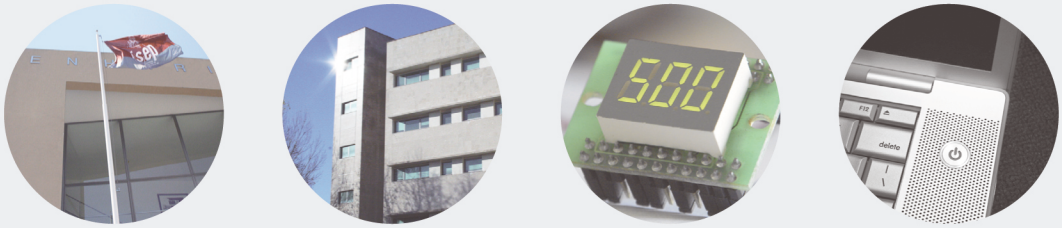
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema HMI 7"

TIAGO MANUEL DA SILVA CAVADAS
julho de 2019

POLITÉCNICO
DO PORTO

Sistema HMI 7”

Tiago Manuel da Silva Cavadas

Tese/Dissertação

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2019

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação,do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores - Automação e Sistemas

Aluno: Tiago Manuel da Silva Cavadas, N 1100497, 1100497@isep.ipp.pt

Orientação Científica: Professor Manuel Gericota, mgg@isep.ipp.pt

Empresa: Ricardo Malheiro, Lda

Supervisão: Eng. Ricardo Malheiro, ricardomalheiro@rmtech.pt

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

17 de Julho de 2019

Agradecimentos

Ao longo do meu percurso académico, a minha trajetória incluiu desafios, tristezas, incertezas, alegrias e muitos percalços que fizeram de mim a pessoa que sou hoje. Desde já um muito obrigado a todas as pessoas, cada uma à sua maneira, que me acompanharam e contribuíram para ultrapassar todas as adversidades.

Em primeiro lugar, gostaria de agradecer ao professor Manuel Gericota, não só pela motivação, paciência e ajuda demonstrada no desenvolvimento de toda a tese, mas também, pelos seus ensinamentos nas aulas da unidade curricular PHADI, no qual tive a possibilidade de o conhecer, e perceber que seria um excelente orientador.

Ao engenheiro Ricardo Malheiro, deixo também meu agradecimento, por toda a confiança, disponibilidade e ajuda. Posso afirmar que todos os dias aprendo eletrónica e sem dúvida que a minha vida, profissional e pessoal, não seria a mesma sem o seu conhecimento.

Quero também agradecer aos meus pais e irmãs, pelo esforço, preocupação, e dedicação em todo o meu percurso académico, e pela presença no momentos, bons e menos bons.

A todos os amigos, Orlando, Eunice, Pedro, Ana, Vítor, João, Esteves e Ivone, que estiveram presentes desde o primeiro ano do ISEP, deixo também um especial agradecimento.

Por fim, à Luísa, que sempre me estimula a crescer científica e pessoalmente, pelas inúmeras trocas de impressões, correções e comentários ao trabalho. Acima de tudo, pelo inestimável apoio, paciência e compreensão reveladas ao longo destes anos. Por isso muito obrigado.

Esta página foi intencionalmente deixada em branco.

Resumo

A rápida evolução humana e a procura por estratégias facilitadoras do seu dia-a-dia, implicam uma sinergia próxima com a tecnologia que deve adaptar-se às necessidades do seu utilizador. Neste contexto, surgem as interfaces HMI (*Human Machine Interface*) responsáveis pela interligação Homem-Máquina.

A empresa Ricardo Malheiro, Lda, fundada em 2007 e especializada no desenvolvimento, fabrico, comercialização de equipamentos eletrónicos de baixa tensão e assemblagem de cablagem, aceitou o desafio do desenvolvimento de um sistema HMI próprio, adaptado à necessidade de cada um dos seus clientes.

A versatilidade deste sistema permite-lhe ser aplicado a diferentes áreas, como a indústria automóvel e domótica, entre outras. O sistema, modular e *plug & play*, substituí facilmente um sistema já obsoleto, por exemplo, interrutores num painel de um veículo.

Nesta tese desenvolveu-se um sistema HMI que integra um *display* de sete polegadas (1024x600) com tecnologia *touch* resistiva. O sistema contém um armazenamento de 128 Mbytes para imagens e implementa funções de leitura e controlo de luminosidade, tensão de alimentação e temperatura.

Este sistema tem uma interligação com o exterior que usa protocolos de comunicação série através dos quais são controladas todas as funções do sistema.

Palavras-chave

HMI, *touch* resistivo

Esta página foi intencionalmente deixada em branco.

Abstract

The rapid human evolution and the search for strategies that can help to facilitate everyday life imply a close synergy with technology, which must be adapted to the needs of its user. In this context, assume great importance the Human-Machine Interfaces (HMI) responsible for human-machine interconnection.

The company Ricardo Malheiro, Lda, founded in 2007 and specialized in the development, manufacturing, and marketing of low-voltage electronic equipment and cable hardness assembly, accepted the challenge of developing its own HMI system, adapted to the needs of each of its customers.

The versatility of this system allows it to be applied to different areas, such as the automotive industry and home automation, amongst others. The system is modular and plug & play, easily replacing an already obsolete system, for example, based on switches, in a vehicle panel.

Thus, in this thesis, an HMI system was developed that integrates a seven-inch display (1024x600) with resistive touch technology. It contains a 128 Mbytes storage for images and implements functions of reading and control of luminosity, supply voltage, and temperature.

This system has an interconnection with the outside that uses serial communication protocols that are employed to control all system functions.

Keywords

HMI, *touch* resistivo

Esta página foi intencionalmente deixada em branco.

Conteúdo

Lista de Figuras

Esta página foi intencionalmente deixada em branco.

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

List of Acronyms

CAS *Column Access Strobe*

CS *Chip Select*

CSX *Chip Select*

DMA *Direct Memory Access*

DSI *Display Serial Interface*

EIA *Electronic Industries Alliance*

FMC *Flexible Memory Controller*

GUI *Graphical User Interface*

HAL *Hardware Abstraction Layer*

HBP *Horizontal Back Porch*

HFP *Horizontal Front Porch*

HMI *Human Machine Interface*

I2C *Inter-Integrated Circuit*

ISEP *Instituto Superior de Engenharia do Porto*

LCD *Liquid Crystal Display*

LDR *Light Dependent Resistor*

LED *Light Emitting Diode*

LIN *Local Interconnect Network*

LTDC *LCD-TFT Display Controller*

LVDS *Low-voltage Differential Signaling*

OLED *Organic Light-Emitting Diode*

PWM *Pulse Width Modulation*

QSPI *Quad Serial Peripheral Interface*

RD *Read*

RGB *Red, Green, Blue*

RS *Register Select*

R/W *Read/Write*

SCL *Serial Clock*

SDA *Serial Data*

SDI *Serial Data In*

SDO *Serial Data Out*

SDRAM *Synchronous Dynamic Random-Access Memory*

SPI *Serial Peripheral Interface*

SVGA *Super Video Graphics Array*

TEDI *Tese/Dissertação*

TFT *Thin Film Transistor*

TIA *Telecommunications Industry Association*

UART *Universal Asynchronous Receiver-Transmitter*

VBP *Vertical Back Porch*

VFP *Vertical Front Porch*

WR *Write*

XGA *Extended Graphics Array*

Esta página foi intencionalmente deixada em branco.

Capítulo 1

Introdução

1.1 Contexto

Este projeto foi desenvolvido no âmbito da Tese/Dissertação (*TEDI*) do Mestrado em Engenharia Eletrotécnica e de Computadores do Instituto Superior de Engenharia do Porto (*ISEP*), em total colaboração com a empresa Ricardo Malheiro, Lda (RMtech)

A Ricardo Malheiro, Lda é uma empresa que atua no ramo da eletrónica e tem como objetivo satisfazer os seus clientes oferecendo-lhes um produto único de acordo com as suas especificações, acompanhando-o desde o seu desenvolvimento até à sua produção final. A empresa atua ainda no ramo da subcontratação para assemblagem de produtos previamente desenvolvidos.

Todos os projetos de desenvolvimento existentes na empresa são analisados e aceites pela equipa "gestão de projetos", que é responsável por garantir que todos os desenvolvimentos são concluídos dentro das especificações e prazos requeridos.

Nos últimos anos tem-se notado um grande acréscimo de pedidos para uso de *displays* para interação com os seus sistemas, alguns destes já desenvolvidos no passado. Como tal, a equipa de gestão decidiu desenvolver um produto HMI (*Human Machine Interface*) de sete polegadas que fosse

compatível com antigos sistemas e de fácil implementação.

Apesar da existência de uma grande oferta de produtos idênticos ao pretendido, sempre que possível, a Ricardo Malheiro, Lda tem como preferência o desenvolvimento os seus próprios produtos. Desta forma pode dar uma melhor garantia aos seus clientes.

1.2 Objetivos

O objetivo deste trabalho é o desenvolvimento de um sistema HMI de 7" universal, que inclua funções necessárias para diferente aplicações, recolhidas e analisadas previamente pela equipa de gestão.

Os objetivos a ter em consideração são os seguintes:

- *Display* 7" com resolução 1024x600;
- Tecnologia *touch* resistiva;
- Controlo de luminosidade;
- Sensores:
  - Luminosidade;
  - Temperatura;
- Besouro;
- Interface RS485 ou LIN;

1.3 Estrutura do documento

O presente trabalho é composto por sete capítulos cujo conteúdo é discriminado nos seguintes pontos:

- **Introdução** - Contextualização e objetivos;

- **Estado da Arte** - O que é um sistema HMI e qual foi a sua evolução. Quais a tecnologias presentes em *displays* e definição do termo resolução. Interfaces e controladores existentes para comunicação com um *display*. Que tecnologias *touch* estão presentes num *display*;
- **Requisitos e arquitetura do sistema** - Indicação dos requisitos impostos para a realização deste projeto e a sua respetiva arquitetura;
- **Desenvolvimento** - Descrição dos componentes usados neste projeto e respetiva função;
- **Implementação** - Descrição funcional do software, recorrendo essencialmente a fluxogramas, para a realização das funções requisitadas;
- **Resultados** - Apresentação dos testes efetuados ao hardware e software do sistema desenvolvido;
- **Conclusão e trabalho futuro** - Reflexão sobre o trabalho realizado e perspetivas de futuro.

Esta página foi intencionalmente deixada em branco.

Capítulo 2

Estado da Arte

2.1 O que é um sistema HMI

Um sistema HMI é uma interface entre o Homem e a máquina. Uma HMI permite ao utilizador a inserção de dados e a visualização dos resultados no equipamento. Estes resultados podem também ser apresentados graficamente, proporcionando uma interpretação mais rápida ao utilizador.

Os sistemas HMI foram utilizados ao longo dos anos em vários setores, tais como: indústria automóvel, médica, bancária, aeroespacial, entre outros. Atualmente estas tecnologias são tão significativas que já estão incluídas na área da inteligência artificial, por exemplo. [19]

2.2 Evolução dos Sistemas HMI

Os primeiros registos do desenvolvimento de sistemas HMI datam da década de 60, pois até lá as tecnologias eram limitadas e dispendiosas, sendo a introdução de dados nas máquinas um processo entediante e com grande probabilidade de erro. A primeira interface HMI terá sido criada em 1962, sob a forma de um jogo de computador designado Spacewar.

Em meados do século XX, as tecnologias HMI conhecidas correspondiam

aos teclados e ratos de computador. Xerox PARC em 1973 desenvolveu o Alto PC, sendo o primeiro pc com uma interface gráfica (GUI - *graphical user interface*). Em 1980 apareceram novos sistemas HMI como joysticks, tablets, comandos, etc.

No mundo da indústria, os equipamentos existentes em linhas de produção eram, na sua maioria, controlados por botões de pressão e os alertas eram dados através de sinais luminosos. Com o aparecimento dos sistemas HMI, os botões de pressão foram sendo substituídos por outras alternativas. Contudo, muitas empresas optaram por dar continuidade aos tradicionais botões pela sua simplicidade e baixo custo. Em algumas aplicações, o uso destes botões fará mais sentido que um sistema HMI. Se apenas for necessário um botão para ligar e desligar um equipamento, a implementação de um sistema HMI pode implicar um custo mais elevado, assim como a exigência de formação dos operadores.

Com o aparecimento dos computadores, os trabalhos em sistemas HMI aumentaram exponencialmente. Os sistemas com uso de botões físicos foram alterados gradualmente para computadores e o foco mudou de hardware para software. Nesta altura, o desafio era a interface gráfica de fácil usabilidade, pois estes sistemas exigiam uma grande curva de aprendizagem.

Os sistemas foram desenvolvidos inicialmente com muita pouca comunicação entre os engenheiros e o utilizador, o que resultou em sistemas de pouca qualidade e de baixa eficácia. Mais tarde, um grupo de especialistas projetou de novo os sistemas HMI focando os aspetos das motivações humanas, perceção visual e aprendizagem. [19]

2.3 *Display*

Um *display* é um dispositivo de apresentação visual de informação. Existem diferentes tipos de tecnologias associadas aos *displays*, sendo as mais conhecidas designadas por LCD-TFT (*thin film transistor - liquid crystal display*)

e OLED (*organic light-emitting diode*).

Tal como o nome indica, a tecnologia LCD-TFT consiste num conjunto de cristais líquidos que no seu estado normal (sem corrente elétrica) estão posicionados em paralelo com a superfície. Quando lhes é aplicada uma tensão estes mudam de direção ficando perpendiculares à superfície e alterando as suas propriedades óticas. Portanto, dependendo da corrente aplicada aos cristais, é possível controlar os seus movimentos. Os cristais não emitem luz própria e, por isso, é necessária a existência de uma fonte de luz por trás, designada por *backlight*, responsável pelo brilho do *display*. Consoante a orientação dos cristais, estes deixam ou não passar essa luz.[20].

Para visualização das cores é necessário um filtro de cores que é constituído por vários pontos designados de píxeis. Cada píxel contém três sub-píxeis que são responsáveis pelas cores vermelho, verde e azul, daí o nome RGB (*red*, *green* e *blue*). Todas as restantes cores podem ser geradas a partir destas três cores. A tensão de cada sub-píxel é controlada por um transístor que se encontra numa película designada de TFT (*Thin film transistor*) [20].

Ao contrário da tecnologia LCD-TFT que utiliza uma camada de cristais líquidos, a OLED utiliza díodos orgânicos intercalados entre duas camadas de vidro. Estes possuem luz própria, não necessitando de uma fonte externa de *backlight*, reduzindo assim o consumo do *display* e facilitando a visibilidade a partir de qualquer ângulo [21].

Para este projeto o foco são os *displays* com uma tecnologia LCD-TFT, pois são mais fáceis de encontrar no mercado e de custo mais acessível.

2.4 Interface *display*

Um *display* não possui memória e, como tal, é necessário uma atualização continua da imagem apresentada. A velocidade de refrescamento/atualização deve ser tal que não provoque uma cintilação visível na imagem,

pelo que 100 Hz de velocidade de refrescamento é suficiente. Uma velocidade mais elevada só conduz a um aumento da energia gasta e do aquecimento.

Os *displays* podem ser divididos em duas categorias, os que contêm um controlador interno e os que são controlados por um controlador externo. Normalmente, quando estes incluem um controlador interno, o microntrolador apenas precisa de enviar comandos com as inicializações e o envio de imagens só é feito uma única vez, sendo a função de atualização assumida somente pelo controlador. As interfaces utilizadas para este tipo de configuração são SPI, I2C, Motorola 6800 e Intel 8080 (ver figura 2.1) [1].

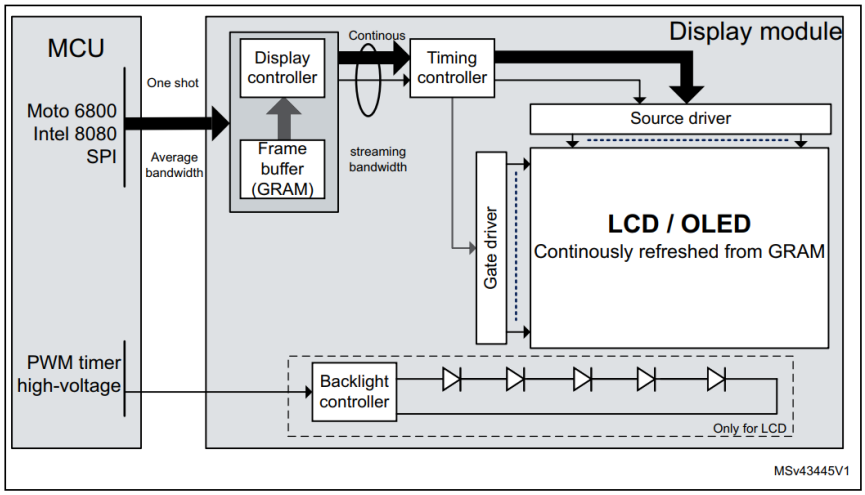


Figura 2.1: Arquitetura de um *display* com controlador e memória interna. Adaptado de [1]

Já os *displays* que não incluam um controlador e memória interna usam uma interface com controlo de sincronismo designada de RGB (ver figura 2.2). Nesta interface, o controlador externo, que pode ser o próprio microcontrolador do sistema ou outro controlador dedicado, tem de enviar continuamente as informações das cores correspondentes a cada píxel. O envio da informação para o *display* é efetuado a uma frequência pré-definida que será diretamente proporcional à velocidade de atualização da imagem [1].

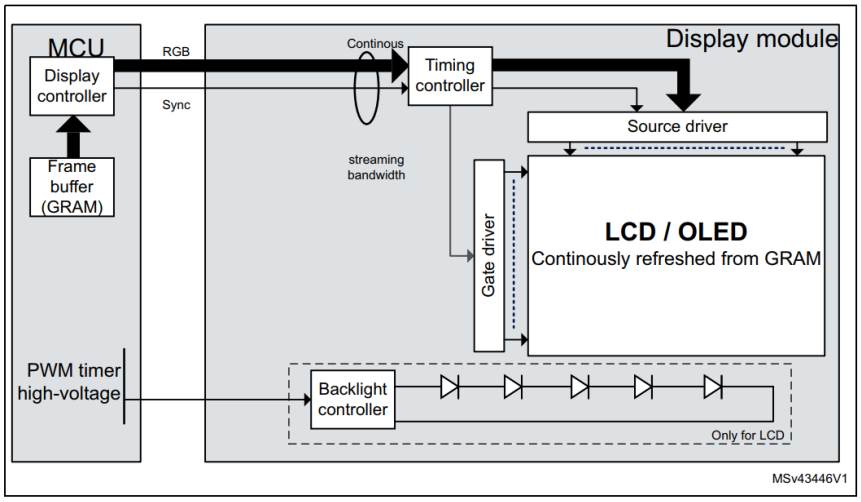


Figura 2.2: Arquitetura de um *display* sem controlador nem memória interna. Adaptado de [1]

Nos pontos seguintes é feita uma breve introdução às principais interfaces referidas anteriormente.

Interface SPI

A interface SPI utiliza quatro linhas (ver figura 2.3). A linha CSX (*chip select*), responsável por selecionar o *driver* a comunicar, as linhas SDI (*serial data in*) e SDO (*serial data out*), utilizadas para envio e receção de dados, respetivamente e, por último, a linha de *clock* que é responsável pelo sincronismo dos dados.

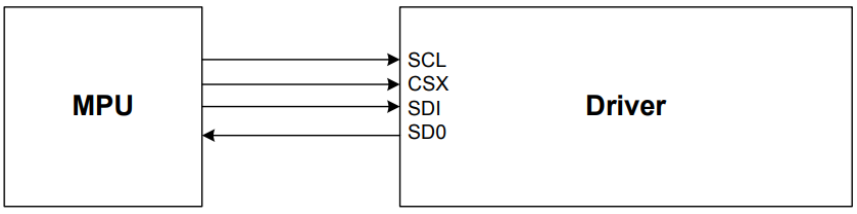


Figura 2.3: Interface SPI entre um *display* e microcontrolador. [2]

Interface I2C

A interface I2C é semelhante à SPI. No entanto, utiliza apenas três fios para controlo do envio e receção de dados (ver figura 2.4). Os fios de seleção do *driver* (CSX) e de *clock* (SCL), são utilizados de igual forma. O terceiro fio (SDA-*serial data*) é responsável pelo envio e receção de dados, ao contrário do que acontece na comunicação SPI onde são utilizados dois fios.

