An approach for sending sensor data in environment that lacks internet connection

Nusret Halitia, Arbana Kadriub, Mattias Hanssonc, Fitim Durmishia

aBridge-Technology, SEE University Technological Park, Ilindenska bb, 1000, Tetove, Macedonia

bSEE University, Ilindenska bb, 1000, Tetove, Macedonia

cInnotel AB, Mässans Gata 18, 412 51 Göteborg, Sweden

**Abstract**. Sensors are rapidly being integrated everywhere, in every aspect of our lives, ranging from home automation to wearable sensors for pervasive healthcare management. We hear ubiquitously for devices that are ubiquitous. In recent years, increasingly we are listening and talking about phrases like “Internet of Things”, “M2M”, “Context awareness”, “Web of Things”, which announce the new era to come, where we will be surrounded with smart devices which communicate with each other to easer and beautify our lives. Nevertheless, these new technologies will face with factual difficulties, among which the most expressed are their distributed nature of the machinery and how this data can be sent, which is usually done in wireless environment or using SMS. This article reports about a new way to send the sensor data, using the traditional telephony system. Namely, the idea is to use phone calls to send sensor information to a central place that will understand what values are sent from a sensor. The model described is universal for any kind of sensor, independently from its nature and the data that it emits. The algorithm for encoding and decoding sensor data is explained, together with the mathematical model that grounds our algorithm. At the end, we give a practical implementation of our model for one type of sensor, using an open source framework for developing communications applications, which can be replicated to any type of sensor.

**Keywords:** sensor data, mobile environment, communication network, sensor applications

1. **Introduction**

The Internet of Things (IoT) is a new paradigm that is rapidly achieving ground in the scenario of modern wireless telecommunications [1]. The main concept of this paradigm is that everyday objects can be armed with identifying, sensing, networking and processing possibilities to enable them communication with each other another and other devices and services over the Internet to achieve some beneficial purpose. Critical hardware infrastructure used to build IoT includes: RFID, NFC and Sensor Networks [2]. Most of the sensors utilized until now are wireless, among which employed to build wireless sensor networks are: wireless personal area network (WPAN) (e.g.Bluetooth), wireless local area network (WLAN) (e.g. Wi-Fi), wireless metropolitan area network (WMAN) (e.g. WiMAX), wireless wide area network (WWAN) (e.g. 2G and 3G networks), and satellite network (e.g. GPS). Sensor networks also use two types of protocols for communication: non-IP based (e.g: Zigbee and Sensor-Net) and IP-based protocols (NanoStack, PhyNet, and IPv6) [3]. Wireless Sensor Networks (WSNs) are defined as in self-organizing wireless networks aimed to observe physical measurements, to pass their values through the network to a main station where these values can be kept and processed further [4-9]. The concept of *sensor web* is introduced which is related to the idea of connection of all the sensors in the world and their data together to achieve shared goals [10]. There are different sensor web systems, and most of these systems are dedicated to one type of application only (Mercury dedicated for Parkinson monitoring, CenceMe is dedicated for social network communication etc.) [11]. To develop an effective IoT environment, there are incorporated standards to follow aspiring enable real time integration of heterogeneous sensors [12-15]. Despite the transport of the sensor data to a central location, there can be also a flow of data in reverse direction - from the sink to the sensor [16].

All the above presented research is assuming that there is a wireless sensor network that enables sending flow of sensor emitted information to some central location and in reverse direction. But, there are situations when there is no wireless connection or it has very high cost. The need for sending data in situation of no internet connection was the main drive for the research presented in this paper. We first give an overview of the context and the architecture that we use to set up our solution to the need of sending information in an alternative way. Then we give the algorithm designed for sending the information over the predefined architecture, together with an illustrative implementation of our algorithm for a sensor type. At the end, we discuss the value of our research and the possible application.

1. **Describing the context**

The main question is how to send information gained from different sensors through an environment that lacks internet connection, but there is available some sort of traditional telephony system (PSTN, GSM, satellite phone etc.). In the absence of intermediators in form of wireless connection, the only possible costless solution remains the phone calls. So, now the question that naturally arises is how to send data only by phone calls, from different sensors to a central place that will recognize the information from the made phone calls. Sensors are of different categories and they send data of different type like location, temperature, speed, direction etc. (Figure 1). So, this central place should understand which sensor sends data, what is the value of this data and its category.

