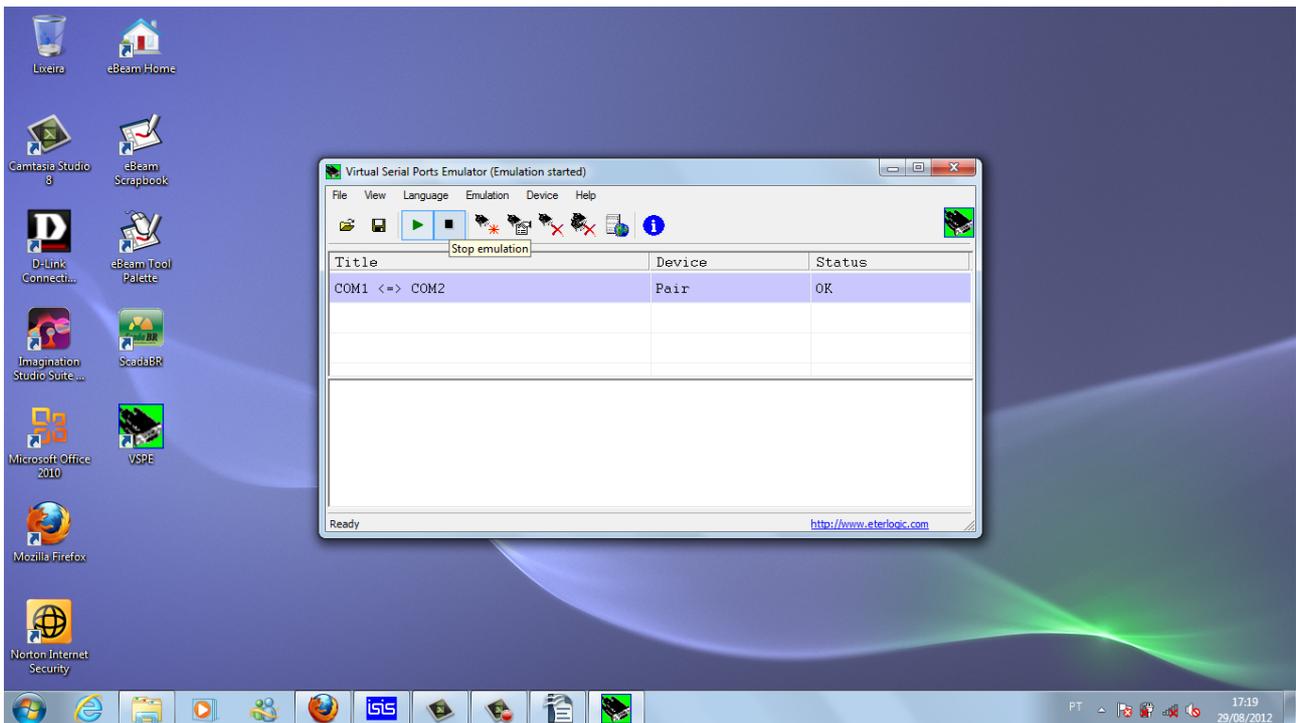


## MODBUS COM ARDUINO

Vamos fazer um Atmega328P comunicar-se com o ScadaBR. Eu testei esse programa com um Arduino Uno e funcionou muito bem. Para obter mais praticidade, resolvi utilizar o Proteus para simular o arduino Uno. A comunicação com o ScadaBR vai fazer uso do protocolo Modbus, vamos precisar também de um emulador de porta serial - o Virtual Serial Port Emulator ou outro equivalente, no meu caso, criei um par de portas seriais COM1 e COM2. A COM1 ficou sendo utilizada pelo componente COMPIM do Proteus e a COM2 ficou sendo utilizada pelo ScadaBR.

