Lino Flávio Monteiro Dias

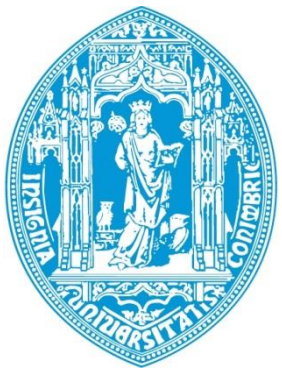
Smart Switch

Dissertação de Mestrado

Coimbra 2014

· U C ·

Universidade de Coimbra



Departamento de Engenharia Electrotécnica e de Computadores
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Smart Switch

Lino Flávio Monteiro Dias

Orientador: Professor Doutor António Paulo Mendes Breda Dias Coimbra

Júri:
Presidente: Professor Doutor Manuel Marques Crisóstomo
Vogais: Professor Doutor António Paulo Mendes Breda Dias Coimbra
Professor Doutor Marco Alexandre Cravo Gomes

Coimbra, Setembro 2014

Agradecimentos

Apesar de devido à sua finalidade académica uma dissertação seja um trabalho individual, existem contributos que têm de ser realçados. Como tal, desejo expressar os meus sinceros agradecimentos:

Ao Professor Doutor António Paulo Mendes Breda Dias Coimbra, pela sua competência científica, aconselhamento e disponibilidade durante a orientação desta dissertação.

O meu agradecimento à minha esposa Tânia e filha Maria, por serem o meu suporte, motivação e inspiração durante a realização desta dissertação.

Os meus agradecimentos aos meus pais e sogros pela motivação e incentivo.

À minha avó Teresa que me educou e nunca me deixou desistir desta meta.

Aos meus primos Nuno e Tiago pela disponibilidade demonstrada.

Aos meus amigos Emanuel Martins e Ricardo Faustino por estarem sempre disponíveis em me ouvir e por todo o apoio que me deram.

Resumo

Os equipamentos electrónicos desempenham um papel extremamente importante no nosso quotidiano. Existe cada vez mais a necessidade de realizar o mínimo esforço em actividades diárias e rotineiras, deixando essa tarefa e gestão para sistemas inteligentes.

O trabalho realizado nesta dissertação teve como objectivo principal a criação de um dispositivo electrónico enquadrado no âmbito da domótica, possível de controlar remotamente e de fácil integração em circuitos convencionais já projectados e implementados.

Foi elaborado um dispositivo com um teclado de toque (*touch control*) que comunica via Rádio Frequência (RF), tem a possibilidade de fazer *dimming* em sistemas de iluminação e controla estores ou qualquer outra carga (resistiva ou indutiva) não superior a 400W.

Juntamente com o dispositivo foi criada uma interface com o utilizador (programada no programa Monodevelop), aplicada a um sistema de iluminação com 2 lâmpadas que comunica através da porta USB de um computador e que pode ser compilada em diferentes sistemas operativos.

Foram testados experimentalmente 2 sistemas "*Smart Switch"* com 2 lâmpadas cada um, onde as lâmpadas eram actuadas através da interface com o utilizador compilada para Windows.

Palavras Chave:

Domótica, *touch* control, módulo de RF, *dimming*

Abstract

Electronic devices play an extremely important role in our daily lives. There is a need to make the minimum effort in daily and routine activities leaving this task and management for intelligent systems.

The main goal of this dissertation was the creation of an electronic device that can be fitted in home automation (domotics) that could be controlled remotely and be easily integrated in conventional installations already designed and implemented.

It was designed a device with a touch control that communicates by radio frequency (RF) and has the ability to make dimming for lighting systems and control blinds or any charges (resistive or inductive) that not exceed 400W.

Along with the device was created a user interface (programmed with Monodevelop) applied to a lighting system with 2 lamps that communicate by USB port of a computer and can be compiled in several operating systems.

2 Smart Switch systems were tested with 2 lamps each controlled by the user interface developed and compiled for Windows.

Key Words:

Home Automation, touch control, RF module, dimming

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Esta dissertação teve como motivação desenvolver um interruptor de luz doméstico com novas capacidades adicionais relativamente aos vulgares interruptores de luz mecânicos, nomeadamente: ser capaz de suportar comando por toque (*touch control*) sem botão mecânico; actuar como *dimmer* de luz (regular a intensidade da luz), ou outras cargas (ex. estores eléctricos); substituir os existentes interruptores simples, de escadas, inversores e interruptores actuados por movimento; ter a capacidade de ser controlado remotamente via Rádio Frequência (RF) e que fosse de fácil implementação em instalações existentes.

Foi feita uma pesquisa na Internet sobre os interruptores existentes no mercado e suas funcionalidades e verificou-se que havia uma lacuna: Há interruptores *touch control* para comando de cargas, mas que não têm comunicação por RF (exemplo ilustrado na Figura 1); Existem também interruptores *touch control* com comunicações RF, mas que não possuem o controlo da carga no mesmo módulo. São enviadas mensagens por RF para uma base que se encontra no quadro eléctrico da habitação que gere o controlo da iluminação (exemplo ilustrado na Figura 2). Estes equipamentos são interessantes quando o seu uso/instalação é previsto na fase de construção de uma habitação.

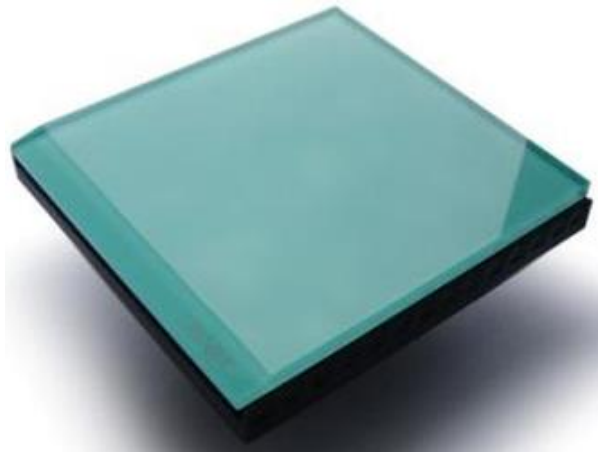


Figura 1 – Interruptor PLCBUS (R 2220HE).

Figura 2 – Interruptor Szhoma (hlc208).

1.1. Objectivos

Foram definidos os seguintes objectivos para esta dissertação:

- Desenvolver o *Hardware* para um interruptor *touch control* para controlo de luminárias e estores com os seguintes requisitos:
  - *Dimming* para sistemas de iluminação
  - Sensor de luminosidade
  - Sensor de movimento
  - Comunicação por RF
  - Comando de 2 cargas
  - 6 Teclas, 3 teclas para cada carga (ver Figura 3)
  - Montagem numa caixa de aparelhagem eléctrica standard

- Desenvolver o *Firmware* para o interruptor inteligente para comando de luminárias;

- Desenvolver uma interface no computador (PC) que permita via RF actuar os interruptores inteligentes.

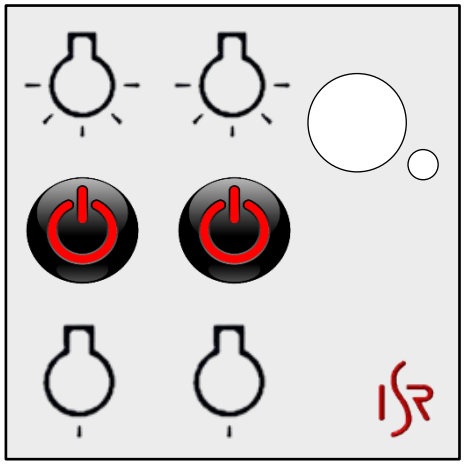


Figura 3 – Desenho do interruptor desenvolvido.

1.2. Contribuições

Foi desenvolvida uma Placa de circuito Impresso (PCB) que contém 3 módulos: o de teclado; o de interface e controlo e o de potência. A dimensão destes módulos estava limitada ao tamanho de uma caixa de aparelhagem eléctrica standard.

Para o módulo de teclado foi feito um estudo sobre dimensionamento das teclas *touch*, e foi desenhado o *footprint* da tecla tendo em conta esse estudo e o espaço físico disponível [12][13][14].

Para o módulo de interface e controlo foram feitas pesquisas para os componentes a usar, de modo a escolher aqueles que fossem mais baratos. Depois de escolhidos, foram desenhadas todas as suas *pads* individualmente para depois construir os seus *footprints*.

No módulo de potência foram também escolhidos os componentes mais baratos, foram desenhadas as suas *pads* e posteriormente o seu *footprint.*

Para desenhar as pistas e planos de todos os módulos foi necessário estudar cada componente individualmente. Por exemplo, no módulo de RF é de boa prática não colocar muitas pistas na zona da sua antena, pois elas vão tirar performance ao módulo.

No desenvolvimento do *firmware* foi necessário dominar vários módulos do PIC24: conversor analógico para digital (ADC); *Charge Time Measurement Unit* (CTMU); UART; *Serial Peripheral Interface* (SPI); vários tipos de interrupções e remapeamento de pinos. Foi necessário adaptar as librarias da CTMU e do módulo de RF para o *hardware* elaborado. Foi necessário criar dois protocolos de comunicação: um para os *Smart Switch* e outro para o módulo de controlo dos *Smart Switch* conectado ao PC.

Os componentes para os 3 módulos foram montados manualmente.

Por fim, para a interface para PC desenvolvida foi necessário aprender C#, e aprender como aceder às portas USB do PC onde está ligado o módulo que comanda os *Smart Switch.*

1.3. Organização da Dissertação

A Dissertação encontra-se organizada da seguinte forma:

- O capítulo 2 descreve o desenvolvimento de todo o Hardware e Firmware elaborado;
- O capítulo 3 apresenta os testes e resultados experimentais realizados;
- O capítulo 4 tece algumas conclusões finais e trabalho futuro.

Capítulo 2

Desenvolvimento do Hardware e Firmware

O PCB desenvolvido encontra-se ilustrado na Figura 5 e Figura 6. Este é composto por 3 módulos (Figura 4):

- Módulo de teclado com 6 teclas *touch control*;
- Módulo de interface e controlo com um microprocessador (PIC24), um módulo de RF, um NodeID (identificador para uma rede RF), um sensor de luminosidade, um sensor de movimento, 4 LEDs e comunicação com o PC;
- Módulo de potência que gere a alimentação para todos os módulos e controlo das 2 saídas para lâmpadas ou estores.

Figura 4 – Módulos do PCB desenvolvido.

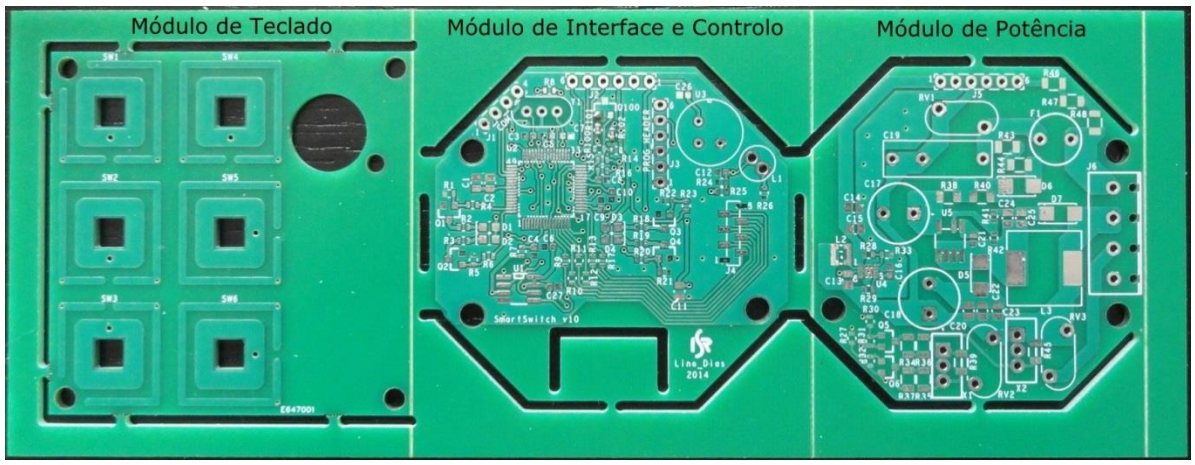


Figura 5 – PCB desenvolvido (lado de cima).

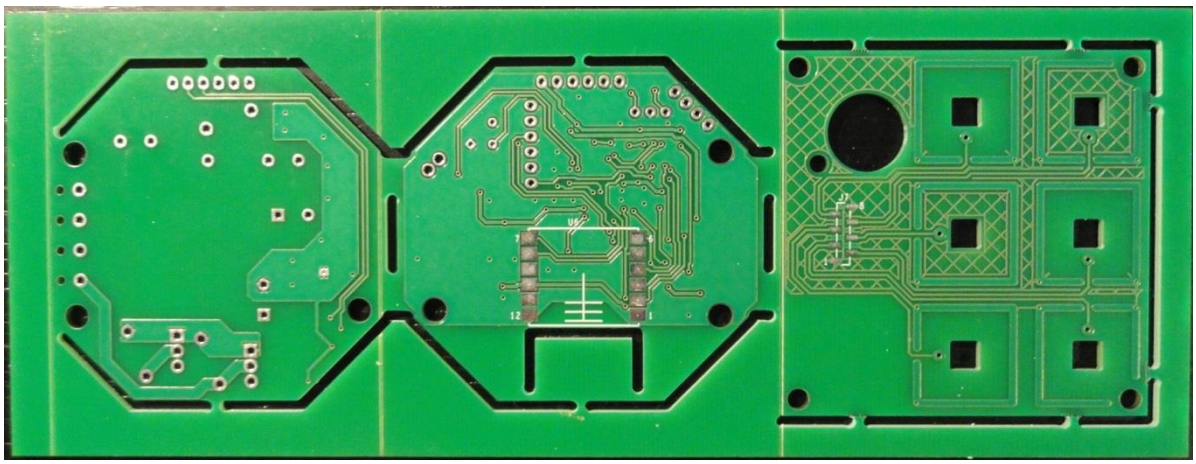


Figura 6 – PCB desenvolvido (lado de baixo).

Para redução de custos, os 3 módulos foram projectados num só painel com pontos de quebra entre os módulos (*Scoring*). Depois de montados os componentes, cada módulo foi separado dos restantes nos pontos de quebra projectados para o efeito. Os módulos de potência e interface são ligados electricamente por um *header* e, mecanicamente, por espaçadores (ver Figura 7).

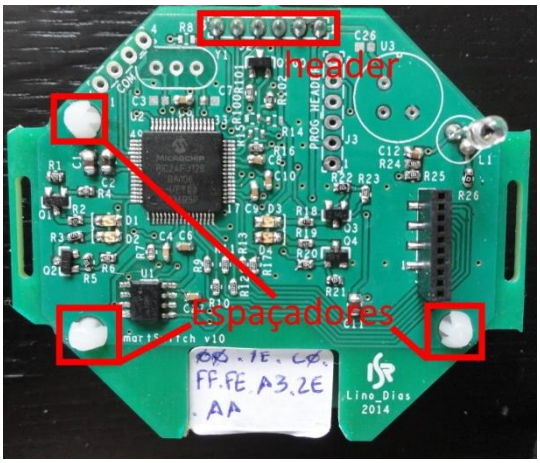


Figura 7 – Localização do header e espaçadores no módulo de interface e controlo.

Foram mandados fabricar 3 PCBs. Num dos PCBs a parte de interface está a ser usada para comunicações com um PC, tirando partido da UART do PIC24. Nos 2 sistemas restantes usados para controlar cargas, a UART não é utilizada [16].

Para desenvolver o hardware foram utilizados os seguintes programas:

- OrCAD 16 – *Capture* – Para o desenho do esquemático;
- OrCAD 16 - *PCB Editor* – Para o desenho do PCB, e desenho dos footprints de todos os componentes;
- OrCAD 16 - *Pad Designer* – Para o desenho das Pads para os componentes.

Para desenvolvimento do Firmware foi utilizado:

- Linguagem C;
- Programa MPLAB X (compilador C30) para a programação do microcontrolador PIC24FJ128GA106 da Microchip, que é um ambiente integrado de desenvolvimento de software em C (linguagem usada), Assembly, programação e debugging do código desenvolvido [15].

Para simulação de circuitos foi usado o programa Proteus 8.

2.1. Módulo de teclado

Este módulo é composto por seis teclas, sendo cada tecla um sensor capacitivo (Figura 8). Foi escolhido usar o PIC24 pois era, da arquitectura de 16 bits, o mais barato. Para a leitura dos sensores foi usada a CTMU (*Charge Time Measurement Unit*) do PIC24 [9][10][11].

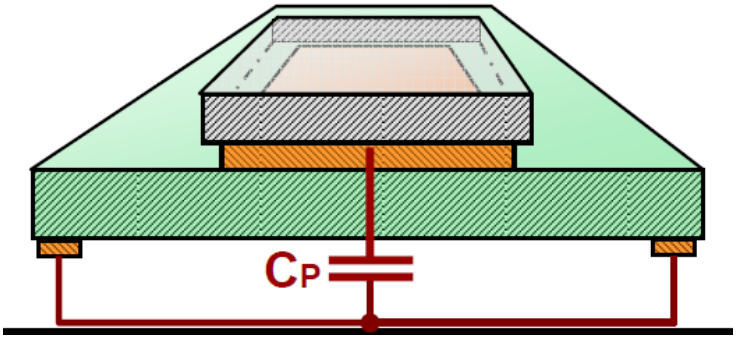


Figura 8 – Exemplo de um sensor capacitivo (Cp é a capacidade do sensor).

Quando configuramos a CTMU temos de definir qual das fontes de corrente constante vamos utilizar (0.55µA, 5.5µA ou 55µA) para carregar o condensador, e quanto tempo vai durar esse carregamento. Depois do condensador ser carregado pela fonte de corrente, vamos ler a tensão que está em cada uma das teclas através do ADC.

A Figura 9 ilustra a interacção entre os módulos ADC e CTMU do PIC24. O condensador C_{AD} representa o condensador usado pelo módulo de ADC para fazer as conversões.

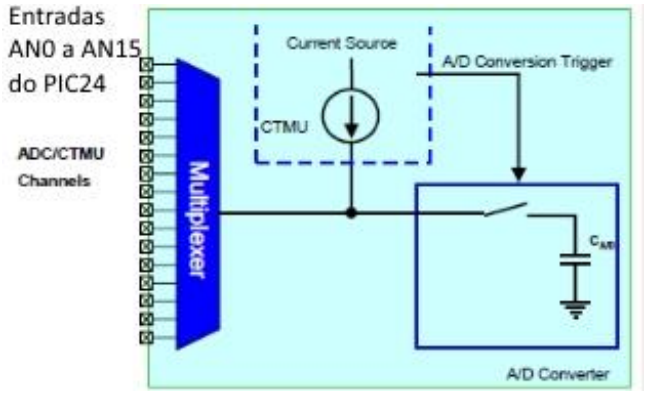


Figura 9 – Interacção dos módulos ADC e CTMU do PIC24.

