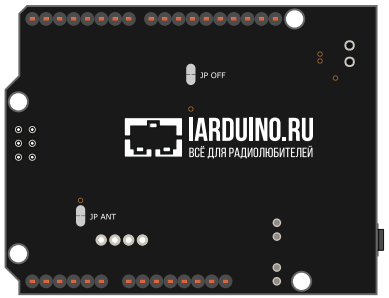
GSM/GPRS Shield A9: примеры подключения к сети интернет



Общие сведения

Плата расширения GSM/GPRS Shield A9 подарит вашему устройству на Arduino мобильную связь для осуществления входящих/исходящих голосовых звонков, отправки/получения SMS и приёма/передачи данных через мобильный интернет.

На основе GSM/GPRS Shield A9 можно создать радионяню с радиусом действия всего земного шара, сделать SMS-сигнализацию складского помещения или разработать систему мониторинга климатический условий в вашей загородной тепличке. Главное условие — SIM-карта с наличием мобильной связи.

Плата GSM/GPRS Shield работает с беспроводными стандартами связи 2G GSM/GPRS. К таким оператором в России относится популярная тройка: MTS, Megafon и Beeline. Для установки SIM-карты на плате расположен соответствующий слот под карту. Если вы хотите использовать SIM-карту другого оператора или находитесь в другой стране, уточните у своего провайдера поддержку стандарта GSM.

В рамках данной статьи рассмотрены примеры подключения GSM/GPRS Shield A9 к сети интернет. Всю остальную полезную информацию вы найдете в нашем общем руководстве по использованию GSM/GPRS Shield A9.

Подключение и настройка

Плата расширения GSM/GPRS Shield A9 предусмотрена для установки на платформы форм-фактора Arduino Shield R3. Для коммуникации с контроллером используются контакты интерфейса UART (TX и RX) с дополнительным пином управления PWR.

Для связи двух устройств в интерфейсе UART существует правило: линия TX подключается к выводу RX, а линия RX к выводу TX. В Arduino UART часто называют Serial. Подробнее про интерфейс UART читайте в нашей статье на вики.

В зависимости от управляющей платформы, интерфейс UART может располагаться на разных пинах. Выберите свой вариант подключения GSM/GPRS-модуля.

GSM/GPRS A9 к Arduino Leonardo

На Arduino Leonardo и других платформах с микроконтроллером ATmega32U4, данные по USB и аппаратный интерфейс UART не связаны между собой. Это даёт возможность подключить GSM/GPRS Shield A9 к аппаратному UART платформы на пинах `RX-0` и `TX-1` .

