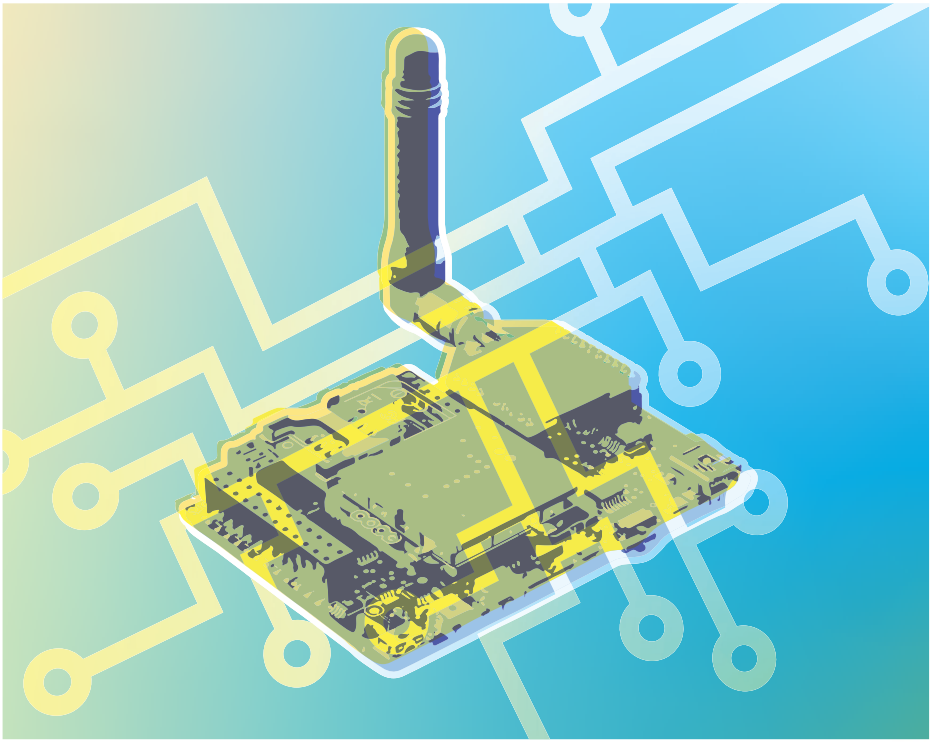
А. С. Сомов

ЛАБОРАТОРНАЯ РАБОТА «Сбор и визуализация данных с помощью платформы интернета вещей Libelium Waspmote»



Учебно-методическое пособие

ENTER IOT WIRELESS National Technology Initiative

Сомов А.С. Лабораторная работа «Сбор и визуализация данных с помощью платформы интернета вещей Libelium Waspmote» – М: Сколковский институт науки и технологий, 2019. – 30 с.

УДК 004.7

В учебно-методическом пособии приводится пример лабораторной работы по сбору и визуализации данных с применением платформы Libelium Waspmote и сервиса ThingSpeak. В качестве учебного примера использованы данные, получаемые с датчиков температуры в составе платы газоанализатора для платформы Libelium Waspmote. Указанная задача является типичным примером применения систем «интернета вещей». Учебно-методическое пособие может быть использовано как преподавателями, так и обучающимися средних общеобразовательных школ для проведения лабораторных практикумов.

Центр компетенций НТИ по направлению «Технологии беспроводной связи и интернета вещей» основан в июне 2018 года на базе Сколковского института науки и технологий для содействия российским коммерческим и государственным компаниям в преодолении технологических барьеров и создании конкурентоспособных продуктов и услуг для мирового рынка в области технологий интернета вещей и беспроводной связи, прежде всего сотовой связи следующих поколений (5G и 6G), индустриального интернета вещей (IIoT) и обработки промышленных данных.

Содержание:

Введение

Развитие технологий «интернета вещей» (Internet of Things, IoT) и возможностей их применение в различных отраслях делает актуальным вопрос изучения отдельных их составляющих, в том числе – практического применения датчиков, беспроводных технологий, облачных сервисов и других.

Данное учебно-методическое пособие приводит пример лабораторной работы, включающей типичные IoT задачи: получение и визуализации данных с применением платформы Waspmote и сервиса ThingSpeak.

В качестве учебного примера рассматривается сбор и передача данных с датчика температуры в составе платы газоанализатора для платформы Waspmote в облачный сервис ThingSpeak, позволяющий осуществить дальнейшую визуализацию полученных данных.

Указанная лабораторная работа может быть адаптирована также для других датчиков, совместимых с платформой Waspmote.

1. Информация об оборудовании и программном обеспечении для проведения лабораторной работы

1.1. Описание платформы Waspmote

Платформа Waspmote разработана компанией Libelium и позволяет проводить различные типы лабораторных работ, в т.ч. для измерения параметров окружающей среды.

Основные характеристики платформы включают [1]:

- Ультранизкое энергопотребление (7 μA);
- Наличие более 120 датчиков, которые могут быть подключены к платформе;
- Поддержку 16 беспроводных технологий: 4G / 3G / GPRS / GPRS+GPS / LoRaWAN / LoRa / Sigfox / 868 МГц / 900 МГц / ZigBee / 802.15.4 / DigiMesh / WiFi / RFID/NFC / BT / BLE;
- Возможности беспроводного программирования (OTA);
- Поддержку промышленных протоколов: RS-232, RS-485, Modbus, CANBus.

