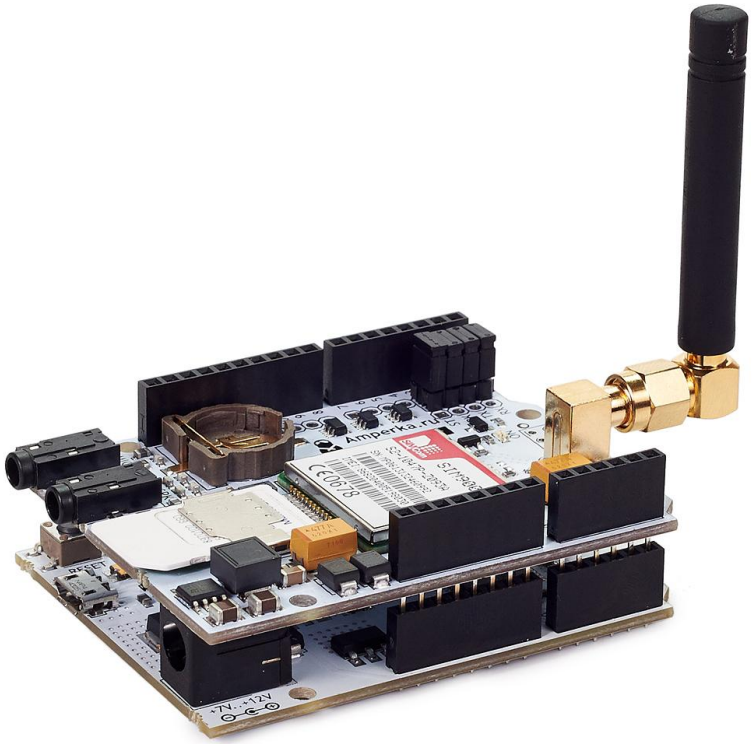
GPRS Shield

GPRS Shield — это плата расширения, позволяющая Arduino работать в сетях сотовой связи по технологиям GSM/GPRS для приёма и передачи данных, SMS и голосовой связи. С ней вы сможете управлять удалёнными объектами с помощью мобильного телефона.

Подключение и настройка

1. Установите SIM-карту на GPRS Shield, а GPRS Shield — на управляющую платформу, например Arduino или Iskra Neo.
2. Убедитесь в наличии и правильности соединения джамперов на плате GPRS устройства:
    - `RX` и `TX` с `0` и `1` пином соответственно;
    - `PK` с `2` пином;
    - `ST` с `3` пином.

3. Подключите внешнюю антенну через SMA-разъём.



4. Скачайте и установите библиотеку , для управления GPRS-устройством.

Примеры работы GPRS Shield

Теперь рассмотрим несколько простых примеров работы GPRS-устройства.

Исходящие звонки

Чтобы позвонить с GPRS Shield’a, воспользуйтесь скетчем, приведенным ниже. Если открыть монитор последовательного порта, можно проследить порядок действий.

GPRS_OutgoingCall.ino

```
// библиотека для работы с GPRS устройством
#include <GPRS_Shield_Arduino.h>

// библиотека для эмуляции Serial порта
// она нужна для работы библиотеки GPRS_Shield_Arduino
#include <SoftwareSerial.h>

// номер на который будем звонить
#define PHONE_NUMBER  "+79263995140"

// создаём объект класса GPRS. По умолчанию скорость общения с ним 9600
бод
// с помощью него будем давать команды GPRS шилду
GPRS gprs;

void setup()
{
  // включаем GPRS шилд
```

```
  gprs.powerUpDown();
  // открываем последовательный порт для мониторинга действий в
программе
  Serial.begin(9600);
  while (!Serial) {
    // ждём, пока не откроется монитор последовательного порта
    // для того, чтобы отследить все события в программе
  }
  // проверяем есть ли связь с GPRS устройством
  while (!gprs.init()) {
    // если связи нет, ждём 1 секунду
    // и выводим сообщение об ошибке
    // процесс повторяется в цикле
    // пока не появится ответ от GPRS устройства
    delay(1000);
    Serial.print("Init error\r\n");
  }
  // вывод об удачной инициализации GPRS Shield
  Serial.println("GPRS init success");
  // сообщаем о наборе номера
  Serial.print("Start to call ");
  Serial.print(PHONE_NUMBER);
  // звоним по указанному номеру
  gprs.callUp(PHONE_NUMBER);
}

void loop()
{
}
```

Входящие звонки

Произведём входящий звонок на GPRS Shield. Не забудьте подключить наушники и микрофон в соответствующие разъёмы на плате, чтобы вы смогли слышать абонента, а он вас.

