

# Special Project - ASE IoT

---

COME UTILIZZARE UN APPLICATION KIT INFINEON TC2X7 PER L'IOT  
STABILENDO UNA CONNESSIONE CON L'ASSISTENTE VOCALE ALEXA

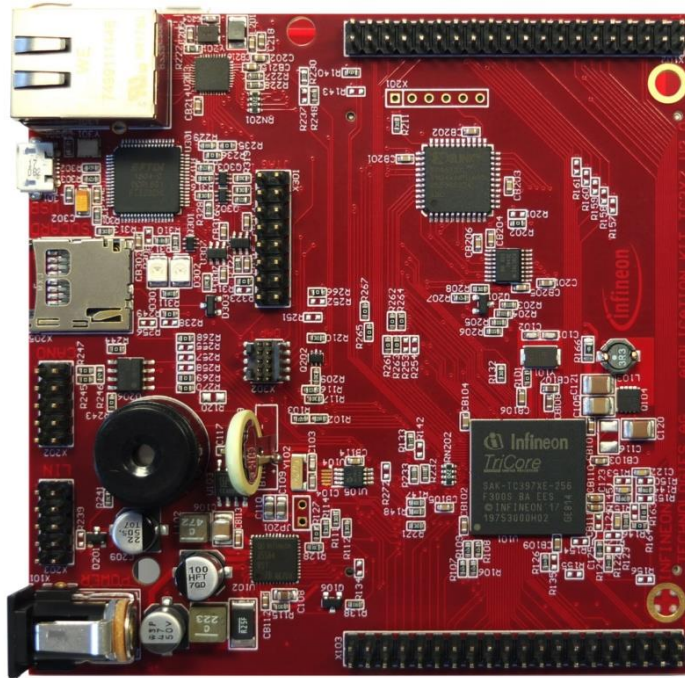
Alessandro Ciullo

s310023

# STRUMENTAZIONE

---

Application Kit  
Infineon TC2x7



Sensore di Temperatura  
DS18B20



# Ambiente di sviluppo

---



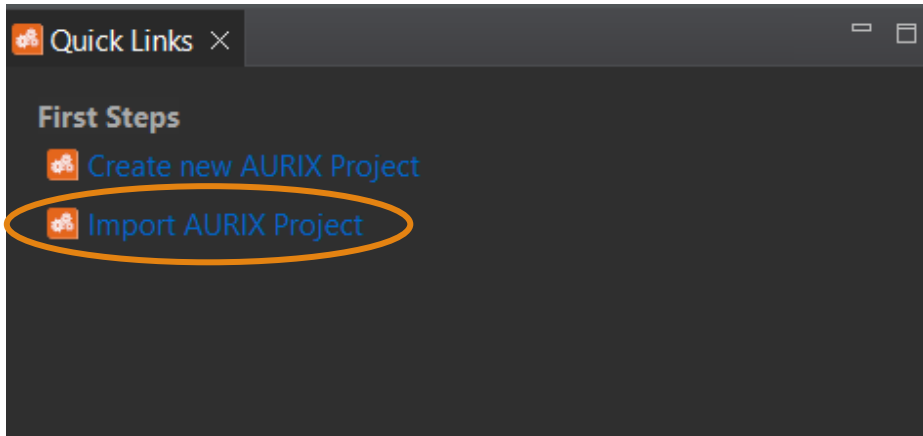
- AURIX Development Studio

<https://www.infineon.com/cms/en/product/promopages/aurix-development-studio/>



- Visual Studio Code

<https://code.visualstudio.com/>

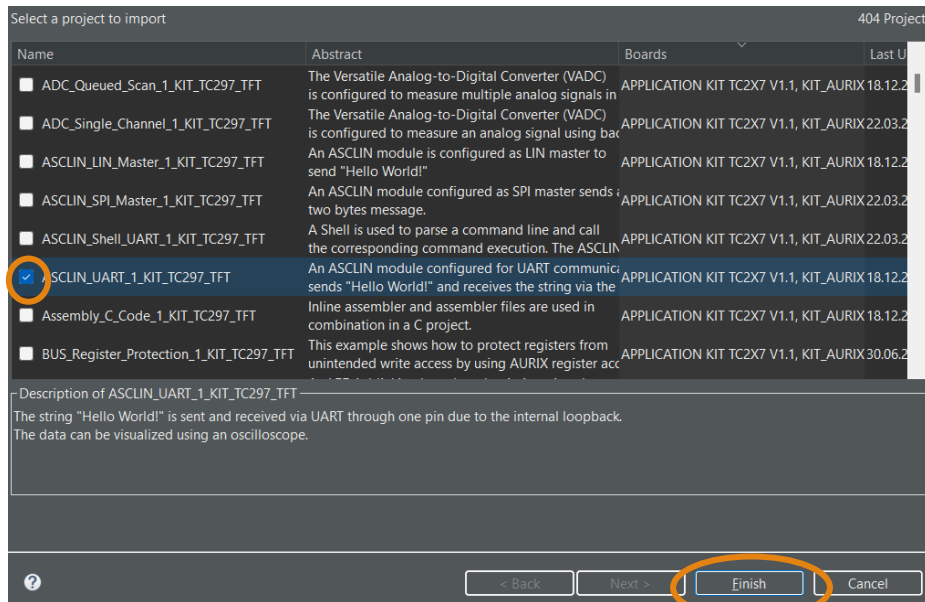


# Ambiente di sviluppo: AURIX Development Studio

AURIX Development Studio è un IDE basato su Eclipse, molto utile per programmare board Infineon Aurix. Al suo interno è già integrata la possibilità di importare le demo open source offerte da infineon, premendo sul comando “Import AURIX Project” mostrato nella figura in alto.