Рисунок 1 – Платформа Waspmote

Технические характеристики платформы Waspmote [1]:

Параметр	Характеристика
Микроконтроллер	ATmega1281
Частота	14 МГц
ОЗУ (SRAM)	8 Кбайт
ПЗУ (FLASH)	128 Кбайт
Слот для карт памяти	MicroSD
Вес	20 гр
Размеры	73.5 x 51 x 13 мм

Более подробное описание технических характеристик платформы Waspmote представлено по ссылке [1].

Платформа имеет богатый набор внешней периферии: разъем питания mini-USB, переключатель питания, 3 разноцветных све-

тодиода, разъем подключения аккумуляторной батареи, разъем подключения солнечной батареи, кнопка сброса, разъемы подключения внешних плат расширения и плат с датчиками, разъемы для модулей беспроводной связи, датчик акселерометра. Внешний вид верхней части основного модуля платформы представлен на рисунке 2:

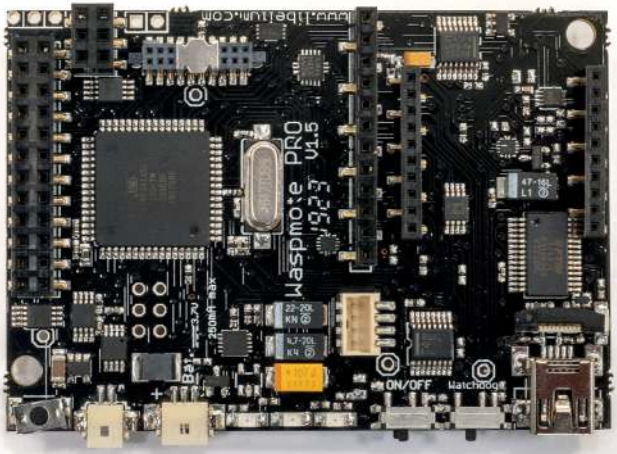


Рисунок 2 – Внешний вид верхней части основного модуля платформы Waspmote

Кроме перечисленной периферии модуль имеет часы реального времени RTC и возможность подключения карт памяти microSD. Внешний вид нижней части основного модуля платформы представлен на рисунке 3:

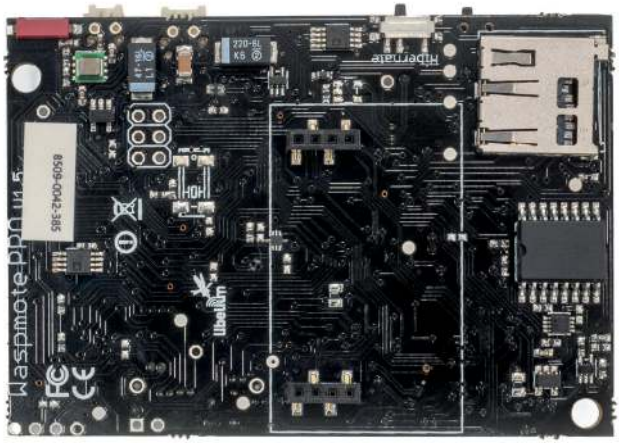
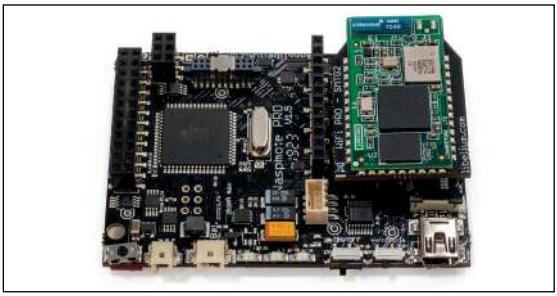


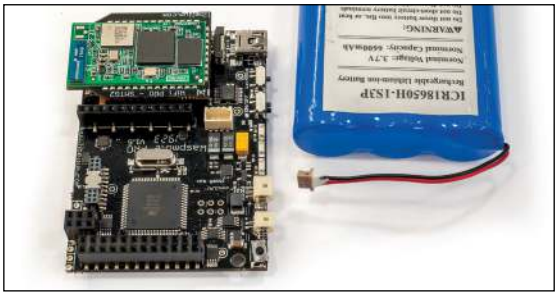
Рисунок 3 – Внешний вид нижней части основного модуля платформы Waspmote

Выполнение лабораторной работы требует сборки платформы в определенной конфигурации. Для получения работоспособной платформы к основному модулю необходимо дополнительно подключить: плату газоанализатора, датчик температуры, модуль беспроводного интерфейса, внешнюю аккумуляторную батарею, кабель mini-USB. Последовательность действий по сборке платформы находится в инструкции по эксплуатации платформы Waspmote [2]. Ниже Вы найдете в п. 1-6 краткие указания по сборке:

1 – присоединение антенны к модулю беспроводной передачи данных WiFi (выполняется при отсутствии встроенной в модуль антенны)