GPRS_IncomingCall.ino

```
// библиотека для работы с GPRS устройством
#include <GPRS_Shield_Arduino.h>

// библиотека для эмуляции Serial порта
// она нужна для работы библиотеки GPRS_Shield_Arduino
#include <SoftwareSerial.h>

// создаём объект класса GPRS. По умолчанию скорость общения с ним 9600
бод
// с помощью него будем давать команды GPRS шилду
GPRS gprs;

void setup()
{
  //настраиваем пин №13 в режим выхода,
  pinMode(13, OUTPUT);
  // подаём на пин 13 «низкий уровень»
  digitalWrite(13, LOW);
  // включаем GPRS шилд
  gprs.powerUpDown();
  // открываем последовательный порт для мониторинга действий в
программе
  Serial.begin(9600);
  while (!Serial) {
    // ждём, пока не откроется монитор последовательного порта
```

```
    // для того, чтобы отследить все события в программе
  }
  // проверяем есть ли связь с GPRS устройством
  while (!gprs.init()) {
    // если связи нет, ждём 1 секунду
    // и выводим сообщение об ошибке
    // процесс повторяется в цикле,
    // пока не появится ответ от GPRS устройства
    delay(1000);
    Serial.print("Init error\r\n");
  }
  // вывод об удачной инициализации GPRS Shield
  Serial.println("GPRS init success");
  // сообщаем об ожидании звонка
  Serial.println("Wait to call ");
}

void loop()
{
  // ожидаем звонка
  if (gprs.ifcallNow()) {
    // если поступает входящий звонок,
    // подаём на пин 13 «высокий уровень», чтобы
    // зажечь встроенный на Iskra светодиод
    digitalWrite(13, HIGH);
    // выводим сообщение о входящем вызове
    Serial.println("Incoming call");
    // по истечении 5 секунд берём трубку
    delay(5000);
    gprs.answer();
    // выводим сообщение о начале разговора
    delay(1000);
    Serial.println("Call a conversation");
    while (!gprs.ifcallEnd()) {
      // ждём пока месть абонент не положит трубку
    }
    // выводим сообщение о конце разговора
    Serial.println("Call over");
    // гасим светодиод
    digitalWrite(13, LOW);
  }
}
```

Отправка SMS

Помимо приёма и совершения звонков модуль может принимать и отправлять короткие текстовые сообщения — SMS. Ниже представлен пример отправки текстового сообщения.

GPRS_SendSMS.ino

```
// библиотека для работы с GPRS устройством
#include <GPRS_Shield_Arduino.h>

// библиотека для эмуляции Serial порта
// она нужна для работы библиотеки GPRS_Shield_Arduino
#include <SoftwareSerial.h>

// номер на который будем отправлять сообщение
#define PHONE_NUMBER  "+79263995140"
// текст сообщения, которое будем отправлять
#define MESSAGE  "Hello from GPRS Shield"
```

```
// создаём объект класса GPRS. По умолчанию скорость общения с ним 9600
бод
// с помощью него будем давать команды GPRS шилду
GPRS gprs;

void setup()
{
  // включаем GPRS-шилд
  gprs.powerUpDown();
  // открываем последовательный порт для мониторинга действий в
программе
  Serial.begin(9600);
  while (!Serial) {
    // ждём пока не откроется монитор последовательного порта
    // для того, чтобы отследить все события в программе
  }
  // проверяем, есть ли связь с GPRS-устройством
  while (!gprs.init()) {
    // если связи нет, ждём 1 секунду
    // и выводим сообщение об ошибке;
    // процесс повторяется в цикле,
    // пока не появится ответ от GPRS-устройства
    delay(1000);
    Serial.print("Init error\r\n");
  }
  // вывод об удачной инициализации GPRS Shield
  Serial.println("GPRS init success");
  // сообщаем о написании и отправке СМС по указанному номеру
  Serial.println("Start to send message ...");
  // отправляем сообщение по указанному номеру с заданным текстом
  gprs.sendSMS(PHONE_NUMBER, MESSAGE);
}

void loop()
{
}
```

Чтение SMS

Прочитаем сообщения, которые будут поступать на сим-карту, установленную в модуль. Вся информация будет выводиться в монитор Serial порта.