Quando escolhemos os canais (AN0 a AN15) onde estão ligadas as teclas, o multiplexer do PIC24 conecta-as directamente à CTMU (sequencialmente) e faz logo a aquisição e conversão das amostras pelo ADC, uma vez que a CTMU tem um trigger para o ADC.

Seguidamente, vai ser explicado de que forma é possível usar a CTMU para aplicações teclas *touch* usadas como sensores capacitivos:

- A corrente instantânea num condensador é dada pela expressão:

$$I = C.\frac{dV}{dt} \qquad (1)$$

- Se a corrente for constante, podemos integrar a equação (1)

$$I = C.\frac{V}{t} \qquad (2)$$

- Retrabalhando a equação (2) chegamos a esta equação:

$$\frac{I.t}{C} = V \qquad (3)$$

Significa então que se um condensador for carregado com uma corrente constante "I" durante um período de tempo fixo "t", o valor da Capacidade "C" é inversamente proporcional ao da tensão "V". Quando tocamos com o dedo no sensor capacitivo estamos a introduzir uma capacidade em paralelo (C_F) com a capacidade no sistema, (C_P), como mostra a Figura 10.

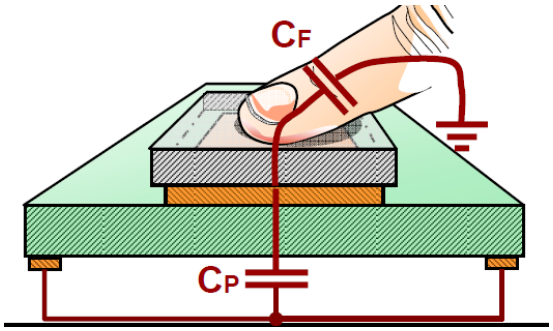


Figura 10 – Efeito capacitivo provocado pelo toque de um dedo no sensor.

Tipicamente a capacidade do sensor pertence à gama entre 30pF e 100pF, e a capacidade do corpo humano entre 7pF e 20pF [19]. Então quando introduzimos um dedo no sensor, no mínimo iremos ter uma capacidade de 37pF. Estipulando então um tempo de carga de 10µs ("t") e usando uma corrente constante de 5.5µA, pela equação (3) verifica-se o seguinte:

- Quando o sistema não é tocado com o dedo

$$\frac{I.t}{C} = V \Leftrightarrow \frac{5.5\mu * 10\mu}{30p} = V \Leftrightarrow V = 1.833v$$

- Quando o sistema é tocado com o dedo

$$\frac{I.t}{C} = V \Leftrightarrow \frac{5.5\mu * 10\mu}{37p} = V \Leftrightarrow V = 1.486v$$

A forma de onda típica do módulo CTMU é a que está ilustrada na Figura 11 [11].

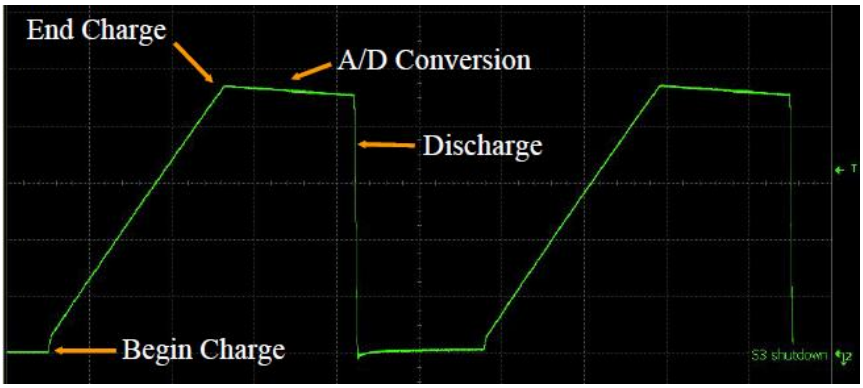


Figura 11 – Forma de onda típica da CTMU.

Através da medição da tensão é possível que o firmware determine se uma tecla foi premida ou não. A Figura 12 representa os níveis de tensão para tecla livre (*Not Touched*) ou tecla pressionada (*Touched*) [11].

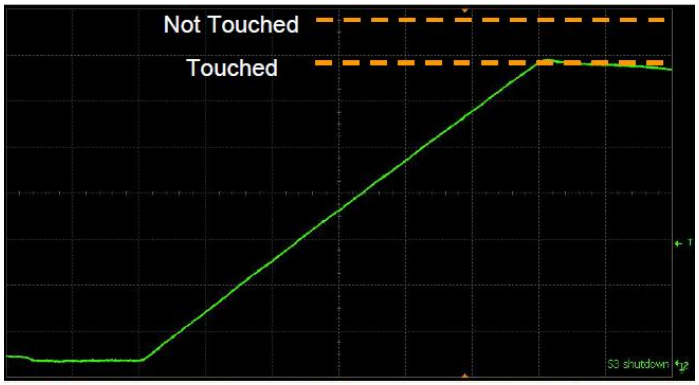


Figura 12 – Níveis de tensão para tecla livre e tecla pressionada.

O firmware desenvolvido para o módulo de teclado foi organizado em dois ficheiros, o "InitTeclado.c" e o "EstadoTeclado.c" (as funções encontram-se no Apêndice C). Na Figura 13 está ilustrado o fluxograma relativo às funções que estão no ficheiro "InitTeclado.c".

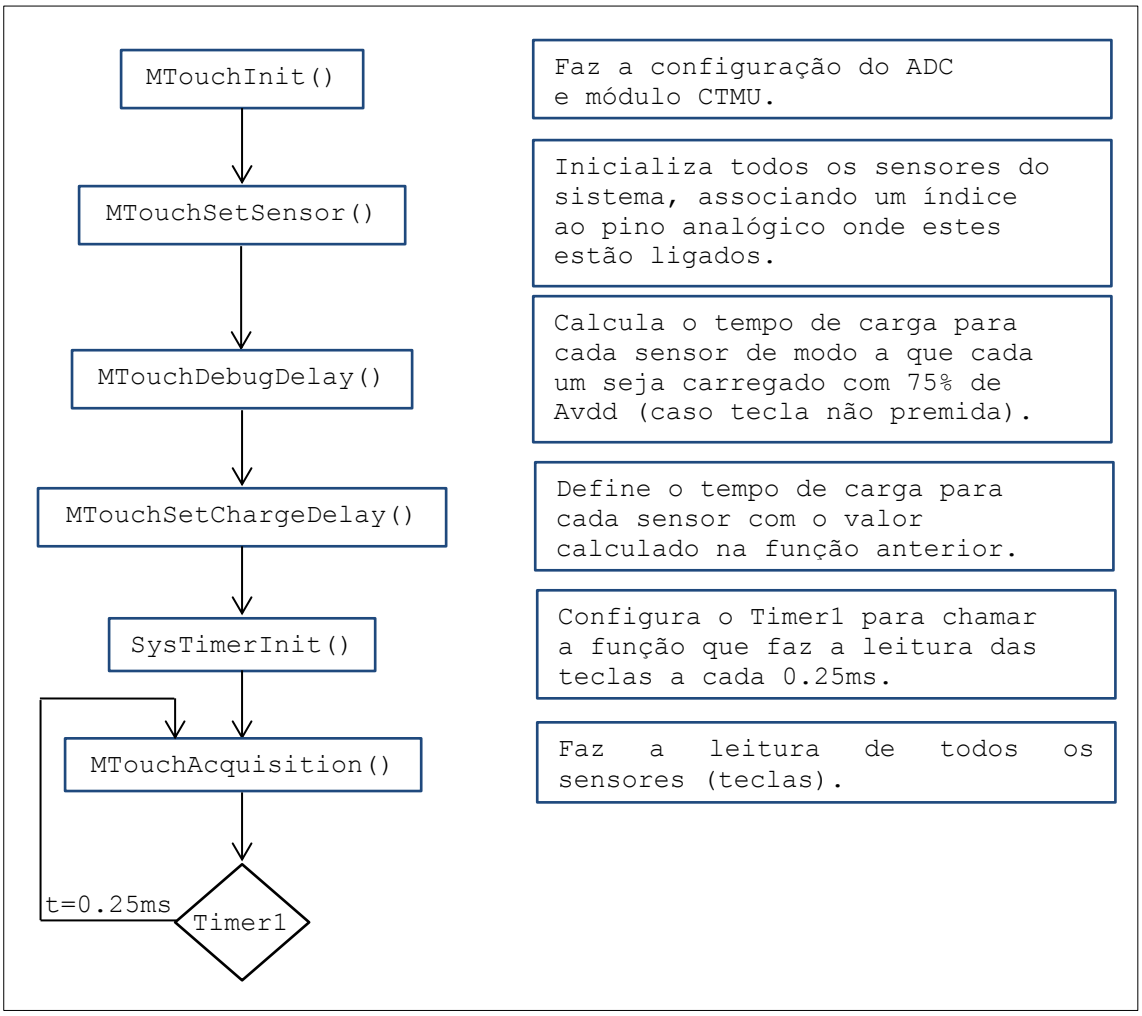


Figura 13 – Fluxograma da função InitTeclado() existente no ficheiro “InitTeclado.c”.

O fluxograma ilustrado na Figura 14 representa as funções que foram criadas no ficheiro "EstadoTeclado.c".

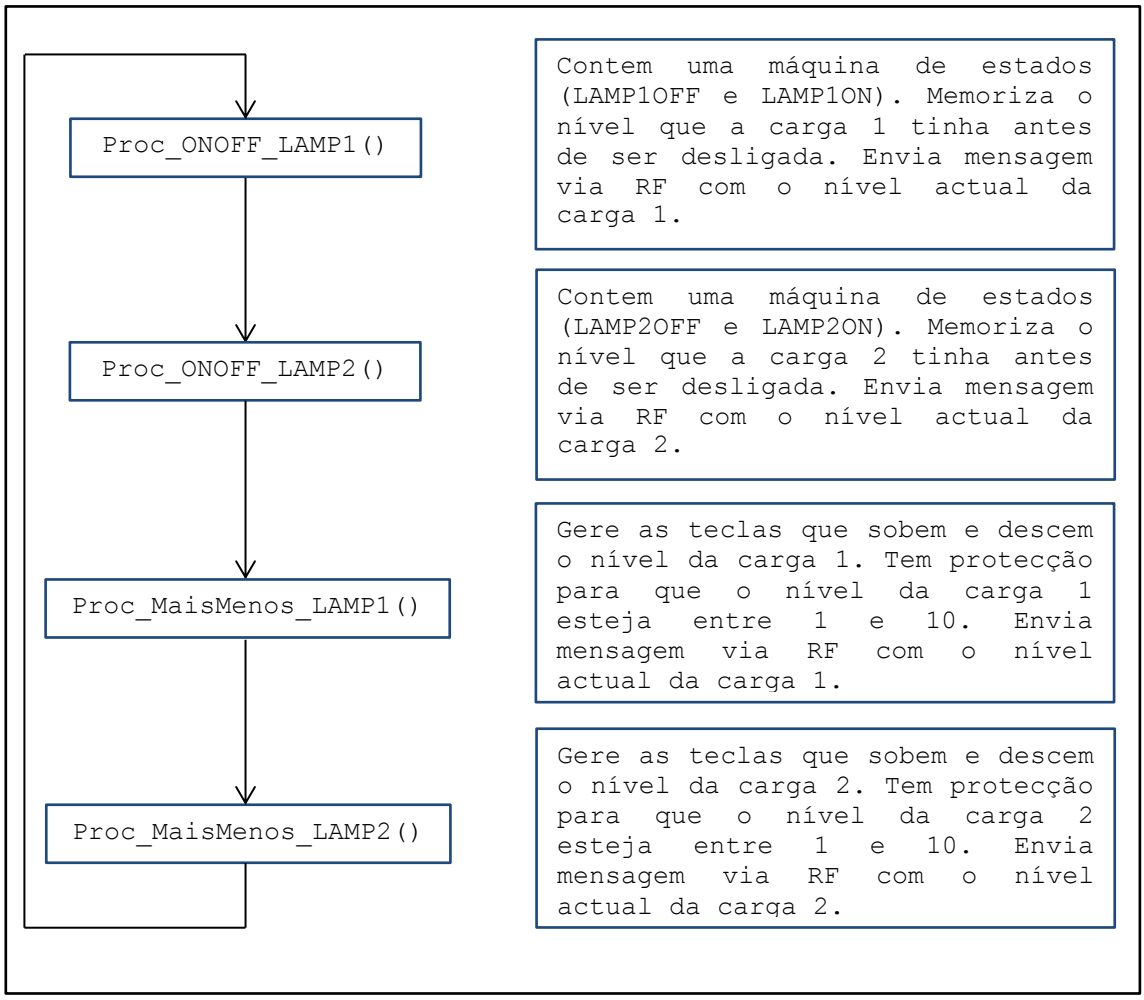


Figura 14 – Fluxograma das funções existentes no ficheiro "EstadoTeclado.c".

2.2. Módulo de interface

Este módulo é constituído pelos seguintes componentes:

- Microprocessador PIC24FJ128GA106 da Microchip. Foi usado o seu oscilador interno, a CTMU, o ADC, o SPI e a UART [6];
- Módulo de RF da Microchip MRF24J40MA;
- EEPROM de 2k com NodeID da Microchip 25AA02E48;
- Foto-transístor da Vishay TEPT4400;
- Sensor de movimento (Passive Infrared Sensor (PIR)) da Panasonic EKMC1601111.

2.2.1. Módulo RF

Este módulo suporta os protocolos de comunicação ZigBee®, MiWi™, MiWi™ P2P e protocolos não standard ("*Proprietary Protocols"*). Possui um cristal integrado, um regulador de tensão interno, um módulo de gestão e uma antena com um alcance até 120m. De notar, que este módulo RF está certificado pela *Federal Communications Commission* (FCC) dos Estados Unidos da América, pela *Industry Canada* (IC) e pelo *European Telecommunications Standards Institute* (ETSI) [7][8][18].

O módulo de RF seleccionado está ilustrado na Figura 15.

Figura 15 – Módulo de RF MRF24J40MA.

O protocolo usado foi o MiWi™ P2P [8], topologia "*peer-to-peer"*, pela sua simplicidade de implementação. Nesta topologia não é necessário que haja um emparelhamento entre os vários sistemas, basta apenas saber a identificação de cada um (através do *NodeID*) para transmitir a mensagem ("*Pay Load*"). Na **Tabela 1** está exemplificado o formato do pacote da topologia MiWi™ P2P.

Bytes	2	1	2	2/8	0/2	8	Variável	2
	Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Pay Load	Frame Check Sequence

Tabela 1 – Pacote da topologia MiWi™ P2P.

Quando queremos enviar uma mensagem para um destino específico (mensagem unicast) só nos temos de preocupar com o endereço do destino ("*Destination Address"*) e com a mensagem a enviar ("*Pay Load"*), pois a Microchip fornece uma biblioteca que se encarrega do restante.

Foi necessário criar um protocolo de comunicação na mensagem *Pay Load* para que os sistemas soubessem que acção tomar.

O protocolo para a variável *Pay Load* está descrito na Tabela 2.

	Pay Load	
	Byte 1	Byte 2
Carga 1	0x01	0 - 10
Carga 2	0x02	

Tabela 2 – Protocolo comunicação da mensagem *Pay load* para a carga 1.

Foram definidos 2 bytes para a mensagem Pay Load porque se quisermos acrescentar parâmetros no futuro o protocolo continua funcional. No byte 1 (rxMessage.Payload[0] do Quadro 1) é definido qual das cargas é actuada, e no byte 2 (rxMessage.Payload[1] do Quadro 1) define-se o nível da carga. O Quadro 1 mostra a função "HandleRfMsg()" implementada para gerir este protocolo.

```
void HandleRfMsg(void){

  if( MiApp_MessageAvailable() ){

    switch(rxMessage.Payload[0]){
      case 1: // carga 1
        if ( rxMessage.Payload[1] <= Nivelmaxcargas ){
          nivelcarga1 = rxMessage.Payload[1];
         }

        break;
      case 2: // carga 2
        if ( rxMessage.Payload[1] <= Nivelmaxcargas ){
          nivelcarga2 = rxMessage.Payload[1];
        }

        break;
    }

    MiApp_DiscardMessage();
  }
}  // HandleRfMsg()
```

Quadro 1 – Implementação da função HandleRfMsg()

2.2.2. EEPROM com NodeID

Qualquer sistema que use RF precisa de um NodeID, isto é, um MAC address (número de 64 bits) que o identifique numa rede IEEE 802.15.4 (padrão que especifica a camada física e efectua o controlo do acesso a redes sem fios pessoais de baixas taxas de transmissão).

Prevendo a necessidade de se ter de guardar configurações do sistema, foi escolhida uma EEPROM que vem programada de fábrica com um NodeID, tendo assim as duas funcionalidades num só chip [5][18].

Este componente liga-se ao PIC24 através de SPI. O seu endereço pode ser lido desde a posição de memória FAh até FFh (48bits). Na Tabela 3 está exemplificado o mapeamento da memória para o NodeID.

Description	24-bit Organizationally Unique Identifier			24-bit Extension Identifier		
Data	00h	04h	A3h	12h	34h	56h
Array Address	FAh					FFh

Tabela 3 – Mapeamento da memória do NodeID de 48 bits.

Contudo, o NodeID tem de ser de 8 bytes, ou seja 64 bits (tamanho usado no protocolo implementado para o módulo de RF). Caso queiramos usar uma EEPROM com um NodeID de 48bits em substituição de uma de 64 bits vem expresso no datasheet que a memória seja mapeada da forma exemplificada na Tabela 4. Acrescenta-se “FFh” e “FEh” nos bytes 4 e 5 da memória.

Foi escolhida a EEPROM de 48 bits por ser mais barata que a de 64 bits.

Description	24-bit Organizationally Unique Identifier			40-bit Extension Identifier				
Data	00h	04h	A3h	**FFh**	**FEh**	12h	34h	56h

Tabela 4 – Conversão do mapeamento da memória do NodeID de 48 bits para 64 bits.

Foi usada a EEPROM com NodeID da Microchip com a referência 25AA02E48, ilustrada na Figura 16.

Figura 16 – EEPROM com NodeID 25AA02E48.

