Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Mestrado em Engenharia Eletrónica e de Computadores

Marco Caldas – 7804

Pedro Faria – 7848

Tiago Miranda – 7537

Pedro Silva – 12345

Trabalho Pratico sobre a

Identificação do Condutor e ajuste do veículo

Laboratórios Integrados II

Julho de 2016

Resumo

Este projeto consiste no desenvolvimento e implementação de um sistema que permita que um automóvel reconheça a identidade do seu condutor.

A verificação da identidade do condutor será efetuada através de visão por computador, com recurso a imagens obtidas por uma câmara apontada à face deste.

O módulo de processamento de imagem deverá comparar a face presente na imagem com faces de referência existentes numa base de dados com os condutores previamente introduzidos no sistema. O processamento de imagem será implementado em PC. A comunicação entre o PC e os restantes dispositivos do automóvel será efetuada através do barramento CAN, um *standard* de comunicação comum nos automóveis atuais.

Este trabalho compreende portanto o desenvolvimento de um sistema integrado que engloba projeto de *hardware* e *software*.

Índice

Índice de Figuras

1 Introdução

A indústria automóvel procura constantemente novas formas de adaptar o funcionamento dos seus produtos às conveniências e conforto do cliente.

Este projeto consiste no desenvolvimento e implementação de um sistema que permita que um automóvel reconheça a identidade do seu condutor, o que permitirá o ajuste automático de parâmetros do interior do automóvel (posição do banco, volume do rádio, etc.) às preferências do condutor que estiver a conduzi-lo em cada momento, conforme representado na figura 1.

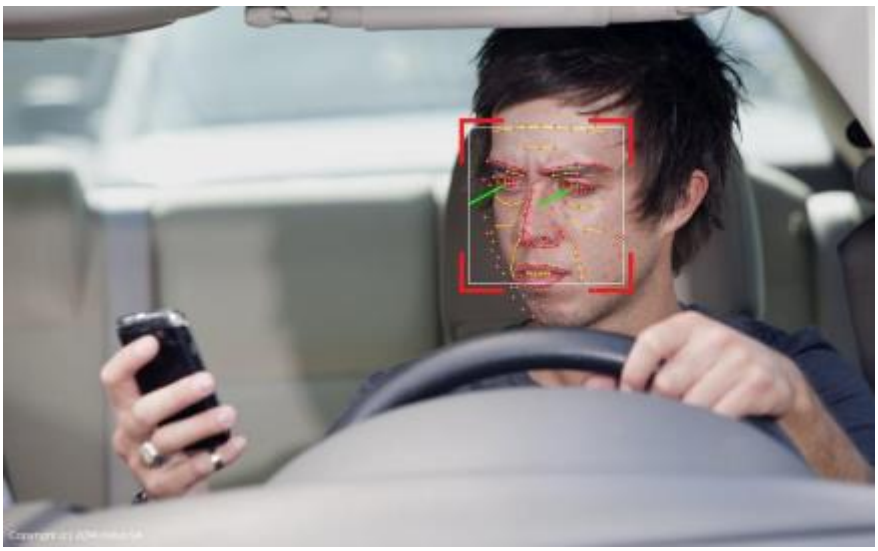

Figura 1 - Reconhecimento da Identidade do Condutor

A verificação da identidade do condutor será efetuada através de visão por computador, com recurso a imagens obtidas por uma câmara apontada à face deste, tipicamente montada junto ao retrovisor do automóvel. O módulo de processamento de imagem deverá comparar a face presente na imagem com faces de referência existentes numa base de dados com os condutores previamente introduzidos no sistema. O processamento de imagem será implementado em PC. A comunicação entre o PC e os restantes dispositivos do automóvel será efetuada através do barramento CAN, um *standard* de comunicação comum nos automóveis atuais.

Este trabalho compreende portanto o desenvolvimento de um sistema integrado que engloba projeto de *hardware* e software. Neste projeto, a aplicação que correr no PC implementa funções de processamento e análise de imagem e comunica com um dispositivo através de USB, que por sua vez permite a comunicação com uma rede industrial CAN.

Para tal o trabalho foi dividido em várias partes deforma a que a adquisição dos dados seja mais focada na análise do condutor, em seguida iremos abordar os procedimentos necessários para a adquisição do sinal.

1.1 Proposta de Trabalho

O bloco baseado no PC deverá constituir um módulo de identificação de faces de forma a ativar operações no interior do veículo, conforme figura 2.

O bloco baseado em sistemas embebidos incluirá o desenvolvimento de dois módulos de *hardware*:

- Módulo master, que estará ligado ao PC, e será responsável pela gestão e transmissão dos dados resultantes da aplicação de processamento e análise de imagem, através de uma rede de comunicação industrial baseada no protocolo CAN BUS.
- Módulo de expansão I/O, que estará ligado ao barramento CAN, traduzirá nas suas saídas os dados resultantes da análise de imagem, transmitidos através do módulo master.

