Capire e usare LoRa®

6

di PIER CALDERAN

In questa sesta e ultima puntata spieghiamo come usare la tecnologia LoRaWAN abbinata alla geolocalizzazione tramite GPS.

Siamo dunque giunti alla conclusione del nostro corso dedicato alla tecnologia LoRa e all'implementazione di reti wireless low-power LoRaWAN: dopo avervi introdotto lo standard di comunicazione radio LoRa ed aver progettato e proposto dell'hardware con cui sperimentare la tecnologia attraverso tutta una serie di applicazioni didattiche, tra cui un nodo (dispositivo finale) ed un gateway, procediamo e concludiamo con una nuova applicazione pratica. Dunque, sfruttando il nostro dispositivo finale LoRa e un piccolo modulo ricevitore GPS, vi spiegheremo come costruire un sistema di geolocalizzazione basato sulla rete satellitare globale che può esservi utile in molteplici situazioni, anche in virtù della possibilità di comunicare i dati di localizzazione attraverso la rete LoRa.
Il modulo GPS che useremo per il nostro esperimento è basato sul chip NEO-6, prodotto dall'azienda svizzera U-blox (sito ufficiale *https://www.u-blox.com/en/product/neo-6-series*) già nota per i suoi prodotti rivolti alla localizzazione satelitare. Questo chip ha dimensioni davvero ridotte, in rapporto alle prestazioni e frequentemente si trova in commercio montato in una breakout board dedicata (**Fig. 1**). Tale scheda è peraltro disponibile presso la Futura Elettronica (*www.futurashop.it*) con il codice prodotto 2846-GPSNEO6M sula quale si trova il modulo GPS U-Blox NEO-6M, corredato di batteria di backup per i dati e antenna in ceramica.
Di seguito le caratteristiche tecniche del nostro modulo.

- Chipset: U-BLOX NEO-6M;
- Banda di lavoro: L1 (1.575,42 MHz).
- 50 canali con oltre 1 milione di correlatori effettivi;
- Sensibilità in Tracking: -161 dBm.
- Sensibilità in acquisizione: -148 dBm.
- Dati predefiniti: WGS84.
- Riacquisizione: 0,1 secondi (media).
- Hot start: 1 secondo (media).
- Warm start: 35 secondi.

Fig. 1 - Il modulo breakout con il chip NEO-6.

- Cold start: 38 secondi.
- C/A code: 1.023 MHz stream.
- Altitudine: 18.000 metri (60.000 piedi max).
- Velocità massima: 515 metri/secondo
- Accelerazione: < 4g.
- Frequenza di aggiornamento della posizione: ogni 5 Hz.
- Uscita seriale: 9.600 Baud, 8N1
- Uscita dati: formato NMEA predefinito per U-blox 6 (GGA, GLL, GSA, GSV, RMC e VTG).
  Contrariamente a ANTARIS 4, ZDA è disabilitato per impostazione predefinita.
- Batteria di backup per i dati: Sì.
- Collegamenti: VCC, GND, TX, RX.
- Alimentazione: da 3 Vcc a 5 Vcc.
- Dimensioni del modulo (LxAxP): 30x23x4 mm.
- Dimensioni Antenna (LxWxH): 25x25x8 mm.
- Temperatura di funzionamento: da -40°C a +80°C.

COLLEGAMENTO AL DISPOSITIVO FINALE LORA

Fra le caratteristiche del modulo GPS leggiamo che la comunicazione è seriale, quindi ideale per la connessione con il nostro chip ATmega328 del dispositivo finale LoRa.

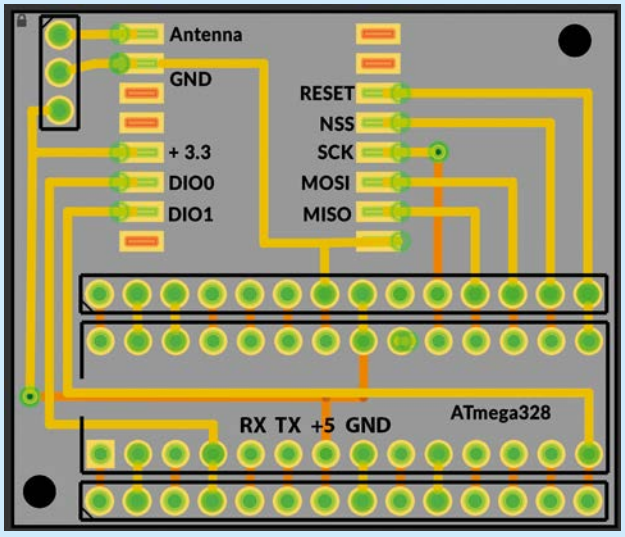

Fig. 2 - Il modulo SC102 con ATmega328.

