

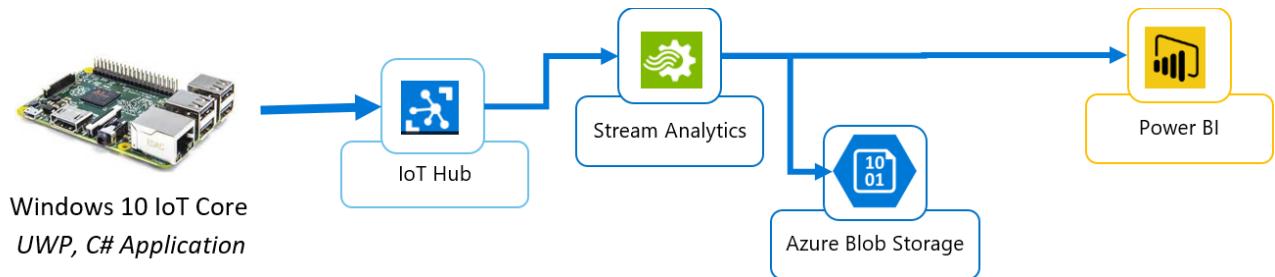
Indice

1 - Requisiti e strumenti necessari	2
2 – Windows 10 IoT Core.....	2
3 – Configurazione della RPI	4
3a - Networking	6
3b - Remote	6
4 - Connessione a Microsoft Azure.....	8
5 - L'applicazione UWP per recuperare e mostrare temperatura, luce, coordinate dell'accelerometro	13
5a - Deployment da remoto sulla RPI.....	18
6 – Il percorso dei dati: dalla RPI ad Azure IoT Hub.....	19
7 - Analisi e gestione dei dati: Azure Stream Analytics e Blob Storage	22
7a - Azure Blob Storage	22
7b - Azure Stream Analytics.....	22
7c - Visualizzazione dei dati memorizzati.....	26
8 - Visualizzazione dei dati sotto forma di grafico: Power BI	28
8a - Office365 Enterprise E3 Trial	29
8b - Registrazione a PoweBI	31
8c – Un nuovo output per Stream Analytics: PowerBI	31
8d – La Dashboard su PowerBI	34
9 - Conclusioni	36

IoT Day - Hands on Lab

In questo laboratorio realizzeremo una semplice applicazione IoT che prende dati di temperatura, luce e le coordinate dell'accelerometro utilizzando una Raspberry PI 3 e le invia ad Azure, memorizzandole in un Blob Storage e inviandole infine a Power BI per visualizzare i grafici in tempo reale.

Di seguito la struttura dell'applicazione.



1 - Requisiti e strumenti necessari

Per realizzare questo progetto è indispensabile avere un PC con installato Windows 10. Inoltre, sono necessari i seguenti hardware e strumenti di sviluppo e monitoring (in verde quelli forniti durante il laboratorio):

- [Raspberry Pi 3](#) con micro SD da almeno 8GB
- [GHI Electronics FEZHAT shield](#)
- [Azure account](#)
- [Visual Studio 2015 Community Edition Update 3](#) con installata la Windows SDK più recente
- [Windows 10 IoT Core Dashboard](#)
- Windows IoT Remote Client (Preview) – scaricabile direttamente dallo store
- [Azure Device Explorer](#) (clicca qui per il [download](#), scroll down fino al file SetupDeviceExplorer.msi)
- [Microsoft Azure Storage Explorer](#)

2 – Windows 10 IoT Core

In questo laboratorio le schede vengono fornite con Windows 10 IoT Core già installato, quindi non occorre seguire la procedura indicata in questo punto, che è stata inserita a scopo puramente informativo. Si può quindi saltare direttamente al [punto 3](#).

La Raspberry PI3 supporta Windows 10 IoT Core, scaricabile gratuitamente seguendo la procedura step by step a questo link: <https://developer.microsoft.com/it-it/windows/iot/getstarted> selezionando nell'ordine: *Raspberry PI 3, Install into my blank micro SD card, Windows 10 IoT Core Insider Preview*, come mostrato nella figura sottostante.

Get Started

The setup and installation steps are different based on what hardware you have.

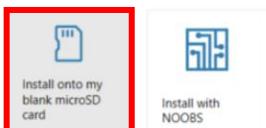
Choose your hardware, installation media, and OS version.

1 Select your hardware

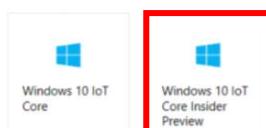
[Help me choose a device](#)



2 Select your installation media



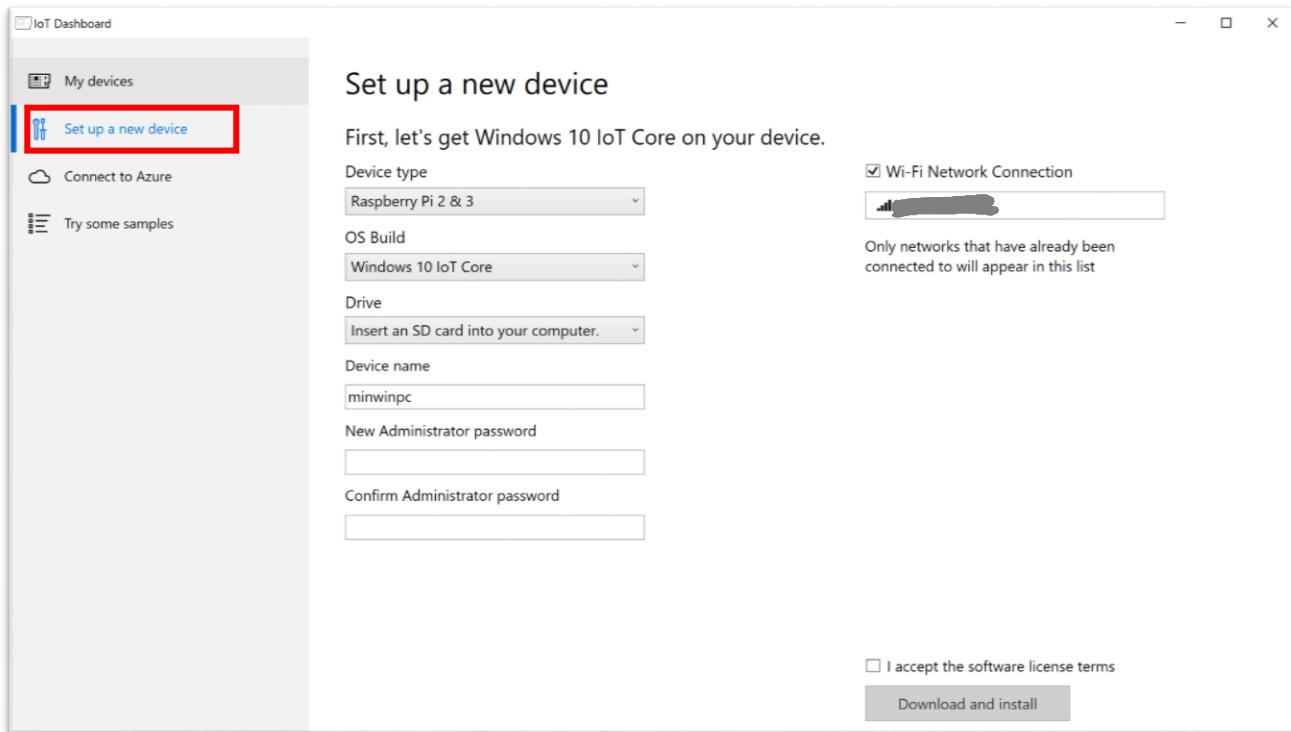
3 Select your OS version



A seguire, una procedura a 4 step ben dettagliati fornisce tutte le informazioni e i tool necessari per procedere all'installazione del sistema operativo sulla scheda e al monitoring del dispositivo (Windows 10 IoT Core Dashboard).

Per installare Windows 10 IoT Core sulla RPI, occorre aprire Windows 10 IoT Core Dashboard, nella sezione *Set Up a new device*. Impostare i seguenti parametri:

- **Device Type:** Raspberry Pi 2 o 3
- **OS Build:** Windows 10 IoT Core
- **Drive:** deve corrispondere alla Micro SD che andrà inserita nella RPI
- **Device Name:** scegliere un nome per la propria scheda (sarà il nome con cui la scheda comparirà nel Windows 10 IoT Core Dashboard. Nel mio caso Batman 😊)
- **Password:** scegliere una password che dovrà essere usata ogni volta che si accede alla web page della scheda.

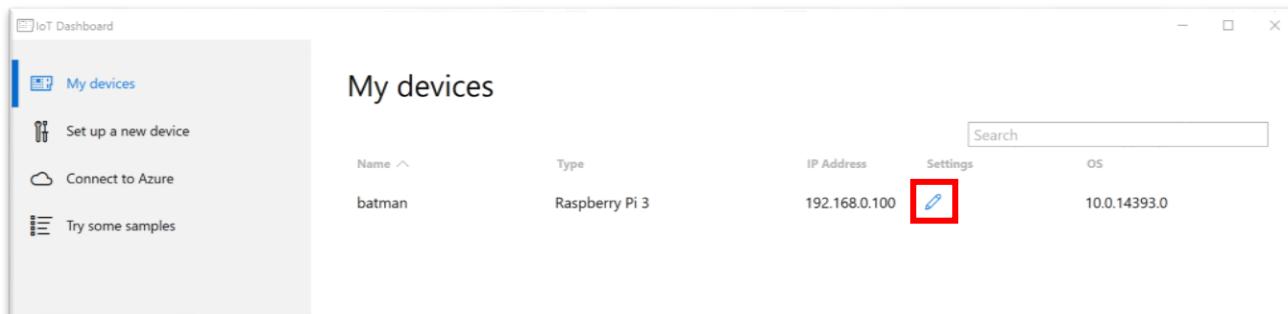


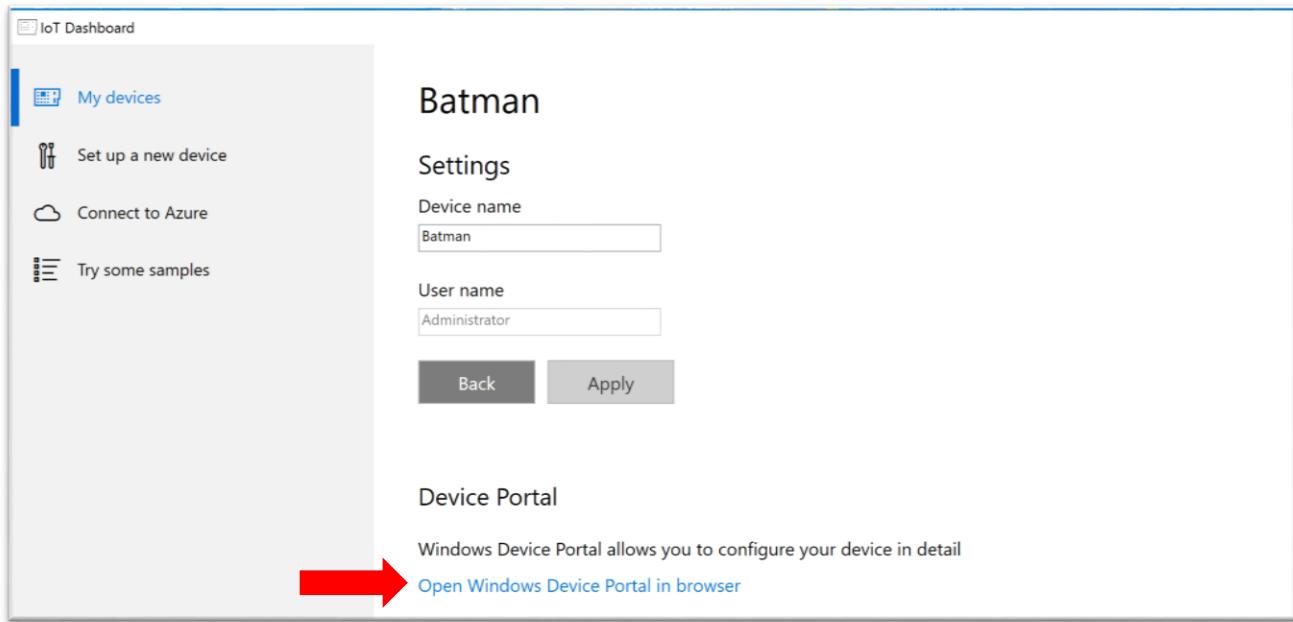
A questo punto verrà installato Windows 10 IoT Core sulla micro SD, che andrà poi inserita nella RPI. Ora è tutto pronto per accendere la scheda e iniziare a programmare!

3 – Configurazione della RPI

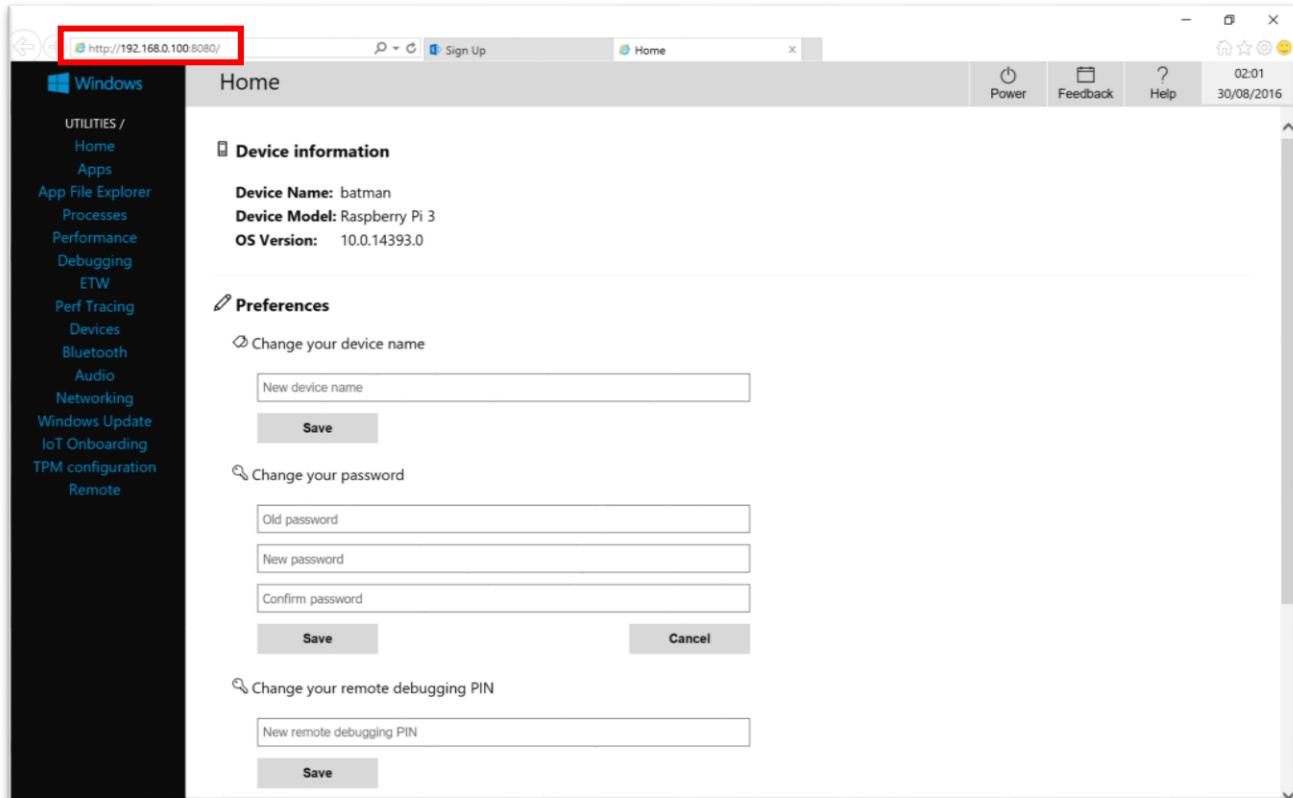
Requisito fondamentale per permettere la comunicazione tra il pc e la RPI è che siano connessi entrambi alla stessa sottorete (tramite cavo ethernet o wifi).

Per la prima connessione tra PC e scheda si consiglia di utilizzare il cavo ethernet e di connettere entrambi i dispositivi allo stesso router. Nel momento in cui la scheda compare nell'elenco *My Devices* del Windows 10 IoT Core Dashboard, come mostrato nella figura sottostante, è possibile accedere alla web page che contiene tutte le informazioni sulla RPI. Per farlo, cliccare sulla *matita* e poi sul link *Open Windows Device Portal in browser*.





