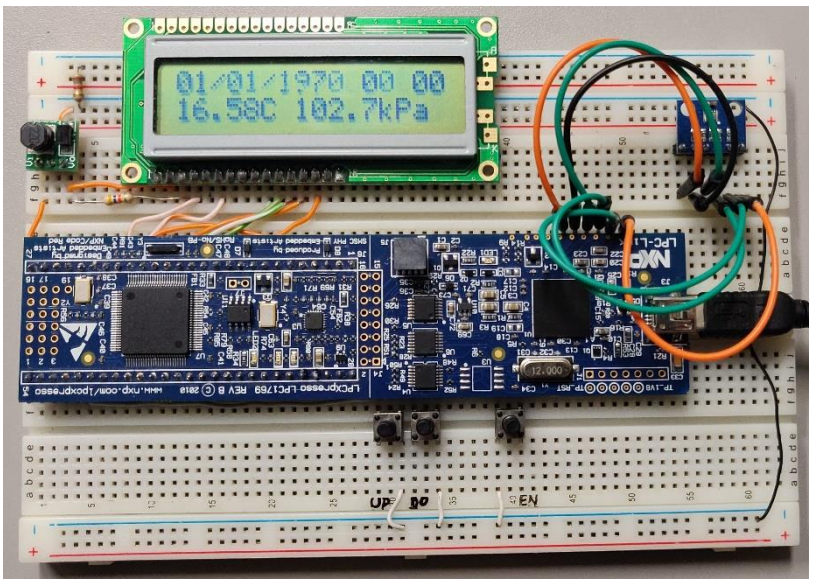
ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Projeto – Estação Meteorológica

40619 André Dias
38866 Manuel Dias
43924 Ricardo Romano



Docente: Pedro Sampaio

Relatório do Projeto Estação Meteorológica realizado no âmbito da Unidade Curricular Sistemas de Embebidos I, do curso de licenciatura em Engenharia Informática e de Computadores Semestre de Inverno 2019/2020

Janeiro de 2020

Resumo

O presente documento foi desenvolvido no âmbito da Unidade Curricular Sistemas Embebidos I de Licenciatura do Curso de Engenharia Informática de Computadores. Trata-se de um relatório do Projeto que visa aplicar a matéria adquirida nas aulas sobre desenvolvimento de aplicações para sistemas embebidos, desenvolver aplicações usando uma arquitetura abstrata e por camadas e desenvolver software de sistema de interface com o hardware. Para tal, o projeto foi contextualizado no desenvolvimento de um sistema que implementa uma estação meteorológica.

Durante o seu desenvolvimento, foi necessário recorrer a matéria adquirida durante as aulas teóricas para delinear uma arquitetura abstrata. Esta tem como objetivo produzir software que permite simplificar o processo de manutenção da aplicação. Também foi necessário recorrer a software produzido durante as aulas práticas para ter acesso às funcionalidades dos periféricos necessários para desenvolver a aplicação.

No final, a produção da aplicação foi executada com sucesso, permitindo a visualização da temperatura ambiente, pressão atmosférica, calendário e relógio e também fazer a sua manutenção. Também foram adicionadas funcionalidades extras ao sistema que permitem uma interação mais intuitiva com o utilizador, como um aviso através de um LED.

Lista de Símbolos

GPIO: General Purpose Input/Ouput

RTC: Real Time Clock

SPI: Serial Peripheral Interface

Índice

Lista de Figuras

1.Introdução

No âmbito da disciplina Sistemas de Embebidos I, foi proposto fazer um projeto que pretende:

- Desenhar estruturas de hardware baseadas em microcontroladores;
- Desenvolver software de sistema de interface com o hardware;
- Desenvolver software de aplicação para sistemas embebidos;
- Depurar o hardware e o software realizado;

Este projeto tem como objetivo realizar um sistema autónomo que implementa uma estação meteorológica para monitorização dos valores da temperatura ambiente e da pressão atmosférica.

Para tal, durante o semestre, foram realizados trabalhos laboratoriais que introduziam todos os periféricos e interfaces essenciais para o desenvolvimento do projeto.

Estes periféricos e interfaces introduzidos são:

- Systick Timer;
- Pinos GPIO;
- Timers;
- RTC;
- SPI;
- Memória Flash;

1.1 Organização do documento

O restante relatório é constituído por quatro capítulos.

No primeiro capitulo é contextualizado o problema em questão, no qual surgiu a necessidade de desenvolver este projeto e é exposto uma arquitetura cuja a solução final terá que cumprir.

No segundo capitulo é relatado todos os componentes e dispositivos utilizados para executar este projeto e a solução final a nível de hardware.

No terceiro capitulo é exposto todo o software desenvolvido para a concretização da aplicação.

No quarto e último capitulo são feitas as conclusões do projeto.

2.Formulação do Problema

Com este projeto, pretende-se criar um sistema autónomo a partir de um microcontrolador que utilize diferentes periféricos e uma aplicação. Este sistema deve ter como metas a reutilização de código, facilidade de manutenção, eficiência e todos os periféricos e interfaces serem independentes da aplicação.
Na secção 2.1, é exposto a contextualização do desenvolvimento deste sistema. Na secção 2.2, é exposto a arquitetura do sistema e a sua composição.

2.1 Contextualização

Este projeto contextualiza-se na necessidade de desenvolver um sistema autónomo que implementa uma estação meteorológica para monitorização dos valores da temperatura ambiente e da pressão atmosférica.

Este sistema é composto por vários dispositivos, como um microcontrolador, botões, um monitor LCD e um sensor de temperatura e humidade. A partir destes, o utilizador pode interagir com o sistema a partir dos botões (**U**, **D** e **E**) e do monitor. Para o funcionamento do sistema, é necessário desenvolver uma aplicação que assenta sobre as necessidades do sistema.

Esta aplicação tem que ser automática, sendo executada após a ligação da energia elétrica, e contém dois modos de funcionamento distintos, o modo normal e o modo de manutenção.

No modo normal, o sistema mostra ao utilizador a partir do monitor LCD a informação relativa ao calendário, relógio, temperatura ambiente e pressão atmosférica. Este modo é considerado o modo automático, ou seja, quando o sistema é eletricamente alimentado entra primeiro neste modo.

