Bidirectional GPRS connection with Telit GM862-PY

Tom Hannelius
Research assistant
TUT/ACI

TABLE OF CONTENTS

1. OVERVIEW

This document was written based on a research project done for the Institute of Automation and Control (ACI) at Tampere University of Technology (TUT) in 2006. The goal of the project was to update ACI's SMS controlled Metso ND800 valve application with a bidirectional GPRS connection using Telit's EZ10 with GM862-PY wireless GSM/GPRS modem. Interactive control was to be achieved without the need for initializing the connection from the modem's side. As a result the valve is now controlled locally through an operator panel and remotely by SMS messaging and HTTP requests (web browser) via the software developed in Python residing in the GM862-PY.

The document starts off with at look at the system setup in chapter 3. A look is taken at the devices that the application builds on.

Chapter 4 provides a description of the network environment that the application functions in.

Chapter 5 digs into the brains of the application, the software which is written in Python. Use case, class and sequence diagrams are presented to give an insight into the software functionality.

An operator's manual is provided in chapter 6. We take a look at how the application is operated and go through the process of developing applications and uploading them to the modem.

Chapter 7 lists some personal thoughts and ideas that arose during the project.

This document is mainly directed to ACI personnel operating the application. This said though, general instructions and troubleshooting that help develop software for Telit platforms are provided.

2. TERMS AND DEFINITIONS

AJAX	Asynchronous JavaScript and XML
EZ10	The GM862-PY modem + board, housing, interfaces (power, antenna, SIM, RS232, RJ11) and status leds
GM862-PY	The GSM/GPRS cellular modem with Python engine
GPIO	General purpose input output
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HTTP	Hypertext Transfer Protocol
NVM	Non-volatile Memory
PLC	Programmable logic controller
RAM	Random Access Memory
SMS	Short message service

3. SYSTEM SET UP

3.1 Local environment

Figure 1 points out the devices in the local environment. Updates to the earlier environment [Sim] include the EZ10, the Nokeval 7470 and the switch. The switch gives the option to control the valve manually through the PLC or remotely by SMS messaging and HTTP requests through the EZ10.

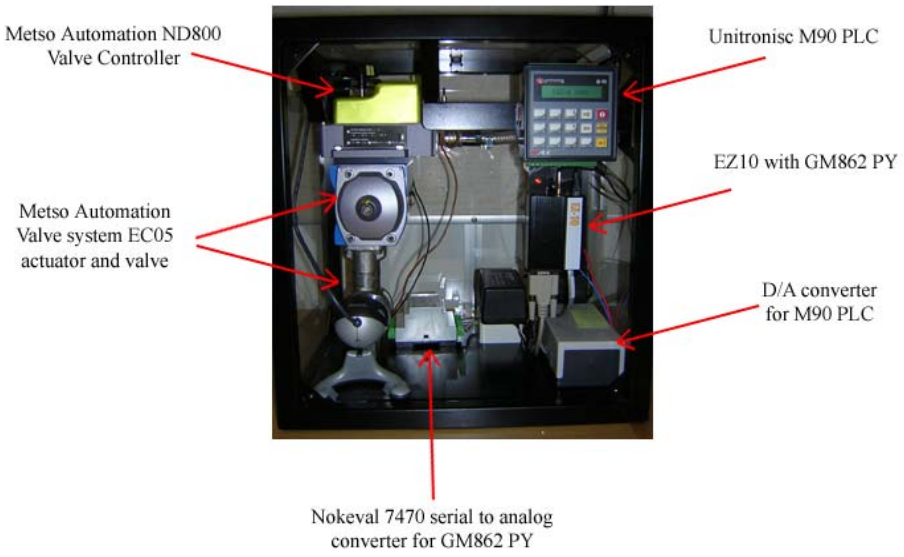

Figure 1. Devices in the local environment

3.2 PLC, valve controller and valve system

The Metso ND800 valve controller and Metso Automation valve system EC05 (actuator and valve) build up the process that is controlled locally or remotely. The process and local control functionalities are discussed in [Sim]. Remote control is covered in this document.

3.3 Nokeval 7470 serial to analog converter

The Nokeval 7470 is a serial bus controlled analog output unit. It provides four mA or V outputs that can be controlled by an RS232 or RS485 bus.

Before initial use the 7470 must be configured. This is done easily with a PC or a hand-held programmer 6790. In the case of PC configuration, the PC it is connected to the 7470's RS232 or RS485 bus or to the 3,5mm POL connector located on the front panel. A program called MekuWin is used to configure the converter. Another useful software Sicala is used to test the connection and configuration. For information on the hand-held programmer and more information on the 7470 please refer to [Nokeval]. For information on configuring and testing the 7470 please refer to [MekuWin] and [Sicala].

3.3.1 Nokeval 7470 in the application

The 7470 is connected between the EZ10 and the Metso ND800 valve controller to provide a conversion of serial data to an analog message. An RS232 bus using RX, COM and TX pins is used between the EZ10 and the 7470 (figure 2).

Figure 2. Nokeval 7470 serial input from EZ10 and power input

Two wires are used to connect the 7470 through the switch to the Metso ND800 (figure 3 and figure 4). In addition to these connections the 7470 requires an operating voltage of 24Vdc.

Figure 3. Nokeval 7470 output to the valve controller

Figure 4. Valve controller input from Nokeval 7470

A message from 0 to 100 is constructed in the Python script based on user input. The 7470 converts the serial data to a standard current message of 4mA to 20mA. 0 corresponds to 4 mA and 100 to 20 mA. The Metso ND800 controls the actuator based on the provided current and eventually the actuator turns the valve to the desired position.

An RS232 – POL cable was used to configure and an RS232 to test the 7470. These cables are marked as seen in figure 5.

Figure 5. Nokeval 7470 configuration and test cables

3.4 EZ10 with GM862-PY

The GM862-PY is a telecommunications module that integrates a Python engine allowing self controlled operations. The GM862-PY includes features such as GPRS (class 10), voice, circuit switched data transfer, fax, phone book, SMS and on board SIM reader.

The EZ10 is a ready for use subsystem for wireless communication. It provides a board for the GM862-PY and the external interfaces shown in figure 6.