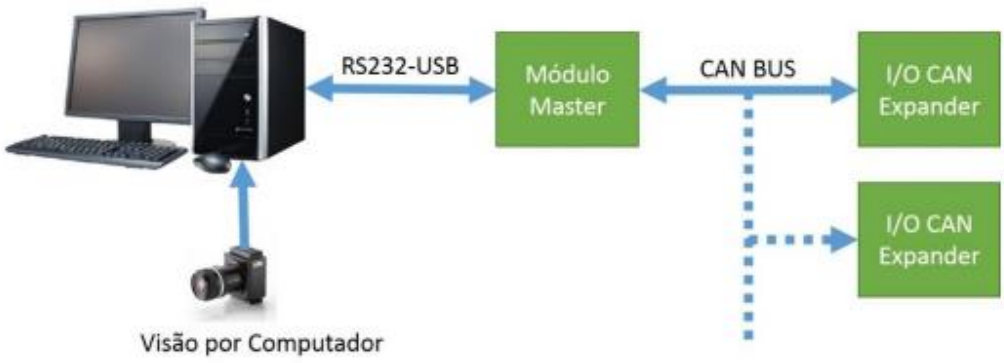

Figura 2- Arquitetura do Sistema

1.1.1 Módulo Master

Este módulo de *hardware* será utilizado para realizar a interface entre o módulo de *software* desenvolvido no PC e a rede CAN, conforme representado na figura 3.

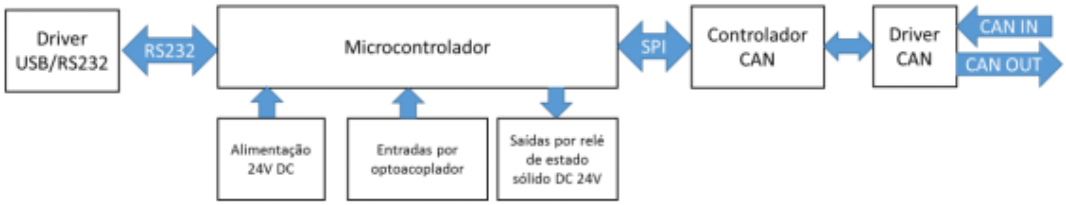

Figura 3 - Esquema do Módulo Master

Para tal, no projeto deverão ser consideradas as seguintes características:

- Ligação ao PC por USB – deverá para isso incorporar um conversor USB/RS232 (FT232R);
- Transmissão de dados através do protocolo CAN – deverá para isso integrar um microcontrolador (Atmel 328), que comunicará com o PC através do protocolo RS232, e com um controlador Stand-Alone CAN (MCP2515) através da interface SPI. A regularização dos níveis de tensão para rede CAN deverá ser estabelecida através de um driver CAN (MCP2551SN);
- Alimentação por fonte externa 12Vdc;
- Disponibilização de um conjunto de entradas (por optoacoplador SFH618A) e saídas (por relé de estado sólido – G3MB-202P) com indicação luminosa do estado.

1.1.2 Módulo de expansão I/O CAN (hardware).

Este módulo de *hardware* será utilizado como módulo de expansão da rede CAN de entradas e saídas, conforme figura 4.

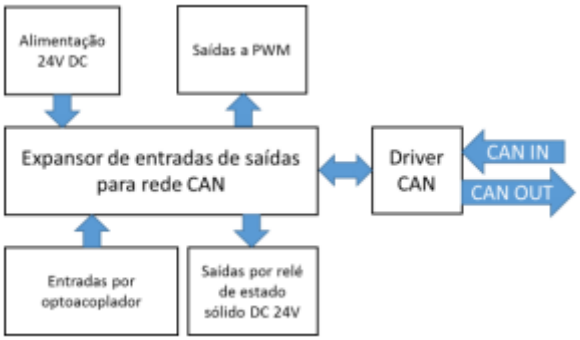

Figura 4 - Esquema módulo Slave

Para tal, no projeto deverão ser consideradas as seguintes características:

- A implementação do protocolo CAN deverá ser assegurada por um bloco expansor de entradas e saídas (MCP25050).
- A regularização dos níveis de tensão para rede CAN deverá ser estabelecida através de um driver CAN (MCP2551SN);
- CAN In e CAN Out;
- Alimentação por fonte externa 12 V DC;
- Disponibilização de um conjunto de entradas (por optoacoplador SFH618A) e saídas (por relé de estado sólido – G3MB-202P) com indicação luminosa do estado

1.1.3 Módulo de comunicação PC <-> Módulo Master CAN (software)

Definição de um protocolo de comunicação, no PC e no módulo Master, que permita:

- Enviar os dados da análise de imagem para o PC com identificação do módulo CAN para onde os dados serão transmitidos;
- Pedir ao módulo Master o código de identificação de todos os módulos ligados à rede CAN.

1.1.4 Módulo de comunicação Módulo Master CAN <-> Módulo de expansão I/O CAN (software)

Definição de um módulo de comunicação no módulo Master e configuração do módulo de expansão, que permita:

- Configurar os registos do controlador Stand-Alone CAN para que este funcione no modo pretendido;
- Converter as mensagens recebidas do PC no protocolo CAN;
- Transmitir a mensagem através da rede CAN;
- Consultar o código de identificação de todos os módulos ligados à rede CAN;

1.1.5 Módulo de processamento de imagem: identificação de faces para ações no automóvel.

Este projeto consiste no desenvolvimento e implementação de um sistema que permita que um automóvel reconheça a identidade do seu condutor, o que permitirá o ajuste automático de parâmetros do interior do automóvel (posição do banco, volume do rádio, etc.)

