Connect a chip to Azure

A journary around the Azure SDK for small device (bare metal)

Mirco Vanini

Microsoft MVP Developer Technologies
Proxima Software


altitudo


beanTech
IT moves your business


OVERNET.
upgrade your digital skills


[stesi]
Powered by Innovation


ADAWEN
Add Value Group


AZUREBOX


BETA 80 GROUP


elettrone
intelligenza energetica


fairmat


jmatica

Agenda

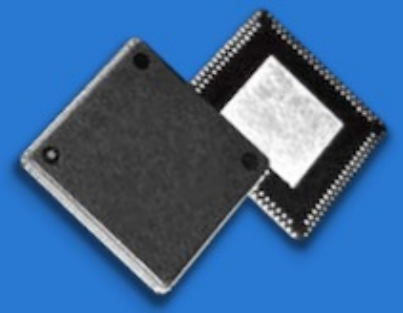
- MCU/CPU
- Azure IoT SDKs
- Azure SDK for Embedded C
- Tools
- Demo

MCU, CPU, what's the difference?

- A Micro Controller Unit (MCU) is a small processor, usually a synonym for low power consumption, and small size. Modern ones like STM32F7, ESP32, TI CC1352R are ARM Cortex M based, embedding few hundreds of kilobytes of RAM, small flash and a decent few hundred million of hertz clock cadence.

- A Central Processing Unit (CPU) is a larger processor, in this category, you'll find all the traditional x86, x64, but as well other ARM processors like the one used in Raspberry Pi.

The microcontroller (MCU)
a low-cost, single chip computer



9 BILLION new MCU devices
built and deployed every year

Azure IoT products and services

Azure IoT Operations

Capture asset data, process it at the edge, and send it to the cloud with Azure Arc–enabled services

Azure IoT Hub

Connect, manage, and scale billions of IoT devices from the edge to the cloud.

Azure Digital Twins

Build next-generation IoT spatial intelligence solutions by replicating real physical spaces and creating connected environments.

Azure IoT Edge

Extend cloud intelligence and analytics by moving workloads and business logic from the cloud to edge devices.

Azure Sphere

Securely connect MCU-powered devices from the silicon to the cloud.

Windows for IoT

Long term OS support and services to manage devices.

Azure RTOS

Making embedded IoT development and connectivity easy.

Microsoft Azure IoT SDKs

- Azure IoT SDK for Python
- Azure IoT SDK for Node.js
- Azure IoT SDK for Java
- Azure IoT SDK for Golang
- Azure IoT SDK for .NET
- Azure IoT SDK for C
- Azure IoT SDK for Embedded C
- Azure IoT middleware for Azure RTOS
- Azure IoT middleware for FreeRTOS

innova community altitudo beanTech IT moves your business #GLOBALAZURE 2024 OVERNET. upgrade your digital skills [stesi] Powered by Innovation CloudGen

Azure IoT C SDKs

- The Azure IOT Hub Device SDK allows applications written in C99 or later or C++ to communicate easily with Azure IoT Hub, Azure IoT Central and to Azure IoT Device Provisioning.

- Support Azure IoT TLS over TCP/IP

- Support SHA-256 (optional): necessary to generate the secure token for authenticating the device with the service. Different authentication methods are available and not all require SHA-256.
- Have a Real Time Clock or implement code to connect to an NTP server: necessary for both establishing the TLS connection and generating the secure token for authentication.
- Having at least 64KB of RAM: the memory footprint of the SDK depends on the SDK and protocol used as well as the platform targeted. The smallest footprint is achieved targeting microcontrollers.