Figure . Sending sensor data through phone calls

So, we have a set of categories {}, where each category **ci** has a predefined set of **mj** physical measurements which should be reported do the central place. More sensors can belong to one category, but one sensor belongs to one and only one category. In this way, we can say that for each sensor **si**, which belong to a category **cj** (where 0 < j < n+1), we have a set **Si** of **mj** physical measurements, which should be transmitted:



**Location sensors**

**Weather sensors**

**Pollution sensors**



Si = ... (1)



If for example sensor 4 and sensor 6 belong to the category 2, which holds five physical measurements, to these sensors we will associate the sets S4 and S6 as follows:

S4 = ... (2)



S6 = ... (3)



1. **Architecture**

Figure 2 illustrates our architecture - we suppose that on one side, sensors side, we have diverse sensors that can emit diverse information, which are equipped with phone numbers and can make phone calls.

And on the other side we have a centralized station with a certain amount of phone numbers that is used to gain the information back from the phone calls made to this station.

Thus, on the sensors side, every sensor has somehow to encode the information that it wants to transmit. This encoding should be through a list of subsequent phone calls at the central station, which then uses this list of calls to decode the information gained and sent by a sensor.

The central station has a database of all sensors that can make calls, which includes their phone numbers, together with the category to which they belong. In this way, the phone number of the caller automatically will give us information about the sensor id (which sensor) and category (type of the sent data). It means that we need no encoding for these two elements. The only information that needs encoding is the value of the data. Since there is need for more calls do encode the data, we should identify which call belongs to which part of information.

Figure . Distribution of bits for different categories depending from the physical measurements which each category holds

**phone\_number\_n**

**phone\_number1**

**phone\_number2**

**calls**

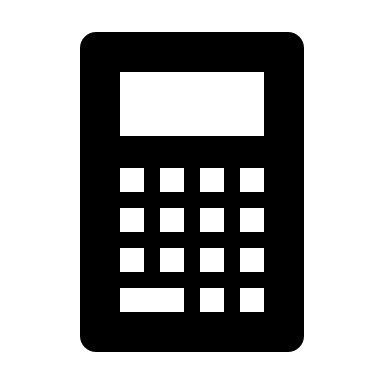
**calls**

**calls**

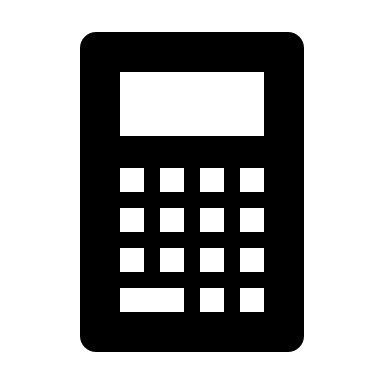
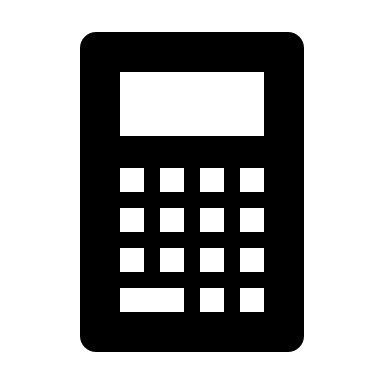
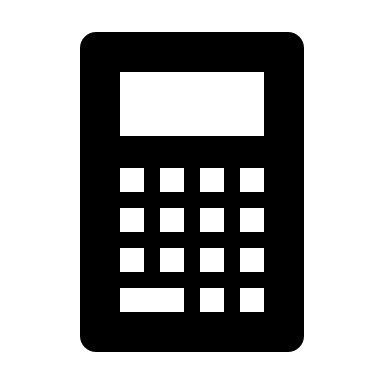
.

.

.



**Number range**



.

.

.

.

.

.



1. **Algorithm for encoding and decoding sensor data**

The main resource for our architecture is a list of phone numbers, set aside with the sole purpose to be called. Normally, they have a fixed part (prefix) for the **K** first digits. In this way, all that remains for encoding are *N - K* digits, where *N* is the length of the phone numbers. This part will be called *extension* further on. For example, if we have a list of 10000 numbers, it means that we can use for encoding all numbers at the range [0000 … 9999][[1]](#footnote-1).

