



UNESP – UNIVERSIDADE ESTADUAL PAULISTA
CAMPUS DE SOROCABA

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Disciplina: Redes Industriais de Comunicação

***Prática 7 – Comunicação entre Servidor OPC UA ESP32 e
Cliente (App e PC)***

Orientador: Prof. Eduardo Paciência Godoy

Aluno: Daniel Augusto Carneiro de Souza

Versão 1

Sorocaba

2021

Introdução

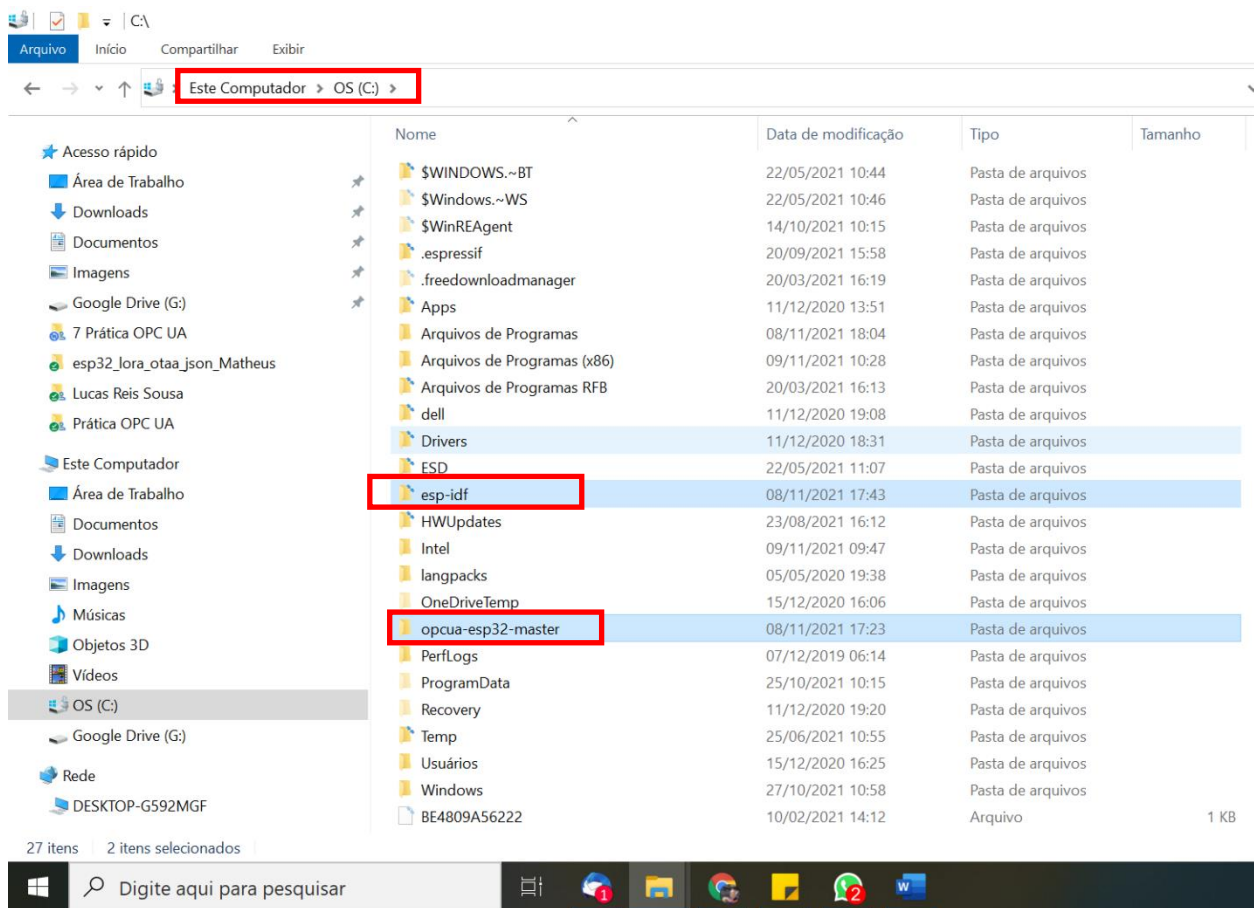
Esta prática tem como finalidade demonstrar as funcionalidades do padrão OPC UA através de um servidor OPC UA desenvolvido no ESP32, bem como realizar conexões com diversos dispositivos através do mecanismo de transporte de dados TCP/IP do OPC UA.

Material necessário

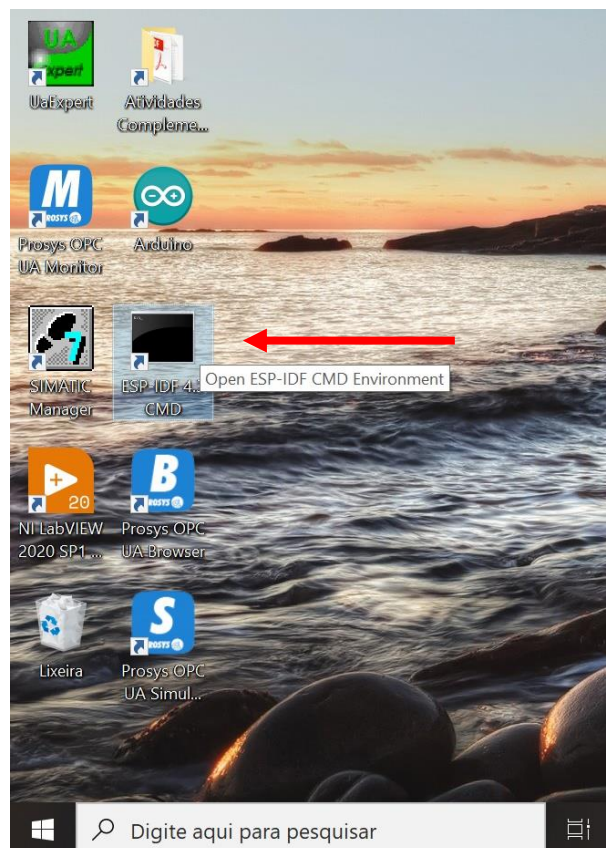
- UAExpt (Unified Automation) (instalador disponível na pasta de arquivos da Prática): <https://www.unified-automation.com/downloads/opc-ua-clients.html>;
- Aplicativo Prosys OPC UA Client (versão para Android: https://play.google.com/store/apps/details?id=com.prosysopc.ua.android2&hl=en_US&gl=US)

1. Configuração do Servidor OPC UA do ESP32

- Para criar o servidor, usaremos um código desenvolvido pelo professor e o aluno Daniel Souza em seu trabalho de TG. O código está pronto para ser compilado e gravado na ESP32, porém foi desenvolvido com o ambiente de desenvolvimento da Espressif (fabricante do ESP32), chamado de ESP-IDF;
- Para a instalação desse ambiente ESP-IDF, acessar procedimento detalhado no link: https://youtu.be/N5BnTShl_RY;
- Instalar arquivo “**esp-idf-tools-setup-2.3.exe**” disponível na pasta compartilhada da prática;
- Após a instalação, mover a pasta “**esp-idf**” criada na área de trabalho do seu computador para a raiz C: do seu computador (**C:\esp-idf**);
- Extrair a pasta compactada “**opc-esp32-master.rar**” disponível na pasta compartilhada da prática também para a raiz C: do seu computador (**C:\opcua-esp32-master**);



- Abrir ESP-IDF usando ícone criado na área de trabalho do seu computador



```

ESP-IDF 4.3 CMD - "C:\.espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef
Python 3.8.7
Using Git in C:\.espressif\tools\idf-git\2.30.1\cmd\
git version 2.30.1.windows.1
Setting IDF_PATH: C:\esp-idf

Adding ESP-IDF tools to PATH...
C:\.espressif\tools\xtensa-esp32-elf\esp-2020r3-8.4.0\xtensa-esp32-elf\bin
C:\.espressif\tools\xtensa-esp32s2-elf\esp-2020r3-8.4.0\xtensa-esp32s2-elf\bin
C:\.espressif\tools\xtensa-esp32s3-elf\esp-2020r3-8.4.0\xtensa-esp32s3-elf\bin
C:\.espressif\tools\riscv32-esp-elf\1.24.0.123_64eb9ff-8.4.0\riscv32-esp-elf\bin
C:\.espressif\tools\esp32ulp-elf\2.28.51-esp-20191205\esp32ulp-elf-binutils\bin
C:\.espressif\tools\esp32s2ulp-elf\2.28.51-esp-20191205\esp32s2ulp-elf-binutils\bin
C:\.espressif\tools\cmake\3.16.4\bin
C:\.espressif\tools\openocd-esp32\v0.10.0-esp32-20210401\openocd-esp32\bin
C:\.espressif\tools\ninja\1.10.2\
C:\.espressif\tools\idf-exe\1.0.1\
C:\.espressif\tools\ccache\3.7\
C:\.espressif\tools\dfu-util\0.9\dfu-util-0.9-win64
C:\esp-idf\tools

Checking if Python packages are up to date...
Python requirements from C:\esp-idf\requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

idf.py build

C:\esp-idf>

