

Sviluppare un'applicazione LUIS ed integrarla con attuatore/sensore

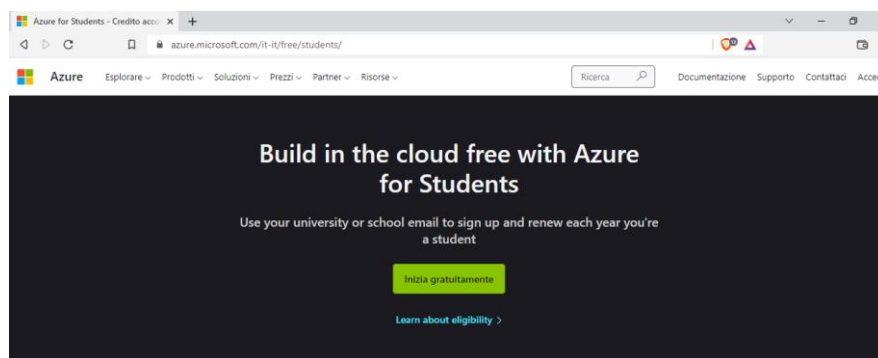
Lo scopo di questa attività è controllare l'attuatore mediante comandi vocali analizzati da LUIS ed ottenere i dati del sensore mediante sintesi vocale, interagendo con un server Flask o MQTT installato su Raspberry.

Nella sezione A verrà illustrato come configurare LUIS, nella sezione B come sviluppare un programma Python (sotto Windows) che interagisce con LUIS e nella sezione C come utilizzare la sintesi vocale. Nella sezione D, infine, verranno date indicazioni su come integrare il tutto con l'attuatore ed il sensore.

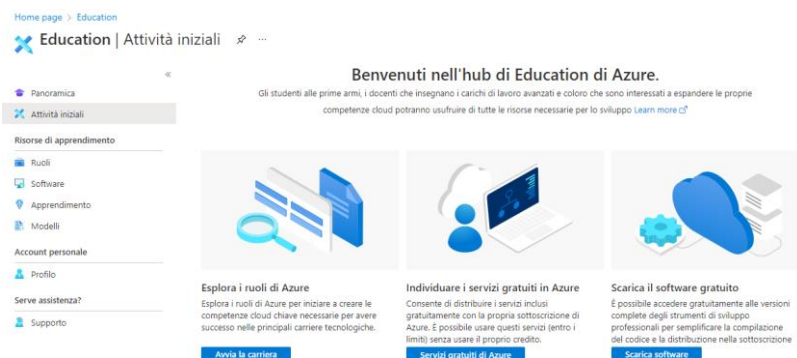
Sezione A: configurazione LUIS

Parte 1: creazione di un account Azure e creazione dell'applicazione LUIS

- a. Accedere a <https://azure.microsoft.com/it-it/free/students/>



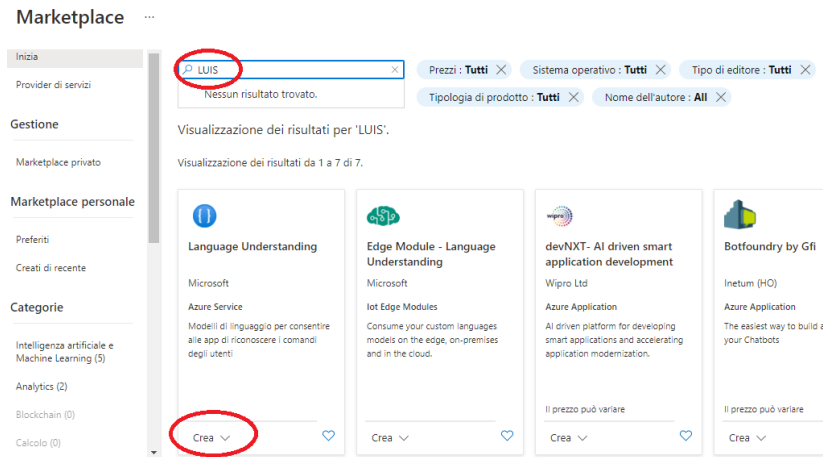
- b. Scegliere "Inizia gratuitamente", loggarsi con l'account scolastico ed eseguire la procedura di registrazione



