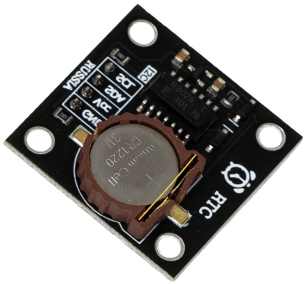
Часы реального времени RTC: руководство по использованию



Линейка часов реального времени поможет вашему устройству стать пунктуальным и выполнять задачи по расписанию. Часы высчитают время для подачи еды питомцу из автоматической кормушки, внесут в график переворачивания яиц в инкубаторе или зажгут ёлку на новый год.

Часы реального времени (RTC — англ. Real Time Clock) служат для получения текущей секунды, минуты, часа, дня, месяца и года без затрат ресурсов микроконтроллера. Модуль пригодиться для создания будильников, сигнализаций и снятия показаний с датчиков по графику.

Список моделей

В нашем магазине мы предлагаем целую линейку часов реального времени на любой вкус и цвет.

Модель	Чип	Основное напряжение VCC	Резервное напряжение VBAT	Форм-фактор
RTC DS1307Z	DS1307Z	5 В	CR2032 / 3 В	---
RTC DS3231	DS3231	3,3–5 В	CR2032 / 3 В	---
RTC DS1307Z (Trema-модуль)	DS1307Z	3,3–5 В	CR1220 / 3 В	Trema-модуль
RTC DS3231 (Trema-модуль)	DS3231	3,3–5 В	CR1220 / 3 В	Trema-модуль
RTC RX8025 (Trema-модуль)	RX8025	3,3–5 В	CR1220 / 3 В	Trema-модуль

Питание

В следующей таблице показано, как себя ведут RTC-модули при наличии или отсутствии основного (внешнего) и дополнительного (от батарейки) питания:

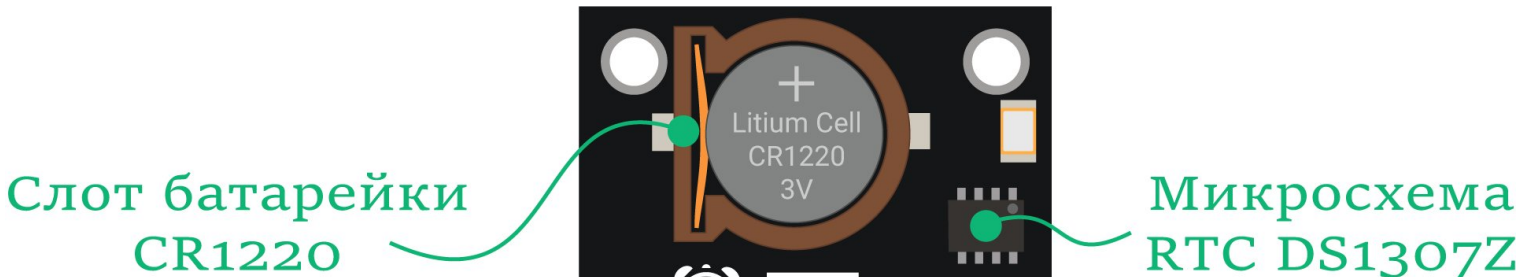
Основное питание VCC	Резервное питание VBAT	Результат
Да	Да	Часы работают от основного питания.
Да	Нет	Часы работают от основного питания.
Нет	Да	Часы работают от резервного питания. RTC продолжают отсчитывать время, но не отвечают на запросы внешнего контроллера.
Нет	Нет	Часы выключены и все регистры сброшены.

Подробности

Рассмотрим подробнее несколько различных моделей RTC-модулей.

- RTC DS1307Z (Trema-модуль)
- RTC DS3231 (Trema-модуль)
- RTC RX8025 (Trema-модуль)

RTC DS1307Z (Trema-модуль)



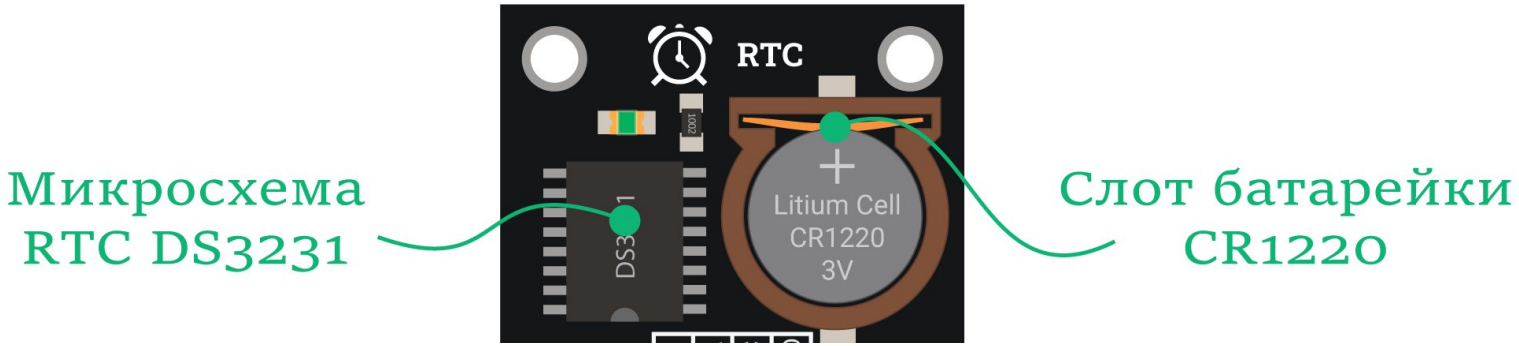
Сердце часов — микросхема RTC DS1307Z, которая занимается подсчётом времени.

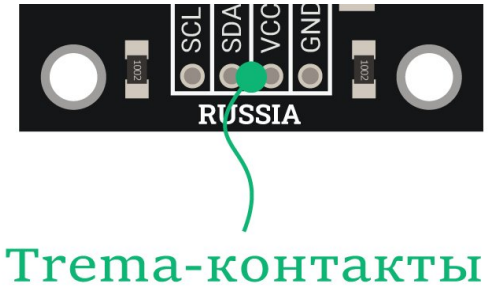
На плате также расположен слот для часовой батарейки размера CR1220 на 3 вольта. Благодаря дополнительному автономному питанию от таблетки, при отключении электропитания часы продолжат идти. Перед началом использования текущие дата и время устанавливаются единожды, а затем они могут быть прочитаны пока жива батарейка.

Плата с часами спроектирована в форм-факторе Trema-модулей — это унифицированный формат, который облегчает подключение датчика к внешним контроллерам, например Arduino или Raspberry Pi.

> С подробностями разобрались, теперь можете смело переходить к подключению часов DS1307Z к контроллеру и примерам работы.

RTC DS3231 (Trema-модуль)





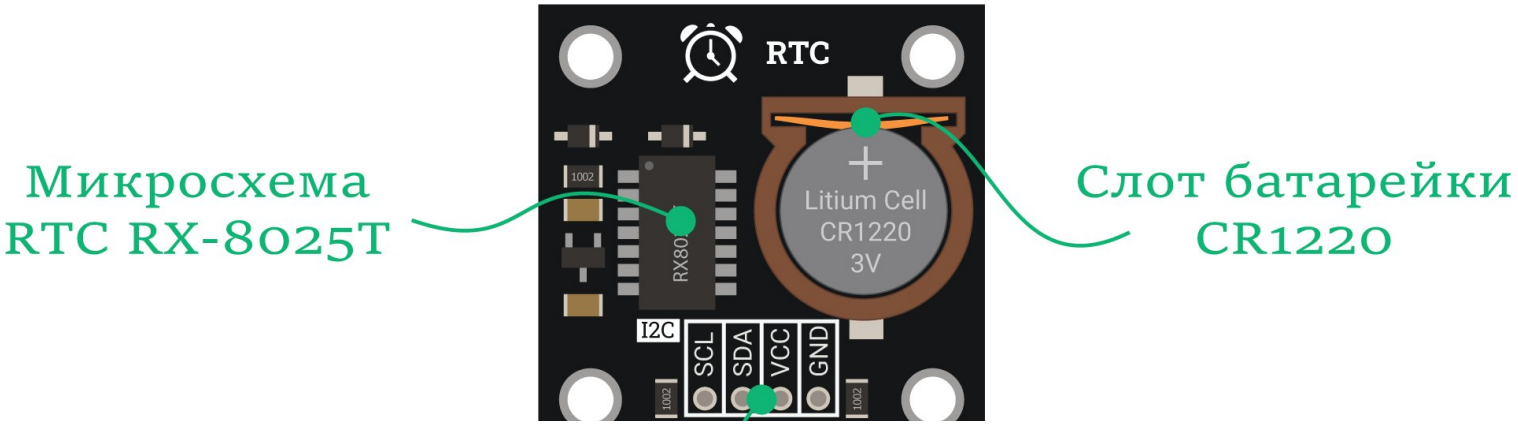
Сердце часов — микросхема RTC DS3231, которая занимается подсчётом времени.

На плате также расположен слот для часовой батарейки размера CR1220 на 3 вольта. Благодаря дополнительному автономному питанию от таблетки, при отключении электропитания часы продолжат идти. Перед началом использования текущие дата и время устанавливаются единожды, а затем они могут быть прочитаны пока жива батарейка.

Плата с часами спроектирована в форм-факторе Trema-модулей — это унифицированный формат, который облегчает подключение датчика к внешним контроллерам, например Arduino или Raspberry Pi.

> С подробностями разобрались, теперь можете смело переходить к подключению часов DS3231 к контроллеру и примерам работы.

RTC RX8025 (Trema-модуль)



Сердце часов — микросхема RTC RX8025, которая занимается подсчётом времени.