```

- Inserir comando para acessar pasta do arquivo: **cd “endereço do arquivo”**. No nosso caso digitar: **cd “C:\opcua-esp32-master”**

```

ESP-IDF 4.3 CMD - "C:\.espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef

C:\esp-idf>cd "C:\opcua-esp32-master"

C:\opcua-esp32-master>idf.py menuconfig
Executing action: menuconfig
Running ninja in directory c:\opcua-esp32-master\build
Executing "ninja menuconfig"...
[0/1] Re-running CMake...
-- ccache will be used for faster recompilation
-- Project is not inside a git repository, or git repository has no commits; will not use 'git describe' to determine PROJECT_VERSION.
-- Building ESP-IDF components for target esp32
-- Project sdkconfig file C:/opcua-esp32-master/sdkconfig
-- Could NOT find Perl (missing: PERL_EXECUTABLE)
CMake Warning (dev) at C:/esp-idf/components/mbedtls/CMakeLists.txt:114 (target_sources):
  Policy CMP0076 is not set: target_sources() command converts relative paths
  to absolute. Run "cmake --help-policy CMP0076" for policy details. Use
  the cmake_policy command to set the policy and suppress this warning.

  A private source from a directory other than that of target "mbedcrypto"
  has a relative path.
This warning is for project developers. Use -Wno-dev to suppress it.

-- App "opcua_esp32" version: 1
-- Adding linker script C:/esp-idf/components/esp_rom/esp32/ld/esp32.rom.ld
-- Adding linker script C:/esp-idf/components/esp_rom/esp32/ld/esp32.rom.api.ld
-- Adding linker script C:/esp-idf/components/esp_rom/esp32/ld/esp32.rom.libgcc.ld

```

- Acessar menu de configuração do ESP32 na ESP=IDF com o comando: **idf.py menuconfig**

```

ESP-IDF 4.3 CMD - "C:\espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef
C:\espressif\tools\esp32s2ulp-elf\2.28.51-esp-20191205\esp32s2ulp-elf-binutils\bin
C:\espressif\tools\cmake\3.16.4\bin
C:\espressif\tools\openocd-esp32\v0.10.0-esp32-20210401\openocd-esp32\bin
C:\espressif\tools\ninja\1.10.2\
C:\espressif\tools\idf-exe\1.0.1\
C:\espressif\tools\ccache\3.7\
C:\espressif\tools\dfu-util\0.9\dfu-util-0.9-win64
C:\esp-idf\tools

Checking if Python packages are up to date...
Python requirements from C:\esp-idf\requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

idf.py build

C:\esp-idf>cd "C:\opcua-esp32-master"
C:\opcua-esp32-master>idf.py menuconfig
Executing action: menuconfig
Running ninja in directory c:\opcua-esp32-master\build
Executing "ninja menuconfig"...
[0/1] cmd.exe /C "cd /D C:\opcua-esp32-master\build && C:\...ENV_FPGA= --output config C:/opcua-esp32-master/sdkconfig"
C:/esp-idf/Kconfig:14: warning: IDF_ENV_FPGA has 'option env="IDF_ENV_FPGA"', but the environment variable IDF_ENV_FPGA
is not set
Loaded configuration 'C:/opcua-esp32-master/sdkconfig'
No changes to save (for 'C:/opcua-esp32-master/sdkconfig')

```

- Com a tela de configuração aberta, acessar “*Connection Configuration*” para conectar-se a sua rede Wi-Fi (atualmente o ESP só suporta redes 2,4 GHz);

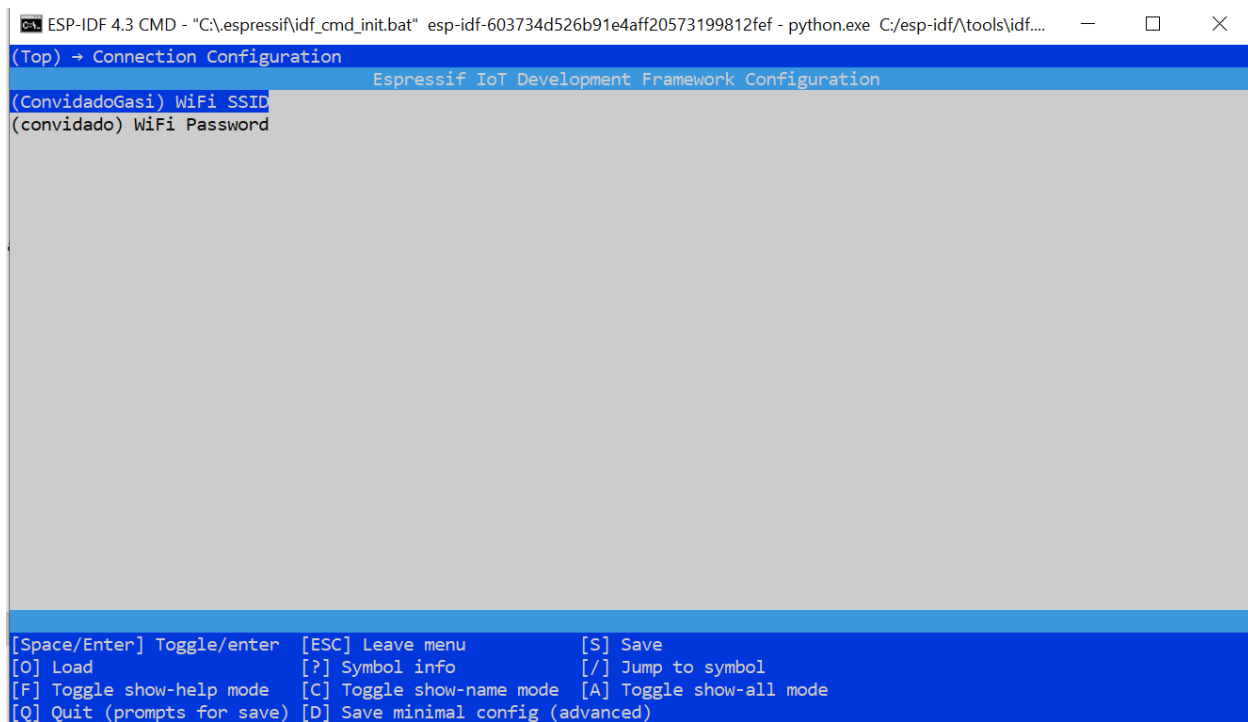
```

ESP-IDF Command Prompt (cmd.exe) - "C:\Users\rosah\espressif\idf_cmd_init.bat" "C:\Users\rosah\AppData\Local\Programs\Python\Python37\..."
(Top)
Espressif IoT Development Framework Configuration
SDK tool configuration --->
Build type --->
Application manager --->
Bootloader config --->
Security features --->
Serial flasher config --->
Partition Table --->
Connection Configuration --->
Compiler options --->
Component config --->
Compatibility options --->

[Space/Enter] Toggle/enter  [ESC] Leave menu          [S] Save
[O] Load                    [?] Symbol info           [/] Jump to symbol
[F] Toggle show-help mode    [C] Toggle show-name mode [A] Toggle show-all mode
[Q] Quit (prompts for save)  [D] Save minimal config (advanced)

```

- Digite nome e senha da sua rede. Depois salve e carregue no ESP;



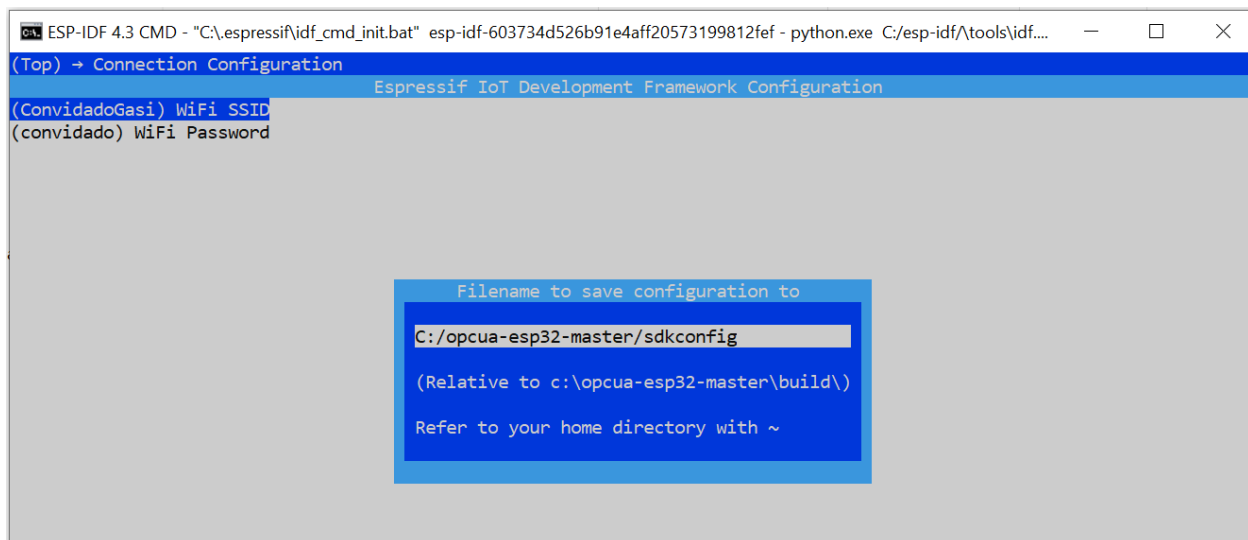
ESP-IDF 4.3 CMD - "C:\espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef - python.exe C:/esp-idf/tools/idf...

