Содержание

1.Введение

1.1 Задача учебной практики

Задача: Игра на скорость реакции – зажечь случайный светодиод, игрок должен нажать правильную кнопку

1.2 Теоретические сведения

1.2.1 Подключения устройств

1.2.1.1 Компоненты:

1) Arduino (Uno, TYPE-C)

2) Макетная плата (HALF)

3) Сенсорная кнопка TTP223

4) Резисторы постоянные 0,25 Вт 1 кОм 5%

5) Светодиоды LED 3мм

6) Oled дисплей 0.96 I2C 128x64 (белый)

7) Пьезопищатель с выводом

1.2.1.2 Физическое подключение:

1) Подключение Arduino к компьютеру:

Подключаем Arduino с помощью TYPE-C провода к копьютеру в свободный USB порт

2) Подключение светодиодов к макетной плате с Arduino:

3) Подключение кнопки TTP223 к макетной плате с Arduino:

4) Подключение Oled дисплея к макетной плате с Arduino:

5) Подключение пьезопищателя к макетной плате с Arduino:

1.2.2 Настройка рабочей среды

1.2.2.1 Скачиваем и устанавливаем Arduino IDE (см..Источники 1)

1.2.2.2 Скачиваем и устанавливаем драйвера для Arduino - CH340[1] (см. Источники 2)

1.2.2.3.1 В Arduino IDE устанавливаем библиотеки Arduino_Sensorkit(1.4.0), ATMAC_EEPROM(1.0.0), ezBuzzer(1.0.0)

1.2.2.3.2 Подключаем библиотеки в коде: (см. Источники 3 и ПРИЛОЖЕНИЕ 1)

[1] Я могу быть не точен насчёт установки этого ПО, так как в Arduino IDE проект собирался в аудитории без него

2. Основная часть

2.1 Описание работы программы

2.1.1 Функциональное назначение

Программа реализует игру на проверку реакции с использованием Arduino.

Главные функции:

1. Управление пользователями (от 1 до 3) с личной статистикой

2. Система уровней сложности игры (1 – лёгкий уровень, 2 – средний уровень, 3 – трудный уровень)

3. Визуальное отображение раунда игры, выбора пользователя, сложности игры с помощью 5 светодиодов и отображение информации на OLED экране

4. Звуковой сигнал с помощью сенсорной кнопки

5. Сохранение результатов в EEPROM памяти

2.1.2 Архитектура проекта

Класс ReactionGame инкапсулирует всю логику игры

Описание класса ReactionGame: (Описание полей и их назначение см. Приложения 2. Жизненный цикл программы см. ПРИЛОЖЕНИЕ 3)

2.2 Листинг кода с объяснением

1) Конструктор для класса ReactionGame: (Схему алгоритма см. ПРИЛОЖЕНИЕ 4)

```
ReactionGame::ReactionGame(int button, const int leds[], int buzzerPin) :
  buttonPin(button), ledPins(leds), piezo(buzzerPin) {
    for(int i = 0; i < 3; i++) {
      for(int j = 0; j < 3; j++) {
        topTimes[i][j] = 0;
```

```
      }
    }
}
```

Объяснение: Передаём в конструктор 3 аргумента: Значение пина кнопки, значения пинов LED светодиодов, значение пина пьезопищателя. С помощью вложенного цикла обнуляем статистику.

2) Функция вывода сообщения (Схему алгоритма см. ПРИЛОЖЕНИЕ 5)

```
void displayMessage(const String& msg) {
  display.clearDisplay();
  display.setCursor(0,0);
  display.println(msg);
  display.display();
  Serial.println(msg);
}
```

Объяснение: это функция типа `void`, принимающая `const String` по ссылке. Функция очищает экран, устанавливает курсор на 0, 0, выводит сообщение, обновляет дисплей и выводит сообщение

3) Функция выбора с заданным диапазоном (Схему алгоритма см. ПРИЛОЖЕНИЕ 6)

```
int ReactionGame::any_choice_with_button(int* options,
int optionsCount) {
    int currentOption = 0;
    unsigned long lastChangeTime = 0;
    const int cycleDelay = 500;

    while(true) {
      unsigned long currentTime = millis();
      if(currentTime - lastChangeTime > cycleDelay) {
        for(int i = 0; i < optionsCount; i++) {
          digitalWrite(ledPins[options[i]], i ==
currentOption ? HIGH : LOW);
        }
        lastChangeTime = currentTime;
        currentOption = (currentOption + 1) %
optionsCount;
```

```
      }
      if(digitalRead(buttonPin)) {
        delay(50);
        while(digitalRead(buttonPin));
        return options[currentOption];
      }
    }
}
```

Объяснение: Это функция, которая принимает указатель на начало массива, значение диапазона, до которого функция будет выбирать значения, возвращающая int. Объявляется переменная для результата, инициализированная 0, переменная типа беззнаковый long для последнего времени изменения, и переменная типа const int со значением 500, обозначающая задержу для цикла. Создаётся цикл, в котором объявляется переменная типа unsigned long, с обозначением правильного времени, инициализированная результатом функции millis(). Далее идёт проверка, если из правильного времени вычесть последнее время изменения, и результат будет больше времени задержки для цикла, то – создаётся цикл с перебором до значения диапазона, в котором в функции digitalWrite подаёт из массива светодиодов, со значениями переданного массива, сигнал HIGH или LOW сигнал на цифровой вывод. Выйдя из цикла, программа обновит значение последнего времени изменения на правильное время и обновит значение правильного диапазона на + 1 к переменной и совершит деление по остатку от диапазона. Далее, выйдя из первой проверки, функция переходит в другую, где digitalRead читает значение с пина кнопки, делает задержку в размере 50 миллисекунд для избежания кнопки от возможного дребезжания, и пока функция читает кнопку – программа вернёт выбор пользователя.

