Table of contents

1) Introduction

What is message2action?
The program processes messages like SMS and mails and reacts on the content of the message. Sometimes this functionality is called "server control".
It is a program running on Raspberry Pi (following simply "Pi") or better Raspian and is developed in programming language C. The program was never tested on other platforms.
In the message you send to your Pi you can write commands like "stop" or "start" and you can specify in the configuration file (ini-file) of message2action what should be done when the message has been received by your Pi.
General spoken message2action uses a file-based interface. Or in other words: it reads all files in specified directories and processes the files. It is not an online-program.
The idea why I started to develop the program was my wish to start my PC without having it connected to the internet. So I wanted to send a SMS to my Pi with a command, which tells the Pi to send WOL (wake on LAN) to my PC. To do with message2action does avoid the problem that I don't need dyndns and portforwarding in my router or something like that. And my PC is still not visible from the internet.
With this idea starting I thought it would be great to have a general interface running on a Pi. The interface can be used to start anything in my network like my PC, to ask for a status of any device in my network like temperature sensors and so on.

2) Features

Here is a list of most important features:
-reads text files (mails, SMS and Telegram) and processes the commands in the text files by reading keywords in the message
-creates text files (mails, SMS and Telegram) when an answer should be send back (for example the result of a status-request)
-level of information (information, warnings and errors) can be specified what is saved in a log file
-maximum size of log file can by specified

-the text files for mails, SMS and Telegram are kept on your Pi and can be used together with the log file for analyzing purposes
-maximum 10 mail addresses, 10 cell phone-numbers and 10 chat-IDs for Telegram can be specified as valid contacts
-warning messages will be send out when a message comes from an invalid sender (mail, SMS and Telegram)
-maximum 10 commands (programs, scripts) can be specified to be executed by sending keywords CMD00 ... CMD09 in a message
-internal commands are available to stop program (message2action), reboot and shutdown your Pi by sending keywords in a message

3) Requirements

What do you need, what are the requirements to use message2action?

3.1) SMS

You need a surfstick which can be connected to your Pi with USB. For the surfstick you need a SIM-card which is able to receive and send SMS.
Currently I use the "SMS Server Tools 3" (http://smstools3.kekekasvi.com) for sending and receiving the SMS with a surfstick. The software can be installed with

```
sudo apt-get install smstools
```

I've connected a Vodafone surfstick called K3-772Z with a prepaid SIM card from T-Mobile with the Pi. Note that I had to wipe out the SIM-lock of the surfstick first in order to use the SIM-card from T-Mobile.
In order to switch the surfstick from data-mode to modem-mode you can use USB Modeswitch (see http://www.draisberghof.de/usb_modeswitch). You should install the data-package, too. Both can be installed with

```
sudo apt-get install usb-modeswitch usb-modeswitch-data
```

The mentioned "SMS Server Tools" save the received SMS in files. The default-directory is /var/spool/sms/incoming. message2action reads all files in this directory and searches for commands in the files. After reading the files are moved to directory for processed files.
When message2action wants to send messages as SMS the files are saved in the outgoing directory (default: /var/spool/sms/outgoing).

3.2) Mails

You need an account at a mail-provider. This account is the point of contact to your Pi, let's call it Pi-account. This means that you send from another account (let's call this one sender-account) a mail to the Pi-account. In order to send mails to your Pi you have to send mails from your sender-account to the Pi-account. And you will get mails from the Pi-account send to your sender-account, too. I use for receiving mails from the Pi-account fetchmail and for sending out mails I use ssmtp.
fetchmail saves the received mails in files. You can specify the directory were fetchmail should save the mails. message2action reads all files in this directory and searches for commands in the files. After reading the files are moved to directory for processed files. Use

```
sudo apt-get install fetchmail
```

in order to install fetchmail.
In order to send mails out message2action creates a text file which is used as an input for a shell-script which is called by sendmails.service (this file is provided by me). The shell-script calls ssmtp for sending the mail from the Pi-account to the receiver-account. Use

```
sudo apt-get install ssmtp
```

in order to install ssmtp.

3.3) Telegram

You need either the app for Telegram or you use a web-browser in order to log into Telegram.
You need a Telegram account and a bot first of all. Concerning your bot you need the token which is related with your bot. The token looks like this: 380096123:ABGxGlaeQTbUR_fWZFG2RT8u6MhABCxLwmw
You need to know the chat id of the chat you are using to communicate with your bot. The chat id is 9 digit number. Please search in internet for tutorials in order to find out your chat id. Configure the chat id (or several chat ids if you have) in the ini file.
The sending of messages to your pi is done by curl. If curl is not installed, use

```
sudo apt-get install curl
```

in order to install it.

In order to send messages to your chat a shell script send_telegram_to_bot.sh is used. This script scans in the directory for outgoing messages and sends out each message by calling curl.
In order to receive messages from your chat a python script called telegrambot.py is used.
In telegrambot.py the library telepot is used to read the chat content. You have to install telepot with

```
sudo pip install teleport
```

Another library which is used is called pathlib. You can install it with this command:

```
sudo pip install pathlib
```

And you have to install python-pip with

```
sudo apt-get install python-pip
```

4) Command line parameters

When you call message2action without any parameters you will get a small help about available command line parameters. You need minimum 1 parameter.
Here is the explanation of the parameters:

in=<incoming messages> : Use for <incoming messages> these values:

in=sms :
message2action will scan the directory for incoming sms (default: /var/spool/sms/incoming) if there are files which were received by the surfstick. When it detects a file (they are named as "GSM1.XYZ") message2action opens it and will extract the last line of the file. In the last line of the file you will find the text which was entered in the message. This line contains the command which should be executed by message2action. After the file is read it will move the file to a directory for saving processed files. You can specify the directory in the ini-file (default is /home/pi/message2action/smsprocessed).

in=mail :
message2action will scan the directory for incoming mails (default: /home/pi/mail) if there are files which were received by fetchmail.
Note: You have to configure fetchmail so it will save the received mails in this directory. See section for fetchmail for details.

