Leitura de sensores analógicos

A leitura de sensores analógicos é um pouco mais trabalhosa do que de sensores de resposta binária. Por serem analógicos, eles respondem aos estímulos do mundo externo com tensões entre 0V e 5V que são mapeadas nas portas analógicas em valores entre 0 e 1023. Esse valor é obtido com o uso do método analogRead, informando a porta.

```
digitalRead( <porta> );

<porta>
   porta à qual o sensor analógico está ligado
resposta
   a resposta do método será um valor entre 0 e 1023
```

Algumas vezes esse valor é utilizado diretamente, mas algumas vezes necessitamos convertê-lo em alguma unidade da grandeza que está sendo medida. Para isso, necessitamos estabelecer uma relação entre os valores medidos pelo sensor e os respectivos valores de resposta (entre 0 e 1023) da porta analógica. Podemos então usar duas técnicas: o cálculo de uma constante de conversão ou o mapping dos valores. O problema do primeiro método é que necessitamos utilizar regras de 3 combinadas e caso os sensores meçam grandezas que não partam de 0 (ou que usem negativos, como um sensor de temperatura), ainda necessitamos considerar um offset (deslocamento) a ser somado ao resultado obtido. Como isso é muito comum (ou seja, dificilmente o primeiro valor lido pelo sensor é 0), o Arduino possui um método especial que nos ajuda a converter o valor lido entre 0 e 1023 na unidade da grandeza medida, de acordo com o valor mínimo e máximo obtido pelo sensor.

Mapping de valores

O mapping de valores é o mapeamento da leitura de entrada (algo entre 0 e 1023) de acordo com a grandeza medida pelo sensor, que busca facilitar a conversão na unidade de medida que nos interessa. O Arduino nos oferece para isso o método *map*:

```
map(<valorLido>, <minPorta>, <maxPorta>, <minSensor>, <maxSensor>);

<valorLido>
   valor lido na entrada analógica que queremos descobrir a quanto equivale

<minPorta>
   valor gerado na porta digital quando o valor mínimo da grandeza desejada é lido

<maxPorta>
   valor gerado na porta digital quando o valor máximo da grandeza desejada é lido

<minPorta>
   valor mínimo da grandeza desejada lido pelo sensor

<maxPorta>
   valor máximo da grandeza desejada lido pelo sensor
```

Vamos imaginar que temos um sensor de temperatura que consegue medir temperaturas entre 10ºC e 100ºC. Vamos imaginar também que quando esse sensor está medindo 10°C, o valor gerado na porta analógica é de 55 e quando está medindo 100ºC, o valor gerado na porta analógica é de 985. Essas informações são detalhadas sempre no datasheet dos componentes. Agora a pergunta: caso obtivermos uma leitura de valor 800 na porta, a quanto isso equivale em graus celsius neste sensor? Para descobrir, usamos o map: o primeiro parâmetro é o valor lido que queremos converter em graus celsius (800), o segundo e o terceiro parâmetros são os valores mínimo e máximo gerados na porta analógica pelo sensor (55 e 985) e o quarto e quinto parâmetros são os valores mínimo e máximo lidos pelo sensor (10 e 100).

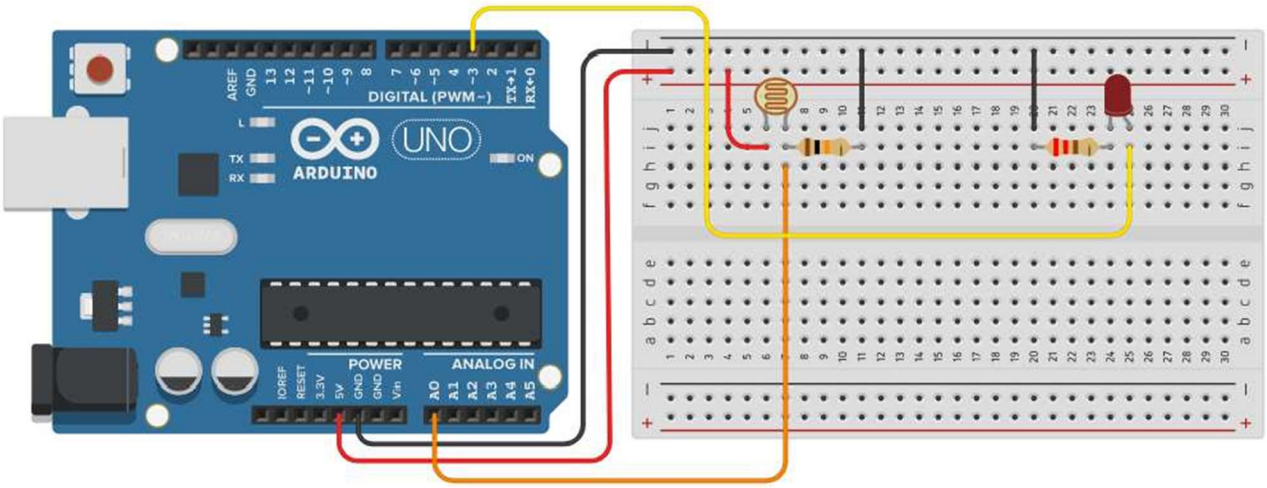
float graus = map(800, 55, 985, 10, 100)

Descobriríamos facilmente que um valor de 800 lido na porta analógica, corresponde à aproximadamente 82ºC e o mesmo método poderíamos utilizar para descobrir a equivalência em graus celsius para qualquer valor lido.

