Technologia GSM w elektronice (8)

Open AT – obsługa SMTP, wysyłanie wiadomości MMS

W poprzednim odcinku kursu programowania modemów AirPrime Q26 firmy Sierra Wireless przedstawiliśmy system zdalnej aktualizacji aplikacji (DOTA). Dziś zajmiemy się wysyłaniem wiadomości e-mail oraz MMS.

Współczesne modemy i telefony GSM mają możliwość nie tylko wysyłania wiadomości SMS, ale również umożliwiają przesyłanie wiadomości multimedialnych, wieloczęściowych MMS, transmisję danych z użyciem protokołu TCP/IP oraz wysyłanie wiadomości e-mail. Dostępna w środowisku *Sierra Wireless Software Suite* biblioteka internetowa WIP (Wavecom IP) zawiera między innymi obsługę e-mail z użyciem protokołów SMTP i POP3 oraz daje umożliwia wysyłanie wiadomości MMS.

Listing 1. Przykładowy do wysyłania e-mail z użyciem protokółów SMTP i POP

```
#include "adl_global.h"
#include "wip.h"
#include "wip_smtp.h"    // WIP smtp services
const u16 wm_apmCustomStackSize = 1024*4;
ascii tekst[100];
ascii * PinCode = "";
ascii * GPRS_APN="internet";
ascii * GPRS_USER="";
ascii * GPRS_PASSWORD="";
wip_bearer_t bearer_handle;
#define SMTP_CLIENT_TEST_PORT       25
const ascii * SMTP_STR_HOSTNAME   = "smtp.wp.pl";
const ascii * SMTP_STR_USERNAME   = "testxxx";      //login do konta SMTP
const ascii * SMTP_STR_PASSWORD   = "testxxx_pass"; //hasło do konta SMTP
const ascii * SMTP_STR_SENDERNAME = "Sierra Module";   //nazwa nadawcy
const ascii * SMTP_STR_SENDER     = "testxxx@wp.pl"; //adres nadawcy
const ascii * SMTP_STR_REC        = "ac@testmail.pl";              //adres odbiorcy
const ascii * SMTP_STR_CCREC      = „test_tr@gmail.com";  //adres odbiorcy w kopii

const ascii * SMTP_STR_SUBJ       = "Mail from Sierra Module";
// Pre formatted mail without header (wip smtp generates the header)
const ascii * SMTP_STR_BODY       = "Przykladowa tresc wiadomosci e-mail            \r\n"
                                    "                                               \r\n"
                                    "recipients lists.\r\n"
                                    "                 \r\n"
                                    " Pozdrawiam,     \r\n"
                                    "The WipSender... \r\n"
                                    "                 \r\n";

typedef struct
{
  wip_channel_t CnxChannel;          // session channel
  wip_channel_t DataChannel;         // data channel
  u8           *pdataBuffer;         // mail data pointer
  u32           dataLength;          // mail data length
} smtp_ClientTestCtx_t;

smtp_ClientTestCtx_t smtp_ClientTestCtx;
static s32 smtp_ClientTestPutFile(void);
static void poll_creg( u8 Id );
static void smtp_ClientTestCnxHandler(wip_event_t *ev, void *ctx)
{
  smtp_ClientTestCtx_t *pSmtpClientCtx = (smtp_ClientTestCtx_t *)&smtp_ClientTestCtx;
  u32 StatusCode = 0;
  u32 ErrCode = 0;
  ascii **pErrorString;
  u32 ret = 0;
  TRACE(( 4, "smtp_ClientTestCnxHandler: %d 0x%x\n", ev->kind, ctx ));
  switch(ev->kind)
  {
    case WIP_CEV_OPEN:
      TRACE(( 4, "smtp_ClientTestCnxHandler: WIP_CEV_OPEN\n" ));
      // open the DATA channel
```

Wysyłanie wiadomości e-mail

Na **listingu 1** zamieszczono przykładową aplikacje wysyłającą wiadomość e-mail, z użyciem do tego celu zewnętrznego serwera poczty. Przed skompilowaniem aplikacji należy w treści zmienić parametry serwera SMTP oraz adresy nadawcy i odbiorcy.