На плате также расположен слот для часовой батарейки размера CR1220 на 3 вольта. Благодаря дополнительному автономному питанию от таблетки, при отключении электропитания часы продолжат идти. Перед началом использования текущие дата и время устанавливаются единожды, а затем они могут быть прочитаны пока жива батарейка.

Плата с часами спроектирована в форм-факторе Trema-модулей — это унифицированный формат, который облегчает подключение датчика к внешним контроллерам, например Arduino или Raspberry Pi.

> С подробностями разобрались, теперь можете смело переходить к подключению часов RX8025 к контроллеру и примерам работы.

Подключение и настройка

Рассмотрим подключение различных модулей часов реального времени.

- RTC DS1307Z (Trema-модуль)
- RTC DS3231 (Trema-модуль)
- RTC RX8025 (Trema-модуль)

RTC DS1307Z (Trema-модуль)

Часы RTC DS1307Z подключается к управляющей электронике через группу из четырёх контактов.

Контакт	Функция	Подключение

SDA	Линия данных шины I²C	Подключите к пину SDA микроконтроллера.
SCL	Линия тактирования шины I²C	Подключите к пину SCL микроконтроллера.
VCC	Питание	Подключите к питанию микроконтроллера.
GND	Земля	Подключите к земле микроконтроллера.

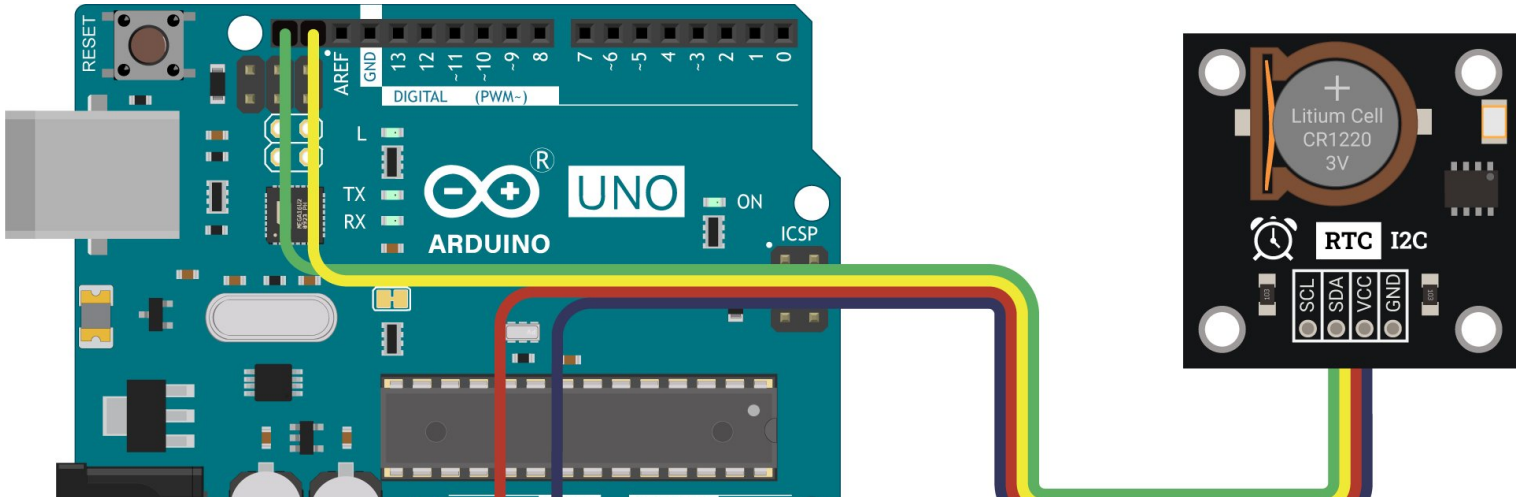
Что понадобится

- 1× RTC DS1307Z (Trema-модуль)
- 1× Arduino Uno
- 1× Соединительные провода «папа-мама»
- 1× Кабель USB (A — B)

Рекомендуем также обратить внимание на дополнительные платы расширения:

- Trema Shield поможет подключить модуль к Arduino с помощью аккуратного шлейфа.
- Trema Set Shield поможет подключить модуль к Arduino без проводов вовсе.

Схема устройства



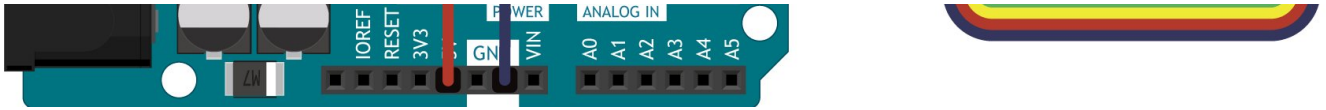


Схема устройства с Trema Shield

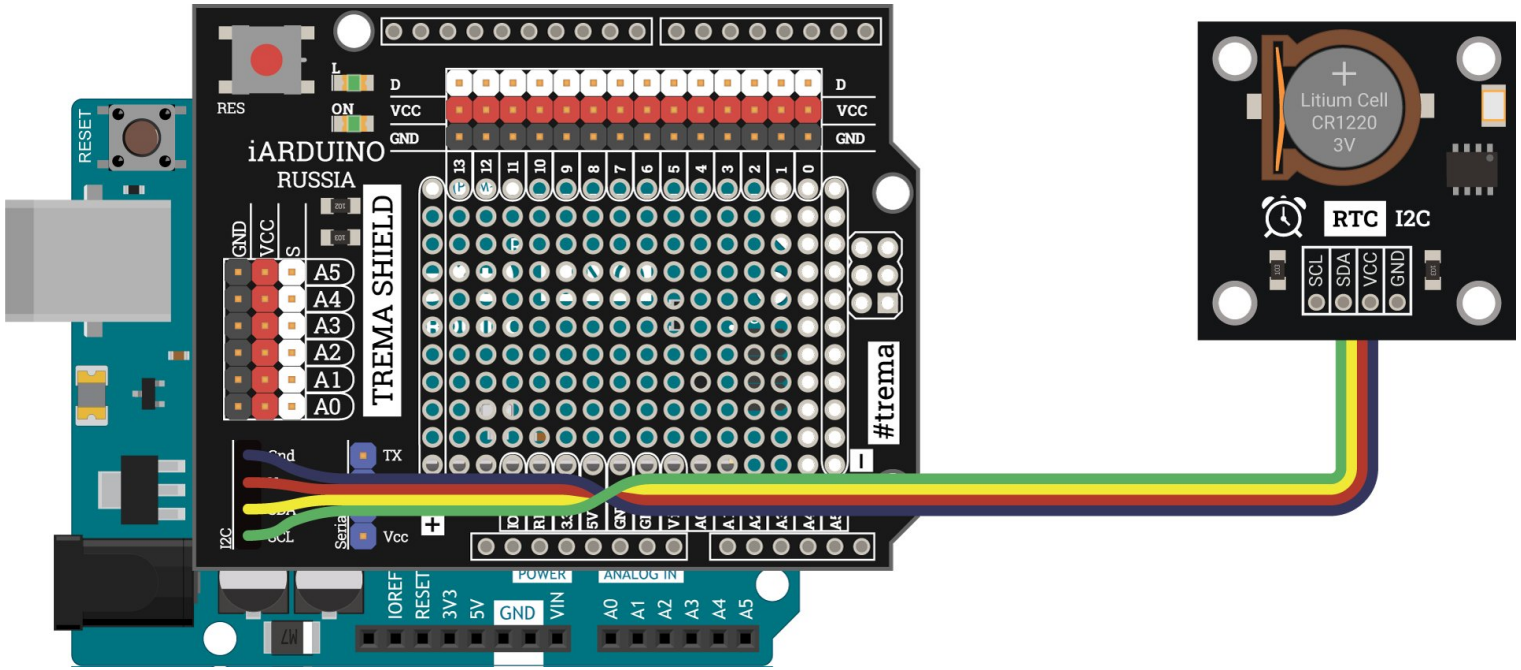
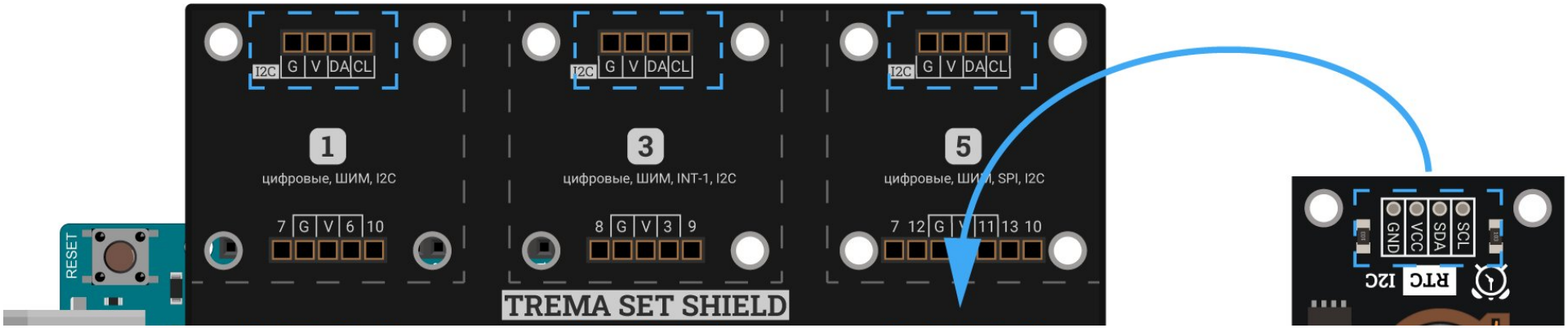
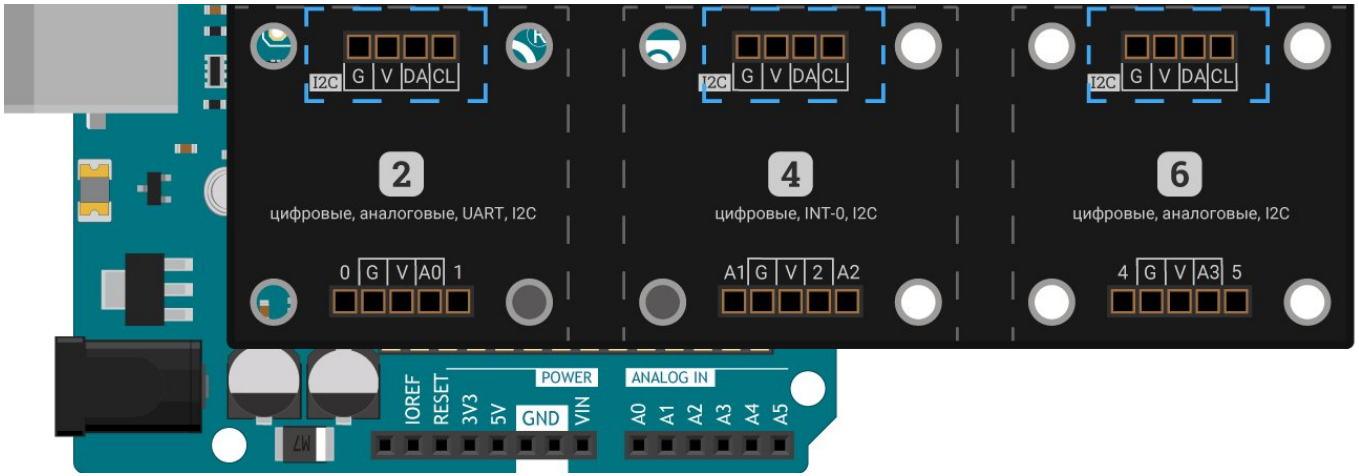