Una volta aperto il catalogo, ci basterà selezionare la demo più adatta ai nostri fini e premere il pulsante finish per completare l’importazione.

E’ possibile importare progetti anche dal nostro filesystem locale facendo click su **File → Open Projects from File System** e selezionando la cartella desiderata.

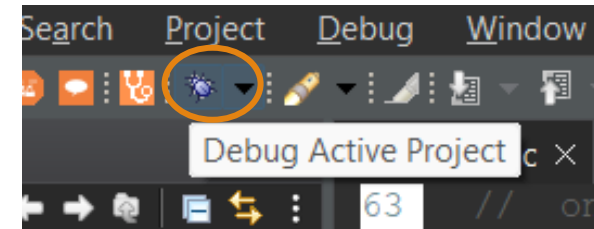
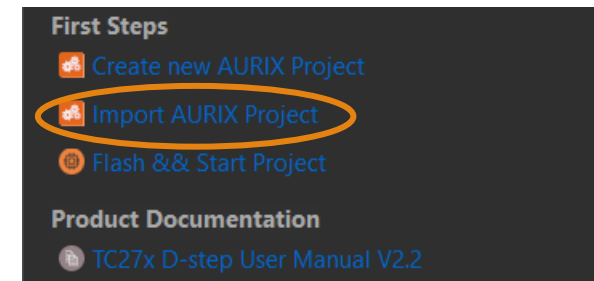


# Ambiente di sviluppo: AURIX Development Studio



Una volta importato il progetto per eseguirlo sulla nostra board è necessario dapprima impostarlo come **progetto attivo** (tasto destro sul progetto → Set Active Project), e poi flasharlo tramite cavo USB premendo il comando in figura **Flash && Start Project**.

Per debuggare il nostro programma basta selezionare l'icona mostrata nella figura in basso e passare alla vista *Debugging* nella quale è possibile controllare il flusso di esecuzione e monitorare le variabili tramite i *watchers*.



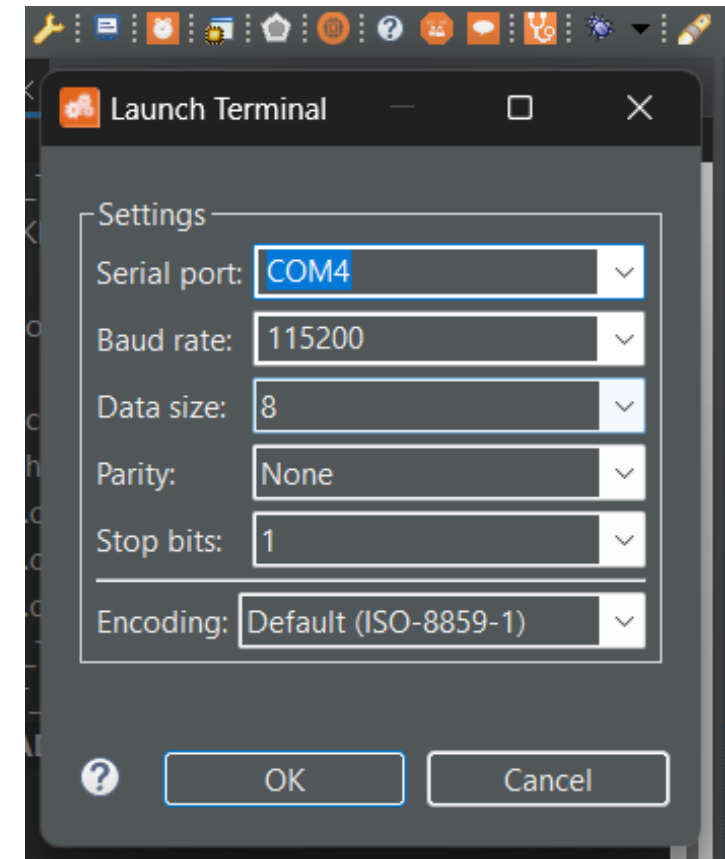
# Ambiente di sviluppo: AURIX Development Studio