Что понадобится

- 1× Arduino Leonardo
- 1× GSM/GPRS Shield A9
- 1× Кабель micro-USB

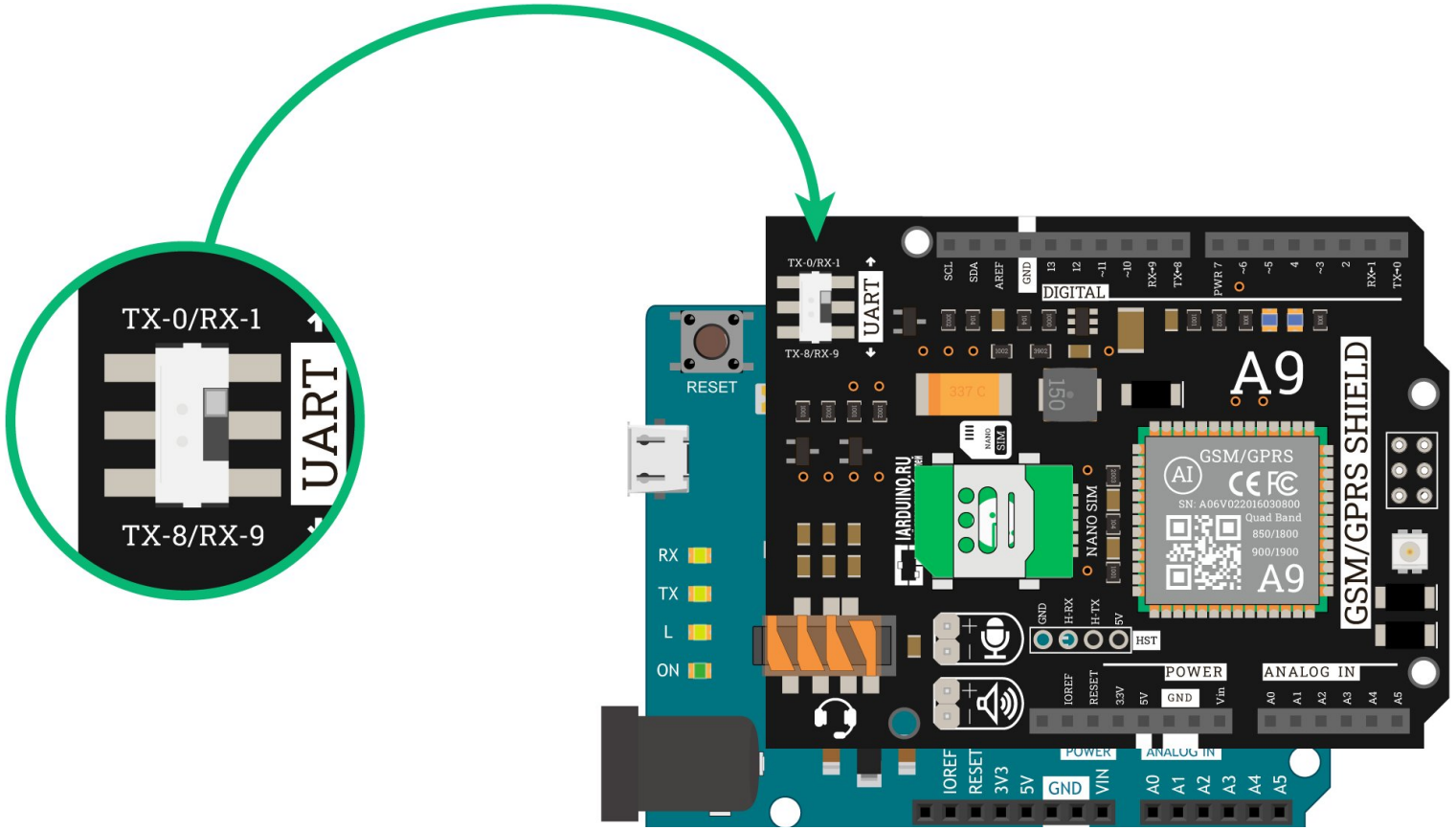
Таблица сигналов

Контакт GSM/GPRS Shield A9	Контакт Arduino Leonardo

TX-0	RX-0
RX-1	TX-1
PWR	7

Схема устройства

1. Вставьте SIM-карту в GSM/GPRS Shield A9.
2. Установите GSM/GPRS Shield A9 сверху на управляющую платформу Arduino Leonardo.
3. Установите переключатель шины UART в положение `TX-0/RX-1` .



Код инициализации

```
/*
*  Код инициализации GSM/GPRS Shield A9 с платами Arduino Leonardo
*  Подробности: https://wiki.iarduino.ru/page/gsm-gprs-shield-internet
*/

// Подключаем библиотеку GprsModem
#include <GprsModem.h>

// Назначаем GPIO пин включения GSM/GPRS-модуля
constexpr int PIN_PWR = 7;

// Создаём объект класса GprsModem
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
// и пин включения GSM/GPRS-модуля
GprsModem myModem(Serial1, PIN_PWR);

// Создаём объект класса GprsClient
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
GprsClient myClient(Serial1);

void setup() {
}

void loop() {
}
```

GSM/GPRS A9 к Arduino Uno

На Arduino Uno и других платформах с микроконтроллером ATmega328, данные по USB и аппаратный интерфейс UART связаны между

собой. Это не даёт возможность подключить GSM/GPRS Shield A9 к аппаратному UART платформы на пинах `RX-0` и `TX-1`. Выход есть — программный UART, который можно назначить на другие пины управляющей платы. В примере будем использовать программный UART на пинах `RX-8` и `TX-9`.

Что понадобится

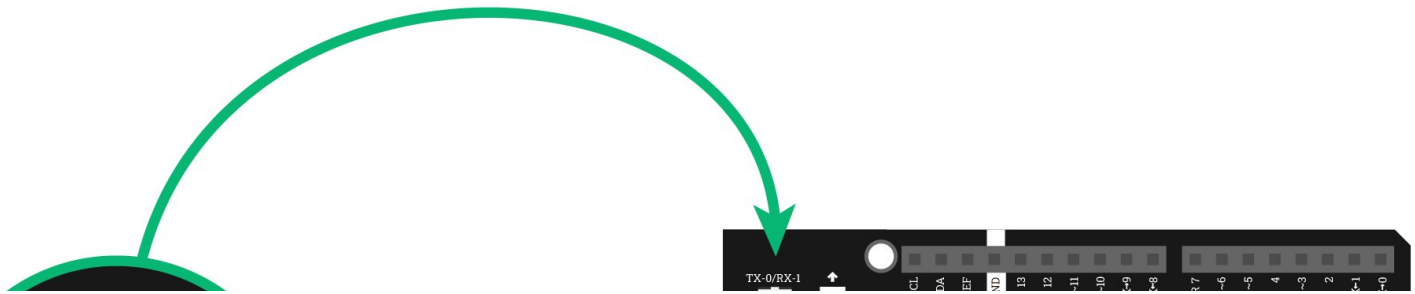
- 1× Arduino Uno
- 1× GSM/GPRS Shield A9
- 1× Кабель USB (A — B)