Схема устройства с Trema Set Shield





МОЖНО УСТАНОВИТЬ В ЛЮБУЮ ЯЧЕЙКУ

Программная настройка

1. Настройте плату Arduino Uno в среде Arduino IDE.
2. Скачайте и установите библиотеку `iarduino_RTC` . Для инсталляции рекомендуем использовать нашу инструкцию по установке библиотек для Arduino.
3. Переходите к примерам работы.

RTC DS3231 (Trema-модуль)

Часы RTC DS3231 подключается к управляющей электронике через группу из четырёх контактов.

Контакт	Функция	Подключение
SDA	Линия данных шины I²C	Подключите к пину SDA микроконтроллера.
SCL	Линия тактирования шины I²C	Подключите к пину SCL микроконтроллера.
VCC	Питание	Подключите к питанию микроконтроллера.
GND	Земля	Подключите к земле микроконтроллера.

Что понадобится

- 1× RTC DS3231 (Trema-модуль)
- 1× Arduino Uno
- 1× Соединительные провода «папа-мама»
- 1× Кабель USB (A — B)

Рекомендуем также обратить внимание на дополнительные платы расширения:

- Trema Shield поможет подключить модуль к Arduino с помощью аккуратного шлейфа.
- Trema Set Shield поможет подключить модуль к Arduino без проводов вовсе.

Схема устройства

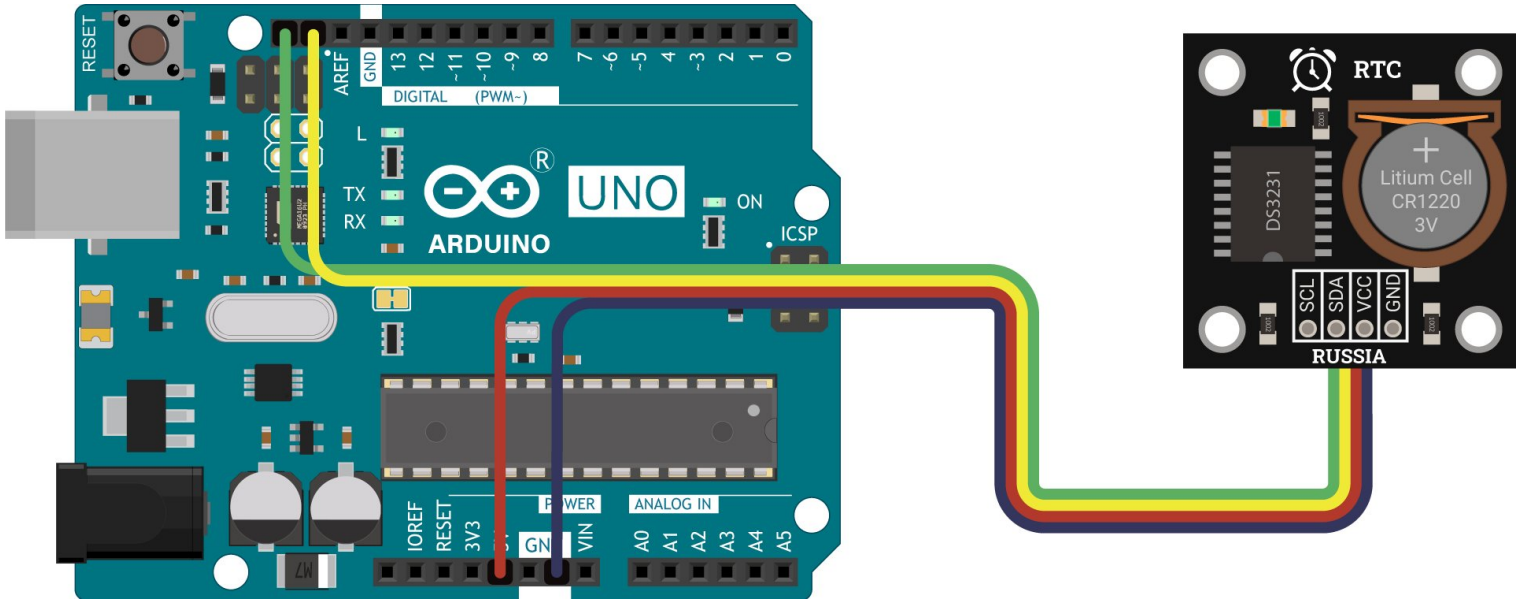
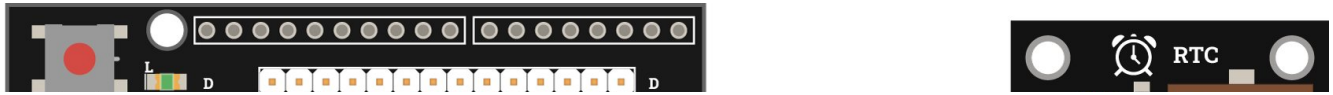


