



IoT Day

Hands on Lab

Internet of Things:

Windows 10 IoT Core, UWP Application, Azure IoT Hub, Stream Analytics, Blob Storage, Power BI.

Vediamo come mettere insieme i pezzi per trasmettere i dati dai sensori al cloud.

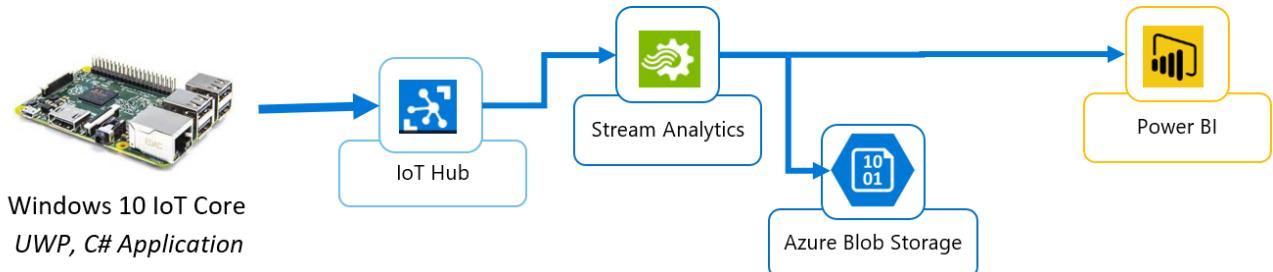
Indice

1 - Requisiti e strumenti necessari	3
2 – Windows 10 IoT Core.....	3
3 – Configurazione della RPI	5
3a - Networking	7
3b - Remote	7
3c – TPM Configuration	9
4 - Connessione a Microsoft Azure.....	10
5 - L'applicazione UWP per recuperare e mostrare temperatura, luce, coordinate dell'accelerometro	15
5a - Deployment da remoto sulla RPI.....	21
6 – Il percorso dei dati: dalla RPI ad Azure IoT Hub.....	22
7 - Analisi e gestione dei dati: Azure Stream Analytics e Blob Storage	24
7a - Azure Blob Storage	25
7b - Azure Stream Analytics.....	25
7c - Visualizzazione dei dati memorizzati	29
8 - Visualizzazione dei dati sotto forma di grafico: Power BI	31
8a - Office365 Enterprise E3 Trial	32
8b - Registrazione a PoweBI	33
8c – Un nuovo output per Stream Analytics: PowerBI	33
8d – La Dashboard su PowerBI	36
9 - Conclusioni	38

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.

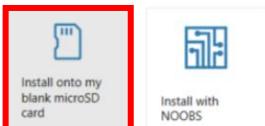
Choose your hardware, installation media, and OS version.

1 Select your hardware

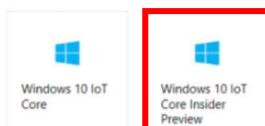
[Help me choose a device](#)



2 Select your installation media



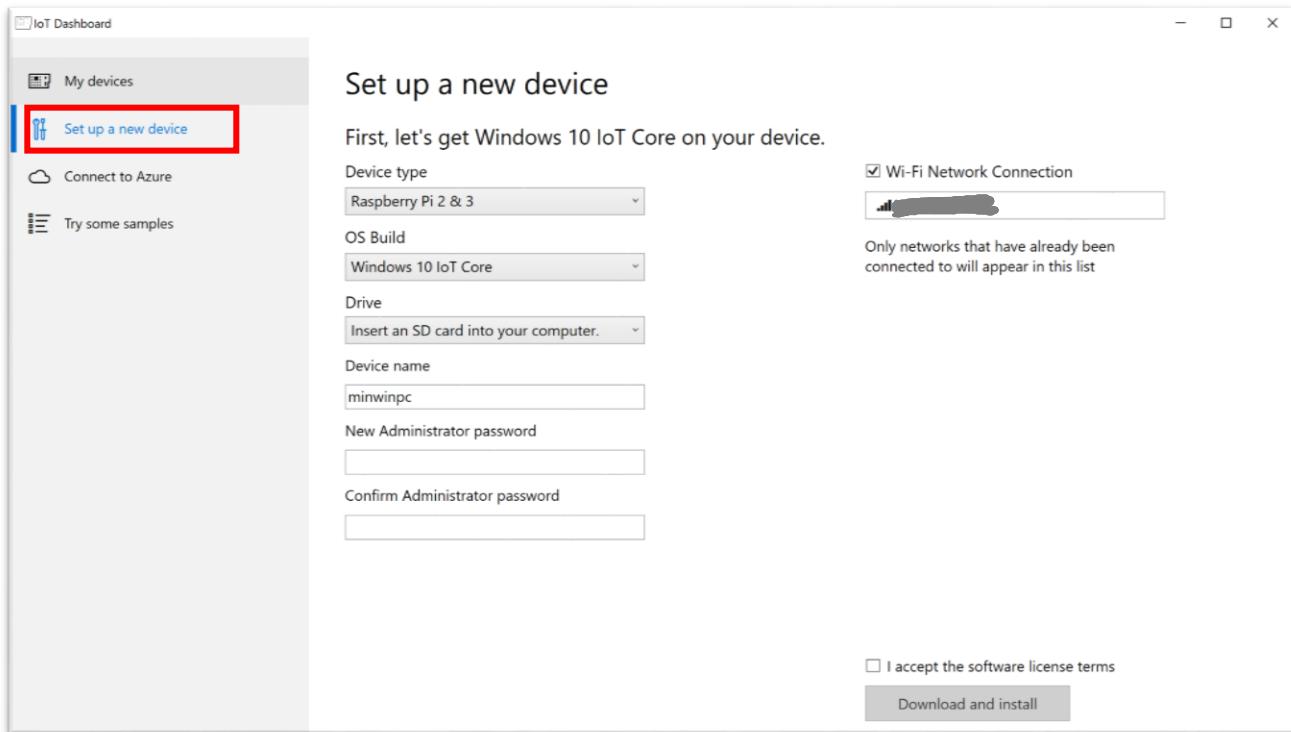
3 Select your OS version



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. Le schede preconfigurate per questo laboratorio usano la seguente password: iotLab



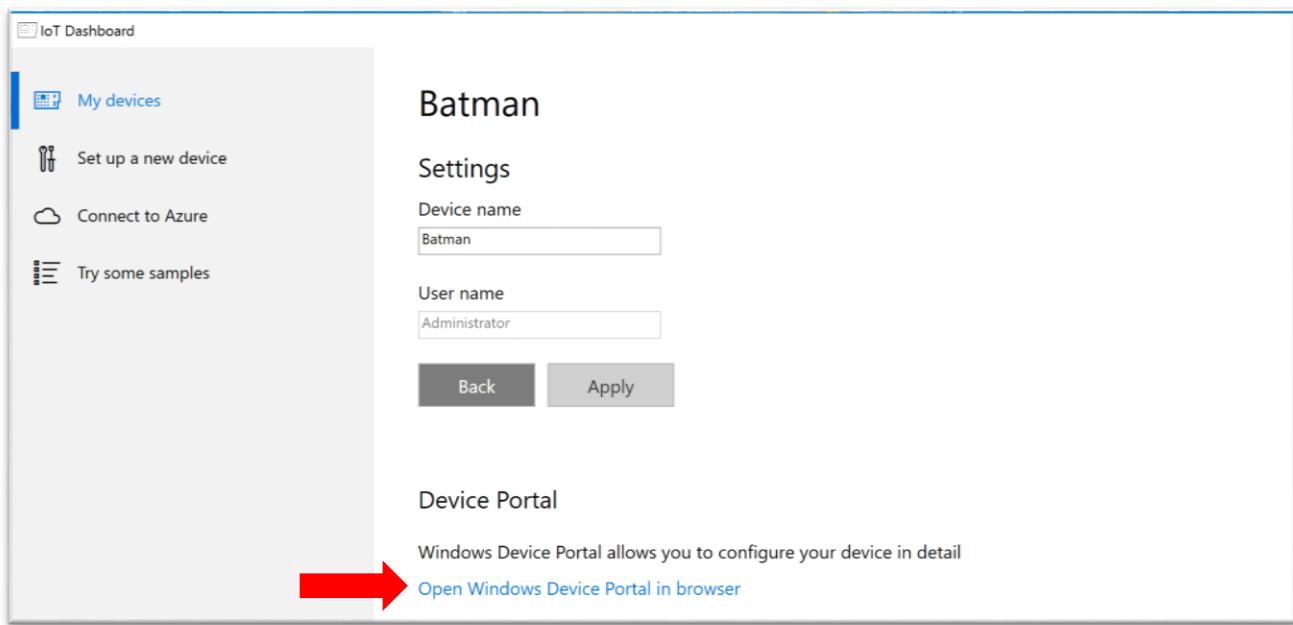
A questo punto verrà installato Windows 10 IoT Core sulla micro SD, che andrà poi inserita nella RPI. Ora è tutto pronto per accendere la scheda e iniziare a programmare!

3 – Configurazione della RPI

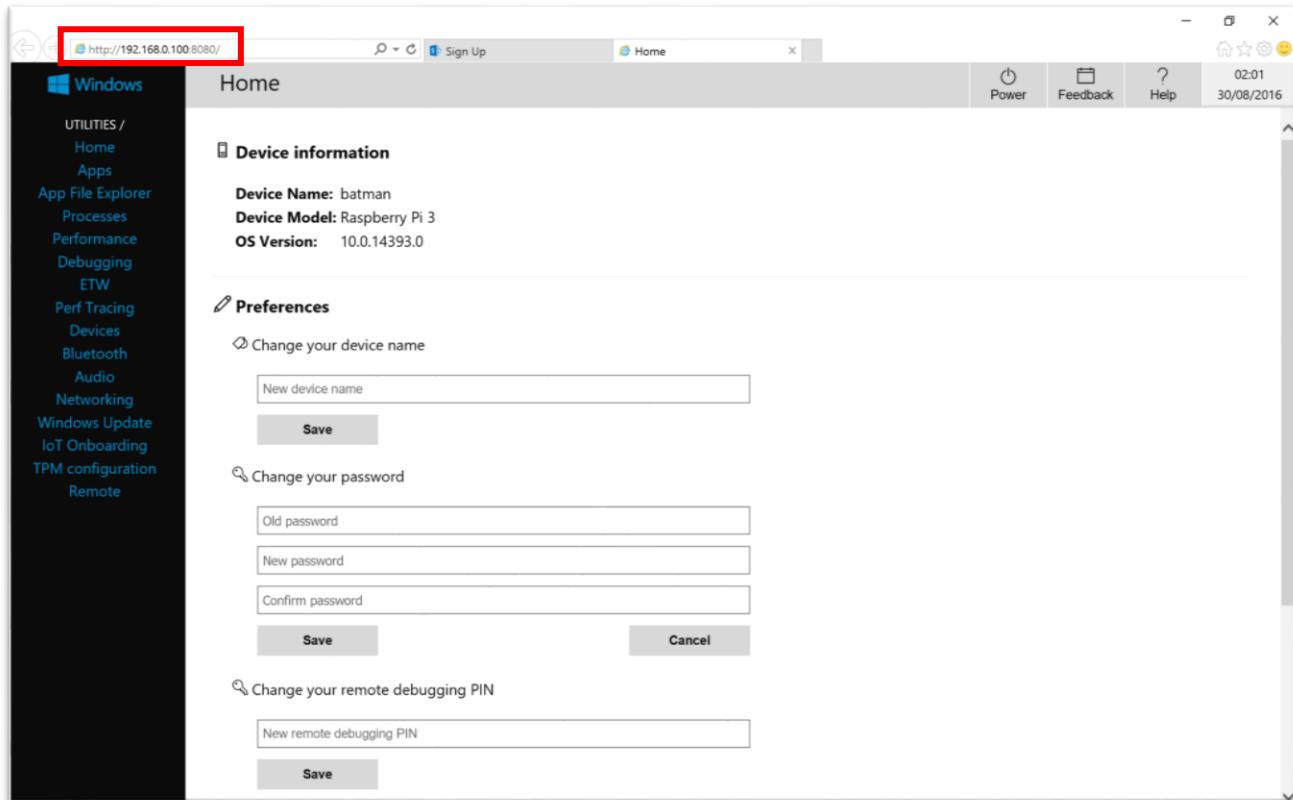
Requisito fondamentale per permettere la comunicazione tra il pc e la RPI è che siano connessi entrambi alla stessa sottorete (tramite cavo ethernet o wifi).

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*.

Name	Type	IP Address	Settings	OS
batman	Raspberry Pi 3	192.168.0.100		10.0.14393.0



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. La password per le schede preconfigurate per questo laboratorio è: *iotLab*. 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. The left sidebar has a dark theme with white text. The 'Networking' option is highlighted with a red arrow. The main content area has a light gray header bar with the title 'Networking'. Below it, under 'WiFi adapters', a dropdown menu shows 'Broadcom 802.11n Wireless SDIO Adapter'. Under 'Available networks', there is a table:

SSID	PROFILE	INFRA	SIGNAL	SECURITY	ENCRYPTION	CHANNEL
MSFTCORP		✓	■■■	WPA2	AES	
MSFTGUEST		✓	■■■	Open	None	

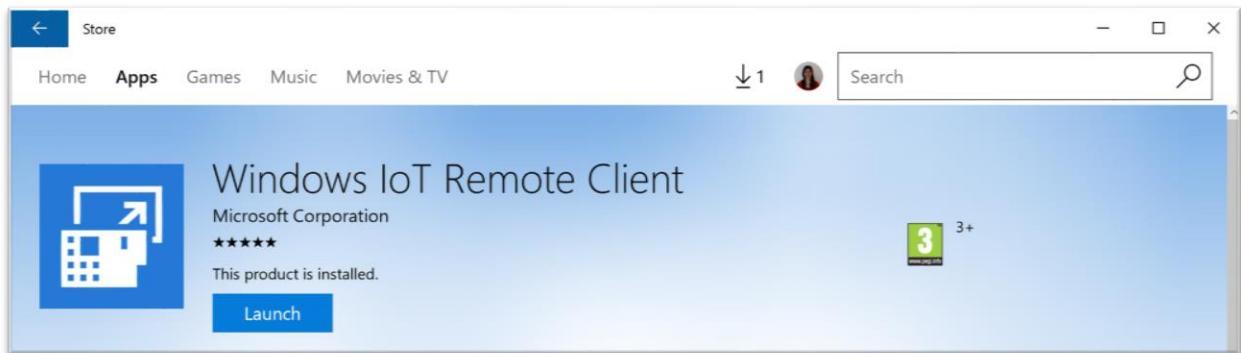
Below this is the 'IP configuration' section, which displays the following details:

- Description: Bluetooth Device (Personal Area Network)
- Type: Ethernet
- Physical address: b8-27-eb-2e-9a-63
- IPv4 address: 0.0.0.0
- Subnet mask: 0.0.0.0

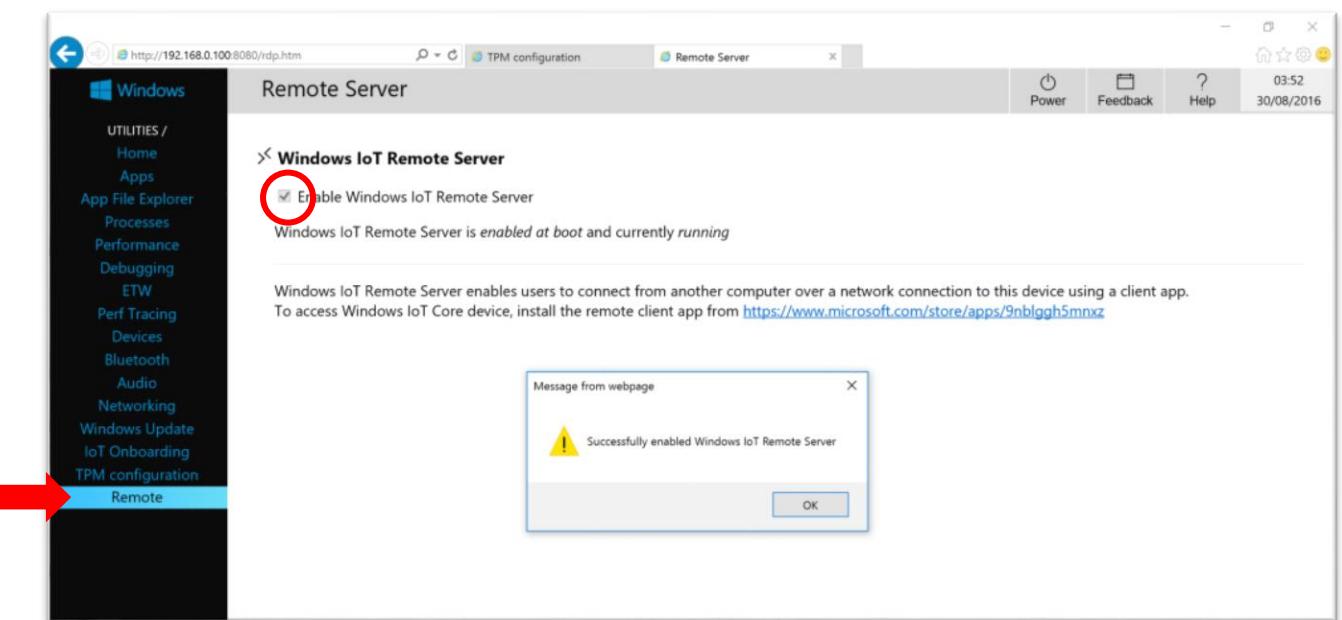
3b - Remote

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

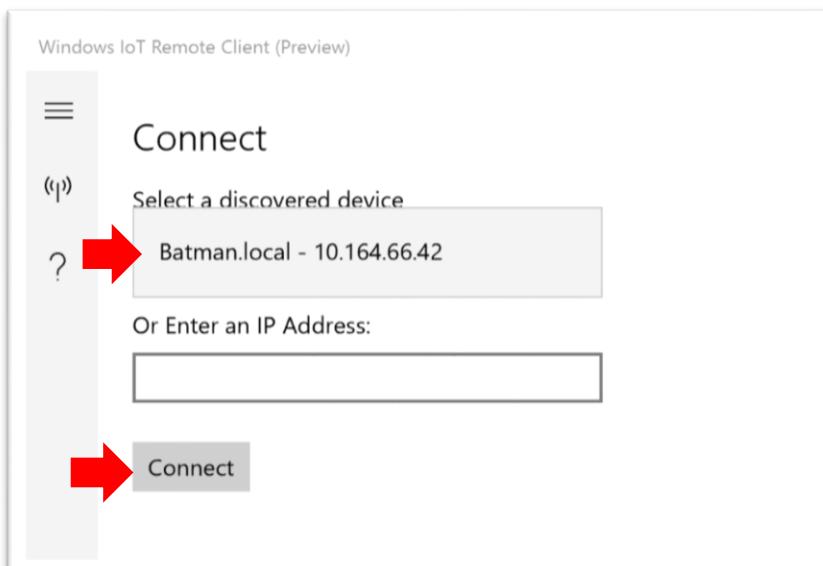
- 1 – scaricare l'app dal Windows Store: Windows IoT Remote Client (Preview)



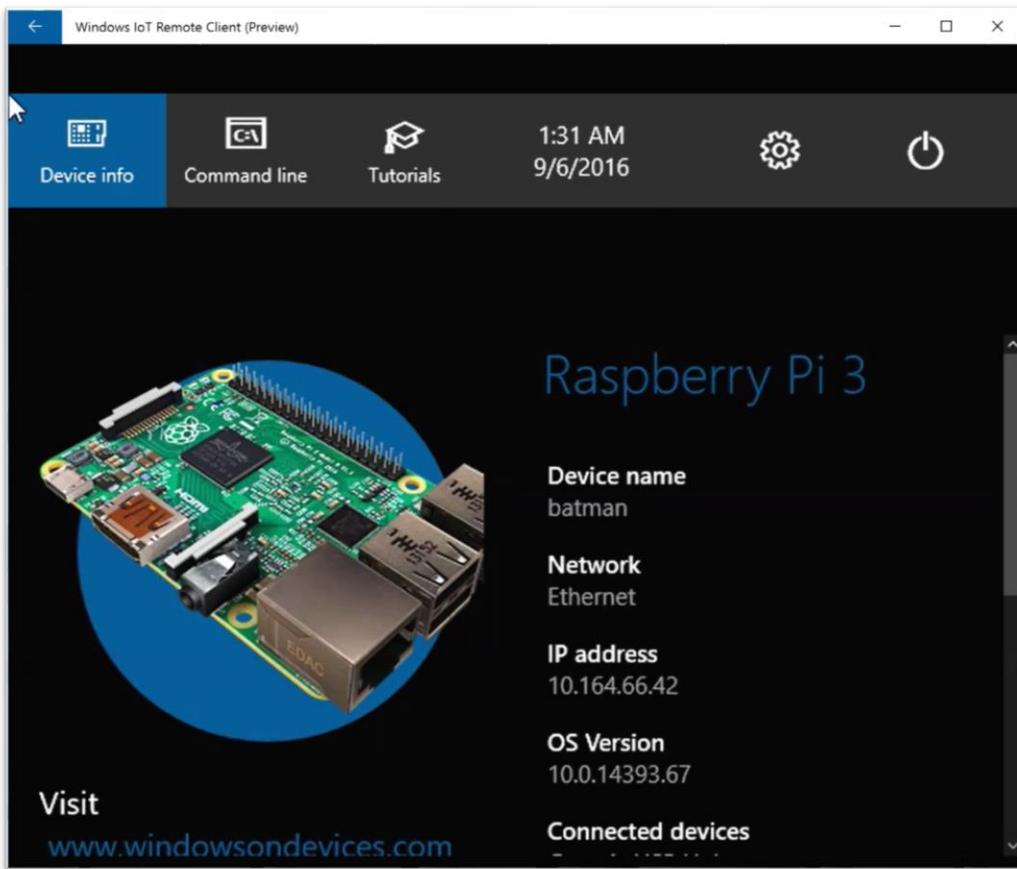
2 – Nella webpage della RPI alla sezione *Remote* abilitare il flag *Enable Windows IoT Remote Server*, come mostrato in figura



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.



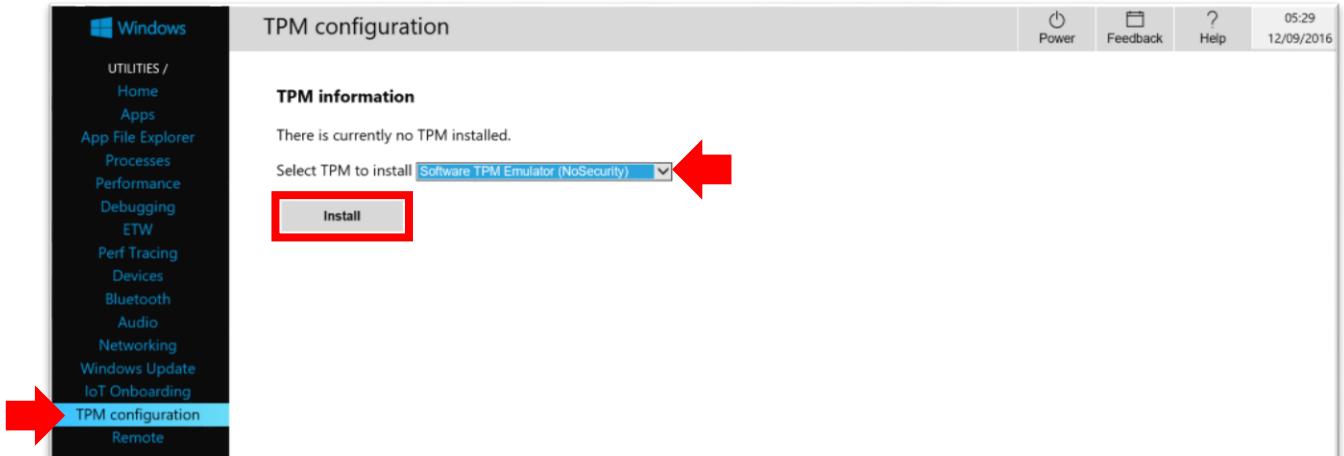
3c – TPM Configuration

Il Trusted Platform Module è un modulo in grado di fornire le funzioni base di generazione, cifratura e decifrazione di chiavi. Grazie ad esso è garantita la sicurezza delle chiavi generate dal modulo o in esso memorizzate, in quanto è del tutto indipendente dal sistema operativo e, se l'hardware viene alterato, il TPM non rilascia le chiavi memorizzate. Nel nostro caso, useremo un TPM Software (in quanto la RPI non possiede il modulo fisico) per memorizzare la chiave di connessione con IoT Hub.

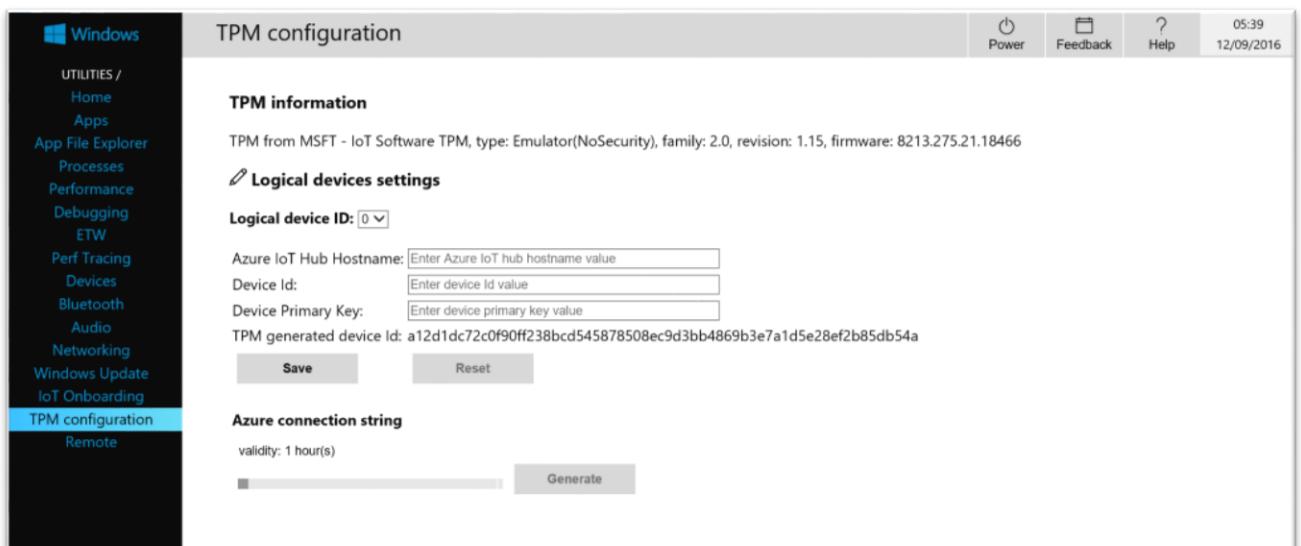
Nel momento in cui effettueremo la registrazione del dispositivo sul nostro IoT Hub ([punto 4](#)), la chiave sarà memorizzata nel TPM della scheda e, per poterla utilizzare e quindi per accedere al nostro IoT Hub e inviarci i dati, dovremo usare all'interno della nostra UWP Application i metodi della classe *Microsoft.Device.Tpm* tramite i quali possiamo generare un token di accesso in maniera del tutto trasparente al sistema operativo.

Per ulteriori informazioni sul TPM potete consultare la documentazione a questo link:
<https://developer.microsoft.com/en-us/windows/iot/docs/tpm>

Nella webpage della RPI, apriamo la sezione TPM Configuration e installiamo il Software TPM Emulator, come mostrato in figura.



A questo punto verrà richiesto il riavvio della scheda, accettiamo e, una volta completato, la schermata della sezione TPM Configuration sarà simile a quella mostrata in figura.

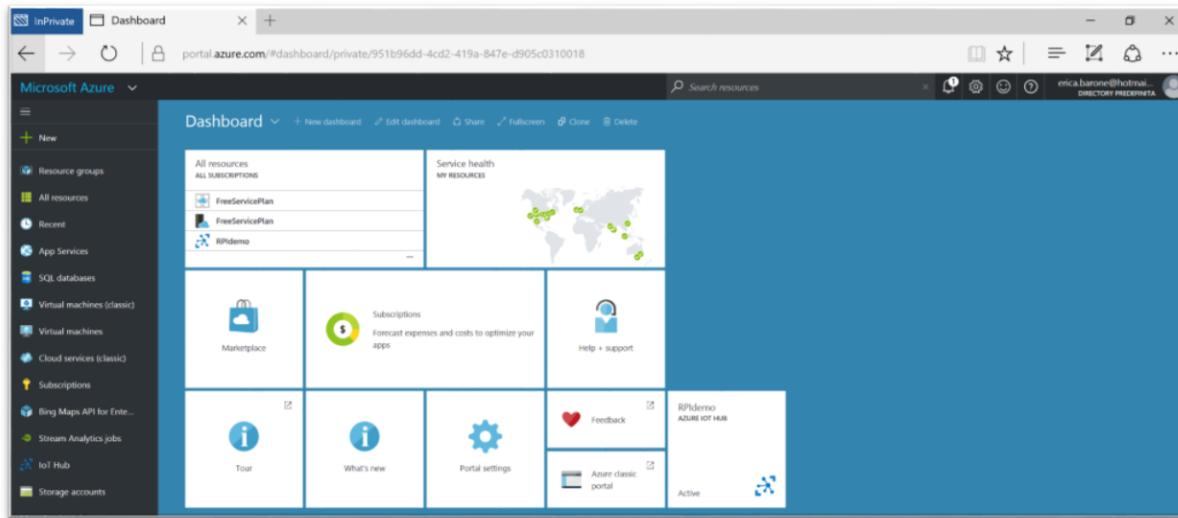


Vediamo ora come fare il *provisioning* del dispositivo, ovvero come generare la chiave da usare per la comunicazione tra il dispositivo e l'IoT Hub, che verrà poi memorizzata e gestita dal TPM.