A questo punto si aprirà una web page che contiene tutti i dati e le configurazioni della vostra RPI. Per accedervi, inseriamo username (default:*Administrator*, come potete vedere dall'immagine qui sopra) e la password che avete selezionato al punto precedente. Si aprirà una pagina web simile a quella sottostante.



Vi faccio notare che è possibile accedere alla RPI anche direttamente dal browser, inserendo l'indirizzo IP della scheda e la porta 8080.

Per procedere con questo hands on lab è necessario che siano configurati correttamente il WiFi della scheda, il TPM (che vedremo al punto 4) e che sia abilitato il remote desktop. Vediamo come fare.

3a - Networking

Per questo laboratorio, le schede sono già state connesse al wifi della sala, è indispensabile connettere il pc allo stesso. Verificare se la scheda è visibile tra i MyDevices di Windows 10 IoT Core Dashboard. In caso positivo, si può passare direttamente al [punto 3b](#).

Nella webpage della RPI, alla sezione *Networking*, selezioniamo la Wifi della scheda (se c'è un dongle, funzionano correttamente entrambi i profili – wifi integrata o dongle) e identifichiamo dall'elenco sottostante la Wifi a cui ci vogliamo collegarci. Vi ricordo che è fondamentale che RPI e PC siano connessi alla stessa sottorete.

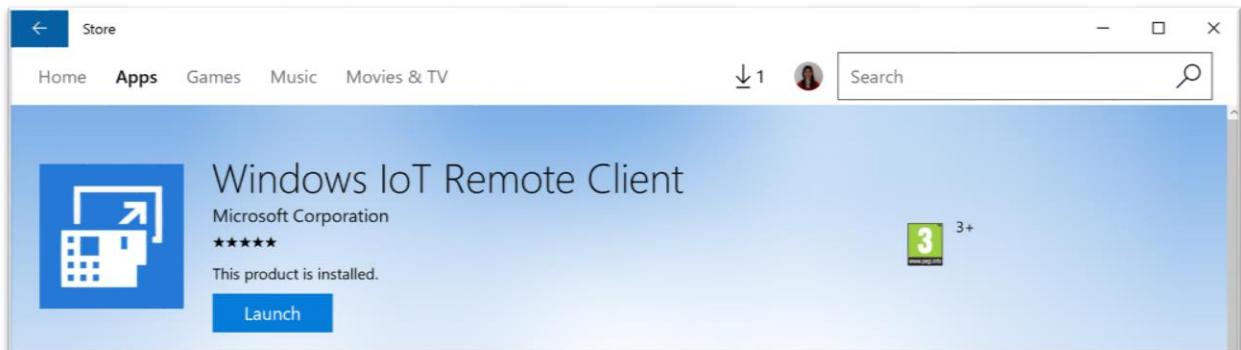
Una volta selezionata la rete WiFi, verrà chiesto di inserire la password. Da questo momento in poi, ad ogni avvio la scheda si conterà automaticamente alla rete WiFi selezionata.

The screenshot shows the Windows 10 IoT Core Dashboard interface. On the left, a sidebar lists various utilities like Home, Apps, File Explorer, Processes, Performance, Debugging, ETW, Perf Tracing, Devices, Bluetooth, Audio, and Networking. The Networking option is highlighted with a red arrow. The main content area is titled 'Networking' and shows 'WiFi adapters'. A dropdown menu is open, showing 'Broadcom 802.11n Wireless SDIO Adapter' with a red arrow pointing to it. Below this is a 'Profiles' section with a blue bar and 'Connect' and 'Delete' buttons. A note says: '* Changing your network profile will change your IP address and require you to reconnect to Device Portal'. A large red box highlights the 'Available networks' section, which lists 'MSFTCORP' and 'MSFTGUEST' with details like SSID, PROFILE, INFRA, SIGNAL, SECURITY, ENCRYPTION, and CHANNEL. At the bottom, there's an 'IP configuration' section with fields for Description (Bluetooth Device (Personal Area Network)), Type (Ethernet), Physical address (b8-27-eb-2e-9a-63), IPv4 address (0.0.0.0), and Subnet mask (0.0.0.0). A 'Refresh' button is also present.

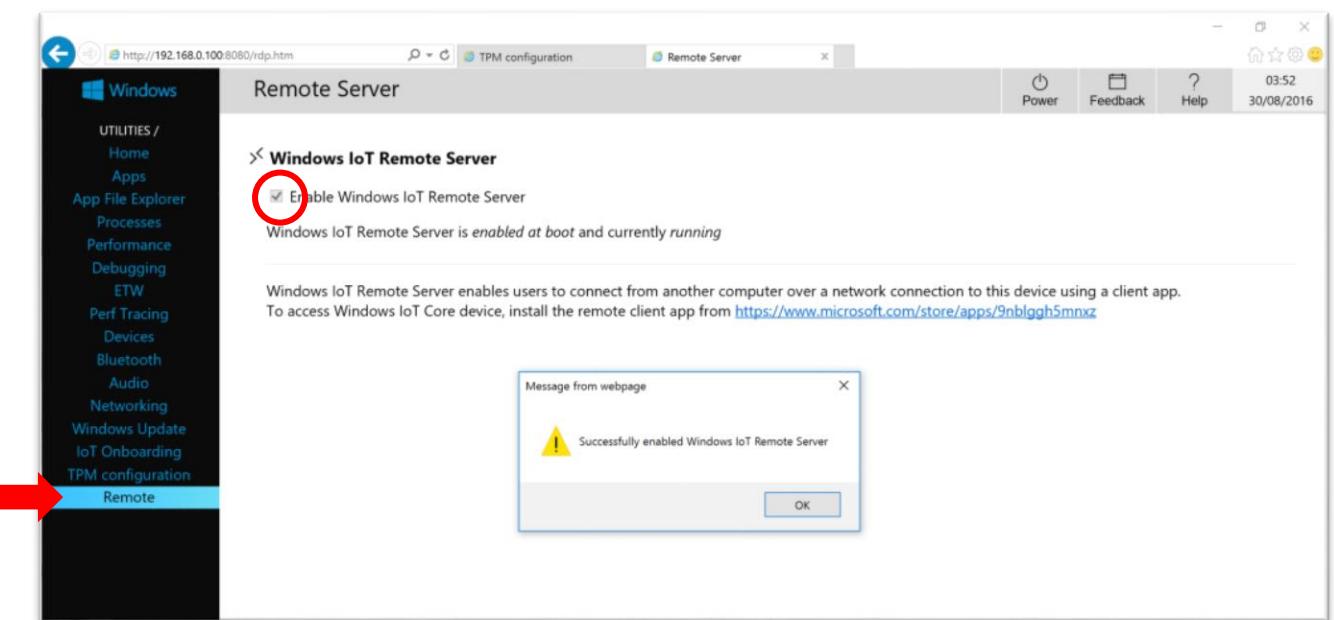
3b - Remote

Per abilitare la possibilità di visualizzare da remoto l'interfaccia grafica dell'applicazione che sta girando sulla RPI sono necessarie pochi semplici passi:

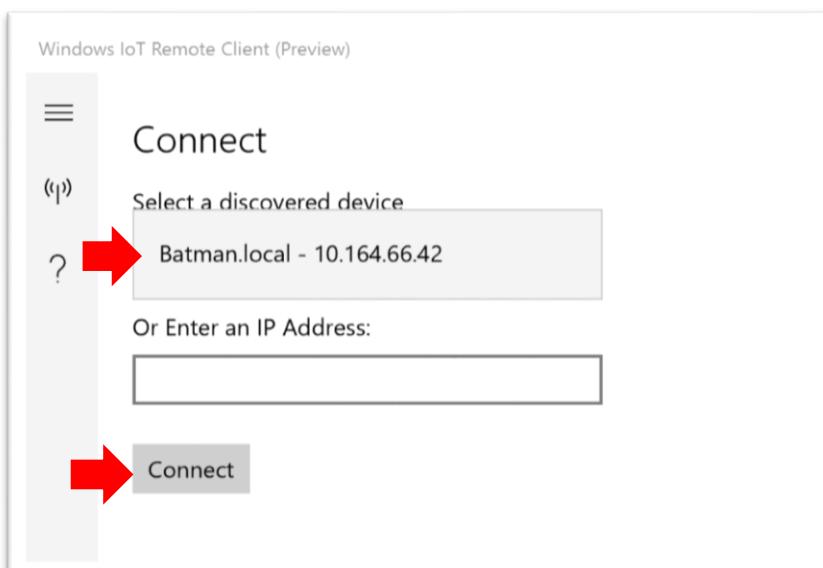
- 1 – scaricare l'app dal Windows Store: Windows IoT Remote Client (Preview)



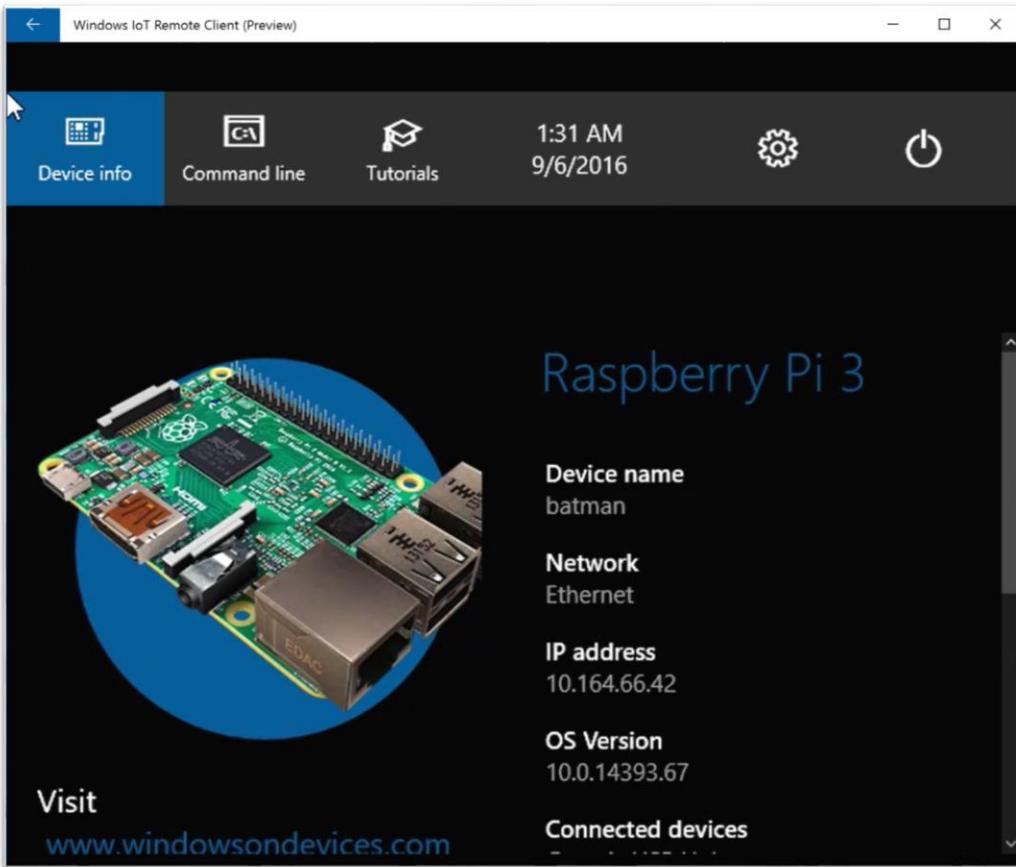
2 – Nella webpage della RPI alla sezione *Remote* abilitare il flag *Enable Windows IoT Remote Server*, come mostrato in figura



3 – aprire l'app Windows IoT Remote (Preview) e selezionare o inserire il nome o l'indirizzo IP della nostra RPI, come mostrato in figura.



Una volta connesso, l'applicazione mostrerà l'interfaccia grafica dell'app che sta girando sulla RPI. Di default, ora che non abbiamo ancora caricato nulla sulla scheda, questo è quello che vedremo.



4 - Connessione a Microsoft Azure

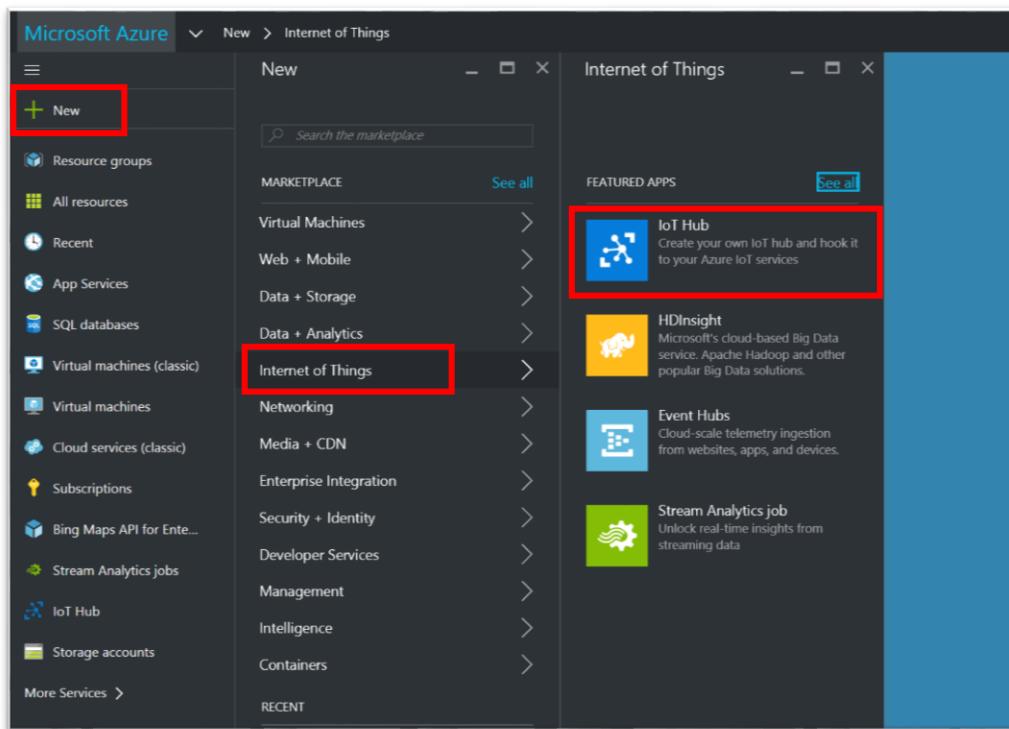
Per prima cosa occorre creare un IoT Hub sulla nostra sottoscrizione Azure. Questo sarà il collettore di tutti i dati che invieremo dalla nostra RPI e il punto di accesso al cloud.

Apriamo il browser (in private mode) e accediamo al portale di Azure: <http://portal.azure.com>.

A screenshot of the Microsoft Azure portal dashboard. The left sidebar shows navigation options like Resource groups, All resources, Recent, App Services, SQL databases, Virtual machines (classic), Cloud services (classic), Subscriptions, and more. The main dashboard area has several sections:

- All resources:** Shows a list of resources including "FreeServicePlan", "FreeServicePlan", and "RPIdemo".
- Service health:** A map showing the status of various Azure regions.
- Marketplace:** A link to the Azure Marketplace.
- Subscriptions:** A section to forecast expenses and costs to optimize your apps.
- Help + support:** A link to help and support.
- Tour:** A link to the tour.
- What's new:** A link to what's new.
- Portal settings:** A link to portal settings.
- Feedback:** A link to feedback.
- RPIdemo AZURE IOT HUB:** A card indicating the IoT hub is active.
- Anure classic portal:** A link to the Anure classic portal.

Per creare un nuovo IoT Hub selezioniamo *New > Internet of Things > IoT Hub*



Si aprirà a questo punto una tab in cui dovremo inserire i dati come riportati nella figura seguente

The screenshot shows the 'Hub IoT' creation wizard. On the left, the configuration fields are shown:

- * Nome: iothub4lab (highlighted with a red arrow)
- * Piano tariffario e livello di scalabilità: S1 - Standard (highlighted with a red arrow)
- * Unità di IoT Hub: 1
- * Partizioni da dispositivo a cloud: 4 partizioni
- * Sottoscrizione: Visual Studio Ultimate con MSDN
- * Gruppo di risorse: LabIoT (highlighted with a red arrow)
- * Località: Europa settentrionale (highlighted with a red arrow)
- Abilita Gestione dei dispositivi (ANTEPRIMA) (checkbox)

On the right, the 'Scegliere il piano tariffario e il livello di scalabilità' (Choose tariff plan and scalability level) panel displays three plans:

S1 Standard	S2 Standard	S3 Standard
400k messaggi/unità/giorno	6M messaggi/unità/giorno	300M messaggi/unità/giorno
Da dispositivo a cloud telemetria	Da dispositivo a cloud telemetria	Da dispositivo a cloud telemetria
Da cloud a dispositivo messaggistica	Da cloud a dispositivo messaggistica	Da cloud a dispositivo messaggistica
200 units maximum	200 units maximum	10 units maximum
50,00 USD PER UNITÀ DI IOT HUB		

A fourth plan, F1 Free, is shown in a separate box:

F1 Free
8k messaggi/unità/giorno
Da dispositivo a cloud telemetria
Da cloud a dispositivo messaggistica
1 unit
È consentito un solo hub IoT gratuito per sottoscrizione.
0,00 USD PER UNITÀ DI IOT HUB

Verifichiamo che il nome scelto sia unico (in caso contrario ci comparirà un punto esclamativo rosso di fianco al nome e dovremo necessariamente cambiarlo).