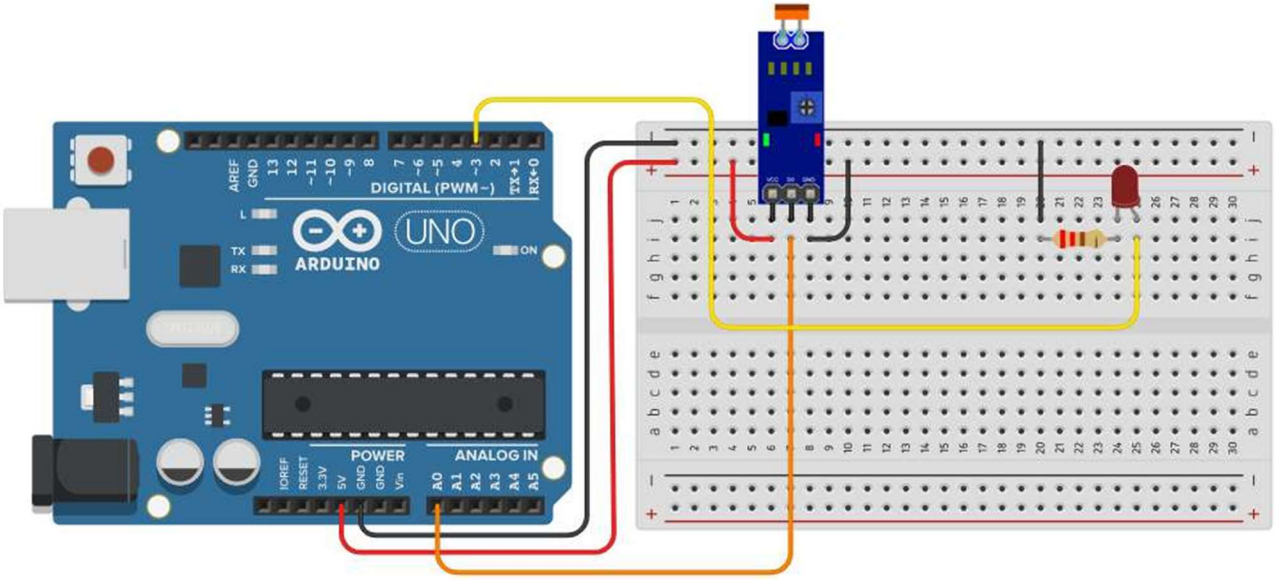
Sensor LDR: exemplo detalhado de captura direta de valor

O sensor LDR como já comentamos é um sensor que muda a sua resistência de acordo com a luminosidade que incide sobre ele. Se conectarmos um de seus pinos à uma porta analógica, conseguimos captar um valor que é maior quanto mais luz é detectada. Aqui, não queremos converter para Lux (unidade formal para a quantidade de luz percebida), mas apenas ler e tomar alguma decisão baseada em seu valor bruto vindo da porta. Assim, a simples leitura com analogRead é suficiente. Em nosso exemplo, vamos então conectar um pino do LDR à porta A0 e quando o valor lido na porta for menor do que 100 (luz muito baixa) ativamos um led para iluminar (ligado à uma porta digital, no nosso caso a porta 3). Aqui, é importante ressaltar que os LDRs podem ser encontrados avulsos (possuindo apenas dois pinos) ou já montados em encapsulamentos com 3 pinos. A grande diferença destes últimos é que colocam o pino de saída do LDR já em série com um resistor de pullup (geralmente de 10kΩ) para reforçar a leitura. Assim, nosso esquema eletrônico será como um dos apresentados a seguir:

Esquema 1: LDR avulso (sem encapsulamento em placa, incluindo o resistor de pullup):



Esquema 2: LDR já encapsulado em uma placa com pullup incluído na mesma (neste exemplo, o sensor tem como primeiro pino o VCC, o pino do meio o OUT que é a saída de dados e o terceiro pino o GND).



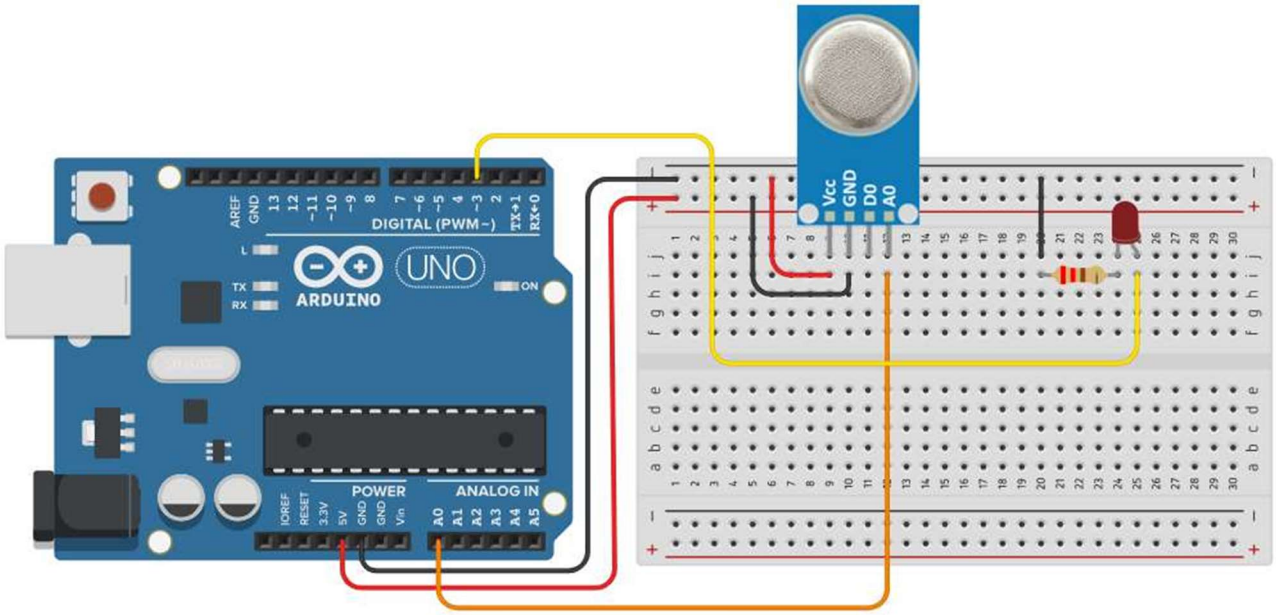
Agora para a implementação de nossa lógica de ligar o led quando estiver escuro, teremos:

```
void setup(){
  pinMode(A0, INPUT);
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);
  Serial.begin(9600);
}

void loop(){
  delay(100);
  int valorLido = analogRead(A0);
  Serial.println("Valor da luminosidade: " + (String)valorLido);

  if(valorLido < 100) {
    digitalWrite(3, HIGH);
  } else {
    digitalWrite(3, LOW);
  }

}
```

