Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Departamento de Engenharia Elétrica
ELE0629 – Tópicos Especiais em Sistemas Embarcados – 2020.2

Projeto de aplicação com periféricos do ESP32

Pedro Henrique de Souza Fonsêca dos Santos

Introdução

Em um mundo cada vez mais interligado, a integração de equipamentos se tornou um dos pilares da nova indústria. O objetivo é de cada vez mais extrair dados e informações que auxilie nas melhorias dos sistemas. Nesse contexto, aparecem em foco dois conceitos, o de Sistemas Embarcados e o de *Internet of Things* (IoT).

Sistemas Embarcados são sistemas embutidos com um função específica, seja ela coletar informações, firmar uma comunicação ou auxiliar no processamento de dados. Esses sistemas necessitam de uma inteligência, tarefa essa, feita por um microcontrolador, que funcionará como o coração do sistema. Além disso, a integração com a rede, tornam mais interessantes esses sistemas. O termo IoT é exatamente para isso, sendo a interligação entre os sistemas embarcados via conexão sem fio.

Nesse projeto, será mostrada uma aplicação IoT que se utiliza de um microcontrolador, o ESP32, desenvolvido pela Espressif, para coletar dados de temperatura, por meio do sensor de temperatura LM35, e iluminância, por meio do resistor dependente de luz, o LDR. Esses dados serão enviados pela internet e poderão ser acessados pelo aplicativo Telegram.

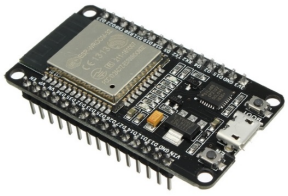

Figura 1: ESP32

Figura 2: Aplicativo Telegram

Desenvolvimento

Para o concepção desse projeto, temos em destaque duas importantes APIs (Interface de programação de aplicações) e uma biblioteca utilizadas no desenvolvimento, a API do conversor analógico-digital (ADC), a API para a comunicação *Wi-Fi* e uma biblioteca para se comunicar com o aplicativo Telegram. Essas APIs e biblioteca tornarão mais fácil o desenvolvimento da aplicação.

1 Conversor analógico-digital

A grande questão do conversor analógico-digital envolve principalmente a sua calibração e a transfomação das grandezas. Para a calibração, é utilizada a API da Espressif `driver/adc.h`, que possui uma estrutura de calibração. Com ela, passamos parâmetros como, qual canal será usado, qual a atenuação será necessária, a resolução do conversor e sua tensão de referência. Assim, ele cria uma estrutura que, ao coletar o sinal de tensão, a modifica para uma leitura correta. A estrutura para a calibração pode ser vista abaixo:

```
#include <driver/adc.h>
#include <esp_adc_cal.h>

esp_adc_cal_characteristics_t adc_cal; // Estrutura de calibracao

void setup()
{
  adc1_config_width(ADC_WIDTH_BIT_12);
  adc1_config_channel_atten(ADC1_CHANNEL_7, ADC_ATTEN_DB_0); //
     Configura o canal

  esp_adc_cal_value_t adc_type = esp_adc_cal_characterize(
     ADC_UNIT_1, ADC_ATTEN_DB_0, ADC_WIDTH_BIT_12, 1100, &adc_cal)
     ; // Inicializa a calibracao

 }
```

Com essa estrutura, foi possível criar as funções para extrair do LM35 e do LDR. Para a temperatura, é pego a tensão na saída do LM35. Essa tensão é proporcional ao a temperatura medida, tendo uma variação de 10mV para cada grau celsius variado. Dessa forma, a função de extrair a temperatura ficou da seguinte forma:

```
String getTemperature(){
  unsigned long int voltage = 0;
  String celsius;
  for(int i = 0; i < 100; i++){
    voltage += adc1_get_raw(ADC1_CHANNEL_7);
    delay(30);
  }
  voltage /= 100;
  voltage = esp_adc_cal_raw_to_voltage(voltage, &adc_cal);
  celsius = String(0.1*voltage);
```

```
  return celsius;
}
```

