Un enfoque para enviar datos de sensores en un entorno que carece de conexión a Internet

Nusret Haliti[a], Arbana Kadriu[b], Mattias Hansson[c], Fitim Durmishi[a]

[a]Bridge-Technology, SEE University Technological Park, Ilindenska bb, 1000, Tetove, Macedonia

[b]SEE University, Ilindenska bb, 1000, Tetove, Macedonia

[c]Innotel AB, Mässans Gata 18, 412 51 Göteborg, Sweden

Resumen . Los sensores se están integrando rápidamente en todas partes, en todos los aspectos de nuestras vidas, desde la automatización del hogar hasta sensores portátiles para la administración generalizada de la atención médica. Escuchamos de manera ubicua los dispositivos que son ubicuos. En los últimos años, cada vez escuchamos y hablamos más sobre frases como " Internet of Things", "M2M", "Context awareness", "Web of Things", que anuncian la nueva era por venir, donde nos rodearemos de dispositivos inteligentes que se comunicarán entre sí para facilitar y embellecer nuestras vidas. Sin embargo, estas nuevas tecnologías se enfrentarán con dificultades reales, entre las cuales las más analizadas corresponden a la naturaleza distribuida de la maquinaria y cómo se pueden enviar los datos, lo que generalmente se hace en un entorno inalámbrico o usando SMS. Este artículo explica una nueva forma de enviar los datos del sensor, utilizando el sistema de telefonía tradicional. A saber, la idea es usar llamadas telefónicas para emitir información del sensor a un lugar central que comprenderá qué valores se envían desde un sensor. El modelo descrito es universal para cualquier tipo de sensor, independientemente de su naturaleza y de los datos que emite. Se explica el algoritmo para codificar y descodificar los datos del sensor, junto con el modelo matemático que fundamenta nuestro algoritmo. Al final, damos una implementación práctica de nuestro modelo para un tipo de sensor, utilizando un marco de código abierto, para desarrollar aplicaciones de comunicaciones que se puede replicar en cualquier tipo de sensor.

Palabras clave: datos del sensor, entorno móvil, red de comunicación, aplicaciones de sensores.

1. Introducción

Internet de las cosas (IoT) es un nuevo paradigma que ha alcanzando rápidamente terreno en el escenario de las telecomunicaciones inalámbricas modernas[1]. El concepto principal de este paradigma es que los objetos cotidianos pueden ser armados para identificar, detectar, interconectar y procesar posibilidades que permita la comunicación entre sí y otros dispositivos y servicios a través de internet y así lograr algún propósito beneficioso crítico. La infraestructura de hardware utilizada para construir IoT incluye: RFID, NFC y Sensor Redes[2]. La mayoría de los sensores utilizados hasta ahora son inalámbricos, entre los que se utilizan para construir redes inalámbricas. Las redes de sensores son: red de área personal inalámbrica (WPAN) (por ejemplo, Bluetooth), red de área local inalámbrica (WLAN) (por ejemplo, Wi-Fi), red de área metropolitana inalámbrica (WMAN) (por ejemplo, WiMAX) red inalámbrica de área extensa (WWAN) (por ejemplo, redes 2G y 3G), y red satelital (por ejemplo, GPS). Las redes de sensores también utilizan dos tipos de protocolos para la comunicación: no IP basados (p.ej.: Zigbee y Sensor-Net) y protocolos basados en IP (NanoStack, PhyNet e IPv6)[3]. Las redes inalámbricas de sensores (WSN) se definen como redes inalámbricas autoorganizadas destinadas a observar las mediciones físicas, para pasar

sus valores a través de la red a una estación principal donde estos valores pueden mantenerse y procesarse[4-9]. El concepto de *sensor web* es introducido, el cual está relacionado con la idea de conexión de todos los sensores en el mundo y sus datos en conjunto para lograr objetivos compartidos[10]. Hay diferentes sistemas de sensores web, y la mayoría de los sistemas en sí se dedican solo a un tipo de aplicación (Mercurio está dedicado para la supervisión de Parkinson, CenceMe está dedicado a la comunicación de redes sociales, etc.)[11].

Para desarrollar un entorno efectivo IoT, se han incorporado estándares para conseguir la integración en tiempo real de sensores heterogéneos [12-15]. A pesar de que hay transporte de los datos del sensor a una ubicación central, también puede haber un flujo de datos en dirección inversa, desde una ubicación central hasta el sensor[16].

Todas las investigaciones presentadas anteriormente suponen que existe una red de sensores inalámbricos que permite enviar el flujo de información emitida por el sensor a alguna ubicación central y en dirección inversa. Pero hay situaciones en las que no hay conexión inalámbrica o tiene un costo muy elevado. La necesidad de enviar datos en una situación sin conexión a Internet fue el impulso principal para la investigación presentada en este documento. Primero damos una visión general del contexto y la arquitectura que utilizamos para configurar nuestra solución a la necesidad de enviar información de una manera alternativa. Luego proporcionamos el algoritmo diseñado para enviar la información a través de la arquitectura predefinida, junto con una implementación ilustrativa de nuestro algoritmo para un tipo de sensor. Al final se discute el valor de nuestra investigación y la posible aplicación.