E' inoltre integrata nell'IDE la possibilità di aprire un terminale connesso alla porta seriale desiderata. Basta premere sull'icona evidenziata sulla destra e poi su OK.

I parametri impostati di default sono già adatti alla comunicazione con la nostra board.

```
COM4 ×  
temp: 25.44  
temp: 25.44  
temp: 25.44  
temp: 25.38  
temp: 25.44  
temp: 25.38  
temp: 25.44
```



# Installazione real time operating system

---

Considerato l'obiettivo del progetto, l'installazione di un Sistema Operativo Real Time è essenziale per via della complessità dei protocolli di schedulazione dei processi e della pila protocollare TCP/IP, non implementata nativamente nelle librerie open source associate alla board Infineon.

Nel nostro caso abbiamo scelto di installare **freeRTOS**, il sistema operativo attualmente più diffuso nell'ambiente IoT.

Difficoltà incontrate: la board monta un processore con architettura proprietaria, quindi l'unica versione reperibile di freeRTOS compatibile è un **lite kit** che si occupa principalmente dello **scheduling dei processi**.

Una volta scaricato il progetto è possibile importarlo tramite AURIX Development Studio.



# Realizzazione Interfaccia seriale

---

L'**application kit TC2x7** fa uso di un collegamento **USB-microUSB** sia per l'alimentazione che per la comunicazione seriale.

Tra le librerie open source di Infineon (\*) sono disponibili diversi esempi di comunicazione seriale: per il nostro progetto abbiamo scelto di utilizzare come base di partenza **ASCLIN\_Shell\_UART\_1\_KIT\_TC297\_TFT**, progetto esempio che implementa una shell molto semplice.

La scelta è stata fatta in previsione dell'eventualità che si rivelasse necessaria una comunicazione bidirezionale durante l'avanzamento del progetto.

Abbiamo, infine, ripulito e caricato nel nostro progetto la libreria sopraccitata.

\* [https://github.com/Infineon/AURIX\\_code\\_examples/tree/master/code\\_examples](https://github.com/Infineon/AURIX_code_examples/tree/master/code_examples)