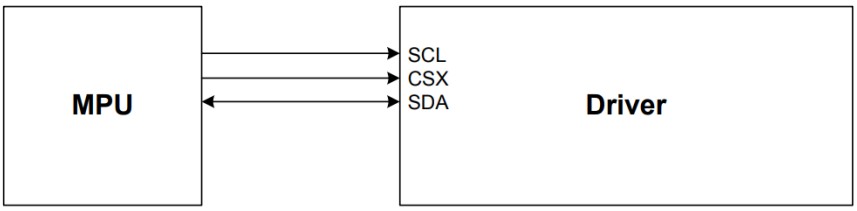


Figura 2.4: Interface I2C entre um *display* e microcontrolador. [2]

Interface Motorola 6800 e Intel 8080

As interfaces Motorola 6800 (M68) e Intel 8080 (I80) têm um funcionamento idêntico. A interface Motorola 6800 contém apenas um fio para configuração de envio e receção de dados (fio R/W) e um de *enable*. Na interface Intel 8080 a configuração é feita em dois fios independentes (fios RD e WR) e não contém qualquer *enable*. Ambas incluem o fio de seleção do *display* (CS), de *register select* (RS) e dos dados referentes à cor a visualizar em cada píxel. Estes dados podem conter até 16 fios (ver figuras 2.5 e 2.6).

Para efetuar um envio de informação usando a interface Intel 8080 é necessário usar o pino "WR"onde, normalmente, na transição do estado de um para zero são transmitidos os dados. A leitura de dados é efetuada de forma semelhante usando o pino "RD"[22]. O envio ou receção na interface Motorola 6800 é efetuado através um pulso no fio *enable* "E"e, dependendo do sinal presente na linha "R/W", zero ou um, é feita uma leitura ou escrita [22].

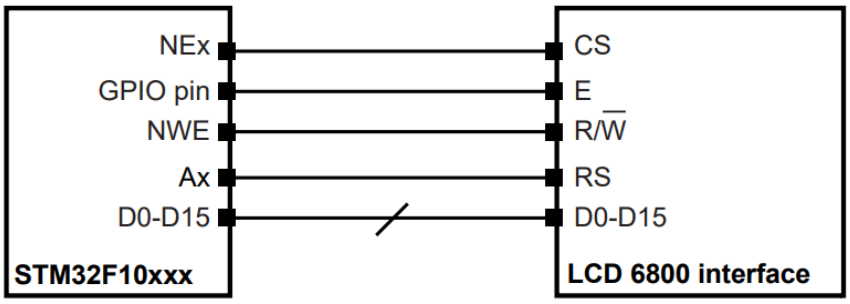


Figura 2.5: Interface Motorola 6800 (16 bits cores) entre um *display* e microcontrolador. [3]

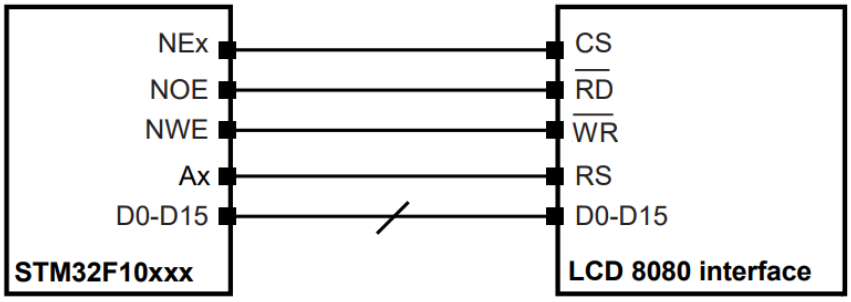


Figura 2.6: Interface Intel 8080 (16 bits cores) entre um *display* e microcontrolador. [3]

Interface RGB

A interface RGB com controlo de sincronismo contém normalmente um total de 28 linhas de interface (ver figura 2.7), onde 24 destas correspondem aos dados de cada píxel e as restantes quatro ao controlo de sincronismo. Nas linhas de sincronismo estão presentes os sinais "VSYNC"e "HSYNC"que são responsáveis por controlar os sinais de varrimento vertical e horizontal, respetivamente. O sinal "HCLK"é referente ao *clock* para envio de dados e o "DE"faz a seleção do *display*. Como referido anteriormente, este tipo de comunicação deve ter um envio contínuo para o display. Para esse efeito, é necessário um microcontrolador com um periférico específico ou um driver dedicado. Na secção 2.6 é possível visualizar alguns dos componentes que

podem ser utilizados para estas funções.

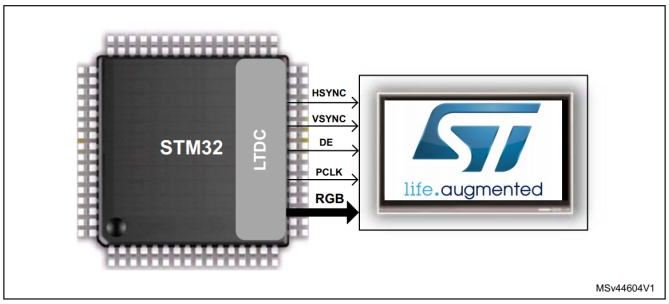


Figura 2.7: Interface RGB com controlo de sincronismo entre um *display* e microcontrolador. [3]

Interface LVDS

A interface LVDS utiliza três a quatro pares de linhas diferenciais, dependendo do número de dados a transmitir, dados estes que correspondem à interface RGB. Na figura 2.8 estão representados os dados referentes à interface RGB distribuídos numa comunicação LVDS. Os sinais que estão no interior do quadrado vermelho são referentes aos sinais de sincronismo.

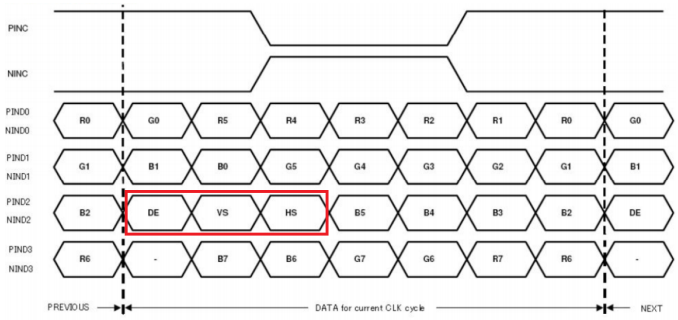


Figura 2.8: Interface LVDS entre um *display* e microcontrolador. [4]

Interface MIPI-DSI

A interface MIPI-DSI é uma comunicação série que utiliza uma interligação até seis fios em três pares diferenciais. Um *display* que use este tipo de

interface contém normalmente um *driver* que converte os dados recebidos para uma interface RGB. Os dados podem ser enviados de duas formas distintas, usando um *frame* longo ou um curto. Como é possível ver na figura 2.9, o *frame* curto contém quatro bytes. O primeiro é referente ao ID (*identification*), os dois seguintes correspondem a informações a enviar e o último é um controlo de erros. O *frame* longo pode conter até 65541 bytes. À semelhança do *frame* curto, o primeiro byte enviado é o ID, seguindo-se outros dois que indicam a quantidade de bytes de informação a enviar. Depois destes segue-se um byte para controlo de erros. Em seguida, são enviados os respetivos dados, e os últimos dois bytes são referentes a um *checksum* para controlo de erros (ver figura 2.10). Ambos os *frames* contêm um byte fixo inicial que indica o início do *frame* e um no final que indica o fim [1].

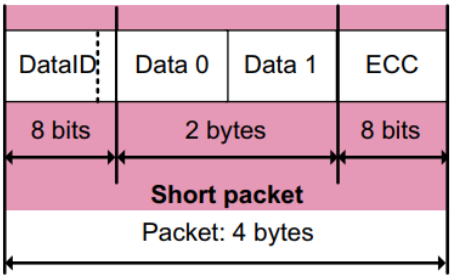


Figura 2.9: *Frame* curto MIPI-DSI. [1]

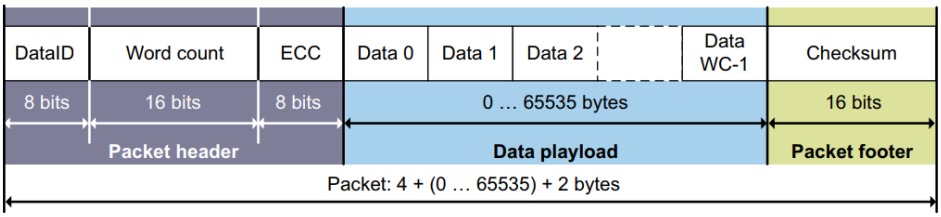


Figura 2.10: *Frame* longo MIPI-DSI. [1]

As interfaces série I2C e SPI usam poucas linhas de interligação em

comparação com as Motorola 6800, Intel 8080 e RGB o que, para efeitos de desenho do circuito impresso, pode ser uma vantagem. Em contrapartida, o envio de dados é muito lento. As comunicações série LVDS e MIPI-DSI também usam poucas linhas de interligação e ao serem diferenciais podem atingir velocidades elevadas. Contudo as pistas no circuito impresso têm de ser desenhadas cuidadosamente, tendo de obedecer a critérios específicos (ver secção 4.1) [1].

2.4.1 Resolução *display*

A resolução de um *display* é referente ao número de píxeis horizontais e verticais que este suporta. Um *display* de 7" cuja resolução esteja especificada como 800x480, indica que este contém 800 píxeis horizontais e 480 píxeis verticais. O aumento de resolução nem sempre implica uma maior nitidez da imagem. Ou seja, um *display* de 10" com uma resolução de 1024x600 apresenta uma imagem com uma definição idêntica ao exemplo anterior de 7". Contudo, se o tamanho do *display* se mantiver 7" e a resolução for de 1024x600, a imagem apresentada tem uma maior nitidez, pois um maior número de píxeis está distribuído pelo mesmo tamanho de ecrã [23].

Na figura 2.11 são ilustrados três círculos de igual tamanho mas com um número diferente de píxeis. Nestas imagens é possível visualizar que com o aumento do número de píxeis a definição dos círculos também aumenta. Ou seja, como referido anteriormente, uma imagem com o mesmo tamanho e uma resolução maior é mais nítida. Contudo, uma maior resolução implica a necessidade de uma memória maior para o armazenamento das imagens.

Os *displays* são normalmente fabricados seguindo um tamanho e resolução típicos. Na figura 2.12 estão representadas as resoluções mais utilizadas em sistemas HMI e a comparação entre os seus tamanhos.

Um outro fator a ter em consideração num *display* é o número de cores que este suporta. O número de cores atribuído a uma imagem é designado de

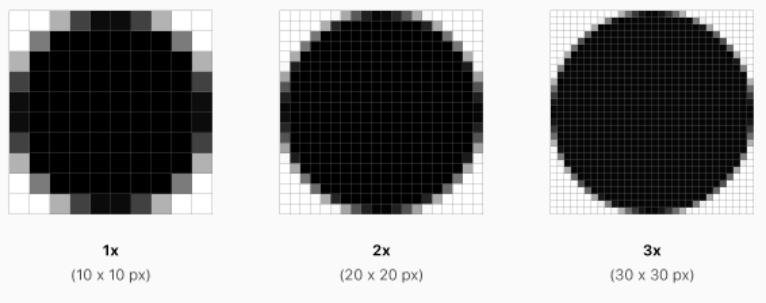


Figura 2.11: Comparação de 3 círculos com diferentes resoluções. [5]

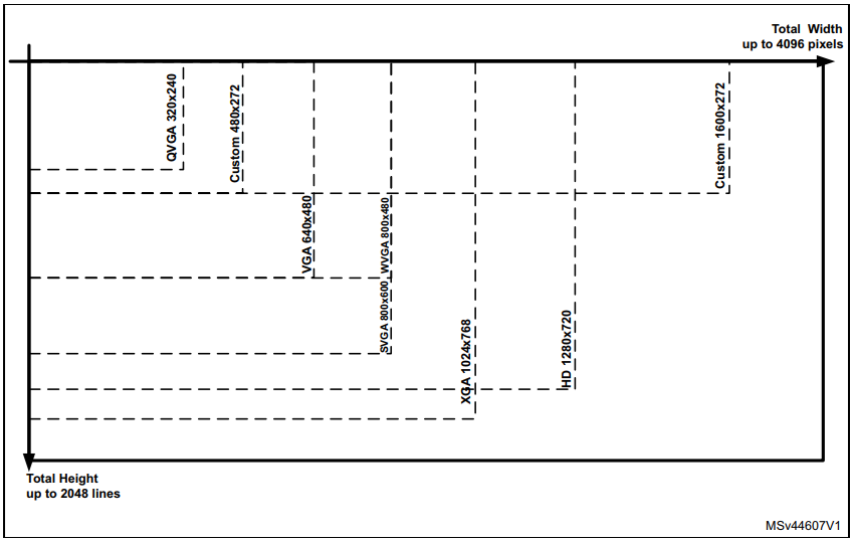


Figura 2.12: Resoluções de *displays* normalizados existentes em aplicações HMI. [6]

profundidade de cor e tem como termo em inglês *color depth*. Quanto mais elevado for o número de cores, melhor é a qualidade da figura a apresentar.

A unidade usada para a profundidade de cor num *display* tem como termo em inglês *bits per píxel* (bpp) e, tal como o nome indica, corresponde ao número de bits atribuídos a cada píxel. O número de cores é obtido pela potência de dois do número de bits, ou seja 16 bpp corresponde a 65536 cores (2^{16}). [24]

Na tabela 2.1 é possível visualizar alguns dos bits por píxel e o seu

respetivo número de cores.

Bits Por Píxel	Número de Cores
1	2
2	4
4	16
8	256
16	65536
18	262143
24	16777216
32	16777216 + Transparência

Tabela 2.1: Bits por píxel e respetivas combinações de cores.

As profundidades de cor 1 bpp, 2 bpp, ou 4 bpp são normalmente atribuídas a imagens brancas, pretas ou tons de cinzento. Já as profundidades 8 bpp, 16 bpp, 24 bpp e 32 bpp são atribuídas a imagens com cores.

A interface com 16 bpp é normalmente designada de RGB565. Os cinco bits mais significativos correspondem à cor vermelha, os seis bits seguintes à cor verde e os últimos cinco à cor Azul. Um complemento é a interface RGB666 com 18 bpp onde cada cor primária usa 6 bits. RGB888 usa 24 bits sendo atribuído um byte para as cores anteriores referidas, a interface com 32 bpp usa três bytes para atribuição da cor pretendida e um byte para o nível de transparência da imagem. A interface de 32 bit designa-se de ARGB8888.

Na figura 2.13 é possível visualizar uma comparação de uma imagem com diferentes profundidades de cor.

2.5 *Displays touch*

Os *displays touch* podem ser divididos em duas categorias, resistiva e capacitiva, cujas tecnologias são completamente distintas. Nos pontos seguintes são apresentadas estas tecnologias assim como as suas diferenças.

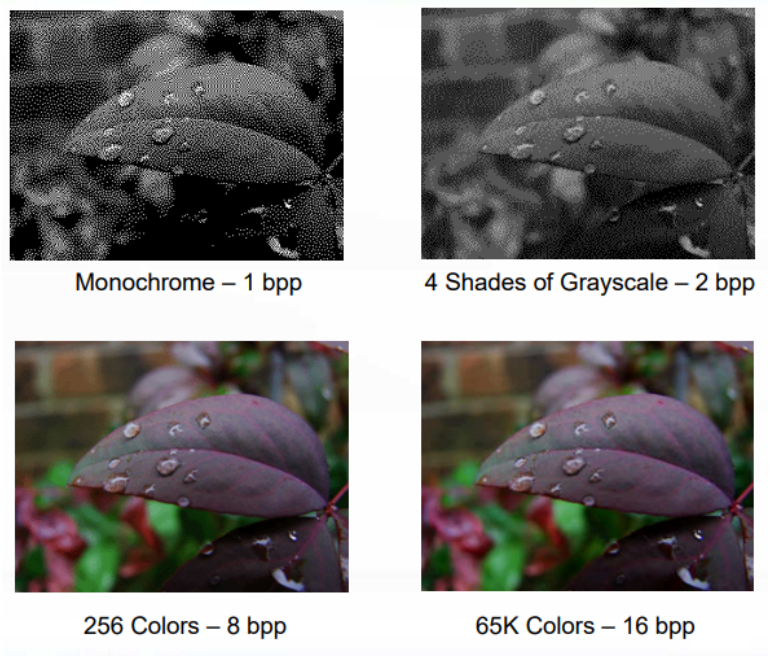


Figura 2.13: Comparação das profundidade de cores 1 bpp, 4 bpp, 8 bpp e 16 bpp. [7]

2.5.1 Tecnologia *touch* resistiva

Um *display* com tecnologia *touch* resistiva é constituído por duas camadas. Uma camada inferior, geralmente de vidro ou acrílico, e uma camada superior constituída pelo material polietileno. Ambas as camadas estão revestidas por um material resistivo que se encontra separado por pequenos espaçadores (ver figura 2.14). Quando a tela superior é pressionada, os dois materiais resistivos encontram-se gerando assim um sinal analógico que corresponde à posição que foi pressionada [25].

As arquiteturas mais populares são de 4 e 5 fios que funcionam de forma semelhante, dando a possibilidade de determinar posições bidimensionais (eixo X e Y). Contudo, também é possível uma terceira posição, designada por Z, que corresponde à pressão exercida sobre o ecrã.

Para efetuar a aquisição de valores relativos ao eixo X é necessário aplicar

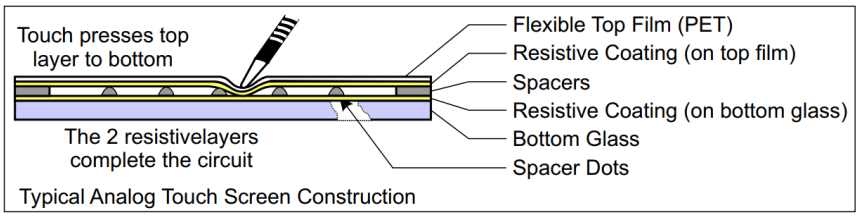


Figura 2.14: Camadas presentes num ecrã resistivo. [8]

uma determinada tensão num dos fios correspondente a esse mesmo eixo e, através do divisor de tensão formado pelas camadas resistivas, é possível identificar a posição que foi pressionada. Para uma medição no eixo Y, o processo usado é semelhante, aplicando uma tensão na linha do respetivo eixo (ver figura 2.15) [8] [25].

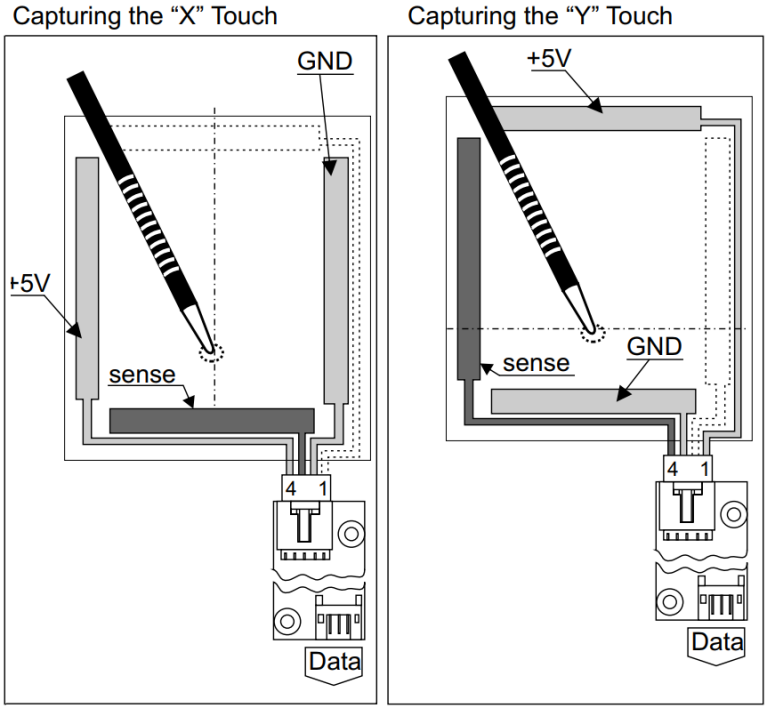


Figura 2.15: Leitura dos dados no eixo do X, à esquerda e, à direita, dos dados no eixo do Y. [8]

2.5.2 Tecnologia *touch* capacitiva

Esta tecnologia é uma das mais utilizadas, principalmente pelo seu uso na maioria dos telemóveis. Esta é normalmente constituída por duas camadas conectadas na parte inferior de um vidro ou acrílico. Cada uma das camadas possui uma matriz de elétrodos em forma de diamante para uma melhor deteção da capacidade induzida pelos dedos. Cada uma destas camadas efetua medições apenas numa dimensão. Ou seja, cada camada regista posições apenas num dos eixos, uma no "X"e a outra no "Y". (ver figura 2.16)[26].