No Quadro 2 está representada a função "GetEUI()", que vai ler o "*Extended Unique Identifier"* (EUI) do sistema, e adaptar o NodeID de 48bits para 64bits.

```
void GetEUI(void){

  U8 i;

  EE_CS = 0; // faz o enable do pino chip select do nodeId
  Nop();

  SPIPut(3);    // comando de leitura
  SPIPut(0xFA); // endereço de inicio do EUI48

  for( i = 0; i < 6; i++){
    myLongAddress[i] = SPIGet();
  }

  EE_CS = 1; // faz o disable do pino chip select do nodeId

  // reorganizar para ser 64bits
  myLongAddress[7] = myLongAddress[5];
  myLongAddress[6] = myLongAddress[4];
  myLongAddress[5] = myLongAddress[3];

  myLongAddress[3] = 0xFF;
  myLongAddress[4] = 0xFE;

}  // GetEUI()
```

Quadro 2 – Implementação da função GetEUI()

2.2.3. Foto-transístor

Um requisito do *"Smart Switch"* era que tivesse um sensor de luminosidade. Foi feita uma pesquisa por foto-díodos e foto-transístores com um espectro entre 400 e 700 nm (espectro visível) e o mais barato era o foto-transístor TETPT4400 da Vishay, daí o uso deste especificamente.

O circuito para leitura do foto-transístor encontra-se ilustrado na Figura 17.

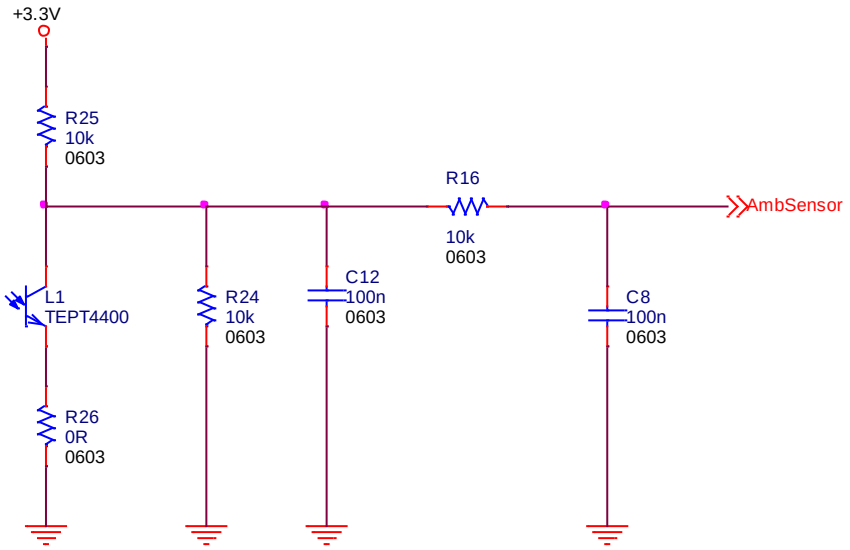


Figura 17 – Circuito de leitura do foto-transístor.

Quando não existe luz o foto-transístor não conduz. Deste modo foram colocadas 2 resistências de 10k (R25 e R24) em série para que, no máximo, a tensão que fica no pino do processador PIC24 (sinal AmbSensor) seja de 1.65V (tensão em R24), uma vez que a corrente que passa no foto-transístor é nula. A resistência R26 de 0 ohm foi colocada caso fosse necessário alterar o divisor de tensão (como não foi necessário colocou-se 0 ohm, ou seja, um curto-circuito).

Quando existe luz, o foto-transístor vai conduzir, ou seja vai haver corrente a passar nele. Quanto mais luz houver mais ele conduz, menor vai ser a tensão em R24, pois à medida que o foto-transístor vai conduzindo mais, menor vai ser a corrente a passar em R24. Juntamente com C12, R24 e R25 formam um filtro passa baixo acoplado ao filtro passa baixo formado por R16 e C8. Foram colocados 2 filtros passa baixo por questões de durabilidade e fiabilidade do sinal, caso um filtro deixe de funcionar está lá o outro.

A entrada do PIC24 apenas precisa de um bom condensador para que no momento da amostragem da tensão não "morra" o sinal da entrada. C8 faz também essa tarefa.

Para ler o sinal "AmbSensor" teve de ser configurado o módulo do ADC do PIC24. Foram criadas três funções para o módulo de ADC:

- "initADC()" que configura o módulo de ADC do PIC24, e está representada no Quadro 3.

```
void initADC(int inputscanreg){

  // Guarda as configuracoes anteriores do modulo ADC
  _AD1CSSL = AD1CSSL;
  _AD1CON1 = AD1CON1;
  _AD1CON2 = AD1CON2;
  _AD1CON3 = AD1CON3;

  AD1CSSL = inputscanreg;

  //ADC config
  AD1CON1 = 0x00E0; // Internal counter ends sampling and starts conversion
(auto-convert)
  AD1CON2 = 0x0400;
  AD1CON3 = 0x1F02; // 3*Tcy, Auto sample time bits = 31 Tad

  AD1CON1bits.ADON = 1; //A/D Converter module is operating

}  // initADC()
```

Quadro 3 – Implementação da função initADC()

- "readADC()" que retorna o valor lido pelo módulo de ADC, e está representada no Quadro 4.

```
int readADC(int input_sensor_channel){  // Rotina de leitura do ADC

  ADC1BUF0=0; // limpa buffer do ADC

  _AD1CHS = AD1CHS; // faz o backup do registo AD1CHS

  AD1CHSbits.CH0SA = input_sensor_channel;  // select analog input channel

  AD1CON1bits.SAMP = 1;  // start sampling, automatic conversion will follow

  while (!AD1CON1bits.DONE);{  // wait to complete the conversion 0-not done
                               // 1- done
    AD1CON1bits.ASAM=0;
  }

  AD1CON1bits.DONE=0;

  return ADC1BUF0;  // return the conversion result

}  // readADC()
```

Quadro 4 – Implementação da função readADC()

- "restoreADC()" que coloca as configurações do ADC como estavam antes de ser chamada a função de leitura do sensor de luminosidade. Esta operação tem de ser feita porque temos apenas um ADC no PIC24, e as configurações do ADC para a conversão do sinal do sensor de luminosidade e da leitura do teclado são diferentes. A função "restoreADC()" está representada no Quadro 5.

```
void restoreADC(void){

  AD1CSSL = _AD1CSSL;
  AD1CON1 = _AD1CON1;
  AD1CON2 = _AD1CON2;
  AD1CON3 = _AD1CON3;
  AD1CHS  = _AD1CHS;

}  // restoreADC()
```

Quadro 5 – Implementação da função restoreADC()

Para tomar acções do valor lido pelo ADC do sensor de luminosidade foi criada a função "readamblight()", que liga todos os LEDs do módulo de interface assim que o valor lido pelo sensor seja maior que um valor estipulado ("lighsensorconstant") para o qual se quer que liguem os Leds. A função "readamblight()" está representada no Quadro 6.

```
void readamblight(){

  unsigned long valoradcluminosidade;

  initADC(inputscanreg_amblight);

  valoradcluminosidade = readADC(input_amblight);   // valor de tensao do
sensor de

// luminosidade

  if (valoradcluminosidade > lighsensorconstant) {

    LED1 = 1;  // liga LED1
    LED2 = 1;  // liga LED2
    LED3 = 1;  // liga LED3
    LED4 = 1;  // liga LED4

  } else {

      LED1 = 0;  // desliga LED1
      LED2 = 0;  // desliga LED2
      LED3 = 0;  // desliga LED3
      LED4 = 0;  // desliga LED4
    }

  restoreADC();  // repoe as confiuracoes do ADC que estavam antes da
                 // leitura do sensor de luminosidade

}  // readamblight()
```

Quadro 6 – Implementação da função readamblight()

2.2.4. Sensor de movimento

Era requisito do *"Smart Switch"* que houvesse a possibilidade de incorporar um sensor de movimento. Foi feita uma pesquisa sobre os sensores existentes e foi escolhido o Panasonic EKMC1601111 por ser o mais barato.

Este sensor já tem incorporado um estabilizador da tensão de alimentação, um amplificador, um comparador e uma lente como se pode verificar na Figura 18.

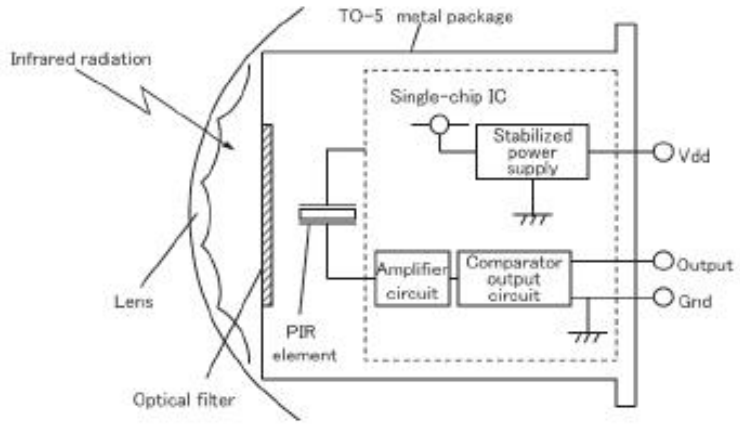


Figura 18 – Diagrama de blocos do sensor de movimento Panasonic EKMC1601111.

Para usar este sensor basta ligar a sua saída directamente ao pino do processador PIC24 e ver se o sinal está a "0" ou a "1". Para o seu correto funcionamento, depois de alimentar o sensor é necessário esperar 30 segundos para que este estabilize (Twu). A partir dos 30 segundos sempre que a saída for a "1" é porque o sensor detectou movimento (ver Figura 19). O sensor tem um alcance de 5 metros.

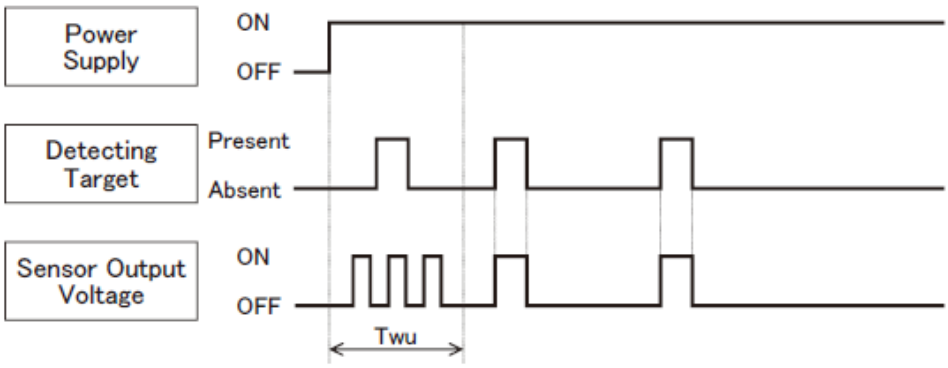


Figura 19 – Comportamento do sensor Panasonic EKMC1601111.

Para ler o sinal de saída do sensor de movimento foram usadas as funções “initADC()”,“readADC()” e “restoreADC()” descritas no ponto 0, e foi criada a função “detectmov()” que está no Apêndice C.

2.3. Módulo de potência

Para este módulo, e devido ao espaço existente, tínhamos de fazer uma fonte de alimentação que não necessitasse de um transformador. O facto é que os transformadores deixaram de ser usados em aplicações electrónicas devido à norma europeia de consumo de standby.

Depois de alguma pesquisa verificou-se que havia uma família de conversores AC (corrente alternada) / DC (corrente contínua) da *Power Integrations* (LNK302, 304, 305 e 306) que podia ser usada [1][2][3]. Depois de calcular os consumos equivalentes do sistema inteiro verificou-se que o consumo estava próximo de 190mA. Isto excluiria imediatamente os LNK302 (80mA) e 304 (170mA). Foi usado o LNK306 (380mA) (o LNK305 (280mA) não estava disponível para venda). Este conversor foi regulado para os 5V para alimentar os LEDs, os triacs das cargas e o regulador para 3.3V [4]. Foi escolhida a tensão de 5V pois o regulador para 3.3V não suporta mais de 5.5V na sua entrada.

Na Figura 20 está ilustrado o circuito do conversor AC / DC implementado no nosso sistema. Esta é uma topologia não isolada do tipo *buck-boost.* (a topologia *buck-boost* é a versão não isolada do conversor do tipo *Flyback* onde o transformador é substituído por um indutor (L3)).

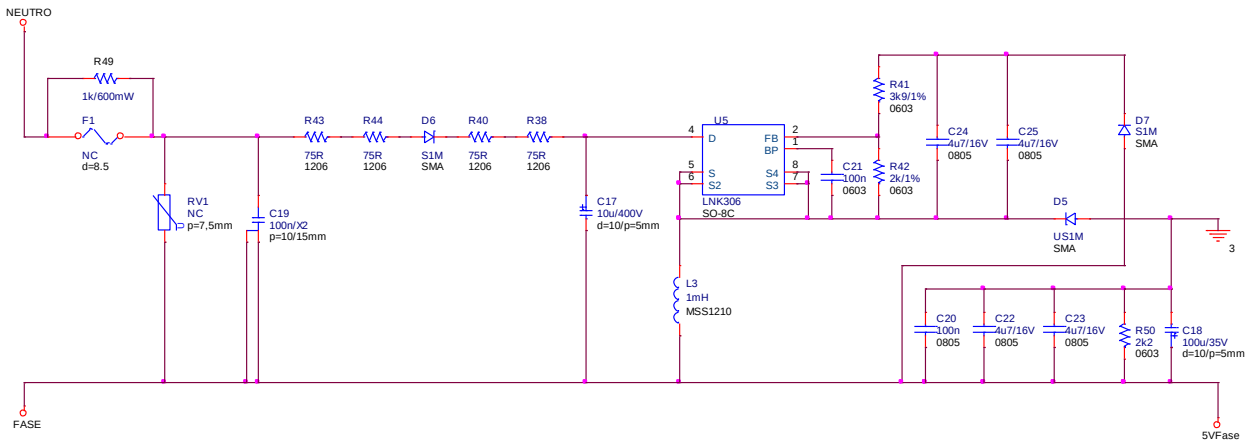


Figura 20 – Fonte não isolada topologia buck-boost regulada para 5V.

Vamos então dividir o esquema em duas partes para que seja mais perceptível a sua compreensão. Na Figura 21, R49 e C19 formam um filtro RC na malha de entrada do circuito. As resistências R38, R40, R43, R44 e R49 servem para limitar a corrente de entrada. Foram colocadas 4 resistências de 75R por causa de suportarem a potência de entrada do circuito, uma vez que são resistências de um tamanho muito pequeno relativamente às convencionais. A tensão AC é filtrada e rectificada de meia onda pelo díodo D6 e condensador C17.

RV1 representa um varístor, que não está montado fisicamente no nosso PCB, mas pode ser necessário para melhorar os ensaios de compatibilidade electromagnética.

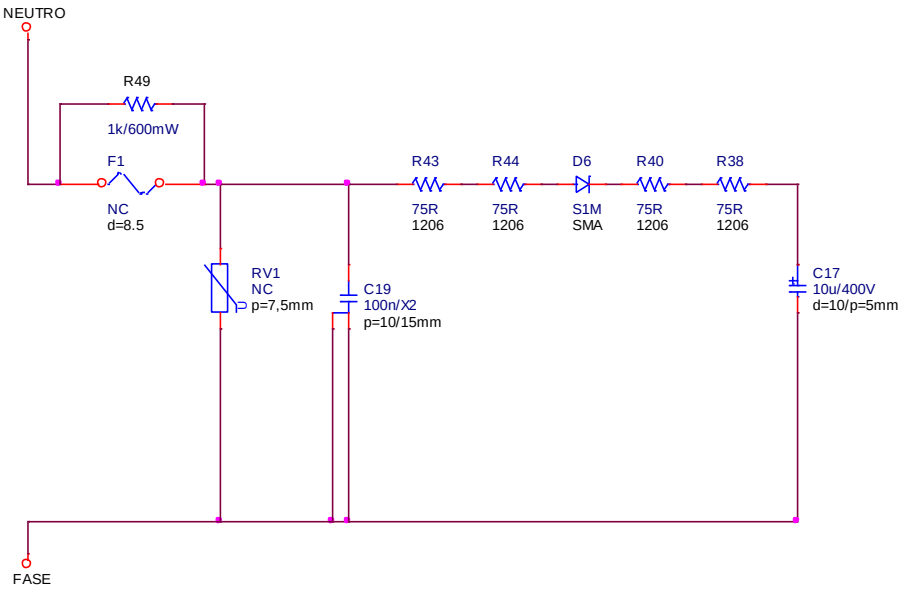


Figura 21 – Filtro e entrada RC e rectificação de meia onda (Legenda: NC- não colocado).

Na Figura 22, o *"LinkSwitch-TN"* U5, o díodo D7 e o indutor L3 formam o chamado buck-boost.

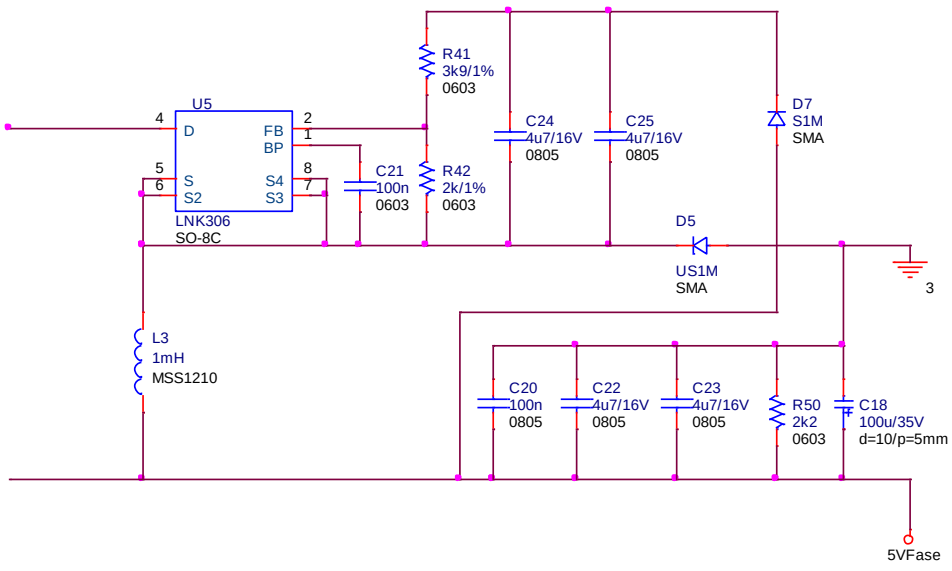


Figura 22 – Circuito com o *buck-boost* e componentes para regulação da saída.

