



Universidade Federal da Bahia
Escola Politécnica
2022.2

Sistema supervisório com Node-RED via Modbus

Discentes:

André Paiva

Caroline Alves

Jeisiane Macedo

Pedro Augusto Correia

Salvador, 01 de dezembro de 2022

Introdução

Nos últimos anos, presenciamos uma crescente procura das empresas pela adoção da automatização de processos.

Tal automatização promove redução de custos e aumento na produtividade.

Neste contexto, destaca-se a importância dos sistemas supervisórios.

Importância de Sistemas Supervisórios

- Monitoramento de dados de um processo
- Controle de dados de um processo
- Aumento da produtividade
- Redução de custos

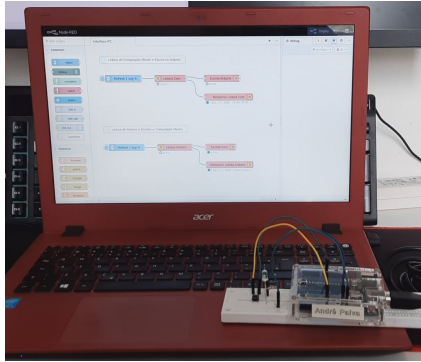
Projeto desenvolvido

- Prova de conceito de um sistema supervisorio
- Demonstração de funcionamento e configuração

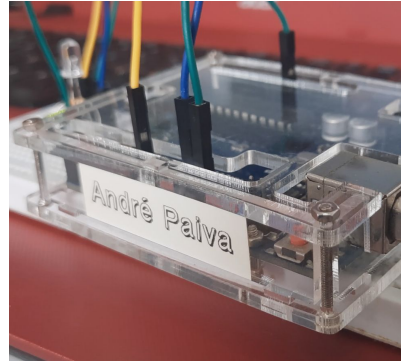
Componentes de Hardware



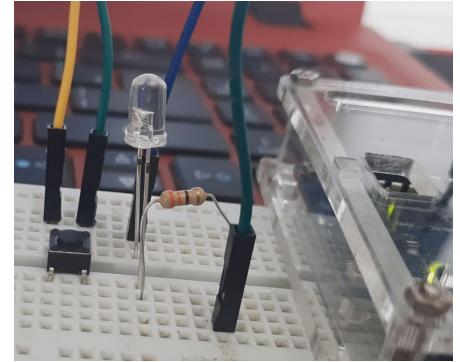
Core PC



Interface PC

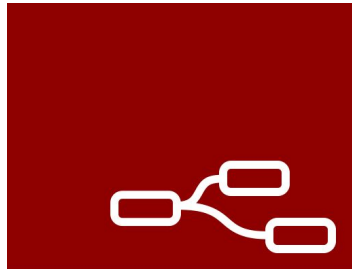


Arduino



Circuito

Componentes de Software



Node-RED

Supervisório/IHM
e CLP



Comunicação entre
Supervisório e CLP



Comunicação CLP e
arduino

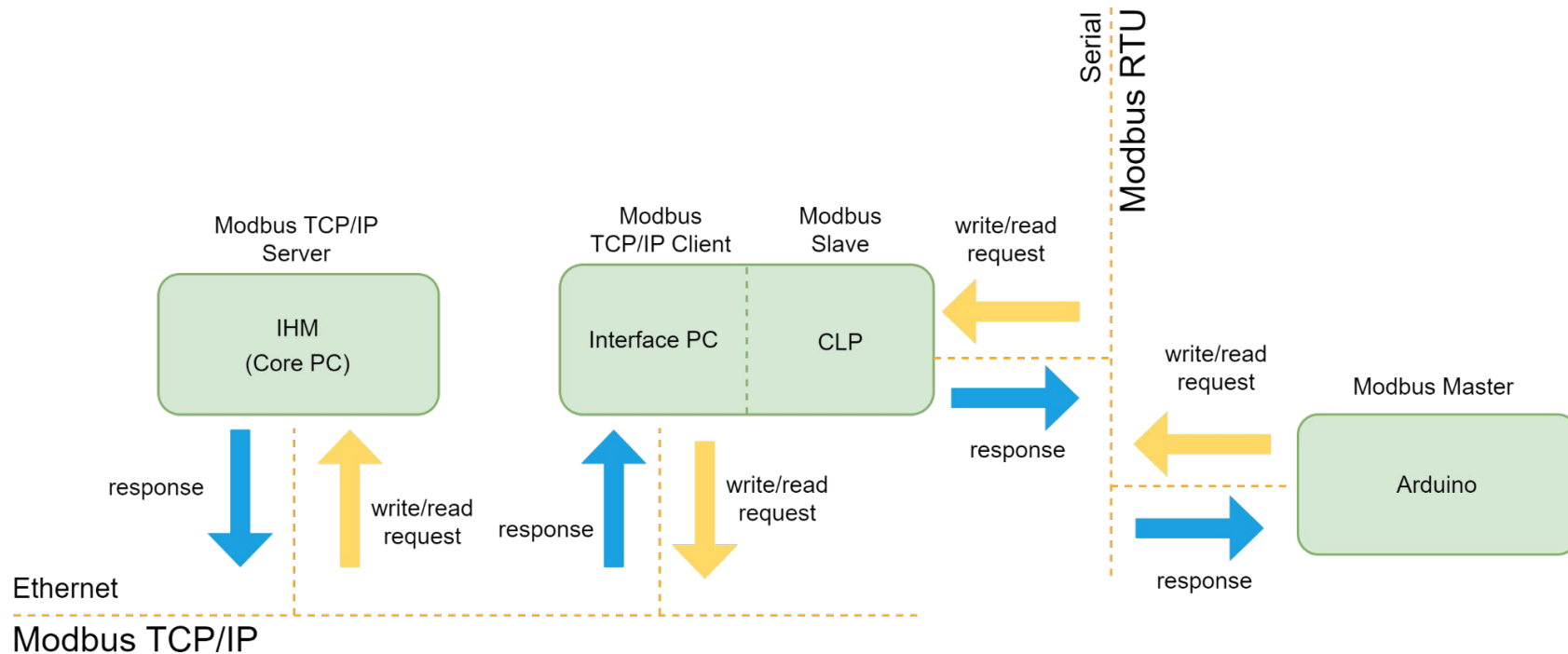
Node-RED

- Ferramenta versátil que facilita a criação de aplicações IoT desenvolvida pela IBM.
- Possui uma abordagem gráfica para programação baseada em fluxos através do navegador.
- As conexões são realizadas por nós com propósitos definidos de execução. Cada nó recebe o dado, o processa de acordo com sua função e passa adiante.
- O projeto de fluxo pode ser exportado via JSON.

Modbus

- É um protocolo aberto para transmissão de informações entre dispositivos, via portas serial ou Ethernet, desenvolvido pela Modicon e pertencente atualmente à Schneider.
- É um dos protocolos mais utilizados na automação industrial devido à sua facilidade de implementação.
- Arquitetura Mestre/Escravo (ou Cliente/Servidor).
- A transmissão pode ser feita em modo RTU ou ASCII. Esses modos não se comunicam entre si. A versão TCP/IP utiliza o modo RTU.

