Universidade Federal do Maranhão
Departamento de Informática
Laboratório de Sistemas Distribuídos Inteligentes (LSDi)

Relatório: Monitoramento do Ambiente com Arduino

Discente: Amanda Almeida Cardoso
Orientador: Prof. Dr. Francisco Silva e Silva
Junho, 2024

1 Introdução

O projeto foi desenvolvido com objetivo de monitorar o ambiente de uma sala enviando os dados de temperatura, umidade e presença de gás via BLE para uma aplicação Android, e através dessa aplicação, enviar comandos para ligar/desligar e mudar a temperatura do ar condicionado. O ar condicionado em questão é da marca Gree e o protocolo do seu controle remoto é yay1f1.

2 Hardware

Para o projeto, foram utilizados os seguintes componentes:

- Arduino Uno
- Sensor DHT11
- Sensor de Gás MQ-135
- Módulo HM-10
- Emissor IR
- Resistores

3 Software

Foi utilizada a biblioteca Software Serial que permite a comunicação serial com pinos digitais de uma placa Arduino. Inicialmente, foi feito uso do aplicativo BLE Scanner para realizar conexão com o módulo HM-10 e acessar seus serviços. Posteriormente, desenvolveu-se uma aplicação android própria para monitorar o ambiente e alterar a temperatura do ar condicionado.

Para a parte de alteração da temperatura, fez-se uso da biblioteca IRremote que permite enviar e receber sinais de infravermelho. Foram testadas diferentes formas de envio que serão discutidas brevemente. A solução escolhida foi salvar os arrays IR na memória Flash do Arduino, alternativa mais eficaz para lidar com a pouca memória RAM do dispositivo.

4 Funcionamento

Consiste em obter os dados dos sensores e enviar as informações para uma aplicação cliente que irá estar conectada com o módulo e poderá acessar as informações através do serviço disponibilizado. Essa aplicação também é capaz de enviar informações ao módulo. No caso desse projeto, a informação enviada é referente a temperatura que se deseja. O usuário digita um valor numérico que pode ser 0, para desligar a temperatura, ou um valor entre 16 e 30. O Arduino, ao receber esse número, irá enviar o array de infravermelho equivalente para que ocorra a mudança de temperatura do ar condicionado. E caso o ar condicionado esteja desligado, ele irá ligar automaticamente na temperatura desejada.

O arduino espera receber o valor digitado seguido de um ";" para marcar o final. O aplicativo elaborado pelos autores faz essa alteração automaticamente, então o usuário pode apenas digitar o valor, sem a adição do ponto e vírgula. Porém caso seja utilizado algum outro aplicativo que realize a conexão BLE, é necessário que o usuário digite a temperatura no formato "NUMERO;" para que o dispositivo consiga identificar o final do comando.

5 Conexões físicas

As seguintes conexões físicas precisam ser feitas:

HM-10	Arduino
TX	Digital pin 10
RX	Digital pin 11
VCC	5V
GND	GND

Table 1: Conexões HM-10

DHT-11	Arduino
Data	Digital pin 4
VCC	5V
GND	GND

Table 2: Conexões DHT-11

MQ-135	Arduino
AO	Analog pin 0
VCC	5V
GND	GND

Table 3: Conexões MQ-135

Emissor IR	Arduino
Data	Digital pin 3
VCC	5V
GND	GND

Table 4: Conexões Emissor IR

O HM-10 é um dispositivo de 3.3V. O breakout board converte o VCC +5V para 3.3v para alimentar o HM-10, mas o pino RX ainda é 3.3v. Isso significa que é necessário criar uma divisão de voltagem utilizando resistores para que a energia que chegue no pin RX seja de 3.3V. Para fazer isso, utilizamos três resistores de 1K ohm.