Our encoding/decoding algorithm is based on numerical systems, representing values as list of bits. Since one sensor can emit more than one value and different values can have different length, the number of bits used for some value description will depend from the value itself. For example, *longitude*, which has approximately 9 digits, can be described with 32 bits, while *speed*, which can be maximum 180 miles/hour number, can be described with 8 bits.

In general, for the category **Cj**, if its **i-th** value (physical measurement) is described with **m** bits, then all potential values will be in the range [0000 . . . 2m-1]. And vice versa, the available range determines how many values our system can describe. Hence, if we want to use 2048 different values, we can represent all possible numbers in the interval [0000 … 2047] with 11 bits, given that 211 = 2048. With 13 bits, we can represent all potential values within the range [0000 … 8191], as 213 = 8192.



For every sensor, depending form the physical measurements that it emits, the bits’ distribution is reorganized. Figure 3 gives a generalized picture of this distribution, for sensors of **m** different categories:

Figure . Bits distribution for different metrics

...

...



b11 bits

b12 bits

b1i bits

b1n bits

...

...



...

...



b21 bits

b22 bits

b2i bits

b2n bits

...

...



...

...



bm1 bits

bm2 bits

bmi bits

bmn bits

...

...



**.**

**.**

**.**

**. .**

**Sensor from category 1**

**Sensor from category 2**

**Sensor from category m**

Figure 4 gives us a picture of this distribution for a sensor that gives information on:

* temperature, with values [-50, 170]
* humidity, values in [900, 1100]
* wind, values in [0, 255]
* direction, values in [0, 360]

Figure . A sensor emitting values for temperature, humidity, wind, direction

**temp, 8 bits**

**wind, 8 bits**

**direct, 9 bits**

**hum, 8 bits**

There are values with low number of digits and if this is the case, one call only can transmit them. But there are also values with higher number of digits, which need more calls to transmit the value to the central station. Moreover, usually one sensor gives more than one information, for example pollution and humidity. In these circumstances, again we need to make more phone calls. On the other side, these calls should identify themselves to which part of information belong to. Somehow, now that the system reads the call, it should automatically categorize to which part of the information it belongs. Aiming this purpose, we define a new metric named *offset*, which associates the value with the concrete part of information. There is need for a function that takes the call value from the *phone calls domain* C and relates it to a value from the *offset* *set O*:

*f: C -> O, f(c) = o* ... (4)

where *o* is integer number that shows that the value belongs to the *o*-th part of the information, while O is an ordered set of integer values from 1 to P, calculated as:

*P = int(NR/256),* where NR is the number range ... (5)

Since in most of the cases the number range is not a multiple of 256, there will be some bits left, which can be used for sending some additional information. If we denote this part by T, we have that:

*T = fractional(NR/256) \* 256 ... (6)*

If there are 10000 phone numbers available on the central part for calling, then:

P = int(10000/256) =39, so there will be 39 slots. And there will also remain one 4-bit slot (0.0625 \* 256=16 = 24).

Let’ s say that we have **t**physical measurements for a category **Cj**, which have values V1, V2, ..., Vt, at some point of time, where 20 ≤ Vi ≤ 216. Roughly, if a sensor sends out **t** values, and every *i-th* value occupies a predefined number of *bi* bits, then the overall number of bits need to express all the values is equal to *b1+b2 + ... +bt*. Figure 5 gives a representation of this setting:

**...**

**...**

**v1**

**v2**

**vi**

**vt**

**b1 bits**

**b2 bits**

**bi bits**

**bt bits**

**...**

**...**

Figure . A sensor emitting different values for t different metrics, each occupying a predefined number of bits

Along these lines, the extension will be split in two parts – the identificatory part and the value part. The first **n** bits will be used for the identificatory and the remaining **m** bits will be reserved for the value. The idea behind the identificatory is that now that its value is read, it will automatically tell the position to which that part of information belongs. It means that it will give the offset on the horizontal ax from Figure 5.