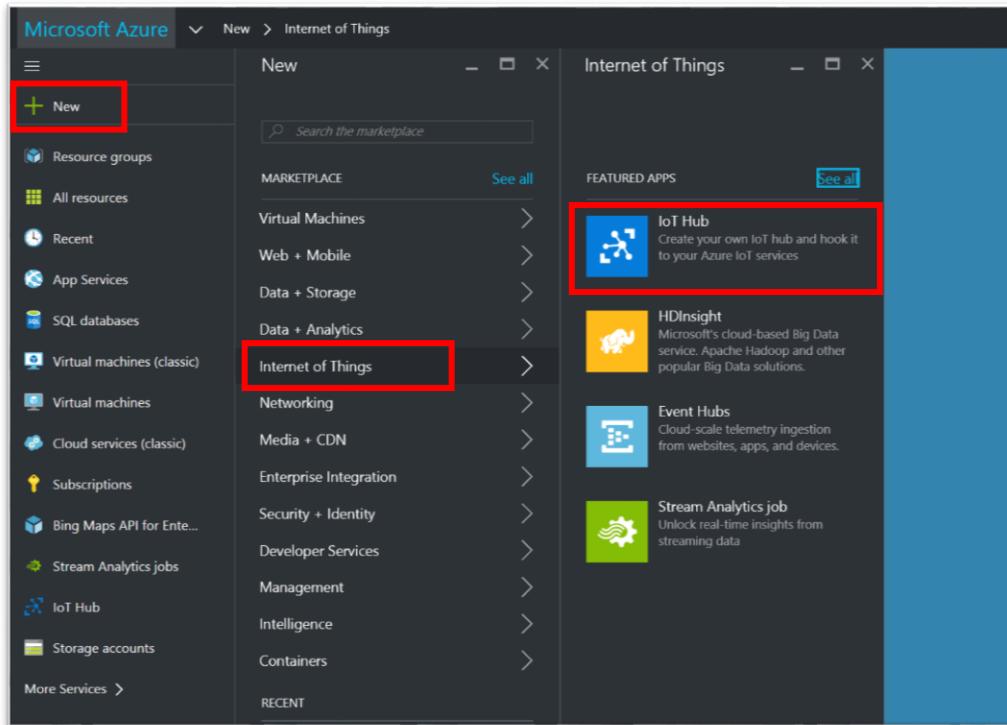
4 - Connessione a Microsoft Azure

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.

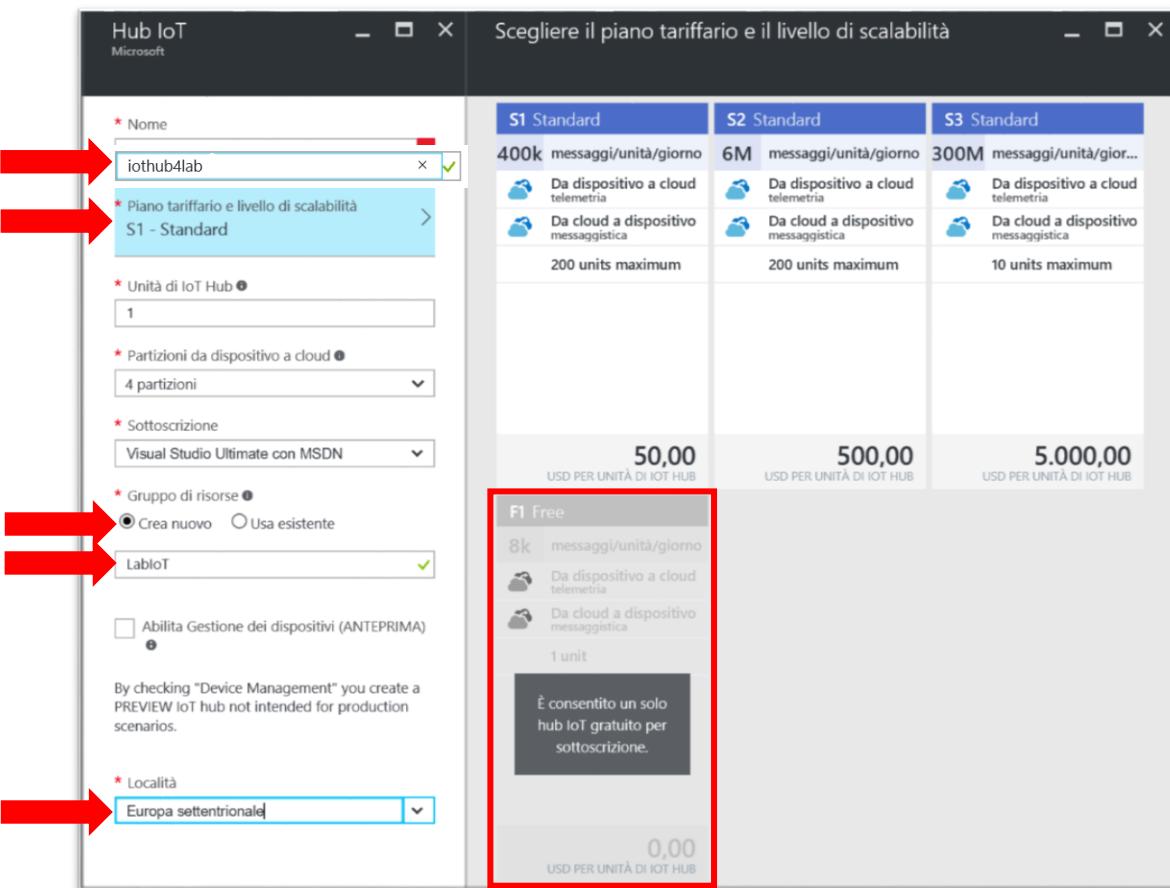
Apriamo il browser (in private mode) e accediamo al portale di Azure: <http://portal.azure.com>.



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



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.

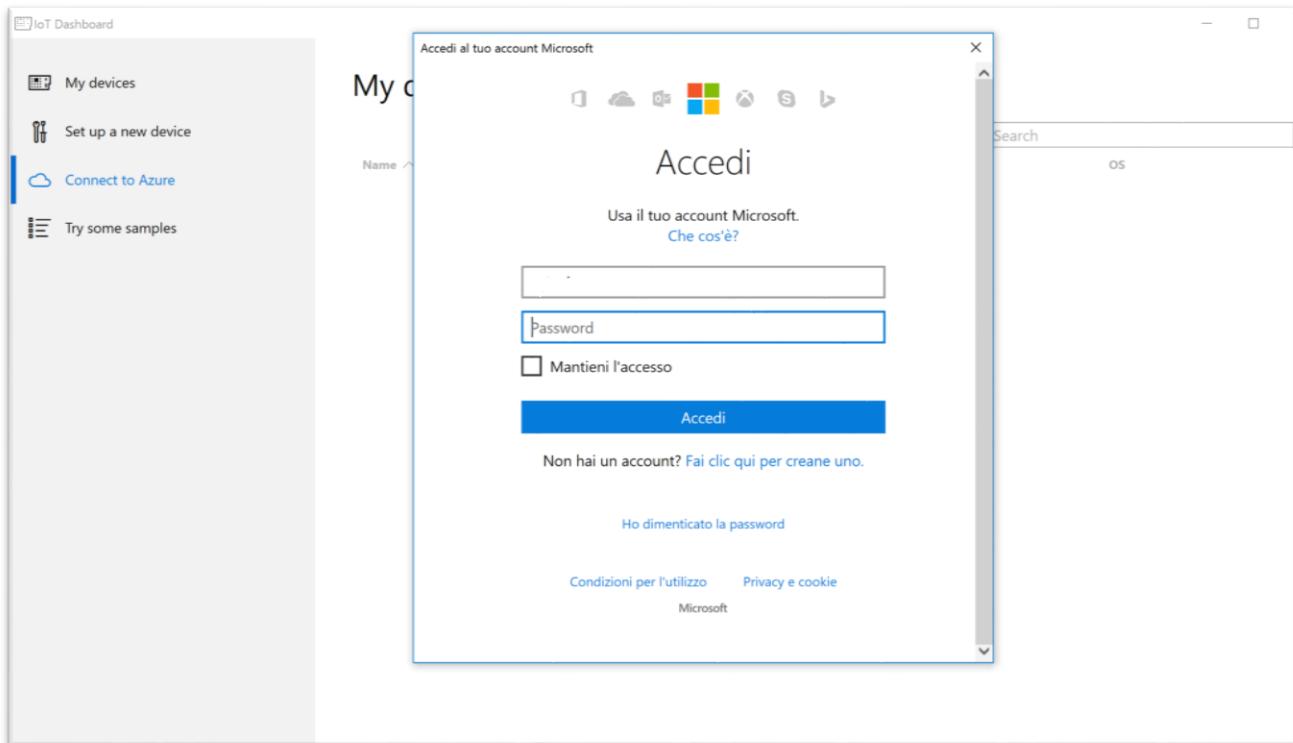
Per quanto riguarda il piano tariffario, se nella vostra sottoscrizione non sono presenti altri IoT Hub allora potete anche selezionare il piano gratuito F1 Free (nell'immagine sopra è disabilitato in quanto è consentito avere un solo IoT Hub gratuito per ciascuna sottoscrizione). Ai fini di questo laboratorio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

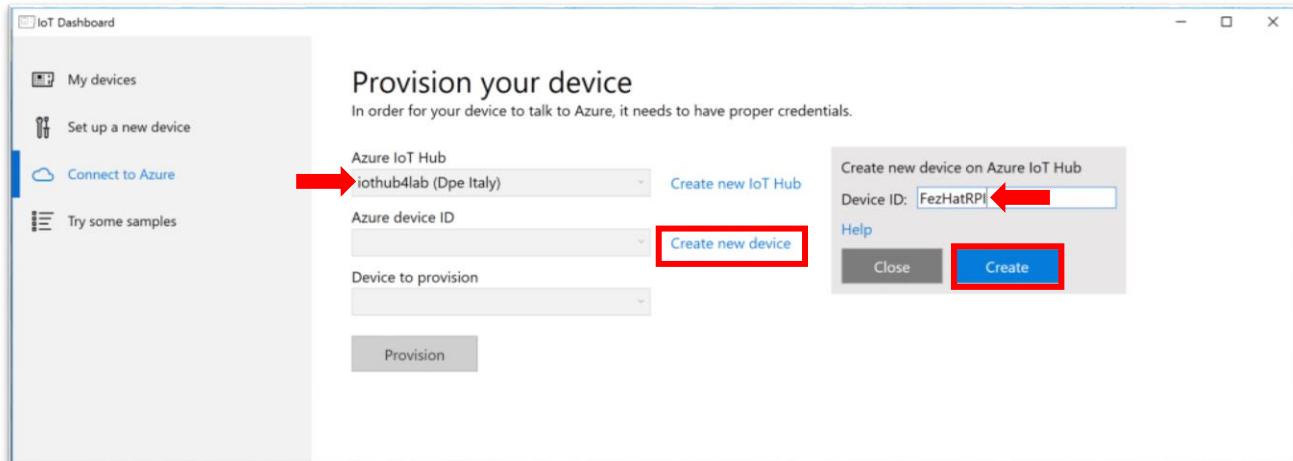
Assicuriamoci che venga completata la creazione dell'IoT Hub, poi possiamo tornare alla scheda.

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).

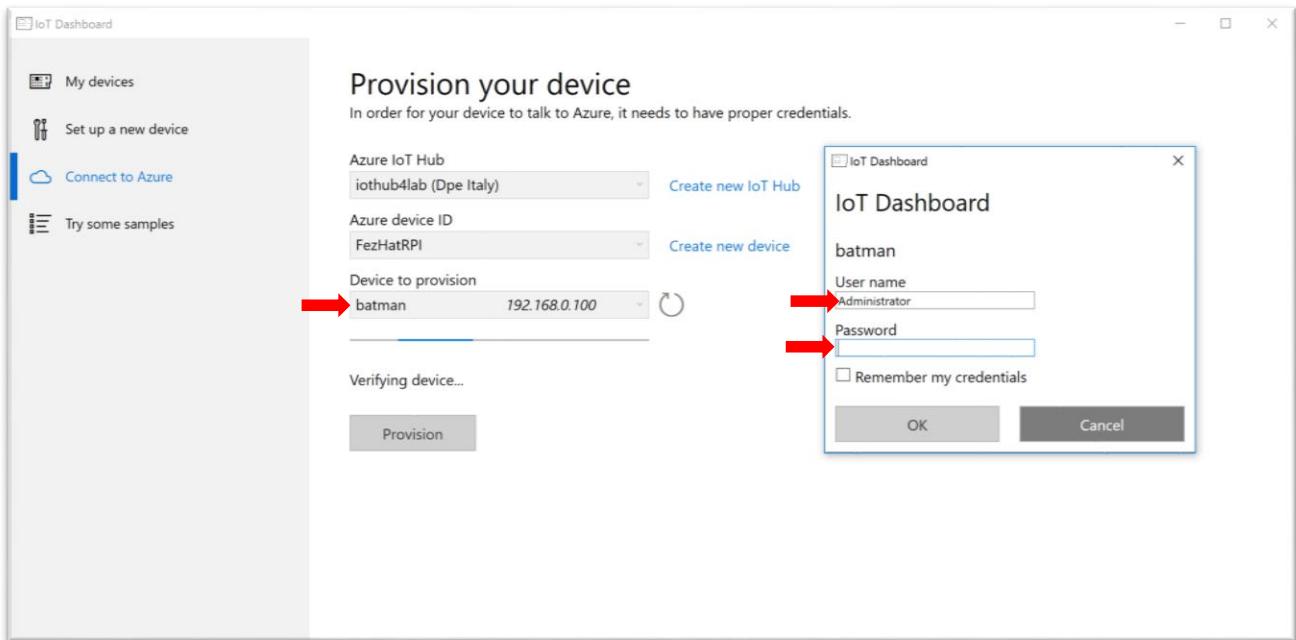


Una volta eseguito l'accesso, ci ritroveremo una schermata simile a quella che segue, in cui dovremo selezionare l'IoT Hub appena creato (cliccando *Create new IoT Hub* avremmo potuto anche crearlo da qui, ma lo abbiamo già fatto nella maniera tradizionale dal portale).