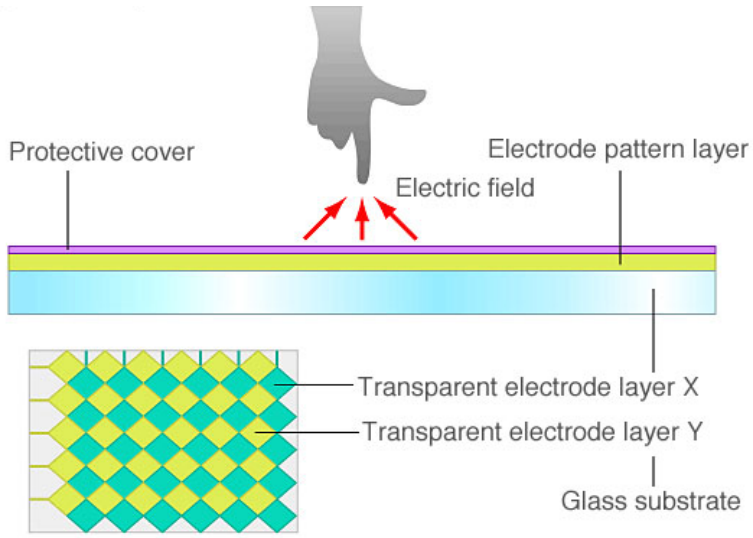


Figura 2.16: Camadas presentes num ecrã capacitivo. [9]

Ao aplicar uma determinada tensão à matriz de elétrodos existente numa das camadas é criado um campo eletrostático uniforme, que é alterado quando deteta um toque no ecrã. A alteração do campo eletrostático na zona do toque pode ser medida através da variação da capacidade elétrica do elétrodo. A localização que foi pressionada é obtida depois de um pré processamento dos sinais lidos nos sensores capacitivos[26].

Em comparação com os ecrãs resistivos, a tecnologia capacitiva tem como vantagens uma maior durabilidade, a capacidade de interpretar vários toques

ao mesmo tempo (*multi-touch)* e uma boa resistência à agua. Contudo a tecnologia dos ecrãs resistivos têm um custo mais acessível e permite o uso de luvas ou qualquer outro objeto, ao contrário do capacitivo que apenas permite dedos ou um objeto condutor [9] [27]. O principal fator que contribui para a diminuição da durabilidade de um sistema resistivo é a necessidade de uma ação mecânica entre as camadas. Um sistema capacitivo, sendo a camada superior normalmente em vidro ou acrílico, será mais resistente em comparação com um ecrã resistivo, que necessita que a camada superior seja de polietileno.

2.6 Microcontrolador e *Driver*

Existem várias soluções no mercado para interligar um microcontrolador a um *display*. A interligação pode ser efetuada de três formas distintas, tais como: através de um *driver* responsável pela conversão da comunicação do microcontrolador para o *display*; o *display* incluir um *driver* dedicado; ou o microcontrolador comunicar diretamente com o *display*.

Para as interfaces onde apenas é utilizado o microcontrolador ou *driver* e estes tenham pouca memória de armazenamento interno, pode ser necessário o uso de uma memória externa.

Nos pontos seguintes são apresentados microcontroladores e *drivers* de fabricantes distintos de forma a perceber o que estes têm em comum e quais as suas vantagens.

A figura 2.17 apresenta alguns dos *drivers* que a empresa EPSON disponibiliza e as suas características mais relevantes. A interface do *display* está disponível para uma configuração de 18 bits ou 24 bits na maioria dos componentes e pode controlar um *display* com uma resolução XGA ou SVGA correspondendo a 1024x768 ou 800x600, respetivamente. Estes componentes necessitam também de uma memória externa que pode ir até 64 MBytes de armazenamento.

Product	CPU Interface Support	LCD Interface Support				Color Depth (Max.)	External Memory Capacity	Package
		Monochrome STN	Color STN	TFT	Typical Resolution			
S1D13513F01A	16-bit I/F, Serial I/F	8-bit	8-bit	18-bit	XGA	MSTN:64 grayscale CSTN:256K colors TFT:256K colors	Up to 16MB SDRAM	QFP22-208
S1D13513B01B	16-bit I/F, Serial I/F	8-bit	8-bit	18-bit / 24-bit	XGA	MSTN:64 grayscale CSTN:256K colors TFT:16M colors	Up to 64MB SDRAM	PBGA1UC256
S1D13515F00A B00B	16-bit I/F, Serial I/F	n/a	n/a	18-bit / 24-bit	XGA	16M colors	Up to 64MB SDRAM	QFP22-256 PBGA1UC256
S1D13517F00A	8-bit /16-bit I/F	n/a	n/a	18-bit / 24-bit	SVGA	16M colors	Up to 16MB SDRAM	QFP15-128

Figura 2.17: Especificação controladores LCD EPSON. [10]

Na figura 2.18 é possível visualizar o diagrama de blocos do *driver* S1D15317 da "EPSON".

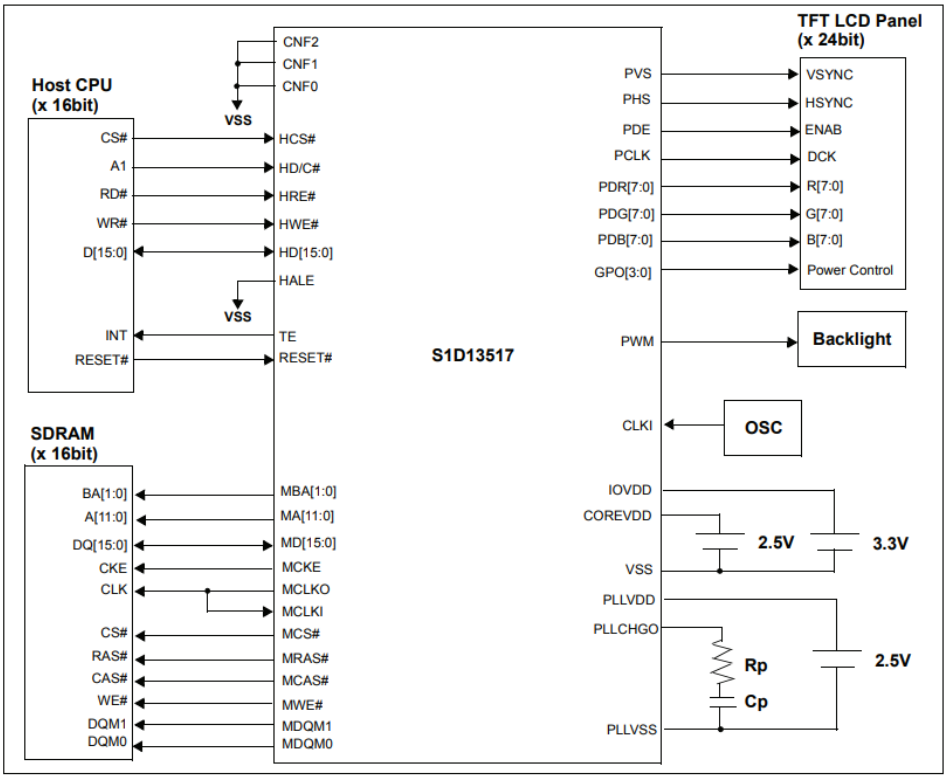


Figura 2.18: Diagrama de blocos do *driver* S1D15317. [11]

Este driver contém pinos dedicados para uma memória externa de 16 bits

e é capaz de converter uma interface Motorola 6800 em RGB. Contém ainda um pino dedicado para controlo do *backlight*. Todas as implementações que recorrem a *drivers* são semelhantes à visualizada na figura 2.18, podendo conter ou não uma memória externa e o controlo do *backlight*.

O fabricante STMicroelectronics disponibiliza uma vasta gama de microcontroladores com possibilidade de uma interconexão com um *display* sem o auxílio de um outro controlador. A interface responsável por esta comunicação é designada de LTDC.

Uma outra interface existente nos microcontroladores da STMicroelectronics, designada FMC (*Flexible Memory Controller*), é responsável pela interligação de uma possível memória externa que pode tornar-se necessária quando a memória interna do microcontrolador não é suficiente. Esta interface suporta 8, 16 ou 32 bits de dados, quatro bancos de dados com a configuração de 11 bits de endereçamento de colunas e 13 bits para endereçamento de linhas.

Na figura 2.19 é possível visualizar as principais características, como velocidades de *clock* e alguns dos periféricos existentes nos microcontroladores com interface LTDC deste fabricante.

STM32 lines	FLASH (bytes)	On chip SRAM (bytes)	Quad-SPI	Max AHB frequency (MHz)	Max FMC SRAM and SDRAM frequency (MHz)	Max pixel clock (MHz)	DMA2D	MIPI -DSI host
STM32F429/ 439	Up to 2 M	256 k	No	180	90	83	Yes	No
STM32F469/ 479	Up to 2 M	384 k	Yes	180	90	83	Yes	Yes
STM32F7x6	Up to 1 M	320 k	Yes	216	100	83	Yes	No
STM32F7x7	Up to 2 M	512 k	Yes	216	100	83	Yes	No
STM32F7x8/ STM32F7x9			Yes	216	100	83	Yes	Yes

Figura 2.19: Comparação de alguns dos microcontroladores da STMicroelectronics. [6]

O fabricante Microchip disponibiliza uma vasta gama de microcontroladores com capacidade para operar diretamente com um display. Estes pertencem à família PIC32MZ e a sua arquitetura é de 32 bits. A velocidade máxima de *clock* desta família é 200 MHz, disponibilizando uma interface para uma memória externa e uma interface QSPI (ver secção 4.5.1) para uma memória *flash*. A resolução máxima suportada por estes microcontroladores é 800x480 e uma interface até 24 bits (RGB888) [28].

Os microcontroladores da Microchip e os controladores externos da Epson contêm soluções para armazenamento até três imagens, o que facilita na projeção de animações e manipulações das imagens. O fabricante STMicroelectronics apenas dispõe de microcontroladores com memória interna para duas imagens. Contudo, estes permitem 32 bits de profundidade de cor e uma resolução máxima de 1280x720. Os restantes integrados vistos apenas atingem resoluções até 800x480 e 24 bits de interface.

Todos os componentes referidos contêm ainda funções que facilitam a transparência da imagem. Estas função podem ser usadas, por exemplo, para alterar a imagem visível no *display* para uma outra, sobrepondo ambas com uma ligeira transparência (ver figura 2.20)

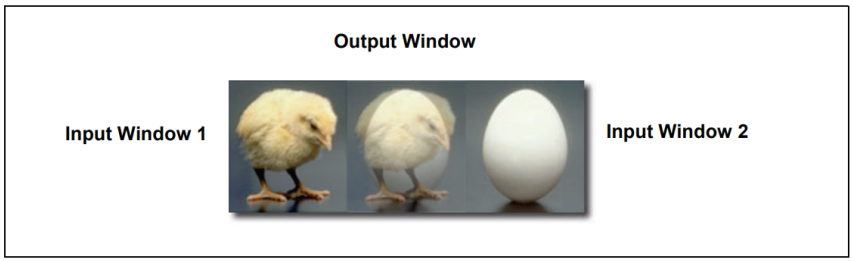


Figura 2.20: Uso da transparência entre duas imagens. [11]

Neste capítulo foram apresentadas as principais opções existentes em termos de componentes para o desenvolvimento da interface HMI. No próximo, são definidos e justificados os requisitos do sistema a desenvolver e as escolhas feitas a nível dos componentes a usar.

Esta página foi intencionalmente deixada em branco.

Capítulo 3

Requisitos e Arquitetura do Sistema

Neste capítulo, numa primeira fase, são referenciados quais os requisitos impostos e como estão distribuídos em *hardware*. Posteriormente, são especificadas de forma mais completa quais as características de cada componente utilizado.

3.1 Requisitos

Os sistemas HMI de 7" atualmente existentes no mercado que se pretende abranger contêm, na sua maioria, uma resolução de 800x480. Com o objetivo de desenvolver uma solução mais competitiva, optou-se pela utilização de um *display* de 7" com uma resolução de 1024x600. O *display* tem de conter uma tecnologia *touch* resistiva, dando assim a possibilidade do utilizador usar as funcionalidades *touch* com luvas.

Relativamente às imagens a apresentar no *display*, são necessárias três imagens com resolução igual à do *display*. Uma destas é usada como imagem de apresentação e as duas restantes como imagens de fundo de duas camadas distintas. Além das imagens referidas são necessárias cerca de 50 imagens

com tamanho indefinido usadas essencialmente para botões e indicações. Todas as imagens a apresentar no *display* são previamente guardadas em memória *flash* para serem apresentadas sempre que forem solicitadas. Para dimensionamento da memória consideram-se quatro imagens de fundo com uma resolução de 1024x600, uma a mais do que especificado para possibilitar o uso de uma terceira camada caso seja necessário, e um mínimo de 50 imagens auxiliares com uma dimensão de 150x150. Estas dimensões são relativas a botões os quais, podem ter dimensões mais pequenas. Todas as imagens devem ter uma quantidade mínima de 16 bits por píxel, (RGB565 [6]).

As diferentes aplicações que incluem o sistema HMI, são constituídas ainda por outros módulos, módulos estes constituídos essencialmente por entradas e saídas com softwares dedicados para a solução em causa. A interligação a usar entre o sistema HMI e os restantes módulos será baseada em barramentos RS485 [29] ou LIN [30] que correspondem às interligações usadas em todos os restantes módulos já existentes na empresa RMtech. Estas interligações não necessitam de funcionar em simultâneo, pois uma aplicação onde os vários módulos existentes comuniquem através de um barramento RS485 não necessita de um barramento de comunicação LIN e vice-versa. Através do barramento selecionado são enviados ao microcontrolador do sistema HMI todas as funções a executar, funções estas como a mudança de imagem, atualização de um botão e possíveis indicações para o utilizador. Por sua vez, o microcontrolador deve retornar as informações necessárias, sendo estas maioritariamente referentes a coordenadas *touch* e dados provenientes de outros periféricos constituintes do sistema.

Relativamente às informações *touch*, estas têm de ser analisadas de forma contínua e guardadas em memória. Sempre que haja uma solicitação prévia por parte do *master* para receber informações *touch*, devem ser enviadas as coordenadas da posição onde foi detetado o toque no ecrã. No caso de

não haver qualquer toque nos últimos 100ms é enviado um *frame* com a informação que não houve qualquer toque.

Este projeto tem como maior relevância o *display* e o *touch*, pois são os principais componentes que interagem diretamente com o utilizador. Contudo, o sistema necessita de funcionalidades adicionais que vão ao encontro das diferentes aplicações necessárias, tornando assim o sistema mais completo e universal. Os periféricos e funções a adicionar ao sistema são os seguintes:

- Controlo da luminosidade do *display*;
- Sensor de temperatura;
- Sensor de luminosidade;
- Leitura da tensão de alimentação;
- Besouro;
- Carregamento de imagens pela comunicação exterior.

O controlo de luminosidade do *display* é essencialmente para adaptar o painel a diferentes locais, sendo possível aumentar a luminosidade em locais com muita luz, ou diminuir esta em locais com pouca luz, de forma a não perturbar o utilizador com alta intensidade da luminosidade do *display*. Este controlo apenas será feito quando houver um pedido no barramento de comunicação enviado pelo exterior.

Os sensores de temperatura e luminosidade são ambos usados para resposta a um pedido externo específico destas mesmas funcionalidade. Este pedido pode ser efetuado para aplicações onde seja necessário distinguir dia e noite, para posteriormente controlar o nível de luminosidade no ecrã. O controlo de luminosidade não pode ser feito automaticamente pois em algumas aplicações o ajuste da luminosidade do ecrã não é usado. Caso o utilizador necessite de alterar a luminosidade do ecrã recorrendo ao sensor, tem

de previamente pedir a informação referente a este e, com base nos dados recebidos, enviar o correspondente comando para alteração da luminosidade para o valor pretendido.

O besouro é usado essencialmente para alertas e sinalização de toques no sistema *touch*. Este é acionado sempre que há um pedido pelo barramento de comunicações externo. Os pedidos para toque de besouro são de dois tipos: um constante, até receber um comando para o desligar, e um outro com a duração de 200ms, usado essencialmente quando um botão é pressionado.

À semelhança de outros periféricos já referidos, a alteração das imagens será efetuada pelo exterior, imagens estas previamente armazenadas em memória. O acesso à memória é efetuado pelo barramento de comunicações externo, onde o sistema tem de reconhecer que os dados recebidos são referentes a uma imagem e armazenar estes numa posição de memória conhecida.

De forma a tornar o sistema HMI universal, o *software* do microcontrolador responsável pelo controlo de todos os periféricos deve ser sempre o mesmo, independentemente da aplicação. Assim, a maioria das ações de interação com o utilizador são previamente pedidas pelo barramento de comunicação externo (RS485/LIN), como alteração de imagens, envio de informações dos seus periféricos, entre outros. Ou seja, numa solução final o sistema HMI necessita que um outro módulo lhe comunique qual a função a executar, módulo este que é normalmente designado de *master*, sendo os restantes módulos designados de *slaves*.

Para dar a possibilidade do sistema HMI ser usado como *master* em determinadas aplicações sem alterar o *software* do microcontrolador principal, foi adicionado um microcontrolador auxiliar a programar com um *software* específico, variável consoante a aplicação.

O sistema HMI pode assim funcionar como *master*, utilizando o micro auxiliar, para controlar todos os módulos da aplicação final, ou então como

slave, onde um módulo externo é responsável por enviar os comandos diretamente para o microcontrolador principal.

3.2 Arquitetura

A configuração dos dois modos, *master* ou *slave*, é feita por *hardware*, onde para uma configuração como *slave* (ver figura 3.1), é feita a interligação entre o barramento de comunicação externa e o microcontrolador principal através de duas resistências de 0 Ω, funcionado como um *shunt*. Por sua vez, para uma configuração como *master*(ver figura 3.2), é feita uma interligação, usando mais uma vez resistências de 0 Ω, entre o barramento de comunicação externa e o microcontrolador auxiliar, e deste para o microcontrolador principal. Desta forma, dependendo da aplicação, o *hardware* é montado de maneira diferente. Contudo, o microcontrolador principal não sofre qualquer alteração em hardware.

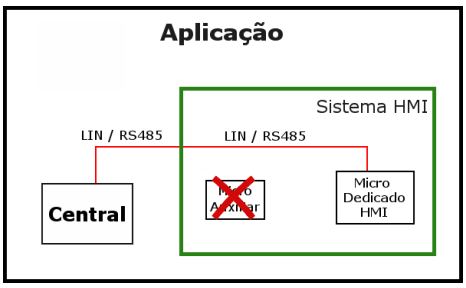


Figura 3.1: Aplicação do sistema HMI como *Slave*.

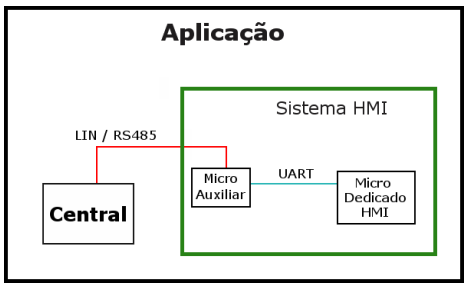


Figura 3.2: Aplicação do sistema HMI como *Master*.

Um exemplo de aplicação em que se usa o microcontrolador auxiliar numa aplicação real é uma máquina de café cujo desenvolvimento da parte de eletrónica foi efetuado pela empresa RMtech. A máquina contém um módulo *slave* que controla todos os motores existentes na máquina. Porém, este módulo apenas atua sobre os motores se receber essa ordem por parte do *master*. Nesta aplicação o *master* é um módulo com pulsadores e um pequeno *display* de caracteres de 16x2. Para uma possível substituição do

módulo de pulsadores pelo sistema HMI em desenvolvimento, mantendo o restante *hardware*, é necessário usar um microcontrolador auxiliar. O programa deste microcontrolador controla o *slave* já existente e as restantes funções do microcontrolador principal do sistema HMI necessárias para esta aplicação.

