AppZone от Telit:

практическое создание приложений

Технология AppZone, активно интегрируемая компанией Telit в модули сотовой связи и позволяющая запускать полноценные многозадачные приложения в 2G/3G/4G- модулях, в общих чертах была описана в [1]. В рамках данной статьи будет рассмотрен основной функционал AppZone, доступные приложению ресурсы, а также пример ее использования применительно к режимам энергосбережения.

Алексей Рудневский
rudnevsky.a@atoma.spb.ru

Все модули Telit, построенные на чипсетах Intel, поддерживают или Python, или AppZone (табл. 1). Эти технологии являются взаимоисключающими. В стандартных прошивках предусмотрена работа с Python, в соответствии с консервативным подходом компании к совместимости программного обеспечения (ПО). Это позволяет потребителям использовать все свои старые наработки, включая скрипты Python, на любых современных модулях. Для новых же проектов Telit предлагает выбор — или Python или AppZone. Последняя технология обладает рядом существенных преимуществ, таких как существенно более высокое быстродействие, многозадачность с системой приоритетов, наличие прерываний с фиксированным временем реакции, широкий набор инструментов для управления периферией и др.

Рассмотрим в качестве примера процесс создания небольшого тестового приложения и все его этапы, начиная с установки необходимого ПО и до инструментального измерения результатов работы. Параллельно созданию приложения можно будет оценить удобство использования как самой оболочки, так и аппаратно-программного интерфейса (API) AppZone для программиста.

Поскольку одним из ключевых параметров модуля сотовой связи является энергопотребление в различных режимах, тестовое приложение как раз и позволит оценить его при работе с AppZone.

Итак, для работы с AppZone потребуются следующие аппаратные средства:

- Компьютер с ОС Windows XP или выше.
- Отладочный набор Telit EVK M2M Air с одним или несколькими мезонинами: GE910, UE910, HE910 или LE910 V2 [2].
- При отсутствии EVK M2M Air можно использовать терминалы GT-HE910-EUG или GT863-3EU [3], но при этом будет существенно ограничена возможность работы с периферией модуля (в терминалах большая часть интерфейсов не выведена наружу).

На компьютер необходимо установить следующее программное обеспечение:

- Собственно само приложение ADE AppZone, включающее оболочку Eclipse, компилятор gcc, файлы заголовков, скелетное приложение, а также встроенную терминальную программу AT Console. Установка AppZone на компьютер происходит без особенностей, основные этапы показаны на рис. 1.
- Java Runtime Environment (JRE — среда выполнения для Java) — в случае, если JRE ранее не устанавливалась [4].
- USB-драйверы модулей Telit [5]. Они необходимы, если управление модулем будет производиться через USB-интерфейс.

Таблица 1.

Модуль	GL868-DUAL V3	GL865 V3	GE910 V3	GE866-QUAD	GE910-QUAD	GE910-GNSS	UL865	UE910	HE910-G	UE866	LE910 V2
Стандарты связи	GSM/GPRS						GSM/GPRS/UMTS/HSPA+				GSM/GPRS/UMTS/HSPA+/LTE
Наличие GNSS	*					GPS/ГЛОНАСС	*	*	GPS	*	*
Python	1.5.2				2.7.2						
Объем flash-памяти, Мбайт	0,8				2						
Объем RAM, Мбайт	1				2						
AppZone	–				+						**
Объем flash-памяти, Мбайт	–				3	3	5				
Объем памяти RAM	–				256 кбайт		2 Мбайт				

Примечания: * — модули сотовой связи поддерживают подключение и работу с внешними навигационными модулями; ** — ведется разработка