in=telegram:
message2action will scan the directory for incoming telegrams (default: /home/pi/telegram) if there are files which were received by telepot.

out=<outgoing messages> : Use for <outgoing messages> the same values as for <incoming messages>

Values for in and out have must be given only when you start the program for processing. out is not required when you want to stop the program.

display=y :
Parameter display will let message2actoin display information, warnings and errors on the screen. If you leave out this parameter the program will only tell its version. The suppressing the display is useful for running the program as a daemon started by systemd.

log=y :
The log option of the start parameter will start message2action in a way that it will write information, errors and warnings to the log-file. You will find in the log-file information like received filename, the found command in the message and so on.
This is useful if you want to find out, what's going on when something does not work as you expect. You can specify the full path + filename of the log-file in the configuration file. If you do not give parameter for logging then no logging for information and warnings will be done: only errors are written to log file.

stopfile : This parameter can be given together with in=<incoming messages>. stopfile will create a file which contains the command for stopping message2action. In order to use parameter stopfile you have to enter it in a separate terminal. After calling message2action with the stopfile command the other message2action-instance will stop because it will find a file with the stop-command in it. So the main purpose of the parameter stopfile is to stop another instance of message2action.

stopsignal : This parameter can be given together with in=<incoming messages>.
This option lets the program stop in another way. When message2action is started it will write its own process-ID to a file, a pid-file. The name of the file can be specified in the ini-file and should be different for all message-types (SMS, mail and Telegram).
If you call message2action with parameter stopsignal then the new instance of message2action will read the pid-file and sends the SIGTERM (15) signal to the process-ID which was saved in pid-file. When the process with the mentioned process-ID receives the SIGTERM signal then it will stop immediately.

?:
A short help will be printed out on the screen.

config:
This parameter will print out on the screen the complete configuration as specified in the ini-file. This function is useful if you do some changes in the ini-file and you want to check if the changes take effect. The ini-file is read only once when the program starts. This means that you have to stop message2action and start again when you have made changes to the ini-file.

daemon=y or daemon=n:
When message2action runs as a systemd-service it tries to catch keyboard hits. But this does not make sense and results in error messages “Error: Terminalsettings could not be determined!” and “Error: Terminalsettings could not be set to new values!”. In order to deal with this mode a new command line parameter was implemented called daemon. It can have the values y or n. If you specify daemon=y then the program does not try anymore to catch keyboard hits.
The default value is “n” for “no”.

When the program is running you can press any so called “readable” key in order to stop it. “readable” keys are for example escape-button, all characters a to z, all numbers. The shift- and ctrl-key do not belong to the “readable” keys.

Here are some examples for calling message2action:
#1: You want to process incoming SMS and want to send back any results with SMS, too. You do not want to display anything on the screen and you do not want logging:

```
message2action in=sms out=sms
```

#2: You want to process incoming mail and want to send back with SMS. You want to display what's going on but logging is not needed:

```
message2action in=mail out=sms display=y
```

#3: You want to process incoming mail and want to send back with mail. You need to see everything on the screen. Further on you want to have a log-file:

```
message2action in=mail out=mail display=y log=y
```

#4: You want to stop message2action by sending a signal to stop. The program currently processes SMS. You have to enter in another terminal:

```
message2action in=sms stopsignal
```

5) Description of ini-file

In the ini-file you can specify values like the path, where the processed file will be saved, the command for stopping message2action and so on.

Every character behind the "#" sign will be ignored. So you can use "#" for saving comments. Please see ini-file with the comments in order to understand better what the parameters mean.
Right after the value there must be equal sign "=" and after the equal sign there has to be the value. For example
CMD03='/home/pi/demo' is correct. But
CMD03 ='/home/pi/demo' is NOT correct and
CMD03= '/home/pi/demo' is NOT correct, too.

5.1) Configuration of program

message2action_stop_filename, default: message2action.stop
When you call in another terminal message2action with command line parameter stopfile then the last called instance of message2action will create a file called message2action.stop. It will save this file in the incoming-directory for the message type you have specified with command line parameter in=.
The file will contain the string representing the stop-message (default: stop).
Depending on the message type it will take the first cell-phone-# (in=sms), the first mail address (in=mail) or the first chat-ID (in=telegram) which is configured in ini-file. See next sub-section for details.

path_logfile, default: /home/pi/message2action/message2action.log
This is the filename where messag2action will save information for logging in. In the log file you will find lines with leading "Info:", "Warning:" and "Error:" and a timestamp behind it.
"Info:" : This is only for information like the found command in a SMS or a status.
"Warning:" : Ooops, something went not in a perfect way. For example there was an invalid cell-phone-# detected.
"Error:" : Something went really wrong. For example the log-file could not be opened for appending lines or file could not be saved due to insufficient rights.

max_size_logfile, default: 500
The value specifies the maximum size of the log-file expressed in kBytes (1 kByte = 1024 Bytes). What does happen, when the log-file extends the maximum size? Then the log-file will be divided by 2 and the younger part of the log-file will be kept. So you will always be able to see the latest lines.

garbage_collection_days, default: 30
With parameter garbage_collection_days you can limit the maximum age of processed files expressed in days. Message2action will search in the directory for processed files for the incoming kind of messages for old files. If message2action finds files which are older than the limit of days the files will be deleted. For example if you specify 30 days in inifile and call message2action with in=sms then message2action will search for old processed sms-messages. Garbage collection protects you that # of processed files increase without any limit even you do not need them anymore.

