*e*HOUSE

São Paulo - SP

MICROPROCESSADORES

Alunos:

Iris de Oliveira Corcino dos Santos - 31652964

Arthur Henry F dos Santos - 31672914

Tainá Akemy Fonseca Ohats - 31616623

Gabriel de Sena Matos dos Santos - 31621120

Thamires Furtuoso Alexandre - 31687121

Danilo Kenji Assunção Katayama - 31621961

1 INTRODUÇÃO

O objetivo deste projeto é a construção de um supervisório que ofereça maior acessibilidade de controle residencial ao usuário por qualquer dispositivo conectado em um modem “internet” através de um "IP". Para isso foi desenvolvido um sistema utilizando principalmente o Software ScadaBR, o microcontrolador Arduino e o RaspBerry PI para interface gráfica.

Neste trabalho será descrito o desenvolvimento da interação da interface do scadaBR com o Arduino e os meios para que seja possível a comunicação entre ambos, assim como as configurações de código, biblioteca e plataforma.

Apesar da simplicidade das aplicações apresentadas, o desenvolvedor/usuário tem a possibilidade de ampliar e adaptar seu projeto pessoal para outros ambientes e necessidades a partir do que foi aqui apresentado.

2 Lista de Materiais

- ARDUINO UNO
- RASPBERRY
- SHIELD MAX 485
- 6x 1K OHM
- 6x RELES JQC-3F(T73) 5V
- 6 TRANSISTORES BC548
- 1 SERVO SG90
- 1 COOLER 5V
- FITA DE LED OU LED INDIVIDUAL
- DHT11 (SENSOR DE TEMPERATURA E HUMIDADE)
- CONVERSOR ADAPTADOR USB PARA RS485 BORNE 2 PINOS

3 Sensor de Temperatura

O Sensor de Umidade e Temperatura DHT11 é um sensor de temperatura e umidade que permite fazer leituras de temperaturas entre 0 a 50 Celsius e umidade entre 20 a 90

O elemento sensor de temperatura é um termistor do tipo NTC e o sensor de Umidade é do tipo HR202, o circuito interno faz a leitura dos sensores e se comunica a um microcontrolador através de um sinal serial de uma via. Nas figuras 1 pode-se observar as pinagens.
Especificações:
– Modelo: DHT11 (Datasheet)
– Faixa de medição de umidade: 20 a 90– Faixa de medição de temperatura: 0º a 50ºC
– Alimentação: 3-5VDC (5,5VDC máximo)
– Corrente: 200uA a 500mA, em stand by de 100uA a 150 uA
– Precisão de umidade de medição: 5,0– Precisão de medição de temperatura: 2.0 ºC
– Tempo de resposta: 2s

– Dimensões: 23 x 12 x 5mm (incluindo terminais)

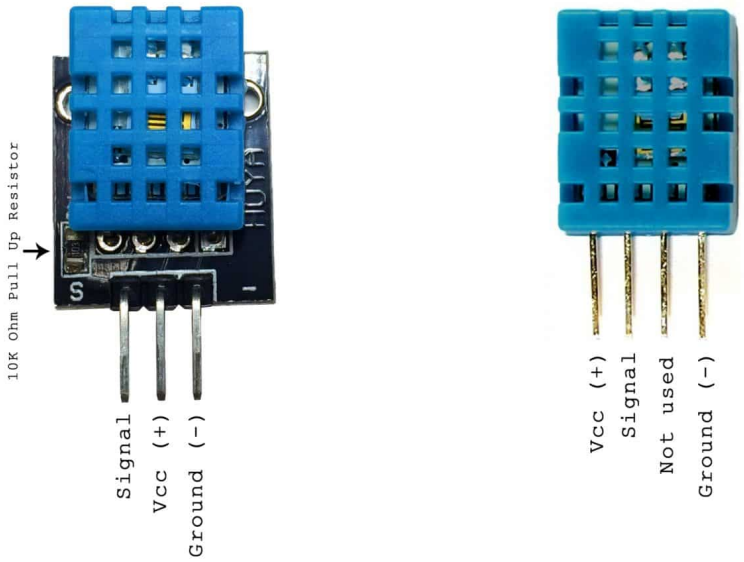

Figura 1: Sensor de Temperatura.

