Circuit Documentation

Summary

This circuit is designed to interface a GPS module (NEO 6M) and a WiFi module (ESP8266-01) with an Arduino UNO microcontroller. The circuit includes pushbuttons, LEDs, resistors, and an NPN transistor to facilitate user interaction and signal control. The Arduino UNO reads GPS data and communicates with the ESP8266-01 to send data over WiFi. LEDs are used to indicate the status of the WiFi connection and data transmission.

Component List

Microcontroller

- **Arduino UNO**: A microcontroller board based on the ATmega328P. It has digital input/output pins, analog inputs, a USB connection for programming, and power management features.

Communication Modules

- **WiFi module ESP8266-01**: A WiFi module that allows microcontrollers to connect to a wireless network and make simple TCP/IP connections using Hayes-style commands.
- **GPS NEO 6M**: A GPS module that provides location data such as latitude, longitude, and velocity.

Passive Components

- **Resistors (1k Ohms)**: Four resistors with a resistance of 1000 Ohms each, used for current limiting and voltage division.
- **Pushbutton**: A simple switch that connects two contacts when pressed.

Active Components

- **NPN Transistor (CBE)**: A bipolar junction transistor used for switching or amplification.
- **LED: Two Pin (green)**: A green light-emitting diode used as an indicator.
- **LED: Two Pin (red)**: A red light-emitting diode used as an indicator.

Wiring Details

Arduino UNO

- **GND**: Connected to the ground pins of the GPS NEO 6M, NPN Transistor, green LED, red LED, and ESP8266-01.
- **3.3V**: Powers the GPS NEO 6M and ESP8266-01 through a 1k Ohm resistor.

- **D9**: Connected to the RX pin of the GPS NEO 6M.
- **D8**: Connected to the TX pin of the GPS NEO 6M.
- **D3**: Connected to a 1k Ohm resistor.
- **D2**: Connected to the TX pin of the ESP8266-01.
- **D12**: Connected to a 1k Ohm resistor leading to the anode of the red LED.
- **D13**: Connected to a 1k Ohm resistor leading to the anode of the green LED.

WiFi module ESP8266-01

- **RX**: Connected to the emitter of the NPN Transistor.
- **GPIO0**: Connected to the pushbutton.
- **GND**: Shared ground with the Arduino UNO and pushbutton.
- **CH-PD Chip power down**: Connected to 3.3V through a 1k Ohm resistor.
- **+3V3**: Powers the module.

GPS NEO 6M

- **VCC**: Powered by the 3.3V output from the Arduino UNO.
- **RX**: Receives data from the D9 pin of the Arduino UNO.
- **TX**: Sends data to the D8 pin of the Arduino UNO.
- **GND**: Shared ground with the Arduino UNO.

Pushbutton

- **Pin 3 (out)** and **Pin 4 (out)**: Connected to the ground.
- **Pin 1 (in)** and **Pin 2 (in)**: Connected to the GPIO0 of the ESP8266-01.

NPN Transistor (CBE)

- **Collector**: Connected to the ground.
- **Base**: Connected to a 1k Ohm resistor which is connected to the D3 pin of the Arduino UNO.
- **Emitter**: Connected to the RX pin of the ESP8266-01.

LEDs

- **Green LED**: Anode connected to a 1k Ohm resistor which is connected to the D13 pin of the Arduino UNO. Cathode connected to the ground.
- **Red LED**: Anode connected to a 1k Ohm resistor which is connected to the D12 pin of the Arduino UNO. Cathode connected to the ground.

Documented Code

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>
```

```
// Configurações do GPS
SoftwareSerial SerialGPS(8, 9);
TinyGPS GPS;
float lat, lon, vel;
unsigned long data, hora;
unsigned short sat;

// Configurações do ESP01
SoftwareSerial ESP01(2, 3); // RX, TX
#define LED_SUCCESS 13
#define LED_ERROR 12

void sendData(String command, const int timeout, bool debug) {
  Serial.println("Enviando comando: " + command);
  ESP01.print(command);

  long int time = millis();
  bool responseReceived = false;

  while ((time + timeout) > millis()) {
    while (ESP01.available()) {
      String response = ESP01.readString();
      responseReceived = true;

      if (debug) {
        Serial.println("Resposta: " + response);
      }

      if (response.indexOf("ERROR") != -1) {
        Serial.println("Erro detectado na resposta do ESP-01: " +
response);
        digitalWrite(LED_ERROR, HIGH);
        digitalWrite(LED_SUCCESS, LOW);
        return;
      }
    }
  }
```

```
  // Handle timeout case
  if (!responseReceived) {
    Serial.println("Timeout: No response from ESP-01");
    digitalWrite(LED_ERROR, HIGH);
    digitalWrite(LED_SUCCESS, LOW);
  } else {
    digitalWrite(LED_SUCCESS, HIGH);
    digitalWrite(LED_ERROR, LOW);
  }
}

void setup() {
  // Inicializa comunicação serial
  Serial.begin(9600);
  SerialGPS.begin(9600);