Na figura 3.3 está representado o diagrama de blocos do sistema HMI desenvolvido neste projeto, incluindo todos os periféricos necessários e o tipo de comunicação usada entre estes e o microcontrolador principal .

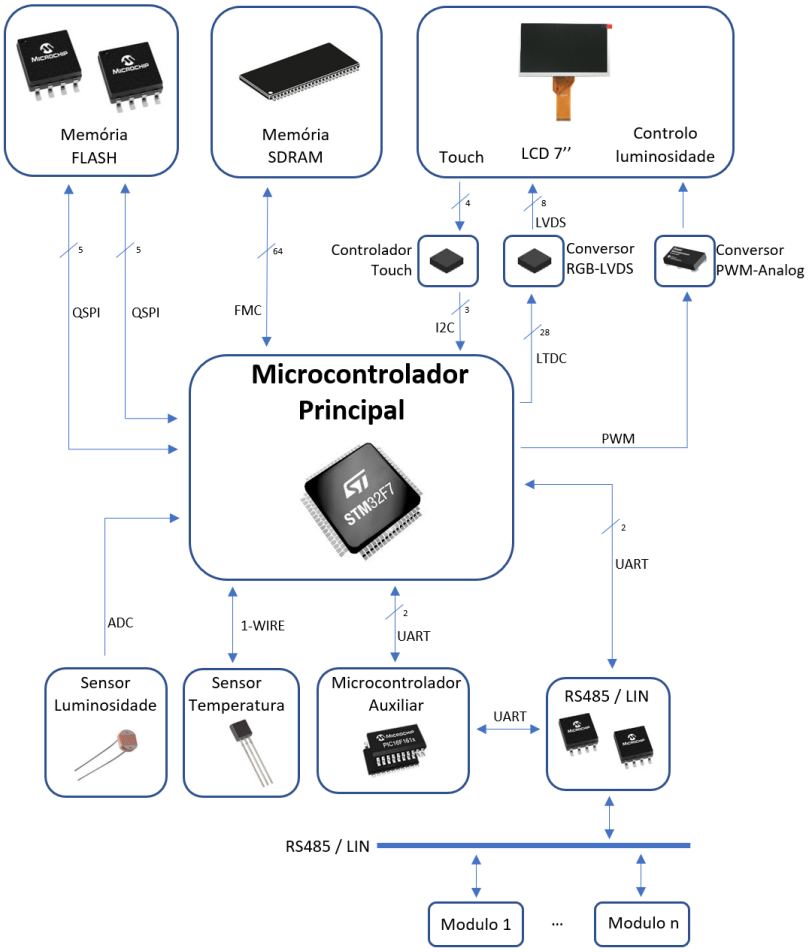


Figura 3.3: Diagrama de blocos do sistema HMI.

Tendo por base a análise dos requisitos descritos neste capítulo, passou-

se à fase de desenvolvimento onde são descritos os componentes usados e o seu funcionamento, fase essa descrita no próximo capítulo.

Esta página foi intencionalmente deixada em branco.

Capítulo 4

Desenvolvimento

Este capítulo tem como objetivo descrever de forma mais completa quais os processos usados para o desenvolvimento do sistema. Nos pontos que se seguem são descritos os componentes e ferramentas usados, quais as suas funções e de que forma interagem com o sistema.

4.1 Desenvolvimento do Hardware

Para o desenho do circuito impresso e esquemático foram usadas as ferramentas *PADS Layout* e *PADS xDxDesigner*, respetivamente. Ambas as aplicações foram projetadas pela companhia *Mentor Graphics* fundada em 1981 e adquirida pela *Siemens* em meados de 2016. A companhia *Mentor Graphics* tem a sua sede nos Estados Unidos da América, tendo suporte e desenvolvimento um pouco por todos os Continentes e dedica-se na sua maioria ao desenvolvimento de softwares de projeto e verificação de componentes e ferramentas eletrónicas [31]. Nas figuras 4.1 e 4.2 é possível visualizar o ambiente gráfico destas duas ferramentas registados durante o desenvolvimento do sistema HMI.

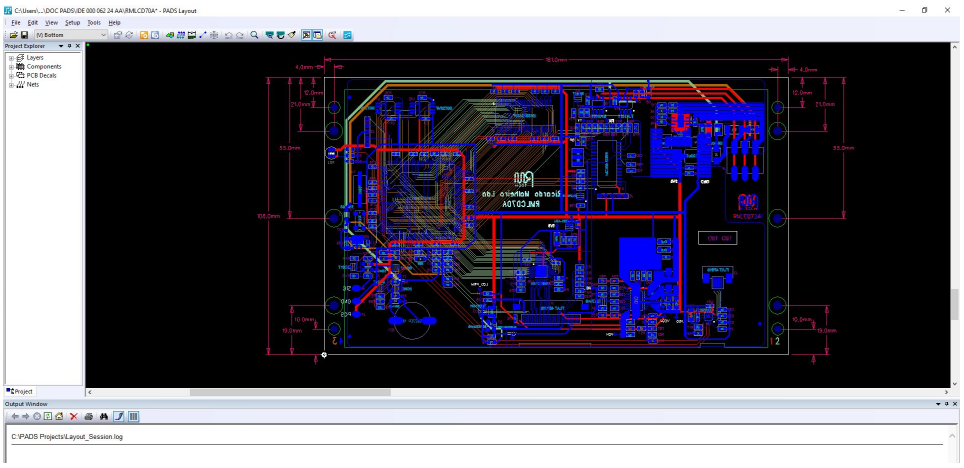


Figura 4.1: Ambiente gráfico do programa PADS Layout.

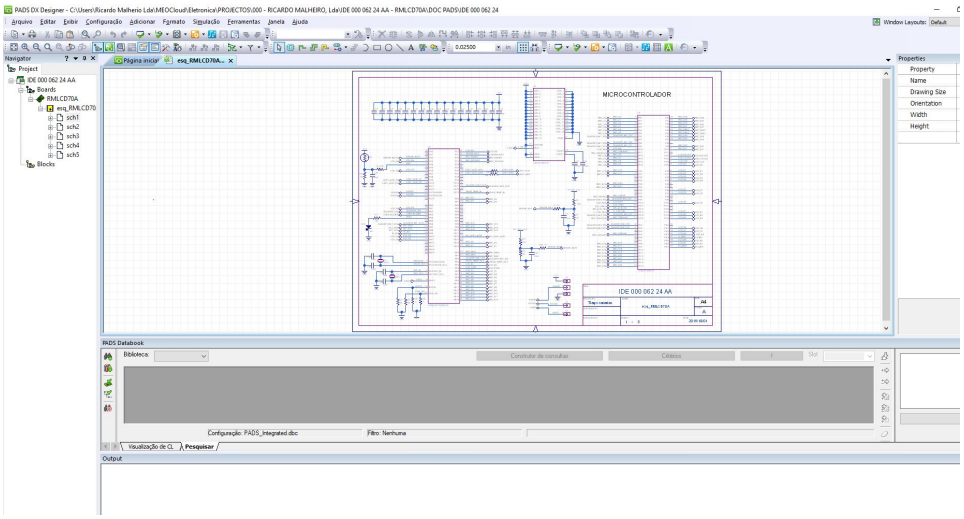


Figura 4.2: Ambiente gráfico do programa PADS xDxDesigner.

Para o desenho da placa foram tidas em conta todas as sugestões fornecidas pelos fabricantes de cada componente, principalmente no desenho das pistas de relógio e dos barramentos de comunicação, devido às frequências elevadas dos sinais que têm de suportar, como a comunicação LVDS (*Low-voltage differential signaling*) (ver secção 2.4) responsável pela interligação entre o *display* e o conversor de RGB/LVDS [17].

As recomendações do fabricante do display para a traçagem das pistas LVDS estão apresentadas na figura 4.3, onde é indicado que sendo "W"a

largura de uma pista diferencial , "S"a distância entre duas pistas diferenciais e "D"dois pares adjacentes de pistas, é recomendado que: S seja menor que 2W e D igual a 2S.

O circuito impresso foi desenvolvido com quatro camadas, onde todos os componentes estão na traseira do circuito impresso (*bottom*). Desta forma, o *display* pode ser aplicado na frente (*top*) sem risco de curto circuito com os outros componentes.

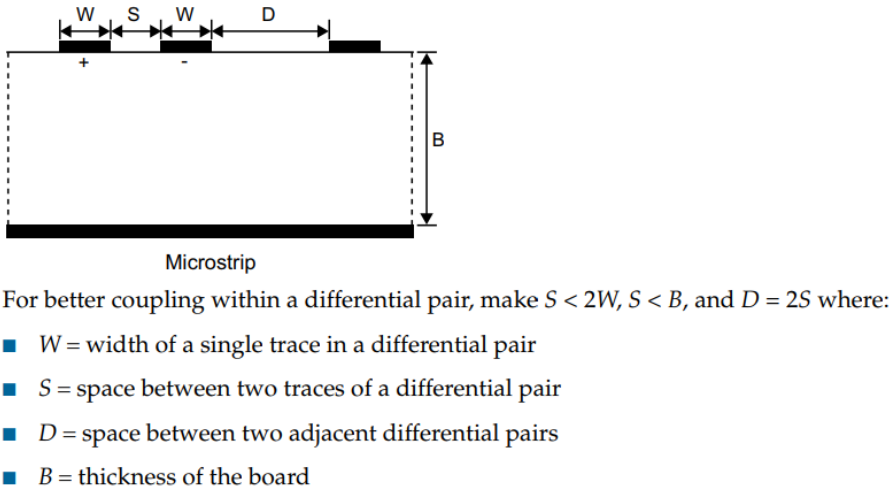


Figura 4.3: Especificação das dimensões das pistas LVDS. [12]

4.2 Desenvolvimento do Software

O programa utilizado para desenvolvimento do software do microcontrolador principal (SM32F746BET6) foi desenvolvido pela companhia *Ac6* em parceria com a *STMicroelectronics* e designa-se por *System Workbench for STM32* [32]. O programa é baseado em eclipse, e permite a programação em dois tipos de linguagem, C ou C++ [33].

Para o desenvolvimento deste projeto a linguagem utilizada foi C, com suporte de bibliotecas disponibilizadas pelo fabricante do microcontrolador,

ST Microelectronics. Estas bibliotecas designam-se por *hardware abstraction layer* (HAL) e facilitam a utilização dos periféricos diminuindo normalmente o tempo despendido em programação.

Para auxílio das configurações do microcontrolador, como periféricos, *clock* e interrupções, foi usado o programa *STM32CubeMX*, desenvolvido também pela empresa *ST Microelectronics* para facilitar a configuração dos seus próprios microcontroladores.

A aplicação *STM32CubeMX* foi usada inicialmente para a escolha do microcontrolador dentro dos disponíveis da gama STM32F7 com características semelhantes. Através deste programa foi possível atribuir todas as funções necessárias aos respetivos periféricos do microcontrolador, verificando facilmente a existência de possíveis conflitos, como, por exemplo, o uso de dois periféricos distintos atribuídos no mesmo pino físico do componente.

Na figura 4.4 é possível ver a configuração do microcontrolador usada para este projeto, onde os pinos representados a verde indicam que já se encontram configurados. Ainda na mesma figura é possível verificar que muitos dos pinos não estão configurados para nenhuma função. Isto deve-se ao elevado número de pinos disponíveis neste microcontrolador, 208 pinos. Contudo o microcontrolador inferior seria de 176 pinos mas já não dispunha de alguns periféricos que seriam necessários para este projeto, como o LTDC (LCD-TFT *display controller*) que é responsável pela comunicação RGB com o *display*.

4.3 Microcontrolador

O microcontrolador STM32F746BET6, escolhido para este projeto, opera a 32 bits e é baseado na arquitetura ARM Cortex-M7. Este componente contém uma memória flash de 1 Mbyte e 320 Kbytes de SRAM. [13]

A frequência máxima de *clock* do microcontrolador é 216 MHz, mas para esta aplicação foi suficiente operar a uma frequência de *clock* de 200

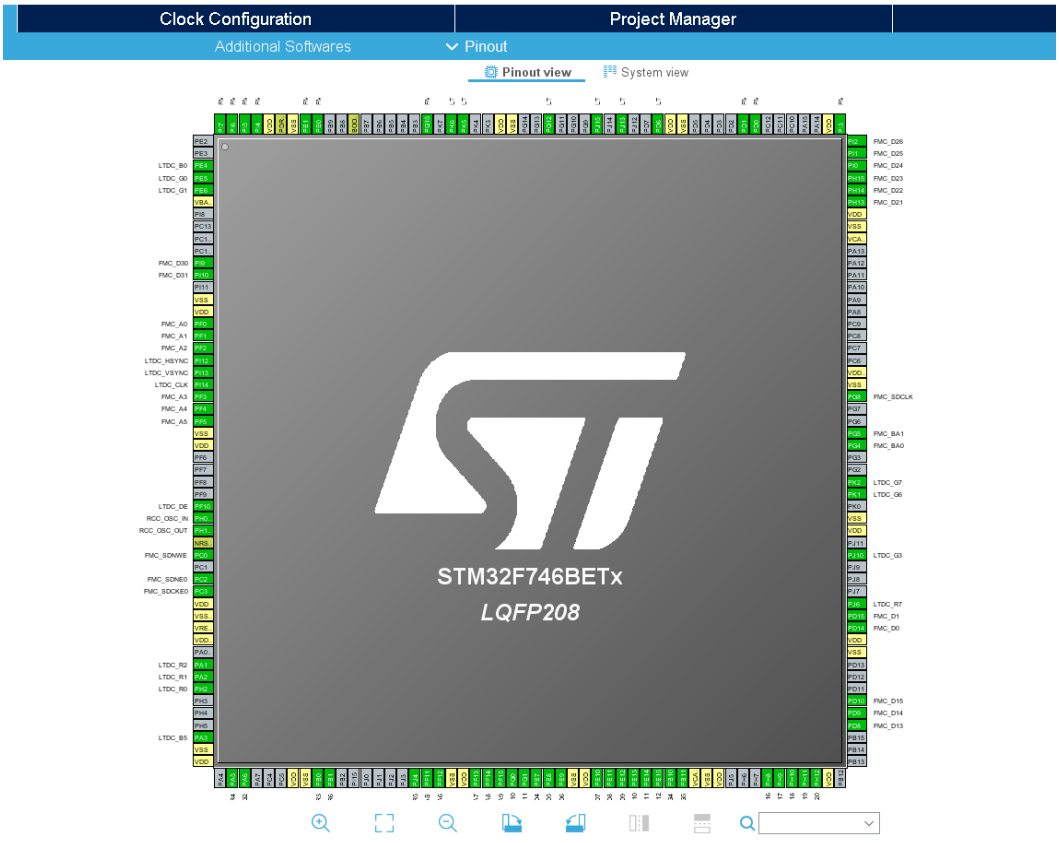


Figura 4.4: *Pin out* do microcontrolador *STM32F746BET6*.

MHz. O periférico FMC (*Flexible Memory Controller*) selecionado para a comunicação com a memória SDRAM externa (ver secção 4.5.2), apenas pode operar a uma frequência de 100 MHz. Uma vez que para obter este valor é necessária a divisão da frequência de *clock* por dois, a seleção da frequência do microcontrolador de 200 MHz (ao invés de 216 MHz), facilita esta operação.

O facto do microcontrolador disponibilizar um elevado número de periféricos (ver figura 4.5), embora nem todos sejam usados nesta aplicação, o seu baixo custo, em comparação com microcontroladores semelhantes de outros fabricantes, e o facto de não ser necessária uma aplicação de programação dispendiosa, foram algumas das razões que levaram à escolha deste

microcontrolador. Um outro motivo foi a prática de operação já existente na empresa RMtech com microcontroladores com uma arquitetura de 32 bits da família STM32F4 que são semelhantes à família STM32F7 usada neste sistema.

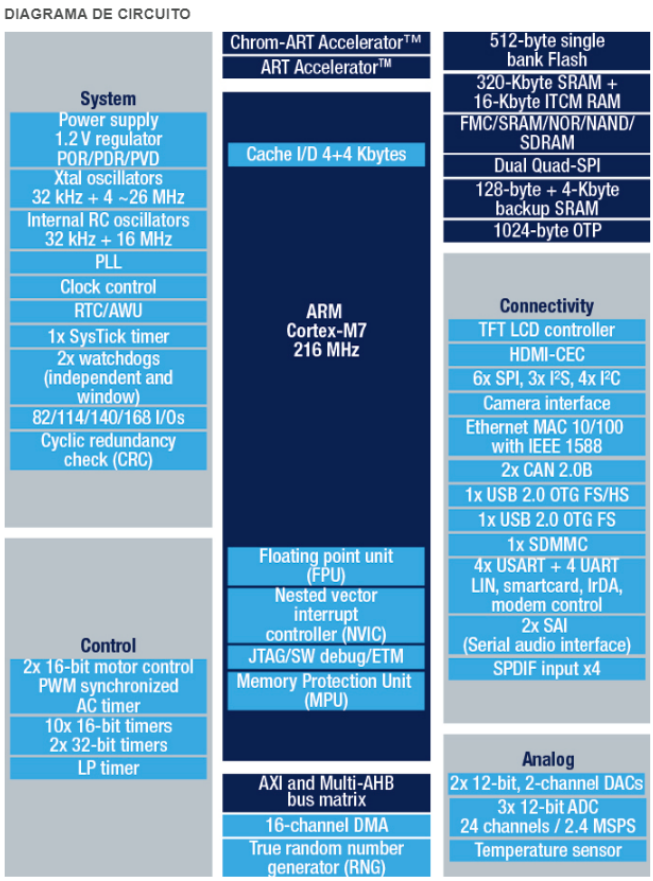


Figura 4.5: Diagrama do microcontrolador STM32F746BET6. [13]

O microcontrolador auxiliar utilizado pertence à família PIC18F do fabricante Microchip, e tem como referência PIC18F25K83.

A implementação deste microcontrolador como auxiliar foi um requisito da empresa que recorre regularmente a microcontroladores desta família, para uma arquitetura de 8 bits. As ferramentas de desenvolvimento já existentes para estes, e o conhecimento das suas arquiteturas, contribuíram para

a sua escolha.

4.4 Comunicação RS485/LIN

A interface de comunicação entre o exterior e o sistema HMI utiliza o protocolo RS485 ou LIN. A seleção do protocolo usado depende da aplicação e dos restantes módulos interligados na mesma interface. Sendo este sistema versátil para ambos os protocolos (RS485 e LIN), é possível comunicar com todos os módulos já desenvolvidos na empresa RMtech.

Atualmente todos os sistemas desenvolvidos na empresa RMtech seguem um protocolo criado internamente, que consiste numa comunicação onde a cada 100 ms o *master* envia um ou mais *frames*, dependendo do número de *slaves*, com pedidos ou ações que o *slave* deve executar. O *frame* é constituído por vários *bytes* que contém informações do endereço a que se destinam os dados, que tipo de dados estão a ser enviados, e um *checksum* para confirmação que o *frame* foi enviado corretamente. O *slave* sempre que recebe um *frame* correto do *master* tem de responder nos 3 ms imediatos à receção, caso contrário o *master* irá registar que ouve uma falha. Ao fim de cinco falhas seguidas, equivalente a 500 ms sem resposta, é acionado um alerta que é apresentado ao utilizador dependente da aplicação.

Na figura 4.6 está exemplificado o modo de como a comunicação é efetuada entre um *master* e um sistema HMI *slave*.

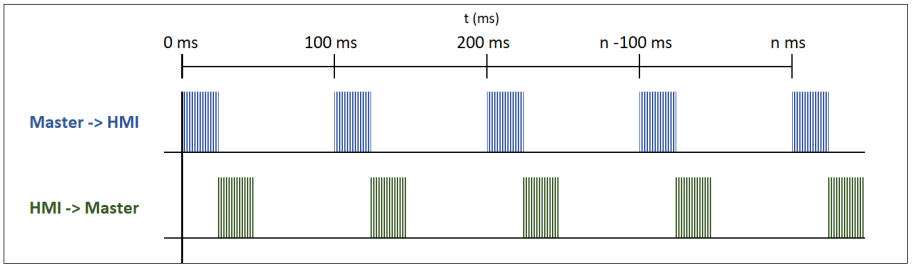


Figura 4.6: Protocolo comunicações RMtech entre um master e um *slave* (HMI).