(Top) → Connection Configuration

Espressif IoT Development Framework Configuration

(ConvidadoGasi) WiFi SSID
(convidado) WiFi Password

[Space/Enter] Toggle/enter [ESC] Leave menu [S] Save
[O] Load [?] Symbol info [/] Jump to symbol
[F] Toggle show-help mode [C] Toggle show-name mode [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)



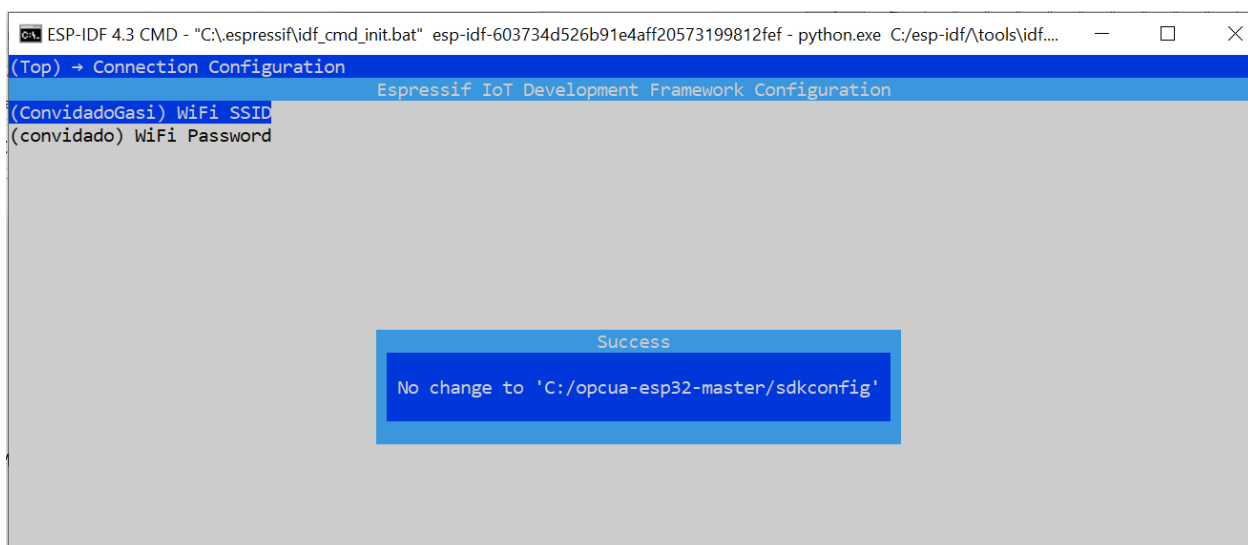
ESP-IDF 4.3 CMD - "C:\espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef - python.exe C:/esp-idf/tools/idf...

(Top) → Connection Configuration

Espressif IoT Development Framework Configuration

(ConvidadoGasi) WiFi SSID
(convidado) WiFi Password

Filename to save configuration to
C:/opcua-esp32-master/sdkconfig
(Relative to c:\opcua-esp32-master\build\
Refer to your home directory with ~



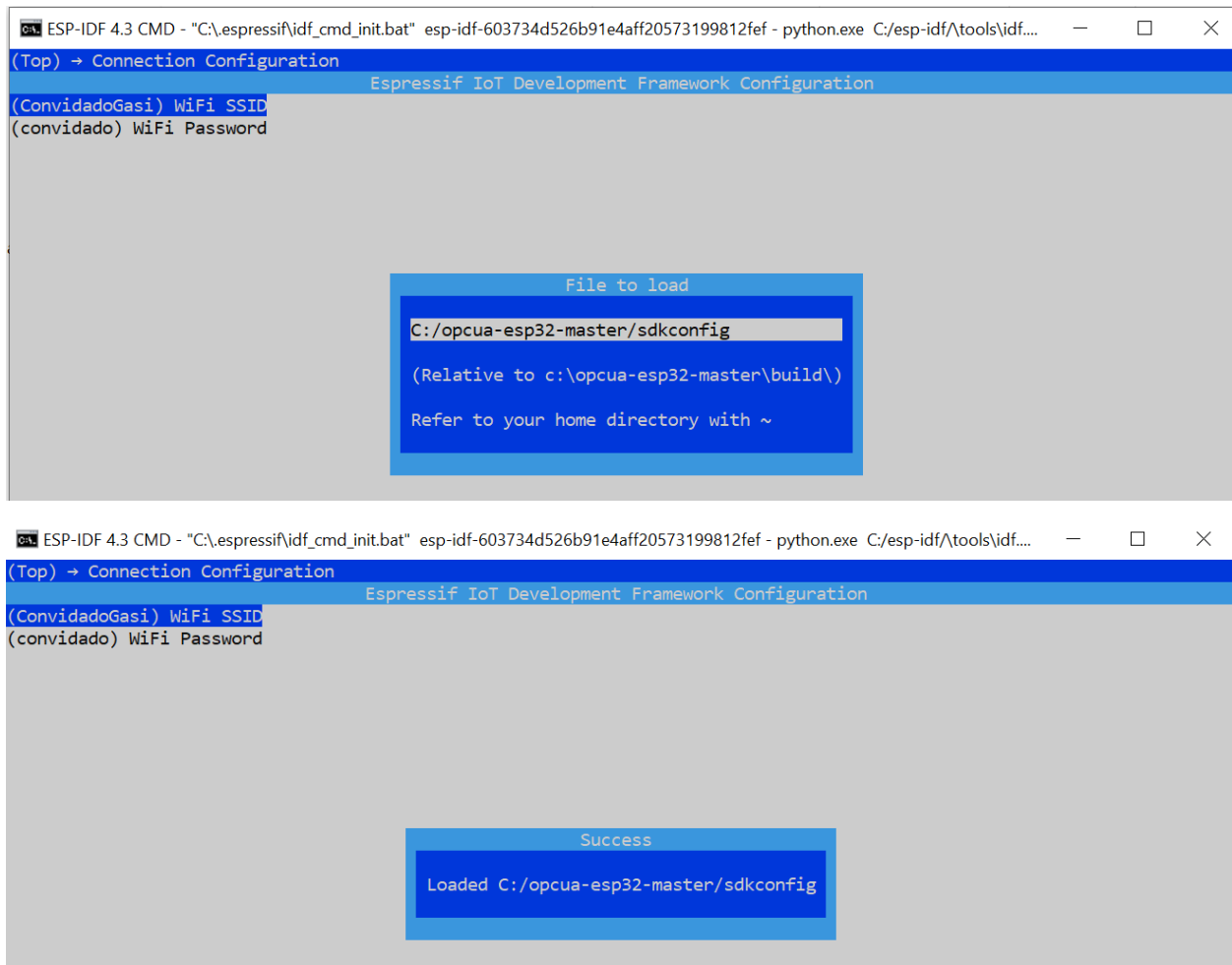
ESP-IDF 4.3 CMD - "C:\espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef - python.exe C:/esp-idf/tools/idf...

(Top) → Connection Configuration

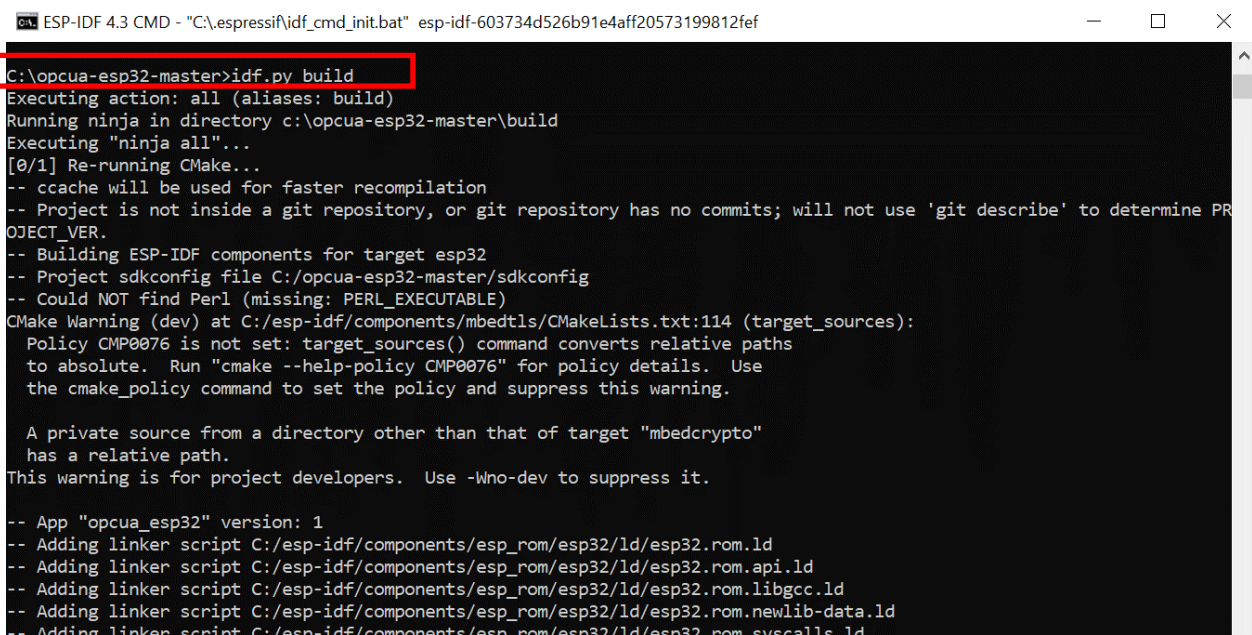
Espressif IoT Development Framework Configuration

