

# Модуль Smartli

---

Объект модуля протеза, который даёт доступ к системным параметрам самого протеза:

- [Напряжение модуля питания протеза](#)
- [Состояние кнопки управления](#)
- [Управление вибромотором](#)
- [Примеры работы с модулем](#)
  - [Уведомление о севшем модуле питания](#)
  - [Работа с кнопкой](#)

## Синтаксис

```
Smartli.Vibrate()  
Smartli.GetPowerRaw()  
Smartli.GetPowerVolts()  
Smartli.SwitchPressedTime()  
Smartli.SwitchState()  
Smartli.Speaker()
```

## Состояние модуля питания

---

Уровень заряда батареи, элемента питания, аккумулятора определяется по уровню остаточного напряжения на контактах аккумулятора. Модуль питания протеза строится на основе двух элементов питания, суммарным напряжением 8,4 В при полном заряде, 7,6 В при минимально допустимом заряде. Блок управления протезом Smartli Education оснащён микроконтроллером со встроенным АЦП с разрешением оцифровки 10 бит, что обеспечивает диапазон значений: 0 .. 1023. Для нашего модуля питания имеем диапазон сырых значений: Максимальный уровень напряжения, которое может оцифровать АЦП микроконтроллера - 5В. Для работы с напряжением выше, на плате реализовано схемотехническое решение, позволяющее снижать диапазон входного напряжений в два раза.

Таким образом получаем диапазон измеряемых напряжений: 3,8В .. 4,2В. В отсчетах АЦП: 777 .. 859

Стоит отметить, что указанный выше диапазон носит исключительно ознакомительный характер, и ваши значения могут несколько отличаться

## int GetPowerRaw()

---

Метод позволяет получить значение уровня заряда модуля питания протеза в условных единицах: 0 .. 1023. Стоит понимать, что значения 0 фактически получить не получится - при таких напряжениях микроконтроллер не будет работать.

## Пример

```
int rawPower = Smartli.GetPowerRaw(); // читаем состояние заряда модуля питания протеза
```

## int GetPowerVolts()

Метод позволяет получить значение уровня заряда модуля питания протеза в вольтах: 0 .. 10V. Шаг дискрета составляет: 0.1V Важно: метод возвращает значение с плавающей точкой (дробное). При использовании целочисленного типа данных (int, char и пр) дробная часть будет отброшена и проигнорирована. 3,9V => 3V - примерно так будет выглядеть действие присвоения

Используйте типы данных double | float для переменных, в которых будете хранить значение заряда модуля питания протеза.

## Пример

```
float voltsPower = Smartli.GetPowerVolts();
```

## Кнопка управления протезом

Большинство электронных устройств оснащаются тем или иным видом контактного управления. Традиционным элементом контактного управления является кнопка. Традиционно кнопка - это два замыкаемых электрических контакта. При замыкании электрических контактов возникает так называемых **дребезг**. Дребезг контактов - это когда значение, считываемое микроконтроллером с порта подключения кнопки меняет свое значение с 0 на 1 и обратно за время соизмеримое со временем чтения состояния кнопки. Это приводит к тому, что однократное нажатие кнопки, может восприниматься микроконтроллером как многократное. Чтобы этого не происходило применяют программную фильтрацию дребезга. Работает это примерно так:

1. Читают состояние кнопки
2. Ждут XX времени
3. Читаю состояние кнопки
4. Если состояние не изменилось => считаем, что кнопка в это состоянии, иначе что-то не так.

## int SwitchState(int debounce = 2)

Метод чтения состояния кнопки, где: debounce - задержка для фильтрации дребезга. Указывается в миллисекундах. По умолчанию равняется 2мс.

Метод возвращает одно из трёх значений: 0 - кнопка не нажата 1 - кнопка нажата 2 - в процессе проверки состояния кнопки, состояние кнопки изменилось. В случае, когда состояние 2 возникает часто, рекомендуется увеличить значение **debounce**

#### Пример

```
int buttonState = Smartli.SwitchState(); // читаем состояние кнопки
```

---

### int SwitchPressedTime(int debounce = 2)

Метод позволяет определить время, в течении которого кнопка остаётся нажатой. Фактически debounce вносит погрешность в измерения:

1. Кнопка должна быть нажата дольше двойного значения debounce
2. Итоговое возвращаемое значение длительности нажатия всегда меньше фактически измеренного на  $\text{debounce} * 2$  Таким образом получаем, что средняя вносимая в измерения длительности погрешность равняется длительности задержки.

#### Пример

```
int buttonPressedTime = Smartli.SwitchPressedTime(); // читаем длительность нажатия
```

---

### Индикация и обратная связь

Для индикации состояния или режима внутри модуля управления протезом Smartli Education, возможно применять штатный вибромотор. Вибромотор может находиться в одном из двух состояний:

1. Покой - мотор выключен и не вибрирует
2. Активное - на мотор подаётся напряжение питания, мотор совершает вращательные движения и вибрирует.

---

### void Vibrate(int period, int count)

Метод позволяет управлять вибромотором протеза.

## Параметры

**int period** - значение периода вибрации, задаётся в миллисекундах, фактическая длительность вибрации равна 1/2 от заданного значения периода.

**int count** - количество импульсов вибрации, задаётся в единицах.

## Пример

```
Smartli.Vibrate(500, 3);
```

где: 500 - это период импульса вибрации 3 - это количество импульсов вибрации

Это означает, что вибромотор произведёт 3 такта вибрации, каждый такт будет иметь 250 мс активного состояния - мотор вибрирует, и 250 мс пассивного состояния - мотор находится в покое и не вибрирует.

---

## Примеры программ

---

### Уведомление о севшем модуле питания

---

#### float GetPowerVolts()

```
#include <smartlieducation.h>

void setup() {

}

void loop() {
    // Читаем значение напряжения модуля питания
    float voltage = Smartli.GetPowerVolts();

    // проверяем уровень
    if (voltage < 8.0) {
        // если уровень низкий, но не критичный
        // совершаем 3 вибрации по 300 мс каждая
        Smartli.Vibrate(300, 3);
    } else if (voltage < 7.5) {
        // если уровень критический
        // совершаем одну длительную вибрацию
        Smartli.Vibrate(600, 1);
    }
}
```

```
}  
  
    delay(500);  
}
```

## Работа с кнопкой

---

### int SwitchState()

```
#include <smartlieducation.h>  
  
void setup() {  
  
}  
  
void loop() {  
    // читаем состояние кнопки  
    int buttonState = Smartli.SwitchState(100);  
  
    if (buttonState === 1) {  
        // если кнопка нажата  
        // совершаем короткую одинарную вибрацию  
        Smartli.Vibrate(100, 1);  
        delay(100);  
    } else if (buttonState === 2) {  
        // если кнопка была отжата или нажата в процессе проверки состояния кнопки  
        // вибрируем 3 раза  
        Smartli.Vibrate(400, 3);  
        delay(100);  
    }  
}
```

### int SwitchPressedTime()

```
#include <smartlieducation.h>  
  
void setup() {  
  
}  
  
void loop() {  
    // определяем длительность нажатия кнопки  
    int buttonPressedTime = Smartli.SwitchPressedTime(100);  
  
    delay(200);  
}
```

```
// вибрируем с длительностью, равной длительности нажатия кнопки  
Smartli.Vibrate(buttonPressedTime, 1);  
}
```

---