O protocolo RS485 foi desenvolvido em conjunto pela *Telecommunications Industry Association* e *Electronic Industries Alliance* (TIA/EIA) em meados de 1998. Este protocolo é baseado numa comunicação UART (*Universal asynchronous receiver/transmitter*) onde através de um *driver* dedicado se converte a transmissão em duas linhas diferenciais [29]. Com este tipo de protocolo é possível uma comunicação de 10 Mbps com cerca de 10 metros de distância entre o primeiro e o último módulo, assim como é possível o aumento da distância até 1000 metros mas, de igual forma, a frequência tem de ser reduzida para cerca de 10 Kbps [34].

A especificação do protocolo LIN foi desenvolvida pela *LIN Consortium* que em 1999 lançou a sua primeira versão. Esta tecnologia é implementada em veículos como vertente da tecnologia CAN, pois a implementação da tecnologia CAN tem custos muito elevados em comparação ao LIN. O LIN tem ainda como vantagem a comunicação ser feita apenas com um fio a uma tensão típica de 12 V e poder atingir velocidades até 20 Kbps [30]. Tal como o protocolo RS485, esta comunicação é baseada numa comunicação UART que quando aplicada a um *driver* específico converte o sinal para uma tensão mais elevada. Assim, quando recebe dados converte-os para uma tensão de 3,3 V compatível com o microcontrolador.

4.5 Gestão de memórias

O sistema é constituído por três memórias externas: uma SDRAM (*Synchronous dynamic random-access*) e duas *flash*. As memórias *flash* são responsáveis por guardar as imagens a apresentar no *display*, e a memória SDRAM é usada pelo microcontrolador para fazer uma atualização contínua ao *display*.

4.5.1 Memória FLASH

As memórias em uso são fornecidas pelo fabricante *Microchip* e pertencem à família *NOR* sendo designadas por *Superflash*. Este tipo de memória foi criado pela *Silicon Storage Technology* (SST) e tem como principal vantagem tempos de escrita e limpeza de registos de memória mais rápidos relativamente às memórias NOR usuais [14]. Como é possível ver na figura 4.7, o tempo de escrita reduz cerca de 6 vezes e o tempo de apagar toda a memória reduz de 64 s para 50 ms.

Estas memórias disponibilizam dois tipos de comunicação, SPI (*Serial Peripheral Interface*) e QSPI (*Quad Serial Peripheral Interface*), onde ambas as comunicações estão disponíveis para uma frequência máxima de *clock* de 104 MHz.

Parameter	SST38VF640X 64 Mb		Competitor A 64 Mb		Competitor B 64 Mb	
	Typ	Max	Typ	Max	Typ	Max
Read	–	90 ns	–	90 ns	–	90 ns
Page Read (Word in page after initial access)	–	25 ns	–	25 ns	–	25 ns
Program	7 µs	10 µs	60 µs	–	50 µs	–
Write Buffer Programming	28 µs	40 µs	240 µs (200 µs)*	–	240 µs (200 µs)*	–
Erase: Sector (4 KWord)	18 ms	25 ms	N/A	N/A	N/A	N/A
Erase: Block (32 KWord)	18 ms	25 ms	0.5 sec	3.5 sec	0.5 sec	–
Erase: Full Chip	40 ms	50 ms	64 sec	128 sec	64 sec	128 sec

Figura 4.7: Comparação de uma memória *Superflash* da *Microchip* com duas memórias NOR de um outro fabricante. [14]

A comunicação QSPI é idêntica à comunicação SPI, com a diferença de esta usar quatro linhas de dados ao invés de duas como no SPI usual. Na figura 4.8 é possível visualizar um exemplo da comunicação QSPI. Nas quatro linhas de comunicação são enviados os dados necessários na transição

de *clock* previamente configurada. Na mesma figura, as siglas representadas por C0 e C1 indicam o respetivo comando, de A0 a A5 corresponde a 24 bits de endereço e as restantes siglas H0, H1 representam um byte de informação.

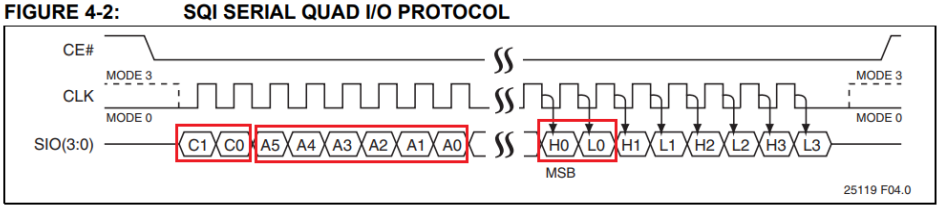


Figura 4.8: Protocolo QSPI [15].

O microcontrolador em uso dispõe de dois periféricos internos para comunicação com duas memórias que usem comunicação QSPI. São usadas duas memórias iguais, onde uma delas contém até quatro imagens de fundo e as restantes imagens são armazenadas na segunda memória, assim como possíveis informações que porventura possam ser necessárias armazenar.

Sendo a dimensão de uma imagem de fundo de igual dimensão à do *display* (1024x600) e dimensionando as memórias para uma capacidade de RGB888 (24 bits), considerando o maior número de bits entre as resoluções RGB565 e RGB888, são necessários aproximadamente 15 Mbits para armazenar uma imagem (ver a expressão 4.1). Ou seja, para armazenar quatro imagens de fundo são necessários cerca de 60 Mbits.

$$1024 \times 600 \times 24 = 14745600 \tag{4.1}$$

Para as restantes imagens são necessários cerca de 27 Mbits que corresponde a 50 imagens com uma dimensão de 150x150 (ver a expressão 4.2).

$$150 \times 150 \times 24 \times 50 = 27000000 \tag{4.2}$$

De forma a diminuir a logística interna e reduzir custos, as duas memórias

são de 64 Mbits cada, ficando livre o restante espaço da segunda memória.

4.5.2 Memória SDRAM

A figura visível no *display* que continuamente é sincronizada pela comunicação LTDC tem de estar armazenada em memória. Como o microcontrolador apenas contém 320 KBytes de DRAM e 1 MBytes de *flash* não é possível armazenar a imagem na memória interna do microcontrolador, pois uma imagem de resolução máxima (1024x600 com RGB888) ocupa 1,8 MBytes. Sendo a memória do microcontrolador insuficiente foi implementada uma memória SDRAM externa.

Para a interligação do microcontrolador à memória SDRAM foi usado o periférico FMC (*flexible memory controller*). A frequência máxima possível para o periférico FMC é de 100 MHz, sendo esta frequência obtida pela divisão por dois da frequência de *clock* principal do microcontrolador (ver secção 4.3).

A memória usada foi decidida por conveniência da empresa, contendo 16 MBytes de memória, distribuídos de igual parte por quatro bancos. Contudo, apenas são usados cerca de 1,8 MBytes, o necessário para o armazenamento de uma imagem com a resolução máxima. A memória contém um barramento de dados de 32 bits e uma frequência máxima de 166 MHz (ver figura 4.9).

O periférico FMC para o microcontrolador em uso suporta até quatro bancos com suporte para um barramento de dados de 8, 16 ou 32 bits. Para a configuração deste periférico foi usada uma vez mais a ferramenta *STM32CubeMX*, tendo sido necessário preencher os parâmetros de acordo com a memória em uso. Estes parâmetros fazem referência ao número de bancos, número de bits para endereçamento e latência CAS (*Column Access Strobe*). Foram usados quatro bancos com 12 bits e 8 bits para endereçar as linhas e colunas, respetivamente, e uma latência de dois ciclos de *clock*

que correspondem a uma frequência de 100MHz, frequência de operação do periférico FMC (ver figura 4.9). À semelhança dos dados referidos, os restantes parâmetros a preencher estão especificados de forma explícita na folha de características da memória. Estes parâmetros são referentes a tempos de execução de funções como *Exit self-refresh*, *self-refresh*, entre outros. Na figura 4.10 é possível verificar o ambiente gráfico da ferramenta *STM32CubeMX* onde se encontram exemplificados os vários parâmetros a configurar nesta plataforma.

KEY TIMING PARAMETERS

Parameter	-6	-7	-75E	Unit
Clk Cycle Time				
CAS Latency = 3	6	7	–	ns
CAS Latency = 2	10	10	7.5	ns
Clk Frequency				
CAS Latency = 3	166	143	–	Mhz
CAS Latency = 2	100	100	133	Mhz
Access Time from Clock				
CAS Latency = 3	5.4	5.4	–	ns
CAS Latency = 2	6.5	6.5	5.5	ns

ADDRESS TABLE

Parameter		4M x 32
Configuration		1M x 32 x 4 banks
Refresh Count	Com./Ind.	4K / 64ms
	A1	4K / 64ms
	A2	4K / 16ms
Row Addresses		A0 – A11
Column Addresses		A0 – A7
Bank Address Pins		BA0, BA1

Figura 4.9: Características relativas à SDRAM. [16]

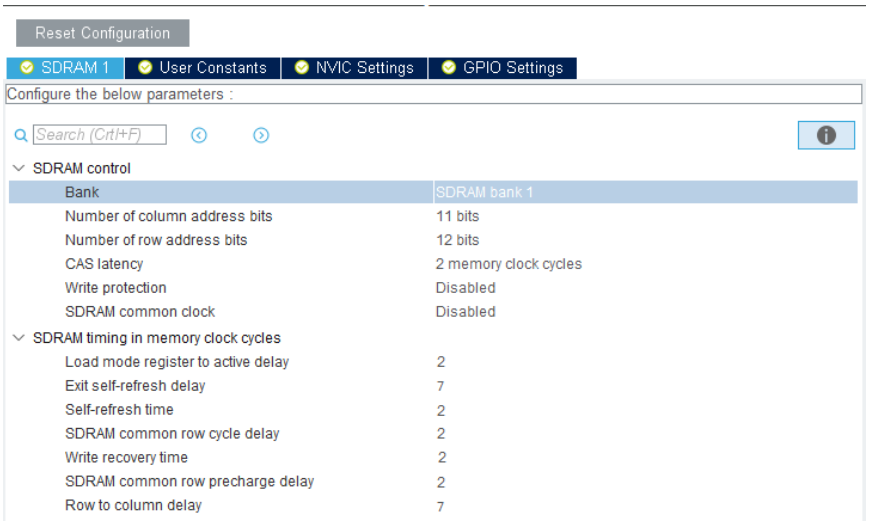


Figura 4.10: Configuração do periférico FMC através ferramenta *STM32CubeMX*.

4.6 Display

O *display* usado neste projeto é fabricado pela empresa Displaytech e designa-se DT070BTFT [4]. Este contém uma resolução de 1024x600 e tecnologia *touch* resistiva. No entanto poderia ser usado um outro *display* de 7" com uma resolução idêntica, devido à interface normalizada comum a outras marcas.

O controlo do *display* através do microcontrolador é composto por duas partes: o controlo de imagens e receção de dados do painel resistivo. Relativamente ao controlo das imagens, este é transmitido pelo microcontrolador através do periférico LTDC que utiliza uma comunicação RGB (ver secção 2.4). Contudo, os *displays* com resoluções 1024x600 existentes no mercado, apenas dispõem de um comunicação LVDS (ver secção 2.4). Para resolver este problema, optou-se pelo uso de um *driver* para converter a comunicação RGB em LVDS [17].

O *driver* em questão trabalha a uma frequência máxima de *clock* de 165 MHz e tem a particularidade de disponibilizar um pino para a configuração das linhas de comunicação LVDS usadas. Este tipo de configuração está também disponível no *display* o que torna os dois completamente compatíveis. As comunicações disponíveis são entre 6 bits e 8 bits de comunicação que correspondem, respetivamente, a 3 e 4 linhas diferenciais. A diferença entre estes dois modos é uma resolução de 24 bits (RGB888 [6]) e 18 bits (RGB666[6]), (ver figura 4.11). Para este projeto as configurações atribuídas ao *display* e ao *driver* foram ambas para uma comunicação de 8 bits, podendo assim utilizar-se uma resolução RGB888.

4.6.1 *LCD-TFT display controller*

Para a configuração do periférico LTDC é necessária a definição da frequência de comunicação pretendida para comunicar com o *display*. O *display* em uso suporta frequências entre os 45 MHz e os 57 MHZ (ver excerto da folha de

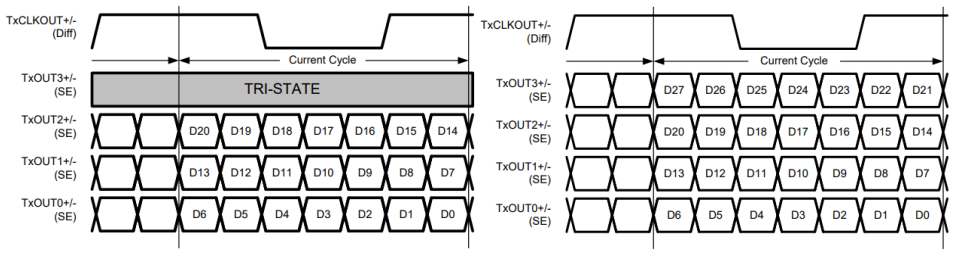


Figure 12. DS90C185 LVDS Map — 18B_MODE = H **Figure 13. DS90C185 LVDS Map — 18B_MODE = L**

Figura 4.11: Modos de comunicação disponíveis no *driver* RGB-LVDS. [17]

características na figura 4.12). Utilizando uma frequência de 52 MHz é possível a transferência de uma imagem de fundo completa em aproximadamente 16,4 ms, o que equivale a cerca de 61 Hz (ver expressões 4.3 e 4.4). Ou seja, com uma frequência de 52 MHz é possível fazer *refresh* total ao *display* sem a sua atualização ser percetível por um ser humano.

ITEM		SYMBOL	MIN	TYP	MAX	UNIT
LVDS Input Signal Sequence	CLK Frequency	tclk	45	51.2	57	MHz

Figura 4.12: Velocidade de *clock* do *display*. [4]

$$Tempo\ refresh\ display = pixeis\ horizontal \times linhas \times \frac{1}{Freq\ LTDC} =$$

$$= 1344 \times 635 \times \frac{1}{52000000} = 0,0164s = 16,4ms \tag{4.3}$$

$$Freq\ refresh\ display = \frac{1}{Tempo\ refresh\ display} = \tag{4.4}$$

$$= \frac{1}{0,0164} = 60,976Hz$$

Como é possível verificar pela figura 4.13 a frequência definida para o periférico LTDC é influenciada por vários fatores, nomeadamente, o número de camadas escolhidas para a aplicação que pode variar entre uma ou duas, o número de cores a usar (8, 16, 24 ou 32 bits), o uso de uma memória externa de 16 bits ou 32 bits e o uso de periférico DMA2D (*Direct memory access* 2D).

Used LTDC layers	Color depth (bpp)	Maximum pixel clock (MHz)			
		LTDC		LTDC + DMA2D	
		SDRAM 16-bit	SDRAM 32-bit	SDRAM 16-bit	SDRAM 32-bit
1 layer	32	42	74	25	39
	24	56	83	34	52
	16	83	83	51	78
	8	83	83	83	83
2 layers	32/32	21	37	12	20
	32/24	24	42	14	23
	32/16	28	49	17	28
	32/8	34	59	21	34
	24/24	29	49	17	27
	24/16	34	59	20	33
	24/8	42	74	26	42
	16/16	43	74	25	41
	16/8	57	83	34	56
	8/8	83	83	51	82

Figura 4.13: Frequências possíveis para o periférico LTDC. [6]

O DMA (*Direct memory access*) é um periférico que permite o acesso a determinados registos, sem a intervenção do *software*. Ou seja, quando previamente configurado possibilita a interligação de vários módulos, como por exemplo: quando uma quantidade pré-definida de dados é recebida pela comunicação UART, estes podem ser transferidos diretamente para um outro periférico, SPI, I2C, etc .

O DMA2D é um DMA dedicado para a manipulação de imagens, onde é possível que uma imagem armazenada num determinado endereço de origem seja transferida para um endereço de destino sofrendo alterações previamente configuradas. Estas alterações podem ser, por exemplo, preencher um retângulo de uma imagem com uma cor predefinida, a conversão do for-

mato de cores da imagem de origem para o endereço de destino, misturar duas imagens com diferentes tamanhos, entre outras [6].

Nesta aplicação é usada uma memória SDRAM de 32 bits e uma comunicação de 24 bits (RGB888) conforme a especificação. O microcontrolador possibilita a configuração até duas camadas. No entanto, para esta aplicação não é necessário mais que uma, pois o objetivo não é a manipulação das imagens, como transparências, mas apenas apresentar as imagens que forem requeridas, e essa função pode ser feita apenas com uma camada sobrepondo as várias imagens pretendidas escrevendo-as na mesma posição de memória da SDRAM.

Recorrendo ao DMA2D é possível a utilização de dois *buffers* no carregamento de imagens, diminuindo assim o risco de possíveis distorções ao enviar a imagem para o *display*. Estes dois *buffers* trabalham em simultâneo, estando um dos *buffers* a enviar um determinado *frame* para o LTDC e o outro *buffer* a processar dados referentes ao próximo envio. Estes dois *buffers* são sincronizados e invertem as suas funções com o sinal VSYNC que representa um sincronismo vertical (ver figura 4.14).

Utilizando as configurações referidas anteriormente, uma memória SDRAM com 32 bits, DMA2D e apenas uma camada, é possível utilizar o periférico LTDC a uma frequência máxima de 52 MHz (ver figura 4.13), frequência essa que corresponde à usada neste projeto.

A informação RGB enviada pelo periférico LTDC é constituída pelo sinal de *clock*, um sinal de sincronismo vertical (VSYNC) e um de sincronismo horizontal (HSYNC) e ainda 24 linhas para envio dos dados a carregar em cada píxel.

O *display* em uso é constituído por 600 linhas com 1024 píxeis cada. Cada uma das linhas é iniciada pelo sinal de sincronismo horizontal (HSYNC) seguindo-se os sinais *horizontal back porch* (HBP), área ativa do *display* (1024 píxeis) e *horizontal front porch* (HFP).

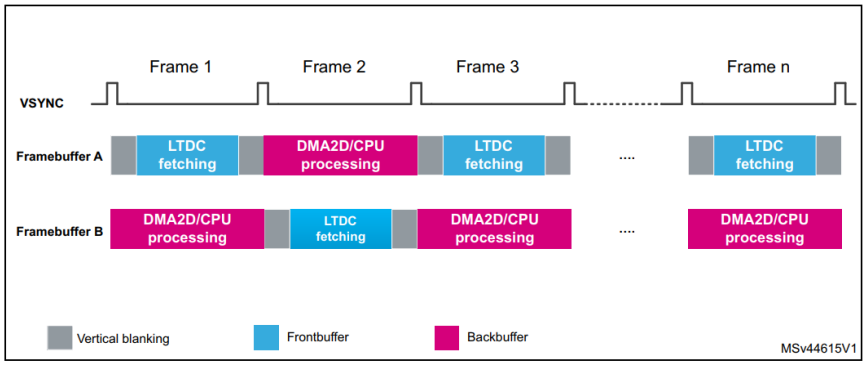


Figura 4.14: Sincronização de periférico LTDC recorrendo a dois *buffers*. [6]

O início do envio da informação da primeira linha é sincronizado pelo sinal vertical (VSYNC), que à semelhança da informação horizontal contém o sinal *vertical back porch* (VBP), seguindo-se a informação de todas as linhas referidas no parágrafo anterior e, por fim, os pulsos referentes ao *vertical front porch* (VFP).

Na figura 4.15 está representado o tipo de comunicação RGB para um *display* onde a área visível horizontal é apenas de 480 píxeis.

