UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Tecnologia

Relatório Microcontroladores II

FRDMKL25Z - Buzzer e ldr

Larissa Mori Piovezam 200869
Rafael Danelon Correia 103841
Raphaela Carvalho Cruz 157111

Prof. Dr. Talia Simões

Limeira/SP

Outubro

2019

a. Introdução

A placa de desenvolvimento Freedom FRDM-KL25Z NXP é equipada com MCU KL25Z128 com 128KB de memória flash e 16KB de memória SRAM, que pode rodar à 48MHz.
A placa contém um acelerômetro, led RGB, sensor touch, dois conectores mini USB para sua programação e alimentação além de possibilitar instalação de conectores para acesso à GPIO.

b. Materiais

- *Buzzer*
- Sensor LDR(*Light Dependent Resistors*)
- *Protoboard* e *jumpers*
- Computador provido de Ubunto
- Placa FRDMKL25Z e cabo USB
- Compilador Mbed armazenado na nuvem

c. Objetivos

Evidenciar o funcionamento do sensor LDR e da "campainha" chamada de *buzzer* instalados na *protoboard* e a placa Freedom através do software para criação e simulação de programas escritos no ambiente Mbed. Os códigos deste relatório são direcionados ao funcionamento do buzzer condicionado à leitura do ldr.

d. Desenvolvimento

Esquemático da ligação dos componentes na protoboard e placa Freedom KL25Z.

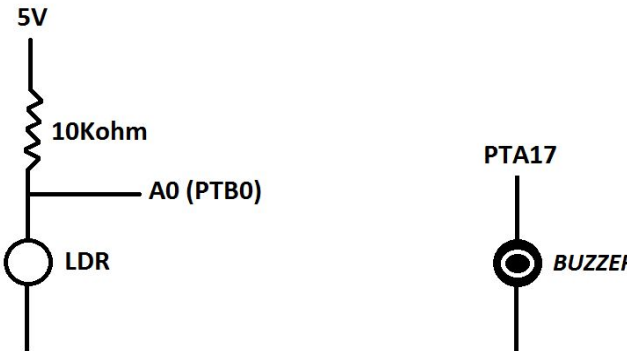

Figura 1 - Esquemático de ligações passados em aula.

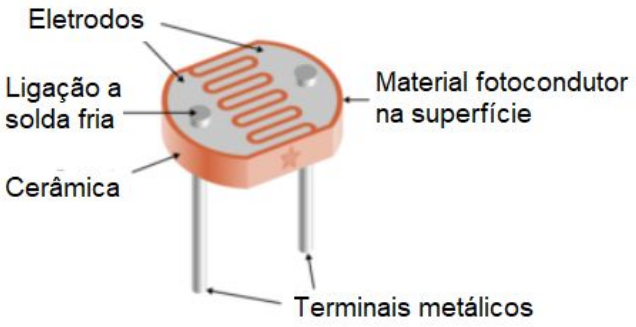

Figura 2 - Esquemático da composição de um fotoresistor.[4]

Códigos para simulação de mudança de cores do led rgb conforme exposição luminosa..

main.cpp

```
#include <mbed.h>
AnalogIn ldr (A0);
DigitalOut buzzer(PTA17);
float x;

int main()
{
    while(true)
    {
        x=ldr.read();
        if(x<0.2){
            buzzer = 1;
            }
            else
            {
             buzzer=0;
            }
    }
}
```

Primeiramente, em todos os códigos, deve-se incluir a biblioteca oficial mbed, neste caso, a única utilizada. Em seguida, define-se a entrada analógica do LDR na porta A0, a saída digital do buzzer conectada em PTA17 e a variável x do tipo *float*. Em seguida, no int main(), ocorre a configuração da lógica do programa a operar. Enquanto for verdade, laço *while(true)*, a variável x recebe a leitura do fotoresistor LDR. Após o recebimento da leitura, pela variável x, em caso de esta receber valor menor que 0.2, nível de luminosidade do ambiente, o *buzzer* permanece desligado, do contrário, se a variável x for superior a 0.2, ou seja, receber maior iluminação, o *buzzer* será acionado. As estruturas *if* e *while* são finalizadas, nesta ordem e por último o *int main()*. Sucintamente, o código opera da seguinte forma: o fotoresistor ldr capta a intensidade da luz ambiente, esta sendo baixa, a campainha permanece desligada porém, em caso contrário, a campainha é acionada.