Il collegamento è davvero molto semplice e presuppone che abbiate costruito il dispositivo finale seguendo le nostre istruzioni della prima puntata del corso o abbiate montato il kit LoRa acquistato da Futura Elettronica (shop on-line *www.futurashop.it*, codice prodotto: 8350-SC102). In ogni caso, facendo riferimento alla **Fig. 2**, effettuate questi collegamenti:

- pin 5 contrassegnato come RX al pin TX del GPS;
- pin 6 contrassegnato come TX al pin RX del GPS;
- pin 7 contrassegnato come +5 al pin VCC del GPS;
- pin 8 contrassegnato come GND al pin GND del GPS.

LO SKETCH LORAWAN GPS TRACKER

Lo scopo del nostro esperimento è quello di leggere i dati di geolocalizzazione dalla porta seriale del modulo GPS e inviarli al gateway LoRaWAN, che si incaricherà di mandarli a TTN. Prendendo spunto dallo sketch LoRaWAN, che abbiamo usato tante volte nelle scorse puntate, è bastato creare una funzione di comunicazione seriale con il modulo GPS. Una volta inviati i pacchetti al gateway, vedremo poi come utilizzare i dati GPS per visualizzare la posizione del nostro dispositivo finale su una mappa di Google Maps. Per fare questo, avete bisogno della libreria open source TinyGPS++ di Mikal Hart, disponibile nel repository GitHub seguente:

https://github.com/mikalhart/TinyGPSPlus

La libreria TinyGPS++ può essere installata manualmente, oppure da file zip. Nella gestione librerie di Arduino è possibile installare la versione precedente TinyGPS, che però ha qualche problema.
Nel **Listato 1** potete vedere la funzione aggiunta al codice originale per la rilevazione delle coordinate GPS.
Oltre a questa libreria, abbiamo pensato di usare anche la libreria SoftwareSerial (già integrata nell'IDE di Arduino) che, come ben sappiamo, permette di usare due pin digitali qualsiasi come porta seriale TX/RX. In questo modo potete sfruttare la UART dell'ATmega328 per effettuare il debug sul monitor seriale. Nell'istanza "ss" creata da SoftwareSerial abbiamo previsto l'uso del pin 3 e del pin 4 rispettivamente come RX e TX. Ricordiamo che i pin fisici 5 e 6 dell'ATmega328 corrispondono ai pin digitali 3 e 4 di Arduino.

FORMATO LPP (LOW POWER PAYLOAD)

Per l'invio dei dati GPS useremo il formato LPP (Low Power Payload) perché il cloud myDevices (*https://mydevices.com*), altrimenti conosciuto come Cayenne, utilizza questo formato. Come vedremo, attiveremo un account Cayenne myDevices per visualizzare la mappa con Google Maps con le coordinate esatte del nostro dispositivo.
I dati delle coordinate forniti dal modulo rcevitore GPS hanno una struttura del tipo di quella proposta qui di se-

Listato 1 - FUTURA LORAWAN GPS TRACKER

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

#define LPP_GPS 0x88

static const int RXPin = 3, TXPin = 4;
static const uint32_t GPSBaud = 9600;

TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);

static const PROGMEM u1_t NWKSKEY[16] = { 0xBB, 0x1C, 0x40, 0xDB, 0x68, 0x3D, 0x87, 0x87, 0xF2, 0x8C, 0x89, 0x8B, 0xD7, 0x89, 0x2F, 0xD4 };
static const u1_t PROGMEM APPSKEY[16] = { 0x60, 0xE5, 0xC2, 0x75, 0xBB, 0xE6, 0xE1, 0xDD, 0xD9, 0x0D, 0x36, 0x57, 0xFE, 0x9E, 0xD6, 0x9C };
static const u4_t DEVADDR = 0x260113A4 ;

const unsigned TX_INTERVAL = 20;

void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

  uint8_t coords[11];
  static osjob_t sendjob;
  static osjob_t initjob;
  uint8_t cursor = 0;
  uint8_t channel;