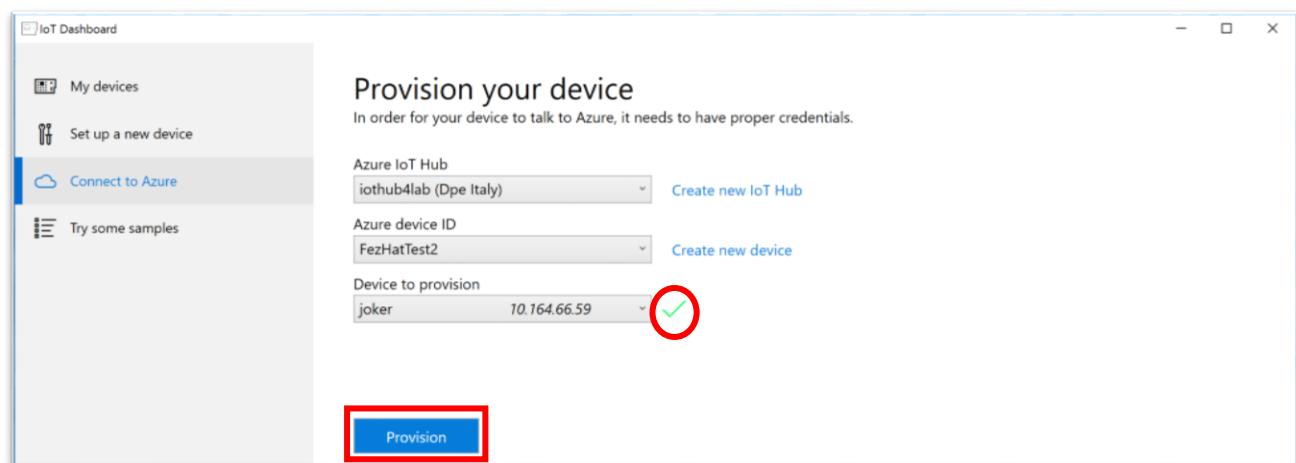
Dobbiamo invece creare un nuovo *device ID*, quindi clicchiamo su *Create New Device* ed inseriamo come id un nome a nostra scelta, in questo caso abbiamo scelto *FezHatRPI*.



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](#) (la password di default è *iotLab*).

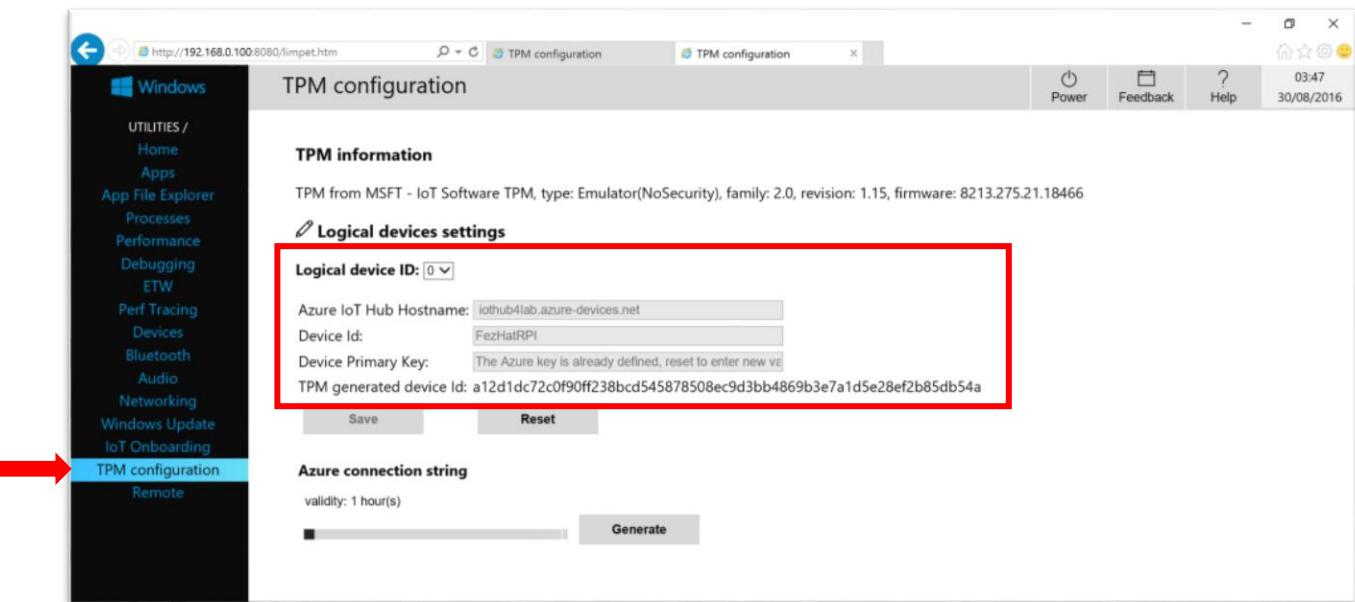
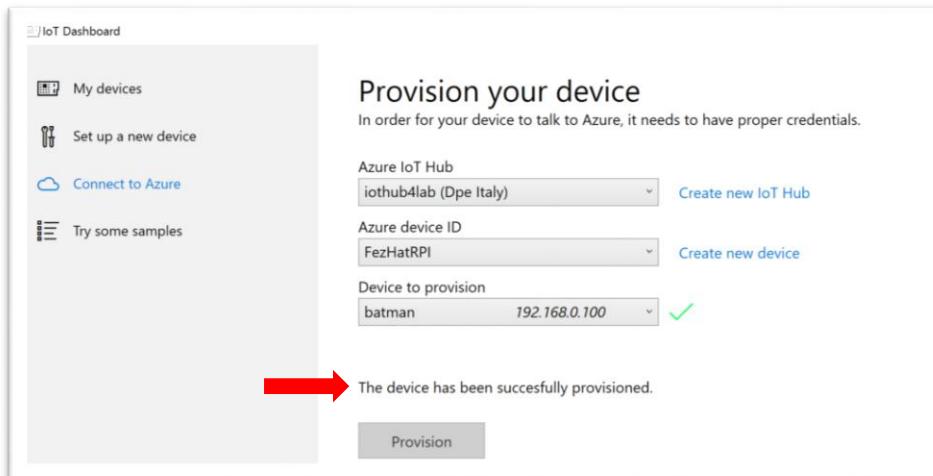


A questo punto ci ritroveremo una schermata come quella che segue e dovremo procedere al provisioning del device.



Dal momento che abbiamo già installato il TPM Software Emulator sulla RPI al punto 3c, dovremmo vedere una spunta verde di fianco al nome della nostra RPI. Possiamo quindi cliccare il tasto *Provision*. Questa operazione ci permette di registrare il dispositivo sul nostro IoT Hub e utilizzare le informazioni di connessione per generare una chiave che verrà memorizzata e gestita dal TPM.

Verifichiamo che il provisioning vada a buon fine e che nella sezione TPM Configuration della web page della RPI siano presenti le informazioni di connessione con il nostro IoT Hub, come mostrato nelle due immagini che seguono.



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

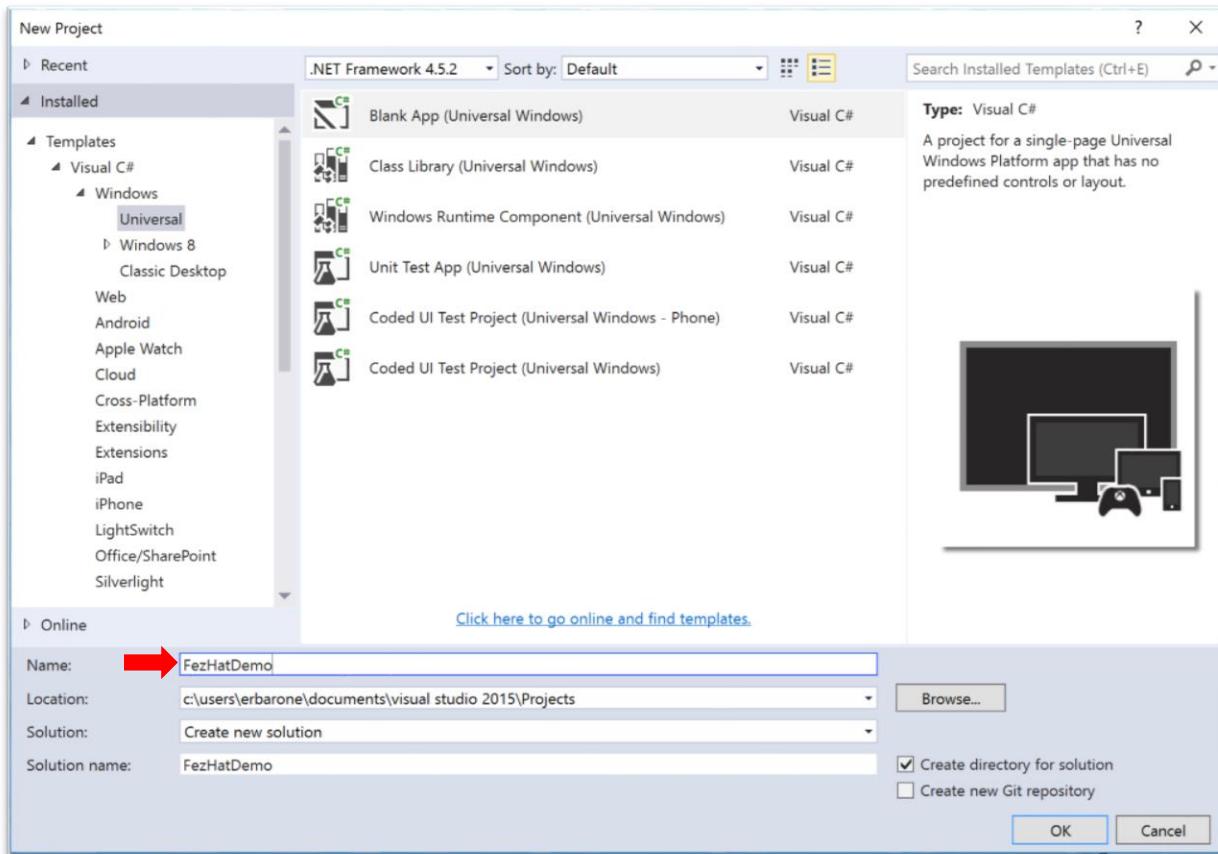
5 - L'applicazione UWP per recuperare e mostrare temperatura, luce, coordinate dell'accelerometro

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

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

Potete scaricarla e analizzarne il codice già pronto oppure se invece vogliamo realizzarla partendo da zero, ecco la procedura da seguire.

Apriamo Visual Studio 2015, assicurandoci di avere installato l'Update 3. Dobbiamo creare un nuovo progetto UWP, quindi selezioniamo *File > New > Project > Installed > Templates > Visual C# > Universal* e selezioniamo la *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*.

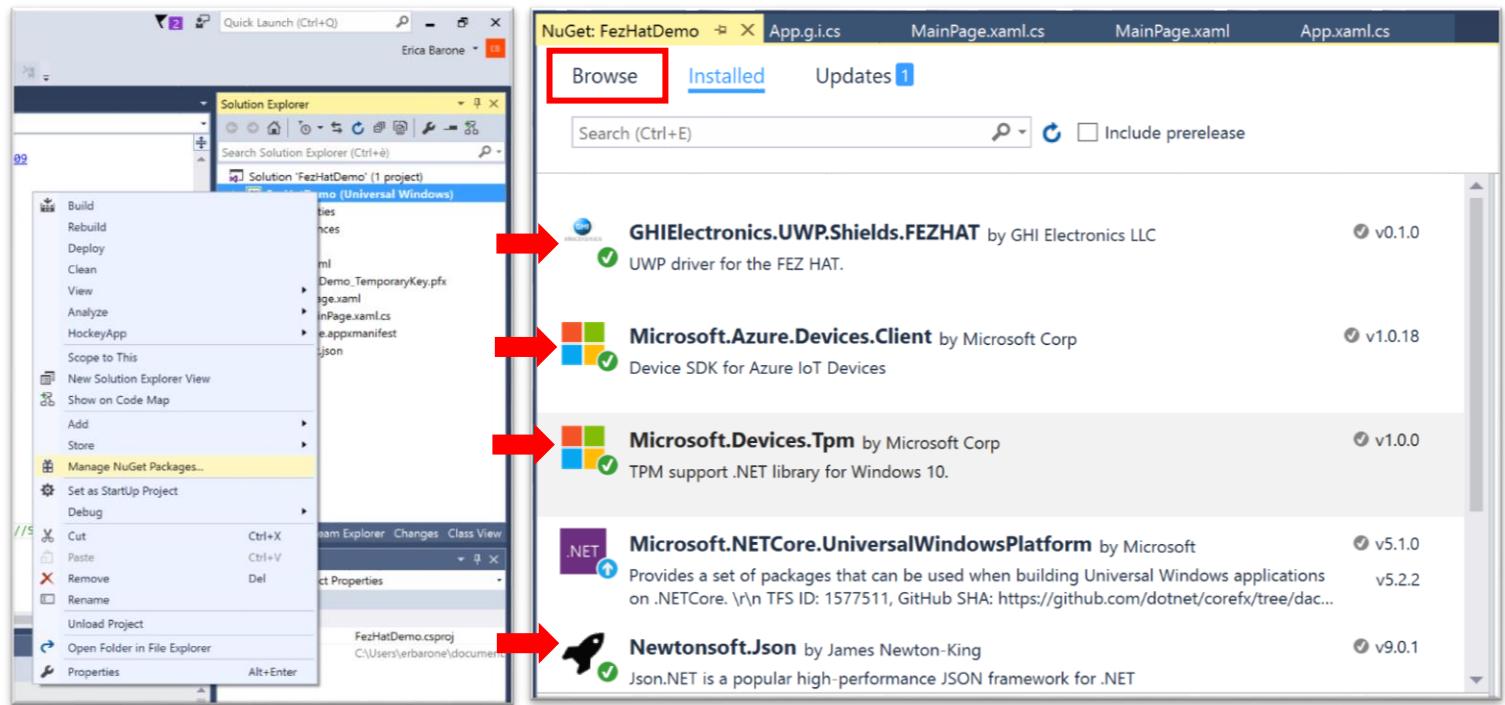
Cominciamo con l'installazione dei package NuGet. *Tasto destro sul progetto > Manage NuGet Packages*. Scarichiamo e installiamo i seguenti package, digitandoli nella sezione *Browse*:

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

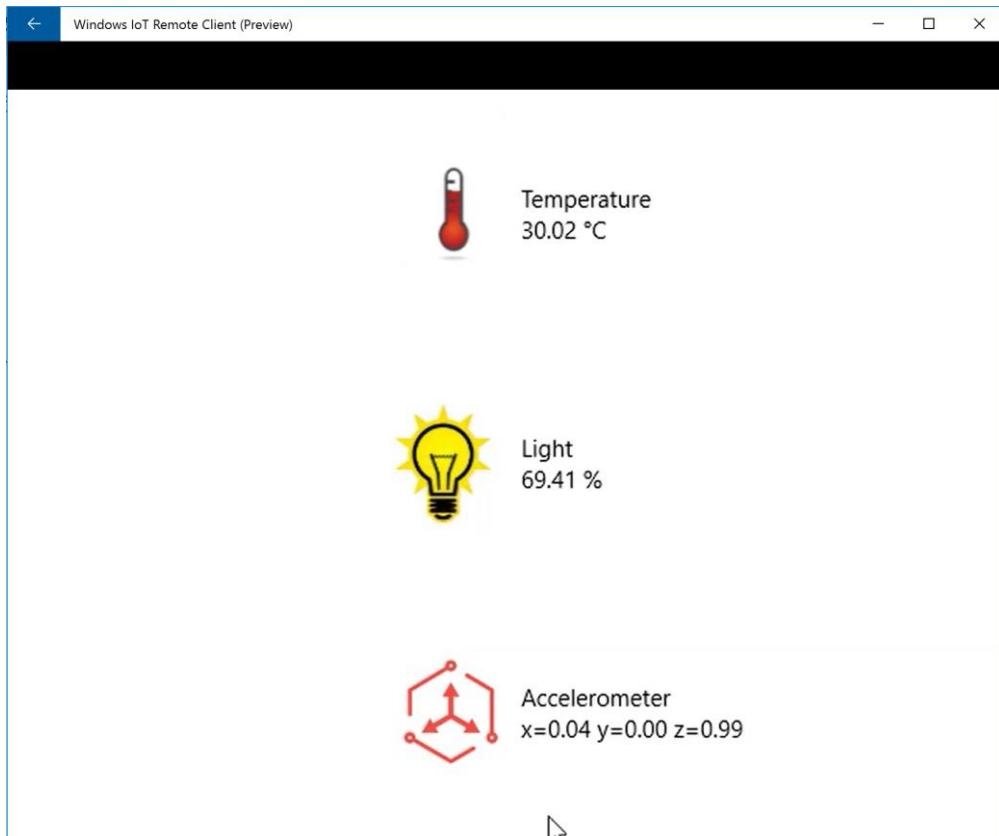
Microsoft.Devices.Tpm -> Per utilizzare il TPM configurato al punto precedente

GHIElectronics.UWP.Shaileds.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 vogliamo ottenere è questo, ma per capire come fare apriamo la *MainPage.xaml*.



Apriamo ora la *MainPage.xaml* per analizzare l'interfaccia grafica dell'applicazione.

```

<Page
    x:Class="FezHatDemo.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:FezHatDemo"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>

        <Image Grid.Column="0" Grid.Row="0" Source="Assets/temperature.jpg" Height="80" Width="80" HorizontalAlignment="Right"></Image>
        <StackPanel Grid.Column="1" Grid.Row="0" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10, 0, 0, 0">
            <TextBlock Text="Temperature"></TextBlock>
            <TextBlock x:Name="Temp" Text="not available"></TextBlock>
        </StackPanel>

        <Image Grid.Row="1" Grid.Column="0" Source="Assets/light.jpg" Height="80" Width="80" HorizontalAlignment="Right"></Image>
        <StackPanel Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10, 0, 0, 0">
            <TextBlock Text="Light"></TextBlock>
            <TextBlock x:Name="Light" Text="not available"></TextBlock>
        </StackPanel>

        <Image Grid.Row="2" Grid.Column="0" Source="Assets/accelerometer.png" Height="70" Width="70" HorizontalAlignment="Right"></Image>
        <StackPanel Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="10, 0, 0, 0">
            <TextBlock Text="Accelerometer"></TextBlock>
            <TextBlock x:Name="Accel" Text="not available"></TextBlock>
        </StackPanel>

    </Grid>
</Page>

```

In pratica dobbiamo utilizzare 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

Di seguito vediamo la *MainPage.xaml.cs*, potete copiare e incollare l'intero codice riportato qui sotto.

COSTRUTTORE

```
public sealed partial class MainPage : Page
{
    private static DeviceClient _deviceClient = null;
    private static FEZHAT _hat = null;
    private DispatcherTimer _timer = null;

    public MainPage()
    {
        this.InitializeComponent();

        //init device and shield
        TpmDevice mytpmdevice = new TpmDevice(0);
        _deviceClient =
DeviceClient.CreateFromConnectionString(mytpmdevice.GetConnectionString(86400)); //SAS Token 24h
initFezhat();
    }

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

TIMER TICK

```
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);
}
```

INIT FEZHAT

```
public async void initFezhat()
{
    _hat = await FEZHAT.CreateAsync();
}

}
```

Analizziamo ora i singoli pezzi di codice che compongono la *MainPage.xaml* (*riporto di seguito alcune porzioni di codice per spiegarle in modo più dettagliato, se avete copiato e incollato il codice della pagina precedente non occorre aggiungere altro*)

Cominciamo con il **costruttore**. 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. Usiamo infatti i metodi della classe *Microsoft.Devices.Tpm* per generare il SAS Token e gestire appunto la connessione tra il dispositivo e IoT Hub. Vi faccio notare che il SAS Token generato dal TPM ha una scadenza limitata nel tempo, impostata per questa demo a 24h (ovvero 86400 sec).