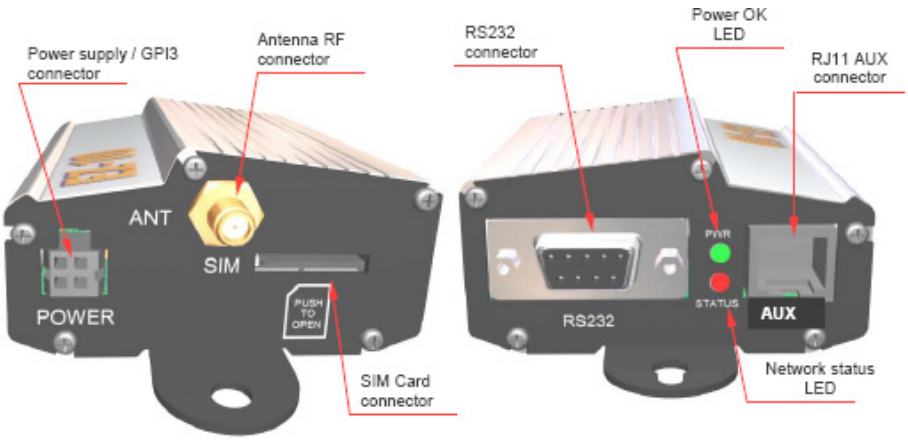

Figure 6. EZ10 interfaces [M2MDesc]

3.4.1 Telit Easy Script

The Telit Easy Script feature allows driving the GM862-PY module internally by scripts written Python, a high level programming language. After writing a Python script, a developer uploads the script to the module and sets it as executable. Starting the module in script mode will cause the script to run. (See ch. 6.2 Command/script mode and 6.3 Writing, uploading and executing scripts). The Python version is 1.5.2+ so built-in types, functions and libraries are restricted to this version.

The module has an NVM of 3MB. This space is allocated for user scripts and other files. The file system in the module allows a user to read and write files on a single level. No subdirectories are supported. A RAM of 1.5MB is reserved for the Python engine's usage.

In addition to Python's libraries six libraries are available to establish interaction with the GM862-PY. A short description of these libraries is provided in table 1. For more please refer to [M2MEasy].

MDM	Interface to the AT command parser engine
SER	Interface to the serial port
GPIO	Interface to the handle the internal general purpose input output
MOD	Miscellaneous functions
IIC	Interface to the IIC bus master
SPI	Interface to the SPI bus master

Table 1. Internal libraries

3.4.2 EZ10 configuration

The EZ10's serial port is used to connect the module to the valve controller through the Nokeval 7470 (figure 7). The serial port is controlled by the script through the SER interface.

A led is connected to the GPIO and controlled by the script through the GPIO interface. The led provides status info (figure 7).

Figure 7. EZ10 at work

4. NETWORK

4.1 Physical topology

The physical topology of the environment is pictured in figure 8.

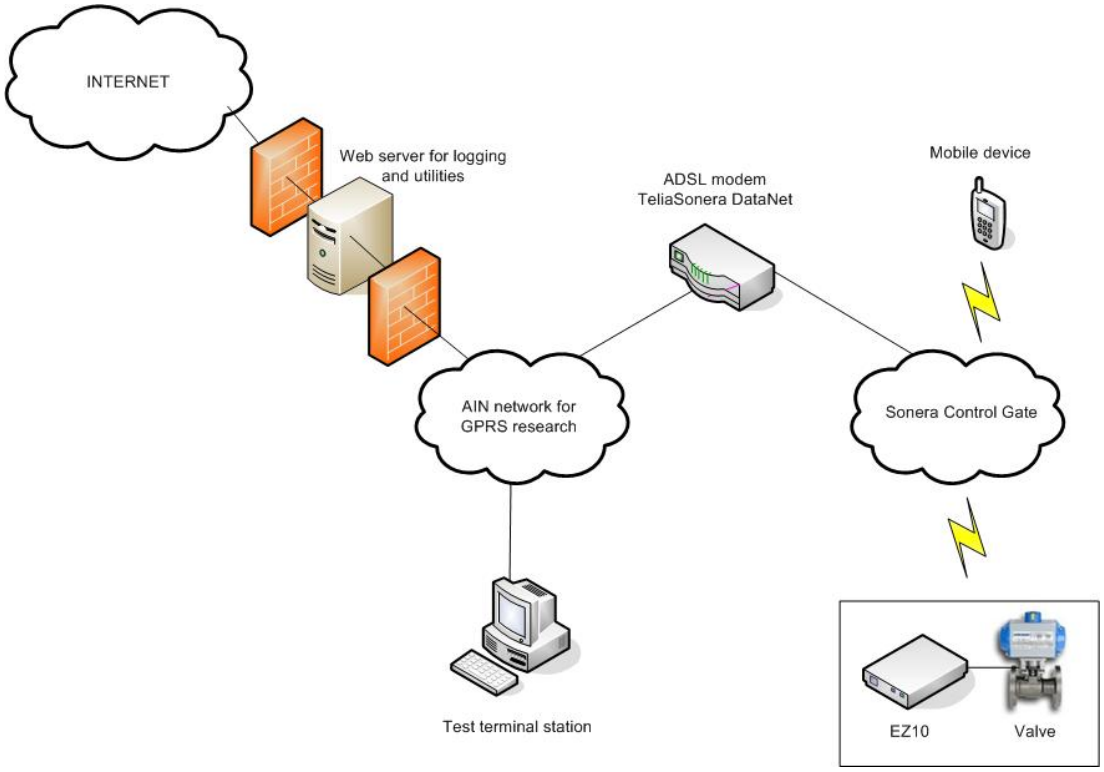

Figure 8. Physical topology

The EZ10 with GM862-PY is connected to the process. A PIN card provided by network service provider Sonera is inserted in to the module. The PIN card provides Sonera's Control Gate service which, in addition to basic GSM functionality, enables bidirectional machine-to-machine communication in GPRS and data networks.

For a more detailed explanation of the network please refer to [Ruoh].

4.2 SMS messaging

SMS messages are sent to and from the module according to basic SMS messaging rules. No restrictions on the sender's phone number are implemented. (See ch. 7 Own notes for implementing restrictions on the sender's phone number.)

4.3 HTTP communication

HTTP requests can be sent to the module with PC's connected to the AIN (Automation and Information Networks) network for GPRS research. The process is not accessible with computers outside this network. Requests are denied by a firewall separating the research network from the outside internet. In addition, the GM862-PY implements a firewall that can be configured with AT commands to allow or disallow certain IP addresses.

5. SOFTWARE

5.1 Use cases

The use case diagram in figure 9 shows all the possible actors and use cases in the application. A remote user is a person who sends control signals to the module by SMS messaging or HTTP requests and can read a log file on the web. A remote user does not have direct access to the process. A local user is a person who has direct access to the process.