// Pin mapping
  const lmic_pinmap lmic_pins = {
  .nss = 10,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 9,
  .dio = {2, 8, LMIC_UNUSED_PIN},
};

// Coordinate GPS
void get_coords () {
  bool newData = false;
  unsigned long chars;
  unsigned short sentences, failed;

  unsigned long age;

float latGPS, lonGPS, altGPS, hdopGPS;  // variabili di lettura float dal GPS



  // Analizziamo per un secondo i dati GPS e riportiamo alcuni valori chiave
  for (unsigned long start = millis(); millis() - start < 1000;) {

    while (ss.available()) {
      char c = ss.read();

      if (gps.encode(c)) {
        newData = true;
      }
    }
  }
 if ( newData ) {
    latGPS=gps.location.lat();
    lonGPS=gps.location.lng();
    if (gps.altitude.isValid())
         altGPS = gps.altitude.meters();
     else
        altGPS=0;
    hdopGPS = 0; //gps.hdop.value();
}

int32_t latitude = latGPS * 10000;  // variabile intera signed a 32 bit per la latitudine
```

```
int32_t longitude = lonGPS * 10000; // variabile intera signed a 32 bit per la longitudine
int16_t altitude = altGPS * 100; // variabile intera signed a 16 bit per l'altitudine
int8_t hdop = hdopGPS; // variabile intera a 8 bit HDOP ignorato, chiude il conteggio di 11 byte)

  // array nel formato LPP
coords[0] = 0x01;  // channel
coords[1] = LPP_GPS;  // type
coords[2] = latitude >> 16; // primo byte latitudine
coords[3] = latitude >> 8; // secondo byte latitudine
coords[4] = latitude; // terzo byte latitudine
coords[5] = longitude >> 16; // primo byte longitudine
coords[6] = longitude >> 8; // secondo byte longitudine
coords[7] = longitude; // terzobyte longitudine
coords[8] = altitude; // primo byte altitudine
coords[9] = altitude >> 8; // secondo byte altitudine
coords[10] = hdop; // primo byte HDOP