2. Describiendo el contexto

La pregunta principal es cómo enviar información obtenida de diferentes sensores a través de un entorno que carece de conexión a Internet, pero donde está disponible algún tipo de sistema de telefonía tradicional (PSTN, GSM, teléfono satelital, etc...). En ausencia de intermediarios en forma de conexión inalámbrica, la única solución sin costo posible sigue siendo las llamadas telefónicas. Entonces, ahora la pregunta que surge naturalmente es cómo enviar datos solo por llamadas telefónicas, desde diferentes sensores a un lugar central que reconocerá la información de las llamadas telefónicas realizadas. Los sensores son de diferentes categorías y envían datos de diferentes tipos, como ubicación, temperatura, velocidad, dirección, etc. (Figura 1). Entonces, este lugar central debería comprender qué sensor envía datos, cuál es el valor de estos datos y su categoría.

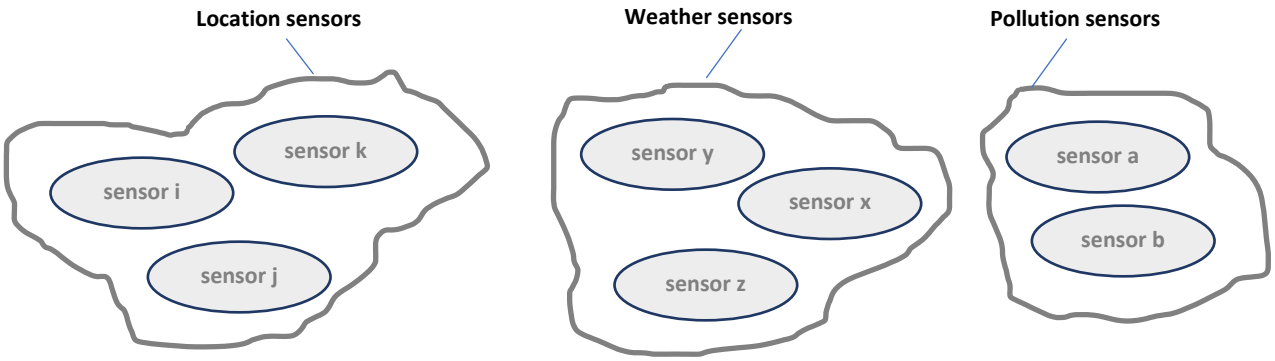


Figura 1. Envío de datos del sensor a través de llamadas telefónicas

Entonces, tenemos un conjunto de n categorías $\{c_1, c_2, c_3, \ldots, c_{n-1}, c_n\}$, donde cada categoría $\mathbf{c}_i$ tiene un conjunto predefinido de $\mathbf{m}_j$ medidas físicas que deben informarse al lugar central. Más sensores pueden pertenecer a una categoría, pero un sensor pertenece a una y solo una categoría. De esta forma, podemos decir que para cada sensor $\mathbf{s}_i$, que pertenece a una categoría $\mathbf{c}_j$ (donde $0 < j < n+1$), tenemos un conjunto $\mathbf{S}_i$ de $\mathbf{m}_j$ mediciones físicas, que deben ser transmitidas:

$$S_i = \{x_{i1}^{C_j},\ x_{i2}^{C_j},\ x_{i3}^{C_j}, \ldots,\ x_{im_j-1}^{C_j},\ x_{im_j}^{C_j}\} \ldots (1)$$

Si, por ejemplo, el sensor 4 y el sensor 6 pertenecen a la categoría 2, que contiene cinco mediciones físicas, a estos sensores asociaremos los conjuntos S_4 y S_6 de la siguiente manera:

$$S_4 = \{x_{4_1}^{C_2},\ x_{4_2}^{C_2},\ x_{4_3}^{C_2},\ x_{4_4}^{C_2},\ x_{4_5}^{C_2}\} \ldots (2)$$

$$S_6 = \{x_{6_1}^{C_2},\ x_{6_2}^{C_2},\ x_{6_3}^{C_2},\ x_{6_4}^{C_2},\ x_{6_5}^{C_2}\} \ldots (3)$$

3. Arquitectura

La figura 2 ilustra nuestra arquitectura: suponemos que, por un lado, tenemos diversos sensores que pueden emitir información diversa, que están equipados con números de teléfono y pueden hacer llamadas telefónicas.

Y, por otro lado, tenemos una estación centralizada con una cierta cantidad de números telefónicos que se utiliza para recuperar la información de las llamadas telefónicas realizadas a esta estación.

Por lo tanto, en el lado de los sensores, cada sensor debe codificar de algún modo la información que desea transmitir. Esta codificación debe realizarse a través de una lista de llamadas telefónicas posteriores en la estación central, que luego utiliza esta lista de llamadas para decodificar la información obtenida y enviada por un sensor.