Entendendo o código: inicialmente definimos a porta A0 (à qual está ligado o pino de saída do LDR) como de entrada e a porta 3 (à qual está ligado o pino positivo do led) como saída, já definindo seu estado como baixo para o led inicie desligado. Após, dentro do loop, a cada ciclo de tempo lemos a porta A0 e o valor obtido na porta analógica (entre 0 e 1023) é armazenado na variável *valorLido*. Caso *valorLido* for menor do que 100 (luminosidade muito baixa detectada) o led é aceso. Caso contrário, o led é apagado.

Sensor de gás

Os sensores de gás permitem detectar a presença de gases específicos em um ambiente e estimar a sua concentração aproximada (com uma precisão mediana, mas suficiente para detecções simples). Cada sensor é específico para um gás e assim temos sensores de dióxido de carbono, monóxido de carbono, oxigênio, hidrogênio, entre outros. O uso, a exemplo do exemplo do LDR, também é bastante simples, pois estes sensores analógicos possuem um pino (geralmente identificado como A0), através do qual lemos um valor entre 0 e 1023 (maior, quanto maior a concentração do gás detectada). Assim, podemos tanto ligar um led ou soar um alarme avisando da detecção do gás (ou da concentração perigosa a partir de um determinado limite) ou tomar outras medidas desejadas. Em nosso exemplo ilustrativo, iremos utilizar um sensor de monóxido de carbono para avisar quando sua concentração em um ambiente não ventilado (uma sala com portas e janelas fechadas e lareira ligada, por exemplo) for perigosa, acendendo um led de alerta (ligado à nossa porta digital 3). Assim, teremos o seguinte esquema eletrônico:

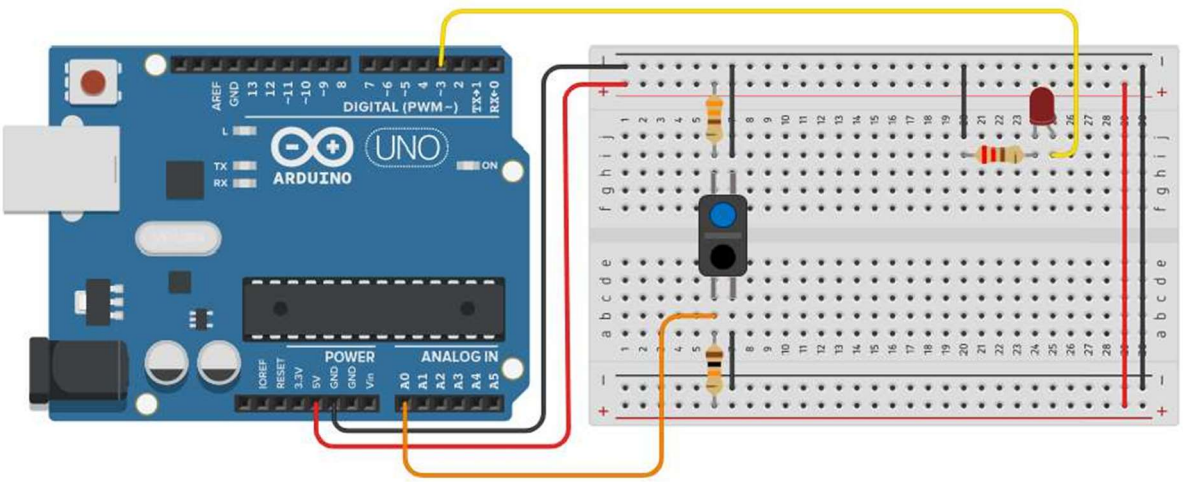


Nossa lógica será muito similar ao exemplo do LDR, pois necessitamos a cada loop ler a porta A0 e descobrir o valor que o sensor de gás está enviando para ela (entre 0 e 1023) para, de acordo com esse valor, ligar ou não o led. Vamos considerar que o limite máximo do valor lido e que representa a concentração de monóxido de carbono no ambiente é de 400. Assim, teremos o seguinte código:

```
void setup(){
  pinMode(A0, INPUT);
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);
  Serial.begin(9600);
}

void loop(){
  delay(100);
  int valorLido = analogRead(A0);
  Serial.println("Concentração de monóxido de carbono : " + (String)valorLido);

  if(valorLido > 400) {
    digitalWrite(3, HIGH);
  } else {
    digitalWrite(3, LOW);
  }

}
```