4) Функция мигания всех светодиодов (Схему алгоритма см. ПРИЛОЖЕНИЕ 7)

```
void ReactionGame::light_up_LEDS(int times) {
  for(int i = 0; i < times; i++) {
    for(int j = 0; j < 5; j++) {
      digitalWrite(ledPins[j], HIGH);
```

```
      }
      delay(200);
      for(int j = 0; j < 5; j++) {
        digitalWrite(ledPins[j], LOW);
      }
      delay(200);
    }
  }
```

Объяснение: Это функция, принимающая значение int, обозначающее время. Функция создаёт вложенный цикл, первый идёт перебором до принимаемого значения, второй идёт до 5, подаёт HIGH сигнал на каждый светодиод, зажигая его, Далее выходит из второго цикла, задерживает выполнение на 200 миллисекунд, создаёт третий цикл, в котором выключает все светодиоды, передавая на каждый сигнал LOW, выходит из цикла, делает задержку, и выполняет эти два цикла до тех пор, пока отрабатывает первый цикл.

5) Функция сохранения рекордов (Схему алгоритма см. ПРИЛОЖЕНИЕ 8)

```
void ReactionGame::saveTopTimes() {
  int address = currentUser * 3 * sizeof(int);
  for(int i = 0; i < 3; i++) {
    EEPROM.put(address + i * sizeof(int),
topTimes[currentUser][i]);
  }
}
```

Объяснение: Это функция типа void, не принимающая аргументов, которая ничего не возвращает.В функции создаётся переменная типа int, которая обозначает адрес, инициализирует её так: берёт значение текущего пользователя, которое умножается на 3 и на размер типа int. Далее создаётся цикл, который отрабатывает 3 раза, в нём отрабатывается функция, первый аргумент – адрес + значение итератора (1, 2, 3) и результат умножается на размер типа int, вторым аргументом передаётся значение массива TopTimes, в котором значение текущего пользователя становится равным текущему итератору цикла. То есть, в теле цикла вычисляется адрес и кладётся в нужную ячейку памяти значение рекорда

6) Функция получения рекордов (Схему алгоритма см. ПРИЛОЖЕНИЕ 9)

```
void ReactionGame::loadTopTimes() {
  int address = currentUser * 3 * sizeof(int);
  for(int i = 0; i < 3; i++) {
    EEPROM.get(address + i * sizeof(int),
topTimes[currentUser][i]);
  }
}
```

Объяснение: Это функция типа void, не принимающая аргументов, которая ничего не возвращает. Функция совершает аналогичные действия по сравнению с функцией `loadTopTimes`, только в цикле, функция получает данные из нужных ячеек

7) Функция получения случайного значения (Схему алгоритма см. ПРИЛОЖЕНИЕ 10)

```
int ReactionGame::get_random_LED() {
  return random(0, 5);
}
```

Объяснение: Это функция, которая не принимает аргументов и возвращает int. С помощью функции random, функция возвращает случайное значение.

8) Функция проверки нажатия кнопки (Схему алгоритма см. ПРИЛОЖЕНИЕ 11)

```
bool ReactionGame::check_button_press() {
  unsigned long start_Time = millis();
  while(millis() - start_Time < delayMs[level - 1]) {
    if(digitalRead(buttonPin)) {
      reactionTime = millis() - start_Time;
      return true;
    }
  }
  return false;
}
```

Объяснение: Это функция, которая не принимает аргументов и возвращает bool. Функция создаёт переменную типа unsigned long, которая обозначает время старта, которая равна результата функции millis(). Далее

создаётся цикл, условие которого – пока результат millis() – время старта меньше массива delayMs, которое считает от level – 1, то идёт проверка, если идёт чтение пина кнопки, то обновляем значение переменной времени реакции. И возвращается true. Если цикл не отрабатывает – возвращается false

9) Функция запуска игры (Схему алгоритма см. ПРИЛОЖЕНИЕ 12)

```
void ReactionGame::launch_game() {
  pinMode(buttonPin, INPUT_PULLUP);
  for(int i = 0; i < 5; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
  selectUser();
  loadTopTimes();
}
```

Объяснение: Это функция, не принимающая аргументов, которая ничего не возвращает

10) Функция выбора пользователя (Схему алгоритма см. ПРИЛОЖЕНИЕ 13)

```
void ReactionGame::selectUser() {
  int users[] = {0, 1, 2};
  currentUser = any_choice_with_button(users, 3);
  String msg = "User: " + String(currentUser + 1);
  displayMessage(msg);
}
```

Объясение: Это функция, которая не принимает аргументов, возвращающая int. В теле функции: Создаётся массив из 3 элементов, инициализируется 0, 1 и 2 значениями. Обновляется переменная текущего пользователя, пользователь должен выбрать значение. Создаётся переменная, которая инициализируется сообщением текущего пользователя. Далее выводится сообщение

11) Функция выбора уровня (Схему алгоритма см. ПРИЛОЖЕНИЕ 14)

```
void ReactionGame::setLevel() {
  int levels[] = {0, 1, 2};
  level = any_choice_with_button(levels, 3) + 1;
  String msg = "Level: " + String(level);
  displayMessage(msg);
}
```

Объяснение: функция совершает аналогичные операции, по сравнению с функцией selectUser, но выводится сообщение о выбранном уровне сложности

