Projeto Final de Microcontroladores 2

Alarme de Temperatura

Aline Romanini 145157

Marlon Felipe 121207

Osvaldo Torezan 147558

Novembro de 2016

Conteúdo

1. Placa de Desenvolvimento Freescale KL25Z

Desenvolvida pela Freescale, a placa de prototipagem KL25Z foi contém como controlador um microprocessador ARM Cortex M0, que proporciona vários canais de comunicação, múltiplas portas I/O e também a possibilidade de trabalhar com uma enorme gama de sensores robustos, seja para uso comercial ou apenas hobby. Abaixo descreveremos sucintamente algumas características importantes da placa, e também abordaremos em tópicos algumas delas.

Tabela 1 – Principais Características KL25Z [1].

Microcontrolador	ARM Cortez M0+
Frequência de Operação	48 MHz
Pinos de GPIO	66
Protocolos de Comunicação	SPI, I2C, Serial
Conversor Analogico Digital (ADC)	16 bit
Conversor Digital Analógico (DAC)	12 bit
Armazenamento	16KB Ram, 128KB Flash
Periféricos Integrados	Acelerômetro e Capacitive Touch Sensor

Abaixo podemos conferir com as imagens, o aspecto geral da placa e também seu esquema de pinos, com os protocolos de comunicação já embutidos.

Figura 1 – Aspecto geral da placa KL25Z [1].

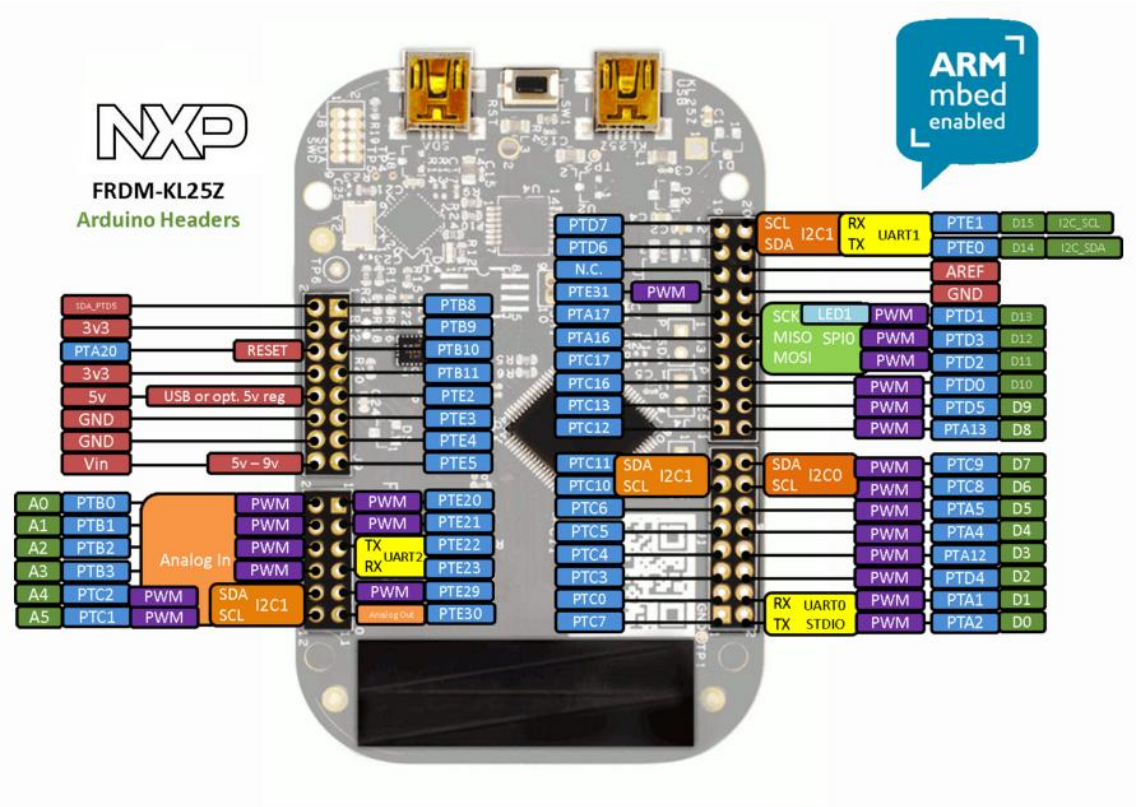


Figura 2 – Esquema de pinagem da placa KL25Z [1].