O modo de manutenção permite que o utilizador defina a unidade em que a temperatura é apresentada, bem como alterar o calendário e o relógio. Para fazer a passagem do modo normal para o modo de manutenção, é preciso manter os botões **U** e **D** pressionados simultaneamente pelo menos durante dois segundos. Neste modo de funcionamento os botões **U** e **D** servem para navegar nos menus e o botão **E** para aceder ao menu selecionado. Durante o acerto do calendário e do relógio, os botões **U** e **D**, quando pressionados, promovem o incremento ou o decremento dos valores dos campos do calendário (ano, mês e dia) e dos campos do relógio

(horas e minutos). O botão **E**, quando pressionado, promove a mudança do campo a acertar, confirmando o seu valor, e no último campo realiza o retorno ao menu.

Durante a escolha da unidade de temperatura a apresentar os botões **U** e **D**, quando pressionados, promovem a seleção da unidade (Celsius ou Fahrenheit). O botão **E**, quando pressionado, confirma a seleção e realiza o retorno ao menu.

O sistema guarda na memória Flash interna do microcontrolador a unidade em que a temperatura é apresentada.

2.2 Arquitetura

O sistema a desenvolver, cujo diagrama de blocos é apresentado na Figura 1, será implementado tendo como base a placa de desenvolvimento LPCXpresso LPC1769 da NXP que inclui um microcontrolador LPC1769 e a placa BMP280 que contém o sensor BMP280. O sistema disponibilizará interface local para o utilizador três botões de pressão **U**, **D** e **E** (6 mm Tact Switches) e um monitor LCD MC1602C, baseado no controlador HD44780.

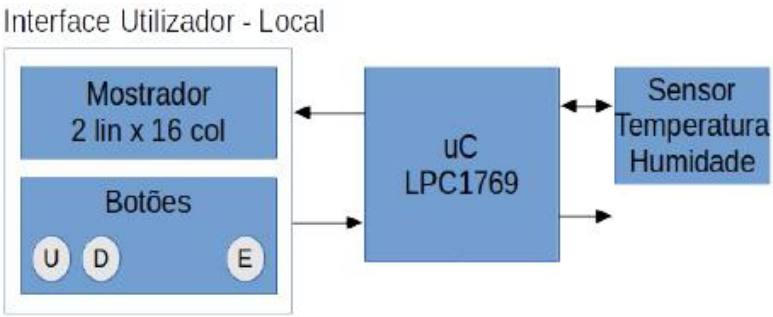


Figura 1 - Diagrama de blocos do sistema a desenvolver

3. Hardware

Para ir ao encontro das metas e objetivos propostos, foram disponibilizados dispositivos para implementar o sistema em questão. Os dispositivos utilizados foram:

- Breadboard, cabos e resistências;
- Placa de desenvolvimento LPCXpresso LPC1769 da NXP;
- Botões de pressão;
- Monitor LCD MC1602C;
- Placa com o sensor BMP280;

Neste capitulo é apresentado todos os dispositivos utilizados, toda a informação essencial, a responsabilidade de cada componente no sistema e o esquema elétrico final.

3.1 Breadboard, Cabos e Resistências

Para montar o sistema totalmente, foi necessário obter certos objetos que permitem que os restantes dispositivos fiquem montados, que garanta a estabilidade física dos componentes e as ligações entre eles. Para tal, foi recorrido o uso de duas **Breadboards** e **Cabos**. É possível averiguar a sua utilidade na Figura 2. No entanto, há certos dispositivos que foi necessário recorrer ao uso de **Resistências**.

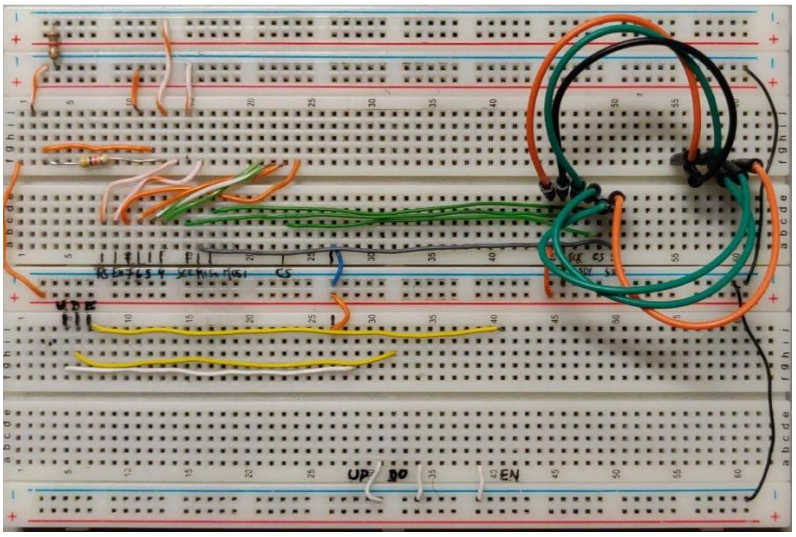


Figura 2 - As duas Breadboards, fios e resistências montadas

Foi necessário ter alguns cuidados ao montar os componentes na Breadboard devido à sua construção interna. Na Figura 3, é possível verificar pelas linhas verdes a conexão interna da

Breadboard. É preciso de ter isto em conta para evitar curtos circuitos que pode provocar danos materiais irreversíveis.

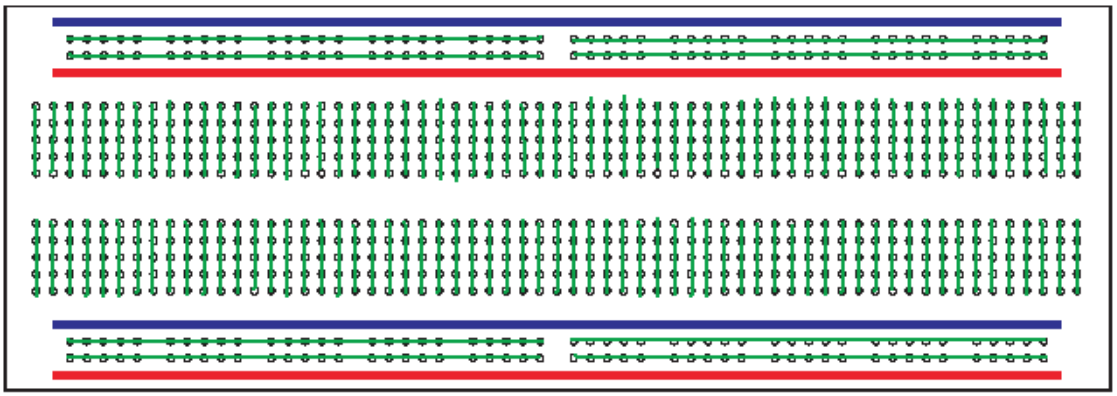


Figura 3 - Circuito interno de uma Breadboard [1]

3.2 Placa de desenvolvimento LPCXpresso LPC1769

Este sistema, tendo como objetivo ser autónomo, foi necessário utilizar um microcontrolador. O microcontrolador escolhido foi o **LPC1769**.