12) Функция старта раунда игры (Схему алгоритма см. ПРИЛОЖЕНИЕ 15)

```
void ReactionGame::start_round() {
  int ledInx = get_random_LED();
  digitalWrite(ledPins[ledInx], HIGH);
  unsigned long startTime = millis();

  if(check_button_press()) {
    digitalWrite(ledPins[ledInx], LOW);
    piezo.beep(100);

        if(result_counts < 3 || reactionTime < topTimes[currentUser][2]) {
      if(result_counts < 3) {
                topTimes[currentUser][result_counts] = reactionTime;
        result_counts++;
      } else {
        topTimes[currentUser][2] = reactionTime;
      }

      for(int i = 1; i < result_counts; i++) {
                    if(topTimes[currentUser][i] < topTimes[currentUser][i - 1]) {
          int temp = topTimes[currentUser][i - 1];
                topTimes[currentUser][i - 1] = topTimes[currentUser][i];
          topTimes[currentUser][i] = temp;
        }
      }
      saveTopTimes();
    }

    String msg = "Time: " + String(reactionTime) + " ms";
    displayMessage(msg);
  } else {
    digitalWrite(ledPins[ledInx], LOW);
    light_up_LEDS(3);
```

```
        displayMessage("Too slow!");
      }
    }
```

Объяснение: Это функция, не принимающая аргументов. В функции происходит: выбор случайного светодиода, проверка реакции игрока и вывод звука, обновление статистики, сортировка результатов методом пузырька (пузырьковая сортировка), сохранение результата в EEPROM, вывод времени реакции, если игрок не успел, то выключаются светодиоды и мигают 3 раза

13) Функция показа рекордов (Схему алгоритма см. ПРИЛОЖЕНИЕ 16)

```
void ReactionGame::showTopTimes() {
  String msg = "Top for User " + String(currentUser + 1) + ":\n";
  for(int i = 0; i < 3; i++) {
    if(topTimes[currentUser][i] > 0) {
      msg += String(i + 1) + ": " + String(topTimes[currentUser][i]) + " ms\n";
   }
  }
  displayMessage(msg);
}
```

Объяснение: Это функция, которая не принимает аргументов и ничего не возвращает. В теле функции: создаётся переменная типа String, которая обозначат сообщение с рекордами, и выводится с помощьюю цикла

Заключение

Я изучил тему программирования микроконтроллеров Arduino в ходе работы над задачей и получил практический опыт, который я могу применить в дальнейшем, если у меня будет задача, связанная с темой микроконтроллеров. Я научился находить информацию на интернет-ресурсах, смотря и проверяя код разных людей. Я научился работать в новом для меня среде разработки, при поиске нужных мне библиотек и сборке проекта. Мне было полезно узнать, как программировать физический объект, и результатом моих действий станет то, что я написал и подключил. Мне интересен этот проект, поэтому я старался решать проблемы в срок и выполнять задачи, а так же, я добавил от себя систему пользователей. К сожалению, у меня не получилось завершить работу проекта из за недостатка проводов, и у меня не хватило бы деталей, даже при альтернативном подключении всех деталей. В дальнейшем, я планирую завершить этот проект, потому что это важно лично для меня и для моей учёбы

4.Список использованных источников

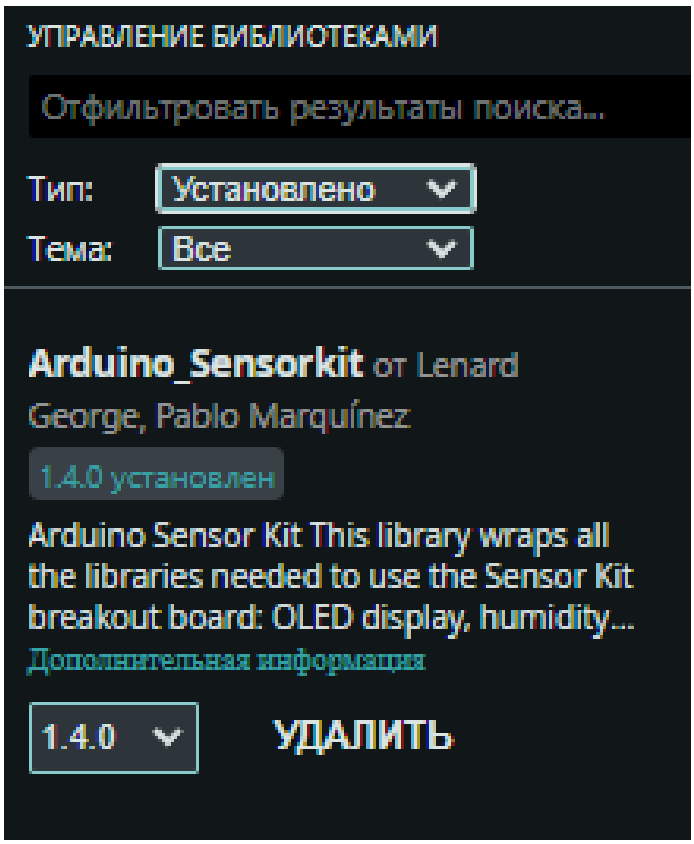
Источники 1 https://www.arduino.cc/en/software/
Источники 2 https://wiki.amperka.ru/articles:driver-ch340
Источники 3 https://iarduino.ru/file/
Источники 4 https://arduino.ru/Reference
Источники 5 https://lesson.iarduino.ru/
Источники 5 https://alexgyver.ru/lessons/library-using/

5.Приложения

ПРИЛОЖЕНИЕ 1

```
#include "Arduino.h"
#include "ezBuzzer.h"
#include "EEPROM.h"
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_GFX.h>
```

ПРИЛОЖЕНИЕ 2



ПРИЛОЖЕНИЕ 3

Поле	Тип данных	Назначение
buttonPin	const int	Пин для подключения кнопки (TTP223)
ledPins	const int* (указатель на начало массива)	Массив пинов светодиодов (5 шт.)
topTimes	int[3][3]	Двумерный массив («матрица») лучших результата
piezo	ezBuzzer	Объект управления пьезопищалкой
reactionTime	int	Время реакции пользователя
level	int	Уровень игры
delayMs	int[3]	Время задержки реакции
result_counts	int	Подсчёт результата
currentUser	int	Текущий пользователь