O pino de feedback (FB, pino nº2) é internamente colocado a 1.65V, sendo assim a tensão de saída regulada através do divisor de tensão feito entre as resistências R41 e R42.
O condensador C21 é chamado de condensador de *bypass* que funciona para o desacoplamento das altas frequências e para armazenar energia. Este condensador além de fornecer a energia para o U5 e controla também o seu mecanismo de *auto-restart.* Por fim, a resistência R50 serve para reduzir os picos de carga do condensador C18 que tendem a fazer subir a tensão de saída. Foram usados 3 condensadores na saída (C20, C22 e C23) devido a ocuparem menos espaço relativamente a só um condensador com a capacidade equivalente à dos 3 condensadores usados.

Para obter os 3.3V para alimentar o PIC24, o módulo de RF, o NodeID e os sensores de luminosidade e movimento necessitávamos de um regulador que não ocupasse muito espaço (poucos componentes) e que fosse eficiente. Por estes motivos foi escolhido o conversor *buck-boost* da *Texas Instruments* TPS63031 que além de ter mais de 96% de eficiência só necessita de um circuito com um indutor, um condensador na entrada e outro na saída (a tensão de saída é fixa – 3.3V).

Foi previsto o uso de um conversor da mesma família do que foi usado (TPS63030), daí a existência dos componentes R28, R29, R33 e C16. Caso fosse usado o conversor TPS63030, que tem uma tensão de saída regulável, R29 teria de passar a não colocado no circuito e a tensão de saída passa a ser estipulada pelo divisor de tensão entre R28 e R33. O circuito usado está ilustrado na Figura 23.

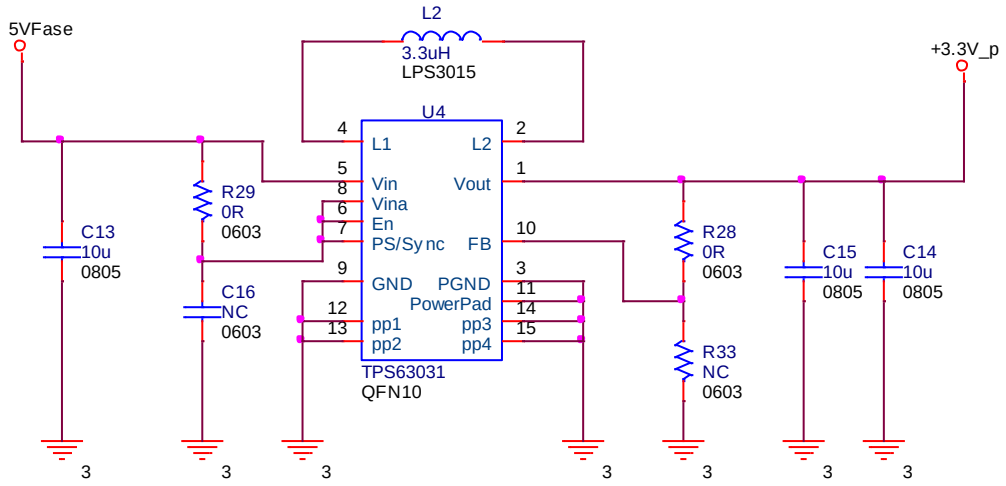


Figura 23 – Conversor *buck-boost* para os 3.3V (Legenda: NC- não colocado).

2.3.1. Circuito de *Dimming*

No módulo de potência, foi necessário desenvolver um circuito que actuasse as cargas, e que no caso da iluminação tivesse a possibilidade de fazer *dimming*. A actuação para cargas resistivas é feita por triac, daí o uso deste no circuito. O triac tem a peculiaridade de ligar imediatamente quando damos ordem para tal, mas quando damos ordem para desligar ele só desliga na próxima passagem por zero da tensão da rede.

Na Figura 24 está representado como se faz *dimming* de 20%, 50% e 90% de uma lâmpada.

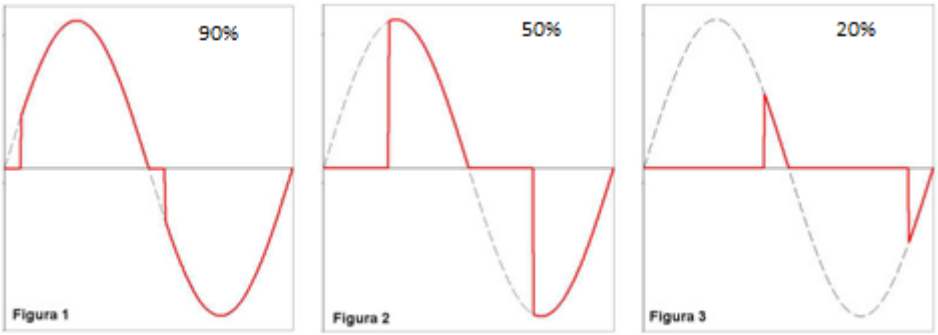


Figura 24 – *Dimming* a 20%, 50% e 90%.

Analisando a Figura 24, verificamos que só é possível fazer o *dimming* detectando a passagem por zero da tensão da rede (230 Vac). Na Figura 25 está implementado o circuito usado para detecção da passagem por zero da tensão da rede.

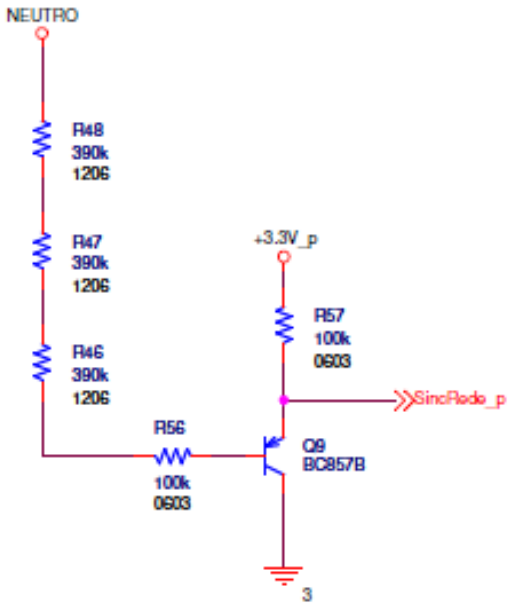


Figura 25 – Circuito de detecção da passagem por zero da tensão da rede.

Na Figura 26 está a simulação do sinal que vai aparecer no pino do PIC24, sinal “SincRede_p” (onda cor de rosa).

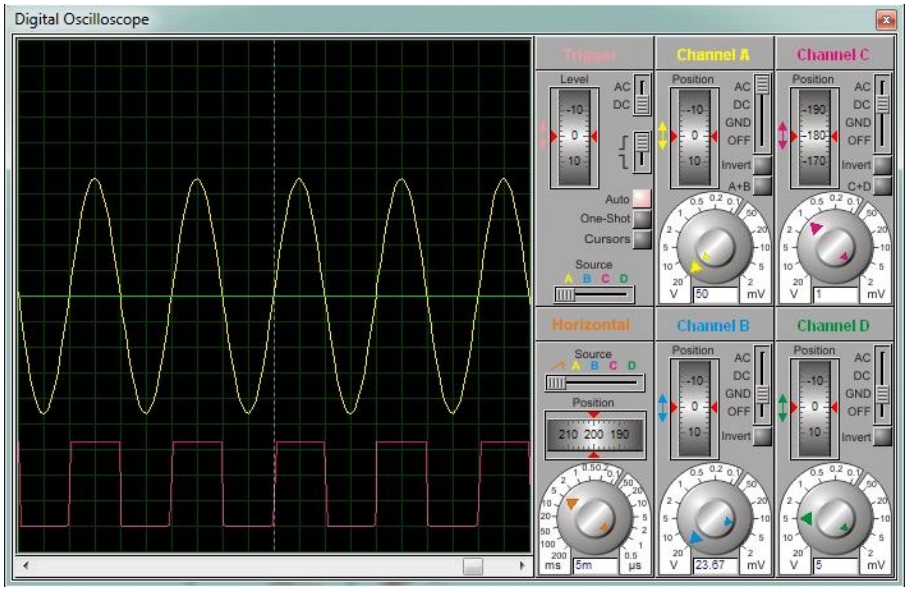


Figura 26 – Simulação do circuito que detecta a passagem por zero da tensão da rede.

Em termos de firmware foi criada uma interrupção INT1 no pino onde esta ligado o sinal “SincRede_p”, para que fosse activada na fase descendente da onda (função “IntZeroPassInit()” representada no Quadro 7).

```
void IntZeroPassInit(void){

  INTCON2 = 0x1F;        // Setup INT1 interupt on negative edge
  IFS1bits.INT1IF = 0;   // Reset INT1 interrupt flag
  IEC1bits.INT1IE = 1;   // Enable INT1 Interrupt Service Routine
  IPC5bits.INT1IP = 7;   // set high priority

}  // IntZeroPassInit()
```

Quadro 7 – Implementação da função IntZeroPassInit()

Para fazer o *dimming* foi programado, dentro da interrupção INT1, um modo de *Pulse-Width Modulated* (PWM) para cada uma das cargas [17].

O circuito com triac para controlo das cargas é o que está ilustrado na Figura 27 (para a carga 2 o circuito é igual ao da carga 1).

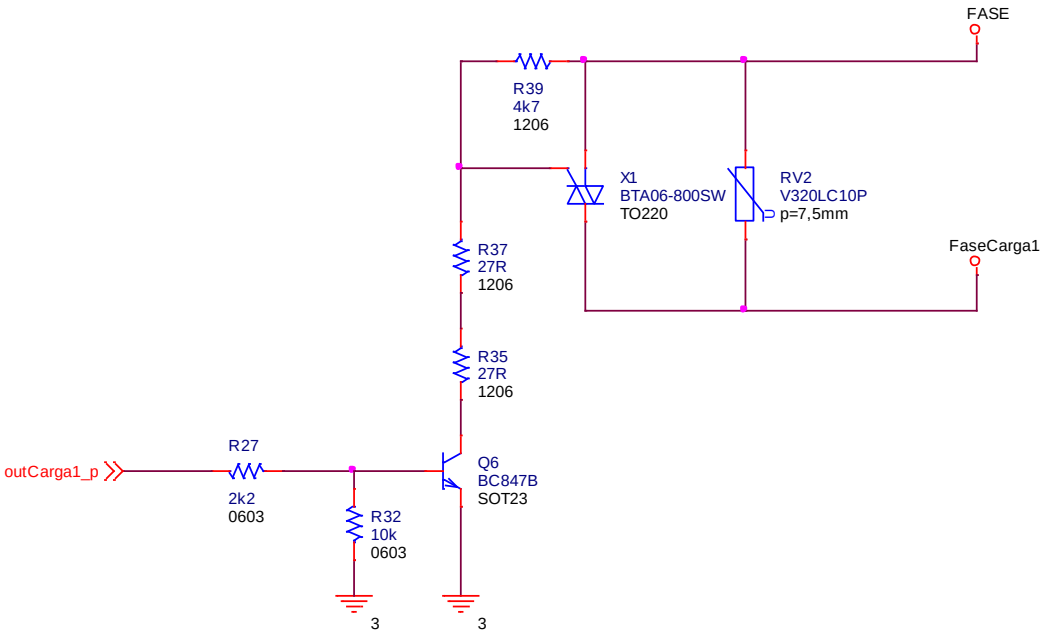


Figura 27 – Circuito para actuação do triac da carga 1.

Quando o pino outCarga1_p é colocado a "1" o triac liga, quando é colocado a "0" o triac desliga na próxima passagem por zero da tensão da rede. O varístor RV2 serve para absorver algum pico de tensão que seja criado pela comutação do triac, de modo a não danificar a carga.

O firmware elaborado dentro da interrupção INT1 para o *dimming* das cargas está representado no Quadro 8.

```
void __attribute__((__interrupt__, no_auto_psv)) _INT1Interrupt(void) {

//#############################################################################################
//######################### Configuracao do PWM para a carga 1 ##############################

  T2CON = 0x8010; // enable timer 2, Tcy, 1:8
  TMR2 = 0;

  PWMCharge1 = 0;


// 1. OC1 Initialization

  OC1CON1 = 0;
  OC1CON2 = 0;

// 2. Calculate the desired pulse on time value based upon TCY and write it into OCxR.

  if (nivelcarga1 <= 0 || nivelcarga1 > Nivelmaxcargas){

    nivelcarga1=0;
    OC1R = 0;
    OC1RS = 0;

  } else if (nivelcarga1 > 0 || nivelcarga1 <= Nivelmaxcargas){

           OC1R = (Nivelmaxcargas-nivelcarga1)*pulsoPWM;
           // 3. Calculate the period value based upon TCY and write it into OCxRS.
           OC1RS = periodoPWM;
         }

// 4. Write 0x1F to the SYNCSEL<4:0> bits (OCxCON2<4:0>) to select
// self-synchronization.

  OC1CON2bits.SYNCSEL = 0x1F;

// 5. Set the required clock source.

  OC1CON1bits.OCTSEL = 0b000; // Using timer 2

// PWM Fault Condition Status bit (1-PWM Fault condition has occurred
// (cleared in HW only)

  OC1CON1bits.OCFLT0 = 1;

// 6. Set the OCM<2:0> bits (OCxCON1<2:0>)

  OC1CON1bits.OCM = 0b101;  // 0b101 to select Double Compare Continuous Pulse mode

  // ligar o pino RP4 como saída PWM do OC1
  PWMCharge1 = 18;

//#############################################################################################
//######################### Configuracao do PWM para a carga 2 ##############################

  T3CON = 0x8010; // enable timer 3, Tcy, 1:8
  TMR3 = 0;

  PWMCharge2 = 0;

// 1. OC2 Initialization
  OC2CON1 = 0;
  OC2CON2 = 0;
```

```
// 2. Calculate the desired pulse on time value based upon TCY and write it into OCxR.

  if (nivelcarga2 <= 0 || nivelcarga2 > Nivelmaxcargas){

    nivelcarga2=0;
    OC2R = 0;
    OC2RS = 0;

  } else if (nivelcarga2 > 0 || nivelcarga2 <= Nivelmaxcargas){

            OC2R = (Nivelmaxcargas-nivelcarga2)*pulsoPWM;
            // 3. Calculate the period value based upon TCY and write it into OCxRS.
            OC2RS = periodoPWM;
          }


// 4. Write 0x1F to the SYNCSEL<4:0> bits (OCxCON2<4:0>) to select
// self-synchronization.

  OC2CON2bits.SYNCSEL = 0x1F;

// 5. Set the required clock source.

  OC2CON1bits.OCTSEL = 0b001; // Using timer 3

// PWM Fault Condition Status bit (1-PWM Fault condition has occurred
// (cleared in HW only)

  OC2CON1bits.OCFLT0 = 1;

// 6. Set the OCM<2:0> bits (OCxCON1<2:0>)

  OC2CON1bits.OCM = 0b101;  // 0b101 to select Double Compare Continuous Pulse mode

  // ligar o pino RP4 como saída PWM do OC2
  PWMCharge2 = 19;

  IFS1bits.INT1IF = 0;    //Clear the INT1 interrupt flag

}  // _INT1Interrupt()
```

Quadro 8 – Implementação da função _INT1Interrupt()

Capítulo 3

Testes e Resultados Experimentais

Foram realizadas várias experiências em cada um dos módulos projectados (teclado, interface e potência). Antes de enviar os ficheiros para fabrico do sistema, foram montados numa *bread board* todos os circuitos projectados, e feito firmware para garantir que todos funcionavam como o esperado. A montagem está ilustrada na Figura 28.

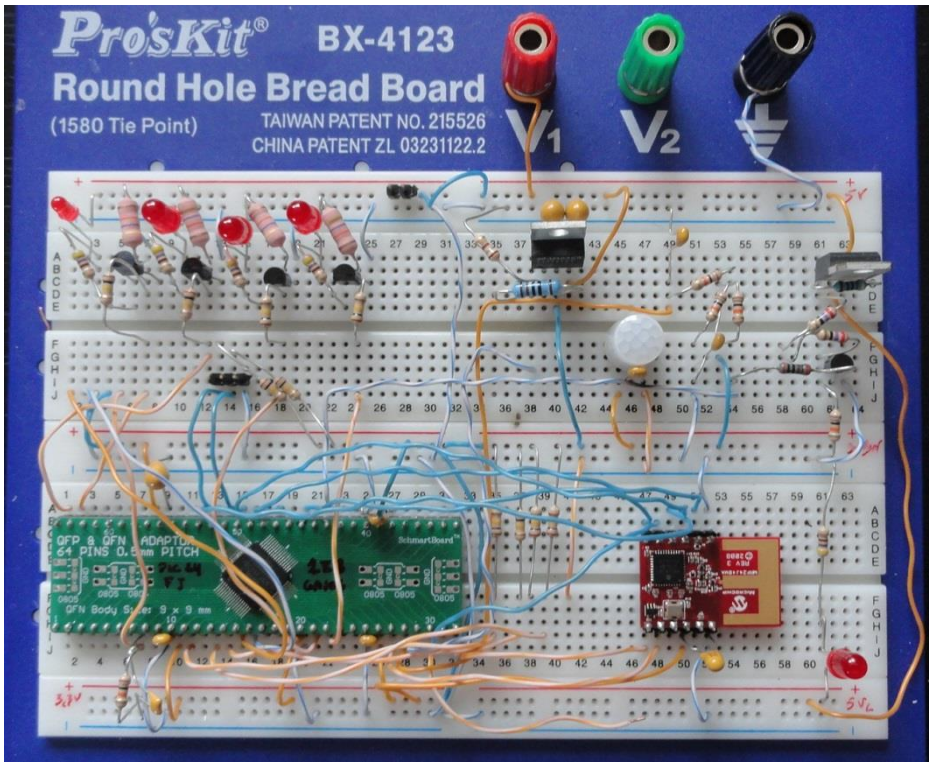


Figura 28 – Montagem dos circuitos usados no *"Smart Switch"*.

3.1. Ensaios realizados com o módulo de teclado

Para testar o módulo de teclado foram feitos 2 ensaios. No primeiro ensaio foi retirada a forma de onda correspondente ao sinal de tensão de uma das teclas sem estar premida e estando premida, para podermos comparar com o sinal teórico que está ilustrado na Figura 11.

Na Figura 29 está ilustrada a situação onde não é premida nenhuma tecla (tecla solta) e na Figura 30 está ilustrada a situação onde é premida uma tecla.

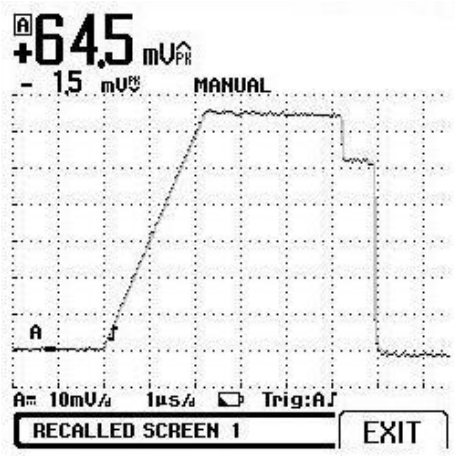


Figura 29 – Sinal medido para tecla solta.

