

# Integrating Microcontrollers into TANGO Control Systems with a Standardized Library and Dynamic Device Server

Antonio Bartalesi,  
Laura Torres Garcia

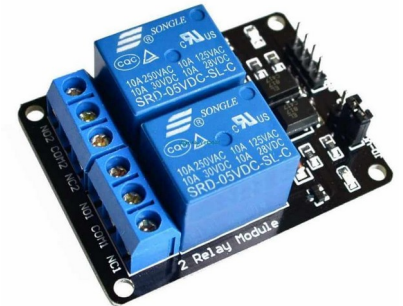
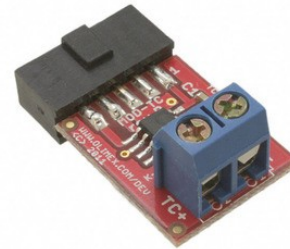
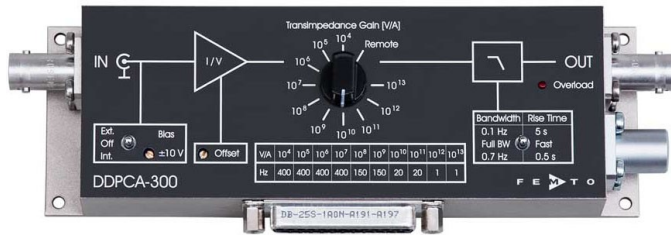
A project for the **MAX IV Laboratory**

39th Tango Community meeting at INAF

# The MCMAX project, introduction

In some occasions, it is necessary to interface very simple hardware components with the control system. For example:

- Devices with analog or digital GPIOs.
- Simple sensors using protocols like SPI or I2C
- Custom electronics or old equipment



# What is available

There are various ways to achieve this, most commonly:

Solution	Pros	Cons
Raspberry PI (or equivalent)	Cheap, Scientists make it themselves	Almost impossible to maintain
PLC based	Robust, Well integrated	Cost, uses PLC resources, Hard to install, deploy, update
Pandabox (or equivalent)	Great hardware capabilities, Well integrated	Cost

# Drawbacks of what is available

These systems are working well, but they have **drawbacks** for the common simple applications:

- Custom made solution are impossible to maintain or support.
- The PLC has a complex and lengthy deployment workflow.
- The fully customizable electronics systems are expensive and better employed in other applications.

# How to bridge this gap

A project was created:

- Define a standardized **microcontroller based** solution
- Fully integrated in the TANGO control system
- Which allows writing simple hardware functionality
- Using the same CI/CD workflow of the control system

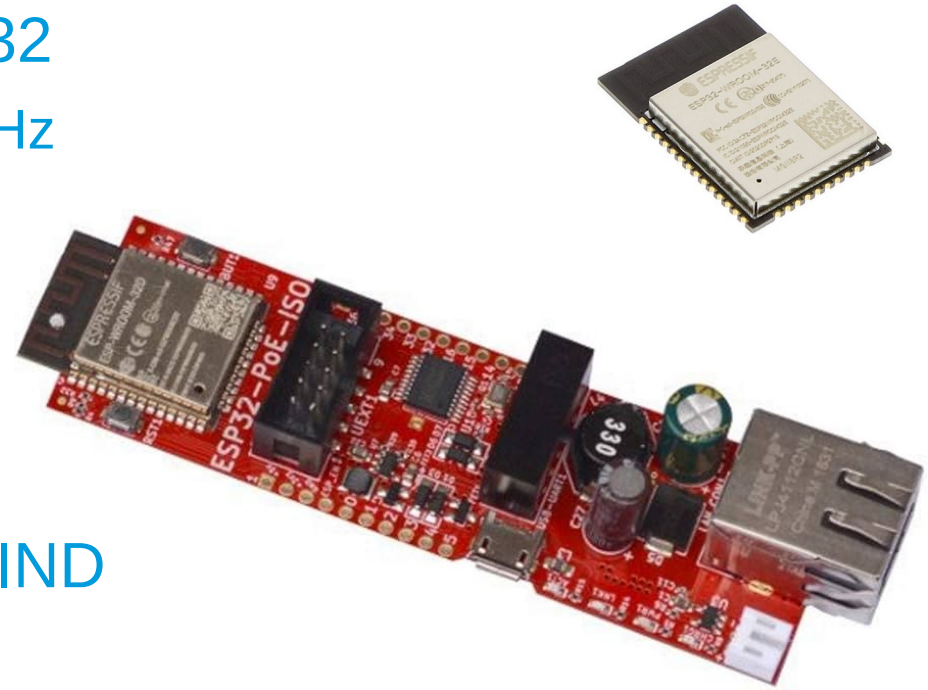
# How to achieve this

## Steps:

- 1) Select a commercial microcontroller and possibly a board.
- 2) Select/define a protocol to communicate to the control system.
- 3) Write libraries for the microcontroller to communicate in a standard way with a server.
- 4) Write a general tango device with dynamic attributes and commands

