INSTITUTO FEDERAL DO ESPÍRITO SANTO

BÁRBARA FABRI
REGINALDO GUTTER
RICARDO BRENNER

PROJETO FINAL – PARTE 3

SERRA
2018

BÁRBARA FABRI
REGINALDO GUTTER
RICARDO BRENNER

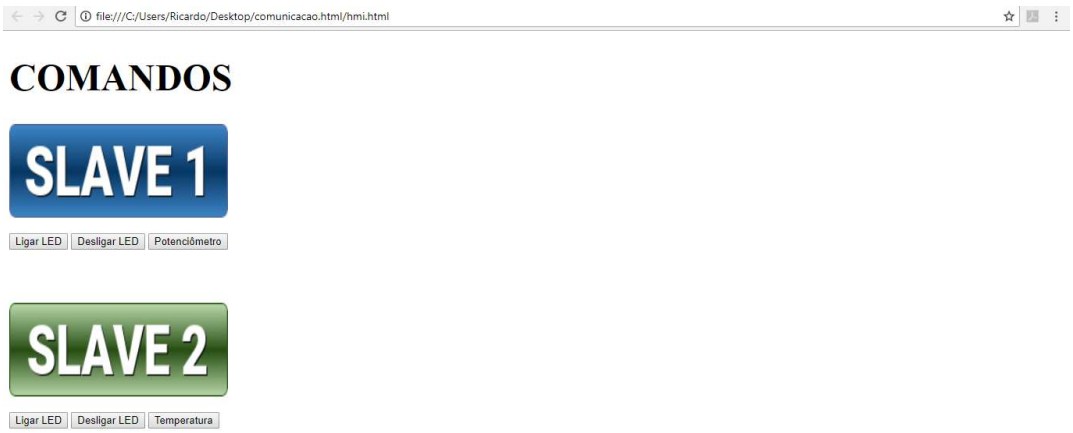
PARTE 3 – COMUNICAÇÃO MESTRE/ESCRAVO POR RS485 E INTERFACE WEB BÁSICA

Trabalho acadêmico apresentado à disciplina de Controle de Processos do curso de Engenharia de Controle e Automação, 9º período, do Instituto Federal do Espírito Santo - campus Serra, orientado pelo professor Rafael Emerick.

SERRA
2018

A terceira etapa do presente projeto consistiu na substituição da conexão entre o elemento mestre e o PC por uma rede Wifi, promovendo o controle e as leituras através de uma interface web HTTP básica.

A interface foi pensada para ser básica e de fácil entendimento para o usuário, por esta razão consiste da identificação dos escravos com os respectivos botões utilizados para promover o acionamento das funções:



Essa página promove a interação entre o usuário e o mestre através de requisições ao IP do elemento mestre. O trecho de código da página html abaixo ilustrado auxiliará a explicar este processo:

```
function slave2(sw)                          // Função para interagir com slave 2 a partir do clique no botão
{
    if (sw == 0)                             //Se sw retornar valor zero, acessa o respectivo IP para desligar led
    {
        window.open("http://192.168.137.113/desligaled2", "minhaJanela2", "top = 415, left = 300,height=100,width=400");
    }
    else if (sw == 1)                        //Se sw retornar valor um, acessa o respectivo IP para desligar led
    {
        window.open("http://192.168.137.113/ligaled2", "minhaJanela2", "top = 415, left = 300, height=100,width=400");
    }
    else if (sw == 2)                        //Se sw retornar valor dois, acessa o respectivo IP para desligar led
    {
        window.open("http://192.168.137.113/leentrada2", "minhaJanela2", "top = 415, left = 300, height=100,width=400");
    }
}


</script>
```