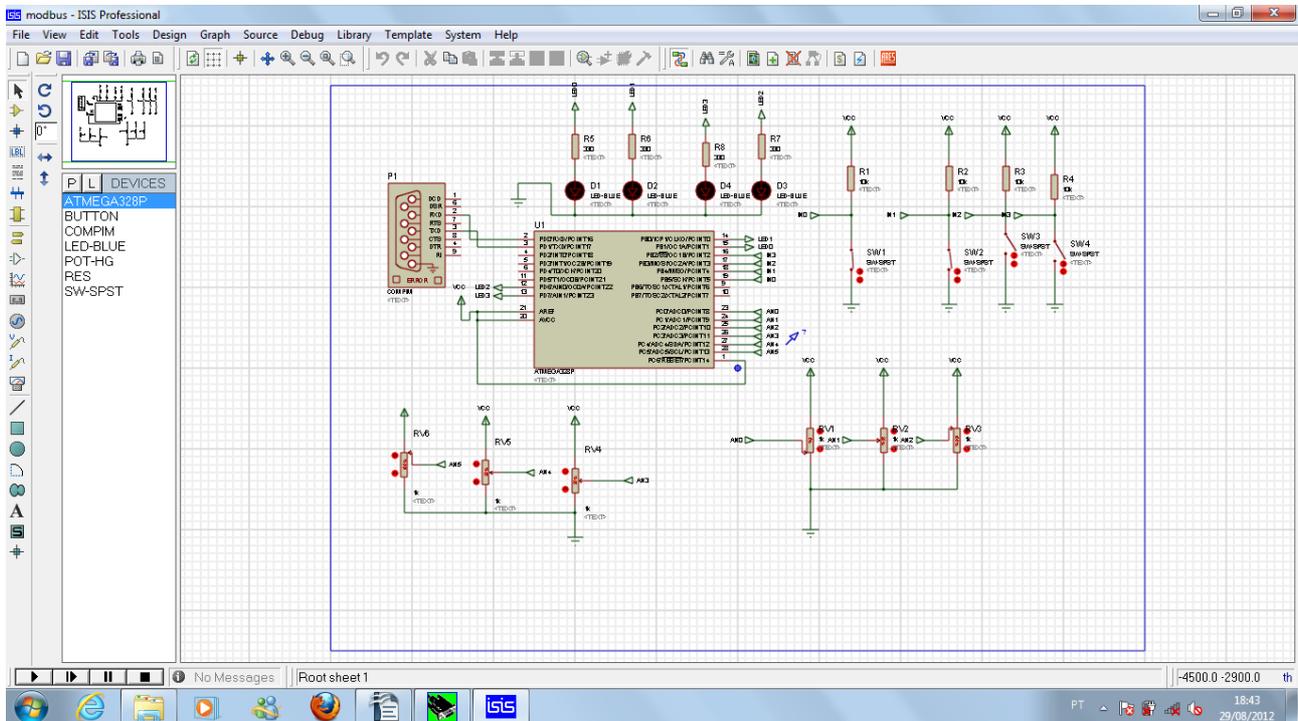
Na tela abaixo, temos o Virtual Serial Port Emulator apresentado o par de portas recém-criadas.



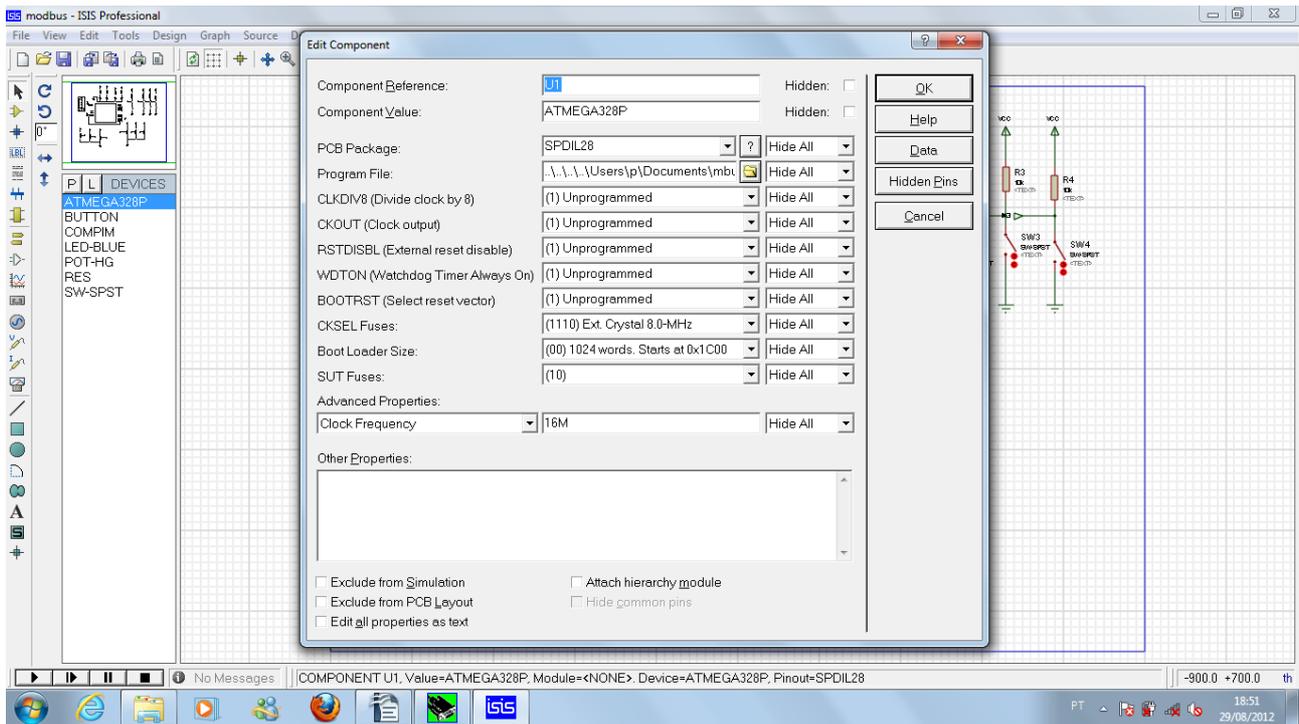
Agora precisamos criar um pequeno circuito no Proteus com os componentes exibidos abaixo. Devemos ter chaves e potenciômetros para simular sensores e ligá-los aos pinos digitais e às entradas analógicas bem como ligar leds para simular as cargas que desejamos acionar. O nosso circuito será capaz de lê o estado lógico de 4 chaves, lê