La estación central tiene una base de datos de todos los sensores que pueden hacer llamadas, que incluye sus números de teléfono, junto con la categoría a la que pertenecen. De esta forma, el número de teléfono de la persona que llama automáticamente nos dará información sobre la identificación del sensor (qué sensor) y la categoría (tipo de datos enviados). Significa que no necesitamos codificación para estos dos elementos. La única información que necesita codificación es el valor de los datos. Como hay más llamadas para codificar los datos, debemos identificar qué llamada pertenece a qué parte de la información.

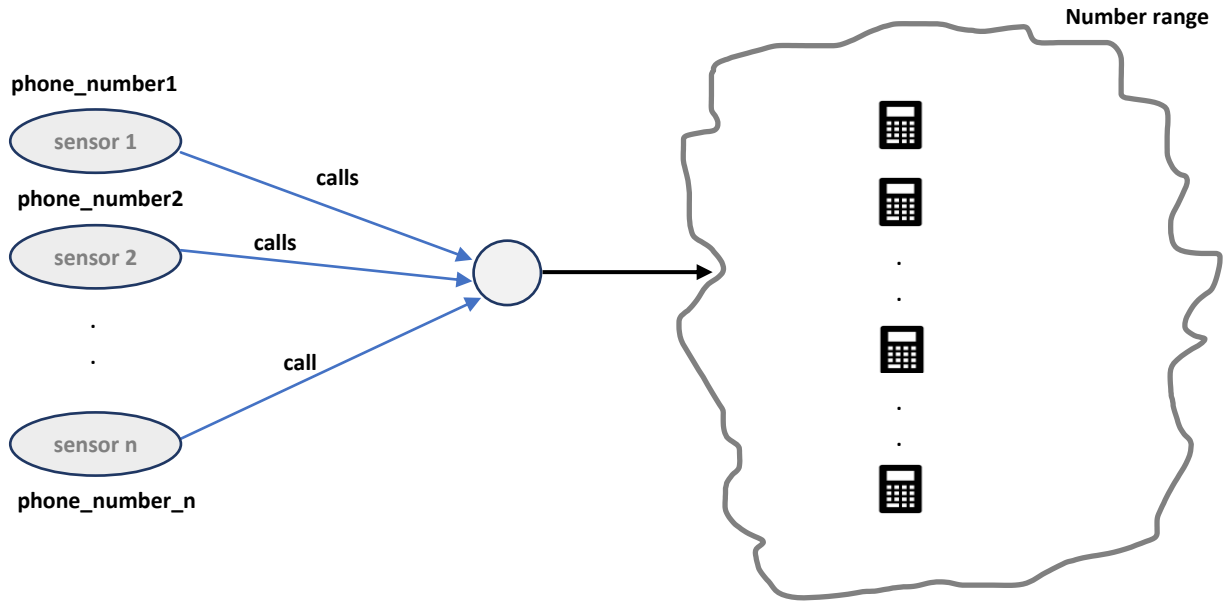


Figura 2. Distribución de bits para diferentes categorías dependiendo de las medidas físicas que cada categoría contiene

4. Algoritmo para codificar y decodificar datos de sensores

El recurso principal para nuestra arquitectura es una lista de números de teléfono, separados con el único propósito de recibir llamados. Normalmente, tienen una parte fija (prefijo) para los primeros dígitos **K.** De esta forma, todo lo que queda para la codificación son dígitos *N - K*, donde *N* es la longitud de los números telefónicos. Esta parte se llamará *extensión* más adelante. Por ejemplo, si tenemos una lista de 10000 números, significa que podemos usarla para codificar todos los números en el rango [0000...9999] [1].

Nuestro algoritmo de codificación/descodificación se basa en sistemas numéricos que representan valores como una lista de bits. Como un sensor puede emitir más de un valor y diferentes valores pueden tener diferentes longitudes, el número de bits utilizado para cierta descripción del valor dependerá del valor en sí. Por ejemplo, la *longitud*, que tiene aproximadamente 9 dígitos, se puede describir con 32 bits, mientras que la *velocidad*, que puede ser un número máximo de 180 millas/hora, se puede describir con 8 bits.

En general, para la categoría $\mathbf{C}_j$, si su valor **i-ésimo** (medición física) $\boldsymbol{x}_i^{C_j}$ se describe con **m** bits, entonces todos los valores potenciales estarán en el rango [0000...2^m-1]. Y viceversa, el rango disponible determina cuántos valores puede describir nuestro sistema. Por lo tanto, si queremos usar 2048 valores diferentes, podemos representar todos los números posibles en el intervalo [0000...2047] con 11 bits, dado que 2^{11} = 2048. Con 13 bits, podemos representar todos los valores potenciales dentro del rango [0000...8191], como 2^{13} = 8192.