// Monitor seriale con i dati in formato LPP
Serial.println(coords[0], HEX);
Serial.println(coords[1], HEX);
Serial.println(coords[2], HEX);
Serial.println(coords[3], HEX);
Serial.println(coords[4], HEX);
Serial.println(coords[5], HEX);
Serial.println(coords[6], HEX);
Serial.println(coords[7], HEX);
Serial.println(coords[8], HEX);
Serial.println(coords[9], HEX);
Serial.println(coords[10], HEX);
}
```

guito, almeno per quanto riguarda la latitudine, la longitudine e l'altitudine:

LATITUDINE	LONGITUDINE	ALTITUDINE
45.863067	9.989356	508.90 971

Come potete vedere, si tratta di valori *float* che devono per forza essere convertiti in *integer* per poter essere inviati in una trasmissione seriale. In altre parole, è necessario che i pacchetti inviati dal dispositivo finale al gateway debbano essere formattati in un array di byte. Dalla documentazione di Cayenne leggiamo le indicazioni del formato Low Power Payload (LPP), traducibile come "carico utile a bassa potenza", creato appositamente per non "pesare" nella trasmissione dei dati al cloud. Il Cayenne Low Power Payload fornisce un modo comodo e semplice per inviare dati su reti LPWAN come LoRaWAN. Il Cayenne LPP è conforme alla limitazione della dimensione del carico utile, che può essere ridotta fino a 11 byte (nel caso del GPS) e consente al dispositivo di inviare più dati del sensore contemporaneamente. Inoltre, il Cayenne LPP consente al dispositivo di inviare diversi dati del sensore in diversi frame.
Per fare ciò, ogni dato del sensore deve essere preceduto da due byte:

- **Data channel**: identifica in modo univoco ogni sensore nel dispositivo attraverso i frame, per es. "GPS".
- **Data type:** identifica il tipo di dati nel frame, per esempio "gradi".

Il payload, ossia il contenuto utile di dati nella stringa generata, ha la struttura mostrata qui di seguito:

1 byte	1 byte	n byte	1 byte	1 byte	m byte	···	···
Data1 Ch.	Data1 Type	Dati1	Data2 Ch.	Data2 Type	Dati2	···	···

TIPI DI DATI

I tipi di dati sono conformi alle linee guida IPSO Alliance Smart Objects, che identifica ogni tipo di dati con un "ID oggetto".
Tuttavia, come mostrato di seguito, viene eseguita una conversione per adattare l'ID oggetto in un singolo byte.
LPP_DATA_TYPE = IPSO_OBJECT_ID - 3200
Ciascun tipo di dati può utilizzare 1 o più byte per inviare i dati in base alla **Tabella 1**.
Quello che ci interessa della tabella è il data type "GPS Location" che è identificato con ID 136 (dec) o 0x88 (hex). Da qui si deduce che se un dispositivo con GPS invia, per esempio un payload di 11 byte di questo tipo:

Payload (hex): 01 88 | 06 76 5f | f2 96 0a | 00 03 e8

Tabella 1

Data type	IPSO	LPP Dec	LPP Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0x0	1	1
Digital Output	3201	1	0x1	1	1
Analog Input	3202	2	0x2	2	0.01 Signed
Analog Output	3203	3	0x3	2	0.01 Signed
Illuminance Sensor	3301	101	0x65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	0x66	1	1
Temperature Sensor	3303	103	0x67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	0x68	1	0.5 % Unsigned
Accelerometer	3313	113	0x71	6	0.001 G Signed MSB per axis
Barometer	3315	115	0x73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	0x86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	0x88	9	Latitude: 0.0001° Signed MSB Longitude: 0.0001° Signed MSB Altitude: 0.01 m Signed MSB

La conversione del payload per l'intestazione è di due byte:
Data channel = 01 (canale dati 1)
Data type = 88 (tipo di dati GPS)
Seguono 9 byte di dati:
Latitude = 06 76 5f = latitudine è 42.3519 gradi (3 byte)
Longitude = f2 96 0a = longitudine sarà -87.9094 gradi -879094 (3 byte)
Altitude = 00 03 e8 = altitudine sarà 10 metri (2 byte)
HDOP = ignorato (1 byte)
In totale, il formato LPP per il GPS è di 11 byte e, una volta capito come costruirlo, è facile creare un array di 11 byte per inviare il pacchetto LoRaWAN. Per prima cosa si dichiarano le variabili float per la lettura seriale dal GPS. Notare che l'ultimo byte relativo a HDOP viene ignorato, anche se comunque viene letto dal modulo GPS e inviato nel pacchetto di 11 byte. Il cosiddetto parametro HDOP, che è l'acronimo di Horizontal Diluition Of Precision, indica la qualità della geometria dei satelliti utilizzati dal ricevitore.