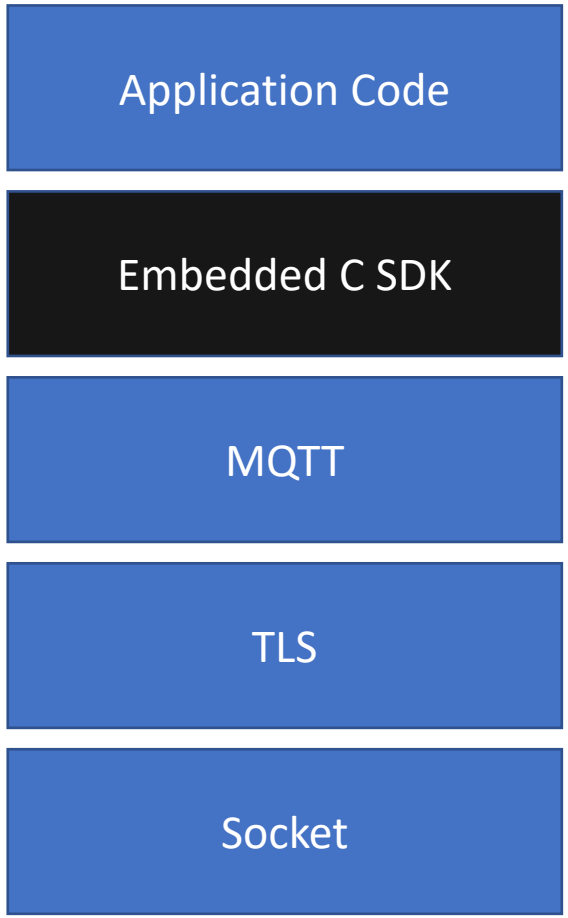
Azure IoT C SDKs

- Azure IoT SDK for Embedded C is an alternative for **constrained devices** which enables the BYO (bring your own) network approach: IoT developers have the freedom of choice to bring MQTT client, TLS and Socket of their choice to create a device solution.

- Azure IoT middleware for Azure RTOS builds on top of the embedded SDK and tightly couples with the Azure RTOS family of networking and OS products. This gives you very performant and small applications for real-time, constrained devices.

- Azure IoT middleware for FreeRTOS builds on top of the embedded SDK and takes care of the MQTT stack while integrating with FreeRTOS. This maintains the focus on constrained devices and gives users a distilled Azure IoT feature set while allowing for flexibility with their networking stack.

Azure IoT C SDK



- Telemetry
- CDC messages
- Device Twin
- Direct Methods
- Device Provisioning

The core implementation of the SDK is IoTHubClient library that contains the implementation of the lowest API layer in the SDK.

The library depends on other open-source libraries:

The Azure C shared utility library, which provides common functionality for basic tasks (such as strings, list manipulation, and IO) needed across several Azure-related C SDKs.

The Azure uAMQP library, which is a client-side implementation of AMQP optimized for resource constrained devices.

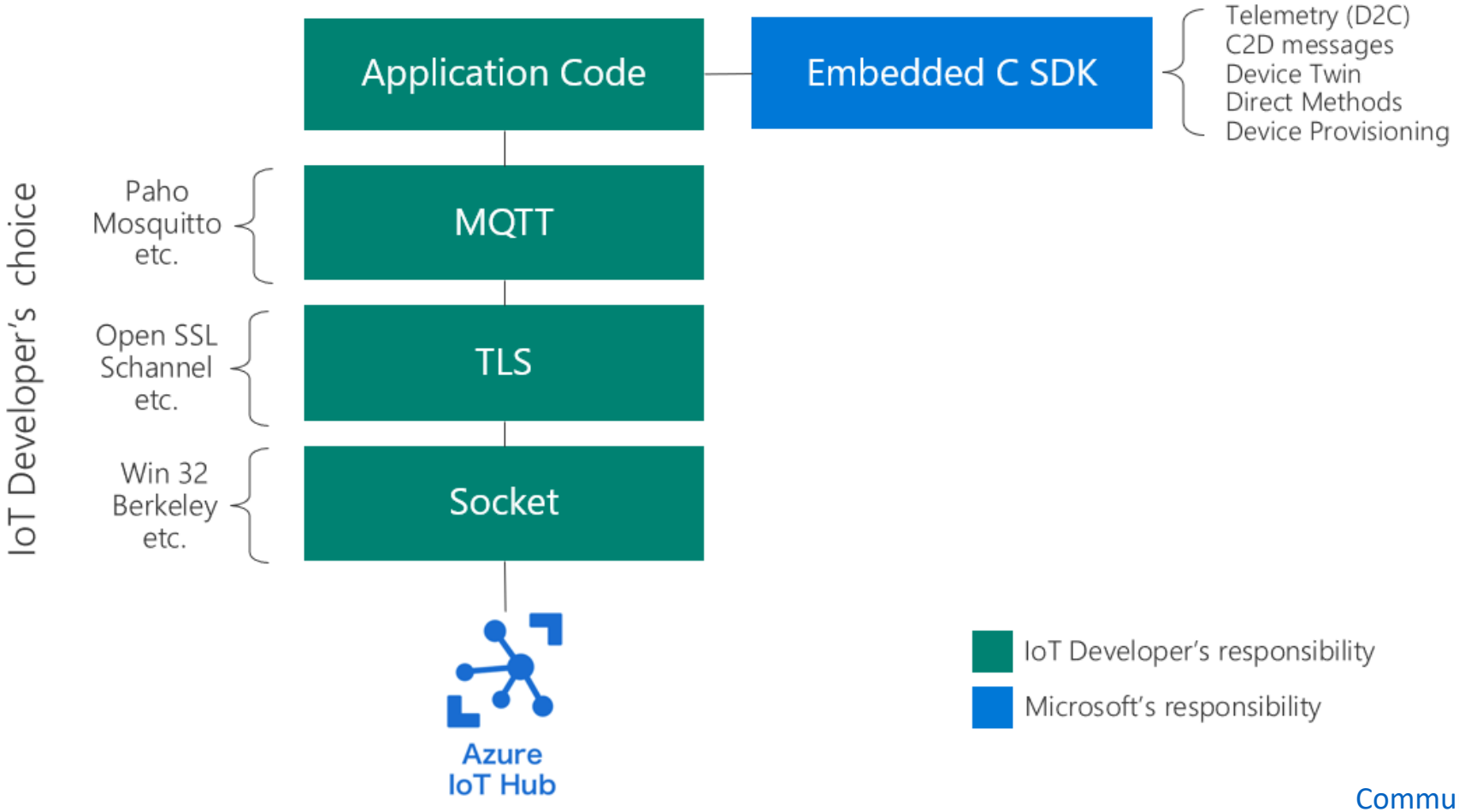
The Azure uMQTT library, which is a general-purpose library implementing the MQTT protocol and optimized for resource constrained devices.

