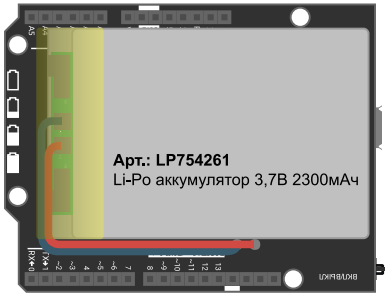
Battery Shield для автономного и резервного питания Arduino



Общие сведения:

Battery Shield - это источник автономного питания для 5В плат Arduino, позволяющий сделать Ваши устройства по настоящему мобильными. Battery Shield устанавливается на Arduino снабжая её питанием, как обычный аккумулятор снабжает питанием Ваш смартфон, планшет и т.д. Если к шинам питания Arduino подключены иные устройства, они также получат питание от Battery Shield. Уровень заряда LiPo (литий-полимерного) аккумулятора можно контролировать либо программно (по шине I2C), либо визуально (по светодиодному индикатору на плате). При необходимости аккумулятор можно зарядить через порт micro USB (питание Arduino не исчезнет во время заряда аккумулятора), блок зарядного устройства автоматически включается, выключается и выбирает тип заряда аккумулятора в зависимости от уровня его разряженности.

Источник автономного питания выполнен в виде Shield, что удобно при его использовании с платами Arduino Uno, Arduino Mega, Arduino Leonardo и им подобных плат Arduino с рабочим напряжением питания 5В. Если Вы используете платы Arduino Nano или Arduino Pro Mini 5V 16MHz, то их так же можно запитать от Battery Shield, без проводов, предварительно установив Arduino в модуль Trema Shield NANO.

Использование Battery Shield не только превратит Ваши устройства в мобильные, но и избавит Вас от необходимости использования силовых проводов, блоков питания, батарейных отсеков с аккумуляторами, преобразователями и т.д.

Спецификация:

- Входное напряжение питания зарядного устройства: 5 В (порт micro USB).
- Выходное напряжение питания модуля: 5 В (постоянного тока).
- Напряжение заряда аккумулятора: 4,2 В.
- Тип аккумулятора: LiPo (литий-полимерный) 3,7 В.
- Ток заряда аккумулятора: до 2,1 А.
- Ток на выходе модуля: до 1 А (в пиках до 1,5 А).
- Время обнаружения перегрузки по току: 30 мс.
- Время обнаружения КЗ нагрузки: до 200 мкс.
- Время пробуждения: 50 мс.

- Порог срабатывания защиты от перегрева: 125 °C.
- Порог срабатывания защиты от пониженного питания на входе micro USB: 4,5 В.
- Частота повышающего DC-DC преобразователя: 650 кГц.
- Интерфейс: I2C.
- Адрес на шине I2C: 0x75.
- Рабочая температура: 0 ... 70 °C.
- Габариты: Shield.

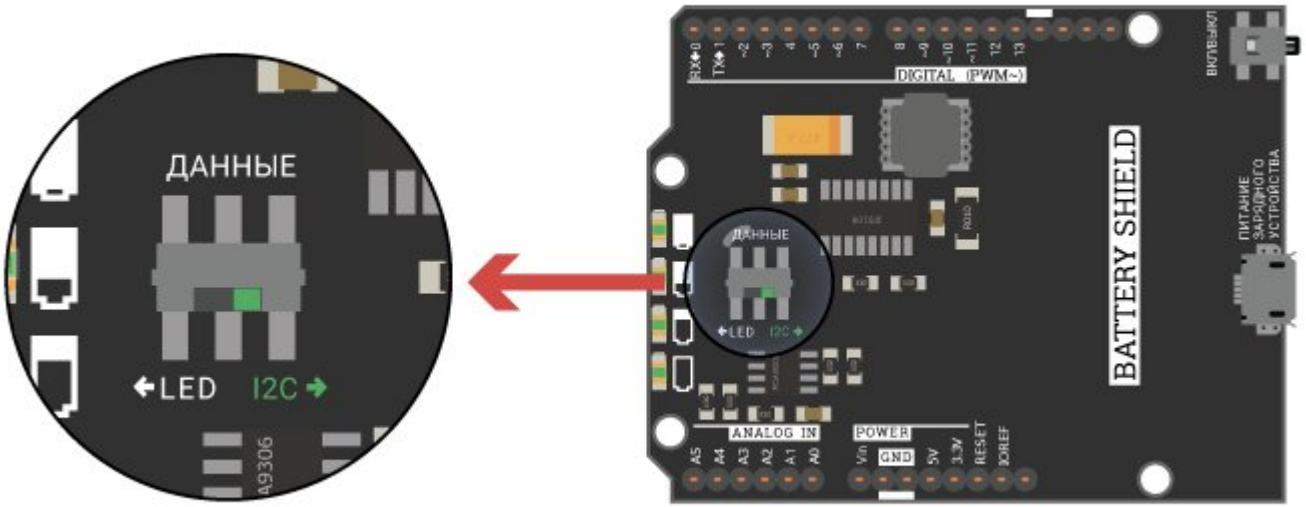
Подключение:

- Перед установкой Battery Shield дважды нажмите кнопку на плате, что бы выключить модуль.
- Battery Shield устанавливается на Arduino или Trema Shield NANO, так чтобы все штекеры разъёмов модуля установились в гнёзда разъемов Arduino.
- Колодка «POWER» на плате модуля должна совпадать с одноимённой колодкой на плате Arduino или Trema Shield NANO.

Отключение спящего режима:

Для предотвращения случайной разрядки батареи в модуле предусмотрена функция автоматического выключения при простое без нагрузки. Функция по умолчанию включена, если микроконтроллер не подключён или подключён, но не инициировал модуль или переключатель модуля находится в положении LED. Для отключения этой функции необходимо:

1) Перевести переключатель в положение I2C:



2) Подключить Shield к Arduino

3) Скачать и установить библиотеку Battery_Shield. О том как устанавливать библиотеки можно узнать здесь.

4) Добавить следующие строки в скетч:

```
#include <Battery_Shield.h>             // Подключаем библиотеку Battery_Shield.
Battery_Shield pwrBank;                 // Объявляем объект pwrBank для работы с функциями и методами библиотеки Battery_Shie
                                        //
void setup(){                           //
    pwrBank.begin(0.0128f);             // Инициируем работу с Battery Shield, указывая номинал сопротивления (0.128 Ом) уста
}                                       //
```

Подробнее о Battery Shield:

Battery Shield построен на базе чипа IP-5108 оснащенным блоком управления заряда/разряда аккумулятора блоком управления повышающего DC-DC преобразователя, многоканальным управлением питанием, 14-ти битным АЦП для чтения напряжений в различных цепях схемы, защитой от перегрузки по току (на входе и выходе), от короткого замыкания, от перенапряжения, от перезарядки аккумулятора, от перегрева чипа. При срабатывании защиты, выходное напряжение отключается, для возобновления работы Battery Shield, необходимо подать питание на порт micro USB. Для согласования логических уровней шины I2C используется чип PCA9306. Контролировать текущее состояние аккумулятора и процесса его заряда, можно как программно (по шине I2C), так и визуально (посредством светодиодов на плате модуля). Установить подходящий Вам метод контроля можно используя переключатель на плате модуля.

Специально для Battery Shield, нами разработана библиотека Battery_Shield, которая позволяет управлять источником автономного питания по шине I2C. Для работы библиотеки, переключатель на плате должен находиться в положении «I2C». Библиотека позволяет: выключать модуль, включать/выключать зарядное устройство, получать силу тока аккумулятора, получать силу тока в цепи нагрузки, получать напряжение на аккумуляторе, получать напряжение на аккумуляторе без нагрузки, получать напряжение в цепи нагрузки, получать % заряда аккумулятора, получать используемый тип заряда аккумулятора (заряд не осуществляется, TK - заряд малым током, CC - заряд постоянным током, CV - заряд постоянным напряжением, Time Over). Дополнительно можно получить текущее КПД повышающего DC-DC преобразователя, а так же точное сопротивление резистора в цепи аккумулятора, используемого для расчёта силы тока аккумулятора.

Для включения модуля необходимо однократно нажать на единственную кнопку на плате.
Выключить модуль можно либо двойным нажатием на ту же кнопку, либо программно (по шине I2C).