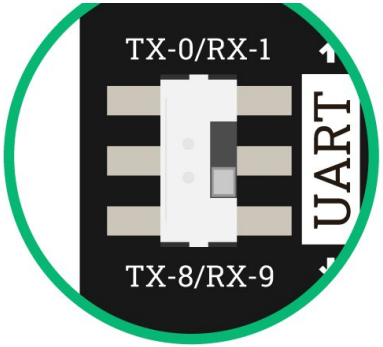
Таблица сигналов

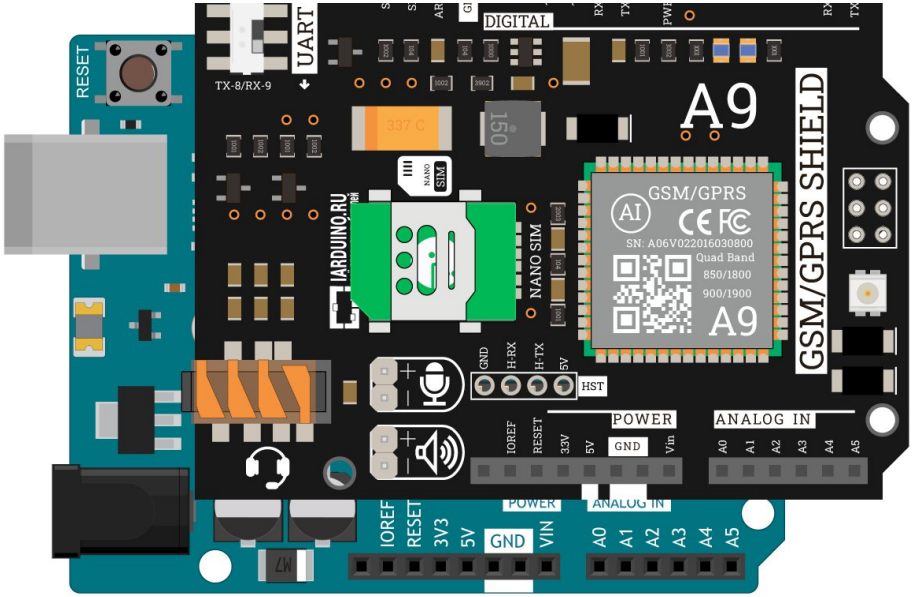
Контакт GSM/GPRS Shield A9	Контакт Arduino Uno
TX-8	RX-8
RX-9	TX-9
PWR	7

Схема устройства

1. Вставьте SIM-карту в GSM/GPRS Shield A9.
2. Установите GSM/GPRS Shield A9 сверху на управляющую платформу Arduino Uno.
3. Установите переключатель шины UART в положение `TX-8/RX-9`.







Код инициализации

```
/*
*  Код инициализации GSM/GPRS Shield A9 с платами Arduino Uno
*  Подробности: https://wiki.iarduino.ru/page/gsm-gprs-shield-internet
*/

// Подключаем библиотеку GprsModem
#include <GprsModem.h>
// Подключаем библиотеку SoftwareSerial
#include <SoftwareSerial.h>

// Назначаем GPIO пины для связи с GSM/GPRS Shield A9
// Пин включения GSM/GPRS-модуля
constexpr int PIN_PWR = 7;
// Пин приёма данных из контроллера в GSM/GPRS-модуль
constexpr int PIN_RX = 8;
```

```
// Пин передачи данных из контроллера в GSM/GPRS-модуль
constexpr int PIN_TX = 9;

// Создаём объект mySerial для работы с функциями библиотеки SoftwareSerial
// В параметрах указываем пины RX и TX
SoftwareSerial mySerial(PIN_RX, PIN_TX);

// Создаём объект класса GprsModem
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
// и пин включения GSM/GPRS-модуля
GprsModem myModem(mySerial, PIN_PWR);

// Создаём объект класса GprsClient
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
GprsClient myClient(mySerial);

void setup() {
}

void loop() {
}
```

GSM/GPRS A9 к Arduino Mega

На Arduino Mega и других платформах с микроконтроллером ATmega2560, данные по USB и аппаратный интерфейс UART связаны между собой. Это не даёт возможность подключить GSM/GPRS Shield A9 к аппаратному UART платформы на пинах `RX-0` и `TX-1`. Однако на платах форм-фактора Mega есть ещё дополнительно три аппаратных UART:

- UART1: RX-19 и TX-18
- UART2: RX-17 и TX-16
- UART3: RX-15 и TX-14

В примере будем использовать аппаратный UART3 на пинах `RX-15` и `TX-14` .

Что понадобится

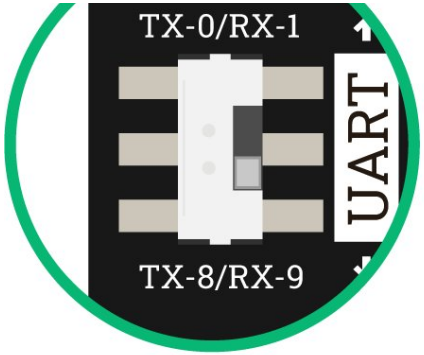
- 1× Arduino Mega 2560
- 1× GSM/GPRS Shield A9
- 1× Кабель USB (A — B)
- 1× Соединительные провода «папа-папа»