Методы класса	Тип данных	Аргумент(ы)	Способ инкапсуляции	Назначение
light_up_LEDS	void	int	private	Зажигание всех светодиодов
saveTopTimes	void	-	private	Сохранение рекордов
loadTopTimes	void	-	private	Получение рекордов

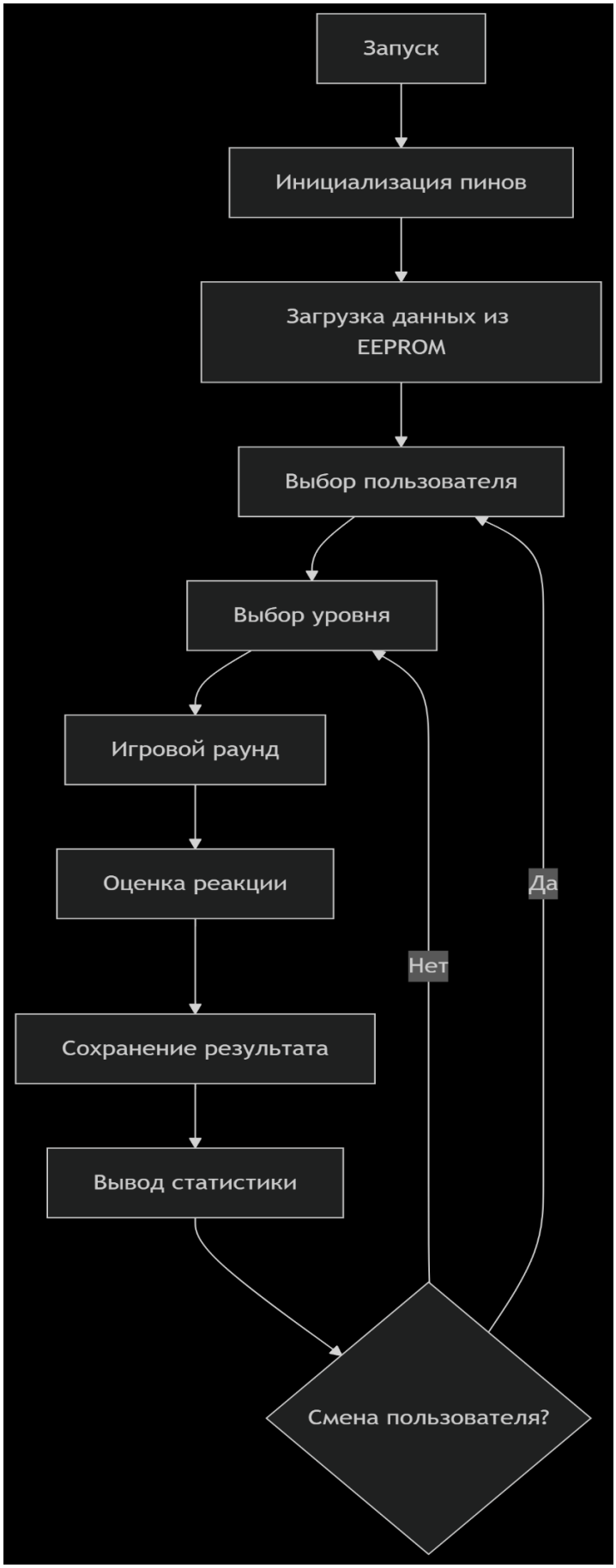
get_random_LED	int	-	private	Получение случайного значения для светодиода
check_button_press	bool	-	private	Проверка на нажатие кнопки пользователем
displayMessage	void	const String&	public	Вывод сообщения на экран
launch_game	void	-	public	Загрузка игры
selectUser	void	-	public	Выбор пользователя
setLevel	void	-	public	Установка уровня пользователем
start_round	void	-	public	Старт раунда
showTopTimes	void	-	public	Вывод рекордов

Конструктор класса ReactionGame	Аргумент(ы)	Способ инкапсуляции	Назначение
ReactionGame()	int, const int[], int	public	Создание объекта класса

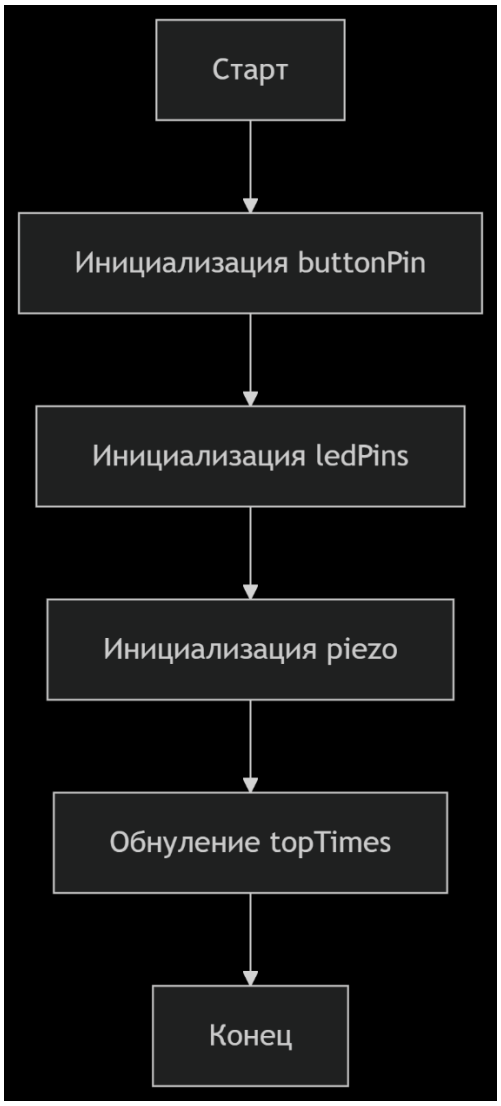
Деструктор класса ReactionGame	Аргумент(ы)	Способ инкапсуляции	Назначение
~ReactionGame()	-	public	Уничтожение объекта класса