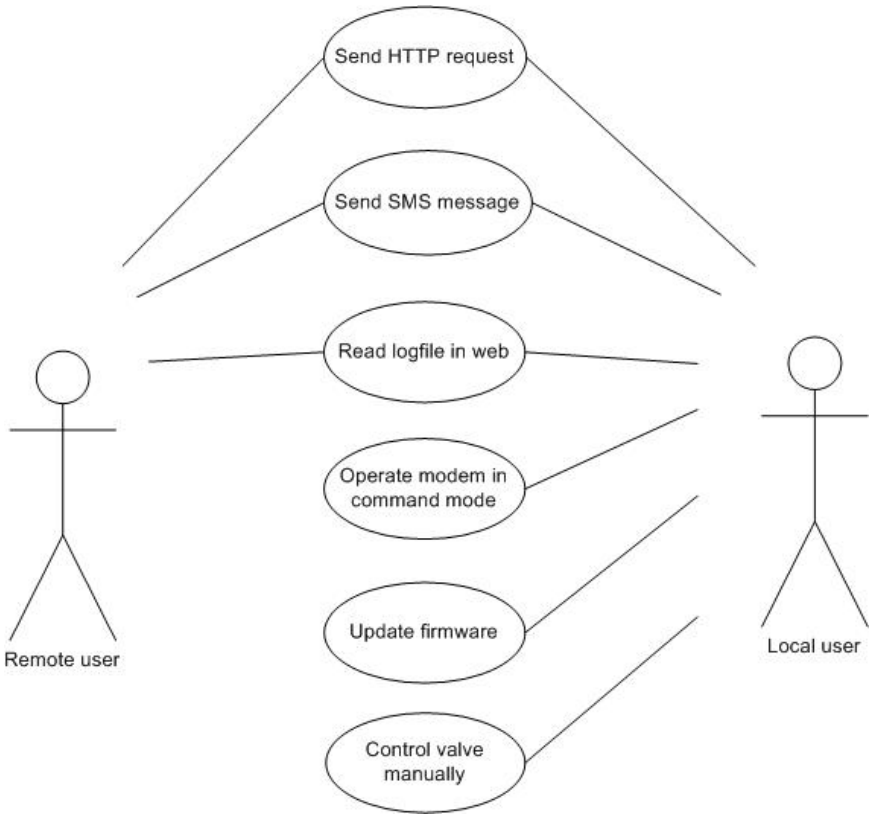

Figure 9. Use case diagram

5.2 Classes

The Python software consists of the classes seen in the class diagram (figure 10). In addition to these classes, three Python scripts are included in the application: timers.py, utils.py and mainA.py. timers.py offers methods for creating and handling timers. utils.py offers general purpose methods for logging, setting the real time clock (RTC), connecting to a server, sending a signal to the signal converter etc. mainA.py contains the Python idiom called by the Python engine at start up.

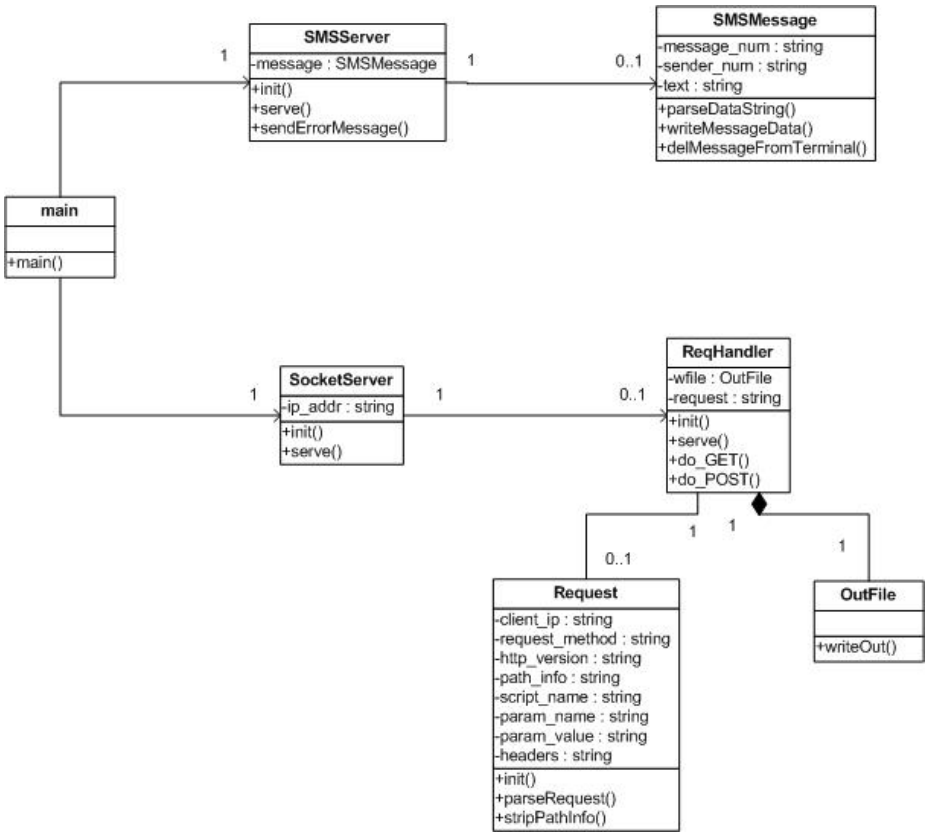

Figure 10. The main classes in the application

After initializations at start up, the main method starts to listen to the AT interface and delegates incoming SMS message indications to an SMSServer object and HTTP requests to a SocketServer object. These objects handle the data they receive and respond in an appropriate manner.

5.3 Sequences

5.3.1 Handling an SMS message

The sequence diagram in figure 11 describes what happens when the module receives an SMS message.