main.cpp

```
#include <mbed.h>
AnalogIn ldr (A0);
DigitalOut buzzer(PTA17);
PwmOut ledr(LED1); //setando pwm pro led vermelho
PwmOut ledg(LED2); //setando pwm pro led verde
PwmOut ledb(LED3); //setando pwm pro led azul
float x;

int main()
  {
    while(true)
    {
      x = ldr.read();
       if(x<0.33)
       {
        ledg=1;
        ledb=1;
        ledr=1;
        buzzer=1;
        buzzer=0;
            for (float p = 0.0f ;p<1.0f ; p += 0.1f)
            {
             ledr = p;
             wait(0.1);
            }
      if(x>0.33 && x<0.6)
      {
       ledg=1;
       ledb=1;
       ledr=1;
       buzzer=0;
            for (float p = 0.0f ;p<1.0f; p += 0.1f)
            {
             ledg = p;
             wait(0.1);
            }
       }
       if(x>0.6)
       {
       ledg=1;
       ledb=1;
       ledr=1;
       buzzer=1;
            for (float p = 0.0f ;p<1.0f; p += 0.1f)
            {
             ledb = p;
             wait(0.1);
            }
      }
     }
    }
```

Primeiramente, em todos os códigos, deve-se incluir a biblioteca oficial mbed, neste caso, a única utilizada. Em seguida, define-se a entrada analógica do LDR na porta A0, a saída digital do buzzer conectada em PTA17 as três saídas pwm do led, sendo vermelho, verde e azul associados aos leds 1, 2 e 3, ou seja, ledr, ledg e ledb e a variável x do tipo *float*. Em seguida, no *int main()*, ocorre a configuração da lógica do programa a operar. Enquanto for verdade, laço *while(true)*, a variável x recebe a leitura do fotoresistor LDR. Após o recebimento da leitura, pela variável x, em caso de esta receber valor menor que 0.33, nível de luminosidade do ambiente, o *buzzer* permanece desligado e o laço *for* é lido, implementando o valor de LED RGB parte de 0, e é implementado de 0.1 em 0.1 multiplicado pela

função f até atingir o valor 1.0 equivalente à variável p, posteriormente recebida especificamente pela "saída vermelha" do LED RGB, emitindo assim, progressivamente um aumento de luz vermelha emitida.
Caso a leitura que x recebe estar dentro do intervalo 0.33 a 0.6, nível de luminosidade menor que a do ambiente, o LED RGB emite a luz verde e o *buzzer* ainda permanece desligado permanece desligado e o laço *for* é lido, o valor de LED RGB parte de 0, e é implementado de 0.1 em 0.1 multiplicado pela função f até atingir o valor 1.0 equivalente à variável p, posteriormente recebida especificamente pela "saída verde" do LED RGB, emitindo assim, progressivamente um aumento de luz verde emitida.
Entretanto para leituras em que x excede o valor 0.6, nível baixíssimo ou nulo de luminosidade do ambiente, o *buzzer* é acionado e o laço *for* é lido, o valor do LED RGB parte de 0, e é implementado de 0.1 em 0.1 multiplicado pela função f até atingir o valor 1.0 equivalente à variável p, posteriormente recebida especificamente pela "saída azul" do LED RGB, emitindo assim, progressivamente um aumento de luz azul emitida.