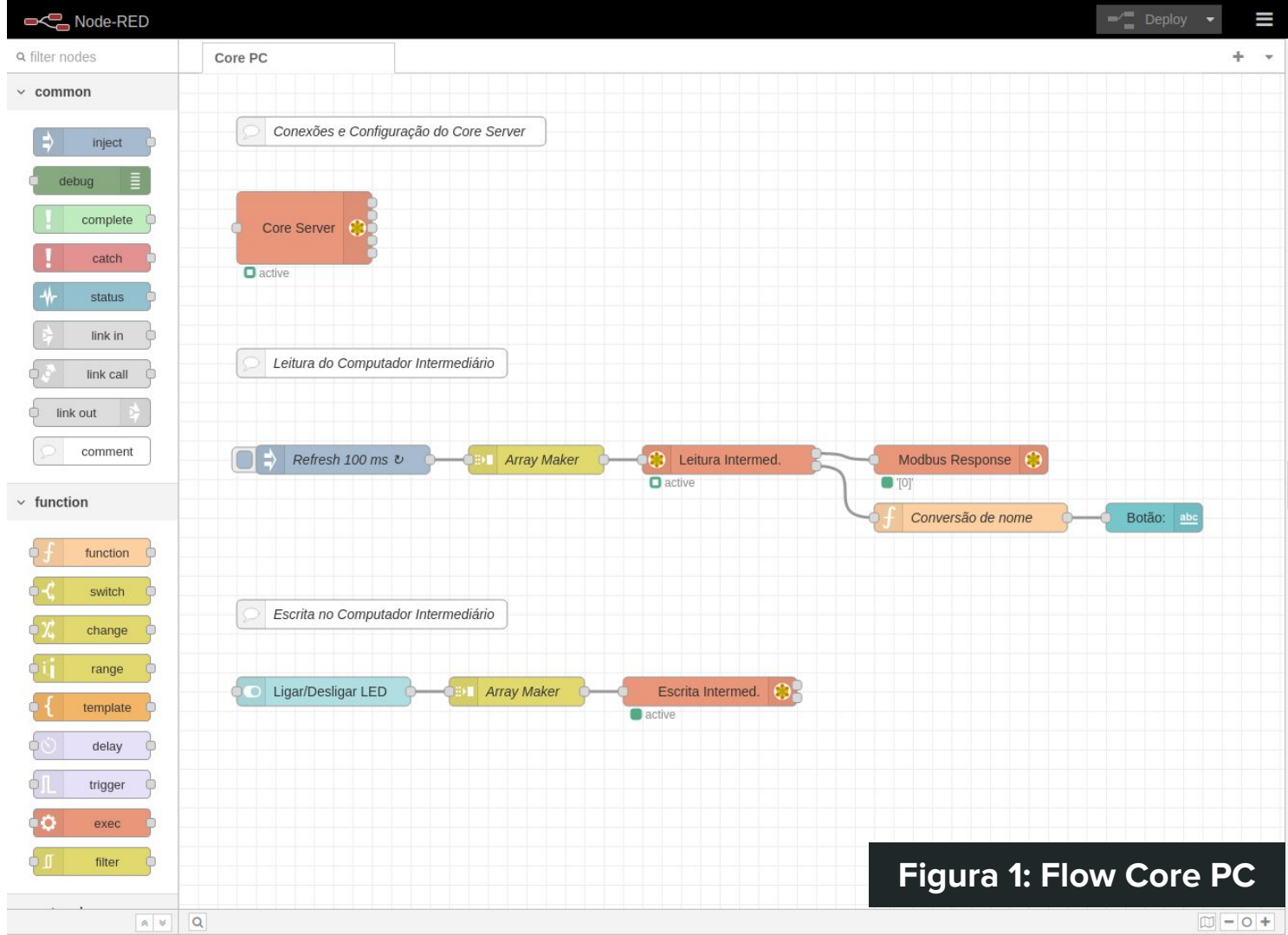
Arquitetura da aplicação

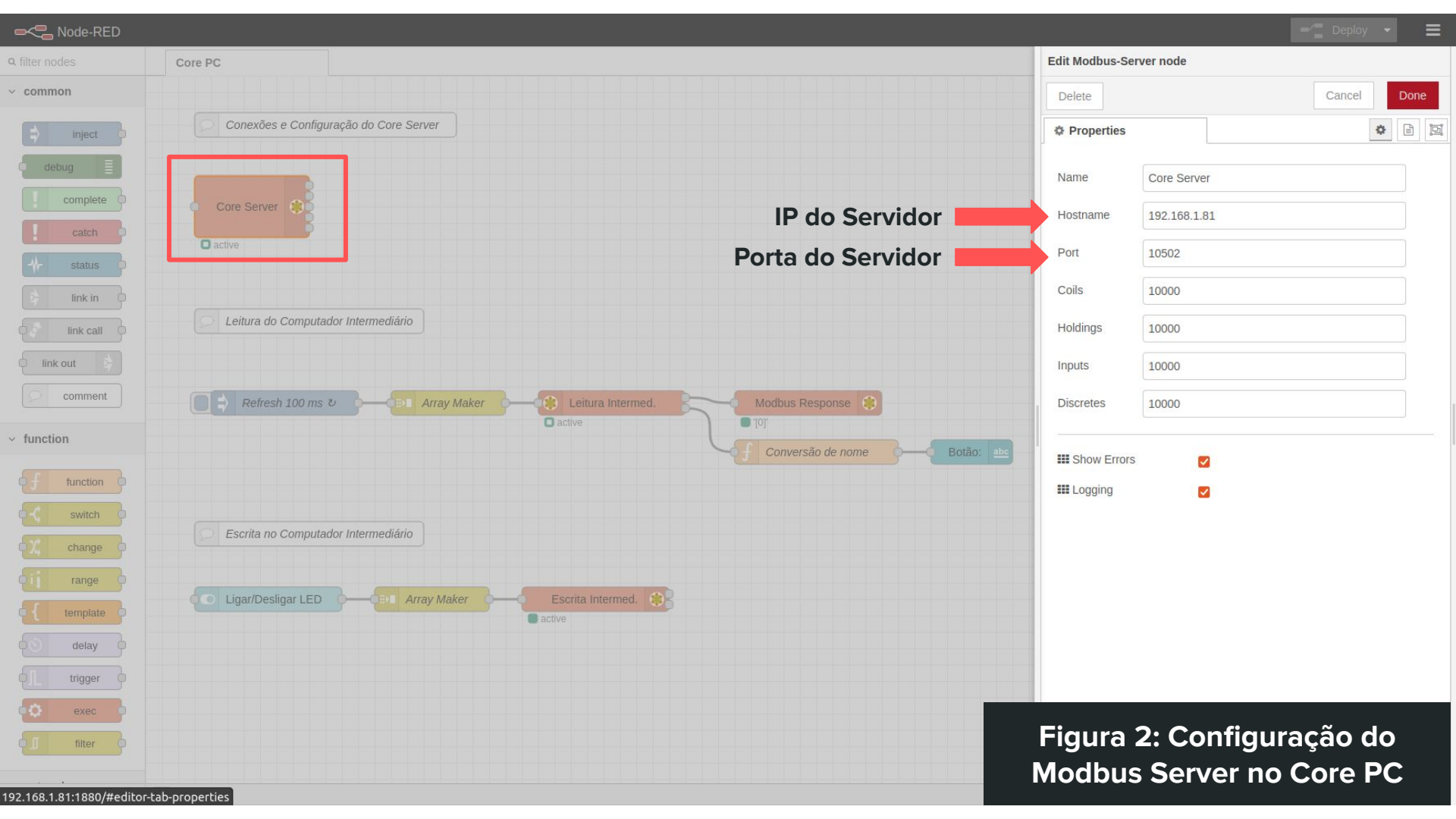


Desenvolvimento do projeto

Core PC

O Core PC é um PC que fica em uma das extremidades da rede, sendo utilizado como um Modbus Server e como interface humano-máquina por meio de um Dashboard que possibilita controle e monitoramento.





IP do Servidor
Porta do Servidor

Figura 2: Configuração do Modbus Server no Core PC

Node-RED

filter nodes

common

Inject

debug

complete

catch

status

link in

link call

link out

comment

function

switch

change

range

template

delay

trigger

Core PC

Conexões e Configuração do Core Server

Core Server

active

Leitura do Computador Intermediário

Refresh 100 ms

Array Maker

Leitura Intermed.

active

Modbus Response

0

Conversão de nome

Botão:

Escrita no Computador Intermediário

Ligar/Desligar LED

Array Maker

Escrita Intermed.

