TÓPICOS AVANÇADOS

MÓDULOS ESPECIAIS E COMUNICAÇÃO

Além dos sensores e atuadores vistos, o Arduino conta com uma série de módulos especiais (alguns deles que podem inclusive funcionar como entrada ou saída de dados de forma alternada). Entre os seus diversos fins, três são muito importantes e agregam novas possibilidades aos nossos projetos: gravação da dados e comunicação. Por isso, iremos ver nesse capítulo o módulo de cartão MicroSD (que permite a gravação de dados de nosso Arduino em arquivos que podem ser lidos e manipulados) e módulos de comunicação bluetooth, ethernet e wifi (que permitem a comunicação do Arduino com outros dispositivos além do seu acesso à internet para o recebimento e o envio de informações e mesmo para funcionarem como servidores web fornecendo webpages para controle e visualização de informações).

Módulo de Cartão MicroSD

Até o momento, trabalhamos com o entendimento dos sensores e atuadores para adquirir informações do mundo externo (compreendendo o que está acontecendo no entorno do Arduino) e agir sobre ele (seja comunicando ou seja executando ações dos mais diversos tipos). Porém, em nenhum momento, nos preocupamos com uma funcionalidade essencial na grande maioria dos projetos: o registro das informações, armazenando os dados lidos pelos sensores, as ocorrências programadas para serem detectadas e as ações executadas pelo sistema. Quando esse registro nos basta em um sistema local, o Módulo de Cartão MicroSD acaba se tornando muito interessante. Como o próprio nome nos indica, ele consegue dar à nossa placa a possibilidade de ler e escrever em um cartão MicroSD, abrindo ou criando arquivos e guardando neles informações pertinentes. Veremos a seguir os principais métodos necessários além de um exemplo de uso do módulo especial.

Principais métodos para trabalhar com arquivos e MicroSD

Entre os principais métodos necessários para trabalhar com arquivos e cartões SD, estão a verificação da existência de um cartão no leitor, a verificação da existência de um determinado arquivo, a criação de um arquivo novo, a abertura de um arquivo existente, a escrita nesse arquivo (inclusão de linhas) e a exclusão do arquivo.

Verificar se o cartão MicroSD está presente no leitor

```
if(SD.begin(<porta>)) { ... }
```

Abrir um arquivo existente através do nome (ou criar novo) em modo escrita

```
File <nome_desejado> = SD.open(<caminho_nome_extensao>, FILE_WRITE);
```

Abrir um arquivo existente através do nome (ou criar novo) em modo leitura

```
File <nome_desejado> = SD.open(<caminho_nome_extensao>);
```

Escrever uma linha em um arquivo aberto

```
<nome_objeto_file>.println("Conteudo");
```

Fechar e salvar um arquivo após editar

```
<nome_objeto_file>.close();
```

Percorrer um arquivo aberto pegando o seu conteúdo

```
while(<nome_objeto_file>.avaiable(){... <nome_objeto_file>.read(); ...}
```