- Перед первым включением Battery Shield (после покупки), подайте питание на порт micro USB (не менее чем на 2 секунды).
- Для включения модуля необходимо однократно нажать на единственную кнопку на плате.
- Для выключения модуля необходимо выполнить двойное нажатие на единственную кнопку на плате (выключайте модуль перед его установкой на Arduino).
- Для заряда аккумулятора подайте питание на порт micro USB (при наличии питания от micro USB, модуль включится и не будет выключаться при нажатии на кнопку).
- При срабатывании защиты Battery Shield (перегрузка по току, КЗ, перегрев и т.д), выходное напряжение модуля отключается, для

возобновления работы Battery Shield, необходимо его включить, однократно нажав на кнопку.

- При срабатывании защиты аккумулятора, его схема отключит питание на выходе, для возобновления работы аккумулятора необходимо подать питание на порт micro USB.
- Для визуальной индикации (посредством светодиодов) состояния аккумулятора и его зарядки, переведите выключатель на плате модуля в положение «LED».
- Для управления модулем и получения данных по шине I2C, переведите выключатель на плате модуля в положение «I2C».
- Обратите внимание на то, что в режиме «LED» светодиодная индикация потребляет ток.
- Если модуль находится без нагрузки (ток на выходе ниже 120 мА) дольше 32 секунд, то он автоматически выключится.
- Для работы с Battery Shield по шине I2C предлагаем воспользоваться библиотекой Battery_Shield.
- Библиотека Battery_Shield запрещает автоматическое выключение модуля при отсутствии нагрузки.

Примеры:

Вывод тока и напряжения:

```
#include <Battery_Shield.h>              // Подключаем библиотеку Battery_Shield.
Battery_Shield pwrBank;                  // Объявляем объект pwrBank для работы с функциями и методами библиотеки Battery_Shie
                                         //
void setup(){                            //
    Serial.begin(9600);                  // Инициируем передачу данных в монитор последовательного порта на скорости 9600.
    pwrBank.begin(0.0128f);              // Инициируем работу с Battery Shield, указывая номинал сопротивления (0.128 Ом) уста
}                                        // Номинал сопротивления Вашего источника автономного питания указан на вкладыше к Ba
                                         //
void loop(){                                                                 //
    Serial.println("-------------------------");                            //
    Serial.println((String) "Vbat=" + pwrBank.voltmeter(BATTERY)   + "B."); // Выводим напряжение аккумулятора в В.
    Serial.println((String) "Ibat=" + pwrBank.amperemeter(BATTERY) + "A."); // Выводим силу тока  аккумулятора в А.
    Serial.println((String) "Vout=" + pwrBank.voltmeter(OUTPUT)    + "B."); // Выводим напряжение на выходе    в В.
    Serial.println((String) "Iout=" + pwrBank.amperemeter(OUTPUT)  + "A."); // Выводим силу тока  на выходе    в А.
    delay(1000);                                                             // Приостанавливаем выполнение скетча на 1 секунд
}                                                                            //
```

Данный пример будет постоянно выводить I_{BAT}, V_{BAT}, I_{OUT}, V_{OUT} в монитор последовательного порта.

Вывод состояния и заряда аккумулятора:

```
#include <Battery_Shield.h>              // Подключаем библиотеку Battery_Shield.
Battery_Shield pwrBank;                  // Объявляем объект pwrBank для работы с функциями и методами библиотеки Battery_Shie
                                         //
void setup(){                            //
    Serial.begin(9600);                  // Инициируем передачу данных в монитор последовательного порта на скорости 9600.
    pwrBank.begin(0.0128f);              // Инициируем работу с Battery Shield, указывая номинал сопротивления (0.128 Ом) уста
}                                        // Номинал сопротивления Вашего источника автономного питания указан на вкладыше к Ba
                                         //
void loop(){                             //
    switch(pwrBank.getState()){          // Выводим текст в зависимости от значения возвращённого функцией getState():
        case CHARGING_IDLE:              Serial.println(       F("Аккумулятор не заряжается.")
        case CHARGING_TK:                Serial.println(       F("Аккумулятор заряжается малым током.")
        case CHARGING_CC:                Serial.println(       F("Аккумулятор заряжается постоянным током.")
        case CHARGING_CV:                Serial.println(       F("Аккумулятор заряжается постоянным напряжением.")
        case CHARGING_TO:                Serial.println(       F("Аккумулятор не заряжается по причине истечения времени заряд
        default:                         Serial.println(       F("Режим заряда аккумулятора неизвестен.")
    }                                    Serial.println((String) "Текущая ёмкость аккумулятора " + pwrBank.getLevel() + "%"
                                         Serial.println(       F("-------------------------")
    delay(1000);                         // Приостанавливаем выполнение скетча на 1 секунду.
}                                        //
```

Данный пример будет постоянно выводить состояние зарядного устройства, тип заряда и текущую ёмкость аккумулятора (в т.ч. и во время заряда).