É importante notar que, para adequado funcionamento do laço *for* dentro de cada condição *if*, deve-se estipular o estado desligado para cada led, dado que a lógica é invertida, o estado desligado é equivalente ao estado 1. Similarmente, o mesmo é válido para o *buzzer*, iniciado em estado desligado, e por padrão, ao contrário do led, desligado equivale ao estado 0. Além disso, é necessário o intervalo de 0.1 segundos declarado na função *wait()* evidenciando melhor as trocas de estado.

e. Resultados

O teste do primeiro código, se trata do funcionamento do buzzer de acordo com a luminosidade em apenas duas condições, uma à "nível ambiente" sem acionamento de campainha e outra, considerando uma forte iluminação atrelado ao acionamento do buzzer. O código do desafio, um pouco mais elaborado por também incluir aumento gradativo de intensidade do led, permite observar que, em luz ambiente, a tensão medida no sensor de luminosidade era baixa, e por conta disso a saída do LED RGB era a cor vermelha. Posteriormente, ao fazer sombra sobre o sensor pode-se notar que a tensão aumentou e logo a saída tornou-se verde. Finalmente, quando os dedos eram colocados no sensor, restringindo ao máximo possível a luminosidade recebida, a tensão tornava-se ainda maior, e portanto, a saída tornou-se cor azul e o *buzzer* era acionado.

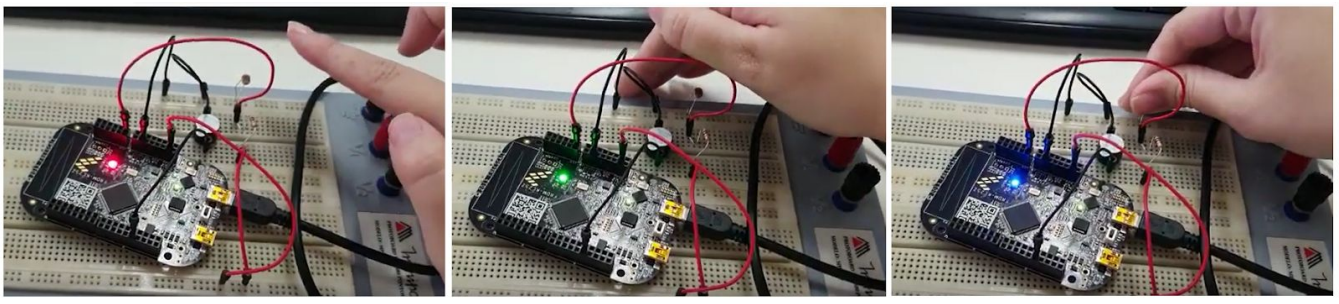

Figura 3- Circuito na protoboard associado à KL25Z.

O código do desafio em funcionamento pode ser visto na íntegra no seguinte link:
https://www.youtube.com/watch?v=Y7lubwkAUDM

f. Conclusão

As simulações na placa trazem a possibilidade de fazer projetos diferentes utilizando ferramentas, físicas e em nuvem, de forma fácil e acessível a fim de testar funcionamento de lógicas e projetos, e neste caso, os códigos operam de forma a evidenciar a variação de sensibilidade fotovoltaica do LDR associado ao funcionamento do *buzzer*.

g. Referências

[1] Documentação Mbed OS5. Disponível em:<https://os.mbed.com/docs/mbed-os/v5.9/introduction/index.html> Acesso em 15 de outubro de 2019.

[2] Apresentando a Freedom KL25Z. Disponivel em:<https://www.filipeflop.com/blog/apresentando-frdm-kl25z/> Acesso em 15 de outubro de 2019.

[3] Instructables circuits DIY LittleBits. Disponível em: <https://www.instructables.com/id/DIY-littleBits-Introduction/> Acesso em 18 de outubro de 2019.

[4] Mech4study. Types of Lighting Sensorshttps://www.mech4study.com/2018/12/types-of-lighting-sensors.html

[5] UNICAMP. Aulas teóricas e práticas de microcontroladores II.