5.2) Sending and receiving SMS

receivingfrom_cell_phone00 to receivingfrom_cell_phone09, default: 4999
With these parameters you can specify up to 10 cell-phone-# which are recognized as valid ones. Maybe you want to enable your family members, your team or something like this to send SMS to your Pi. You can specify their cell-phone-# here. Each cell-phone-# has the same rights to send all commands to your Pi and to receive the result back.
If a non-valid cell-phone is detected in the incoming SMS, the command will be ignored and a warning message is send to all valid phone-#. The message looks like
"Warning: invalid sender-phonenumber in incoming SMS: %s!". Variable %s contains the cell-phone-# from which the SMS came from.
With this feature no one else than the specified cellphone-# can cause any actions on your Pi.

path_incoming_SMS, default: /var/spool/sms/incoming/
This is the path where message2action will read files related to incoming SMS. This is the directory where SMS-Tools will save the received sms-files.

path_outgoing_SMS, default: /var/spool/sms/outgoing/

This is the path where message2action will save the SMS you want to send out to a cell-phone.

path_processed_SMS, default: /var/local/message2action/processedsms/
In this directory message2action will move the processed files from the incoming-directory. This leads to a growing directory of processed files. But the incoming-directory will be kept empty. With this marker you can check if message2action really scans successfully for incoming SMS.

fullpath_smsfile_PID, default: /home/pi/message2action/smspid
When message2action starts it will save the process-id of its own process in this file. This is needed so that you can send a signal to the running process, for example the SIGTERM (15). The file will be deleted when the program ends in normal way. There are different files containing the process-id for each type of message.

wait_time_next_sms, default: 2
When the program is running, it will loop in order to detect new incoming SMS-files in the incoming directory. With this parameter you can specify the number of seconds before the program will check for new files. This value should correspond with SMS Servertools. For example if SMS Servertools look every 60 seconds for new SMS then it does not make sense to specify here a wait time of 10 seconds.

5.3) Sending and receiving mail

mail_address_00 to mail_address_09: default: dummyuser@pathfinder-outlander.lit
You can specify up to 10 mail-addresses which are accepted as valid ones. The thinking behind is the same as for cell-phone-#. If a non-mail-address is detected in the incoming mail, the command will be ignored and a warning message is send to all mail-addresses. The message looks like
"Warning: invalid mail address in incoming mail: %s!". Variable %s contains the address from which the mail came from. With this feature no one else than the specified mail-addresses can cause any actions on your Pi.
I recommend to add as one mail address the account from which your provider sends mails like “unknown mail address”. For example this account for T-Online from Deutsche Telekom is mailer-daemon@t-online.de.

path_incoming_mail, default:/home/pi/mail/
Inbound-directory where message2action scans for received mails.

path_outgoing_mail, default:/home/pi/mail/outgoing/
Outbound-directory where mails are saved which have to be send out.

fullpath_mailfile_PID, default: /home/pi/message2action/mailpid
Same meaning as for SMS.

wait_time_next_mail, default: 30
Same procedure as for SMS. Note: message2action will not call for new mails but it will look if there are new mails saved as text-files in the incoming directory.

5.4) Sending and receiving telegrams

telegram_bot_token, default: my_telegram_bot_token
This is the token related to your bot in Telegram. Currently only one bot is supported but you can specify up to 10 chat-ids for up to 10 chats with your bot.

telegram_chat_id_00 to telegram_chat_id_09, default: 999999999
These are the chat-ids of chats to which your bot is invited. Please note, that currently group chats can not be accessed by the methods how I setup with telepot.
If a chat-id is detected in the incoming telegram which is not specified here, the command will be ignored and a warning message is send to all chat-ids. The message looks like the one which is created for mails and SMS.

With this feature no one else than the specified chat-ids can cause any actions on your Pi.

path_incoming_telegram, default:/home/pi/telegram/
Inbound-directory where message2action scans for received telegrams.

path_outgoing_telegram, default:/home/pi/telegram/outgoing_to_bot/
Outbound-directory where telegrams are saved which have to be send out to a bot.

path_processed_telegram, default:/home/pi/message2action/telegramprocessed/
The path were processed telegrams are saved.

fullpath_telegram_PID, default: /home/pi/message2action/telegrampid
Same meaning as for SMS and mail.

wait_time_next_telegram, default: 2
Same procedure as for SMS. Note: message2action will not call for new telegrams but it will look if there are new telegrams saved as text-files in the incoming directory.

5.5) Description of commands send in messages to your pi

First of all I distinguish between so called "build-in" commands and "external" commands. "Build-in" commands are those which are implemented in the source-code of message2action. Examples are stop, shutdown, and WOL (wakeup on LAN). In the ini file you can specify the words for each command which you have to send in your message to your Pi. "External" commands are those which will execute programs or even scripts. The path to the program or script is a parameter of the ini-file.

5.6) "Build-in" commands

message2action_stop_command, default: stop
If you send the word "stop" in the message to your PI then message2action will stop running. If you don't like "stop" and want to have "stopprogram" instead you can change this in the ini-file.
It is useful to have this command because if you have found any security issues you can terminate the program from anywhere by simply sending a message with the stop-command to your Pi.

shutdown_pi_command, default: shutdown
If you send the word "shutdown" then message2action will stop and then your Pi (!) will be shutdown. The reason for this command is the same as for stop and it goes a step further: you can shut down your Pi from anywhere.

reboot_pi_command, default: reboot
This command will stop message2action and let reboot your Pi. This may be useful of you feel that something has crashed or hangs on your system and your only chance to recover is a reboot the complete computer.

status_pi_command, default: status
This is the command to send back a status of the program to outbound channel. The status covers total # of processed SMS, mails and Telegrams. Further on total # of errors and total # of warnings since start of the program are reported. # of processed SMS, mails and Telegrams covers only received messages but not the send out ones.

message2action_help_command, default: help
When you send this command to your Pi then you will get a list of possible commands you can send to your Pi. This is helpful if you do not know exactly which command you should send or how it is written exactly.

message2action_questionmark_command, default: ?
This is the same as the message2action_help_command but shorter and more easy to remember.

message2action_config_command, default: config

This command returns the content of the inifile. The result is the same as when you call the program with command line parameter “config”.

WOL00 to WOL09, default: WOL00=1A:1B:1C:1D:1E:1F;host=summerrain
WOL01=2A:2B:2C:2D:2E:2F;host=skyfall
WOL02=3A:3B:3C:3D:3E:3F;ip=10.10.10.10
WOL03=4A:4B:4C:4D:4E:4F;ip=10.10.10.11

The values are MAC-addresses. Behind WOL00 to WOL09 you can write a MAC-address of a device. The MAC-address is used to send the "wake on LAN" message to the MAC-address.This causes the device with this MAC-address (for example a PC with a NIC) to wake up. Maximum 10 MAC-addresses can be saved in the ini-file.