Este microcontrolador é um **Cortex-M3** desenhado para aplicações embebidas oferecendo um nível alto de integração e baixo consumo a frequências de trabalho de 120MHz. É constituído por 512kB de memória Flash, 64kB de memória de dados, USB Device/Host/OTG, três canais de SPI, quatro Timers, um Real-Time Clock com um consumo muito baixo, consegue oferecer até 70 pinos GPIO entre outros periféricos [2].

Para utilizar este microcontrolador, foi recorrido ao uso de uma placa de desenvolvimento já previamente feita, a **LPCXpresso LPC1769** [3], apresentada na Figura 4. Esta placa dispõe de uma interface de depuração, **LPC-Link**.

Figura 4 - Ilustração da placa de desenvolvimento LPCXpresso LPC1769 [3]

Esta placa, consegue cumprir todos os requerimentos necessários para implementar o sistema, pois oferece uma variedade de interfaces e comunicações para vários periféricos e também é

possível programar o microcontrolador e depurar a aplicação desenvolvida facilmente. Também, esta é responsável por alimentar o resto do sistema, fornecendo uma tensão de 3,3V.

No entanto, foi necessário ter certos cuidados ao montar esta placa na Breadboard pela sua conexão interna, explicado na secção 3.1.

3.3 Botões de Pressão

Como referido anteriormente, este sistema requer uma interface que interaja com um utilizador. Esta interface é composta por um input do utilizador e outra por um output do sistema. Como input do utilizador, foi recorrido a três botões de pressão PTS645SH952LFS [4], demonstrado na Figura 5.

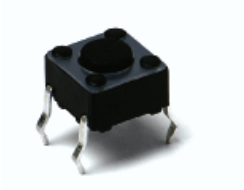


Figura 5 - Botão de pressão PTS645 [4]

Para utilizar os botões, é preciso verificar o seu esquema elétrico. A Figura 6, demonstra o esquema elétrico do botão de pressão e é possível verificar que os pinos 1, 2 e 3, 4 estão constantemente ligados. Só quando se carrega no botão, todos os pinos ficam ligados todos entre si.

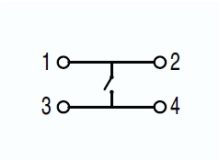


Figura 6 - Esquema elétrico do botão de pressão [4]

Devido à sua constituição, é preciso tomar certas precauções para montar na Breadboard e ligar à placa de desenvolvimento. Como não ligar os botões diretamente na mesma zona que o microcontrolador, mas sim montar numa zona diferente e utilizar um cabo para fazer a ligação entre o botão e a placa e também entre o botão e o GND, como é demonstrado na Figura 7. Na Figura 7, também é demonstrado através de duas linhas azuis do lado esquerdo paralelas entre si

a maneira a evitar de montar, pois esta pode causar um curto circuito entre dois pinos do microcontrolador.

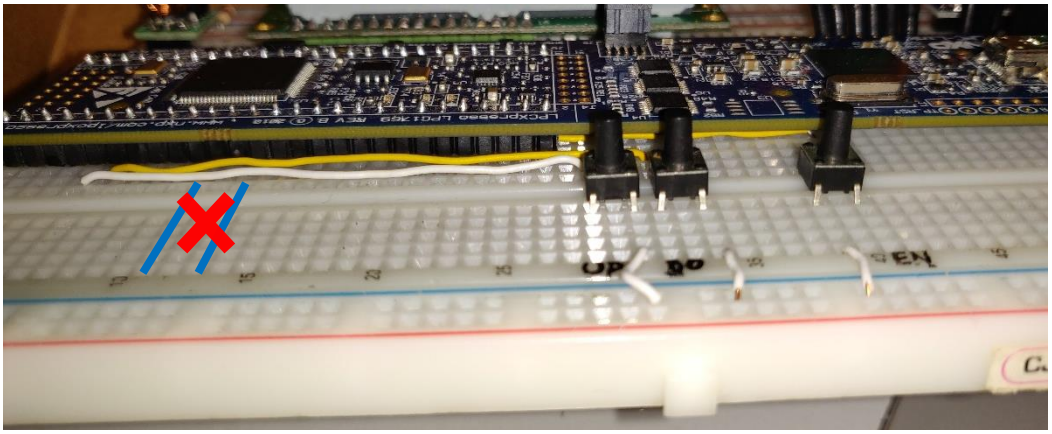


Figura 7 - Montagem dos botões de pressão

3.4 Monitor LCD MC1602C

Como mencionado na secção 3.3, o sistema reque uma interface que interaja com um utilizador e que esta interface é composta por um input do utilizador e outra por um output do sistema. Nesta secção é relatado que dispositivo o sistema utiliza para dar output ao utilizador, que é o Monitor LCD MC1602C, representado na Figura 8.

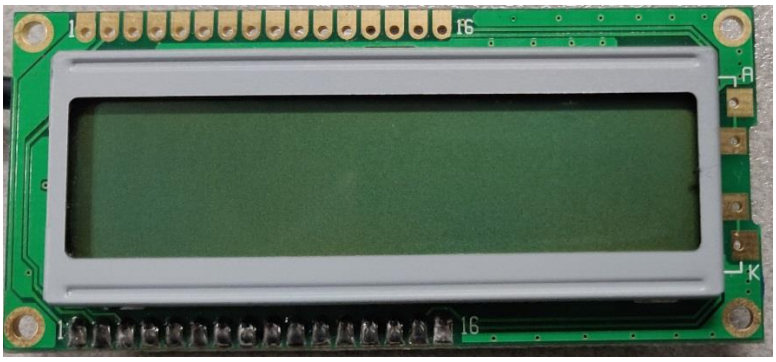


Figura 8 - Monitor LCD MC1602C

Na Figura 8 é possível verificar que os pinos estão numerados de 1 a 16. Para saber qual é a função de cada pino, foi consultado a ficha técnica do monitor. Nesta, é exposta uma tabela que demonstra os símbolos e as funções a que corresponde cada pino. Esta tabela também se encontra disponível na Figura 9.