Aplikacja rozpoczyna działania od wpisania kodu PIN i nawiązania sesji GPRS, co było już omawiane poprzednio. Dla uproszczenia usunąłem treść *open_and_start_bearer(), poll_creg_callback(), poll_creg(), SimHandler (),* a także uprościłem *funkcję evh_bearer()*, pozostawiając jedynie wywołanie funkcji *smtp_ClientTestCreate()* po wystąpieniu zdarzenia *WIP_BEV_IP_CONNECTED*. Funkcje te są identyczne, jak w dotychczas prezentowanych przykładach i dla większości osób uważnie śledzących kurs nie będzie stanowiło problemu ich odtworzenie.

Po nawiązaniu sesji GPRS następuje połączenie z serwerem SMTP (*wip_SMTPClientCreateOpts*), a po otwarciu połączenia przesłanie danych za pomocą funkcji *wip_putFileOpts()*.

Przedstawiona na listingu 1 aplikacja pozwala na wysyłanie wiadomości e-mail jako zwykłego tekstu. Jeśli chcielibyśmy wysyłać wiadomości w formacie HTML lub wiadomości z załącznikami, to również jest to możliwe.

W przedstawionym przykładzie tworzeniem nagłówka zajmuje się biblioteka WIP, a podajemy jedynie treści wiadomości do wysłania. Jest jednak możliwość samodzielnego tworzenia całej wiadomości wraz z nagłówkiem. W tym celu parametr *WIP_COPT_SMTP_FORMAT_HEADER* wywołania funkcji *wip_putFileOpts()* powi-