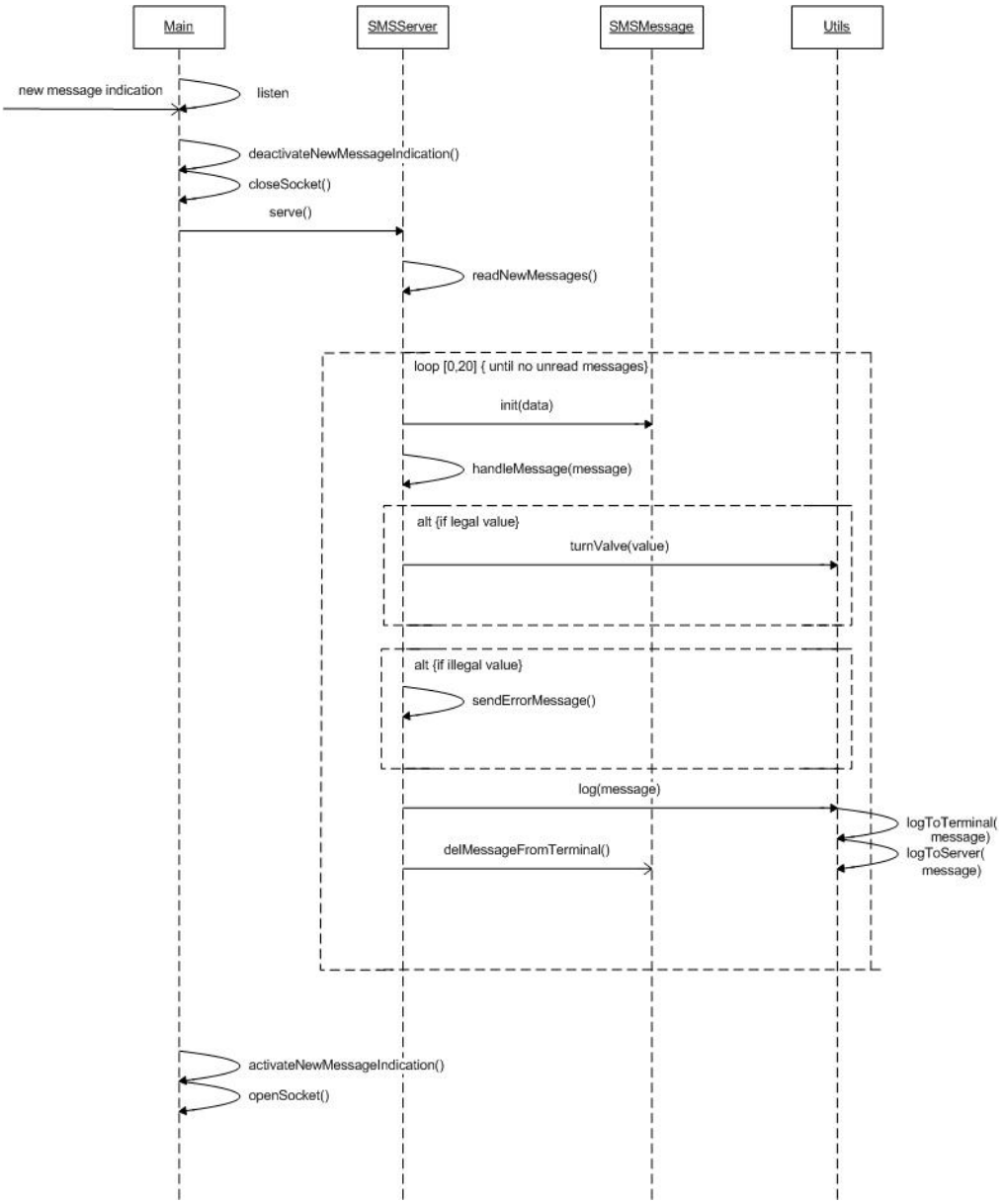

Figure 11. Handling an SMS message

The main method is listening to the AT interface for new SMS message indications and HTTP requests. When it detects a new SMS message indication, it first deactivates the indication of new SMS messages and closes the listening socket. These are done because possible new SMS message indications and HTTP requests arriving at the AT interface might cause unwanted behaviour in the script while it is processing an SMS message.

The indication of new SMS messages is detected at the AT interface but no message data is transferred during this process. SMSServer asks for the unread messages' data with readNewMessages(). The result of this function call can contain one to twenty new units of SMS message data (20 is the maximum amount of SMS messages that fit in to the SIM card). Each message is parsed and saved in to an SMSMessage object and based on the message's data the valve is turned or an error message is sent back to the sender's number.

After handling the SMS message a log message is written to a log file in the module and on a remote server. Log information can thus be read during execution from a predefined address. The log file in the module can only be read in command mode and thus can not be viewed while the application is running.

After handling the message/messages, new message indication is turned back on and the socket is opened in listen.

5.3.2 Handling an HTTP request

The sequence diagram in figure 12 describes what happens when the module receives a valid HTTP POST request.

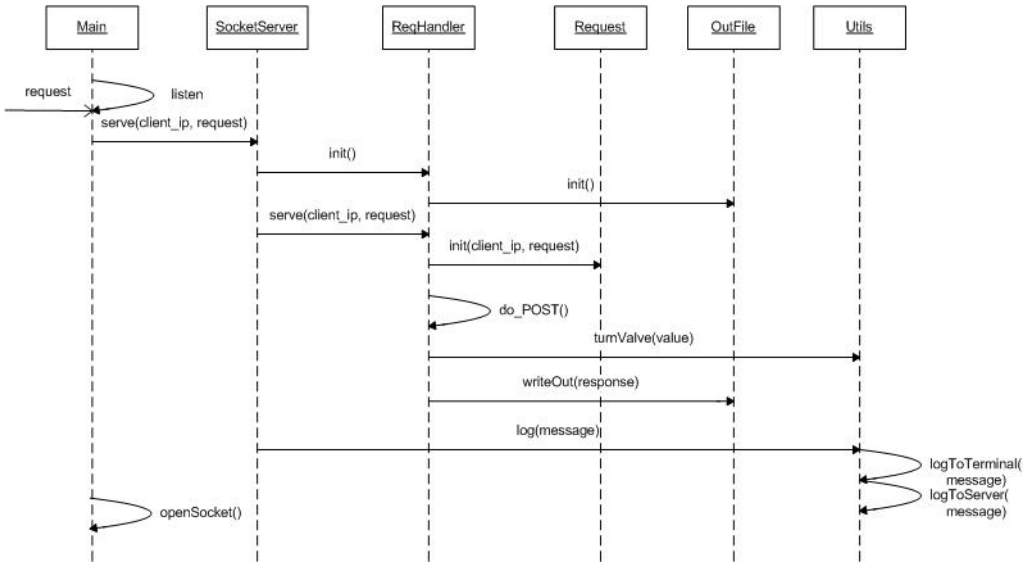

Figure 12. Handling a valid HTTP POST request

The main method is listening to the AT interface for new SMS message indications and HTTP requests. After receiving a new HTTP request it calls the SocketServer's serve() method with the client's IP address and the HTTP request, both parsed from the initial HTTP request read from the AT interface.

The SocketServer creates a request handler (ReqHandler) which in turn creates an OutFile. OutFile is responsible for writing responses out to the client through the AT interface. ReqHandler parses the request and saves the data in a Request object. If the request was valid the valve is turned and a response is written to the client by calling OutFile's method writeOut.