[www.mundoarduino.com](http://www.mundoarduino.com)

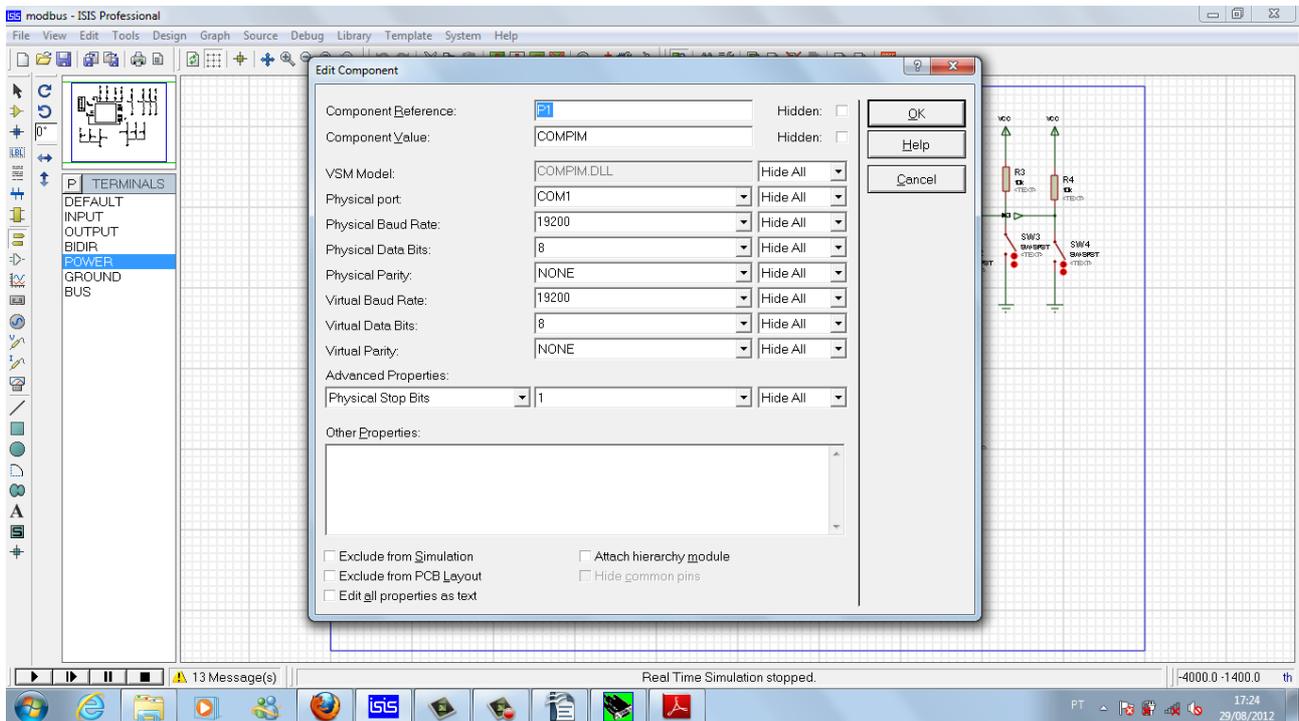
a tensão presente em 6 entradas analógicas e acionar 4 cargas. Também não devemos esquecer de adicionar o componente COMPIM responsável pela comunicação serial com o ScadaBR. Fazemos as ligações conforme a figura abaixo:



Clicando no Atmega328P podemos configurar o micro para executar o arquivo hex responsável pela implementação do protocolo Modbus. Você pode fazer o download do arquivo no site [www.mundoarduino.com](http://www.mundoarduino.com).