4 Interface de Comunicação

A comunicação do projeto é feita na seguinte ordem:
O Usuário aciona através da Tela gráfica, que é acessada pelo link do ScadaBr, o comando que deseja (acender/apagar a luz, ou acionar o sensor de temperatura e humidade). Este comando é enviado para um moldem que através de uma comunicação Wi-fi com o Raspberry (onde está o ScadaBR). O raspberry irá fazer a leitura e enviará o comando para que o arduíno faça a execução. Os dados passados pelo arduíno são convertidos através de um RS485 que é uma interface de comunicação Half duplex, ou seja só faz a emissão ou recebimento de dados por vez, através de uma Shield Max 485 fará a conversão da linguagem do RS485 (analógica) na linguagem RX TX do arduíno. Para essa conversão são ligadas as saídas A e B da shield Max485 nos pinos RXTX do arduíno e curto circuitam as outras 2 saídas do Max485 e conectam a um outro pino qualquer do arduíno que fará o comando de entrada e saída. E o arduíno fará a execução do comando para a casa.

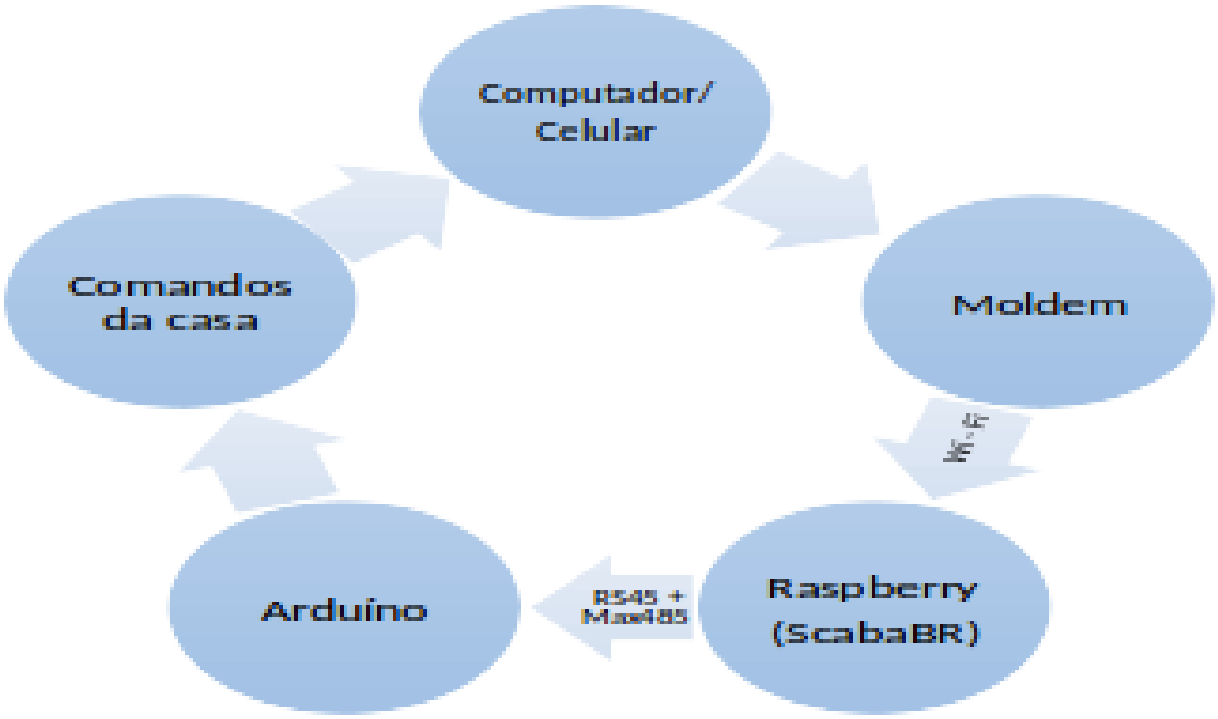

Figura 2: Fluxograma de Comunicação.