Глобальная переменная	Тип	Аргумент(ы)	Назначение
display	Adafruit_SSD1306	SCREEN_WIDTH, SCREEN_HEIGHT, &Wire	Создание объекта класса дисплея

ПРИЛОЖЕНИЕ 4

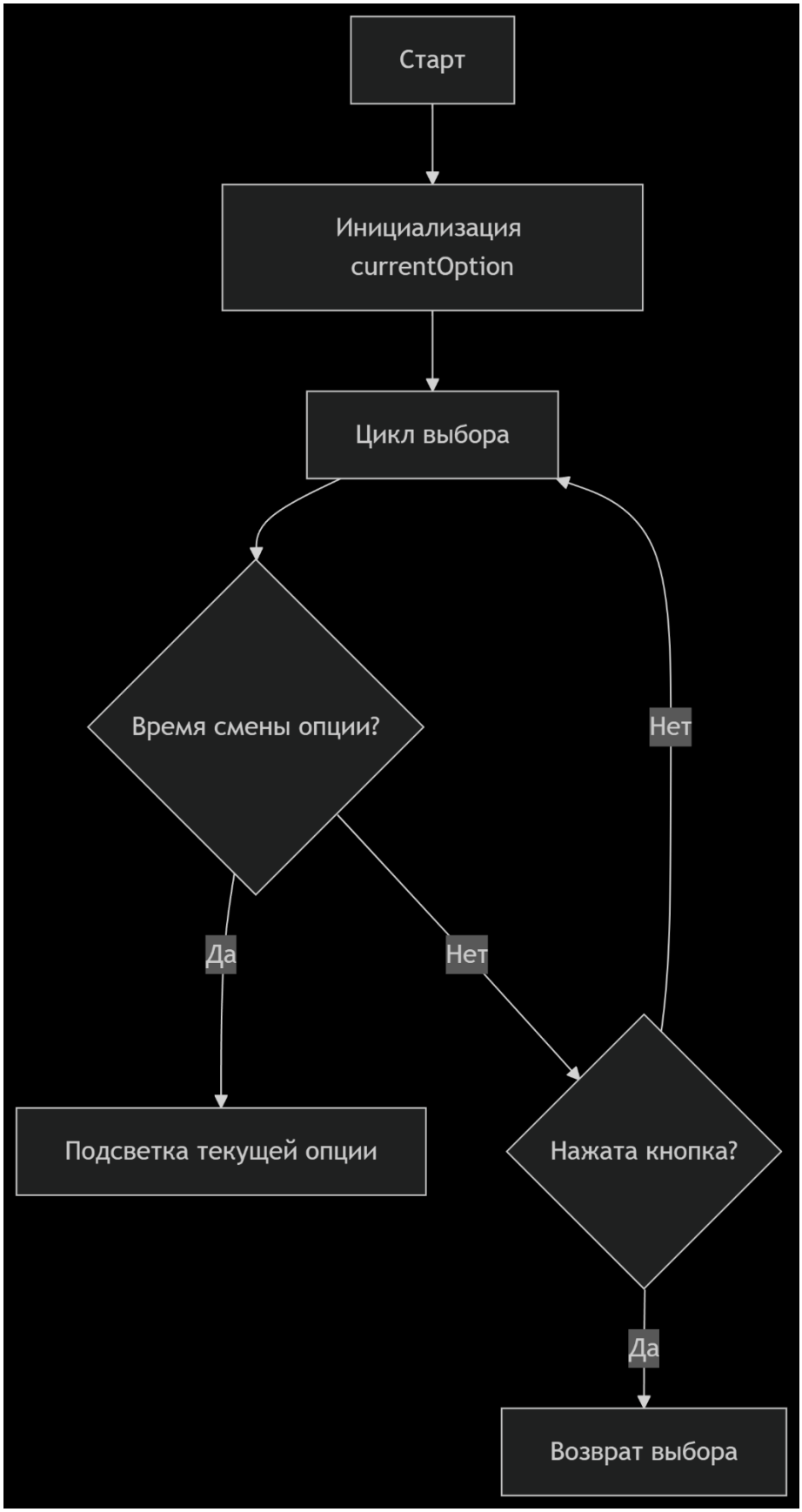


ПРИЛОЖЕНИЕ 5

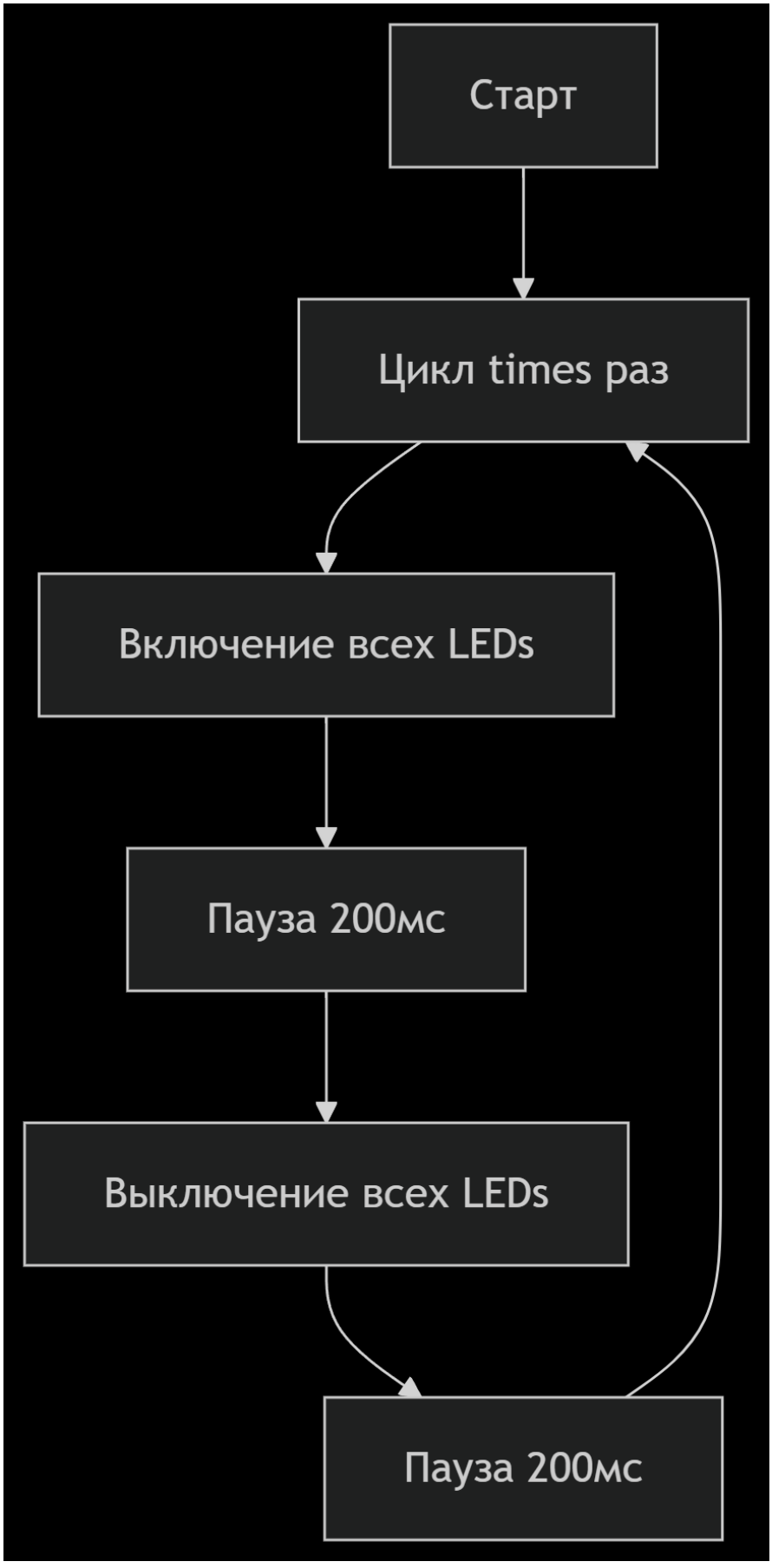


ПРИЛОЖЕНИЕ 6

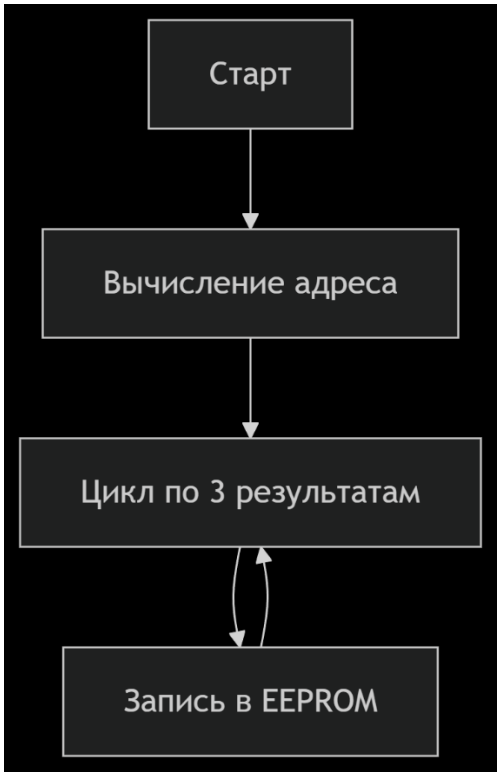
ПРИЛОЖЕНИЕ 6