# The MCMAX electronics

- Microcontroller: ESPRESSIF ESP32
  - Dual core Xtensa LX6 ISA @240 MHz
  - 520 KiB RAM
  - 34 GPIOs\*
  - 4M Flash (or more)
  - OTA firmware update
- Board: OLIMEX ESP32-POE-ISO-IND
  - Open hardware
  - Ethernet with PoE



# The MCMAX protocol and libraries

- Protocol: **MQTT**
  - Lightweight
  - Capable
  - Open source implementations
  - Docker images for brokers
- **Libraries** and development environment: ESP-IDF
  - Same manufacturer as the SoC, can flash and debug
  - Available also as docker container (compile in CI)
  - Vscode and Eclipse plugin, LSP and command line interface
  - Modular (thanks to component registry), capable and open source
  - Available on <https://gitlab.com/ABartalesi/esp-mcmax>





# How to use the MCMAX libraries

To develop firmware on the esp-idf platform, a developer needs to:

- Start a project using esp-idf and git.
- Include the mc-max component, using standard espressif idf commands.
- Define a set of tango attributes and tango commands, using the C structures included in the mcmmax libraries.
- Define a callback for each writable attribute or command.
- Compile, flash and debug the microcontroller as usual.

The developer can really focus on the **firmware** only.  
(Tango, self description, MQTT, OTA, are handled by the library)



# MQTT clarification



10.0.0.1  
(client)



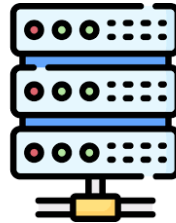
ESPRESSIF

```
#include esp-mqtt.h
```

Subscribe and publish



10.0.0.2  
(broker)

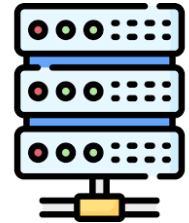


FlashMQ docker image

Subscribe and publish



10.0.0.3  
(client)



PyTango

```
pip install paho-mqtt pytango
```

# The MCMAX tango device server

## The **Tango device server** features:

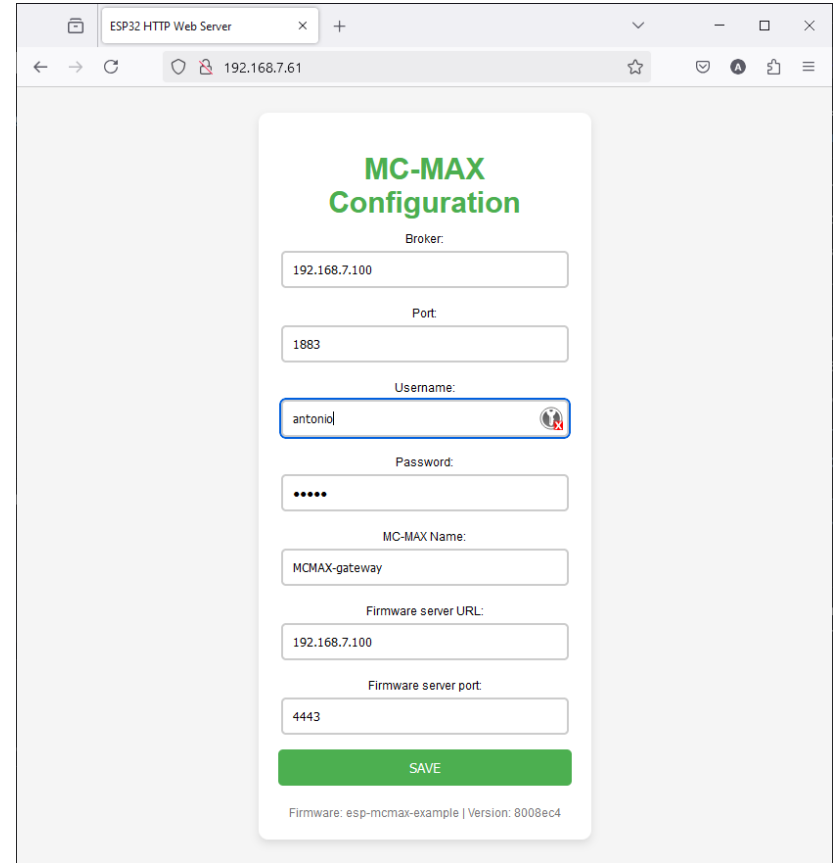
- One single tango device for all the different firmwares.
- One tango property to connect to the right device.
- During init, it listens to the micro self description, and creates attributes and commands dynamically.
- Reactive, thanks to MQTT protocol and Tango events.
- **No pytango development necessary** for a new project.
- Available on pip and conda:  
<https://gitlab.com/ABartalesi/mc-max-tango-device-server>