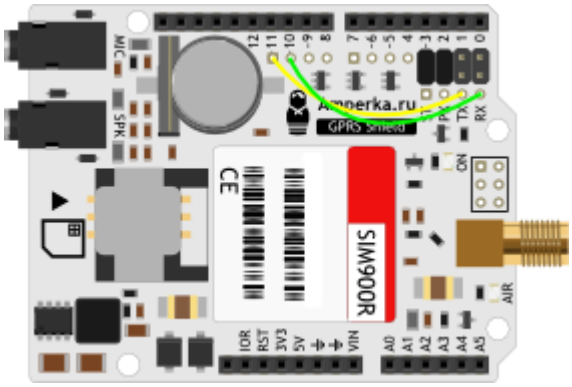
GPRS_ReadSMS.ino

```
// библиотека для работы с GPRS устройством
#include <GPRS_Shield_Arduino.h>

// библиотека для эмуляции Serial-порта
// она нужна для работы библиотеки GPRS_Shield_Arduino
#include <SoftwareSerial.h>

// длина сообщения
#define MESSAGE_LENGTH 160

// номер сообщения в памяти сим-карты
int messageIndex = 0;

// текст сообщения
char message[MESSAGE_LENGTH];
// номер, с которого пришло сообщение
char phone[16];
// дата отправки сообщения
char datetime[24];
```

```
// создаём объект класса GPRS. По умолчанию скорость общения с ним 9600
бод
// с помощью него будем давать команды GPRS шилду
GPRS gprs;

void setup()
{
  // включаем GPRS-шилд
  gprs.powerUpDown();
  // открываем последовательный порт для мониторинга действий в
программе
  Serial.begin(9600);
  while (!Serial) {
    // ждём пока не откроется монитор последовательного порта
    // для того, чтобы отследить все события в программе
  }
  // проверяем, есть ли связь с GPRS-устройством
  while (!gprs.init()) {
    // если связи нет, ждём 1 секунду
    // и выводим сообщение об ошибке;
    // процесс повторяется в цикле,
    // пока не появится ответ от GPRS-устройства
    delay(1000);
    Serial.print("Init error\r\n");
  }
  // выводим сообщение об удачной инициализации GPRS Shield
  Serial.println("GPRS init success");
  Serial.println("Please send SMS message to me!");
}

void loop()
{
  // проверяем наличие непрочитанных сообщений
  // и находим их номер в памяти сим-карты
  messageIndex = gprs.isSMSunread();
  if (messageIndex > 0) {
    // если есть хотя бы одно непрочитанное сообщение,
    // читаем его
    gprs.readSMS(messageIndex, message, MESSAGE_LENGTH, phone,
datetime);

    // Удаляем прочитанное сообщение из памяти Сим-карты
    gprs.deleteSMS(messageIndex);

    // выводим номер, с которого пришло смс
    Serial.print("From number: ");
    Serial.println(phone);

    // выводим дату, когда пришло смс
    Serial.print("Datetime: ");
    Serial.println(datetime);

    // выводим текст сообщения
    Serial.print("Recieved Message: ");
    Serial.println(message);
  }
}
```

Работа через SoftwareSerial

На платах Arduino, прошивка которых проходит через пины `0` и `1` (Arduinio Uno, Arduino Mega 2560, Arduino ADK и Iskra Mini), прошивку микроконтроллера необходимо производить со снятыми джамперами `TX` и `RX`. Если же вы вам необходима одновременная работа GPRS-шилда и коммуникация с компьютером снимите джампера с `TX` и `RX` и припаяйте провода между луженым отверстием рядом со снятым джампером и луженым



отверстием рядом с нужным пином. На этой картинке мы перекинули управляющие пины `TX` и `RX` с `0` и `1` пина Arduino — на `10` и `11` пин соответственно.