After responding to a request and closing the connection, a log message is written to a log file in the module and on a remote server. Log information can thus be read during execution from a predefined address. The log file in the module can only be read in command mode and thus can not be viewed while the application is running.

5.4 AT commands

AT commands (Hayes Command Set) are used to communicate with the module. The module performs different actions based on the AT commands provided to it. Telit's modules support different sets of AT commands. When working with Telit's modules, make sure that you are referring to the AT Commands Reference Guide that corresponds to your module's version. Reference guides can be downloaded from www.m2m-platforms.com

Figure 13 shows a diagram describing the AT commands sent from the Python script to the module's AT interface during initialization.

For a complete list of AT commands supported by the GM862-PY and more information on the AT commands in figure 13, please refer to [M2MAT].

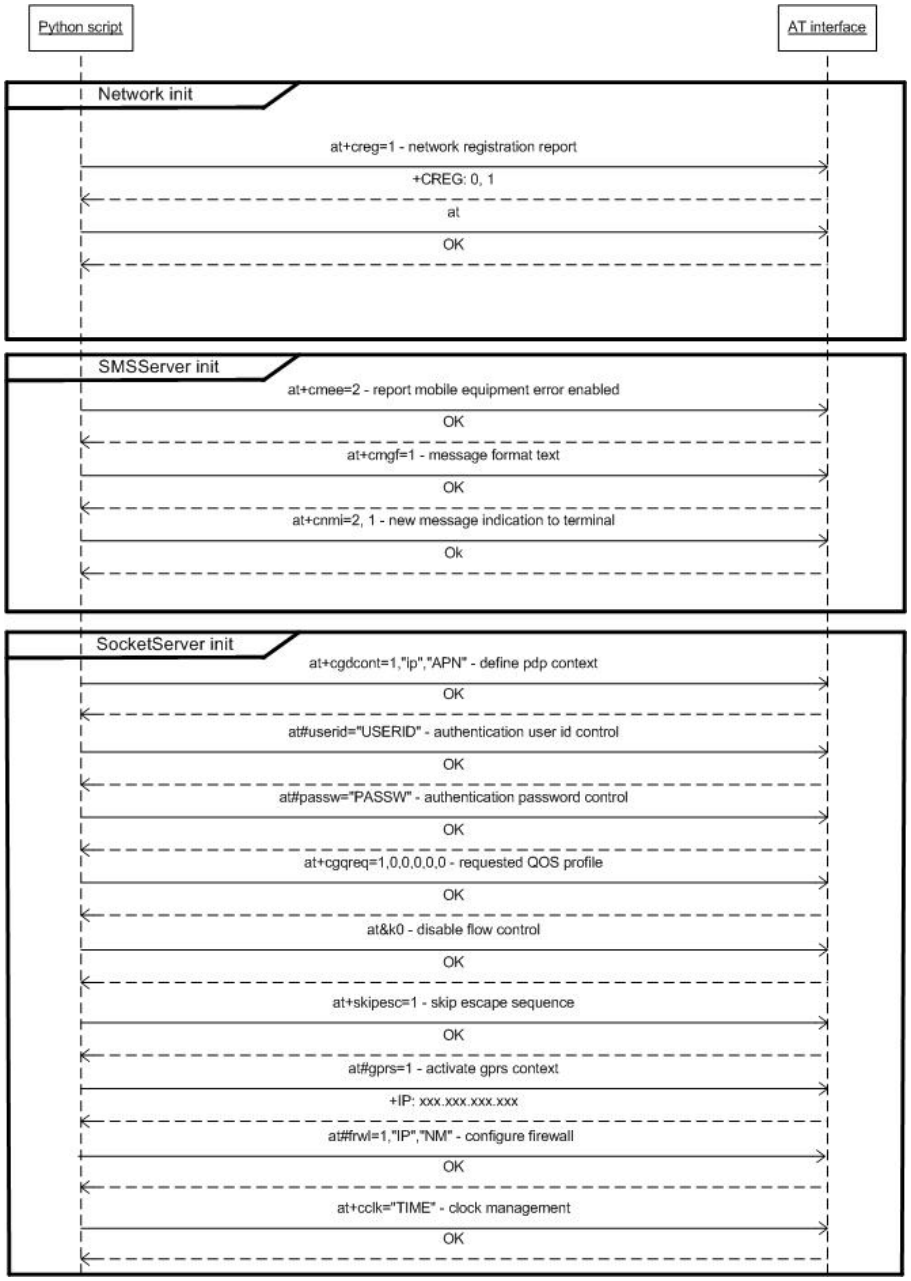

Figure 13. AT commands sent during initialization

6. OPERATOR'S MANUAL

6.1 Using the application

6.1.1 Starting up the application

Make sure that all the power cables except the EZ10's are plugged in. Make sure that the pressure hose is plugged in. Make sure that a PIN card is inserted. Start the module is script mode (see ch. 6.2 Command/script mode). When the application has ran all the initialization scripts the red led connected to the module's GPIO port will fire. If the red light does not fire or fires and soon after dies please refer to chapter 6.4 Troubleshooting.

Once powered up in script mode the Python engine runs the script that was set as the executable script. After initializations you will see log messages starting to flow to *public server address*. Among other things you will find the module's IP address there. The IP address is needed for connecting to the module via HTTP requests (figure 16).

Once up and running the valve can be controlled by SMS messaging or HTTP requests.

6.1.2 Sending an SMS message to the module

To send an SMS message you need to know the number of the PIN card inserted in to the module. The syntax of a valid message is shown in figure 14.

<text> <number>

Figure 14. The syntax of a valid SMS message

<text> is the not case sensitive word "valve" and <number> is an integer between 0 and 100. Zero or more whitespaces can be in front and/or behind the message. One or more whitespaces has to be between <text> and <number>. An example of a valid SMS message is show in figure 15.

Figure 15. A valid SMS message

After sending an SMS message to the module the red led will flicker once. The Python script will start processing the message and turn the valve if the message was valid or send an error message back to the sender if the message was invalid. If the led does not flicker, the valve is not turned and an error message is not sent back or if you have any other problems concerning SMS messaging, please refer to chapter 6.4 Troubleshooting.

6.1.3 Connecting to the module using an HTTP client

The Python script implements a basic web server capable of handling HTTP GET and POST requests. The functionality of the web server is restricted to GET and POST requests with a certain syntax. Any requests that differ from this syntax are responded to with an HTTP error code and an error message. The connection can be achieved with any HTTP capable client.

To connect to the module you must know the module's IP address. The network service provider (NSP) provides the module an IP address dynamically at GPRS context activation (application start up) from the range *"private ip address"* – *"private ip address"*. The allocated IP address can be read after initializations from *"public server address"*.