# Configuring the microcontroller

The microcontroller **hosts** a web server:

- Important configuration parameters:
  - MQTT broker address, port and credentials
  - MQTT name (must match tango property)
  - Firmware updates server address and port
  - Firmware name and version
- Settings are kept between firmware updates.



The screenshot shows a web browser window titled "ESP32 HTTP Web Server" with the address bar displaying "192.168.7.61". The main content is a form titled "MC-MAX Configuration" with the following fields:

- Broker: 192.168.7.100
- Port: 1883
- Username: antoniol
- Password: (masked with dots)
- MC-MAX Name: MCMAX-gateway
- Firmware server URL: 192.168.7.100
- Firmware server port: 4443

A green "SAVE" button is located below the fields. At the bottom of the form, it says "Firmware: esp-mcmax-example | Version: 8008ec4".

# CI/CD workflow

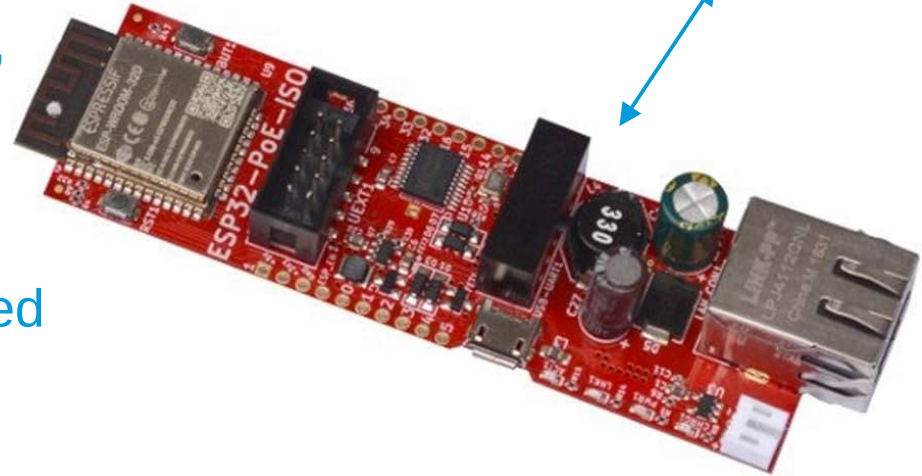
The current workflow:

- On new git commits, a new firmware image is compiled using esp-idf docker image.
- Firmware version is obtained with ``git describe``.
- A json file with firmware metadata is created.
- Firmware binary and metadata are uploaded to the firmware server.



# OTA firmware update

- The microcontroller contacts a web server (URL is configurable).
- This can be done either automatically or manually.
- If the version in the metadata file is newer, the microcontroller downloads the new firmware, installs it to the alternate partition and reboots.
- Configuration is preserved, since it is stored in a separate partition.



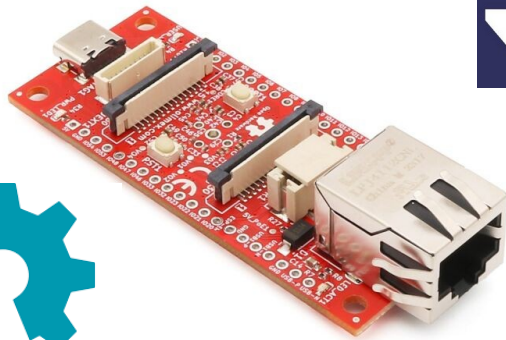
# Bonuses

Espressif is moving to more open source tools:

- Amazon version of FreeRTOS with n-core support.
- RISC-V based microcontrollers.



For the MCMAX project, the RISC-V chip ESP32-P4 has already been tested successfully!



# Thank you!

## Questions?

Links:

<https://gitlab.com/ABartalesi/esp-mcmax>

<https://gitlab.com/ABartalesi/mc-max-tango-device-server>

<https://bartalesi.eu/>

<mailto:antonio.bartalesi@gmail.com>