So, we need a function that will project every value **vi**, depending from its offset to a value equal to **xi**, which will serve for the phone calls. Offset depends from the number of bits used for the previous calls in the ax defined in Figure 5:

... (7)



Now, the values **xi** for are calculated as follows:

... (8)



*5. 1. Encoding*

1. As a first step in encoding the source number is its multiplication by 10x, where x is the number of digits after the decimal place that we want to consider, depending from the number of bits that we want to occupy for that physical measurement. For example, if we have an information that has eight digits after the decimal place, but we want consider only the first six digits, then this number should be multiplied by 1000000. The multiplication constant is predefined for every metric particularly. If there are no digits after the decimal place, the first step is skipped.
2. After the multiplication operation, we take only the integer part of the gained number. If the integer part is positive number, we move to the next step. Otherwise, we negate it using the two’ s complement logic by adding 2b, where b is the number of bits used for representation. For example, if 32 bits are used, then number 4294967296 will be added, which is equal to 232. Another possibility is to shift the values from the negative part to the positive part gaining in this way a new range of values that starts from 0. If the range is let say [-r, p], after the shifting, it will become [0, p + r].
3. We convert the result obtained from the previous step to its hexadecimal equivalent. The bytes from this value are separated and used to make a list of phone call to transmit the value. If the phone number range is at most 10000, then the calls will be byte by byte. This means that to every single byte corresponds one single phone call to the central station. Otherwise, if we have more available phone numbers for call, we can make phone calls using two successive bytes.
4. As the last step, the offset is added to each byte returned in the third step of our encoding algorithm, respectively to its position. In this way, if we have slots of 8 bits per physical measurement, then to the first byte is added 0, to the second byte is added 256, to the third byte is added 512, to the fourth 768 and so on.

*5. 1. Decoding*

The decoding process is straight forward:

1. We take the phone calls and start decoding for each call independently (the order is not important).
2. For each call, we extract the last X digits (X depends from the number range). Than we check the offset, trying to identify to which part of information it belongs. If it is greater than *offseti* and less than *offseti+1*, it gives us that this value refers to the *i-th* part of the overall metric value. For example, if it is greater than 768 and less than 1023, it means that refers to the fourth part of the overall measurement. When we define the order, we subtract *offseti* from the extracted value. The result is converted to its hexadecimal value. The decoding’s order is not important, the calls are decoded independently to each other.
3. After the processing of all phone calls is finished, since the order for all hexadecimal numbers is known, they are merged per the order.
4. We compare the final number with the biggest positive number (maximum value that can be represented in unsigned long int) that will suit in **b** bits when employing the "two's complement" notation[[2]](#footnote-2). If it is bigger, then we subtract the number by 2b-1 and divide it by 10x, where x is defined in the encoding part too. If not, the final number is just divided by 10x. The returned value is the original value gotten at the sensor side.

*Example:*

Let’ s take a value equal to *-67.7264727* for longitude to illustrate how the encoding/decoding works for a concrete value, supposing that we have used 32 bits for this particular sensor:

*Encoding:*

1. Since in this case there are 7 digits after the decimal place, we multiply it by 107, which results in -677264727.
2. As -677264727 < 0, we calculate the two’ s complement by adding 232 = 4294967296, which outputs to 3617702569.
3. Because the hexadecimal value for 3617702569 is D7A1C2A9, the first byte is D7, which decimal equivalent is equal to 215. Processing similarly for all four bytes we get the numbers 215, 161, 194, 169.
4. We add the identificatory digit to each byte correspondingly and now our list of numbers is 215, 417, 706, 937 (215 = 0 + 215, 417 = 161 + 256, 706 = 194 + 512, 706 = 169 + 768).

Thus, these numbers will be used now to make four calls that end with the just calculated values. The sensor side will make these calls to the central station. Here the calls should be decoded to revert the original value.