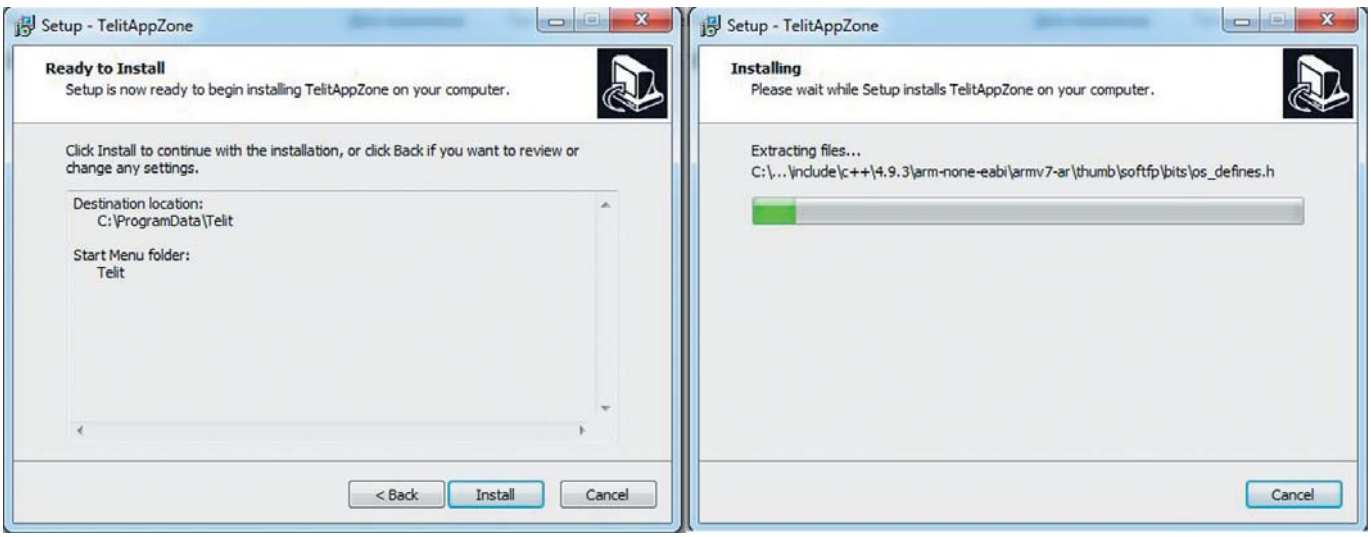

Рис. 1. Установка AppZone ADE

Также в модули должна быть предварительно записана специальная прошивка с поддержкой AppZone, причем версия прошивки должна совпадать с версией ADE (Application development environment) (рис. 2). При серийном применении модулей с AppZone рекомендуется заранее заказывать поставку уже с необходимой прошивкой.

После установки всего необходимого ПО можно приступать к созданию нового проекта. Назовем его CFUN (рис. 3) — по названию команды ***AT+CFUN***, управляющей энергопотреблением модуля [6].

При создании проекта нужно выбрать компилятор GCC — он уже встроен в ADE и отдельная его установка не требуется (рис. 4). После создания нового проекта появляется скелетная программа (sceleton), которую программист наполняет необходимым содержимым для реализации алгоритма выполнения приложения. Рассмотрим далее основные части скелета программы, а также заполним их в нашем примере.

