Stack SMS

Chema Alonso Pablo González Lucas Fernández Francisco Ramírez

CDO, Telefónica
Madrid, Spain

chema@11paths.com
pablo.gonzalez@11paths.com
lucas.fernandezaragon@telefonica.com
franciscojose.ramirezvicente@telefonica.com

Abstract- *To send text messages through SMS protocol is still quite popular nowadays. In some locations, sending an SMS is free, so it is widely used, and it does not require access or Internet connection. This is a strength because it opens a way of communication when data connectivity is not available. IoT and devices without an Internet connection but GSM, is probably is the most viable option. Stack SMS (SDK and architecture under Open Source license) can take advantage of how to use another communication channel using only GSM. Furthermore, Stack SMS is designed as a communication protocol and it permits implementation of any type of complex applications using SMS as a basis. It also offers a library or framework that allows the creation of all types of utilities that can be run within Stack SMS.*

1. INTRODUCCTION

SMS or Short Message Service has been operating since 1985 without any major significative changes. In fact, the same model is still working even with the new 4G and 5G networks sending messages through GSM standard. These messages still have a format of 140 characters, just to send standard text messages between phones. Its main advantage is that it works on GSM networks. Hence, this is the only way to send information without making a call when there is no data network.

With SMS it is possible to add various types of information within the message. Thus, it is possible to design a client application that could receive those SMS with this new type of data and run commands or operations on the device which receive the SMS. With SMS Stack it is possible to build a data frame with several information headers that can be processed in the device to run the commands. It works like a communication protocol on SMS. On that account, it would be possible to send a whole file between devices using only a GSM network. For example, it would be useful when natural disasters occur disconnecting all Internet services, but the GSM network still works.

Stack SMS can be implemented in various ways. It can be adapted, depending on the user's needs. IoT is a wide-open field where Stack SMS offers lots of possibilities because of the encapsulation within the SMS of complex commands and actions. It is a way to take back control when there is no Internet connection or the signal strength is weak, but GSM band is still reliable.

1.1. SMS standard protocol architecture

As mentioned before, the SMS messages are limited to 120 bytes (this size can vary in many countries), but various character set exists that can be used based on its codification. For example, 140 characters with 8 bits or 70 characters with 16 bits. But all GSM systems support the main one codification mode, GSM 7 bits.

	0	1	2	3	4	5	6	7
0	@	Δ	SP	0	¡	P	¿	p
1	£	_	!	1	A	Q	a	q
2	$	Φ	"	2	B	R	b	r
3	¥	Γ	#	3	C	S	c	s
4	è	Λ	¤	4	D	T	d	t
5	é	Ω	%	5	E	U	e	u
6	ù	Π	&	6	F	V	f	v
7	ì	Ψ	'	7	G	W	g	w
8	ò	Σ	(	8	H	X	h	x
9	Ç	Θ	)	9	I	Y	i	y
10	LF	Ξ	*	:	J	Z	j	z
11	Ø	Note 1	+	;	K	Ä	k	ä
12	ø	Æ	,	<	L	Ö	l	ö
13	CR	æ	-	=	M	Ñ	m	ñ
14	Å	ß	.	>	N	Ü	n	ü
15	å	É	/	?	O	§	o	à

Figure 1. GSM Alphabet 7 bit ASCII

An SMS message has the following packet format

- *SMSC length*. It shows information about the service status or information about the provider who sends the SMS.
- *Timestamp SMSC*. Time information to synchronize all SMS sent by the SMSC.
- Sender telephone number. Telephone number where the SMS send.
- *Protocol Identifier.* TP-PID,
- *TP-DCS data codification*. Codification used in the SMS.
- *Message length*. TP-UDL
- *User data (text message).* TP-UD. Data sent by the user.

Stack SMS takes advantage of the entire SMS infrastructure and encapsulates the sending protocol within "User Data" (TP-UD) in base64 text format (header in hexadecimal, body in base64 / encryption). Therefore, users will receive a traditional SMS, as the client application is responsible for extracting, sending and executing the encapsulated instructions within the text message.