1.1. Microcontroladores ARM Cortex-M

Os microcontroladores ARM da família Cortex-M são caracterizados pelo seu tamanho reduzido e pelo baixo consumo de energia. Desenvolvido e aprimorado para ajudar os desenvolvedores em uma rápida prototipagem, agregando baixo custo de desenvolvimento e versatilidade de conexão, os microcontroladores ARM Cortex-M estão presentes em dispositivos comerciais que requerem um compromisso entre custo e desempenho [2].

Especificamente, na placa KL25Z, utiliza-se um processador da família M0+, as quais possuem a característica de ser os mais eficientes em consumo de energia [3]. Tal microcontrolador proporciona além do baixo consumo energético e alto desempenho, uma maior versatilidade de programação, pois o mesmo é uma excelente alternativa aos microcontroladores de 8 bits, comumente usados para aplicações com baixo custo (Arduino por exemplo) [3].

Abaixo podemos conferir algumas características e benefícios da família Cortex-M0+ [3]:

- Baixo consumo energético, aproximadamente 9.4uW/MHz;
- Alta versatilidade de conexão com periféricos utilizando protocolos comumente encontrados em sensores e atuadores (SPI, I2C e Serial);
- Compatibilidade de instruções com a família de microcontroladores Cortex-M0.

Podemos destacar também a extrema praticidade de termos um microcontrolador SoC (Systems on Chips), onde encontramos em apenas um único chip, uma implementação de hardware para memórias, unidades de processamento, entradas e saídas digitais e analógicas, protocolos de comunicação, dentre outros. Na figura abaixo podemos conferir um exemplo de um sistema SoC.