No.	Symbol	Level	Function
1	VSS	–	Power Supply (0V, GND)
2	VDD	–	Power Supply for Logic
3	VEE (Vo)	–	Power Supply for LCD Drive
4	RS	H / L	Register Select Signal
5	R/W	H / L	Read/Write Select Signal H : Read L : Write
6	E	H / L	Enable Signal (No pull-up Resister)
7	DB0	H / L	Data Bus Line / Non-connection at 4-bit operation
8	DB1	H / L	Data Bus Line / Non-connection at 4-bit operation
9	DB2	H / L	Data Bus Line / Non-connection at 4-bit operation
10	DB3	H / L	Data Bus Line / Non-connection at 4-bit operation
11	DB4	H / L	Data Bus Lin
12	DB5	H / L	Data Bus Line
13	DB6	H / L	Data Bus Line
14	DB7	H / L	Data Bus Line
15	LEDA	--	No Connection
16	LEDK	--	No Connection

Figura 9 - Atribuição de pinos do monitor

Como é possível verificar na Figura 9, o monitor tem vários pinos com funções diferentes.
Os pinos 1 a 3 são pinos de alimentação. Estes são responsáveis por alimentar toda a parte lógica do monitor e o controlador do LCD.
Os pinos 4 a 6 são pinos de controlo. Estes são responsáveis por controlar a escrita, a leitura do monitor e a configuração do monitor. Por convenção definida pelo docente, foi proposto que neste projeto só seria feito escritas no monitor.
Os pinos 7 a 14 são pinos de dados. Estes são responsáveis por receber dados de um dispositivo externo e/ou enviar. O monitor também é capaz de fazer operações de 4 bits usando apenas os pinos 11 a 14. Por convenção, foi proposto que o monitor faça operações apenas a 4 bits.
Os pinos 15 e 16 são pinos de alimentação para um LED com o propósito de iluminar o monitor para ambientes com pouca iluminação.

Para o monitor funcionar, é preciso alimentá-lo através dos seus pinos de alimentação. Na Figura 10 é apresentado um exemplo fornecido pelo fabricante de como se deve alimentar o monitor. Como é possível verificar, o VEE(Vo) é alimentado a partir de um divisor de tensão de duas resistências, R1 e R2. Apesar de que na figura é representado um potenciómetro, foi convencionado de que não se ia usar. O fabricante sugere também que a soma das resistências seja entre 10 Ω e 20KΩ. As resistências usadas são:

- R1=4,7KΩ
- R2=1,8KΩ

O que significa que temos no total a soma das resistências é 6,5KΩ, desprezando a resistência dos cabos e da Breadboard.

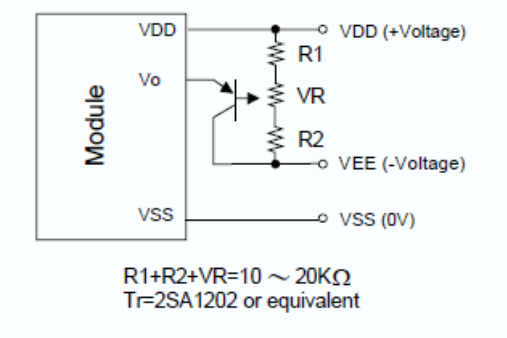


Figura 10 -Exemplo da alimentação do monitor

No entanto, ao verificar a característica elétrica do pino de alimentação VDD, demonstrado na Figura 11, notou-se que a placa LPC não tem tensão suficiente para alimentar este monitor.

Ta = 25°C, V_{SS} = 0V

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Units
Supply Voltage (Logic)	V_{DD} - V_{SS}	--	4.5	--	5.5	V
Supply Voltage (LCD Drive)	V_{DD} - V_O	Shown in 3.1				V
High Level (Input Voltage)	V_{IH}	V_{DD} = 5.0V	2.2	--	V_{DD}	V
Low Level (Input Voltage 0	V_{IL}	V_{DD} = 5.0V	-0.3	--	0.6	V

Figura 11 - Característica Elétrica dos pinos

Por isso, foi necessário recorrer a um step-up, que faz aumentar os 3,3V de alimentação da placa para 5V.

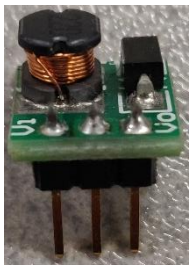


Figura 12 - Step-up

3.5 Sensor BMP280

Para obter dados sobre a temperatura ambiente e a pressão atmosférica, foi recorrido ao uso de um sensor de temperatura e humidade, o BMP280, demonstrado na Figura 13.

Figura 13 - Sensor de temperatura e humidade BMP280

Este sensor tem a capacidade de comunicar com outros módulos a partir dos protocolos I2C ou SPI. Foi definido pelo docente que para este projeto, o protocolo utilizado para comunicar com este dispositivo será o SPI.

A nível do protocolo de SPI, este dispositivo é compatível com o modo três e quatro fios, trabalhando a uma frequência máxima de 10 MHz.

A partir da ficha técnica fornecida pelo fabricante, pode ser verificado que a tensão de alimentação deste dispositivo é compreendida entre 1,71V e 3,6V, demonstrado também na Figura 14, o que significa que este pode ser alimentado diretamente a partir da placa LPC sem recorrer ao uso de step-ups ou reguladores de tensão.

Sensor supply voltage	V_{DD}	ripple max. 50mVpp	1.71	1.8	3.6	V
Interface supply voltage	V_{DDIO}		1.2	1.8	3.6	V
Supply current	$I_{DD,LP}$	1 Hz forced mode, pressure and temperature, lowest power		2.8	4.2	µA
Peak current	I_{peak}	during pressure measurement		720	1120	µA
Current at temperature measurement	I_{DDT}			325		µA
Sleep current[1]	I_{DDSL}	25 °C		0.1	0.3	µA
Standby current (inactive period of normal mode) [2]	I_{DDSB}	25 °C		0.2	0.5	µA
Relative accuracy pressure	A_{rel}	700 ... 900hPa		±0.12		hPa
V_{DD} = 3.3V		25 . . . 40 °C		±1.0		m

Figura 14 - Características elétricas da alimentação do sensor BMP280

3.6 Esquema Elétrico do Sistema

Após apresentar todos os dispositivos e componentes compreendidos no sistema, foi construído um esquema elétrico, demonstrado na Figura 15. A partir desta, é possível fazer todas as ligações do sistema.