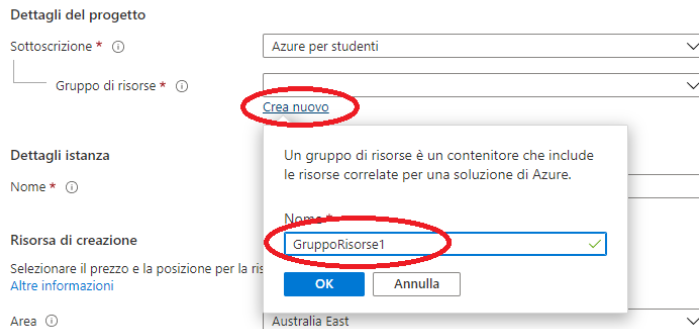
- c. Scegliere "Home page" (in alto a sinistra) e selezionare "Crea una risorsa"



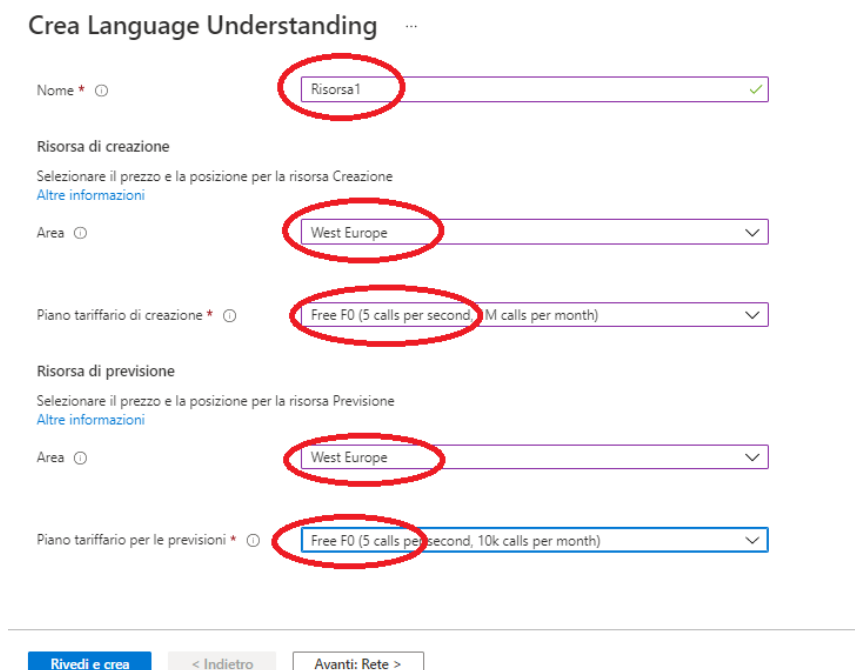
- d. Cercare “LUIS” e poi scegliere “Crea” nella finestra di “Language understanding”



