Sistemas Reativos – Miniprojeto 4

Alunos:
Lucas Santos
Carlos Leandro

Objetivos

- Oferecer uma interface WEB amigável, pois o acesso via navegador é simples e pouco custoso;
- Permitir controle remoto para locais de difícil acesso;
- Possibilitar o tratamento diversos controladores (independentes ou não) através de uma central única.

Acesso WEB

- Utilizamos a placa NODEMCU para atuar como um servidor WEB.
- De forma simples, permite hospedar uma ou mais páginas para que os usuários enviem comandos e obtenham informações sobre o sistema de interesse.
- Utilizamos parâmetros de query para enviar os comandos.
- Neste projeto apenas oferecemos comandos, mas informações das páginas podem ser atualizadas em tempo real através da geração dinâmica da página e uso de scripts no lado do cliente para atualização.

```
---- Funcao de criacao e configuracao da pagina do webserver
local criar_webserver = function()
    local listen_callback = function(conn)
        local receive_callback = function(client, request)
            local pagina = ""
            local get = {}
            local _, _, metodo, path, vars = string.find(request, "([A-Z]+) (.+)?(.+) HTTP")
            if (metodo == nil) then
                _, _, metodo, path = string.find(request, "([A-Z]+) (.+) HTTP")
            end
            if (vars ~= nil) then
                for k, v in string.gmatch(vars, "(%w+)=(%w+)&*") do
                    get[k] = v
                end
            end
            if (get.pin == "horario") then
                estado_led1 = toggle_led(estado_led1, led1)
                print("+")
            elseif (get.pin == "meio") then
                estado_led1 = toggle_led(estado_led1, led1)
                print("M")
            elseif (get.pin == "antihorario") then
                estado_led1 = toggle_led(estado_led1, led1)
                print("-")
            end
            client:send(configurar_pagina(), function()
                client:close()
            end)
            collectgarbage()
        end
        conn:on("receive", receive_callback)
    end
    servidor=net.createServer(net.TCP)
    servidor:listen(80, listen_callback)
end
```

```
    local setup = function()
        gpio.mode(led1, gpio.OUTPUT)
        gpio.mode(led2, gpio.OUTPUT)
        configurar_wifi()
        criar_webserver()
    end

    setup()
```

```
-- Funcoes responsaveis por configurar um webserver e executar comandos
---- Funcao de criacao da pagina html do webserver
local configurar_pagina = function()
    local pagina = "<h1>NODE Web Server</h1>"
    pagina = pagina.."<p style=\"margin-left: 50px;\">Mudar Posição do Servo</p>"
    pagina = pagina .. "<a href=\"?pin=horario\" style=\"margin-left: 10px; margin-right:10px;\"><button>Horário</button></a>"
    pagina = pagina .. "<a href=\"?pin=meio\" style=\"margin-left: 10px; margin-right:10px;\"><button>Voltar p/ Meio</button></a>"
    pagina = pagina .. "<a href=\"?pin=antihorario\" style=\"margin-left: 20px;\"><button>Anti-Horário</button></a>"
    return pagina
end
```

Comunicação Serial

- Utilizamos o Arduino UNO para controlar diretamente os atuadores. Portanto necessitamos uma forma de comunicar os comando recebidos via NODEMCU para o arduino.
- Utilizando o protocolo UART (padrão do serial para ambas as placas) pudemos fazer uma conexão.

Comunicação Serial – Limitações

- O nosso sistema não conta com alimentação externa, embora em campo essa seja a realidade.
- A entrada USB utiliza a mesma porta serial que o padrão das placas. Portanto, tivemos de escolher:
  - 1 – Utilizar a biblioteca SoftwareSerial que simula a comunicação serial UART em outras portas das placas.
  - 2 – Utilizar a Tx e Rx padrão das placas.

Comunicação Serial – Limitações

- No caso 1, temos a vantagem de poder (sem endereçamento) tratar a comunicação para diversas placas arduino (unicast), e com pouco custo para todas as placas (multicast).
- No caso 2, podemos tratar o multicast naturalmente (basta conectar as placas em paralelo) em troca de ter de adicionar um endereçamento para lidar com o unicast.
- No caso 2, não podemos utilizar o buffer de leitura. Ele pode ser corrompido através do preenchimento indevido pela entrada USB.
- Embora exista essa limitação , optamos por utilizar neste projeto o método 2 (utilizar a mesma porta que o USB facilita o debug). A comunicação no sistema flui apenas do NODE para o Uno e é simples, permitindo tratar explicitamente cada caso.

Arduino UNO - Controlador

- A placa controla um servomotor.
- Dependendo da entrada recebida ele alterna a posição do braço do servomotor.
- Como este não era o foco do sistema, utilizamos a biblioteca Servo presente no arduino que facilita o controle de servomecanismos, abstraindo o controle PWM.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);
  Serial.begin(115200);
  Serial.flush();
  servo.attach(8);
  servo.write(pos);
}
```

```
// the loop function runs over and over again forever
void loop() {
  char input = Serial.read();
  if (input != -1) {
    Serial.flush();
    Serial.println(input);
    if (input == '+') {
      if (pos > 90)
        pos = 80;
      else if (pos != 20)
        pos = pos-10;

      toggleLED();
      servo.write(pos);
    }
    else if (input == '-') {
      if (pos < 90)
        pos = 100;
      else if (pos != 170)
        pos = pos+10;

      toggleLED();
      servo.write(pos);
    }
    else if (input == 'M') {
      pos = 90;
      toggleLED();
      servo.write(pos);
    }
  }
  delay(100);
}
```

Apêndice – Esquemático