BÁNKI KÖZLEMÉNYEK
3. ÉVFOLYAM 1. SZÁM

Examining the applicability of Lora technology in an intelligent integral railway environment

A Lora technológia alkalmazhatóságának vizsgálata intelligens integrált vasúti környezetben

Zoltan Papp
Óbuda University, Budapest, Hungary
papp.zoltan@kvk.uni-obuda.hu

In this paper, we describe the difficulties of using LORA MOTE in the railway environment. First of all, we should have created a complex test environment in C#, which is provide us with the appropriate data from the working system. From these data, we can conclude that how can we use LORA technology in a strong electromagnetic radiation and dusty and oily environment such as railway environment.

***Keywords*:** Lora, Mote, Measurement, Railway, Data.

Ebben a cikkben a LORA MOTE vasúti környezetben való használatának problémáit írjuk le. Először is, C #-ban létre kellett hoznunk egy komplex tesztkörnyezetet, amely szolgáltatja számunkra a szükséges adatokat a mérőrendszerből. Ezekből az adatokból arra lehet következtetni, hogy hogyan használhatjuk a LORA technológiát erős elektromágneses sugárzásban, poros és olajos környezetben, mint amilyen a vasúti környezet.

***Kulcsszavak*:** Lora, Mote, mérés, vasút.

1 INTRODUCTION

„The implementation of integrated systems would provide high security, comfortable and cost-effective operation and prestige for the users. Integration means that the processes of each subsystem mutually interact with one another. In many cases, the subsystems have a complementary relationship. The integrated structure enables the development of a more complex rail system, which can ensure safe operation in a cheaper and more efficient way. The driving factor of the design method of the Intelligent Railway System is that: the devices communicate with each other using a unique protocol via optical hard-wire network without any centralized control. The operation of the network is guaranteed on the required security level outdoors, even if any of the system elements are destroyed. The remaining system elements still operated the network.

The aspects of network configuration are that a single hard-wire cut-off cannot cause damage of communication, therefore it is necessary to implement a loop structure of a network. In the case of double hard-wire cut-off occurs devices have secondary communication channels, which are LORA. Using this wireless communication, the system will be able to remain in good working condition for maximum capacity and safety of railway operation.” [6]

„The principle of subsidiarity may be applied at the introduction of a new intelligent system. According to this principle, problems must be solved where they emerge, so higher-level intervention is only necessary if the problem cannot be solved on its level and it would interfere with the operation of the entire network. With the application of this principle it is possible to create an ICT-based transportation system with ambient intelligence – which typically uses wireless communication methods - that fits this structure.” [7]

In Lora networks, nodes communicate to the base station via the gateway. In our system (our local network), the IoT station responsible for the gateway and the network access. Nodes are capable of bidirectional communication with unique encryption. The data packets are extremely short. (8 bytes in our local network)

Data transfer rate can be adjusted depending on transmitter power and distance. The nodes can transmit or receive messages at a single frequency at a time. The gateway is capable of bidirectional communication with multiple nodes.

According to the edict of the NMHH (National Media and Information Office) which rules of band 868 MHz ISM (Industrial, Scientific and Medical) Lora network in Hungary. This band may have a transmission fill factor of up to 1%. This means that after 1-second transmission at a given frequency, then 99-second pause must be maintained. This rule ensures the proper communication of the devices and there isn't any device which reserves the whole frequency band. The Lora nodes constantly switch between the channels in their communication to avoid data collision. if it's occurring, the redundant message has a very small chance of re-collision.

2 LORA COMMUNICATION

2.1 Lora communication can be divided into three classes.

2.1.1 Class A:

Transmission is based on a two-way, asynchronous communication with ALOHA protocol. In this case, we can use the nodes in a "low power sleep mode", which means it consumes the least amount of energy. The transmission from the nodes to the server is possible at any time. The server communication must be buffered on it while the nodes open the communication channel.

2.1.2 Class B:

Transmission provides scheduled communication, which naturally consumes more energy than Class A. Messages delay can be set up for up to 128 seconds, which is still suitable for use with the battery.