As principais tarefas associadas ao processamento de imagem são:

1) Identificar o melhor setup para aquisição da imagem (câmara, iluminação, ótica, etc);

2) Calibrar a distorção radial e tangencial provocada pela lente;

3) Detetar a(s) face(s) na imagem, escolhendo a principal ;

4) Segmentar a face;

5) Efetuar uma extração de características da face detetada:

a) Definir quais as características relevantes a extrair;

6) Comparar características da face detetada com características de faces existentes na base de dados;

Os dados a enviar para a rede CAN são:

- Se foi identificada uma face
- A identidade da face identificada;

1.1.6 Módulo de interface com o utilizador

Deverá ser desenvolvida uma interface com o utilizador que permita uma fácil e intuitiva interação com o sistema.

Esta deverá respeitar os seguintes requisitos:

- O interface deverá indicar o estado da ligação ao Módulo Master
- O interface deverá poder listar todos os módulos ligados à rede CAN
- A imagem da câmara deverá aparecer em tempo real no ecrã o PC
- A deteção de faces deverá ser assinalada na imagem
- A não deteção de faces deverá ser assinada numa ou mais saídas digitais de um ou mais módulos de expansão I/O; a(s) saída(s) dos módulos a ativar deverão ser configuráveis
- O interface deverá permitir a criação, edição e remoção de registos de faces na base de dados; para cada face deverá ser possível configurar quais as saídas de quais módulos de expansão I/O deverão ser ativados/desativados
- O interface deverá permitir a receção de pedidos de criação de novo registo de face na base de dados provenientes do módulo master, com a face identificada no momento;

2 Projeção do trabalho

No desenvolvimento deste projeto, foi necessário fazer uma análise e estudo do que foi proposto e definiu-se o trabalho em quatro partes:

- Esquematização do projeto proposto e fazer protótipos dos módulos de *hardware,* onde dividimos o projeto em duas partes modulo *Master* e Modulo *Slave*;
- Programação MCP25050, através do *software* fornecido pelo docente; (placa *slave);*
- Programação do AtMega328 através do Arduíno e o MCP2515. (módulo master)
- Projeção e criação do *software* de adquisição de fases.

Para a criação da placa foi utilizado o *software* EAGLE, onde criamos duas placas, sendo que as entradas são protegidas por optocopoladores e varístores, a saída funcionam com reles de estado solido com indicação luminosa do seu estado, cada ciruito integrado tem referências propiás de ligação como o cristal que foram seguidas e referenciadas pelo *DataSheet*

O funcionamento do módulo *slave* vai depender maioritariamente da programação do MCP25050. Esta programação é feita através de um *software* dedicado á programação de MCP25050 chamado “MCP250xxProgrammer.exe”. Este possui uma interface gráfico que torna a programação do mesmo mais fácil e intuitiva a programação dos registos de GPIOs e CAN, onde define-se quais os *GPIOs* de entrada e de saída e a programação dos filtros e máscaras das mensagens.

Quanto ao módulo master, o funcionamento deste vai depender da programação do AtMega328 que está em comunicação com o MCP2515 que faz a interação com a rede CAN. A programação microcontrolador é através do IDE do Arduíno que possui bibliotecas (mcp_can.h) de interação com dipositivos CAN. Este módulo faz a interface entre o módulo de software desenvolvido no PC e a rede CAN.

Para o processamento de imagem do computador foi criado um interface com o utilizador. Este faz a deteção da fase do condutor que guarda as frames e deteta o condutor que aparece na imagem.

2.1 Desenvolvimento do Projeto

Neste capítulo é descrito o processo de construção do projeto. Cada subcapítulo representa cada uma do estudo mencionado na introdução. Mencionando os passos que foram efetuados assim como a descrição técnica da elaboração do projeto.

2.1.1 Criação da PCB

Um dos objetivos deste trabalho é o desenho de uma placa de circuito impresso (PCB) para a comunicação (can e série), assim como a atuação do reles de estado solido, que representam as saídas e a leitura do estado dos botões que representam as entradas, esta parte do trabalho esta dividida em duas partes, conforme já foi mencionado, placa principal e placa secundária. (Master / Slave).

2.1.1.1 Placa Principal (Master)

A placa Principal é composta por 3 componentes principais o atmega 328, mcp2515 e o mcp 2551. Sendo que para cada componente são necessário periféricos conforme descritos no datasheet, também é constituída por reguladores de tensão 7805 que garantem uma tensão estabilizada de 5Vdc, também constituem a placa os reles de estado sólidos para as saídas, assim como os optoacopoladores para proteger a entrada, na figura 5 esta representado o esquemático da placa principal.