active

Figura 3: Core PC Button Reader Refresh

Período para leitura dos dados enviados pelo Interface PC

Edit inject node

DeleteCancelDone

Properties

NameRefresh 100 ms

msg. payload

=

a-z

x

msg. topic

=

a-z

x

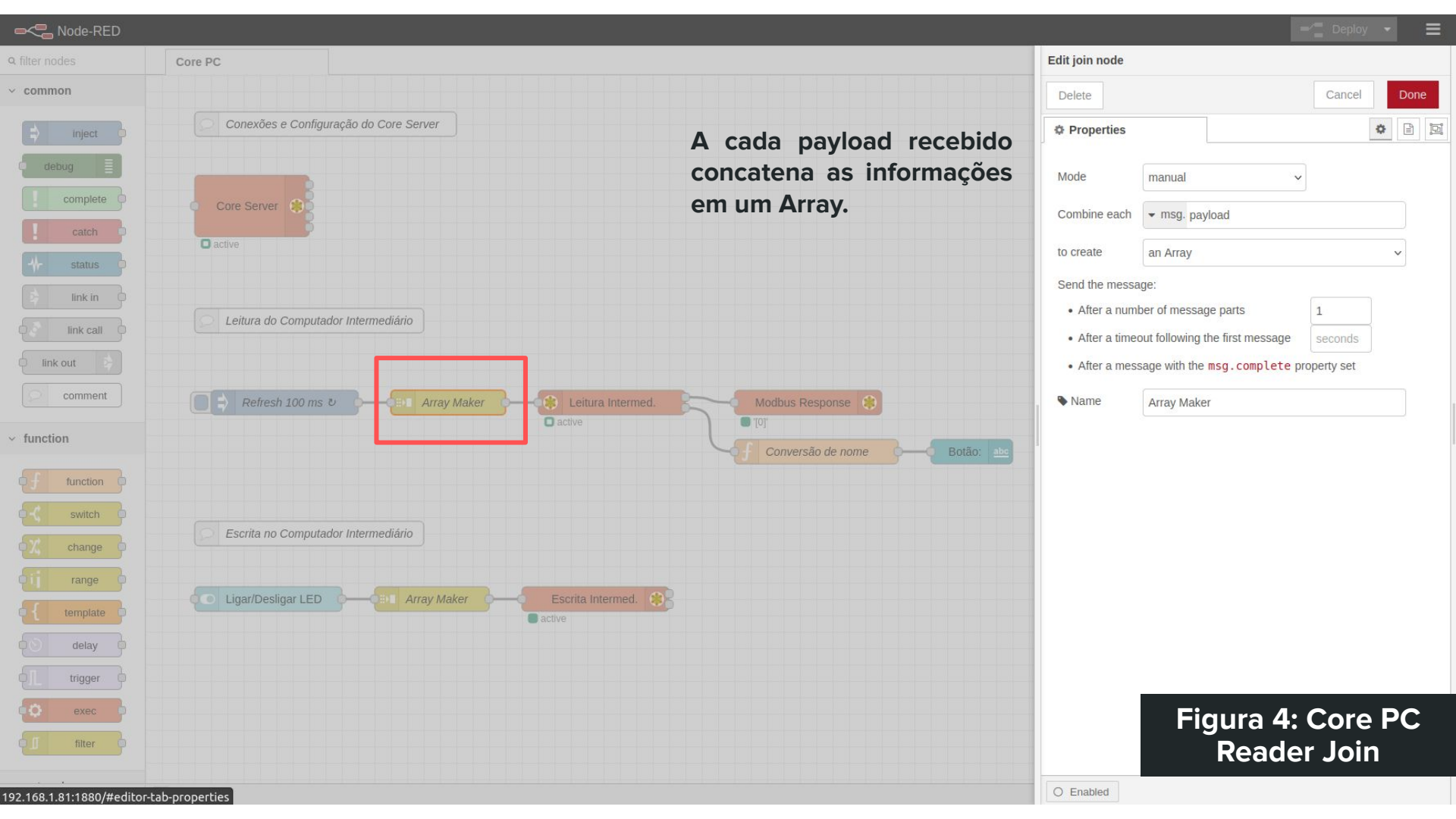
+ add

Inject once after0.1seconds, then

Repeatinterval

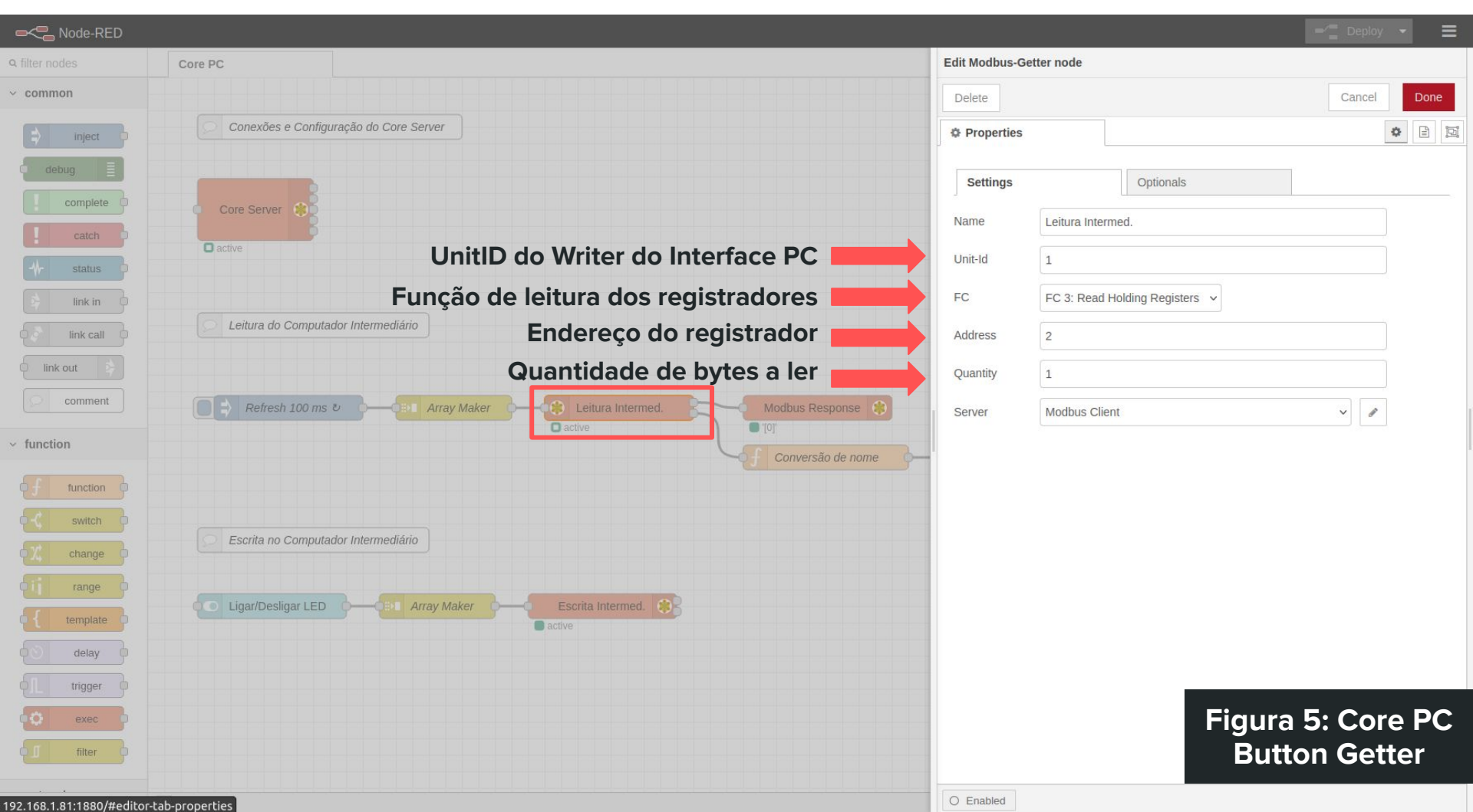
every0.1seconds

Enabled



A cada payload recebido concatena as informações em um Array.

Figura 4: Core PC Reader Join



Node-RED

filter nodes

common

inject

debug

complete

catch

status

link in

link call

link out

comment

function

function

switch

change

range

template

delay

trigger

exec

filter

Core PC

Conexões e Configuração do Core Server

Core Server

active

Leitura do Computador Intermediário

Refresh 100 ms

Array Maker

Leitura Intermed.

active

Modbus Response

10

Conversão de nome

Escrita no Computador Intermediário

Ligar/Desligar LED

Array Maker

Escrita Intermed.

active

Edit function node

Delete

Cancel

Done

Properties

Name

Conversão de nome

Setup

On Start

On Message

On Stop

```
1 if (msg.payload.data[0] == 1) {
2     msg.payload = {
3         value: "Pressionado"
4     }
5 }else{
6     msg.payload = {
7         value: "Solto"
8     }
9 }
10 return msg;
```

Código para conversão do estado do botão para texto

0: Solto

1: Pressionado

Enabled

Figura 6: Core PC Button Function

Node-RED

filter nodes

Core PC

common

inject

debug

complete

catch

status

link in

link call

link out

comment

function

function

switch

change

range

template

delay

trigger

exec

filter

Conexões e Configuração do Core Server

Core Server

active

Leitura do Computador Intermediário

Refresh 100 ms

Array Maker

Leitura Intermed.

active

Modbus Response

[0]

Conversão de nome

Botão: abc

Escrita no Computador Intermediário

Ligar/Desligar LED

Array Maker

Escrita Intermed.

active

Label

Valor Solto/Pressionado

Edit text node

Delete

Cancel

Done

Properties

Group

[Home] Painel de Controle

Size

auto

Label

Botão:

Value format

{{msg.payload.value}}

Layout

label value

label value

label value

label value

label value

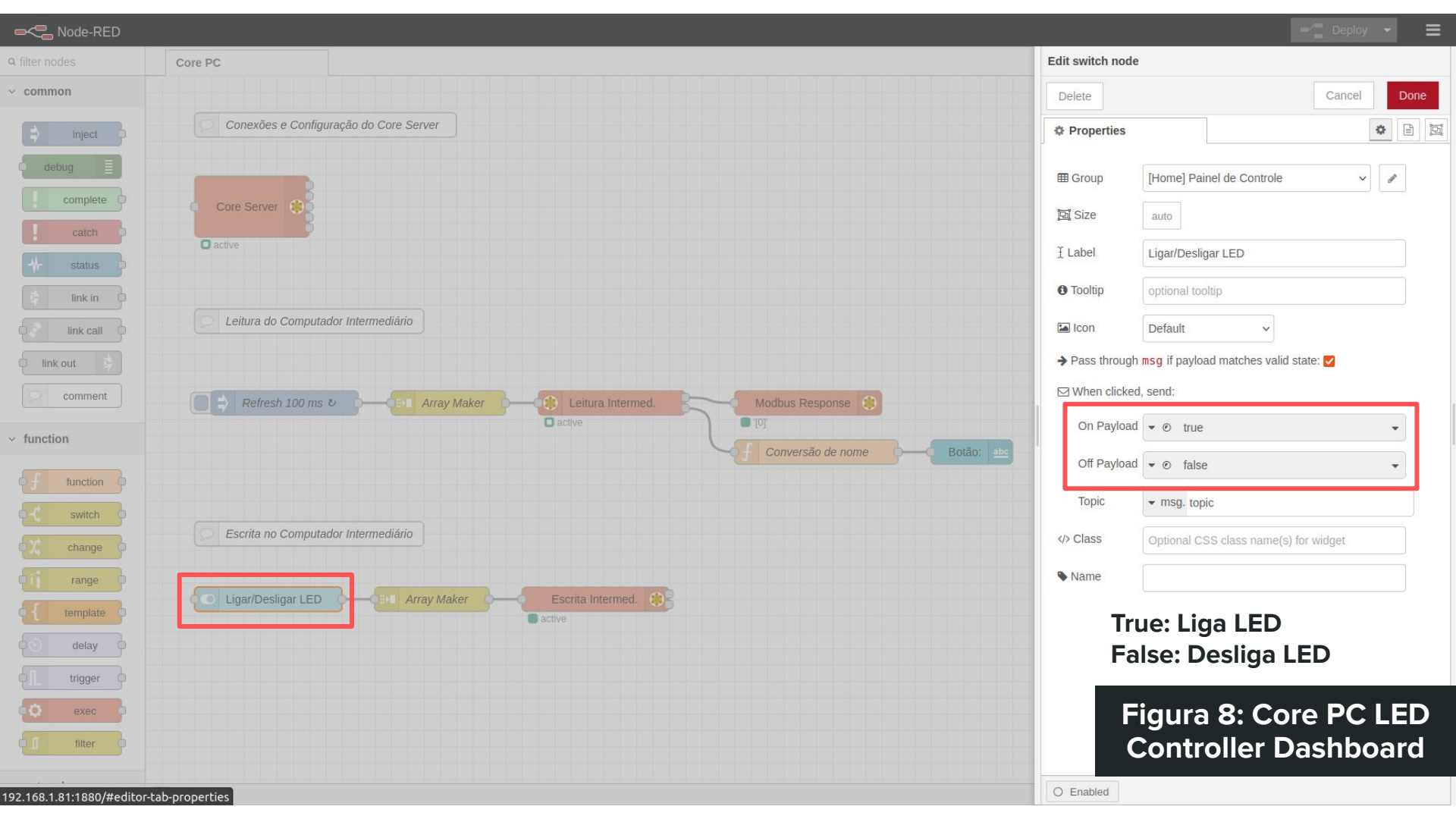
Class

Optional CSS class name(s) for widget

Name

Enabled

Figura 7: Core PC Button Reader Dashboard



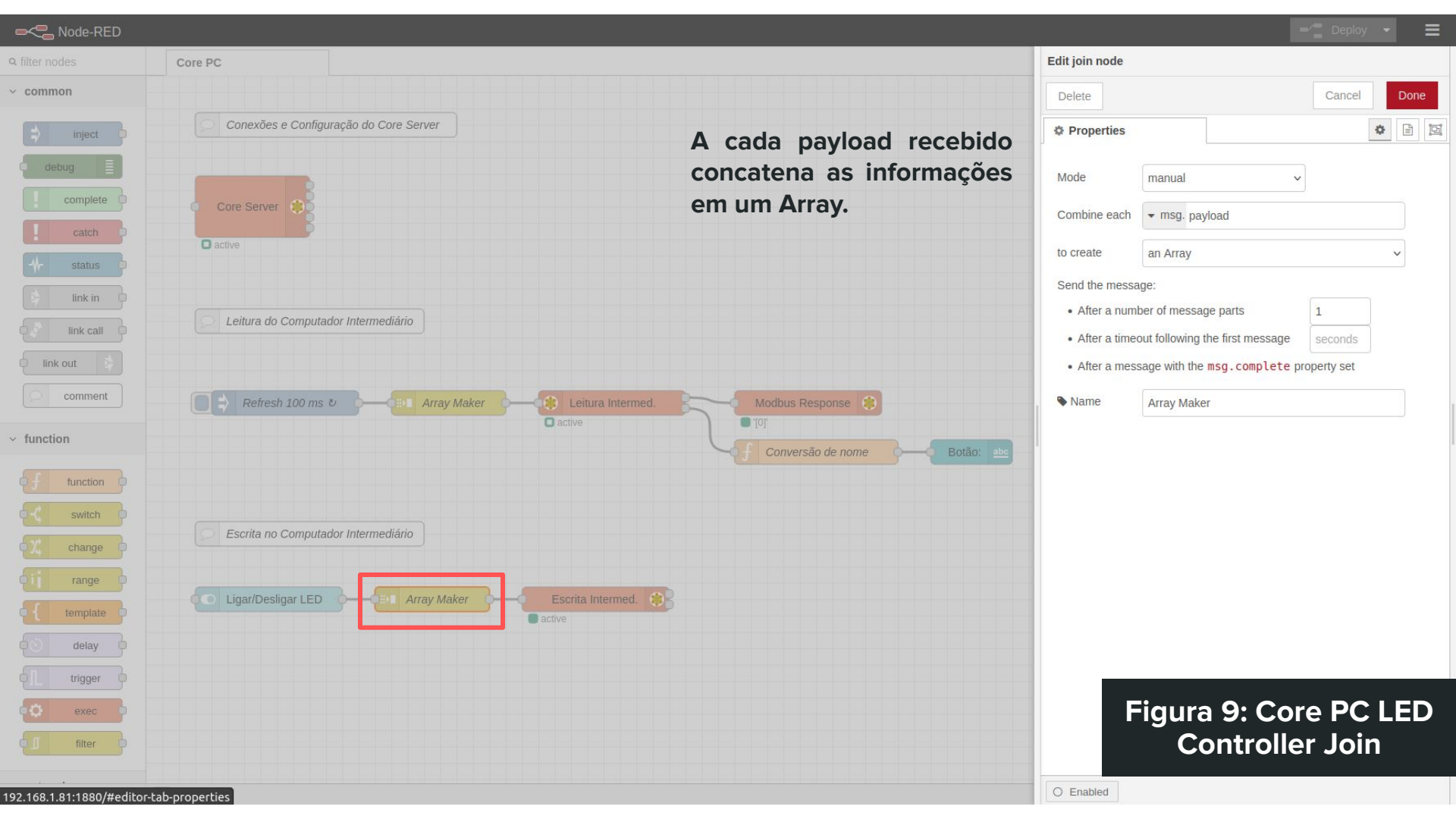
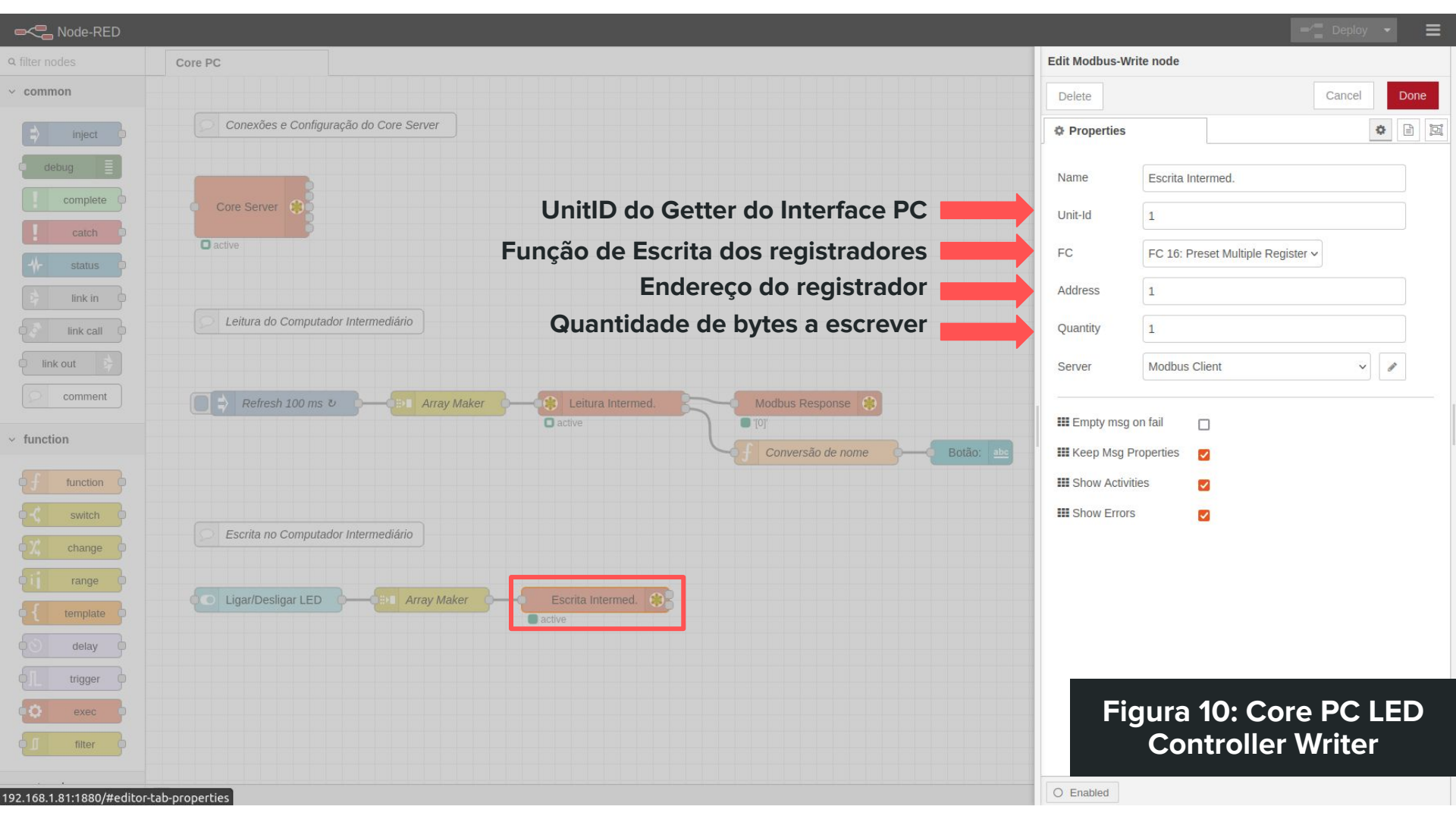


Figura 9: Core PC LED Controller Join



Node-RED

filter nodes

Core PC

common

- inject
- debug
- complete
- catch
- status
- link in
- link call
- link out
- comment

function

- function
- switch
- change
- range
- template
- delay
- trigger
- exec
- filter

Conexões e Configuração do Core Server

Core Server

active

Leitura do Computador Intermediário

Refresh 100 ms

Array Maker

Leitura Intermed.

active

Escrita no Computador Intermediário

Ligar/Desligar LED