2. STACK SMS

The stack or Stack SMS presented in this document has the option of implementing a group of protocols and SDK tools, focusing on connections, that be used to encapsulate upper protocols. Consequently, sending data through Stack SMS is possible.

2.1. Stack SMS architecture

In this section, the composition of the lower protocol of the SMS Stack is shown. This protocol has functions similar to those of TCP in

the TCP / IP architecture [3]. The following image shows the frame format and the description of the fields that compose the protocol:

2.2.1. *Frame format*

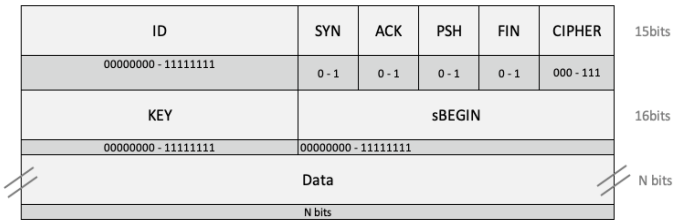


Figure 2: Stack SMS frame format layout.

ID (8 bits): identifies the protocol used in the encapsulation, that is, in the data field (Data).

SYN-ACK - PSH - FIN (1 bit each): flags for connection control and traceability of the information. If it is greater than 140 characters, it splits into several SMS. At his point is where these flags come into play to allow synchronizing and gathering all the information in case of being segmented. The following figure shows the use of each flag:

- *SYN.* This flag, as it happens in the TCP protocol, shows the beginning of a connection to another device using an SMS.
- *ACK.* This flag indicates that the data was received. It works the same as a TCP / IP assent.
- *PSH.* This flag shows the data within the "data" field.
- *FINISH.* This flag indicates the end of the connection.

Stack SMS allows optimizing the connection starting point and its completion using the same SMS. A frame could have SYN, PSH, and FIN active, so it is possible to start a new connection, send data and close the connection in the same message. This technique can be used sending a small amount of information.

CIPHER (3 bits): shows the encryption applied to the data encapsulated within the protocol. The types are: 0 (000) for Base64, 1 (001) for AES + Base64 and 2 (010) for public key. This format could be extended with new implementations. The field remains open, and 3 bits have been reserved to apply any other modes of encryption.

KEY (8 bits): This field stored the internal SMS Stack id to identify the message, mandatory for later synchronization and recovery tasks. When the connection starts, it generates and sends a key (Key). The receiver notices an open connection and can identify when a data packets with the PSH flag is for one application or another. Each connection creates a new link between both sides. This connection chain works similarly like the TCP/IP protocol ports.

sBEGIN (8 bits): message location or position, necessary for traceability in case of sending several SMS messages. This field links to the Key field. With only one Key we can receive different encapsulated data, and we can figure out the data position within this field.

Data (N bits): this field stores all the information sent. Size varies depending on the country.

Once the header is created, the information is codified in hexadecimal just to save space and reduce the SMS to send.

2.2.2. Connection format

Stack SMS connection format tries to be as simple as possible, acting as a transport layer for any other service implemented (some examples below). A new message connection layout (new message) can be seen in figure 3. The information to be sent is prepared with the type of encryption (CIPHER) or which part of the message is sending (sBEGIN).

The second layout details the connections created when an error occurs during the creation of a new message (new message error, figure 4). These messages keep all the connection information to achieve traceability and manage the error.

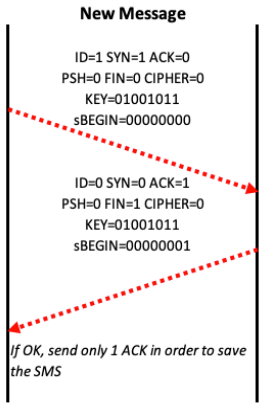


Figura 3. New message conecction format (communication start point)