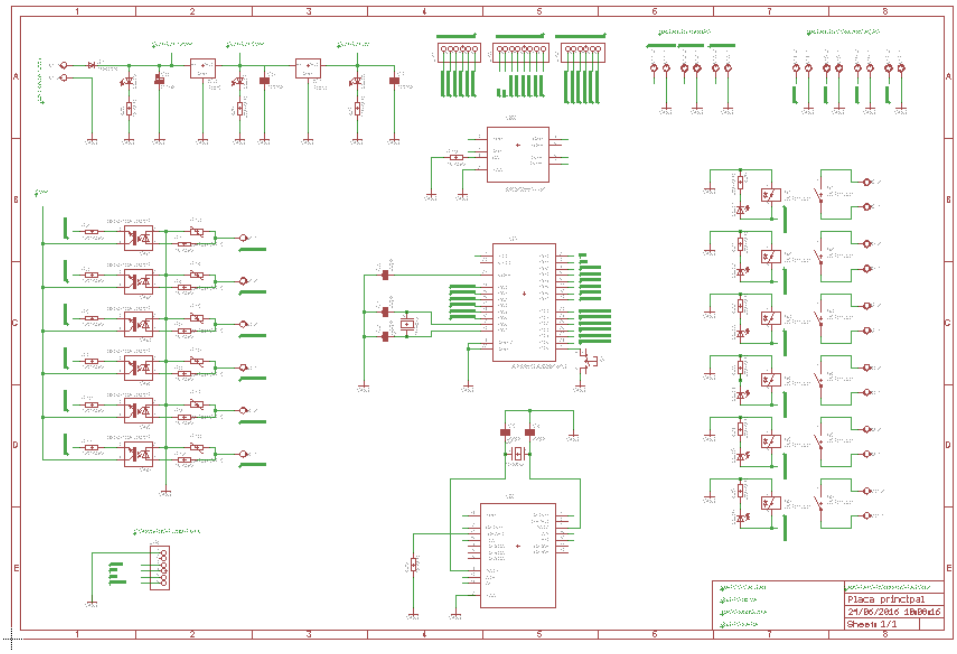

Figura 5 - Esquema Eagle da placa principal

2.1.1.2 Placa Principal Atmega 328

O Atmega é um microcontrolador que gere as comunicações serie (Pc- atmega328) e can (master-slaves), acoplado a este componente existe 2 entradas e 6 saídas digitais e o controlador stand-alone MCP 2515 com o qual comunica por SPI, conforme na figura 6.

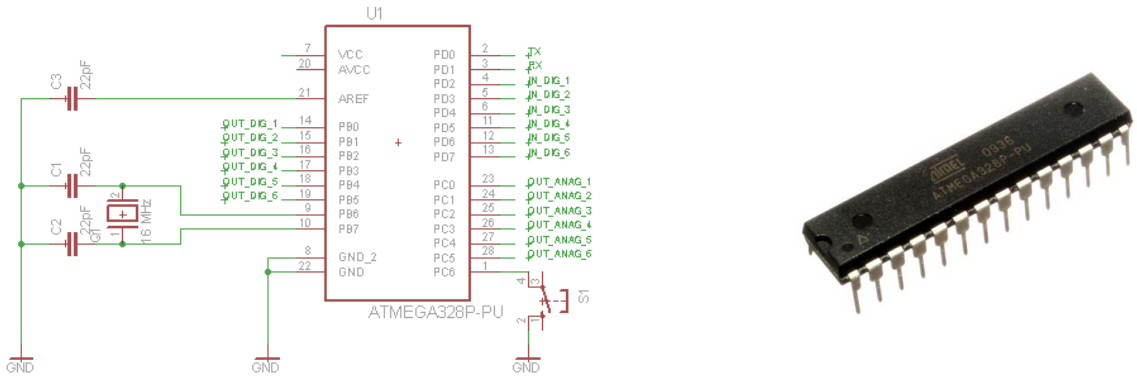

Figura 6 - Atmega 328

2.1.1.3 Placa Principal MCP 2515

O MCP2515 é um controlador stand –alone implementa o CAN. É capaz de transmitir e receber dados em formato standard, extended data e remote frames. O MCP2515 tem duas máscaras de aceitação e seis filtros de aceitação que são usados para filtrar mensagens indesejadas, reduzindo assim a sobrecarga ao microcontrolador. A interface com o microcontrolador e feita através de SPI, conforme na figura 7.

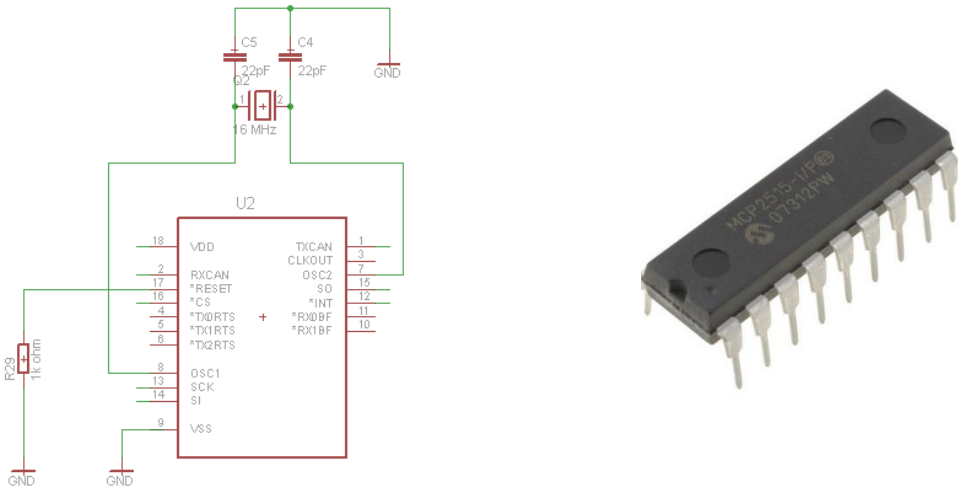

Figura 7 - MCP 2515

2.1.1.4 Placa Principal MCP 2551

O MCP2551 é um dispositivo que tem a função de interface entre o controlador CAN (MCP 2515) e o barramento CAN, conforme na figura 8.