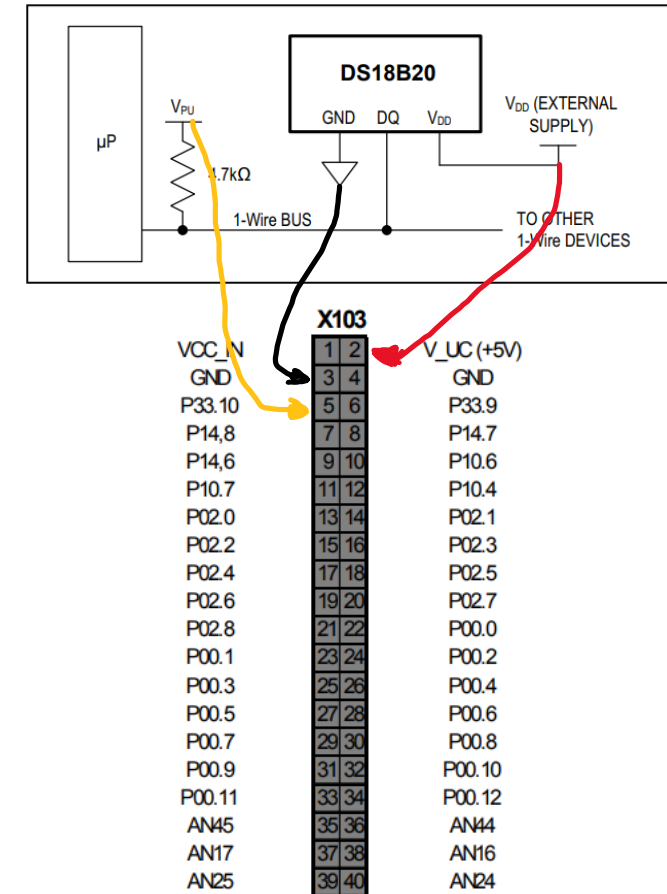
# Integrazione sensore di temperatura

Il sensore di temperatura **DS18B20**, come mostrato in figura, dispone di 3 pin: GND, DQ, Vdd.

Per alimentare il sensore è necessario fornire tra GND e Vdd una tensione compresa tra **3.0V** e **5.5V**.

La board **TC297** in nostro possesso fornisce una tensione di alimentazioni per dispositivi GPIO di **5V**, che rispetta ampiamente le specifiche del sensore.

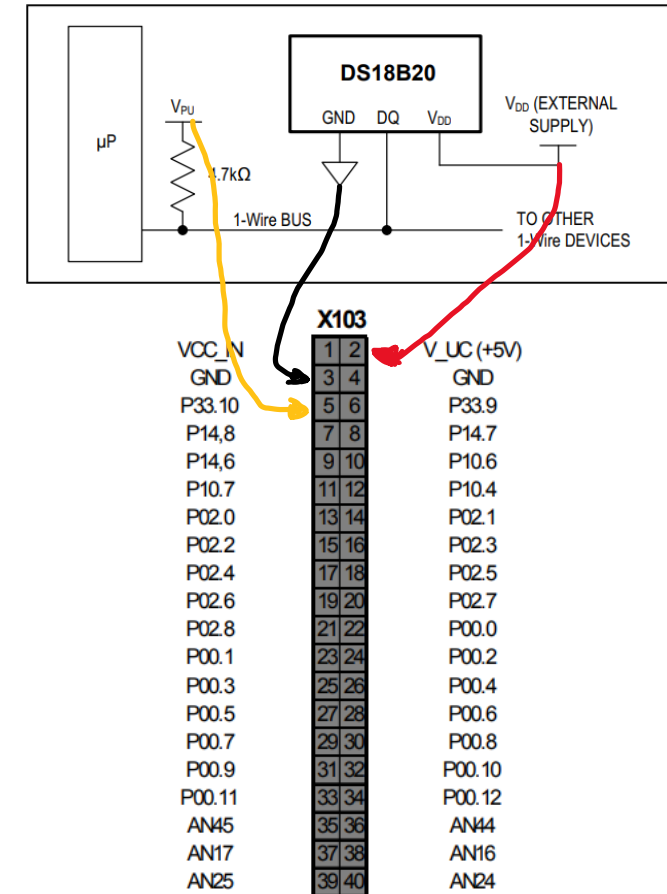
Abbiamo quindi connesso tramite una Breadboard il pin GND con il pin 3 del connettore **X103** (in figura) e il pin Vdd con il pin 2(V\_UC) della connettore **X103**.