Рис. 2. Соответствие версии ADE прошивке модуля

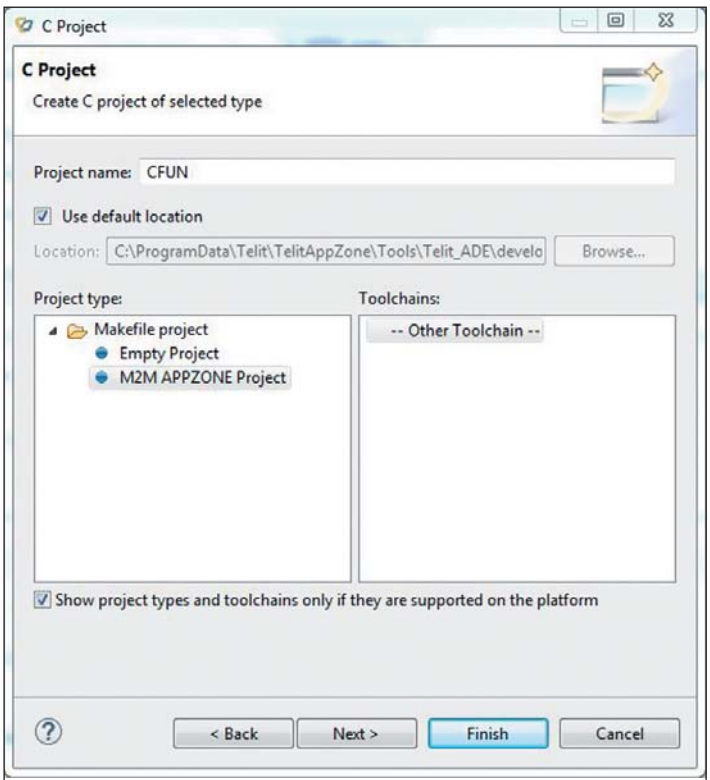

Рис. 3. Создание нового проекта AppZone

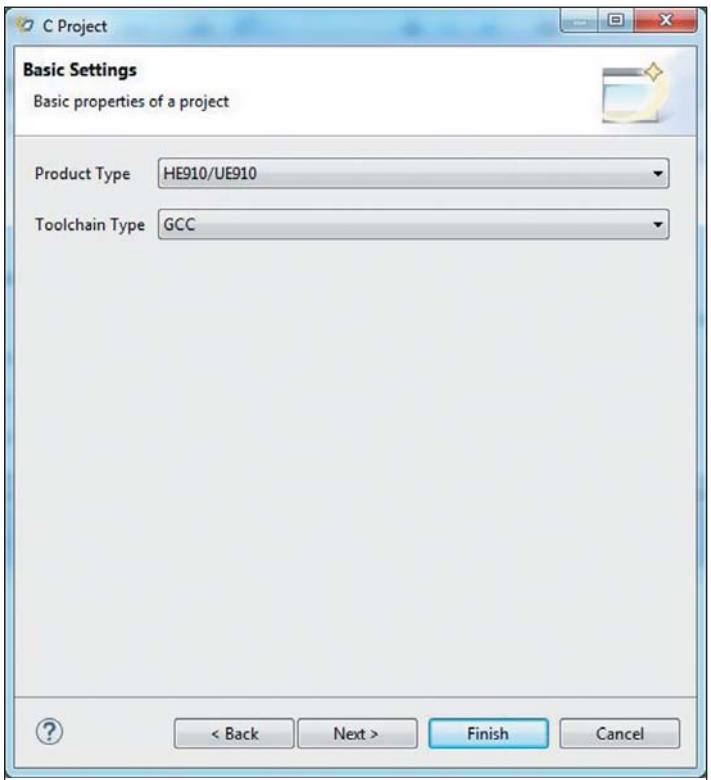

Рис. 4. Выбор платформы и компилятора

M2M_proc1.c — M2M_procX.c

Каждый файл соответствует одной задаче (task), выполняемой приложением. Максимальное количество задач в системе — 32, чем больше номер задачи, тем ниже ее приоритет. По умолчанию в скелете один файл *M2M_proc1.c* и, соответственно, одна задача.

Запишем в задачу команду управления энергосбережением ***AT+CFUN=5***:

```
INT32 M2M_msgProc1(INT32 type, INT32 param1, INT32 param2)
{
  LOG_INFO(«Put module into power sawing mode AT+CFUN=5\r\n»);
  m2m_os_sleep_ms(1000);
m2m_os_iat_send_at_command(«AT+CFUN=5\r»,1);
return 0;
}
```

Помимо ***M2M_msgProc1***, файл *M2M_proc1.c* содержит также функцию ***M2M_msgProcCompl***, вызывающуюся при завершении задачи.