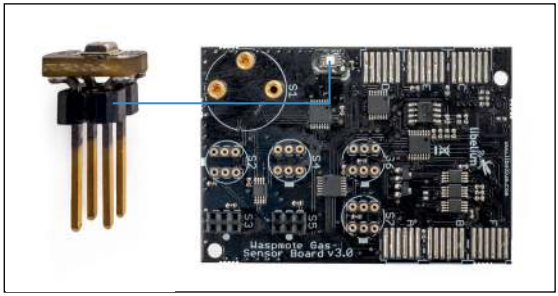


2 – установка модуля беспроводной передачи данных WiFi на платформу Waspmote

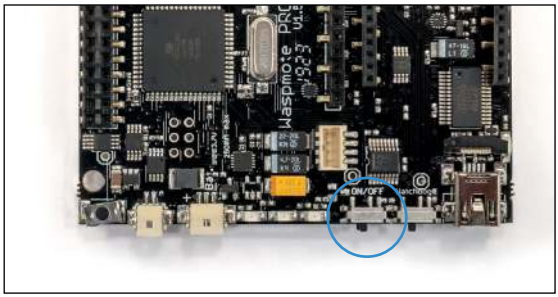


3 – подключение внешнего аккумулятора к Waspmote

4 – подключение платы газоанализатора к Waspmote



5 – подключение датчика температуры к плате газоанализатора (датчик необходимо установить в разъем S8 соблюдая полярность (совместить белые маркеры, обозначающие 1 контакт)



6 – включение питания производится с помощью переключателя ON/OFF на основном модуле Waspmote

Платформа готова к работе, для подключения к ПК необходимо использовать кабель mini-USB.

1.2. Описание сервиса ThingSpeak

Сервис ThingSpeak [3] – это облачная платформа, которая позволяет работать с задачами «интернета вещей» (Internet of Things, IoT), включая

сбор, визуализацию и анализ данных в реальном времени. Сервис позволяет интегрировать функции математики из MATLAB, что значительно расширяет ее возможности в области анализа и обработки данных. ThingSpeak часто используется для создания прототипов IoT систем и их проверки на уровне идеи, когда предполагается использование значительного объема аналитики.

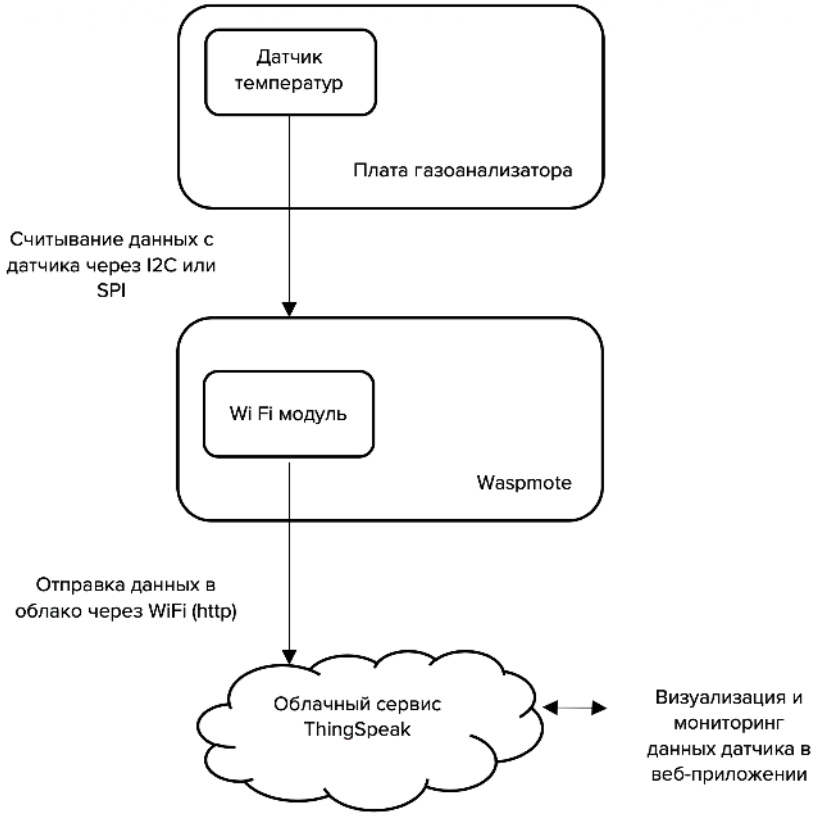
Платформа ThingSpeak состоит из каналов (частных или открытых), куда можно направлять данные с различных устройств для их хранения и визуализации. Каждый канал может иметь несколько полей для данных и различные другие параметры:

ID канала	Генерируется автоматически при создании канала и используется для индентификации канала при отправке данных на него
Имя	Указывается уникальное имя для платформы ThingSpeak
Описание	Добавляется по желанию пользователя
Поле N°	Дает возможность указать до 8 полей различного типа данных
Метаданные	Вводится информация о данных канала в формате JSON, XML или CSV
Теги	Указываются ключевые слова, которые позволяют идентифицировать канал
URL	Указывается ссылка на веб-сайт (если имеется), который связан с каналом ThingSpeak
Высота над уровнем моря	Указывается позиция датчика (в метрах), с которого поступают данные
Месторасположение	Указывается широта, долгота, высота, которые позволят выводить информацию в виде карты

2. Лабораторная работа «Сбор и визуализация данных с помощью платформы интернета вещей Libelium Waspmote»

Цель работы: познакомиться с работой платформы Waspmote и сервиса ThingSpeak для задач интернета вещей на примере сбора и визуализации данных с датчика температуры.