Array Maker

Escrita Intermed.

active

Edit modbus-client node

Delete Cancel Update

Properties

Name: Modbus Client

Type: TCP

Host: 192.168.1.81

Port: 10502

TCP Type: RTU-BUFFERED

Unit-Id: 1

Timeout (ms): 1000

Reconnect on timeout: ☒

Reconnect timeout (ms): 2000

Unitid's in parallel: ☒

Log states changes: ☐

Queue Logging: ☐

Log failures: ☒

Queue commands: ☒

Queue delay (ms): 1

Enabled 2 nodes use this config On all flows

192.168.1.81:1880/#editor-tab-properties

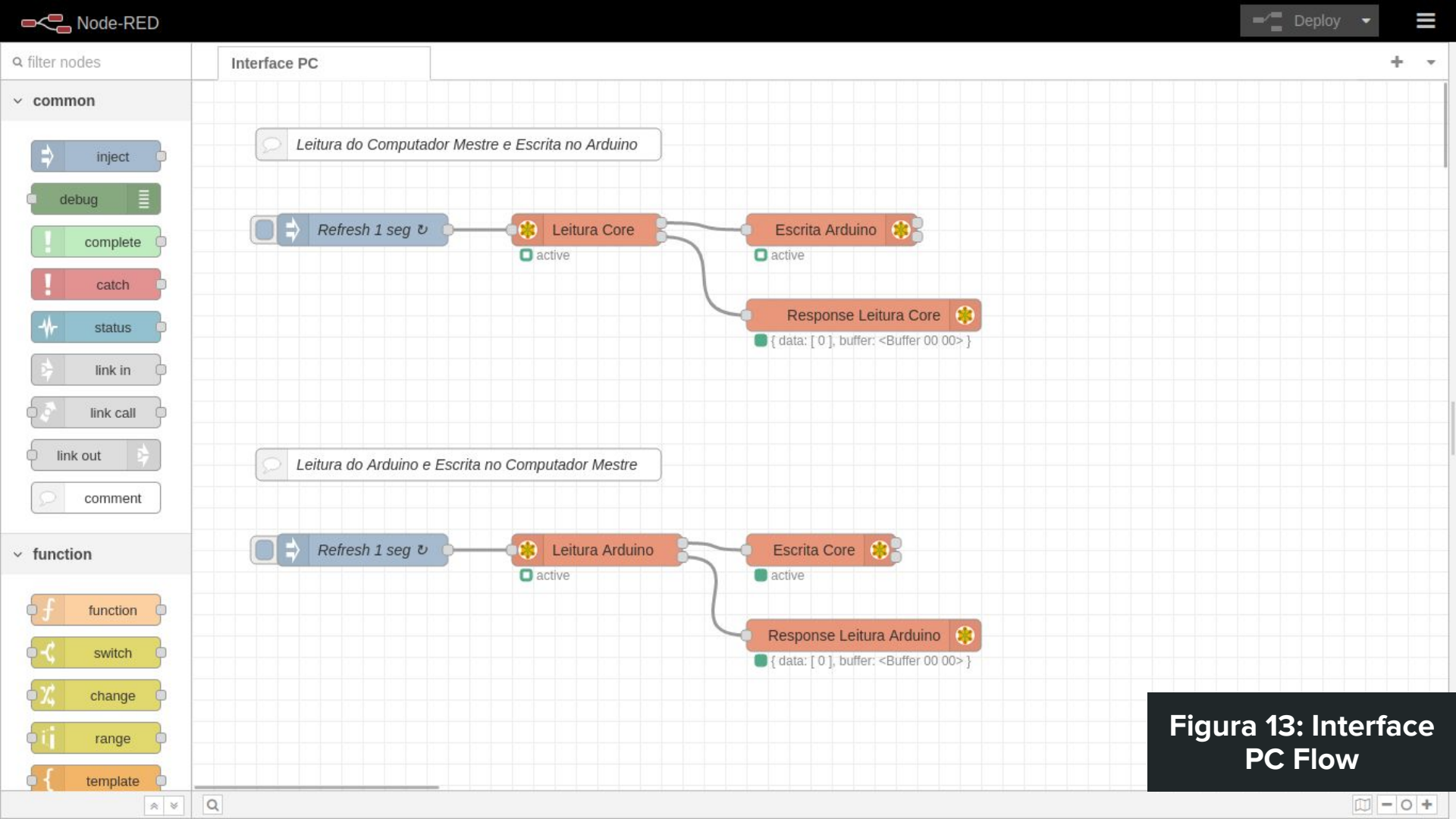
Figura 11: Modbus Client Config



**Figura 12: Core
PC Dashboard**

Interface PC

O Interface PC é um PC configurado para simular o comportamento de um CLP, servindo apenas para interfaceamento e transmissão de informações por entre as extremidades da rede configurada (Core PC e Arduino).