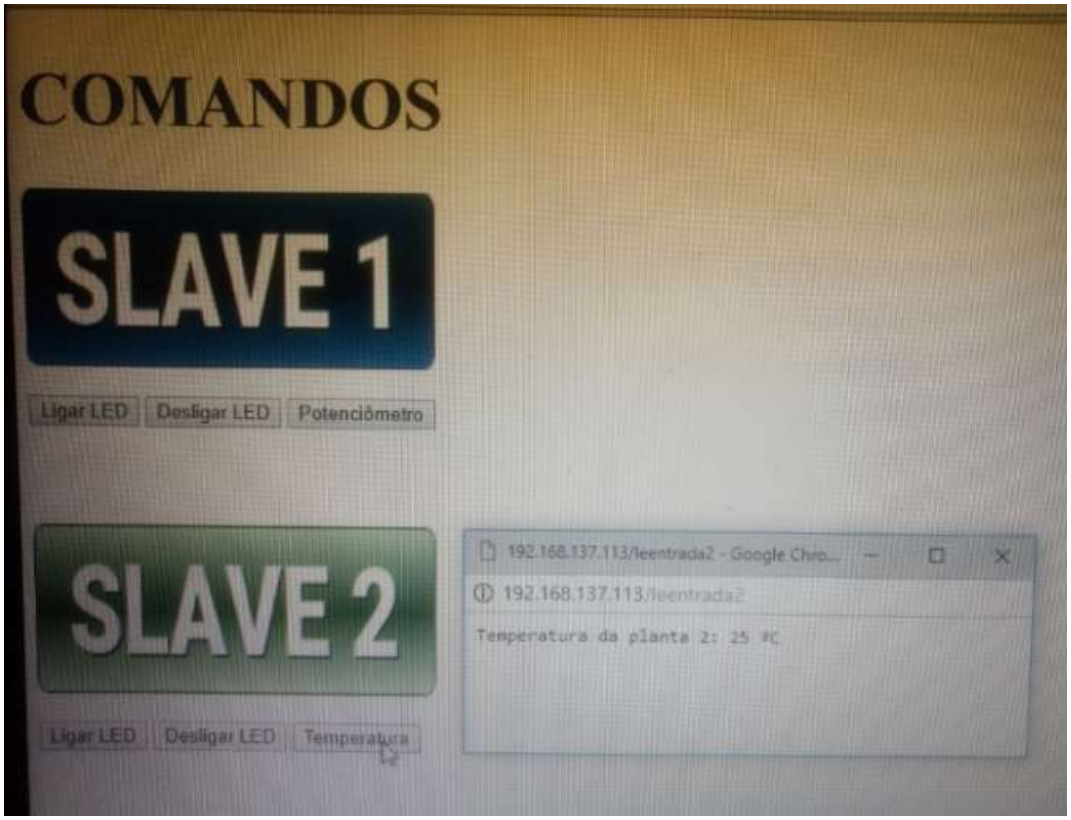
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Filter by type or text

No problems have been detected in the workspace so far.

Aqui pode-se observar a função que solicita ao mestre os serviços relacionados ao escravo 2. Por exemplo, ao clicar sobre o botão “Desligar LED” do slave 2 a

página irá abrir uma nova aba com o endereço “192.168.137.113/desligaled2”. A extensão após o endereço IP mudará para cada tipo de serviço.

Assim será feito uma solicitação ao mestre, que identificará o processo através da extensão do IP, fará a busca dos dados e então retornará para a página aberta o resultado da solicitação. A foto abaixo mostra o resultado de uma requisição de temperatura da planta 2:



O nodemcu consiste de um sistema embarcado com aparatos que permitem comunicação wireless. Ele pode funcionar como Acess Point, mas neste caso foi utilizado na função station:

ESP8266 operating in the **Station** mode

Para este experimento o notebook foi utilizado como acess point, roteando uma rede exclusiva para comunicação com o nodemcu.

A programação do nodemcu (elemento mestre) foi feita em linguagem LUA na IDE do Arduíno, e consiste de uma evolução do projeto programado na parte 2. As nuances do programa, bem como bibliotecas utilizadas e comentários podem ser verificadas no repositório git já informado neste relatório, na pasta de nome “Parte 3”. Vale a pena destacar o trecho onde o mestre identifica a função a ser executada a partir da solicitação de seu IP:

```
void loop(){

   cliente = servidor.available();                                    //Diz ao cliente que há um servidor disponivel.

   if (cliente == true)                                               //Se houver clientes conectados, ira enviar o HTML.
   {
      String req = cliente.readStringUntil('\r');                     //Faz a leitura do Cliente.
      Serial.println(req);                                            //Printa o pedido no Serial monitor.

      if (req.indexOf("/ligaled1") != -1)                             //solicitação para acionar o led do escravo 1
        {
             liga_led1();
        }
      else if (req.indexOf("/desligaled1") != -1)                      //solicitação para desligar o led o led
        {
             desliga_led1();
        }
      else if (req.indexOf("/leentrada1") != -1)                       //solicitação para acionar o led do escravo 1
        {
             entrada_1();
        }
```

No interior da função “loop” o mestre permanece informando que existe um servidor disponível, e caso haja cliente conectado ele buscará o serviço a ser executado através do comando req.indexOf(), que identificará a função e executará o respectivo processo.