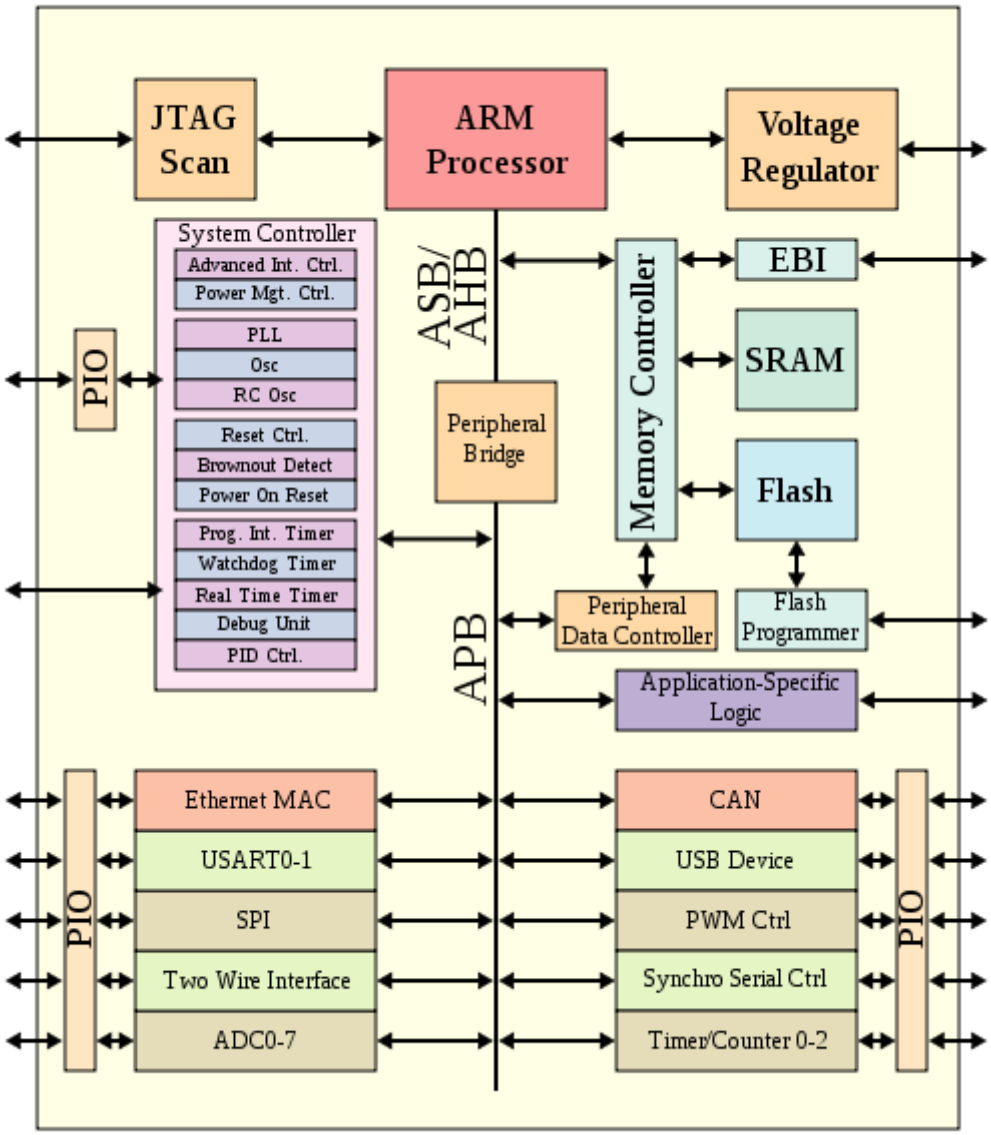


Figura 3 – SoC [4].

1.2. Protocolo I2C

O protoloco I2C (Inter Integrated Circuit) permite realizarmos a conexão entre um dispositivo *Master* (dispositivo controlador) e vários dispositivos *Slave* (dispositivo controlado) através de apenas duas linhas de comunicações.

No protocolo I2C, temos duas linhas de comunicação que são:

- **SDA**: Utilizada para transferência dos dados de forma serial
- **SCL**: Utilizada para sincronia através de clock, gerada sempre pelo dispositivo *Master*.

Com isso, eliminamos o problema de se realizar múltiplas comunicações entre um microcontrolador e vários dispositivos, bastando apenas conectá-los ao protocolo I2C quando disponível. Na figura abaixo, encontramos uma rápida abstração de hardware para o protocolo I2C, onde vemos a conexão entre um dispositivo *Master* e um dispositivo *Slave.*