M2M_main.c

Здесь находятся три функции: ***M2M_main, M2M_suspend u M2M_resume***. Последние две зарезервированы для использования в будущем, а вот ***M2M_main***, как следует из названия, — это точка входа приложения. Модифицируем стандартную функцию следующим образом:

```
void M2M_main (INT32 argc, CHAR argv[M2M_ARGC_MAX][M2M_ARGV_MAXTOKEN + 1])
{
  LOG_INFO(«CFUN test start \r\n»);
  m2m_os_iat_set_at_command_instance(1,1);
  m2m_network_enable_registration_location_unsolicited();
  AT_parser = m2m_os_create_task(M2M_OS_TASK_STACK_M, 1, M2M_OS_TASK_
MBOX_M, M2M_atParserProc);
  m2m_os_iat_send_at_command(«AT+CREG?\r»,1);
}
```

В нашем примере функция создает связь между логическим портом и экземпляром обработчика AT-команд, разрешает модулю регистрацию в сети сотовой связи и запрашивает результат регистрации командой ***AT+CREG?***. Сам обработчик (***M2M_atParserProc***) находится в отдельном файле *M2M_atParser.c*. Этот файл не входит в скелет программы, и подробное его рассмотрение выходит за рамки данной статьи. Тем не менее, полные исходные коды приложения CFUN можно найти в [7].

M2M_atRsp.c

В этом файле находится функция ***M2M_onReceiveResultCmd***, которая вызывается при поступлении ответов модуля на AT-команды. В нашем случае функция передает управление обработчику AT-команд, упомянутому чуть выше:

```
INT32 M2M_onReceiveResultCmd(CHAR *atCmd, INT32 len, UINT16 logPort)
{
m2m_os_send_message_to_task(AT_parser, 0, (INT32)atCmd, len);
return 1;
}
```

M2M_hwEvents.c

Файл содержит следующие функции:

- ***M2M_onWakeup*** — обработка сигнала будильника.
- ***M2M_onInterrupt*** — обработка аппаратных прерываний от GPIO. Прерывания могут формироваться от восьми портов модуля, заданное время реакции 130 мс.
- ***M2M_onHWTimer*** — работа с аппаратными таймерами. Возможна установка до семи таймеров заданной длительностью от 100 мкс до 300 мс.
- ***M2M_onUSbCableEvent*** — обработка события подключения или отключения кабеля USB.
- ***M2M_onKeyEvent*** — обработка нажатия кнопки включения модуля.

В нашем примере описанные в данном файле аппаратные интерфейсы не используются, поэтому скелетный файл не модифицируется.

M2M_initialize.c

Файл включает функцию ***InitUserInterface***, выполняющуюся однократно при старте приложения, а также несколько пока неиспользуемых функций, предназначенных для развития AppZone в будущем.

M2M_net.c

В файле описаны функции, связанные с сетевыми событиями:

- ***M2M_onNetEvent*** — обработка событий, связанных с функционированием модуля в сети мобильной связи, например активация PDP-контекста.
- ***M2M_onRegStatusEvent*** — события, связанные с регистрацией в сети, например изменение статуса регистрации. В нашем примере приложение отслеживает статус сети, поэтому стандартная функция модифицируется:

```
void M2M_onRegStatusEvent (UINT16 status, UINT8* location_area_code, CHAR* cell_id,
UINT16 Act)
{
  LOG_INFO(«status: %d cellID: %s \r\n», status, cell_id);
  if((status == 1) && (strcmp(cell_id, «FFFF»)!= 0)) {
      LOG_INFO(«Module registered, ready to receive data call\r\n»);
      m2m_network_disable_registration_location_unsolicited();
  }
}
```

- ***M2M_onMsgIndEvent*** — обработка поступившего SMS.
- ***M2M_onIP6RawEvent*** — прием пакета IPv6.