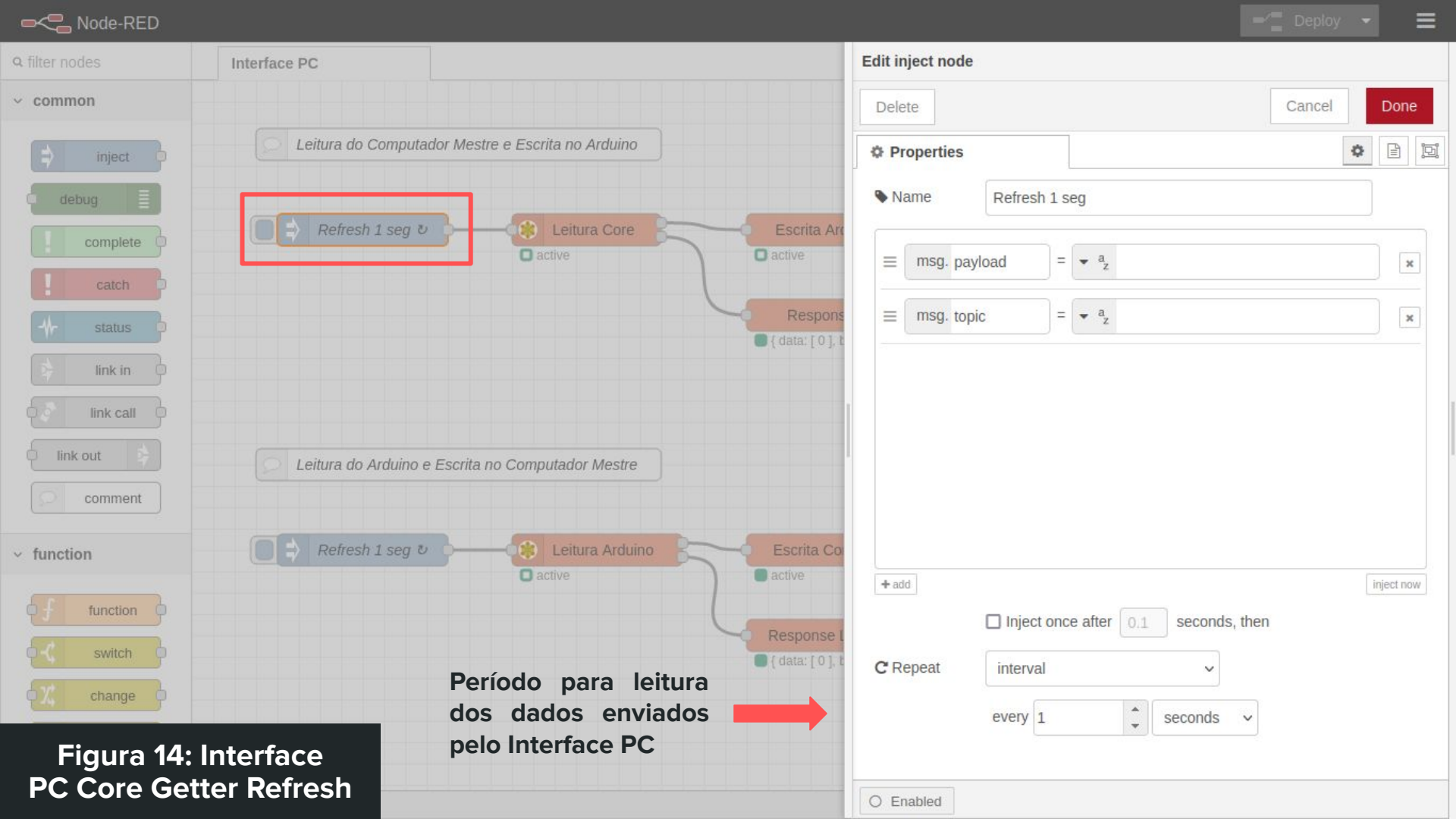
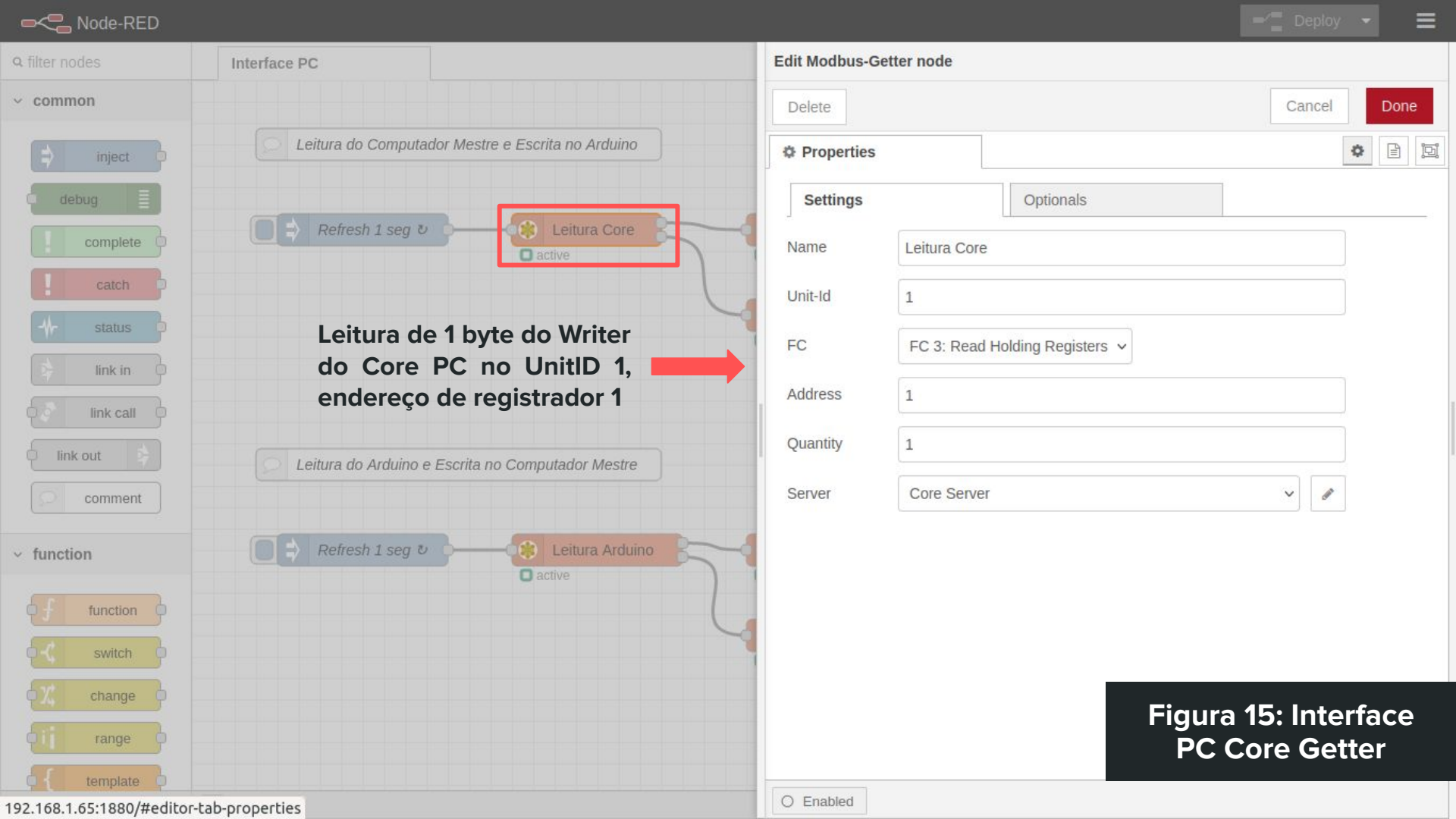
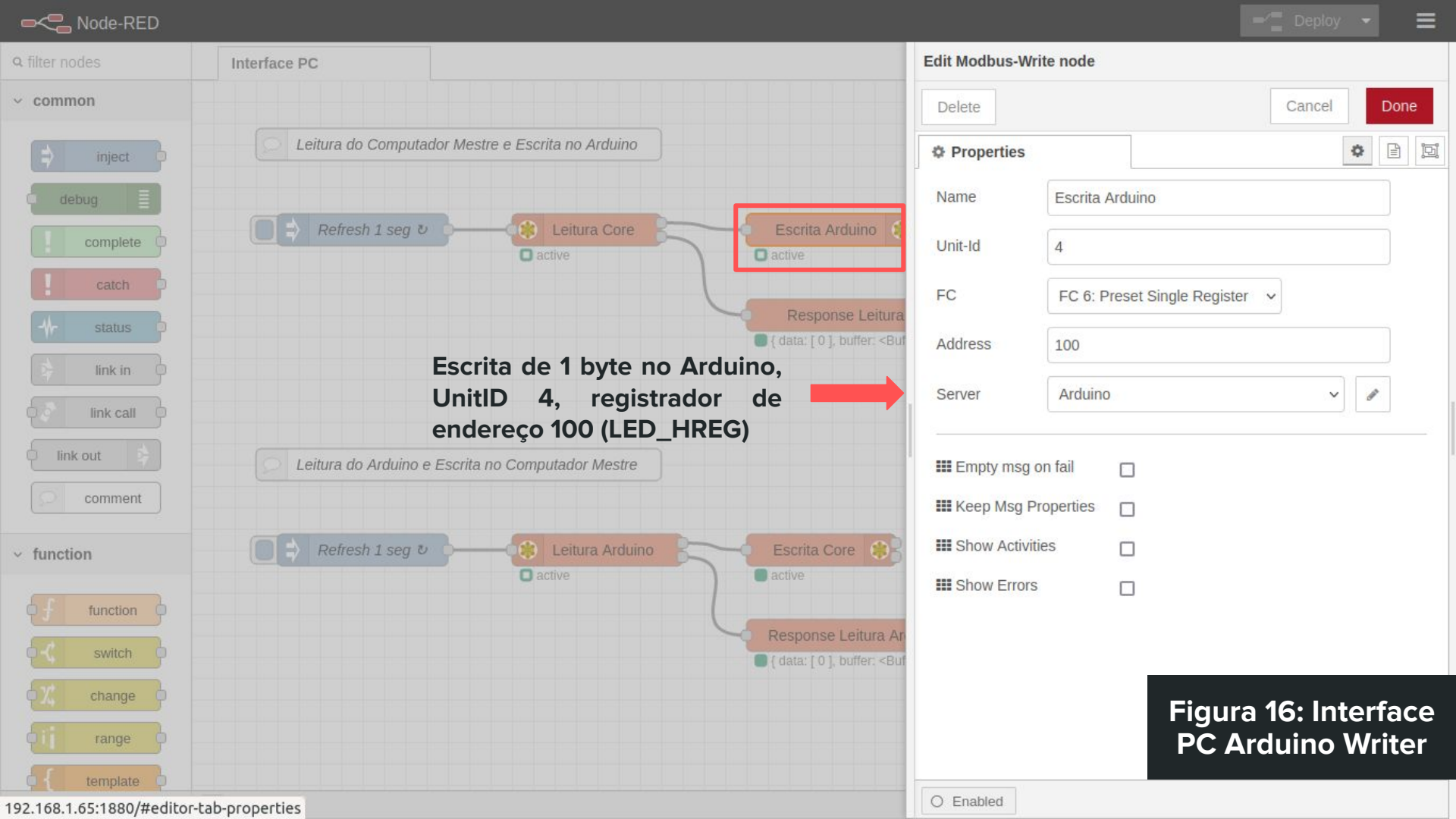