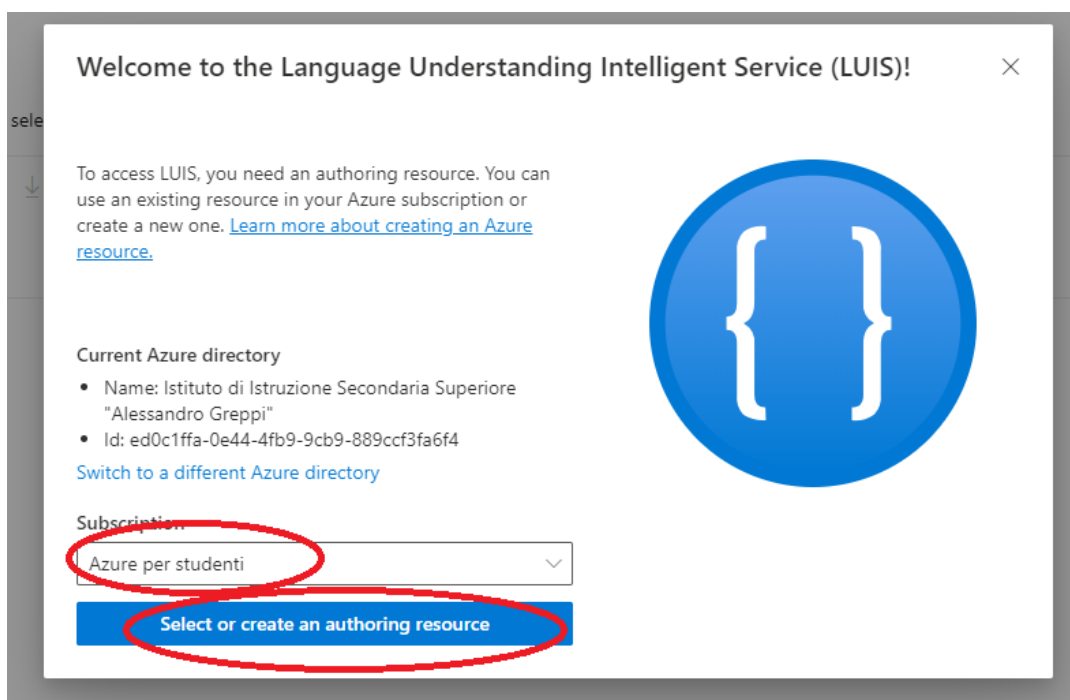
- e. Creare un gruppo di risorse con nome a piacere (ad esempio *GruppoRisorse1*)



- f. Assegnare un nome alla risorsa (ad esempio *Risorsa1*), scegliere zona (*West Europe*) e piano tariffario (*F0*) e clicca su “Rivedi e crea”



- g. Clicca su “Crea” e attendi qualche istante; scegliere poi “Vai al gruppo di risorse” dove si noterà che sono state create le risorse *Risorsa1* e *Risorsa1-Authoring*
- h. Accedere poi a <https://www.luis.ai/> e loggarsi con le proprie credenziali scolastiche



- i. Specificare la risorsa di Authoring

Choose an authoring resource

Switching your authoring resource will also switch to your apps. You can switch back at any time. [Learn more about resources in Azure.](#)

Azure directory *

Istituto di Istruzione Secondaria Superiore "Alessandro Greppi" ▾

Note: Switching directory will cause the page to refresh.

Azure subscription *

Azure per studenti ▾

Authoring resource

Risorsa1-Authoring ▾

Pricing tier: Free (F0)

Managed identity: Disabled

Create a new authoring resource

Done Cancel

- j. Scegliere "New App", assegnare un nome all'applicazione (ad esempio *App1*) e scegliere la risorsa creata precedentemente.

Conversation apps

Azure subscription: Azure per studenti / Authoring

+ New app ▾ Rename ▾ Export ▾

Create new app

Name *

App1

Culture * ⓘ

Italian ▾

Description

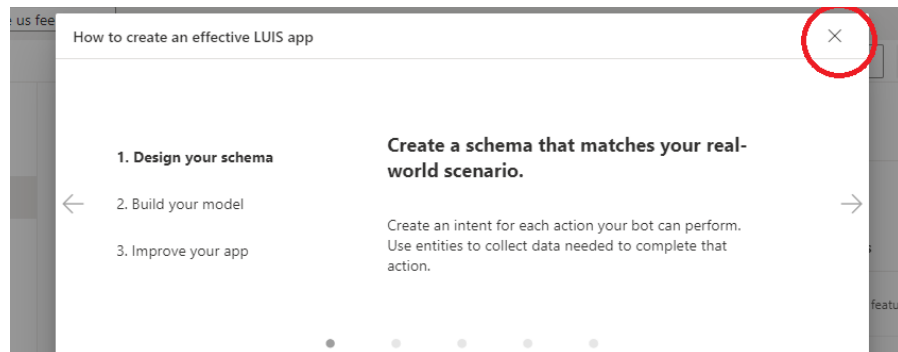
Type app description ...