Схема устройства с Trema Shield



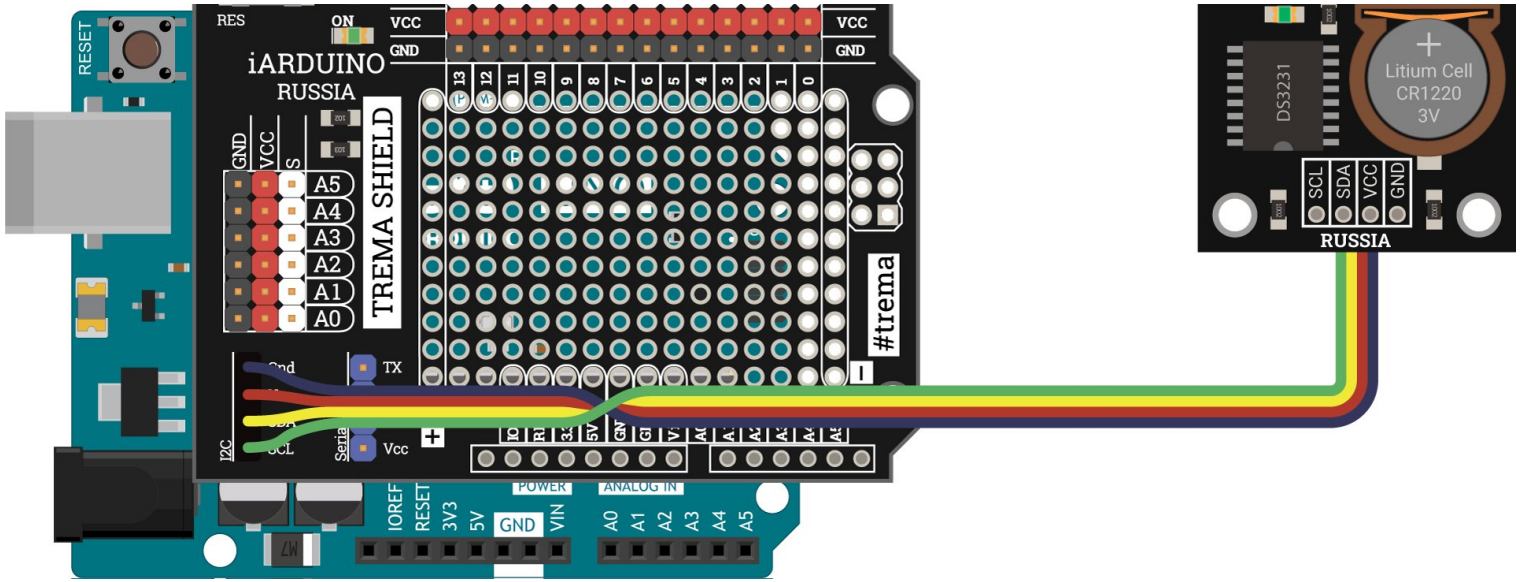
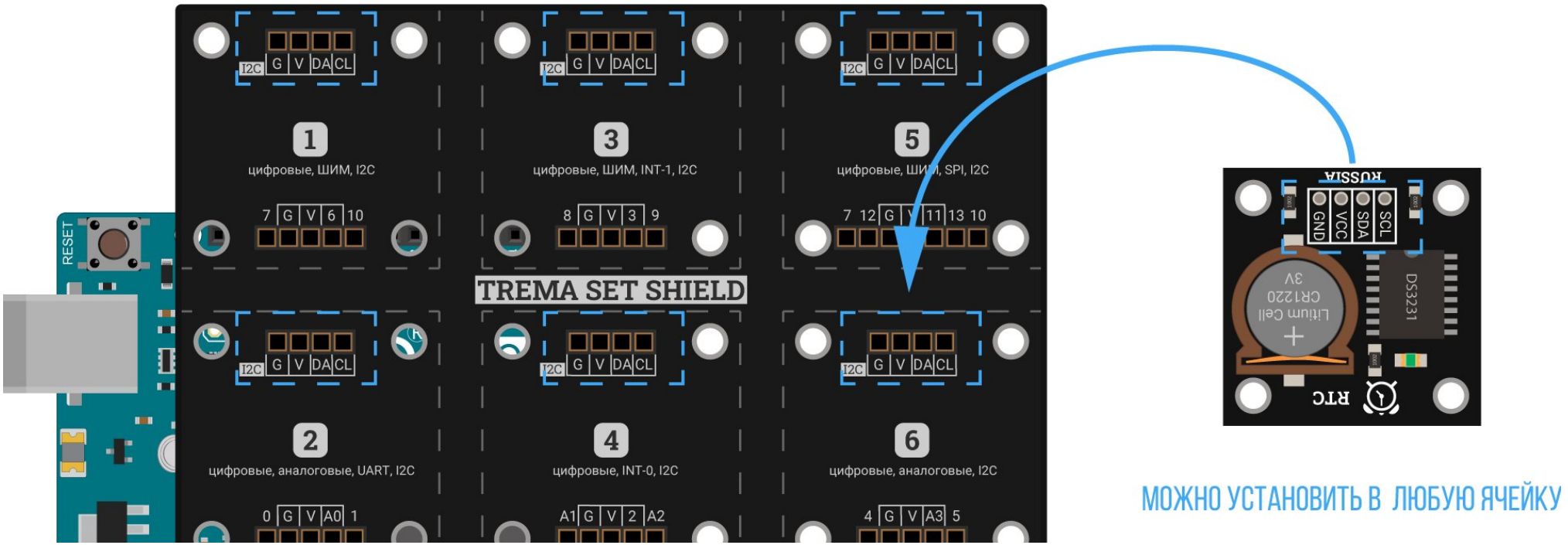
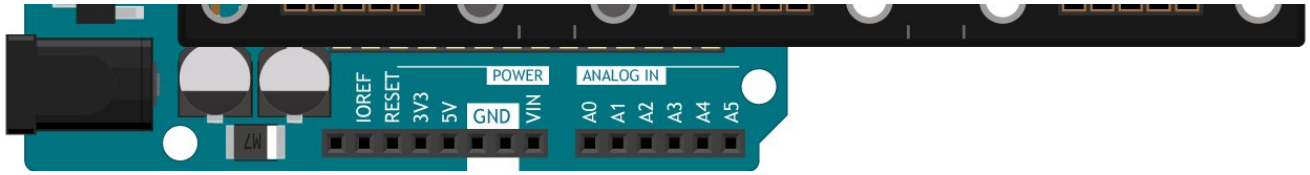


Схема устройства с Trema Set Shield



МОЖНО УСТАНОВИТЬ В ЛЮБУЮ ЯЧЕЙКУ



Программная настройка

1. Настройте плату Arduino Uno в среде Arduino IDE.
2. Скачайте и установите библиотеку `iarduino_RTC` . Для инсталляции рекомендуем использовать нашу инструкцию по установке библиотек для Arduino.
3. Переходите к примерам работы.

RX8025 (Trema-модуль)

Часы RTC RX8025 подключается к управляющей электронике через группу из четырёх контактов.

Контакт	Функция	Подключение
SDA	Линия данных шины I²C	Подключите к пину SDA микроконтроллера.
SCL	Линия тактирования шины I²C	Подключите к пину SCL микроконтроллера.
VCC	Питание	Подключите к питанию микроконтроллера.
GND	Земля	Подключите к земле микроконтроллера.

Что понадобится

- 1× RTC RX8025 (Trema-модуль)
- 1× Arduino Uno
- 1× Соединительные провода «папа-мама»
- 1× Кабель USB (A — B)

Рекомендуем также обратить внимание на дополнительные платы расширения:

- Trema Shield поможет подключить модуль к Arduino с помощью аккуратного шлейфа.
- Trema Set Shield поможет подключить модуль к Arduino без проводов вовсе.

Схема устройства

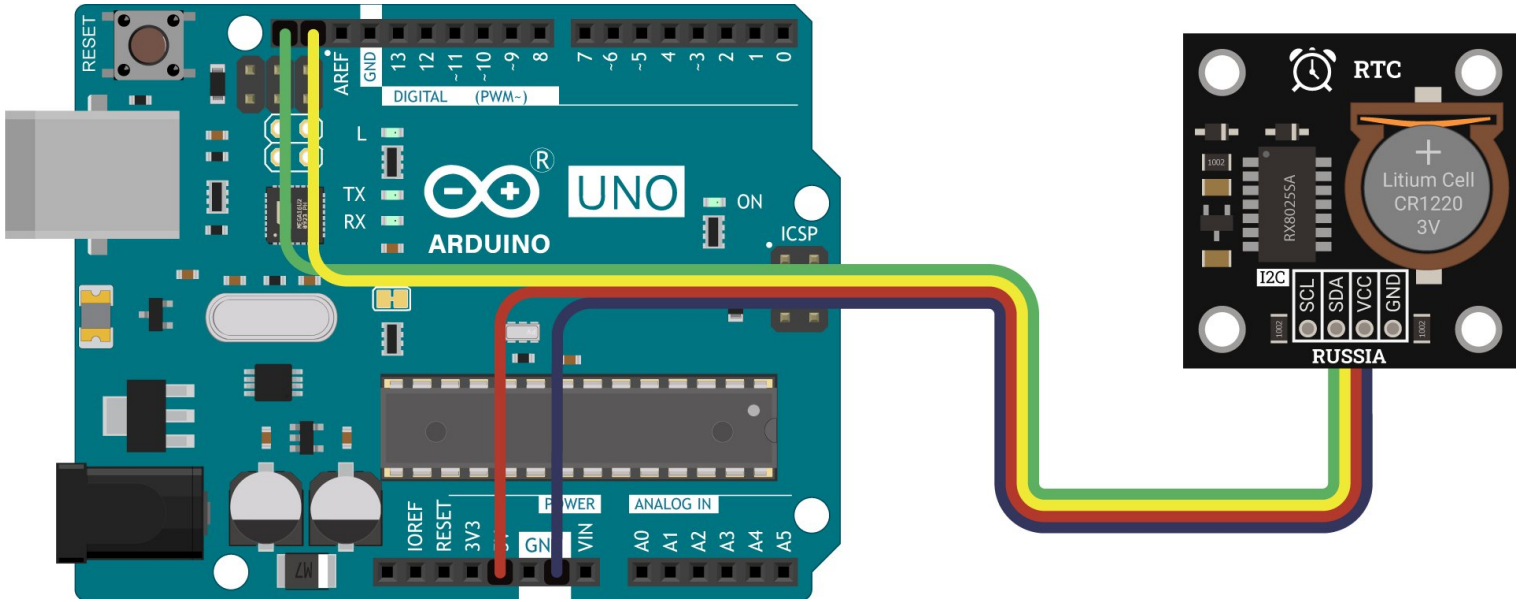
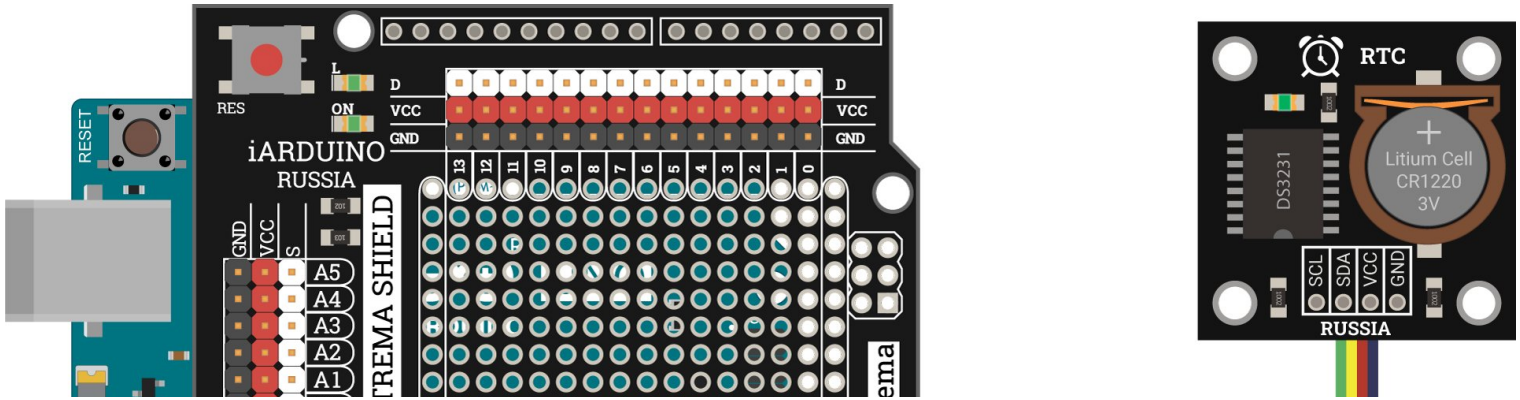