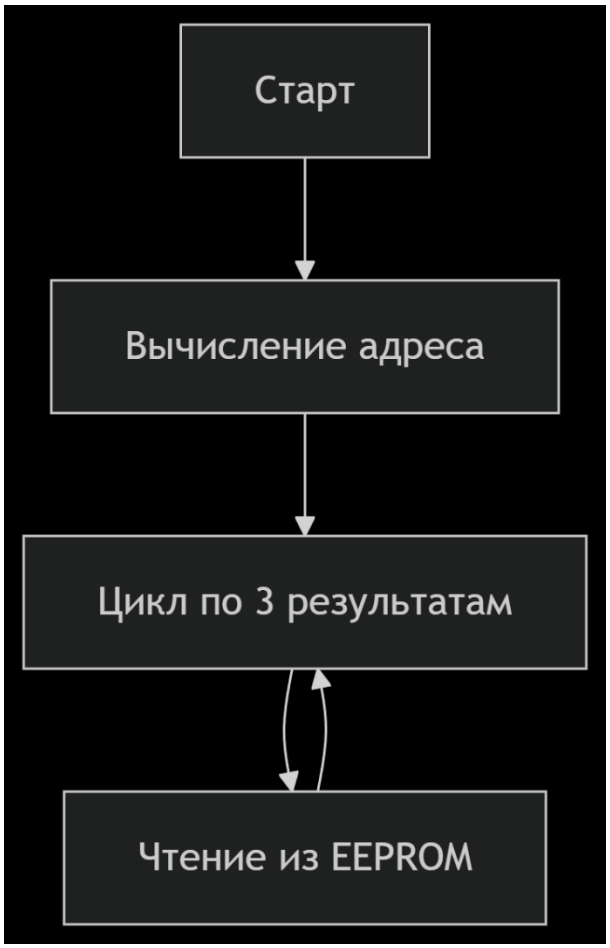
ПРИЛОЖЕНИЕ 7



ПРИЛОЖЕНИЕ 8

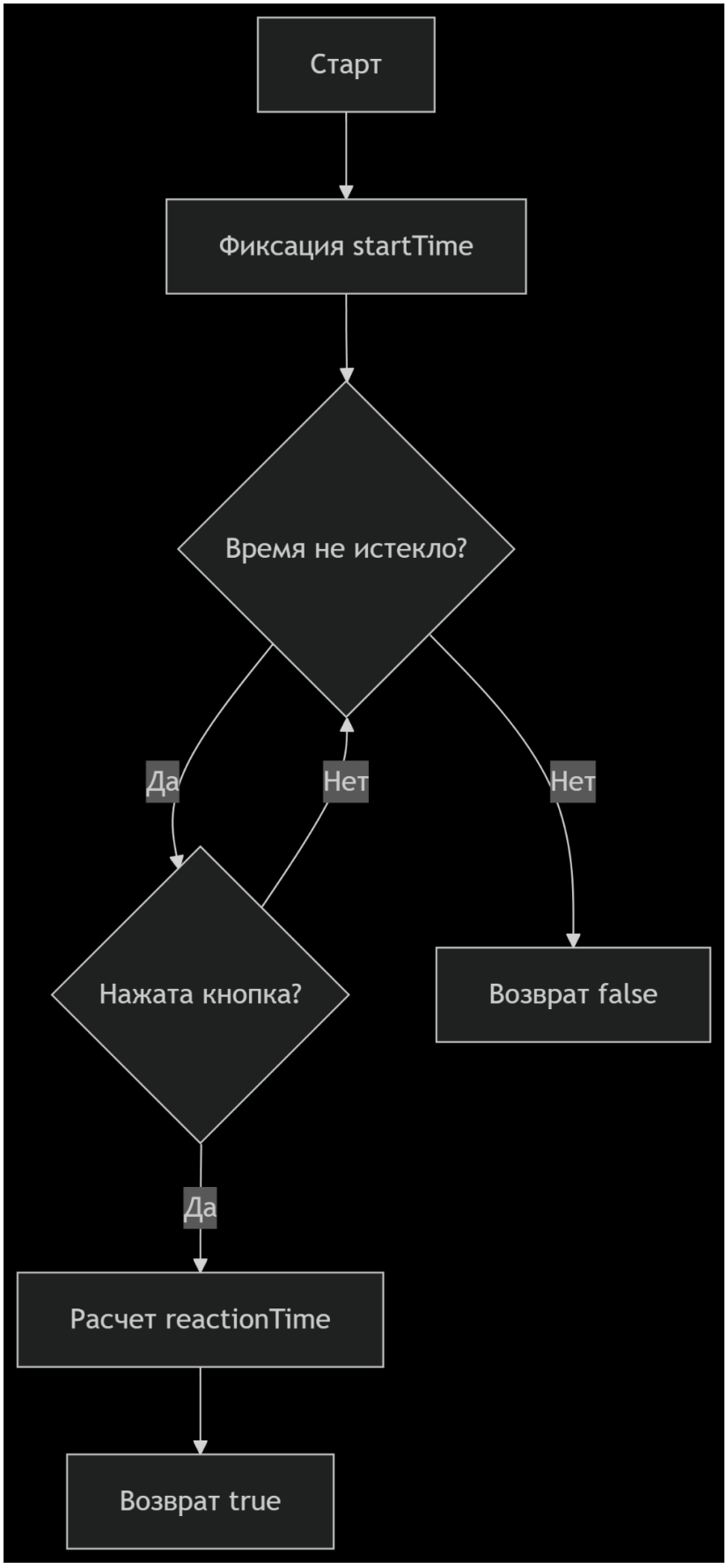


ПРИЛОЖЕНИЕ 9

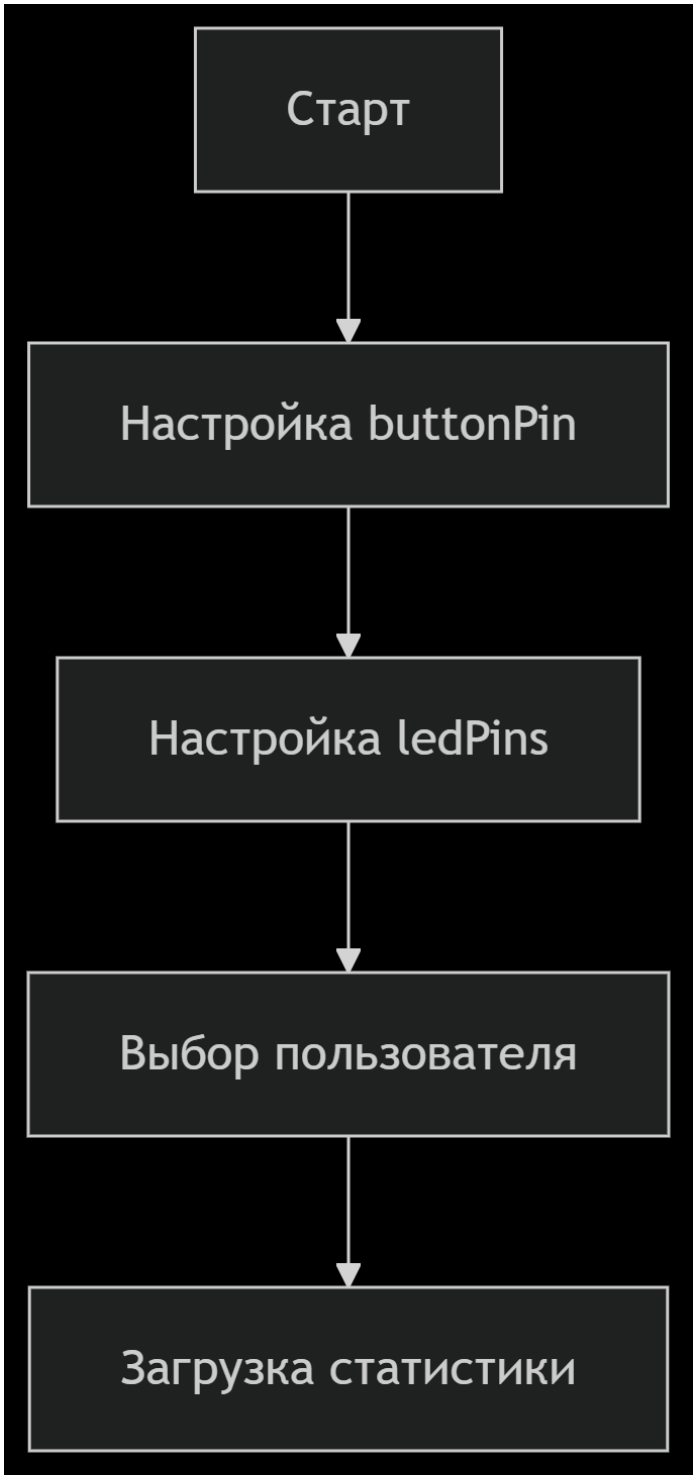