Пример исходящего звонка через SoftwareSerial

GPRS_OutgoingCall_Software.ino

```
// библиотека для работы с GPRS устройством
#include <GPRS_Shield_Arduino.h>

// библиотека для эмуляции Serial порта
// она нужна для работы библиотеки GPRS_Shield_Arduino
#include <SoftwareSerial.h>

// номер на который будем звонить
#define PHONE_NUMBER  "+79263995140"

// создаём объект класса GPRS
// и передаём ему номера пинов ST, PK, RX и TX.
// с помощью него будем давать команды GPRS шилду
GPRS gprs(2, 3, 11, 10);

void setup()
{
  // включаем GPRS шилд
  gprs.powerUpDown();
  // открываем последовательный порт для мониторинга действий в
программе
  Serial.begin(9600);
  while (!Serial) {
    // ждём, пока не откроется монитор последовательного порта
    // для того, чтобы отследить все события в программе
  }
  // проверяем есть ли связь с GPRS устройством
  while (!gprs.init()) {
    // если связи нет, ждём 1 секунду
    // и выводим сообщение об ошибке
    // процесс повторяется в цикле
    // пока не появится ответ от GPRS устройства
```

```
    delay(1000);
    Serial.print("Init error\r\n");
  }
  // вывод об удачной инициализации GPRS Shield
  Serial.println("GPRS init success");
  // сообщаем о наборе номера
  Serial.print("Start to call ");
  Serial.print(PHONE_NUMBER);
  // звоним по указанному номеру
  gprs.callUp(PHONE_NUMBER);
}

void loop()
{
}
```

Работа с AT-командами

Данный раздел рассказывает о том, как работать с GPRS Shield на более низком уровне, без дополнительных библиотек. Если вам достаточно тех методов, которые предоставляет штатная библиотека, можете пропустить этот раздел.

Введение

С внешним миром модуль общается посредством AT-команд. Все команды делятся на базовые, так называемые S-команды, и расширенные, добавленные в стандартах GSM07.05–07.07. Практически все команды работают в 3 режимах — тестовом, чтения и записи.

- В тестовом режиме возвращается `OK`, если команда поддерживается или возможные значения данных в параметре команды. Тестовый режим определяется окончанием команды в виде `=?`
- В режиме чтения возвращаются текущие значения параметра, отличается от тестового наличием в конце просто символа `?`
- В режиме записи после `=` идут новые значения параметров.

Настройки порта

По умолчанию модуль настроен на 9600 8N1:

- 9600 – скорость;
- 8 – бит в посылки;
- N – нет контроля чётности;
- 1 – стоп бит.

Для проверки поддерживаются AT-команды:

Команда	Ответ	Описание
AT+IPR?	+IPR: 0 OK	**Скорость порта:** 0 – автоматически 1200 2400 4800

Команда	Ответ	Описание
		9600 19200 38400 57600 115200
AT+ICF?	+ICF: 3,3 OK	**Настройки передачи.** Первый параметр: Бит в посылке чётность/стоп бит 1 – 8/0/2 2 – 8/1/1 3 – 8/0/1 4 – 7/0/2 5 – 7/1/1 6 – 7/0/1 Второй параметр – чётность: 0 – нечётный 1 – чётный 3 – нет
AT+IFC?	+IFC: 0,0 OK	**Контроль передачи данных.** Первый параметр – терминалом от модуля Второй параметр – модулем от терминала 0 – нет контроля 1 – программный 2 – аппаратный

Если вы хотите изменить их, введите AT-команду, замените знак ? на = и введите нужные вам параметры из таблицы. Все настройки этих команд сохраняются в энергонезависимой памяти.

Информация о модуле и состояние

Команда	Ответ	Описание
AT+GCAP	+GCAP:+FCLASS,+CGSM OK	Возможности модуля
AT+GMM	SIMCOM_SIM900 OK	Идентификатор модуля
AT+GMR	Revision:1137B09SIM900M64_ST OK	Ревизия
AT+GSN	01322600XXXXXXX OK	IMEI

В документе с перечнем AT-команд можно найти документацию на все поддерживаемые команды.