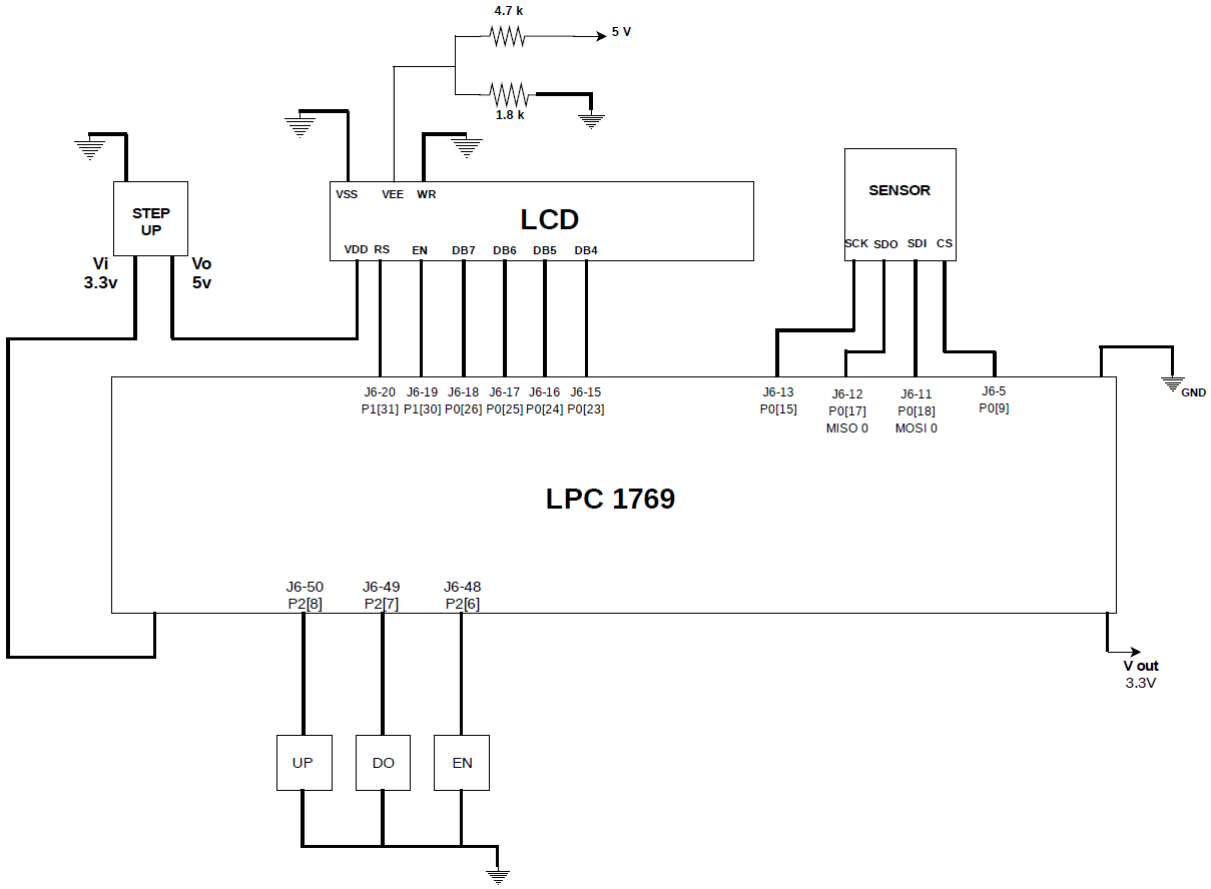


Figura 15 - Esquema Elétrico do Sistema

4. Middleware e Software

Para ir ao encontro das metas e objetivos propostos, foi necessário reunir todos os elementos para se conseguir desenvolver a aplicação desejada. Para tal, foi preciso dividir a aplicação por camadas, onde cada uma destas é independente das posteriores e anteriores, permitindo modificar os elementos que constitui cada camada com mais facilidade.

Neste capítulo é relatado todo o código realizado em prol de desenvolver a aplicação, incluindo também algum código produzido durante as aulas laboratoriais.

Na secção 4.1 é relatado como foi abordado a aplicação no modo geral, revelando a sua arquitetura e a sua composição. Na secção 4.2 é retratado o **Middleware** que demonstra o código produzido nas aulas. Na secção 4.3 é esclarecido o desenvolvimento da aplicação, retratando todo o **Software** desenvolvido para a implementação da aplicação.

4.1 Arquitetura da Aplicação

Como mencionando anteriormente, foram reunidos todos os elementos a desenvolver para implementar uma aplicação que vai assentar no sistema desenvolvido. Esta aplicação é composta por três partes principais, o **Software**, o **Middleware** e o **Hardware**.

O **Hardware** já foi exposto no capitulo anterior todos os seus componentes e como todos eles se interligam. O **Software** é a camada responsável pela a aplicação a desenvolver. Esta é responsável pela lógica de negócio, neste caso, toda a lógica mencionada na contextualização do projeto. O **Middleware** é a camada ponte que interliga todas as necessidades do **Software** e do **Hardware**.

Na Figura 16, estão representados o layout da aplicação por camadas e a constituição das camadas principais em subcamadas.

