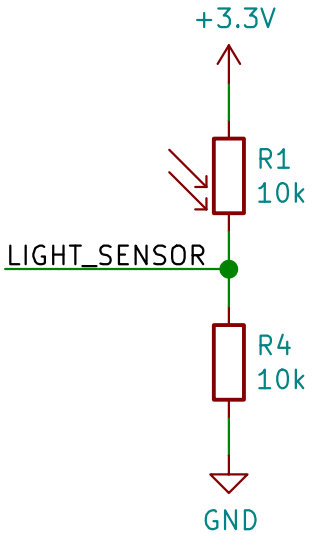
Датчик освещённости

Вы, наверное замечали, что у большинства смартфонов экран меняет свою яркость в зависимости от освещения? Магии здесь нет, всё дело в датчике освещенности. Чем наши часы хуже? Ночью яркость дисплея следует уменьшить, иначе он будет освещать всю комнату и мешать спать. К счастью в наборе имеется элемент, который можно использовать в качестве датчика освещённости, а именно — фоторезистор.

> Фоторезистор — *полупроводниковый прибор, изменяющий величину своего сопротивления при облучении светом. Не имеет p-n перехода, поэтому обладает одинаковой проводимостью независимо от направления протекания тока.* // Wikipadia

К сожалению, говорить о стабильности зависимости сопротивления от падающего света от изделия к изделию не приходится. Снимать абсолютные значения им не получится, но он хорошо подойдёт на роль датчика освещённости. Однако, его перед использованием придётся откалибровать, т.е. замерить сопротивление в темноте и под действием света.

У вас, уже появился вопрос: а как собственно можно измерить сопротивление? Само это значение нам в принципе ни к чему, а вот напряжение мы измерить можем достаточно просто, ведь в состав микроконтроллера входит такая периферия как АЦП — аналого-цифровой преобразователь, суть работы которого мы уже рассмотрели. Добавив в схему ещё одно сопротивление можно получить делитель напряжения.



В качестве R_1 установлен фоторезистор, сопротивление которого меняется от 10-20 кОм при свете до 1 МОм в темноте. Подключая фоторезистор к питанию (+3,3В), а R_2 (возьмём равным 100 кОм) к земле, можно прикинуть диапазон напряжений, которое будет наблюдаться между R_1 и R_2 из соотношения:

$$U_2 = \frac{R_2}{R_1 + R_2}$$

Так для R_1 = 10 кОм, напряжение V_{adc} = 3 В, для 100 кОм — 1,65 В, а при 1 МОм — 0,3 В, т.е. в теории должен перекрывать почти весь динамический диапазон АЦП.

С физической частью мы разобрались, приступим к программированию. Как и раньше обособим код относящийся к датчику в отдельный модуль (файлы `light_sensor.c` и `light_sensor.h`). Однако перед этим обговорим, что именно мы хотим получить.

Нам нужны две функции, одна инициализирует периферию, а вторая запускает преобразование и возвращает уровень яркости дисплея. Допустим для дневного режима яркость будет составлять 12, а для ночного 2. Разница между условным днём и условной ночью будет решаться простым неравенством — если значение АЦП больше заданного порога, значит это день, в противном случае это ночь. Усложняя и развивая данную функцию можно добавить гистерезис, а также плавный переход с одного уровня яркости к другому. Что же касается АЦП, то мы его настроем в независимый режим работы с одиночным измерением. Это не самое лучшее решение, но мы его выбрали из-за простоты.

Составим заголовочный файл.

```
#ifndef __LIGHT_SENSOR_H__
#define __LIGHT_SENSOR_H__

#include "stm32f10x.h"

void light_sensor_init(void);
uint8_t light_sensor_get_brightness(void);

#endif /* LIGHT_SENSOR_H */
```

В файле исходного кода нам необходимо задать константы ночной и дневной яркости, а так же пороговое значение.

```
// light_sensor.h
#define NIGHT_DISPLAY_BRIGHTNESS        2
#define DAY_DISPLAY_BRIGHTNESS          12
#define LIGHT_SENSOR_THRESHOL           2047
```

Согласно схеме, датчик подключен к нулевой ножке порта А, то есть к модулю `ADC1`, `0` каналу.

```
void light_sensor_init(void) {
    // Включаем тактирование порта A, альтернативной функции и ADC1
    RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;
    RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;

    // Вход, аналоговый
    GPIOA->CRL &= ~GPIO_CRL_MODE0;
    GPIOA->CRL &= ~GPIO_CRL_CNF0;

    // Независимый режим работы
    ADC1->CR1 &= ~ADC_CR1_DUALMOD;
    // Устанавливаем количество каналов (1)
    ADC1->CR1 &= ~ADC_CR1_DUALMOD;
    // Однократный режим
    ADC1->CR1 &= ~ADC_CR1_DISCEN;
    // Отключаем многоканальный режим
    ADC1->CR1 &= ~ADC_CR1_SCAN;
    // Однократный режим
```

```
    ADC1->CR2 &= ~ADC_CR2_CONT;
    // Выравнивание к правой стороне
    ADC1->CR2 &= ~ADC_CR2_ALIGN;
    // Отключаем преобразование по триггеру
    ADC1->CR2 &= ~ADC_CR2_EXTTRIG;
    // Регулярные каналы, последовательность из 1 канала
    ADC1->SQR1 &= ~ADC_SQR1_L;
    // Настройка регулярного канала
    ADC1->SMPR2 &= ~ADC_SMPR2_SMP0;
    ADC1->SMPR2 |= ADC_SMPR2_SMP0_0;
    ADC1->SQR3 &= ~ADC_SQR3_SQ1_0; // one channel
    // Включаем АЦП
    ADC1->CR2 |= ADC_CR2_ADON;

    // Сбрасываем калибровку
    ADC1->CR2 |= ADC_CR2_RSTCAL;
    while((ADC1->CR2 & ADC_CR2_RSTCAL) == ADC_CR2_RSTCAL);

    // Запускаем калибровку
    ADC1->CR2 |= ADC_CR2_CAL;
    while((ADC1->CR2 & ADC_CR2_CAL) != ADC_CR2_CAL);
}
```

Осталось реализовать функцию получения яркости дисплея.

```
uint8_t light_sensor_get_brightness(void) {
    // Запуск преобразования и ожидание его завершения
    ADC1->CR2 |= ADC_CR2_ADON;
    while((ADC1->CR2 & ADC_SR_EOC) == ADC_SR_EOC);

    if ((uint16_t)ADC1->DR < LIGHT_SENSOR_THRESHOL) {
        return NIGHT_DISPLAY_BRIGHTNESS;
    }
    return DAY_DISPLAY_BRIGHTNESS;
}
```

Код урока можно найти на github: **CMSIS**.

Назад | **Оглавление** | **Дальше**