Azure SDK for Embedded C – Design philosophy

- Optimize for size
- Customer choice
- No dynamic memory
- Full embrace MQTT
- BYO Network stack
- Our “Everywhere SDK*”

*on any device that can connect to a network

Azure SDK for Embedded C – IoT client libraries



The Azure IoT Client library is created to facilitate connectivity to Azure IoT services alongside an MQTT and TLS stack of the user's choice.

This means that this SDK is NOT a platform but instead is a true SDK library.

From a functional perspective, this means that the user's application code (not the SDK) calls directly to the MQTT stack of their choice.

The SDK provides utilities (functions, default values, etc.) that help make the connection and feature set easier to use.

Azure IoT Client MQTT State Machine

Communicate with your IoT hub using the MQTT protocol

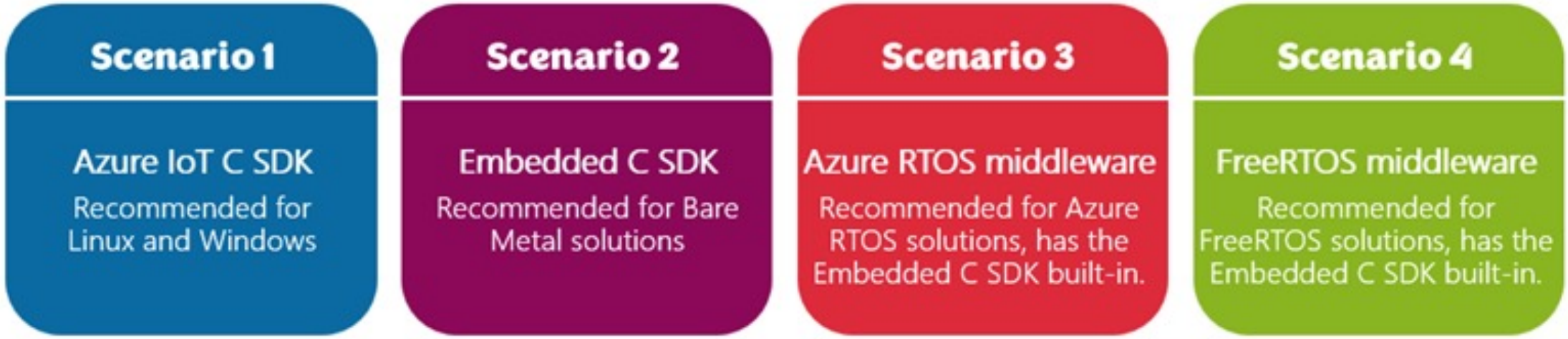
Azure SDK for Embedded C – Key Features

√ feature available √* feature partially available (see Description for details) × feature planned but not supported