2.1.3 Class C:

Communication is similar to Class A communication, but there is an essential difference: it can reduce the delay in communication to the downstream (nodes) so that two downward "windows" are always open and hold the end device in active mode until it receives the data packet. This allows the server to initiate communications at any time. Recommended in cases where continuous performance is required. For example, this is a wireless firmware upgrade.

2.2 Lora connection method can be classified into two types.

- ABP, Activation by Personalization
- OTAA, Over-the-Air Activation

2.2.1 ABP (Activation by Personalization) connection settings

For the ABP connection, the node must be pre-registered on the server interface and must be pre-configured the encryption keys used for communication. (Dev Addr, NwkSKey, and AppSKey)

Advantages:

- The device does not need resources to perform the connection procedures
- encryption key has no expiry date
- No specific DevEUI or AppKey setting is required.

Disadvantages:

- the encryption keys can be stolen
- encryption keys can be detected before activation
- Network settings cannot be entered during connection time

1. table Lora abbreviations

EUI	Extended Unique Identifier - a globally unique id
DevEUI	Device EUI - set by manufacturer, unique per device
AppEUI	Application EUI - identifies the end application
AppKey	Application Key - used in OTAA to generate session keys.
DevAddr	Device Address - identifies a device on a particular network.
NwkSKey	Network Session Key - encrypts the packet metadata.
AppSKey	Application Session Key - encrypts the packet payload.
nonce	A number that is only used once in cryptographic messages.
DevNonce	A random nonce sent from device to network during a join request to prevent rogue devices re-playing the join request.
AppNonce	A nonce sent from network to device during a join response that allows the device to generate the session keys.
NetID	Network Identifier - uniquely identifies the network.
DevAddr	Device Address - identifies the device within the network.
DLSettings	Downlink Settings - data rates to be used for receiving.
RxDelay	Receive Delay - time between transmit and receive.
CFList	Channel Frequency List - frequency setting for each channel.

2.2.2 OTAA (Over-the-Air Activation) connection settings

For the OTAA connection, the DevEUI, AppEUI, and AppKey keys must be configured on the server. Based on AppKey, NwkSKey and AppSKey are created. With these two keys, the message will be encrypted within the Lora network. The device transmits an AppKey as a connection request and then the server compares the preconfigured AppKey with transmitted Appkey. If Its match the server create the key for encryption of communication. The device stores these keys and uses it for the communication.

Advantages:

- The server creates encrypted keys only when activated, so it is safer.
- If the network changes, a re-connection will be established automatically, without having to re-configured it.

Disadvantages:

- DevEUI, AppEUI, AppKey must be defined in advance
- The device must be able to request and store an encryption key

Different communication rate adjustments are available within the connection modes. From DR0 to DR7 level. Measuring node (Lora Mote) can only be adjusted to DR5.

2.table : Lora communication rate

DR	Bit rate	message length max.	Transmission Density
DR0	292 bps	59 byte	fill max. 1%,
DR1	537 bps	59 byte	fill max. 1%,
DR2	976 bps	59 byte	fill max. 1%,
DR3	1757 bps	123 byte	fill max. 1%,
DR4	3125 bps	230 byte	fill max. 1%,
DR5	5468 bps	230 byte	fill max. 1%,
DR6	10937 bps	230 byte	fill max. 1%,
DR7	50000 bps	230 byte	fill max. 1%,

It is possible to temporarily switch between class A and class C in battery mode Our measuring device is also capable of switch between these two communication modes.

1. figure Lora Mote device

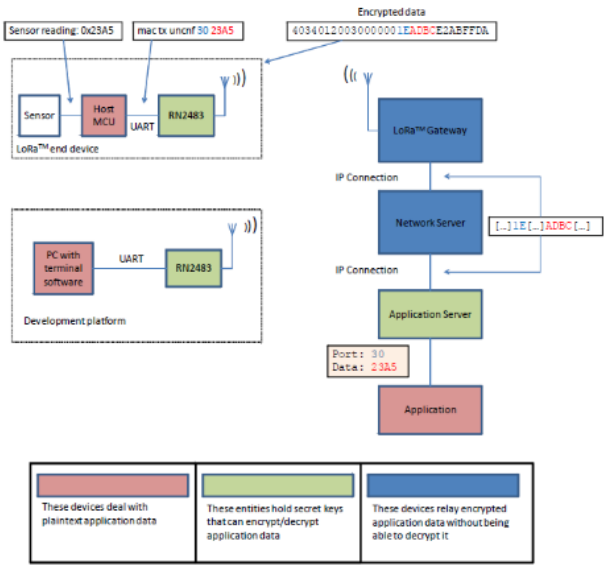
For the test we used Lora Mote, which is certified in 2015. The LORA Mote works with a RN2483 radio module and it can communicate with the computer via an USB port. The development KIT consists a built in power supply, which is suitable for class-a, class-c communication. The device has a light and heat sensor, and it could transmit data packets (messages) to the server on a schedule way or at event of touching a button.