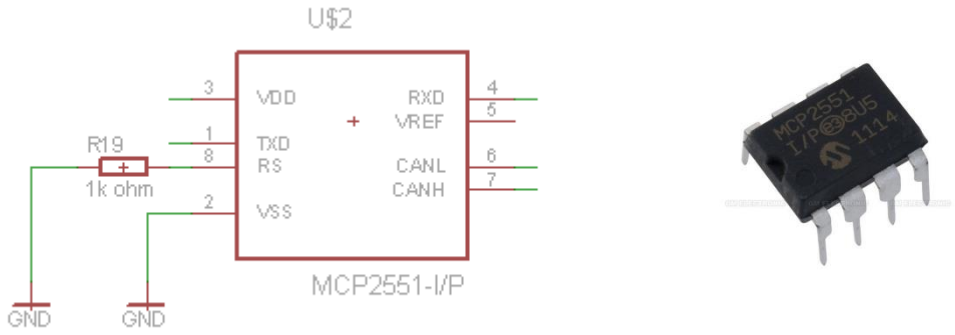

Figura 8 - MCP 2551

2.1.1.5 Comunicação com o PC

Para a comunicação com o PC foi usado um módulo conversor Serie-usb, foi colocado na placa pinos de interface com este conversor, na figura 9 esta representado o conversor.

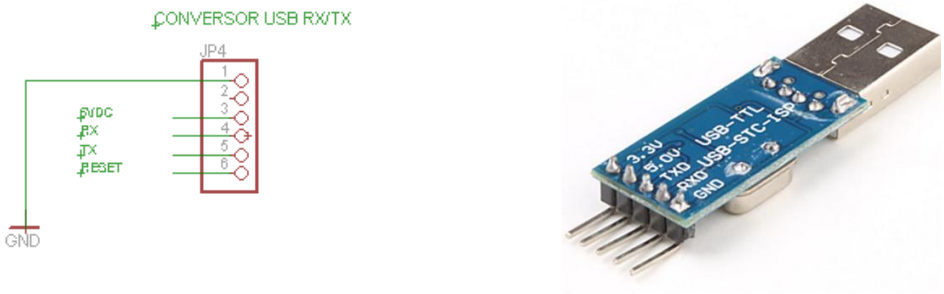

Figura 9 - Conversor Serie- USB

2.1.1.6 PCB

A placa de circuito impresso foi desenhada tendo em conta as normas e regras para a concessão da mesma assim como as limitações da máquina de produção de PCB da escola, utilizando o ficheiro previamente definido, conforme figura 10.

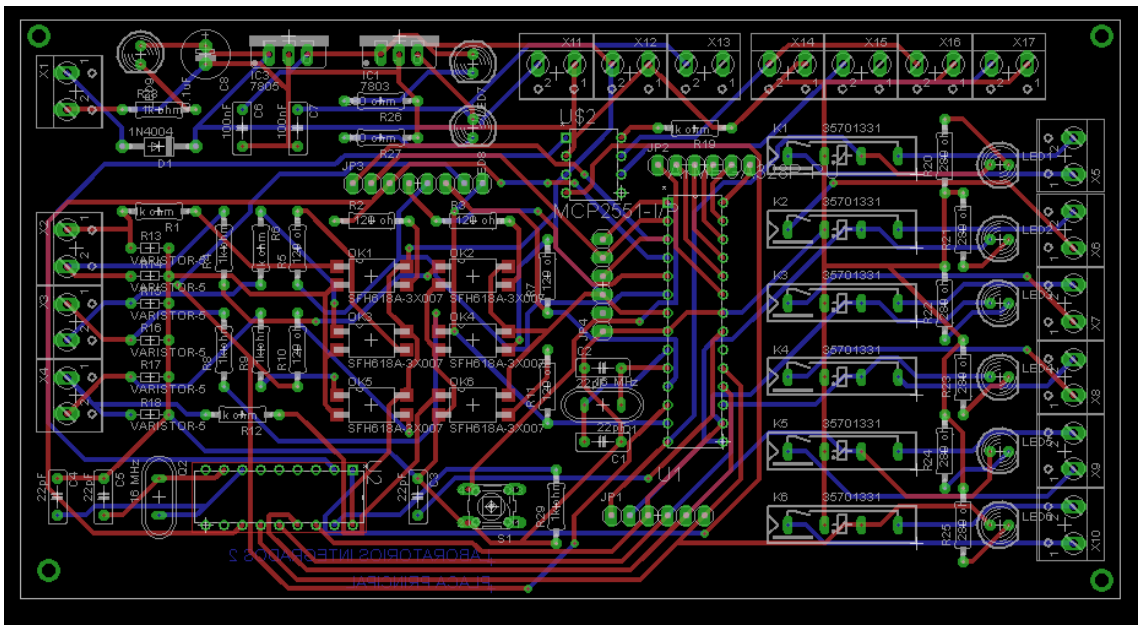

Figura 10 - PCB

2.1.1.7 Placa Secundaria (Slave)

A placa secundária é composta por 3 componentes principais o MCP25050, mcp2515 e o mcp 2551. Sendo que para cada componente são necessário periféricos conforme descritos no datasheet, também é constituída por reguladores de tensão 7805 que garantem uma tensão estabilizada de 5Vdc, também constituem a placa os reles de estado sólidos para as saídas, assim como os optoacopoladores para proteger a entrada, na figura 11 esta representado o esquemático da placa principal.

