

BASE

AUTOMAÇÃO

SERVIÇOS . EQUIPAMENTOS . SOLUÇÕES



Upgrade de hardware Elrest para Schneider

- Objetivo
- Serviços desenvolvidos
- Antiga arquitetura de automação
- Nova arquitetura de automação
- Tecnologias utilizadas
- Software
- Exemplo de telas
- Vantagens da migração dos hardwares e softwares
- Contatos

- Apresentar o exemplo de uma solução de upgrade de hardware industrial completo desenvolvida pela equipe da BASE Automação, com serviços diversificados, tais como parametrizações dos novos módulos, conversão e programação dos novos equipamentos baseado na programação antiga, testes da nova arquitetura garantindo o mesmo funcionamento ou superior, dessa forma potencializando o processo anterior, aumentando a confiabilidade e a capacidade.



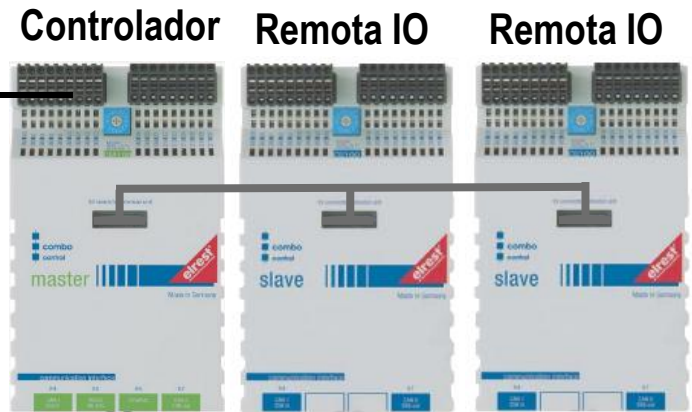
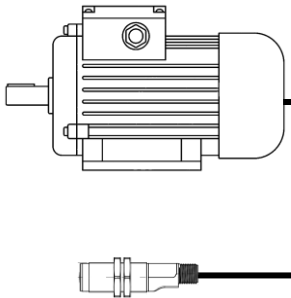
- Conversão e adequações da lógica do controlador.
- Conversão das telas.
- Organização das lógicas, inserção de comentários e descrições.
- Configuração dos novos módulos de I/O.
- Configuração da rede de comunicação.
- Comissionamento e startup da nova arquitetura.



Arquitetura de automação baseada em Elrest – (Solução antiga)

Equipamentos Elrest

Equipamentos em Campo



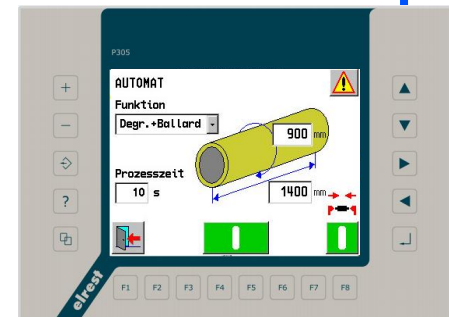
Rede CanOpen

Supervisório



Rede Ethernet

IHM



Arquitetura de automação baseada em Schneider – (Solução Nova)

Equipamentos Schneider

Supervisório



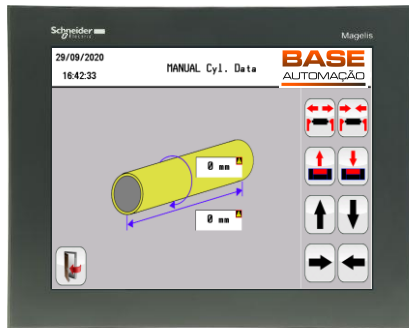
Controlador



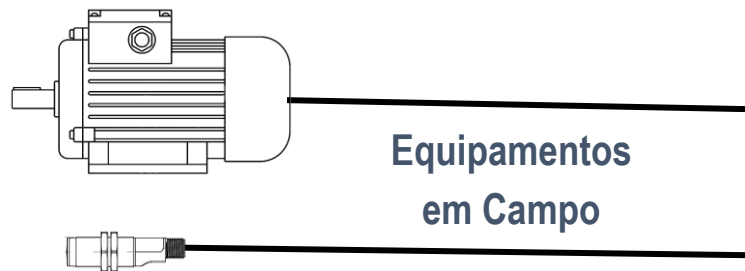
Rede Ethernet



IHM



Equipamentos em Campo



Controlador lógico programável – CLP

Elrest Combo Control Master CM110

➤ Comunicação:

- Ethernet TCP/IP
- Serial
- CANOpen
- Modbus TCP
- FTP client/server
- SNMP client/server
- Web server (WebVisu & XWeb system)

elrest[®]
INNOVATION FOR AUTOMATION



Controlador lógico programável – CLP

Elrest Combo Control Master CM110

➤ Memória:

- 4 MB para programação
- 2 MB Memória de sistema RAM

➤ IOs:

- 16 Entradas e saídas digitais (selecionáveis)
- 4 Entradas analógicas
- 4 Saídas analógicas

➤ Expansão:

- Expansível através de módulos combo escravos

➤ Software de Programação:

- ElaDesign – baseado em CoDeSys

elrest[®]
INNOVATION FOR AUTOMATION



Interface homem máquina - IHM

Schneider Harmony GTO 7.5

➤ Tela:

- 5.7 Polegadas

➤ Resolução:

- 320 x 240 px

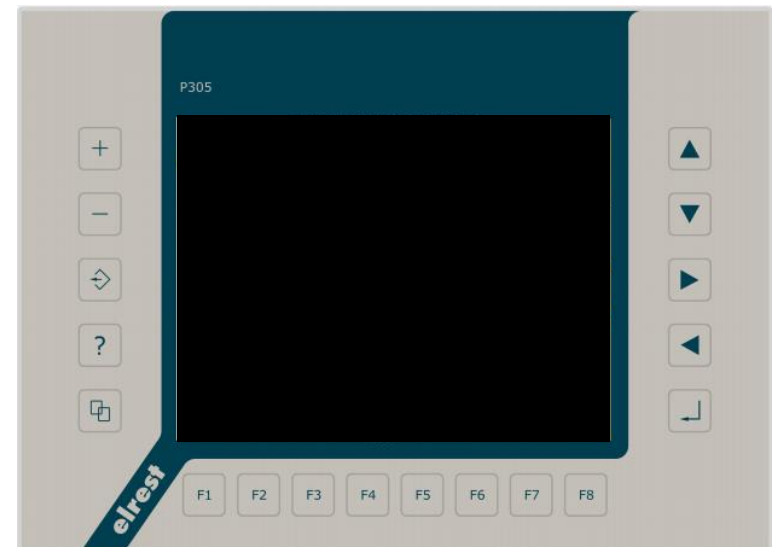
➤ Interface:

- 18 teclas frontais / Touchscreen

➤ Software de programação:

- ElaDesign – baseado em CoDeSys

elrest[®]
INNOVATION FOR AUTOMATION



Controlador lógico programável – CLP Schneider Modicon M251 Ethernet CAN

➤ Comunicação:

- Ethernet/IP slave device
- IEC VAR ACCESS
- Modbus TCP cliente
- Modbus TCP server
- Modbus TCP slave device
- FTP client/server
- SNMP client/server
- SQL cliente
- Web server (WebVisu & XWeb system)
- OPC UA server



Controlador lógico programável – CLP Schneider Modicon M251 Ethernet CAN

➤ Expansão:

- Máximo de 7 slots por rack
- Máximo de 14 remotas

➤ Memória:

- 8 MB para programação
- 64 MB Memória de sistema RAM

➤ Software de Programação:

- SoMachine – baseado em CoDeSys

Schneider
Electric



Interface homem máquina - IHM

Schneider Harmony GTO 7.5

➤ Tela:

- 7.5 Polegadas

➤ Resolução de cores:

- 65536

➤ Resolução:

- 640 x 480 px

➤ Interface:

- Touchscreen

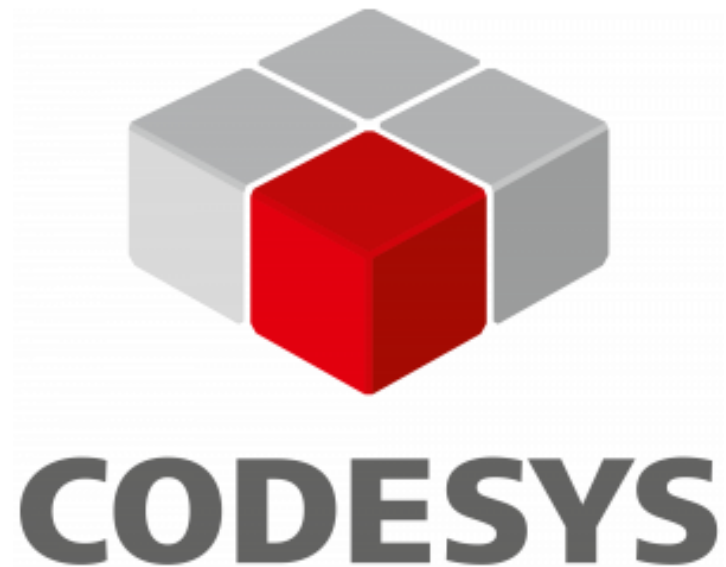
➤ Software de programação:

- Vijeo Designer

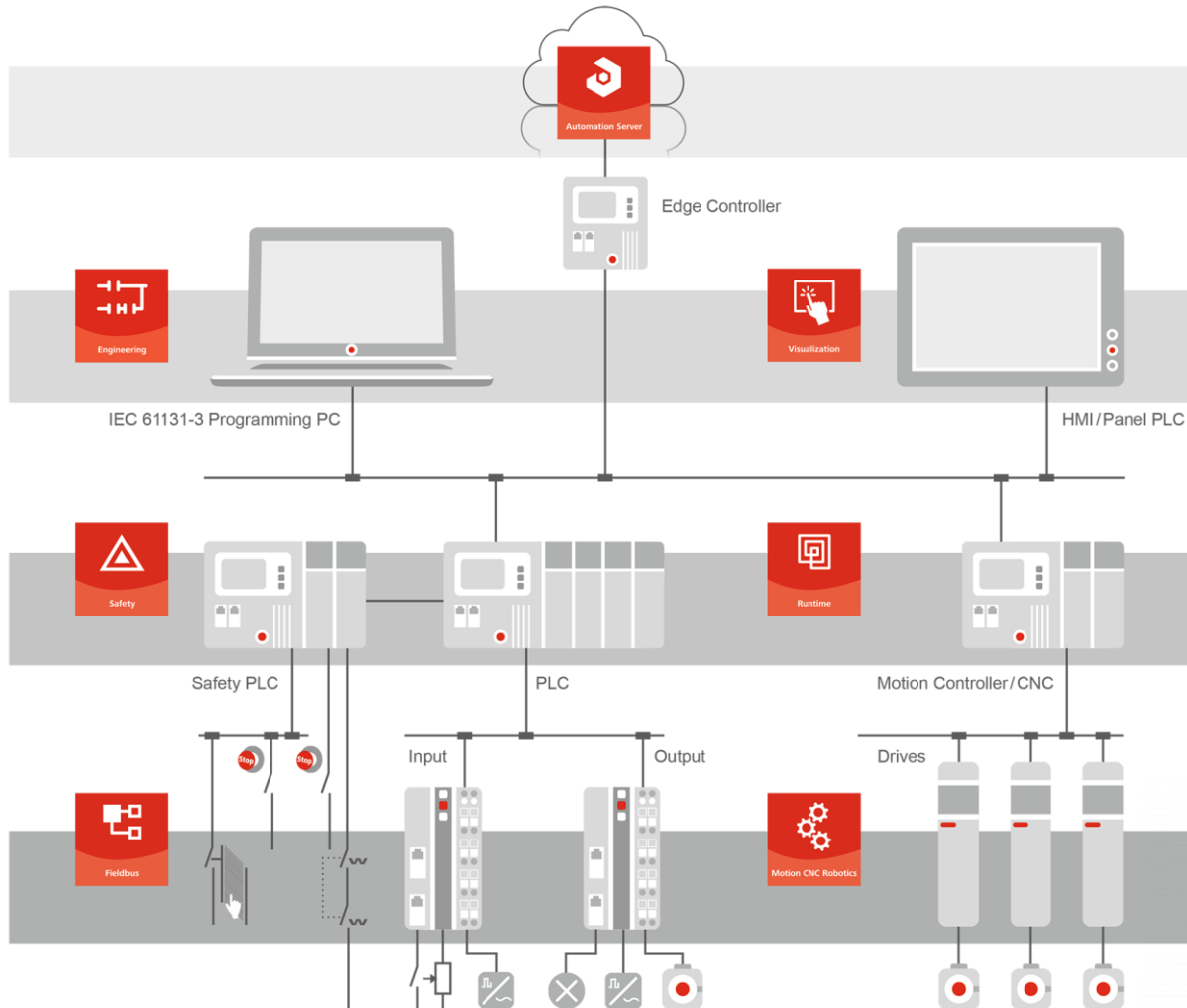


- CODESYS é uma plataforma de software para tecnologia de automação industrial. O núcleo da plataforma é a ferramenta de programação IEC-61131-3 “Controller Development System“, ela oferece aos usuários soluções integradas orientadas para a prática e configuração conveniente de aplicativos de automação. O objetivo é fornecer suporte prático para as tarefas diárias.

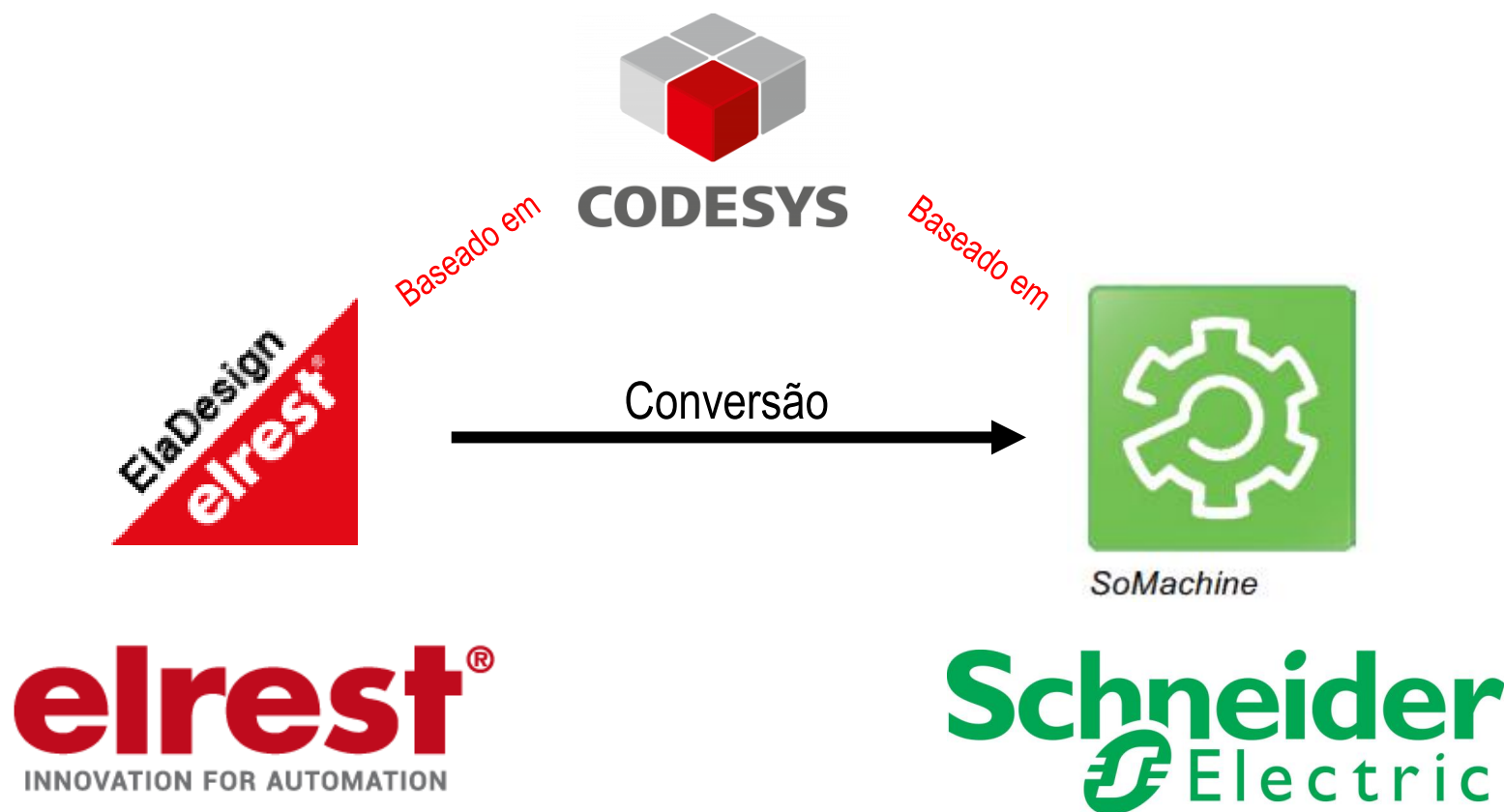
O software Codesys é um software independente e os fabricantes de dispositivos usam o CODESYS para implementar seus próprios componentes de automação programáveis ou configuráveis. Existem Milhões de dispositivos únicos compatíveis com CODESYS, mais de 1.000 tipos de dispositivos diferentes de mais de 400 fabricantes e dezenas de milhares de usuários finais CODESYS