We used PDTIOT-ISS00 - IoT station as a network server. This server is capable to create a stand-alone, self-supporting LoraWan network. It has an outdoor design with 15 km operating range. It is working with Linux operating system, which means easy to integrate into our existing systems. It can be configured by a web-based graphics interface where we can set up many parameters. The server has a 49 channels demodulator.

2. figure: Lora IoT station device

2.3 Operation of Lora Technology

„A simple use case is described in Figure 3 where an end device, containing a host MCU which reads a sensor, commands the RN2483 to transmit the sensor readingover the LoRa network. Data are encrypted by the RN2483 and the radio packet is received by one or multiple gateways which forward it to the network server. The network server sends the data to the application server which has the key to decrypt the application data. Similarly, a development platform may consist of an RN2483 directly connected over UART to a PC which becomes the host system in this case. Users can then type commands into the module using a terminal program."[1]



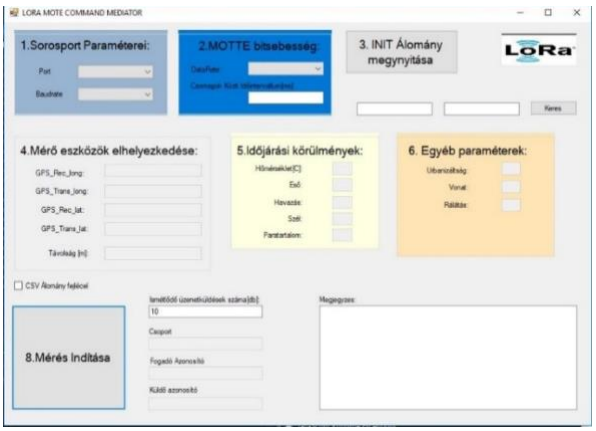
3. figure: Lora technology network diagram

"The LoRaWAN network protocol does not guarantee that all messages will be reaches the network server, but it will guarantee that, the most important messages (priority messages) will be must arrive at the server. " [2]

That's why the Lora network is suit for many IoT devices to communicate with the central server. For example, smart homes, smart city applications and smart devices. In our case the Lora network can be the perfect answer for the additional communication method for train control system. It is not a fail-safe subsystem, so it won't use to control trains, such as GSM-R. For the only task, which we will use for it, is to get the information of devices state from the Inteligent Railway System.

Currently the evolution of the LORA technology is unstoppable. There are many developed interface program found in the market, but we haven't found the appropriate one - which is suit for us, so we decided to develop a custom one for the measurements in the laboratory.

3 LORA MOTE COMMAND MEDIATOR: VERSION: 2.10 LORA COMMUNICATION MEASUREMENT SERIES IN LABORATORY CONDITIONS



4. figure: Lora mote command mediator gui

Lora Technology is not yet fully defined and it has many problems, such as software support. That is why we decided to write our own software, which records the measurements parameters. The basic purpose of the software is to save the

measurement environment, the measurement location, time and data, in an common file type on the client side.

The program was written in visual studio 2017 using C#, under Windows 10. Finally, it converted to the - easy to use - runnable format (exe).

On the server (IoT station) side (web interface) we have the possibility to download measured data in CSV extension. The CSV file contains the received messages and their exact time.

We first created a basic initialization (ini) file for recording the measurement environment.

Eg. precise location, weather conditions, train type (electric or diesel), etc.

The exact naming and filling of our base file (ini) is very important. The ini file name will be the base of the output csv file name and the contents of the ini file (metering environment) will be recorded to the csv file with the measured data.