Ricordiamoci questo nome perchè ci servirà poi per connettere la RPI ad Azure.

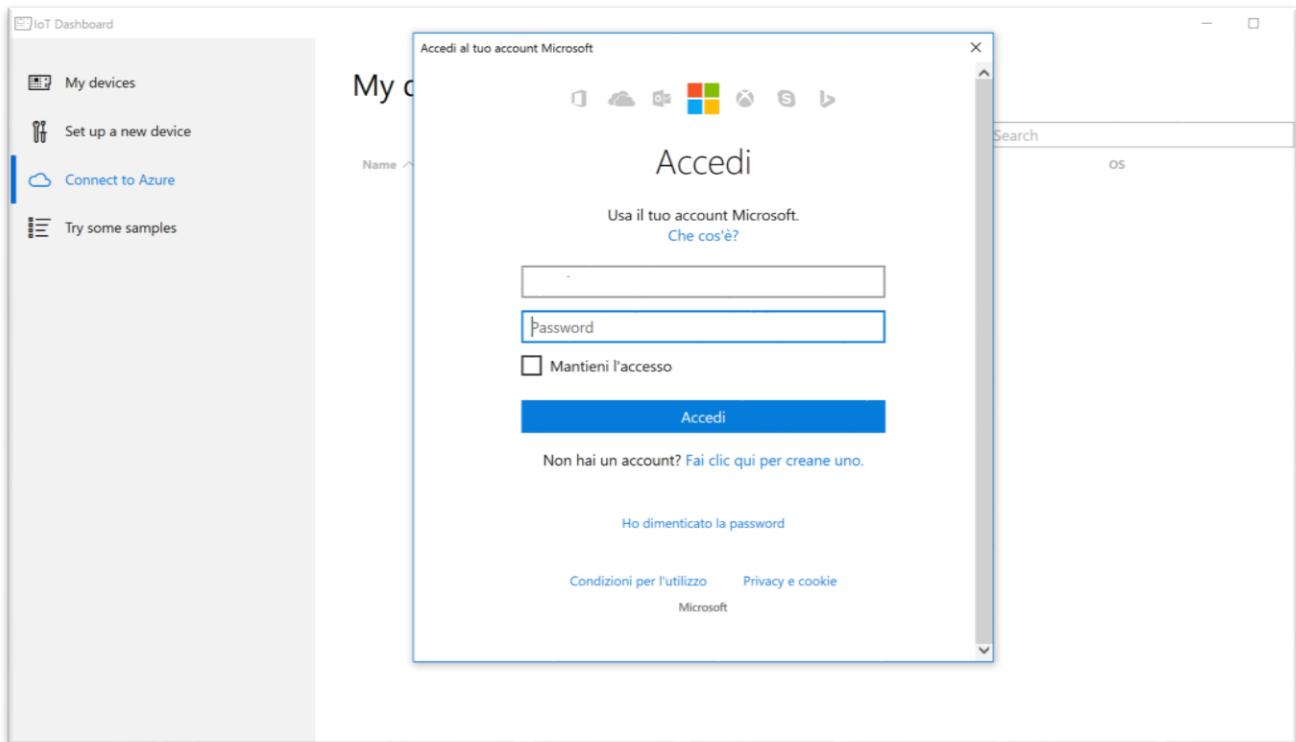
Per quanto riguarda il piano tariffario, se nella vostra sottoscrizione non sono presenti altri IoT Hub allora potete anche selezionare il *piano gratuito F1 Free* (nell'immagine sopra è disabilitato in quanto è consentito avere un solo IoT Hub gratuito per ciascuna sottoscrizione). Ai fini di questo laboratorio è sufficiente il piano gratuito, anche se saranno disabilitati alcuni parametri di configurazione.

Per quanto riguarda il *Gruppo di Risorse* o *Resource group*, facciamo attenzione a creare uno nuovo e chiamiamolo *LabIoT*. Per tutti i servizi che creeremo successivamente selezioneremo questo stesso resource group, in modo da avere tutto ciò che serve in un unico gruppo.

Infine, posizioniamolo in Europa Settentrionale.

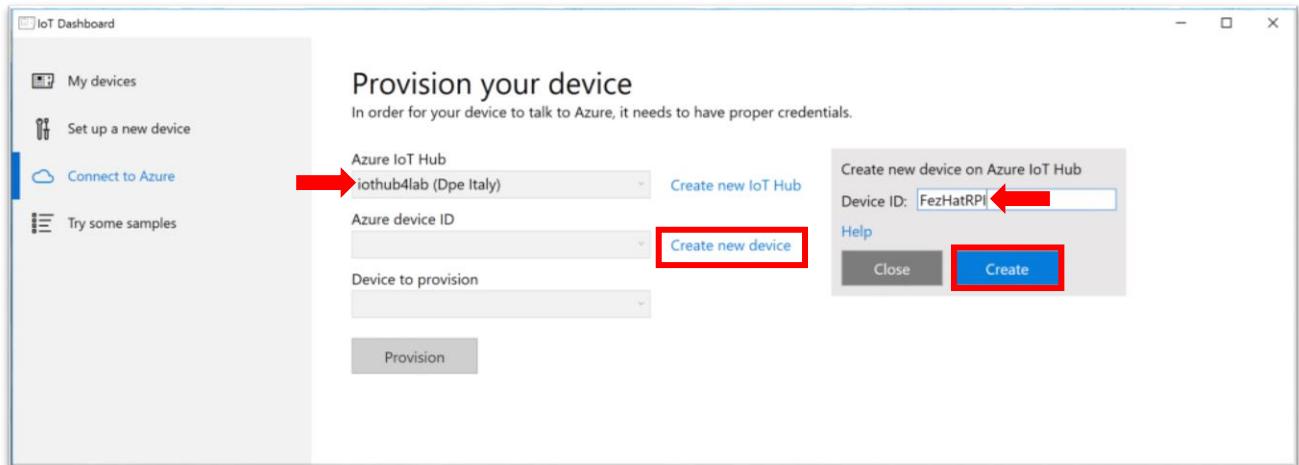
A questo punto possiamo tornare alla scheda.

Apriamo la Windows IoT Core Dashboard, alla sezione Connect to Azure e inseriamo le credenziali del nostro account (quello in cui abbiamo appena creato l'IoT Hub).

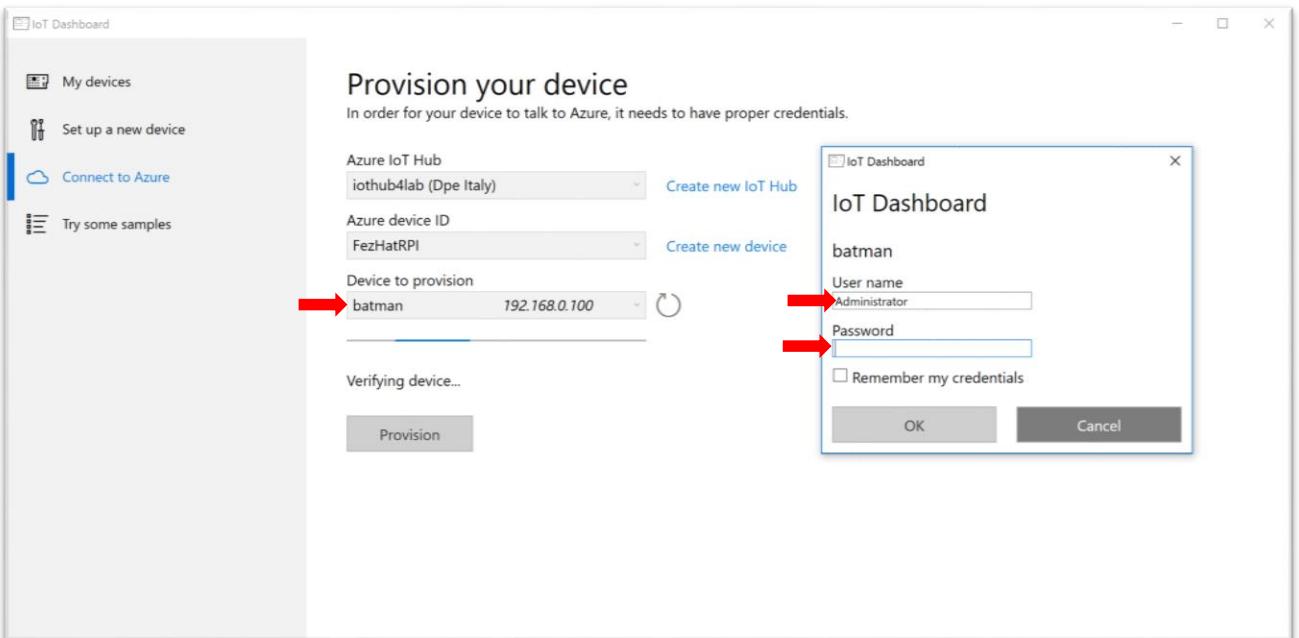


Una volta eseguito l'accesso, ci ritroveremo una schermata simile a quella che segue, in cui dovremo selezionare l'IoT Hub appena creato (cliccando *Create new IoT Hub* avremmo potuto anche crearlo da qui, ma lo abbiamo già fatto nella maniera tradizionale dal portale).

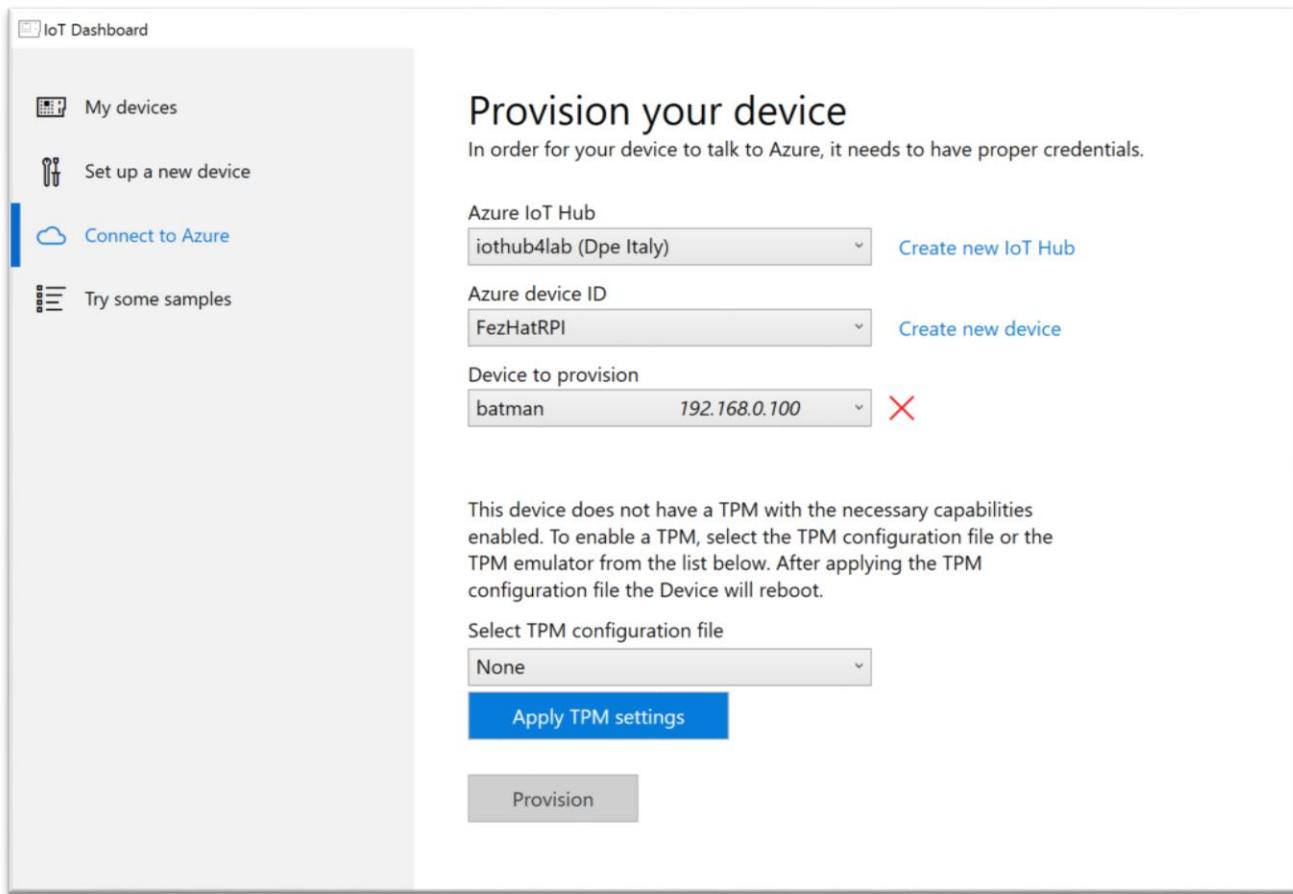
Dobbiamo invece creare un nuovo *device ID*, quindi clicchiamo su *Create New Device* ed inseriamo come id *FezHatRPI*.



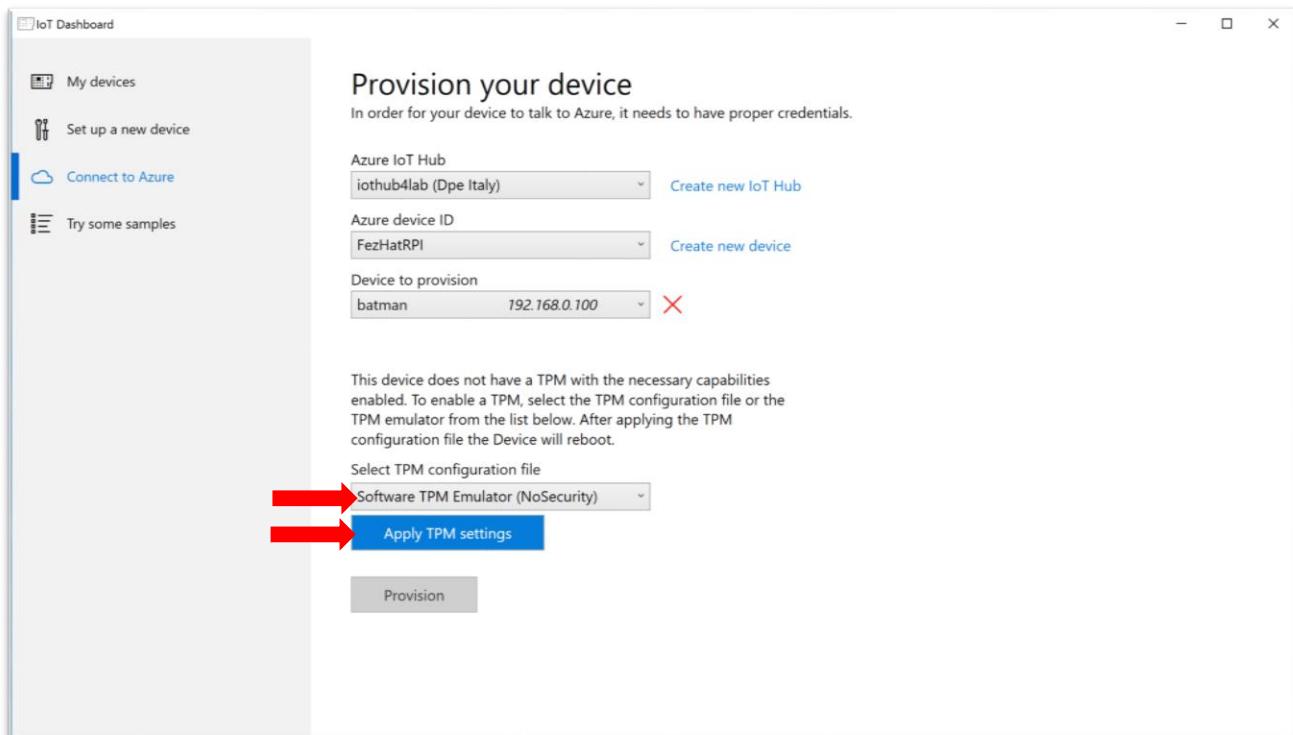
Nella sezione *Device to provision*, selezioniamo la nostra RPI. Ci verranno chieste le credenziali di accesso, le stesse usate per accedere alla webpage della RPI, che sono state impostate al [punto 2 \(Windows 10 IoT Core\)](#).