Алгоритм работы создаваемой платформы:



2.1 Задание 1 «Создание облачного хранилища данных»

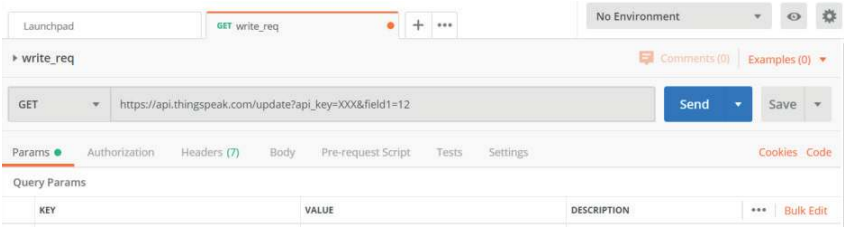
Шаг 1 – Настройка сервиса ThingSpeak

- Перейдите по ссылке https://thingspeak.com и создайте аккаунт.
- Создайте канал для загрузки данных о температуре и заполните необходимую информацию о канале, а именно – укажите имя канала (например, my_data), выберите и укажите имя для первого поля данных (например, temp), остальные параметры можно оставить без изменений.
- В разделе настроек общего доступа к каналу «Sharing» укажите «Share channel view with everyone», тем самым доступ к данным будет открыт любому пользователю по ссылке https://thingspeak.com/channels/123456 (где вместо 123456 нужно подставить ID канала, выданный сервисом ThingSpeak).
- Обратите внимание на раздел API keys, данные о ключе для записи (Write API Key) понадобятся для формирования API запросов к интернет-сервису.
- В правой части меню приведены примеры API запросов. В случае запроса на запись он выглядит так: GET https://api.thingspeak.com/update?api_key=XXX&field1=0 (вместо XXX будет отображаться значение ключа Write API Key).

Шаг 2 – Передача данных на канал ThingSpeak

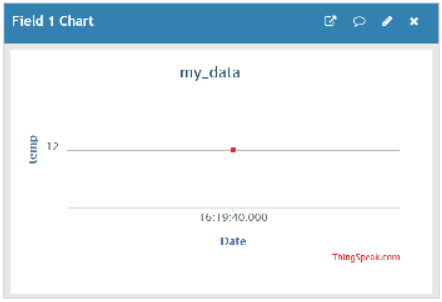
- Скачайте и установите программу Postman [4]. Данная программа позволяет сформировать API запрос и отправить его на интернет-сервис.
- После запуска программы создайте новый API запрос (File – New – Request). Задайте запросу произвольное имя (например, write_req). Выберите или создайте папку для сохранения результатов запроса (create collection – «status»). Вместо status можно использовать любое имя. Завершите создание запроса кнопкой «Save to status». В результате создан пустой шаблон API запроса.
- Укажите тип запроса GET. Скопируйте в поле адреса запрос, полученный на предыдущем шаге (https://api.thingspeak.com/update?api_key=XXX&field1=0).

- Вместо XXX подставьте значение ключа Write API Key вашего канала на сервисе ThingSpeak.
- Вместо значения «0» для поля field1, укажите любое другое число, например, «12».
- Отправьте запрос на интернет-сервис кнопкой «Send». Успешная



отправка запроса будет иметь статус «200 OK».

- Проверьте, что на вашем канале теперь лежат первые данные. Для этого перейдите по ссылке https://thingspeak.com/channels/123456 (где вместо 123456 нужно подставить ID канала, выданный сервисом ThingSpeak). Данная ссылка может быть открыта как в браузере на ПК,



так и на смартфоне.

*Бесплатный аккаунт на платформе ThingSpeak позволяет передавать данные не чаще, чем 1 раз в 20 секунд. Все переданные данные с меньшим интервалом будут проигнорированы.

В ходе выполнения данного задания было создано облачное хранилище данных, API ключи доступа к нему был проверены и использованы при передачи данных. Данные были переданы с помощью API запроса,

отправленного программой Postman. В задании 5 будет применен аналогичный подход передачи данных, однако API запрос будет формировать уже сама платформа Waspmote.

2.2 Задание 2 «Настройка платформы Waspmote»

Шаг 1 – Установка Libelium SDK

- Перейдите на сайт http://www.libelium.com
- Откройте раздел Development - Waspmote - SDK and applications.
- Скачайте среду Waspmote Pro IDE.
- В данном разделе также находится руководство пользователя к среде

Waspmote Pro IDE.

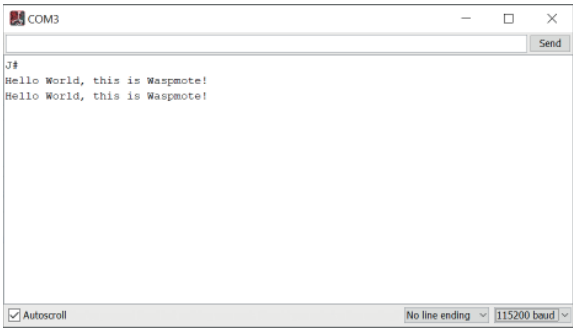
- Установите и проведите настройку платформы Waspmote Pro IDE, следуя инструкции из руководства пользователя [5].
- Запустите среду Waspmote Pro IDE.

Шаг 2 – Настройка платформы Waspmote

Перед выполнением этого задания платформа Waspmote должна быть собрана, как было показано в пунктах 1-6 раздела 1.1 «Описание платформы Waspmote».

- Подключите собранную платформу Waspmote к ПК, используя кабель mini-USB.
- Установите переключатель питания в положение «ON» (для данной лабораторной работы переключатели «Watchdog» и «Hibernate» должны находиться в одинаковом с переключателем ON/OFF положении).
- Необходимо определить номер COM порта, который был назначен платформе. Для этого откройте диспетчер устройств и найдите его в разделе «COM & LPT».