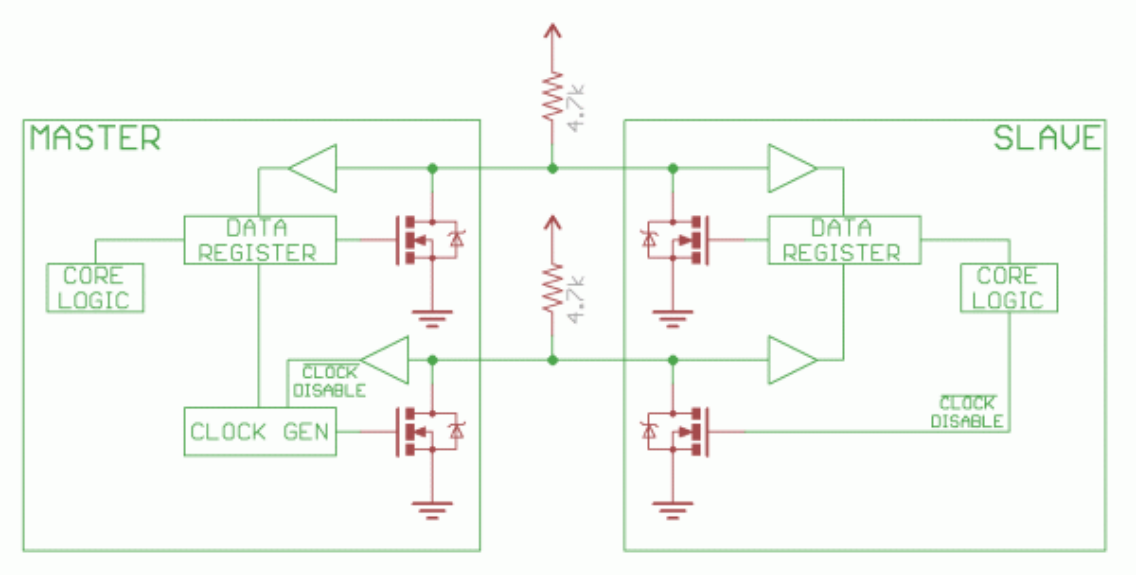


Figura 5 – Abstração de Hardware para o protoloco I2C [5].

1.3. Protocolo SPI

O protocolo SPI (Serial Peripheral Interface) nos permite realizar a comunicação entre um dispositivo *Master* (dispositivo controlador) e vários dispositivos *Slave* (dispositivo controlado) através de apenas três linhas de comunicações.

No protocolo SPI, temos três linhas de comunicação que são:

- **SCK**: Utilizada para sincronia através de clock, gerada sempre pelo dispositivo *Master*.
- **MOSI**: *Master Output Slave Input,* que é a entrada de dados no dispositivo *Slave* e a saída de dados no dispositivo *Master*
- **MISO**: *Master Input Slave Output,* que é a entrada de dados no dispositivo *Master* e a saída de dados no dispositivo *Slave.*

Com isso, eliminamos o problema de se realizar múltiplas comunicações entre um microcontrolador e vários dispositivos, bastando apenas conectá-los ao protocolo SPI quando disponível. Na figura abaixo, encontramos uma rápida abstração de hardware para o protocolo SPI, onde vemos a conexão entre um dispositivo *Master* e um dispositivo *Slave.*

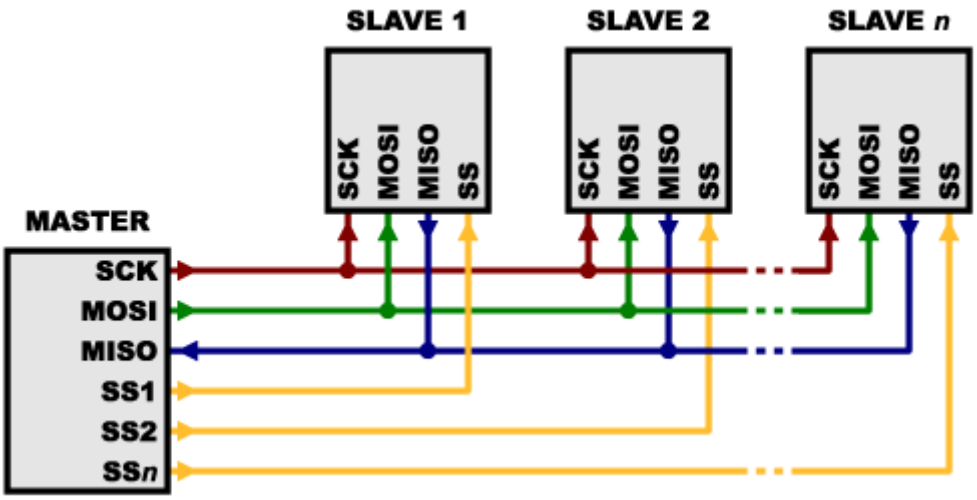


Figura 6 – Abstração de Hardware par ao protocolo SPI [6].

2. Projeto Final

O projeto final utilizando a placa KL25Z consiste em um alarme de temperatura,
onde o usuário ira selecionar uma temperatura limite para o sistema, através dos botões de interação (aumento e diminuição da temperatura). O projeto contém um display LCD 16x2 para exibição das informações.

Após selecionar uma temperatura, o sistema irá verificar com uma frequência de aquisição dos dados do sensor a cada segundo, a temperatura ambiente. Quando a temperatura ambiente for maior que a temperatura selecionada, o sistema acionará um aviso luminoso através do LED vermelho da própria placa, simulando um atuador.