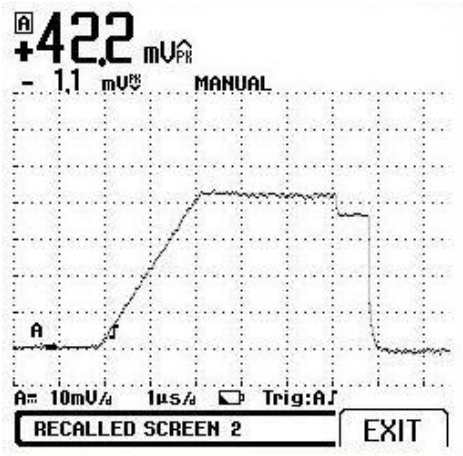


Figura 30 – Sinal medido para tecla premida.

No segundo ensaio foram definidas acções para quando fossem detectadas teclas premidas. Se fosse detectada a tecla superior esquerda o LED superior esquerdo ligava, se fosse detectada a tecla inferior esquerda o LED inferior esquerdo ligava e se fosse detectada a tecla central esquerda os LEDs superior e inferior direito ligavam. Foi feita a mesma analogia para as teclas do lado direito, e os resultados estão ilustrados nas figuras seguintes:

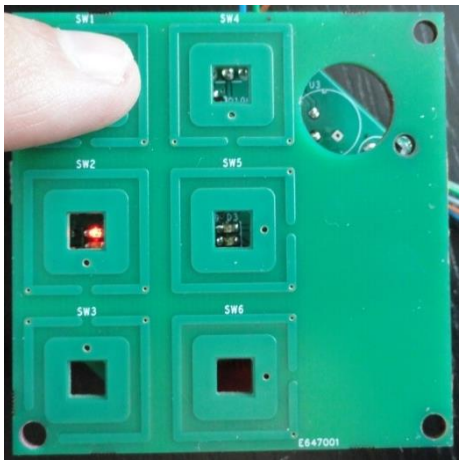


Figura 31 – Leitura do botão superior esquerdo (aumento da luz da lâmpada/carga 1).

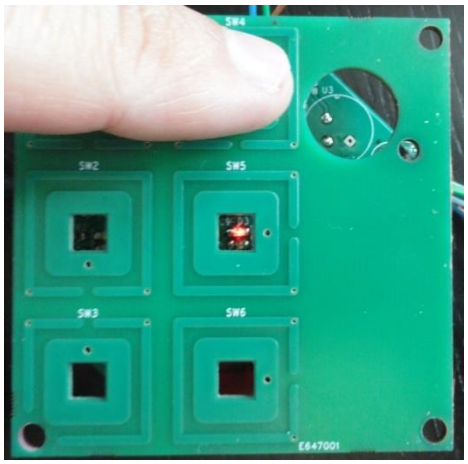


Figura 32 – Leitura do botão superior direito (aumento da luz da lâmpada/carga 2).

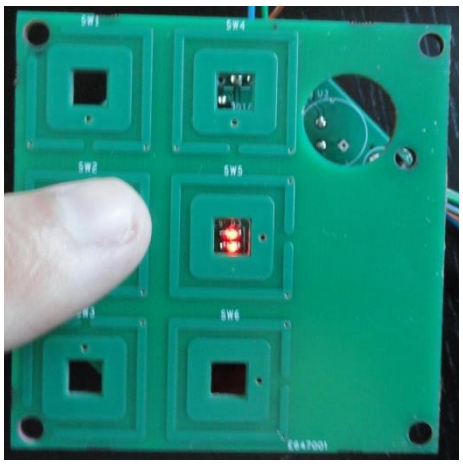


Figura 33 – Leitura do botão central esquerdo (On/Off da luz da lâmpada/carga 1).

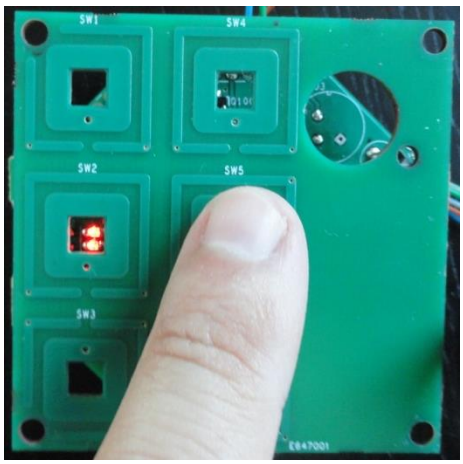


Figura 34 – Leitura do botão central direito (On/Off da luz da lâmpada/carga 2).

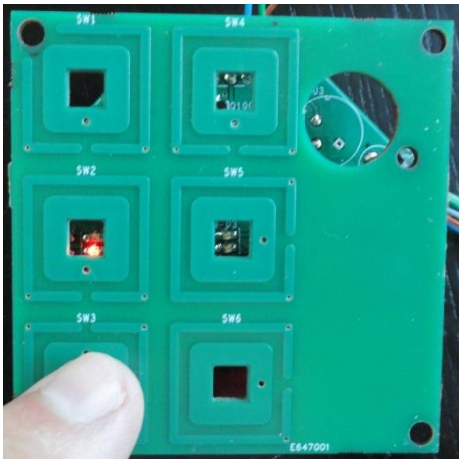


Figura 35 – Leitura do botão inferior esquerdo (diminuição da luz da lâmpada/carga 1).

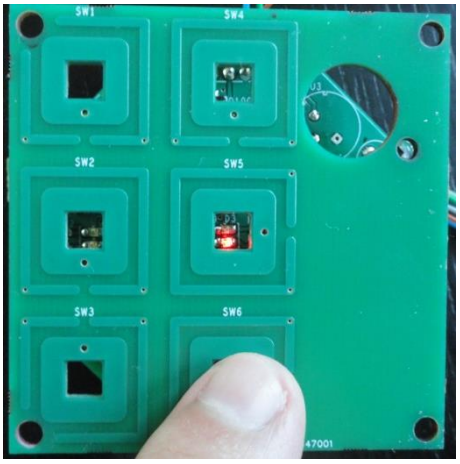


Figura 36 – Leitura do botão inferior direito (diminuição da luz da lâmpada/carga 2).

Nota: Nos ficheiros desta dissertação existe um vídeo que mostra o momento em que se esteve a fazer estes ensaios.

3.2. Ensaios realizados com o módulo de interface

No módulo de interface foram realizados ensaios com o sensor de luminosidade e sensor de movimento. O sensor de luminosidade foi programado para que fossem ligados todos os LEDs do sistema quando a luz ambiente estivesse apagada, e para que não ligasse nenhum LED quando a luz ambiente estivesse ligada. Estão ilustrados na Figura 37 e Figura 38 essas situações.

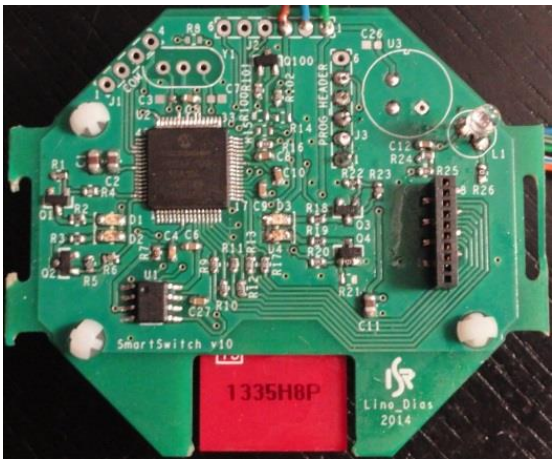


Figura 37 – Luz ambiente ligada.

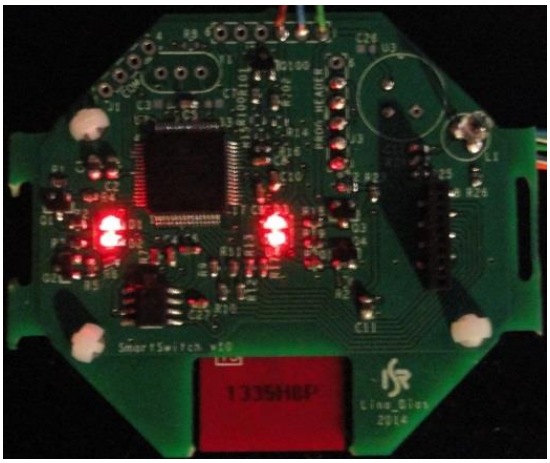


Figura 38 – Luz ambiente desligada.

Nota: Nos ficheiros desta dissertação existe um vídeo que mostra o momento em que se esteve a fazer este ensaio.

O sensor de movimento foi testado na bread board, pois só existia um sensor para ser usado. Foi estipulado que quando o sensor de movimento detectasse algo ligava os quatro LEDs durante três segundos. O sinal retirado com o osciloscópio do pino de saída do sensor está ilustrado na Figura 39.

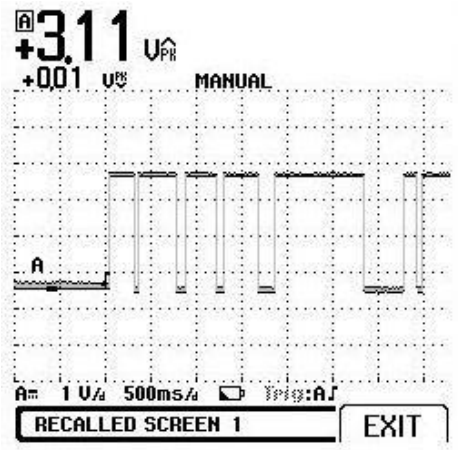


Figura 39 – Sinal de saída do sensor de movimento.

Na Figura 40 está representada a acção tomada pelo sistema quando detectou a mão.

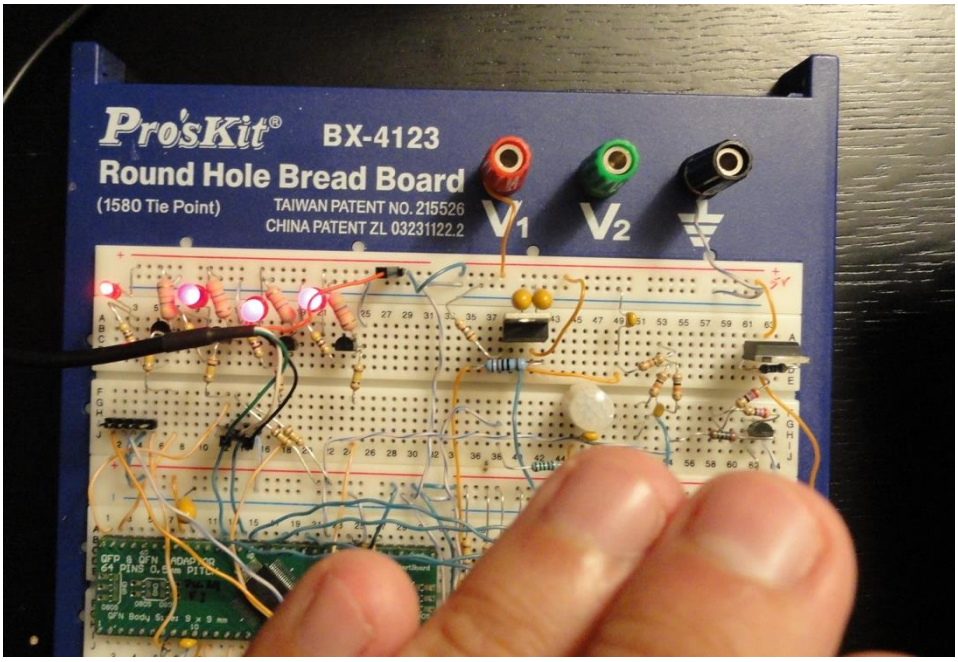


Figura 40 – Detecção do sensor de movimento.

Nota: Nos ficheiros desta dissertação existe um vídeo que mostra o momento em que se esteve a fazer este ensaio.

3.3. Ensaios realizados com o módulo de potência

O módulo de potência foi testado em três fases. Na primeira fase foi montado apenas no PCB a fonte que regula para os 5V, isto para que se houvesse algum problema com esta tensão não avariasse a fonte dos 3.3V que só suporta 5.5V na sua entrada (facto já referido anteriormente).

Quando foi verificado que estava tudo a funcionar como o projectado na fonte dos 5V foi colocada uma resistência de 20Ω à saída da fonte, isto para que puxássemos 250mA ($R = \frac{U}{I} = \frac{5}{250.10^{-3}} = 20\Omega$) do conversor AC / DC para avaliar o seu comportamento. Verificou-se que não existia nenhuma anomalia no circuito.

Na segunda fase foi montada a fonte dos 3.3V, e como era esperado funcionava como o que tinha sido projectado.

Na terceira e última fase foram montados os dois circuitos que fazem o driver das cargas. Foram retirados os sinais do osciloscópio de uma das saídas quando foi feito *dimming* dos 10% até aos 100%.

Vê-se claramente nas figuras seguintes que o circuito projectado para o sincronismo de rede funciona na perfeição, e que o *dimming* está implementado correctamente. No canal "A" está registada a onda correspondente à tensão da rede, e no canal "B" a tensão aplicada na carga.

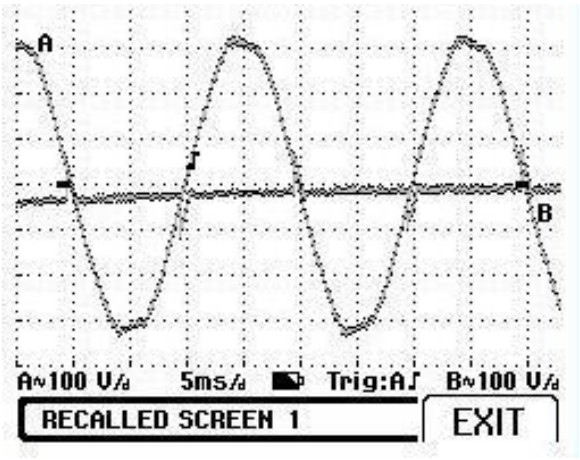


Figura 41 – *Dimming* a 10%.

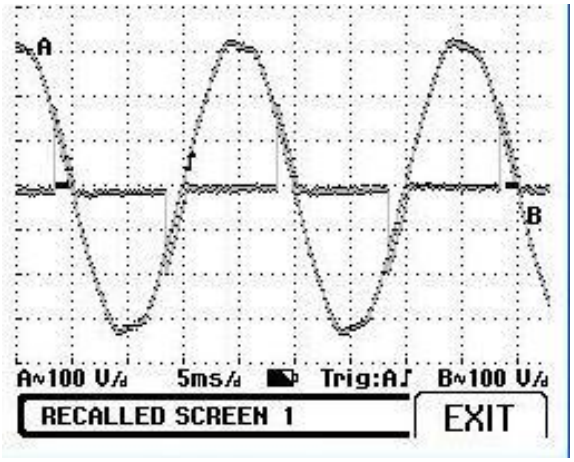


Figura 42 – *Dimming* a 20%.

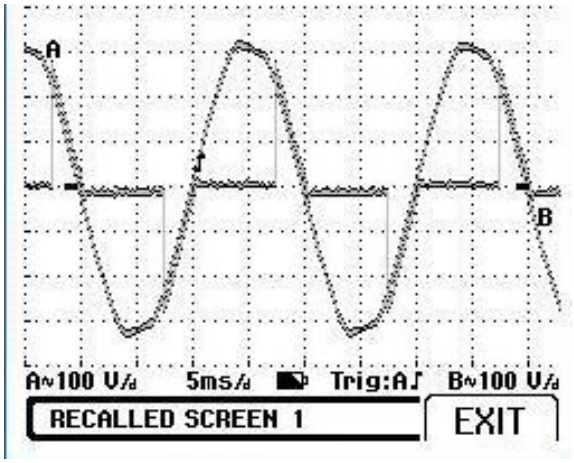


Figura 43 – *Dimming* a 30%.

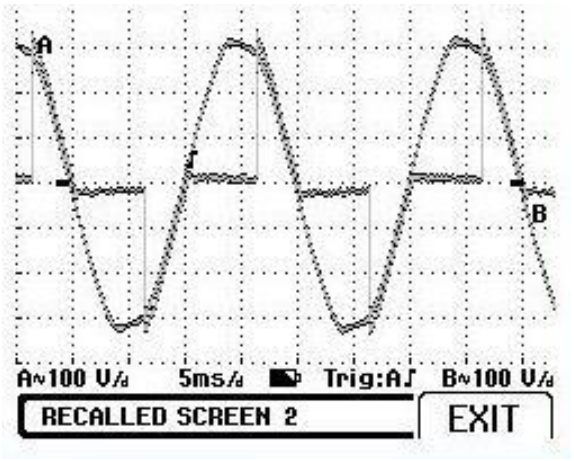


Figura 44 – *Dimming* a 40%.

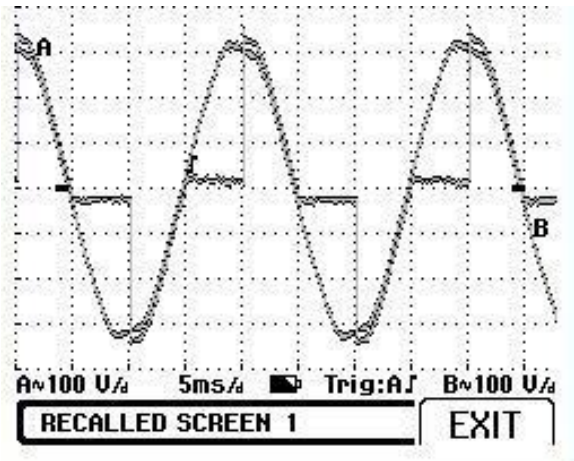


Figura 45 – *Dimming* a 50%.

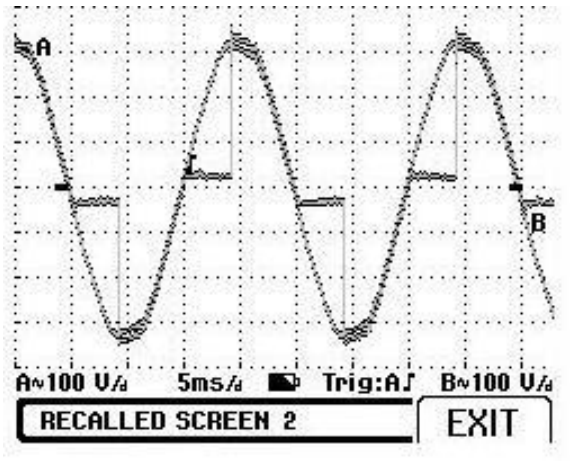


Figura 46 – *Dimming* a 60%.