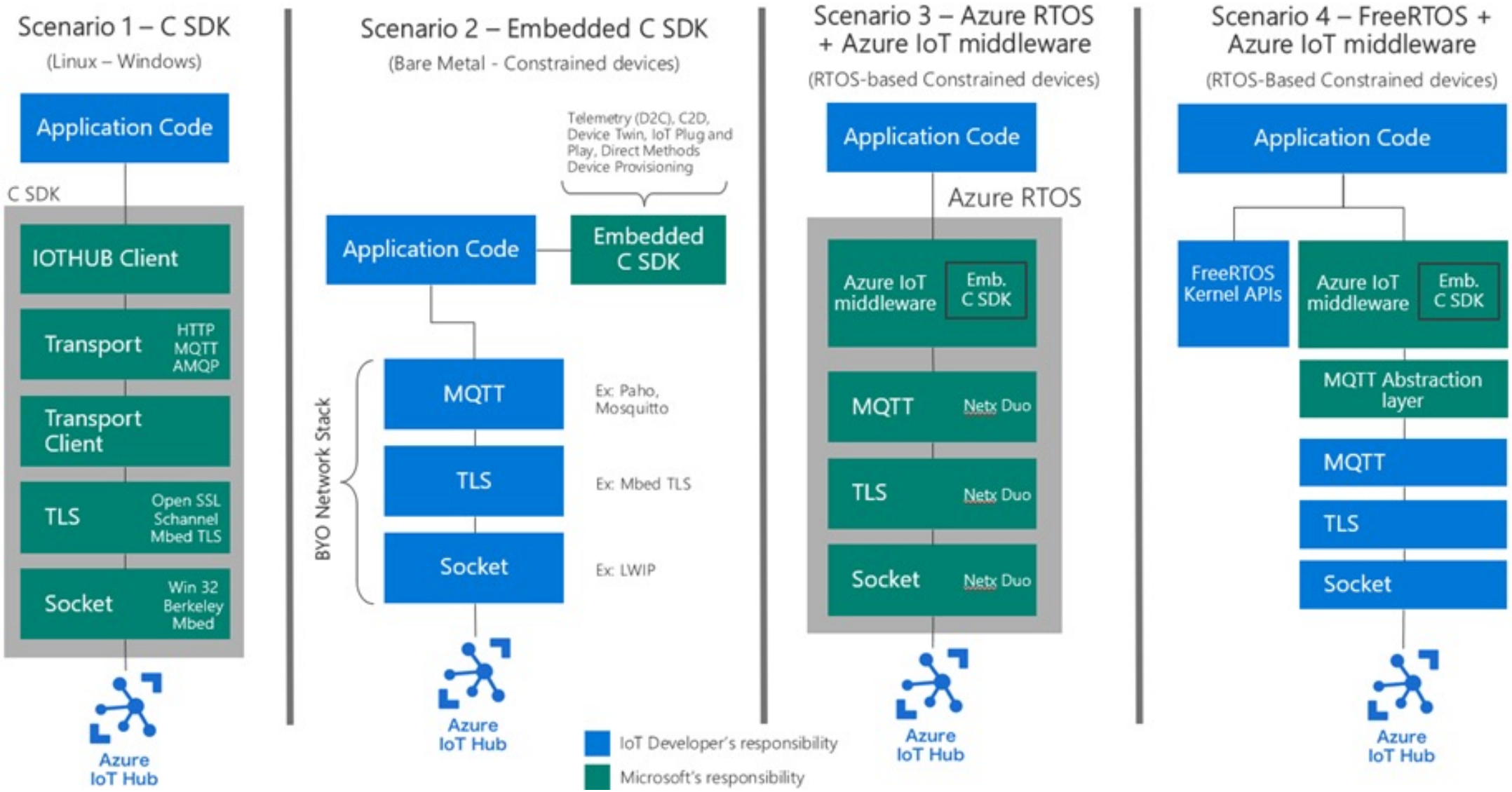
Feature	Azure SDK for Embedded C	Description
Send device-to-cloud message	√	Send device-to-cloud messages to IoT Hub with the option to add custom message properties.
Receive cloud-to-device messages	√	Receive cloud-to-device messages and associated properties from IoT Hub.
Device Twins	√	IoT Hub persists a device twin for each device that you connect to IoT Hub. The device can perform operations like get twin document, subscribe to desired property updates.
Direct Methods	√	IoT Hub gives you the ability to invoke direct methods on devices from the cloud.
DPS - Device Provisioning Service	√	This SDK supports connecting your device to the Device Provisioning Service via, for example, individual enrollment using an X.509 leaf certificate.
Protocol	MQTT	The Azure SDK for Embedded C supports only MQTT.
Retry Policies	√*	The Azure SDK for Embedded C provides guidelines for retries, but actual retries should be handled by the application.
IoT Plug and Play	√	IoT Plug and Play enables solution builders to integrate smart devices with their solutions without any manual configuration.

inn0va community altitudo beanTech IT moves your business #GLOBALAZURE OVERNET. upgrade your digital skills [stesi] Powered by Innovation CloudGen