5 ARDUINO

5.1 Bibliotecas

Em nosso trabalho foram utilizadas as bibliotecas:

```
#include <Modbus.h>
#include <ModbusSerial.h>
#include <Servo.h>
#include <DHT.h>
```

Sendo que as bibliotecas Modbus são complementares e funcionam para protocolar nossa comunicação RS485 com a RX, TX do arduíno usando como auxílio o MAX485. Foi utilizado o RS485 porque ele suporta até 32 escravos e uma rede de 150 m, ele é muito utilizado na industria e seu principal uso é em automações residenciais. A biblioteca para o servomecanismo (Servo.h) envia os comandos pela forma de PWM na qual os parâmetros importantes são a frequência com que o pulso se repete (geralmente a cada 20ms), e a duração de cada pulso,

que indica o ângulo para o qual o servo deve apontar.

5.2 Explicação do Código

Para que seja possível a comunicação entre o ScadaBr e o arduíno é necessário a criação de endereços para os sinais que queremos monitorar e/ou controlar.

```
const int L1 = 0; // Endere o do L1 - L6 respons veis pelo controle dos LED s
    tipo Coil (Bin rio OUTPUT)
const int L2 = 1;
const int L3 = 2;
const int L4 = 3;
const int L5 = 4;
const int L6 = 5;
const int ServoMOTOR = 6; // Endere o do Servo Motor tipo Status do Coil (
    Bin rio OUTPUT)
const int COOLER = 7; // Endere o do Cooler tipo Status do Coil (Bin rio
    OUTPUT)
const int temp = 8; // Endere o da temperatura tipo Registrador de Entrada (
    Anal gico INPUT)
const int umid = 9;// Endere o da umidade tipo Registrador de Entrada (
    Anal gico INPUT)
```

As constantes inteiras atribuídas aos nossos componentes, são os endereçamentos escolhidos para cada um deles, ou seja, quando o ScadaBr receber a indicação de que é necessário enviar o comando de “0” (desliga) ou “1” (liga) para o endereçamento 1, a lâmpada 2 (L2) é que receberá esse comando.

```
mb.addCoil(L1);
  mb.addCoil(L2)
  mb.addCoil(L3);
  mb.addCoil(L4);
  mb.addCoil(L5);
  mb.addCoil(L6);
  mb.addCoil(ServoMOTOR);
  mb.addCoil(COOLER);
  mb.addIreg(temp);
  mb.addIreg(umid);

servo1.attach(pinServo); // Configura  o do pino do servo motor
  pinMode(Lamp1,OUTPUT); // pinos de sa da e Entrada
  pinMode(Lamp2,OUTPUT);
  pinMode(Lamp3,OUTPUT);
  pinMode(Lamp4,OUTPUT);
  pinMode(Lamp5,OUTPUT);
  pinMode(Lamp6,OUTPUT);
  pinMode(pinServo,OUTPUT);
  pinMode(pinCooler,OUTPUT);
  pinMode(pinDHT,INPUT);

}
```

Só indicar um endereço para cada componente não é suficiente para indicar o tipo de sinal