Following the MAC-address you can use ';' to separate the MAC-address from either a host-name or an IPV4 IP-address.
A hostname can be specified with
host=myhostname
An IP-address can be specified with
ip=123.123.123.123
The hostname or the IP-address can be used to check if the device (for example a PC) is online after you have send WOL-command to the device.

online_status_command_suffix, default: stat
This suffix can be set following the command WOL00 to WOL09.
Example: You have sent WOL04 in order to send the WOL-command to the device mentioned after WOL04 in ini-file. Now you want to know if the device is online or not.
So you send "WOL04stat" in your message. Then you will get an answer if the device is online or not. The check is done by simply executing a ping to the desired device. In order to use this feature you have to specify either an IP-address or a hostname after the MAC-address.

check_online_status_waittime, default: 30
After sending out a wake-on-LAN to one of the devices this amount in seconds will be waited and then it will be checked, if the mentioned devices became online or not. You will get an answer from messag2action if the device is online or not. A value = 0 disables this feature, you will not get an answer automatically.

CMD00 to CMD09, no default.
It’s a full path to a program or script. You can specify commands (programs, script) which should be executed when you send a message with the command-# (for example CMD03) to your Pi. These words in the message can be used to execute any program or script on your Pi. For example if you want to restart a web-server which is running on your Pi then you can specify the complete path included with command line parameters in the ini-file. You can use the 10 commands in order to add new functions to message2action.

6) Commands in messages

The way how you can send commands in messages to your Pi depends on the type of the message. All lowercase characters are transformed to uppercase ones.

6.1) SMS

When you type the sms for example on your phone then you simply enter the command as the text.

6.2) Mail

The command is send in the subject of the mail and not in the body. You can leave the body empty because the body will be stripped from the mail.

6.3) Telegram

The command is entered in the chat with your bot. Here is an example how it looks like when you are sending the status-command to your Pi:

I have entered first “status” as a command and I received the number of messages, warnings and errors. After that I send the wake on LAN to my PC called “asus551” and after the wait period the PC was online.

7) Sample files

In the subdirectory samples I have stored some sample files which are useful for testing. With these sample files it is not required to send a SMS from you cellphone, an email or a telegram to your Pi for testing purposes. Instead you can copy a sample file to the incoming-directory for the specific message-type and see what happens.

7.1) SMS

All sample files for SMS have name GSM1.*.
Here is an example: You want to start a program on your Pi which is called by /home/pi/testdirectory/testprogram.sh and you want to call the program with command-line-parameter "invoice tell". You want to start the program when you send CMD03 to your Pi. Then you have to add this line in your ini-file:
CMD03=/home/pi/testdirectory/testprogram.sh invoice tell
After that you edit a sample file (you can take for example file GSM1.CMD03) and add in the first line a cellphone-# which is specified in the ini-file.
In the last line you put CMD03 and save the sample-file. As the last step you copy the sample-file to the incoming-directory for SMS (default: /var/spool/sms/incoming).
Please note that there's no plausibility-check covering the CMD00 ... CMD09 values.
In the directory /home/pi/samples you will find several files for the processing of sms-files. They all have a name beginning with "GSM1.". You can modify these files and copy them to the incoming directory of SMS. When message2action is running it will see the file and will process it.

7.2) Mail

All sample files for SMS have name mail.*.

Assume that you want to start the same program as described in the example for SMS. Then you have to send an email to the account which will be read out by fetchmail. If fetchmail is running then it will get the mail as a text file. In the section of fetchmail in this document you will see how to configure fetchmail so that message2action will work properly.
In the directory /home/pi/samples you will find several files as examples for mails which were received by fetchmail. You can modify these files and copy them to the incoming directory of mails (by default: /home/pi/mails/). When message2action is running it will see the file and will process it.

7.3) Telegram

All sample files for SMS have name telegram.*.
The procedure is the same as for SMS and mail. Put the file in outgoing directory for telegrams (default: /home/pi/telegram/outgoing_to_bot).

8) Installation and configuration on your Pi

On my Pi I've installed the program in directory /home/pi/message2action. All following instructions and descriptions refer to this directory. I recommend that you create directory /home/pi/message2action first before you continue.
Save the archive called message2action.tar in your directory /home/pi/message2action and extract the archive with this command:
tar -x -f message2action.tar

You will get several files and subdirectories in /home/pi/message2action. The files are the following ones:
message2action : the executable binary file
message2action.c : is the sourcecode
message2action.ini : ini file containing the configuration. Note, that you have to configure on your own this file.
message2action.txt: documentation in text-format.
message2action.pdf: documentation in PDF-format which contains some pictures.
sendmails.sh: this script is called by sendmails.service in order to send out the mails which were created by message2action.
The directories are the following ones:
/fetchmail : Files for configuring fetchmail on your own.
/ssmtp : Files for configuring ssmtp on your own.
/telegram: Files for configuring telegram on your own.
/mailprocessed : It is an empty one and it will be used as default for saving the mails which were processed by message2action.
/smsprocessed : It is an empty one and it will be used as default for saving the sms-files which were processed by message2action.
/telegramprocessed: It is an empty one and it will be used as default for saving the telegram-files which were processed by message2action.
/samples: sample files, see section Sample files for details. You will find dummy-SMS, mails and telegrams which can be used for testing purposes.
/systemd : files for setting up message2action as a service which will be started during boot of your Pi and will be stopped during shutdown. Concerning details about the files for systemd please see section systemd. You will also find service- and timer-files in order to start fetchmail for getting mails to your Pi. ssmtp is used to send out mails and there are also service- and timer-files available to be use by systemd. These files are examples, feel free to use them as you like. Please note, that fetchmail and ssmtp have to be installed (and configured) by yourself.
I've added a script for installation the systemd files, it is called installservicetimer.sh.

8.1) ini file

Open the ini file message2action.ini with an editor and configure the fields for your own purposes. See section Description of ini file for details.