C SDK and Embedded C SDK usage scenarios



C-based SDK technical usage scenarios



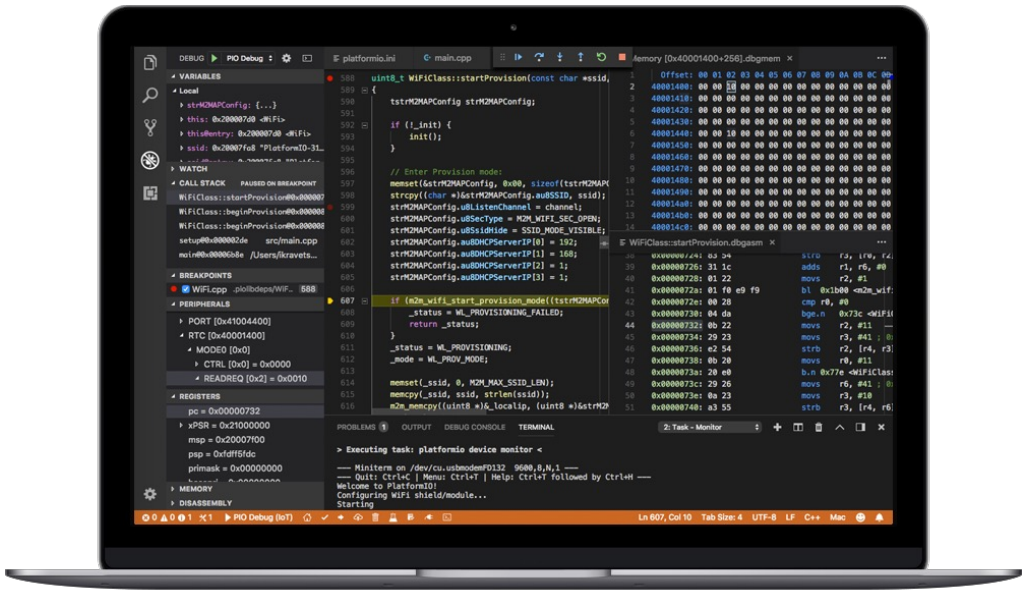
Azure IoT Features Supported by each SDK

	Azure IoT C SDK	**Azure SDK for Embedded C**	**Azure IoT middleware for Azure RTOS**	**Azure IoT middleware for FreeRTOS**
SAS Client Authentication	Yes	Yes	Yes	Yes
x509 Client Authentication	Yes	Yes	Yes	Yes
Device Provisioning	Yes	Yes	Yes	Yes
Telemetry	Yes	Yes	Yes	Yes
Cloud-to-Device Messages	Yes	Yes	Yes	Yes
Direct Methods	Yes	Yes	Yes	Yes
Device Twin	Yes	Yes	Yes	Yes
IoT Plug-And-Play	Yes	Yes	Yes	Yes
Telemetry batching (AMQP, HTTP)	Yes	No	No	No
Uploads to Azure Blob	Yes	No	No	No
Automatic integration in IoT Edge hosted containers	Yes	No	No	No

innova community altitudo beanTech IT moves your business #GLOBALAZURE OVERNET. upgrade your digital skills [stesi] Powered by Innovation CloudGen