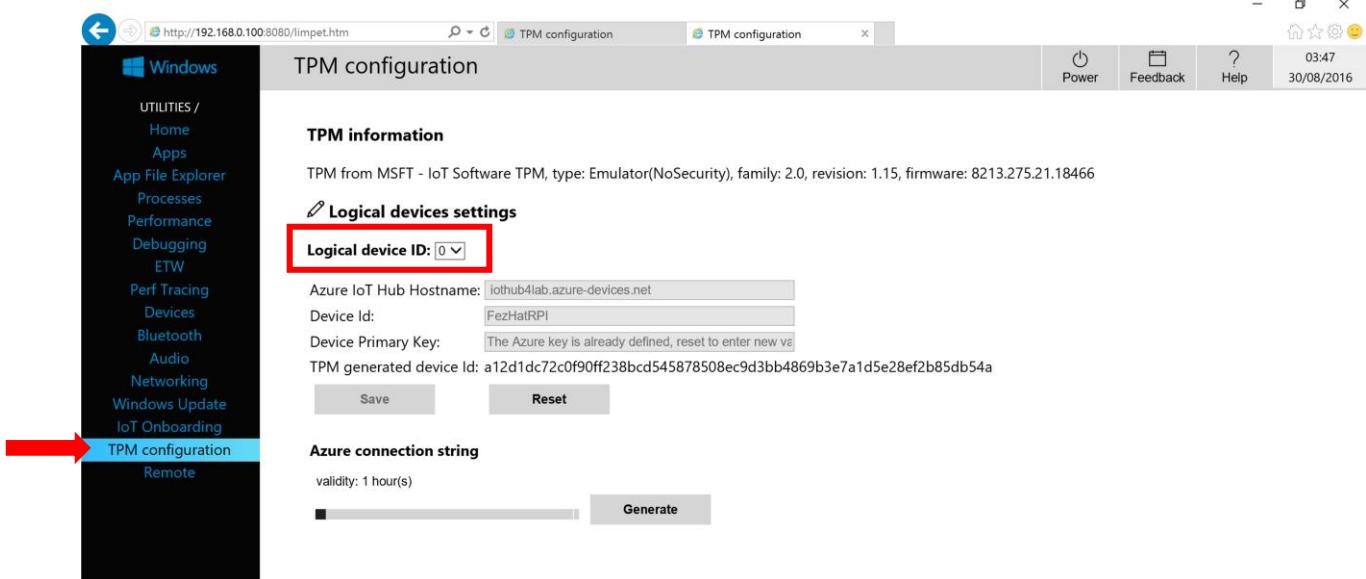
A questo punto ci ritroveremo una schermata come quella che segue e dovremo procedere al provisioning del device.



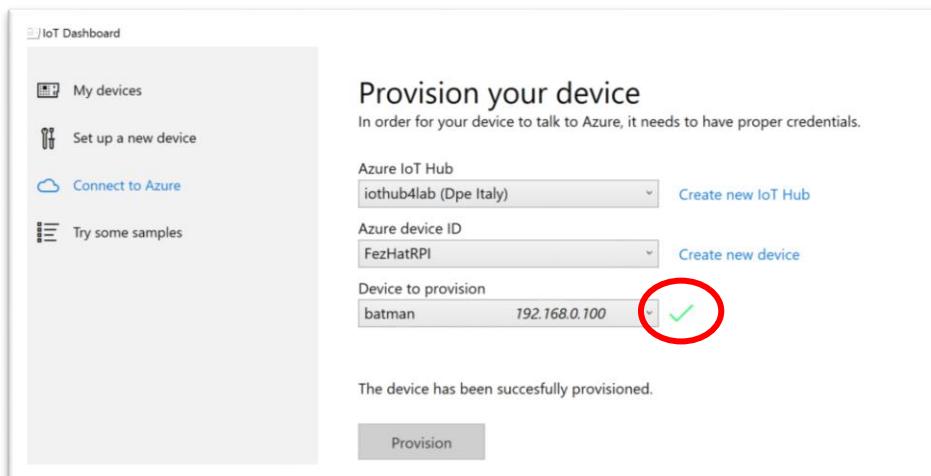
Nel caso della RPI, il TPM configuration file da selezionare è quello software, in quanto la scheda non contiene un modulo hardware dedicato. Per ulteriori informazioni sul TPM potete consultare la documentazione a questo link: <https://developer.microsoft.com/en-us/windows/iot/docs/tpm>



La RPI verrà riavviata e si torna alla schermata principale del Windows 10 IoT Dashboard. Una volta terminato il reboot, verifichiamo che le impostazioni siano corrette anche sulla webpage della scheda, alla sezione TPM Configuration:



Vediamo che i dati corrispondono a quelli che abbiamo inserito nell'IoT Dashboard. Se così non fosse, riapriamo la Dashboard nella sezione Connect to Azure e verifichiamo che il Provisioning sia stato completato (dobbiamo vedere una spunta verde di fianco al nome della nostra RPI).

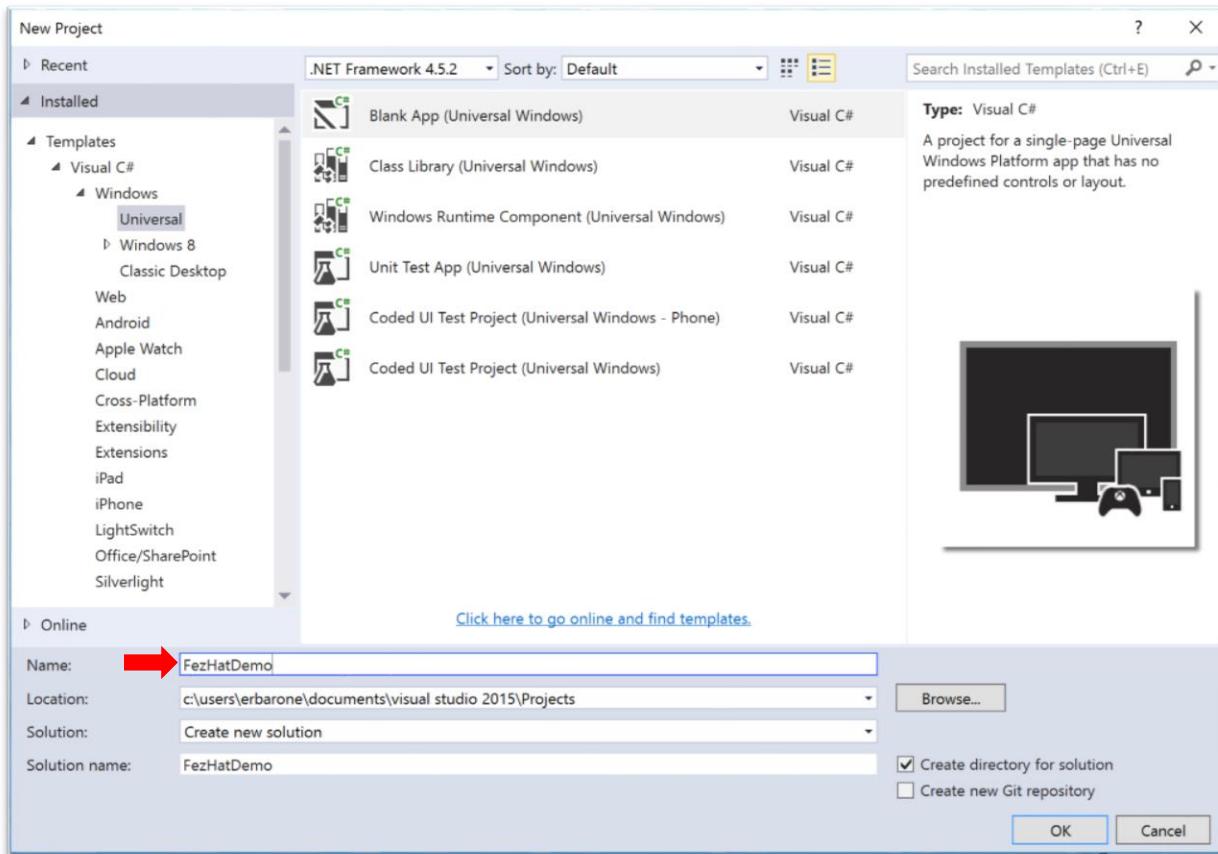


Ora il nostro dispositivo è stato registrato sul nostro IoT Hub e quindi è autorizzato ad inviar gli dati. Vediamo nel prossimo capitolo come recuperare i dati dai sensori presenti sullo shield FezHat e inviarli al cloud, tramite una semplice UWP Application.

5 - L'applicazione UWP per recuperare e mostrare temperatura, luce, coordinate dell'accelerometro

Sulla nostra RPI abbiamo installato Windows 10 IoT Core, ora dobbiamo creare una UWP Application per recuperare i dati e inviarli ad Azure.

Apriamo Visual Studio 2015, assicurandoci di avere installato l'Update 3. Dobbiamo creare un nuovo progetto UWP, quindi selezioniamo *File > New > Project > Installed > Templates > Universal > Blank App* come mostrato in figura. Assegnamo il nome *FezHatDemo*.



L'applicazione è molto semplice: si tratta di prendere i dati di temperatura, luce e le coordinate dell'accelerometro, visualizzarle nella UI ed inviarle all'IoT Hub. Per questo motivo è sufficiente lavorare sulla *MainPage*.

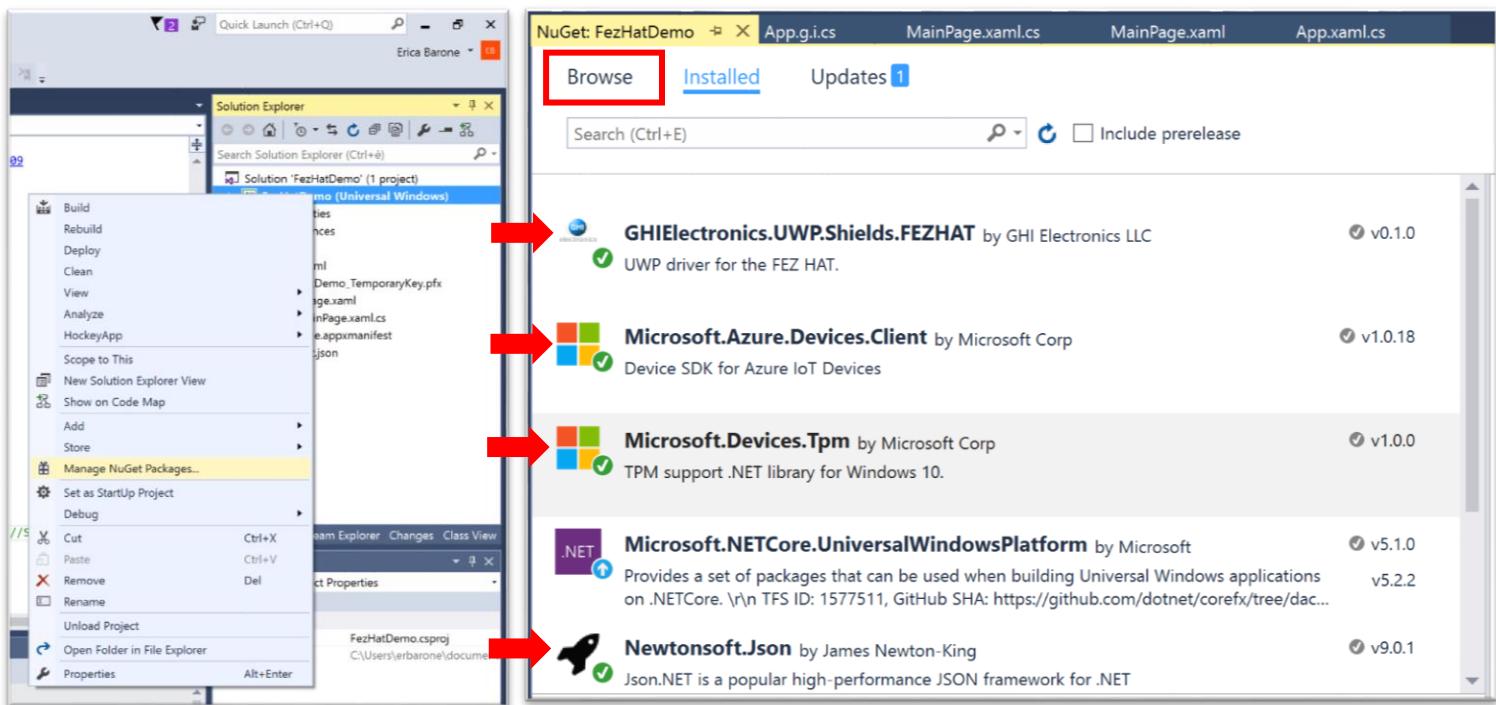
Cominciamo con l'installazione dei package NuGet. *Tasto destro sul progetto > Manage NuGet Packages*. Scarichiamo e installiamo i seguenti package, digitandoli nella sezione *Browse*:

Microsoft.Azure.Device.Client -> Per gestire la comunicazione tra l'app e Azure IoT Hub

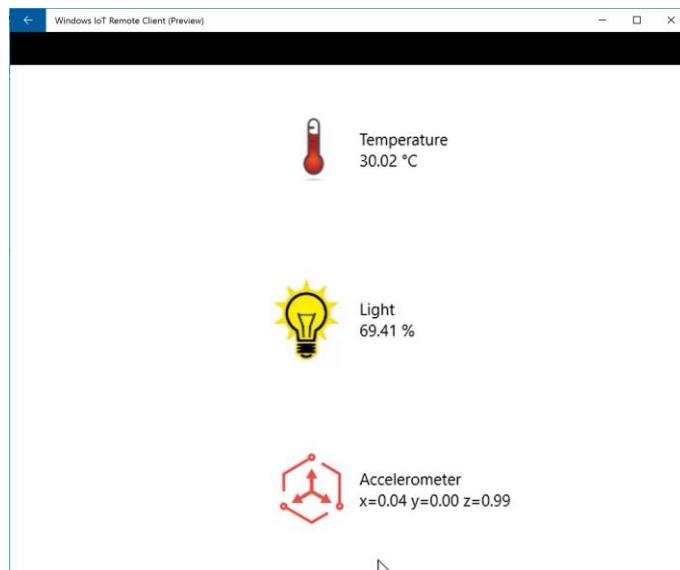
Microsoft.Device.Tpm -> Per utilizzare il TPM configurato al punto precedente

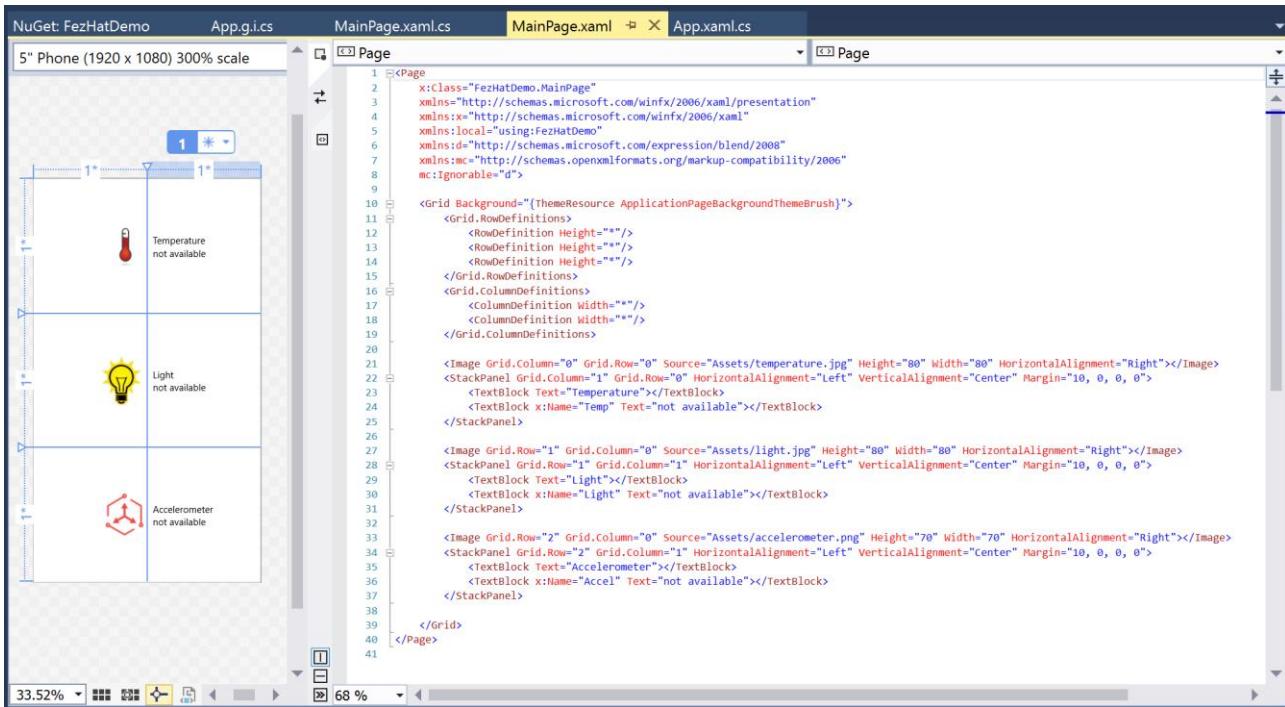
GHIElectronics.UWP.Shafts.FEZHAT -> Per recuperare i dati dai sensori presenti sullo shield della scheda

Newtonsoft.Json -> Per la serializzazione del messaggio da inviare all'IoT Hub



Il risultato finale che dobbiamo ottenere è questo, per capire come fare apriamo la *MainPage.xaml*.