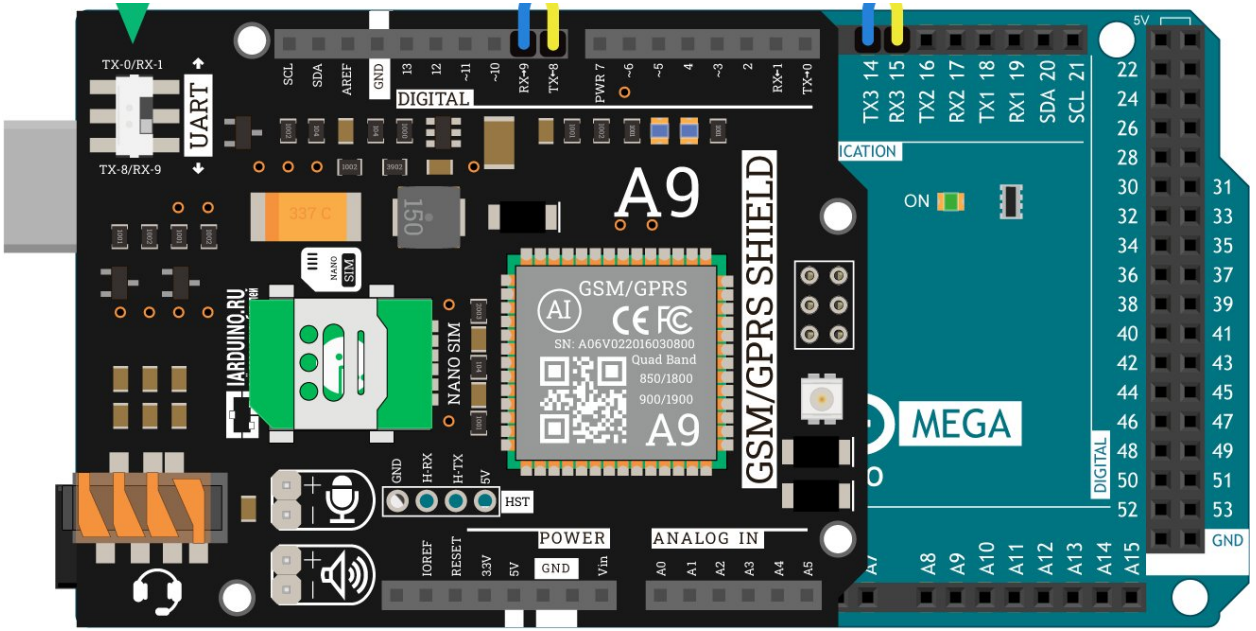
Таблица сигналов

Контакт GSM/GPRS Shield A9	Контакт Arduino Mega 2560
TX-8	RX-15
RX-9	TX-14
PWR	7

Схема устройства

1. Вставьте SIM-карту в GSM/GPRS Shield A9.
2. Установите GSM/GPRS Shield A9 сверху на управляющую платформу Arduino Mega 2560.
3. Установите переключатель шины UART в положение `TX-8/RX-9.`
4. Соедините контакты GSM/GPRS Shield A9 `TX-8` и `RX-9` с платой Arduino Mega с помощью проводов «папа-папа» согласно таблице сигналов.





Код инициализации

```
/*
*  Код инициализации GSM/GPRS Shield A9 с платами Arduino Mega
*  Подробности: https://wiki.iarduino.ru/page/gsm-gprs-shield-internet
*/

// Подключаем библиотеку GprsModem
#include <GprsModem.h>

// Назначаем GPIO пин включения GSM/GPRS-модуля
constexpr int PIN_PWR = 7;

// Создаём объект класса GprsModem
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
// и пин включения GSM/GPRS-модуля
```

```
GprsModem myModem(Serial3, PIN_PWR);

// Создаём объект класса GprsClient
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
GprsClient myClient(Serial3);

void setup() {
}

void loop() {
}
```

Примеры работы

Приведённые ниже примеры написаны для работы GSM/GPRS Shield A9 с контроллером Arduino Leonardo. Если у вас другая Arduino, например Arduino Uno или Arduino Mega, необходимо изменить инициализацию в начале кода программы. Все базовые варианты мы рассмотрели в разделе подключение и настройка.

Получение статуса модема

Для начала запросим уровень статуса модема.

```
/*
*  Код запроса статуса модема
*  для GSM/GPRS Shield A9 с контроллером Arduino Leonardo
*  Если у вас другой контроллер, необходимо изменить инициализацию в начале кода
*  Подробности: https://wiki.iarduino.ru/page/gsm-gprs-shield-internet
*/

// Подключаем библиотеку GprsModem
#include <GprsModem.h>

// Назначаем GPIO пин включения GSM/GPRS-модуля
```

```
constexpr int PIN_PWR = 7;

// Создаём объект класса GprsModem
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
// и пин включения GSM/GPRS-модуля
GprsModem myModem(Serial1, PIN_PWR);

void setup() {
  // Открываем консоль
  Serial.begin(9600);

  // Ждём готовность модема GSM/GPRS Shield к работе
  Serial.print("Инициализация модема, подождите пожалуйста...");
  while (myModem.status() != GPRS_OK) {
    Serial.print(".");
    myModem.begin();
  }
  Serial.println("Инициализация модема прошла успешно");

  // Получаем последний статус
  switch (myModem.status()) {
    case GPRS_OK: {
      Serial.println("Модуль готов к работе");
      break;
    }
    case GPRS_REG_NO: {
      Serial.println("Модем не зарегистрирован в сети оператора связи");
      break;
    }
    case GPRS_SPEED_ERR: {
      Serial.println("Не удалось согласовать скорость UART");
      break;
    }
```

```
    case GPRS_UNKNOWN: {
      Serial.println("Статус неизвестен");
      break;
    }
    case GPRS_SLEEP: {
      Serial.println("Режим ограниченной функциональности");
      break;
    }
    case GPRS_SIM_NO: {
      Serial.println("Нет SIM-карты");
      break;
    }
    case GPRS_SIM_FAULT: {
      Serial.println("SIM-карта неисправна");
      break;
    }
    case GPRS_SIM_ERR: {
      Serial.println("SIM-карта не прошла проверку");
      break;
    }
    case GPRS_REG_FAULT: {
      Serial.println("Оператор отклонил регистрацию модема");
      break;
    }
    case GPRS_SIM_PIN: {
      Serial.println("Требуется ввод PIN-кода");
      break;
    }
    case GPRS_SIM_PUK: {
      Serial.println("Требуется ввод PUK1");
      break;
    }
```

```
    default: {
      break;
    }
  }

  Serial.println("Обновляем статус модема...");
  switch (myModem.updateStatus()) {
    case GPRS_OK: {
      Serial.println("Модуль готов к работе");
      break;
    }
    case GPRS_REG_NO: {
      Serial.println("Модем не зарегистрирован в сети оператора связи");
      break;
    }
    case GPRS_SPEED_ERR: {
      Serial.println("Не удалось согласовать скорость UART");
      break;
    }
    case GPRS_UNKNOWN: {
      Serial.println("Статус неизвестен");
      break;
    }
    case GPRS_SLEEP: {
      Serial.println("Режим ограниченной функциональности");
      break;
    }
    case GPRS_SIM_NO: {
      Serial.println("Нет SIM-карты");
      break;
    }
    case GPRS_SIM_FAULT: {
      Serial.println("SIM-карта неисправна");
```

```
      break;
    }
    case GPRS_SIM_ERR: {
      Serial.println("SIM-карта не прошла проверку");
      break;
    }
    case GPRS_REG_FAULT: {
      Serial.println("Оператор отклонил регистрацию модема");
      break;
    }
    case GPRS_SIM_PIN: {
      Serial.println("Требуется ввод PIN-кода");
      break;
    }
    case GPRS_SIM_PUK: {
      Serial.println("Требуется ввод PUK1");
      break;
    }
    default: {
      break;
    }
  }
}

void loop() {
}
```