Demo Tools

platformio.org

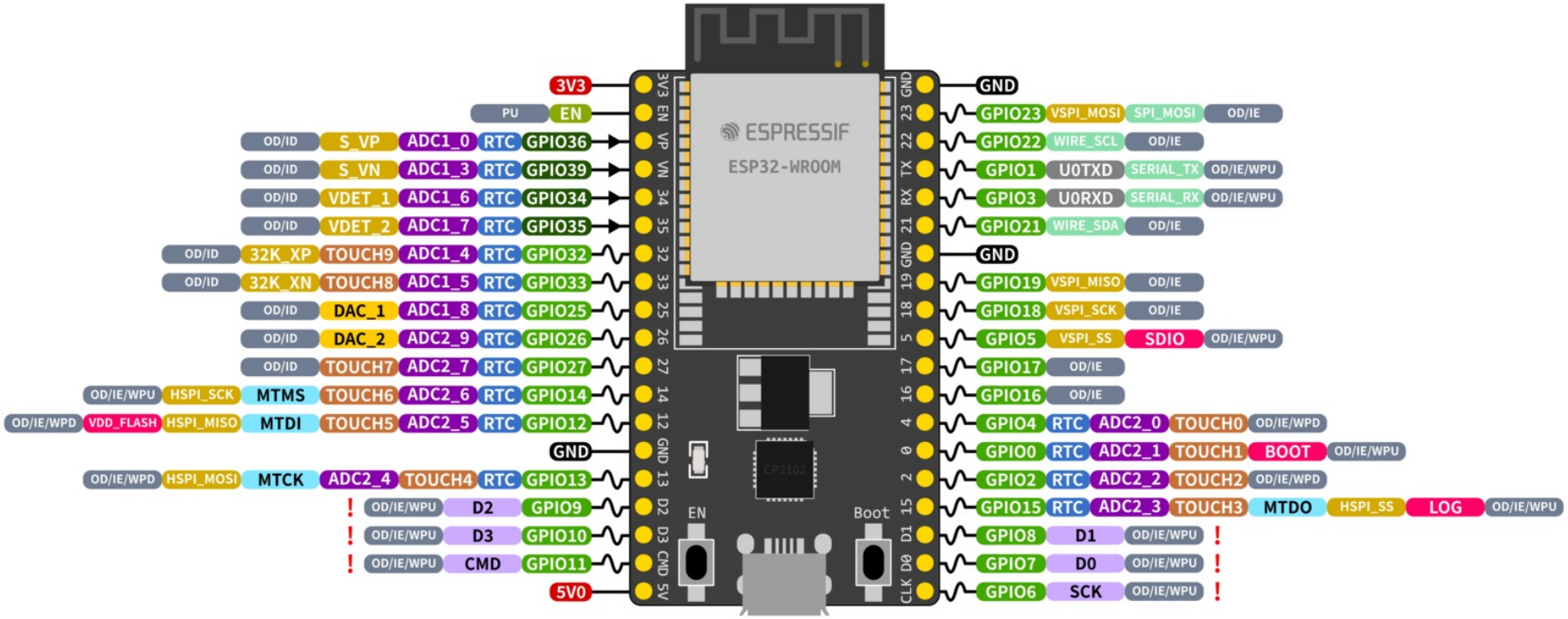


Professional collaborative platform for embedded development

- A lightweight but powerful cross-platform source code editor
- Smart code completions based on variable types, function definitions, and library dependencies
- Multi-projects workflow with easy navigation around project codebase, multiple panes, and themes support
- Seamless integration with PlatformIO Home (UI) with board and library managers
- Intuitive project wizard and a wide range of example projects
- Built-in Terminal with PlatformIO Core (CLI) and powerful Serial Port Monitor

ESP32-DevKitC

ESPRESSIF



ESP32 Specs

32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
BLuetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C,
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

PWM Capable Pin
GPIOX GPIO Input Only
GPIOX GPIO Input and Output
DAC_X Digital-to-Analog Converter
DEBUG JTAG for Debugging
FLASH External Flash Memory (SPI)
ADCX_CH Analog-to-Digital Converter
TOUCHX Touch Sensor Input Channel
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
ARDUINO Arduino Related Functions
STRAP Strapping Pin Functions

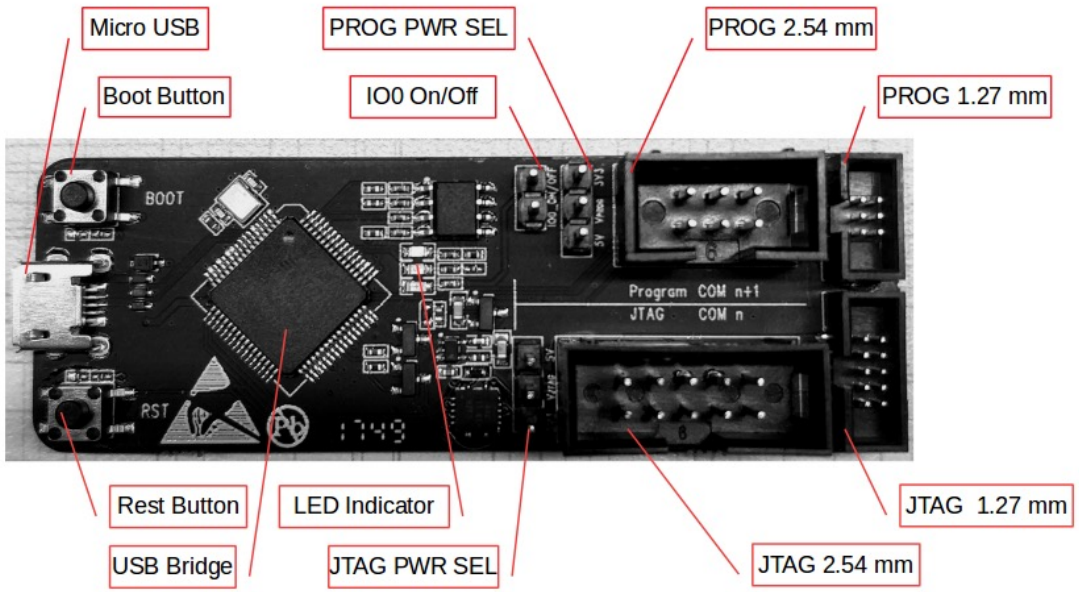
RTC RTC Power Domain (VDD3P3_RTC)
GND Ground
PWD Power Rails (3V3 and 5V)
! Pin Shared with the Flash Memory *Can't be used as regular GPIO*

GPIO STATE

WPU: *Weak Pull-up (Internal)*
WPD: *Weak Pull-down (Internal)*
PU: *Pull-up (External)*
IE: *Input Enable (After Reset)*
ID: *Input Disabled (After Reset)*
OE: *Output Enable (After Reset)*
OD: *Output Disabled (After Reset)*