Prediction resource ⓘ

Risorsa1 ▾

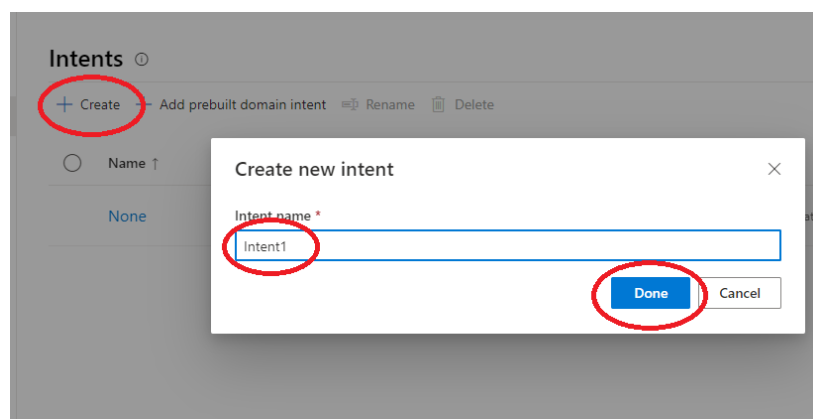
Done Cancel

- k. Chiudere la finestra di help che appare e passare alla fase successiva di configurazione.

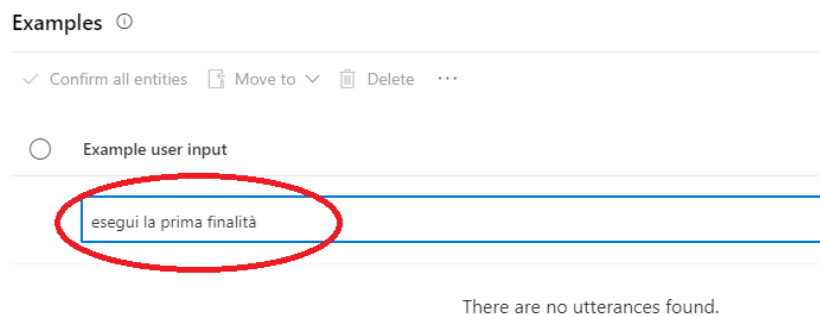


Parte 2: definizione del modello

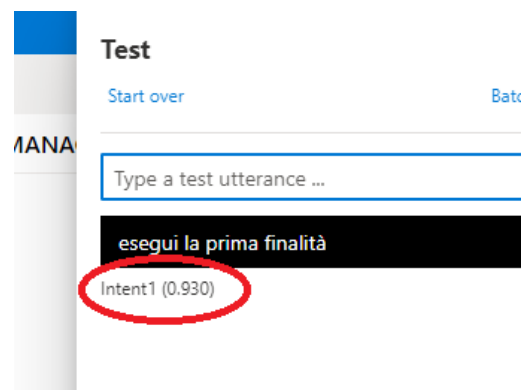
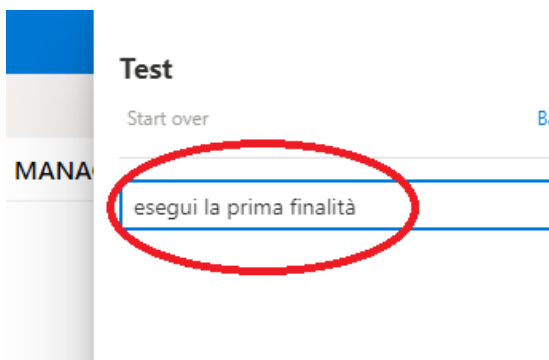
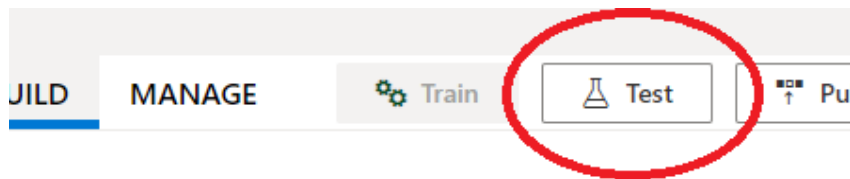
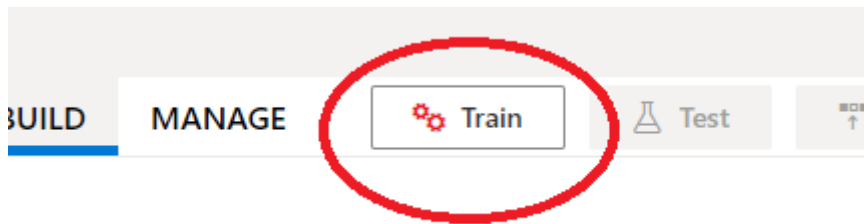
- a. Creare la prima **finalità** (*intent*) ed assegnare un nome (ad esempio *Intent1*)