(ConvidadoGasi) WiFi SSID
(convidado) WiFi Password

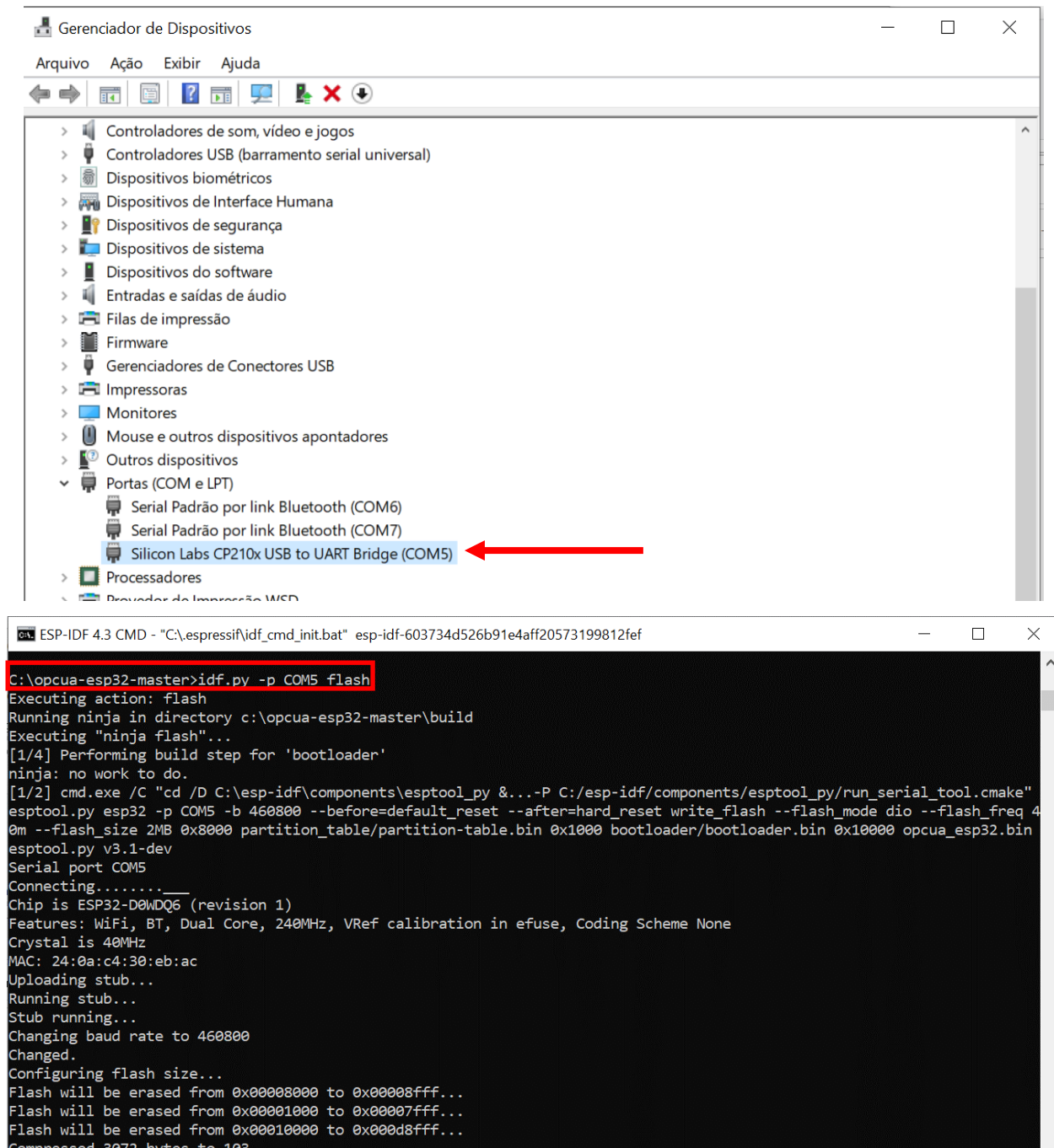
Success
No change to 'C:/opcua-esp32-master/sdkconfig'



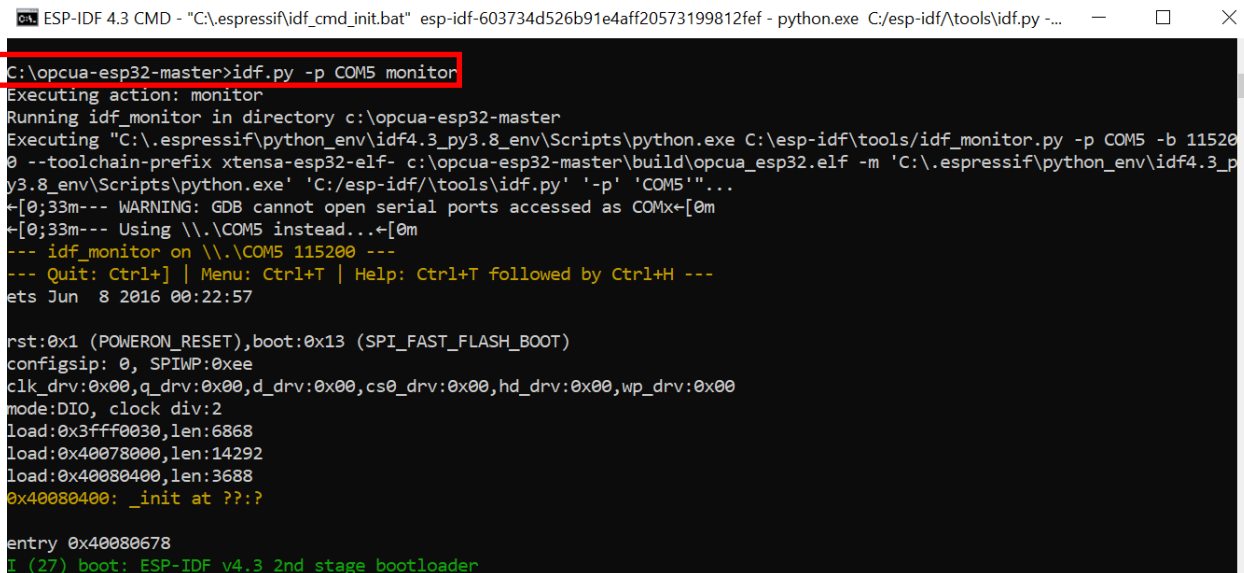
- Com as configurações realizadas, aperte “ESC” até sair do menu.
- Para compilar o programa, digite o comando: **idf.py build**



- Após compilação, para gravação do código compilado na ESP32, digite o comando: **idf.py -p COMX flash**. A porta COMX representa sua porta COM criada após a conexão do ESP32 na USB no computador. Acesse o Gerenciador de Dispositivos do seu Windows para verificar qual a porta COM.



- Após a programação do ESP32, para verificar a operação, digite o comando: **idf.py -p COMX monitor**. A porta COMX representa sua porta COM criada após a conexão do ESP32 na USB no computador. Caso falhe a conexão automática entre o IDF e a ESP32, tente realizar o comando novamente e apertar o botão BOOT/PRG da ESP32 se aparecer a mensagem “Conecting...” na tela;