In pratica è stato utilizzato il controllo *Grid* per inserire le immagini e le *Textblock* in maniera opportuna. Il testo di default per tutti i valori è “*not available*”, ma viene aggiornato ogni secondo tramite il codice C# contenuto in *MainPage.xaml.cs*.

Il code behind dell'applicazione è costituito da 3 semplici parti:

- Un costruttore, che provvede all'inizializzazione e al setup del timer
- Il tick del timer, dentro il quale è contenuta la logica di recupero dei dati dai sensori e invio al cloud, oltre all'update della UI
- L'inizializzazione dello shield FezHat

Cominciamo dal costruttore.

```
1 reference
public MainPage()
{
    this.InitializeComponent();

    //init device and shield
    TpmDevice mytpmdevice = new TpmDevice(0);
    _deviceClient = DeviceClient.CreateFromConnectionString(mytpmdevice.GetConnectionString()); //SAS Token: 60 min
    initFezhat();

    //timer setup and start -> 1 msg every second
    _timer = new DispatcherTimer();
    _timer.Interval = TimeSpan.FromSeconds(1);
    _timer.Tick += _timer_Tick;
    _timer.Start();

}
```

E' interessante notare come, grazie al TPM configurato sulla RPI, la connessione con Azure IoT Hub possa avvenire senza la necessità di inserire la *connection string* all'interno del codice sorgente. Questo garantisce ovviamente una maggiore sicurezza.

Focalizziamo ora l'attenzione sul tick del timer, che contiene praticamente l'intera logica dell'applicazione.

```

1 reference
private async void _timer_Tick(object sender, object e)
{
    if (_hat == null)
        return;

    //update UWP app UI
    double temp, light, x, y, z;
    temp = _hat.GetTemperature();
    light = _hat.GetLightLevel();
    _hat.GetAcceleration(out x, out y, out z);

    Temp.Text = $"{temp:N2} °C";
    Light.Text = $"{light:P2}";
    Accel.Text = $"x={x:N2} y={y:N2} z={z:N2}";

    //create message
    var message = new
    {
        deviceID = "Batman",
        temperatureToSend = temp,
        lightToSend = light,
        xToSend = x,
        yToSend = y,
        zToSend = z
    };

    var messageString = JsonConvert.SerializeObject(message);
    var messageToSend = new Message(Encoding.UTF8.GetBytes(messageString));

    //send message to IoTHub
    await _deviceClient.SendEventAsync(messageToSend);
}

}

```

Come potete vedere, recuperare i dati dai sensori utilizzando i metodi messi a disposizione dal package *GHIElectronics.UWP.Shields.FEZHAT* è davvero molto semplice

```

//update UWP app UI
double temp, light, x, y, z;
temp = _hat.GetTemperature();
light = _hat.GetLightLevel();
_hat.GetAcceleration(out x, out y, out z);

```

Questi dati vengono poi utilizzati per aggiornare la UI dell'applicazione e poi vengono impacchettati all'interno di un messaggio, serializzati e infine inviati al nostro IoT Hub tramite un semplicissimo metodo, fornito dal package *Microsoft.Azure.Device.Client*

```

//create message
var message = new
{
    deviceID = "Batman",
    temperatureToSend = temp,
    lightToSend = light,
    xToSend = x,
    yToSend = y,
    zToSend = z
};

var messageString = JsonConvert.SerializeObject(message);
var messageToSend = new Message(Encoding.UTF8.GetBytes(messageString));

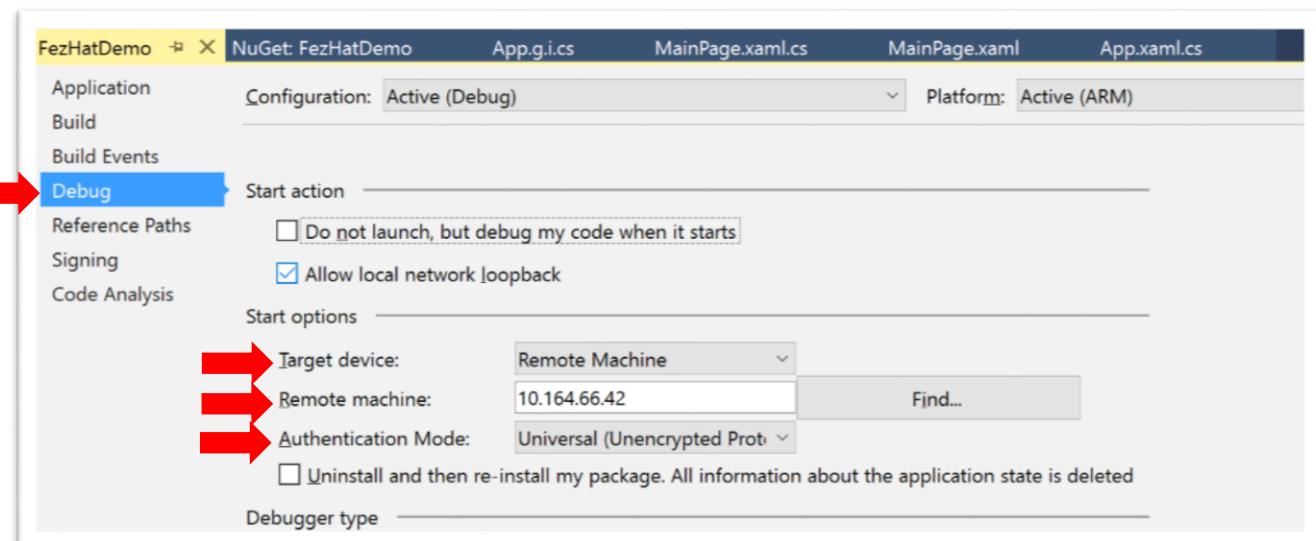
//send message to IoTHub
await _deviceClient.SendEventAsync(messageToSend);

```

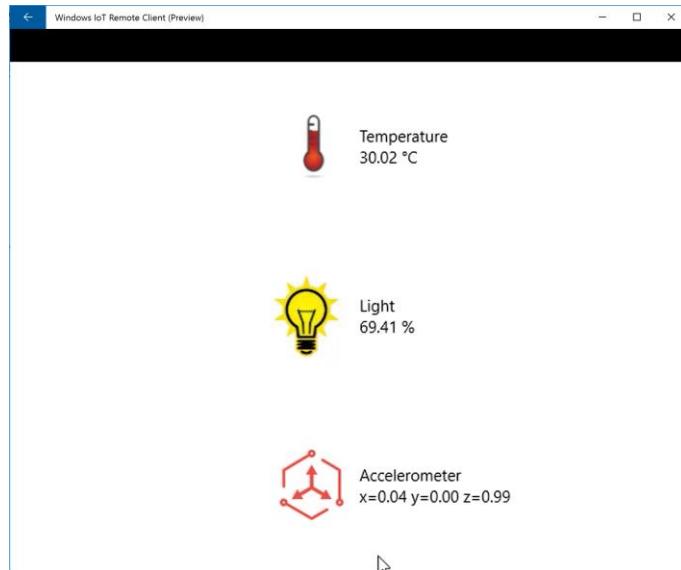
L'applicazione è disponibile su GitHub al seguente link: <https://github.com/erryB/LabIoT>

5a - Deployment da remoto sulla RPI

Prima di far partire l'applicazione però, dobbiamo configurare il deployment remoto sulla nostra RPI. Per farlo, clicchiamo con il *tasto destro sul progetto > Properties*. Nella sezione *Debug* selezioniamo *Remote Machine* e inseriamo il nome o l'indirizzo IP della nostra RPI, verificando che l'*Authentication mode* sia *Universal*, come mostrato in figura.



Ora possiamo finalmente lanciare la nostra applicazione e verificare tramite l'app *Windows IoT Remote Client (Preview)* che il risultato sia quello sottostante, con i numeri che si aggiornano ogni secondo.



Possiamo provare a muovere la RPI, ad abbassare la luce o alzare la temperatura e verificare l'aggiornamento dei dati in locale. Ma come facciamo a verificare che questi dati vengano effettivamente mandati al nostro IoT Hub? Lo vediamo nel prossimo punto.

6 – Il percorso dei dati: dalla RPI ad Azure IoT Hub

Per verificare che i dati raccolti dai sensori arrivino effettivamente al nostro IoT Hub, apriamo la dashboard di Azure: <http://portal.azure.com> e selezioniamo il resource group creato al [punto 4](#), poi il nostro IoT Hub.

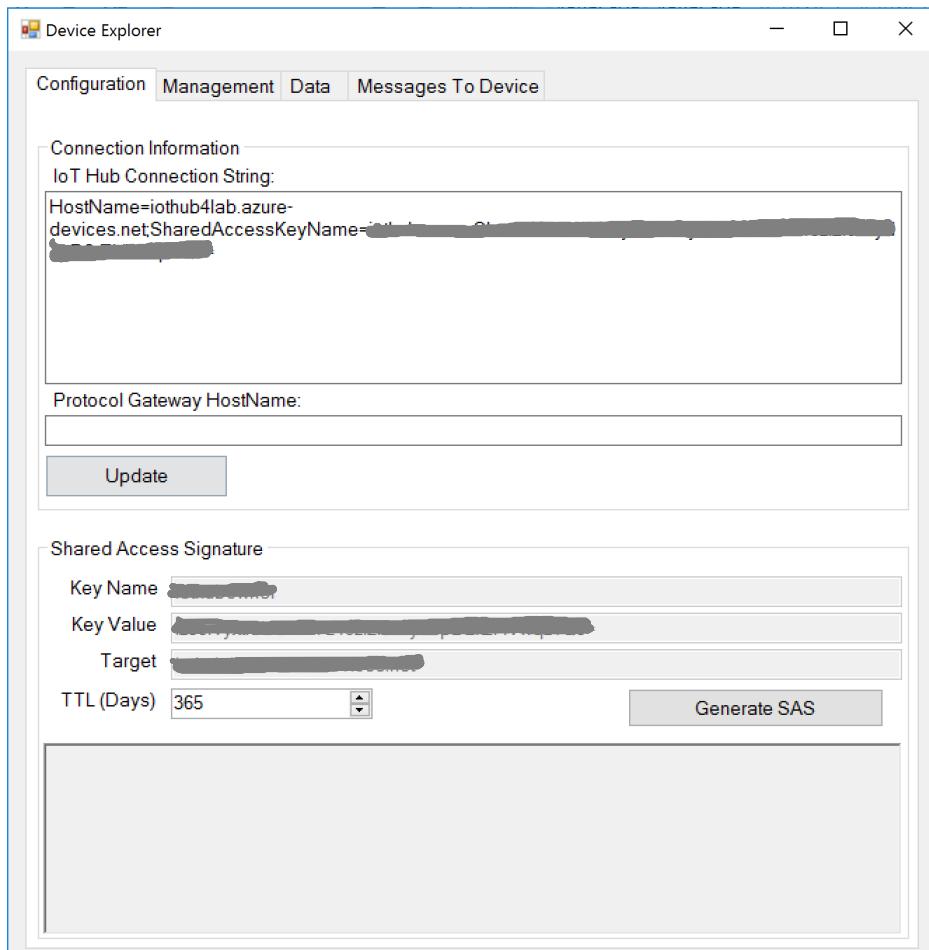
Una volta selezionato l'IoT Hub, ci troveremo di fronte una schermata simile a questa.

Informazioni di base	Utilizzo
ID sottoscrizione: 0fe8fa123-74ca-43ce-b688-63d0acfa1d5e	07/09/2016 UTC IOTHUB4LAB 119 /400K DISPOSITIVI 1
Nome sottoscrizione: Dipe Italy	
Ultima distribuzione: 31/08/2016 (Operazione riuscita)	
Località: Europa occidentale	
	Monitoraggio IOTHUB4LAB 10 40 50 40 30 20 10 0 CONNECTED DEVICES 0 TELEMETRY MESSAGES SENT 44

Nel riquadro evidenziato in figura possiamo verificare l'utilizzo dell'IoT Hub: in questo caso vediamo che c'è un solo dispositivo registrato (la nostra RPI) e oggi questo dispositivo ha inviato 119 messaggi al mio IoT

Hub. Ovviamente i vostri numeri potranno essere diversi, ma dovranno comunque aumentare quando l'applicazione sulla RPI è in funzione. Vi faccio notare che è possibile connettere più device allo stesso IoT Hub e che il limite massimo di messaggi giornalieri varia in base al piano tariffario selezionato al [punto 4](#). Se avete creato un IoT Hub gratuito il vostro limite sarà 8K messaggi al giorno invece dei 400K che vedete in figura.

Ma c'è anche un altro strumento che ci permette di verificare se i messaggi arrivano effettivamente all'IoT Hub, fornendo anche una semplice ma utilissima visualizzazione dei messaggi stessi. Si tratta dell'[Azure Device Explorer](#).



E' uno strumento molto semplice, ma utilissimo per il monitoring dell'IoT Hub e quindi per fare debug delle nostre applicazioni. Per prima cosa, occorre associarlo con il nostro IoT Hub.

Come mostrato in figura, nella tab *Configuration* deve essere inserita la *connection string*, che possiamo copiare direttamente dal portale di Azure, come evidenziato in figura.

A questo punto, cliccando su *Update* nel Device Explorer sarà stabilita la connessione con l'IoT Hub, e nella tab *Data* sarà possibile visualizzare i messaggi che gli arrivano, specificando il nome dell'IoT Hub (che, dal momento che ne abbiamo uno solo, dovrebbe comparire automaticamente) e il nome del device che si vuole monitorare (che può essere selezionato tramite la combobox). Cliccando infine sul tasto *Monitor* possiamo vedere i messaggi che arrivano in formato Json. Di seguito il risultato finale.

```

6863,"xToSend":0.0390625,"yToSend":0.015625,"zToSend":0.984375}]
07/09/2016 10:50:09> Device: [FezHatRPI], Data:
[{"deviceId":"Batman","temperatureToSend":28.687782805429869,"lightToSend":0.8784313725490
196,"xToSend":0.0390625,"yToSend":0.01171875,"zToSend":0.98828125}]
07/09/2016 10:50:10> Device: [FezHatRPI], Data:
[{"deviceId":"Batman","temperatureToSend":28.687782805429869,"lightToSend":0.8745098039215
6863,"xToSend":0.04296875,"yToSend":0.01171875,"zToSend":0.984375}]
07/09/2016 10:50:11> Device: [FezHatRPI], Data:
[{"deviceId":"Batman","temperatureToSend":28.687782805429869,"lightToSend":0.8784313725490
196,"xToSend":0.04296875,"yToSend":0.015625,"zToSend":0.9921875}]
07/09/2016 10:50:12> Device: [FezHatRPI], Data:
[{"deviceId":"Batman","temperatureToSend":28.687782805429869,"lightToSend":0.8745098039215
6863,"xToSend":0.0390625,"yToSend":0.015625,"zToSend":0.98828125}]
07/09/2016 10:50:13> Device: [FezHatRPI], Data:
[{"deviceId":"Batman","temperatureToSend":28.687782805429869,"lightToSend":0.8745098039215
6863,"xToSend":0.04296875,"yToSend":0.01953125,"zToSend":0.9921875}]

```

7 - Analisi e gestione dei dati: Azure Stream Analytics e Blob Storage

A questo punto, i nostri dati vengono rilevati dai sensori sulla RPI e inviati correttamente all'IoT Hub. Ora dobbiamo utilizzarli in modo che ci forniscano un valore!