Listing 1. c.d.

```
      smtp_ClientTestPutFile();
      break;
    case WIP_CEV_PEER_CLOSE:
      TRACE(( 4, "smtp_ClientTestCnxHandler: WIP_CEV_PEER_CLOSE\n" ));
      if (pSmtpClientCtx->CnxChannel != NULL)
      {
        wip_close(pSmtpClientCtx->CnxChannel);
        pSmtpClientCtx->CnxChannel = NULL;
      }
      break;
    case WIP_CEV_ERROR:
      ret = wip_getOpts( pSmtpClientCtx->CnxChannel,
                         WIP_COPT_ERROR, &ErrCode,
                         WIP_COPT_END);
      ret = wip_getOpts( pSmtpClientCtx->CnxChannel,
                         WIP_COPT_SMTP_STATUS_CODE, &StatusCode,
&pErrorString,
                         WIP_COPT_END);
      TRACE(( 4, "smtp_ClientTestCnxHandler: WIP_CEV_ERROR %d \n", ErrCode));
      wip_debug("[ERROR SMTP]: %d %s \n", StatusCode, pErrorString);
      if (pSmtpClientCtx->CnxChannel != NULL)
      {
        wip_close(pSmtpClientCtx->CnxChannel);
        pSmtpClientCtx->CnxChannel = NULL;
      }
      break;
    default:
      break;
  }
}

static void smtp_ClientTestDataWriteHandler(void)
{
  smtp_ClientTestCtx_t *pSmtpClientCtx = (smtp_ClientTestCtx_t *)&smtp_
ClientTestCtx;
  while( pSmtpClientCtx->dataLength > 0 )
  {
    int WrittenBytes;
    WrittenBytes = wip_write( pSmtpClientCtx->DataChannel,
                              pSmtpClientCtx->pdataBuffer,
                              pSmtpClientCtx->dataLength );
    if (WrittenBytes <= 0)
    {
      break;
    }
    pSmtpClientCtx->pdataBuffer    += WrittenBytes;
    pSmtpClientCtx->dataLength     -= WrittenBytes;
  }
  if (pSmtpClientCtx->dataLength == 0)
  {
    TRACE(( 4, "smtp_ClientTest: Close DATA channel\n" ));
    if (pSmtpClientCtx->DataChannel != NULL)
    {
      wip_close(pSmtpClientCtx->DataChannel);
      pSmtpClientCtx->DataChannel = NULL;
    }
    TRACE(( 4, "smtp_ClientTest: Close SESSION channel\n" ));
    if (pSmtpClientCtx->CnxChannel != NULL)
    {
      wip_close(pSmtpClientCtx->CnxChannel);
      pSmtpClientCtx->CnxChannel = NULL;
    }
  }
}

static void smtp_ClientTestDataHandler(wip_event_t *ev, void *ctx)
{
  smtp_ClientTestCtx_t *pSmtpClientCtx = (smtp_ClientTestCtx_t *)&smtp_
ClientTestCtx;
  s32 ret = 0;
  ascii **pErrorString;
  u32 ErrorCode;
  u32 StatusCode;


  TRACE(( 4, "smtp_ClientTestDataHandler: %d 0x%x\n", ev->kind, ctx ));

  switch(ev->kind)
  {
    case WIP_CEV_OPEN:
      TRACE(( 4,"smtp_ClientTestDataHandler: WIP_CEV_OPEN\n" ));

      // Send a normal text mail body
      pSmtpClientCtx->pdataBuffer = SMTP_STR_BODY;
      pSmtpClientCtx->dataLength  = wm_strlen(SMTP_STR_BODY);
      break;
    case WIP_CEV_PEER_CLOSE:
      TRACE(( 4, "smtp_ClientTestDataHandler: WIP_CEV_PEER_CLOSE\n" ));
      break;
    case WIP_CEV_ERROR:
      ret = wip_getOpts(pSmtpClientCtx->DataChannel,
                        WIP_COPT_ERROR, &ErrorCode,
                        WIP_COPT_END);
      ret = wip_getOpts(pSmtpClientCtx->DataChannel,
                        WIP_COPT_SMTP_STATUS_CODE, &StatusCode, &pErrorString,
                        WIP_COPT_END);
      TRACE(( 4, "smtp_ClientTestDataHandler: WIP_CEV_ERROR %d\n",
ErrorCode));
      wip_debug("[ERROR SMTP]: %d %s \n", StatusCode, pErrorString);
      break;
    case WIP_CEV_WRITE:
      smtp_ClientTestDataWriteHandler();
      break;
```

nien mieć wartość 0. Aplikację z takim właśnie sposobem wysyłania wiadomości można znaleźć w przykładach dostępnych po zainstalowaniu IDE.

Biblioteka WIP jest inicjalizowana z opcją *WIP_NET_OPT_DEBUG_PORT*. Pozwala to na korzystanie z funkcji *wip_debug()* do wyświetlania informacji diagnostycznych biblioteki WIP, co w przypadku problemów z komunikacją z serwerem SMTP znacznie ułatwi analizę.

Wysyłanie wiadomości MMS

Na **listingu 2** zamieszczono aplikację wysyłającą wiadomość MMS na wskazany numer telefonu lub adres e-mail. Aby dołączyć grafikę (JPG) lub dźwięk (AMR), należy najpierw utworzyć plik (pliki) z rozszerzeniem *.h* zawierający dany plik zmieniony do postaci wektorów danych. Do tego celu można użyć dostępnego w Internecie programu *bin2h.exe*. Po utworzeniu pliku z danymi umieszczamy go metodą kopiuj-wklej w katalogu */inc* projektu w *M2MStudio*, a następnie dołączamy za pomocą dyrektywy *#include*.

Aby prawidłowo wysyłać wiadomości MMS niezbędne jest odpowiednie skonfigurowanie APN, serwera Proxy oraz numeru portu. Niezbędne parametry można znaleźć na stronach operatorów. W przykładzie posłużyłem się ustawieniami dla sieci Orange.

