закрывает файл после записи
- Устанавливает флаг ошибки при проблемах с SD-картой

Параметры:

- `cycleNumber` (int) - номер текущего цикла измерений
- `lightLevel` (float) - измеренный уровень освещенности

Возвращает: Нет

Использует:

- Библиотеку SdFat
- Переменную `dataFileName` для имени файла
- Устанавливает `isSdError = true` при ошибке

---

3. `sendDataViaSerial()`

Назначение: Отправка данных измерений через последовательный порт.

```
void sendDataViaSerial(float lightLevel)
```

Описание:

- Передает значение освещенности через SoftwareSerial
- Используется для мониторинга в реальном времени
- Формат передачи: числовое значение с переносом строки

Параметры:

- `lightLevel` (float) - уровень освещенности для отправки

Возвращает: Нет

Использует:

- Объект `mySerial` (SoftwareSerial)

---

4. `performInitialMeasurement()`

Назначение: Выполнение начального (нулевого) измерения.

```
void performInitialMeasurement()
```

Описание:

- Устанавливает УФ-диод в состояние ВЫКЛ
- Включает красный лазер для измерения
- Выполняет измерение освещенности
- Выключает красный лазер
- Записывает результат в файл
- Отправляет результат через последовательный порт
- Увеличивает счетчик циклов

Параметры: Нет **Возвращает:** Нет

Вызывает:

- `measureLightLevel()` - для измерения освещенности
- `writeDataToFile()` - для записи в файл
- `sendDataViaSerial()` - для отправки данных
- Увеличивает `cycleCounter++`

Использует пины:

- `UV_DIODE_PIN` (8) - устанавливает в LOW
- `RED_LASER_PIN` (7) - включает и выключает

---

5. `performRegularMeasurement()`

Назначение: Выполнение регулярных измерений по таймеру.

```
void performRegularMeasurement()
```

Описание:

- Проверяет, прошло ли достаточно времени с предыдущего измерения
- Устанавливает УФ-диод в состояние ВЫКЛ
- Включает красный лазер
- Выполняет измерение освещенности
- Выключает красный лазер
- Обновляет время предыдущего измерения
- Записывает и отправляет результат
- Увеличивает счетчик циклов

Параметры: Нет **Возвращает:** Нет

Вызывает:

- `measureLightLevel()` - для измерения освещенности
- `writeDataToFile()` - для записи в файл
- `sendDataViaSerial()` - для отправки данных
- Обновляет `previousTime = millis()`
- Увеличивает `cycleCounter++`

Использует пины:

- `UV_DIODE_PIN` (8) - устанавливает в LOW
- `RED_LASER_PIN` (7) - включает и выключает

---

6. `processIncomingCommands()`

Назначение: Обработка входящих команд через последовательный порт.

```
void processIncomingCommands()
```

Описание:

- Проверяет наличие входящих данных в SoftwareSerial
- Читает данные до разделителя ';'
- Разбирает команду с помощью библиотеки GParser
- Выполняет соответствующую команду
- Отправляет подтверждение или результат

Параметры: Нет **Возвращает:** Нет

Обрабатываемые команды:

- **369** - Установка параметров эксперимента
- **1** - Запуск цикла измерений
- **2** - Остановка измерений
- **3** - Включение красного лазера (тестовый режим)
- **4** - Выключение красного лазера

Использует:

- Библиотеку GParser для разбора команд
- Объект `mySerial` для приема/передачи

---

7. `runMeasurementCycle()`

Назначение: Основной цикл управления измерениями.

```
void runMeasurementCycle()
```

Описание:

- Управляет последовательностью измерений
- Включает УФ-диод после завершения адсорбции
- Выполняет начальное и регулярные измерения
- Останавливает измерения по завершению всех циклов

Параметры: Нет **Возвращает:** Нет

Вызывает:

- `performInitialMeasurement()` - при первом цикле
- `performRegularMeasurement()` - для регулярных измерений
- Устанавливает `isCatalysisActive = false` по завершению

Логика работы:

1. Проверяет активность катализа (`isCatalysisActive`)
2. Если счетчик = 0, выполняет начальное измерение
3. После циклов адсорбции включает УФ-диод
4. Выполняет регулярные измерения по таймеру
5. Останавливается после всех циклов катализа

Функции Arduino Framework

8. `setup()`

Назначение: Инициализация системы при запуске.

```
void setup()
```

Описание:

- Инициализирует последовательные порты
- Настраивает датчик освещенности BH1750
- Настраивает пины управления как выходы
- Инициализирует SD-карту
- Устанавливает начальные состояния устройств

Инициализирует:

- Serial (9600 бод) - для отладки
- mySerial (9600 бод) - для управления
- I2C шину (Wire)
- Датчик BH1750 в режиме высокого разрешения
- Пины 7 и 8 как выходы
- SD-карту на пине 10

---

9. `loop()`

Назначение: Главный цикл программы.

```
void loop()
```

Описание:

- Бесконечный цикл обработки
- Вызывает основные функции системы

Вызывает:

- `processIncomingCommands()` - обработка команд
- `runMeasurementCycle()` - выполнение измерений

Вспомогательные функции (неявные)

Управление устройствами:

- `digitalWrite(RED_LASER_PIN, HIGH/LOW)` - управление лазером
- `digitalWrite(UV_DIODE_PIN, HIGH/LOW)` - управление УФ-диодом
- `lightMeter.begin()` - инициализация датчика
- `sdCard.begin()` - инициализация SD-карты
- `sdCard.open()` - открытие файла

Измерение времени:

- `millis()` - получение времени выполнения
- Задержки: `delay(100)` - между измерениями

Работа с данными:

- `String()` - преобразование чисел в строки
- `Serial.println()` - вывод отладочной информации

Последовательность выполнения

```
setup()
├── Инициализация Serial портов
├── Настройка датчика BH1750
├── Настройка пинов управления
└── Инициализация SD-карты

loop() [повторяется]
├── processIncomingCommands()
│   ├── Чтение команд
│   ├── Разбор параметров
│   └── Выполнение действий
└── runMeasurementCycle()
    ├── Если активен и счетчик=0: performInitialMeasurement()
    │   ├── measureLightLevel()
    │   ├── writeDataToFile()
    │   └── sendDataViaSerial()
    ├── Если время пришло: performRegularMeasurement()
    │   ├── measureLightLevel()
    │   ├── writeDataToFile()
    │   └── sendDataViaSerial()
    └── Проверка завершения циклов
```

Зависимости функций


setup()
loop()
processIncomingCommands()
runMeasurementCycle()
performInitialMeasurement()
measureLightLevel()
BH1750.measurementReady()
BH1750.readLightLevel()
performRegularMeasurement()
writeDataToFile()
sdCard.open()
sdCard.println()
sendDataViaSerial()
mySerial.println()