que este emite ou recebe, portanto com o trecho de programação descrito acima é atribuído o tipo de sinal à cada componente, no caso, os leds, o motor e o cooler, são componentes do tipo sinal output coil, ou seja eles recebem, do ScadaBr, o comando de 0 ou 1 que indicam se eles devem ligar ou desligar, abrir ou fechar (no caso do motor), já a temperatura e a umidade, servem para indicar um valor, ou seja eles são do tipo registradores input, já que recebem um dado externo e o registram na tela gráfica do ScadaBr.

```
void loop() {
  mb.task(); // In cio da comunica  o serial
  digitalWrite(Lamp1, mb.Coil(L1)); // Atribuindo o valor da vari vel L1 ao
    pino Lamp1
digitalWrite(Lamp2, mb.Coil(L2)); // Atribuindo o valor da vari vel L2 ao pino
    Lamp2
digitalWrite(Lamp3, mb.Coil(L3)); // Atribuindo o valor da vari vel L3 ao pino
    Lamp3
digitalWrite(Lamp4, mb.Coil(L4)); // Atribuindo o valor da    vari vel L4 ao
    pino Lamp4
digitalWrite(Lamp5, mb.Coil(L5)); // Atribuindo o valor da vari vel L5 ao pino
    Lamp5
digitalWrite(Lamp6, mb.Coil(L6)); // Atribuindo o valor da vari vel L6 ao pino
    Lamp6
digitalWrite(pinCooler, mb.Coil(COOLER)); // Atribuindo o valor da vari vel
    COOLER ao pino que liga o Cooler

  byte temperature = 0; // configura  o da temperatura no DHT11
  byte humidity = 0; // configura  o da umidade no DHT11
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess)
    { // Leitura das vari veis temperatura e umidade
     return;
  }
  mb.Ireg(temp,temperature); // Atribui  o da temperatura ao registrador da
    temperatura
  mb.Ireg(umid,humidity); // Atribui  o da umidade ao registrador da umidade

  if(mb.Coil(ServoMOTOR)==HIGH){ // Condi  o de abertura do port o
    servo1.write(179);
  }else{
    servo1.write(0);
  }
```

O loop da programação é bastante simples, o digitalWrite indica que os componentes, Lamp1 por exemplo, devem receber os comandos através do coil, ou registradores que estão conectados ao ScadaBr. Finalizando a programação, temos um IF, que indica a condição para a abertura do portão.

6 ScadaBR

ScadaBR é um software livre, gratuito e de código-fonte aberto, para desenvolvimento de aplicações de Automação, Aquisição de Dados e Controle Supervisório. Para a realização deste projeto o ScadaBR4 foi instalado no sistema operacional `LINUX`, o qual contém o tutorial para o meio de instalação no `INDEX`, página 11.

Junto ao ScadaBR deve ser instalado um `SERVIDOR DE DADOS` chamado de `TOMCAT` que é o responsável por intermediar a comunicação do comando do usúario por meio do ScadaBR para o arduino.

Para usurfruir da aplicação do ScadaBR sobre o arduino, deve-se dar o *START* no TOMCAT. E, então, é gerado um endereço na internet com base no `LOCALHOST` em que o sistema foi instalado. Ao abrir este endereço em um navegador pode ser feita as configurações em `DataSource` e então, definir a interface de comunicação como `MODBUS Serial`, figura 3. E nas propriedades apresentadas, figura 4, pode ser definido os parâmetros e `tags`(Entradas/Saídas) desta comunicação assim como é descrito o tipo de paramêtro (analógico/digital), seu endereço de rede e com qual escravo (arduino) que está conectado.