O componente COMPIM deverá ser configurado conforme a figura abaixo

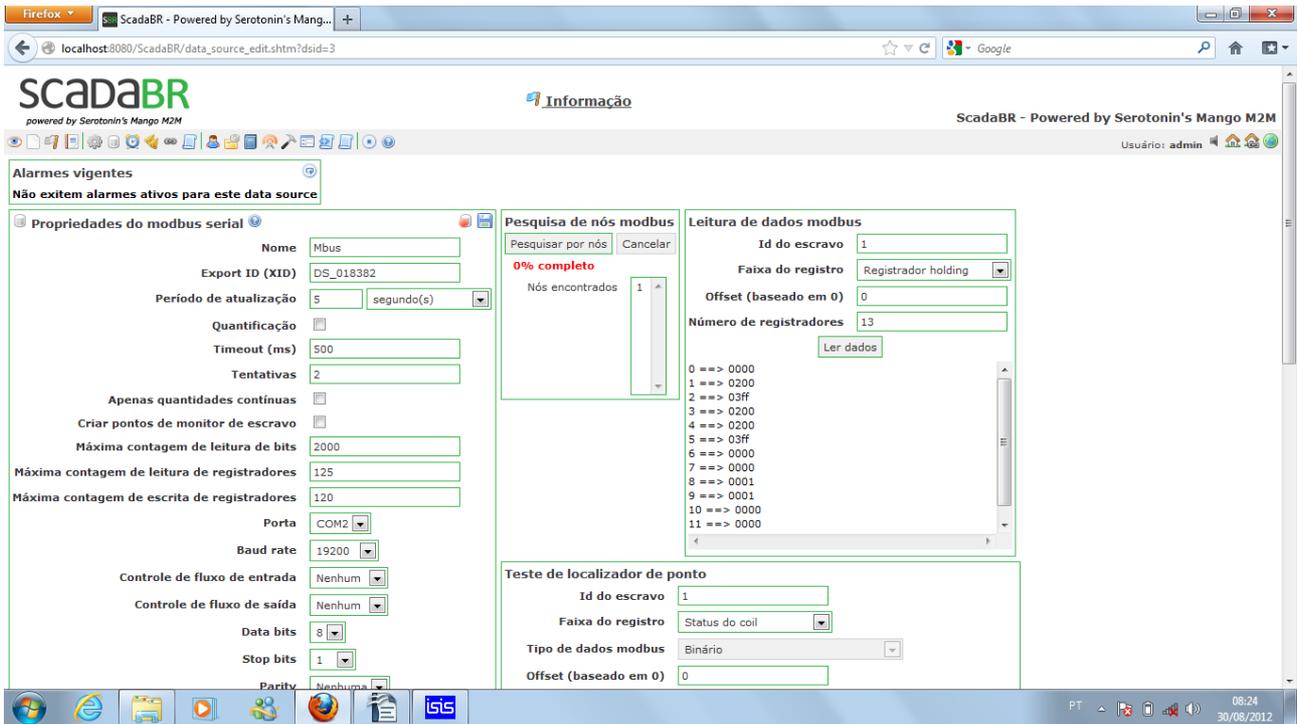


O ScadaBR ficará responsável pela exibição dos estados das chaves, leitura das portas analógicas e pela atuação das cargas, no nosso caso, as cargas são apenas leds.