Currently, the software is under development yet. This version is themselves generating the CSV file together with the measured data. There is no need for any terminal program for programming the mote.

3.1 Using the software in a laboratory measurements step by step

The first step is: connect the Lora mote to the computer using a serial port (USB).

After that, the ini file must be named and filled in with the data of the measurement environment.

Then start LoRa_Usart_Command_Mediator_V2.exe.

It automatically detects active communication ports. Select the appropriate COM port from the group of serial communication box.

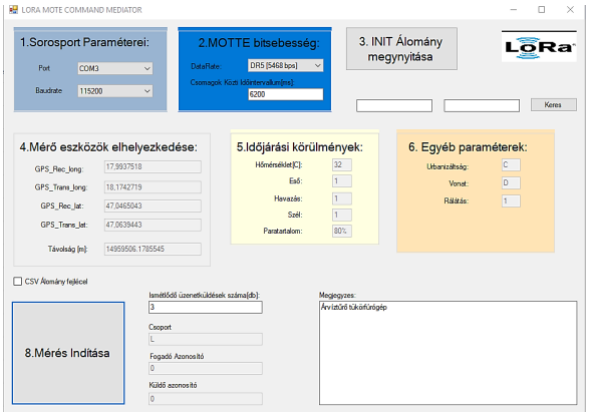
Second step: we can set the rate of communication which is mentioned above. The default rate is dr5. According to the description of Mote, our node not able to communicate a higher rate. (eg. dr6, dr7) Source: RN2483 Lora Technology Module User's Guide.

When we set the communication rate, the delay time the dialogue box has automatically appeared. This is important if you want to send more than one message because you cannot reach the 1% fill of the band.

To calculate the delay time, the sx1272 lora calculator software was used [4], which can be downloaded free of charge from the Internet. Unfortunately, we did not get reliable data from this software, so we implemented the calculation method of the correct delay time in our software.

Third step: load the ini extension file int the group of number 3.

The following figure shows the ini file already loaded.



5. figure: ini file already loaded

Next step: depending on the devices you can enter the ID of the sending device or the receiving device. Then in this parameter box, you can set the type of measurement (L = Lora) and the number of messages. If the number of messages value is high then we are able to perform the overload test. That's why this is a very important parameter for us.

The content of the message automatically completed and they have the largest possible size. In our present case, the size of the message is 8 byte which includes 16 number in hexadecimal format. This message contents the exact sending time.

After that, click to „start" button to start the measurement.

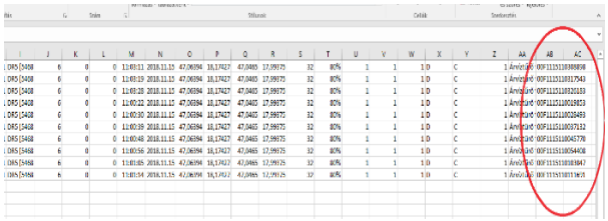
Our program establishes an OTAA connection with the server and it begins to send "unconfirmation messages" continuously. This type of message (uncnf) seemed to be the most reliable way to make measurements.

As long as the server gets the "uncnf" message it doesn't send a confirmation. If we use a normal (cnf) message, the mote will be waiting untill it receives the confirmation. While the Mote waits for the confirmation message from the server, it is not capable to send another message.

The program running list:

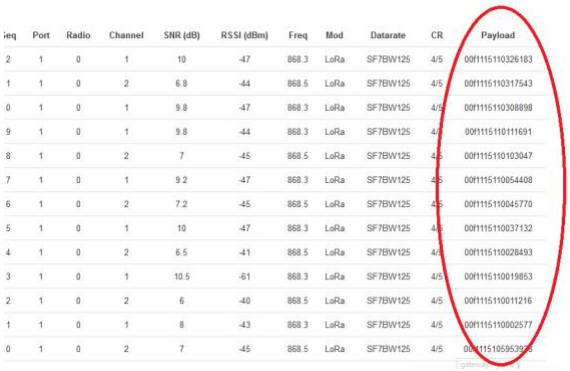
- Starts the measurement
- Restores factory default settings
- sets the important default settings for us
- Links to server with OTAA (class-c) type of connectivity
- the Mote ID key is verified by the pre-registered keys in the gateway (Dev EUI, App EUI, App Key)
- The gateway sends a feedback to the Mote for allowing communication between the two devices and sending the encryption key that the two devices will use during communication
- Starts to send uncnf messages with the predefined data rate and delay
- After the Mote sends all the messages, an "ok" reply is received from it

If the program has run successfully, it will create a measurement client-side database in the csv extension. As we see on the figure 6 in the red ring section.