*Decoding:*

1. We take the phone calls, which in this case are four, which are of the format \*\*\*\*\*\*\*215, \*\*\*\*\*\*\*417, \*\*\*\*\*\*\*706, \*\*\*\*\*\*\*937.
2. We take the last three digits from, let say the second value 417. Now we check the range and conclude that it is bigger than 256 and less than 511. It means that the order is 2 and we need to minus 256 from 417 resulting in this way with value 161. Now we convert it to its hexadecimal value which is A1. Similarly, if we take the last call for the value 937, after the defining the order and subtraction, we convert it to its hexadecimal equivalent A9. Using the same logic, we have for the first call the hexadecimal equivalent D7 and the third one C2.
3. We merge the hexadecimal numbers per the order. The final number is D7A1C2A9.
4. Decimal equivalent of D7A1C2A9 is 3617702569.
5. Since 231 = 3617702569 > 2147483648, we have:

(3617702569 – 4294967296) / 107 = -677264727 / 107 = **-67.7264727**, which is equal to the original value from the sensor side. Kjo duhet te jete vetm si shembull edhe sepse te edhe google edhe ne vende tjera e perdorin te njejten logjike.

All the description above was about a single value. If a sensor sends out more than one value, the order numbering of the subsequent values will continue at the previous end. For example, if the sensor send out a latitude value, the ordering of the latitude will start from five, since we needed four calls for the longitude metric. Everything else, has the same logic.

1. **Implementation and results**

We have tested our algorithm using Libelium device to send location data to a central server, where we use Asterisk, a framework for constructing communications applications[[3]](#footnote-3). This framework uses a dial plan to deal with the phone calls that it recognizes. In our case, now that a call is recognized, we call a script that has the phone number and the extension as input arguments.

This script then decodes the value (extension) and puts the decoded value to a database (in this case it is implemented in PostgreSql) that holds all the emitted data for different sensors:

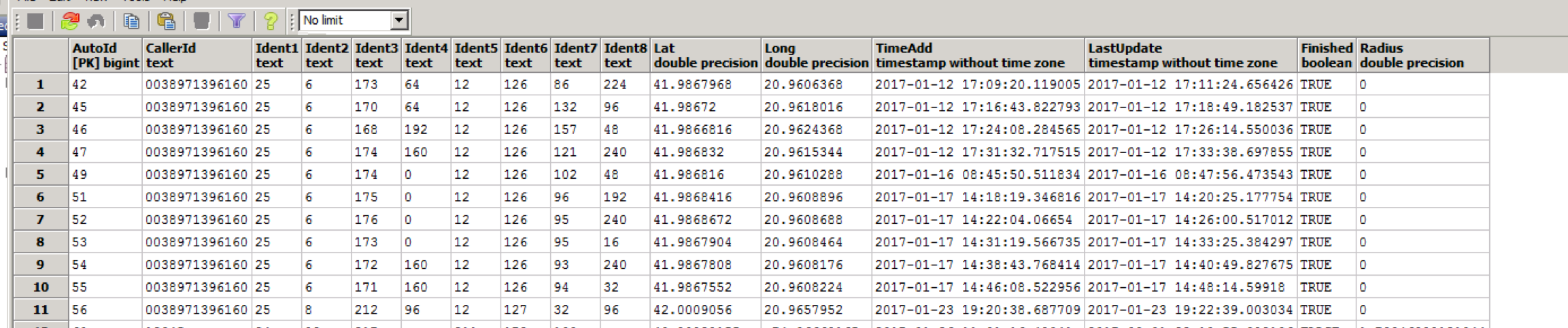


Figure . Database of the emitted sensor data

For every phone call, a new record is added, filling one by one all the values for each identificatory separately. The Libelium device is programed in that way that initially the first, most values for longitude and latitude is sent:

USB.print("La4: ");

numberToCall(Lat[0]);

USB.print(" - ");