Para cada sensor, dependiendo de las medidas físicas que emite, la distribución de los bits se reorganiza. La figura 3 proporciona una imagen generalizada de esta distribución, para sensores de **m** categorías diferentes:

[1] El operador del teléfono definirá el número de teléfono inicial, lo importante es que los números de teléfono sean una lista de números consecutivos

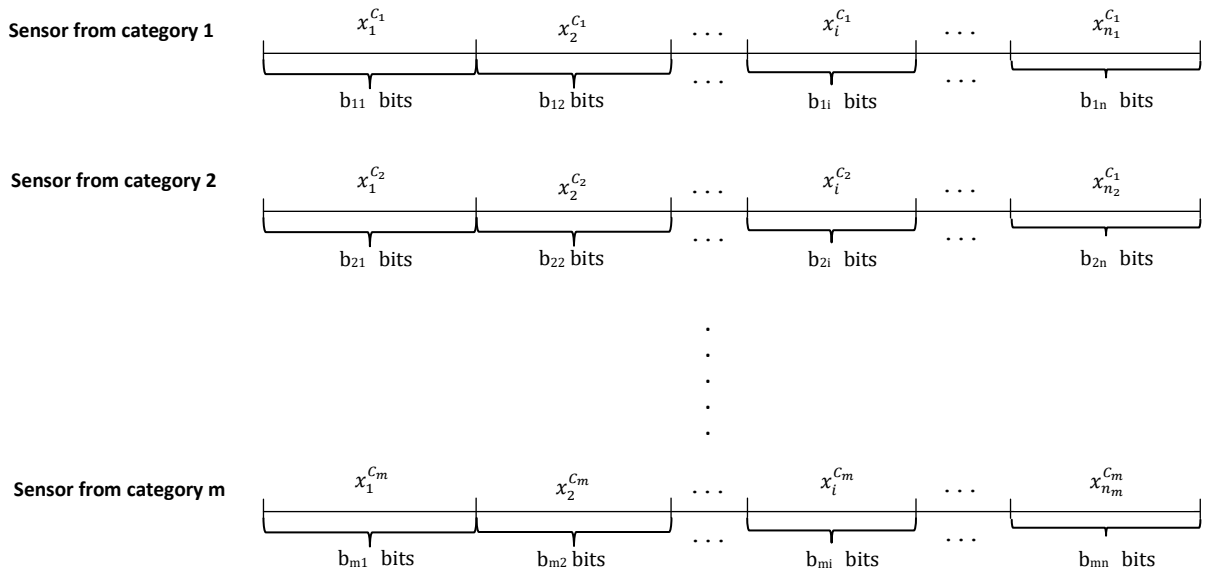


Figura 3. Distribución de bits para diferentes métricas

La Figura 4 nos da una imagen de esta distribución para un sensor que brinda información sobre:

- temperatura, con valores [-50, 170]
- humedad, valor s en [900, 1100]
- viento, valores en [0, 255]
- dirección, valores e n [0, 360]

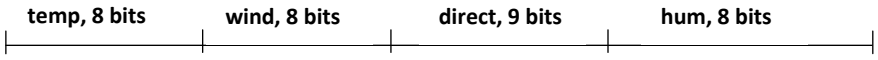


Figura 4. Un sensor que emite valores de temperatura, humedad, viento y dirección

Hay valores con pocos dígitos y, si este es el caso, solo una llamada puede transmitirlos. Pero también hay valores con mayor número de dígitos, que necesitan más llamadas para transmitir el valor a la estación central. Además, generalmente un sensor proporciona más de una información, por ejemplo, contaminación y humedad. En estas circunstancias, nuevamente tenemos que hacer más llamadas telefónicas. Por otro lado, estas llamadas deberían identificarse a qué parte de la información pertenece. De alguna manera, ahora que el sistema lee la llamada, debe categorizar automáticamente a qué parte de la información pertenece. Con este objetivo, definimos una nueva métrica denominada *desplazamiento*, que asocia el valor con la parte concreta de la información. Se necesita una función que tome el valor de la *llamada del dominio de llamadas telefónicas* C y lo relacione con un valor del *desplazamiento establecido O*:

$$f: C \rightarrow O, f(c) = o \ldots (4)$$

dónde *o* es un número entero que muestra que el valor pertenece a la parte *o-ésima* de la información , mientras que O es un conjunto ordenado de valores enteros de 1 a P, calculado como:

$P = int(NR/256)$, donde NR es el rango de números... (5)

Como en la mayoría de los casos el rango de números no es un múltiplo de 256, quedan algunos bits, que pueden usarse para enviar información adicional. Si denotamos esta parte por T, tenemos que:

$$T = fractional(NR/256) * 256 \ldots (6)$$

Si hay 10000 números de teléfono disponibles en la parte central para llamar, entonces:

P = int (10000/256) = 39, por lo que habrá 39 espacios . Y también quedará una ranura de 4 bits (0.0625 * 256 = 16 = 2^4).