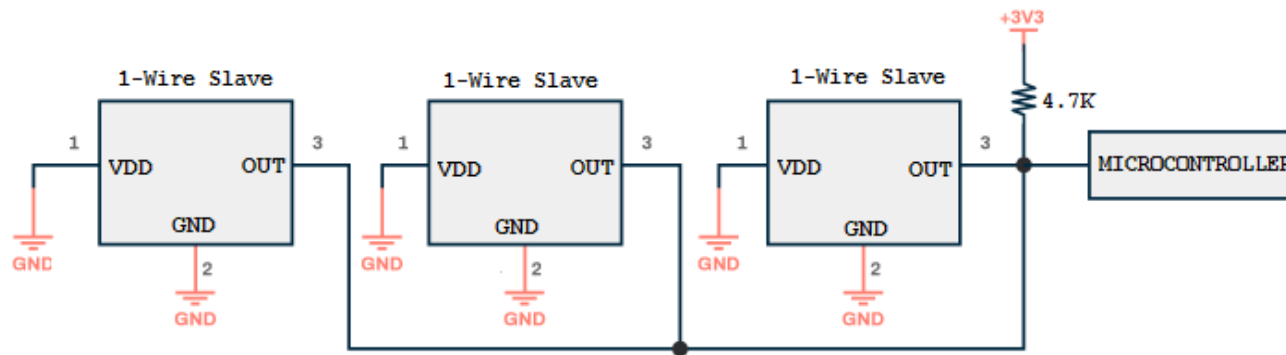
# Integrazione sensore di temperatura

Per quanto riguarda il pin **DQ** (connesso al pin 5 del connettore **X103**), utilizzato per lo scambio di dati, è necessario frappare tra la linea di comunicazione e la tensione una resistenza di pull up di circa 5k $\Omega$  affinché il sensore funzioni correttamente.

Come vedremo nelle prossime slide il sensore **DS18B20** occupa il ruolo di **Slave**, mentre la nostra board Infineon copre il ruolo di **Master** nella comunicazione tra i device.



# Integrazione sensore di temperatura: 1-Wire



La comunicazione avviene secondo il protocollo **1-Wire**, usato per stabilire una comunicazione seriale tra un microcontrollore e uno o più dispositivi mediante l'utilizzo di un singolo filo (come il nome lascia immaginare) connesso in modalità open drain e tenuto, in stato di idle, al livello alto di tensione tramite una resistenza di pull up, come mostrato in figura.

# Integrazione sensore di temperatura: 1-Wire

Per gestire più device contemporaneamente connessi al singolo filo, 1-Wire fa uso di un sistema di wake-up basato sull'univoco **registration number** salvato nella memoria ROM di ciascun dispositivo abilitato al protocollo.

I comandi per la lettura e gestione dell'**ID** sono inviati dal **Master** sul data-wire, e le possibili operazioni sono mostrate in figura.

1-Wire Command	Value	Description	Notes
Skip ROM	0xCC	Ignore device ID(s)	Intended to be used if there is only one device on the bus. However, it can also be used to send subsequent commands to <i>all</i> devices
Read ROM	0x33	Read a device's ID	Used when there is only one device on the bus
Search ROM	0xF0	Begin enumerating IDs	
Match ROM	0x55	Select a device with a specific ID	Next 64 bits to be written will be the known ID

# Integrazione sensore di temperatura: 1-Wire

---

E' dunque necessario realizzare il porting della libreria OneWire per la board TC297, rimappando i pin necessari alla comunicazione. Durante questa fase abbiamo semplificato la libreria per adattarla alla comunicazione con un solo dispositivo e conseguentemente diminuendone la complessità.

Infine abbiamo schedulato il task in figura, lanciato con intervalli di 1 secondo, che si occupa di inviare sull'interfaccia seriale la temperatura misurata dal sensore nel formato: «temp: 24.83\r\n».

```
static __attribute__((__noreturn__)) void OsTasks_temperature(void *arg)
{
    while(1) {
        vTaskDelay(pdMS_TO_TICKS(1000));
        broadcastConvert();
        float x = getTemperature();
        char temp[20];

        sprintf(temp, "temp: %.2f\r\n", x);
        printToConsole(temp);
    }
}
```

# Comunicazione con PC

---

Poiché impossibilitati a connettere la board direttamente ad internet, per via della mancanza di librerie adeguate non coperte da NDA, abbiamo ripiegato sull'utilizzo di un PC come hub intermediario.

Abbiamo quindi preparato uno script in **Python** che si occupa di fare il **fetch** della temperatura dalla board connessa tramite USB.

Per realizzare la comunicazione seriale con la scheda è stata usata la libreria:



• <https://pypi.org/project/pyserial/>

# Comunicazione con PC: Pyserial

---



La libreria **Pyserial** può essere installata lanciando il comando: *pip install pyserial*

Una volta installata va importata nel nostro script: `import serial`

Adesso si può aprire l'interfaccia seriale eseguendo : `ser = serial.Serial('/dev/ttyUSB0', 115200)`

Per leggere il buffer d'interfaccia si fa uso del comando: `lines = ser.read_all()` o `temp = ser.readline()`

Facciamo inoltre uso dell'attributo `ser.in_waiting` per tenere traccia del numero di byte contenuti nel buffer.

# Comunicazione con la rete: Adafruit



Per condividere in rete i valori fetchati dalla board utilizziamo **Adafruit IO**, ed in particolare alla funzionalità **Feed** offerta nel portale.