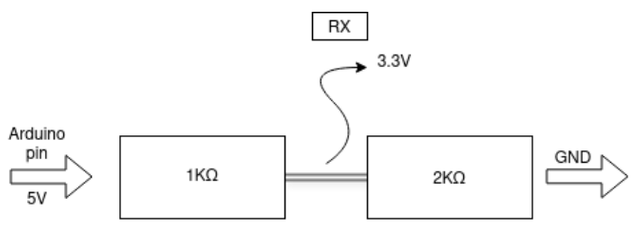

Figure 1: Esquema de divisão de voltagem

Também é importante notar que a comunicação utilizada é a UART, dessa forma, o RX (receptor) do dispositivo é conectado ao TX (transmissor) do outro dispositivo e vice versa. Isso quer dizer que os pinos RX e TX do HM10 devem estar conectados respectivamente aos pinos que farão o papel de TX, RX no Arduino (isso será configurado no código através da biblioteca Software Serial).

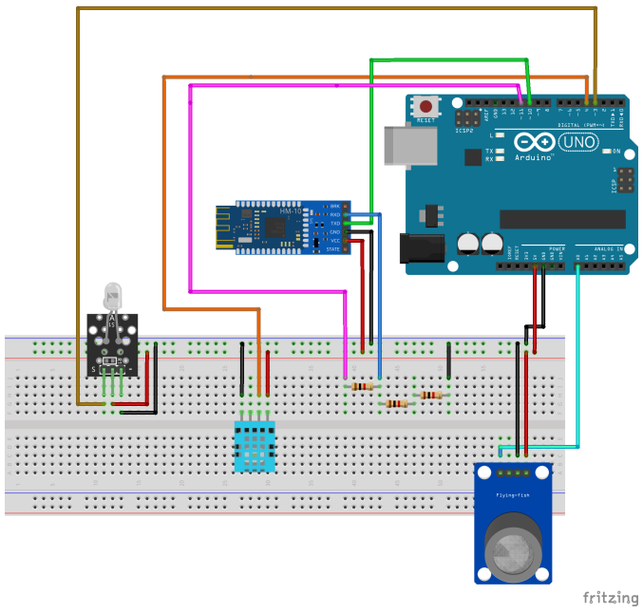

Figure 2: Conexões físicas

6 Configurando o módulo

Para configurar o módulo, é necessário um código para se comunicar via comandos AT com o HM-10. Para isso, utilizamos as funções read() e write() da biblioteca Software Serial e os comandos abaixo:

- Checar se o módulo está disponível para receber os comandos AT:
  >AT?
  Resposta esperada: OK

- Configurar o serviço de Sensoriamento Ambiental:
  >AT+UUID0x181A
  Resposta esperada: OK+Set: UUID0x181A

- Configurar a primeira característica Temperatura:
  >AT+CHAR0x2A6E
  Resposta esperada: OK+Set: CHAR0x2A6E

- Configurar a segunda característica (WRITE ONLY):
  >AT+FFE2?
  Resposta esperada: 0 ou 1. Se for 0, a segunda característica não está ativa. Se for 1, está ativa.
  Para ativá-la, enviamos o comando:
  >AT+FFE21
  Resposta esperada: OK+Set:1
  A segunda característica é automaticamente configurada com o UUID da primeira +1.

- Salvando as mudanças
  >AT+RESET
  Resposta esperada: OK+RESET

O código utilizado para configurar o módulo foi obtido de Martyn Currey (1).

Uma vez que o módulo esteja configurado, é possível fazer a descoberta de seus serviços e características utilizando um aplicativo que se conecte ao HM10. A troca de informações entre o módulo e o aplicativo é proporcionada pelas funções print (envia os dados dos sensores para o aplicativo através da conexão BLE) e read (recebe uma informação que é digitada no aplicativo: a temperatura desejada).

7 Sinais IR

Para conseguir alterar as temperaturas do ar condiconado, o primeiro passo é utilizar um código que consiga gravar os sinais de infravermelho gerados pelo controle remoto do aparelho. Os sinais IR de qualquer ar condiconado são maiores que outros, por isso, é preciso um código que esteja apto a gravar longos sinais. Nesse projeto, utilizamos o código disponibilizado pelo Analys IR: *Arduino Record Long Air Conditioner Infrared Signals*.