Aplikacja zaczyna się od wpisania kodu PIN i nawiązania sesji GPRS. Tym razem korzystamy jednak z APN-u dla MMS, a nie dla Internetu. Po nawiązaniu polaczenia za pomocą funkcji *wip_mmsCreateOpts()* konfigurujemy takie parametry, jak: nadawca, odbiorca i tryb. Podajemy również funkcję *status_callback*, która zostanie wykonana po wysłaniu MMS lub wskutek wystąpienia błędu.

Po skonfigurowaniu parametrów następuje tworzenie ciała wiadomości MMS. Następuje to poprzez złożenie różnych składowych (tekst, grafika, dźwięk), za pomocą funkcji *wip_mmsAddPart()*. Wielkość każdej z załączanych części nie powinna przekroczyć 2 MB. W kolejnym kroku następuje nawiązanie połączenia HTTP przez określony port do wskazanego adresu IP. Po otwarciu kanału http następuje przesłanie treści wiadomości MMS za pomocą funkcji *wip_mmsSend()*.

Podsumowanie

Dokładny opis używanych funkcji API oraz informacje o pozostałych możliwościach biblioteki WIP są w dokumentacji, którą znajdziemy w menu *Pomoc* środowiska *M2MStudio* (*Help –> Help Contents –> Plug-ins Documentation –> WIP Open AT Plug-in Package*).

Więcej informacji na temat produktów Sierra Wireless można znaleźć na stronach producenta: *www.sierrawireless.com* lub kontaktując się z firmą ACTE Sp. z o.o., która jest oficjalnym dystrybutorem opisywanych produktów oraz zapewnia ich pełne wsparcie techniczne.

Adrian Chrzanowski
Acte Sp. z o.o.

Listing 1. c.d.

```
    default:
      break;
  }
}

static s32 smtp_ClientTestCreate(void)
{
  wip_channel_t CnxChannel;
  s32 ret = 0;
  CnxChannel = wip_SMTPClientCreateOpts(SMTP_STR_HOSTNAME,
                                        smtp_ClientTestCnxHandler,
                                        NULL,
                                        WIP_COPT_PEER_PORT, SMTP_CLIENT_TEST_
PORT,
                                        WIP_COPT_USER, SMTP_STR_USERNAME,
                                        WIP_COPT_PASSWORD, SMTP_STR_PASSWORD,
                                        WIP_COPT_SMTP_AUTH_TYPE, WIP_SMTP_
AUTH_MIME64,
                                        WIP_COPT_END);
  if (CnxChannel == NULL)
  {
    TRACE(( 1, "cannot create smtp session channel\n" ));
    return(-1);
  }
  else
  {
    u32 port, authtype;
    ascii **username, **password, **hostname;
    ret = wip_getOpts( CnxChannel,
                       WIP_COPT_ADDR, &hostname,
                       WIP_COPT_USER, &username,
                       WIP_COPT_PASSWORD, &password,
                       WIP_COPT_PEER_PORT, &port,
                       WIP_COPT_SMTP_AUTH_TYPE, &authtype,
                       WIP_COPT_END);