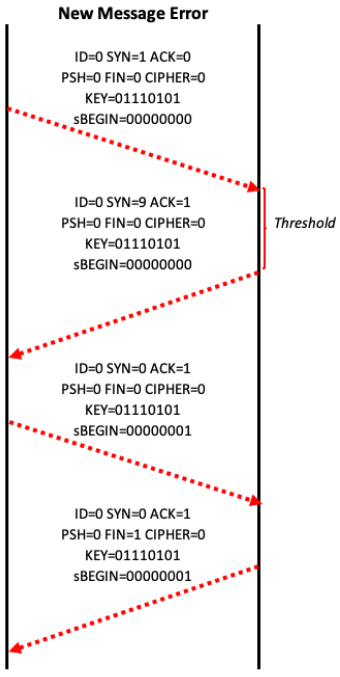


Figure 4. Connection format that describes an error during the handshake setup.

2.2.3. Fields encoding

As seen above, the SMS Stack information is encoded into base64, but the headers have a particular treatment. The method used to encode the headers is through a bit mask. The header has a fixed size, so it applies padding (adding additional bits) for each of the fields when it is necessary (for

example, if you need to complete the standard header size). All the fields are encoded and concatenated using hexadecimal format.

2.2.4. Encryption and codification.

Stack SMS has an option to encode (Base64) and encryption (AES) [4]:

- *Base64*: this encoding is applied within the message body to be transmitted. During the sending data process, Base64 converts the information in printable ASCII characters (AZ, az, and 0-9) for simplification. Stack SMS divide the message into parts based on its length, is encoded in Base64 and finally linked with the rest of the headers. During the reception phase, it sorts all the message, removes the headers, merges all the information and finally, deciphers the data.

- *AES*: once again, the content of the message body is retrieved, but now a secret is shared using a challenge (a "response" signal must be received to validate the action). Next step, after getting the secret, it encrypts the main message and split it into several "n" parts based on the message length. The final step sorts the message, remove the headers, join all the parts and finally is decipher.

To conclude, it is possible to send the data without encryption if it is necessary. The base Stack SMS protocol is open to any modifications.

3. UPPER PROTOCOLS

One of the most interesting features of Stack SMS is the possibility to emulate upper application protocols. Thus, the users can create protocols that can be encapsulated within Stack SMS. The following cases are real Stack SMS implementation examples.

3.1. Latch [5]

Latch gives the possibility to add a new security layer for any service implemented (Latch SDKs are available for Android and iOS). This tool allows using a virtual latch to block or authorize any operations already configured with Latch app.

Stack SMS allows full integration with Latch. It works encapsulating the communications between Latch and the app protected with it.

Header:

OperationID	ID	AccountID	STATUS	Itsok	Mode	16bits
variable	variable	variable	0 - 1	0-1	00-11	

Figure 5. Latch integration header for Stack SMS.

The "header" fields have the following functions:

OperationID (optional, variable size): operation code checked by Latch.. It is related to the 'latches' associated with an app within Latch.

ID (variable size): packet identifier.

Account-ID: this label is related to the Latch app to identify the main "latch." It is a mandatory field. It is possible to check or modify the status, depending on the value of the field.

STATUS (1 bit): Latch status, open or closed.

Itsok (1 bit): operation execution verification.

Mode (2 bits): 01 to apply the status of Latch and 10 to check it.

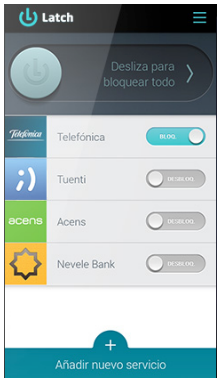
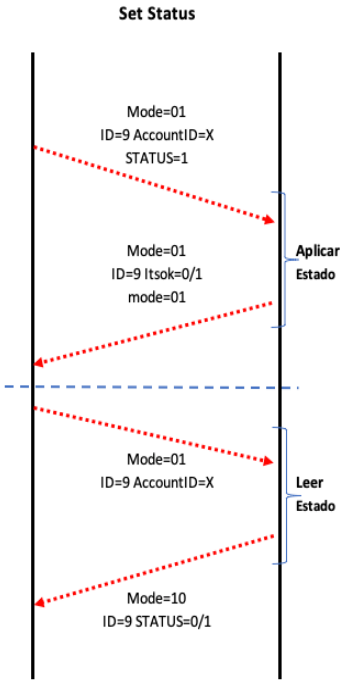


Figure 6. Latch control panel example.