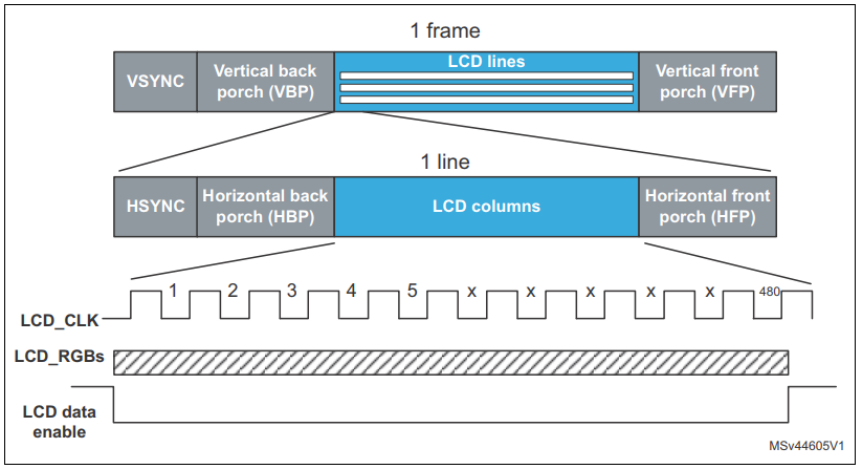


Figura 4.15: Comunicação RGB de uma área visível horizontal de 480 píxeis. [6]

Os sinais HBP, BFP, VBP e VFP são constituídos apenas por pulsos

de *clock* não contendo quaisquer dados. O número de pulsos em cada um destes sinais é especificado pelo fabricante de cada *display*, assim como para os sinais VSYNC, HSYNC e a área visível. Estes valores foram inseridos na ferramenta *STM32CubeMX* para a inicialização do periférico LTDC.

O microcontrolador contém um registo de 32 bits que é referente à cor de fundo do *display*, cor esta que é apresentada após a inicialização do periférico LTDC no microcontrolador. A cor de fundo é por pré definição preta, sendo mantida para esta aplicação. Para a configuração da camada é necessário definir qual a sua resolução, à qual foram atribuídos 1024 píxeis por 600 linhas e a sua posição inicial no *display* que será (0,0). Para este projeto a camada usada foi projetada para conter o tamanho total do *display*. Contudo, esta poderia ser reajustada para um tamanho inferior caso fosse pretendido.

4.6.2 Controlo da luminosidade do display

O *display* é constituído por 21 LEDS, para controlo da luminosidade, que se encontram distribuídos por sete séries de três LEDs (ver figura 4.16), cuja tensão de alimentação recomendada para a luminosidade máxima é de 9,6 V.

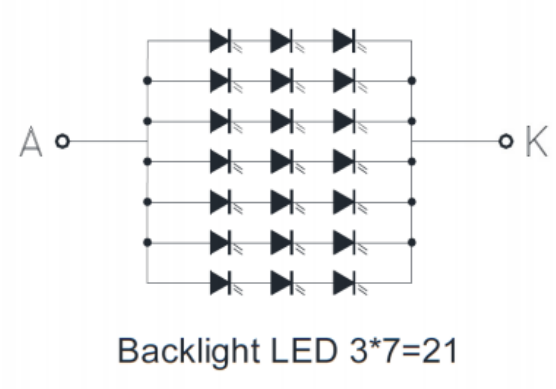


Figura 4.16: Circuito dos LEDS existentes no display. [4]

Para efetuar o ajuste da luminosidade é necessário variar a tensão aplicada aos LEDs. Para esse efeito foi usado o *driver* TS6104 que fornece uma tensão de saída dependente do PWM (*Pulse-Width Modulation*) aplicado no pino *enable*. Na figura 4.17 está representada a implementação usada para o controlador de luminosidade referido. Na correspondente folha de características do controlador TS6104 é sugerido que a frequência a aplicar no PWM esteja entre 100 Hz e os 500 Hz, correspondendo a um período de 10 ms e 2 ms, respetivamente. Como tal, para o controlo da luminosidade é usada uma frequência de 200 Hz, 5 ms de período. O PWM é gerado pelo microcontrolador através do periférico *timer 9*. Para a configuração deste periférico foi necessário iniciar dois registos, *Prescaler* e *Counter Period*.

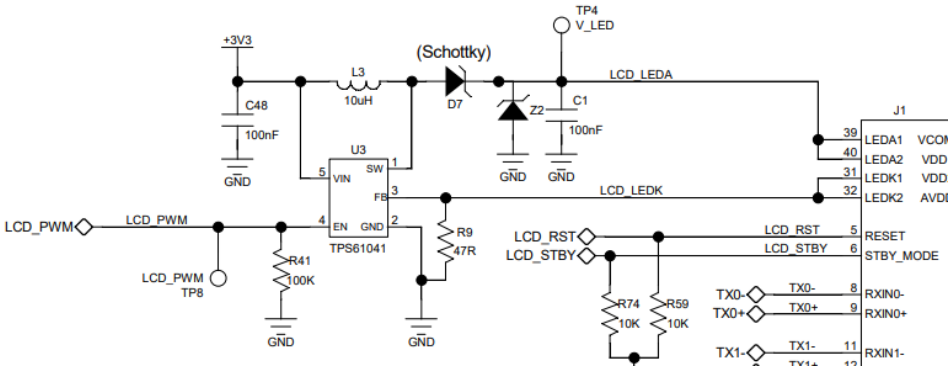


Figura 4.17: Circuito usado para o controlo da luminosidade do *dispay*.

O *Counter Period* é dependente do período de PWM pretendido e da frequência de *clock* do *timer* 9. Como se verifica na expressão 4.5, para obter um período de 5 ms com frequência de *clock* de 1 MHZ, o registo *Counter Period* terá de ser carregado com o valor 5000 numa base decimal. A frequência de *clock* de 1 MHz é obtida através do registo de *prescaler* que corresponde a um divisor da frequência principal do microcontrolador (ver expressão 4.6).

$$counterPeriod(16bits) = \frac{T_{PWM}}{\frac{1}{F_{clkPWM}}} = \frac{0,005}{0,000001} = 5000 \quad (4.5)$$

$$Prescaler(16bits) = \frac{f_{clkAPB1}}{F_{clkPWM}} = \frac{200000000}{1000000} = 200 \quad (4.6)$$

4.6.3 Touch

A informação do *touch* resistivo é disponibilizada pelo *display* através de quatro pinos designados por X-, X+, Y- e Y+. Para converter a informação analógica em coordenadas do *display* foi usado o *driver* STMPE811. Este *driver* pode ser configurado com dois tipos de interface: SPI ou I2C, tendo como frequência máxima 1 MHz e 400 KHz, respetivamente (ver figura 4.18).

A interface escolhida foi SPI podendo assim atingir velocidades mais elevadas. Esta configuração foi efetuada através de *hardware*, colocando o pino "IN1"a 3,3 V. O *driver* disponibiliza também um pino de saída digital que funciona como interrupção. Este muda o seu estado de um para zero sempre que deteta que o *touch* foi pressionado.

Symbol	Parameter	Test condition	Value			Unit
			Min	Typ	Max	
$CLKI2C_{max}$	I^2C maximum SCLK	V_{IO} = 1.8 - 3.3 V	400	–	–	kHz
$CLKSPI_{max}$	SPI maximum clock	V_{IO} = 1.8 V	800	–	–	kHz
		V_{IO} = 3.3 V	1000	–	–	kHz

Figura 4.18: Velocidade de comunicação do integrado STMPE811. [18]

O *driver* possibilita a receção de dados até 3 eixos, 'X', 'Y' e 'Z', onde a informação de cada um dos eixos X e Y é armazenada em registos de 12 bits e o eixo Z num registo de 8 bits. A receção dos valores referentes aos eixos X e Y pode ser configurada entre 10 e 12 bits, correspondendo a um máximo de 1023 e 4095, respetivamente. Para esta aplicação, a configuração usada para ambos os eixos é de 10 bits. Para o eixo do 'X' a informação

recebida corresponde ao número de píxeis horizontais (1024). Contudo, para o eixo do 'Y' o valor máximo 1023 é convertido para uma resolução de 600. Assim, as coordenadas de cada eixo têm uma relação direta com cada píxel facilitando a sua interpretação.

O eixo 'Z' não foi usado pois este apenas indica a pressão que é exercida no ecrã, função esta que não se aplica a nenhuma das aplicações pretendidas.

Para o funcionamento correto do *driver* é necessário uma pré inicialização de alguns parâmetros, como a definição do número de eixos a usar e o tempo de conversão do sinal analógico em digital. A conversão do sinal analógico em digital usa, por omissão, uma frequência interna de 6,5 MHz. No entanto através da divisão do *clock* é possível configurar a frequência para 3,25 MHz e 1,625 MHz. Depois de definir a frequência interna é necessário configurar qual o tempo pretendido para a conversão do sinal analógico, tempo este que pode variar entre 5,5 us e 56,4 us (ver figura 4.19). Para este sistema as configurações usadas são de 3,25 MHz para a frequência de *clock* e um tempo de conversão de 64 ciclos de *clock*, correspondendo a 19,8 us.

Sample time setting	Conversion time in ADC clock	6.5 MHz (154 ns)	3.25 MHz (308 ns)	1.625 MHz (615 ns)
000	36	5.5 µs (180 kHz)	11 µs (90 kHz)	22 µs (45 kHz)
001	44	6.8 µs (147 kHz)	13.6 µs (74 kHz)	27 µs (36 kHz)
010	56	8.6 µs (116 kHz)	17.2 µs (58 kHz)	34.4 µs (29 kHz)
011	64	9.9 µs (101 kHz)	19.8 µs (51 kHz)	39.6 µs (25 kHz)
100	80	12.3 µs(81.5 kHz)	24.6 µs (41 kHz)	49.2 µs (20 kHz)
101	96	14.8 µs (67.6 kHz)	28.8 µs (33 kHz)	59.2 µs (17 kHz)
110	124	19.1 µs (52.3 kHz)	38.2 µs (26 kHz)	56.4 µs (13 kHz)

Figura 4.19: Tempos de conversão para o integrado STMPE811. [18]

4.7 Sensor de Temperatura

Habitualmente, no desenvolvimento interno de novas aplicações em que seja necessária a leitura de temperatura é usado o sensor DS18B20 [35]. Este

sensor foi o selecionado para o projeto, devido ao seu baixo custo de aquisição e de forma a não aumentar custos na logística.

O sensor DS18B20 usado contém apenas três pinos, onde dois deles são alimentação e o terceiro o pino de comunicação. A alimentação suportada pelo sensor está compreendida entre os 3 V e os 5,5 V. Para o sistema atual são usados os 3,3 V disponíveis na fonte principal para a alimentação do sensor de temperatura.

A comunicação usada pelo sensor é designada de 1-WIRE. Esta comunicação é bidirecional e tal como o nome indica necessita apenas de um fio, podendo conter vários sensores na mesma linha. A linha de comunicações está normalmente num nível lógico '1', através de uma *pull-up* aos 3,3 V. Sempre que é necessário comunicar é feita uma transição para o nível lógico '0'. Na figura 4.20 é possível ver o circuito de ligação do sensor ao microcontrolador usado para este sistema.

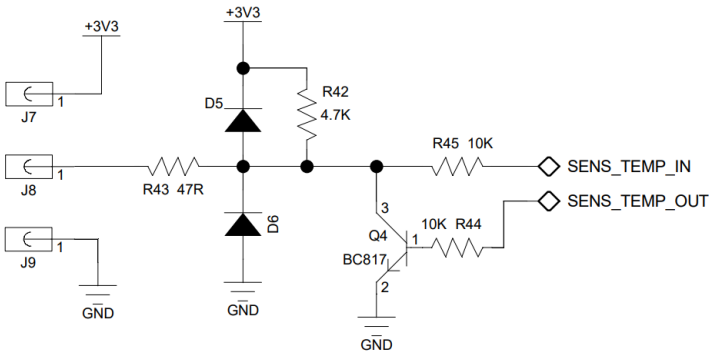


Figura 4.20: Circuito usado para controlo do sensor de temperatura.

4.8 Leituras Analógicas

A leitura da tensão de alimentação e a leitura da luminosidade são feitas através de entradas analógicas disponíveis no microcontrolador principal. Este dispõe de três registos ADC (*Analog-to-Digital Converter*) com 24 canais cada um. Os pinos com ADC usados foram escolhidos por conveniência para a interligação das pistas no circuito impresso.

4.8.1 Leitura da tensão de Alimentação

Conforme a especificação referida no capítulo 3.1, Requisitos, é necessária a leitura da tensão de entrada para o controlo de algumas funções, tais como besouro e envio para o utilizador. Esta informação é exposta para o utilizador no *display*, sempre que for solicitado pelo *master*. Juntamente com os dados referentes ao pedido de apresentação da tensão do sistema, é necessário o envio das coordenadas onde será apresentado no *display* o respetivo valor da tensão de alimentação.

De acordo com a folha de características do periférico ADC, o tempo máximo de conversão da leitura analógica é de 16,40 us [13]. Desta forma, é efetuada uma leitura a cada 100 ms, que corresponde ao período da comunicação com o exterior. Na figura 4.21 está representado o circuito usado na ligação do pino de alimentação ao microcontrolador. A tensão de entrada representada pela *label* "12V/24V_AUX"está conectada ao cátodo de um díodo que se encontra em série com um fusível de *rearme* automático. Deste modo, a entrada fica protegida contra inversão de polaridade, assim como todo o sistema. Uma vez que a queda da tensão num díodo é de 0,7 V, este valor tem de ser assumido nos cálculos na tensão de alimentação.

Como referido nos requisitos, as leituras efetuadas têm de ser armazenadas e enviadas com aproximação às unidades. Contudo, os cálculos efetuados foram para uma escala de 0,5 V de forma a criar uma janela onde variações mínimas da tensão não afetem o valor registado. Ou seja, depois de um

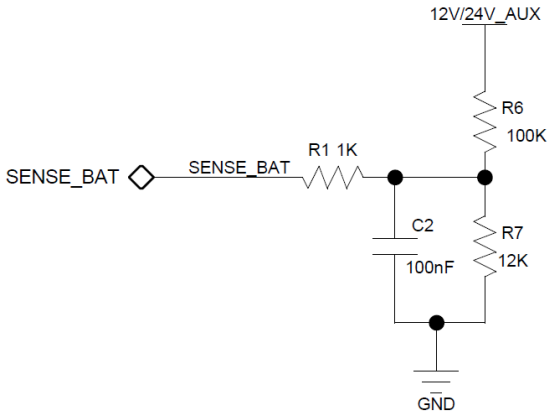


Figura 4.21: Circuito responsável pela leitura da tensão de alimentação.

valor lido e armazenado em memória, apenas é alterado se a próxima leitura for superior ou inferior 0,5 V do valor anteriormente armazenado. Este método tem como objetivo a estabilização da informação de alimentação a ser enviada para o utilizador.

A configuração referente ao número de bits para a conversão analógica digital foi de seis bits, disponibilizando assim 64 estados possíveis. Dividindo a tensão de referência de 3,3 V pelos 64 estados, estamos perante uma resolução de aproximadamente 0,05 V, o suficiente para a escala de 0,5 V pretendida.

Para o cálculo da tensão de entrada foram usadas duas expressões: uma regra de três simples para relacionar o valor recebido do conversor analógico com a tensão aplicada no pino do microcontrolador e outra expressão que relaciona a tensão no pino do microcontrolador com a tensão de alimentação do sistema. Sabendo que, quando a tensão no divisor for 3,3 V - tensão de referência do conversor - o valor lido no conversor é de 63. Com esta informação, é possível relacionar o valor lido no conversor com a tensão do divisor.

Na expressão 4.7 é possível ver um exemplo onde o valor lido pelo con-

versor é 38, correspondendo aproximadamente a 1,99 V. Para converter a tensão do divisor na tensão de alimentação do sistema foi usada a expressão 4.8, sendo o valor R1 referente à resistência de 100 kΩ e o valor R2 referente à resistência de 12 kΩ. Aplicando o exemplo anterior à expressão 4.8, o valor de tensão obtido para alimentação do sistema é aproximadamente de 19,3 V.

$$\frac{3,3V}{V_{div}} = \frac{63}{Valor_{ADC}} \Leftrightarrow \frac{3,3V}{V_{div}} = \frac{63}{38} \Leftrightarrow V_{div} = 1,99V \tag{4.7}$$

$$V_{div} = (V_{in} - 0,7)\frac{R2}{R1 + R2}$$

$$V_{div} = (V_{in} - 0,7)\frac{12000}{100000 + 12000} \tag{4.8}$$

$$V_{in} = \frac{V_{div}}{0,107} + 0,7$$

4.8.2 Leitura da luminosidade

A implementação de um sensor de luminosidade para este sistema não tem influência em nenhum periférico do sistema HMI. Apenas é necessário o envio desta informação para o exterior quando solicitado. Este sensor vem apenas como complemento do sistema para uma aplicação em que eventualmente seja necessário esta função.

Quando a informação da luminosidade é solicitada pelo exterior, é enviado o valor recebido no ADC sem sofrer qualquer alteração, sendo analisado e processado pelo módulo que solicitou o pedido.

O sensor de luminosidade usado tem como nome comum LDR (*Light Dependent Resistor*) e, tal como o nome indica, a sua resistência varia con-

forme o valor da luminosidade. No LDR usado a resistência varia entre os 8 kΩ e os 20 kΩ [36], onde a resistência aumenta com a diminuição da luminosidade. Para o cálculo do divisor de tensão foram usados na resistência R1 os valores extremos de resistência do LDR e para a resistência R2 o valor de 1 KΩ (ver figura 4.22). Desta forma, para uma tensão máxima de alimentação (28 V), o valor mínimo possível no divisor de tensão é de 1,3 V (ver expressão 4.9) e o valor máximo é de 3,033 V (ver expressão 4.10). Assim, os valores da luminosidade que são enviados para o exterior numa escala de seis bits, estão aproximadamente compreendidos entre os valores 25 e 58 (ver expressões 4.11 e 4.12). Estes valores são conhecidos pelos módulos que solicitarem um pedido de informação relativo à luminosidade.

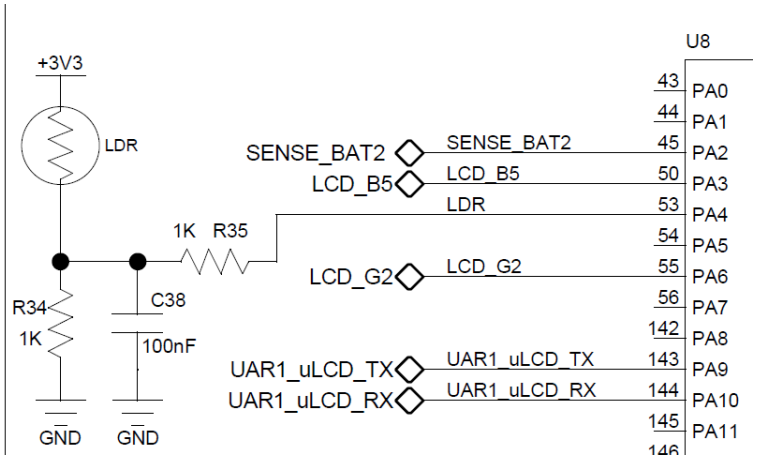


Figura 4.22: Circuito responsável pela leitura da luminosidade.

$$V_{div} = (V_{in} - 0,7)\frac{R2}{R1 + R2} = 27,3\frac{1000}{21000} = 1,3V \tag{4.9}$$

$$V_{div} = (V_{in} - 0,7)\frac{R2}{R1 + R2} = 27,3\frac{1000}{9000} = 3,033V \tag{4.10}$$

$$\frac{3,3V}{V_{div}} = \frac{63}{Valor_{ADC}} \Leftrightarrow \frac{3,3V}{1.3} = \frac{63}{Valor_{ADC}} \Leftrightarrow Valor_{ADC} = 24,8 \quad (4.11)$$

$$\frac{3,3V}{V_{div}} = \frac{63}{Valor_{ADC}} \Leftrightarrow \frac{3,3V}{3,033} = \frac{63}{Valor_{ADC}} \Leftrightarrow Valor_{ADC} = 57,9 \quad (4.12)$$

4.9 Besouro

O besouro para este sistema atua de duas formas distintas: por ordem externa (comunicação RS485/LIN), ou quando deteta que algum botão é pressionado. Na primeira opção deverá tocar permanente até que receba uma nova ordem de paragem. Este tipo de função é essencialmente utilizado em alertas de emergência. Na segunda, sempre que há um toque no ecrã onde as coordenadas são consideradas válidas, o besouro toca durante 200 ms, produzindo um pequeno "BIP", suficiente para o utilizador perceber que um botão foi pressionado.