- Переключитесь в среду Waspmote Pro IDE.
- В меню Tools – Port укажите номер COM порта, назначенный платформе Waspmote.
- Откройте пример кода hello_world. Для этого в меню выберите File – Examples – 01.General – hello_world.
- Загрузите код на платформу с помощью горячих клавиш CTRL + U.
- Светодиоды на платформе Waspmote должны начать мигать.
- Откройте монитор последовательного порта Tools – Serial Monitor, задайте в нем скорость порта 115200. На экране должен появиться приветственный текст «Hello World, this is Waspmote!».



В ходе выполнения данного задания была запущена первая программа на платформе Waspmote. Ее выполнение можно наблюдать не только визуально по моргающим светодиодам, но и в терминальной программе на стороне ПК. Вывод информации в терминал очень полезен для отладки сложных программ, так как дает возможность выводить информативные уведомления о ходе выполнения программы. В следующих заданиях это будет активно использоваться.

2.3 Задание 3 «Измерение температуры окружающей среды»

Создайте в Waspmote Pro IDE программу (или скетч «sketch»), который будет собирать данные с датчика температуры и печатать в терминале на ПК текущее измеренное значение.

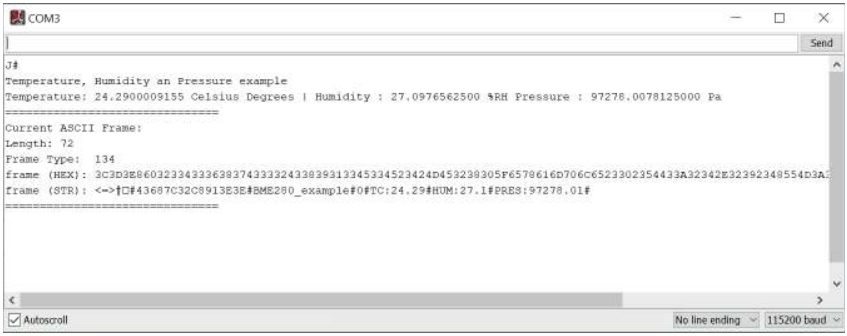
Вы можете воспользоваться примерами из библиотеки программ для платы газоанализатора «GasesBoard v30 Guide» [6].

Шаг 1 – Создание программы измерения температуры окружающей среды

- Откройте пример кода Ga_v30_01_BME280_sensor_reading. Для этого в меню выберите File – Examples – 02.Sensors – Smart_Gases_v30 – Ga_v30_01_BME280_sensor_reading.
- Загрузите код на платформу с помощью горячих клавиш CTRL + U

Шаг 2 – Проверка хода выполнения программы измерения температуры окружающей среды

- Откройте монитор последовательного порта Tools – Serial Monitor, задайте в нем скорость порта 115200. На экране должен появиться текст с измеряемыми данными (температурой, влажностью, давлением).
- Приложите палец к датчику температуры и убедитесь, что температура начала увеличиваться.
- *(опционально) Измените код программы так, чтобы выводилось только значение текущей температуры.



В ходе выполнения данного задания была создана программа по измерению температуры окружающей среды. Программа выполняется на платформе Waspmote и использует данные с датчика температуры, подключенного через плату газоанализатора. Фактически нами был создан «умный» датчик, пригодный для использования в интернете вещей. Следующим этапом настройки «умного» датчика является использование для передачи данных беспроводного соединения WiFi (Задания 4 и 5).

2.4 Задание 4 «Подключение платформы Waspmote к сети WiFi»

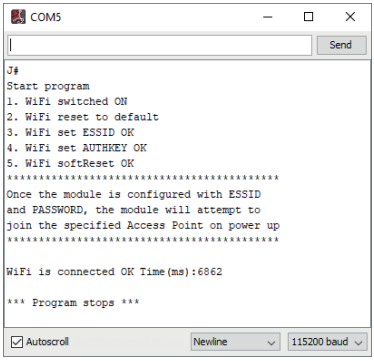
Узнайте, какая WiFi сеть с выходом в Интернет вам доступна. Запишите ее название (SSID) и пароль.

Шаг 1 – Подключение платформы Waspmote к Wi-Fi сети

- Откройте пример кода WIFI_PRO_01_configure_essid. Для этого в меню выберите File – Examples – 03.Communication – WIFI_PRO – WIFI_PRO_01_configure_essid.
- Измените строки кода, содержащие имя сети и пароль к ней
  char ESSID[] = «libelium_AP»;
  char PASSW[] = «password»;
  Замените libelium_AP и password на доступные вам.
- Загрузите код на платформу с помощью горячих клавиш CTRL + U

Шаг 2 – Проверка подключения платформы Waspmote к Wi-Fi сети

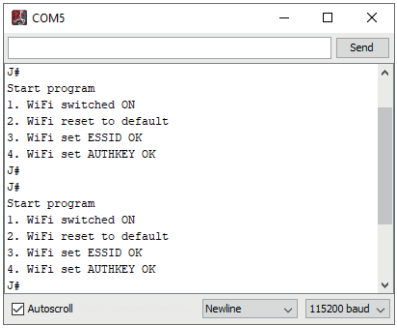
- Откройте монитор последовательного порта Tools – Serial Monitor, задайте в нем скорость порта 115200. На экране должен появиться текст с текущим статусом подключения к сети.