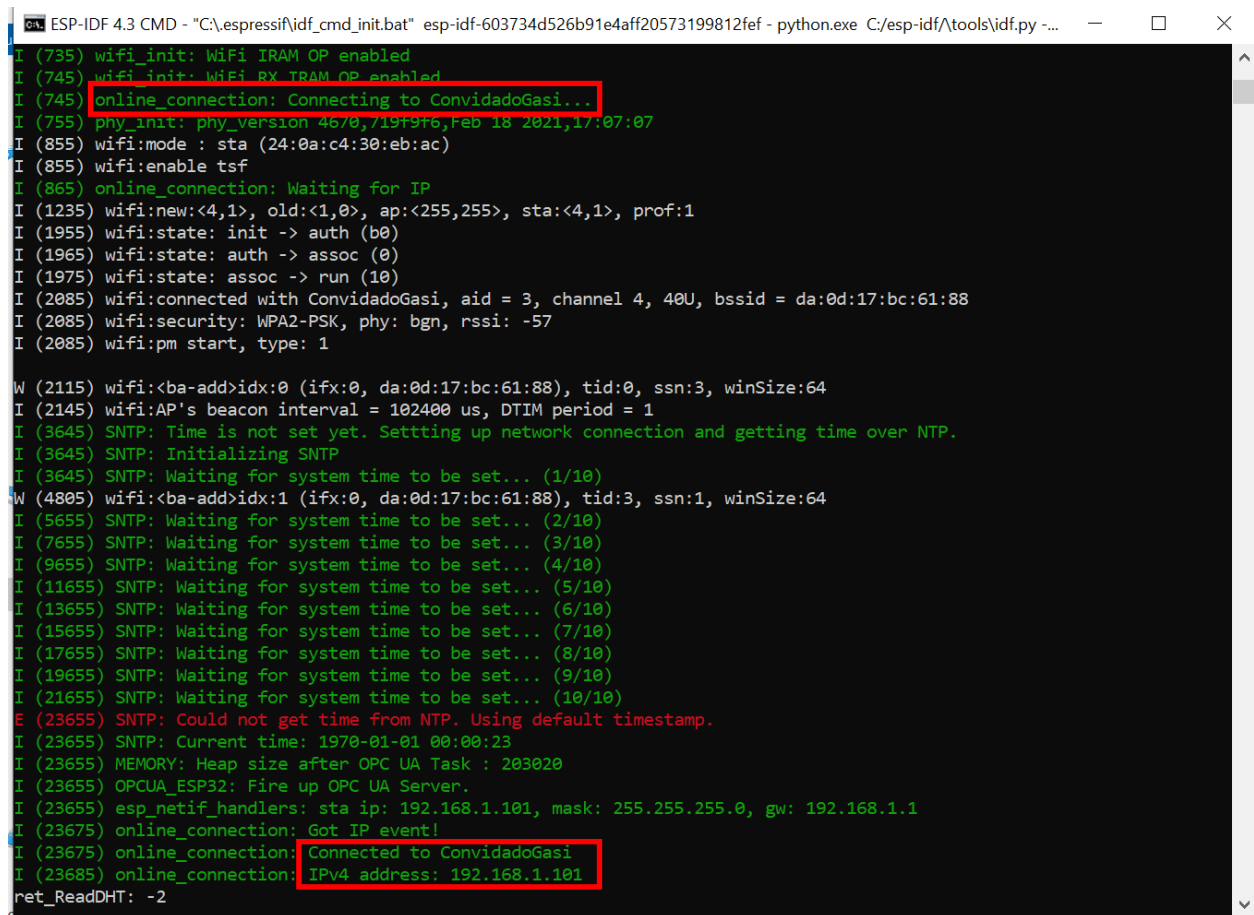


```
ESP-IDF 4.3 CMD - "C:\espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef - python.exe C:/esp-idf/tools/idf.py ...
C:\opcua-esp32-master>idf.py -p COM5 monitor
Executing action: monitor
Running idf_monitor in directory c:\opcua-esp32-master
Executing "C:\.espressif\python_env\idf4.3_py3.8_env\Scripts\python.exe C:\esp-idf\tools\idf_monitor.py -p COM5 -b 115200 --toolchain-prefix xtensa-esp32-elf- c:\opcua-esp32-master\build\opcua_esp32.elf -m 'C:\.espressif\python_env\idf4.3_py3.8_env\Scripts\python.exe' 'C:/esp-idf/tools/idf.py' '-p' 'COM5'"...
+ [0;33m--- WARNING: GDB cannot open serial ports accessed as COMx+ [0m
+ [0;33m--- Using \\.\COM5 instead...+ [0m
--- idf_monitor on \\.\COM5 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:6868
load:0x40078000,len:14292
load:0x40080400,len:3688
0x40080400: _init at ???:?

entry 0x40080678
T (27) boot: ESP-IDF v4.3 2nd stage bootloader
```

- Com a comunicação estabelecida, diversas informações são apresentadas, entre elas o endereço IP do servidor OPC UA da ESP32 que deverá ser usado para conexão:

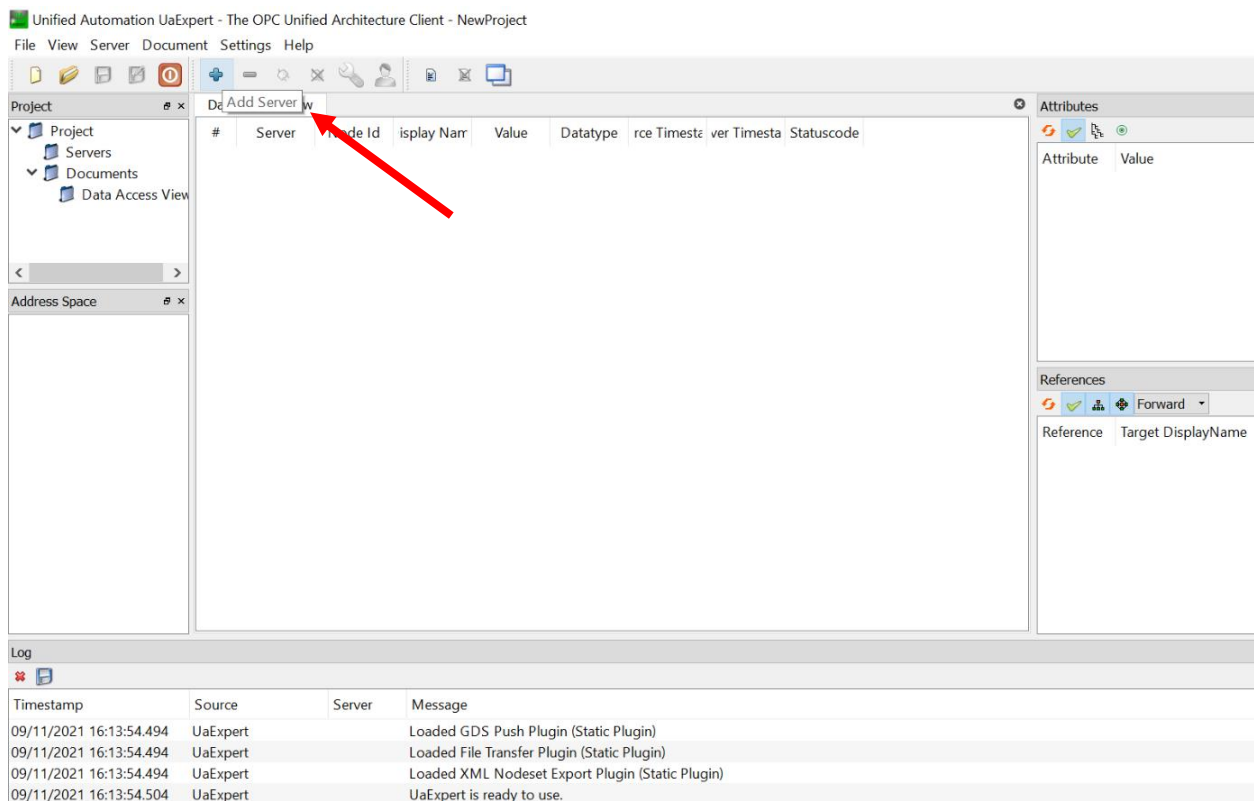


```
ESP-IDF 4.3 CMD - "C:\espressif\idf_cmd_init.bat" esp-idf-603734d526b91e4aff20573199812fef - python.exe C:/esp-idf/tools/idf.py ...
I (735) wifi_init: WiFi IRAM OP enabled
I (745) wifi_init: WiFi RX IRAM OP enabled
I (745) online_connection: Connecting to ConvidadoGasi...
I (755) phy_init: phy_version 4670,719+9t6, Feb 18 2021, 17:07:07
I (855) wifi:mode : sta (24:0a:c4:30:eb:ac)
I (855) wifi:enable tsf
I (865) online_connection: Waiting for IP
I (1235) wifi:new:<4,1>, old:<1,0>, ap:<255,255>, sta:<4,1>, prof:1
I (1955) wifi:state: init -> auth (b0)
I (1965) wifi:state: auth -> assoc (0)
I (1975) wifi:state: assoc -> run (10)
I (2085) wifi:connected with ConvidadoGasi, aid = 3, channel 4, 40U, bssid = da:0d:17:bc:61:88
I (2085) wifi:security: WPA2-PSK, phy: bgn, rssi: -57
I (2085) wifi:pm start, type: 1

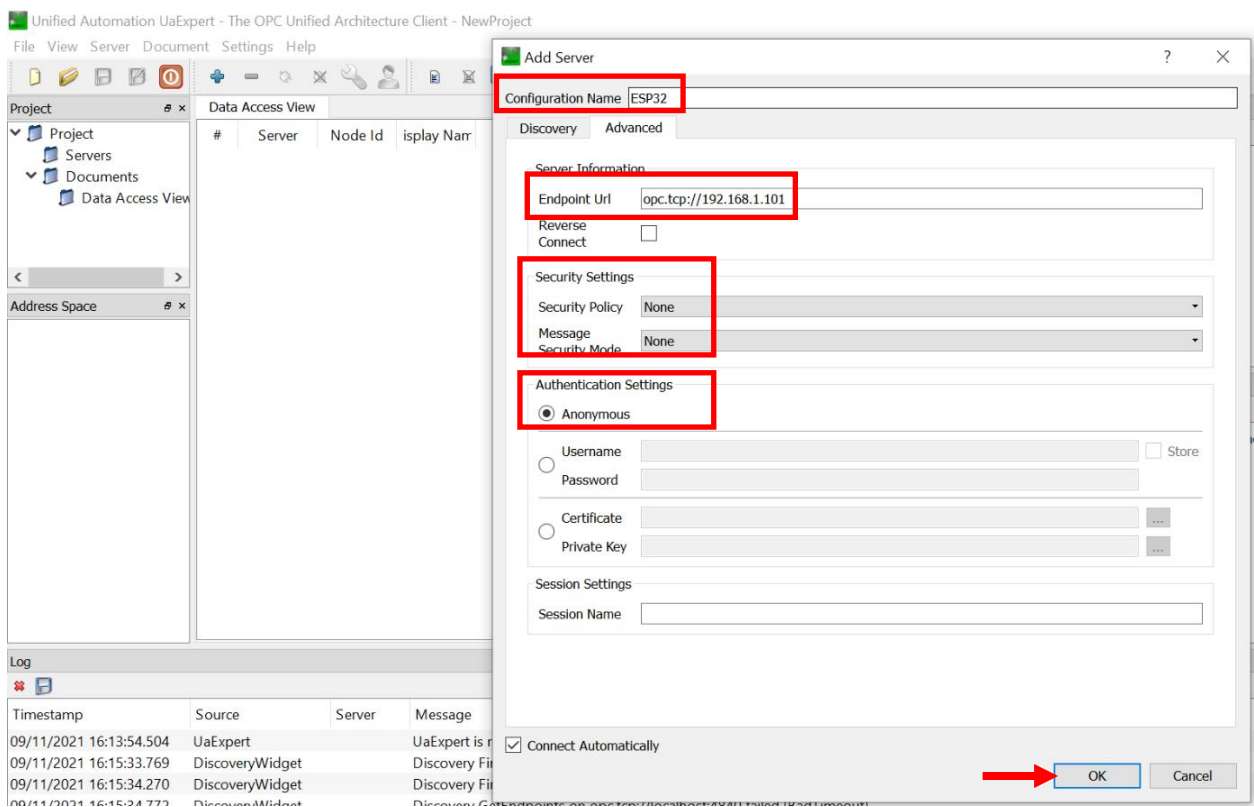
W (2115) wifi:<ba-add>idx:0 (ifx:0, da:0d:17:bc:61:88), tid:0, ssn:3, winSize:64
I (2145) wifi:AP's beacon interval = 102400 us, DTIM period = 1
I (3645) SNTP: Time is not set yet. Setting up network connection and getting time over NTP.
I (3645) SNTP: Initializing SNTP
I (3645) SNTP: Waiting for system time to be set... (1/10)
W (4805) wifi:<ba-add>idx:1 (ifx:0, da:0d:17:bc:61:88), tid:3, ssn:1, winSize:64
I (5655) SNTP: Waiting for system time to be set... (2/10)
I (7655) SNTP: Waiting for system time to be set... (3/10)
I (9655) SNTP: Waiting for system time to be set... (4/10)
I (11655) SNTP: Waiting for system time to be set... (5/10)
I (13655) SNTP: Waiting for system time to be set... (6/10)
I (15655) SNTP: Waiting for system time to be set... (7/10)
I (17655) SNTP: Waiting for system time to be set... (8/10)
I (19655) SNTP: Waiting for system time to be set... (9/10)
I (21655) SNTP: Waiting for system time to be set... (10/10)
E (23655) SNTP: Could not get time from NTP. Using default timestamp.
I (23655) SNTP: Current time: 1970-01-01 00:00:23
I (23655) MEMORY: Heap size after OPC UA Task : 203020
I (23655) OPCUA_ESP32: Fire up OPC UA Server.
I (23655) esp_netif_handlers: sta ip: 192.168.1.101, mask: 255.255.255.0, gw: 192.168.1.1
I (23675) online_connection: Got IP event!
I (23675) online_connection: Connected to ConvidadoGasi
I (23685) online_connection: IPv4 address: 192.168.1.101
ret_ReadDHT: -2
```