Figura 14: Interface PC Core Getter Refresh



**Leitura de 1 byte do Writer
do Core PC no UnitID 1,
endereço de registrador 1**

**Figura 15: Interface
PC Core Getter**



**Escrita de 1 byte no Arduino,
UnitID 4, registrador de
endereço 100 (LED_HREG)**

Edit Modbus-Write node

Delete

Cancel

Done

Properties

Name

Escrita Arduino

Unit-Id

4

FC

FC 6: Preset Single Register

Address

100

Server

Arduino

☐ Empty msg on fail

☐ Keep Msg Properties

☐ Show Activities

☐ Show Errors

**Figura 16: Interface
PC Arduino Writer**

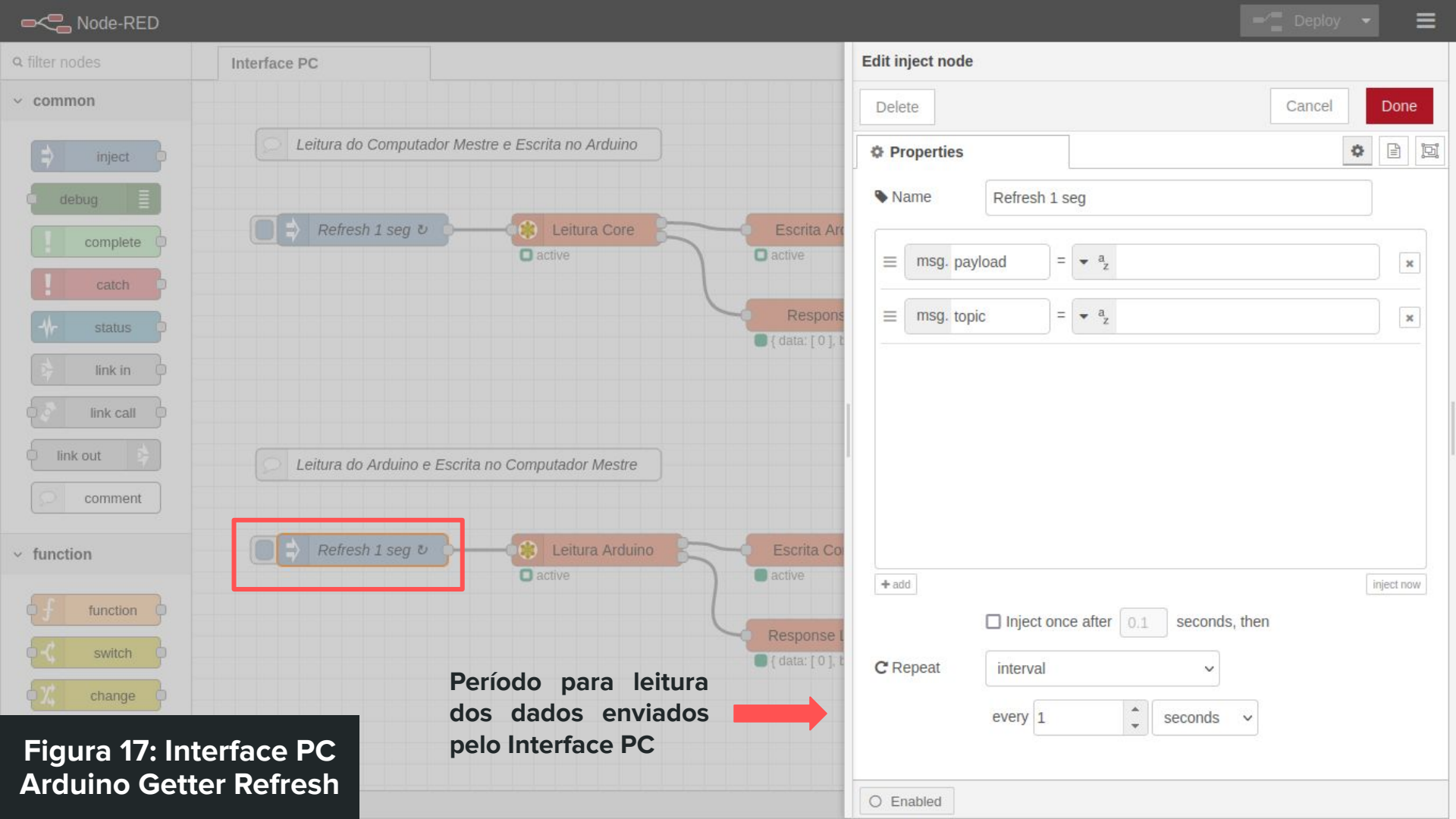


Figura 17: Interface PC Arduino Getter Refresh

Período para leitura dos dados enviados pelo Interface PC

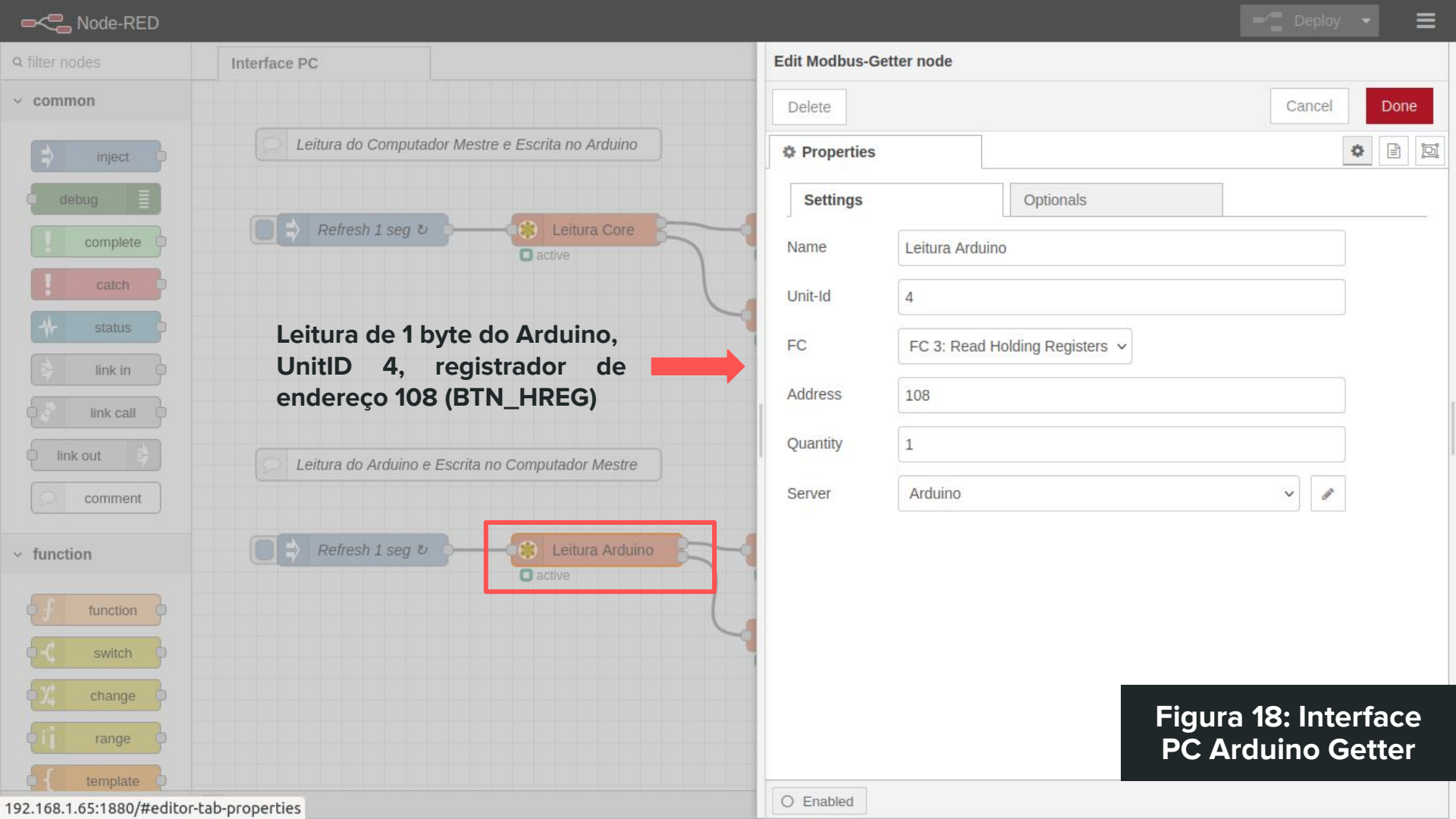
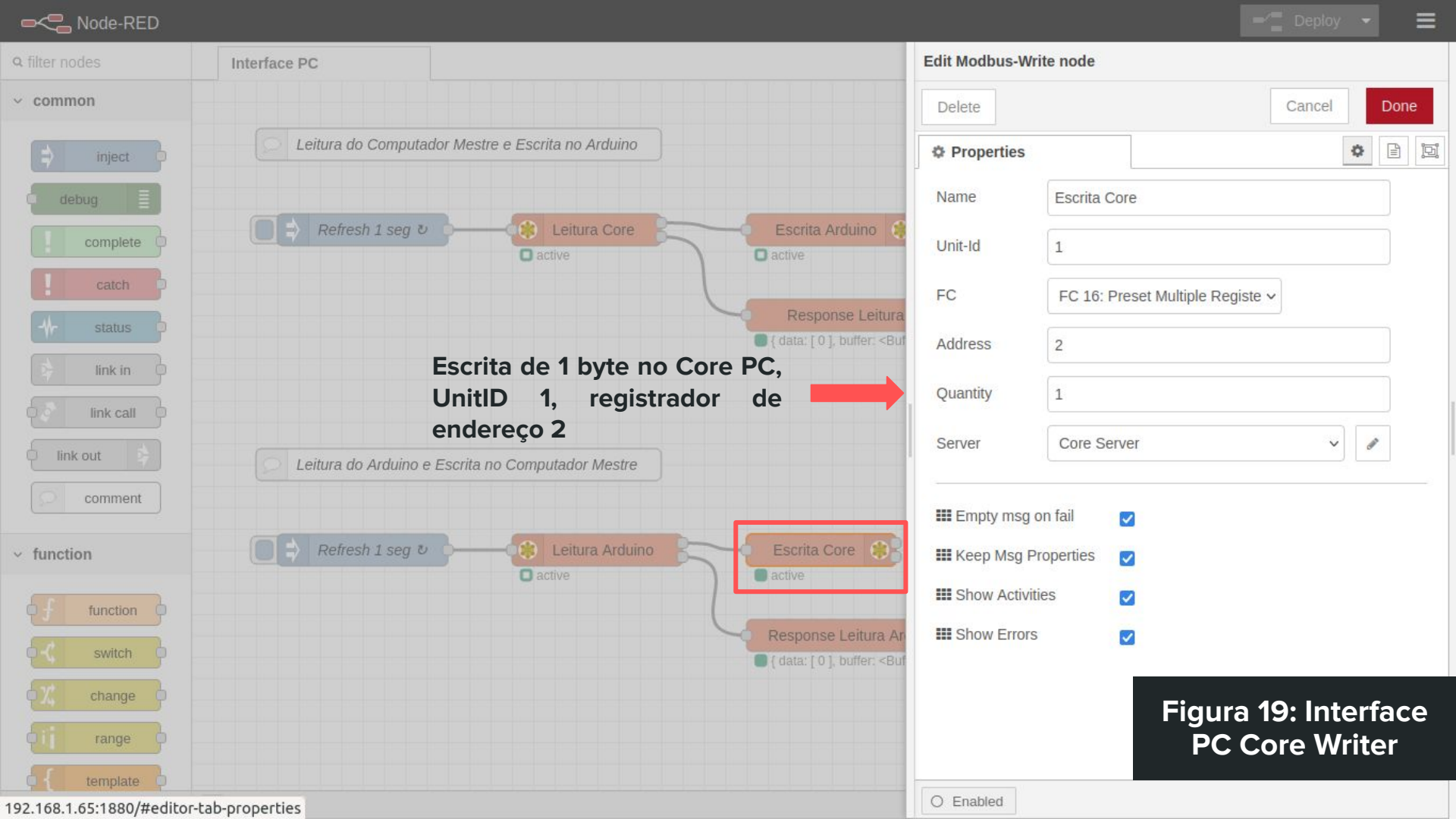
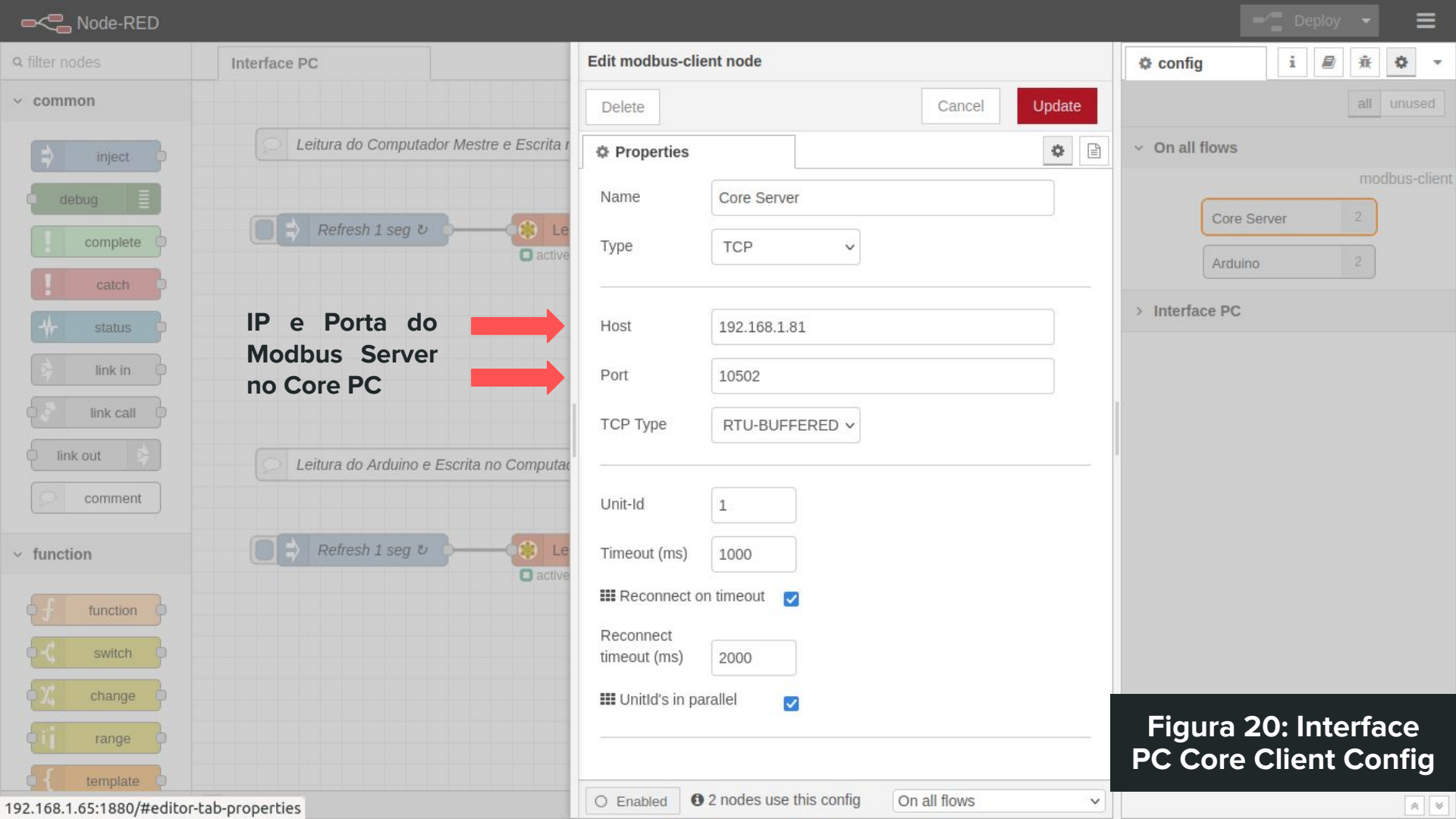