Если отображенный текст соответствует приведенному выше, платформа успешно подключилась к сети. Параметры точки доступа теперь хранятся внутри модуля беспроводной связи, и повторная их настройка в рамках данной лабораторной работы больше не требуется.

Модуль WiFi при запуске потребляет повышенный ток, поэтому питание

платформы только по проводу mini-USB может приводить к неправильному выполнению программы (пример такого выполнения приведен ниже). Для исправления ситуации подключите к платформе Waspmote внешнюю аккумуляторную батарею, как было показано в пункте 3 раздела 1.1 «Описание платформы Waspmote»



В ходе выполнения данного задания платформа Waspmote была подключена к сети WiFi и теперь она имеет доступ в интернет и может отправлять данные в облачное хранилище.

2.5 Задание 5 «Сбор данных о температуре окружающей среды и визуализация собранных данных в облаке»

Задание 5 строится на базе всех предыдущих заданий, поэтому как минимум задания 1 и 4 должны быть успешно выполнены.

Необходимо с помощью платформы Waspmote обеспечить сбор и передачу в облако данных о температуре окружающей среды. Передачу данных в облако можно реализовать при помощи отправки API запроса к интернет-сервису ThingSpeak. Примером формирования такого запроса в среде Waspmote Pro IDE является программа WIFI_PRO_12_http_get.

Шаг 1 – Создание программы сбора данных о температуре окружающей среды и передачи собранных данных в облако

- Откройте пример кода WIFI_PRO_12_http_get. Для этого в меню выберите File – Examples – 03.Communication – WIFI_PRO – WIFI_PRO_12_http_get.
- Измените строку адреса API запроса на соответствующую

созданному в задании 1 каналу данных платформы ThingSpeak:

```
char type[] = «http»; // «http» or «https»
char host[] = «pruebas.libelium.com»;
char port[] = «80»;
char link[] = «getpost_frame_parser.php?counter=1&varA=1&varB=2&var
C=3&varD=4&varE=5&varF=6&varG=7&varH=8&varI=9&varJ=10&varK=11
&varL=12&varM=13&varN=14&varO=15»;
```

Корректным адресом для платформы ThingSpeak является:

```
char host[] = «api.thingspeak.com»;
char link[]  = «update?api_key=XXX&field1=»;
```

Вместо XXX подставьте значение ключа Write API Key вашего канала на сервисе ThingSpeak.

- Добавьте в код библиотеку для работы платы газоанализатора и датчика температуры:

```
#include <WaspSensorGas_v30.h>
```

- Добавьте в конец функции setup() код, активирующий датчик температуры:

```
// Switch ON and configure the Gases Board
Gases.ON();
delay(100);
```

- Добавьте в начало функции setup() код, получающий текущее значение с датчика температуры:

```
float temperature; // Stores the temperature in ºC
// Read environmental variables
temperature = Gases.getTemperature();
```

- Закончите формирование API запроса на запись к онлайн-сервису ThingSpeak. Для этого строку link нужно дополнить текущим значением температуры, конвертированным в char.

После кода, получающего текущее значение с датчика температуры, добавьте код:

```
char body[150];
char buffer1[50];
strcpy(body,link);
Utils.float2String (temperature, buffer1, 3);
strcat(body, buffer1);
```

- Функция getURL формирует HTTP GET запрос, что в данном случае и является требуемым API запросом к платформе ThingSpeak.

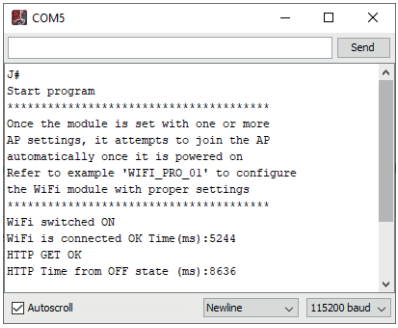
error = WIFI_PRO.getURL(type, host, port, link);

Изначально функция использует для запроса строку link, однако мы ее уже дополнили нашими данными и теперь правильной строкой является body, измените данную строчку кода на:

error = WIFI_PRO.getURL(type, host, port, body);
Загрузите код на платформу с помощью горячих клавиш CTRL + U.

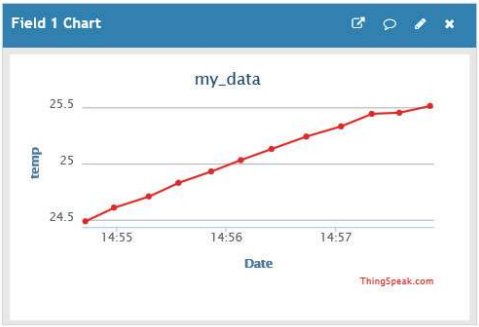
Шаг 2 – Проверка работы программы и визуализация данных в облаке

- Откройте монитор последовательного порта Tools – Serial Monitor, задайте в нем скорость порта 115200. На экране должен появиться текст со статусом выполнения программы.
- Найдите статус ответа сервера на отправленный API запрос
- «HTTP GET OK»



- Проверьте, что на вашем канале теперь периодически появляются новые данные. Для этого перейдите по ссылке https://thingspeak.com/channels/123456 (где вместо 123456 нужно подставить ID канала, выданный сервисом ThingSpeak). Данная ссылка может быть открыта как в браузере на ПК, так и на смартфоне.