Отключение Battery Shield:

```
#include <Battery_Shield.h>              // Подключаем библиотеку Battery_Shield.
```

```
Battery_Shield pwrBank;                  // Объявляем объект pwrBank для работы с функциями и методами библиотеки Battery_Shie
                                         //
void setup(){                            //
    pwrBank.begin(0.0128f);              // Инициируем работу с Battery Shield, указывая номинал сопротивления (0.128 Ом) уста
}                                        // Номинал сопротивления Вашего источника автономного питания указан на вкладыше к Ba
                                         //
void loop(){                             //
    if(millis()>5000){ pwrBank.off(); }  // Если прошло более 5 секунд, то отключаем Battery Shield.
}                                        //
```

Данный пример отключит Battery Shield через 5 секунд после его включения.
Отключение не будет работать, если подано питание на разъем mocro USB.

Программная защита от перегрузки по току:

```
#include <Battery_Shield.h>              // Подключаем библиотеку Battery_Shield.
Battery_Shield pwrBank;                  // Объявляем объект pwrBank для работы с функциями и методами библиотеки Battery_Shie
                                         //
void setup(){                            //
    pwrBank.begin(0.0128f);              // Инициируем работу с Battery Shield, указывая номинал сопротивления (0.128 Ом) уста
}                                        // Номинал сопротивления Вашего источника автономного питания указан на вкладыше к Ba
                                         //
void loop(){                             //
    if(pwrBank.amperemeter(OUTPUT)>0.7){ // Если сила тока на выходе превысит 0.7 А (700 мА), то ...
        pwrBank.off();                   // Отключаем Battery Shield.
    }                                    //
}                                        //
```

Данный скетч отключит Battery Shield если сила тока потребляемая Arduino и другими устройствами превысит 700 мА.

Описание основных функций библиотеки:

Библиотека Battery_Shield позволяет управлять источником автономного питания по шине I2C. Для работы библиотеки, переключатель на плате должен находиться в положении «I2C». Библиотека позволяет: выключать модуль, включать/выключать зарядное устройство, получать значения I_{BAT}, V_{BAT}, I_{OUT}, V_{OUT}, % заряда аккумулятора, текущий тип заряда аккумулятора (TK, CC, CV).

Подключение библиотеки:

```
#include <Battery_Shield.h>              // Подключаем библиотеку Battery_Shield.
Battery_Shield pwrBank;                  // Объявляем объект pwrBank для работы с функциями и методами библиотеки Battery_Shie
```

Функция begin();

- Назначение: Инициализация работы с Battery Shield.
- Синтаксис: begin(RES [, КПД]);
- Параметры:
  - RES - значение сопротивления в цепи аккумулятора, указывается в Ом (тип float).
  - КПД - необязательный параметр, коэффициент полезного действия повышающего DC-DC преобразователя, указывается в % (тип float).
- Возвращаемое значение: (bool) результат инициализации true/false.
- Примечание:
  - RES - сопротивление R_{BAT} значение указано на вкладыше к Battery Shield, оно используется для расчёта I_{BAT}.
  - КПД - значение (по умолчанию 94%) используется для расчёта I_{OUT}.
  - Значение RES выше чем номинал резистора, так как оно учитывает сопротивление дорожек, припоя.
  - Точные значения RES и КПД Вашего Battery Shield можно получить используя функции ohmmeter() и efficiency().
- Пример:

```
pwrBank.begin(0.0128f);                  // Инициализация работы с Battery Shield (Rbat = 0.128 Ом).
```

Функция off();