M2M_shell.c

В этом файле заготовка только одной функции ***M2M_cmdShell***, пока не используемой и предназначенной для использования в будущем.

Помимо скелетной программы, при создании проекта формируются заголовочные файлы, декларирующие основные функции API. Поскольку это основной инструментарий программиста AppZone, рассмотрим их чуть более подробно. Полное описание функций API приведено в [8].

m2m_clock_api.h

Заголовочный файл *m2m_clock_api.h* содержит описание следующих функций:

- установка/чтение текущих времени и даты;
- управление будильником.

m2m_fs_api.h

Здесь описываются основные функции для работы с файловой системой:

- ***m2m_fs_create*** — создание нового файла.
- ***m2m_fs_open*** - открытие существующего файла. Файл может быть открыт на чтение, на запись, на модификацию (без изменения размера), на добавление данных.
- ***m2m_fs_close*** — закрытие файла.
- ***m2m_fs_delete*** — удаление файла. Перед удалением файл должен быть закрыт.
- ***m2m_fs_clear*** — полная очистка файловой системы с форматированием.
- ***m2m_fs_copy*** — копирование файла в файл с новым именем.
- ***m2m_fs_find_first, m2m_fs_find_next*** — поиск файлов.
- ***m2m_fs_rename*** — переименование заданного файла.
- ***m2m_fs_get_size, m2m_fs_get_size_with_handle*** — определить размер файла.
- ***m2m_fs_tell*** — получить текущий указатель на данные в файле
- ***m2m_fs_seek*** — сместить указатель на заданное значение
- ***m2m_fs_truncate*** — сократить размер файла. Фактически размер сократится после закрытия файла.
- ***m2m_fs_getc*** — прочитать символ из файла.
- ***m2m_fs_gets*** — прочитать строку из файла.
- ***m2m_fs_read*** — прочитать из файла заданное количество данных. Файл должен быть открыт только для чтения.
- ***m2m_fs_write*** — записать в файл заданное количество данных. Файл должен быть открыт только для записи.
- ***m2m_fs_mk_dir*** — создать новый каталог в файловой системе.
- m2m_fs_rename_dir — переименовать существующий каталог.
- ***m2m_fs_rm_dir*** — удалить существующий каталог. В каталоге перед удалением не должны содержаться файлы.
- ***m2m_fs_get_free_space*** — получить информацию о свободном месте в файловой системе (в байтах).
- ***m2m_fs_get_nof_files*** — получить информацию об общем количестве файлов в файловой системе.
- ***m2m_fs_last_error*** — получить информацию о последней ошибке при работе с файловой системой.

Таким образом, AppZone позволяет управлять полноценной файловой системой при помощи функций, аналогичных стандартным в ANSI C.