```
_deviceClient = DeviceClient.CreateFromConnectionString(mytpmdevice.GetConnectionString(86400)); //SAS Token 24h
```

Vi faccio notare che nelle applicazioni reali, che restano attive per tempi che vanno ben oltre le 24 ore, deve necessariamente essere gestito un refresh del token, indipendentemente dalla durata impostata in fase di costruzione.

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

Come potete vedere nel codice riportato qui sopra, recuperare i dati dai sensori utilizzando i metodi messi a disposizione dal package *GHIElectronics.UWP.Shields.FEZHAT* è davvero molto semplice, dal momento che esiste questa libreria per Windows 10 IoT Core che si interfaccia direttamente con la scheda.

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

Vi faccio notare che senza questa libreria sarebbe necessario analizzare il datasheet della scheda per capire come i dati vengono letti dai singoli sensori e quindi come i valori ottenuti possono essere utilizzati all'interno delle nostre applicazioni.

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, *SendEventAsync*, fornito dal package *Microsoft.Azure.Devices.Client*. Anche in questo caso, le librerie che abbiamo installato facilitano di molto la serializzazione del messaggio (*Newtonsoft.Json*) e l'invio al nostro IoT Hub (*Microsoft.Azure.Devices.Client*).

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

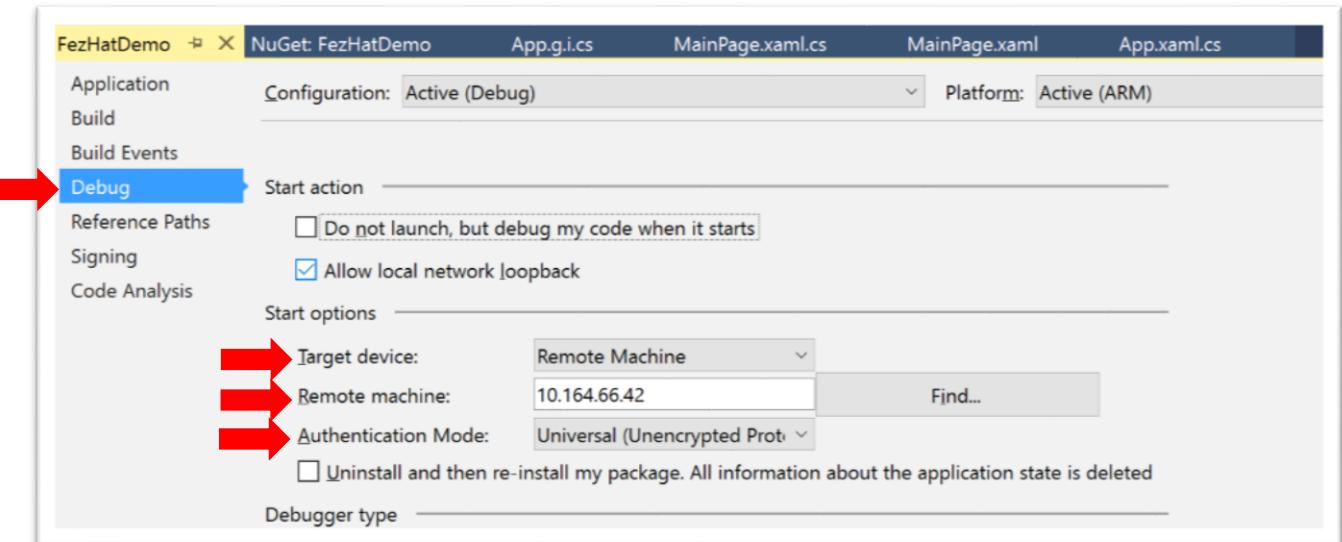
var messageString = JsonConvert.SerializeObject(message);
var messageToSend = new Message(Encoding.UTF8.GetBytes(messageString));

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

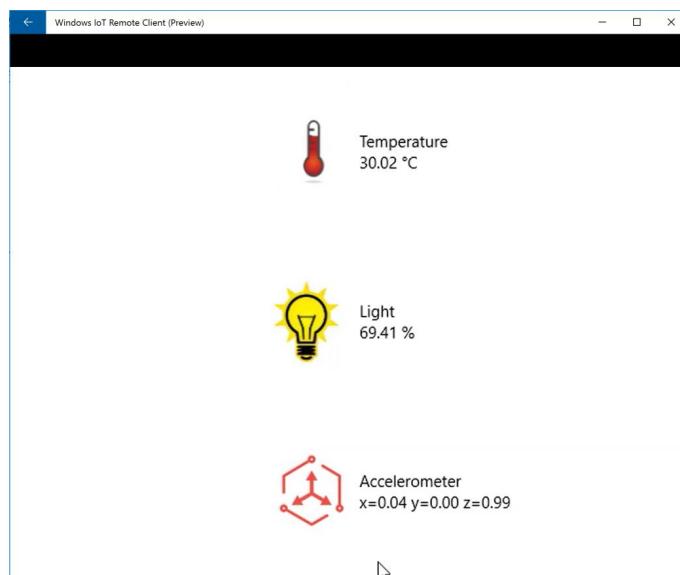
Possiamo ovviamente modificare il deviceId all'interno del codice, in modo che corrisponda al nome che abbiamo assegnato al [punto 2](#).

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

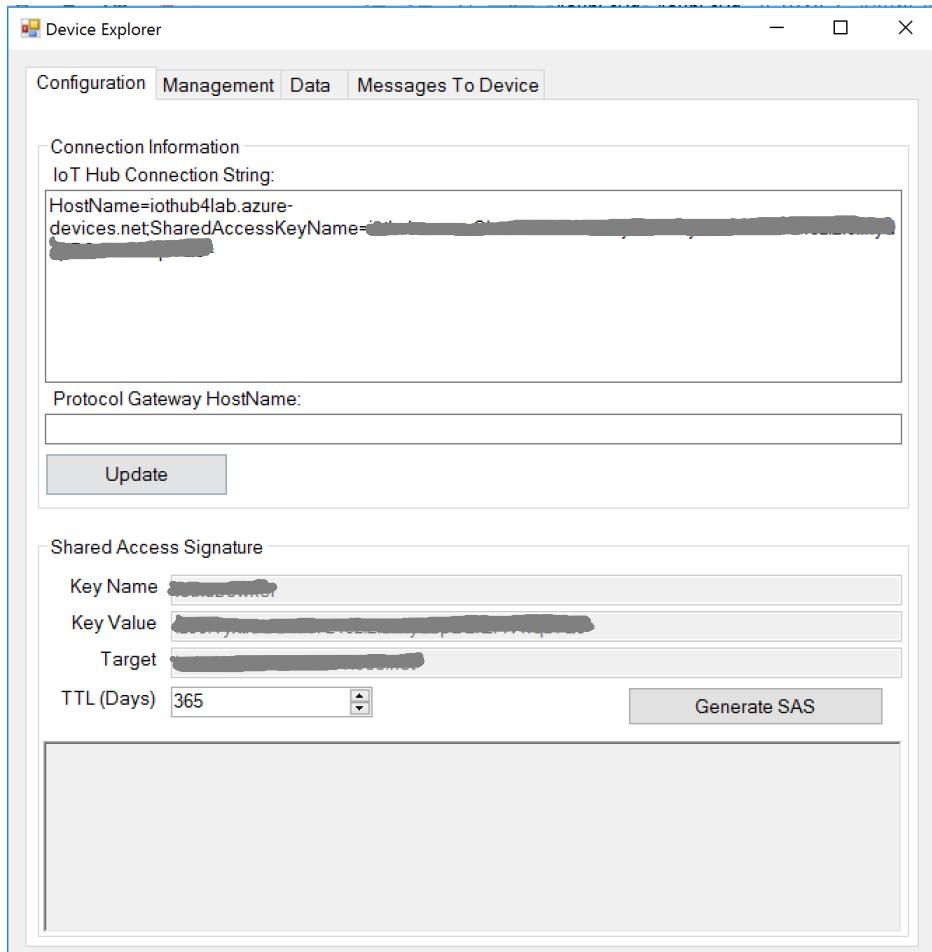
Per verificare che i dati raccolti dai sensori arrivino effettivamente al nostro IoT Hub, apriamo la dashboard di Azure: <http://portal.azure.com> e selezioniamo il resource group creato al [punto 4](#), poi il nostro IoT Hub.

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

The screenshot shows the Azure portal interface with the 'IoT Hub' blade open. On the left, the navigation menu includes 'Tutte le risorse', 'Gruppi di risorse', 'Servizi app', 'Database SQL', 'DocumentDB (NoSQL)', 'Macchine virtuali', 'Servizi di bilanciamento...', 'Account di archiviazione', 'Reti virtuali', 'Log attività', 'Centro sicurezza', 'Fatturazione', 'Guida e supporto', and 'Altri servizi'. The main area displays the 'iothub4lab' IoT Hub details, including its name, ID, creation date (31/05/2016), location (Europa occidentale), and properties like 'Nome host' (iothub4lab.azure-devices.net), 'Plano tariffario e livello di scalabilità' (S1 - Standard), and 'Unità di IoT Hub' (1). A large red box highlights the 'Utilizzo' section, which shows a circular icon with '119 / 400k' and 'DISPOSITIVI 1'. Below this, a line chart titled 'Monitoraggio' shows the number of connected devices over time, with values ranging from 0 to 50. At the bottom, there are sections for 'TELEMETRY MESSAGES SENT' (0) and 'CONNECTED DEVICES' (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.

The screenshot shows the Azure portal interface. On the left, a sidebar lists 'iothub4lab - Criteri di accesso condivisi' under 'IoT Hub'. The main area displays a table of shared access policies:

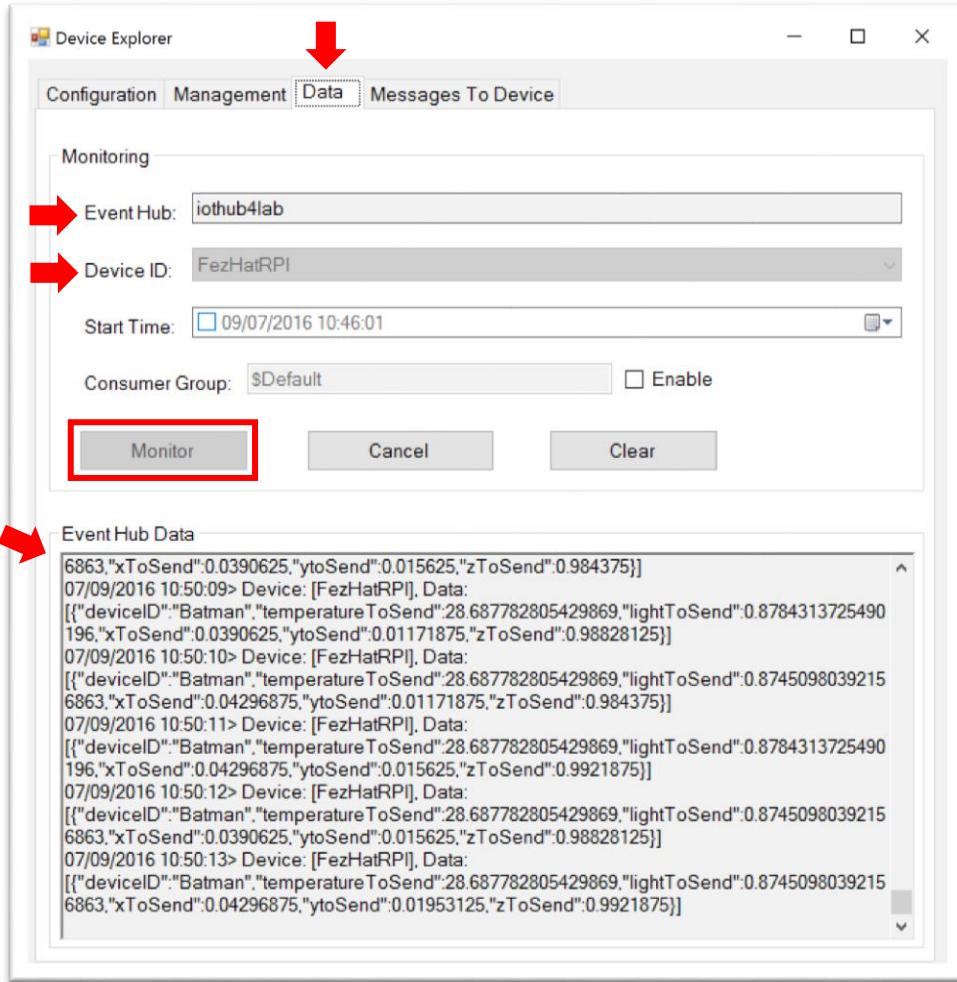
CRITERIO	AUTORIZZAZIONI
iothubowner	scrittura registro, connessione servizio, connes...
service	connessione servizio
device	connessione dispositivo
registryRead	lettura registro
registryReadWrite	scrittura registro

To the right, a detailed view of the 'iothubowner' policy is shown:

- Nome criterio di accesso:** iothubowner
- Autorizzazioni:** (checkboxes checked)
 - Lettura registro
 - Scrittura registro
 - Connessione servizio
 - Connessione dispositivo
- Chiavi di accesso condivise:**
 - Chiave primaria: (redacted)
 - Chiave secondaria: (redacted)
 - Stringa di connessione - chiave primaria:** (redacted) REDACTED
 - Stringa di connessione - chiave secondaria: (redacted)

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.



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 various service categories like Resource groups, All resources, Recent, App Services, etc. A red box highlights the 'New' button. In the center, under 'Data + Storage', a red box highlights the 'Data + Storage' category. On the right, a detailed configuration dialog for creating a storage account is shown. It includes fields for Name (labiotdatastorage), Deployment model (Resource manager), Account kind (General purpose), Performance (Standard), Replication (Read-access geo-redundant storage), Subscription (Dpe Italy), Resource group (radio button set to 'Use existing' with LabIoT selected), Location (West Europe), and a 'Create' button at the bottom.

Assegnamo 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 creating resources. Under "MARKETPLACE", the "Internet of Things" category is selected (red box), and under "FEATURED APPS", the "Stream Analytics job" item is selected (red box).
- Right Window:** Shows the "New Stream Analytics J..." configuration dialog. It includes fields for "Job name" (labeled with a red arrow), "Subscription" (Dpe Italy), "Resource group" (LabiloT, with a red box around the "Use existing" radio button), "Location" (West Europe), and a "Create" button (boxed in red).

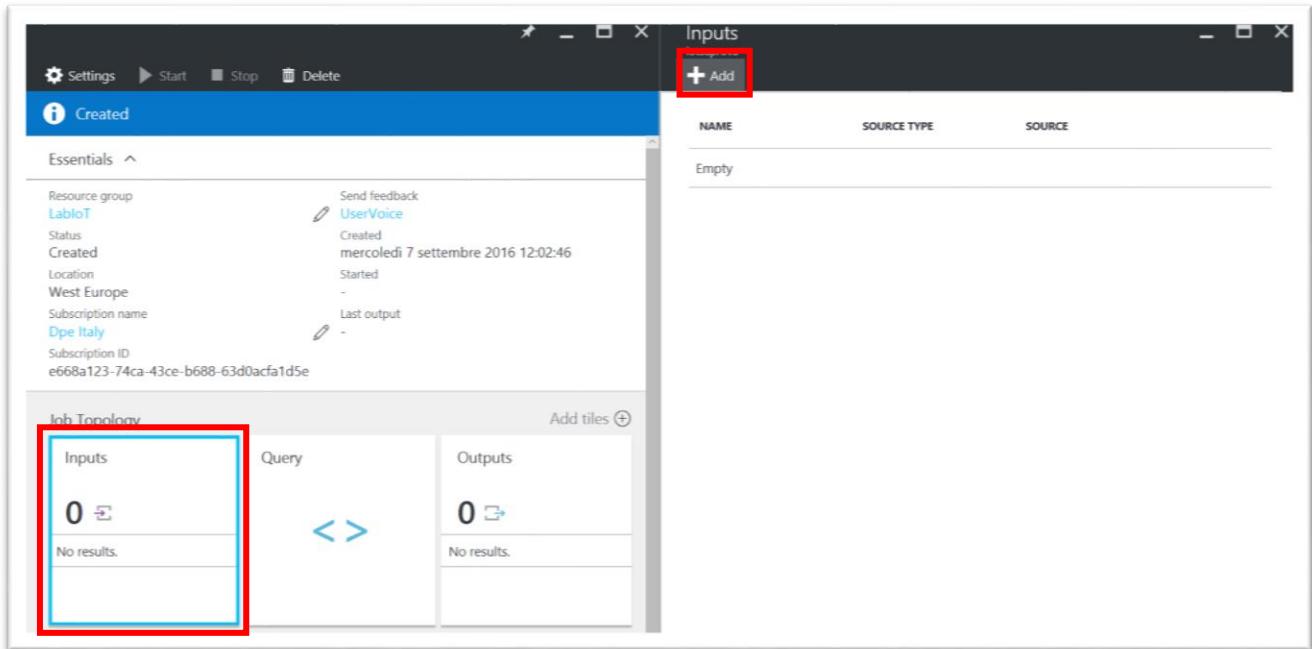
Di nuovo, facciamo attenzione a selezionare *Use existing* e inserire lo stesso *Resource Group* in cui abbiamo inserito l'*IoT Hub* creato al [punto 4](#), come mostrato in figura qui sopra.

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.



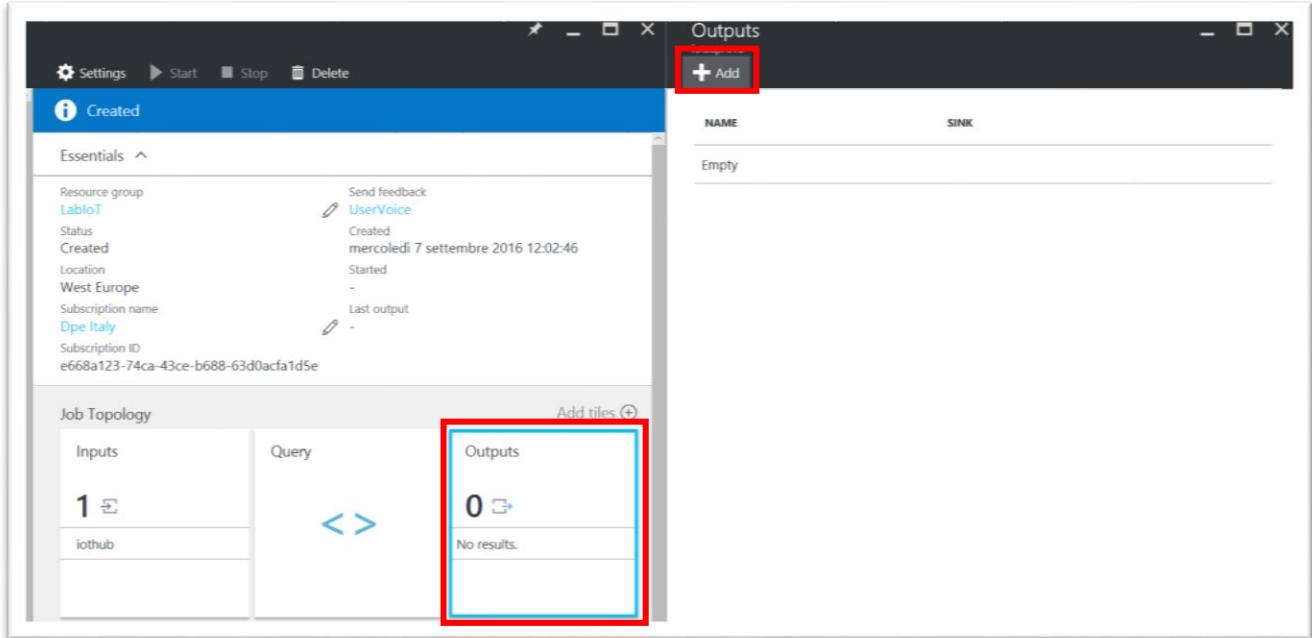
Inseriamo i seguenti parametri e clicchiamo infine sul tasto *Create* in basso.

A screenshot of the 'Add Input' configuration dialog. The 'Input alias' field contains 'iothub' and has a red arrow pointing to it. The 'Source Type' dropdown is set to 'Data stream' and has a red arrow pointing to it. Other fields include 'Subscription' (Use IoT hub from current subscription), 'IoT hub' (iothub4lab), 'Endpoint' (Messaging), 'Shared access policy name' (iothubowner), 'Shared access policy key' (redacted), 'Consumer group' (\$Default), 'Event serialization format' (JSON), and 'Encoding' (UTF-8). At the bottom is a 'Create' button, which is highlighted with a red box.

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*.



The 'New output' dialog shows the following configuration:

- * Output alias: toBlobStorage
- * Sink: Blob storage
- * Subscription: Use blob storage from current subscription
- * Storage account: labiotdatastorage1
- * Container: Create a new container
- * Container: blobcontainer
- Path pattern: (empty)
- Date format: YYYY/MM/DD
- Time format: HH
- * Event serialization format: JSON
- Encoding: UTF-8
- Format: Line separated

A red box highlights the 'Outputs' section in the main Stream Analytics interface, and a red arrow points to the 'Create' button at the bottom of the 'New output' dialog.

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 INTO ToBlobStorage
3 FROM iothub

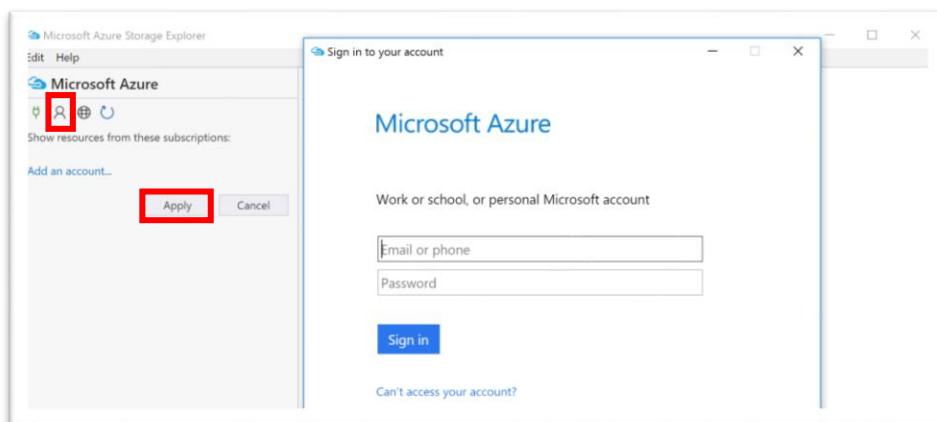
```