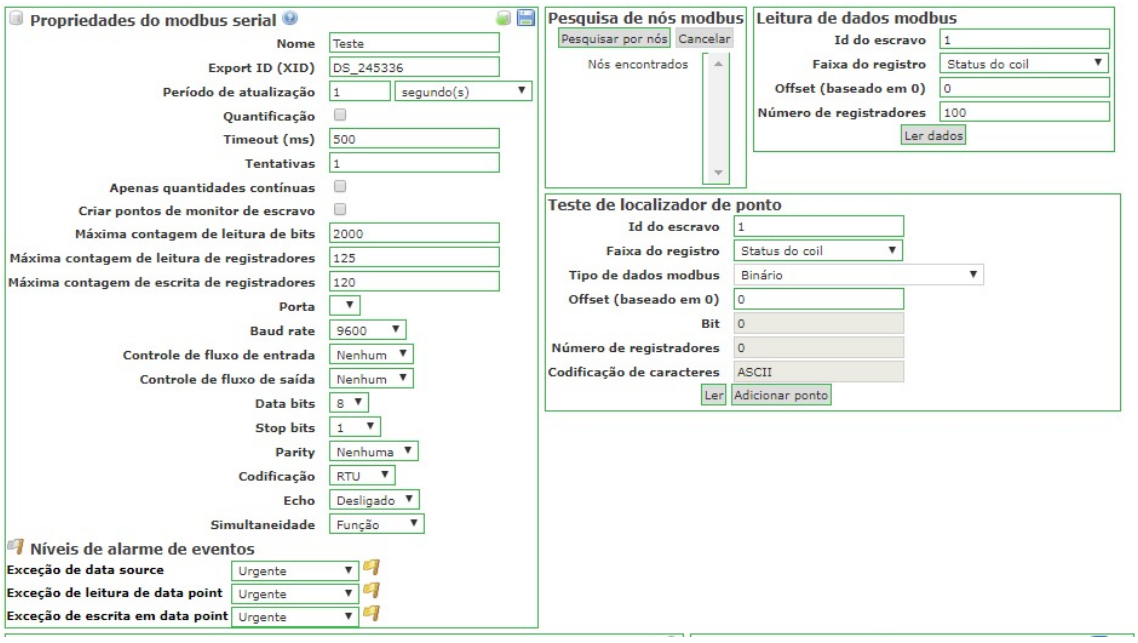

Figura 3: Configuração do Protocolo de Comunicação MODBUS Serial.

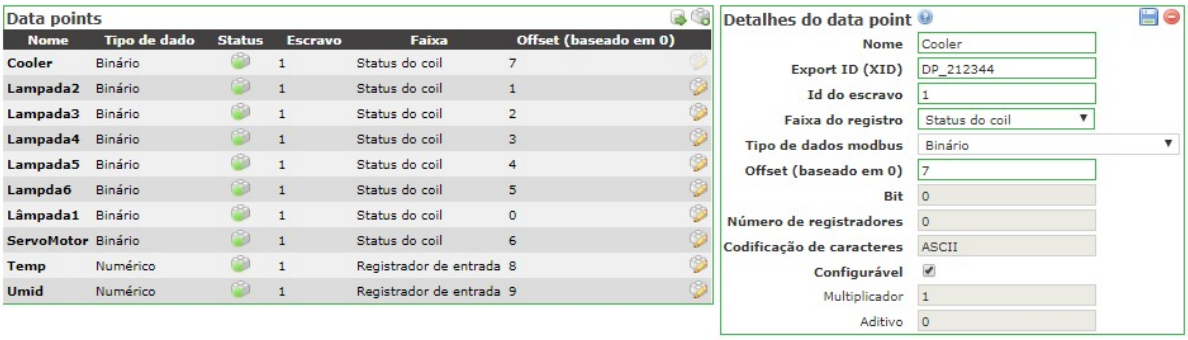

Figura 4: Tags e Propriedades dos Parâmetros.

Feita as configurações, em modo `RUN`, no `Watch List` aparece todos os parâmetros definidos anteriormente. Na figura 5, pode-se observar que as variáveis **temperatura** e **umidade** estão variando automaticamento já que são entradas analógicas, enquanto que algumas das variáveis digitais se encontram em 1 (ligadas) ou 0 (desligadas), sob controle do usuário.

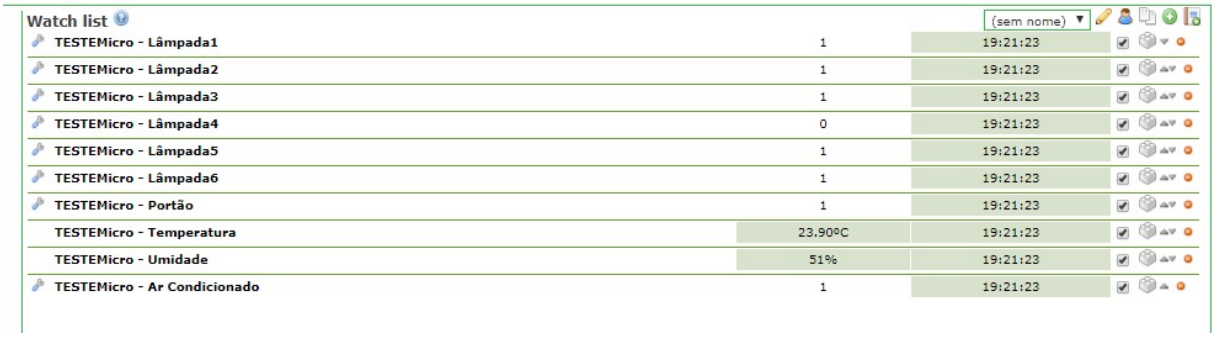

Figura 5: Parâmetros em funcionamento.