Per accedere al portale Adafruit è necessaria soltanto una semplice registrazione, tramite la quale avremo accesso a diverse funzionalità utili per l'IoT.

Tra queste compare l'area Feed mostrata in figura, tramite la quale possiamo **creare delle variabili** utilizzando l'interfaccia grafica ed **aggiornarle da remoto** dai nostri dispositivi.

+ New Feed

+ New Group

Default				<div><div>+</div><div>...</div></div>
Feed Name	Key	Last value	Recorded	
<input type="checkbox"/> temperature	temperature	24.56	1 day ago	



# Comunicazione con la rete: Adafruit

---

Lo stesso script Python discusso precedentemente si occupa anche della comunicazione con la rete ed il sistema **Feed** di **Adafruit**.

Per comunicare con le loro API abbiamo integrato lo script con la libreria **Python** di Adafruit IO, tramite la quale, dopo essersi autenticati con la chiave privata rilasciataci, possiamo aggiornare il valore dei feed desiderati, nel nostro caso il feed 'temperatura'.

Usando le seguenti istruzioni ci autenticiamo con il server Adafruit e fetchiamo l'istanza del feed desiderato:

```
aio = Adafruit_IO.Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)
feed = aio.feeds('temperature')
```

Per poi aggiornarne il valore sul server remoto tramite l'istruzione:

```
aio.send_data(feed.key, temp)
```



- [https://github.com/adafruit/Adafruit IO Python](https://github.com/adafruit/Adafruit_IO_Python)

# Comunicazione con la rete: Adafruit



E' possibile comunicare solo un certo numero di volte per minuto con il server di Adafruit, perciò abbiamo sviluppato un sistema di settaggi e di back off esponenziale per evitare il throttling della banda di rete.

```
DESIRED_UPDATE_TIME = 1  
UPDATE_TIME_THRESHOLD = 500
```

```
except Adafruit_IO.ThrottlingError:  
    print("too fast")  
    if sleeping_time > UPDATE_TIME_THRESHOLD:  
        print("Adafruit can't go faster than selected threshold: TERMINATING")  
        exit()  
    sleeping_time = sleeping_time * 2 #exponential backoff
```

```
if sleeping_time > DESIRED_UPDATE_TIME:  
    sleeping_time = sleeping_time - 1 #Response to exponential backoff, similar to TCP
```

# Interazione con Alexa: IFTTT

---

Tramite il servizio IFTTT (If This Then That) è possibile mettere in comunicazioni servizi diversi tra loro, scatenando, secondo la logica degli **if this-then that** delle azioni in risposta a dei trigger.

L'obiettivo è quindi collegare Adafruit ad Alexa, generando un annuncio ogni qual volta il feed della temperatura viene aggiornato.

Infine avviare l'annuncio sul nostro device dopo un nostro comando vocale tramite una routine di Alexa.



- <https://voicemonkey.io/>
- <https://ifttt.com/>

# Interazione con Alexa: IFTTT

Il nostro IFTTT **applet** è composto dal trigger di Adafruit **'any new data'** come **IF THIS**, e dall'action **'Make announcement'** di Voice Monkey come **THAN THAT**. L'annuncio può essere personalizzato con i dati del feed Adafruit.

Per utilizzare il Trigger di Voice Monkey è però necessario collegare la skill al nostro account Amazon e associare un dispositivo **speaker** su cui lanciare l'azione.

## Any new data

This Trigger fires any time there is new data in your feed.


## Make announcement

This action will make an announcement on your Alexa/Echo device using the text you supply. Ensure you open the Voice Monkey Skill as the final action in your routine. Text can be dynamic e.g. an ingredient from another IFTTT event or applet.

### Speakers

Add and manage a smart speaker device such as an Echo or Echo Show to play announcements or media content.

[+ Add a device](#)

Name	Device ID	Setup And Sync	Delete
pixel	pixel	<a href="#">Setup and sync</a>	

# Creare la routine di Alexa

Per creare la nostra routine, apriamo l'app di Alexa e selezioniamo home→Routine.

Premiamo il tasto + per creare una nuova routine, gli assegniamo un nome e la configuriamo come in figura.

In questa maniera, al nostro comando vocale desiderato, in questo caso *'che temperatura c'è in camera'* Alexa aprirà l'ultimo annuncio di Voice Monkey ricevuto.

Possiamo così ottenere la temperatura del nostro sensore ogni qual volta lo desideriamo.