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

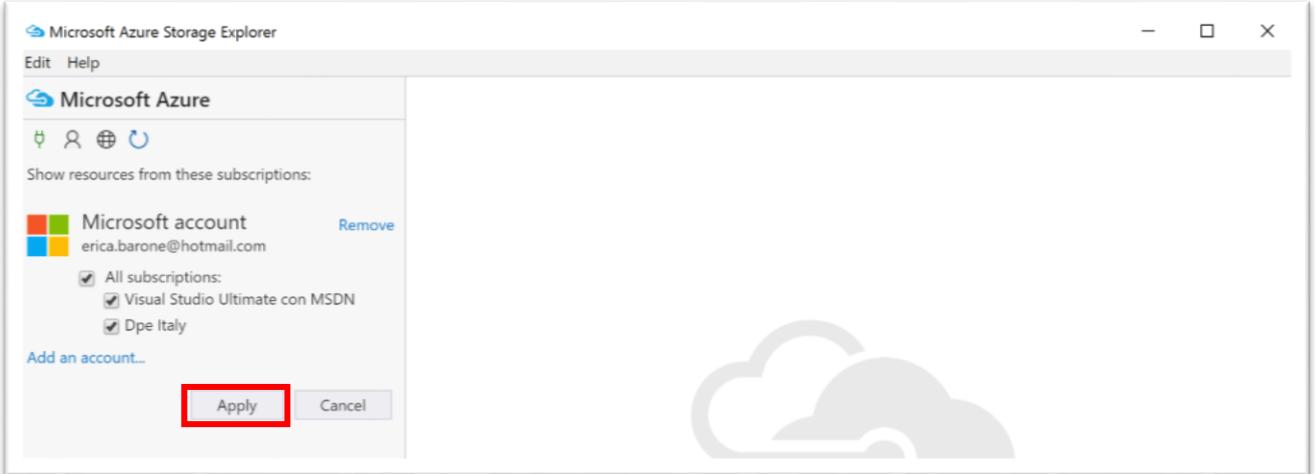


7c - Visualizzazione dei dati memorizzati

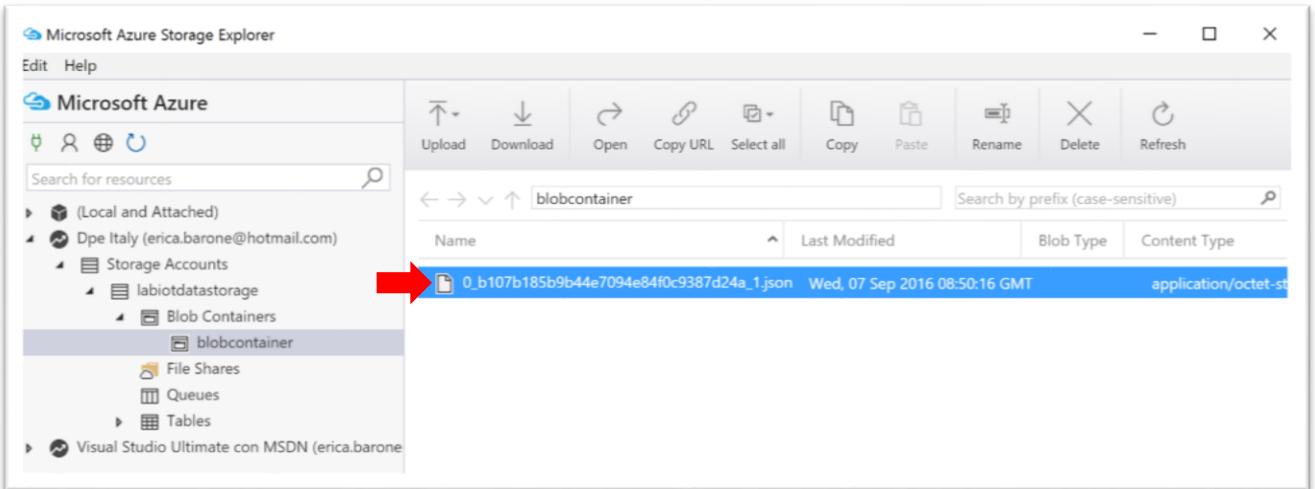
Verifichiamo ora che i dati vengano effettivamente memorizzati all'interno del nostro Blob. Per farlo, utilizziamo l'applicazione [Azure Storage Explorer](#). Una volta scaricato il tool, la prima cosa da fare è collegarlo al nostro account Azure, selezionando l'icona *account* > *Add an account* e si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



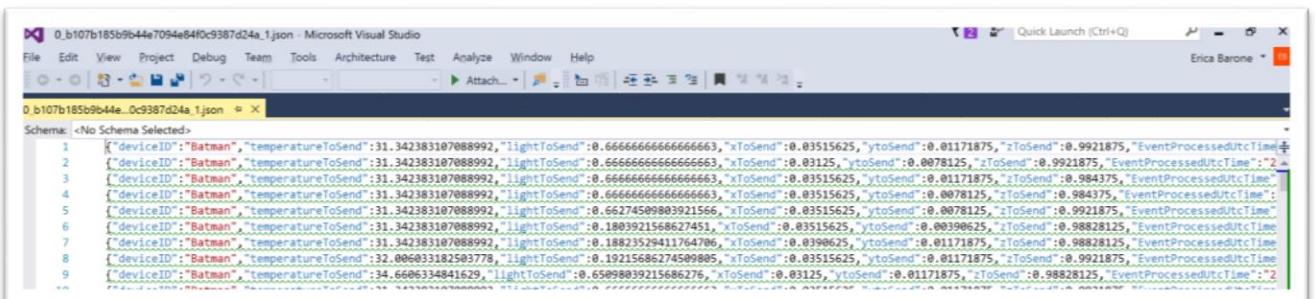
Una volta inserite le credenziali, comparirà il nostro account con tutte le sottoscrizioni ad esso associate (nel mio caso sono due, ma per voi dovrebbe essere una sola). Possiamo selezionare quale visualizzare e quale no, infine clicchiamo su *Apply*.



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.

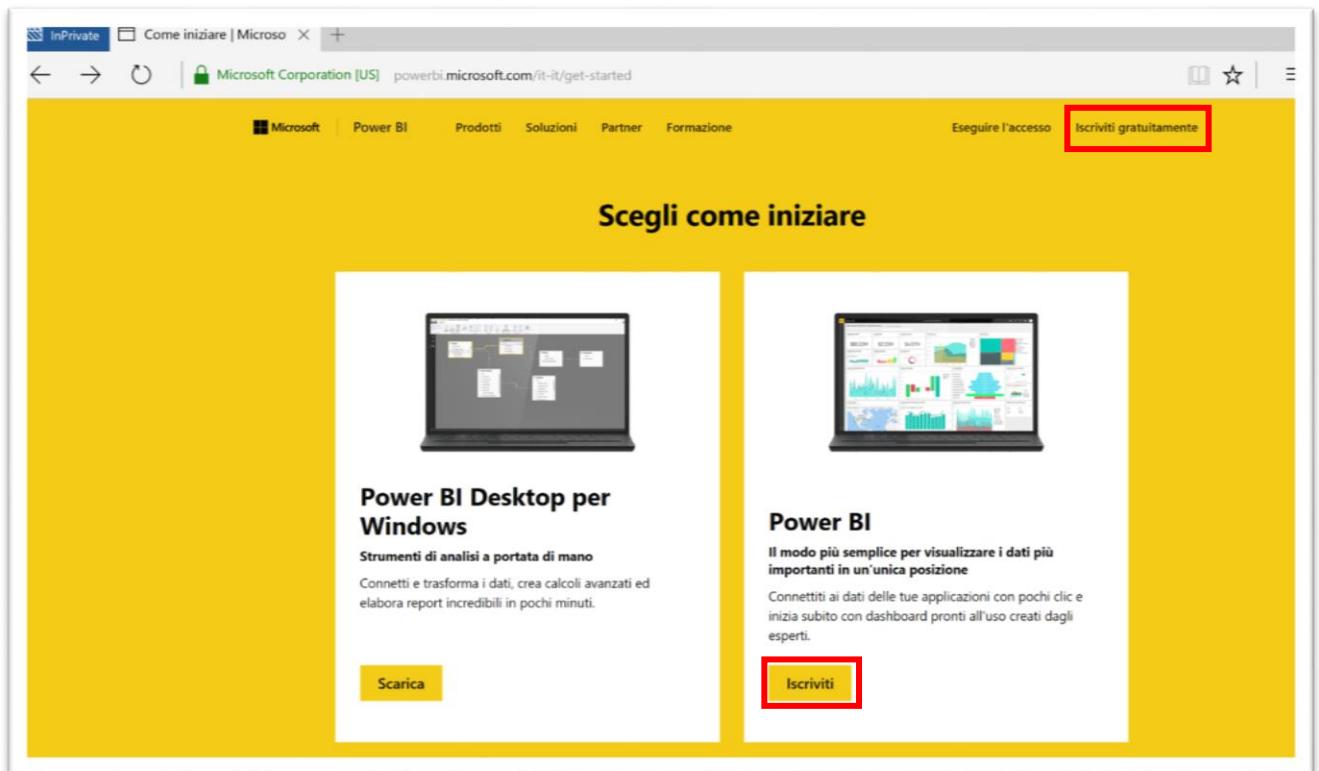


Vi faccio presente che, perchè il blob venga creato, l'applicazione deve girare sulla RPI, inviando i dati ad IoT Hub, in modo che il job dello Stream Analytics (attivato al punto precedente) sia in grado di prenderli in ingresso e inviarli allo storage. Se anche solo uno di questi passaggi non funziona,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

The screenshot shows the Microsoft Office 365 Enterprise E3 trial landing page. At the bottom left, there is a large button labeled "Free trial" with a red arrow pointing to it. The page features a banner for Office 365 Enterprise E3, highlighting its features like integrated collaboration services and advanced compliance features. It also mentions the inclusion of new Office 2016 apps for PC and Mac. A price of £14.70 user/month annual commitment is displayed, along with a "Buy now" button. The background shows a group of people working together in an office environment.

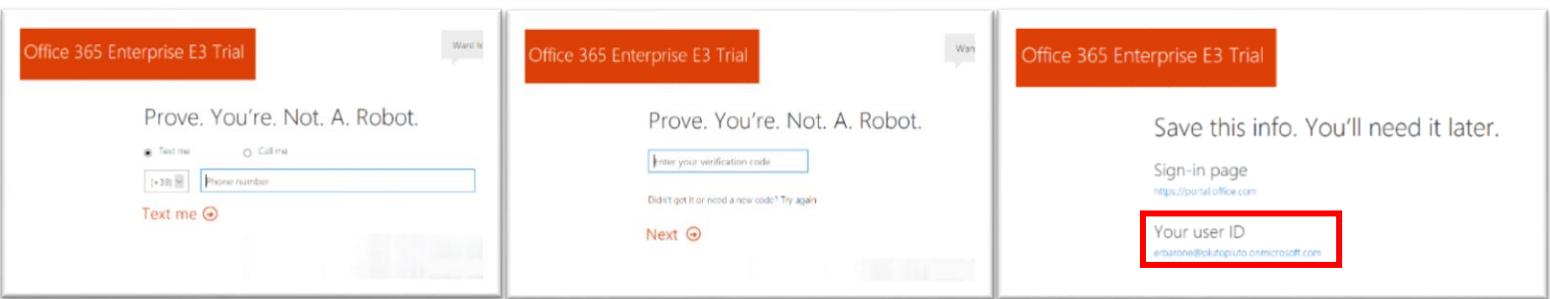
A questo punto ci viene chiesto di inserire alcuni dati per la registrazione. Compiliamo la prima form con i nostri dati, e poi in quella successiva inseriamo quello che sarà il nostro account aziendale, da usare poi per accedere a PowerBI.

The image contains two side-by-side screenshots of the Microsoft Office 365 Enterprise E3 trial registration process.

Left Screenshot: Shows the initial registration form titled "Office 365 Enterprise E3 Trial". It includes fields for First name, Last name, Business email address, Business phone number, Company name, and Your organization size. A red box highlights the "Your organization size" dropdown menu, which is set to "Iceland". A note says "This can't be changed after sign-up. Why not?". Below the form is a button labeled "Just one more step" with a red circle containing a question mark.

Right Screenshot: Shows the "Create your user ID" form. It asks for a User name, which is pre-filled with "username@yourcompany.onmicrosoft.com" and has a red box around it. There are also fields for Create password and Confirm your password. Below these fields is a note: "By clicking [Create my account](#), you agree to our terms and conditions and default communication preferences." Further down, there are sections for communication preferences (Email, Phone) and Microsoft Online Services contact information. At the bottom is a button labeled "Create my account" with a red circle containing a question mark.

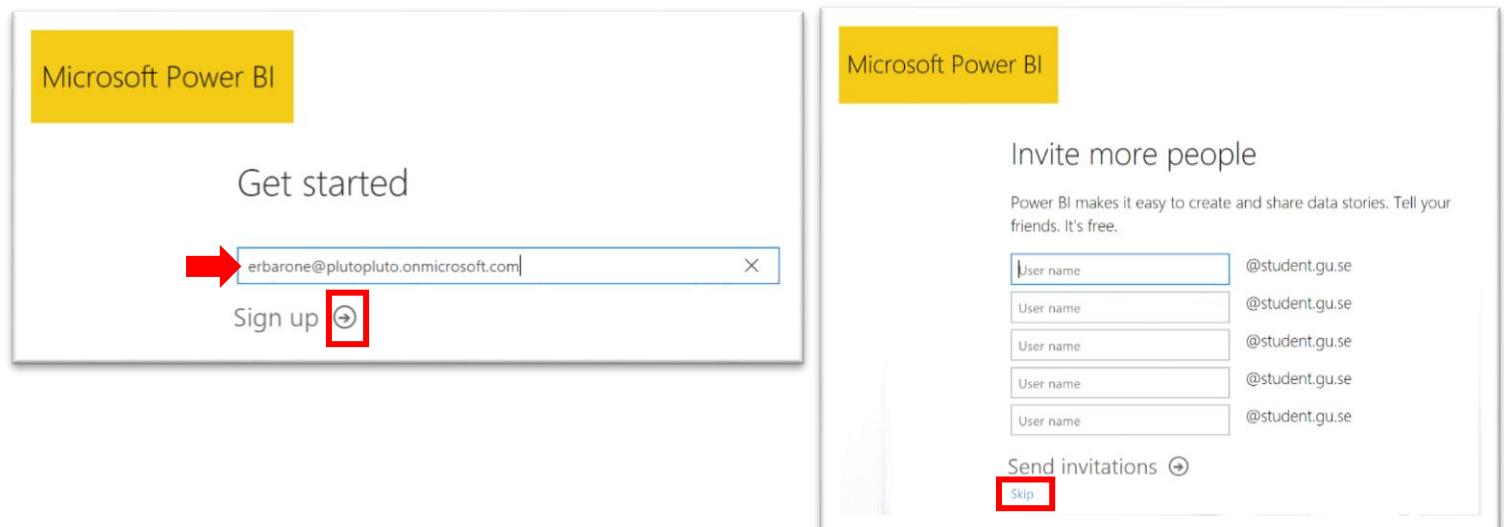
Verrà chiesta una verifica per confermare l'account, procediamo seguendo la procedura guidata.



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*.

ADD AN OUTPUT

Add an output to your job

- SQL Database ?
- Blob storage ?
- Event Hub ?
- Power BI ?
- Table storage ?
- Service Bus Queue ?
- Service Bus Topic ?
- DocumentDB ?
- Data Lake Store PREVIEW ?