8.2) SMS

/etc/smsd.conf: You have to define the device of your surfstick and you have to enable incoming SMS with `incoming = yes`.

8.3) Mail, fetchmail

In directory /home/pi/message2action/fetchmail:
fetchmailparser.sh: No adaption required.
fetchmailrc: Enter here your mailserver, the Pi-account@your domain, the password and the ssl-fingerprint.

8.4) Mail, ssmtp

In directory /home/pi/message2action/ssmtp:
revaliases: adapt here mail-account and mailhub with port.
ssmtpconf: adapt here mail-account and password.

8.5) Telegram

In directory /home/pi/message2action/telegram:
send_telegram_to_bot.sh: No adaption required.
telegramparser.sh: No adaption required.
telegrambot.py: Search for the line `bot = teleport.bot('<insert your token of your bot here>')` and insert your token in the file.

9) systemd

systemd is the successor of initd. Please take the files I've delivered in directory systemd here as an option. Feel free to use them or to find another way to start the programs automatically. In the following sections I've described the service-files. In order to install all files for systemd, you can call installservicetimer.sh. It will copy the service-files to /lib/systemd/system. After the copy-job each service will be registered by the script using systemctl.

9.1) message2action.service

message2action.service will start the program when your Pi boots and will stop the program when you shut down your Pi.
You can check the correct installation of the service message2action.service by entering

```
sudo systemctl status message2action
```

If you want to start message2action as a service you can enter

```
sudo systemctl start message2action
```

In order to stop the service you can enter

```
sudo systemctl stop message2action
```

When you reboot your Pi, message2action will be started with the parameters given in message2action.service.

9.2) getmails.service and getmails.timer

getmails.service calls fetchmail to get mails from your account and saves them in the incoming-directory for mails (default in ini-file: /home/pi/mail). In the subdirectory fetchmail you will find a config file for fetchmail and a parsing-script which is called in the config file. The parsing-script is used to get a unique filename and only the sender-id + the subject of the mail. The body of the mail is ignored, message2action takes only a look at the subject of the mail.
getmails.timer is used to specify how often getmails.service is called to download mails. Default is every 5 minutes. Please note that your mail provider might get confused when you try to download to many times per time your mails. So be careful when you decrease the minutes.

9.3) sendmails.service

sendmails.service starts a script called sendmails.sh with parameter start in order to start the continuous sending out procedure. sendmails.service can stop this sending out procedure by calling the sendmails.sh with parameter stop. In order to stop the sending out procedure you can also call sendmails.sh with parameter stop. The sendmails.sh is delivered only as an example how to send out mails with program ssmtp. sendmails.sh runs as long until it finds a file called /tmp/sendmails.stop. You can create the file to send a signal to sendmails.sh by calling sendmails.sh with parameter stop.

9.4) send_telegram_to_bot.service

This service file is used to send the telegrams which were created by message2action out to your chat-id of your bot. In the service file a shell script send_telegram_to_bot.sh is called which scans for text files and sends out the message with curl.

9.5) telegram-bot.service

This service file starts python script telegrambot.py which uses telepot in order to receive chat-messages. The shell script telegramparser.sh is called as a parser so that the incoming telegrams are saved in files with unique names.

10) Compiling of sourcecode

Feel free to modify the source code. If you want to compile a new release of the program you can use the gcc compiler with these options:
gcc -g –Wall -Wextra -o message2action message2action.c -lpthread
There's no make-file at all. If you have made any modifications or found any bugs please let me know. Maybe I implement your functionality, too.

You can enable a more detailed log and information by defining the so called DEVSTAGE in the sourcecode. If DEVSTAGE is enabled then some more parts of the code will be compiled so that you can see more detailed what's going on. This helped me sometimes for debugging.
In order to enable DEVSTAGE you simply have to define in the source code DEVSTAGE. By default this is disabled right at the beginning of the source code message2action.c

11) Security

The "SMS Server Tools 3" will write the cellphone-# into the sms-file. This cellphone-# is checked by message2action if it is a valid one. In the ini-file you can specify 10 allowed cellphone-#.
This means if someone else is sending a sms to your Pi that message2action will not work with that. All valid cell-phone-# will receive a message with a warning when an invalid phone-# is detected.

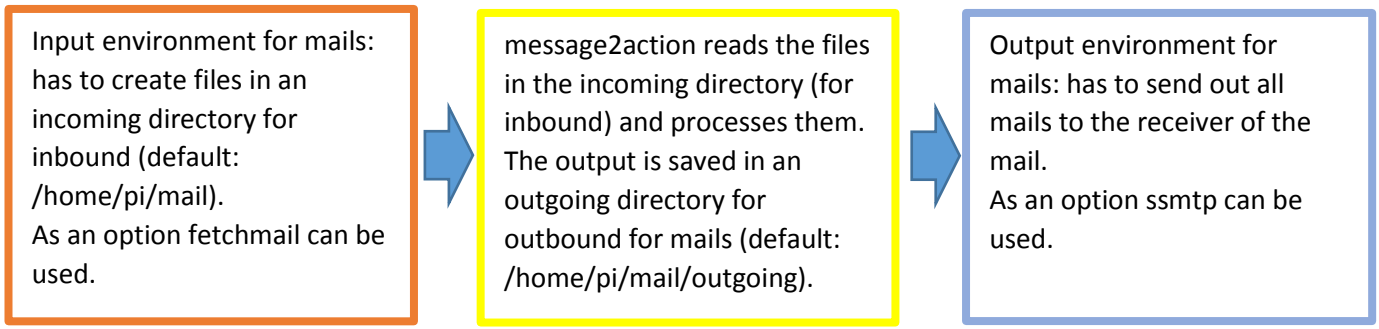
The same is valid for mails. In the ini-file you can specify 10 valid mail-addresses. If your account receives a mail from an unknown account, then all mail-addresses will receive a mail with a warning. This might be a problem if you receive spam mails because every time a spam mail is received then a mail is send out to all registered mail accounts in ini-file.
The same is valid for telegrams. In the ini-file you can specify 10 valid chat-ids. If your bot receives anyhow a message from a chat which is not specified, than all chat-ids will receive a warning message.