Com isso, na própria página do ScadaBR é feita a Representação Gráfica (HMI) para a interação do Usuário com o meio controlado. Neste caso, o HMI deste projeto é uma planta da residência automatizada, figura 6,com botões para ligar/desligar as lampadas, representações para as unidades de temperatura e umidade assim como botões para interação com o Ar Condicionado e Portão.

Figura 6: HMI (Human Machine Interface).

7 Hardware

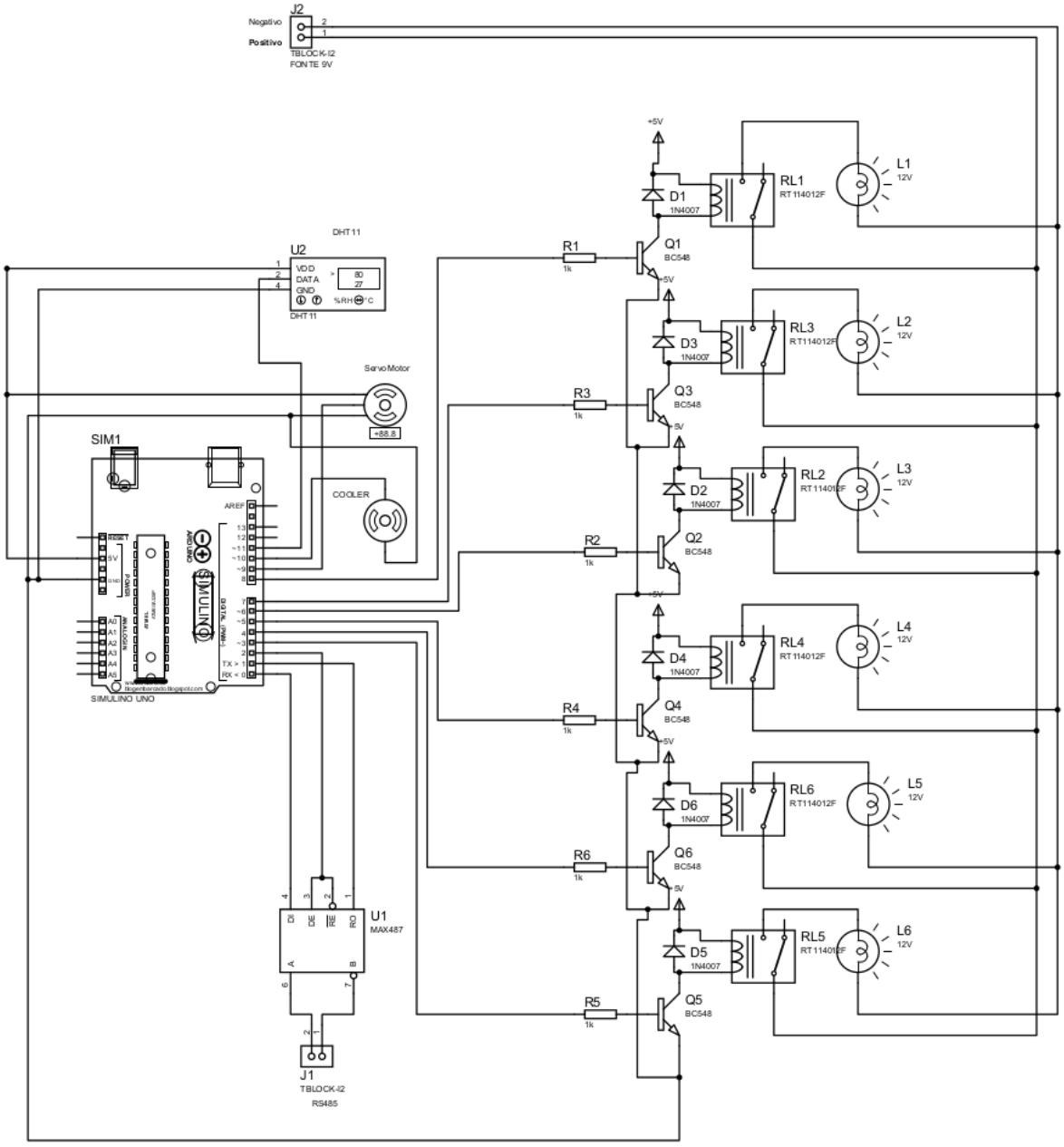

Figura 7: Hardware.