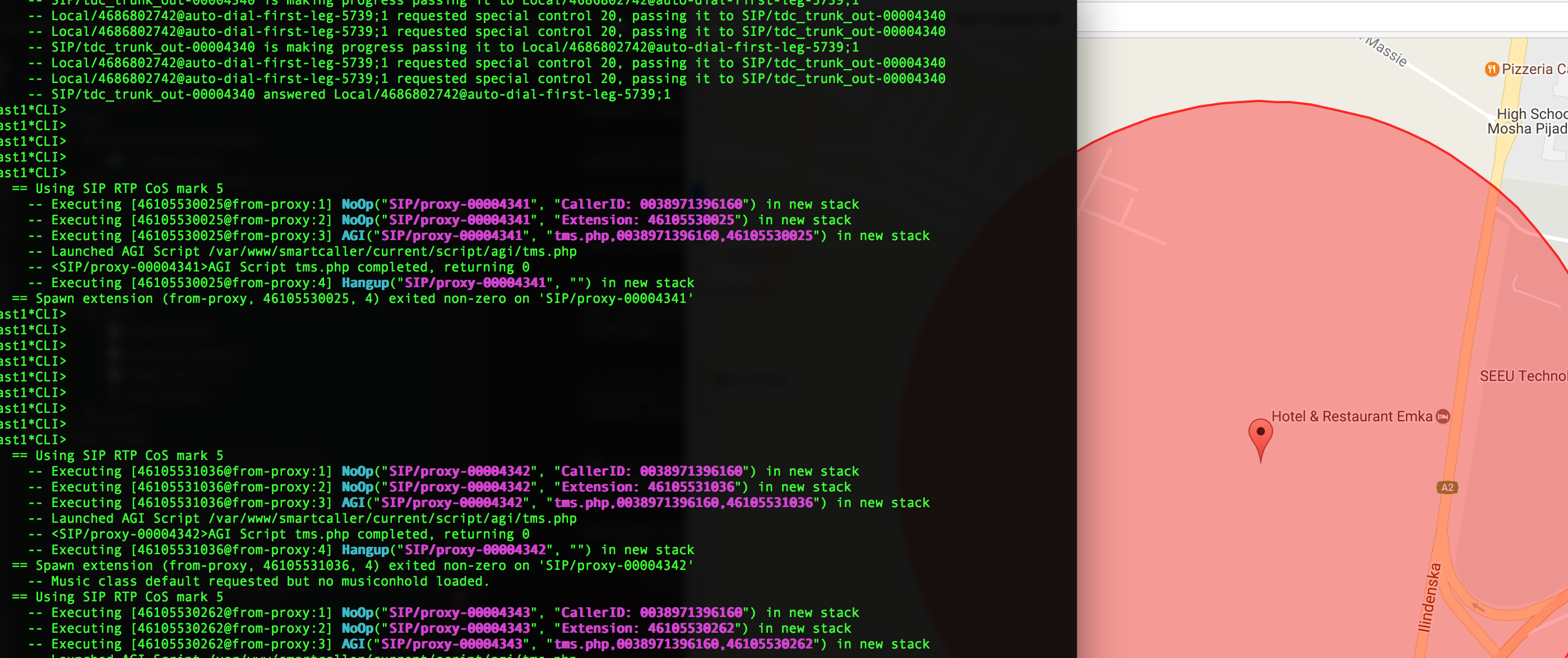
USB.println(Lat[0]);

makeCall();

In this way, even with the first two calls, we have an approximation about the location. Hence, whichever application that uses this information, can be programmed to be event based and updated for every call subsequently. To demonstrate this, we have built a simple web application that shows the current position of the sensor on a map, depending from the last phone calls (Figure 7). Our application is event-based, which is notified every time a new value (identificatory) is added to the database, regarding the sensor that is observed.

aksndalndan

Figure . Real-time execution of our method



We also measured the throughput time required to make twenty successive calls for sensor data, together with their encoding time (we have infrastructure limitations at this stage of the research). Table 1 gives an illustration of the gained results. The average time needed to establish call is approximately 5 seconds. Taking that in our case we have available the phone extension ranging from 0000 until 9999, with one call we can transfer approximately 13.3 bits. From this we can conclude that for this communication channel the capacity is 13.3 bits / 5 sec = 2.66 bits/sec.