Схема устройства с Trema Shield



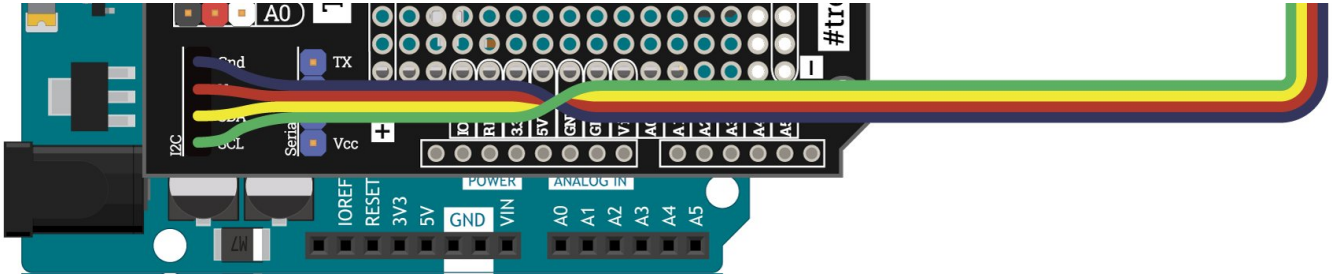
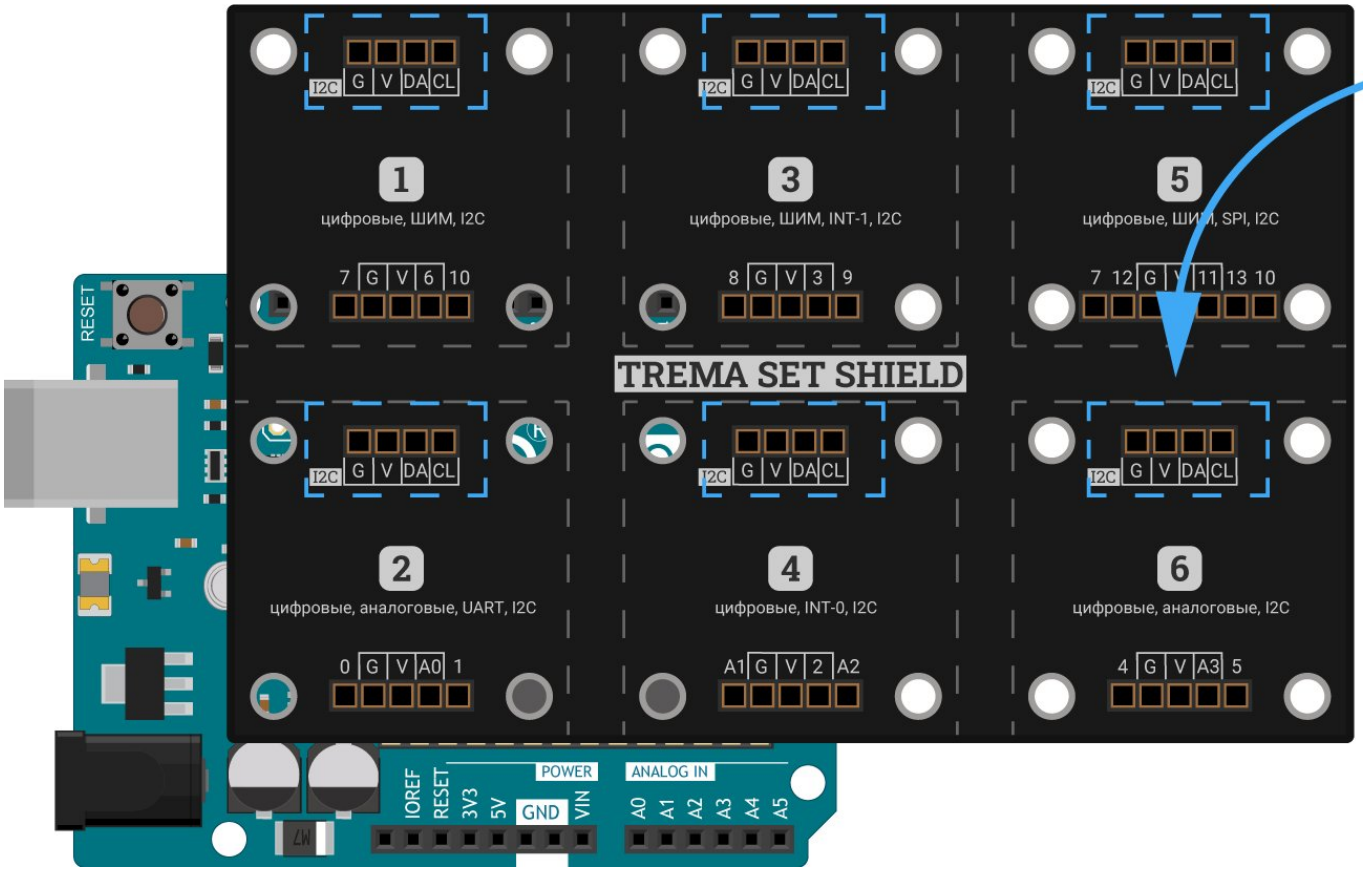
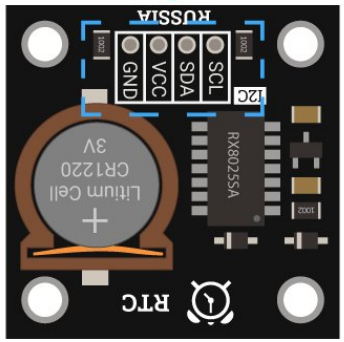


Схема устройства с Trema Set Shield





МОЖНО УСТАНОВИТЬ В ЛЮБУЮ ЯЧЕЙКУ

Программная настройка

1. Настройте плату Arduino Uno в среде Arduino IDE.
2. Скачайте и установите библиотеку `iarduino_RTC` . Для инсталляции рекомендуем использовать нашу инструкцию по установке библиотек для Arduino.
3. Переходите к примерам работы.

Примеры работы

В зависимости от модели часов RTC, используйте соответствующий объект в конструкторе. Весь остальной код подойдёт для всех чипов.

- RTC DS1307Z (Trema-модуль)
- RTC DS3231 (Trema-модуль)
- RTC RX8025 (Trema-модуль)

RTC DS1307Z (Trema-модуль)

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>

// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1307
iarduino_RTC watch(RTC_DS1307);
```

RTC DS3231 (Trema-модуль)

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>

// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS3231
iarduino_RTC watch(RTC_DS3231);
```

RTC RX8025 (Trema-модуль)

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>

// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_RX8025
iarduino_RTC watch(RTC_RX8025);
```

Установка времени

Для начала установим дату и время в RTC-модуль.

Исходный код

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>

// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1307
iarduino_RTC watch(RTC_DS1307);

// В параметрах указываем тип микросхемы – RTC_DS3231
// iarduino_RTC watch(RTC_DS3231);

// В параметрах указываем тип микросхемы – RTC_RX8025
// iarduino_RTC watch(RTC_RX8025);

void setup() {
  // Открываем Serial-порт
  Serial.begin(9600);
  // Инициализируем работу с RTC-модулем
  watch.begin();
  // Устанавливаем время в модуль: 12:30:00 25 октября 2022 года
```

```
  // 00 сек, 30 мин, 12 час, 25 число, октябрь, 2022 год, вторник
  watch.settime(0, 30, 12, 25, 10, 22, 2);
}

void loop() {
  // Если прошла одна секунда
  if (millis() % 1000 == 0) {
    // Выводим временную отметку одной строкой
    Serial.println(watch.gettime("d-m-Y, H:i:s, D"));
  }
}
```

Результат работы

После прошивки устройства, в модуль запишется дата и время указанные в коде программы. Далее «часы затикают» и временная отметка будет выводится в консоль.

Обратите внимание на то, что повторная загрузка скетча, нажатие кнопки Reset на плате Arduino, закрытие и открытие монитора последовательного порта, отключение и подача питания, приведут к тому что код в теле функции setup() так же повторно выполнится и в RTC-модуль вновь запишется указанное время. Если вы не желаете повторной перезаписи времени, сразу после выполнения данного скетча, загрузите скетч вывода времени в котором нет функции `settime()` .

Установка времени автоматически

Время и дату также можно получить из компьютера при компиляции программы.