Figura 18: Interface PC Arduino Getter





IP e Porta do
Modbus Server
no Core PC

Edit modbus-client node

Delete

Cancel

Update

Properties

Name Core Server

Type TCP

Host 192.168.1.81

Port 10502

TCP Type RTU-BUFFERED

Unit-Id 1

Timeout (ms) 1000

Reconnect on timeout ☒

Reconnect
timeout (ms) 2000

Unittid's in parallel ☒

Enabled

2 nodes use this config

On all flows

config

On all flows

modbus-client

Core Server

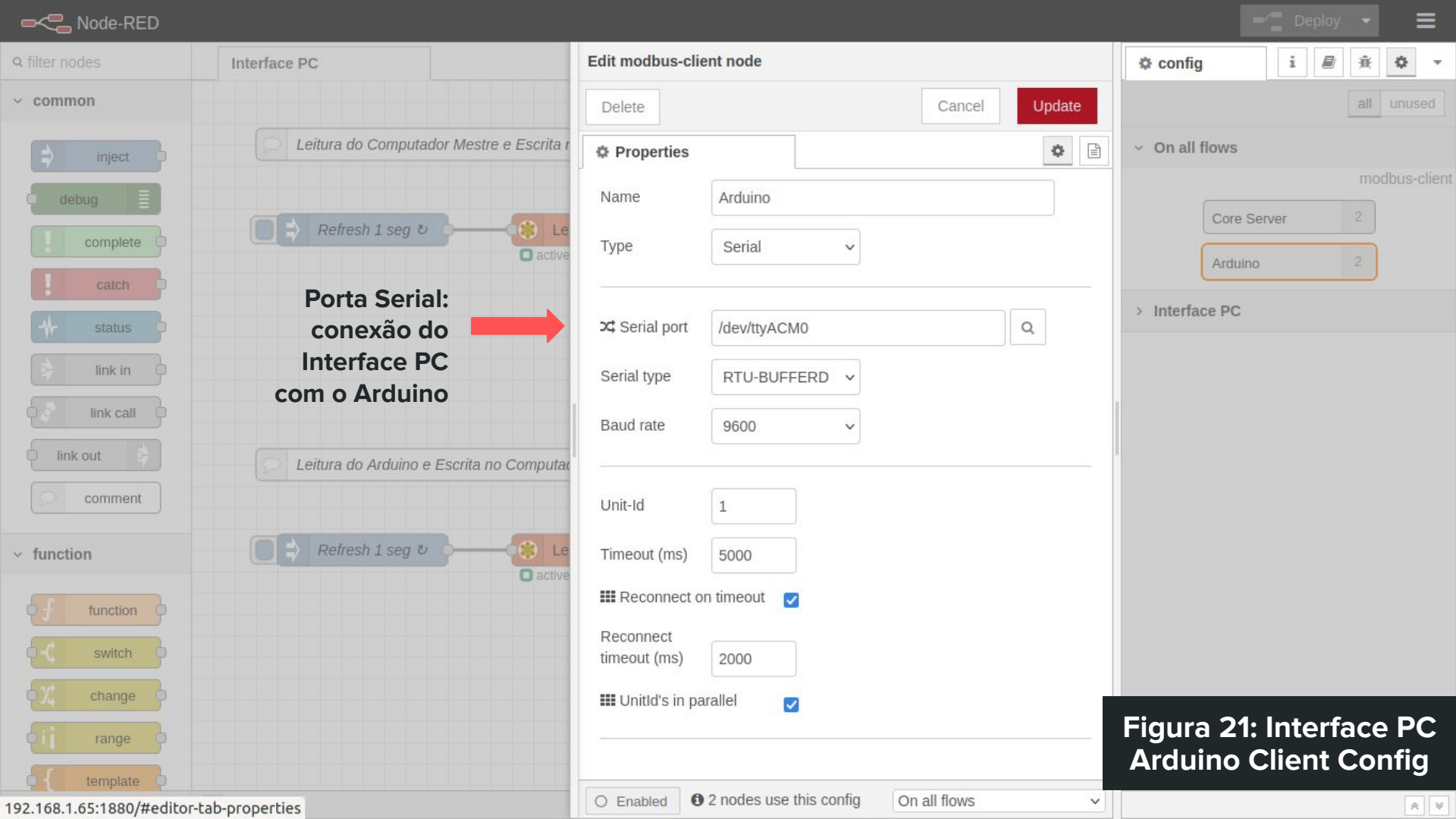
2

Arduino

2

Interface PC

Figura 20: Interface
PC Core Client Config



**Porta Serial:
conexão do
Interface PC
com o Arduino**

Edit modbus-client node

Delete

Cancel

Update

Properties

Name

Type

Serial port

Serial type

Baud rate

Unit-Id

Timeout (ms)

Reconnect on timeout ☒

Reconnect timeout (ms)

Unittld's in parallel ☒

config

i

📄

🔍

⚙️

▼

all

unused

On all flows

modbus-client

Core Server 2

Arduino 2

Interface PC

**Figura 21: Interface PC
Arduino Client Config**

Arduino

O Arduino é a extremidade da rede que se conecta diretamente com o circuito. A nível industrial, seria equivalente ao dispositivo responsável por ler informações de sensores e mandar os comandos para os atuadores.

```

// Definição de constantes
dos pinos utilizados
#define LED 13
#define BTN 8

#include <Modbus.h>
#include <ModbusSerial.h>

// Offset de registradores
const int LED_HREG = 100;
const int BTN_HREG = 108;

// Objeto ModbusSerial
ModbusSerial mb;

```

```

void setup() {
    // Configuração da comunicação Modbus
    mb.config(&Serial, 9600, SERIAL_8N1);

    // Configuração de ID
    mb.setSlaveId(4);

    // Definição de pinos
    pinMode(BTN, INPUT_PULLUP);
    pinMode(LED, OUTPUT);

    // Configuração de registradores
    mb.addHreg(LED_HREG, 0);
    mb.addHreg(BTN_HREG, 0);
}

```

```

void loop() {
    // Chamada da task Modbus
    mb.task();

    // Rotina de atualização do LED
    if(mb.Hreg(LED_HREG) == 1){
        digitalWrite(LED, true);
    }else if(mb.Hreg(LED_HREG) == 0){
        digitalWrite(LED, false);
    }

    // Rotina de leitura do botão
    mb.Hreg(BTN_HREG, !digitalRead(BTN));
}

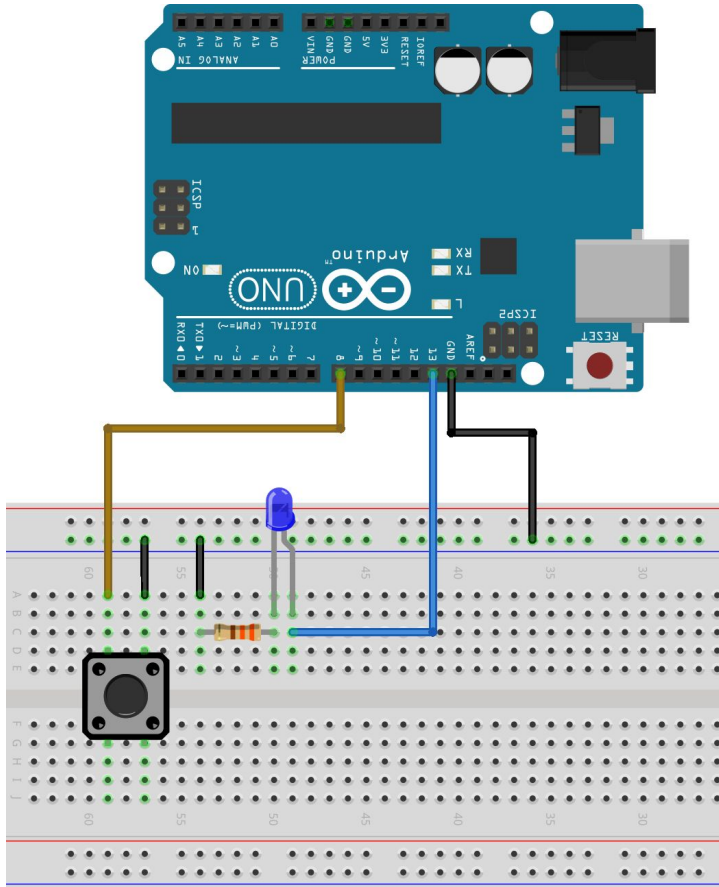
```

Código do Arduino

Circuito

O circuito representa os sensores que podem ser lidos e atuadores que podem ser controlados. Na nossa montagem, o botão simula um sensor, e o LED, um atuador.

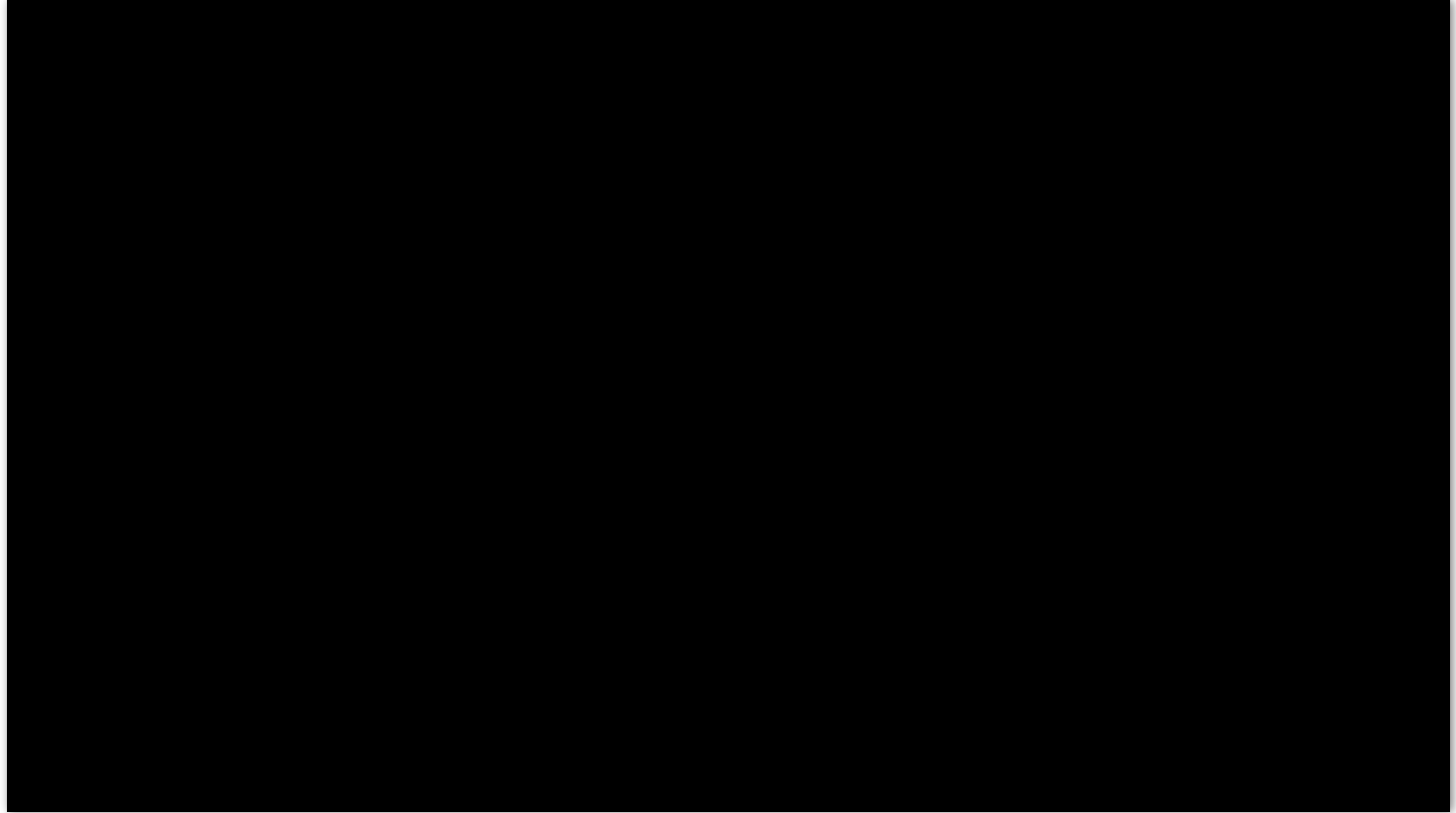




Conexões com o Arduino

- Pino 8: Botão, INPUT_PULLUP
- Pino 13: LED, OUTPUT

Vídeo - Demonstração do funcionamento



Considerações Finais

- Conseguimos fazer uma prova de conceito demonstrando didaticamente algumas aplicações de Redes Industriais.
- Apesar de em indústrias reais a rede geralmente ser de uma escala bem maior, conseguimos reproduzir conceitos essenciais para uma rede industrial, como uma interface de supervisão e controle, dispositivos mediadores, controladores, sensores e atuadores.

Obrigado pela atenção!