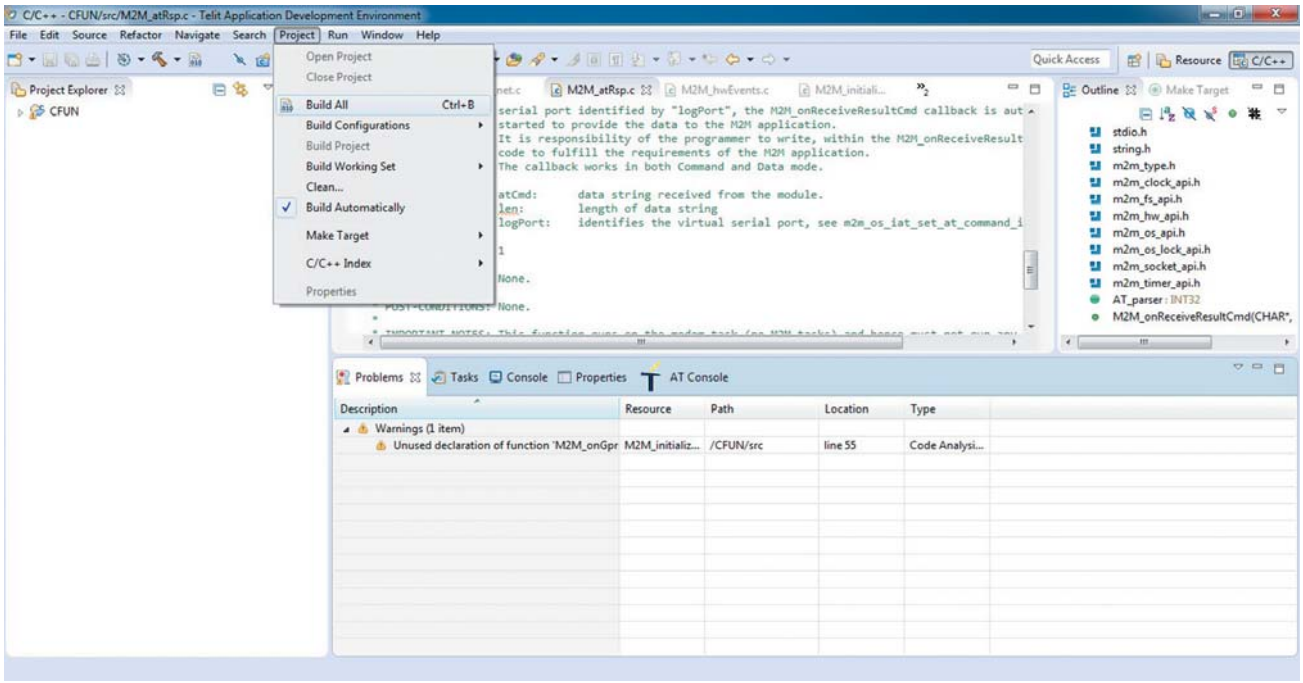

Рис. 5. Компиляция проекта

m2m_hw_api.h

В этом файле содержатся объявления функций API, предназначенных для управления периферийными ресурсами модуля:

- портами ввода/вывода (GPIO);
- счетчиком;
- аппаратными таймерами (как было сказано выше, их семь);
- последовательным портом (UART);
- передачей данных по USB;
- переходом в спящий режим и выключением.

m2m_i2C_api.h

Декларированы функции, управляющие передачей данных по шине I²C.

m2m_spi_api.h

Передача данных по шине SPI.

m2m_socket_api.h

Здесь декларируются функции управления сокетами TCP/IP. AppZone поддерживает работу одновременно с 10 сокетами, как клиентскими, так и серверными, как TCP, так и UDP.

m2m_network_api.h

Прототипы функций, позволяющих управлять взаимодействием модуля с сетью сотовой связи.

m2m_os_api.h и ***m2m_os_lock_api.h***

Эти файлы содержат функции взаимодействия с операционной системой модуля, включая управление задачами, семафоры, обмен сообщениями между задачами и пр.

m2m_sms_api.h

Как можно предположить из названия, здесь описаны функции работы с короткими сообщениями (SMS).

m2m_ipraw_api.h

AppZone поддерживает работу по протоколу IPv6, и в данном файле собраны функции API для работы с ним.

m2m_ssl_api.h

Заголовочный файл содержит прототипы функций API для работы по протоколу SSL. Поддерживается только клиентское соединение по протоколу SSL:

- создание сессии SSL;
- для каждого соединения создается соответствующий контекст;
- для шифрования и передачи данных используется функция ***m2m_ssl_encode_send***;
- после получения зашифрованных данных используется ***m2m_ssl_decode*** для дешиврации.

m2m_timer_api.h

Набор функций для работы с таймерами.