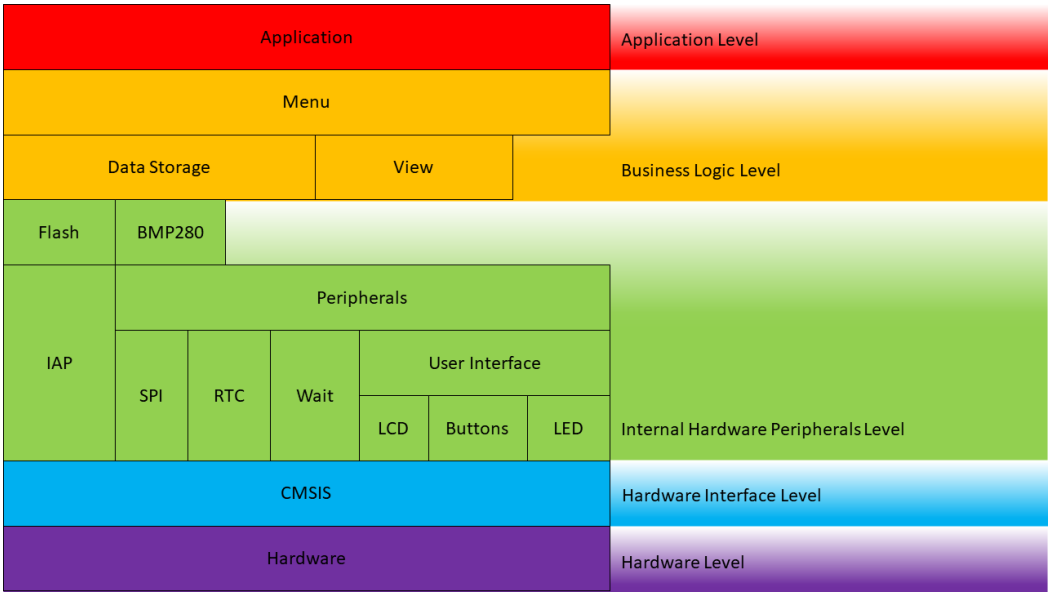


Figura 16 - Layout da aplicação por camadas

A camada **Software** está representada pelas duas primeiras subcamadas da Figura 16 que estão pintadas a vermelho e a laranja. A camada vermelha é o ponto de acesso à aplicação que o microcontrolador vai aceder, que depois permite aceder à camada a seguir, a laranja, que representa toda a lógica do projeto mencionada na contextualização.

A camada **Middleware** está representado pelas duas subcamadas pintadas a verde e azul. A subcamada a verde é constituída por todos os dispositivos que necessitam de uma entidade que interpreta os seus dados ou que interaja com esse elemento. Os dados para serem processados por essas entidades são recebidos ou enviados a partir da subcamada a azul que também tem como responsabilidade interpretar e codificar as instruções em código máquina. Este código máquina depende sempre do **Hardware** escolhido.

A camada **Hardware** está representado pela última subcamada pintada a roxo. Esta camada é constituída pelo microcontrolador escolhido.

4.2 Middleware

Como mencionado anteriormente, esta camada é constituída por duas subcamadas. A primeira subcamada é constituída por elementos que foram produzidos durante as aulas laboratoriais e a segunda é uma camada abstrata desenvolvida pela *Arm Developer*. Como esta última foi obtida gratuitamente, não vai ser relatado a sua constituição.

A produção de todas as entidades e API's nas aulas laboratoriais foram documentadas usando a ferramenta *Doxygen*. Esta documentação explicita como se deve interagir com cada API e o seu funcionamento.

4.3 Software – A Aplicação

Esta camada é constituída por quatro entidades principais, a **Aplicação**, o **Menu**, o **Armazém de Dados**, e a **Vista**. A partir destas, é executada toda a lógica necessária para cumprir todas as necessidades requeridas na contextualização do projeto.

A primeira entidade, a **Aplicação**, tem como objetivo o ponto de partida da aplicação. Esta contém o método *main()* que tem como responsabilidade executar a funcionalidade principal da aplicação.

A entidade **Menu**, é responsável por oferecer e executar os menus, ou os modos de funcionamento, da aplicação e aplicar a lógica do projeto. Esta é composta por três componentes com objetivos diferentes, o **Menu**, o **Menu Normal** e o **Menu de Manutenção**. A componente **Menu** é responsável pela inicialização de todas as interfaces, variáveis e periféricos da aplicação. O **Menu Normal** visa implementar todas as funcionalidades do **Modo Normal**

mencionado na contextualização do projeto. O **Menu de Manutenção** tem como objetivo aplicar todas as necessidades do **Modo de Manutenção** referido na contextualização do projeto.
A entidade **Armazém de Dados**, é responsável por armazenar e processar todos os dados importantes da aplicação, neste caso, constituído pelo calendário, relógio, temperatura e pressão. Também é responsável por verificar a coerência dos dados, apresentando apenas valores válidos para cada campo.
A entidade **Vista**, é responsável por oferecer um output ao utilizador dos dados presentes, dependendo do modo corrente.

Com isto, é possível demonstrar e explicitar o algoritmo de implementação da aplicação.

4.3.1 Algoritmo da Aplicação - Geral

Durante a execução da aplicação, pode ser verificado que esta é constituída por três possíveis estados, a **Inicialização**, **Normal** e **Manutenção**. Na Figura 17, é representado e descrito o comportamento destes estados, de um modo geral.

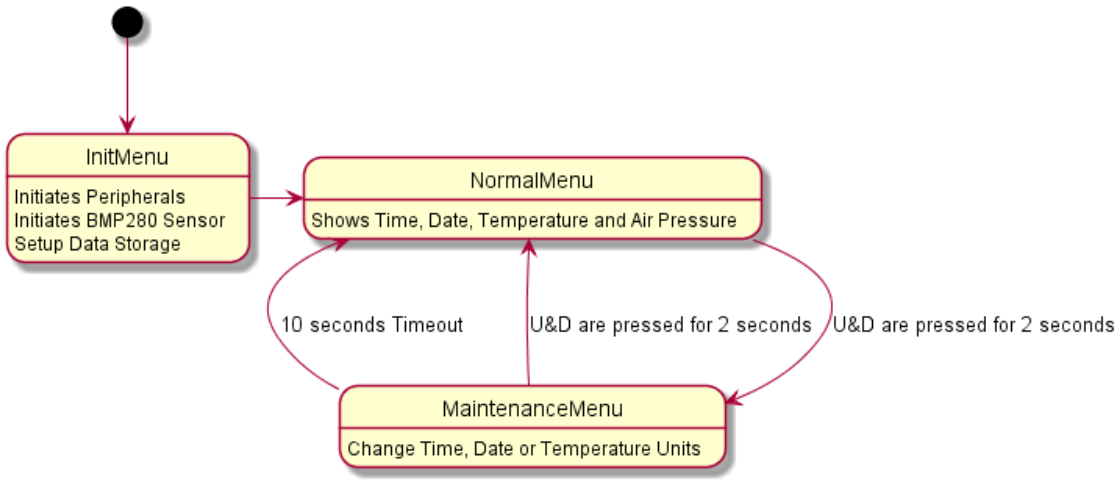


Figura 17 - Estados possíveis da Aplicação

No estado de **Inicialização**, a aplicação é inicializada e é chamado todas as funções de inicialização da aplicação, preparando todas as variáveis, registos e periféricos para um estado definido, como por exemplo, ler da Flash a unidade de medida dos valores da temperatura ou ler os valores de calibração do sensor de temperatura presente no sistema. Com isto, é possível criar uma independência do código com o **Hardware**, desde que este apresente funções e registos semelhantes ao anterior. Este estado só é executado apenas no inicio da aplicação, ou seja, mal o

microcontrolador é alimentado. Para executar novamente este estado é preciso fazer uma reinicialização do sistema, como por exemplo, deixar de alimentar o microcontrolador e voltar a alimentá-lo. Na Figura 18, é possível verificar todos os estados de inicialização que a aplicação passa.