Digamos que tenemos **t** mediciones físicas para una categoría $\mathbf{C_j}$, que tienen valores V_1, V_2, ..., V_t, en algún punto del tiempo, donde $2^0 \leq V_i \leq 2^{16}$. Aproximadamente, si un sensor envía **t** valores, y cada valor *i-ésimo* ocupa un número predefinido de b_i bits, entonces el número total de bits necesita expresar todos los valores es igual a $b_1+b_2 + \ldots + b_t$. La Figura 5 da una representación de esta configuración:

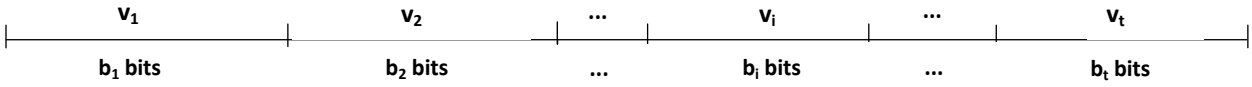


Figura 5. Un sensor que emite diferentes valores para t métricas diferentes, cada uno ocupando un número predefinido de bits

En esta línea, la extensión se dividirá en dos partes: la parte identificatoria y la parte del valor. Los primeros **n** bits se usará para el identificador y los **m** bits restantes se reservarán para el valor. La idea detrás del identificador que ahora que se lee su valor, automáticamente indicará la posición a la que pertenece esa parte de la información. Significa que dará el desplazamiento en el eje horizontal de la Figura 5.

Entonces, necesitamos una función que proyecte cada valor $\mathbf{v_i}$, dependiendo de su desplazamiento a un valor igual a $\mathbf{x_i}$, que servirá para las llamadas telefónicas. El desplazamiento depende del número de bits utilizados para las llamadas anteriores en el hacha definido en la Figura 5:

$$OFFSET_j = \sum_{i=0}^{j-1} 2^{b_i} \ldots (7)$$

Ahora, los valores $\mathbf{x_i}$ se calculan de la siguiente manera:

$$x_j = \sum_{i=0}^{j-1} 2^{b_i} + V_{x_j} \ldots (8)$$

4.1. Codificación

1. Como primer paso para codificar el número fuente es su multiplicación por 10^x, donde x es el número de dígitos después del lugar decimal que queremos considerar, dependiendo del número de bits que queremos ocupar para esa medición física. Por ejemplo, si tenemos una información que tiene ocho dígitos después del lugar decimal, pero queremos considerar solo los primeros seis dígitos, entonces este número debe multiplicarse por 1000000. La constante de multiplicación está predefinida para cada métrica en particular. Si no hay dígitos después del lugar decimal, se omite el primer paso.

2. Después de la operación de multiplicación, tomamos solo la parte entera del número ganado. Si la parte entera es un número positivo, pasamos al siguiente paso. De lo contrario, negamos usando los lógica de complemento dos mediante la adición de 2^b, donde b es el número de bits utilizados para la representación. Por ejemplo, si se utilizan 32 bits, se agregará el número 4294967296, que es igual a 2^{32}. Otra posibilidad es cambiar los valores de la parte negativa a la positiva obteniendo así un nuevo rango de valores que comienza desde 0. Si el rango es digamos [-r, p], después del cambio, se convertirá en [0, p + r].
3. Convertimos el resultado obtenido del paso anterior en su equivalente hexadecimal. Los bytes de este valor se separan y se usan para hacer una lista de llamadas telefónicas para transmitir el valor. Si el rango del número de teléfono es como mucho 10000, entonces las llamadas serán byte por byte. Esto significa que a cada byte corresponde una sola llamada telefónica a la estación central. De lo contrario, si tenemos más números de teléfono disponibles para llamadas, podemos hacer llamadas telefónicas usando dos bytes sucesivos.
4. Como último paso, el desplazamiento se agrega a cada byte devuelto en el tercer paso de nuestro algoritmo de codificación, respectivamente, a su posición. De esta forma, si tenemos ranuras de 8 bits por medición física, entonces al primer byte se agrega 0, al segundo byte se agrega 256, al tercer byte se agrega 512, al cuarto 768 y así sucesivamente.

4.2. Decodificación

El proceso de decodificación es directo:

1. Tomamos las llamadas telefónicas y comenzamos a decodificar para cada llamada de manera independiente (el orden no es importante).
2. Para cada llamada, extraemos los últimos X dígitos (X depende del rango numérico). Verificamos el desplazamiento, tratando de identificar a qué parte de la información pertenece. Si es mayor que el $desplazamiento_i$ menos que el $desplazamiento_{i+1}$, nos da que este valor se refiere a la *i-ésima* parte del valor métrico general. Por ejemplo, si es mayor que 768 y menor que 1023, significa que se refiere a la cuarta parte de la medición global . Cuando definimos el orden, restamos el $desplazamiento_i$ del valor extraído. El resultado se convierte a su valor hexadecimal. El orden de decodificación no es importante, las llamadas se decodifican de forma independiente entre sí.
3. Una vez finalizado el procesamiento de todas las llamadas telefónicas, dado que se conoce el orden de todos los números hexadecimales, se fusionan según el orden.
4. Comparamos el número final con el mayor número positivo (valor máximo que se puede representar en int largo sin signo) que se ajustará en **b** bits cuando se utiliza la notación[1] de "complemento a dos". Si es más grande, restamos el número por 2^{b-1} y lo dividimos por 10^x, donde x también se define en la parte de codificación. Si no, el número final simplemente se divide por 10^x. El valor devuelto es el valor original obtenido en el lado del sensor.