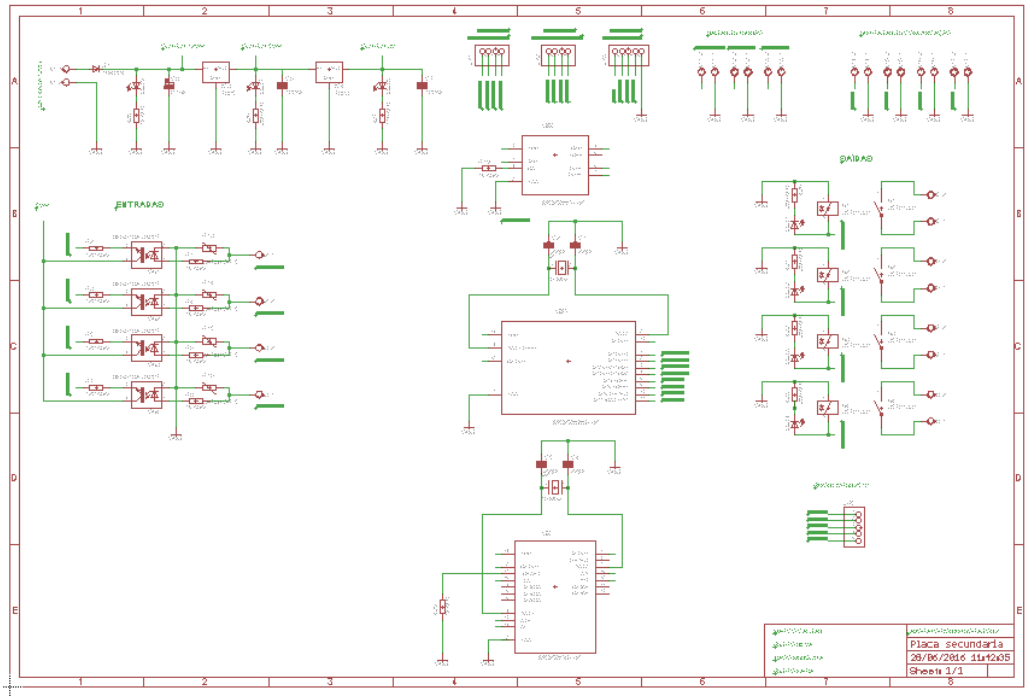

Figura 11 - Placa secundaria

2.1.1.8 Fonte de alimentação

Para a fonte de alimentação que foi projetada da mesma forma para ambas as placas, colocamos reguladores de tensão 7805, 7803 aplicando condensadores de filtragem, led´s de forma a sinalizar os níveis de tensão da fonte, na figura 12 esta representada a fonte

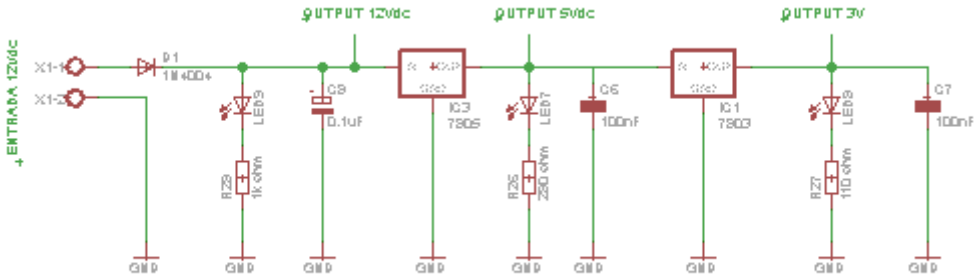

Figura 12 - Fonte de alimentação

2.1.1.9 Entradas digitais protegidas

Para a proteção das entradas digitais colocaram-se optoacopoladores, ligados com resistências e varístores para proteger dos impulsos indesejados, na figura 13 esta representado a entrada.

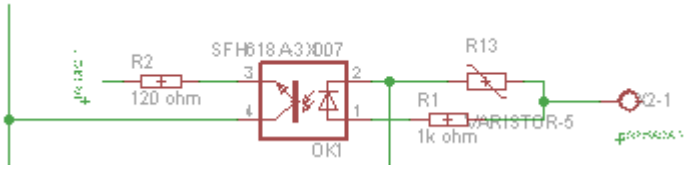

Figura 13 - Entrada Protegida

2.1.1.10 Saídas com reles

Na saída foram colocados reles de estado sólidos, com os leds deforma a indicarem o estado da saída, conforme figura 14.

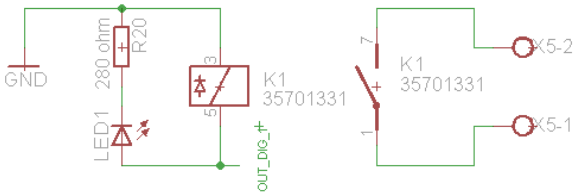

Figura 14 – Saída

2.1.2 Programação do MCP 25050

Ao programar do MCP25050 pelo *software* "MCP250xxProgrammer.exe" tem de responder de acordo com o pretendido e tem de existir coerência com o módulo master.