Пример скетча для работы с использованием AT-команд

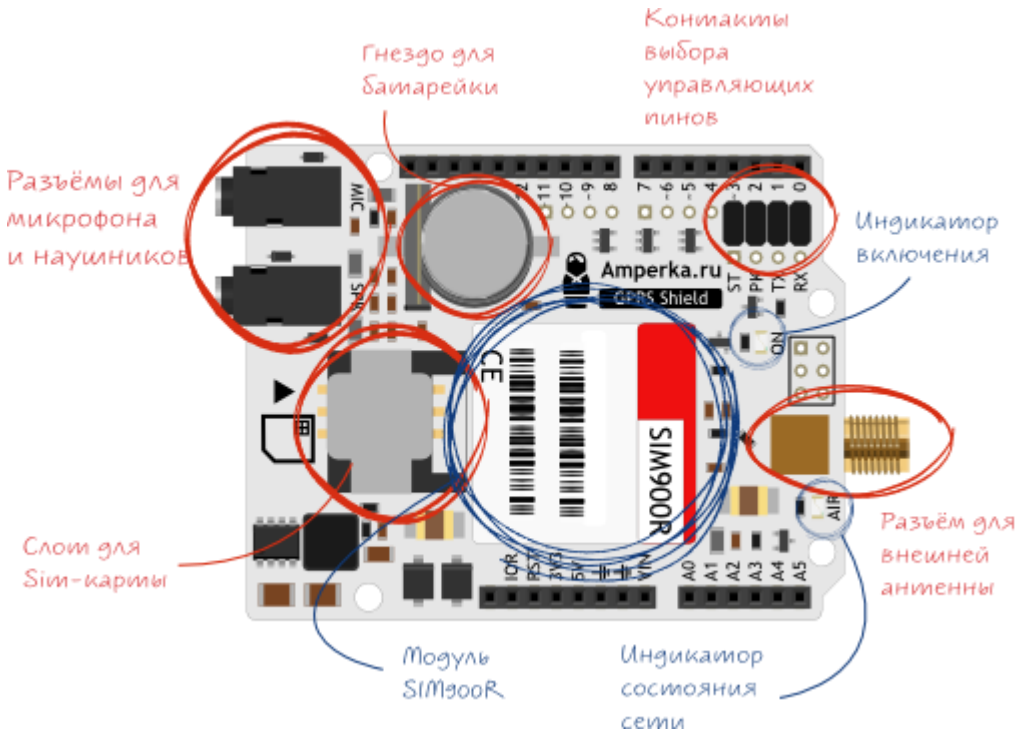
GPRS_AT_Commands.ino

```
// буфер PC Serial
char bufferPC_Serial[64];
// буфер GPRS Serial
char bufferGPRS_Serial[64];

int i = 0;
int j = 0;

void setup()
{
  // включаем GPRS-шилд
  gprs_OnOff();

  // открываем последовательный порт для мониторинга действий в
программе
  Serial.begin(9600);

  // открываем последовательный порт
  // для связи с GPRS-устройством со скоростью 9600 бод
  // (Пример для Arduino Leonardo. Если у вас Arduino Uno, используйте
SoftwareSerial)
  Serial1.begin(9600);

  while (!Serial) {
    // ждём, пока не откроется монитор последовательного порта
    // для того, чтобы отследить все события в программе
  }

  // пока не установится связь с GPRS-устройством, будем крутиться в
теле функции
  gprsTest();
}

void loop()
{
  // считываем данные с компьютера и записываем их в GPRS Shield
  serialPCread();
  // считываем данные с GPRS Shield и выводим их в Serial-порт
  serialGPRSread();
}

void serialPCread()
{
  i = 0;
  // если появились данные с компьютера
  if (Serial.available() > 0) {
    while (Serial.available() > 0) {
      // пока идёт передача данных,
      // записываем каждый байт в символьный массив
      bufferPC_Serial[i++]=(Serial.read());
    }
    // добавляем символ конца строки
    bufferPC_Serial[i] = '\0';
    // записываем данные в GPRS Shield
    Serial1.println(bufferPC_Serial);
    Serial.println("");
    // очищаем буфер PC Serial
    clearBufferPC_Serial();
  }
}

void serialGPRSread()
{
    j = 0;
```

```
    // если появились данные с GPRS Shield
    if (Serial1.available() > 0) {
      while (Serial1.available() > 0) {
        // пока идёт передача данных,
        // записываем каждый байт в символьный массив
        bufferGPRS_Serial[j++]=(Serial1.read());
      }
      // добавляем символ конца строки
      bufferGPRS_Serial[j] = '\0';
      // выводим полученные данные с GPRS Shield в Serial-порт
      Serial.write(bufferGPRS_Serial);
      // очищаем буфер GPRS Serial
      clearBufferGPRS_Serial();
    }
}

void clearBufferPC_Serial()
{
  for (int t = 0; t < i; t++) {
    // очищаем буфер,
    // присваивая всем индексам массива значение 0
    bufferPC_Serial[t] = 0;
  }
}

void clearBufferGPRS_Serial()
{
  for (int t = 0; t < j; t++) {
    // очищаем буфер,
    // присваивая всем индексам массива значение 0
    bufferGPRS_Serial[t] = 0;
  }
}

void gprsTest()
{
  // бесконечный цикл
  while (1) {
    // ждём 1 секунду
    delay(1000);
    j = 0;
    // посылаем в GPRS Shield АТ-команду "AT"
    Serial1.println("AT");
    // если появились данные с GPRS Shield
    if (Serial1.available() > 0) {
      while (Serial1.available() > 0) {
        // пока идёт передача данных,
        // записываем каждый байт в символьный массив
        bufferGPRS_Serial[j++] = Serial1.read();
      }
      // добавляем символ конца строки
      bufferGPRS_Serial[j] = '\0';
      // посылаем АТ-команду "AT"; если GPRS Shield исправен,
      // он должен вернуть команду "AT";
      // сравниваем всё что находиться в буфере GPRS Shield
      // со строкой "AT\r\n\r\nOK\r\n"
      if (strcmp(bufferGPRS_Serial, "AT\r\n\r\nOK\r\n") == 0) {
        // если всё верно выводим в Serial порт "State OK"
        // и выходим из бесконечного цикла
        Serial.println("State OK");
        break;
      } else {
        // если строки разные, значит произошла ошибка
        // выводим сообщение об ошибке и продолжаем цикл
```

```
        Serial.println("Init ERROR");
      }
    }
    // очищаем буфер GPRS Serial
    clearBufferGPRS_Serial();
  }
}

void gprs_OnOff()
{
  // настраиваем пин №2 в режим выхода
  pinMode(2, OUTPUT);
  // проверяем состояние 3 пина
  if (digitalRead(3) != HIGH) {
    // если на нём «низкий уровень»
    // подаём на пин 2 «высокий уровень»
    digitalWrite(2, HIGH);
    // ждём 3 секунды
    delay(3000);
  }
  // подаём на пин 2 «низкий уровень»
  digitalWrite(2, LOW);
}
```