2.1. Sensor de DHT11

Escolhemos este sensor para colocar no projeto, pois o mesmo é de fácil utilização e implementação para a placa KL25Z. Comumente utilizado em aplicações hobbystas, este sensor realiza a leitura da temperatura e umidade ambiente com precisões de ± 2°C e ± 5%RH respectivamente [7].

Para conectamos o sensor na placa KL25Z, utilizamos 3 (três) pinos, sendo:

- Vcc – Alimentação 5V
- GND – Ground (0V)
- Data – Dados da comunicação

Podemos conferir na figura abaixo o modelo de sensor utilizado.

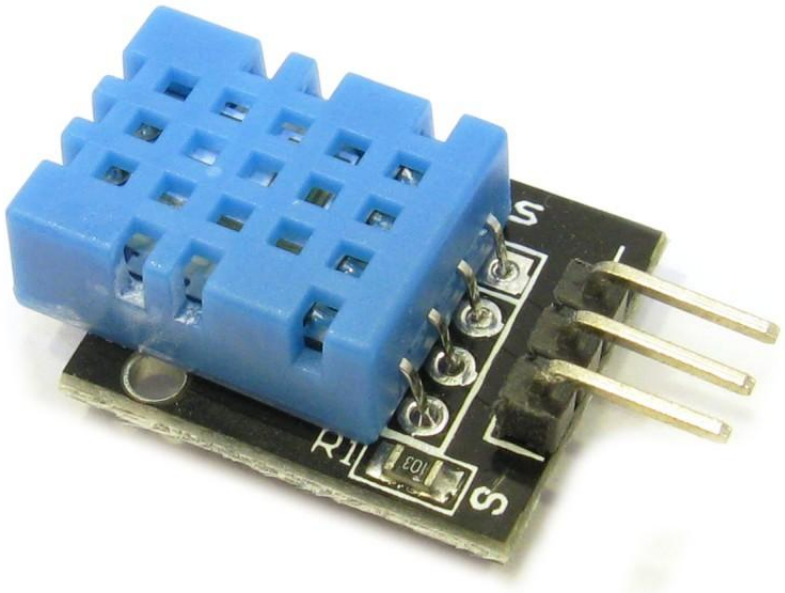


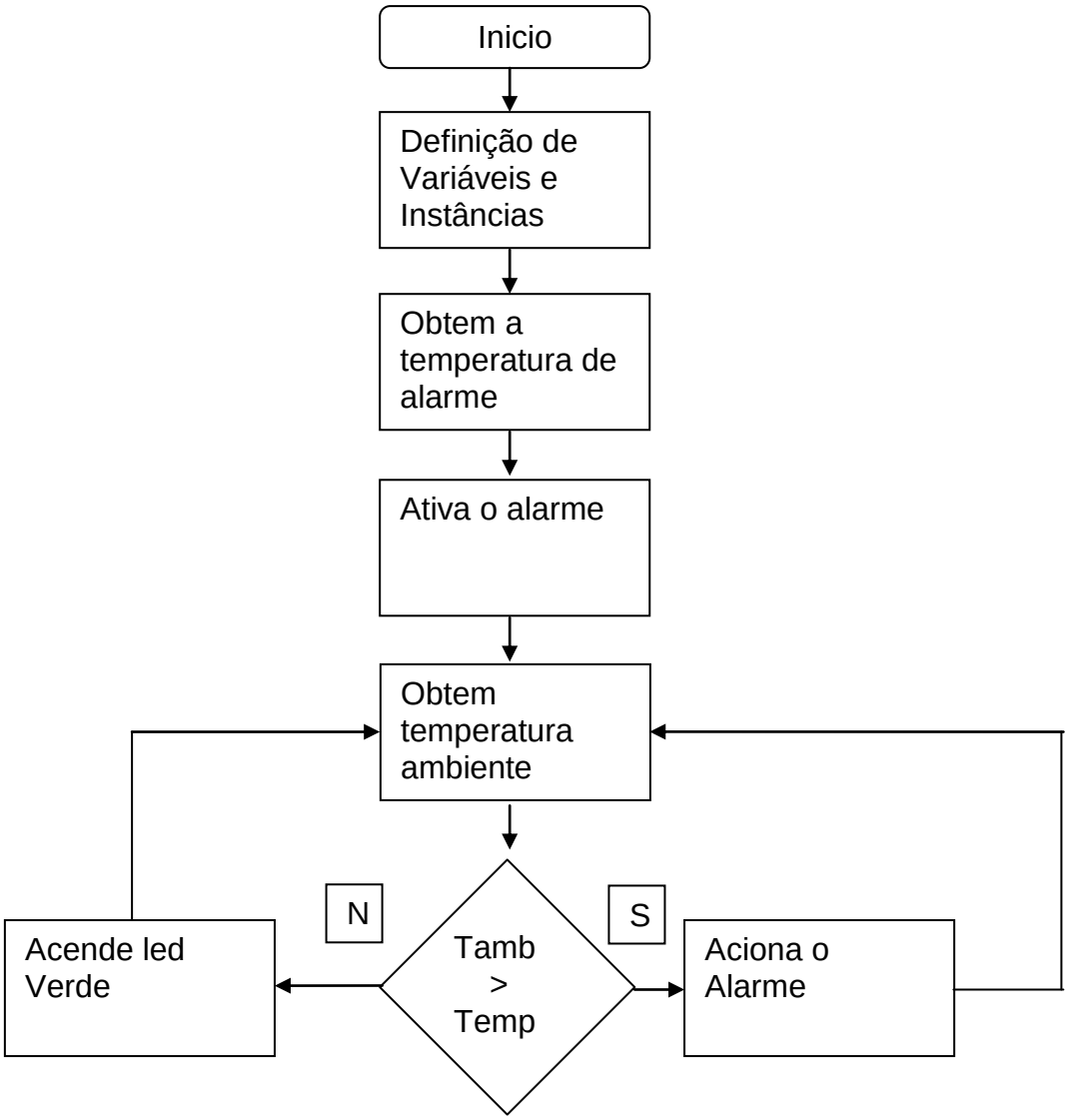
Figura 7 – Sensor de Umidade e Temperatura DHT11.

As especificações do sensor podem ser conferidas na tabela abaixo.

Tabela 2 – Especificações do sensor DHT11 [7].

Parâmetro	Condição	Mínimo	Típico	Máximo
Umidade				
Resolução		1%RH	1%RH	1%RH
Acurácia	25°C		±4%RH	
	0-50°C			±5%RH
Range de Leitura	0°C	30%RH		90%RH
	25°C	20%RH		90%RH
	50°C	20%RH		80%RH
Temperatura				
Resolução		1°C	1°C	1°C
Acurácia		±1°C		±2°C
Range de Leitura		0°C		+50°C

