UNIVERSIDADE CATÓLICA DE PELOTAS

MESTRADO EM ENGENHARIA ELETRÔNICA E COMPUTAÇÃO

DOUGLAS ADALBERTO SCHEUNEMANN

DISCIPLINA DE INSTRUMENTAÇÃO UBÍQUA

TRABALHO 4 – MÓDULO ESP8266 – SENSORES

Pelotas, Setembro de 2015

1 Introdução

O módulo WiFi ESP8266 é um SoC com protocolo TCP/IP integrado e acesso a rede WiFi. O ESP8266 é capaz tanto de hospedar uma aplicação quanto disponibilizar todas as funções de redes WiFi para outro processador via interface UART utilizando protocolo AT. Na Figura 1 pode ser vista uma imagem do módulo ESP8266-01, o qual possui além do pinos TX e RX mais dois pinos GPIO.

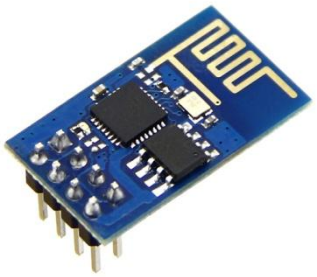

Figura 1: módulo ESP8266-01.

Outras versões do módulo podem ser encontradas com variações na quantidade de pinos de GPIO disponíveis e tipo de antena. O modelo ESP8266-12E conta com antena integrada, nove pinos de GPIO, um conversor ADC de 10 bits, interfaces UART e SPI.

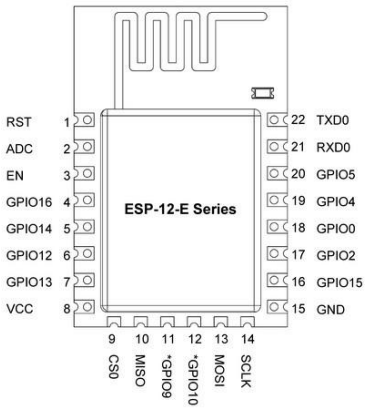

Figura 2: diagrama de pinos do módulo ESP8266-12E

2 Aplicações hospedadas

Os dois métodos mais referenciados para desenvolvimento de aplicações hospedas nos módulos ESP8266, são a utilização da IDE do Arduino ou a utilização de scripts em linguagem LUA. Para utilizar a IDE do Arduino é necessário instalar um plugin disponível neste link, no vídeo pode ser vista uma introdução ao uso do mesmo.

Uma vantagem de utilizar o plugin para Arduino é que a programação do ESP8266 pode ser feita com os mesmos comandos utilizados em outras placas Arduino. Logo, vários códigos desenvolvidos para interface de sensores com Arduino podem ser aplicados para os módulos ESP8266.

2.1 Gravação da aplicação

A gravação de aplicações hospedadas é feita através da interface UART. Utilizando o plugin para a IDE do Arduino a gravação pode ser feita através de uma porta serial do PC ou interface USB/Serial. Os módulos ESP8266 operam com tensões de 3,3 V, logo deve ser utilizado um conversor de nível lógico para adaptar interfaces UART que operam em 5,0 V para 3,3 V.

3 Aplicações com sensores

Em função dos módulos ESP8266 serem suportados pela IDE do Arduino, as aplicações para Arduino em que são utilizados sensores SPI, I2C e 1-Wire podem ser utilizadas no mesmo. Para os módulos que disponibilizam o pino ADC, as aplicações com leitura de dados analógicos também podem ser utilizadas.

Neste link pode ser vista a lista completa de bibliotecas do Arduino suportadas pelos módulos ESP8266.

3.1 Exemplo com sensor 1-wire DS1820

Código de exemplo disponível na IDE do Arduino e compilado para a placa “Generic ESP8266 Module”.

```
#include <OneWire.h>

// OneWire DS18S20, DS18B20, DS1822 Temperature Example
//
// http://www.pjrc.com/teensy/td_libs_OneWire.html
//
// The DallasTemperature library can do all this work for you!
// http://milesburton.com/Dallas_Temperature_Control_Library

OneWire  ds(2);  // on pin 2 (a 4.7K resistor is necessary)

void setup(void) {
  Serial.begin(9600);
}

void loop(void) {
  byte i;
```

```
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius, fahrenheit;

  if ( !ds.search(addr)) {
    Serial.println("No more addresses.");
    Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
    Serial.write(' ');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return;
  }
  Serial.println();

  // the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      Serial.println("  Chip = DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
    case 0x22:
      Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);        // start conversion, with parasite power on
at the end

  delay(1000);     // maybe 750ms is enough, maybe not
  // we might do a ds.depower() here, but the reset will take care of it.

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);         // Read Scratchpad

  Serial.print("  Data = ");
  Serial.print(present, HEX);
```

```
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print(OneWire::crc8(data, 8), HEX);
  Serial.println();

  // Convert the data to actual temperature
  // because the result is a 16 bit signed integer, it should
  // be stored to an "int16_t" type, which is always 16 bits
  // even when compiled on a 32 bit processor.
  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // "count remain" gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    // at lower res, the low bits are undefined, so let's zero them
    if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
    //// default is 12 bit resolution, 750 ms conversion time
  }
  celsius = (float)raw / 16.0;
  fahrenheit = celsius * 1.8 + 32.0;
  Serial.print("  Temperature = ");
  Serial.print(celsius);
  Serial.print(" Celsius, ");
  Serial.print(fahrenheit);
  Serial.println(" Fahrenheit");
}
```