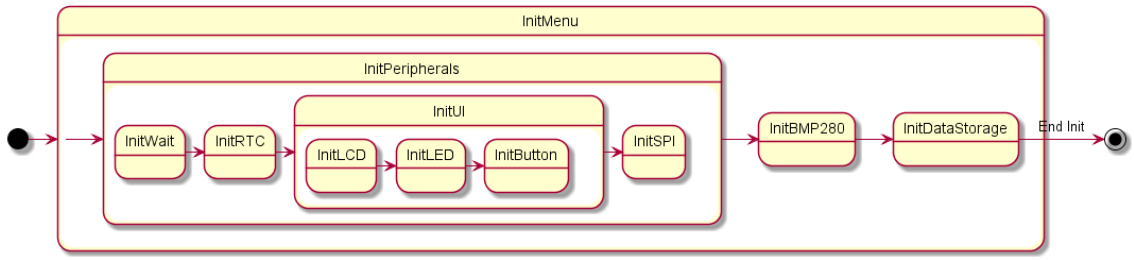


Figura 18 - Algoritmo do estado de Inicialização

No estado **Normal**, a aplicação implementa as funcionalidades do **Menu Normal** que foram anunciadas previamente. Na Figura 19, é demonstrado o comportamento interno deste estado.

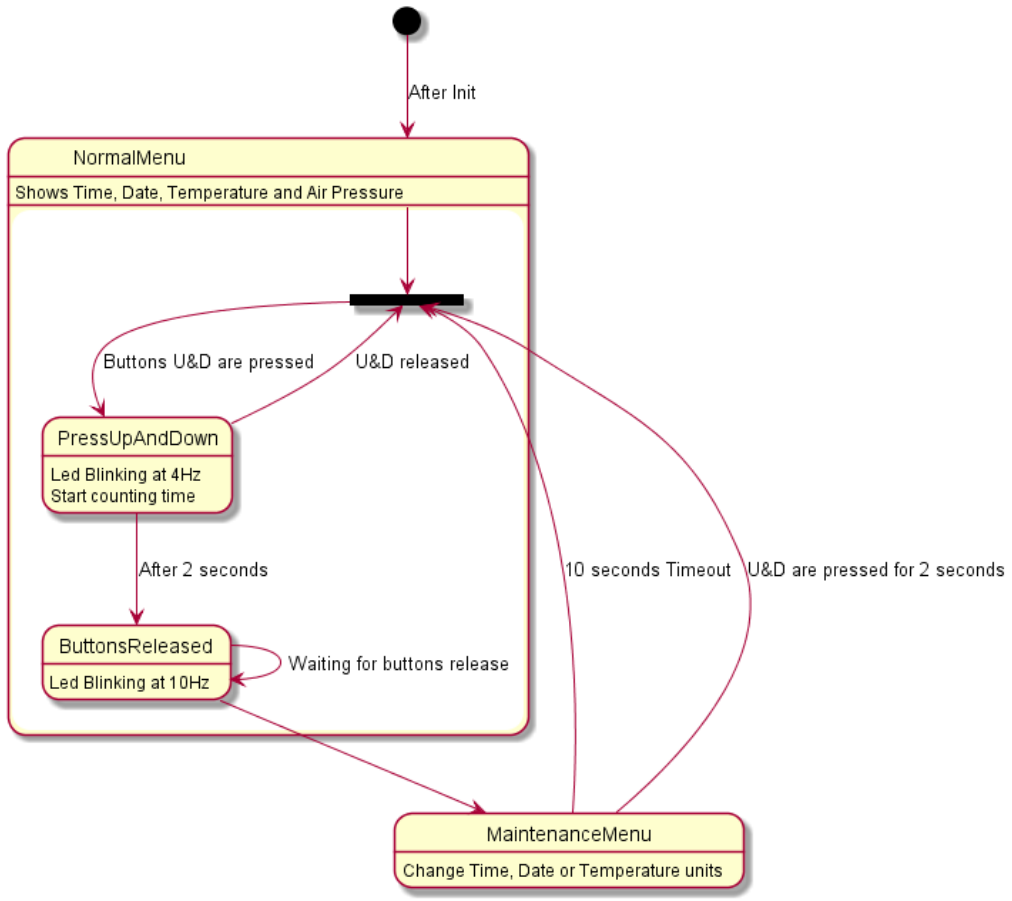


Figura 19 - Algoritmo do estado Normal

No estado de **Manutenção**, implementa todas as funcionalidades do **Modo de Manutenção**. A Figura 20 demonstra o comportamento interno deste estado.