```
float latGPS, lonGPS, altGPS, hdopGPS; //variabili di lettura GPS
int32_t latitude = latGPS * 10000;  // variabile intera a 32 bit per la latitudine
int32_t longitude = lonGPS * 10000; // variabile intera a 32 bit per la longitudine
int16_t altitude = altGPS * 100; // variabile intera a 16 bit per l'altitudine
int8_t hdop = hdopGPS; // variabile intera a 8 bit HDOP ignorato
```

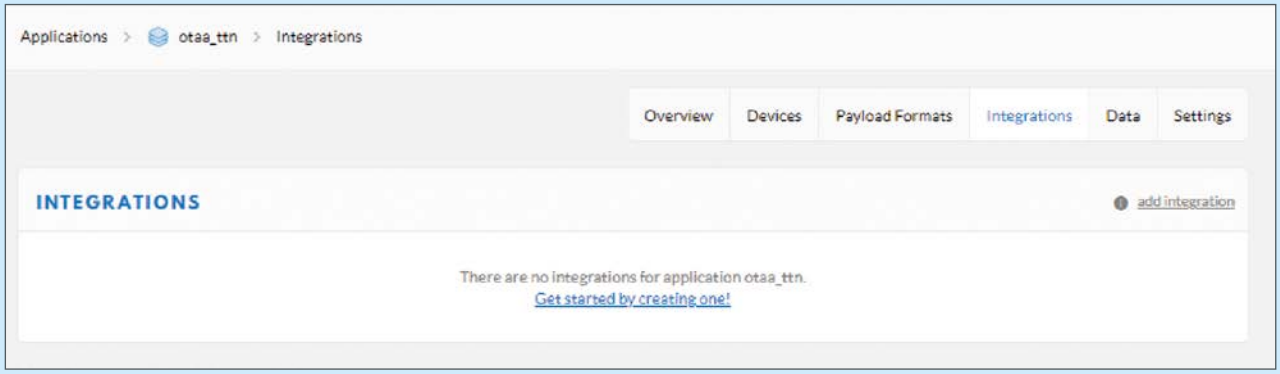

Fig. 3 - Menu Integrations dell'applicazione.

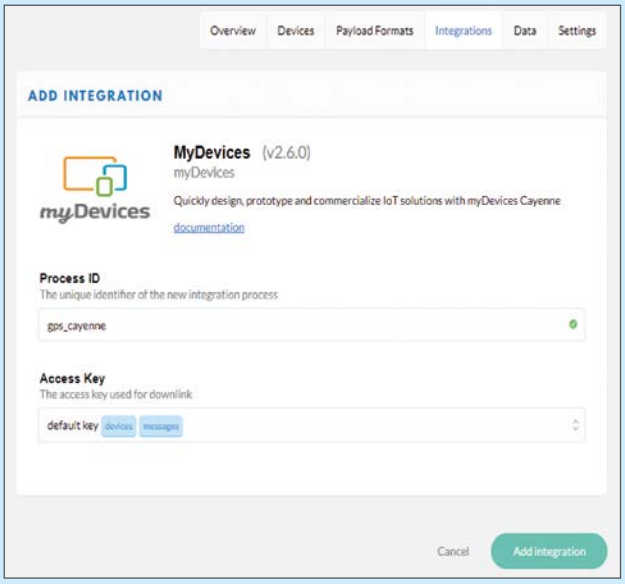

Fig. 4 - Impostazioni di myDevices.

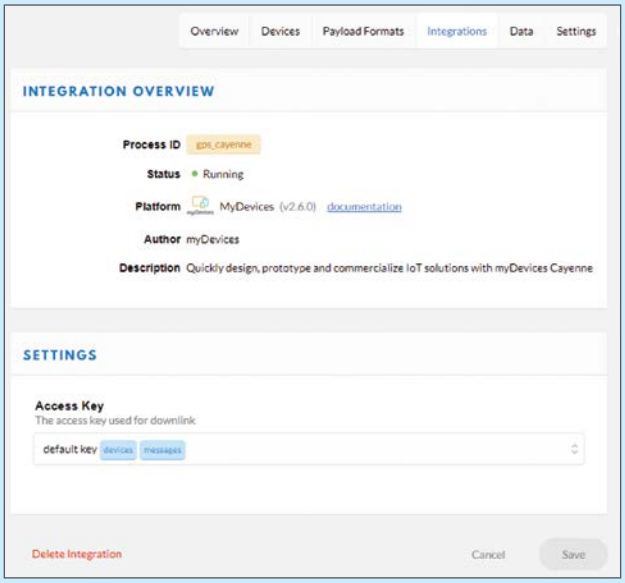

Fig. 5 - L'integrazione myDevices running.