ПРИЛОЖЕНИЕ 10

ПРИЛОЖЕНИЕ 11



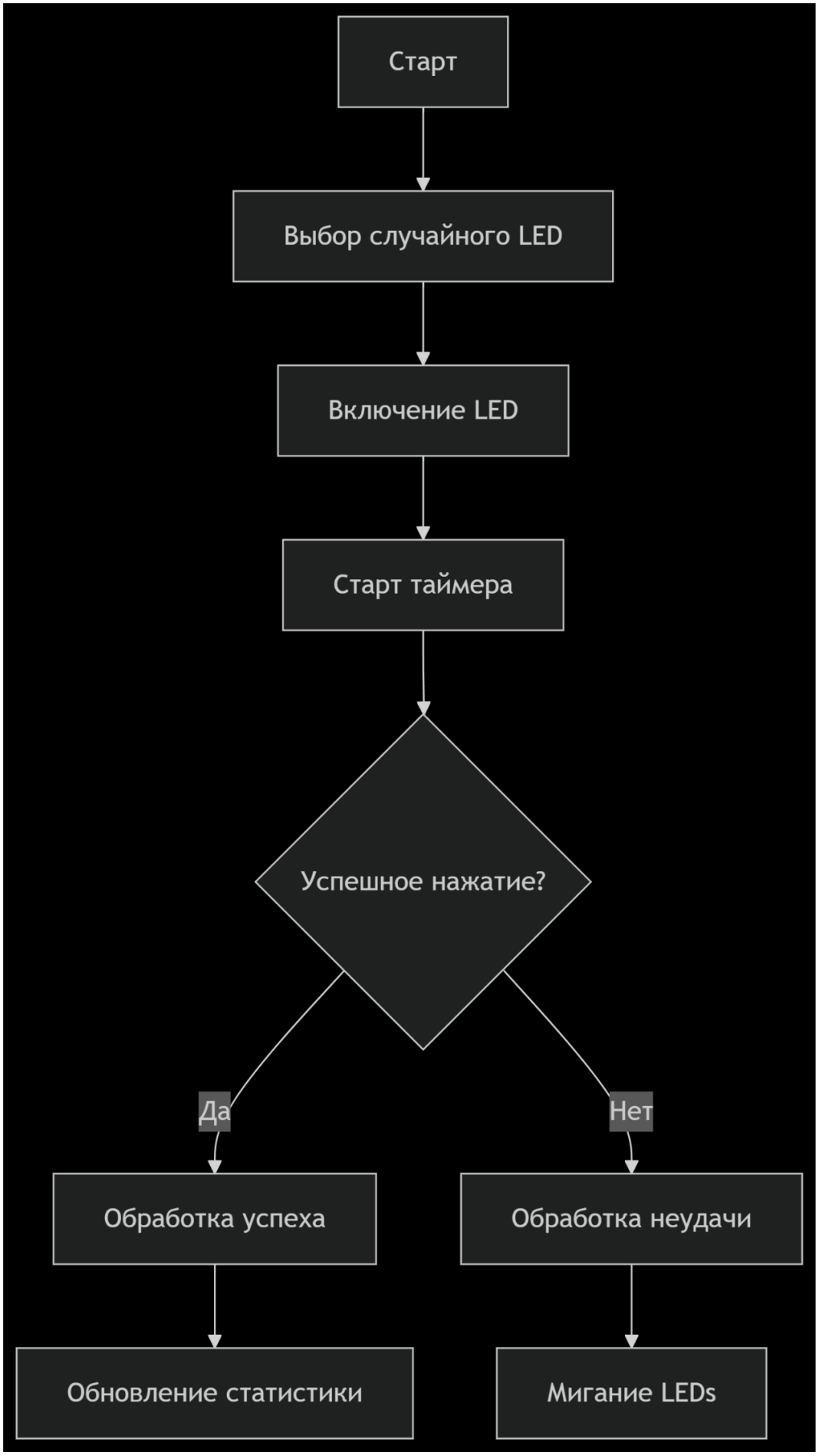
ПРИЛОЖЕНИЕ 12



ПРИЛОЖЕНИЕ 13

ПРИЛОЖЕНИЕ 13

ПРИЛОЖЕНИЕ 14



ПРИЛОЖЕНИЕ 15