Áreas de atuação Software CODESYS



Software de programação



Exemplo de tela de programação de Elrest – (Eladesign – Codesys 2.3)

The screenshot displays the Codesys 2.3 development environment. On the left, a project tree shows a hierarchy of POUs (Programs, Functions, Function Blocks) under the name 'Elrest'. The main editor area contains a ladder logic program for 'PROGRAM PinFun1'. The code is as follows:

```
0001 PROGRAM PinFun1
0002
0003 (* PinFun1 *)
0004
0005 (* Anzeige *)
0006 PinolenZustand := E_PinPressOk;
0007
0008 (* Steuerung *)
0009 IF NOT RemoteBetriebAktiv THEN
0010   IF (FTastePinOpen OR PosFlankPinOpenKey.Q) AND (NOT PinAktiv) AND
0011     (NOT CylSpeedAktiv) AND (NOT AutoFun3Aktiv) AND (NOT PinNotStart) THEN
0012     PinOpenAktiv := TRUE;
0013   END_IF
0014   IF (FTastePinClose OR PosFlankPinCloseKey.Q) AND (NOT PinAktiv) AND
0015     (NOT E_PinPressOk) AND (NOT PinNotStart) THEN
0016     PinCloseAktiv := TRUE;
0017   END_IF
0018   IF T_AutoPinCloseStart AND (NOT PinAktiv) AND (NOT E_PinPressOk) AND (NOT PinNotStart) THEN
0019     PinAutoAktiv := TRUE;
0020     PinCloseAktiv := TRUE;
0021   END_IF
0022   IF ((NOT (DauerFTastePinOpen OR DauerFTastePinClose OR E_PinOpenKey OR E_PinCloseKey)) OR
0023     (DauerFTastePinOpen AND (DauerFTastePinClose OR E_PinOpenKey OR E_PinCloseKey))) OR
0024     (DauerFTastePinClose AND (E_PinOpenKey OR E_PinCloseKey)) OR
0025     (E_PinOpenKey AND E_PinCloseKey)) AND (NOT PinAutoAktiv) AND (NOT AutoFun3Aktiv) THEN
0026     PinStop := TRUE;
0027   END_IF
0028   IF PinAktiv AND (T_AutoPinCloseStop OR FTastePinOpen OR FTastePinClose OR
0029     PosFlankPinOpenKey.Q OR posFlankPinCloseKey.Q) THEN
0030     PinAutoAktiv := FALSE;
0031     PinCloseAktiv := TRUE;
0032   END_IF
0033 END_IF
0034
0035
0036 IF (P_SollCurrentEichung / 10) <= IstCurrent THEN
0037   PinStop := TRUE;
0038 END_IF
0039
0040 IF PinStop THEN
0041   PinStop := FALSE;
0042   PinModus := 99;
0043 END_IF
0044
0045 (* Ablauf *)
0046 CASE PinModus OF
0047   0 :
0048     PinStop := FALSE;
```

The status bar at the bottom of the editor shows the following loading messages:

- Loading library 'C:\Arquivos de programas\WAGO Software\CoDeSys V2.3\Upload\G3_CANopenMaster.lib'
- Loading library 'C:\Arquivos de programas\WAGO Software\CoDeSys V2.3\Upload\G3_CANopenManager.lib'
- Loading library 'C:\Arquivos de programas\WAGO Software\CoDeSys V2.3\Upload\G3_CANdrv.lib'