12) Next features

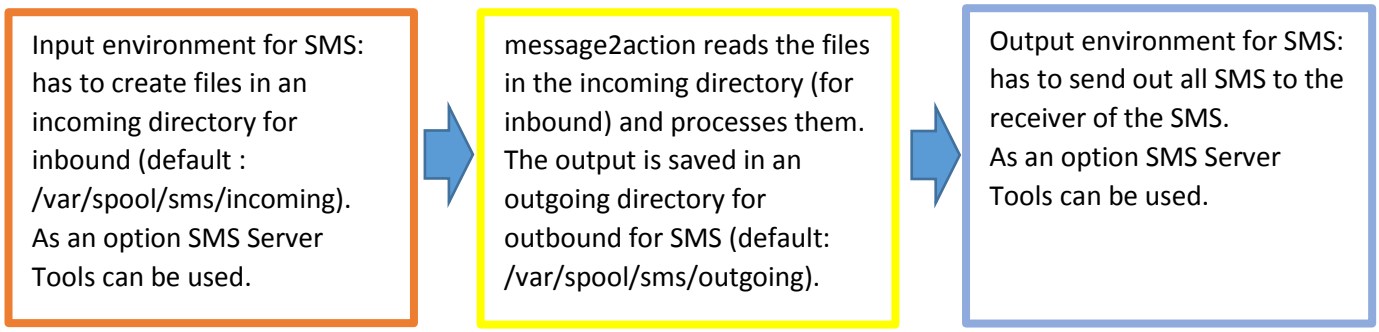
a) I'm thinking about a way to check the status of the GPIO-ports of the PI.
b) In order to view and edit the ini-file you have to use a classic editor. Maybe a GUI will be fine for reading and editing the ini-file.
c) Maybe it would be helpful to save the lines of the log-file in a database instead in a plain file. This may help for analyzing purposes.
d) Maybe it would be helpful to use the mail-accounts implemented in Raspian as a channel, too. So it would be possible to send from a user account on your Pi a mail to message2action.

13) Interfaces for messages

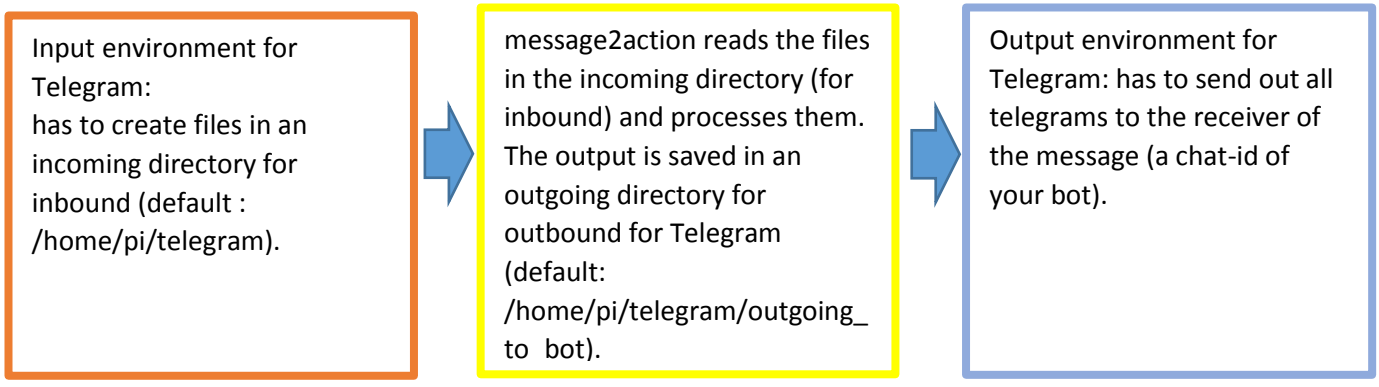
The interfaces to message2action are asynchronous ones and file based. That means that all messages are files which are sent to message2action. And all messages which are returned by message2action are files, too. The consequence of this design is that you have to set up an environment for creating the files, which are to be processed by message2action. And your environment has to be able to pick up the files which were setup by message2action in order to deliver them to the desired receiver. See the following diagram for a general overview for mails:



And here is the corresponding diagram for SMS:



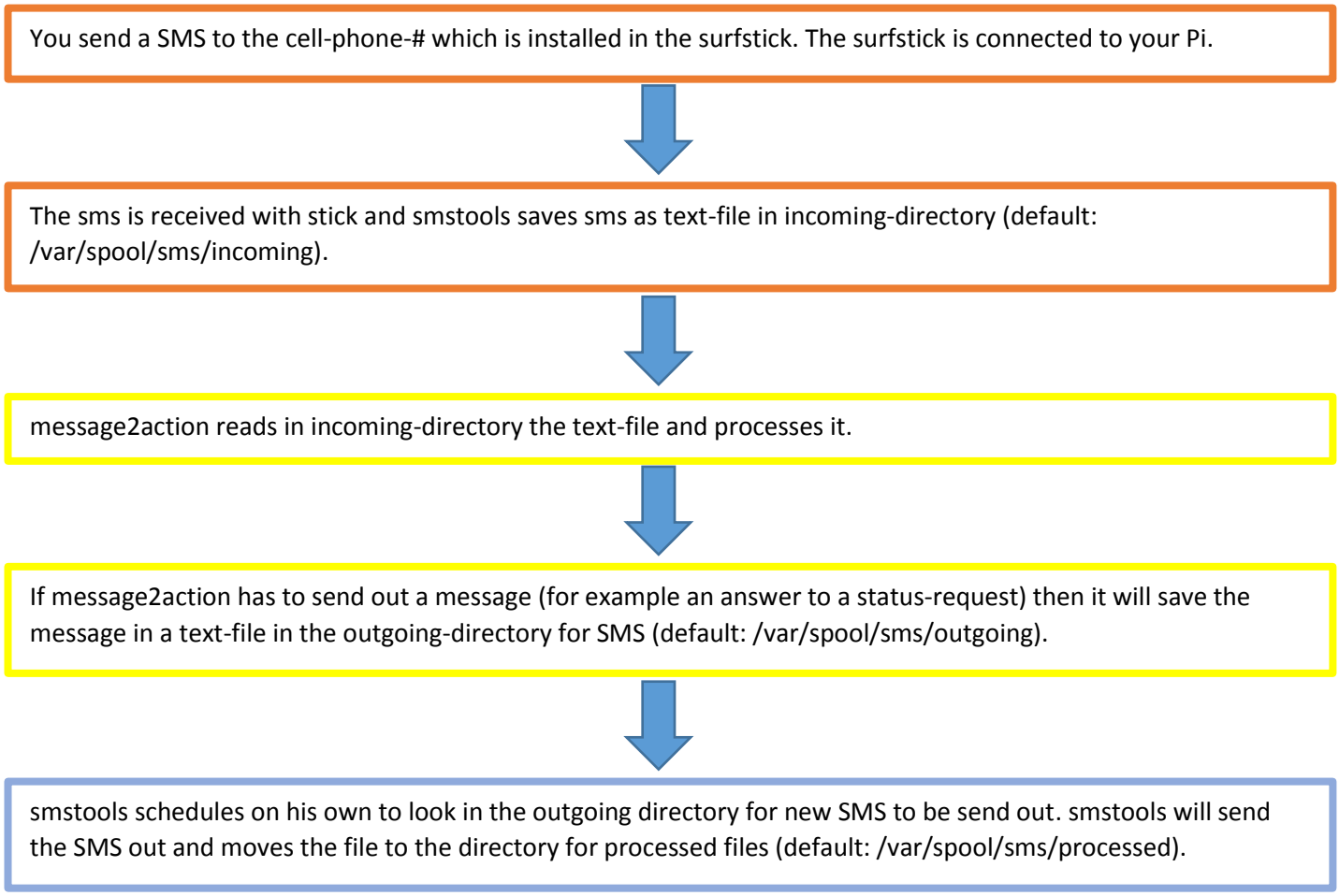
And here is the corresponding diagram for telegrams:



This design makes it easier to integrate the next kind of messages to be processed by messag2action. I know that a processing which happens online would be smarter. But then I would have to implement all channels in my software instead simply using readymade programs like smstools, fetchmail, ssmtp, curl, telepot ….

13.1) SMS

This is the flow for processing SMS on your Pi using message2action. The colors of the input environment, message2action and the output environment are kept as above.



13.1.1) Naming convention and structure of text files

Here is an example of a file which was created by SMStools for an incoming SMS. (real values are replaced by "DUMMY"):

```
From: 4999
From_TOA: DUMMY
From_SMSC: DUMMY
Sent: DUMMY
Received: DUMMY
Subject: DUMMY
Modem: DUMMY
IMSI: DUMMY
Report: DUMMY
Alphabet: DUMMY
Length: 5

CMD01
```

The first line contains the cell phone number after keyword From: from which the SMS came from. This value is used in message2action in order to verify if we have received a SMS from a valid cell phone.
The very last line contains the text which was sent in the SMS. This is the command you send to your Pi and is read by message2action. Before the very last line you will find an empty line.

The files are saved by SMStools with this naming convention: GSM1.abxyz with abxyz representing a random string with 5 characters.

Here's an example of a file which was send out by message2action:

```
To: 4999

Your text to your cell phone
```

The files are saved by message2action with this naming convention: GSM1.<date>_<time>_<# of CPU-ticks>. The <# of CPU-ticks> helps to make the name of the file unique.

13.2) Mail

This is the flow for processing mails on your Pi using message2action using the optional software fetchmail and ssmtp:

You send a mail from your standard account (example: alterego@mailmaster.com) to the account which is related to your Pi (example: mypi@mydomain.com).

Systemd on your Pi calls getmails.service (as configured in getmails.timer). In getmails.service fetchmail is called to download the received mail from account mypi@mydomain.com). fetchmail is calling a shell-script which saves the downloaded mail in the inbound directory for mails (default: /home/pi/mail).

message2action reads in incoming-directory the text-file and processes it.

If message2action has to send out a mail (for example an answer to a status-request) then it will save the message in a text-file in the outgoing-directory for mails (default: /home/pi/mail/outgoing/mail.txt).

systemd calls sendmails.service at boot of your Pi. sendmails.service calls the skript which covers ssmtp (default: /home/pi/message2action/sendmails.sh). ssmtp sends a mail via mypi@mydomain.com to the mail-address from which the original mail came from: alterego@mailmaster.com

13.2.1) Naming convention and structure of text files

Here is an example of a text-file which contains an incoming mail:

```
From: <dummyuser@pathfinder-outlander.lit>
Subject: status
```

The mail address in angle brackets is used by message2action in order to verify if we have received a message from a valid mail account. The text following Subject: contains the command you send to your Pi.

Here is an example of a text file which contains an outgoing mail:

```
To: DUMMYUSER@DUMMY-DOMAIN.INT
```

```
Subject: Info: no further information for you

Message from message2action, see subject for details.
```

The files are saved with this naming convention: <date>_<time>_<# of CPU-ticks>.mail. The <# of CPU-ticks> helps to make the name of the file unique.

13.3) Telegram

This is the flow for processing telegrams on your Pi using message2action using the optional software curl and telepot:

You send a text in your telegram-app or the web-based client to your bot.

Systemd on your Pi calls telegram-bot.service. In telegram-bot.service the python script telegrambot.py is called in order to read text messages you have entered in your chat with your telegram bot. The reading is done by a messagehandler which is started with telepot. If a message is read then the parser telegramparser.sh is called in order to extract the message in a text file. The extracted message is saved in the incoming directory for telegrams (default: /home/pi/telegram).

message2action reads in incoming-directory the text-file and processes it.

If message2action has to send out a message to your bot (for example an answer to a status-request) then it will save the message in a text-file in the outgoing-directory for telegrams (default: /home/pi/telegram/outgoing_to_bot/*.txt).

systemd calls send_telegram_to_bot.service at boot of your Pi. send_telegram_to_bot.service calls the script send_telegram_to_bot.sh. This shell script looks for text files in the outgoing directory for telegrams (default: /home/pi/telegram/outgoing_to_bot). When a text file is found it will be read and send out to your chat id.