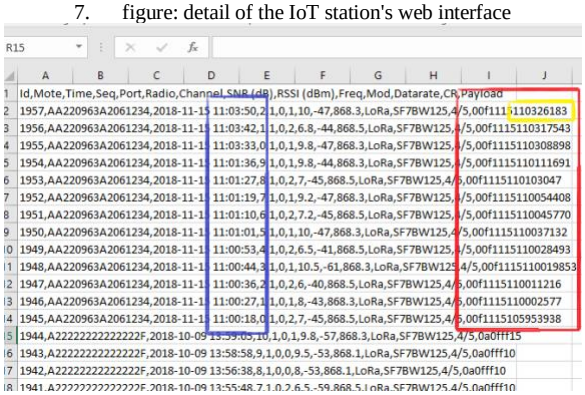
6. figure: detail from client-side csv file

The server-side database can be downloaded from the IoT station's web interface in csv extension. As we can see the received messages in the red ring on the figure 7.



ieq	Port	Radio	Channel	SNR (dB)	RSSI (dBm)	Freq	Mod	Datarate	CR	Payload
2	1	0	1	10	-47	868.3	LoRa	SF7BW125	4/5	00f1115110326183
1	1	0	2	6.8	-44	868.5	LoRa	SF7BW125	4/5	00f1115110317543
0	1	0	1	9.8	-47	868.3	LoRa	SF7BW125	4/5	00f1115110308898
9	1	0	1	9.8	-44	868.3	LoRa	SF7BW125	4/5	00f1115110111691
8	1	0	2	7	-45	868.5	LoRa	SF7BW125	4/5	00f1115110103047
7	1	0	1	9.2	-47	868.3	LoRa	SF7BW125	4/5	00f1115110054408
6	1	0	2	7.2	-45	868.5	LoRa	SF7BW125	4/5	00f1115110045770
5	1	0	1	10	-47	868.3	LoRa	SF7BW125	4/5	00f1115110037132
4	1	0	2	6.5	-41	868.5	LoRa	SF7BW125	4/5	00f1115110028493
3	1	0	1	10.5	-61	868.3	LoRa	SF7BW125	4/5	00f1115110019853
2	1	0	2	6	-40	868.5	LoRa	SF7BW125	4/5	00f1115110011216
1	1	0	1	8	-43	868.3	LoRa	SF7BW125	4/5	00f1115110002577
0	1	0	2	7	-45	868.5	LoRa	SF7BW125	4/5	00f1115105953938