Para o LDR, não há essa proporção tão exata. O circuito do LDR é apenas um divisor de tensão, no qual a resistência do mesmo varia de acordo com a iluminação do local onde se encontra. Com isso, o primeiro passo para a obtenção, é exatamente descobrir a resistência. O código, no entanto, não fornece a resistência e sim a iluminância. Isso foi possível graças a um estudo visto no site *AllAboutCircuits*, que é mostrada uma função que relaciona a resistência com a quantidade de lux. Dessa forma, a função de extrair a iluminância ficou assim:

```
String getIlluminance(){
  unsigned long int voltage = 0;
  double RLDR = 0;
  String lux;
  for(int i = 0; i < 100; i++){
    voltage += adc1_get_raw(ADC1_CHANNEL_6);
    delay(30);
  }
  voltage /= 100;
  voltage = esp_adc_cal_raw_to_voltage(voltage, &adc_cal2);
  RLDR = (10000.0 * (3300 - voltage))/voltage;
  lux = String(12518931*pow(RLDR, -1.405));
  return lux;
}
```

2 *Wi-Fi*

Para a utilização do *Wi-Fi*, utiliza-se a API `WiFi.h`. Por meio dela, é escolhido o modo que ser quer utilizar, no caso o modo *Station*, pois este é utilizado para se conectar a um ponto de acesso, e aos parâmetros de rede sem fio a qual desejar. A estrutura para a conexão com a rede sem fio se encontra a seguir:

```
#include <WiFi.h>

char ssid[] = "Insira sua rede Wi-Fi";
char password[] = "Insira sua senha";

void setup() {
  Serial.begin(115200);

  // WIFI
  Serial.print("Connecting Wifi: ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { // Espera o status ser
     conectado
    Serial.print(".");
    delay(500);
```

```
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}
```

3 Telegram

A utilização do Telegram no projeto só é possível graças ao sistema de bots que o aplicativo tem. Com ele, é possível criar um bot que pode se comunicar com o microcontrolador, recebendo ou enviando dados. Para isso, é preciso ter a biblioteca `UniversalTelegramBot.h`. Criando o bot no Telegram, é recebido um token, que é utilizado para a criação do objeto bot no código. Por meio deste, é possível se comunicar com o Telegram. As funções utilizadas para se comunicar estão comentadas no código abaixo:

```
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

#define BOTtoken "Insira o token recebido no Telegram"

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client); // Cria o objeto

String id, text, msg;
// Strings que armazenam o id do usu rio, a mensagem recebida
// e a mensagem enviada
unsigned long tempo;

void setup()
{
}
void loop()
{
if (millis() - tempo > 2000)  { // Verifica a cada 2 secs

    tempo = millis();//Reseta o tempo

    int newmsg = bot.getUpdates(bot.last_message_received + 1);
       // Faz a verificacao de novas mensagens
    for(int i = 0; i < newmsg; i++){
      id = bot.messages[i].chat_id;
      // Quando recebe novo update, armazena o ID de quem enviou
      text = bot.messages[i].text;
      // Quando recebe novo update, armazena o texto da mensagem
      text.toUpperCase(); // Transforma o texto em maiusculo

      if(text.indexOf("TEMPERATURA") > -1){
```

```
32        // Procura a palavra na String
33          msg = String("Temperatura: " + getTemperature() + " C");
34          bot.sendMessage(id, msg, ""); // Envia a mensagem pro ID
35        }
36        else if(text.indexOf("ILUMINANCIA") > -1){
37          msg = String("Iluminancia: " + getIlluminance() + " lux")
              ;
38          bot.sendMessage(id, msg, "");
39        }
40        else if(text.indexOf("TUDO") > -1){
41          msg = String("Temperatura: " + getTemperature() + " C\
              nIluminancia: " + getIlluminance() + " lux");
42          bot.sendMessage(id, msg, "");
43        } else {
44          bot.sendMessage(id, "Comando invalido", "");
45        }
46      }
47    }
48  }
```

Resultados e discussão

O primeiro passo para o projeto é a criação do bot no Telegram. Para isso, basta entrar no Telegram e procurar o *BotFather*. Nele, digitamos o comando `/newbot` para criá-lo e escolhemos seu nome e username. O nome escolhido para o projeto foi "Projeto Embarcados" e o username foi "ProjetoEmbarcados_bot". Na figura 3 é possível ver o comando usado, os nomes escolhidos e o token recebido no final.