Ejemplo:

Tomemos un valor igual a *-67.7264727* de longitud para ilustrar cómo funciona la codificación/descodificación para un valor concreto, suponiendo que hemos utilizado 32 bits para este sensor en particular:

[1] Este paso solo es necesario si codifica números negativos..

Codificación:

1. Como en este caso hay 7 dígitos después del lugar decimal, lo multiplicamos por 10^7, lo que da como resultado -677264727.
2. Como -677,264,727<0, calculamos el complemento dos mediante la adición de 2^{32} = 4294967296, que da salida a 3617702569.
3. Como el valor hexadecimal para 3617702569 es D7A1C2A9, el primer byte es D7, cuyo equivalente decimal es igual a 215. Procesando de manera similar para los cuatro bytes obtenemos los números 215, 161, 194, 169.
4. Agregamos el dígito identificador a cada byte correspondientemente y ahora nuestra lista de números es 215, 417, 706, 937 (215 = 0 + 215, 417 = 161 + 256, 706 = 194 + 512, 706 = 169 + 768).

Por lo tanto, estos números se usarán ahora para hacer cuatro llamadas que finalizan con los valores recién calculados. El lado del sensor hará estas llamadas a la estación central. Aquí las llamadas deben decodificarse para revertir el valor original.

Decodificación:

1. Tomamos las llamadas telefónicas, que en este caso son cuatro, que tienen el formato ******* 215, ******* 417, ******* 706, ***** ** 937.
2. Tomamos los últimos tres dígitos , digamos el segundo valor 417. Ahora verificamos el rango y concluimos que es más grande que 256 y menos que 511. Significa que el orden es 2 y necesitamos menos 256 de 417 resultando en de esta manera con el valor 161. Ahora lo convertimos a su valor hexadecimal que es A1. De manera similar, si tomamos la última llamada para el valor 937, después de definir el orden y la resta, lo convertimos a su equivalente hexadecimal A9. Usando la misma lógica, tenemos para la primera llamada el equivalente hexadecimal D7 y el tercero, C2.
3. Fusionamos los números hexadecimales por orden. El número final es D7A1C2A9.
4. El equivalente decimal de D7A1C2A9 es 3617702569.
5. Desde 2^{31} = 3617702569 > 2147483648, tenemos:
   (3617702569 – 4294967296) / 10^7 = -677264727 / 10^7 = **-67.7264727** , que es igual al valor original del lado del sensor.

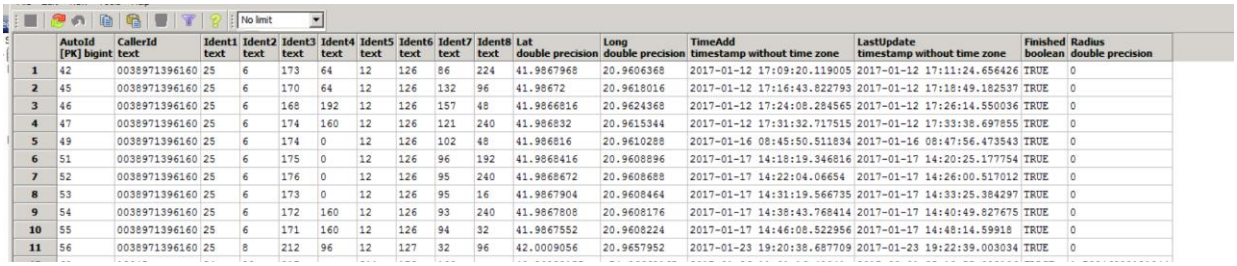
Toda la descripción anterior fue sobre un solo valor. Si un sensor envía más de un valor, la numeración de orden de los valores subsiguientes continuará en el extremo anterior. Por ejemplo, si el sensor envía un valor de latitud, el orden de la latitud comenzará desde cinco, ya que necesitamos cuatro llamadas para la medida de longitud. Todo lo demás, tiene la misma lógica.

5. Implementación y resultados

Hemos probado nuestro algoritmo usando el dispositivo Libelium para enviar datos de ubicación a un servidor central, donde utilizamos Asterisk, un marco para construir aplicaciones de comunicaciones[2] . Este marco utiliza un plan de marcado para atender las llamadas telefónicas que reconoce. En nuestro caso, ahora que se reconoce una llamada, llamamos a una secuencia de comandos que tiene el número de teléfono y la extensión como argumentos de entrada.