7. figure: detail of the IoT station's web interface



	A	B	C	D	E	F	G	H	I	J
1	Id,Mote,Time,Seq,Port,Radio,Channel,SNR (dB),RSSI (dBm),Freq,Mod,Datarate,CR,Payload									
2	1957,AA220963A2061234,2018-11-15	11:03:50,2	1,0,1,10,-47,868.3,LoRa,SF7BW125,4/5,00f1115110326183							
3	1956,AA220963A2061234,2018-11-15	11:03:42,1	1,0,2,6.8,-44,868.5,LoRa,SF7BW125,4/5,00f1115110317543							
4	1955,AA220963A2061234,2018-11-1	11:03:33,0	1,0,1,9.8,-47,868.3,LoRa,SF7BW125,4/5,00f1115110308898							
5	1954,AA220963A2061234,2018-11-1	11:01:36,9	1,0,1,9.8,-44,868.3,LoRa,SF7BW125,4/5,00f1115110111691							
6	1953,AA220963A2061234,2018-11-1	11:01:27,8	1,0,2,7,-45,868.5,LoRa,SF7BW125,4/5,00f1115110103047							
7	1952,AA220963A2061234,2018-11-1	11:01:19,7	1,0,1,9.2,-47,868.3,LoRa,SF7BW125,4/5,00f1115110054408							
8	1951,AA220963A2061234,2018-11-1	11:01:10,6	1,0,2,7.2,-45,868.5,LoRa,SF7BW125,4/5,00f1115110045770							
9	1950,AA220963A2061234,2018-11-1	11:01:01,5	1,0,1,10,-47,868.3,LoRa,SF7BW125,4/5,00f1115110037132							
10	1949,AA220963A2061234,2018-11-1	11:00:53,4	1,0,2,6.5,-41,868.5,LoRa,SF7BW125,4/5,00f1115110028493							
11	1948,AA220963A2061234,2018-11-1	11:00:44,3	1,0,1,10.5,-61,868.3,LoRa,SF7BW125,4/5,00f1115110019853							
12	1947,AA220963A2061234,2018-11-1	11:00:36,2	1,0,2,6,-40,868.5,LoRa,SF7BW125,4/5,00f1115110011216							
13	1946,AA220963A2061234,2018-11-1	11:00:27,1	1,0,1,8,-43,868.3,LoRa,SF7BW125,4/5,00f1115110002577							
14	1945,AA220963A2061234,2018-11-1	11:00:18,0	1,0,2,7,-45,868.5,LoRa,SF7BW125,4/5,00f1115105953938							
15	1944,A22222222222222F,2018-10-09	13:59:05,10	1,0,1,9.8,-57,868.3,LoRa,SF7BW125,4/5,0a0fff15							
16	1943,A22222222222222F,2018-10-09 13:58:58,9,1,0,0,9.5,-53,868.1,LoRa,SF7BW125,4/5,0a0fff10									
17	1942,A22222222222222F,2018-10-09 13:56:38,8,1,0,0,8,-53,868.1,LoRa,SF7BW125,4/5,0a0fff10									
18	1941,A22222222222222F,2018-10-09 13:55:48,7,1,0,2,6.5,-59,868.5,LoRa,SF7BW125,4/5,0a0fff10									

8. figure: detail of the csv file which downloaded from the IoT station's web interface.

The received messages can be seen in the red frame on the figure 8. The timestamp of receiving data is in the blue frame. The yellow frame (in the received message) shows the transmit timestamp.

The following table summarises the quality of the transmit-receive process.

3.table: Summary table

		ABP			OTAA		
		Mote <-- Server	Mote --> Server	Mote <--> Server	Mote <-- Server	Mote --> Server	Mote <--> Server
class A	uncnf	x	ok	ok	x	ok	ok
	cnf	x	ok	ok	x	ok	ok!
class C	uncnf	x	ok	ok	x	ok	ok!
	cnf	x	ok	x	x	ok!	x

In the cases, where the messages were not sent x signs the problem. The cause of problem was that the Mote communication window was not opened. If the server tried to send a message to the Mote, it arrives when the mote starts to send a message simultaneously. We need further testing activity about those cases when the transmission process was successful but the mote does not get a confirmation ("ok!" sign) message from the server. In order to eliminate the transmission process problems, we have to use "ucnf" type of messages on the field. The laboratory measurements were proved it, as it can be seen on table 3.

REFERENCES

[1] RN2483 Lora Technology Module Command Reference User's Guide.pdf

[2] https://www.elektro-net.hu/konstruktor/7735-epitsunk-egyutt-nyilt-lorawan-halozatot (05/12/2018)

[3] https://lora-alliance.org/about-lorawan (23/11/2018)

[4] http://sx1272-lora-calculator.software.informer.com/download/ (24/11/2018)

[5] https://www.newieventures.com.au/blogtext/2018/2/26/lorawan-otaa-or-abp (06/12/2018)

[6] Jozsef, Papp: Embedded Control System with Shared Logic for railroad transport, In: Innorail Special Edition for Innotrans 2016; 09/2016; pp 40-41.

[7] Jozsef, Papp, Daniel Tokody, Dr. Gyorgy Schuster: The challenges of the intelligent railway network implementation: Initial thoughts from Hungary; MECHEDU 2015. Szabadka, Szerbia, 2015.05.14-2015.05.15. Szabadka: Subotica Technical College of Applied Sciences, 2015. pp. 179-185. ISBN:978-86-918815-0-4