2.1.2.1 Análise

Os atributos mais relevantes da rede CAN, para este projeto, são as trocas de mensagens entre dispositivos CAN.

Sendo assim, existem 2 tipo de CAN IDs de mensagens: *Standard e Extended*. A principal diferença entre estes é que o *Extended* consegue possuir mais configurações de *ID*. Estes *ID* são identificações dos filtros e as máscaras de aceitação das mensagens. Estes vão filtrar as mensagens indesejadas, reduzindo assim a sobrecarga ao microcontrolador. Para este projeto será utilizado os *Standard Ids*.

De acordo com o datasheet, o MCP25050 possui 3 tipo de mensagens:

- Input massagens: são mensagens que o MCP25050 recebe para modificar os seus registos.
- Information Request Messages (IRM): são mensagens que o MCP25050 recebe para pedidos de informação.
- Output Messages: são mensagens que o MCP25050 transmite que possui as respostas pedidas pelos IRM.

O MCP25050 possui 2 barramentos de receção de mensagens (RXF0 e RXF1). Cada um destes barramentos possui um filtro de aceitação que estão diretamente relacionados com os 2 tipos de receção de mensagens. Se o MCP25050 receber um IRM irá para o buffer 0 e se receber um input message irá para o buffer 1.

Quanto á transmissão de mensagens, este possui 3 barramentos: TXID0,TXID1 e TXID2. O TXID0 contem o id usado em mensagens "on bus", o TXID1 contem o id usado para o envio de acknowledge e o TXID2 contem o id em mensagens de edge detection, threshold saturation, etc.

Na página 25 do *datasheet* possui as configurações das mensagens para cada tipo de mensagem. Também possui a informação que os ID de Input Message e IRM devem ser diferentes.

2.1.2.2 Programação do GPIOs

Portanto, em relação á programação dos GPIOs, definiu-se:

GPIOs de entrada: 5, 6, 10 e 11.

GPIOs de saída: 1, 2, 3, 4.

Neste sentido, os pinos de entrada estarão conectados a botões e os de saída estarão conectados aos relés e a LEDs para indicar o estado do mesmo.

Inclusive, com a existência de 'Edge detect' quando os GPIOs de entrada passam para o estado baixo, provocam uma interrupção ao master, enviando uma trama de dados, na figura 15 esta representado o GPIO

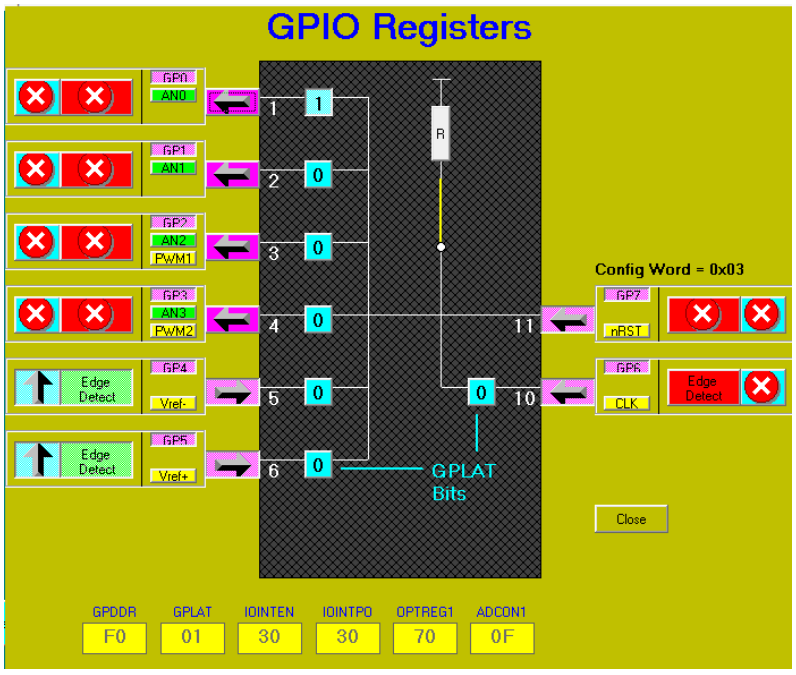

Figura 15 - Registos GPIO

2.1.2.3 Programação dos registos CAN

Como já mencionado, para o microcontrolador saber quais mensagens aceitar e interpretar é necessário programar os filtros e as máscaras de aceitação. Para isso define-se os ID dos barramentos de receção e transmissão de mensagens.

Na configuração dos registos para mensagens de leitura dos registos A/D e para mensagens de escrita em registos os bits 0,1 e 2 do identificador standard devem ser 0.

Partindo deste principio e considerando que os ids utilizados serão todos do tipo standard, o registo TXDI0SIDH (8 msb do id standard) pode tomar qualquer valor. Os bits 5-7 do registo TXID0SIDL (3 lsb do id standard) devem ser 0. O bit 3 (EXIDE) deverá ser também 0 pois o identificador pretendido é standard.