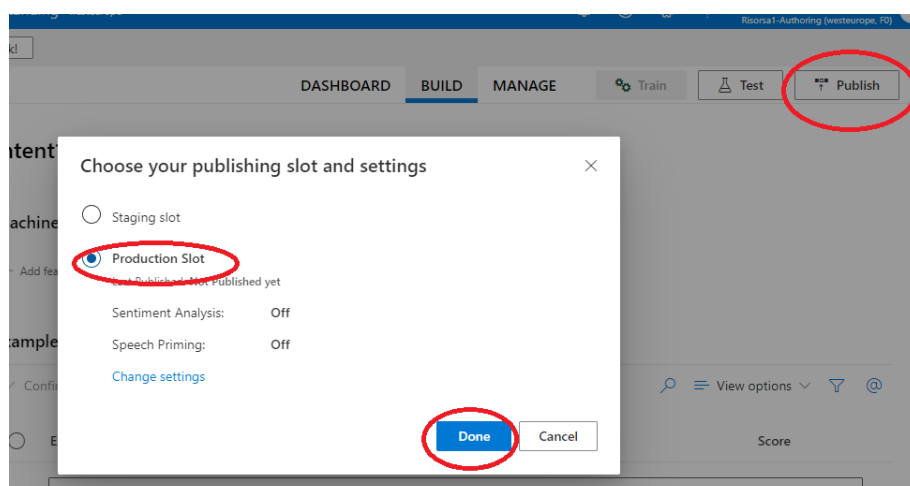
- b. Definire la prima **espressione di riconoscimento** (*utterance*), ad esempio: “*esegui la prima finalità*” e premere invio