```
// array del formato LPP
coords[0] = 0x01;  // channel
coords[1] = LPP_GPS;  // type
coords[2] = latitude >> 16;  // byte 1 latitudine
coords[3] = latitude >> 8;   //  byte 2 latitudine
coords[4] = latitude;        //  byte 3 latitudine
coords[5] = longitude >> 16; // byte 1 longitudine
coords[6] = longitude >> 8;  // byte 2 longitudine
coords[7] = longitude;       // byte 3 longitudine
coords[8] = altitude;        // byte 1 altitudine
coords[9] = altitude >> 8;   // byte 2 altitudine
coords[10] = hdop;           // byte HDOP
```

Nell'output del monitor seriale potete vedere questo una volta eseguito il bitwise sui byte ricevuti dal GPS:

Starting LPP
Data channel: 1
Data type; 88
Data 1 latitude: 6
Data 1 latitude: FF
Data 1 latitude: 86
Data 2 longitude: 1
Data 2 longitude: 86
Data 2 longitude: 37
Data 3 Altitude: DE
Data 3 Altitude: C1
Data 3 HDOP: 94

Si avranno così valori interi delle coordinate:
6FF86 = 458630 = latitudine
18637 = 99895 = longitudine
DEC1 = 57025 = altitudine

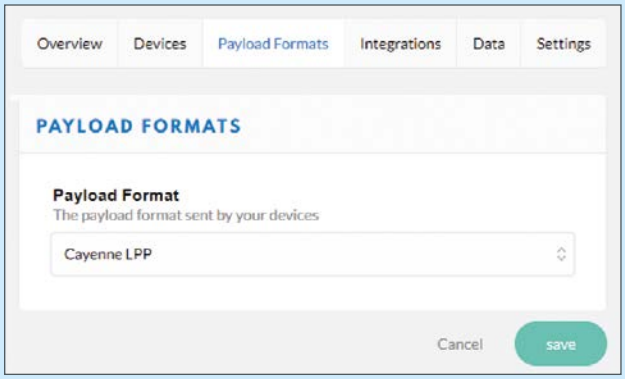

Fig. 6 - Il payload format impostato su Cayenne LPP.

Basta dividere i valori interi per ottenere i decimali:
458630 / 10000 = 45,863 ° latitudine
99895 / 10000 = 9,9895 ° longitudine
57025 / 100 = 570,25 m altitudine

INTEGRAZIONE TTN

The Things Network mette a disposizione diverse integrazioni e a seconda dell'integrazione scelta, un'applicazione di TTN può comunicare con il servizio cloud scelto in maniera differente.
Attualmente (ossia al momento della pubblicazione di questa puntata del corso) TTN mette a disposizione le seguenti integrazioni:

- AllThingsTalk Maker;
- AllThingsTalk;
- Collos;
- Semtech Corporation;

- Data Storage;
- The Things Industries B.V.;
- EVRYTHNG;
- HTTP Integration;
- IFTTT Maker;
- myDevices;
- OpenSensors;
- TTN Mapper;
- JP Meijers;
- TagoIO;
- ThingSpeak;
- MathWorks;
- Ubidots.