*Бесплатный аккаунт на платформе ThingSpeak позволяет передавать данные не чаще, чем 1 раз в 20 секунд. Все переданные данные с меньшим интервалом будут проигнорированы.



В ходе выполнения данного задания был создан полноценный умный датчик интернета вещей. Датчик собирает информацию об окружающей среде, может при необходимости ее обрабатывать и отправлять в облачное хранилище для визуализации.

2.6 Задание 6* (опционально) «Сбор данных об окружающей среде, обработка данных и их визуализация в облаке»

Добавьте два новых поля для данных на созданном ранее канале интернет-сервиса ThingSpeak. Заполните эти поля данным о текущей влажности и атмосферном давлении. Добавьте в платформу Waspmote обработку собранных данных перед их передачей в облако. Например, можно считать среднее значение за несколько последних измерений или можно выделить на канале ThingSpeak отдельное поле «alert», в которое сообщать о превышении измеренными значениями заранее заданных пороговых показателей.

3. Источники

[1] Описание технических характеристик платформы Waspmote http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf

[2] Инструкция по эксплуатации платформы Waspmote http://www.libelium.com/downloads/documentation/waspmote_technical_guide.pdf

[3] Облачная платформа ThingSpeak https://thingspeak.com/

[4] Программа создания и запуска API запросов Postman https://www.getpostman.com/

[5] Руководство пользователя среды Waspmote Pro IDE
http://www.libelium.com/downloads/documentation/waspmote_ide_user_guide.pdf

[6] Библиотека программ для платы газоанализатора http://www.libelium.com/downloads/documentation/gases_sensor_board_3.0.pdf

4. Листинги кода лабораторной работы

4.1 Листинг 1 Код «hello_world»

```
void setup()
{
  // Opening UART to show messages using ‘Serial Monitor’
  USB.ON();
}

void loop()
{
  // Blinking LEDs
  Utils.blinkLEDs(1000);

  // Printing a message, remember to open ‘Serial Monitor’ to be able to see this message
  USB.println(F(«Hello World, this is Waspmote!»));

  // A little delay
  delay(2000);
}
```

4.2 Листинг 2 Код «Ga_v30_01_BME280_sensor_reading»

```
// Library include
#include <WaspSensorGas_v30.h>
#include <WaspFrame.h>

float temperature; // Stores the temperature in ºC
float humidity;    // Stores the realitve humidity in %RH
float pressure;    // Stores the pressure in Pa

char node_ID[] = «BME280_example»;

void setup()
{
  USB.ON();
  USB.println(F(«Temperature, Humidity an Pressure example»));
  // Set the Waspmote ID
  frame.setID(node_ID);

  ////////////////////////////////////////////
  // 1. Turn on the board
  ////////////////////////////////////////////

  // Switch ON and configure the Gases Board
  Gases.ON();
  delay(100);
}

void loop()
{
  ////////////////////////////////////////////
  // 2. Read sensors
  ////////////////////////////////////////////

  // Read enviromental variables
  temperature = Gases.getTemperature();
  humidity = Gases.getHumidity();
```

```
pressure = Gases.getPressure();

// Print of the results
USB.print(F(«Temperature: «));
USB.print(temperature);
USB.print(F(« Celsius Degrees |»));

USB.print(F(« Humidity : «));
USB.print(humidity);
USB.print(F(« %RH»));

USB.print(F(« Pressure : «));
USB.print(pressure);
USB.print(F(« Pa»));

USB.println();

//////////////////////////////////////////////
// 3. Create ASCII frame
//////////////////////////////////////////////

// Create new frame (ASCII)
frame.createFrame(ASCII, node_ID);
// Add temperature
frame.addSensor(SENSOR_GASES_TC, temperature);
// Add humidity
frame.addSensor(SENSOR_GASES_HUM, humidity);
// Add pressure
frame.addSensor(SENSOR_GASES_PRES, pressure);
// Show the frame
frame.showFrame();

delay(3000);
```