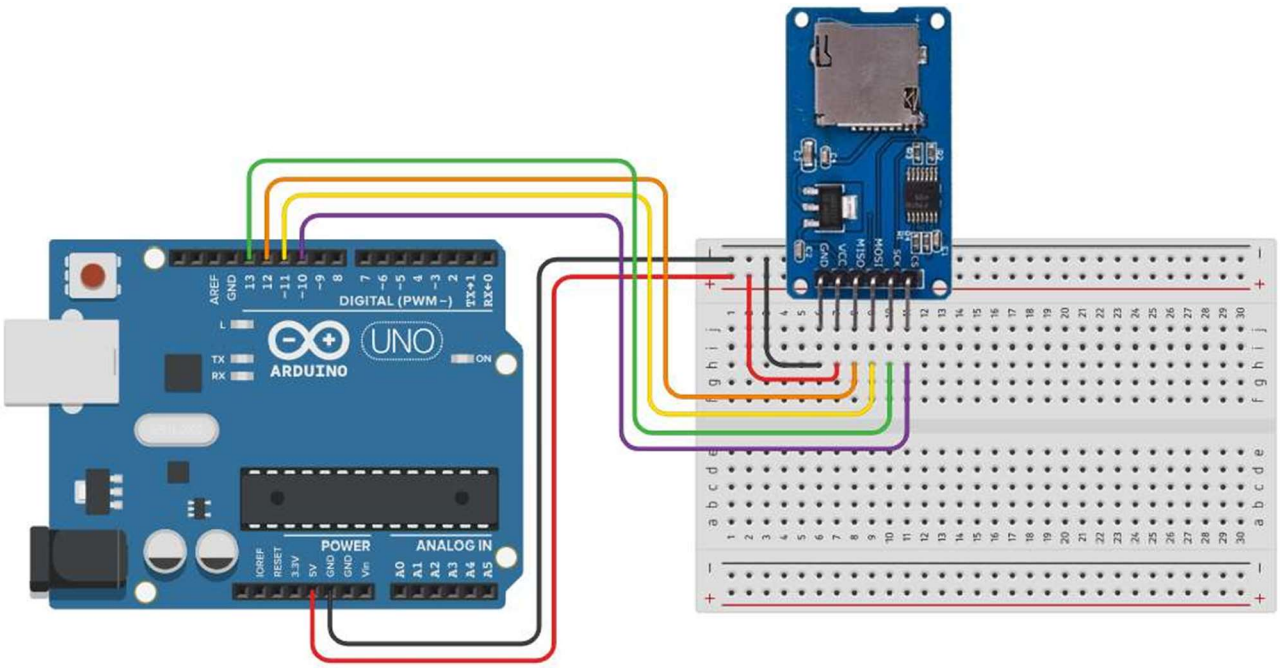
Apagar do MicroSD um arquivo existente (pelo nome)

```
SD.remove(caminho_nome_extensao);
```

Esquema eletrônico de ligação do leitor

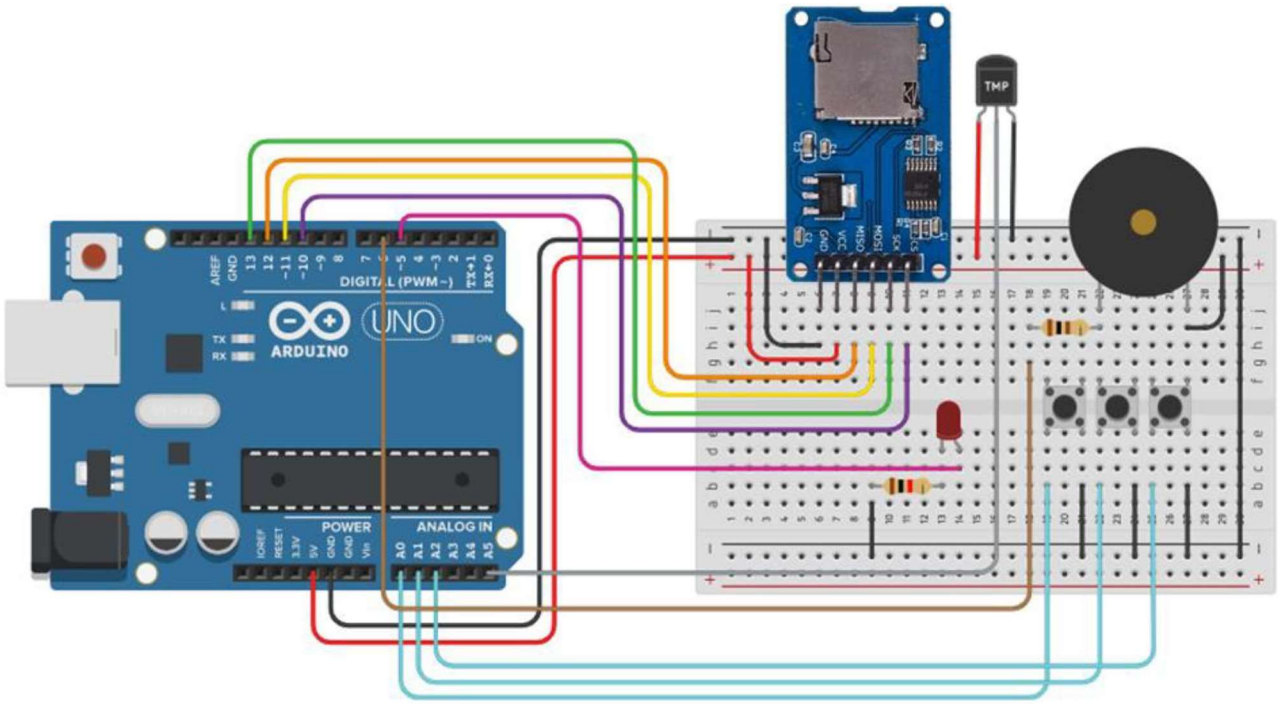
Os leitores de cartão MicroSD compatíveis com o Arduino utilizam a comunicação SPI, um tipo de comunicação Serial especial para periféricos externos à placa e que oferecem uma troca de mensagens padronizada (já utilizamos esta interface no projeto do leitor RFID visto neste livro). Nela, as portas são fixas pois respeitam os pinos especiais que implementam a SPI (SCK, MOSI, MISO) e no Arduino Uno estão organizados da porta digital 11 até a porta digital 13. O único pino que permite variação é o CS que em nosso