Conecction:



. Figure 7. Connection format to apply a Latch state or check it.

3.2. Chat

The following example shows how SMS Stack can be implemented as a simple chat application, offering new features such as the creation of groups, among others. The

difference of this chat with others is that it can work without an Internet connection; only a GSM connection is necessary.

Header:

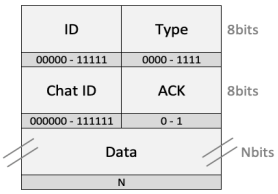


Figure 8. Chat header implementation using Stack SMS.

ID (5 bits): identifies the data packet

Type (4 bits): operation performed within the chat (new group, new message, etc.).

ChatID (6 bits): unique identifier of the chat

ACK (1 bit): confirmation of reception signal.

Data (N bits): chat text to send.

Connection:

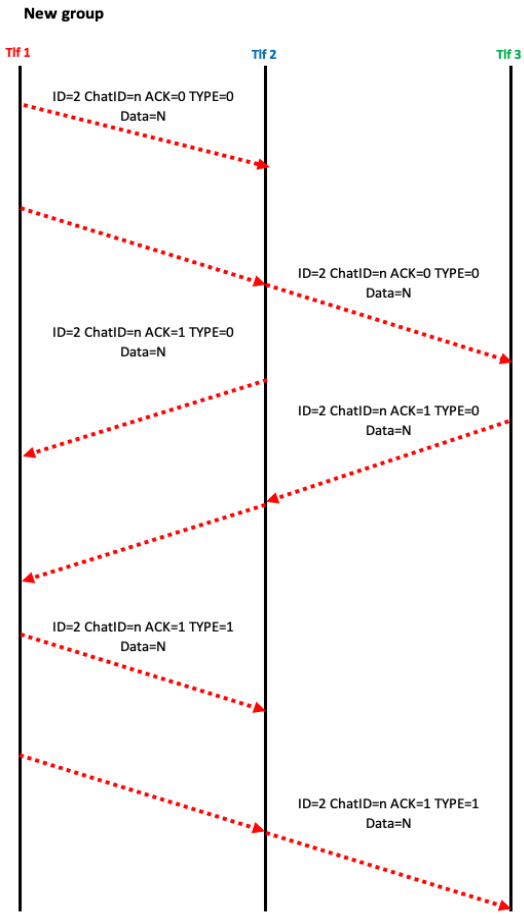


Figure 9. New group chat format connection using Stack SMS.

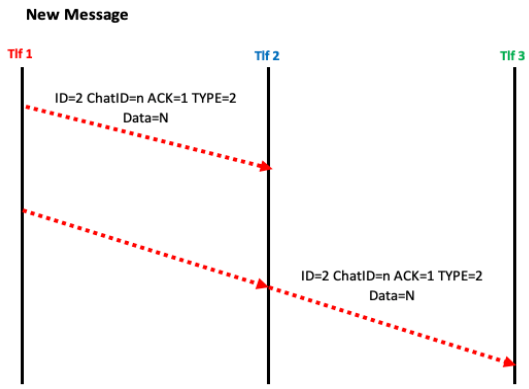


Figure 10. New chat message connection format using Stack SMS.

3.3. Guess the word game

The next example shows a way to implement a variant of the Hangman Game [6]. The goal is to guess a word by asking for a specific letter in each turn. For every letter that is not part of the word to guess, one life will be lost (maximum number of attempts). The game ends when the word is guessed within the established turns or when more lives are no longer available.

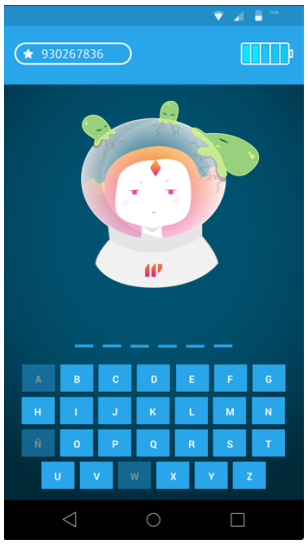


Figure 11. Game implementation (wuth Stack SMS) by ElevenPaths.