4.3 Листинг 3 Код «WIFI_PRO_01_configure_essid»

```
#include <WaspWIFI_PRO.h>


// choose socket (SELECT USER'S SOCKET)
///////////////////////////////////////
uint8_t socket = SOCKET0;
///////////////////////////////////////

// WiFi AP settings (CHANGE TO USER'S AP)
///////////////////////////////////////
char ESSID[] = «libelium_AP»;
char PASSW[] = «password»;
///////////////////////////////////////

// define variables
uint8_t error;
uint8_t status;
unsigned long previous;



void setup()
{
  USB.println(F(«Start program»));


  ////////////////////////////////////////////////
  // 1. Switch ON the WiFi module
  ////////////////////////////////////////////////
  error = WIFI_PRO.ON(socket);

  if (error == 0)
  {
    USB.println(F(«1. WiFi switched ON»));
```

```
}
else
{
  USB.println(F(«1. WiFi did not initialize correctly»));
}


////////////////////////////////////////////////
// 2. Reset to default values
////////////////////////////////////////////////
error = WIFI_PRO.resetValues();

if (error == 0)
{
  USB.println(F(«2. WiFi reset to default»));
}
else
{
  USB.println(F(«2. WiFi reset to default ERROR»));
}


////////////////////////////////////////////////
// 3. Set ESSID
////////////////////////////////////////////////
error = WIFI_PRO.setESSID(ESSID);

if (error == 0)
{
  USB.println(F(«3. WiFi set ESSID OK»));
}
else
{
  USB.println(F(«3. WiFi set ESSID ERROR»));
}
```

```
////////////////////////////////////////////////
// 4. Set password key (It takes a while to generate the key)
// Authentication modes:
//    OPEN: no security
//    WEP64: WEP 64
//    WEP128: WEP 128
//    WPA: WPA-PSK with TKIP encryption
//    WPA2: WPA2-PSK with TKIP or AES encryption
////////////////////////////////////////////////
error = WIFI_PRO.setPassword(WPA2, PASSW);

if (error == 0)
{
  USB.println(F(«4. WiFi set AUTHKEY OK»));
}
else
{
  USB.println(F(«4. WiFi set AUTHKEY ERROR»));
}


////////////////////////////////////////////////
// 5. Software Reset
// Parameters take effect following either a
// hardware or software reset
////////////////////////////////////////////////
error = WIFI_PRO.softReset();

if (error == 0)
{
  USB.println(F(«5. WiFi softReset OK»));
}
else
{
  USB.println(F(«5. WiFi softReset ERROR»));
}
```

```
  USB.println(F(«*****************************************»));
  USB.println(F(«Once the module is configured with ESSID»));
  USB.println(F(«and PASSWORD, the module will attempt to «));
  USB.println(F(«join the specified Access Point on power up»));
  USB.println(F(«*****************************************\n»));

  // get current time
  previous = millis();
}

void loop()
{
  ////////////////////////////////////////////////
  // Join AP
  ////////////////////////////////////////////////

  // Check if module is connected
  if (WIFI_PRO.isConnected() == true)
  {
    USB.print(F(«WiFi is connected OK»));
    USB.print(F(« Time(ms):»));
    USB.println(millis()-previous);

    USB.println(F(«\n*** Program stops ***»));
    while(1)
    {}
  }
  else
  {
    USB.print(F(«WiFi is connected ERROR»));
    USB.print(F(« Time(ms):»));
    USB.println(millis()-previous);
  }
}
```

4.4 Листинг 4 Код «WIFI_PRO_12_http_get»

```
// Put your libraries here (#include ...)
#include <WaspWIFI_PRO.h>
#include <WaspSensorGas_v30.h>

// choose socket (SELECT USER'S SOCKET)
///////////////////////////////////////
uint8_t socket = SOCKET0;
///////////////////////////////////////

// SERVER settings
///////////////////////////////////////
char type[] = «http»; // «http» or «https»
char host[] = «api.thingspeak.com»;
char port[] = «80»;
char link[]  = «update?api_key=XXX&field1=»;
///////////////////////////////////////

uint8_t error;
uint8_t status;
unsigned long previous;

void setup()
{
  USB.println(F(«Start program»));
  USB.println(F(«*************************************»));
  USB.println(F(«Once the module is set with one or more»));
  USB.println(F(«AP settings, it attempts to join the AP»));
  USB.println(F(«automatically once it is powered on»));
  USB.println(F(«Refer to example 'WIFI_PRO_01' to configure»));
  USB.println(F(«the WiFi module with proper settings»));
  USB.println(F(«*************************************»));

    // Switch ON and configure the Gases Board
  Gases.ON();
  delay(100);
}
```

```
void loop()
{

float temperature; // Stores the temperature in ºC
// Read enviromental variables
temperature = Gases.getTemperature();

char body[150];
char buffer1[50];
strcpy(body,link);
Utils.float2String (temperature, buffer1, 3);
strcat(body, buffer1);

 // get actual time
 previous = millis();

 ////////////////////////////////////////////////
 // 1. Switch ON
 ////////////////////////////////////////////////
 error = WIFI_PRO.ON(socket);
 if (error == 0)
 {
  USB.println(F(«WiFi switched ON»));
 }
 else
 {
  USB.println(F(«WiFi did not initialize correctly»));
 }

 ////////////////////////////////////////////////
 // 2. Join AP
 ////////////////////////////////////////////////

 // check connectivity
 status =  WIFI_PRO.isConnected();

 // Check if module is connected
 if (status == true)
```

```
{
  USB.print(F(«WiFi is connected OK»));
  USB.print(F(« Time(ms):»));
  USB.println(millis()-previous);

  // http request
  error = WIFI_PRO.getURL( type, host, port, body);

  // check response
  if (error == 0)
  {
    USB.println(F(«HTTP GET OK»));
    USB.print(F(«HTTP Time from OFF state (ms):»));
    USB.println(millis()-previous);

    USB.print(F(«\nServer answer:»));
    USB.println(WIFI_PRO._buffer, WIFI_PRO._length);
  }
  else
  {
    USB.println(F(«Error calling ‘getURL’ function»));
    WIFI_PRO.printErrorCode();
  }
}
else
{
  USB.print(F(«WiFi is connected ERROR»));
  USB.print(F(« Time(ms):»));
  USB.println(millis()-previous);
}

////////////////////////////////////////////////
// 3. Switch OFF
////////////////////////////////////////////////
WIFI_PRO.OFF(socket);
USB.println(F(«WiFi switched OFF\n\n»));
delay(10000);
}
```

Сколковский институт
науки и технологий
Территория Инновационного Центра
“Сколково”, Большой бульвар д.30, стр.1
Москва 121205
Телефон: +7 (495) 280 14 81

Email
iot@skoltech.ru
Сайт
https://iot.skoltech.ru/