  // Configura os LEDs como saída
  pinMode(LED_SUCCESS, OUTPUT);
  pinMode(LED_ERROR, OUTPUT);
  digitalWrite(LED_SUCCESS, LOW);
  digitalWrite(LED_ERROR, LOW);

  // Informa o início da configuração
  Serial.println("Iniciando teste de comunicação ESP01...");

  // Testa diferentes velocidades de comunicação
  int baudRates[] = {9600, 115200, 57600, 38400};
  bool connected = false;

  for(int i = 0; i < 4; i++) {
    Serial.print("Tentando baud rate: ");
    Serial.println(baudRates[i]);

    ESP01.begin(baudRates[i]);
    delay(1000);
```

```
    // Tenta comando AT 3 vezes
    for(int attempt = 0; attempt < 3; attempt++) {
      Serial.print("Tentativa ");
      Serial.print(attempt + 1);
      Serial.print(" com ");
      Serial.print(baudRates[i]);
      Serial.println(" baud");

      ESP01.println("AT");
      delay(1000);

      if(ESP01.available()) {
        while(ESP01.available()) {
          char c = ESP01.read();
          Serial.write(c);
        }
        connected = true;
        Serial.print("Sucesso! Baud rate correto: ");
        Serial.println(baudRates[i]);
        break;
      }
    }

    if(connected) break;
  }

  if(!connected) {
    Serial.println("ERRO: Não foi possível estabelecer comunicação com
ESP01");
    Serial.println("Por favor, verifique:");
    Serial.println("1. Conexões dos pinos TX/RX (TX->2, RX->3)");
    Serial.println("2. Alimentação 3.3V e GND");
    Serial.println("3. Pino CH_PD conectado ao 3.3V");
    digitalWrite(LED_ERROR, HIGH);
    while(1) { delay(1000); } // Para a execução
  }
```

```
  Serial.println("ESP01 respondeu! Continuando configuração...");
  digitalWrite(LED_SUCCESS, HIGH);
  delay(1000);

  // Continua com a configuração normal apenas se encontrou a comunicação
  Serial.println("Reiniciando ESP01...");
  sendData("AT+RST\r\n", 2000, true);
  delay(2000);

  // Configura o modo Wi-Fi
  Serial.println("Configurando modo WiFi...");
  sendData("AT+CWMODE=1\r\n", 2000, true);
  delay(1000);

  // Conecta ao Wi-Fi
  Serial.println("Conectando ao WiFi...");
  sendData("AT+CWJAP=\"DFRANCY DRYWALL
CENTRO\",\"dfrancy2023drywall\"\r\n", 10000, true);
  delay(5000);  // Aguarda mais tempo para conexão WiFi

  // Mostra o endereço IP
  Serial.println("Obtendo IP...");
  sendData("AT+CIFSR\r\n", 2000, true);
  delay(1000);

  // Configura múltiplas conexões
  Serial.println("Configurando conexões múltiplas...");
  sendData("AT+CIPMUX=1\r\n", 2000, true);
  delay(1000);

  // Inicia o servidor na porta 80
  Serial.println("Iniciando servidor...");
  sendData("AT+CIPSERVER=1,80\r\n", 2000, true);

  Serial.println("Configuração concluída!");
}
```

```
void loop() {
  // Verifica se há dados do GPS disponíveis
  while (SerialGPS.available()) {
    if (GPS.encode(SerialGPS.read())) {
      // Obtém hora e data
      GPS.get_datetime(&data, &hora);

      // Obtém latitude e longitude
      GPS.f_get_position(&lat, &lon);

      // Obtém velocidade
      vel = GPS.f_speed_kmph();

      // Obtém número de satélites
      sat = GPS.satellites();

      // Formata os dados em JSON
      String json = "{\"latitude\": " + String(lat, 6) + ", ";
      json += "\"longitude\": " + String(lon, 6) + ", ";
      json += "\"velocidade\": " + String(vel) + ", ";
      json += "\"satelites\": " + String(sat) + ", ";
      json += "\"hora\": \"" + formatTime(hora) + "\", ";
      json += "\"data\": \"" + formatDate(data) + "\"}";

      // Envia os dados ao cliente
      sendData("AT+CIPSEND=0," + String(json.length()) + "\r\n", 1000,
true);
      sendData(json + "\r\n", 1000, true);

      // Debug no monitor serial
      Serial.println("Dados enviados: " + json);
    }
  }
}

String formatTime(unsigned long hora) {
  char buffer[9];
```

```
  sprintf(buffer, "%02lu:%02lu:%02lu", (hora / 1000000) - 3, (hora %
1000000) / 10000, (hora % 10000) / 100);
  return String(buffer);
}

String formatDate(unsigned long data) {
  char buffer[11];
  sprintf(buffer, "%02lu/%02lu/%02lu", data / 10000, (data % 10000) / 100,
data % 100);
  return String(buffer);
}
```