Exemplo de tela de programação de Elrest – (Eladesign – Codesys 2.3)

```
0001 PROGRAM RemoteFun4
0002 VAR
0003   dummy: BOOL;
0004 END_VAR
0005
0006 (* RemoteFun4 *)
0007 NachHost_WDHost := VonHost_WDHost;
0008 VonHost_InitBefehl := VonHost_InitBefehl;
0009
0010 IF P_RemoteBetrieb = 0 THEN
0011   RemoteTempReglerNotAktiv := FALSE;
0012   RemoteBetrieb := FALSE;
0013 ELSE
0014   RemoteBetrieb := TRUE;
0015
0016 RemoteBetriebAktiv := VonHost_RemoteBetrieb AND PuitMaskActive(MaskeRemoteWechsel);
0017 IF VonHost_RemoteBetrieb AND (NOT TimerWDSlave.Q) THEN
0018   AnzRemote := 1;
0019 END_IF
0020 IF (NOT VonHost_RemoteBetrieb) AND (NOT TimerWDSlave.Q) THEN
0021   AnzRemote := 0;
0022 END_IF
0023 IF (TimerWDSlave.Q) THEN
0024   AnzRemote := 2;
0025 END_IF
0026
0027 NachHost_Remote := RemoteBetriebAktiv;
0028 NachHost_RemoteAktiv := NachHost_Remote AND VonHost_RemoteBetrieb;
0029
0030 TimerWDSlave (IN := (VonHost_WDSlave <> NachHost_WDSlave), PT := P_TimeWDSlave);
0031 IstTimeWDSlave := TIME_TO_REAL(TimerWDSlave.Et);
0032 IF VonHost_WDSlave = NachHost_WDSlave THEN
0033   NachHost_WDSlave := NachHost_WDSlave + 1;
0034 END_IF
0035
0036 RT_TimerWDSlave(CLK:=TimerWDSlave.Q);
0037 IF RT_TimerWDSlave.Q THEN
0038   D_RemoteZeitWD := DT_TO_STRING(SysRtcGetTime(dummy));
0039 END_IF
0040 IF TimerWDSlave.Q THEN
0041   VonHost_RemoteBetrieb := FALSE;
0042 END_IF
0043
0044 IF RemoteBetriebAktiv THEN
0045   CylLen := VonHost_CylLaenge;
0046   CylUmfang := VonHost_CylUmfang;
0047   Soillayer := VonHost_Soillayer;
0048
0049 IF ((VonHost_RemoteBefehl = S_Vorprozess) OR (VonHost_RemoteBefehl = S_Runprozess)) AND
0050 (VonHost_InitBefehl) THEN
0051   Err_HostDaten := (CylLen < P_CylLenMin) OR (CylLen > P_CylLenMax) OR
```

Exemplo de tela de programação Schneider – (SoMachine - Codesys 3.5)

The screenshot displays the Schneider SoMachine Codesys 3.5 software interface. The main window shows a ladder logic program for a current regulator, titled "CurrentReglerFun". The program includes logic for setting the regulator mode based on various inputs and conditions.

```
1 (* CurrentReglerFun *)
2
3 IF CurrentReglerStop[FALSE] THEN
4   CurrentReglerStop[FALSE] := FALSE;
5   CurrentReglerModus[0] := 99;
6 END_IF
7
8 CASE CurrentReglerModus[0] OF
9   0 :
10    CurrentReglerStop[FALSE] := FALSE;
11    IF CurrentReglerAktiv[FALSE] THEN
12      CurrentReglerModus[0] := M_InitCurrentRegler[0_1];
13    END_IF
14
15    M_InitCurrentRegler[0_1] :
16    IF (P_CurrentIMin[0] = 0) AND (P_CurrentUMin[0] = 0) THEN
17      CurrentReglerModus[0] := M_Motorisch[0_2];
18    END_IF
19
20    IF (P_CurrentIMin[0] = 0) AND (P_CurrentUMin[0] <> 0) THEN
21      CurrentReglerModus[0] := M_Spannung[0_3];
22
23      ReglerCurrent[0] := P_SollCurrentEichung[0] * P_CurrentUMin[0] / 100;
24    END_IF
25
26    IF (P_CurrentIMin[0] <> 0) AND (P_CurrentUMin[0] = 0) THEN
27      CurrentReglerModus[0] := M_Strom[0_4];
28
29      ReglerCurrent[0] := SollCurrent[0] * P_CurrentIMin[0] / 100;
30    END_IF
31
32 (*****
33    M_Motorisch[0_2] :
34      FunktionAktiv[FALSE] := FALSE;
35      A_CurrentOn[FALSE] := TRUE;
36
37 (* Strom Plus *)
38    IF ((SollCurrent[0] - CurrentDiff2In[0]) > IstCurrent[0]) AND (NOT FunktionAktiv[FALSE]) THEN
39      FunktionAktiv[FALSE] := TRUE;
40      CurrentPlusTakter[FALSE] := FALSE;
41      CurrentMinusTakter[FALSE] := FALSE;
42
43      A_CurrentDown[FALSE] := FALSE;
44      A_CurrentUp[FALSE] := TRUE;
```

The interface also shows a project tree on the left, a controller list on the right, and a status bar at the bottom indicating the current state (e.g., "RUN", "SIMULATION").

Exemplo de tela de programação Schneider – (SoMachine - Codesys 3.5)

The screenshot displays the Schneider SoMachine Codesys 3.5 software interface. The main window shows a ladder logic program for configuring an IP address. The code includes comments in Portuguese and Ladder Logic (LAD) instructions. The program defines pointers for receiving and transmitting data, checks for Ethernet link status, and configures UDP peers for a specific local IP address and port. The status variable 'state' is used to track the configuration process.