O besouro usado pertence à família *piezo* e tal como o LDR e o sensor de temperatura é o normalmente usado na empresa RMtech. Este besouro contém eletrónica interna que gera uma frequência de 4 KHz, pelo que desta forma apenas é necessário aplicar uma tensão fixa para gerar o som pretendido. A tensão a aplicar para este besouro está compreendida entre os 8 V e os 16 V tendo um consumo de 8 mA a uma tensão continua de 12 V [37].

Como o sistema HMI pode funcionar até uma tensão de 28 V é necessário reduzir a tensão a aplicar no besouro, pois este não suporta mais que 16 V, como referido no parágrafo anterior. Para este efeito foi usado um circuito extra de configuração, onde em circuitos de 24 V a corrente passa por um *zenner* de 12 V colocado em série com o besouro, fazendo assim com que

haja uma queda de tensão de 12 V (ver figura 4.23). Na fase de inicialização do microcontrolador o pino de configuração está num nível lógico zero. Este estado é apenas alterado quando recebe informação do ADC responsável pela leitura da tensão de entrada e a informação recebida refere que estamos perante um sistema de 12 V. Caso contrário, o pino de configuração continua a um nível lógico zero. Desta forma é possível garantir que em condições normais a tensão do besouro não irá exceder os 16 V.

O circuito de configuração usado podia ser substituído por uma fonte de alimentação comutada onde seria possível converter a tensão de alimentação para 12 V. Contudo, sendo os componentes atualmente usados *zenners*, resistências e transístores que existem em grande quantidade na empresa, é possível reduzir o custo deste circuito em comparação com uma fonte comutada.

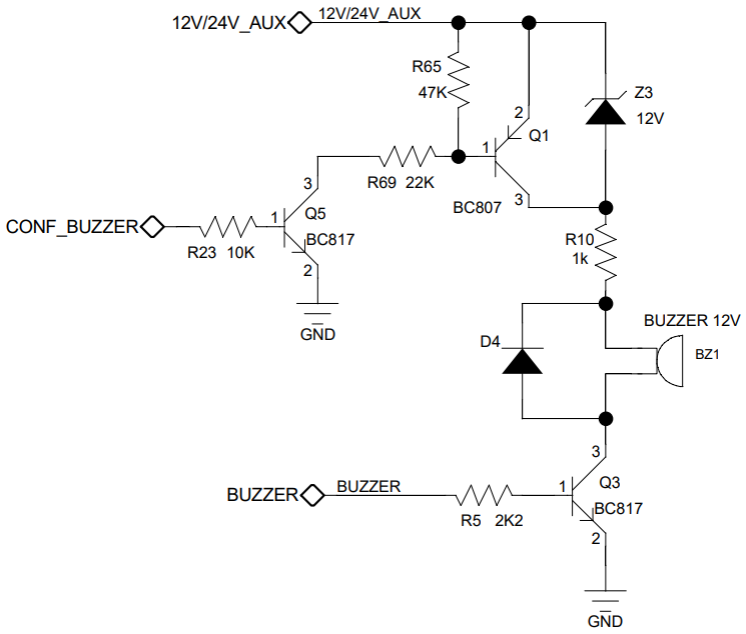


Figura 4.23: Circuito de controlo do besouro.

Tendo por base a análise dos componentes e periféricos referidos neste capítulo, passou-se à fase de implementação, descrevendo-se as várias fases do funcionamento de todos os componentes do sistema, fase essa descrita no próximo capítulo.

Esta página foi intencionalmente deixada em branco.

Capítulo 5

Implementação

Neste capítulo é descrita a implementação usada em *software* para controlo dos vários periféricos constituintes do sistema. São usados três fluxogramas como auxilio à interpretação do sistema, onde um dos fluxogramas corresponde ao programa principal e os restantes a duas interrupções, UART e TIMER.

5.1 Programa principal

Ao iniciar o microcontrolador STM32F746BET6, as primeiras operações efetuadas são as inicializações de *clock* e dos os restantes periféricos usados, como UART, *TIMER*, LTDC, ADC, entre outros. Em seguida, o microcontrolador está continuamente a ler informação dos seus periféricos e a responder ao exterior de acordo com o que lhe é pedido. Como referido na secção 4.8.1, as comunicações com o exterior são efetuadas num período de 100 ms. Desta forma, o programa usa essa mesma base de tempo para executar as suas funções. Ou seja, depois de executar as funções que dependem apenas de si próprio, como leitura da tensão de alimentação, temperatura, luminosidade e *touch*, o programa espera pela receção da comunicação externa e, estando esta completa e validada, responde com as informações

que lhe forem pedidas, processando em seguida os dados recebidos que são essencialmente o carregamento de imagens no display.

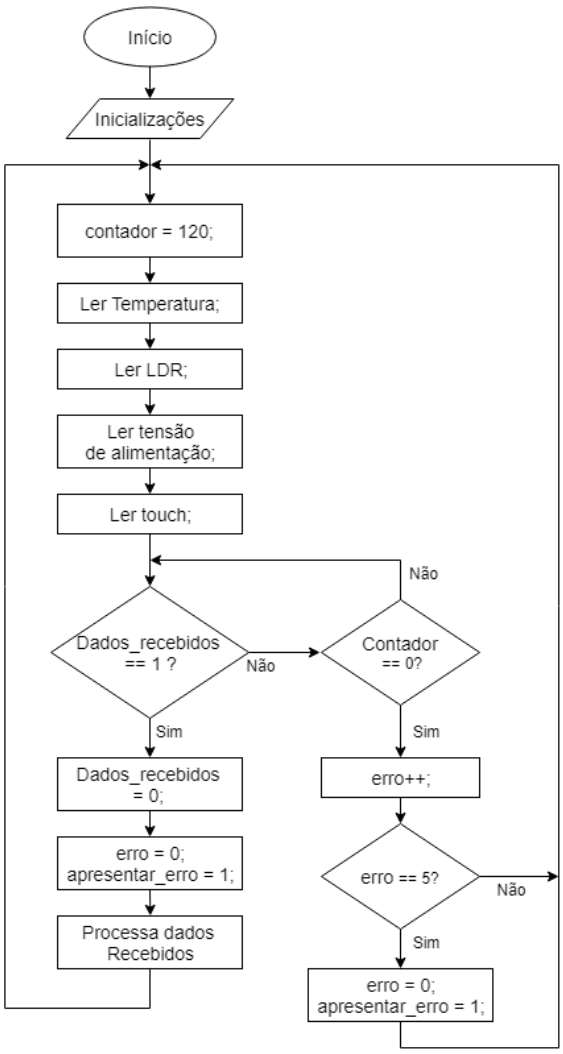


Figura 5.1: Fluxograma do sistema HMI.

Como é possível verificar no fluxograma da figura 5.1, é usada uma variável auxiliar, designada de "contador", que tem como objetivo garantir que o programa não fica parado mais de 120 ms à espera da comunicação externa. Ou seja, se porventura houver uma falha de comunicação na aplicação, e o sistema HMI necessitasse desta mesma comunicação para

prosseguir, os dados enviados para o exterior quando fossem retomadas as comunicações poderiam não corresponder a uma informação atualizada.

5.2 Interrupções TIMER

Com auxílio da aplicação *STM32CubeMX* foi configurado um dos *timers* disponibilizados pelo microcontrolador de forma a interromper o sistema a cada 1 ms. Dentro desta interrupção apenas é feito um decremento a uma variável de 16 bits caso esta seja diferente de zero (ver figura 5.2). O objetivo desta rotina é usar esta variável para executar uma função de *delay* na rotina principal (main). Sendo uma variável de 16 bits é possível criar um atraso de 65534 ms. No fluxograma da figura5.1 é possível ver um exemplo do uso da interrupção *timer*, onde a variável "contador"é inicialmente carregada com o valor 120. Em seguida são executadas as funções necessárias para o sistema HMI como, por exemplo, a leitura da temperatura e da tensão de alimentação. Esta rotina apenas volta a ser executada caso receba dados pelo exterior, que num funcionamento normal comunica a cada 100 ms, ou então quando a variável "contador"atingir o valor zero, que significa que foram ultrapassados 120 ms sem a receção de qualquer tipo de dados.

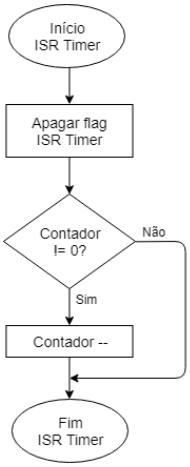


Figura 5.2: Fluxograma referente à interrupção de 1ms.

5.3 Interrupções UART

A receção dos dados do exterior é feita através de uma interrupção processando imediatamente os dados. Nesta função são verificados os bytes recebidos e, caso estes correspondam a um *frame* válido, é ativada uma variável designada por "Dados_recebidos". Esta variável é utilizada pela função principal (main) para obter a informação que foram recebidos dados corretamente do exterior e podem ser processados.

Ainda na interrupção, depois de ativar a variável "Dados_recebidos", é enviado para o exterior a informação que se encontra armazenada no respetivo *buffer* de envio.

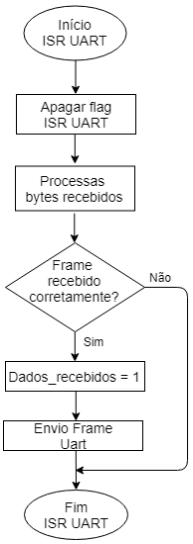


Figura 5.3: Fluxograma referente à interrupção de receção de dados pelo exterior.

Tendo por base a implementação efetuada e descrita neste capítulo, segue-se uma outra fase onde é possível verificar como foram efetuados os testes ao sistema HMI e seus periféricos.

Capítulo 6

Testes e Validações

Numa primeira fase de teste foram montados apenas os circuitos de alimentação no circuito impresso. Desta forma, foi verificado que os valores estavam corretos, nomeadamente 3,3 V na fonte principal que alimenta ambos os microcontroladores, todas as memórias externas e por fim as tensões 3 V, 9,6 V, 18 V e -6 V que correspondem aos pinos "VCOM", "AVDD", "VGH", e "VGL"existentes no *display* [4]. Depois de confirmar todos os valores de tensão foram montados os restantes componentes no circuito impresso de forma a prosseguir com o teste do microcontrolador e restantes periféricos.

Antes do teste total do sistema foram testados individualmente os vários integrados do circuito impresso, assim como os periféricos do microcontrolador que estão interligados a estes. Desta forma é possível também fazer uma depuração do circuito impresso desenhado.

6.1 *Display*

O primeiro teste efetuado foi a confirmação do sinal de *clock* do periférico LTDC. Em seguida, foi feita a inicialização deste periférico com os parâmetros inseridos na aplicação *STM32CubeMX* que dizem respeito ao *display* (ver

secção 4.6.1). Nesta mesma inicialização a cor de fundo foi configurada como branco (0xFFFFFF), o que se verificou no display.

Para confirmar que nenhuma das pistas no circuito impresso estava trocada, numa primeira fase, uma vez que a SRAM ainda não estava configurada, foram feitos testes mudando apenas os bits da cor de fundo e verificando se correspondia à cor pretendida. Os testes à cor de fundo foram feitos para as cores vermelho, verde e azul, onde para cada uma das cores foram acrescentados bits do mais significativo para o menos significativo. Ou seja para teste da cor vermelha o primeiro registo usado foi 0x800000, em seguida 0xC00000, terminado apenas no registo 0xFF0000. Desta forma, verificou-se que as três cores ficavam mais claras com a inserção de cada bit, conforme o que era pretendido (ver figura 6.1). Com este teste foi possível confirmar que o periférico LTDC estava bem configurado, assim como o *display* e o conversor RGB-LVDS estavam bem interligados e a funcionar corretamente.

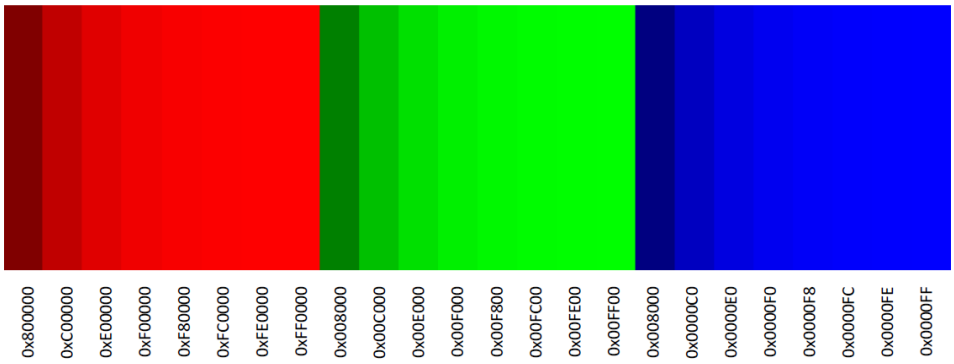


Figura 6.1: Código de cores usado para teste inicial do display.

6.2 Luminosidade

Para efeitos de teste da luminosidade do *display* foram gerados dois *duty-cycle* diferentes, podendo-se assim visualizar no osciloscópio que estes correspondiam ao sinal pretendido. Foi então gerado um *duty-cycle* de 10% e

um outro de *duty-cycle* de 90%, sendo possível visualizar os seus sinais nas figuras 6.2 e 6.3. Em ambas as figuras verificou-se que o período do PWM é de 5 ms como foi previamente configurado e os tempos correspondentes ao nível lógico "1"e "0"também correspondiam ao pretendido.

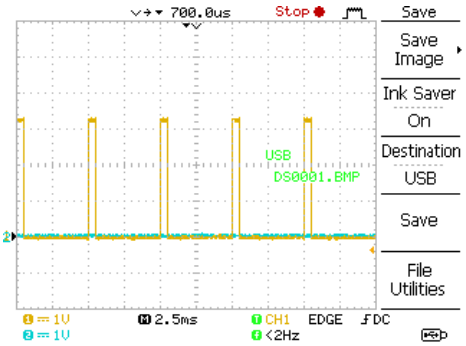


Figura 6.2: PWM com um *duty-cycle* de 10% .

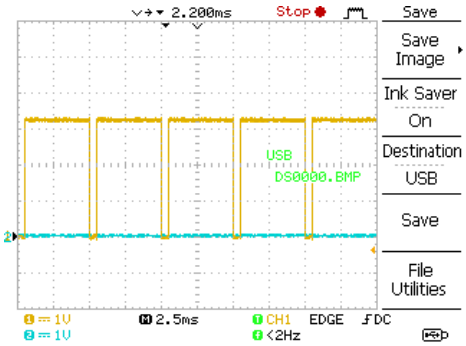


Figura 6.3: PWM com um *duty-cycle* de 90%.

6.3 Teste da memória SDRAM

Para efetuar o teste à memória SDRAM foi usado o método designado de *march test*. O *march test* tem como definição a circulação de um elemento sequencial finito de operações de escrita e leitura aplicada a uma célula ou uma "word", seja aumentando ou diminuindo a ordem de endereços [38]. Existem vários algoritmos associados ao *march test*. Quanto mais operações de escrita e leitura contenha o algoritmo mais robusto é o teste. Contudo, o tempo de teste torna-se muito demorado.

Para efetuar o teste à memória SDRAM foi usado um nível de *march test* relativamente simples designado por "March X", que é capaz de verificar todas as linhas de interligação no circuito impresso da SDRAM ao microcontrolador e que a comunicação entre estes está de acordo com o que é pretendido. As várias iterações efetuadas para teste total da SDRAM foram as seguintes [39]:

1. ⇳ (w0);
2. ⇑ (r0, w1);
3. ⇓ (r1; w0);
4. ⇳ (r0).

Como é possível ver na primeira iteração o símbolo ⇳ indica que todos os bits da memória devem ser escritos com um zero sem que qualquer leitura seja previamente necessária.

Na segunda iteração foi efetuada, de uma forma sequencial ascendente, a leitura de apenas um bit de um determinado endereço, verificando que o seu valor era "0"e nessa mesma posição escrito o valor lógico "1". Esta ação foi repetida sequencialmente para todos os bits de cada endereço. Ou seja, sendo a memória de 32 bits, para cada um dos endereços foi necessário ler cada um dos bits e alterar esses mesmos bits sem modificar nenhum outro. Após efetuada a verificação e escrita de todos os bits de um endereço, este foi incrementado e repetido o mesmo procedimento.

Na terceira iteração é feita a ação inversa da anterior, ou seja, para cada bit de cada endereço tem de ser feita uma leitura onde este tem de conter o nível lógico "1"e, em seguida, escreve-se o nível lógico "0". Nesta iteração o primeiro endereço de teste foi o ultimo endereço da memória, sendo este decrementado até ao endereço inicial.

Na ultima iteração é feita uma leitura a cada um dos endereços, sendo que estes têm de conter todos os bits com um nível lógico "0"que corresponde ao valor hexadecimal 0x000000000 (32 bits).

De forma a reduzir o tempo de teste, apenas foram testados alguns dos endereços. O teste foi efetuado desde o endereço 0x0000 até ao endereço 0x0800 sendo apenas ativo um bit de cada vez, ou seja, inicialmente, como já referido, foi usado o registo 0x0000 em seguida o registo 0x0001, até ao último bit ativo 0x08000. Desta forma, foi reduzido o tempo de teste e foi

possível confirmar que todas as interligações da memória com o microcontrolador estavam corretas.

6.4 Teste da memória Flash

Para o uso do teste da memória flash não foi possível implementar o método usado na memória SDRAM, pois é necessário apagar os dados existentes num determinado bloco da memória antes desta ser reescrita. Desta forma, o método usado para teste de ambas as memórias *flash* teve a seguinte sequencia:

- Apagar toda a memória;
- Verificar que o valor em quatro registos aleatórios foram apagados contendo o valor 0xFF;
- Nos quatro registos escolhidos escrever os valores 0x80, 0x40, 0x20 e 0x01;
- Ler de novo os quatro registos e verificar que estes contêm os valores escritos anteriormente;
- Apagar de novo toda a memória;
- Ler uma vez mais os quatro registos e verificar que estes se encontram apagados.

Os valores 0x80, 0x40, 0x20 e 0x01 correspondem apenas a um bit ativo em cada um dos registos, sendo assim possível confirmar que as quatro interligações de dados existentes entre o microcontrolador e a memória estão corretos. Caso todos os passos de teste se verificassem era ativo um LED indicativo de que a memória estava correta. Numa fase posterior do *software*, onde a apresentação de dados no *display* já estava bastante simplificada, foi efetuado um novo teste às memórias flash, igual ao anterior onde foram

registados sequencialmente no *display* os valores lidos da memória (ver figura 6.4).

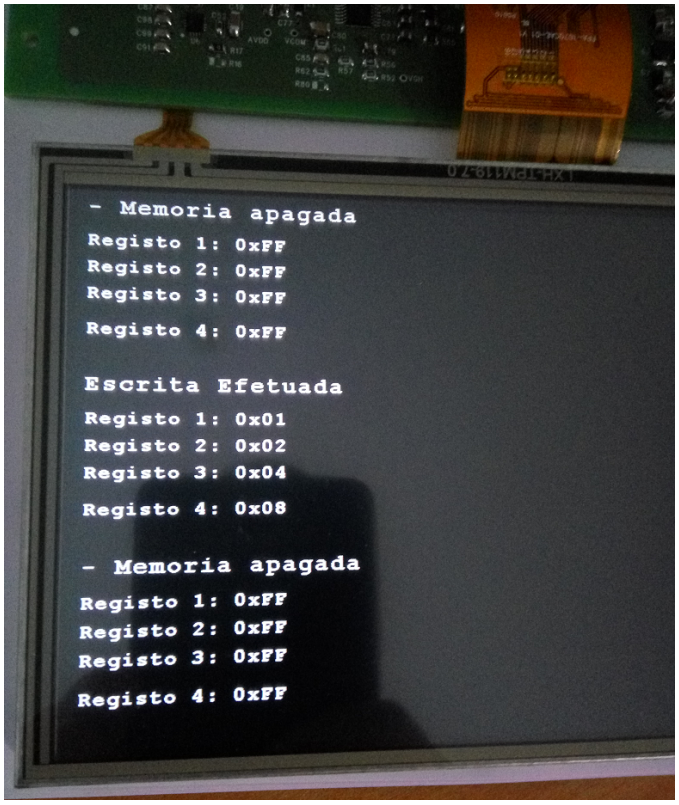


Figura 6.4: Dados obtidos no teste efetuado a uma das memórias flash.