Компиляция и запуск проекта

Для компиляции проекта можно воспользоваться пунктом меню Build All или же горячими клавишами<Ctrl>+<B> (рис. 5). В случае наличия ошибок в файлах проекта их описание появится во вкладке Problems, в случае успешной компиляции в каталоге проекта появится файл *m2mapz.bin*, который необходимо загрузить в модуль. Для загрузки файлов и запуска Telit предлагает воспользоваться специальной утилитой **File System Tool** (рис. 6), входящей в комплект дистрибутива AppZone. Утилита позволяет считывать, записывать и удалять файлы из файловой системы модуля, а также управлять запуском и остановкой приложений AppZone, загруженных в модуль. Альтернативой для работы с файловой системой является использование специальных AT-команд, описанных в [9]. AT-команды могут подаваться как через AT-консоль (AT Console) в ADE (рис. 7), так и через любую привычную пользователю терминальную программу, вплоть до стандартного гипертерминала в Windows.

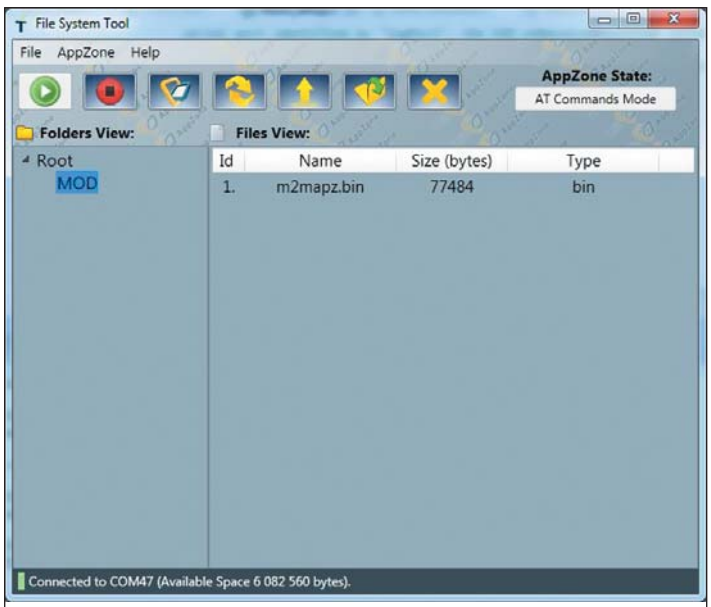

Рис. 6. Утилита File System Tool

Рис. 7. Консоль АТ-команд

Измерение энергопотребления модуля

Для измерения энергопотребления используется простейшая схема, приведенная на рис. 8. По сути, измеряется падение напряжения на резисторе номиналом 1 мОм, увеличенное в тысячу раз. Таким образом, значения измеренных напряжений в милливольтах соответствуют потребляемому току в миллиамперах. В качестве измерительного прибора использовался осциллограф LeCroy LT344L [10], измерялось энергопотребление модулей GE910 и UE910 в следующих режимах:

- стандартный режим без приложения AppZone;
- приложение CFUN, режим энергосбережения выключен;
- приложение CFUN, режим энергосбережения включен.

Результаты измерений приведены в таблице 2. Осциллограмма, показывающая один из результатов (соответствующие ячейки выделены желтым цветом), приведена на рис. 9.

Как свидетельствуют данные таблицы 2, использование AppZone увеличивает энергопотребление модуля приблизительно на 20–30%, что вполне естественно: потребляются дополнительные аппаратные ресурсы. А вот измерение в режиме энергосбережения дало весьма неожиданные результаты: для 3G-модуля UE910 эффект снижения потребления мало заметен, в то время как на GSM/GPRS-модуле GE910 энергопотребление уменьшилось более чем в 4 раза. Это связано с тем, что большую часть потребления 2G-модуля составляет трансивер, работающий в режиме TDMA, и постоянная его работа не требуется. Для 3G-модулей, трансивер которых работает в CDMA, необходима его постоянная работа, что не позволяет радикально снизить энергопотребление.