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
```

Watch 1

Expression	Type	Value	Prepared value	Address	Comment

Messages - Totally 0 error(s), 25 warning(s), 9 message(s)

Last build: 0 25 Precompile: RUN SIMULATION Program loaded Program unchanged Current user: (nobody)

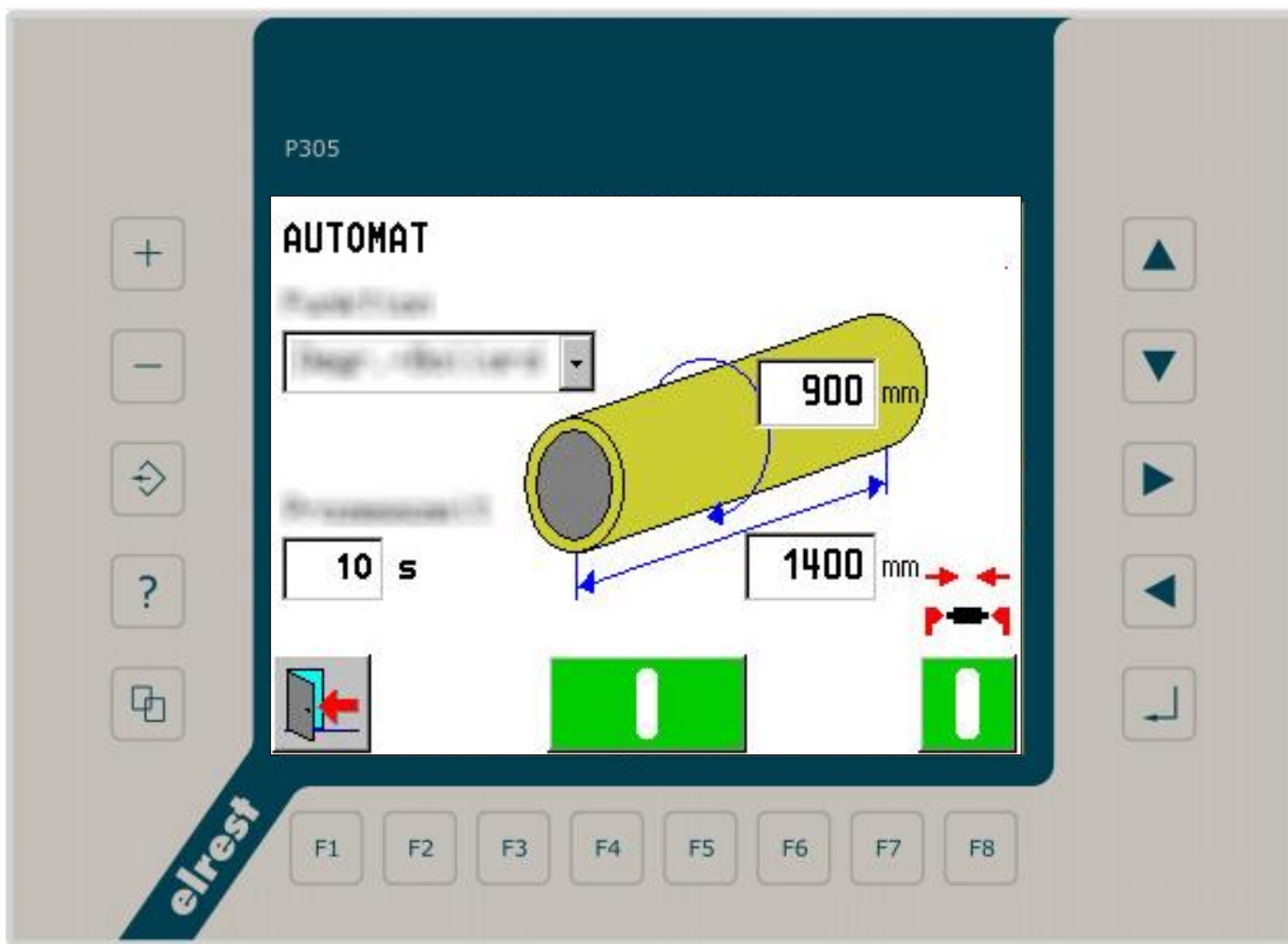
Exemplo de tela na IHM Elrest



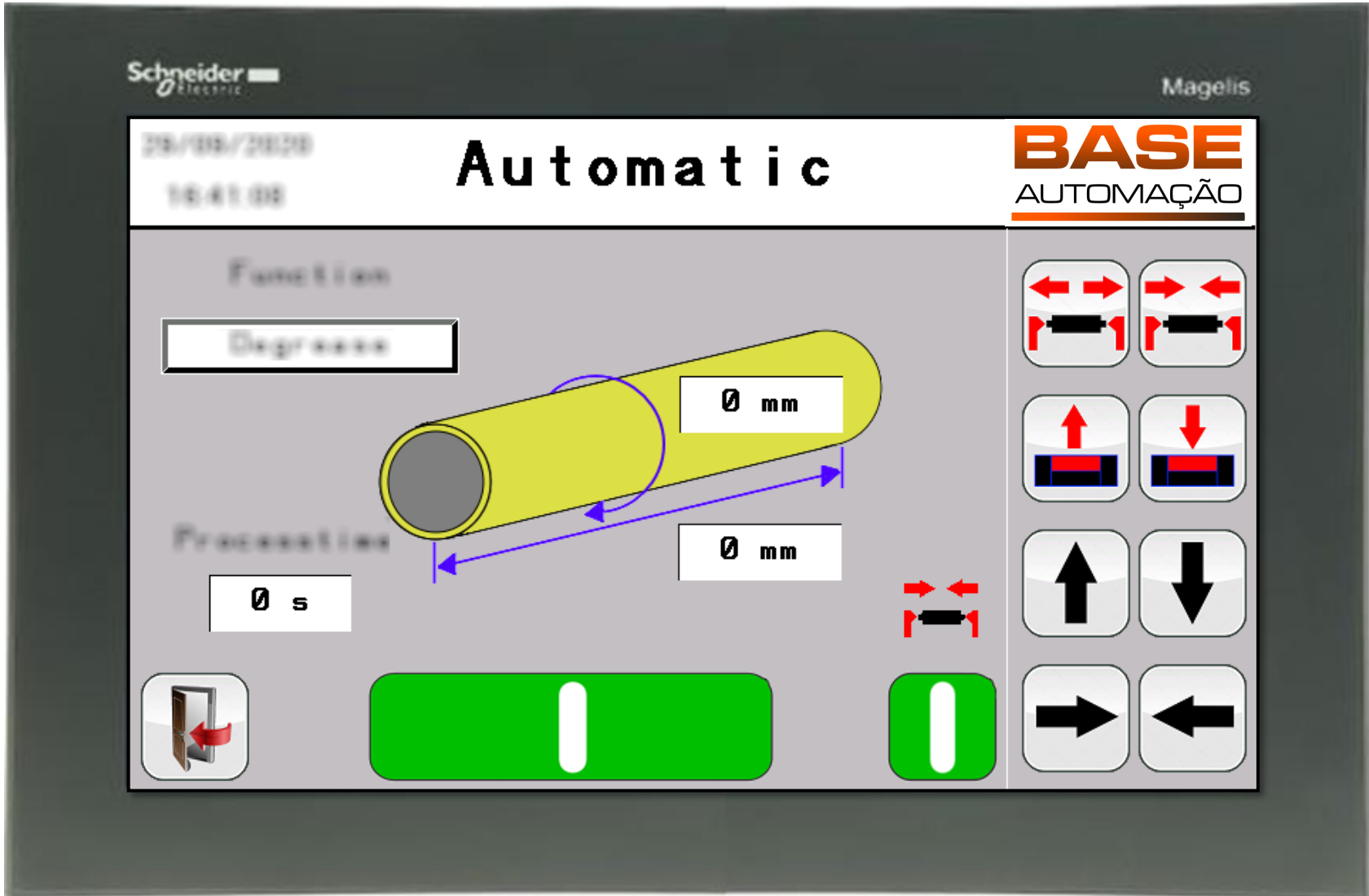
Exemplo de tela convertida para IHM Schneider