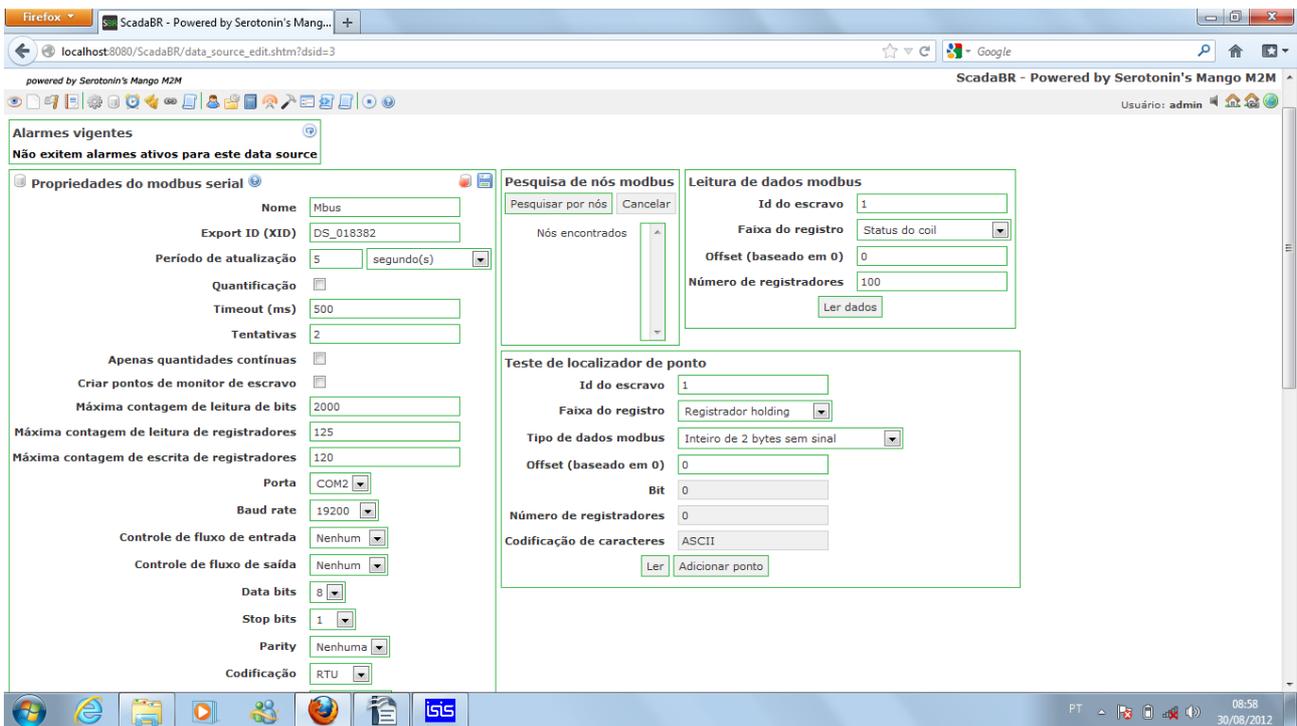
Supondo que você já tenha o ScadaBR instalado, devemos criar um Data Sources para servir de fonte de dados para o ScadaBR. Vamos adicionar um Data Sources do tipo Modbus Serial.



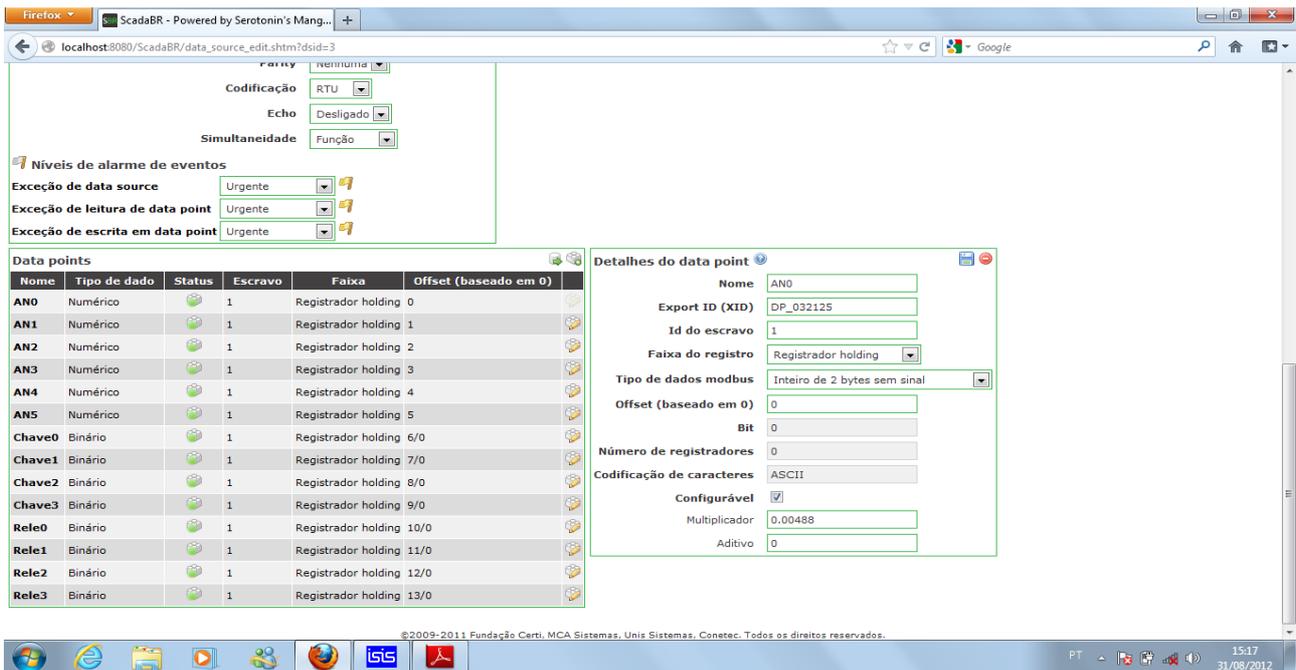
Clicando no ícone do **Data Sources** e depois selecionando tipo **Modbus Serial**, podemos testar a comunicação com o Proteus. Devemos nos assegurar que o Modbus Serial esteja utilizando a porta serial e velocidade correta. Se a simulação no Proteus estiver rodando, ao clicarmos em **Pesquisar por nós** (desabilite o Data Sources antes de executar a pesquisa), será localizado o escravo 1 que é o atmega328P rodando no Proteus. Clicando em **Ler dados** se selecionarmos a Faixa de registros apropriada - **Registrador holding**, bem como o número de registradores que estamos utilizando no nosso caso 14 registradores: 04 para as pinos digitais, 6 para as entradas analógicas e 4 para as cargas, nós teremos acesso direto ao valores atuais dos registradores conforme vemos na figura abaixo:



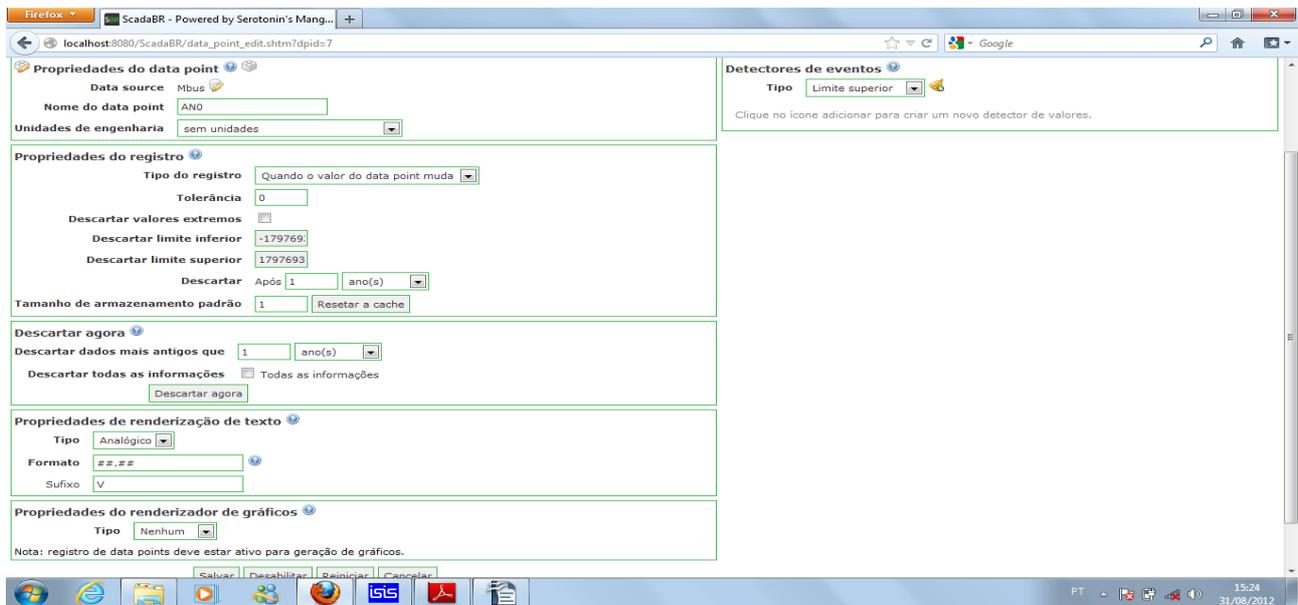
Mas isso não é tudo, nós devemos criar Data um Data Points para cada registrador .No exemplo abaixo, criamos um Data Point para lê a entrada analógica do Atmega328P.



Quando clicarmos em adicionar ponto, estaremos preparando um **Data Point** para o ScadaBR para lê a entrada ADC0 (pino 23) do Atmega328P. Se desejarmos lê outra entrada analógica devemos da mudar o campo **Offset(baseado em 0)**, para 1 e assim sucessivamente até o valor 5, quando então estaremos lendo todas as entradas analógicas do Atmega328P.Veja na figura abaixo, a configuração do data Point para lê a entrada analógica. Para as entradas analógicas podemos inserir um fator multiplicativo e assim lermos as tensões presentes nos pinos variando de 0V a 5V e não de 0 a1023.