Após buscar a informação desejada ou promover a atuação nos escravos, o mestre receberá um frame de resposta, verificará a validade deste frame e

enviará para a página solicitada uma mensagem que contém um retorno sobre o serviço solicitado:

```
checksum();                                                    //verifica se os dados são consistentes
String Json = "";                                              //variável para configurar o dados de resposta a ser enviado
if(checksum()==1 && ctrl==1)                                   //Se os dados são consistentes...
{
  Json  = "HTTP/1.1 200 OK\r\nContent-Type: application/json\r\n\r\n" + String("Potenciometro da planta 1: ") + String(z[4]) + String(" %") +  "\r\n\r\n";
  cliente.print(Json);
  delay(1);
  }
 else if(checksum()==1 && ctrl==2)
 {
  Json  = "HTTP/1.1 200 OK\r\nContent-Type: application/json\r\n\r\n" + String("Led do escravo 1 ligado!") +  "\r\n\r\n";
  cliente.print(Json);
  delay(1);
 }
  else if(checksum()==1 && ctrl==3)
 {
  Json  = "HTTP/1.1 200 OK\r\nContent-Type: application/json\r\n\r\n" + String("Led do escravo 1 desligado!") +  "\r\n\r\n";
  cliente.print(Json);
  delay(1);
 }
  else if(checksum()==1 && ctrl==4)
 {
  Json  = "HTTP/1.1 200 OK\r\nContent-Type: application/json\r\n\r\n" + String("Temperatura da planta 2: ") + String(z[4]) + String(" °C") +  "\r\n\r\n";
  cliente.print(Json);
  delay(1);
 }
```

A variável Json consiste de uma cadeia de string que contém um cabeçalho e por fim as informações a serem escritas na página html requisitante. Neste caso, para simplificação das configurações de retorno, foi declarado que a mensagem ocorreria no protocolo HTTP e que o tipo de informação teria um formato de Json. A informação retornada não consiste de um Json, e esta configuração foi utilizada apenas para simplificar os moldes da mensagem retornada pela função cliente.print().

Uma vez funcional foram feitas algumas análises práticas do processo. Deixando o nodemcu estático, o notebook(ponto de acesso da rede criada) foi sendo gradativamente afastado, e foram dados comandos para analisar a robustez do sistema. Neste caso a rede consistia exclusivamente da comunicação entre o notebook e o nodemcu, logo não havia concorrência pelo ponto de acesso.

Tentou-se analisar os delays com um cronometro, mas os tempos envolvidos eram muito pequenos. O processo mais lento em questão é o de medição de temperatura do escravo 2, devido a questões da biblioteca por ele utilizada. Percebeu-se um aumento no tempo de resposta para a página conforme se afastava o notebook, mas este delay ainda era pequeno. Com 10 metros de distância era perceptível um delay que talvez chegasse a algo próximo de 1 segundo.

Apesar de ser um protótipo o funcionamento do processo foi relativamente estável, e durante a fase de experimentos somente ocorreu falha na transmissão do comando em uma das solicitações. Para utilização em um sistema real teria de se executar muitas melhorias, principalmente no quesito de segurança. Qualquer pessoa com acesso a rede seria capaz de executar um comando ou acessar a leitura dos parâmetros dos escravos.

Além disso, para um processo real, deve-se garantir a estabilidade no sinal da conexão wireless, pois sua qualidade é de fundamental importância para as conexões e tempos envolvidos.

Este experimento permitiu boa consolidação e confrontamento com a teoria apresentada, e tornou simples a visualização do processo de comunicação entre os elementos através de uma rede sem fio.

Os arquivos da parte 3 podem ser obtidos em https://gitlab.com/RBrenner/Comunicacao_Dados.git