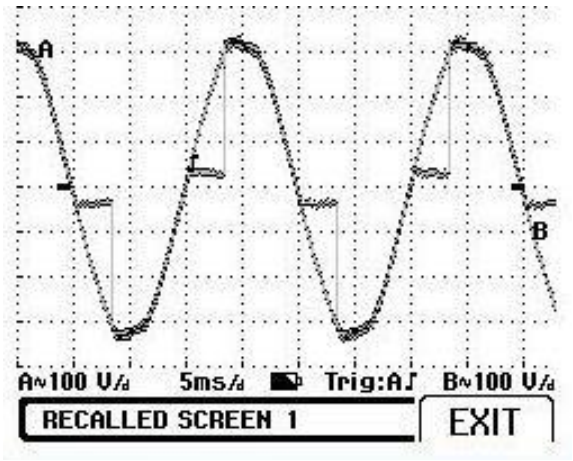


Figura 47 – *Dimming* a 70%.

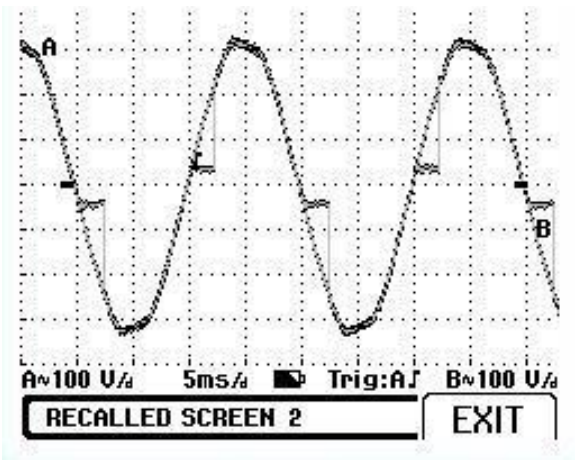


Figura 48 – *Dimming* a 80%.

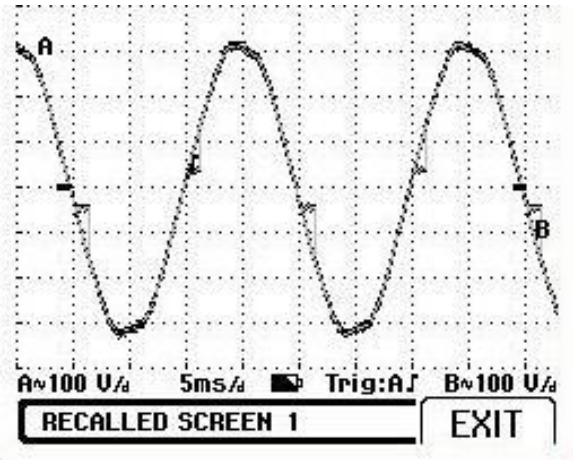


Figura 49 – *Dimming* a 90%.

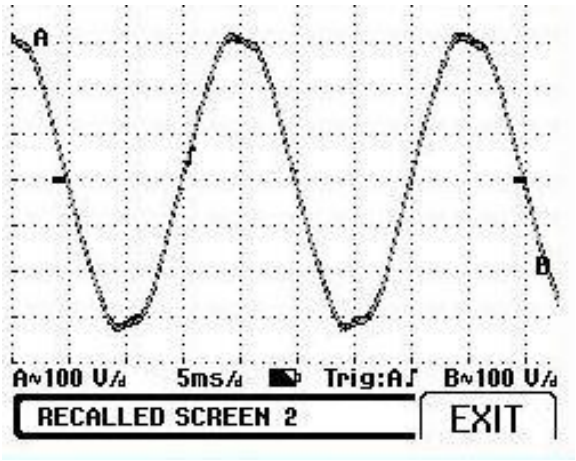


Figura 50 – *Dimming* a 100%.

3.4. Interface com o utilizador para PC para comunicar com os sistemas *"Smart Switch"*

Para controlar os sistemas *"Smart Switch"*, utilizando um PC, foi programada em C# uma interface com o utilizador usando o compilador MonoDevelop (opensource). Para fazer o PC comunicar via RF com os "*Smart Switch"* está a ser usado um módulo de interface e controlo ligado através de um cabo que converte USB em UART (Figura 52) ao PC. O sistema está ilustrado na Figura 51.

Figura 51 – Módulo de interface e controlo com ligação USB para PC.

Figura 52 – Conversor USB-UART CP2102 da *Silicon Labs.*

A interface desenvolvida permite escolher a porta COM onde está ligado o conversor, escolher o valor da intensidade da luz desejado (entre 1 e 10), escolher qual dos 2 sistemas construídos (1 ou 2) actuar e permite ainda enviar mensagem em broadcast para actuar em todos os sistemas simultaneamente. Na Figura 53 está ilustrado o aspecto da interface desenvolvida.

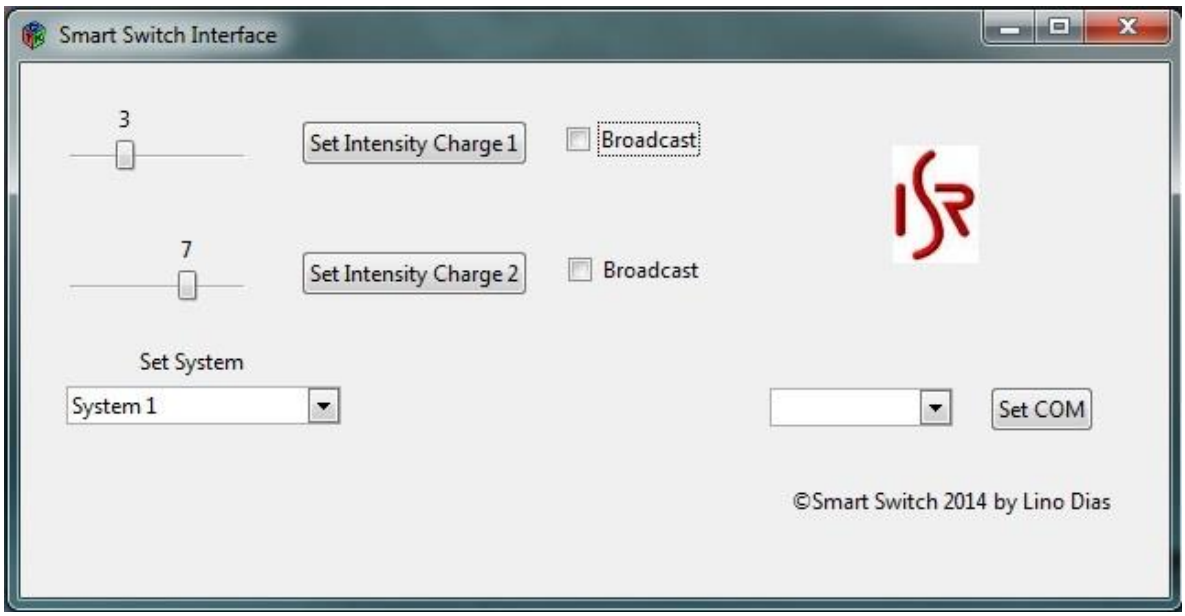


Figura 53 – Aspecto da Interface para PC desenvolvida.

Capítulo 4

Conclusões e Trabalho Futuro

Com este trabalho apresentamos um dispositivo electrónico único, que completa uma lacuna no mercado para quem quer inserir sistemas deste tipo em instalações já existentes. É um dispositivo de fácil instalação, necessita apenas de ser alimentado por Fase e Neutro, e tem a possibilidade de ter duas saídas para cargas, cada uma delas capaz de suportar 400W.

A vertente usada neste trabalho foi a sua aplicação em sistemas de iluminação, mas pode ser também usado em estores eléctricos, ou qualquer outra carga que não exceda os 400W sem ter de alterar o hardware existente.

Os ensaios realizados foram para um sistema de iluminação com duas cargas, conseguindo variar a intensidade de luz que cada uma tem, quer por via do teclado quer pela aplicação feita para PC.

Para as montagens de interruptor de escadas este sistema é muito fácil de implementar em relação aos sistemas convencionais, em que é preciso ligar todos os interruptores entre si. Com o *"Smart Switch"*, podemos ter quantos interruptores quisermos para comandar uma carga, sem eles estarem ligados entre si. Só um dos interruptores está ligado fisicamente à carga, os outros que também a comandam simplesmente estão alimentados pela Fase e Neutro e enviam mensagens por RF para alterar o estado do interruptor que tem a carga ligada.

O *"Smart Switch"* pode ser utilizado de inúmeras maneiras. Pode ser um interruptor convencional de *touch control*, retirando o módulo de RF e fazendo ou não *dimming*. Pode ser usado como dispositivo de segurança usando o seu sensor de movimento (detectar movimento quando não é suposto estar ninguém na habitação).

Para trabalho futuro sugere-se:

- Montagem de mais exemplares *"Smart Switch"* para implementar um protocolo de comunicação do tipo MiWi™ PRO, onde é preciso definir uma rede, onde não vai haver limitação de alcance do RF pois os módulos da rede podem encaminhar a mensagem até ao último elemento da rede.
- Ligar o módulo de interface e controlo do Smart Switch a um access point para que todos os sistemas Smart Switch possam ser controlados por um telemóvel, tablet ou qualquer dispositivo que tenha WiFi.

- Desenvolver um dispositivo táctil (semelhante a um tablet) com um módulo de RF para comandar os *"Smart Switch"*.
- Experimentar o sistema em outras aplicações (estores, como sistema de segurança).
- Submeter o Smart Switch aos ensaios necessários para que seja um produto certificado para venda no mercado.

Referências

[1] Power Integrations, March 2014. Application Note AN-37 LinkSwitch™-TN Family. Design Guide.

[2] Power Integrations Applications Department, April 20, 2005. 3W Non-Isolated Buck Boost Converter using LNK305P. DER-49.

[3] Power Integrations, November 2008. LNK302/304-306. LinkSwitch®-TN Family. Lowest Component Count, Energy-Efficient Off-Line Switcher IC.

[4] Texas Instruments Incorporated, March 2012. TPS63030/TPS63031. High Efficiency Single Inductor Buck-Boost Converter With 1-A Switches.

[5] Microchip Technology Inc., 2008-2013. 25AA02E48/25AA02E64. 2K SPI Bus Serial EEPROMs with EUI-48™ or EUI-64™ Node Identity.

[6] Microchip Technology Inc, 2010. PIC24FJ256GA110 Family Data Sheet. 64/80/100-Pin, 16-Bit, General Purpose Flash Microcontrollers with Peripheral Pin Select.

[7] Microchip Technology Inc, 2008. MRF24J40MA Data Sheet. 2.4 GHz IEEE Std. 802.15.4™ RF Transceiver Module.

[8] Microchip Technology Inc, Yifeng Yang, 2010. AN1204. Microchip MiWi™ P2P Wireless Protocol.

[9] Microchip Technology Inc, Bruce Bohn, 2009. AN1250. Microchip CTMU for Capacitive Touch Applications.

[10] Microchip Technology Inc, 2010. Section 11. Charge Time Measurement Unit (CTMU).

[11] Microchip Technology Inc, 2009. MASTERs 2009. The Worldwide Conference for Embedded Control Engineers. 1337 CTMU Capacitive Touch Sensing Using CTMU.

[12] Microchip Technology Inc, Tom Perme, 2007. AN1102. Layout and Physical Design Guidelines for Capacitive Sensing.

[13] Microchip Technology Inc, Burke Davison, 2010-2013. AN1334. Techniques for Robust Touch Sensing Design.

[14] Microchip Technology Inc, Xiang Gao, 2013. AN1492. Microchip Capacitive Proximity Design Guide.

[15] Microchip Technology Inc, 2007. MPLAB® C30 C Compiler User's Guide.

[16] Microchip Technology Inc, 2009-2013. Universal Asynchronous Receiver Transmitter (UART).

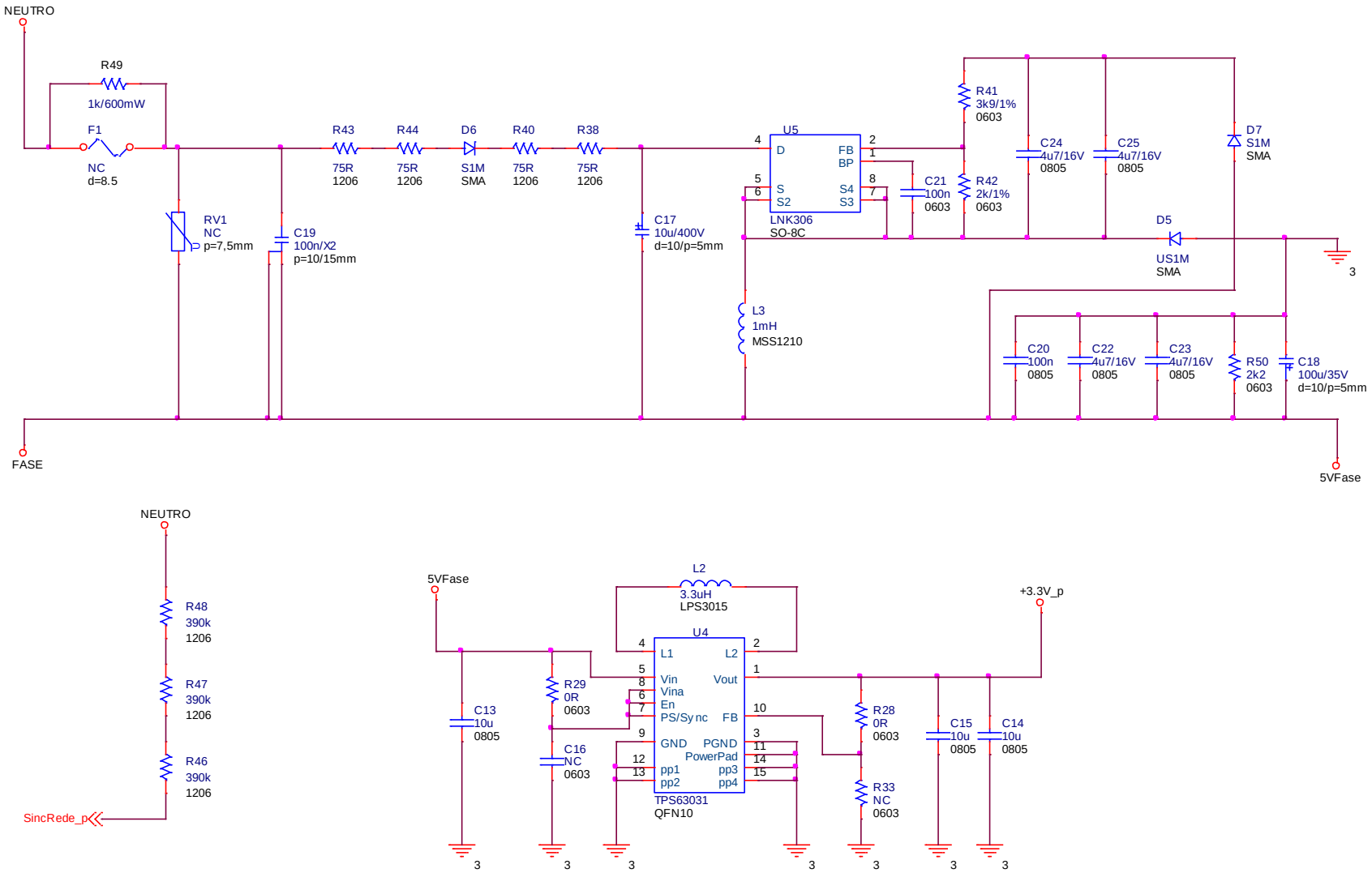
[17] Microchip Technology Inc, 2014. Output Compare with Dedicated Timer.

[18] Microchip Technology Inc, June 2013. Release Notes for Microchip Libraries for Applications v2013-06-15.

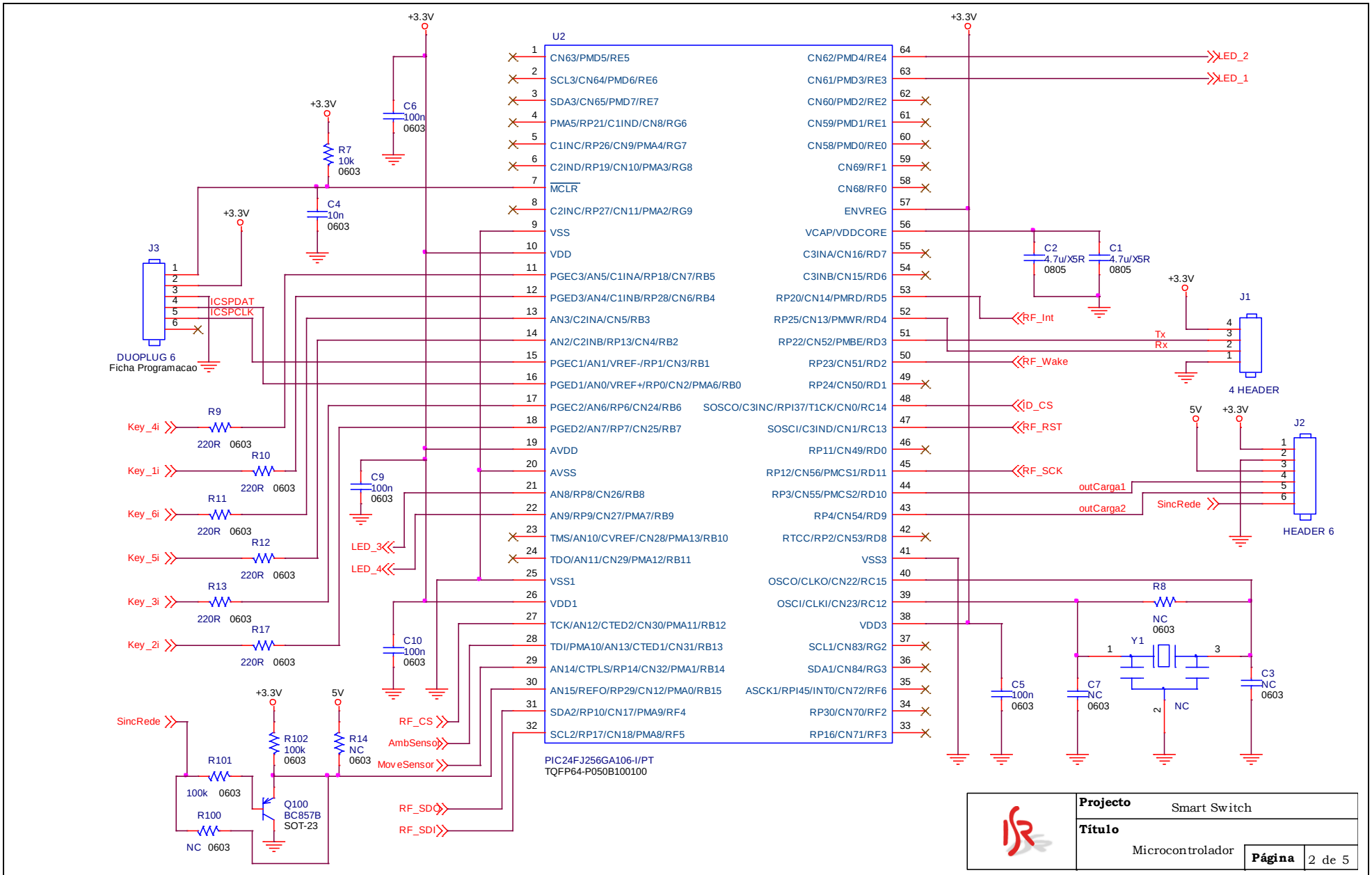
[19] Microchip Technology Inc, 2008. Microchip WebSeminars. Overview of Charge Time Measurement Unit (CTMU).