[2] http://www.asterisk.org/

Este script luego decodifica el valor (extensión) y coloca el valor decodificado en una base de datos (en este caso se implementa en PostgreSql) que contiene todos los datos emitidos para diferentes sensores:



	AutoId [PK] bigint	CallerId text	Ident1 text	Ident2 text	Ident3 text	Ident4 text	Ident5 text	Ident6 text	Ident7 text	Ident8 text	Lat double precision	Long double precision	TimeAdd timestamp without time zone	LastUpdate timestamp without time zone	Finished boolean	Radius double precision
1	42	0038971396160	25	6	173	64	12	126	86	224	41.9867968	20.9606368	2017-01-12 17:09:20.119005	2017-01-12 17:11:24.656426	TRUE	0
2	45	0038971396160	25	6	170	64	12	126	132	96	41.98672	20.9618016	2017-01-12 17:16:43.822793	2017-01-12 17:18:49.182537	TRUE	0
3	46	0038971396160	25	6	168	192	12	126	157	48	41.9866816	20.9624368	2017-01-12 17:24:08.284565	2017-01-12 17:26:14.550036	TRUE	0
4	47	0038971396160	25	6	174	160	12	126	121	240	41.986832	20.9615344	2017-01-12 17:31:32.717515	2017-01-12 17:33:38.697855	TRUE	0
5	49	0038971396160	25	6	174	0	12	126	102	48	41.986816	20.9610288	2017-01-16 08:45:50.511834	2017-01-16 08:47:56.473543	TRUE	0
6	51	0038971396160	25	6	175	0	12	126	96	192	41.9868416	20.9608896	2017-01-17 14:18:19.346816	2017-01-17 14:20:25.177754	TRUE	0
7	52	0038971396160	25	6	176	0	12	126	95	240	41.9868672	20.9608688	2017-01-17 14:22:04.06654	2017-01-17 14:26:00.517012	TRUE	0
8	53	0038971396160	25	6	173	0	12	126	95	16	41.9867904	20.9608464	2017-01-17 14:31:19.566735	2017-01-17 14:33:25.384297	TRUE	0
9	54	0038971396160	25	6	172	160	12	126	93	240	41.9867808	20.9608176	2017-01-17 14:38:43.768414	2017-01-17 14:40:49.827675	TRUE	0
10	55	0038971396160	25	6	171	160	12	126	94	32	41.9867552	20.9608224	2017-01-17 14:46:08.522956	2017-01-17 14:48:14.59918	TRUE	0
11	56	0038971396160	25	8	212	96	12	127	32	96	42.0009056	20.9657952	2017-01-23 19:20:38.687709	2017-01-23 19:22:39.003034	TRUE	0

Figura 6. Base de datos de los datos del sensor emitido

Para cada llamada telefónica, se agrega un nuevo registro, llenando uno por uno todos los valores para cada identificador por separado. El dispositivo Libelium se programa de tal manera que inicialmente se envía el primero, la mayoría de los valores de longitud y latitud:

```
USB.print("La4: ");
numberToCall(Lat[0]);
USB.print(" - ");
USB.println(Lat[0]);
makeCall();
```

De esta manera, incluso con las dos primeras llamadas, tenemos una aproximación sobre la ubicación. Por lo tanto, cualquiera que sea la aplicación que utiliza esta información, se puede programar para que se base en eventos y se actualice para cada llamada posteriormente. Para demostrar esto, hemos creado una aplicación web simple que muestra la posición actual del sensor en un mapa, dependiendo de las últimas llamadas telefónicas (Figura 7).Nuestra aplicación está basada en eventos, que se notifica cada vez que se agrega un nuevo valor (identificatorio) a la base de datos, con respecto al sensor que se observa.

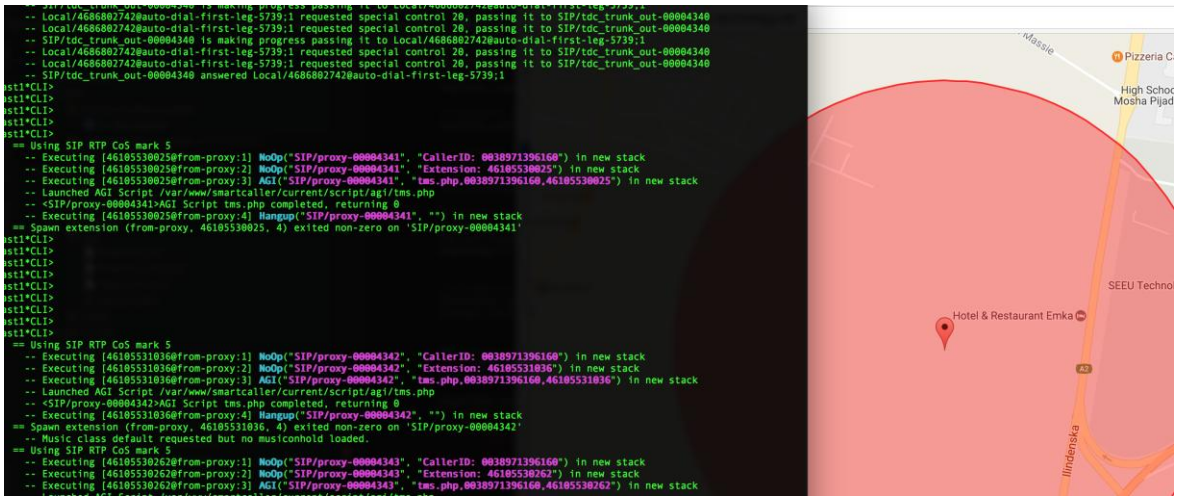


Figura 7. Ejecución en tiempo real de nuestro método

También medimos el tiempo de rendimiento requerido para hacer veinte llamadas sucesivas de datos del sensor, junto con su tiempo de codificación (tenemos limitaciones de infraestructura en esta etapa de la investigación). La Tabla 1 da una ilustración de los resultados obtenidos. El tiempo promedio necesario para establecer una llamada es de aproximadamente 5 segundos. Tomando eso en nuestro caso, tenemos disponible la extensión de teléfono que va desde 0000 hasta 9999, con una llamada podemos transferir aproximadamente 13.3 bits. De esto podemos concluir que para este canal de comunicación la capacidad es de 13.3 bits / 5 seg = 2.66 bits/seg.