2. Configuração do Cliente OPC UA

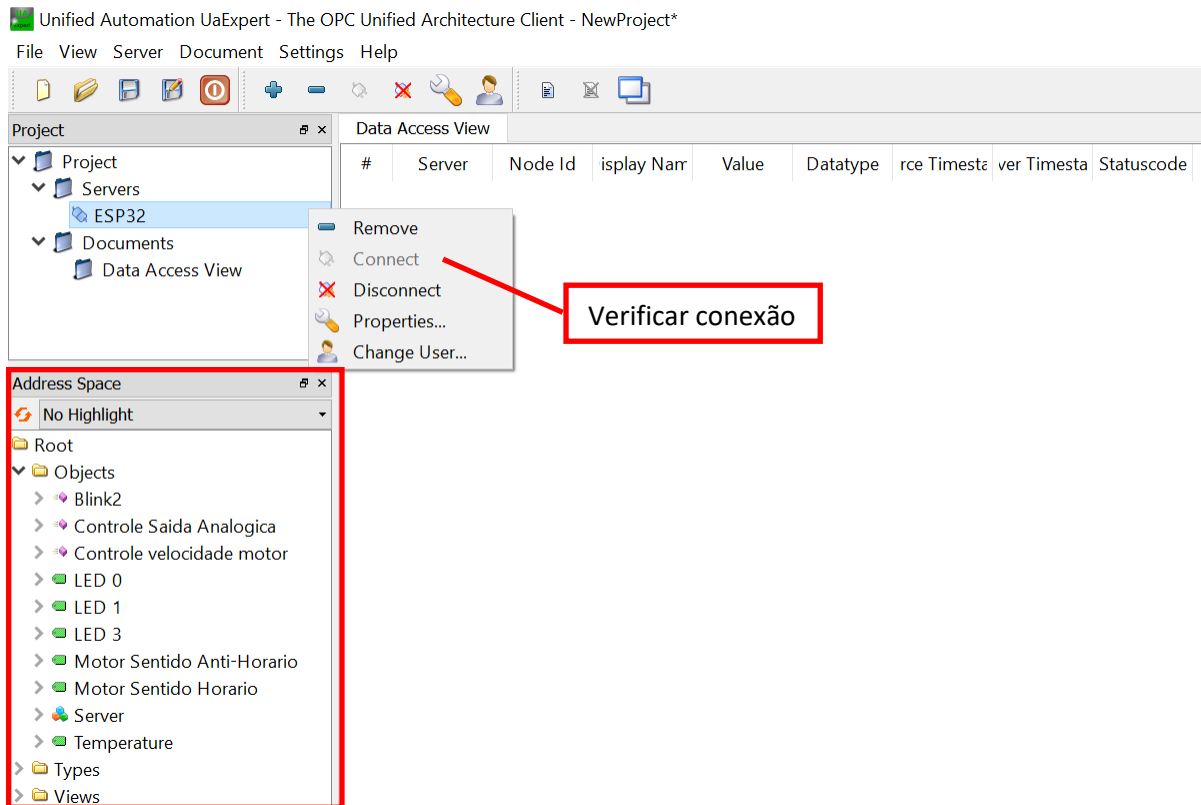
- Abra o aplicativo UAExpert e clique no ícone “+” para adicionar um servidor OPC UA;



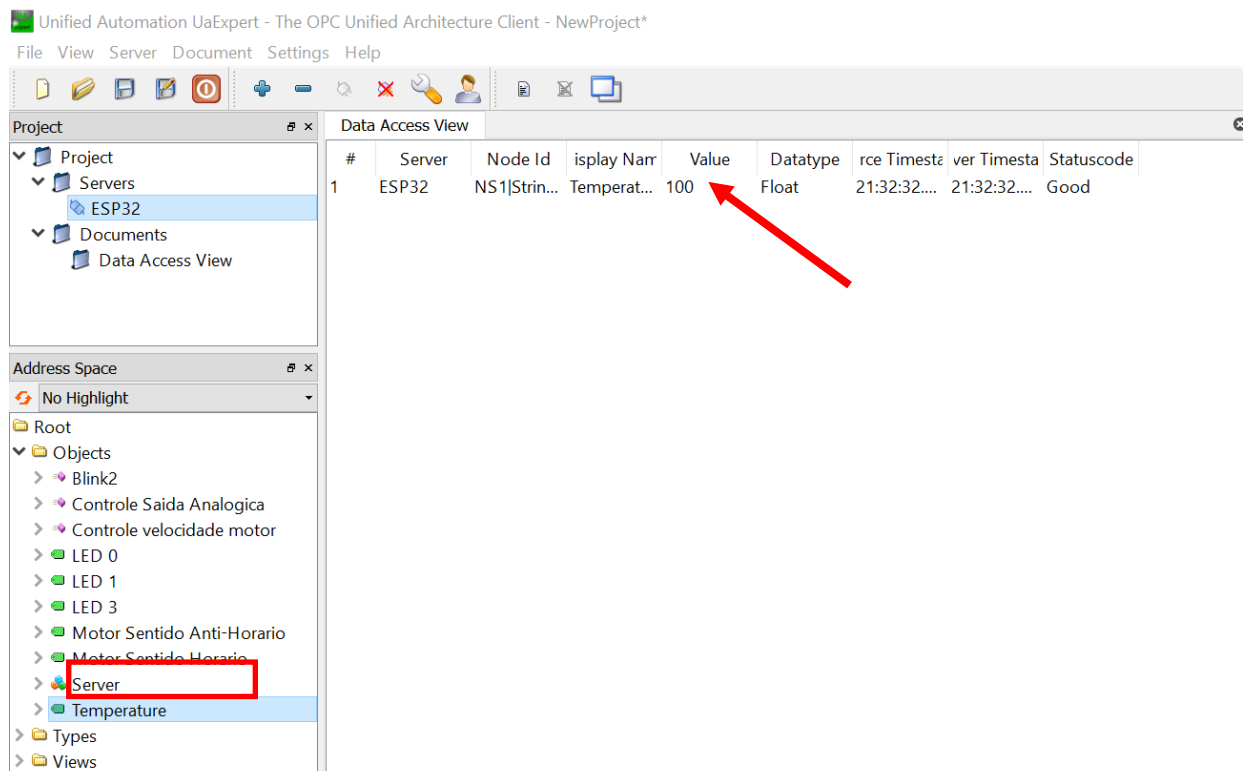
- Selecione a aba “Advanced” de configuração e digite os dados do servidor OPC UA do ESP32. Clique em OK para confirmar.