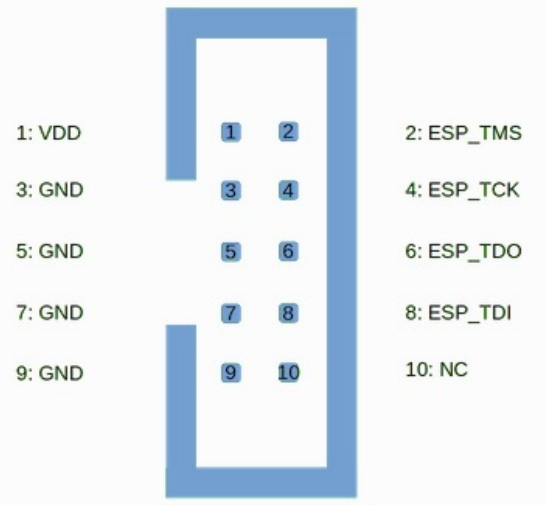
inn0va community
altitudo
beanTech IT moves your business
#GLOBALAZURE
OVERNET. upgrade your digital skills
[stesi] Powered by Innovation
CloudGen

ESP-Prog



ESP-Prog is one of Espressif's development and debugging tools, with functions including automatic firmware downloading, serial communication, and JTAG online debugging. ESP-Prog's automatic firmware downloading and serial communication functions are supported on both the ESP8266 and ESP32 platforms, while the JTAG online debugging is supported only on the ESP32 platform

ESP-Prog JTAG 10-Pin Connector	Board JTAG Pin	Description
1	VDD	Positive Supply Voltage — Power supply for JTAG interface drivers
3	GND	Digital ground
2	ESP_TMS	Test Mode State
4	ESP_TCK	JTAG Return Test Clock
6	ESP_TDO	Test Data Out
8	ESP_TDI	Test Data In

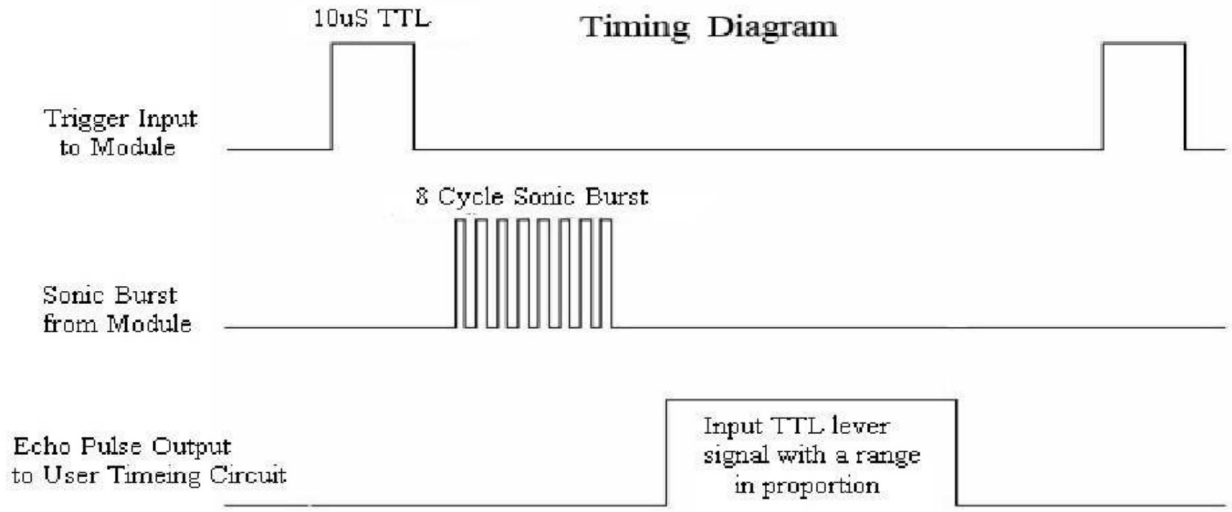


JTAG Wiring Connections

Board Pin	JTAG Tool Pin
IO13	TCK
IO12	TDI
IO15	TDO
IO14	TMS
EN	RST
GND	GND

Ultrasonic Ranging Module HC – SR04

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.