- Назначение: Выключение Battery Shield.
- Синтаксис: off();
- Параметр: отсутствует.
- Возвращаемое значение: (bool) результат выключения true/false.
- Примечание: Функция не работает при наличии питания на micro USB.
- Пример:

```
pwrBank.off();                          // Выключение Battery Shield.
```

Функция charging();

- Назначение: Включение/выключение зарядного устройства.
- Синтаксис: charging(ФЛАГ);
- Параметр: (bool) флаг разрешающий работу ЗУ. true - разрешить / false - запретить.
- Возвращаемое значение: отсутствует.
- Примечание: Если разрешить работу ЗУ но не подать питание на micro USB, то аккумулятор заряжаться не будет.
- Пример:

```
pwrBank.charging(false);                // Запрещение работы зарядного устройства.
```

Функция getLevel();

- Назначение: Получение уровня заряда аккумулятора в %.
- Синтаксис: getLevel();
- Параметр: отсутствует.
- Возвращаемое значение: (uint8_t) уровень заряда (от 0% до 100%).
- Примечание:
  - возвращаемое значение кратно 5%.
  - ёмкость аккумулятора рассчитывается по его напряжению без нагрузки.
  - функция работает вне зависимости от работы зарядного устройства.

- Пример:

```
Serial.println( pwrBank.getLevel() );    // Выводим ёмкость аккумулятора в монитор последовательного порта.
```

Функция getState();

- Назначение: Получение состояния Battery Shield.
- Синтаксис: getState();
- Параметр: отсутствует.
- Возвращаемое значение:
  - CHARGING_IDLE - в данное время аккумулятор не заряжается.
  - CHARGING_TK - аккумулятор заряжается в режиме TK - малым током.
  - CHARGING_CC - аккумулятор заряжается в режиме CC - постоянным током.
  - CHARGING_CV - аккумулятор заряжается в режиме CV - постоянным напряжением.
  - CHARGING_TO - аккумулятор не заряжается, так как истекло отведённое время заряда.
- Пример:

```
switch(pwrBank.getState()){              // Выводим текст в зависимости от значения возвращённого функцией getState():
    case CHARGING_IDLE: Serial.println(  F( "Аккумулятор не заряжается." )                                  ); break;
    case CHARGING_TK:   Serial.println(  F( "Аккумулятор заряжается малым током.")                          ); break;
    case CHARGING_CC:   Serial.println(  F( "Аккумулятор заряжается постоянным током.")                     ); break;
    case CHARGING_CV:   Serial.println(  F( "Аккумулятор заряжается постоянным напряжением.")               ); break;
    case CHARGING_TO:   Serial.println(  F( "Аккумулятор не заряжается по причине истечения времени заряда.")); break;
    default:            Serial.println(  F( "Режим заряда аккумулятора неизвестен.")                        ); break;
}
```

Функция voltmeter();

- Назначение: Получение напряжения.
- Синтаксис: voltmeter(БЛОК);
- Параметр БЛОК - определяет блок схемы на котором требуется измерить напряжение:

    - BATTERY - измерить напряжение на аккумуляторе (U_{BAT}). Вместо BATTERY можно указать INPUT.
    - BATTERY_IDLE - измерить напряжение на аккумуляторе (U_{BAT}) без нагрузки.
    - OUTPUT - измерить напряжение на выходе (U_{OUT}).
- Возвращаемое значение: (float) значение измеренного напряжения в В.
- Пример:

```
Serial.println(pwrBank.voltmeter(BATTERY)); // Выводим напряжение на аккумуляторе в монитор последовательного порта.
Serial.println(pwrBank.voltmeter(OUTPUT));  // Выводим напряжение на выходе модуля в монитор последовательного порта.
```

Функция amperemeter();