Missing an output sink? [Let us know!](#) (Powered by [UserVoice](#) - [Privacy Policy](#))

ADD A MICROSOFT POWER BI OUTPUT

Authorize Connection

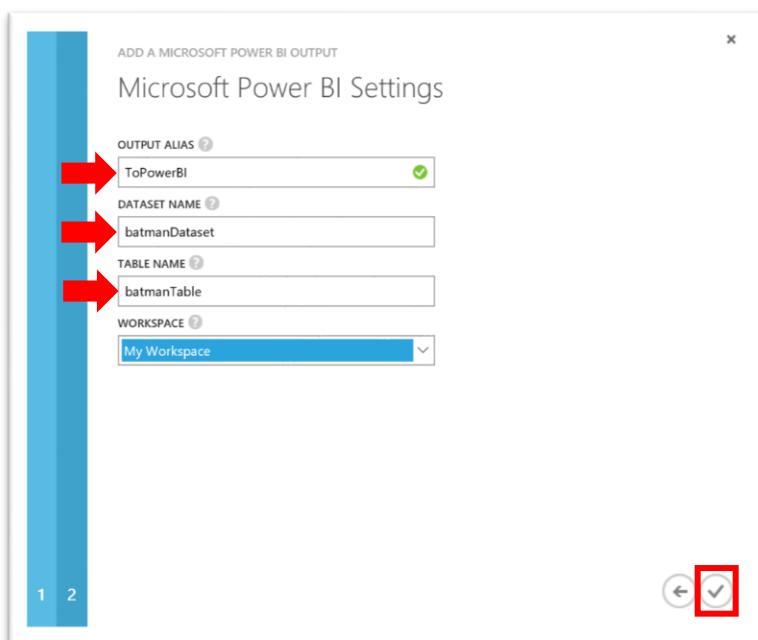
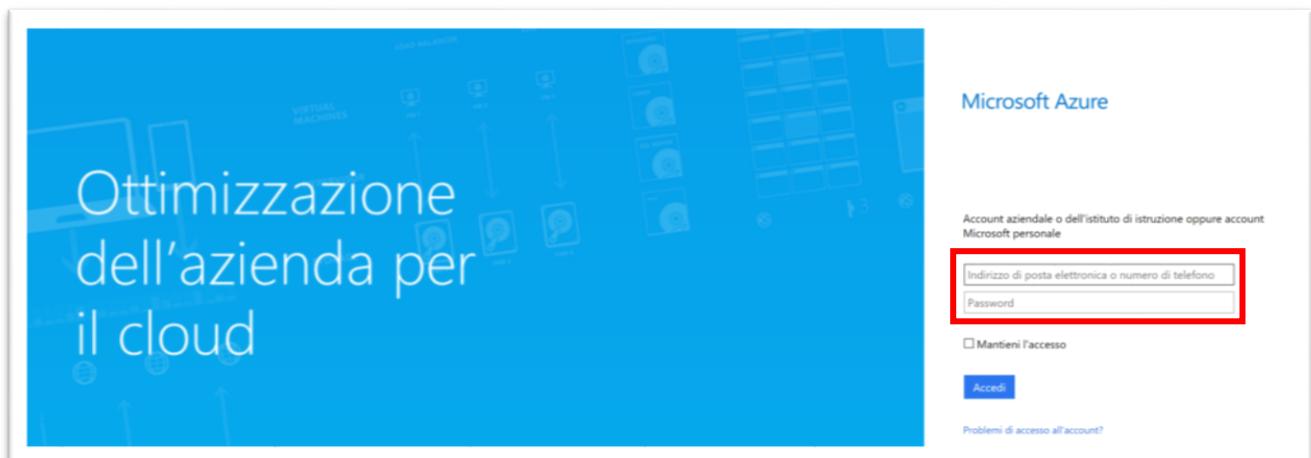
Existing Microsoft Power BI User
Authorize Stream Analytics to access your organizational Microsoft Power BI subscription to [create a live dashboard](#).

Authorize Now

Note: You are granting this output permanent access to your Power BI dashboard. Should you need to revoke this access in the future you can do one of the following:

1. Change the user account password.
2. Delete this output.
3. Delete this job.

New User
Don't have a Microsoft Power BI account yet? [Sign up now!](#)



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.

NAME	TYPE	LOCATION
iothub4lab	IoT Hub	West Europe
labiotsadatastorage	Storage account	West Europe
labiotsa	Stream Analytic...	West Europe

Inputs	Query	Outputs
1 iothub	<>	2 ToBlobStorage ToPowerBI

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

```

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.

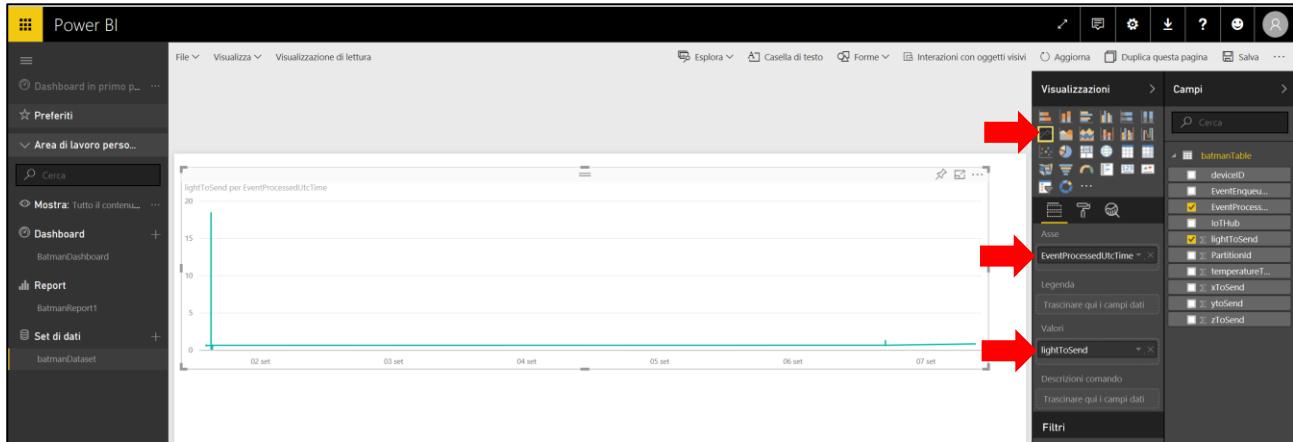
8d – La Dashboard su PowerBI

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*).

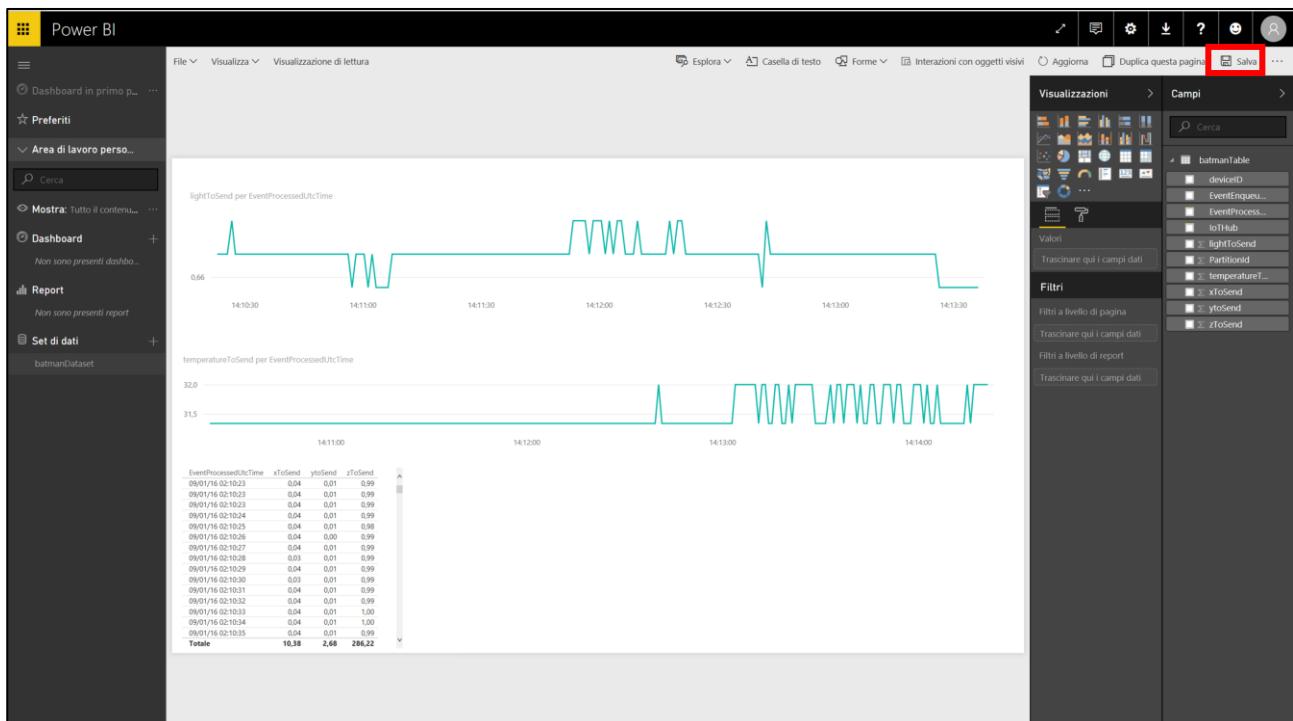
Ora dobbiamo creare un report con alcuni grafici, e in seguito la dashboard.

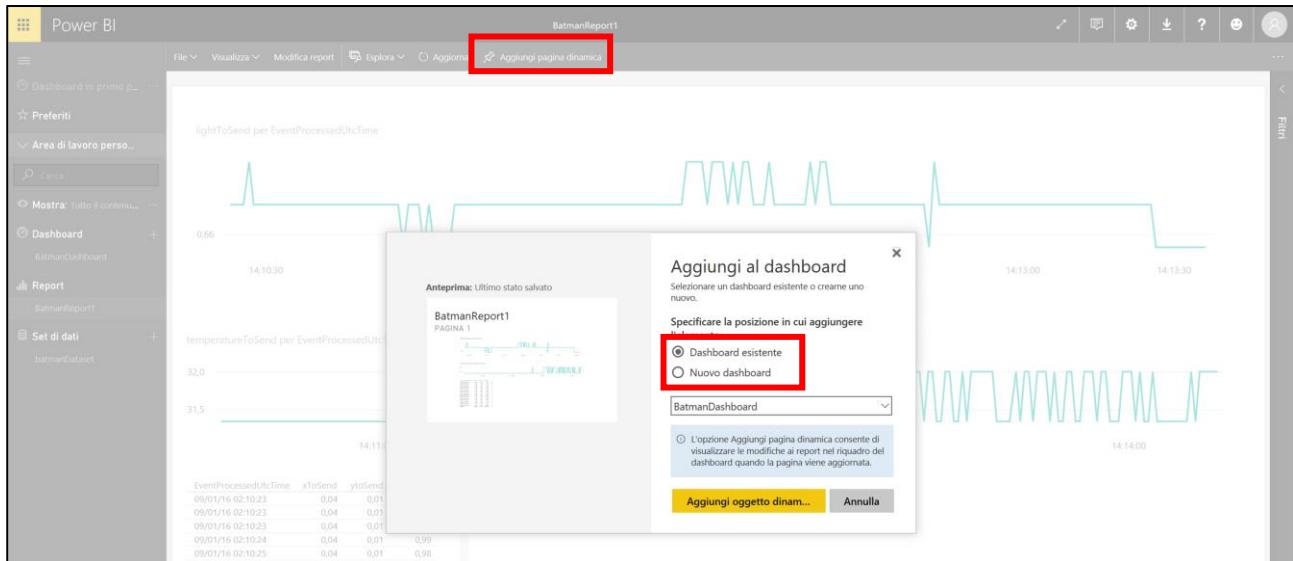
Selezioniamo quindi il tipo di grafico e inseriamo come *Valori* il *lightToSend* (che corrisponde al valore di luce inviato dalla RPI), come *Asse* inseriamo invece *EventProcessedUtcTime* (che identifica l'istante in cui il dato è stato processato dallo Stream Analytics). Questo è il risultato che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



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

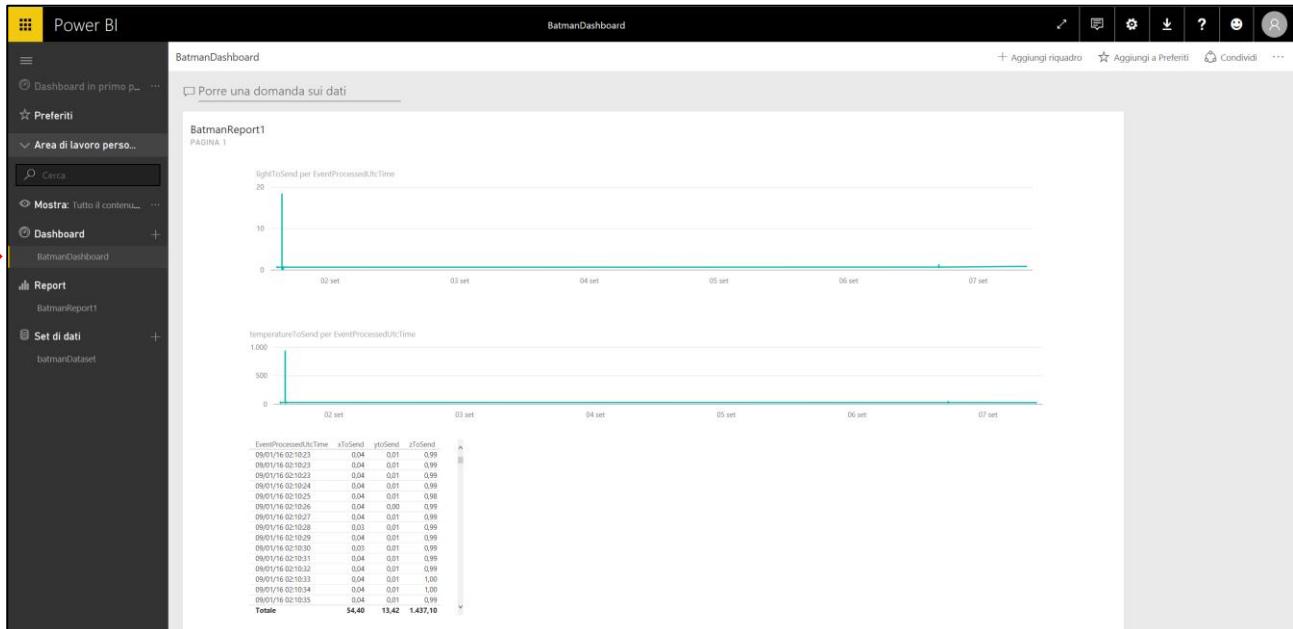
Una volta completato il nostro report, lo salviamo e lo aggiungiamo alla dashboard, cliccando su *Aggiungi pagina dinamica*.





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 😊