Per questo laboratorio vogliamo memorizzare i dati in un Blob Storage e, parallelamente, inviarli a Power BI per averne una rappresentazione grafica.

7a - Azure Blob Storage

Iniziamo dalla creazione del Blob, che sarà poi uno degli output del nostro *Stream Analytics*. Nel portale di Azure <http://portal.azure.com> selezioniamo *New > Data + Storage > Storage Account*.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with a 'New' button highlighted by a red box. The main area is titled 'Data + Storage' and lists several options like Virtual Machines, Web + Mobile, Data + Storage, etc. Under 'Data + Storage', the 'Storage account' option is highlighted with a red box. To the right, a 'Create storage account' dialog box is open. It asks for a 'Name' (set to 'labiotstorage'), a 'Resource group' (set to 'LabIoT' with the 'Use existing' radio button selected), and a 'Location' (set to 'West Europe'). A large red arrow points to the 'Create' button at the bottom right of the dialog.

Assegniamo un nome al nostro Storage e nella selezione del *resource group* ricordiamo di selezionare *Use existing*, inserendo lo stesso gruppo di risorse in cui abbiamo creato il nostro IoT Hub al [punto 4](#) (il cui nome è *LabIoT*).

7b - Azure Stream Analytics

Stream Analytics è un servizio che permette di effettuare analisi dei flussi di dati in tempo reale, partendo da uno (o più) flussi in input e producendo uno (o più) flussi in output. Tramite questo servizio i dati possono essere elaborati in diversi modi, implementando query in un linguaggio SQL-like. Ne vedremo un paio molto (forse troppo 😊) semplici.

Torniamo sul portale di Azure: <http://portal.azure.com> e procediamo alla creazione di un nuovo *Stream Analytics*, come mostrato in figura. Chiamiamolo *labiotsa*.

The screenshot shows three windows from the Microsoft Azure portal:

- Left Window:** Shows the main navigation menu with a red box around the "New" button.
- Middle Window:** Shows the "New" blade for "Internet of Things" services. The "Internet of Things" category is selected (red box), and the "Stream Analytics job" item is highlighted with a red box.
- Right Window:** The "New Stream Analytics J..." configuration dialog. It includes fields for "Job name" (labeled with a red arrow), "Subscription" (Dpe Italy), "Resource group" (LabolT, with a red box around the "Use existing" radio button), "Location" (West Europe), and a "Create" button (boxed in red).

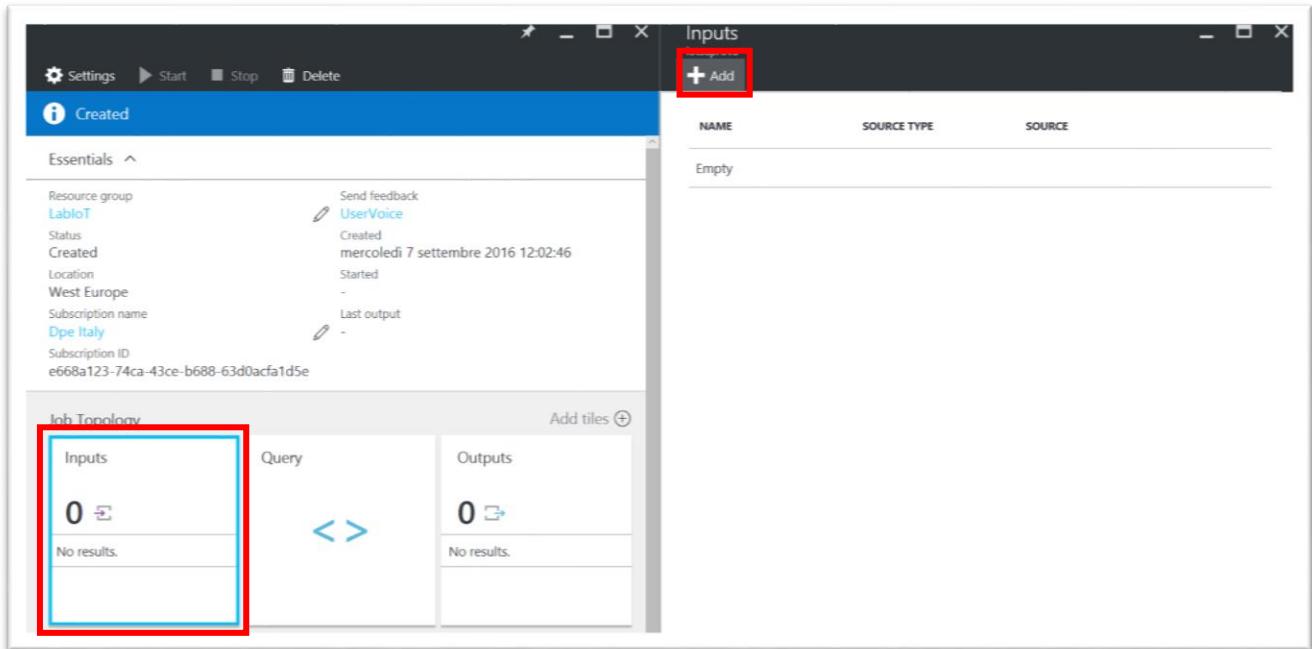
Di nuovo, facciamo attenzione a selezionare *Use existing* e inserire lo stesso *Resource Group* in cui abbiamo inserito l'*IoT Hub* creato al [punto 4](#), come mostrato in figura qui sopra.

Ora dobbiamo configurare *Input*, *Query* e *Output* del nostro servizio, per ottenere questo risultato.

The screenshot shows the Azure Stream Analytics job "labiotsa" in the "Stopped" state. The "Job Topology" section is highlighted with a red box and contains the following details:

Inputs	Query	Outputs
1 iothub	<>	2 ToBlobStorage ToPowerBI

Cominciamo dalla configurazione dell'Input. Clicchiamo su *Inputs > Add* per inserire un nuovo flusso di dati in ingresso.



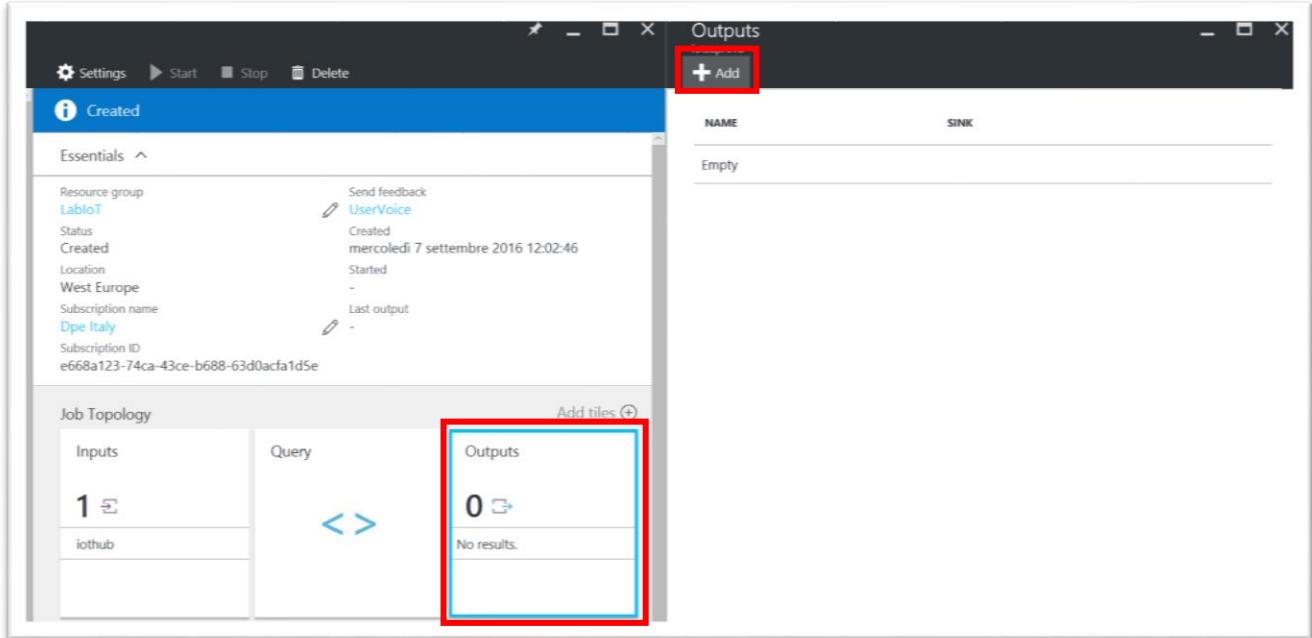
Inseriamo i seguenti parametri e clicchiamo infine sul tasto *Create* in basso.

Input alias	iothub
Source Type	Data stream
Source	IoT hub
Subscription	Use IoT hub from current subscription
IoT hub	iothub4lab
Endpoint	Messaging
Shared access policy name	iothubowner
Shared access policy key	*****
Consumer group	\$Default
Event serialization format	JSON
Encoding	UTF-8

Inseriamo un alias di input (in questo caso *iothub*), selezioniamo IoT Hub come *Source* e lasciamo l'opzione di default per quanto riguarda la *Subscription*. Vediamo che automaticamente gli altri campi si popoleranno con le informazioni necessarie.

Per lo scopo di questo laboratorio, non occorre variare le impostazioni di default.

Ora passiamo all'Output. Come prima, selezioniamo il riquadro *Output* dello Stream Analytics > *Add*.



This screenshot shows the 'New output' configuration dialog. It has several fields with red arrows pointing to specific ones:

- * Output alias: toBlobStorage
- * Sink: Blob storage
- * Subscription: Use blob storage from current subscription
- * Storage account: labiotdatastorage1
- * Container: Create a new container
- * Container: blobcontainer

At the bottom, there's a 'Create' button highlighted by a red box.

Selezioniamo un alias per l'output dello Stream Analytics (in questo caso *toBlobStorage*), il tipo da selezionare nel campo *Sink* è *Blob Storage*. Selezioniamo quindi il nome del Blob creato al [punto 7a](#) e infine creiamo un nuovo *Container*, a cui possiamo assegnare un nome a nostro piacimento (in questo caso *blobcontainer*).

Ricordiamo infine di cliccare il tasto *Create* in basso epr finalizzare la creazione.

A questo punto dobbiamo scrivere la query per far passare i dati dall'input all'output. Selezioniamo il riquadro centrale e scriviamo la seguente query per trasferire *tutti* i dati dall'IoT Hub al Blob storage. Vi faccio notare che sono stati utilizzati gli *alias di input e di output* scelti nel punto precedente. Per semplificare, non viene fatto alcun tipo di elaborazione ai dati in ingresso, ma volendo si possono inserire query anche piuttosto complesse.

```

1 SELECT
2 *
3 INTO
4 ToBlobStorage
5 FROM
6 iothub
    
```

A questo punto lo Stream Analytics è completo (almeno per il momento) e possiamo avviarlo tramite il pulsante *Start* in alto.

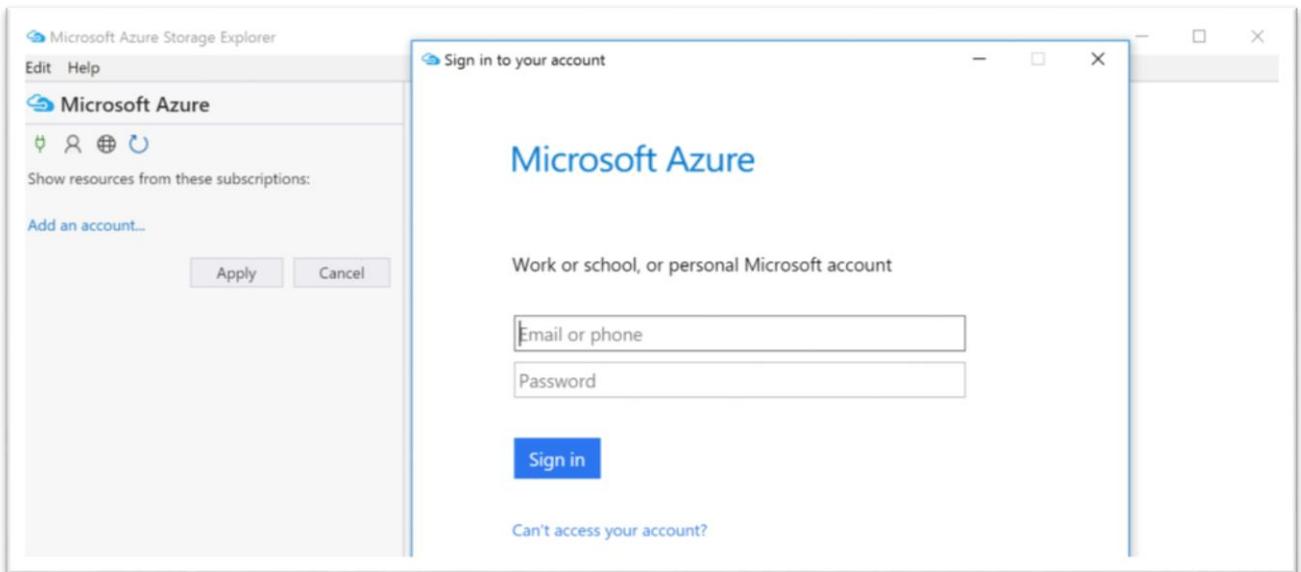


7c - Visualizzazione dei dati memorizzati

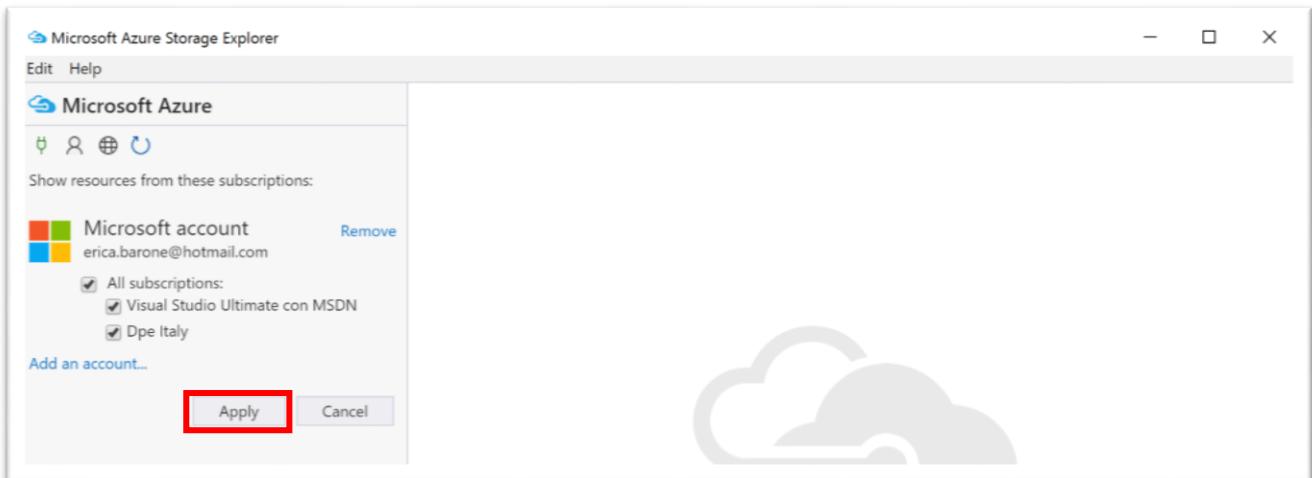
Verifichiamo ora che i dati vengano effettivamente memorizzati all'interno del nostro Blob. Per farlo, utilizziamo l'applicazione [Azure Storage Explorer](#). Una volta scaricato il tool, la prima cosa da fare è collegarlo al nostro account Azure, selezionando l'icona *account > Add an account* e si aprirà la finestra in



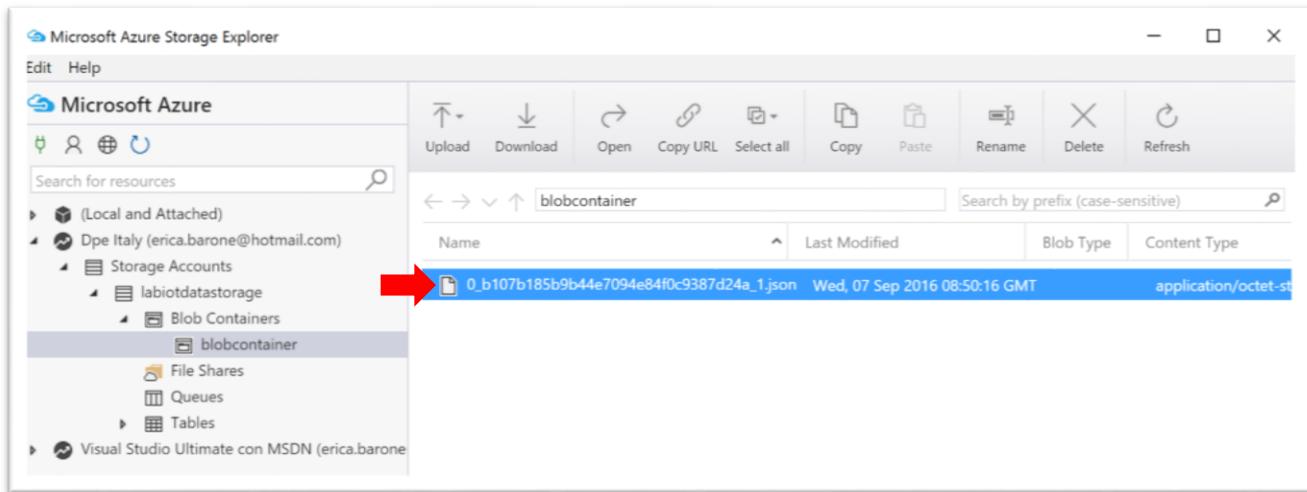
cui inserire le credenziali di accesso, come mostrato in figura.