13.3.1) Naming convention and structure of text files

Here is an example of a text file which contains a message from a chat with a bot:

```
Name: message_from_chat
Chat-ID: 999999999
Message: status
```

The first line is currently always the same. The second line contains the chat-id with your bot. The last line contains the message you have entered in your chat with your bot.

Here is an example for an outgoing message which will be send to the chat-id with your bot:

```
Token: 12345:ffzfzzz55rrdd
Chat-id: 999999999
Message to bot: Info: # of processed telegrams: 1, SMS: 0, mails: 0, errors: 0,
warnings: 0
```

The first line saves the token of your bot. The second line saves the chat-id with your bot. The third line saves the message which will be visible in your chat with the bot.

14) Release notes

In august 2016 the release 0.13 was published of the predecessor called "sms2action". Here are the changes made in the releases of message2action:

14.1) Release 0.27

- Processing mails was added.
- A parameter check_online_status_waittime was implemented in ini-file.
- When program runs it can be stopped by pressing a key on the keyboard in the terminal.
- systemd files getmails.service, getmails.timer and sendmails.service where provided.
- Documentation in PDF is available.
- Commands send to your Pi are capitalized (example: “status” is read as “STATUS”).
- Some bugs were fixed.

14.2) Release 0.28

- A bug was fixed in reading the ini-file
- A bug was fixed concerning reading the line in a mail which contains the command (new-line was assumed to be always at end of line)
- Initializing of global vars now done with constant values defined with statement “define”
- In systemd-files the call of the program was changed

14.3) Release 0.30

- Parameter fullpath_systemd_shell_mail is no more needed because this function was moved to systemd files.
- Parameter fullpath_outgoing_mail is no more needed because a naming procedure for filenames was implemented.
- Fixed some bugs.
- Processing of telegrams was added.
- The naming to files which are used to send messages out was changed.

14.4) Release 0.31

- Added the installation of Python pathlib library in this documentation.
- Fixed a small bug in message2action.

14.5) Release 0.32

- For SMS the way was changed for searching for the command. In some cases there’s no entry in the file for “IMEI:” and I was sure that this happens only for my stick. But that was wrong. Now I simply search in the textfile for an empty line and take this as a delimiter for the command.

14.6) Release 0.33

- 3 new commands were implemented: message2action_help_command, message2action_questionmark_command and message2action_config_command. Now you can get a help by sending “help” or “?” (both are default). If you want to get the current configuration you can use “config” as default. The scripts sendmails.sh and send_telegram_to_bot.sh had to be adapted.

14.7) Release 0.34

- When message2action runs as a systemd-service it tries to catch keyboard hits. But this does not make sense and results in error messages “Error: Terminalsettings could not be determined!” and “Error: Terminalsettings could not be set to new values!”. In order to deal with this mode a new command line parameter was implemented called daemon. It can have the values y or n. If you specify daemon=y then the program does not try anymore to catch keyboard hits.

- Garbage collecting is now possible. In the inifile you can specify with parameter garbage_collection_days the # of days how old files in the processed directories can become. See description of inifile for details.

14.8) Release 0.35

- When message2action receives commands like “reboot” or “shutdown” it tries now to send back an acknowledgement before that happens in order to signal that the command has been received and will be performed. This is also true for commands CMD00 … CMD09.

FAQ (Frequently asked questions)

#1: Q: Why do there exist 2 ways in order to stop the program message2action?
A: The method with parameter stopfile was implemented first. This way is a little bit slower than parameter stopsignal. When you call parameter stopfile then a message-file is created and message2action will read this file. But when you have specified in ini-file the # of seconds to wait in the next cycle, then you have to wait until the cycle has ended. This is realized by the sleep-function in C.
When you call stopsignal then the sleep-function is interrupted immediately. And this is necessary for implementing message2action as a service on your Pi because you want to stop all services very fast when you shut down or reboot your Pi.

#2: Q: What kind of equipment and software do I need in order to test message2action?
A: You neither need SMS-tools nor a surfstick or something like that. You can use and modify the sample-files in order to check if the program fits to your desires. You also do not need any mail-program on you Pi because you can use sample-files, too. Same is valid for Telegram.

#3: Q: I can not find a make-file. Where is it?
A: There does not exist a make-file yet. This is due to the fact, that I currently have only one file to compile. See section Compiling of sourcode for compiling options.

#4: Q: Does there exist a German release of this document?
A: No, only an English one is available. But if you contact me in German then I will answer in German, too.

#5: Why does there exist a getmails.timer but not a sendmails.timer?
A: When I began developing the software I was not a bash-scripter at all. This changed a little bit. I started with getmails.service and added a getmails.timer because I found it easy to let systemd control the scheduling. Some months later I created sendmails.service and I was able then to write a sendmails.sh which has the scheduler inside. I’m not really sure which way is the better one: Give the scheduling-job to systemd or realize this job on your own. So I keep both methods for learning purposes.

#6: Is there also a support for the telegram client (telegram-cli)?
A: No, not anymore. I found it very difficult to implement the telegram cli and I found it more easy to use telegram bots. I managed to get the telegram cli running but for me it was not a satisfying solution.

#7: Can I process more than 1 type of message parallel?
A: No and yes. In order to make this possible I have to find a way so that more than one instance of message2action can write to same log file. So if you accept that you get separated log files you can make a copy of message2action in a new directory and modify the ini file concerning the path for the log file. After that you can start message2action in the new directory.

15) Contact the author of the program

If you have any questions or comments you can reach me by sending an email to: `freeofcharge@t-online.de`.

16) Legal

Free Software Foundation. This is valid for version 2 of the license or (at your option) any later version.

This program was setup for my own usages but I'm sure that minimum 1 other user will find it as useful for his own purposes. I've set it up
WITHOUT ANY WARRANTY. See the GNU General Public License for more details: http://www.gnu.org/licenses/gpl.txt