Исходный код

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>
// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1307
```

```
iarduino_RTC watch(RTC_DS1307);

// В параметрах указываем тип микросхемы – RTC_DS3231
// iarduino_RTC watch(RTC_DS3231);

// В параметрах указываем тип микросхемы – RTC_RX8025
// iarduino_RTC watch(RTC_RX8025);

void setup() {
  // Открываем Serial-порт
  Serial.begin(9600);
  // Инициализируем работу с RTC-модулем
  watch.begin();
  // Устанавливаем время компиляции скетча:
  watch.settime(__TIMESTAMP__);
}

void loop() {
  // Если прошла одна секунда
  if (millis() % 1000 == 0) {
    // Выводим временную отметку одной строкой
    Serial.println(watch.gettime("d-m-Y, H:i:s, D"));
  }
}
```

Результат работы

После прошивки устройства, в модуль запишется дата и время полученные из компьютера при компиляции программы. Далее «часы затикают» и временная отметка будет выводится в консоль.

В скетче использован препроцессорный макрос `__TIMESTAMP__` который передаёт функции `settime()` строку вида: `"Tue Oct 25 12:30:00 2022"` . Вместо макроса можно использовать строку соблюдая последовательность указания временных меток через пробелы: три

символа названия дня недели, три символа названия месяца, две цифры текущего дня в месяце, время в формате ЧЧ:ММ:СС и четыре цифры года.

Вывод времени строкой

Выведем время и дату в консоль одной текстовой строкой.

Исходный код

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>

// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1307
iarduino_RTC watch(RTC_DS1307);

// В параметрах указываем тип микросхемы – RTC_DS3231
// iarduino_RTC watch(RTC_DS3231);

// В параметрах указываем тип микросхемы – RTC_RX8025
// iarduino_RTC watch(RTC_RX8025);

void setup() {
  // Открываем Serial-порт
  Serial.begin(9600);
  // Инициируем работу с RTC-модулем
  watch.begin();
}

void loop() {
  // Если прошла одна секунда
  if (millis() % 1000 == 0) {
    // Выводим время одной строкой согласно указанному шаблону
```

```
    Serial.println(watch.gettime("d-m-Y, H:i:s, D"));
  }
}
```

Результат работы

После прошивки устройства, «часы затикают» и временная отметка будет выводится в консоль.

> Дата и время выводится по шаблону функции date из PHP. Для изменения формата вывода, читайте документацию на библиотеку `iarduino_RTC` в конце статьи.

Вывод времени числами

Выведем время и дату в консоль через целочисленные переменные.

Исходный код

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>

// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1307
iarduino_RTC watch(RTC_DS1307);

// В параметрах указываем тип микросхемы – RTC_DS3231
// iarduino_RTC watch(RTC_DS3231);

// В параметрах указываем тип микросхемы – RTC_RX8025
// iarduino_RTC watch(RTC_RX8025);

// Объявляем переменные для получения значений:
```

```
// day - день, mon - месяц, year - год,
// hour - часы, min - минуты, sec - секунды, week - день недели
uint8_t day, month, year, hour, minute, second, week;

void setup() {
  // Открываем Serial-порт
  Serial.begin(9600);
  // Инициируем работу с RTC-модулем
  watch.begin();
}

void loop() {
  // Если прошла одна секунда
  if (millis() % 1000 == 0) {
    // Считываем текущее время из модуля в буфер библиотеки.
    watch.gettime();
    // Получаем из буфера библиотеки → текущий день месяца: от 1 до 31
    day = watch.day;
    // Получаем из буфера библиотеки → текущий месяц: от 1 до 12
    month = watch.month;
    // Получаем из буфера библиотеки → текущий год: от 0 до 99 (от 2000 до 2099)
    year = watch.year;
    // Получаем из буфера библиотеки → текущие часы: от 0 до 23
    hour = watch.Hours;
    // Получаем из буфера библиотеки → текущие минуты: от 0 до 59
    minute = watch.minutes;
    // Получаем из буфера библиотеки → текущие секунды: от 0 до 59
    second = watch.seconds;
    // Получаем из буфера библиотеки → текущий день недели: от 0 до 6 (0-ВС, 1-ПН...6-СБ)
    week = watch.weekday;
    // Выводим временную отметку через целочисленные переменные
    Serial.print(day);
    Serial.print("-");
```

```
    Serial.print(month);
    Serial.print("-");
    Serial.print(year);
    Serial.print(", ");
    Serial.print(hour);
    Serial.print(":");
    Serial.print(minute);
    Serial.print(":");
    Serial.print(second);
    Serial.print(", ");
    Serial.println(week);
  }
}
```

Результат работы

После прошивки устройства, «часы затикают» и временная отметка будет выводится в консоль.

Будильник

Не засиделся ли ты за компьютером, может завтра на работу? Заведём будильник на 7:00 утра в будний день.

Исходный код

```
// Подключаем библиотеку для работы с RTC
#include <iarduino_RTC.h>

// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1307
iarduino_RTC watch(RTC_DS1307);

// В параметрах указываем тип микросхемы – RTC_DS3231
// iarduino_RTC watch(RTC_DS3231);
```

```
// В параметрах указываем тип микросхемы – RTC_RX8025
// iarduino_RTC watch(RTC_RX8025);

void setup() {
  // Открываем Serial-порт
  Serial.begin(9600);
  // Инициируем работу с RTC-модулем
  watch.begin();
}

void loop() {
  // Если прошла одна секунда
  if (millis() % 1000 == 0) {
    // Выводим временную отметку одной строкой
    Serial.println(watch.gettime("d-m-Y, H:i:s, D"));
    // Если на часах 7 утра в будний день
    if (watch.Hours == 7 && watch.minutes == 0 && watch.seconds == 0 &&
        watch.weekday!=0 && watch.weekday != 6) {
      // Выводим сообщение: «07:00 пора на работу!»
      Serial.println("07:00 пора на работу!");
    }
  }
}
```

Результат работы

После прошивки устройства, «часы затикают» и временная отметка будет выводится в консоль. А при достижении 7:00 по будням, сработает будильник с призывом идти на работу.

Библиотека для Arduino

Для работы с линейкой часов реального времени с контроллерами Arduino мы разработали библиотеку iarduino_RTC.

Поддерживаемые модели

- RTC DS1307Z
- RTC DS3231
- RTC DS1307Z (Trema-модуль)
- RTC DS3231 (Trema-модуль)
- RTC RX8025 (Trema-модуль)

Установка

Для старта скачайте и установите библиотеку iarduino_RTC. Для инсталляции рекомендуем использовать нашу инструкцию по установке библиотек для Arduino.

Подключение

- Назначение: подключение библиотеки.
- Синтаксис: `#include <iarduino_RTC.h>`
- Примечания:
  - Библиотека подключается в самом начале программы.
  - Подключение библиотеки обязательное действие, иначе функции работать не будут.
- Примеры:

```
// Подключаем библиотеку для работы с RTC-модулем
#include <iarduino_RTC.h>
```

Конструктор

- Назначение: создание объекта для работы с функциями библиотеки `iarduino_RTC` .
- Синтаксис: `iarduino_RTC watch(uint8_t chipRTC, uint8_t pinSS, uint8_t pinSCK, uint8_t pinMOSI)`
- Параметры:
  - `chipRTC` : модель выбранных часов реального времени.

        - `RTC_DS1302` : указывайте для часов на базе чипа DS1302. Обмен данными будет на шине SPI.
        - `RTC_DS1307Z` : указывайте для часов на базе чипа DS1307Z. Обмен данными будет на шине I²C.
        - `RTC_DS3231` : указывайте для часов на базе чипа DS3231. Обмен данными будет на шине I²C.
        - `RTC_RX8025` : указывайте для часов на базе чипа RX8025. Обмен данными будет на шине I²C.
    - `pinSS` : пин выбора устройства на шине SPI. Параметр необходимо указывать, только для устройств на шине SPI, в нашем случае при создании объекта модели `DS1302` .
    - `pinSCK` : тактовый сигнал на шине SPI. Параметр необходимо указывать, только для устройств на шине SPI, в нашем случае при создании объекта модели `DS1302` .
    - `pinMOSI` : пин выходных данных на шине SPI. Параметр необходимо указывать, только для устройств на шине SPI, в нашем случае при создании объекта модели `DS1302` .
- Возвращаемое значение: нет
- Примечания:
    - Конструктор вызывается в самом начале программы.
    - Вызов конструктора обязателен, иначе функции работать не будут.
    - Для часов на базе микросхем `DS1307` , `DS3231` и `RX8025` — указывать дополнительные сигнальные пины не нужно.
- Примеры:

При использовании RTC-модуля на микросхеме DS1302

```
// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1302
const int pinSS = 10;
const int pinSCK = 13;
const int pinMOSI = 11;
iarduino_RTC watch(RTC_DS1302, pinSS, pinSCK, pinMOSI);
```

При использовании RTC-модуля на микросхеме DS1307Z

```
// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS1307
```

```
iarduino_RTC watch(RTC_DS1307);
```

При использовании RTC-модуля на микросхеме DS3231

```
// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_DS3231
iarduino_RTC watch(RTC_DS3231);
```

При использовании RTC-модуля на микросхеме RX8025

```
// Создаём объект watch для работы с функциями библиотеки iarduino_RTC
// В параметрах указываем тип микросхемы – RTC_RX8025
iarduino_RTC watch(RTC_RX8025);
```

Функция begin()

- Назначение: инициализация работы с модулем.
- Синтаксис: `void begin()`
- Параметры: нет
- Возвращаемое значение: нет
- Примечания:
  - Вызов функции `begin()` обязателен, иначе часы работать не будут.
  - Функция `begin()` достаточно вызвать один раз.
- Пример:

```
// Инициируем работу с RTC-модулем
watch.begin();
```

Функция settime()

В библиотеке доступны две перегруженных функции `settime()`:

- Функция `settime()` с указанием даты и времени числами
- Функция `settime()` с указанием даты и времени строкой

Функция settime() с указанием даты и времени числами

- Назначение: установка даты и времени.
- Синтаксис: `void settime(int second, int minute, int hour, int day, int month, int year, int weekday)`
- Параметры:
  - `second`: секунды. Доступный диапазон значений от `0` до `59`.
  - `minute`: минуты. Доступный диапазон значений от `0` до `59`. Если параметр устанавливать не нужно, указывайте значение `-1`.
  - `hour`: часы. Доступный диапазон значений от `0` до `23`. Если параметр устанавливать не нужно, указывайте значение `-1`.
  - `day`: число. Доступный диапазон значений от `1` до `31`. Если параметр устанавливать не нужно, указывайте значение `-1`.
  - `month`: месяц. Доступный диапазон значений от `1` до `12`. Если параметр устанавливать не нужно, указывайте значение `-1`.
  - `year`: год без учёта века. Доступный диапазон значений от `1` до `99`. Если параметр устанавливать не нужно, указывайте значение `-1`.
  - `weekday`: день недели. Доступный диапазон от `0` до `6`, где `0` ВС, `1` ПН ... `6` СБ. Если параметр устанавливать не нужно, указывайте значение `-1`.
- Возвращаемое значение: нет
- Примечания:
  - Все параметры, кроме `second`, можно не указывать.
  - Если параметр устанавливать не нужно, указывайте значение `-1`. Подходит для всех параметров, кроме `second`.
  - Время и дату также можно указать одной строкой или установить через количество секунд прошедших с начала эпохи Unix с помощью функции `settimeUnix()`.
- Примеры:

```
// Устанавливаем временную отметку в модуль: 12:30:00 25 октября 2022 года Вторник
// 00 сек, 30 мин, 12 час, 25 число, октябрь, 2022 год, вторник
watch.settime(0, 30, 12, 25, 10, 22, 1);
```

```
// Устанавливаем временную отметку в модуль: 12:30:00
watch.settime(0, 30, 12);
// Равносильная запись
watch.settime(0, 30, 12, -1, -1, -1, -1);
```

```
// Устанавливаем временную отметку в модуль: 25 октября 2022 года
watch.settime(-1, -1, -1, 25, 10, 22);
// Равносильная запись
watch.settime(-1, -1, -1, 25, 10, 22, -1);
```

```
// Устанавливаем временную отметку в модуль: 25 октября 2022 года Вторник
watch.settime(-1, -1, -1, 25, 10, 22, 1);
```

Функция settime() с указанием даты и времени строкой

- Назначение: установка даты и времени.
- Синтаксис:
  - `void settime(String time)`
  - `void settime(const char* time)`
- Параметры:
  - `time` : Строка устанавливаемого времени в формате `"Www Mmm DD HH:MM:SS YYYY"`
    - `Www` : Сокращённое название дня недели (анг.) указывается тремя символами (`Sun` =Вс, `Mon` =Пн, `Tue` =Вт, `Wed` =Ср, `Thu` =Чт, `Fri` =Пт, `Sat` =Сб)
    - `Mmm` : Сокращённое название месяца (анг.) указывается тремя символами (`Jan` , `Feb` , `Mar` , `Apr` , `May` , `Jun` , `Jul` , `Aug` , `Sep` , `Oct` , `Nov` , `Dec`)
    - `DD` : День месяца, указывается двумя цифрами от `01` до `31` .
    - `HH:MM:SS` : Время в 24 часовом формате от `00:00:00` до `23:59:59` .

    - `YYYY` : Год, указывается четырьмя цифрами от `0000` до `9999` .
- Возвращаемое значение: нет
- Примечания:
    - В качестве параметра можно использовать препроцессорным макрос `__TIMESTAMP__` , который возвращает строку описанного выше формата с указанием даты и времени последней компиляции скетча.
    - Время и дату также можно указать числами или установить через количество секунд прошедших с начала эпохи Unix с помощью функции `settimeUnix()` .
- Примеры:

```
// Устанавливаем временную отметку в модуль: Вторник 25 октября 2022 года 12:30:00
watch.settime("Tue Oct 25 12:30:00 2022");
```

```
// Устанавливаем в модуль текущее время с компьютера
watch.settime(__TIMESTAMP__);
```

Функция gettime()

В библиотеке доступны две перегруженных функции `gettime()` :

- Функция `gettime()` без параметров
- Функция `gettime()` с параметром

Функция gettime() без параметров

- Назначение: чтение даты и времени.
- Синтаксис: `void gettime()`
- Параметры: нет
- Возвращаемое значение: нет
- Примечания:
    - Функция обновляет значения приватных переменных: `seconds` , `minutes` , `hours` , `Hours` , `midday` , `day` , `weekday` ,

`month`, `year` и `Unix`. Подробности смотрите в описании конкретных переменных.
    - Функцию `gettime()` можно вызывать с параметром `time`, тогда чтение даты и времени будет происходить по шаблону функции date из PHP. Подробности описаны в функции `gettime()` с параметром.
- Пример:

```
// Считываем текущее время из модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущий день месяца
day = watch.day;
// Получаем из буфера библиотеки → текущий месяц
month = watch.month;
// Получаем из буфера библиотеки → текущий год
year = watch.year;
// Получаем из буфера библиотеки → текущие часы
hour = watch.Hours;
// Получаем из буфера библиотеки → текущие минуты
minute = watch.minutes;
// Получаем из буфера библиотеки → текущие секунды
second = watch.seconds;
// Получаем из буфера библиотеки → текущий день недели
week = watch.weekday;
```

Функция gettime() с параметром

- Назначение: чтение даты и времени по шаблону функции date из PHP.
- Синтаксис:
    - `void gettime(String time)`
    - `void gettime(const char* time)`
- Параметры:
    - `time` - строка шаблон, по которой будет составлена возвращаемая строка.
- Возвращаемое значение: строка где символы шаблона заменены на временные метки.

- `s` : секунды от `00` до `59`
- `i` : минуты от `00` до `59`
- `h` : часы в 12-часовом формате от `00` до `12`
- `H` : часы в 24-часовом формате от `00` до `24`
- `d` : день месяца от `00` до `31`
- `w` : день недели от `0` до `6` (`0` -ВС, `1` -ПН, ... , `6` -СБ)
- `D` : день недели в текстовом виде: `Mon` , `Tue` , `Wed` , `Thu` , `Fri` , `Sat` и `Sun`
- `m` : месяц от `01` до `12`
- `M` : месяц в текстовом виде: `Jan` , `Feb` , `Mar` , `Apr` , `May` , `Jun` , `May` , `Jul` , `Aug` , `Sep` , `Oct` , `Nov` и `Dec`
- `Y` : год от `2000` до `2099`
- `y` : год от `00` до `99`
- `a` : полдень `am` или `pm`
- `A` : полдень `AM` или `PM`

- Примечания:
  - Строка `time` не должна превышать 50 символов.
  - Функция не только возвращает время, но и обновляет значения приватных переменных: `seconds` , `minutes` , `hours` , `Hours` , `midday` , `day` , `weekday` , `month` , `year` и `Unix` .
  - Функцию `gettime()` можно вызывать без параметра.
- Примеры:

```
// Переменная для хранения временной метки
String time;
// Считываем текущее время в 24-формате в переменную time
time = watch.gettime("H:i:s");
// Выведем полученные данные в консоль
Serial.println(time); // Пример результата: 12:30:00
```

```
// Переменная для хранения временной метки
String time;
```

```
// Считываем текущее время в 12-формате в переменную time
time = watch.gettime("h:i:s A");
// Выведем полученные данные в консоль
Serial.println(time); // Пример результата: 12:30:00 PM
```

```
// Переменная для хранения временной метки
String timestamp;
// Считываем текущую временную метку в переменную timeStamp
timeStamp = watch.gettime("d-m-Y, H:i:s, D");
// Выведем полученные данные в консоль
Serial.println(timeStamp); // Пример результата: 25-10-2022, 12:30:00, Tue
```

```
// Выведем сразу текущую временную метку в консоль
Serial.println(watch.gettime("d-m-Y, H:i:s, D")); // 25-10-2022, 12:30:00, Tue
```

Функция blinktime()

- Назначение: мигание одним из значений времени при использовании функции `gettime()` с параметром.
- Синтаксис: `void blinktime(uint8_t value, float frequency = 1)`
- Параметры:
  - `value` : параметр времени от `0` до `8` :
    - `0` : не мигать.
    - `1` : мигать секундами, параметр `s` .
    - `2` : мигать минутами, параметр `i`
    - `3` : мигать часами, параметр `h` или `H` .
    - `4` : мигать числом в месяце, параметр `d` .
    - `5` : мигать месяцем, параметр `m` или `M` .
    - `6` : мигать годом, параметр `y` или `Y` .
    - `7` : мигать днём недели, параметр `w` или `D` .
    - `8` : мигать полднём, параметр `a` или `A` .

    - `frequency` : частота мигания в Герцах. По умолчанию частота мигания установлена `1` , т.е. 1 Герц.
- Возвращаемое значение: нет
- Примечания:
    - Функция `blinktime` настраивает мигание параметров при использовании функции `gettime()` .
- Примеры:

```
// Настраиваем мигание минутами с частотой 1 Гц (значение по умолчанию)
watch.blinktime(2);
```

```
// Настраиваем мигание минутами с частотой 5 Гц
watch.blinktime(2, 5);
```

Функция period()

- Назначение: установка периода обращения к RTC-модулю.
- Синтаксис: `void period(uint8_t value)`
- Параметры:
    - `value` : отрезок времени, в течении которого к модулю может быть отправлен только один запрос. Доступный диапазон значений указывается в минутах от `0` до `255` . При значении `0` , вызов функции `gettime()` всегда генерирует запрос к модулю.
- Возвращаемое значение: нет
- Примечания:
    - Данная функция указывает функции `gettime()` откуда брать текущее время и дату: из модуля (не чаще заданного периода) или рассчитать в библиотеке (без обращения к модулю).
    - Функция полезна, если шина передачи данных сильно нагружена другими устройствами.
- Примеры:

```
// Устанавливаем период обращения к RTC-модулю в одну минуту
// Если в течении одной минуты вызвать несколько функций gettime():
// Первый запрос → дата и время придут с модуля
```