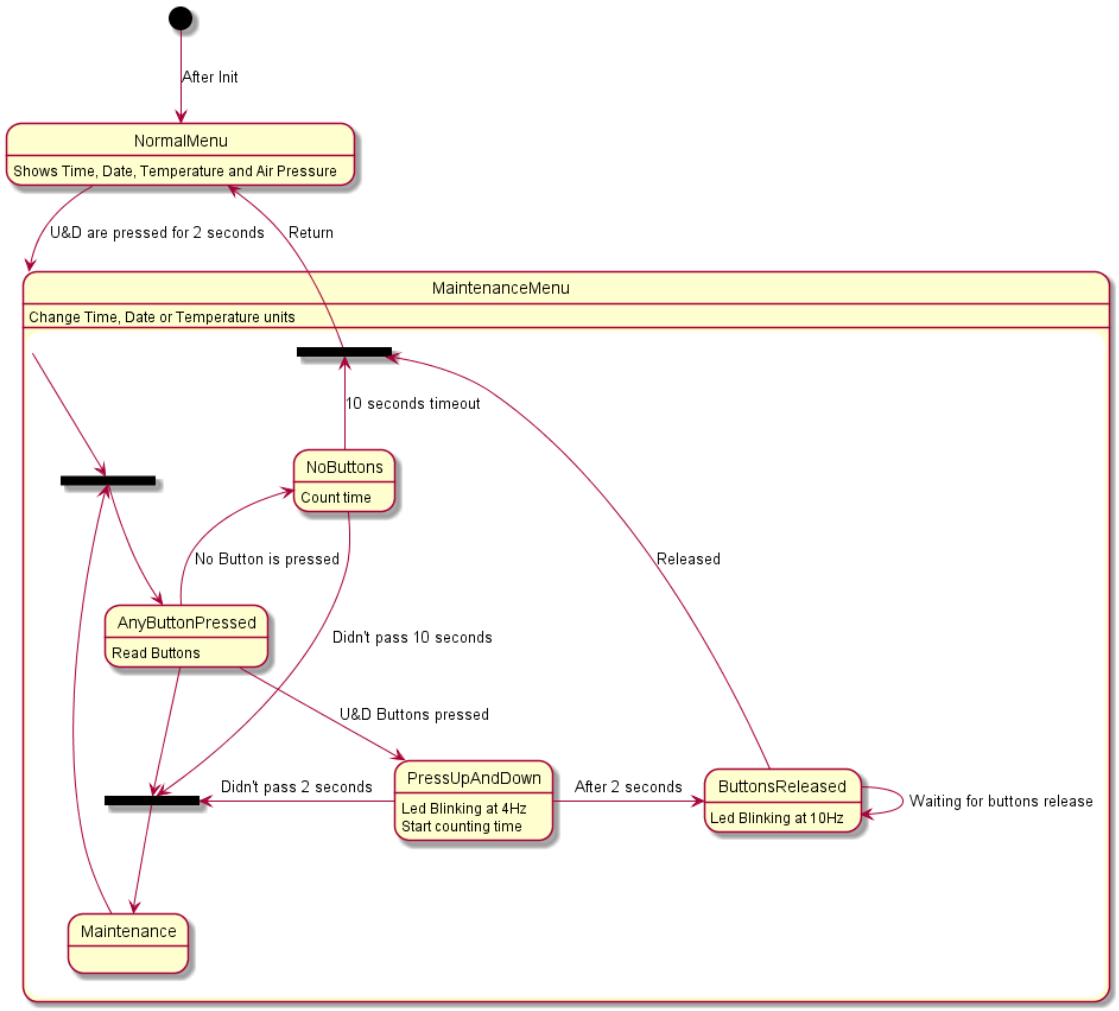


Figura 20 - Algoritmo do estado de Manutenção

4.3.2 Algoritmo da Aplicação – Mudança de Estados

Para a aplicação conseguir mudar entre os vários estados, surgiram duas soluções possíveis. A primeira solução consiste em ter uma variável que represente um estado e a partir de uma instrução de condição, a aplicação ou muda de estado ou permanece no mesmo. A segunda solução consiste em dar a cada estado a responsabilidade de devolver um endereço de uma função que representa o estado seguinte, sendo o próprio ou o próximo estado.

As vantagens da primeira são a simplicidade da solução e a facilidade de interpretação para um programador que nunca tenha visto o código. Também, como neste caso só existem dois

estados, através da arquitetura do microcontrolador, das possíveis instruções *Assembly* e quantidade de *pipelines*, esta solução consegue ser otimizada a partir do compilador. As vantagens da segunda solução são a eficiência a nível de tempo de execução e ocupação de memória e a independência que a camada responsável pela lógica tem perante a aplicação que, consequentemente, permite adicionar com mais facilidade mais estados sem alterar o código da aplicação.

Apesar das vantagens da primeira solução, foi adotada a segunda, pois esta oferece um nível de abstração maior, permitindo adicionar mais estados no futuro. Também permite evitar alguns erros de programação de distração que a primeira solução tem, como por exemplo um estado devolver um valor que não consiste na condição de mudança de estado, enquanto que esta, basta passar a desreferenciação da função com a mesma assinatura.

4.3.3 Algoritmo da Aplicação – Menu de Manutenção

Durante a implementação do **Menu de Manutenção**, foi reparado que existem vários caminhos que utilizam os mesmos procedimentos, mas tinham finalidades diferentes. Por isso, dentro deste menu existem dois modos possíveis, o modo de **Escolha** e o de **Manutenção**.
No primeiro modo, o utilizador escolhe que tipo de manutenção quer fazer, alterar calendário, relógio ou as unidades da temperatura. Quando o utilizador carregar no botão **E**, passa para o modo de **Manutenção**. Neste modo o utilizador modifica os dados daquele campo, no entanto este modo comunica com a entidade **Armazém de Dados** para incrementar ou decrementar os campos correntes.
Isto é possível recorrendo a ponteiros de funções que indicam o que é que a aplicação tem de executar, dependendo do modo corrente.

4.3.4 Algoritmo da Aplicação – Funcionalidades Extras

Na planificação da aplicação, foi decidido adicionar certas funcionalidades à aplicação que visam facilitar a interação do utilizador com o sistema. Estas são:

- Adicionar um LED que representa o estado corrente da aplicação (LED ligado = **Estado Manutenção**, LED desligado = **Estado Normal**);
- Quando o utilizador carregar nos botões **U** e **D**, o LED piscar a uma frequência de 4 Hz e, quando passarem dois segundos, piscar a uma frequência de 10Hz;

- No **Estado Manutenção**, se o utilizador não interagir com o sistema durante 10 segundos seguidos, este retorna para o **Estado Normal**. Se a aplicação estiver a meio de uma alteração e passar os 10 segundos, as alterações não são gravadas;

6. Conclusões

Neste projeto, a implementação de uma aplicação que assenta num sistema, desenvolvendo uma estação meteorológica, foi executada com sucesso.

Foi desenvolvido uma arquitetura abstrata e simples o suficiente para, no caso de se querer alterar uma das camadas intermédias, o código a alterar seja mínimo.

No processo do desenvolvimento da aplicação, surgiram dúvidas que, com a ajuda do docente, ajudaram a complementar conhecimentos satisfazer as necessidades do enunciado do trabalho.

Bibliografia

[1] S. Anwar, “researchgate,” 1 Outubro 2008. [Online]. Available: https://www.researchgate.net/figure/Breadboard-Internal-Connections_fig3_265932780. [Acesso em janeiro 2020].

[2] “NXP.com,” NXP, [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/lpc1700-cortex-m3/512kb-flash-64kb-sram-ethernet-usb-lqfp100-package:LPC1769FBD100. [Acesso em Janeiro 2020].

[3] “embeddedartists.com,” Embedded Artists, [Online]. Available: https://www.embeddedartists.com/products/lpc1769-lpcxpresso/. [Acesso em Janeiro 2020].

[4] “dznh3ojzb2azq.cloudfront.net,” [Online]. Available: https://dznh3ojzb2azq.cloudfront.net/products/Tactile/PTS645/documents/datasheet.pdf. [Acesso em Janeiro 2020].