Beyond the fact of implementing this game, the critical point is to show the variety of applications that can be created, just by designing the data packets frames according to our needs.

Header:

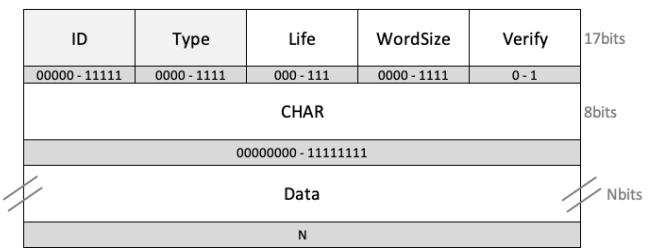


Figure 12. Guess the word game header.

ID (5 bits): packet identifier

Type (4 bits): type 0 is a new game and type 1 is used to send a new letter to check.

Life (optional, 3 bits): indicates the maximum number of lives available (attempts)

Wordsize (optional, 4 bits): assigns the maximum number of letters that make up the full word.

Verify (optional, 1 bit): 0 game lost and 1 game won.

CHAR (8 bits): character in ASCII code used in the game to guess.

Data (optional, size N variable): it stores the complete word guess.

Connection:

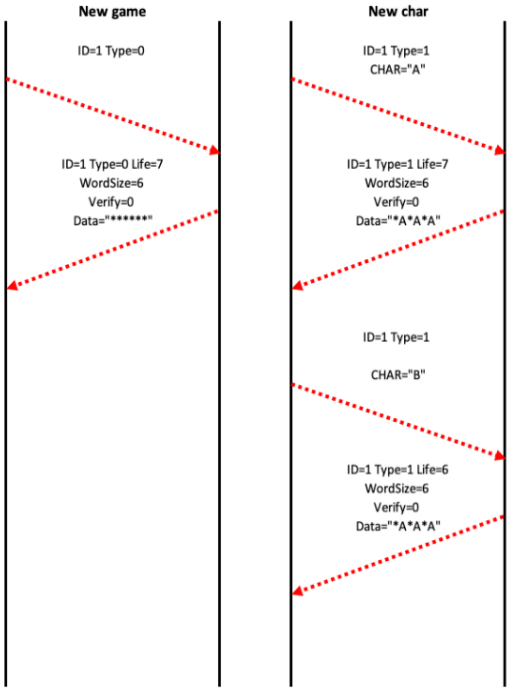


Figure 12. New game connection format (left) and new letter sending.

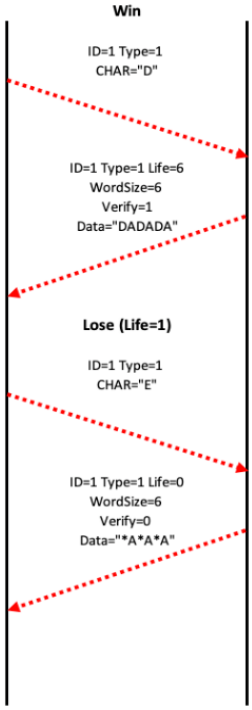


Figure 14. New game and new letter connection format.

3.4. GPIO Raspberry Pi control with Stack SMS (IoT)

The following Stack SMS implementation is focused on IoT, using a Raspberry Pi and the GPIO ports (General Purpose Input Output). It is possible to read and write data from the GPIO ports. This way, it is possible to read data, for instance the one provided from any sensors, send data to relays, engines, etc. using SMS and SMS Stack as a new communication channel. There are many shields (expansion boards) for Raspberry Pi that offer GSM connectivity.