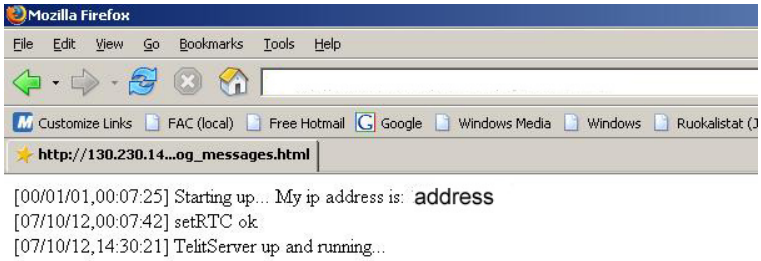

Figure 16. Log page

After typing in the IP address (and only the IP address) to your web browser's address bar and hitting enter a GET request is sent to the module. If the request is valid the Python script will respond with the index.html page (figure 17). No other html pages reside on the module so do not go asking for anything else.

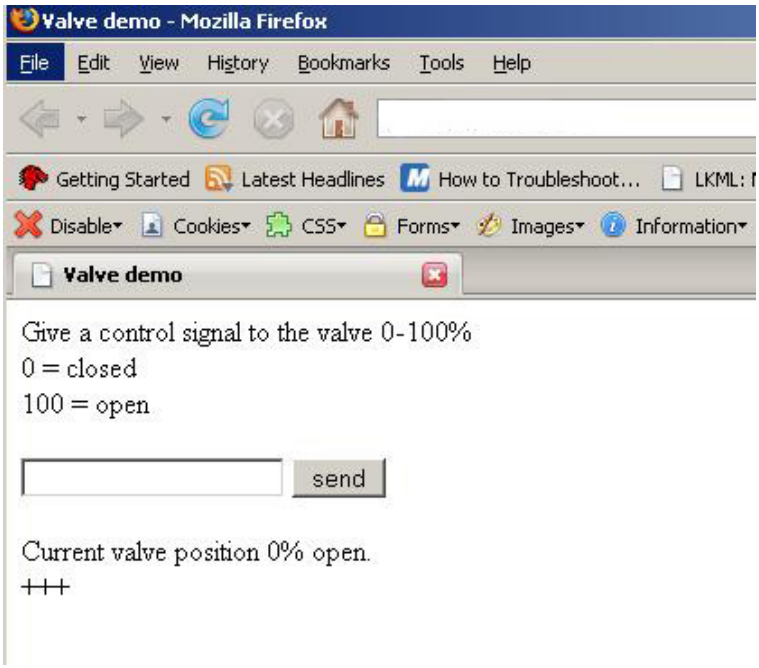

Figure 17. Main page

The index.html page consists of a static HTML part containing a simple form and a dynamic JavaScript part containing the functions that talk to the server. This construction combined with using XML to facilitate the sharing of data is referred to as AJAX (Asynchronous JavaScript and XML).

Applying input to the form and clicking the send button will cause a POST request to be sent to the module. The payload of the POST request will contain the XML data containing the control signal provided as input to the form.

```
<?xml version="1.0"?>
<request>
100
</request>
```

Figure 18. XML message containing valid user input

If the input is valid the valve will be turned to the desired position and an XML response will be returned containing the current valve position.

```
<?xml version="1.0"?>
<response>
Current valve position: 100% open.
</response>
```

Figure 19. XML message containing response

The dynamic part of the index.html page will parse the XML and update the current valve position on the static part (figure 20).

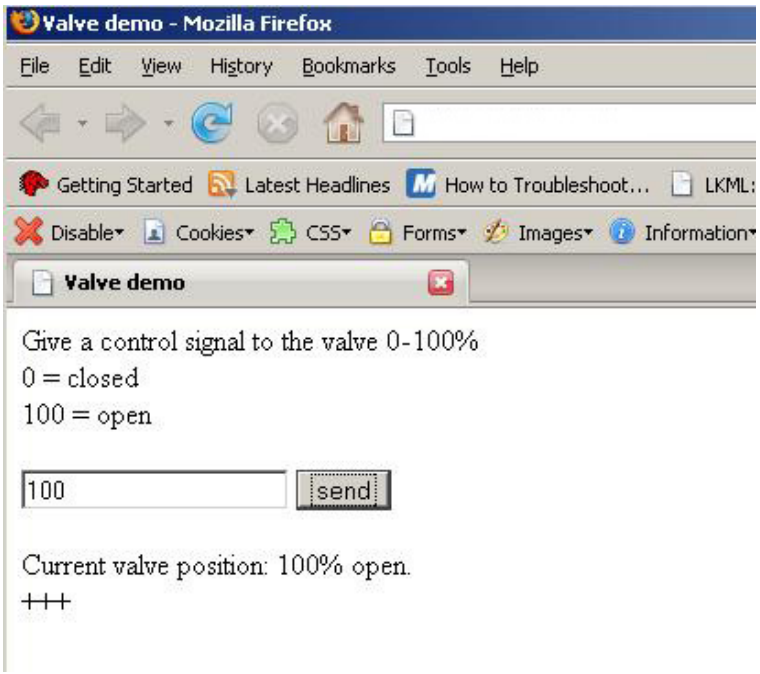

Figure 20. Main page after valid input

If an invalid input is provided, the module will respond with an XML message containing an error message.

```
<?xml version="1.0"?>
<response>
Error: invalid control signal.
</response>
```

Figure 21. XML message containing error message

Once again the dynamic part will parse the XML and update the error message on the static part (figure 22).

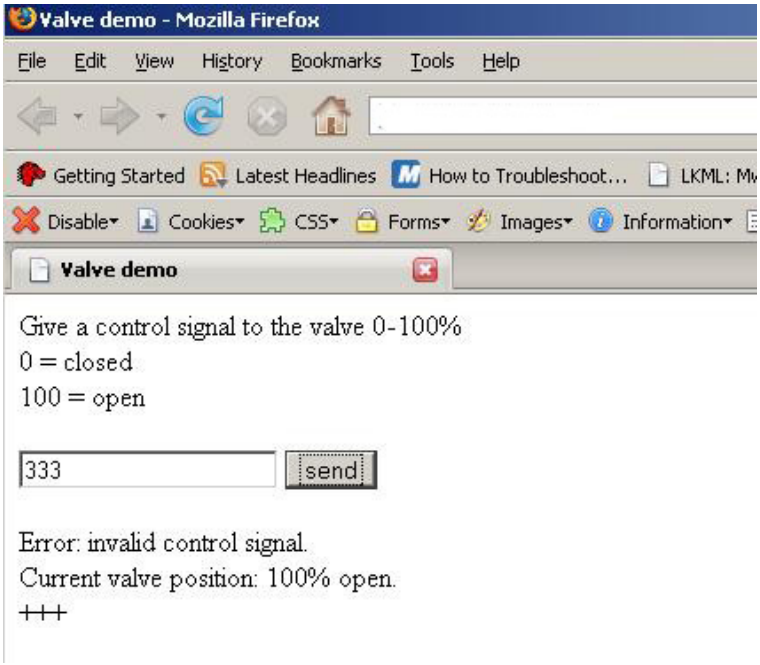

Figure 22. Main page after invalid input

For solving any problems dealing with connecting to the module or sending HTTP requests please refer to chapter 6.4 Troubleshooting.