8 INDEX

8.1 Instalação de SOFTWARES

8.1.1 ScadaBR no RaspBerry (UBUNTU MATE)

```
tomcat=apache-tomcat-6.0.53.tar.gz

java6=jre-6u45-linux-x64.bin

echo -e " - Instalando $java6"

echo -e " - Criando pasta /opt/java"

sudo mkdir -p /opt/java/

echo -e " - Movendo $java6 para /opt/java"

sudo cp $java6 /opt/java

echo -e " - Entrando na pasta /opt/java"

cd /opt/java

echo -e " - Setando Permissoes"

sudo chmod 755 /opt/java/$java6

echo -e " - Instalando $java6"

sudo ./$java6

echo -e " - Criando atalho jre"

sudo ln -s jre1.6.0_45 jre

echo -e " - update-alternatives em andamento"

sudo update-alternatives --install "/usr/bin/java" "java" "/opt/java/jre/bin/
    java" 1

sudo update-alternatives --set java /opt/java/jre/bin/java

sudo update-alternatives --install "/usr/bin/javaws" "javaws" "/opt/java/jre/bin
    /javaws" 1

sudo update-alternatives --set javaws /opt/java/jre/bin/javaws

echo -e " - fim da Instala  o do Java!"

sleep 3

echo -e " - Instalando Tomcat6\n   ______\n\n"

echo -e " - Criando pasta /opt/tomcat6"

sudo mkdir -p /opt/tomcat6

echo -e " - Copiando instalador $tomcat para /opt/tomcat6"

sudo cp ~/$tomcat /opt/tomcat6/

cd /opt/tomcat6

echo -e " - Setando permissoes do $tomcat"

```

```
sudo chmod +x $tomcat

echo -e " - Descompactando $tomcat"

sudo tar xvf $tomcat

echo -e " - Copiando ScadaBR para o Destino"

sudo cp ~/ScadaBR.war /opt/tomcat6/apache-tomcat-6.0.53/webapps/

echo -e " - Iniciando tomcat6: /opt/tomcat6/apache-tomcat-6.0.53/bin/startup.sh"

sudo /opt/tomcat6/apache-tomcat-6.0.53/bin/startup.sh
```

Código 1: Funções para instalar o ScadaBR.

8.2 Código de Programação para o Arduino

```
#include <Modbus.h>          // Biblioteca Modbus para utiliza  o do protocolo
    junto com o meio fisico RS485
#include <ModbusSerial.h>
#include <Servo.h>          // Biblioteca para a utiliza  o do servo motor
#include <SimpleDHT.h>     // Biblioteca para a utiliza  o do sensor de
    temperatura e umidade DHT11

//--------------------------------------------- Defini  o dos Pinos
    ---------------------------------------
#define Lamp1 3
#define Lamp2 4
#define Lamp3 5
#define Lamp4 6
#define Lamp5 7
#define Lamp6 8
#define pinServo 9
#define pinCooler 10
#define pinDHT 11
//----------------------------- Atribui  o de Objetos dentro das bibliotecas
    utilizadas -------------------------
SimpleDHT11 dht11(pinDHT); // Objeto da biblioteca DHT11 e setagem do pino
    utilizado
ModbusSerial mb; // Objeto da biblioteca ModBus Serial
Servo servo1; // Objeto da biblioteca do Servo Motor
//------------------------- Endere amento dos Registradores utilizados no
    ScadaBR------------------------------
const int L1 = 0; // Endere o do L1 - L6 respons veis pelo controle dos LED s
     tipo Coil (Bin rio OUTPUT)
const int L2 = 1;
const int L3 = 2;
const int L4 = 3;
const int L5 = 4;
const int L6 = 5;
const int ServoMOTOR = 6; // Endere o do Servo Motor tipo Status do Coil (
    Bin rio OUTPUT)
const int COOLER = 7; // Endere o do Cooler tipo Status do Coil (Bin rio
    OUTPUT)
const int temp = 8; // Endere o da temperatura tipo Registrador de Entrada (
    Anal gico INPUT)
const int umid = 9;// Endere o da umidade tipo Registrador de Entrada (
    Anal gico INPUT)

void setup() {
  mb.config(&Serial,9600,SERIAL_8N1,2); // Configura  o inicial da biblioteca
    MODBUS Serial, configurando a velocidade de comunica  o e o pino de
    controle
  mb.setSlaveId(1); // Configura  o do ESCRAVO