exemplo está ligado à porta digital 4. Deste modo, a ligação eletrônica torna-se simples (apesar de ser inflexível) e o esquema é apresentado abaixo. Tenha o cuidado de verificar se o seu modelo de Leitor de Cartão MicroSD possui alimentação de 5V ou 3.3V pois é comum encontrar leitores dos dois tipos.



Exemplo prático: histórico de temperaturas de um forno de pizza

Como exemplo de uso do Leitor de Cartão MicroSD, vamos imaginar um sistema de controle de temperatura de um determinado ambiente monitorado. O sistema deve ter 3 botões: o primeiro inicia um monitoramento onde a cada 1 segundo a temperatura é medida e onde, com o passar do tempo, devem ser identificadas a menor mínima e a maior máxima do período (ligando o led para indicar monitoramento ativo); o segundo finaliza o período (que deve ser contabilizado por um contador crescente) e paralisa o monitoramento, gravando uma linha de log em um arquivo texto chamado *"logtemperaturas.txt"* com o número do período, duração em segundos, temperatura mínima e temperatura máxima medidas (e desliga o led para indicar monitoramento inativo); o terceiro abre o arquivo de log e exibe todo o seu conteúdo no monitor serial. A cada medição (ou seja, a cada um segundo), cada nova mínima ou máxima descoberta deve disparar um beep como indicativo. Para a implementação de nosso exemplo, vamos utilizar o circuito eletrônico mostrado a seguir:



Em relação ao código, necessitamos apenas de alguns dos métodos de manipulação de cartão SD e arquivos além de identificar o clique dos botões e, durante o monitoramento, testar a temperatura medida para verificar se ela é menor que a menor mínima encontrada até o momento ou maior do que a maior máxima encontrada até o momento. Deste modo, teremos um código como o a seguir:

```
#include <SPI.h>
#include <SD.h>

int segundos, período = 0;
float  minima,  maxima;
bool ativo = false;

void setup(){
  Serial.begin(9600);
  pinMode(A0, INPUT_PULLUP);
  pinMode(A1, INPUT_PULLUP);
  pinMode(A2, INPUT_PULLUP);
  pinMode(A5, INPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);

  while(!SD.begin(10)){
    Serial.println("Por favor, insira um cartão MicroSD válido!");
    delay(5000);
  }
  Serial.println("Cartão MicroSD válido! Iniciando o sistema...");

}
```

```
void loop() {

  if(digitalRead(A0) == LOW){
    ativo = true;
    segundos = 0;
    periodo++;
    minima = 999;
    maxima = -999;
    digitalWrite(6, HIGH);
  }

  if(digitalRead(A1) == LOW){
    if(ativo == true){
      ativo = false;
      File arquivo = SD.open("logtemperaturas.txt", FILE_WRITE);
      arquivo.println("Periodo:" + (String)periodo + "|Tempo:" + (String)segundos +
                      "|Mínima:" + (String)minima + "|Máxima:" + (String)maxima;
      arquivo.close();
      digitalWrite(6, LOW);
    }

  }

  if(digitaRead(A2) == LOW){
    File arquivo = SD.open("logtemperaturas.txt");
    Serial.println("");
    Serial.println("*********************************************************");
    Serial.println("*************    LOG    DE    TEMPERATURAS    *************");
    Serial.println("*********************************************************");

    while(arquivo.available()){
      Serial.write(arquivo.read());
    }

    arquivo.close;
  }

  if(ativo == true){
    int valorLido = analogRead(A0);
    float temperaturaLida = map(valorLido, 0, 150, 0, 350);

    if(temperaturaLida < minima){
      minima = temperaturaLida;
      tone(6, 528);
      delay(1000);
      noTone(6);
    } else if(temperaturaLida > maxima){
      maxima = temperaturaLida;
      tone(6, 528);
      delay(1000);
      noTone(6);
    } else {
      delay(1000);
    }

  }
```

Entendendo o código-fonte: Iniciamos o código-fonte com a importação das duas bibliotecas que permitem o acesso a cartões SD (*SD.h*) e a comunicação especial que reconhece o leitor SD como um periférico (*SPI.h*). Além disso, são declaradas variáveis para o armazenamento da mínima e máxima encontradas, o total de segundos decorridos, o

número do período e uma indicação de se o monitoramento está ativo ou parado. No setup, inicializamos a Serial e indicamos o tipo das portas A0, A1, A2 (utilizadas pelos botões como INPUT_PULLUP), A5 (utilizada pelo sensor de temperatura LM35 como INPUT) e 7 (utilizada pelo buzzer como OUTPUT). Após, ainda dentro do setup, testamos se existe um cartão MicroSD presente no leitor, ficando o código preso dentro do while enquanto um cartão não for encontrado e seguindo adiante (para o loop) quando for. No loop, possuímos um condicional para testar o pressionamento de cada um dos 3 botões. Caso o botão em A0 seja pressionado, ligamos o led, inicializamos as variáveis, zerando *segundos*, incrementando *periodo* e atribuindo true para a variável *ativo* (indicando que o monitoramento está ativo), 999 para a variável *minima* (para que qualquer valor lido primeiro seja menor do que ela) e -999 para a variável *maxima* (para que qualquer valor lido primeiro seja maior do que ela). Caso o botão em A1 seja pressionado, se o monitoramento estiver em andamento, desligamos o led, atribuímos false à variável *ativo*, abrimos o arquivo "logtemperaturas.txt" (para escrita) e damos um println dentro dele, escrevendo uma nova linha com as informações de período, tempo, mínima e máxima do período (fechando o arquivo logo em seguida). Caso o botão A2 seja pressionado, escrevemos na Serial um cabeçalho que indica o log de temperaturas, abrimos novamente o arquivo "logtemperaturas.txt" (para leitura) e enquanto ainda tivermos dados não lidos dele, percorremos e escrevemos o conteúdo na Serial. Finalmente, a cada ciclo do loop, capturamos a temperatuda do LM35 (utilizando um map para converter o valor capturado em graus celsius) e testamos se ele é maior do que a maior máxima ou menor do que a menor mínima capturadas até o momento (substituindo o valor caso seja e emitindo um beep para informar uma nova mínima ou máxima encontrada).