6.2 Command / script mode

The module can be operated in two modes: command and script. When in command mode the module responds to user input provided with a data terminal application on a PC connected to the modem's RS232 interface. A terminal application that I found very useful, Round Solutions GSM terminal or RSTerm, can be found at www.roundsolutions.com/techdocs. Hyperterminal is another application that suites as well.

When in script mode, the module executes the script that has been set as the executable script.

6.2.1 Starting up in command mode

If the module detects the RS232 DTR pin being high (0V, RS232 cable connected and connection opened) as it powers up it will start in command mode.

To start in command mode:
1. connect your PC to the module with an RS232 cable, make sure the module's power cable is disconnected
2. open a connection to the connected COM port with your terminal application
3. power up the module
4. test the connection by typing in AT, if everything works normally OK should be printed to your terminal application's window

6.2.2 Starting up in script mode

If the module detects the RS232 DTR pin being low (2,8V, connection closed and/or the RS232 cable unconnected) as it powers up it will start in script mode.

To start in script mode:
1. make sure that the connection between your PC and the module is closed and/or the RS232 cable unconnected
2. power up the module

6.3 Writing, uploading and executing scripts

Any text editor can be used to write a Python script. All you have to remember is to save the file with a .py extension. This said though, there are several IDEs that provide an easier way to develop Python software.

The Python version in the GM862-PY is 1.5.2+ so developing software for the module based on a newer Python version makes no sense. The PythonWin IDE combined with TelitPython 1.5.2+v4.1 is an obvious choice. A packet that includes both of these can be downloaded from www.m2m-platforms.com

Once a Python script is written it must be compiled before it is uploaded to the module. Version 4.1 of TelitPython provides an easy way to do this. Just right click on your file in your file manager and choose compile (figure 23).

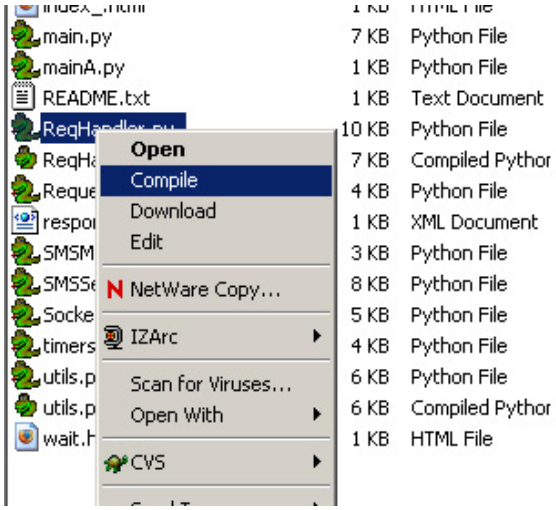

Figure 23. Compile Python file

This option compiles the file and saves it in the same directory with a .pyo extension. Compilation results and possible errors will appear in the active command prompt window. This new option makes software development for the module much faster (see ch. 7 Own notes).

Once your script is compiled it is time to upload it to the module. Use a terminal application i.e. RSTerm and start the module in command mode (figure 24). Click the Telit Python view open (1.), navigate to your working folder from the left (2.), choose your compiled file in “Files in PC folder” window (3.) and click “Upload selected file(s) from PC to folder” (4.).

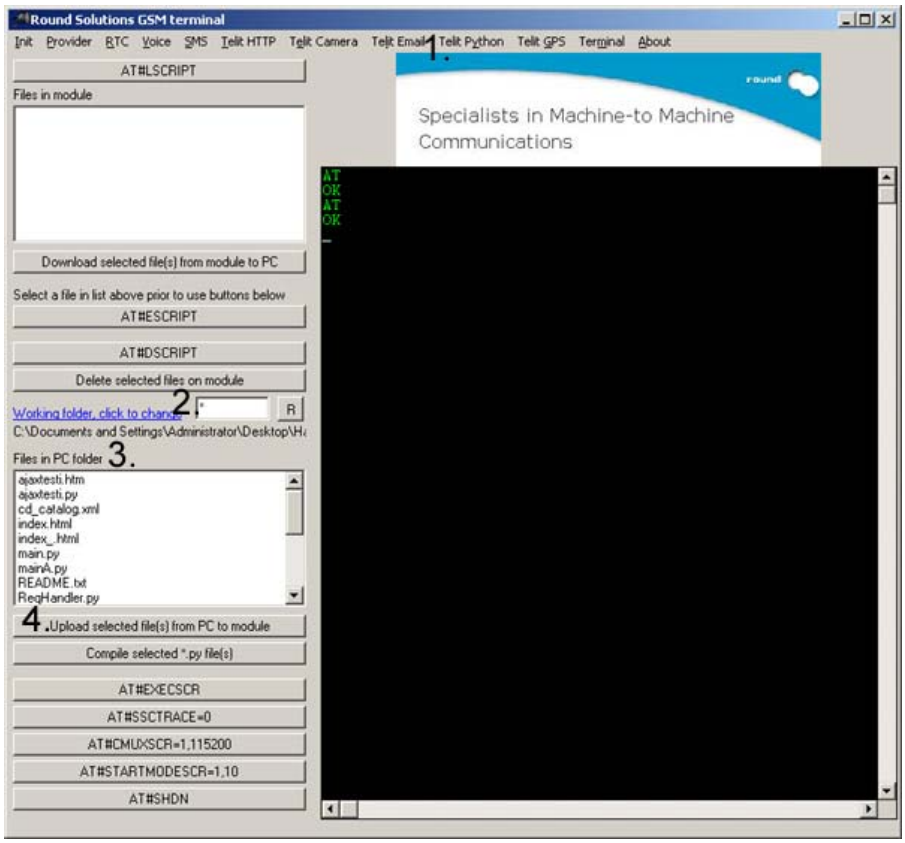

Figure 24. Uploading a file

To make your uploaded application run on start up you must set the file containing the main method called by the Python engine as the executable script. This file must not be compiled on your PC (see ch. 7 Own notes). Select the file from the “Files in module” window (1.) and click AT#ESCRIPT (2.). Your script is now set to run the next time the module starts up in script mode (figure 25).