Tabla 1. Medición del tiempo de rendimiento

Intento	Número de teléfono	Tiempo necesario para establecer una llamada
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

6. Conclusiones y trabajo adicional

En este documento, presentamos una nueva forma de enviar datos de sensores a una ubicación central sin usar canales de comunicación tradicionales y hemos demostrado que este tipo de transporte de datos de sensores es completamente funcional. Este enfoque para la transmisión de los registros del sensor puede dar lugar a posibilidades ilimitadas de aplicaciones y modelos de negocio, tales como la telemática, la medición inteligente (eléctricidad, gas, agua, aire/contaminación, etc…), automotriz (diagnóstico de automóviles, ejecución remota de comandos, etc..), domótica, etc... En nuestro trabajo posterior, planeamos implementar un algoritmo que ejecutará comandos utilizando llamadas telefónicas en la dirección opuesta, desde el receptor a un sensor, como por ejemplo podría ser "*dame información sobre el estado de las luces*". Además, tenemos la intención de explorar formas alternativas de transmitir datos en modo de llamada.

7. Referencias

[1] Luigi Atzori, Antonio Iera, Giacomo Morabi para (2010) El Internet de las cosas: Una encuesta. The International Journal of Computer and Telecommunications Networking, Volumen 54, Número 15, páginas 2787-2805.

[2] Andrew Whitmore, Anurag Agarwal, Li Da Xu (2014) Internet de las cosas: una encuesta de temas y tendencias. Information Systems Frontiers, Volumen 17, Número 2, páginas 261-274.

[3] Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos (2013) Contexto consciente de la informática para Internet de las cosas: una encuesta. Encuestas y tutoriales de comunicaciones IEEE , Volumen: 16 Número: 1, pp. 414 - 454.

[4] MA Matin, MM Islam (2012) Descripción general de la red de sensores inalámbricos. Redes de sensores inalámbricos: tecnología y protocolos, Libro Capítulo 1.

[5] Chris Guy (2006) Redes de sensores inalámbricos. Actas del Sexto Simposio Internacional sobre Instrumentación y Tecnología de Control: Análisis de Señal, Teoría de Medición, Tecnología Fotoelectrónica e Inteligencia Artificial.

[6] FL Lewis (2004) Redes de sensores inalámbricos. Entornos inteligentes: tecnologías, protocolos y aplicaciones (eds DJ Cook y SK Das), John Wiley & Sons, Inc., Hoboken, NJ, EE. UU.

[7] John A. Stankovic (2008) Wireless Sensor Networks. Computador, Volumen: 41, Número: 10.

[8] Chris Townsend, Steven Arms (2005) Redes de sensores inalámbricos: Principios y aplicaciones . Sensores, transductores y detectores, Capítulo 22 del libro.

[9] SI Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci (2002) Wireless redes de sensores: una encuesta. Computer Networks, Volumen 38, Número 4, páginas 393-422.

[10] Botts, M .; Robin, A. (2007) Juntando la red del sensor. Geosciences, pp. 46-53.

[11] Klaithem Al Nuaimi, Mariam Al Nuaimi, Nader Mohamed, Imad Jawhar, Khaled Shuaib (2012) Redes de sensores inalámbricos basados en la web: una encuesta de arquitecturas y aplicaciones. Actas de la 6ª Conferencia Internacional sobre Gestión y Comunicación de la Información ubicua, Artículo No. 113.

[12] Telecomunicaciones e intercambio de información entre sistemas (ISO / IEC JTC 1 / SC 6), https://www.iso.org/committee/45072.html)

[13] Iniciativa de estándares mundiales de internet de las cosas (IoT-GSI) , http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx

[1 4] Actividad conjunta de coordinación sobre la Internet de las cosas y las ciudades inteligentes y Comunidades (JCA IoT y SC & C), https://www.itu.int/en/ITU-T/jca/iot/Documents/ToR/Scope-and-ToR-JCA-IoT-SC-C-18112015.pdf

[15] Actividad conjunta de coordinación sobre Internet de las cosas (JCA-IoT), http://www.geneve-int.ch/joint-coordination-activity-internet-things-jca-iot

[16] Weilian Su, Ozgiir B. Akan, Erdal Cayirci (2006) Protocolos de comunicación para el Sensor Redes. Redes de sensores inalámbricos, Libro Capítulo 2.

[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.