```

```
//============================================================|
    Configura  o de Cada Registrador |
    ================================================

  mb.addCoil(L1);  //                          | Tipo de Registrador |
    Utiliza  o     | Acesso | C digo da Biblioteca |
  mb.addCoil(L2);  //
    |---------------------|----------------|--------|----------------------|
  mb.addCoil(L3);  //                          |        Coil          | Digital
    Output |  R/W   |   addCoil(), Coil()  |
  mb.addCoil(L4);  //                          |   Holding Register   | Analog
    Output  |  R/W   |   addHreg(), Hreg()  |
  mb.addCoil(L5);  //                          |    Input Status      | Digital
    Input  |  Read  |   addIsts(), Ists()  |
  mb.addCoil(L6);  //                          |    Input Register    | Analog
    Input   |  Read  |   addIreg(), Ireg()  |
  mb.addCoil(ServoMOTOR);
  mb.addCoil(COOLER);
  mb.addIreg(temp);
  mb.addIreg(umid);
//
    ================================================================================================

  servo1.attach(pinServo); // Configura  o do pino do servo motor
  pinMode(Lamp1,OUTPUT); // pinos de sa da e Entrada
  pinMode(Lamp2,OUTPUT);
  pinMode(Lamp3,OUTPUT);
  pinMode(Lamp4,OUTPUT);
  pinMode(Lamp5,OUTPUT);
  pinMode(Lamp6,OUTPUT);
  pinMode(pinServo,OUTPUT);
  pinMode(pinCooler,OUTPUT);
  pinMode(pinDHT,INPUT);

}

void loop() {
  mb.task(); // In cio da comunica  o serial
  digitalWrite(Lamp1, mb.Coil(L1)); // Atribuindo o valor da vari vel L1 ao
    pino Lamp1
  digitalWrite(Lamp2, mb.Coil(L2)); // Atribuindo o valor da vari vel L2 ao
    pino Lamp2
  digitalWrite(Lamp3, mb.Coil(L3)); // Atribuindo o valor da vari vel L3 ao
    pino Lamp3
  digitalWrite(Lamp4, mb.Coil(L4)); // Atribuindo o valor da vari vel L4 ao
    pino Lamp4
  digitalWrite(Lamp5, mb.Coil(L5)); // Atribuindo o valor da vari vel L5 ao
    pino Lamp5
  digitalWrite(Lamp6, mb.Coil(L6)); // Atribuindo o valor da vari vel L6 ao
    pino Lamp6
  digitalWrite(pinCooler, mb.Coil(COOLER)); // Atribuindo o valor da vari vel
    COOLER ao pino que liga o Cooler

  byte temperature = 0; // configura  o da temperatura no DHT11
  byte humidity = 0; // configura  o da umidade no DHT11
  int err = SimpleDHTErrSuccess;
  if ((err = dht11.read(&temperature, &humidity, NULL)) != SimpleDHTErrSuccess)
    { // Leitura das vari veis temperatura e umidade
     return;
  }
  mb.Ireg(temp,temperature); // Atribui  o da temperatura ao registrador da
    temperatura
  mb.Ireg(umid,humidity); // Atribui  o da umidade ao registrador da umidade

  if(mb.Coil(ServoMOTOR)==HIGH){ // Condi  o de abertura do port o
     servo1.write(179);
  }else{
     servo1.write(0);
  }


```

```
}
```

Código 2: Código Completo do Arduino.

9 Considerações finais