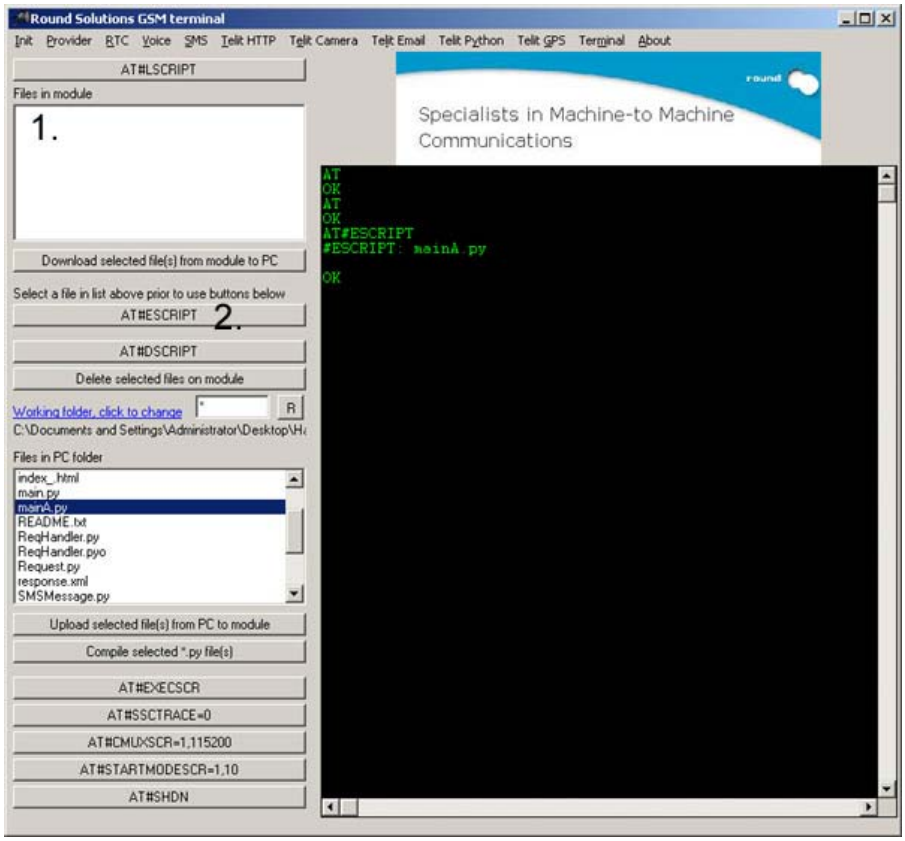

Figure 25. Setting a script as executable

For problems concerning writing, uploading and executing scripts please refer to chapter 6.4 Troubleshooting.

6.4 Troubleshooting

6.4.1 Reading the inner log file

Start up in command mode and type AT#RSCRIPT=”telitServer.log”. The contents of the file will be printed on your terminal software’s screen.

6.4.2 Red led

If the red led does not fire in one minute after starting up with the valve application set as the executable, make sure that you started the module in script mode.

If the red light fires but soon after dies this usually means that no service is provided by the network. Make sure that the PIN card is inserted correctly and try powering up the module again in script mode. If you have problems with start up please refer to the inner log file (see ch. 6.4.1 Reading the inner log file.) for error messages.

6.4.3 SMS messaging

The SIM card can hold a maximum of 20 SMS messages. If the maximum is reached and new messages are sent to the module an indication in the AT interface will not be seen and asking for unread messages will not result in new message data. This is essential if the preferred message storage location is set as the SIM card.

6.4.4 HTTP communication

Only one socket can be opened at a time. Multiple socket connections can not be made.

Only one connection request can be accepted at a time.

6.4.5 Writing, uploading and executing scripts

If the name of a Python script has more than 12 characters, it can not be uploaded to the module.

7. OWN NOTES

Currently the software allows anyone to control the valve through SMS messaging. It does not implement an allow list for phone numbers that are authorized to control the valve. An allow list can and should be implemented and used when parsing the incoming SMS message in class SMSServer.

Occasionally after receiving the first valid control signal through SMS or HTTP after system start up and sending serial data to the 7470, no action is seen. It is not clear, where the point of failure is. The second and proceeding control signals have the desired effect.

It would be convenient to have a led on the EZ10 that tells the mode (command/script) the module is in. A soft reset button would also be convenient to enable easy switching between command and script mode.

The software was developed with Telit's PythonWin v3.1. This version had some restrictions that resulted in a significantly longer time to develop software for the module. Scripts could not be compiled on the PC. They had to first be uploaded to the module and compiled on the module. Compiling a script on the module takes from a few seconds to several minutes depending on the script's size. Telit has released a new version of PythonWin. V4.1 reduces compilation time by enabling compilation on the PC. It also improves some other Python features. USE PYTHONWIN 4.1!!!

Use the latest Firmware. Check versions from www.m2m-platforms.com

The script that is set as executable must always be a .py file. Do not compile the file that you set as executable on your PC. If your application consists of several Python files create a file that only contains a Python idiom that calls the application logic and set this file as executable.

8. REFERENCES

[M2MAT] GM862-PY AT Commands Description
http://www.m2m-platforms.com/data/80000ST10025a_AT_Commands_Reference_Guide_r3.pdf [Read 17.10.2007]

[M2MDesc] Telit GM862-PYTHON Product Description
http://www.m2m-platforms.com/data/80269st10024a_EZ10-QUAD-PY_Prod_Descr_r0.pdf [Read 17.10.2007]

[M2MEasy] Easy Script in Python
http://www.m2m-platforms.com/data/Python%20Easy%20Script_Manual.pdf [Read 17.10.2007]

[MekuWin] Configuration software
http://www.nokeval.com/english/products/software/index.html [Read 17.10.2007]

[Nokeval] User manual
http://www.nokeval.com/english/pdf/Manual/7470manual.pdf [Read 17.10.2007]

[Ruoh] Timo Ruohonen, Bidirectional GPRS in Machine-to-Machine Applications, Master of Science Thesis, April 2004

[Sicala] Test software
http://www.nokeval.com/english/products/software/index.html [Read 17.10.2007]

[Sim] Report of Valve demo application, Petr Simacek, TUT/ACI