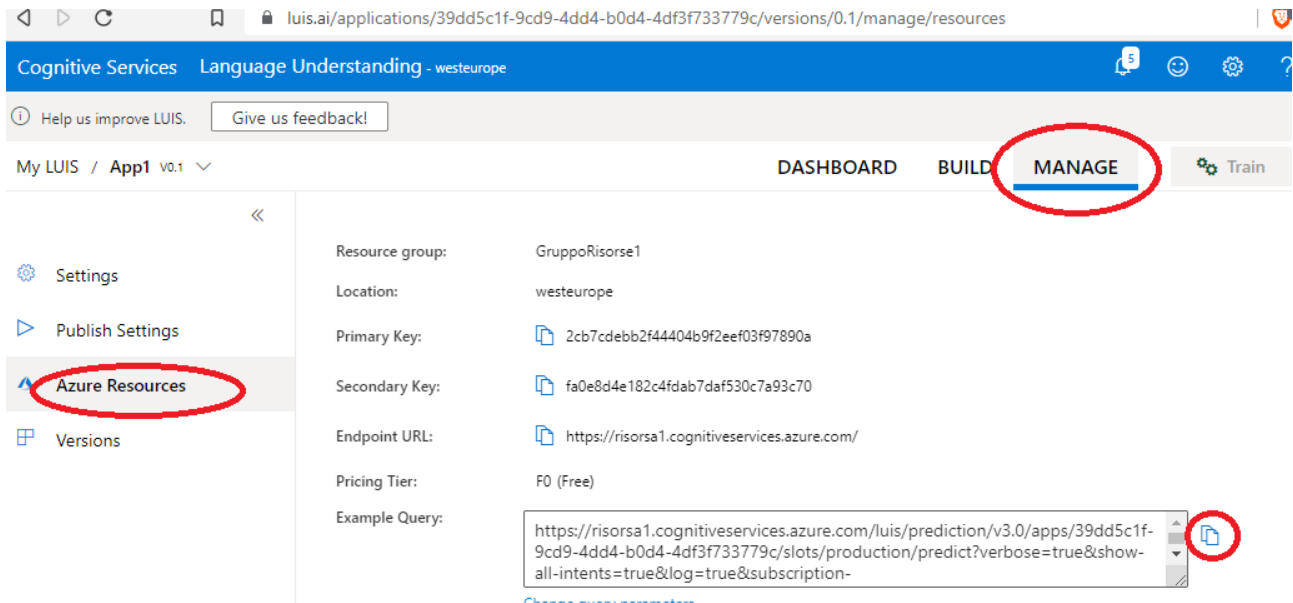
- c. Eseguire il training dell'applicazione e poi passare al test, fornendo l'utterance definito e premendo invio; verificare il corretto riconoscimento dell'intent.



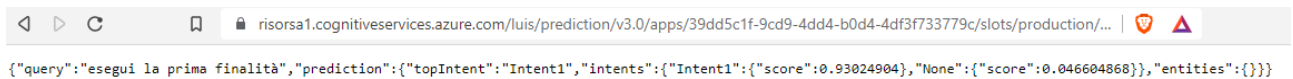
- d. Pubblicare l'applicazione



- e. Recuperare la stringa di query nella sezione “Manage” – “Azure Resource”, provarla nel browser indicando “**esegui la prima finalit **” come parametro e verificare il file JSON ricevuto



```
https://risorsa1.cognitiveservices.azure.com/luis/prediction/v3.0/apps/39dd5c1f-9cd9-4dd4-b0d4-4df3f733779c/slots/production/predict?verbose=true&show-all-intents=true&log=true&subscription-key=2cb7cdebb2f44404b9f2eef03f97890a&query=esegui la prima finalit 
```



- f. Recuperare il codice applicazione (in **giallo**) e il codice sottoscrizione (in **verde**)

Sezione B: programma Python di esempio

1. Installare la libreria

```
Py -m pip install azure-cognitiveservices-speech
```

2. Provare il programma di esempio che produrr  il seguente output:

```
IntentRecognitionResult(result_id=683ebc97b7d3451d917889fb78a41364, text="Esegui la prima finalit .", intent_id=Intent1, reason=ResultReason.RecognizedIntent)
```

```
import time
import azure.cognitiveservices.speech as speechsdk

#-----
# costanti
```

```

#-----
SUBSCRIPTION_KEY="2cb7cdebb2f44404b9f2eef03f97890a"
APPLICATION_ID="39dd5c1f-9cd9-4dd4-b0d4-4df3f733779c"
REGION="westeurope"
#-----
#  configurazione (proxy opzionale)
#-----
intent_config = speechsdk.SpeechConfig(
    subscription=SUBSCRIPTION_KEY,
    region=REGION)
#intent_config.set_proxy("proxy.intranet",3128,"","")
intent_config.speech_recognition_language="it-IT"
#-----
#  creazione riconoscitore vocale
#-----
intent_recognizer =
speechsdk.intent.IntentRecognizer(speech_config=intent_config)
model = speechsdk.intent.LanguageUnderstandingModel(app_id=APPLICATION_ID)
intent_recognizer.add_all_intents(model)
#-----
#  funzione di callback chiamata dopo ogni riconoscimento
#-----
def riconosciuto(evt):
    print(evt.result)
#-----
#  attivazione del riconoscitore
#-----
intent_recognizer.recognized.connect(riconosciuto)
intent_recognizer.start_continuous_recognition()
#-----
#  ciclo di attesa dei riconoscimenti
#-----
while True:
    time.sleep(0.5)
#-----
#  terminazione del riconoscimento
#  (mai attivata in questo esempio)
#-----
intent_recognizer.stop_continuous_recognition()