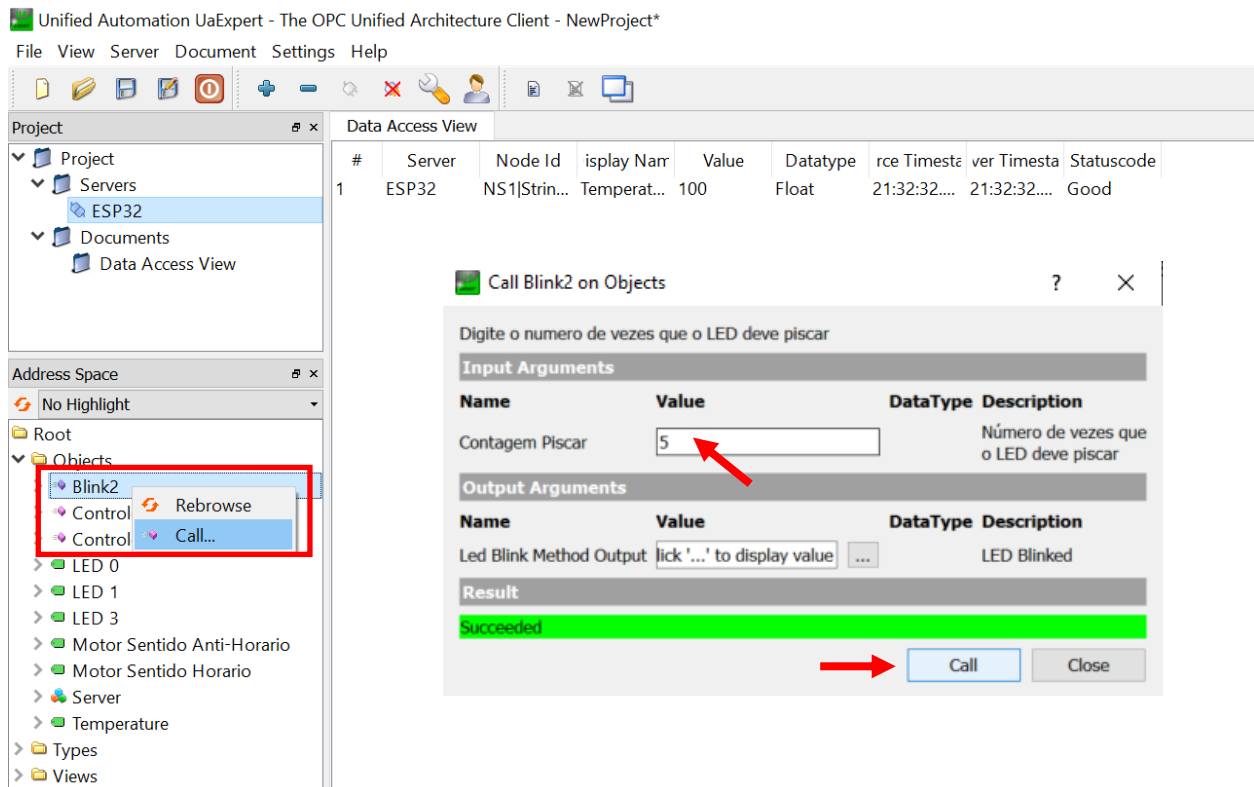
- O Cliente UAExpert deve se conectar ao servidor OPC UA do ESP32. Visualize as informações do Espaço de Endereço (variáveis Tag) do servidor OPC UA:



- Para monitorar (fazer a leitura de uma entrada) de uma Tag, clique sobre a variável na área do Espaço de Endereço e puxe a variável para a área do “Data Access View”. Por exemplo, o valor da *Temperature* pode ser visualizado no campo “Value”;



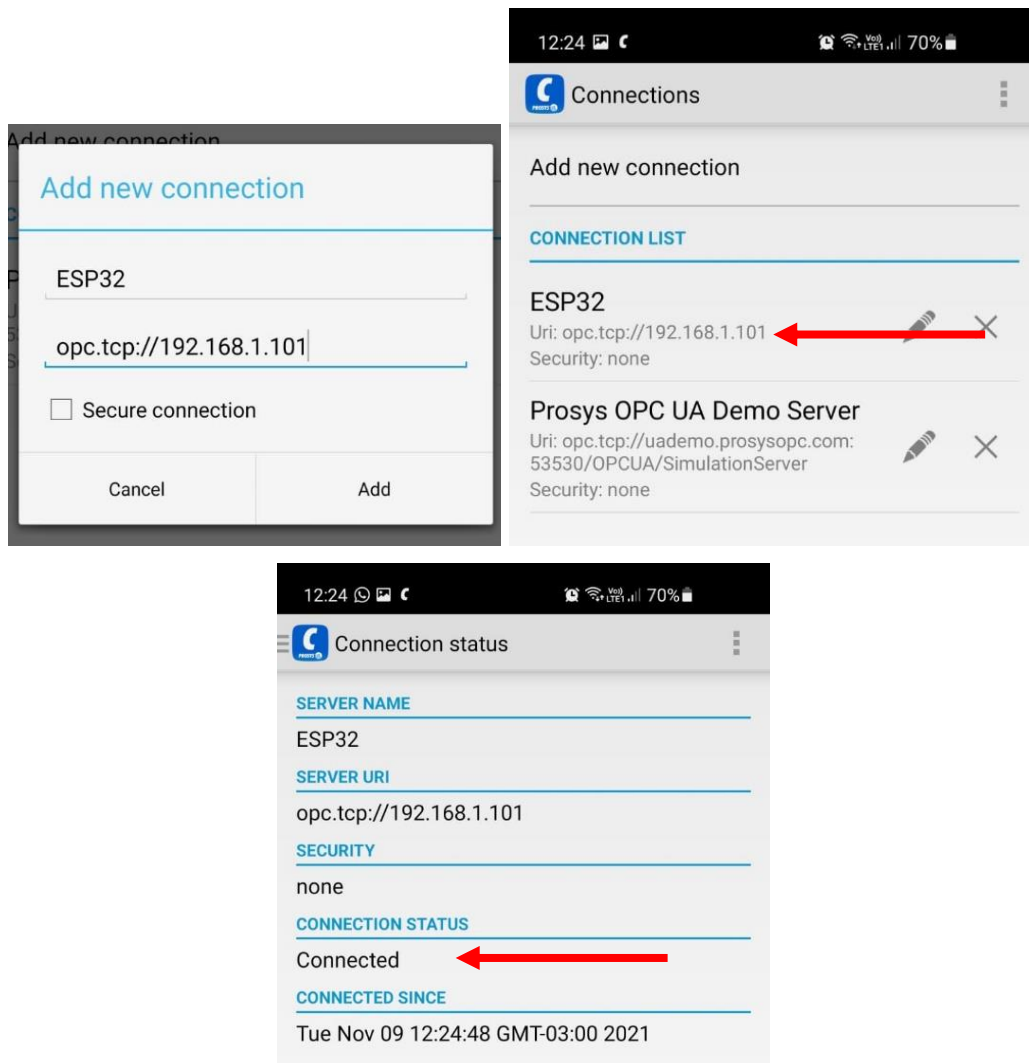
- Para escrever (atualizar uma saída) de uma Tag, clique sobre o nome da variável no Espaço de Endereço e selecione “Call”. Por exemplo, para escrever um valor no Tag “Blink2” para comandar a piscagem (acender/apagar) do LED interno do ESP32, selecione “Call” nessa variável. Na caixa de diálogo digite um novo valor (campo Contagem Piscar) e clique em “Call”:



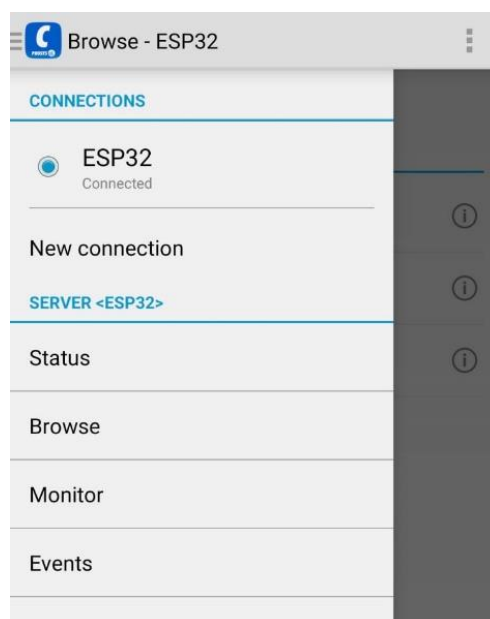
Após o comando, verifique que o LED interno do ESP32 pisca conforme comandado.