AGGIUNGERE L'INTEGRAZIONE A CAYENNE

Fra le molte opzioni, abbiamo scelto myDevices Cayenne perché offre una funzionalità per il GPS tracking davvero completa.
Per configurare l'applicazione per l'inoltro dei dati a Cayenne è necessario prima di tutto aggiungere l'integrazione a un'applicazione di TTN. Per aggiungere l'integrazione Cayenne, seguite i seguenti passaggi.

1. Aprite o create un'applicazione dalla console di TTN (l'applicazione deve avere almeno un dispositivo finale).
2. Dal menu in alto selezionare "Integrations" (**Fig. 3**).
3. Fare clic sul collegamento "Add integrations".
4. Dalla schermata contenente le integrazioni selezionate l'integrazione myDevices.
5. Nel campo "Process ID" della schermata di impostazione di myDevices (**Fig. 4**) inserite un nome qualsiasi, tutto minuscolo: per esempio "gps_cayenne".
6. Selezionare dal menu a discesa del campo "Access Key" la voce "Default key".
7. Fare clic su "Add integration".
8. Apparirà una finestra con l'integrazione myDevices in stato di "running" (**Fig. 5**).
9. Fare clic su "Payload Formats" del menu dell'applicazione.
10. Selezionare nel menu a discesa "Payload Format" la voce "Cayenne LPP" (**Fig. 6**).

LA VISUALIZZAZIONE GPS IN CAYENNE

Per prima cosa bisogna creare un account gratuito sul sito myDevices, scegliendo l'opzione Cayenne dalla home page. Dopo il login seguite questa procedura per creare un progetto.

1. Dal menu laterale, scegliere l'opzione "Add new Project" (**Fig. 7**).
2. Dare un nome qualsiasi al progetto: per esempio, "GPS".
3. Dallo stesso menu scegliere l'opzione "Add Device/Widget".
4. Dalla pagina con i widget e i dispositivi disponibili che si aprirà, scegliere l'opzione LoRa.
5. Nella pagina con tutti i servizi LoRaWAN che si aprirà, selezionare l'icona "The Things Network".