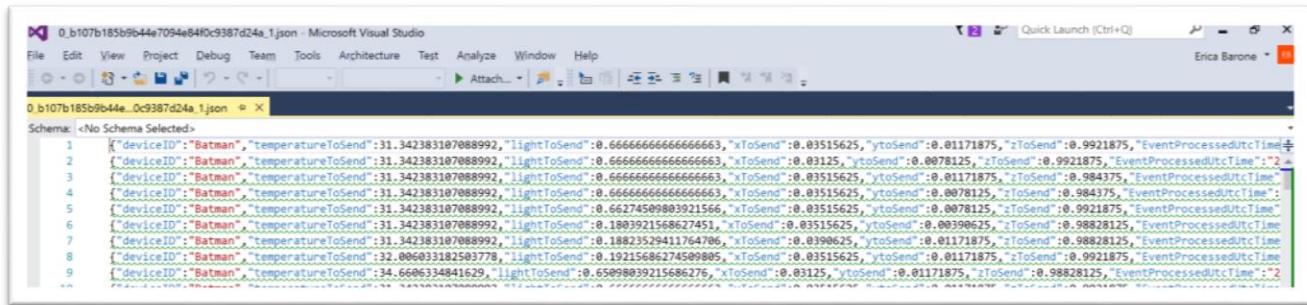
Una volta inserite le credenziali, comparirà il nostro account con tutte le sottoscrizioni ad esso associate (nel mio caso sono due, ma per voi dovrebbe essere una sola). Possiamo selezionare quale visualizzare e quale no, infine clicchiamo su *Apply*.



A questo punto all'interno della sottoscrizione su cui stiamo lavorando vedremo il nostro *Storage Account*, apriamo tutto fino ad arrivare al nostro *blob container* (il quale avrà naturalmente il nome che gli abbiamo assegnato al [punto 7b](#), nella configurazione dell'output dello Stream Analytics. Nel nostro caso *blobcontainer*).



Facendo ora doppio click sul nostro blob saremo in grado di visualizzare (anche direttamente da Visual Studio) i messaggi che sono partiti dalla nostra RPI e sono arrivati al blob, in formato Json.

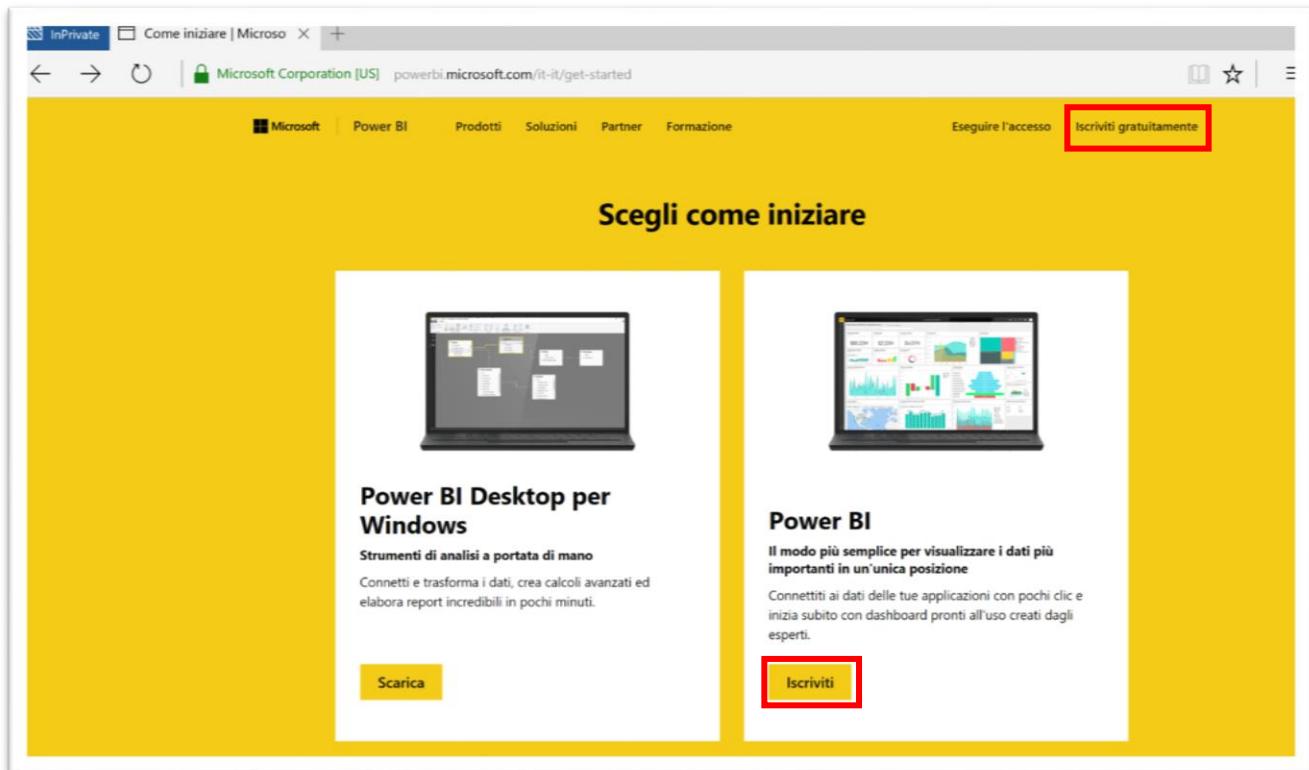


Vi faccio presente che, perchè il blob venga creato, l'applicazione deve girare sulla RPI, inviando i dati ad IoT Hub, in modo che il job dello Stream Analytics (attivato al punto precedente) sia in grado di prenderli in ingresso e inviarli allo storage. Se anche solo uno di questi passaggi non funziona, allora nell'Azure Storage explorer non vedremo nulla.

8 - Visualizzazione dei dati sotto forma di grafico: Power BI

L'ultimo step di questo laboratorio consiste nel creare una dashboard usando Power BI che mostri l'andamento dei dati inviati dalla RPI. Per farlo, la prima cosa da fare è, ovviamente, ottenere l'accesso a Power BI. Se siamo in possesso di un account aziendale è tutto molto semplice, in caso contrario la procedura è un po' più articolata, ma la vediamo insieme passo passo.

Il modo più semplice per verificare il nostro è un account aziendale è provare ad iscriverci gratuitamente, direttamente da <https://powerbi.microsoft.com>



Se l'indirizzo email che utilizziamo è aziendale, allora l'iscrizione sarà immediata e possiamo passare direttamente al [punto 8b](#), in caso contrario invece questo è il messaggio che otterremo:



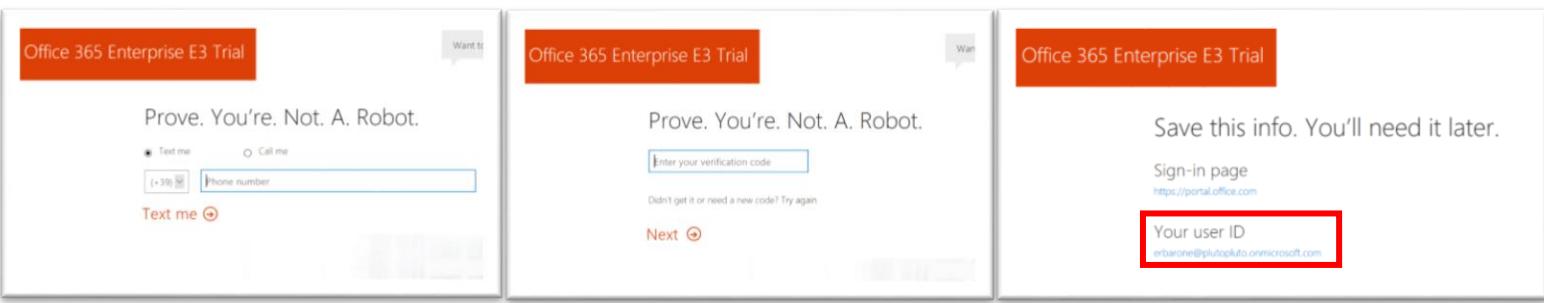
Vediamo allora come creare gratuitamente un indirizzo aziendale da poter utilizzare per accedere a PowerBI.

8a - Office365 Enterprise E3 Trial

Il primo step è attivare la trial gratuita di 30 giorni di O365, la versione Enterprise E3 è quella che ci permette di avere a disposizione anche PowerBI. Inseriamo nel browser questo indirizzo <https://products.office.com/en-gb/business/office-365-enterprise-e3-business-software> e clicchiamo su *Free Trial*.

A questo punto ci viene chiesto di inserire alcuni dati per la registrazione. Compiliamo la prima form con i nostri dati, e poi in quella successiva inseriamo quello che sarà il nostro account aziendale, da usare poi per accedere a PowerBI.

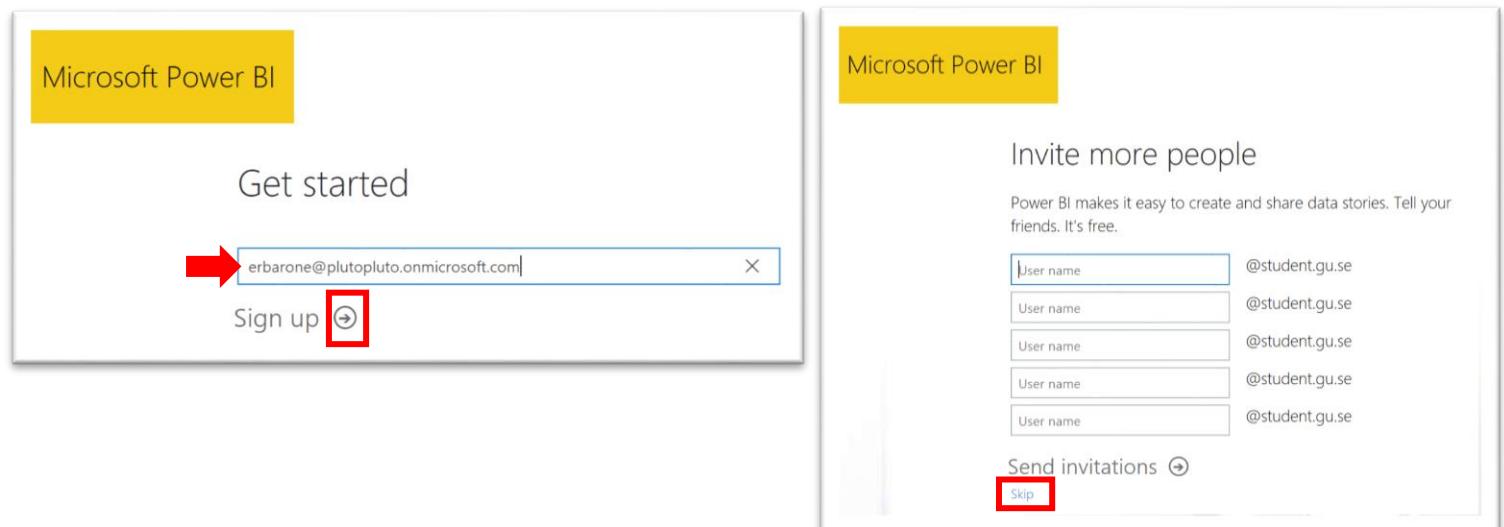
Verrà chiesta una verifica per confermare l'account, procediamo seguendo la procedura guidata.



Evidenziato nell'ultimo step vediamo l'account aziendale da utilizzare per l'accesso a PowerBI. Vi faccio presente che la trial di Office365 durerà 30 giorni.

8b - Registrazione a PoweBI

Torniamo quindi al sito di PowerBI, seguendo gli stessi step descritti [all'inizio del punto 8](#) possiamo ora inserire il nostro nuovo account aziendale, cliccando poi *Skip* nella schermata successiva.



Manca però ancora un passaggio: dobbiamo fare in modo che i dati che arrivano dalla RPI vengano in qualche modo trasferiti come dataset al nostro PowerBI. Per fare questo, sfruttiamo lo Stream Analytics *labiotsa* creato al [punto 7b](#), aggiungendogli un nuovo output.

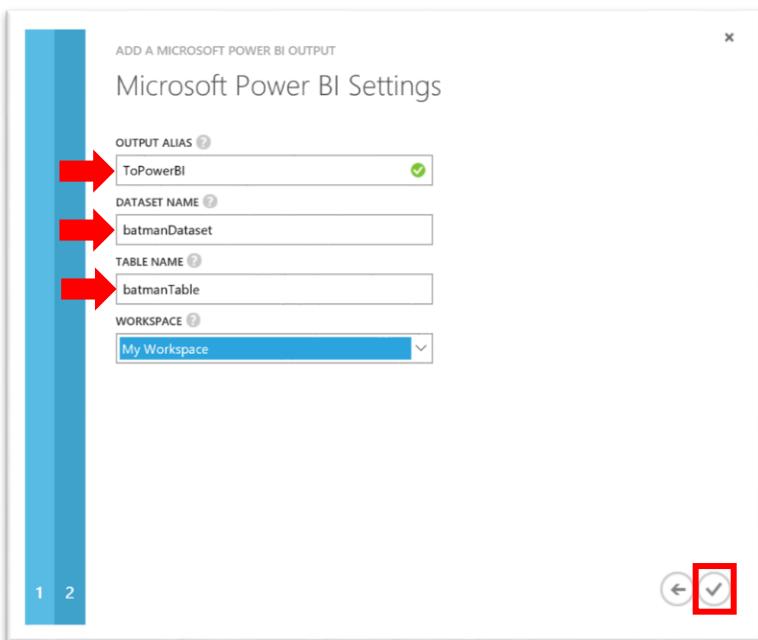
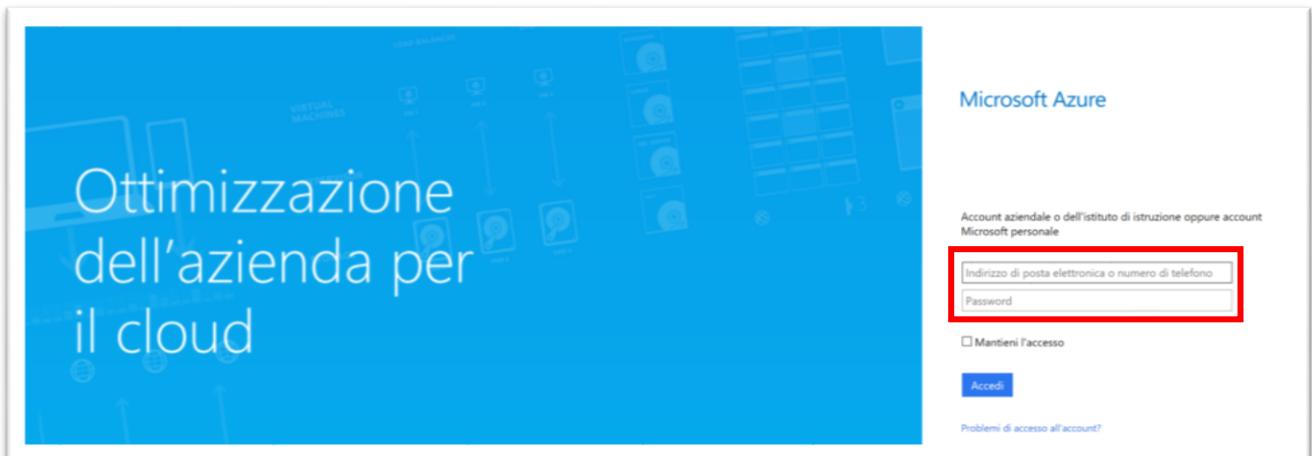
8c – Un nuovo output per Stream Analytics: PowerBI

In questo caso dobbiamo accedere al vecchio portale di Azure, in quanto nel momento in cui questo laboratorio è stato realizzato la procedura di configurazione di PowerBI come output di Stream Analytics non è ancora disponibile sul portale nuovo.