Header:

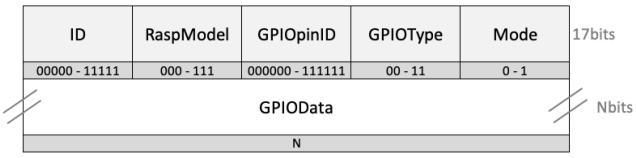


Figure 15. New game and new letter header format.

ID (5 bits): packet identifier

RaspModel (3 bits): Raspberry Pi model to identify the number of GPIO ports. Models A and B (the originals) (code 000) have 26 GPIO pins. Models A +, B + and B2 have 40 GPIO pins (code 001).

GPIOpinID (6 bits): pin number identification. Depending on RaspModel, it may vary. The value represents the pin number instead of the GPIO number, to give more versatility and a high access level to the Raspberry Pi hardware.

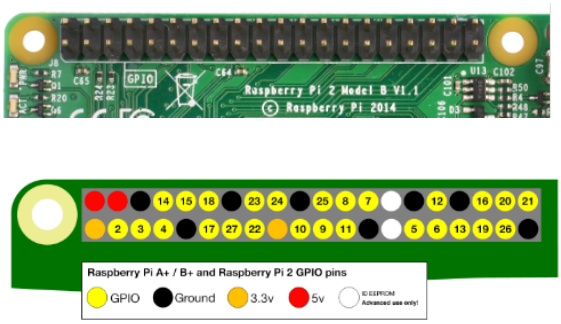


Figure 16. Raspberry Pi GPIO ports avaliable in all the current models. In early models before model B+, pins were reduced by 26 [8].

GPIOType (2 bits): corresponds to:

- *00: GPIO, general purpose*
- *01: I2C, I2C protocol*
- *10: UART, serial communications protocol*
- *11: N / A*

Mode (1 bit): reading (0) or writing (1) towards the port selected.

UARTbit (optional, 1 bit): send start bit serial data (1) and end of the process (0)

ACK (1 bit): reception check

GPIOData (optional, variable size): buffer where the information store during the communication..

Connection:

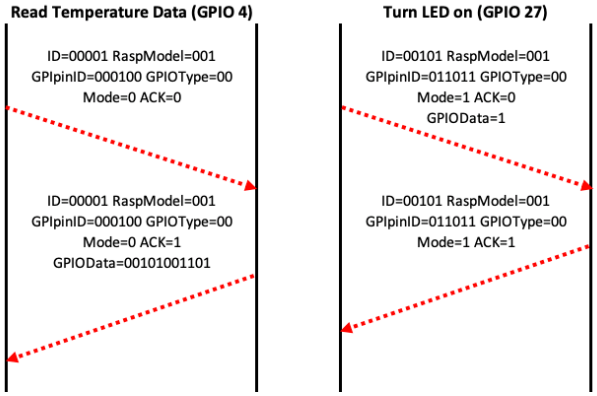


Figure 17. Data Reading connection format on GPIO 4 (for example, to read temperatura data) and LED power on through the GPIO 27 (right).