3.2 Exemplo com sensor 1-wire DS1820 e funções da API WiFi

O código a seguir foi obtido neste link, no mesmo são utilizadas as funções da API WiFi do módulo ESP8266 em conjunto com a API para o sensor DS1820.

```
/*
  Created by Igor Jarc
 See http://iot-playground.com for details
 Please use community fourum on website do not contact author directly

 Code based on https://github.com/DennisSc/easyIoT-
ESPduino/blob/master/sketches/ds18b20.ino

 External libraries:
 - https://github.com/adamvr/arduino-base64
 - https://github.com/milesburton/Arduino-Temperature-Control-Library
```

```
 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 version 2 as published by the Free Software Foundation.
 */
#include <ESP8266WiFi.h>
#include <Base64.h>
#include <OneWire.h>
#include <DallasTemperature.h>

//AP definitions
#define AP_SSID "your ssid"
#define AP_PASSWORD "ap password"
// EasyIoT server definitions
#define EIOT_USERNAME    "admin"
#define EIOT_PASSWORD    "test"
#define EIOT_IP_ADDRESS  "192.168.1.5"
#define EIOT_PORT        80
#define EIOT_NODE        "N13S0"

#define REPORT_INTERVAL 60 // in sec


#define ONE_WIRE_BUS 2  // DS18B20 pin
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);


#define USER_PWD_LEN 40
char unameenc[USER_PWD_LEN];
float oldTemp;


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

  wifiConnect();

  char uname[USER_PWD_LEN];
  String str = String(EIOT_USERNAME)+":"+String(EIOT_PASSWORD);
  str.toCharArray(uname, USER_PWD_LEN);
  memset(unameenc,0,sizeof(unameenc));
  base64_encode(unameenc, uname, strlen(uname));
  oldTemp = -1;
}

void loop() {
  float temp;

  do {
    DS18B20.requestTemperatures();
    temp = DS18B20.getTempCByIndex(0);
    Serial.print("Temperature: ");
    Serial.println(temp);
  } while (temp == 85.0 || temp == (-127.0));

  if (temp != oldTemp)
  {
    sendTeperature(temp);
```

```
    oldTemp = temp;
  }

  int cnt = REPORT_INTERVAL;

  while(cnt--)
    delay(1000);
}

void wifiConnect()
{
    Serial.print("Connecting to AP");
    WiFi.begin(AP_SSID, AP_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
}

void sendTeperature(float temp)
{
   WiFiClient client;

   while(!client.connect(EIOT_IP_ADDRESS, EIOT_PORT)) {
    Serial.println("connection failed");
    wifiConnect();
  }

  String url = "";
  url += "/Api/EasyIoT/Control/Module/Virtual/"+ String(EIOT_NODE) +
"/ControlLevel/"+String(temp); // generate EasIoT server node URL

  Serial.print("POST data to URL: ");
  Serial.println(url);

  client.print(String("POST ") + url + " HTTP/1.1\r\n" +
               "Host: " + String(EIOT_IP_ADDRESS) + "\r\n" +
               "Connection: close\r\n" +
               "Authorization: Basic " + unameenc + " \r\n" +
               "Content-Length: 0\r\n" +
               "\r\n");

  delay(100);
    while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("Connection closed");
}
```

3.3 Utilização do ADC com script LUA

O exemplo mostrado a seguir foi obtido neste link, no mesmo é apresentado um exemplo de utilização do módulo ESP8266 para interface com dispositivos analógicos. A programação foi através de um script em linguagem LUA.

```
-- multisensor.lua
-- ---------------

--configure pins
gpio.mode(0,gpio.OUTPUT)     --  GPIO 16
gpio.mode(5,gpio.OUTPUT)     --  GPIO 14
gpio.mode(6,gpio.OUTPUT)     --  GPIO 12
--set all pins to LOW - no voltage
gpio.write(0,gpio.LOW)
gpio.write(5,gpio.LOW)
gpio.write(6,gpio.LOW)

--  print the ADC value with all pins low
print("\nADC: Both GPIOs LOW start: "..adc.read(0))
--  Send voltage to GPIO 16 completing the circuit for this sensor.
gpio.write(0,gpio.HIGH)
-- short delay before reading, 100ms
tmr.delay(100000)
-- Read the ADC pin or assign to varable var = adc.read(0)
print("\nADC: GPIO 16 HIGH LDR:  "..adc.read(0))
-- Return GPIO 16 to LOW, no voltage.
gpio.write(0,gpio.LOW)
-- short delay 100ms
tmr.delay(100000)
-- repeat for the other two sensors
gpio.write(5,gpio.HIGH)
tmr.delay(100000)
print("ADC: GPIO 14 HIGH Temp:  "..adc.read(0))
gpio.write(5,gpio.LOW)
tmr.delay(100000)
gpio.write(6,gpio.HIGH)
print("ADC: GPIO 12 HIGH Slider:  "..adc.read(0))
gpio.write(6,gpio.LOW)
--  again read ADC, this reading should be very close to the 'start
reading'
print("\nADC: Both GPIOs LOW end: "..adc.read(0))
```

4 Conclusão

Conforme exposto neste trabalho, os módulos ESP8266 são uma boa opção para aplicações de sensoriamento que necessitam de interface wireless, visto que, podem ser facilmente programados com aplicações hospedadas utilizando a IDE do Arduino. Além disso, a base de aplicações para sensores já desenvolvidas para o Arduino pode ser aproveitada.