Получение уровня сигнала

В продолжении запросим уровень сигнала GSM сети.

```
/*
```

```
*  Код запроса уровня сигнала сети
*  для GSM/GPRS Shield A9 с контроллером Arduino Leonardo
*  Если у вас другой контроллер, необходимо изменить инициализацию в начале кода
*  Подробности: https://wiki.iarduino.ru/page/gsm-gprs-shield-internet
*/

// Подключаем библиотеку GprsModem
#include <GprsModem.h>

// Назначаем GPIO пин включения GSM/GPRS-модуля
constexpr int PIN_PWR = 7;

// Создаём объект класса GprsModem
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
// и пин включения GSM/GPRS-модуля
GprsModem myModem(Serial1, PIN_PWR);

void setup() {
  // Открываем консоль
  Serial.begin(9600);

  // Ждём готовность модема GSM/GPRS Shield к работе
  Serial.print("Инициализация модема, подождите пожалуйста...");
  while (myModem.status() != GPRS_OK) {
    Serial.print(".");
    myModem.begin();
  }
  Serial.println("Инициализация модема прошла успешно");

  // Получаем уровень сигнала
  int signalLevel = myModem.getSignalLevel();
  // Выводим уровень сигнала в последовательный порт
  Serial.print("Уровень сигнала: ");
```

```
  Serial.println(signalLevel);
}

void loop() {
}
```

WEB-клиент

В продолжении подключимся к серверу `www.google.com` и запросим данные поиска по ключевому слову `iarduino`. Все данные вы увидите в консоли.

```
/*
*  Код запроса данных с сервера
*  для GSM/GPRS Shield A9 с контроллером Arduino Leonardo
*  Если у вас другой контроллер, необходимо изменить инициализацию в начале кода
*  Подробности: https://wiki.iarduino.ru/page/gsm-gprs-shield-internet
*/

// Подключаем библиотеку GprsModem
#include <GprsModem.h>

// Назначаем GPIO пин включения GSM/GPRS-модуля
constexpr int PIN_PWR = 7;

// Создаём объект класса GprsModem
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
// и пин включения GSM/GPRS-модуля
GprsModem myModem(Serial1, PIN_PWR);

// Создаём объект класса GprsClient
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
GprsClient myClient(Serial1);
```

```
// Данные для получения информации от удалённого узла
char host[] = "www.google.com";
char req[] = "GET /search?q=iarduino HTTP/1.1";
int port = 80;

void setup() {
  // Открываем консоль
  Serial.begin(9600);

  // Ждём готовность модема GSM/GPRS Shield к работе
  Serial.print("Инициализация модема, подождите пожалуйста...");
  while (myModem.status() != GPRS_OK) {
    Serial.print(".");
    myModem.begin();
  }
  Serial.println("Инициализация модема прошла успешно");

  // Инициируем объект клиента
  myClient.begin();

  // Подключаемся к серверу
  Serial.println("Ждём подключения к серверу");
  if (!myClient.connect(host, port)) {
    Serial.println("Не удалось подключится к серверу");
  } else {
    // Делаем GET-запрос на сервер
    myClient.println((String)req);
    myClient.println((String)"Host: " + host);
    myClient.println("Connection: close");
    myClient.println();
  }
}
```

```
void loop() {
  // Получаем данные с сервера
  if (myClient.available() > 0) {
    // считываем данные и печатаем в Serial-порт
    Serial.write(myClient.read());
  }
}
```