Table . Measuring the throughput time

|  |  |  |
| --- | --- | --- |
| **Attempt** | **Tel number** | **Time needed to establish a call** |
| **1** | X0025 | 5.31 |
| **2** | X0145 | 6.40 |
| **3** | X0987 | 5.10 |
| **4** | X0147 | 5.17 |
| **5** | X0472 | 4.98 |
| **6** | X0111 | 5.18 |
| **7** | X0454 | 6.31 |
| **8** | X0333 | 4.54 |
| **9** | X0987 | 4.61 |
| **10** | X0654 | 4.67 |
| **11** | X0321 | 4.82 |
| **12** | X0369 | 4.97 |
| **13** | X0258 | 5.18 |
| **14** | X0147 | 4.97 |
| **15** | X0159 | 4.67 |
| **16** | X0357 | 4.41 |
| **17** | X0000 | 5.23 |
| **18** | X0387 | 4.90 |
| **19** | X0385 | 4.35 |
| **20** | X0085 | 4.67 |

1. **Conclusions and Further Work**

In this paper, we presented a new way of sending sensor data to a central location with no use of traditional communication channels and we have demonstrated that this kind of sensor data transport is fully functional. This approach for sensor records transmission can lead to unlimited possibilities of applications and business models, such as telematics, smart metering (electricity, gas, water, air/pollution etc.), automotive (car diagnostics, remote command execution etc.), home automation and so on. In our further work, we plan to implement an algorithm that will execute commands using phone calls in the opposite direction – from the sink to a sensor, as for example could be “give me information about the lights status”. Furthermore, we intend to explore alternative ways to transmit data in ringing mode.

1. **References**

[1] Luigi Atzori, Antonio Iera, Giacomo Morabito (2010) The Internet of Things: A survey. The International Journal of Computer and Telecommunications Networking, Volume 54 Issue 15, pp. 2787-2805.

[2] Andrew Whitmore, Anurag Agarwal, Li Da Xu (2014) The Internet of Things—A survey of topics and trends. Information Systems Frontiers, Volume 17, Issue 2, pp 261–274.

[3] Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos (2013) Context Aware Computing for The Internet of Things: A Survey. IEEE Communications Surveys & Tutorials, Volume: 16 Issue: 1, pp. 414 – 454.

[4] M.A. Matin, M.M. Islam (2012) Overview of Wireless Sensor Network. Wireless Sensor Networks - Technology and Protocols, Book Chapter 1.

[5] Chris Guy (2006) Wireless sensor networks. Proceedings of Sixth International Symposium on Instrumentation and Control Technology: Signal Analysis, Measurement Theory, Photo-Electronic Technology, and Artificial Intelligence.

[6] F. L. Lewis (2004) Wireless Sensor Networks. Smart Environments: Technologies, Protocols, and Applications (eds D. J. Cook and S. K. Das), John Wiley & Sons, Inc., Hoboken, NJ, USA.

[7] John A. Stankovic (2008) Wireless Sensor Networks. Computer, Volume: 41, Issue: 10.

[8] Chris Townsend, Steven Arms (2005) Wireless Sensor Networks: Principles and Applications. Sensors, Transducers and Detectors, Book Chapter 22.

[9] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci (2002) Wireless sensor networks: a survey. Computer Networks, Volume 38, Issue 4, pp. 393–422.

[10] Botts, M.; Robin, A. (2007) Bringing the sensor web together. Geosciences, pp. 46–53.

[11] Klaithem Al Nuaimi, Mariam Al Nuaimi, Nader Mohamed, Imad Jawhar, Khaled Shuaib (2012) Web-based wireless sensor networks: a survey of architectures and applications. Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, Article No. 113.

[12] Telecommunications and information exchange between systems (ISO/IEC JTC 1/SC 6), <https://www.iso.org/committee/45072.html>)

[13] Internet of Things Global Standards Initiative (IoT-GSI), <http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx>

[14] Joint Coordination Activity on Internet of Things and Smart Cities and Communities (JCA IoT and SC&C), <https://www.itu.int/en/ITU-T/jca/iot/Documents/ToR/Scope-and-ToR-JCA-IoT-SC-C-18112015.pdf>

[15] Joint Coordination Activity on Internet of Things (JCA-IoT), <http://www.geneve-int.ch/joint-coordination-activity-internet-things-jca-iot>

[16] Weilian Su, Ozgiir B. Akan, Erdal Cayirci (2006) Communication Protocols for Sensor Networks. Wireless Sensor Networks, Book Chapter 2.

1. The phone operator will define the starting phone number, what is important is that the phone numbers should be a list of consecutive numbers [↑](#footnote-ref-1)
2. This step is only needed if encode negative numbers. [↑](#footnote-ref-2)
3. http://www.asterisk.org/ [↑](#footnote-ref-3)