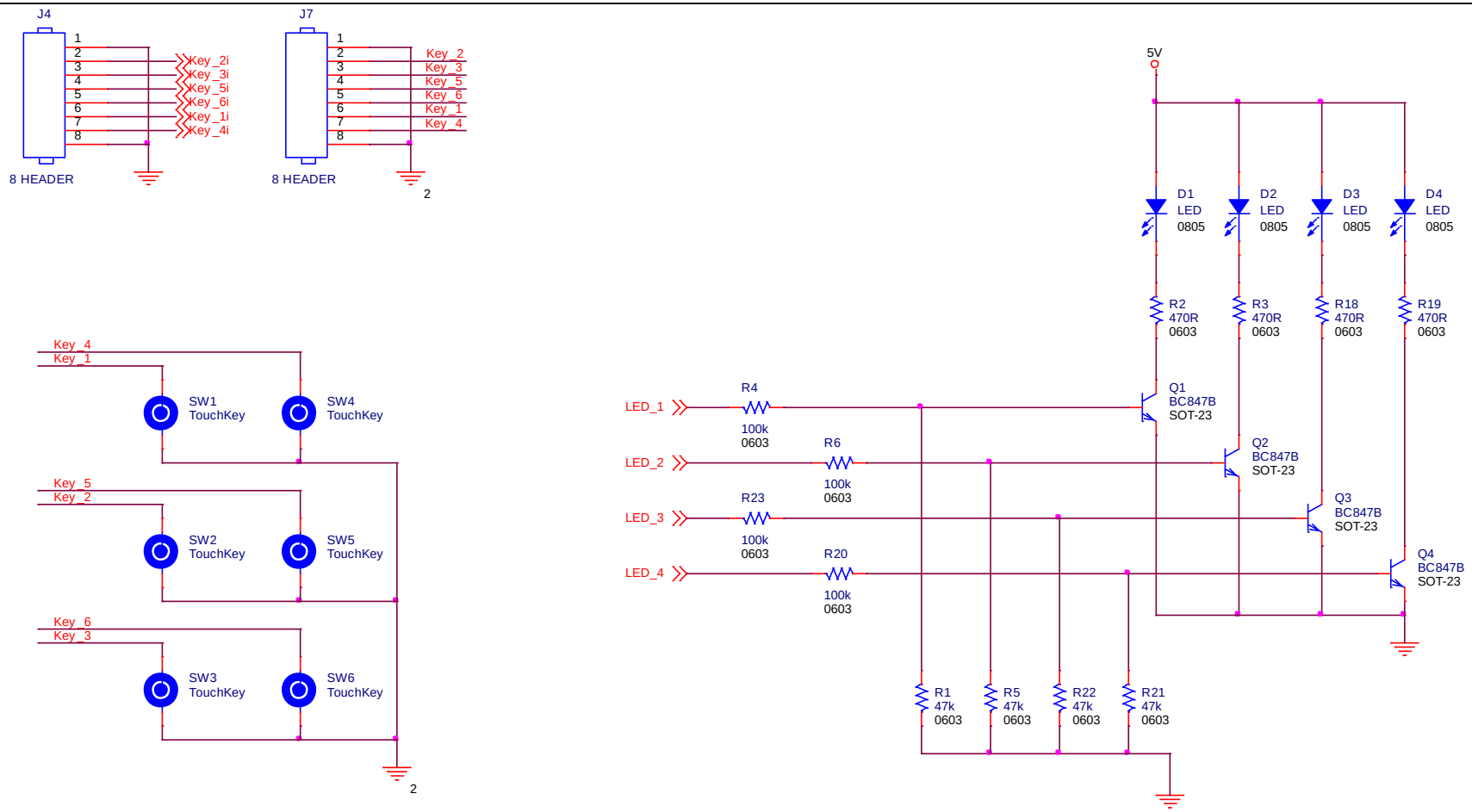
Apêndice A

Esquemático do Smart Switch

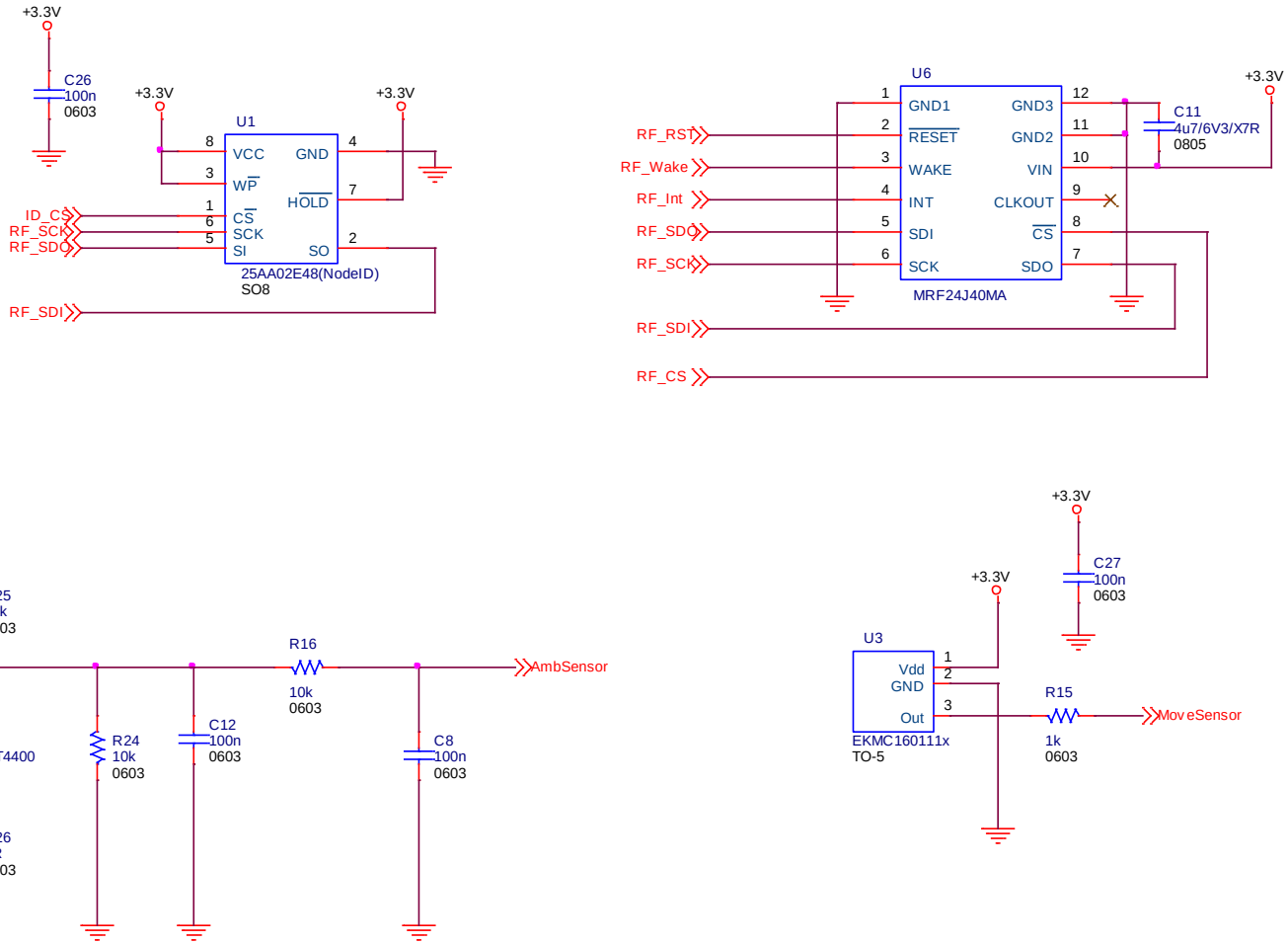


	Elaborou Lino Dias	**Projecto** Smart Switch	**Edição** 1
Substitui a versão: -/-	**Data** 2014/08/18	**Título** Power Supply	**Revisão** 0
			Página 1 de 5

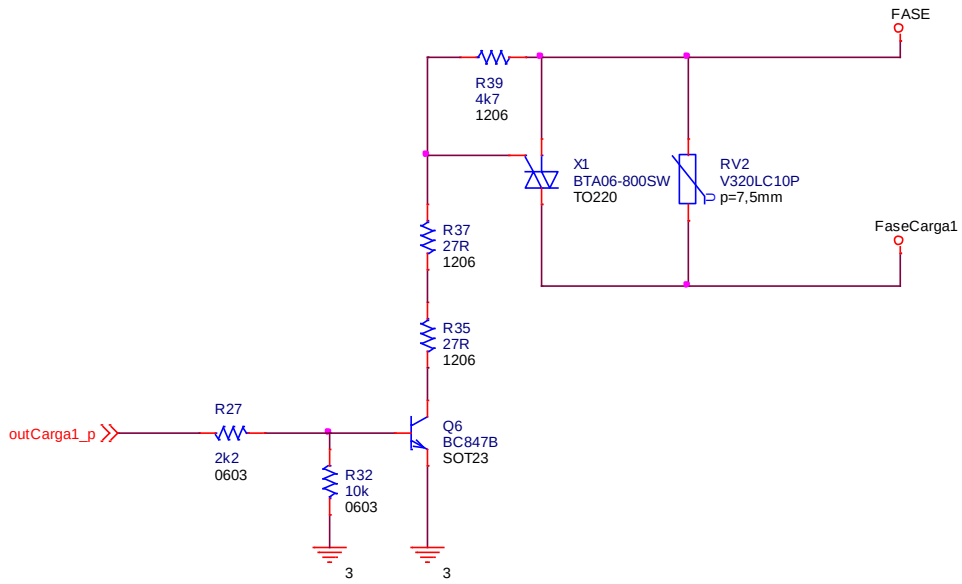

U2
PIC24FJ256GA106-I/PT
TQFP64-P050B100100
CN63/PMD5/RE5
SCL3/CN64/PMD6/RE6
SDA3/CN65/PMD7/RE7
PMA5/RP21/C1IND/CN8/RG6
C1INC/RP26/CN9/PMA4/RG7
C2IND/RP19/CN10/PMA3/RG8
MCLR
C2INC/RP27/CN11/PMA2/RG9
VSS
VDD
PGEC3/AN5/C1INA/RP18/CN7/RB5
PGED3/AN4/C1INB/RP28/CN6/RB4
AN3/C2INA/CN5/RB3
AN2/C2INB/RP13/CN4/RB2
PGEC1/AN1/VREF-/RP1/CN3/RB1
PGED1/AN0/VREF+/RP0/CN2/PMA6/RB0
PGEC2/AN6/RP6/CN24/RB6
PGED2/AN7/RP7/CN25/RB7
AVDD
AVSS
AN8/RP8/CN26/RB8
AN9/RP9/CN27/PMA7/RB9
TMS/AN10/CVREF/CN28/PMA13/RB10
TDO/AN11/CN29/PMA12/RB11
VSS1
VDD1
TCK/AN12/CTED2/CN30/PMA11/RB12
TDI/PMA10/AN13/CTED1/CN31/RB13
AN14/CTPLS/RP14/CN32/PMA1/RB14
AN15/REFO/RP29/CN12/PMA0/RB15
SDA2/RP10/CN17/PMA9/RF4
SCL2/RP17/CN18/PMA8/RF5
CN62/PMD4/RE4
CN61/PMD3/RE3
CN60/PMD2/RE2
CN59/PMD1/RE1
CN58/PMD0/RE0
CN69/RF1
CN68/RF0
ENVREG
VCAP/VDDCORE
C3INA/CN16/RD7
C3INB/CN15/RD6
RP20/CN14/PMRD/RD5
RP25/CN13/PMWR/RD4
RP22/CN52/PMBE/RD3
RP23/CN51/RD2
RP24/CN50/RD1
SOSCO/C3INC/RPI37/T1CK/CN0/RC14
SOSCI/C3IND/CN1/RC13
RP11/CN49/RD0
RP12/CN56/PMCS1/RD11
RP3/CN55/PMCS2/RD10
RP4/CN54/RD9
RTCC/RP2/CN53/RD8
VSS3
OSCO/CLKO/CN22/RC15
OSCI/CLKI/CN23/RC12
VDD3
SCL1/CN83/RG2
SDA1/CN84/RG3
ASCK1/RPI45/INT0/CN72/RF6
RP30/CN70/RF2
RP16/CN71/RF3
+3.3V
C6 100n 0603
R7 10k 0603
C4 10n 0603
J3
DUOPLUG 6
Ficha Programacao
ICSPDAT
ICSPCLK
R9 220R 0603
R10 220R 0603
R11 220R 0603
R12 220R 0603
R13 220R 0603
R17 220R 0603
Key_4i
Key_1i
Key_6i
Key_5i
Key_3i
Key_2i
C9 100n 0603
C10 100n 0603
LED_3
LED_4
5V
SincRede
R102 100k 0603
R14 NC 0603
R101 100k 0603
R100 NC 0603
Q100 BC857B SOT-23
RF_CS
AmbSensor
MoveSensor
RF_SDO
RF_SDI
LED_2
LED_1
C2 4.7u/X5R 0805
C1 4.7u/X5R 0805
RF_Int
RF_Wake
ID_CS
RF_RST
RF_SCK
J1
4 HEADER
Tx
Rx
J2
HEADER 6
outCarga1
outCarga2
R8 NC 0603
Y1
C5 100n 0603
C7 NC 0603
C3 NC 0603
NC
Projecto
Smart Switch
Título
Microcontrolador
Página
2 de 5

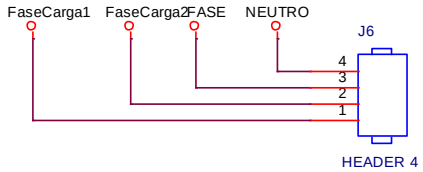


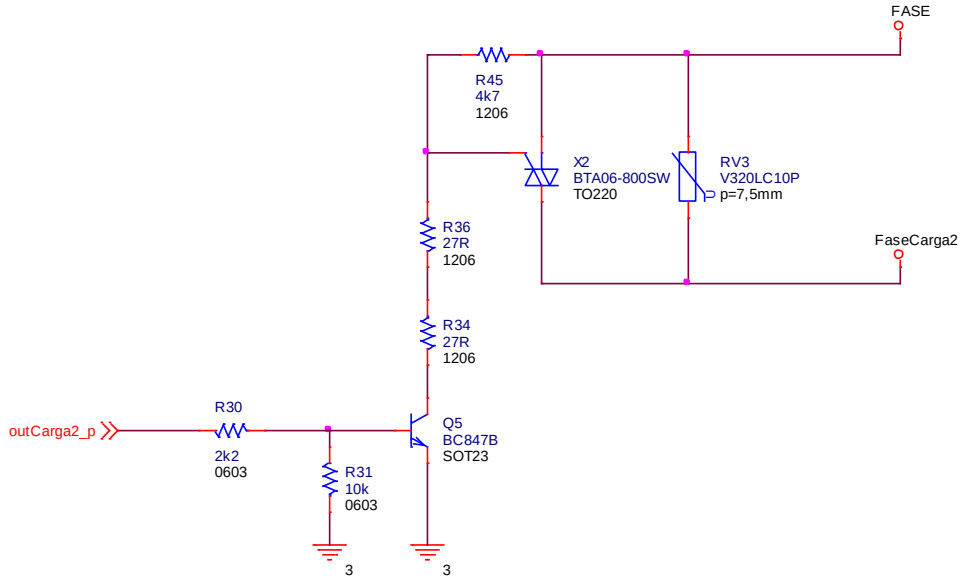
	Projecto	Smart Switch
	Título	Teclado e LEDs
	Página	3 de 5