Sensor óptico reflexivo

O sensor óptico reflexivo é composto de dois elementos: um emissor de luz infravermelha e um receptor de luz infravermelha. Estes elementos são separados por uma "parede" de plástico que não permite que eles "se enxerguem". O receptor só consegue perceber a luz infravermelha emitida quando a mesma reflete em um objeto a sua frente. Assim, este tipo de sensor é muito útil para detectar quando um objeto passa por ele, pois o objeto refletirá a luz infravermelha de acordo com sua distância ou mesmo com o seu material (mais reflexivo ou menos reflexivo ao IR). Com isso, é possível até mesmo identificar alguns tipos de materiais através do sensor. A captura do valor de luz refletida se dá de forma fácil, também através de um pino analógico como nos exemplos anteriores, variando apenas o que fazemos com esse valor. Veja o esquema eletrônico abaixo:



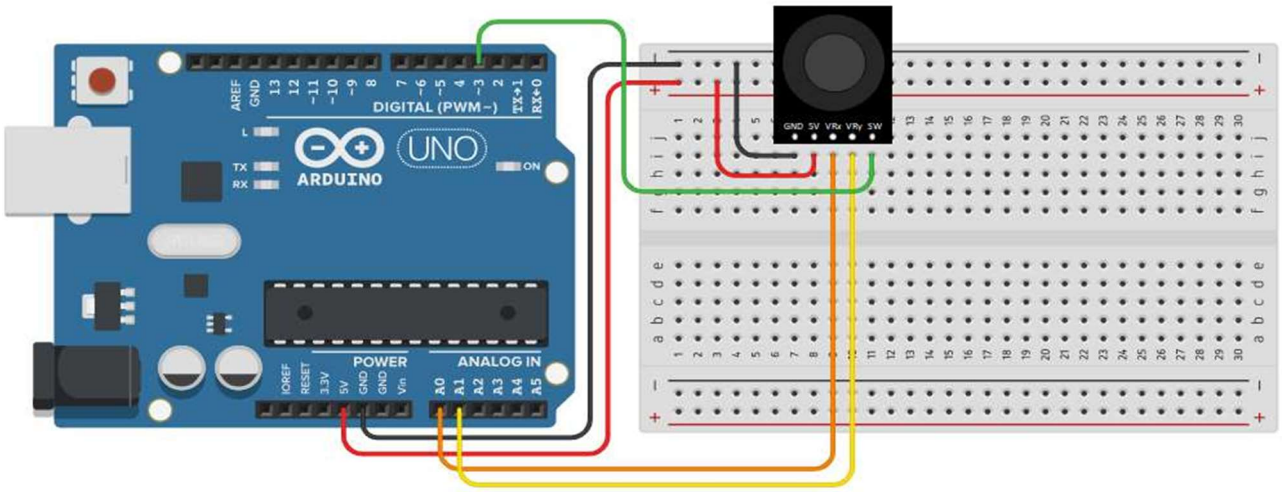
O emissor infravermelho (azul) tem um pino ligado ao negativo e o outro ligado ao positivo em série com um resistor de 330Ω (para baixar sua corrente). Deste modo, permanece o tempo todo ligado emitindo luz. Já o receptor infravermelho (preto) tem um pino ligado ao negativo e o outro pino ligado ao positivo em série com um resistor de pullup de 10kΩ. Desta mesma ligação sai um fio ligando a coluna à porta analógica A0. Quando o receptor perceber luz infravermelha, um valor quantificando essa percepção é enviado à porta analógica. Imagine que queremos usar esse sensor posicionado sobre a lente de um óculos. Sabe-se que quando o olho está aberto, boa parte da luz infravermelha é absorvida por ele, refletindo somente uma pequena parte que gera uma leitura geralmente inferior à 900. Porém, quando o olho é fechado, a pele da pálpebra reflete muito mais luz infravermelha, o que gera uma leitura geralmente superior à 1000. Assim, podemos então considerar valores intermediários (com uma margem de erro) e usar a seguinte lógica para ligar e desligar o led conforme o usuário abre ou fecha os olhos.

```
void setup(){
  pinMode(A0, INPUT);
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);
  Serial.begin(9600);
}

void loop(){
  delay(100);
  int valorLido = analogRead(A0);
  Serial.println("Quantidade de luz IR percebida: " + (String)valorLido);

  if(valorLido < 950) {
    digitalWrite(3, HIGH);
  }

  if(valorLido > 1000) {
    digitalWrite(3, LOW);
  }

}
```

Sensor Joystick

O sensor joystick nos traz um controle analógico capaz de detectar movimentos no eixo x (para esquerda ou direita), no eixo y (para cima ou para baixo) e no eixo z (botão clicado ou não). Assim, diferente dos exemplos anteriores, ele possui dois pinos analógicos e um pino digital. O primeiro pino analógico (chamado de VRx) devolve valores entre 0 e 1023 que quantificam o movimento no eixo x (quando mais perto de 0, mais à esquerda e

quanto mais perto de 1023, mais à direita). O segundo pino analógico (chamado de VRy) devolve valores entre 0 e 1023 que quantificam o movimento no eixo y (quando mais perto de 0, mais para cima e quanto mais perto de 1023, mais para baixo). Já o pino digital (chamado de SW) informa se o botão foi pressionado (LOW) ou não (HIGH). Assim, para utilizarmos o joystick de modo a capturar os movimentos realizados pelo usuário e transformar em ações, devemos ler os 3 valores. Apenas para exemplificar, vamos mostrar como escrever uma mensagem na serial informando qual o movimento o usuário está realizando no joystick. Para isso, vamos analisar o circuito eletrônico abaixo, onde o VRx está ligado à nossa porta A0, o VRy ligado à porta A1 e o SW ligado à porta 3:

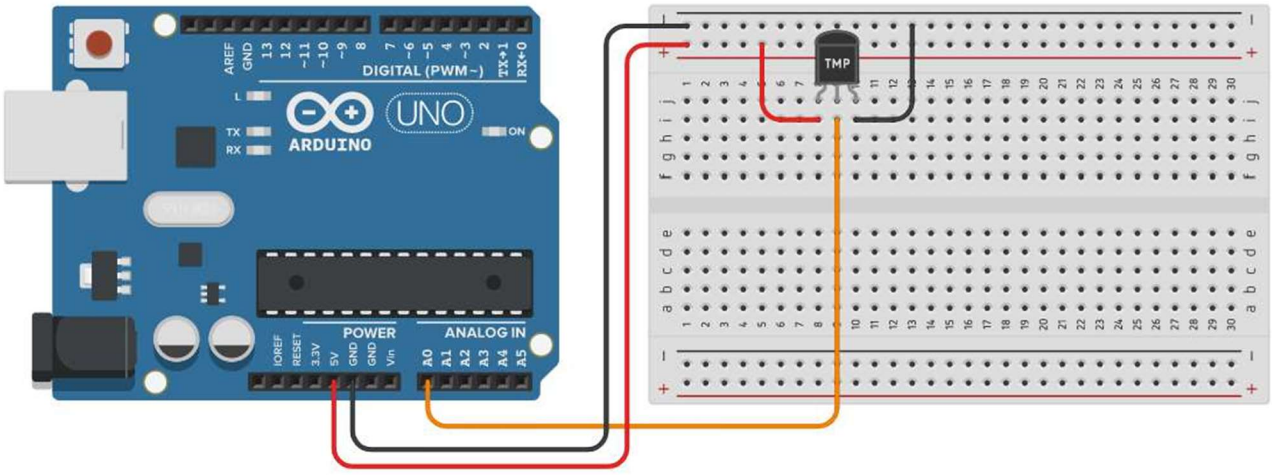


Para a detecção dos movimentos, utilizamos o código abaixo, que pode ser modificado para que ações desejadas sejam efetivamente realizadas.

```
void setup(){
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(3, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop(){
  delay(100);
  if(analogRead(A0) == 0) {
    Serial.println("Esquerda");
  }
  if(analogRead(A0) == 1023) {
    Serial.println("Direita");
  }
  if(analogRead(A1) == 0) {
    Serial.println("Para cima");
  }
  if(analogRead(A1) == 1023) {
    Serial.println("Para baixo");
  }
  if(digitalRead(3) == LOW) {
    Serial.println("Botão clicado");
  }
}
```

Sensor de temperatura LM35_Exemplo detalhado de mapping de valor

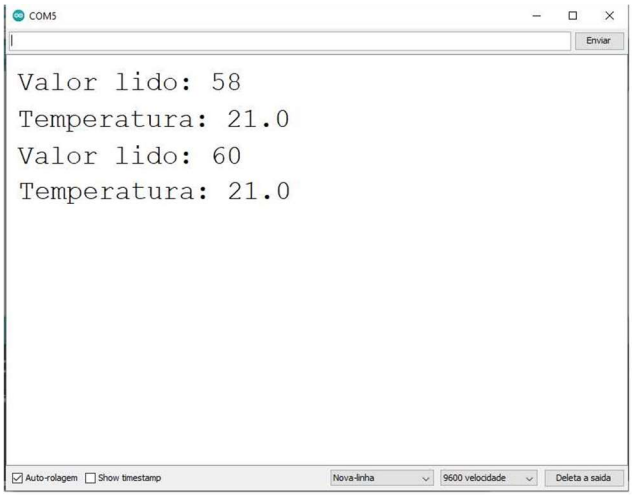
Todos os sensores vistos até agora utilizam diretamente o valor lido na porta analógica. A partir de agora veremos como o mapping pode nos ajudar na conversão destes valores lidos para uma unidade de medida ou para uma referência desejada por nós. Vamos começar com o Sensor de temperatura LM35. Este versátil sensor, nos permite mensurar diversas faixas de temperatura. Segundo o seu datasheet, para uma leitura precisa de temperaturas entre 0 e 150 graus, é gerada na porta analógica um valor entre 0 e 350. Outras temperaturas acima ou abaixo disso são possíveis, mas apresentarão um erro maior. Vamos iniciar com a montagem do circuito eletrônico que é bastante simples, pois o sensor possui um pino VCC à esquerda, um pino de saída (dados) central que ligaremos à nossa porta A0 e um pino de GND à direita. Assim, temos:



Agora, em nosso código, configuramos a porta A0 como entrada e no loop lemos o seu valor analógico (entre 0 e 1023) e logo em seguida, usamos o map fornecendo as informações de máximos e mínimos, para o valor lido seja convertido em graus celsius:

```
void setup(){
  pinMode(A0, INPUT);
  Serial.begin(9600);
}

void loop(){
  delay(100);

  int valorLido = analogRead(A0);
  float temperatura = map(valorLido, 0, 150, 0, 350);

  Serial.println("Valor lido: " + (String)valorLido);
  Serial.println("Temperatura: " + (String)temperatura);
}
```

Entendendo o código: o mapping permitido pelo método map, torna o código muito simples e evita a realização de cálculos massivos e complexos na conversão. No setup a porta A0 está configurada como entrada pois através dela que receberemos o valor vindo de sensor LM35. Inicializamos também a Serial para poder escrever nela (e posteriormente visualizar) os resultados obtidos. Dentro do loop, a cada 100 ms, lemos a porta A0 e armazenamos o valor na variável *valorLido*. Agora, necessitamos converter esse valor lido (qualquer coisa entre 0 e 1023) em graus celsius. Como conhecemos as temperaturas mínima e máxima passíveis de leitura pelo LM35 (0ºC à 150ºC na faixa de valor que queremos utilizar) e qual o valor máximo e mínimo que essas leituras provocam na entrada da porta analógica (0 a 350 conforme datasheet do componente), utilizamos o método map para que o valor lido seja convertido e armazenado na variável temperatura. Após, escrevemos na Serial qual o valor lido através do sensor e qual a temperatura respectiva em graus celsius. Se tudo ocorreu bem, teremos no nosso Monitor Serial uma saída como esta:



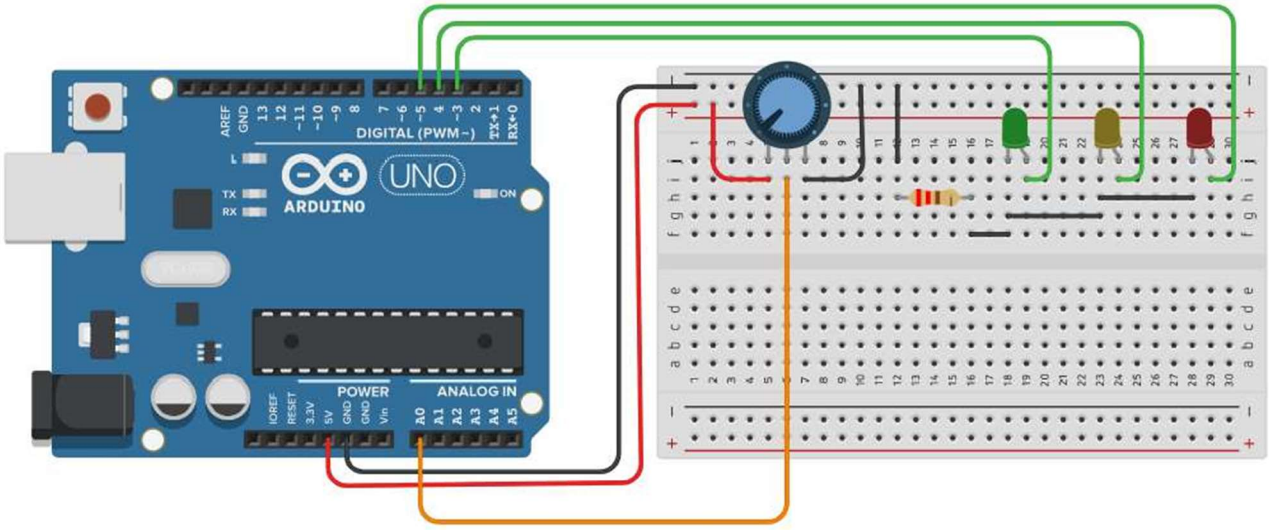
Perceba que a nossa temperatura de saída é sempre "inteira" apesar de mostrada como um decimal. Outra forma de calcular a temperatura de forma ainda mais precisa, segundo o fabricante, caso se necessite dos décimos, seria multiplicar o valor lido por 5000, dividir o resultado por 1024 e dividir novamente o resultado por 10. Assim, teríamos:

*float temperatura = ((valorLido * 5000) / 1024) / 10*

Porém, se estes décimos não são fundamentais para nosso projeto ou análise, torna-se muito mais simples o uso da técnica de mapping através do método map do Arduino.

Potenciômetro

Apesar dos potenciômetros serem componentes eletrônicos do tipo resistor variável, podemos utilizá-los em nossos projetos como se fossem sensores dos quais conseguimos detectar o giro. Para isso, basta conectar as suas extremidades ao 5V e o GND e o seu pino central à uma porta analógica. Isso porque, ao girar o pino, a resistência gerada cria um divisor de tensão entre o pino 5V e central e entre o central o GND. Deste modo, conseguimos modificar a tensão que chega à porta digital, conforme o nosso giro. Para testar, podemos criar um circuito eletrônico como o mostrado abaixo, com o pino central do potenciômetro ligado à porta analógica A0 do Arduino e 3 leds ligados às portas digitais 3, 4 e 5.



Nessa configuração, independente do valor do nosso potenciômetro (que para efeito de exemplo será de 5KΩ), quando seu pino estiver totalmente para a esquerda, o valor lido pela porta analógica será 1000 (mínima resistência implica em máxima tensão) e quando seu pino estiver totalmente para a direita o valor lido pela porta analógica será 0 (máxima resistência implica em mínima tensão). Para inverter, bastaria subtrair o valor lido de 1000 que, com isso, teríamos um valor que aumenta conforme o pino é girado para a direita. Agora, para converter esse valor entre 0 e 1000 em uma resistência entre 0Ω e 5kΩ, podemos utilizar um mapping, mapeando o valor lido com base na escala de valores possíveis na leitura analógica em comparação com a escala de valores possíveis da resistência do potenciômetro. Colocando isso em código, teríamos:

```
void setup(){
  pinMode(A0, INPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  delay(100);
  int valorLido = 1000 - analogRead(A0);
  float resistencia = map(valorLido, 0, 1000, 0, 5000);

  Serial.println("Valor lido: " + (String)valorLido);
  Serial.println("Resistência: " + (String)resistencia);

}
```

Com o método map, o mapping do valor lido (algo entre 0 e 100) para transformação em ohms (entre 0 e 5000) tornou-se simples. Agora, imagine que dentro do loop, logo após as mensagens enviadas para a Serial com os valores, desejássemos que os leds fossem ligados conforme o pino do potenciômetro fosse girado em sentido horário. Aqui, poderíamos utilizar duas lógicas:

- **Lógica 1:** a primeira possibilidade (e mais convencional), é dividir os possíveis valores lidos (de 0 a 1000) em três grupos de valores, sendo o primeiro de 0 a 333 (ligando o led 1), o segundo de 334 à 666 (ligando o led 2) e o terceiro de 667 à 1000 (ligando o led 3). Assim, basta testaria testar o valor lido com três condicionais para identificar em qual grupo (faixa de valor) está e qual o respectivo led que deve ser ligado. Teríamo, então:

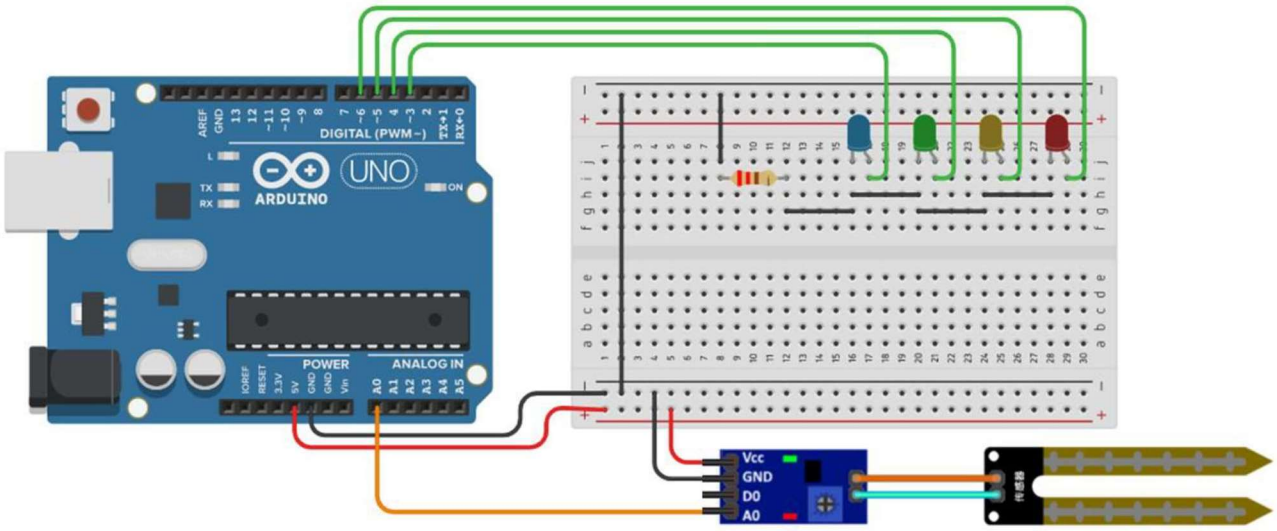
```
if(valorLido >= 0 && valorLido <= 333) {
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
}

if(valorLido >= 334 && valorLido <= 666) {
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
}

if(valorLido >= 337 && valorLido <= 1000) {
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
}
```

- **Lógica 2:** para deixar o código mais elegante, poderíamos utilizar um mapping de valores para converter o valor lido (algo entre 0 e 1000), em um valor entre 1 e 3, que vai nos indicar exatamente qual led ligar. Apesar de necessitarmos de uma linha a mais de código do que na solução anterior, o condicional fica mais simples e não necessita de condição encadeada para testar faixas de valores. Assim, temos:

```
int faixaLed = map(valorLido, 0, 1000, 1, 3);

if(faixaLed == 1) {
  digitalWrite(3, HIGH);
  digitalWrite(5, LOW);
  digitalWrite(3, LOW);
}

if(faixaLed == 2) {
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
}

if(faixaLed == 3) {
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
}
```

Sensor de umidade do solo

O sensor de umidade do solo é especialmente desenvolvido para possibilitar sua introdução no solo a ser monitorado e, de acordo com a umidade percebida, nos devolver um valor analógico que varia de 0 (totalmente seco) até 1023 (totalmente encharcado). Este sensor é composto de duas partes distintas: uma haste, responsável por ficar em contato com o solo e uma placa lógica que captura as micro correntes geradas pela haste, convertendo-as em informação sobre a umidade percebida. A placa geralmente possui 4 pinos, sendo eles o VCC (5V), o GND, o analógico e um pino digital que pode ser utilizado caso não se deseje saber o valor da umidade, mas apenas mudar o estado do sensor quando uma umidade máxima (ajustada através do trimpot presente em seu corpo) for atingida. No nosso exemplo, não iremos utilizar esse pino digital, sendo o pino analógico ligado à nossa porta A0. Para chamar a atenção e ilustrar melhor o nível de umidade do solo, iremos fazer algo similar ao exemplo do potenciômetro, colocando 4 leds que são acionados de acordo com 4 faixas de umidade que vamos estabelecer. Estes leds, serão então ligados nas portas digitais 3, 4, 5 e 6. Deste modo, teremos um circuito eletrônico como o mostrado a seguir:



Para o nosso mapeamento, vamos definir 4 faixas que representem a umidade do solo. Se dividirmos os 1024 valores possíveis por 4, teremos que cada faixa deve contemplar 256 unidades de valor. Assim, poderíamos considerar que valores lidos entre 0 e 255 representam o solo muito seco, valores lidos entre 256 e 511 representam o solo ressecado, valores lidos entre 512 e 765 representam o solo úmido e valores lidos entre 766 e 1023 representam o solo encharcado. Novamente tivemos um certo trabalho para definir as faixas de valor e teremos ainda mais trabalho para, após realizada a leitura, montar condicionais que testem o valor lido em relação às faixas definidas. Isso não é necessário, pois podemos simplificar nossa lógica com o mapping dos valores, transformando entradas entre 0 e 1023 em faixas entre 1 e 4:

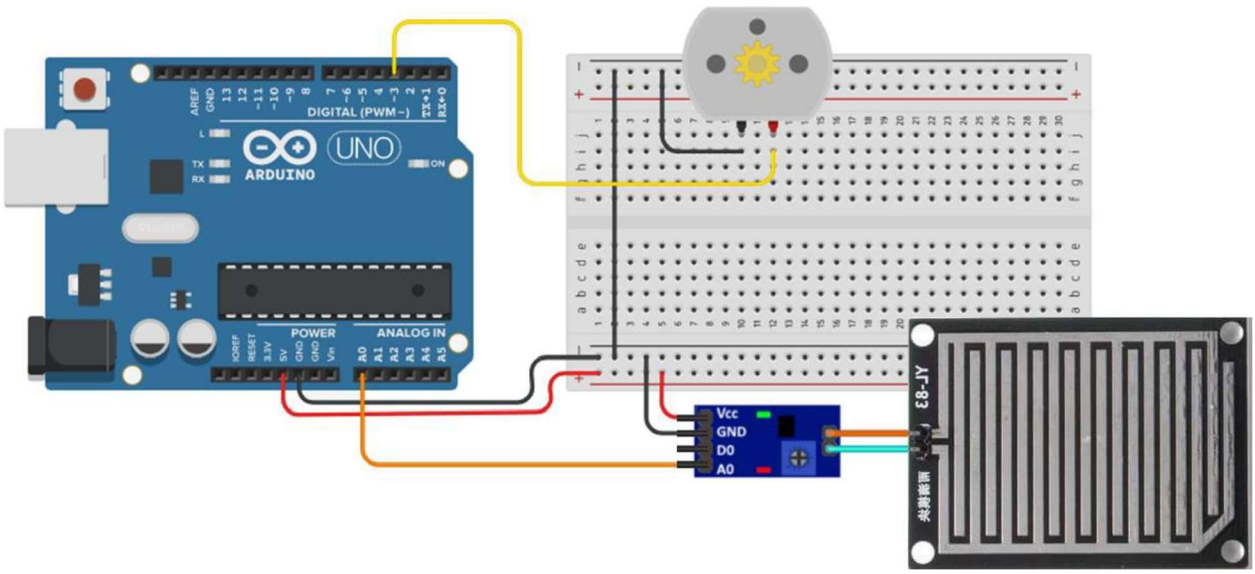
```
void setup(){
  pinMode(A0, INPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  Serial.begin(9600);
}

void loop(){
  delay(100);
  int valorLido = analogRead(A0);
  int faixa = map(valorLido, 0, 1023, 1, 4);

  Serial.println("Valor lido: " + (String)valorLido);
  Serial.println("Faixa de umidade: " + (String)faixa);

  if(faixa == 1) { //totalmente seco
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    digitalWrite(6, HIGH);
  }
```

```
    if(faixa == 2) { //ressecado
      digitalWrite(3, LOW);
      digitalWrite(4, HIGH);
      digitalWrite(5, LOW);
      digitalWrite(6, LOW);
    }
    if(faixa == 3) { //úmido
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, HIGH);
      digitalWrite(6, LOW);
    }
    if(faixa == 4) {  //encharcado
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);
      digitalWrite(6, HIGH);
    }

  }
```

Neste exemplo, estamos apenas ligando e desligando leds como alerta para a umidade percebida no solo, lembrando que quanto mais alta a faixa, mais úmido o solo (por causa do modo como os valores analógicos são lidos, com o 0 sendo a menor umidade e o 1023 sendo a maior umidade). Poderíamos modificar o exemplo para, caso a faixa de solo muito seco seja identificada, ligar um relé (que veremos no capítulo de "Atuadores"), ativando uma bomba de água ou válvula solenoide e realizando a sua rega.

Sensor de respingos

O sensor de respingos é muito similar ao sensor de umidade do solo e conta com uma placa de detecção de líquidos (cujas gotas permitem a passagem de micro correntes) e uma placa que transforma essas micro correntes em informação, devolvendo um valor entre 0 (totalmente molhada) e 1023 (totalmente seca). Os pinos deste sensor são análogos aos do sensor anterior e a nossa única mudança neste exemplo é que em lugar dos leds indicativos, queremos um motor DC alimentado pela porta 3 e que tenha força de, ao perceber um início de chuva, mover uma polia que puxe a corda de um varal para proteger as roupas. Podemos ter, então, um circuito eletrônico como o mostrado abaixo:



Neste exemplo, vamos realizar o mapping com uma lógica um pouco diferente, transformando o valor lido em um percentual de chuva que vá de 0 a 100. O problema é que não é o 0 que representa a placa totalmente seca e o 1023 a placa totalmente molhada e sim o inverso. Como fazemos o mapeamento das duas escalas, se uma está invertida em relação à outra? Simples! Invertendo os valores de uma das escalas. Então, em nosso código, após ler o valor da porta analógica, vamos transformá-lo em um percentual de 0 a 100 mas usando a escala de valor lido na porta invertida (1023 e 0 e não 0 e 1023). Finalmente, caso o percentual de respingos na placa (chuva no nosso caso) for maior do que 20 (que representa 20%) acionamos um motor por 10 segundos para puxar a corda e proteger as roupas estendidas, por exemplo. Deste modo, temos:

```
void setup(){
  pinMode(A0, INPUT);
  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);

  Serial.begin(9600);
}

void loop(){
  delay(100);
  int valorLido = analogRead(A0);
  float percentual = map(valorLido, 1023, 0, 1, 100);

  Serial.println("Valor lido: " + (String)valorLido);
  Serial.println("Percentual de chuva: " + (String)percentual);

  if(percentual > 20) {
    digitalWrite(3, HIGH);
    delay(10000);
    digitalWrite(3, LOW);
  }
```