   wip_debug("Session wip_getOpts(): connect  %s:%d,  U=%s P=%s, %d\n",
              hostname, port, username, password, authtype);
    smtp_ClientTestCtx.CnxChannel  = CnxChannel;
  }
  return(ret);
}
static s32 smtp_ClientTestPutFile(void)
{
  wip_channel_t CnxChannel = (wip_channel_t)smtp_ClientTestCtx.CnxChannel;
  wip_channel_t DataChannel;
  s32 ret = 0;
  TRACE(( 4, "smtp_ClientTestPutFile\n" ));
    DataChannel = wip_putFileOpts(CnxChannel,
                                  NULL,
                                  smtp_ClientTestDataHandler,
                                  NULL,
                                  WIP_COPT_SMTP_SENDERNAME, SMTP_STR_SENDERNAME,
                                  WIP_COPT_SMTP_SENDER, SMTP_STR_SENDER,
                                  WIP_COPT_SMTP_REC, SMTP_STR_REC,
                                  WIP_COPT_SMTP_CC_REC, SMTP_STR_CCREC,
                                  WIP_COPT_SMTP_SUBJ, SMTP_STR_SUBJ,
                                  WIP_COPT_SMTP_FORMAT_HEADER, 1,
                                  WIP_COPT_END);
  if (DataChannel == NULL)
  {
    TRACE(( 1, "cannot create smtp data channel\n" ));
    return(-1);
  }
  else
  {
    ascii **sendername, **sender, **rec, **cc_rec, **bcc_rec, **subject;
    u32 FormatHeader;
    ret = wip_getOpts( DataChannel,
                       WIP_COPT_SMTP_SENDERNAME, &sendername,
                       WIP_COPT_SMTP_SENDER, &sender,
                       WIP_COPT_SMTP_REC, &rec,
                       WIP_COPT_SMTP_CC_REC, &cc_rec,
                       WIP_COPT_SMTP_BCC_REC, &bcc_rec,
                       WIP_COPT_SMTP_SUBJ, &subject,
                       WIP_COPT_SMTP_FORMAT_HEADER, &FormatHeader,
                       WIP_COPT_END);
    wip_debug("Data_getOpts(): sender is %s <%s>, %d, subject=%s\n",
              sendername, sender, FormatHeader, subject);
    wip_debug("rec=%s, cc_rec=%s, bcc_rec=%s\n", rec, cc_rec, bcc_rec);
    smtp_ClientTestCtx.DataChannel = DataChannel;
  }
  return(ret);
}
void evh_bearer( wip_bearer_t b, s8 event, void *ctx)//--
 {
(…)
      case  WIP_BEV_IP_CONNECTED:
            smtp_ClientTestCreate();
        break;
(…)
}

static void open_and_start_bearer( void) {…}
static bool poll_creg_callback(adl_atResponse_t *Rsp) {…}
static void poll_creg( u8 Id ) {…}
void SimHandler ( u8 Event ){…}
void adl_main ( adl_InitType_e InitType )
{
    TRACE (( 1, "Embedded Application : Main" ));
    wip_netInitOpts(WIP_NET_OPT_DEBUG_PORT, WIP_NET_DEBUG_PORT_TRACE,
          WIP_NET_OPT_END);
    adl_simSubscribe (SimHandler, PinCode );
}
```