Режим AT-команд

Методы библиотеки не всегда решают поставленные задачи. Пришло время вылететь из гнезда и пообщаться с модулем на низком уровне через AT-команды. Данный скетч предназначен для разработчиков.

```
/*
*  Код для общение с GSM/GPRS Shield A9 через AT-команды
*  для GSM/GPRS Shield A9 с контроллером Arduino Leonardo
*  Если у вас другой контроллер, необходимо изменить инициализацию в начале кода
*  Подробности: https://wiki.iarduino.ru/page/gsm-gprs-shield-internet
*/

// Подключаем библиотеку GprsModem
#include <GprsModem.h>

// Назначаем GPIO пин включения GSM/GPRS-модуля
constexpr int PIN_PWR = 7;

// Создаём объект класса GprsModem
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
// и пин включения GSM/GPRS-модуля
GprsModem myModem(Serial1, PIN_PWR);
```

```
// Создаём объект класса GprsClient
// В параметрах передаём объект Serial к которому подключён GSM/GPRS-модуль
GprsClient myClient(Serial1);

void setup() {
  // Открываем консоль
  Serial.begin(9600);

  // Ждём готовность модема GSM/GPRS Shield к работе
  Serial.print("Инициализация модема, подождите пожалуйста...");
  while (myModem.status() != GPRS_OK) {
    Serial.print(".");
    myModem.begin();
  }
  Serial.println("Инициализация модема прошла успешно");

  // Инициируем объект клиента
  myClient.begin();
}

void loop() {
  // Если пришли данные с консоли
  // Отправляем их в GSM/GPRS Shield
  if (Serial.available() > 0)
    myClient.write(Serial.read());

  // Если пришли данные с GSM/GPRS Shield
  // Отправляем их в консоль
  if (myClient.available() > 0)
    Serial.write(myClient.read());
}
```

Библиотека для Arduino

GSM/GPRS Shield общается с Arduino через интерфейс UART по протоколу AT-команд. Однако вы можете не вникать в детали программного управления, используйте готовые библиотеки для работы с модулем:

- Библиотека iarduino_GSM служит для работы со звонками и СМС. Подробности и описание методов програмного модуля `iarduino_GSM` читайте в отдельной инструкции по работе модулем со звонками и SMS.
- Библиотека iarduino_GprsClientA9 служит для работы с мобильным интернетом. Подробности и описание методов програмного модуля `iarduino_GprsClientA9` читайте ниже

Установка

Для старта скачайте и установите библиотеку iarduino_GprsClientA9. Для инсталляции рекомендуем использовать нашу инструкцию по установке библиотек для Arduino.

Подключение

- Назначение: подключение библиотеки.
- Синтаксис: `#include <GprsModem.h>`
- Примечания:
  - Библиотека подключается в самом начале программы.
  - Подключение библиотеки обязательное действие, иначе её методы работать не будут.
  - В библиотеках функции называются методами.
- Примеры:

```
// Подключаем библиотеку для работы с GSM/GPRS Shield
#include <GprsModem.h>
```

Классы

В библиотеке `iarduino_GprsClientA9` есть два класса:

- `GprsModem` : отвечает за работу с модемом.

- `GprsClient` : отвечает за работу с интернетом.

Каждый класс имеет свой конструктор и набор методов.

Класс GprsModem

Конструктор

- Назначение: создание объекта для работы с методами класса `GprsModem`
- Синтаксис:
  - `GprsModem myModem(HardwareSerial& Serial, uint8_t pinPWR)`
  - `GprsModem myModem(SoftwareSerial& Serial, uint8_t pinPWR)`
- Параметры:
  - `Serial` : объект или класс используемый для работы с интерфейсом UART.
  - `pinPWR` : пин включения модуля. Для GSM/GPRS Shield A9 указывайте пин `7` .
- Возвращаемое значение: нет
- Примечания:
  - Конструктор вызывается в самом начале программы.
  - Вызов конструктора обязателен, иначе методы библиотеки работать не будут.
  - Serial может быть аппаратный `HardwareSerial` или программный `SoftwareSerial` . Для программного Serial необходимо использовать библиотеку `SoftwareSerial` .
- Пример: все базовые варианты создания объекта мы рассмотрели в разделе подключение и настройка.

Метод begin()

- Назначение: инициализация работы модема GSM/GPRS Shield.
- Синтаксис: `bool begin()`
- Параметры: нет
- Возвращаемое значение:
  - `true` : инициализация модуля прошла успешно.
  - `false` : инициализация модуля прошла не успешно.

- Примечания:
  - Метод `begin()` достаточно вызывать один раз в функции `setup()` .
  - Вызов метода `begin()` обязателен, иначе методы библиотеки работать не будут.
- Пример:

```
// Инициализация модема GSM/GPRS Shield
myModem.begin();
```

Метод coldReboot()

- Назначение: перезагрузка модема GSM/GPRS Shield.
- Синтаксис: `void coldReboot()`
- Параметры: нет
- Возвращаемое значение: нет
- Пример:

```
// Инициализация модема GSM/GPRS Shield
myModem.coldReboot();
```

Метод getSignalLevel()

- Назначение: запрос уровня принимаемого сигнала.
- Синтаксис: `uint8_t getSignalLevel()`
- Параметры: нет
- Возвращаемое значение: число от `0` до `31` .
- Примечания:
  - `0` : уровень сигнала -113 дБм и ниже
  - `1–30` :уровень сигнала от -111 дБм до -53 дБм (шаг 2 дБм)
  - `31` : уровень сигнала -51 дБм и выше.