- Назначение: Получение силы тока.
- Синтаксис: amperemeter(ЦЕПЬ);
- Параметр ЦЕПЬ - ток которой требуется получить:
    - BATTERY - измерить ток в цепи аккумулятора (I_{BAT}). Вместо BATTERY можно указать INPUT.
    - OUTPUT- измерить ток нагрузки в цепи выхода (I_{OUT}).
- Возвращаемое значение: (float) значение силы тока в А.
- Примечание:
    - Значение I_{BAT} может быть отрицательным (это значит что аккумулятор заряжается).
    - Значения I_{BAT} и I_{OUT} рассчитываются в библиотеке, а не измеряются чипом модуля.
    - Значение I_{BAT} зависит от R_{BAT} (от сопротивления указанного в качестве параметра функции begin).
    - Значение I_{OUT} зависит от КПД повышающего DC-DC преобразователя (по умолчанию 94%).
    - Точные значения R_{BAT} и КПД Вашего Battery Shield можно получить используя функции ohmmeter() и efficiency().
- Пример:

```
Serial.println(pwrBank.amperemeter(BATTERY)); // Выводим силу тока аккумулятора в монитор последовательного порта.
Serial.println(pwrBank.amperemeter(OUTPUT));  // Выводим силу тока нагрузки в монитор последовательного порта.
```

Описание дополнительных функций библиотеки:

Дополнительные функции требуют ввода токов измеренных внешними приборами (амперметром или мультиметром). Для выполнения указанных измерений требуются технические навыки и приборы. Любые конструктивные изменения в Battery Shield (в т.ч. обрыв проводов, следы пайки и т.д.) исключают гарантию и работоспособность устройства. Помните что аккумулятор является пожароопасным устройством.

Функция ohmmeter();

- Назначение: Получение сопротивления в цепи аккумулятора R_{BAT} (это значение указывается в качестве параметра функции begin). Чем точнее указано значение R_{BAT} в функции begin(), тем точнее библиотека сможет рассчитывать силу тока аккумулятора I_{BAT}.
- Синтаксис: ohmmeter(ТОК_АККУМУЛЯТОРА);
- Параметр: (float) ТОК_АККУМУЛЯТОРА указывается в А.
- Возвращаемое значение: (float) сопротивление в цепи аккумулятора R_{BAT} в Ом.
- Примечание:
  - Для работы функции необходимо указать ток в цепи аккумулятора, который нужно измерить внешним амперметром или мультиметром, при наличии постоянной нагрузки на выходе Battery Shield.
  - Измеренный ток нужно указать в качестве параметра функции ohmmeter(), загрузить скетч и получить сопротивление.
  - Убедитесь что во время получения сопротивления, ток аккумулятора измеряемый амперметром соответствует току указанному в качестве параметра функции ohmmeter().
  - Для лучших результатов рекомендуется повторить операцию при разных нагрузках, а из полученных сопротивлений рассчитать среднее значение.
- Пример:

```
Serial.print("Ibat=0.550 => Rbat="); Serial.print(pwrBank.ohmmeter(0.550f), 5); Serial.println("Ом."); // Выводим Rbat в Ом с
Serial.print("Ibat=0.700 => Rbat="); Serial.print(pwrBank.ohmmeter(0.700f), 5); Serial.println("Ом."); // Выводим Rbat в Ом с
```

Функция efficiency();

- Назначение: Получение КПД повышающего DC-DC преобразователя (это значение указывается в качестве необязательного, второго параметра функции begin). Чем точнее указано КПД в функции begin(), тем точнее библиотека сможет рассчитывать силу тока на выходе I_{OUT}.
- Синтаксис: efficiency(ТОК_ВЫХОДНОЙ_ЦЕПИ);

- Параметр: (float) ТОК_ВЫХОДНОЙ_ЦЕПИ указывается в А.
- Возвращаемое значение: (float) КПД повышающего DC-DC преобразователя в %.
- Примечание:
  - Для работы функции необходимо указать ток в выходной цепи, который нужно измерить внешним амперметром или мультиметром, при наличии постоянной нагрузки.
  - Измеренный ток нужно указать в качестве параметра функции efficiency(), загрузить скетч и получить КПД.
  - Убедитесь что во время получения КПД, ток в выходной цепи измеряемый амперметром соответствует току указанному в качестве параметра функции efficiency().
  - Для лучших результатов рекомендуется повторить операцию при разных нагрузках, а из полученных КПД рассчитать среднее значение.
- Пример:

```
Serial.print("Iout=0.550 => КПД="); Serial.print(pwrBank.efficiency(0.550f), 5); Serial.println("%."); // Выводим КПД в % с 5
Serial.print("Iout=0.700 => КПД="); Serial.print(pwrBank.efficiency(0.700f), 5); Serial.println("%."); // Выводим КПД в % с 5
```

Применение:

- Создание автономных мобильных устройств, роботов;