```


Parte 3: sintesi vocale

- a. Accedere al portale Azure per la creazione della risorsa di sintesi vocale (usare il link diretto seguente):

<https://ms.portal.azure.com/#create/Microsoft.CognitiveServicesSpeechServices>

- b. Creare alla risorsa (ad esempio *Sintesi1*) e, dopo aver completato la creazione, accedere alla sezione “Chiavi ed endpoint” e copiare la prima chiave:

Cerca (CTRL+/) << Elimina

Panoramica

Log attività

Controllo di accesso (IAM)

Tag

Diagnostica e risoluzione dei problemi

Gestisci risorse

Chiavi ed endpoint

Crittografia

Piano tariffario

Rete

Aiutaci a migliorare Voce. Partecipa al nostro sondaggio!

Informazioni di base

Gruppo di risorse (spostare) : GruppoRisorse1

Stato : Attivo

Località : West Europe

Sottoscrizione (spostare) : Azure per studenti

ID sottoscrizione : 2c310791-9e54-486a-b528-684cf98d962f

Tag (modificare) : Fare clic qui per aggiungere i tag

Attività iniziali Individua Sviluppa Distribuisci

Queste chiavi vengono usate per accedere all'API Servizi cognitivi. Non condividere le chiavi. Archivarle in modo sicuro, ad esempio usando Azure Key Vault. È inoltre consigliabile rigenerare regolarmente queste chiavi. È necessaria una sola chiave per effettuare una chiamata API. Durante la rigenerazione della prima chiave, è possibile usare la seconda chiave per l'accesso continuato al servizio.

Nascondi chiavi

CHIAVE 1

5ba79c9336da495996c3ed52817ecc53

CHIAVE 2

92761862fee34548b24370572b2afc27

Località/Area

westeurope

Endpoint

https://westeurope.api.cognitive.microsoft.com/sts/v1.0/issuetoken

c. Provare il programma di esempio:

```
import time
import azure.cognitiveservices.speech as speechsdk
#-----
#  costanti
#-----
SUBSCRIPTION_KEY="5ba79c9336da495996c3ed52817ecc53"
REGION="westeurope"
#-----
#  configurazione (proxy opzionale)
#-----
speech_config = speechsdk.SpeechConfig(
    subscription=SUBSCRIPTION_KEY,
    region=REGION)
speech_config.speech_synthesis_language="it-IT"
#speech_config.set_proxy("proxy.intranet", 3128, "", "")
#-----
#  creazione sintetizzatore vocale
#-----
speech_synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config)
#-----
#  sintesi
#-----
text="ciao, come va?"
result = speech_synthesizer.speak_text_async(text).get()
print(result)
```

Parte 4: integrazione con il sensore e l'attuatore

L'SDK di Azure gira sotto Windows e Debian 10. Esistono quindi diverse possibilità di integrazione:

- *Installare la versione Debian 10 per Raspberry, configurare un microfono USB e permettere all'applicazione Python-Azure di interagire via RF24 con il sensore e l'attuatore*
- *Installare l'applicazione Python-Azure sotto Windows, un server Flask-ngrok su Raspberry ed interagire via richieste http-JSON (come già fatto in un'attività precedente).*
- *Installare l'applicazione Python-Azure sotto Windows, un server Mosquitto-ngrok su Raspberry ed interagire via MQTT (come già fatto in un'attività precedente).*