6. Si aprirà una pagina con (quasi) tutti dispositivi esistenti in commercio, dove selezionare "Dragino Technology GPS Shield", che è lo shield per Arduino con le stesse caratteristiche del nostro dispositivo.
7. Apparirà una pagina in cui compilare i campi seguenti (**Fig. 8**) e scrivere nel campo "Nome" un testo qualsiasi.
8. Nel campo "DevEUI" incollare il devEUI del dispositivo creato nell'applicazione di TTN.
9. Nel campo "Activation Mode" dovrebbe apparire "Already registered".
10. Nel campo "Location" selezionare "This device moves" oppure "This device doesn't move".
11. Fare clic su "Add device".
12. Apparirà una pagina "Overview" con il messaggio "Waiting for live data".

Non appena il dispositivo finale comunica i dati si vedrà una pagina con la collocazione del dispositivo finale (**Fig. 9**). Nella finestra "Data" del dispositivo Cayenne si possono consultare i dati ricevuti in forma testuale ed è possibile esportarli come file .CSV, facendo clic sul tasto "Download"

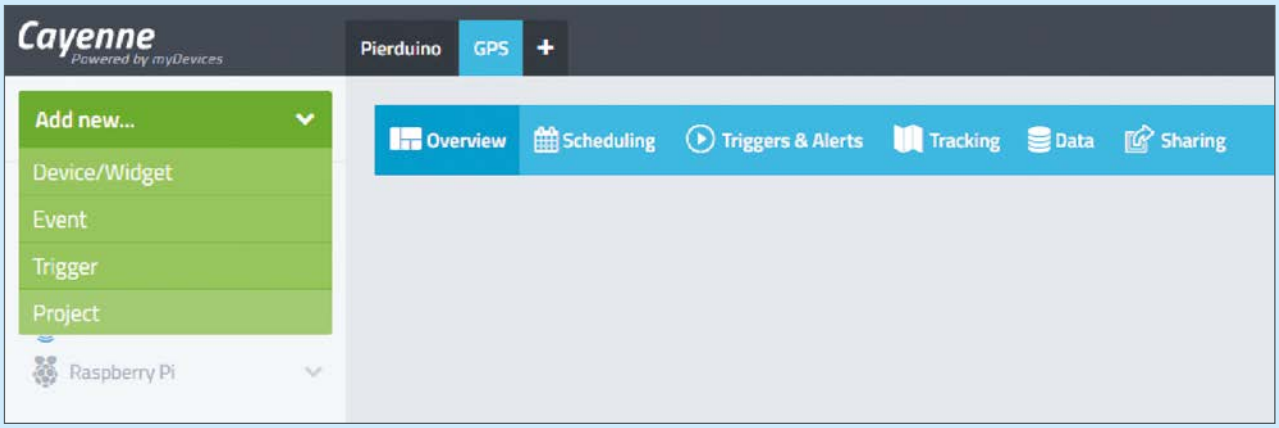

Fig. 7 - Il menù di Cayenne.

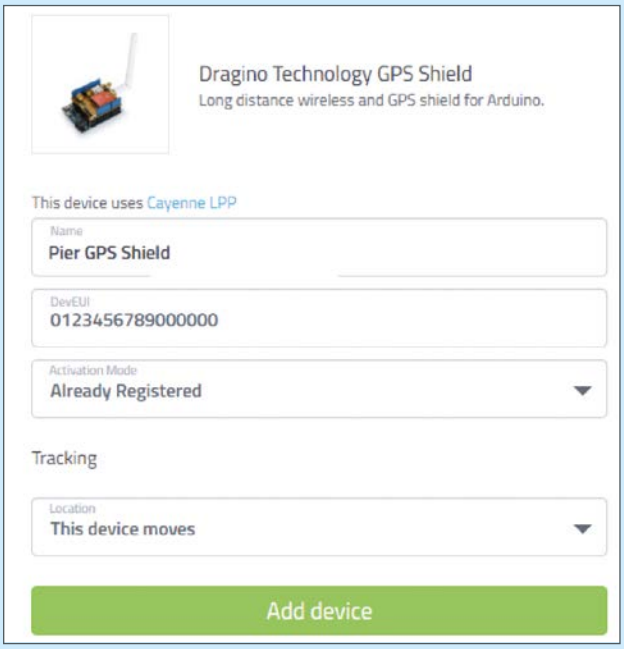

Fig. 8 - Finestra del dispositivo Cayenne.

(**Fig. 10**). Per l'esportazione dei dati sono disponibili varie opzioni, compresa un query per filtrare i dati.

CONCLUSIONE

Con questa puntata si conclude questo corso e, nella speranza che vi sia stato utile, vi auguriamo buon lavoro con i vostri dispositivi LoRa.

Cosa occorre?

Per chi volesse approfondire la tecnologia LoRa è disponibile il libro "Capire e usare LoRa e LoRaWAN" (cod. S5597) a Euro 17,90. L'EndDevice LoRa (cod. SC102) costa Euro 25,00, mentre il Gateway per Raspberry Pi (cod. GWLORA868) costa Euro 29,00. I prezzi si intendono IVA compresa.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 - http://www.futurashop.it

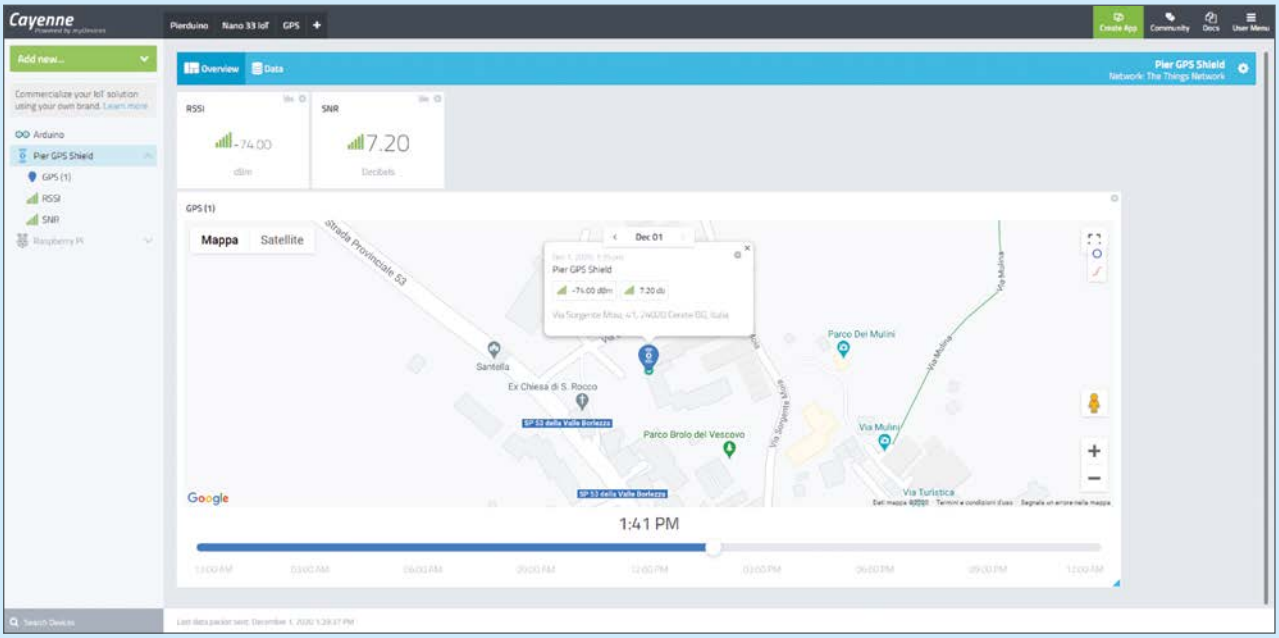

Fig. 9 - La pagina con la location del dispositivo finale.

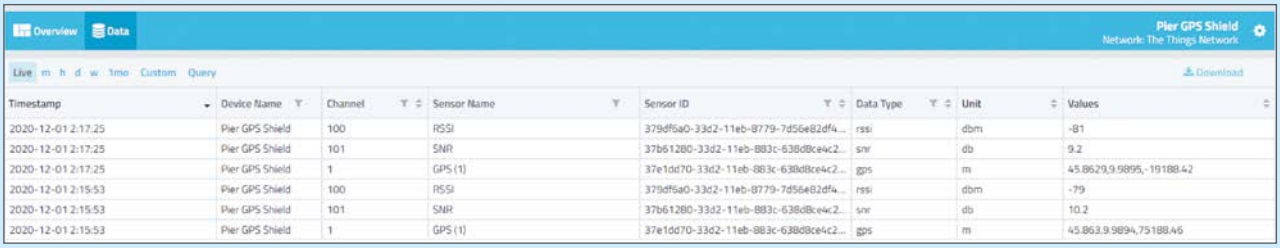

Fig. 10 - La pagina "Data" di Cayenne.