2.2. Fluxograma do Firmware



2.3. Firmware

```
// ------------------------------------
// Projeto Final
// Microcontroladores 2 - 10/10/2016
// Sensor de Temperatura e Umidade DTH11
// ------------------------------------
// Aline Romanini   145157
// Marlon Felipe    121207
// Osvaldo Torezan  147558
// ------------------------------------

// Inclui a Library do mbed
#include "mbed.h"
// Inclui a Library do Sensor de Temperatura e Umidade DHT11
#include "DHT.h"
//Inclui a Library do Display LCD
#include "TextLCD.h"

//Cria uma instância do LCD
TextLCD lcd(PTE0, PTE1, PTE2, PTE3, PTE4, PTE5);

//Cria uma instância do sensor de Temperatura
DHT sensor(PTE20, DHT11);

//Botão para aumentar temperatura de ajuste
DigitalIn btnAumenta(PTC7);
//Botão para diminuir temperatura de ajuste
DigitalIn btnDiminui(PTC0);
//Botão de selecção
DigitalIn btnEnter(PTC5);
//Led interno Vermelho
DigitalOut led(LED1);
//Led interno Verde
DigitalOut ledV(LED2);

//Função que exibe a temperatua no LCD
void ExibeTemperatura(float temperatura)
{
 //Define a posição do LCD na coluna 1 e linha 2, no caso (0,1)
 lcd.locate(0,1);
 //Limpa a linha com 16 espaços
 lcd.printf("                ");
 //Define a posição do LCD na coluna 1 e linha 2, no caso (0,1)
 lcd.locate(0,1);
 //Escreve e temperatua no LCD
 lcd.printf("%.2f",temperatura);
 //Escreve a unidade da temperatura no LCD
 lcd.printf(" oC");
}

//Função Principal
int main()
{
    //Apaga o LED Vermelho da Placa
    led = 1;
    //Acende o LED Verde da Placa
    ledV = 0;

    //Define a tempertura inicial de escolha como 25 graus celcius
    float temperatura = 25;

    //Escreve o texto abaixo na primeira linha do LCD
    lcd.locate(0,0);
    lcd.printf("Set Temp Celsius");

    //Exibe a temperatura inicial no LCD
    ExibeTemperatura(temperatura);
```

```
    //Laço infinito para que o usuário determine a temperatura do alarme.
    //A temperatura será entre o range de temperaturas que o sensor foi construído
    //ou seja, entre 0 e +50°C
    while(1)
    {
     //Aumenta a temperatura
     if (btnAumenta == 0)
     {
      //Espera 100ms para auemtnar a temperatura
      wait(0.1);
      //Caso o usuário tenha mesmo pressionado o botão, e a temperatura seja menor que +50°C, en
tão aumenta 1°C na temperatura e exibe no LCD
      if (btnAumenta == 0 && temperatura < 50)
      {
        temperatura++;
        ExibeTemperatura(temperatura);
        }
     }

     //Diminui a temperatura
     if (btnDiminui == 0)
     {
      //Espera 100ms para auemtnar a temperatura
      wait(0.1);
      //Caso o usuário tenha mesmo pressionado o botão, e a temperatura seja maior que 0°C, entã
o diminui 1°C na temperatura e exibe no LCD
      if (btnDiminui == 0 && temperatura > 0)
      {
        temperatura--;
        ExibeTemperatura(temperatura);
        }
    }

     //Caso o usuário tenha pressionado o botão de "Enter", o firmware encerra o "while" e ativa
 o alarme
     if (btnEnter == 0)
        break;
    }

    //Limpa o display LCD
    lcd.cls();
    //Define o texto na linha 1 e coluna 1, no caso (0,0)
    lcd.locate(0,0);
    //Escreve "Alarme Ativado"
    lcd.printf("Alarm Activate");

    while(1)
    {
     //Chama a função da biblioteca do sensor e requisita a leitura de dados do mesmo
     sensor.readData();
     //Chama a função da biblioteca do sensor e requisita a temperatura em graus Celcius
     float temperaturaS = sensor.ReadTemperature(CELCIUS);
     //Exibe a temperatura ambiente no display LCD
     ExibeTemperatura(temperaturaS);

     //Caso a temperatura ambiente seja maior que a temperatura de alarme
     if (temperaturaS > temperatura)
     {
      // Dispara o alarme com um sinal luminoso no LED Vermelho da Placa
      //Apaga o LED Verde
      ledV = 1;
      //Acende o LED Vermelho
      led = 0;
      wait(0.1);
      led = 1;
      wait(0.1);
```

```
      //}
     }
     else
        ledV = 0; //Acende LED Verde

     //Espera 1 segundo para realizar uma nova leitura do sensor
     wait(1);
    }
}
```

2.4. Documentação e Disponibilização

Todo o projeto que consiste no código fonte as bibliotecas dos componentes está disponibilizado na plataforma do próprio mbed, no link

https://developer.mbed.org/users/OsvaldoTNeto/code/Projeto/

3. Referencias Bibliográficas

[1] https://developer.mbed.org/platforms/KL25Z/

[2] https://www.arm.com/products/processors/cortex-m

[3] https://www.arm.com/products/processors/cortex-m/cortex-m0plus.php

[4] https://en.wikipedia.org/wiki/ARM_architecture#/media/File:ARMSoCBlockDiagram.svg

[5] https://learn.sparkfun.com/tutorials/i2c

[6] https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

[7] http://www.micropik.com/PDF/dht11.pdf