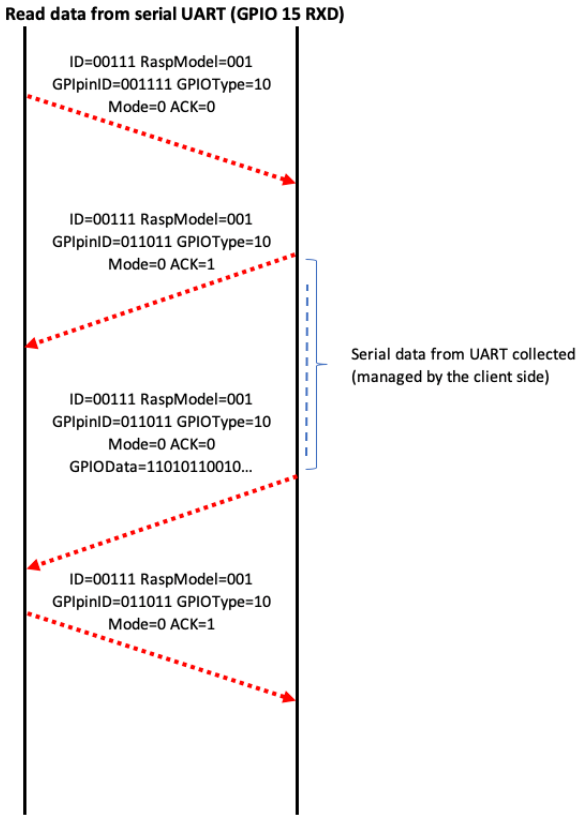
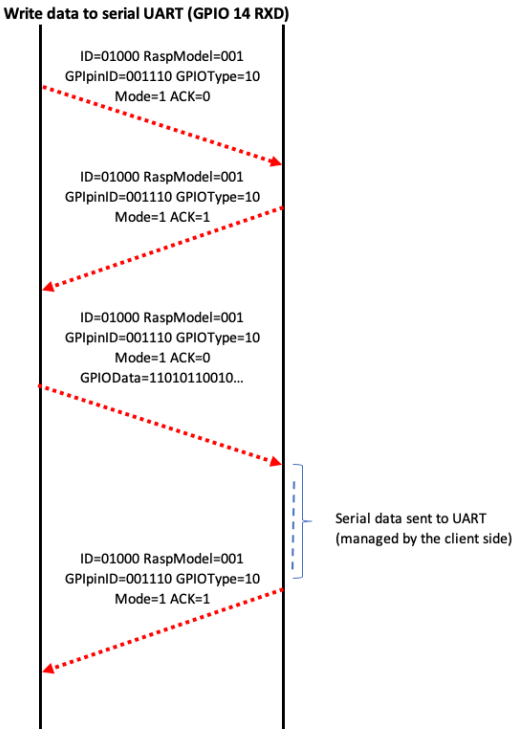


Figure 18. Serial data Reading connection format from the GPIO UART.



. Figure 11. Serial data writting connection format from the GPIO UART.

4. CONCLUSIONS

The practical use and implementation of this SMS protocol still has many possibilities. Stack SMS offers a perfect platform to manage any type of information on a network that does not depend on the Internet. Also, Stack SMS allows the interconnection of various technologies that are not compatible at first. It is not an option to send too much information, mostly because of the fragmentation of the SMS, but it permits the creation of permanent or temporary communication channels (using only GSM) just to run different commands or actions. Stack SMS can be a good communication channel for IoT devices. For example, as we have seen in this document, it is possible to send and receive any kind of information and run commands such as read temperature data, logs, relay activation, turn on or off engines, etc.

In conclusion, only using a SIM card connected to a device, is it possible to implement any kind of task that can be managed through SMS and Stack SMS. The SDK makes the implementation task quite easy and allows the creation of any kind of app, so it can work with SMS. The programmer only needs to design a client app using the SDK available. Stack SMS finally offers a solution that can assure the resilience of the information already sent (like the TCP protocol does) and it offers security during all data transmission (because the data encryption).

5. WORK RECOGNITION

Developed by the Ideas Locas CDO department in Telefonica.

6. REFERENCES

[1] SMS.Technical Specifications. Enlace: https://www.etsi.org/deliver/etsi_ts/123000_123099/123040/12.02.00_60/ts_123040v120200p.pdf

[2] GSM Technical Specification. ETSI.. Enlace: https://www.etsi.org/deliver/etsi_gts/05/0505/05.00.00_60/gsmts_0505v050000p.pdf

[3] W. Stevens, Richard (2001). TCP/IP Illustrated, Vol 1,2 y 3 (20th edición).Addison-Wesley.

[4] Advanced Encryption Standard (AES). FIPSP. Enlace: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

[5] Latch, ElevenPaths., Web oficial: https://latch.elevenpaths.com/

[6] Wikipedia. Juego del Ahorcado. Enlace: https://es.wikipedia.org/wiki/Ahorcado_(juego)

[7] Raspberri Pi. Web oficial. Enlace: https://www.raspberrypi.org/

[8] Wikipedia. Raspberri Pi GPIO: https://www.raspberrypi.org/documentation/usage/gpio/