- Пример:

```
// Запрос уровня принимаемого сигнала
uint8_t signal = myModem.getSignalLevel();
Serial.println(signal);
```

Метод status()

- Назначение: получение последнего статуса модема.
- Синтаксис: `uint8_t status()`
- Параметры: нет
- Возвращаемое значение:
  - `GPRS_OK` : модуль готов к работе.
  - `GPRS_REG_NO` : модем не зарегистрирован в сети оператора связи.
  - `GPRS_SPEED_ERR` : не удалось согласовать скорость UART.
  - `GPRS_UNKNOWN` : статус неизвестен (AT-команды могут не выполнятся).
  - `GPRS_SLEEP` : модуль в режиме ограниченной функциональности.
  - `GPRS_SIM_NO` : нет SIM-карты.
  - `GPRS_SIM_FAULT` : SIM-карта неисправна.
  - `GPRS_SIM_ERR` : SIM-карта не прошла проверку.
  - `GPRS_REG_FAULT` : оператор отклонил регистрацию модема.
  - `GPRS_SIM_PIN` : требуется ввод PIN-кода.
  - `GPRS_SIM_PUK` : требуется ввод PUK-кода.

Метод updateStatus()

- Назначение: получение текущего статуса модема.
- Синтаксис: `uint8_t updateStatus()`
- Параметры: нет
- Возвращаемое значение:

- `GPRS_OK` : модуль готов к работе.
- `GPRS_REG_NO` : модем не зарегистрирован в сети оператора связи.
- `GPRS_SPEED_ERR` : не удалось согласовать скорость UART.
- `GPRS_UNKNOWN` : статус неизвестен (AT-команды могут не выполнятся).
- `GPRS_SLEEP` : модуль в режиме ограниченной функциональности.
- `GPRS_SIM_NO` : нет SIM-карты.
- `GPRS_SIM_FAULT` : SIM-карта неисправна.
- `GPRS_SIM_ERR` : SIM-карта не прошла проверку.
- `GPRS_REG_FAULT` : оператор отклонил регистрацию модема.
- `GPRS_SIM_PIN` : требуется ввод PIN-кода.
- `GPRS_SIM_PUK` : требуется ввод PUK-кода.

Класс GprsClient

Конструктор

- Назначение: создание объекта для работы с методами класса `GprsClient`
- Синтаксис:
  - `GprsClient myClient(HardwareSerial& Serial)`
  - `GprsClient myClient(SoftwareSerial& Serial)`
- Параметры:
  - `Serial` : объект или класс используемый для работы с интерфейсом UART.
- Возвращаемое значение: нет
- Примечания:
  - Конструктор вызывается в самом начале программы.
  - Вызов конструктора обязателен, иначе методы библиотеки работать не будут.
  - Serial может быть аппаратный `HardwareSerial` или программный `SoftwareSerial` . Для программного Serial необходимо использовать библиотеку `SoftwareSerial` .
  - Класс `GprsClient` является дочерним классом класса Stream и наследует все его публичные методы: `available` , `read` , `print` и т.д.

- Пример: все базовые варианты создания объекта мы рассмотрели в разделе подключение и настройка.

Метод begin()

- Назначение: перевод модуля в режим работы с GPRS.
- Синтаксис: `bool begin()`
- Параметры: нет
- Возвращаемое значение:
  - `true` : перевод модуля в режим работы с GPRS прошёл успешно.
  - `false` : перевод модуля в режим работы с GPRS прошёл не успешно.
- Примечания:
  - Метод `begin()` достаточно вызывать один раз в функции `setup()` .
  - Вызов метода `begin()` обязателен, иначе методы библиотеки работать не будут.
- Пример:

```
// Переводим модуль в режим работы с GPRS
myClient.begin();
```