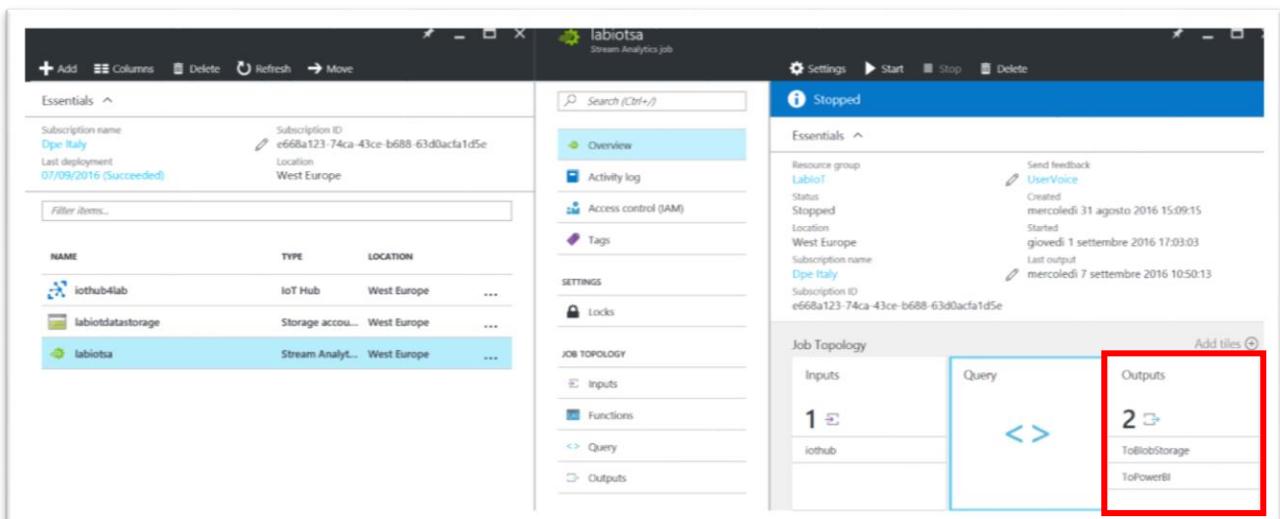
Per accedere al vecchio portale di Microsoft Azure possiamo utilizzare questo link:
<http://manage.windowsazure.com>

La schermata che si apre contiene tutti i *resource group* e tutti i servizi creati dal nuovo portale, ma la UI è un po' diversa. Selezioniamo il nostro Stream Analytics, che al [punto 7b](#) avevamo chiamato *labiotsa* e inseriamo un nuovo output.

Seguendo la procedura guidata, possiamo facilmente creare un output di tipo PowerBI, autorizzando Azure ad accedere tramite le nostre credenziali PowerBI (ovvero l'account aziendale creato al [punto 8a](#)) ed inserendo infine un *alias*, un *dataset name*, un *table name*.



Torniamo ora sul nuovo portale di Azure, più familiare: <http://portal.azure.com> e verifichiamo che il nostro Stream Analytics abbia ora 2 output, come mostrato in figura.



A questo punto però dobbiamo modificare la query, per fare in modo che i dati vengano mandati anche al nostro dataset su PowerBI, oltre al blob. Anche in questo caso abbiamo scelto la procedura più semplice, ovvero il trasferimento di tutti i dati senza modifiche o aggregazioni.

Selezioniamo quindi il riquadro *Query* e aggiungiamo le istruzioni evidenziate in figura. Vi faccio notare che il job non può essere in *running*, in caso contrario non riusciremo a modificare la query.

```

1 SELECT *
2   *
3 INTO ToPowerBI
4 FROM iothub
5
6
7
8
9
10 SELECT *
11   *
12 INTO ToBlobStorage
13 FROM iothub
14
15 ToPowerBI

```

Ora non ci resta che salvare, far ripartire il job di Stream Analytics tramite il tasto *Start*, controllare che l'applicazione sulla RPI stia ancora funzionando e infine verificare che i dati arrivino effettivamente sulla nostra PowerBI.

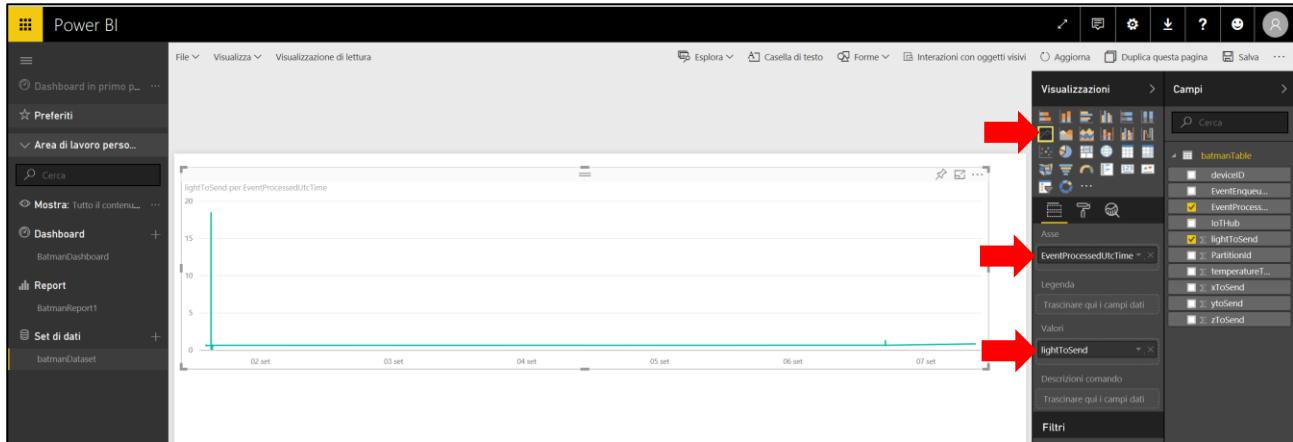
8d – La Dashboard su PowerBI

Torniamo sul sito di PowerBI <https://powerbi.microsoft.com>, accedendo con le credenziali aziendali ottenute al [punto 8a](#). Di seguito vediamo la schermata di default, verifichiamo che a sinistra sia presente un Dataset con lo stesso nome di quello creato al [punto 8c](#) come output di Stream Analytics, nel nostro caso *batmanDataset*.

Sulla sinistra possiamo vedere l'elenco di tutti i campi contenuti in questo Dataset, che corrispondono a quello che l'applicazione sulla RPI invia al cloud (*più alcuni campi "automatici", presenti perchè la query dello Stream Analytics prevede la trasmissione di tutti i campi (*). Volendo possono essere eliminati modificando appunto la query e selezionando, ad esempio, solo quelli di nostro interesse*).

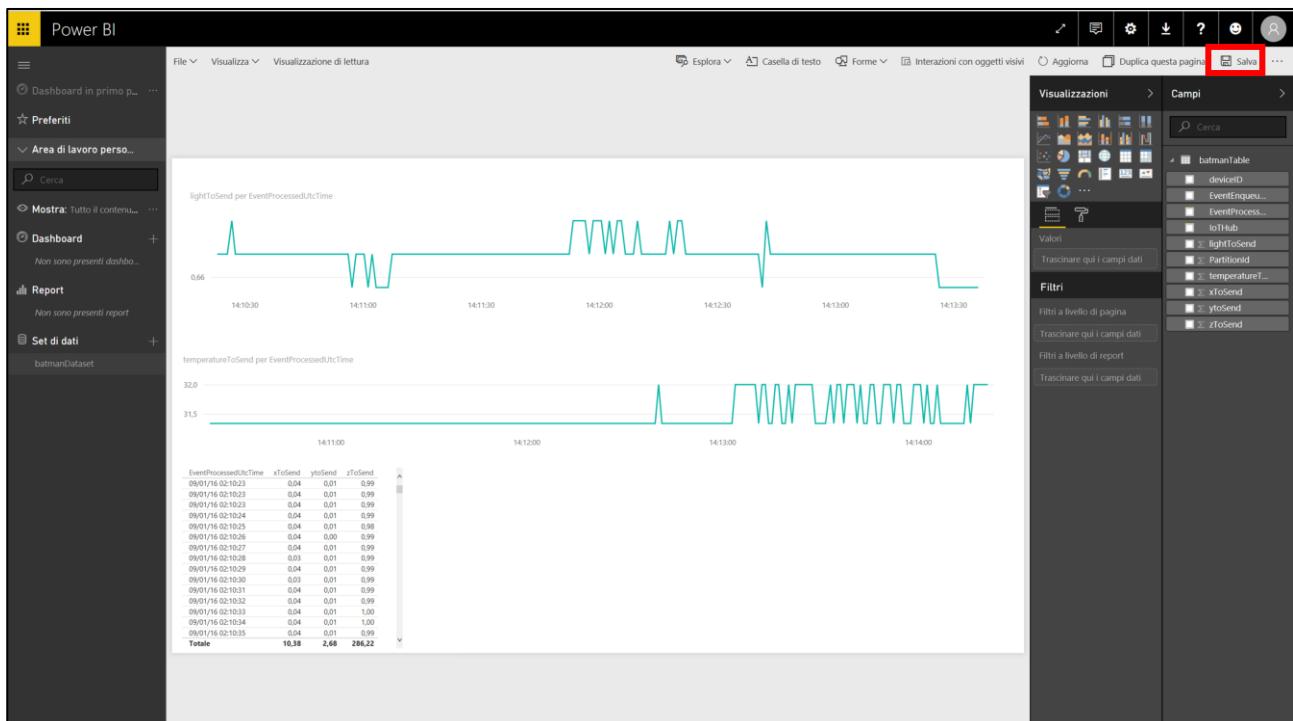
Ora dobbiamo creare un report con alcuni grafici, e in seguito la dashboard.

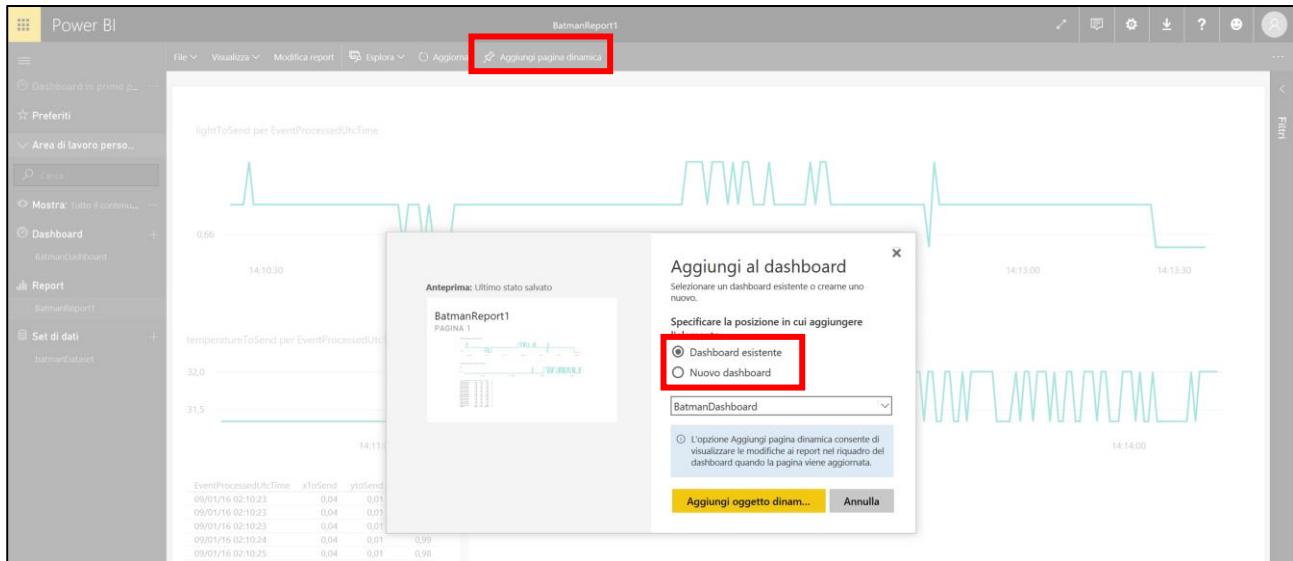
Selezioniamo quindi il tipo di grafico e inseriamo come *Valori* il *lightToSend* (che corrisponde al valore di luce inviato dalla RPI), come *Asse* inseriamo invece *EventProcessedUtcTime* (che identifica l'istante in cui il dato è stato processato dallo Stream Analytics). Questo è il risultato che otteniamo.



Procediamo in modo analogo per tutti gli altri valori che ci interessano, possiamo anche inserire filtri o selezionare tipi di grafico differenti.

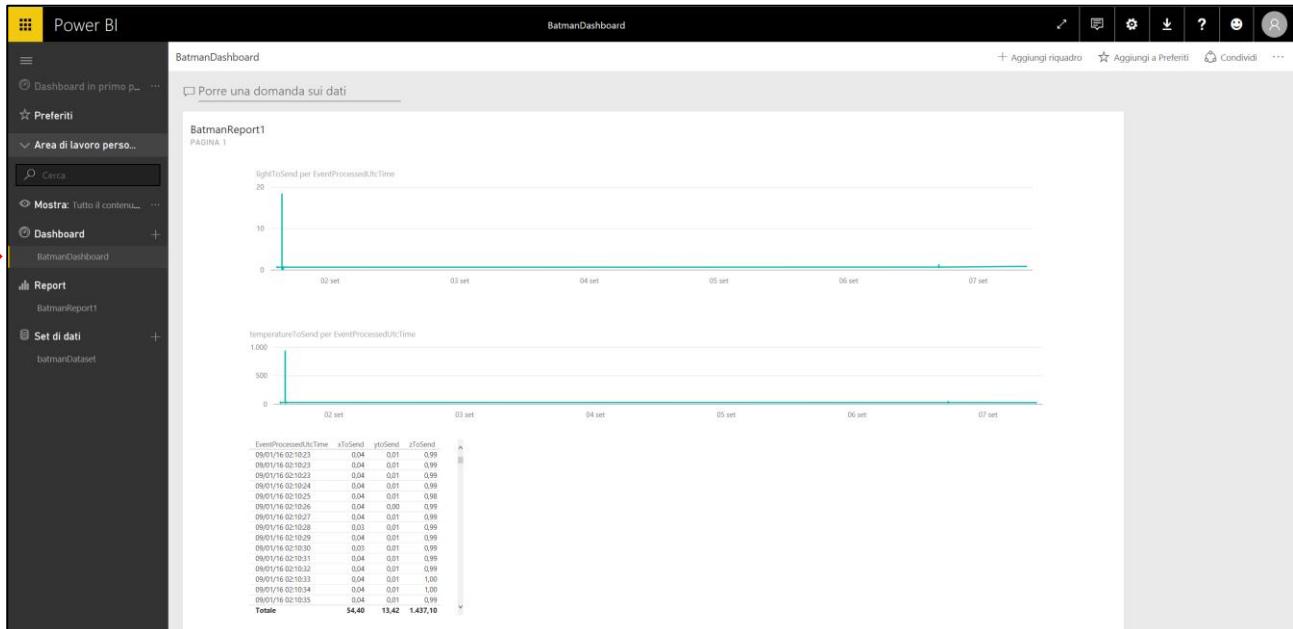
Una volta completato il nostro report, lo salviamo e lo aggiungiamo alla dashboard, cliccando su *Aggiungi pagina dinamica*.





A seconda che sia già stata creata o meno una dashboard, selezioniamo *Dashboard esistente* o *Nuovo Dashboard* nel popup.

Ed ecco che finalmente possiamo visualizzare i grafici dei dati che vengono letti dai sensori presenti sulla scheda direttamente nella nostra dashboard PowerBI.



9 - Conclusioni

In questo laboratorio abbiamo creato tutto ciò che serve per trasferire i dati dai sensori fino alla dashboard. L'applicazione è volutamente molto semplice ma completa, in quanto siamo riusciti a toccare con mano Windows 10 IoT Core, lo sviluppo UWP, alcuni servizi di Azure indispensabili per realizzare applicazioni IoT come IoT Hub e Stream Analytics, strumenti di visualizzazione dei dati come la PowerBI. Ovviamente è possibile (e consigliabile) modificare e integrare questo progetto in tanti modi, sia lato client che lato cloud, per ottenere qualcosa di più complesso, particolare e significativo.

Enjoy 😊