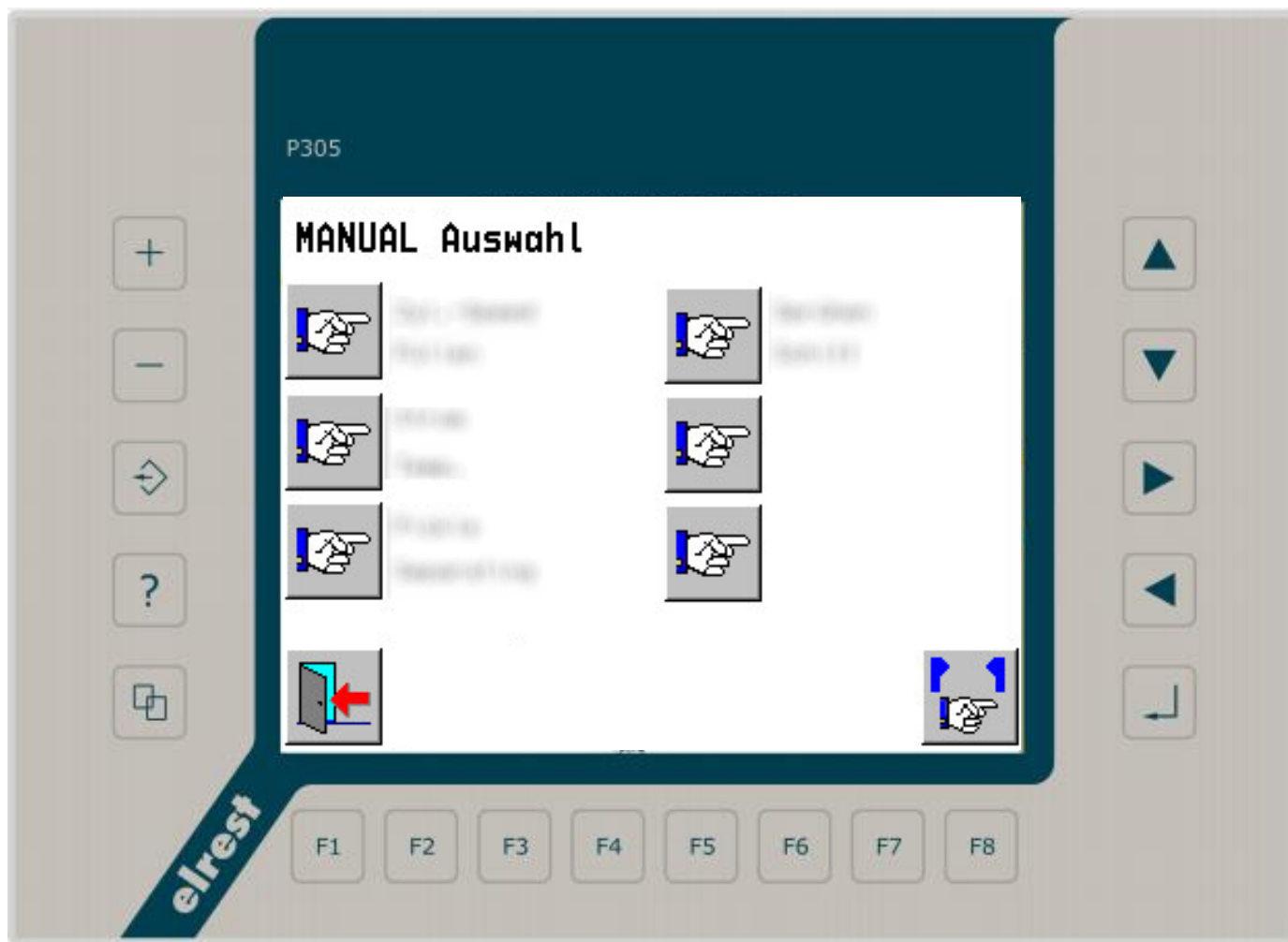
Exemplo de tela na IHM Elrest



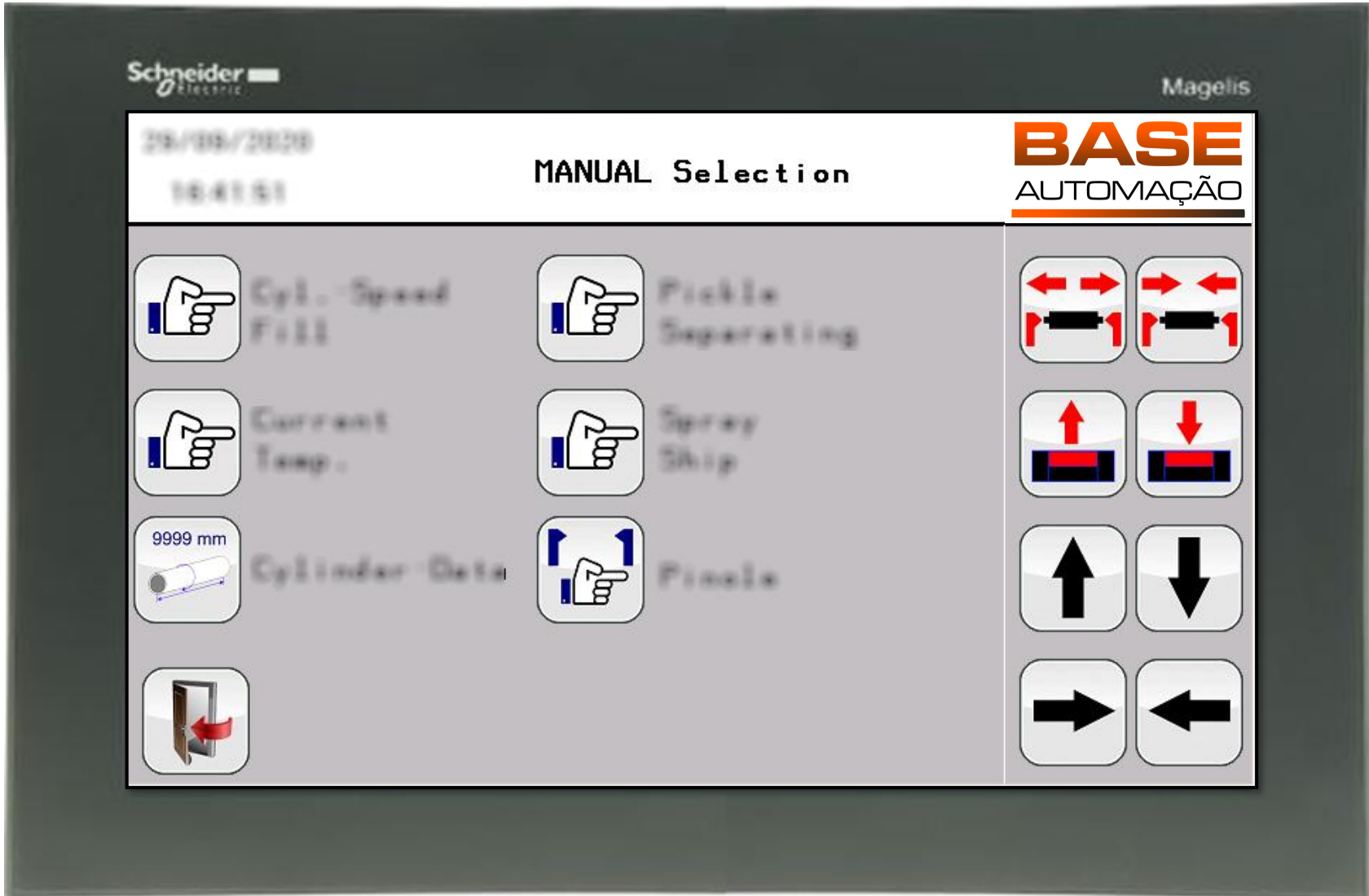
Exemplo de tela convertida para IHM Schneider



Exemplo de tela na IHM Elrest



Exemplo de tela convertida para IHM Schneider



Vantagens da migração dos hardwares e softwares

- Expansão da memória.
- Aumento da velocidade de processamento.
- Maiores possibilidades de programação.
- Inclusão de equipamentos modernos.
- Agilidade e facilidade da rede ethernet.
- Desenvolvimento e manutenção mais acessível.

Contatos



BASE

AUTOMAÇÃO

 **Telefones:** (11) 4456-4321 / (11) 4456-1408 / (11) 97885-1596

 **WhatsApp:** (11) 4456-4321 / (11) 97885-1596

 **E-mail:** comercial@baseautomacao.com.br

 **Site:** baseautomacao.com.br

 **Catálogo virtual:** baseautomacao.com.br/loja

    **/baseautomacao**