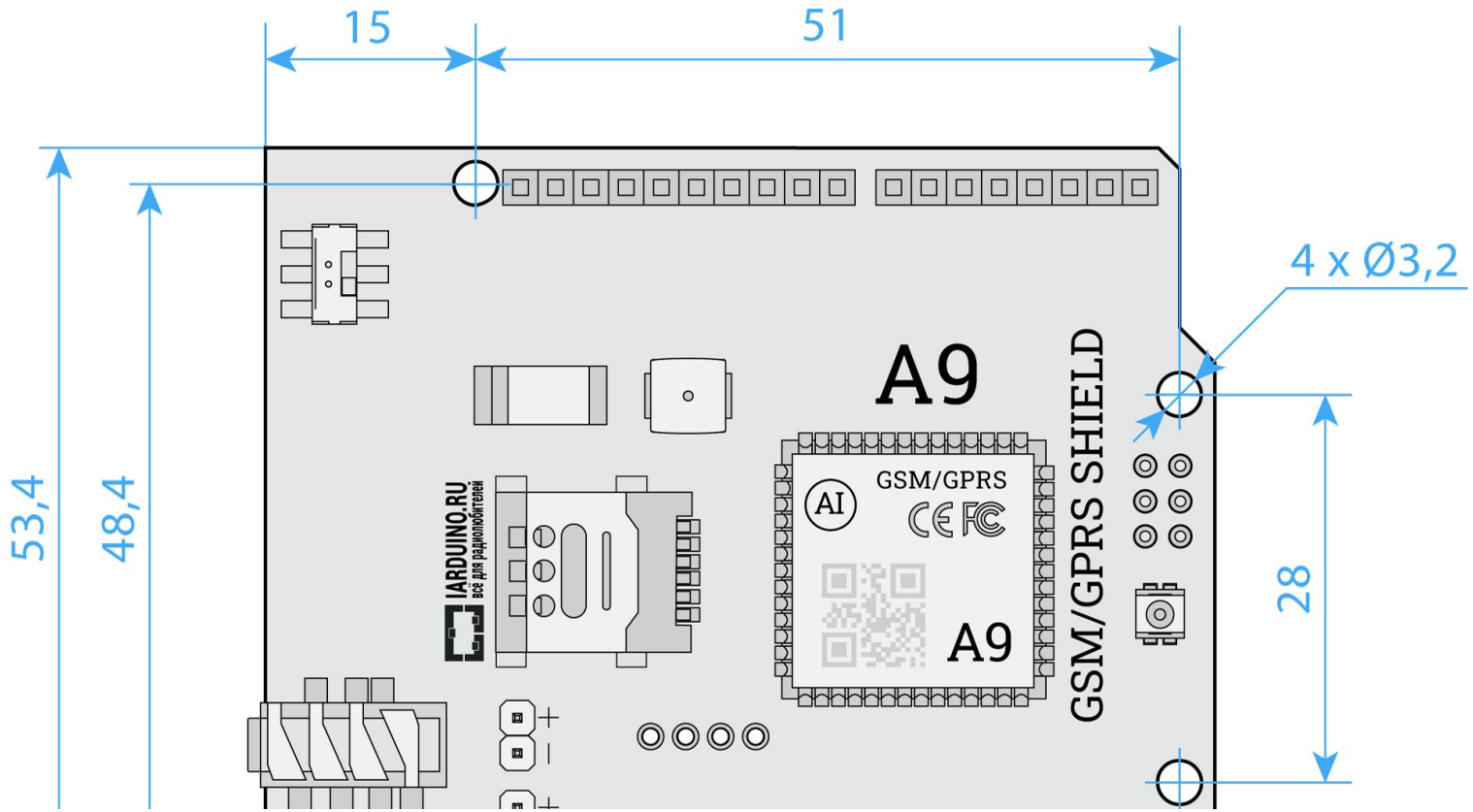
Метод connect()

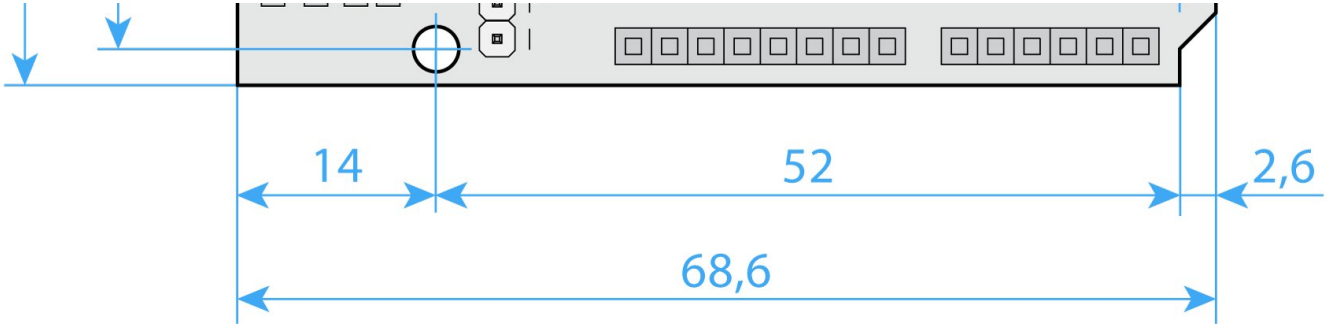
- Назначение: перевод модуля в режим работы с GPRS.
- Синтаксис:
  - `int connect(IPAddress ip, uint16_t port)`
  - `int connect(const char* host, uint16_t port)`
  - `int connect(const char* host, uint16_t port, const char* protocol)`
- Параметры:
  - `ip` : IP-адрес сервера.
  - `host` : имя сервера.
  - `port` : порт подключения.
  - `protocol` : протокол подключения.

- Возвращаемое значение:
  - `1` : успешное подключение.
  - `-1` : таймаут, превышен интервал ожидания.
  - `-2` : сервера не существует.
- Пример:

```
// Подключаемся к серверу с именем www.google.com
// и портом 80
myClient.connect("www.google.com", 80);
```

Габаритный чертёж





Характеристики

- Модель: GSM/GPRS Shield A9
- Стандарт связи: GSM/GPRS
- Возможности: голосовая связь, SMS-сообщения, приём и передача данных.
- Поддержка частот: 850/900/1800/1900 МГц
- Класс GPRS: 10
  - Максимальная скорость загрузки: 85,6 Кбит/сек
  - Максимальная скорость отдачи: 42,8 Кбит/сек
- Слот для SIM-карты: nano SIM (4FF)
- Антенна:
  - Встроенная: разведена на плате
  - Внешняя: подключается через разъём IPX UFL
- Совместимость: контроллеры форм-фактора Arduino R3
- Программный интерфейс: UART с дополнительными пином управления
- Программный протокол: AT-команды
- Входное напряжение питания: 7 – 12 В
- Потребляемый ток:
  - В спящем режиме: до 3 мА
  - В режиме ожидания: до 100 мА
  - В активном режиме (соединение, разговор, SMS): до 500 мА
  - Поиск сети: до 2 А

- Логическое напряжение уровней: 3,3–5 В
- Размеры: Arduino Shield R3