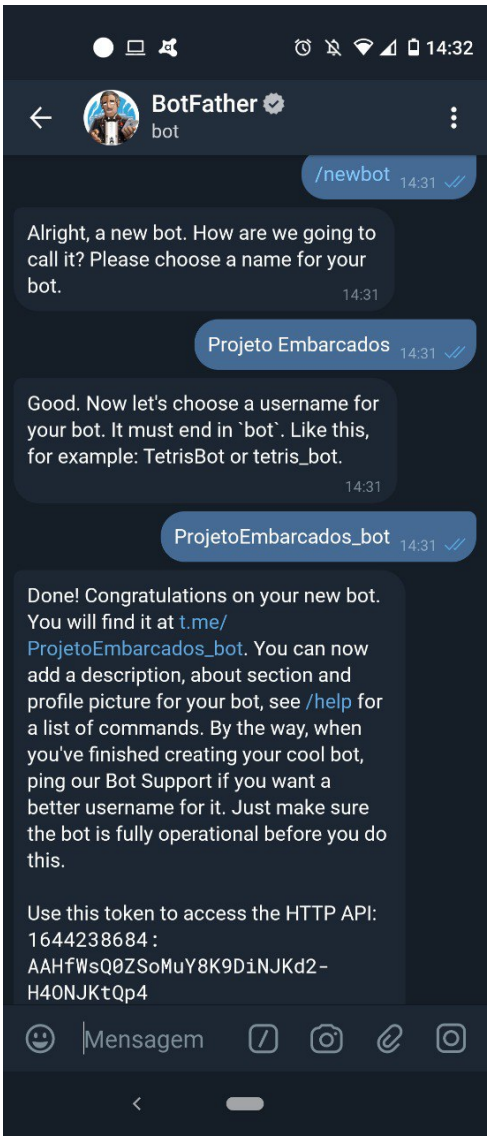

Figura 3: Comandos para a criação do bot

Na figura 3, também é possível ver o link para conversar com o bot. Ao clicar nele, irá abrir a conversa mostrada na figura 4, e para utilizá-lo, é só apertar em "Começar". Agora basta colocar o token no código e realizar o upload do código para a placa de desenvolvimento.

Figura 4: Bot criado esperando o comando para começar.

Com o código carregado e o microcontrolador conetado no *Wi-Fi*, é só começar a requisitar os valores dos sensores. Foram escolhidos três comandos para esse projeto, o de extrair temperatura, chamado apenas por "Temperatura", o de extrair iluminância, chamado por "Iluminancia", e o de extrair os dois, chamado por "Tudo". O comando independe de letras maiúscula ou minúscula, pois, como mostrado no desenvolvimento, há uma função para transformar todas as letras em maiúsculas. Para requisitar, só precisa escrever as palavras, enviar e esperar alguns segundos.

Figura 5: Requisições feitas e suas respectivas respostas.

Conclusão

Por fim, é possível ver que uma aplicação simples e funcional de IoT não é complicada de ser feita, além de trazer benefícios como a automação de diversos pontos da residência. A utilização do Telegram é uma adição grande às possibilidades possíveis para as aplicações, que vão desde à feita nesse projeto, até acionamentos de luzes via relé, controle de televisão ou ar-condicionado via módulo infravermelho, entre outros. E não só as possibilidades, como também a facilidade de utilização do Telegram, sendo muito fácil de criar e usar seus bots e comandos.