3. Aplicativo Cliente OPC UA para celular

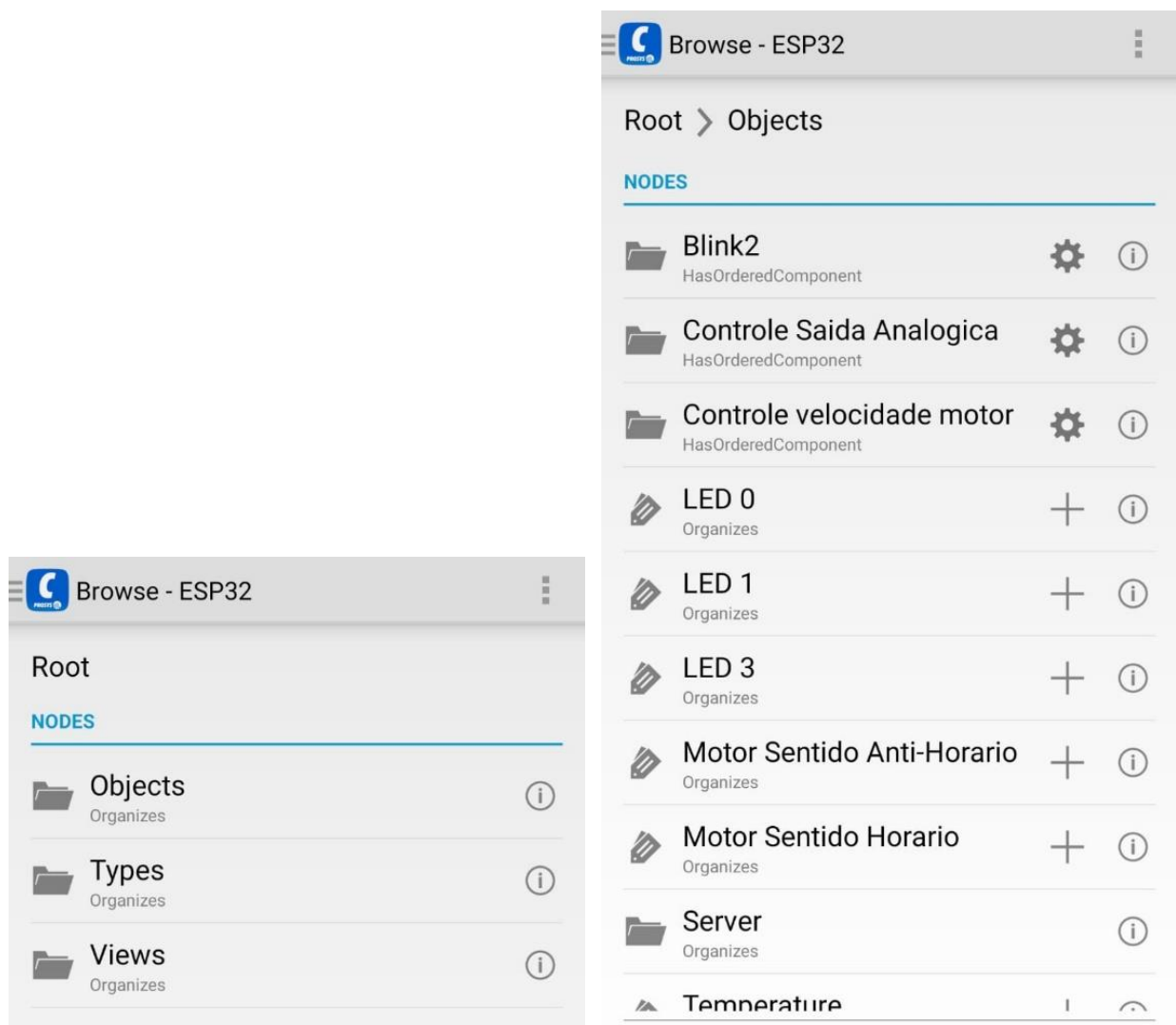
- No celular, abra o aplicativo Prosys OPC UA Client e selecione a opção “Add new connection”;
- Digite o endereço do servidor (o mesmo utilizado no UAExpert)
- Verifique a conexão entre o Cliente e Servidor;



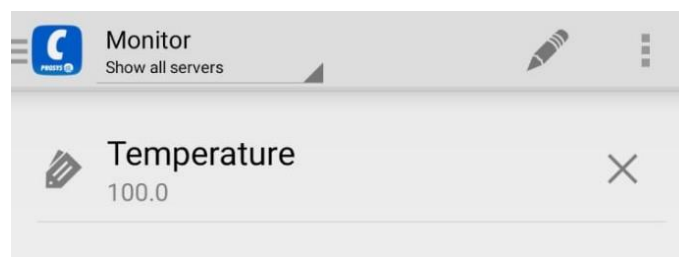
- Clique sobre o botão “*Connection Status*” no canto superior esquerdo e em seguida selecione a opção “*Browse*”:



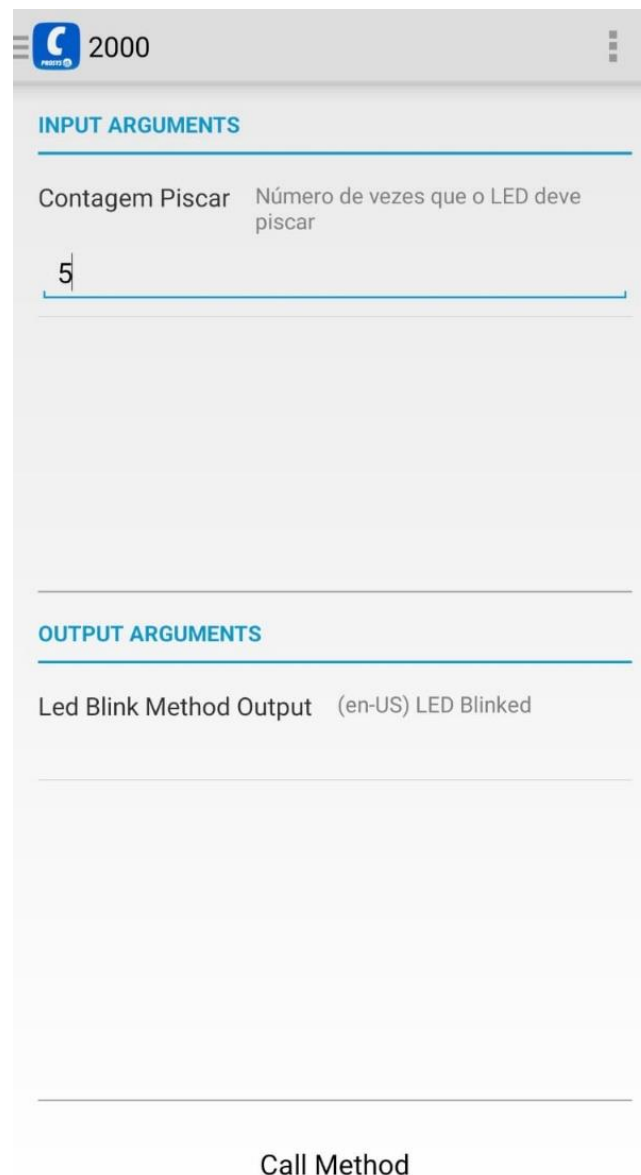
- Acesse as pastas *Objects* para ver os *Tags* disponíveis:



- Para monitorar (fazer a leitura de uma entrada) de uma Tag, clique no sinal “+” ao lado da variável para habilitar o monitoramento. Por exemplo, clique ao lado da variável *Temperature*;
- Abra novamente o menu (canto superior esquerdo) e clique em “*Monitor*” e observe o valor do Tag “*Temperature*”;



- Para escrever (atualizar uma saída) de uma Tag, clique sobre o ícone da Engrenagem de uma das variáveis de saída. Por exemplo, para escrever um valor no Tag “Blink2” para comandar a piscagem (acender/apagar) do LED interno do ESP32, dê um clique sobre a engrenagem ao lado do nome do Tag e digite o novo valor na caixa de diálogo e clique em “Call Method”:



Após o comando, verifique que o LED interno do ESP32 pisca conforme comandado.