Элементы платы



Модуль SIM900R

SIM900R — представитель GSM/GPRS-модулей компании SIMCom. Это — сердце GPRS Shield.

Разъём микрофона и наушников

Если подключить наушники и микрофон в соответствующие разъёмы, во время звонка с абонентом можно будет общаться точно так же как и по обычному телефону.

Слот для SIM-карты

Слот для подключения стандартной сим-карты — Mini Sim. Если у вас сим-карта другого размера, Micro Sim или Nano Sim, воспользуйтесь специальными переходниками.

Контакты выбора управляющих пинов

Контакты GPRS Shield	Контакты Arduino	Использование
RX и TX	0 и 1	Используются для выбора управляющих пинов с микроконтроллером.
PK	2	Используется для включения модуля. Для этого на него необходимо подать высокий уровень на 3 секунды, а затем подать низкий уровень на эту же ножку.
ST	3	Информационный пин о состоянии включения GPRS Shield. Если высокий уровень — шилд включён, если низкий — выключен.

Если в вашем проекте какие-нибудь из этих пинов уже заняты другим устройством, вы можете использовать любой другой свободный цифровой пин. Для этого необходимо снять джампер напротив занятого пина и припаять проводок между луженым отверстием рядом со снятым джампером и таким же отверстием рядом с нужным пином.

Гнездо для батарейки

Гнездо для батарейки CR1225, обеспечивающей работу встроенных часов реального времени. Нужна только при использовании команд, связанных с часами.

Разъём для внешней антенны

SMA-разъём для подключения внешней антенны. Без неё сигнал будет очень слабым.

Индикатор состояния сети

У модуля есть два информационных светодиода — `ON`, который загорается после включения модуля и `AIR`, который мигает в зависимости от состояния сети.

Возможные режимы `AIR` (Горит/Не горит):

- 64мс/800мс — сеть не найдена;
- 64мс/3000мс — сеть найдена;
- 64мс/300мс — идет обмен по GPRS.

Характеристики

- Поддержка частот 850/1800 МГц
- Класс передачи данных GPRS multi-slot class 10/8
- Соответствие стандарту GSM фазы 2/2+
- Поддержка TCP/UDP протоколов
- Полное управление при помощи AT-команд:

    - Стандартный набор – GSM 07.07 & 07.05
    - Расширенный набор – SIMCOM AT
- Возможность подключения аудио гарнитуры и микрофона
- Интерфейс UART 1200–115200 (9600 по умолчанию) бит/с
- Разъём для сим-карты
- Низкое энергопотребление: 1.5 мА (в спящем режиме)
- Выполнен в форм-факторе Arduino

Ссылки

- Библиотека SIM900 для удобства работы с платой
- Пример использования: SMS-розетка
- мануал для изучения AT команд