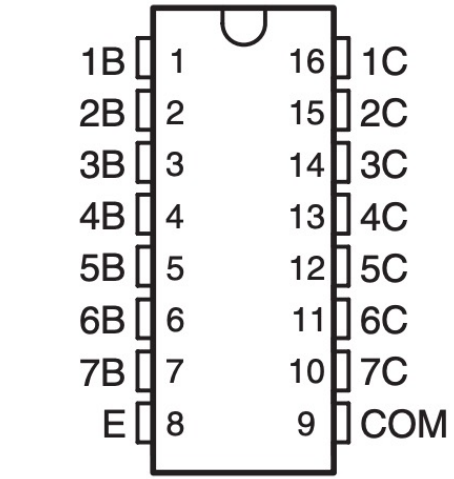


Stepper Motor 5V 4-Phase 5-Wire & ULN2003 Driver Board

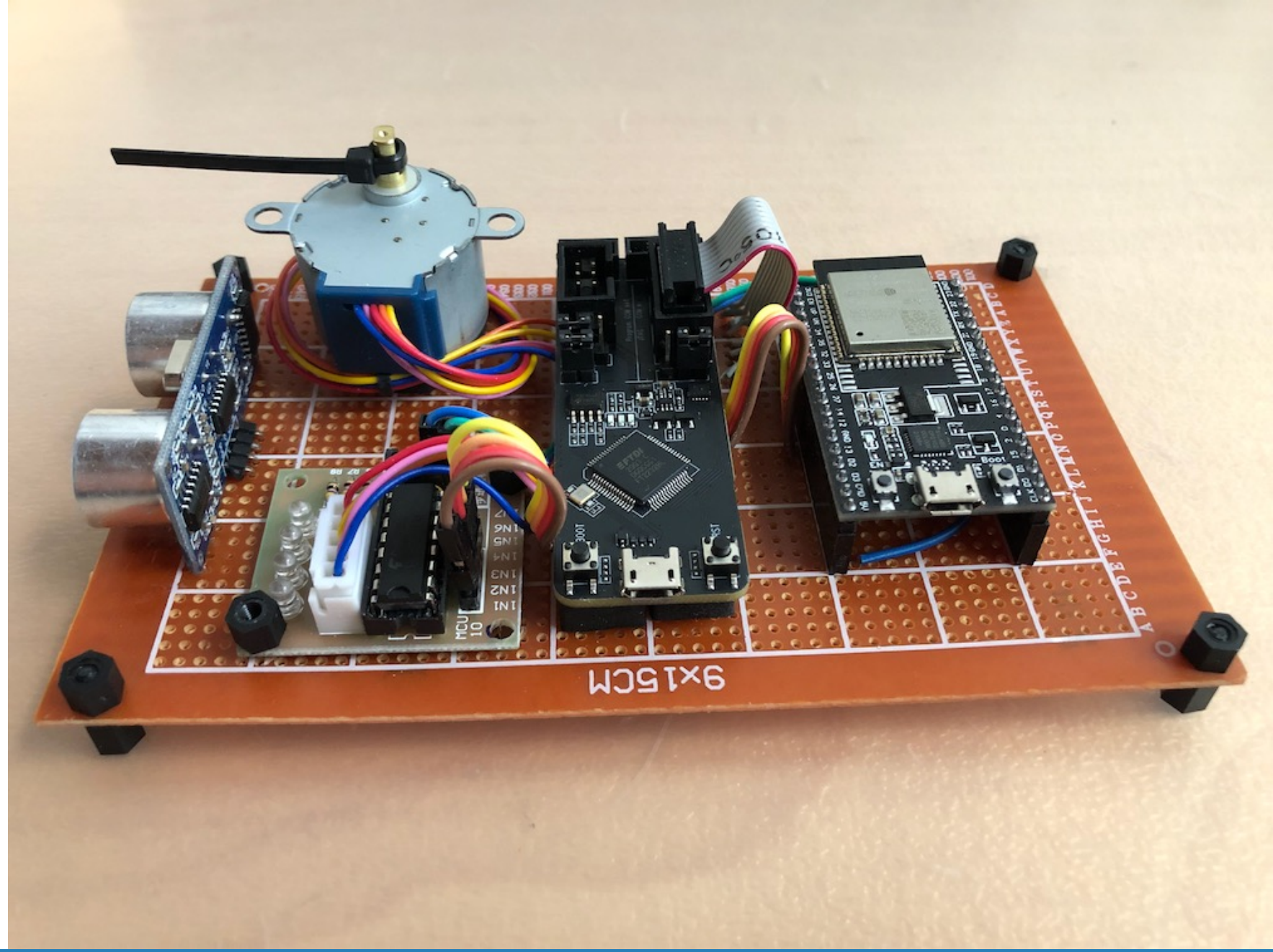
FEATURES

- **500-mA-Rated Collector Current (Single Output)**
- **High-Voltage Outputs: 50 V**
- **Output Clamp Diodes**
- **Inputs Compatible With Various Types of Logic**
- **Relay-Driver Applications**

ULN2002A . . . N PACKAGE
ULN2003A . . . D, N, NS, OR PW PACKAGE
ULN2004A . . . D, N, OR NS PACKAGE
ULQ2003A, ULQ2004A . . . D OR N PACKAGE
(TOP VIEW)



Demo



About me…

Mirco Vanini
Microsoft MVP Developer Technologies

Consultant focused on industrial and embedded solutions using .NET and other native SDKs with over 30 years of experience, XeDotNet community co-founder, speaker and Microsoft MVP since 2012

@MircoVanini
www.proxsoft.it
https://www.linkedin.com/in/proxsoft

Microsoft Most Valuable Professional
inn0va community
altitudo
beanTech IT moves your business
#GLOBALAZURE
OVERNET. upgrade your digital skills
[stesi] Powered by Innovation
CloudGen

Thanks !

Q&A

CloudGen

ominidi verso l'evoluzione