6.5 Sensor touch

O teste efetuado ao sensor *touch* consistiu numa leitura dos eixos X e Y do *display* e a apresentação destes mesmos valores no *display*. Para confirmar que as coordenadas apresentadas no *display* correspondiam ao ponto do ecrã que estava a ser pressionado foram desenhados dois quadrados brancos com uma dimensão de 50 píxeis posicionados nas coordenadas (20,20) e (954,530) (ver figura 6.5).

No primeiro teste efetuado constatou-se que havia um erro significativo

Figura 6.5: Teste efetuado ao sensor *touch*.

nas coordenadas obtidas, erro este que aumentava com a aproximação das bordas do *display*. Assim sendo, verificou-se que uma conversão direta dos valores obtidos do sensor não correspondia à respetiva posição pressionada no ecrã. Para solucionar este problema foram desenhados quatro pequenos quadrados, cada um posicionado em cada canto do *display*, onde foram registados os valores das suas posições e os valores obtidos pelo sensor *touch* quando estes eram pressionados (ver figura 6.6).

		Valores lidos display	Valores desejados
1	X	232	100
	Y	183	100
2	X	238	100
	Y	826	500
3	X	1824	924
	Y	820	500
4	X	1831	924
	Y	184	100

Figura 6.6: Relação valores lidos desejados das coordenadas *touch*

Com base nas expressões 6.1 e 6.2 é possível obter uma relação das coordenadas obtidas com as desejadas, atribuindo valores às constantes a1, b1, c1 e d1 utilizando as expressões 6.3 e 6.4. Nestas expressões é necessário

substituir as variáveis X e Y pelos valores obtidos do sensor, e XDx e YDx pelos valores desejados.

$$Y_D = a_1 Y + b_1[40] \tag{6.1}$$

$$X_D = c_1 X + d_1[40] \tag{6.2}$$

$$\begin{pmatrix} Y_{D1} \\ Y_{D3} \end{pmatrix} = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \begin{pmatrix} Y_1 & 1 \\ Y_3 & 1 \end{pmatrix} [40] \tag{6.3}$$

$$\begin{pmatrix} X_{D2} \\ X_{D4} \end{pmatrix} = \begin{pmatrix} c_1 \\ d_1 \end{pmatrix} \begin{pmatrix} X_2 & 1 \\ X_4 & 1 \end{pmatrix} [40] \tag{6.4}$$

O cálculo destas constantes foi efetuado numa folha de cálculo *excel*, assumindo que os *displays* contêm sempre o mesmo erro. Com estas constantes as expressões iniciais 6.1 e 6.2 podem ser substituídas pelas expressões 6.5 e 6.6.

$$X_D = 0,52X - 22,9 \tag{6.5}$$

$$Y_D = 0,63Y - 15,5 \tag{6.6}$$

Realizando novamente o teste inicial às coordenadas *touch* é possível verificar que os valores obtidos já se encontram dentro dos parâmetros pretendidos. Na figura 6.5 é possível visualizar o valores obtidos e convertidos pelo sensor. Os valores que se seguem ao texto "X:"e "Y:"são referentes aos valores obtidos do sensor, e os valores calculados pelas expressões 6.5 e 6.6

seguem-se ao texto "eixo X:"e "eixo Y:". Desta forma confirma-se que os valores calculados dão origem à coordenada (28,30), que se encontra dentro do quadrado pressionado na posição (20,20) com uma dimensão de 50 píxeis.

6.6 Teste final do sistema

Não havendo uma aplicação final já desenvolvida para o sistema HMI que fosse possível aplicar para fins de teste, foram simulados possíveis comandos da comunicação externa que correspondem a funções que o sistema HMI deverá assumir e executar uma determinada função, tal como aconteceria numa aplicação real.

Numa primeira fase, foram inseridas na memória *flash* duas imagens de fundo, onde uma destas é um logo de apresentação e a outra corresponde a uma camada onde são executadas funções. Ainda no carregamento de imagens foi necessário inserir quatro imagens que correspondem a dois botões, sendo duas destas os botões num estado ativo e as duas restantes os botões num estado desativo.

Na figura 6.7 é possível visualizar a imagem referente ao logo que é apresentada durante dois segundos e em seguida é carregada a figura 6.8 referente à camada de apresentação de funções. Como é possível ver ainda na figura referente à camada de apresentação, esta já contém os dois botões com o *layout* de desativos, que foi a função pré definida.

Figura 6.7: Apresentação da camada inicial.

Figura 6.8: Apresentação de dois pulsadores desativos.

Quando o ecrã do sistema HMI é pressionado as respetivas coordenadas detetadas são registadas e enviadas para o exterior, sempre que o *Master* solicitar esta mesma função. O sistema *Master*, ao receber a informação das coordenadas, analisa-as e se assumir que estas correspondem a algum dos dois botões existentes no ecrã, envia ao sistema HMI a informação para inserir a imagem do botão ativo na respetiva posição. A informação da posição onde se encontra a imagem na memória é enviada pelo *Master* juntamente com a posição onde esta deverá ser carregada no *display* (x0,y0).

Desta forma foi possível testar a comunicação efetuada com o exterior, o protocolo RS485, e as duas funções essenciais do sistema HMI a funcionarem em conjunto, nomeadamente apresentação de imagens no ecrã do *display* a partir de ações efetuadas sobre o painel resistivo.

Ainda usando os dois botões existentes no ecrã foram adicionadas ao *software* do *Master* duas funções a executar quando recebesse a informação que os botões foram pressionados. A função adicionada ao pulsador superior consiste no envio de um *frame* por parte do *Master* para o *display* com o pedido para a tensão de alimentação ser apresentada no *display* numa determinada coordenada. A ação adicionada ao *Master* para o pulsador inferior consiste no pedido de apresentação do valor da temperatura, também esta numa determinada posição do display. Ambas as posições são especificadas pelo *Master* no envio do *frame*. Como é possível ver na figura 6.9, os dois pulsadores encontram-se ativos, sendo também visíveis os valores referentes à tensão de alimentação e temperatura. Quando os pulsadores são pressionados novamente, o *Master* ordena que o sistema carregue ambos os botões com as imagens de desativos, uma vez que estes estavam ativos, e envia dois *frames* para carregar duas imagens, uma a sobrepor o valor da tensão de alimentação e uma outra a sobrepor o valor de temperatura, para desta forma apagar estas mesmas informações do ecrã. As duas imagens carregadas foram adicionadas mais uma vez na memória flash numa posição

conhecida.

Figura 6.9: Apresentação da tensão de alimentação e temperatura no display.

O *software* responsável pelo armazenamento das imagens na memória externa foi incorporado no presente *software* do microcontrolador principal. Contudo, não foi possível efetuar o armazenamento da maneira pretendida, pois o sistema que envia as imagens encontra-se ainda em desenvolvimento por uma outra equipa.

Para os testes efetuados anteriormente, foi necessário armazenar previamente as imagens. Para esse efeito foi desenvolvido um novo *software* responsável por guardar apenas uma imagem num determinado endereço da memória *flash*. Assim, este *software* foi alterado para armazenar as várias imagens. Por último, foi carregado o *sotfware* final com as várias funções implementadas.

Dando por concluídos os testes efetuados ao sistema, segue-se a fase de análise dos resultados obtidos e melhorias a efetuar numa próxima etapa. Estas conclusões são descritas no próximo capítulo.

Esta página foi intencionalmente deixada em branco.

Capítulo 7

Conclusão e Trabalho Futuro

No presente capítulo é feita uma breve análise aos resultados obtidos, tendo como referência os objetivos pretendidos e que implementações podem ser efetuadas para o futuro.

A Ricardo Malheiro, Lda tem como visão a total satisfação dos seus clientes, oferecendo produtos inovadores e personalizados de acordo com as necessidades do cliente.

Ao contrário da tendência do mercado atual, o produto desenvolvido neste projeto utiliza uma resolução de 1024x600 ao invés de 800x480.

Durante o período de desenvolvimento foram feitos estudos/pesquisas aos vários tópicos propostos para este projeto, permitindo que os requisitos especificados inicialmente fossem cumpridos com sucesso.

Como é possível verificar no capítulo 6, Testes e Validações, os valores das tensões presentes nos pinos de alimentação de cada componente e os testes efetuados a cada um dos periféricos, permitiram a confirmação da correta interligação dos vários componentes existentes no circuito impresso.

A interface com módulos externos foi testada numa aplicação real, usando o protocolo RS485. O sistema HMI respondeu corretamente a todos os comandos a que foi submetido. Assim sendo, é possível concluir que todos os requisitos iniciais foram atingidos dentro dos parâmetros pretendidos.

Um dos obstáculos encontrados foi o armazenamento de imagens nas memórias externas. Sem a implementação do envio da informação referente a cada imagem pela interface externa, o armazenamento das imagens através de um software torna-se demasiado lento, aumentando também a possibilidade de erro no local de armazenamento das imagens.

Uma segunda limitação é a forma como o besouro se encontra posicionado na parte de trás do módulo, pois o som produzido em pequenos "BIPs"(quando os botões são pressionados), pode não ser audível em algumas aplicações. Uma solução possível para este problema, será a implementação de um motor de vibração. Desta forma, quando um dos botões for pressionado, o utilizador sentirá uma pequena vibração, além do som produzido pelo besouro.

Para um trabalho futuro, sugere-se a implementação de uma interface *wireless* ou um cartão de memória. Estes não seriam destinados ao controlo do sistema, mas poderiam ser uma mais valia no envio de configurações e armazenamento de dados.

Interfaces CAN, leitura GPS, sensor de proximidade, acelerómetros, e outras tecnologias já integram vários módulos na empresa Ricardo Malheiro, Lda. No entanto, a implementação destas no sistema HMI poderia tornar o produto mais versátil para novas aplicações.

Bibliografia

[1] STMicroelectronics, "AN4860 Application note - DSI Host on STM32F469/479, STM32F7x8/x9 and STM32L4R9/S9 MCUs," 2017. [Online]. Disponível em: `https://www.st.com/resource/en/application_note/dm00287601.pdf`. [Acedido em: 28-Jun-2019].

[2] Li Du, "An Overview of Mobile Capacitive Touch Technologies Trends." [Online]. Disponível em: `https://arxiv.org/pdf/1612.08227`. [Acedido em: 28-Jun-2019].

[3] STMicroelectronics, "TFT LCD interfacing with the high-density STM32F10xxx FSMC," 2008. [Online]. Disponível em: `https://www.st.com/content/ccc/resource/technical/document/application_note/85/ad/ef/0f/a3/a6/49/9a/CD00201397.pdf/files/CD00201397.pdf/jcr:content/translations/en.CD00201397.pdf`. [Acedido em: 28-Jun-2019].

[4] Displaytech Ltd., "TFT LCD Module Product Specification - DT070BTFT-HB-TS," 2016. [Online]. Disponível em: `https://cdn.displaytech-us.com/sites/default/files/display-data-sheet/DT070BTFT-HB-TS-Displaytech-Spec.pdf`. [Acedido em: 28-Jun-2019].

[5] Apple Inc, Image Size and Resolution. [Online]. Disponível em: `https://developer.apple.com/design/human-interface-guidelines/`

ios/icons-and-images/image-size-and-resolution/. [Acedido em: 28-Jun-2019].

[6] STMicroelectronics, "AN4861 Application note - LCD-TFT display controller (LTDC) on STM32 MCUs," 2017. [Online]. Disponível em: https://www.st.com/content/ccc/resource/technical/document/application_note/group0/25/ca/f9/b4/ae/fc/4e/1e/DM00287603/files/DM00287603.pdf/jcr:content/translations/en.DM00287603.pdf. [Acedido em: 28-Jun-2019].

[7] Mauricio Jancic and César Angel Fuoco, "Librerias Graficas con PIC32," 2011. [Online]. Disponível em: http://www.sase.com.ar/2012/files/2012/09/Librerias-Graficas-con-PIC32.pdf. [Acedido em: 28-Jun-2019].

[8] HantouchUSA, "How it works: Touch Screen Specialists 4-Wire Analog ..." [Online]. Disponível em: https://www.sparkfun.com/datasheets/LCD/HOWDOESITWORK.pdf. [Acedido em: 28-Jun-2019].

[9] eizoglobal, "How can a screen sense touch? A basic understanding of touch panels," EIZO. [Online]. Disponível em: http://www.eizoglobal.com/library/basics/basic_understanding_of_touch_panel/. [Acedido em: 28-Jun-2019].

[10] EPSON, " LCD Controller with External VRAM." [Online]. Disponível em: https://global.epson.com/products_and_drivers/semicon/products/display_controllers/pdf/spec_lcd.pdf. [Acedido em: 28-Jun-2019].

[11] EPSON, "S1D13517 Display Controller," 2009. [Online]. Disponível em: http://www.epson.com.cn/ed/lcdc/pdf/13517/x92aa001.f01(S1D13517SpecRev1.0).pdf. [Acedido em: 28-Jun-2019].

[12] Altera Corporation, "Board Design Guidelines for LVDS Systems," 2010. [Online]. Disponível em: `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp_lvdsboard.pdf`. [Acedido em: 28-Jun-2019].

[13] STMicroelectronics, "STM32F745xx STM32F746xx - ARM®-based Cortex®-M7 32b MCU FPU, 462DMIPS, up to 1MB Flash/320 16 4KB RAM, USB OTG HS/FS, ethernet, 18 TIMs, 3 ADCs, 25 com itf, cam & LCD," 2016. [Online]. Disponível em: `https://www.st.com/content/ccc/resource/technical/document/datasheet/96/ed/61/9b/e0/6c/45/0b/DM00166116.pdf/files/DM00166116.pdf/jcr:content/translations/en.DM00166116.pdf`. [Acedido em: 28-Jun-2019].

[14] Microchip, "SuperFlash® Memory Products - High Performance, Low Power Consumption and Superior Reliability." [Online]. Disponível em: `http://ww1.microchip.com/downloads/en/DeviceDoc/25072c.pdf`. [Acedido em: 28-Jun-2019].

[15] Microchip Technology Inc, "SST26VF064B / SST26VF064BA - 2.5V/3.0V 64 Mbit Serial Quad I/O (SQI) Flash Memory," 2015. [Online]. Disponível em: `http://ww1.microchip.com/downloads/en/DeviceDoc/20005119G.pdf`. [Acedido em: 28-Jun-2019].

[16] Integrated Silicon Solution, Inc, "IS42S32400F IS45S32400F - ISSI," 2015. [Online]. Disponível em: `http://www.issi.com/WW/pdf/42-45S32400F.pdf`. [Acedido em: 28-Jun-2019].

[17] Texas Instruments Incorporated, "DS90C185 Low Power 1.8V FPD-Link(LVDS) Serializer," 2013. [Online]. Disponível em: `http://www.ti.com/lit/ds/symlink/ds90c185.pdf`. [Acedido em: 28-Jun-2019].

[18] STMicroelectronics, "STMPE811 - S-Touch® advanced resistive touchscreen controller with 8-bit GPIO expander," 2011. [Online]. Disponível em: `https://media.digikey.com/pdf/DataSheets/STMicroelectronicsPDFS/STMPE811.pdf`. [Acedido em: 28-Jun-2019].

[19] Alipta Ballav and Dr. Mainak Ghosh, "Human Factors Of Human Machine Interaction: Analyzing Future Trends Through The Past And The Present," 2017. [Acedido em: 28-Jun-2019].

[20] JDI, "LCD Basics," Technology : LCD Basics — Japan Display Inc. [Online]. Disponível em: `http://www.j-display.com/english/technology/lcdbasic.html`. [Acedido em: 28-Jun-2019].

[21] Bhrijesh N. Patel and Mrugesh M. Prajapati, "OLED: A Modern Display Technology," 2014. [Online]. Disponível em: `http://www.ijsrp.org/research-paper-0614/ijsrp-p3080.pdf`. [Acedido em: 28-Jun-2019].

[22] Ilya N. Rodionov, Igor V. Nesterenko, Dmitriy V. Telyshev, and Ivan .A. Sapozhkov, "Display Interfaces for the Control Unit of an Implantable Cardiac Pump," 2018. [Online]. Disponível em: `https://publications.rwth-aachen.de/record/723572/files/723572.pdf`. [Acedido em: 28-Jun-2019].

[23] "Tamanho e resolução da imagem." [Online]. Disponível em: `https://paginas.fe.up.pt/~ee03037/tmp/api11/imagem_resol.pdf`. [Acedido em: 28-Jun-2019].

[24] Microchip Technology Inc, "AN1368, Developing Embedded Graphics Applications using PIC® Microcontrollers with Integrated Graphics Controller," 2011. [Online]. Disponível em: `http://ww1.microchip.com/downloads/en/AppNotes/01368a.pdf`. [Acedido em: 28-Jun-2019].

[25] Texas Instruments Incorporated, "Using resistive touch screens for human/machine interface," 2005. [Online]. Disponível em: `http://www.ti.com/lit/an/slyt209a/slyt209a.pdf`. [Acedido em: 28-Jun-2019].

[26] Li Du, "An Overview of Mobile Capacitive Touch Technologies Trends." [Online]. Disponível em: `https://arxiv.org/pdf/1612.08227`. [Acedido em: 28-Jun-2019].

[27] Cypress Semiconductor, "Touchscreens 101: Understanding Touchscreen Technology and Design," 2009. [Online]. Disponível em: `https://www.cypress.com/file/95156/download`. [Acedido em: 28-Jun-2019].

[28] "Integrated Graphics Acceleration and DRAM," PIC32MZ DA Family — Microchip Technology. [Online]. Disponível em: `http://www.microchip.com/design-centers/32-bit/pic-32-bit-mcus/pic32mz-da-family`. [Acedido em: 28-Jun-2019].

[29] "Considerations for Selecting an RS-485 Transceiver in Electronic Power Meters," 2006. [Online]. Disponível em: `http://www.maximintegrated.com/en/app-notes/index.mvp/id/3884`, ultimo acesso 6 Maio 2018. [Acedido em: 28-Jun-2019].

[30] "Local Interconnect Network - univasf.edu.br." [Online]. Disponível em: `http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Protocolo-LIN.pdf`. [Acedido em: 28-Jun-2019].

[31] "Mentor, a Siemens Business," Mentor Graphics. [Online]. Disponível em: `https://www.mentor.com/company/`. [Acedido em: 28-Jun-2019].

[32] Ac6, "HomePage," OpenSTM32 Community Site. [Online]. Disponível em: `http://www.openstm32.org/`. [Acedido em: 28-Jun-2019].

[33] Brian W. Kernighan and Dennis M. Ritchie, "The C Programming Language." [Online]. Disponível em: `https://dipmat.univpm.it/~demeio/public/the_c_programming_language_2.pdf`. [Acedido em: 28-Jun-2019].

[34] Texas Instruments, "RS-485 Reference Guide ," 2014. [Online]. Disponível em: `http://www.ti.com/lit/sg/slyt484a/slyt484a.pdf`. [Acedido em: 28-Jun-2019].

[35] Maxim Integrated, "DS18B20 Prorammale Resoltion 1-Wire Diital Thermometer," 2018. [Online]. Disponível em: `https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf`. [Acedido em: 05-28-Jun-2019].

[36] Lida Optical&Electronic Co., LTD., "CdS Photoconductive Cells GL5528 - Pi Gate." [Online]. Disponível em: `https://pi.gate.ac.uk/pages/airpi-files/PD0001.pdf`. [Acedido em: 28-Jun-2019].

[37] EAST, Piezo Buzzer." [Online]. Disponível em: `https://www.promelec.ru/pdf/efm-250.pdf`. [Acedido em: 28-Jun-2019].

[38] Rob Dekker, Frans Beenker, IEEE, and Loek Thijssen, "A Realistic Fault Model and 567 Test Algorithms for Static Random Access Memories," 1990. [Online]. Disponível em: `http://www.ecs.umass.edu/ece/ece654/ECE654HomePageFiles/Lectures/dekker.pdf`. [Acedido em: 28-Jun-2019].

[39] A.J. van de Goor and I.B.S Tlili, "March tests for word-oriented memories." [Online]. Disponível em: `https://www.cs.york.ac.uk/rts/docs/SIGDA-Compendium-1994-2004/papers/1998/date98/pdffiles/06d_3.pdf`. [Acedido em: 28-Jun-2019].

[40] "TN0074 Technical note - Calibration procedure for a resistive touchscreen system based on the STMPE811," 2011. [Online]. Disponível em:

`https://www.ercankoclar.com/wp-content/uploads/2015/05/Calibration-procedure-for-a-resistive-touchscreen-system-based-on-the-STMPE811.pdf`. [Acedido em: 28-Jun-2019].