```
// Остальные запросы в течении минуты → дату и время высчитает библиотека
watch.period(1);
```

Функция settimeUnix()

- Назначение: установка даты и времени в секундах с начала эпохи Unix.
- Синтаксис: `void settimeUnix(uint32_t second)`
- Параметры:
  - `second` : количество секунд прошедшее с начала эпохи Unix.
- Возвращаемое значение: нет
- Примечания:
  - Начало эпохи Unix — это дата 01.01.1970 00:00:00 GMT.
  - Так как количество секунд с начала эпохи Unix указывается по гринвичу и одинаково во всём мире, то перед установкой времени функцией `settimeUnix()` необходимо однократно указать требуемый часовой пояс функцией `settimezone()` .
  - Время и дату также можно установить привычным способом с помощью функции `settime()` .
- Примеры:

```
// Указываем часовой пояс UTC+3
watch.settimezone(3);
// Устанавливаем временную отметку в модуль
// 12:30:00 25 октября 2022 года UTC+3 (Москва) = 1666690200 секунд с начала эпохи Unix.
watch.settimeUnix(1666690200);
```

Функция gettimeUnix()

- Назначение: чтение даты и времени в формате количества секунд прошедших с начала эпохи Unix.
- Синтаксис: `uint32_t gettimeUnix()`
- Параметры: нет
- Возвращаемое значение:
  - `second` : количество секунд прошедшее с начала эпохи Unix.
- Примечания:

    - Начало эпохи Unix — это дата 01.01.1970 00:00:00 GMT.
    - Так как количество секунд с начала эпохи Unix указывается по гринвичу и одинаково во всём мире, то перед получением времени функцией `gettimeUnix()` необходимо однократно указать используемый часовой пояс функцией `settimezone()`.
    - Время и дату также можно получить привычным способом с помощью функции `gettime()`.
- Примеры:

```
// Указываем используемый часовой пояс UTC+3
watch.settimezone(3);
// Считываем текущее временную метку из модуля
// в формате количества секунд прошедших с начала эпохи Unix
timeUnix = watch.gettimeUnix();
// Выводим данные в консоль
Serial.println(timeUnix);
```

Функция settimezone()

- Назначение: указание часового пояса для функций `settimeUnix()` и `gettimeUnix()`.
- Синтаксис: `void settimezone(int8_t zone)`
- Параметры:
    - `zone`: часовой пояс от `-12` до `12` включительно.
- Возвращаемое значение: нет.
- Примечания:
    - Указанный часовой пояс используется только для расчёта количества секунд прошедших с начала эпохи Unix, указанных функцией `settimeUnix()` или считанных функцией `gettimeUnix()`, так как количество прошедших секунд определяется по гринвичу.
    - Часовой пояс не сохраняется в модуле, но его достаточно однократно указать в теле функции `setup()`. Значение по умолчанию 0.
- Примеры:

```
// Указываем используемый часовой пояс UTC-5
watch.settimezone(-5);
// Выводим количество секунд прошедших с начала эпохи Unix
```

```
Serial.println( watch.gettimeUnix() );
```

Переменная seconds

- Назначение: содержит количество секунд из модуля-RTC.
- Синтаксис: `uint8_t seconds`
- Доступные значения: от `0` до `59`
- Примечания:
  - Для обновления текущей переменой `seconds`, сначала необходимо вызвать функцию `gettime()`.
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущее кол-во секунд
uint8_t seconds = watch.seconds;
// Выводим данные в консоль
Serial.println(seconds);
```

Переменная minutes

- Назначение: содержит количество минут из модуля-RTC.
- Синтаксис: `uint8_t minutes`
- Доступные значения: от `0` до `59`
- Примечания:
  - Для обновления текущей переменой `minutes`, сначала необходимо вызвать функцию `gettime()`.
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущее кол-во минут
uint8_t minutes = watch.minutes;
```

```
// Выводим данные в консоль
Serial.println(minutes);
```

Переменная hours

- Назначение: содержит количество часов в 12-часовом формате из модуля-RTC.
- Синтаксис: `uint8_t hours`
- Доступные значения: от `1` до `12`
- Примечания:
  - Для обновления текущей переменой `hours`, сначала необходимо вызвать функцию `gettime()`.
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущее кол-во часов
uint8_t hours12 = watch.hours;
// Выводим данные в консоль
Serial.println(hours12);
```

Переменная Hours

- Назначение: содержит количество часов в 24-часовом формате из модуля-RTC.
- Синтаксис: `uint8_t Hours`
- Доступные значения: от `0` до `23`
- Примечания:
  - Для обновления текущей переменой `Hours`, сначала необходимо вызвать функцию `gettime()`.
- Примеры:

```
  // Считываем текущее время из RTC-модуля в буфер библиотеки
  watch.gettime();
  // Получаем из буфера библиотеки → текущее кол-во часов
```

```
uint8_t hours24 = watch.Hours;
// Выводим данные в консоль
Serial.println(hours24);
```

Переменная midday

- Назначение: содержит текущее состояние полдня: AM или PM.
- Синтаксис: `uint8_t midday`
- Доступные значения:
  - `0` : AM (Ante Meridiem) — время до полудня. В переводе на 24-часовой формат с 00:00 до 11:59.
  - `1` : PM (Post Meridiem) — время после полудня. В переводе на 24-часовой формат с 12:00 до 23:59.
- Примечания:
  - Для обновления текущей переменой `midday` , сначала необходимо вызвать функцию `gettime()` .
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущее состояние полудня
uint8_t midday = watch.midday;
// Выводим данные в консоль
Serial.println(midday);
```

Переменная day

- Назначение: содержит день месяца из модуля-RTC.
- Синтаксис: `uint8_t day`
- Доступные значения: от `1` до `31`
- Примечания:
  - Для обновления текущей переменой `day` , сначала необходимо вызвать функцию `gettime()` .
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущий день месяца
uint8_t day = watch.day;
// Выводим данные в консоль
Serial.println(day);
```

Переменная weekday

- Назначение: содержит день недели из модуля-RTC.
- Синтаксис: `uint8_t weekday`
- Доступные значения:
  - `0` : воскресенье
  - `1` : понедельник
  - `2` : вторник
  - `3` : среда
  - `4` : четверг
  - `5` : пятница
  - `6` : суббота
- Примечания:
  - Для обновления текущей переменой `weekday` , сначала необходимо вызвать функцию `gettime()` .
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущий день месяца
uint8_t weekday = watch.weekday;
// Выводим данные в консоль
Serial.println(weekday);
```

Переменная month

- Назначение: содержит месяц из модуля-RTC.
- Синтаксис: `uint8_t month`
- Доступные значения:
  - `1` : январь
  - `2` : февраль
  - `3` : март
  - `4` : апрель
  - `5` : май
  - `6` : июнь
  - `7` : июль
  - `8` : август
  - `9` : сентябрь
  - `10` : октябрь
  - `11` : ноябрь
  - `12` : декабрь
- Примечания:
  - Для обновления текущей переменой `month` , сначала необходимо вызвать функцию `gettime()` .
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущий месяц
uint8_t month = watch.month;
// Выводим данные в консоль
Serial.println(month);
```

Переменная year

- Назначение: содержит день недели из модуля-RTC.
- Синтаксис: `uint8_t weekday`
- Доступные значения:
  - `0` : воскресенье
  - `1` : понедельник
  - `2` : вторник
  - `3` : среда
  - `4` : четверг
  - `5` : пятница
  - `6` : суббота
- Примечания:
  - Для обновления текущей переменой `weekday` , сначала необходимо вызвать функцию `gettime()` .
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущий день месяца
uint8_t weekday = watch.weekday;
// Выводим данные в консоль
Serial.println(weekday);
```

Переменная month

- Назначение: содержит месяц из модуля-RTC.
- Синтаксис: `uint8_t month`
- Доступные значения:
  - `1` : январь
  - `2` : февраль
  - `3` : март
  - `4` : апрель

    - `5` : май
    - `6` : июнь
    - `7` : июль
    - `8` : август
    - `9` : сентябрь
    - `10` : октябрь
    - `11` : ноябрь
    - `12` : декабрь
- Примечания:
    - Для обновления текущей переменой `month` , сначала необходимо вызвать функцию `gettime()` .
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущий месяц
uint8_t month = watch.month;
// Выводим данные в консоль
Serial.println(month);
```

Переменная year

- Назначение: содержит год без учёта века из модуля-RTC.
- Синтаксис: `uint8_t year`
- Доступные значения: от `0` до `99`
- Примечания:
    - Для обновления текущей переменой `year` , сначала необходимо вызвать функцию `gettime()` .
- Примеры:

```
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
```

```
// Получаем из буфера библиотеки → текущий год
uint8_t year = watch.year;
// Выводим данные в консоль
Serial.println(year);
```

Переменная Unix

- Назначение: содержит секунды с начала эпохи Unix из модуля-RTC.
- Синтаксис: `uint32_t Unix`
- Доступные значения: от `0` до `4294967295`
- Примечания:
  - Для обновления текущей переменой `Unix`, сначала необходимо вызвать функцию `gettime()`.
  - Начало эпохи Unix — это дата 01.01.1970 00:00:00 GMT.
  - Так как количество секунд с начала эпохи Unix указывается по гринвичу и одинаково во всём мире, то перед обращением к функции `gettime()` необходимо однократно указать используемый часовой пояс функцией `settimezone()`.
- Примеры:

```
// Указываем используемый часовой пояс UTC+3
watch.settimezone(3);
// Считываем текущее время из RTC-модуля в буфер библиотеки
watch.gettime();
// Получаем из буфера библиотеки → текущее количество секунд с начала эпохи Unix
uint8_t unix = watch.Unix;
// Выводим данные в консоль
Serial.println(unix);
```

Библиотеки

- Библиотека для Arduino «iarduino_RTC».
- Как установить библиотеки для Arduino.
- Расширенные возможности библиотек для Arduino на шине I^2C.