Таблица 2. Энергопотребление модулей Telit в различных режимах

Режим работы модуля	GE910		UE910	
	Средний ток I_{mean}, мА	Средне-квадратичный ток I_{rms}, мА	Средний ток I_{mean}, мА	Средне-квадратичный ток I_{rms}, мА
Без AppZone	29,0	29,56	12,55	13,45
Приложение CFUN, энергосбережение выключено	38,65	39,47	15,79	16,94
Приложение CFUN, энергосбережение включено	8,04	8,36	11,67	12,58

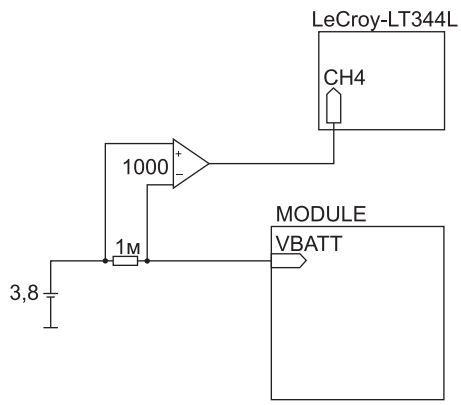

Рис. 8. Структурная схема стенда для измерения энергопотребления

Заключение

Разработчики Telit проделали большую работу по созданию AppZone. Благодаря этому программисты, которые будут использовать данную технологию для создания новых продуктов, получают практически весь необходимый для написания приложений инструментарий из дистрибутива AppZone. Документация Telit достаточно проста и логична, все предлагаемые программные средства могут быть быстро освоены. Все это является частью нового принципа компании «One stop — one shop», символизирующего предоставление всего необходимого для разработки и развития проекта из одного источника. ■

Литература

1. Рудневский А. Технология AppZone в модулях Telit GE910 //Беспроводные технологии. 2014. № 1.
2. http://atoma.spb.ru/sites/default/files/documents/telit_evk2_user_guide_r19.pdf
3. http://atoma.spb.ru/catalog/2413/gt863-3eu
4. http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
5. http://atoma.spb.ru/documentation/3519/drajver-3g-i-4g-modulej-dlja-windows
6. http://atoma.spb.ru/sites/default/files/documents/telit_3g_modules_at_commands_reference_guide_r10.pdf
7. http://atoma.spb.ru/sites/default/files/documents/cfun.zip
8. http://atoma.spb.ru/sites/default/files/documents/appzone_apis_user_guide.pdf
9. http://atoma.spb.ru/sites/default/files/documents/appzone_user_guide_he_ue_ge_r4.pdf
10. http://cdn.teledynelecroy.com/files/manuals/wr2_om_revc.pdf

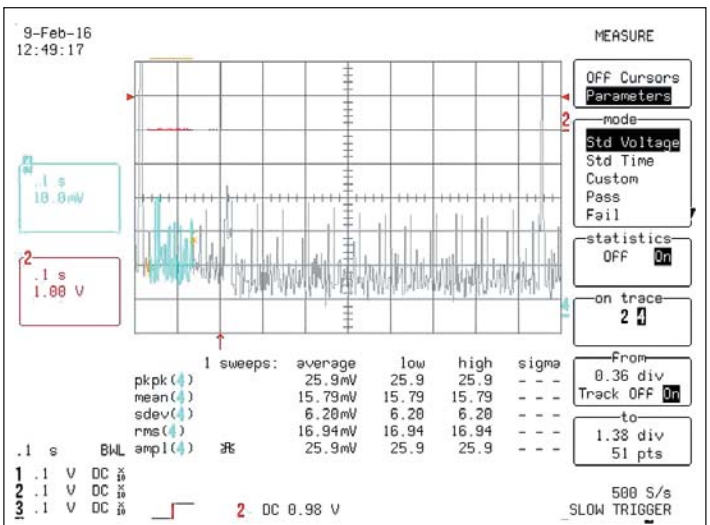

Рис. 9. Осциллограмма энергопотребления модуля