Através do padrão dos arrays, se descobre o protocolo utilizado pelo controle. Nesse caso, o protocolo era desconhecido e uma das alternativas era fazer uma engenharia reversa para decodificar o protocolo. Essa decodificação é essencial para salvar espaço na memória RAM do Arduino.

Os arrays desse modelo de ar condicionado são formados por 419 números. Iniciam-se com o cabeçalho (9064, 4456). Em seguida, temos os sinais de marcas e espaços. Nesse protocolo, a marca é 700, 1650. E o espaço é 700, 700. No array, o cabeçalho aparece 3 vezes e ainda existem pausas de 20.000 e 40.000. Esses padrões são mais perceptíveis ao transformarmos os arrays para binário da seguinte forma:

```
1 = 700, 1650
0 = 700, 700
```

Essa transformação está incluindo apenas os valores de marca e espaço, excluindo os cabeçalhos e outras pausas. Dessa forma, o padrão visto nos arrays é:

```
Hh - header (9064,4456)
S = short (20000)
L = long (40000)
p = pause (700)

    Hh (35bits) pS (32bits) pLHh (35bits) pS (32bits) pLHh (35bits) pS (32bits)
```

Depois que os arrays foram transformados para binário, separamos os arrays binários em bytes. Em seguida, observamos quais bytes mudavam entre uma temperatura e outra. Fizemos uma análise para ver qual o padrão que cada um seguia. Para isso, espelhamos os bytes e fizemos a análise de cada um. Por exemplo, o byte 02:

```
BYTE 2 (temperatura 16 a 30):
normal      espelhado
00000000    00000000   0
10000000    00000001   1
01000000    00000010   2
11000000    00000011   3
00100000    00000100   4
10100000    00000101   5
01100000    00000110   6
11100000    00000111   7
00010000    00001000   8
10010000    00001001   9
01010000    00001010   10
11010000    00001011   11
00110000    00001100   12
10110000    00001011   13
01110000    00001110   14
```

Constatamos que o byte 2 é sempre somado +1 para cada temperatura. Dessa mesma forma foi feito para os outros bytes que mudavam (byte 1, 2, 3, 8, 9, 10, 12, 17, 21 e 25). Ver aqui.

O ponto de partida para ver essas mudanças era comparar os bytes dos arrays binários de cada temperatura com os bytes do array utilizado para desligar o ar condicionado, uma vez que utilizamos o array binário de desligar para poder montar os arrays das temperaturas.

Com esses padrões, foi montado um código que transorforma o array binário de volta no array com o formato original de marcas e espaços.

Entretanto, essa solução se mostrou inviável, pois persistia o problema de falta de espaço na memória RAM. A solução encontrada foi guardar os arrays originais na memória Flash do Arduino. Dessa forma, não foi mais necessário utilizar o código que gerava o array binário ou o que retransformava para o array original.

Para fazer o envio desses arrays para o ar condicionado, utilizamos a função sendRawFlash obtida no AnalysIR (2).

Os códigos e arquivos extras podem ser encontrados aqui

References

1 CURREY, M. *HM-10 Bluetooth 4 BLE Modules*. 2017. Acesso em: 11 jul. 2024. Disponível em: ⟨https://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/⟩.

2 AnalysIR. *Sending long AC Signals from Flash with IRremote*. 2016. Acesso em: 11 jul. 2024. Disponível em: ⟨https://www.analysir.com/blog/2016/04/11/sending-long-ac-signals-flash-irremote/⟩.

3 Phidgets Inc. *IR Remote Control Guide*. 2023. Acesso em: 11 jul. 2024. Disponível em: ⟨https://www.phidgets.com/docs/IR_Remote_Control_Guide#Transmitting_Data⟩.