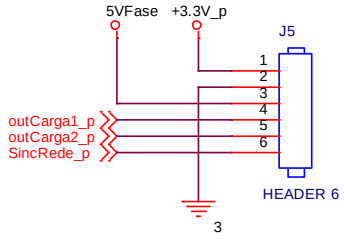


	Projecto Smart Switch	
	Título Node_ID e RF	**Página** 4 de 5





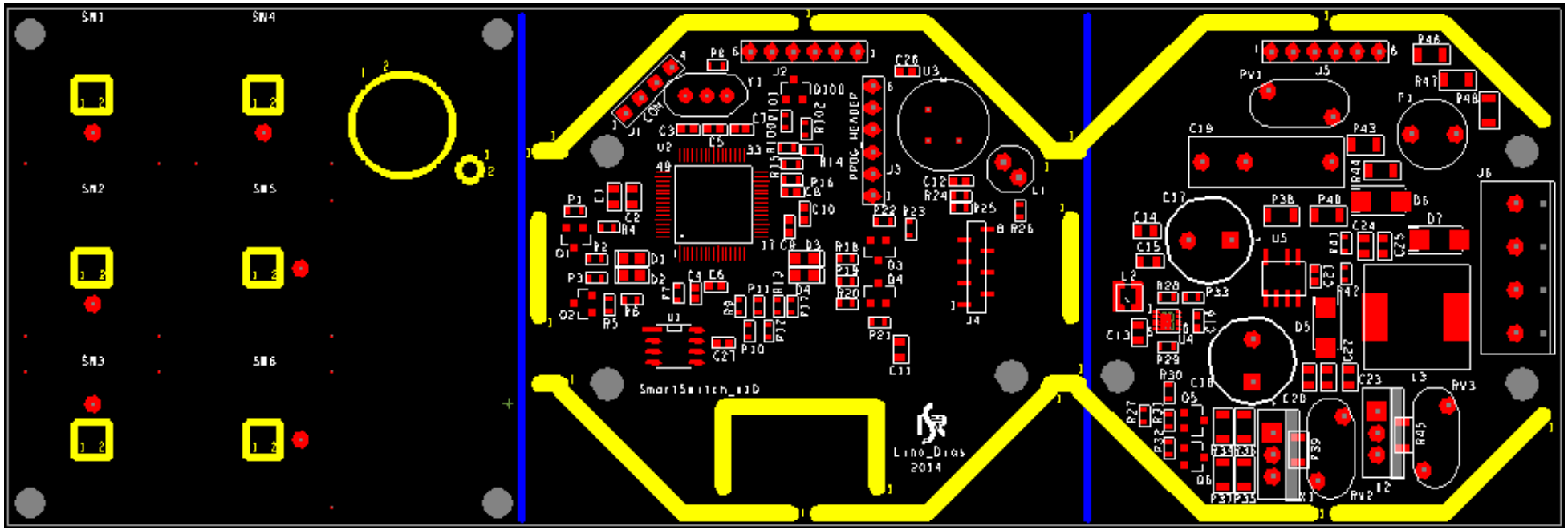




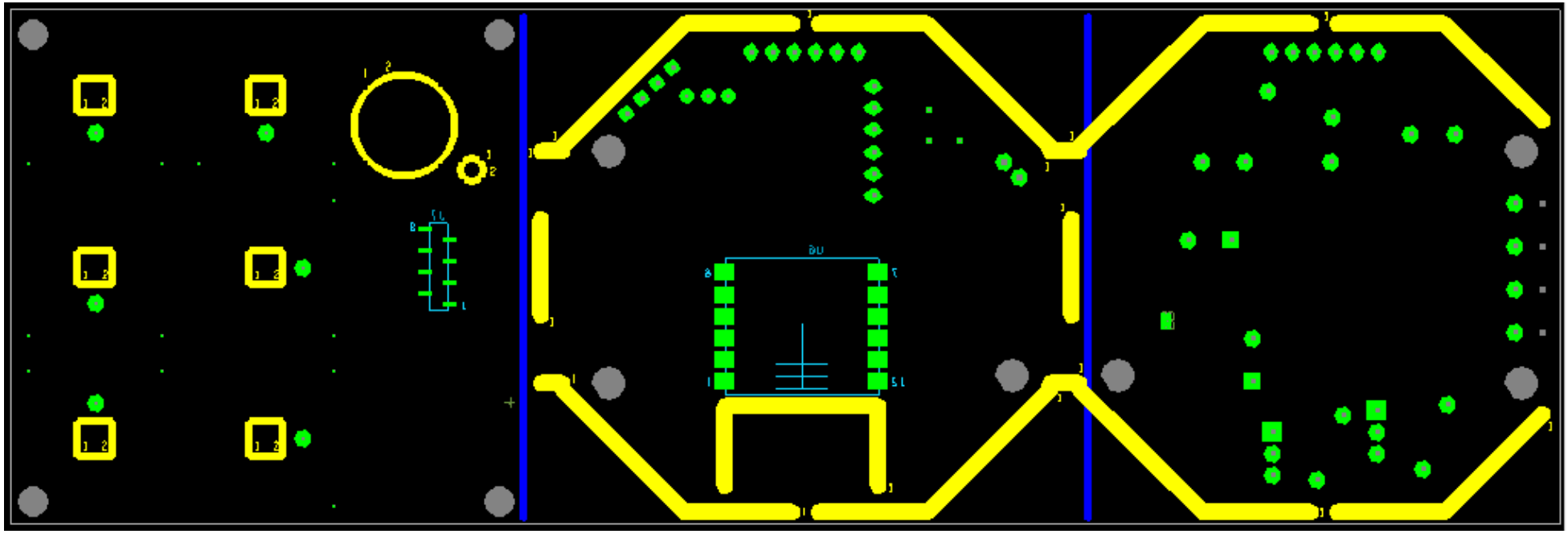
	Projecto Smart Switch	
	Título Cargas	**Página** 5 de 5

Apêndice B

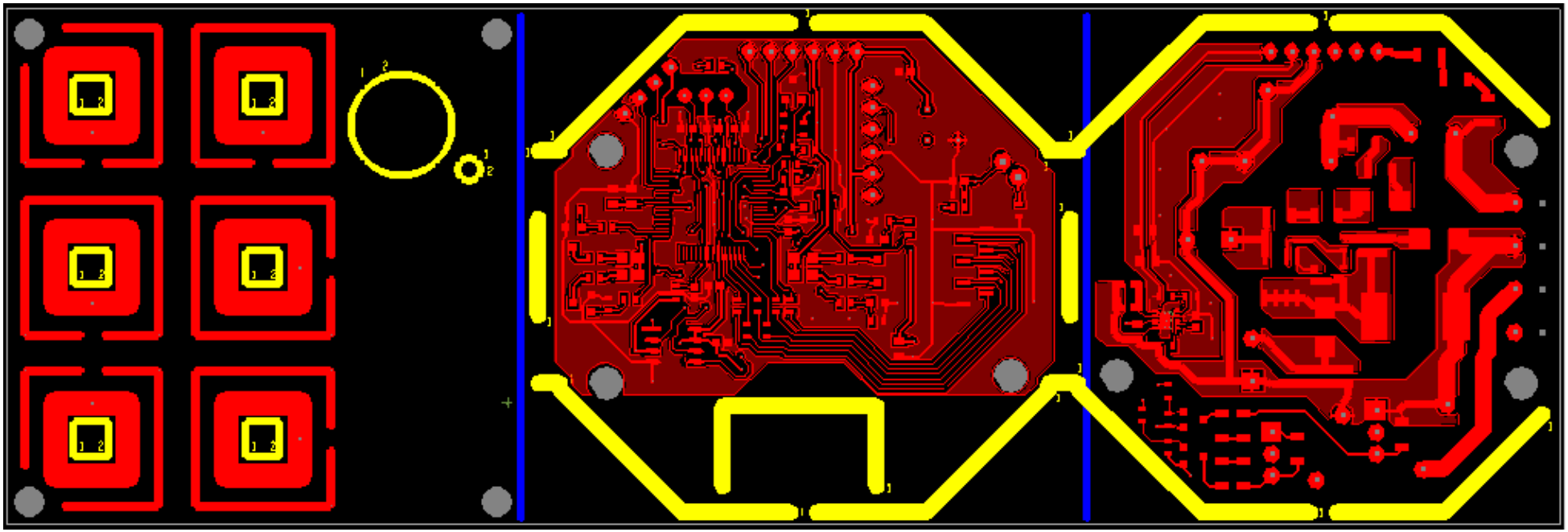
Serigrafia, pistas Top e Bottom do PCB



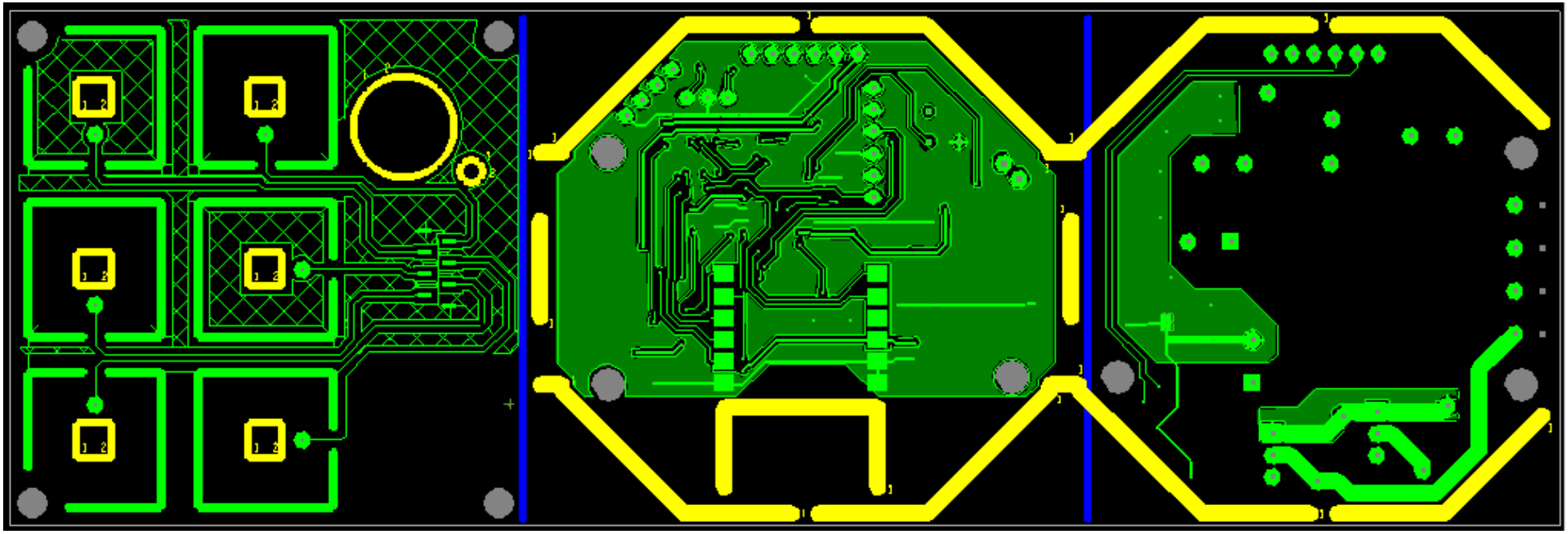
Serigrafia do lado Top do PCB desenvolvido



Serigrafia do lado Bottom do PCB desenvolvido



Pistas do lado Top do PCB desenvolvido



Pistas do lado Bottom do PCB desenvolvido

Apêndice C

Firmware Desenvolvido

```
int InitTeclado(void){

  #ifdef  MTOUCH_DEBUG
    DEBUGDELAY*   pDelay;
  #endif

  // STEP 1 - mTouch library initialization
  MTouchInit(); // activa o ADC e o modulo CTMU

  // STEP 2 - Sensors initialization.
  // sensor #0 – tecla on/off da carga 2
  MTouchSetSensor(0, CH2_TRIS, CH2_LAT, CH2_IO_BIT, CH2_AN_NUM, -1, -1, -1);
  // sensor #1 – tecla "-" da carga 2
  MTouchSetSensor(1, CH3_TRIS, CH3_LAT, CH3_IO_BIT, CH3_AN_NUM, -1, -1, -1);
  // sensor #2 – tecla "+" da carga 1
  MTouchSetSensor(2, CH4_TRIS, CH4_LAT, CH4_IO_BIT, CH4_AN_NUM, -1, -1, -1);
  // sensor #3 – tecla "+" da carga 2
  MTouchSetSensor(3, CH5_TRIS, CH5_LAT, CH5_IO_BIT, CH5_AN_NUM, -1, -1, -1);
  // sensor #4 – tecla "-" da carga 1
  MTouchSetSensor(4, CH6_TRIS, CH6_LAT, CH6_IO_BIT, CH6_AN_NUM, -1, -1, -1);
  // sensor #5 - tecla on/off da carga 1
  MTouchSetSensor(5, CH7_TRIS, CH7_LAT, CH7_IO_BIT, CH7_AN_NUM, -1, -1, -1);

  #ifdef  MTOUCH_DEBUG
    // MTouchDebugDelay(sensorNumber) function calculates an optimal CTMU
    // charge delay value to provide charging sensor about 75% of AVdd.

    pDelay = MTouchDebugDelay(0);
    // Set the adjusted optimal CTMU charge delay value for the sensor.
    MTouchSetChargeDelay(0, pDelay->delay);

    pDelay = MTouchDebugDelay(1);
    MTouchSetChargeDelay(1, pDelay->delay);

    pDelay = MTouchDebugDelay(2);
    MTouchSetChargeDelay(2, pDelay->delay);

    pDelay = MTouchDebugDelay(3);
    MTouchSetChargeDelay(3, pDelay->delay);

    pDelay = MTouchDebugDelay(4);
    MTouchSetChargeDelay(4, pDelay->delay);

 pDelay = MTouchDebugDelay(5);
    MTouchSetChargeDelay(5, pDelay->delay);

  #endif

// STEP 3
  // Timer interrupt initialization to call mTouch acquisition pereodically.
  SysTimerInit();

}  // InitTeclado()

// This callback function called every 1ms by timer.
void SysTimerInterrupt(void)
{
   // Get samples from sensors.
   MTouchAcquisition();
}
```

Quadro 9 – Ficheiro InitTeclado.c

```
void Proc_ONOFF_LAMP1(void){

  switch (Estado_LAMP1){
    case LAMP1OFF:
      if(MTouchGetSensorState(5) == SENSOR_PRESSED){  // le estado do botao
        if (nivelcarga1 == 0){
          nivelcarga1 = nivelcarga1mem;
        }

        MiApp_FlushTx();
        MiApp_WriteData(0x11); // comando feedback saida 1
        MiApp_WriteData(nivelcarga1);
        MiApp_BroadcastPacket(FALSE);

        Estado_LAMP1 = LAMP1ON;
      }
      break;
    case LAMP1ON:
      if(MTouchGetSensorState(5) == SENSOR_PRESSED){
        if (nivelcarga1 != 0){
          nivelcarga1mem = nivelcarga1;
          nivelcarga1 = 0;
        }

        MiApp_FlushTx();
        MiApp_WriteData(0x11); // comando feedback saida 1
        MiApp_WriteData(nivelcarga1);
        MiApp_BroadcastPacket(FALSE);

        Estado_LAMP1 = LAMP1OFF;
      }
      break;
    default:
      Estado_LAMP1 = LAMP1OFF;
      break;
  }
}  // Proc_ONOFF_LAMP1()

void Proc_ONOFF_LAMP2(void){

  switch (Estado_LAMP2) {
    case LAMP2OFF:
      if(MTouchGetSensorState(0) == SENSOR_PRESSED){
        if (nivelcarga2 == 0){
          nivelcarga2 = nivelcarga2mem;
        }


        MiApp_FlushTx();
        MiApp_WriteData(0x12); // comando feedback saida 2
        MiApp_WriteData(nivelcarga2);
        MiApp_BroadcastPacket(FALSE);

        Estado_LAMP2 = LAMP2ON;
      }
      break;
    case LAMP2ON:
      if(MTouchGetSensorState(0) == SENSOR_PRESSED){
        if (nivelcarga2 != 0){
          nivelcarga2mem = nivelcarga2;
          nivelcarga2 = 0;
        }

        MiApp_FlushTx();
        MiApp_WriteData(0x12); // comando feedback saida 2
        MiApp_WriteData(nivelcarga2);
        MiApp_BroadcastPacket(FALSE);

        Estado_LAMP2 = LAMP2OFF;
      }
```

```
      break;
    default:
      Estado_LAMP2 = LAMP2OFF;
      break;
  }
}  // Proc_ONOFF_LAMP2()

void Proc_MaisMenos_LAMP1(void){

  if(MTouchGetSensorState(2) == SENSOR_PRESSED){
    if (Estado_LAMP1 == LAMP1OFF){
      // nao faz nada
    } else{
        if (nivelcarga1 == Nivelmaxcargas){
          // nao faz nada, ja se encontra no nivel maximo
        }
        if (nivelcarga1 >= 1 && nivelcarga1 <= (Nivelmaxcargas-1)){
          nivelcarga1 = nivelcarga1+1;
        }
    }

    MiApp_FlushTx();
    MiApp_WriteData(0x11); // comando feedback saida 1
    MiApp_WriteData(nivelcarga1);
    MiApp_BroadcastPacket(FALSE);

  }  // leitura tecla "+" da LAMP1

  if(MTouchGetSensorState(4) == SENSOR_PRESSED){
    if (Estado_LAMP1 == LAMP1OFF){
      // nao faz nada
    } else{
        if (nivelcarga1 == 1){
          // nao faz nada, ja se encontra no nivel minimo
        }
        if (nivelcarga1 > 1 && nivelcarga1 <= Nivelmaxcargas){
          nivelcarga1 = nivelcarga1-1;
        }
    }

    MiApp_FlushTx();
    MiApp_WriteData(0x11); // comando feedback saida 1
    MiApp_WriteData(nivelcarga1);
    MiApp_BroadcastPacket(FALSE);

  }  // leitura tecla "-" da LAMP1
}  // Proc_MaisMenos_LAMP1()

void Proc_MaisMenos_LAMP2(void){

  if(MTouchGetSensorState(3) == SENSOR_PRESSED){
    if (Estado_LAMP2 == LAMP2OFF){
      // nao faz nada
    } else{
        if (nivelcarga2 == Nivelmaxcargas){
          // nao faz nada, ja se encontra no nivel maximo
        }
        if (nivelcarga2 >= 1 && nivelcarga2 <= (Nivelmaxcargas-1)){
          nivelcarga2 = nivelcarga2+1;
        }
    }

    MiApp_FlushTx();
    MiApp_WriteData(0x12); // comando feedback saida 1
    MiApp_WriteData(nivelcarga2);
    MiApp_BroadcastPacket(FALSE);

  }  // leitura tecla "+" da LAMP2

  if(MTouchGetSensorState(1) == SENSOR_PRESSED){
```

```
    if (Estado_LAMP2 == LAMP2OFF){
      // nao faz nada
    } else{
        if (nivelcarga2 == 1){
          // nao faz nada, ja se encontra no nivel minimo
        }
        if (nivelcarga2 > 1 && nivelcarga2 <= Nivelmaxcargas){
          nivelcarga2 = nivelcarga2-1;
        }
    }

    MiApp_FlushTx();
    MiApp_WriteData(0x12); // comando feedback saida 1
    MiApp_WriteData(nivelcarga2);
    MiApp_BroadcastPacket(FALSE);

  }  // leitura tecla "-" da LAMP2
}  // Proc_MaisMenos_LAMP2()
```

Quadro 10 – Ficheiro EstadoTeclado.c

```
void detectmov(int carga1, int carga2){

  int valoradcmov;

  initADC(inputscanreg_movesensor); // ADC.h
  valoradcmov = readADC(input_movesensor);  //valor de tensao do foto
                                            //transistor

  switch (carga1) {
    case 1:
      if (valoradcmov > 0x30C) {
        nivelcarga1 = Nivelarranquecarga1;
        LED1=1; // liga so para efeitos de debug
        LED2=1; // liga so para efeitos de debug
        LED3=1; // liga so para efeitos de debug
        LED4=1; // liga so para efeitos de debug
        __delay_ms(tempoonsensormovedeteta); // carga temporizada

      } else {
          nivelcarga1 = 0;
          LED1=0;
          LED2=0;
          LED3=0;
          LED4=0;
        }
      break;

    default:
      break;
  }

  switch (carga2) {
    case 1:
      if (valoradcmov > 0x30C) {
        nivelcarga2 = Nivelarranquecarga2;

      } else {
          nivelcarga2 = 0;

        }
      break;

    default:
      break;
  }
}  // detectmov()
```

Quadro 11 – Implementação da função detectmov()