Listing 2. Wysłanie wiadomości MMS

```
#include „adl_global.h”
#include “wip.h”
#include “wip_mms.h”
#include “mms_data.h”
void appli_entry_point();
void cfg_gprs ( void (* entry_point)(void));
#define MMS_APN_URL “http://mms.orange.pl”
#define MMS_APN_IP “192.168.6.104”
#define MMS_APN_PORT 8080
#define GPRS_APN      “mms”
#define GPRS_USER     “mms”
#define GPRS_PASSWORD “mms”
#define GPRS_PINCODE  “”
#define CREG_POLLING_PERIOD 20 /* in 100ms steps */
#define ASSERT( pred) \
if( !(pred)) wip_debug( “ASSERTION FAILURE line %i: “ #pred “\n”, __LINE__)
#define ASSERT_OK( v) ASSERT( 0 == (v))
const u16 wm_apmCustomStackSize = 4096;
static wip_mms_t p_mmsCtrl;
static wip_channel_t HTTPCnxChannel;
static void statuscallback(wip_mms_t mms, u32 status, void * ctx);
static void sendMMS(void);
void appli_entry_point();
static void HTTPCnxChannelFinalizer(void *ctx);
static void statuscallback(wip_mms_t mms, u32 status, void * ctx)
{
  wip_debug(“[MMS_SAMPLE] statuscallback\n”);

  wip_debug(“[MMS_SAMPLE] status: %d\n”, status);

  wip_debug(“[MMS_SAMPLE] Closing MMS Channel\n”);
  wip_mmsClose(p_mmsCtrl);
  p_mmsCtrl = NULL;

  wip_debug(“[MMS_SAMPLE] Closing HTTP Session Channel\n”);
  wip_close(HTTPCnxChannel);
  HTTPCnxChannel = NULL;
}
static void sendMMS(void)
{

  HTTPCnxChannel = wip_HTTPClientCreateOpts(
      NULL,
      NULL,
      WIP_COPT_HTTP_PROXY_STRADDR, MMS_APN_IP,
      WIP_COPT_HTTP_PROXY_PORT, MMS_APN_PORT,
      WIP_COPT_HTTP_VERSION, WIP_HTTP_VERSION_1_1,
      WIP_COPT_FINALIZER, HTTPCnxChannelFinalizer,
      WIP_COPT_END);
  if(!HTTPCnxChannel)
    wip_debug(“[MMS SAMPLE] Cannot create HTTP control channel\n”);
  else
  {
    wip_debug(“[MMS_SAMPLE] Sending MMS\n”);
    wip_mmsSend( p_mmsCtrl, HTTPCnxChannel, (u8*)MMS_APN_URL, 0 );
  }
}
#define MSG       „MMS tekst - czesc pierwsza”
#define MSG2      „MMS tekst - czesc druga”
#define MSG3      „MMS tekst - czesc trzecia”
#define PICTURE_NAME      “samp.jpg”
#define SOUND_NAME“samp.amr”
void appli_entry_point() {
  wip_debug( “[MMS_SAMPLE] Application started\n”);
  p_mmsCtrl = wip_mmsCreateOpts(
      WIP_COPT_MMS_TO_PHONE,”0668xxxxxx”, // numer odbiorcy
       WIP_COPT_MMS_STATUS, statuscallback, NULL,
      WIP_COPT_MMS_SUBJECT,”Sample_MMS”,
      WIP_COPT_MMS_SENDER_VISIBILITY,WIP_MMS_SENDER_SHOW,
      WIP_COPT_MMS_MESSAGE_CLASS,WIP_MMS_MESSAGE_PERSONAL,
      WIP_COPT_MMS_PRIORITY,WIP_MMS_PRIORITY_NORMAL,
      WIP_COPT_MMS_FROM,”0507xxxxxx”, // numer nadawcy
      WIP_COPT_END);
  /* Dodawanie różnych części wiadomości MMS */
  wip_mmsAddPart( p_mmsCtrl, (u8*)MSG, wm_strlen(MSG), WIP_COPT_MMS_PART_TEXT, WIP_COPT_END);
  wip_mmsAddPart( p_mmsCtrl, (u8*)MSG2, wm_strlen(MSG2), WIP_COPT_MMS_PART_TEXT, WIP_COPT_END);
  /* Dodanie obrazu */
  wip_mmsAddPart( p_mmsCtrl, image, sizeof(image), WIP_COPT_MMS_PART_JPG, PICTURE_NAME, wm_strlen(PICTURE_NAME), WIP_
COPT_END);
  /* Dodanie dźwięku */
  wip_mmsAddPart( p_mmsCtrl, sound, sizeof(sound), WIP_COPT_MMS_PART_AMR, SOUND_NAME, wm_strlen(SOUND_NAME), WIP_COPT_
END);
  wip_mmsAddPart( p_mmsCtrl, (u8*)MSG3, wm_strlen(MSG3), WIP_COPT_MMS_PART_TEXT, WIP_COPT_END);
  sendMMS();
}
static void HTTPCnxChannelFinalizer(void *ctx)
{
  wip_debug(“[MMS_SAMPLE] Program terminated\n”);
}

static void evh_bearer( wip_bearer_t b, s8 event, void *ctx) {
  if( WIP_BEV_IP_CONNECTED == event) { appli_entry_point(); }
}
static void open_and_start_bearer( void) {…}
static void poll_creg( u8 Id ) {…}
static void evh_sim( u8 event) {…}
void adl_main ( adl_InitType_e InitType )
{
  TRACE (( 1, “Embedded Application : Main” ));
  wip_netInitOpts( WIP_NET_OPT_DEBUG_PORT, WIP_NET_DEBUG_PORT_TRACE, WIP_NET_OPT_END);
  adl_simSubscribe( evh_sim, GPRS_PINCODE);

}
```