Clicando em **Watch List – Detalhes do Data Point – Editar data Point** poderemos atribuir uma máscara e um sufixo para a grandeza que estamos medindo.Veja a figura abaixo:



Vamos discutir mais sobre o código que roda no micro em outra oportunidade, mas uma boa olhada no trecho abaixo, dá a compreensão necessária para estabelecer o **Offset** necessário para configurar os Data Points tanto para as entradas digitais , analógicas, bem como as rotinas necessárias para escrever nos pinos do Atmega328P e atuar cargas. O programa completo pode ser baixado na Internet – pesquise no google “modbus slave arduino” e encontrará facilmente o arquivo .pde que implementa o protocolo modbus. O trecho de código abaixo contém a parte realmente importante com as modificações necessárias para trabalhar com o Atmega328P.

```
enum {
    MB_SLAVE = 1, /* modbus slave id */
};
/* slave registers example */
enum {
    MB_REG0,
    MB_REG1,
    MB_REG2,
    MB_REG3,
    MB_REG4,
    MB_REG5,
```

```
MB_REG6,  
MB_REG7,  
MB_REG8,  
MB_REG9,  
MB_REG10,  
MB_REG11,  
MB_REG12,  
MB_REG13,  
MB_REGS
```

```
/* total number of registers on slave */
```

```
};
```

```
int regs[MB_REGS]; /* this is the slave's modbus data map */
```

```
void setup()
```

```
{
```

```
/* Modbus setup example, the master must use the same COM parameters */
```

```
/* 115200 bps, 8N1, two-device network */
```

```
configure_mb_slave(19200, 'n', 0);
```

```
boolean chave0=13;
```

```
boolean chave1=12;
```

```
boolean chave2=11;
```

```
boolean chave3=10;
```

```
//boolean rele0=9;
```

```
//boolean rele1=8;
```

```
//boolean rele2=7;
```

```
// boolean rele3=6;
```

```
pinMode(13,INPUT);
```

```
pinMode(12,INPUT);
```

```
pinMode(11,INPUT);
```

```
pinMode(10,INPUT);
```

```
pinMode(9,OUTPUT);
```

```

pinMode(8,OUTPUT);
pinMode(7,OUTPUT);
pinMode(6,OUTPUT);

}

void loop()
{
    /* This is all for the Modbus slave */
    update_mb_slave(MB_SLAVE, regs, MB_REGS);

    regs[MB_REG0]=analogRead(0);
    regs[MB_REG1]=analogRead(1);
    regs[MB_REG2]=analogRead(2);
    regs[MB_REG3]=analogRead(3);
    regs[MB_REG4]=analogRead(4);
    regs[MB_REG5]=analogRead(5);
    regs[MB_REG6]=digitalRead(13);
    regs[MB_REG7]=digitalRead(12);
    regs[MB_REG8]=digitalRead(11);
    regs[MB_REG9]=digitalRead(10);

    switch ( regs[MB_REG10] ) {
        case 1:

            digitalWrite(9,HIGH);
            break;
        case 0:
            digitalWrite(9,LOW);
            break;
        default:
            digitalWrite(9,LOW);
    }

    switch ( regs[MB_REG11] ) {
        case 1:

```

```
    digitalWrite(8,HIGH);
    break;
case 0:
    digitalWrite(8,LOW);
    break;
default:
    digitalWrite(8,LOW);
}
```

```
switch ( regs[MB_REG12] ) {
  case 1:

    digitalWrite(7,HIGH);
    break;
  case 0:
    digitalWrite(7,LOW);
    break;
  default:
    digitalWrite(7,LOW);
}
```

```
switch ( regs[MB_REG13] ) {
  case 1:

    digitalWrite(6,HIGH);
    break;
  case 0:
    digitalWrite(6,LOW);
    break;
  default:
    digitalWrite(6,LOW);
}
```

Veja o exemplo de configuração de um Data Point para lê uma chave com o Atmega328P.

The screenshot shows the ScadaBR web interface in a Firefox browser window. The main configuration area is titled 'Níveis de alarme de eventos' and includes dropdown menus for 'Exceção de data source', 'Exceção de leitura de data point', and 'Exceção de escrita em data point', all set to 'Urgente'. Below this is a table of 'Data points' and a 'Detalhes do data point' panel.

Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)
AN0	Númérico		1	Registrador holding 0	0
AN1	Númérico		1	Registrador holding 1	1
AN2	Númérico		1	Registrador holding 2	2
AN3	Númérico		1	Registrador holding 3	3
AN4	Númérico		1	Registrador holding 4	4
AN5	Númérico		1	Registrador holding 5	5
Chave0	Binário		1	Registrador holding 6/0	6/0
Chave1	Binário		1	Registrador holding 7/0	7/0
Chave2	Binário		1	Registrador holding 8/0	8/0
Chave3	Binário		1	Registrador holding 9/0	9/0
Rele0	Binário		1	Registrador holding 10/0	10/0
Rele1	Binário		1	Registrador holding 11/0	11/0
Rele2	Binário		1	Registrador holding 12/0	12/0
Rele3	Binário		1	Registrador holding 13/0	13/0

The 'Detalhes do data point' panel for 'Chave0' shows the following configuration:

- Nome: Chave0
- Export ID (XID): DP\_971194
- Id do escravo: 1
- Faixa do registro: Registrador holding
- Tipo de dados modbus: Binário
- Offset (baseado em 0): 6
- Bit: 0
- Número de registradores: 0
- Codificação de caracteres: ASCII
- Configurável:
- Multiplicador: 1
- Aditivo: 0

E abaixo a configuração de um Data Point para escrever no Atmega328P um valor binário e assim poder comandar cargas.

The screenshot shows the ScadaBR web interface for configuring data points. The main configuration area includes:

- Codificação:** RTU
- Echo:** Desligado
- Simultaneidade:** Função
- Níveis de alarme de eventos:**
  - Exceção de data source: Urgente
  - Exceção de leitura de data point: Urgente
  - Exceção de escrita em data point: Urgente

The **Data points** table is as follows:

Nome	Tipo de dado	Status	Escravo	Faixa	Offset (baseado em 0)
AN0	Numérico		1	Registrador holding	0
AN1	Numérico		1	Registrador holding	1
AN2	Numérico		1	Registrador holding	2
AN3	Numérico		1	Registrador holding	3
AN4	Numérico		1	Registrador holding	4
AN5	Numérico		1	Registrador holding	5
Chave0	Binário		1	Registrador holding	6/0
Chave1	Binário		1	Registrador holding	7/0
Chave2	Binário		1	Registrador holding	8/0
Chave3	Binário		1	Registrador holding	9/0
Rele0	Binário		1	Registrador holding	10/0
Rele1	Binário		1	Registrador holding	11/0
Rele2	Binário		1	Registrador holding	12/0
Rele3	Binário		1	Registrador holding	13/0

The **Detalhes do data point** panel for 'Rele0' shows the following configuration:

- Nome: Rele0
- Export ID (XID): DP\_253713
- Id do escravo: 1
- Faixa do registro: Registrador holding
- Tipo de dados modbus: Binário
- Offset (baseado em 0): 10
- Bit: 0
- Número de registradores: 0
- Codificação de caracteres: ASCII
- Configurável:
- Multiplicador: 1
- Aditivo: 0

O problema então se resume em ter o cuidado de atribuir o número do escravo, Offset, tipo de dados adequados corretos aos Data Points e para fazer tudo isso, basta observar bem o trecho do código postado nesse tutorial. Obviamente você também é livre para baixar da internet a biblioteca Modbus para Arduino e alterá-la para atender suas necessidades.

Por fim, clicando em **Watch list** você poderá acrescentar todos os Data Points para visualizar aos valores neles armazenados bem como alterar o nível lógico dos pinos configurados como saída e acionar as cargas a eles ligadas. Veja a figura abaixo:

The screenshot shows the ScadaBR web interface in a Firefox browser. The page title is "ScadaBR - Powered by Serotonin's Mango M2M". The URL is localhost8080/ScadaBR/watch\_list.shtm. The user is logged in as "admin".

On the left, there is a "Points" tree view showing a hierarchy of data points: Mbus - AN0 through AN5, Mbus - Chave0 through Chave3, and Mbus - Rele0 through Rele3.

The main area displays a "Watch list" table with the following data:

Point Name	Value	Timestamp	Actions
Mbus - AN0	2,75V	15:08:15	[Icons]
Mbus - AN1	3,6V	15:08:15	[Icons]
Mbus - AN2	2,7V	15:08:15	[Icons]
Mbus - AN3	2,3V	15:08:15	[Icons]
Mbus - AN4	2V	15:08:15	[Icons]
Mbus - AN5	2,5V	15:08:15	[Icons]
Mbus - Chave0	0	15:08:15	[Icons]
Mbus - Chave1	0	15:08:15	[Icons]
Mbus - Chave2	0	15:08:15	[Icons]
Mbus - Chave3	0	15:08:15	[Icons]
Mbus - Rele0	1	15:08:15	[Icons]
Mbus - Rele1	1	15:08:15	[Icons]
Mbus - Rele2	1	15:08:15	[Icons]
Mbus - Rele3	1	15:08:15	[Icons]

At the bottom, there is a "Gráfico" section with date and time filters: "De 2012 Ago 30 15:08" and "Até 2012 Ago 31 15:00". The system clock shows 15:08 on 31/08/2012.