Os restantes bits dos registos TXID0SIDL, TXIDnEID8 e TXIDnEID0 são indiferentes pois representam o extended ID.

O RXM contem a máscara de aceitação, portanto é importante que seja igual ao ID TXID0 isto para que todas as mensagens sejam aceites pala máscara.

O registo RXnSIDH (8 msb do id standard) pode tomar qualquer valor. Os bits 5-7 do registo RXnSIDL (3 lsb do id standard) devem ser 0, como visto em cima. O bit 3 (EXIDE) deverá ser também 0 pois o identificador pretendido é standard.

Os restantes bits dos registos RXnSIDL, RXnEID8 e RXnEID0 são indiferentes pois representam o extended ID, Na figura 16 esta representado o registos de CAN

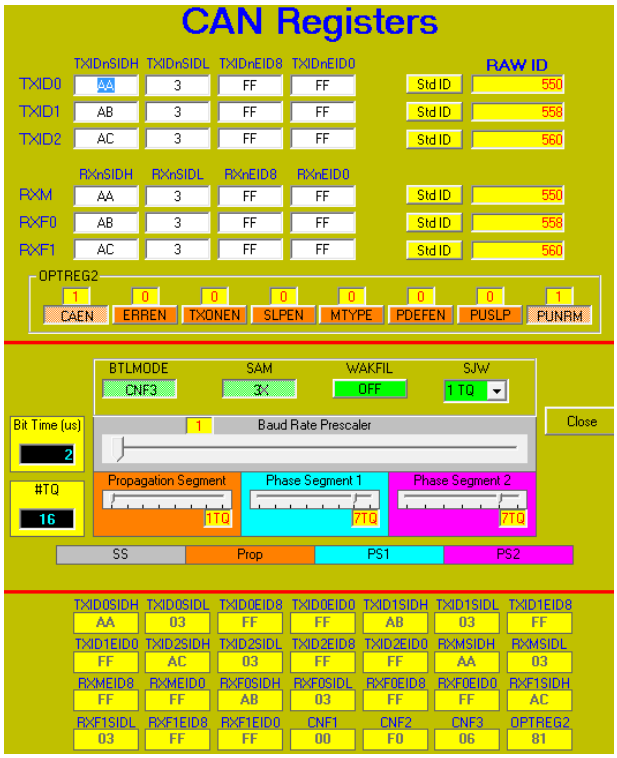

Figura 16 - Registos de CAN

2.1.3 Programação do Arduíno

O software usado para o desenvolvimento do código para o microcontrolador foi o software disponibilizado pela Arduíno, foi escolhido este software devido á existência de bibliotecas para CAN tornando mais fácil a sua implementação.

2.1.3.1 Biblioteca

A biblioteca "mcp_can.h" possui várias funções de interação com a rede CAN. As mais importantes e utilizados neste projeto são as funções de transmissão e receção de mensagens.

```
public:
    MCP_CAN(INT8U _CS);
    INT8U begin(INT8U speedset);                                    /* init can                    */
    INT8U init_Mask(INT8U num, INT8U ext, INT32U ulData);           /* init Masks                  */
    INT8U init_Filt(INT8U num, INT8U ext, INT32U ulData);           /* init filters                */
    INT8U sendMsgBuf(INT32U id, INT8U ext, INT8U rtr, INT8U len, INT8U *buf);   /* send buf
    INT8U sendMsgBuf(INT32U id, INT8U ext, INT8U len, INT8U *buf);   /* send buf                    */
    INT8U readMsgBuf(INT8U *len, INT8U *buf);                       /* read buf                    */
    INT8U readMsgBufID(INT32U *ID, INT8U *len, INT8U *buf);         /* read buf with object ID     */
    INT8U checkReceive(void);                                       /* if something received       */
    INT8U checkError(void);                                         /* if something error          */
    INT32U getCanId(void);                                          /* get can id when receive     */
    INT8U isRemoteRequest(void);                                    /* get RR flag when receive    */
    INT8U isExtendedFrame(void);                                    /* did we recieve 29bit frame? */
};
```

2.1.3.2 Comunicação MCP25050

Para poder trocar mensagens com o microcontrolador, volta-se a verificar as tabelas do datasheet da página 25. Nesta tabela vê-se os comandos das mensagens que podem ser transmitidas e recebidas dependendo do tipo de mensagem que se pretende utilizar.

É necessário enviar mensagens para alterar os registos dos GPIO, sabendo o endereço pretendido como as posições dos GPIOs referente aos bits do registo, utilizando a Input Message.

Para saber o estado de informação dos GPIO's utilizadmos uma IRM, enviando para o MCP o seu respetivo IRM.

2.1.3.3 Atmega328

A programação do AtMega328 faz a comunicação entre o PC e os módulos CAN. Tem de existir coerência com os comandos estabelecidos pelo processamento de imagem e, também, com o módulo slave.
As principais funcionalidades passa pela troca de mensagens entre a rede CAN e o PC. Faz a leitura da ativação dos botões a receção de informação sobre a ativação das GPIOs de saída.

2.1.4 Processamento de Imagem

O *software* utilizado foi Visual Studio com a linguagem de programação C# com o add OpenCV, deteta as fases do utilizador e grava em dez vezes a mesma face de forma a captar várias perspetivas de imagem, na figura 17 esta representado o interface com o utilizador.

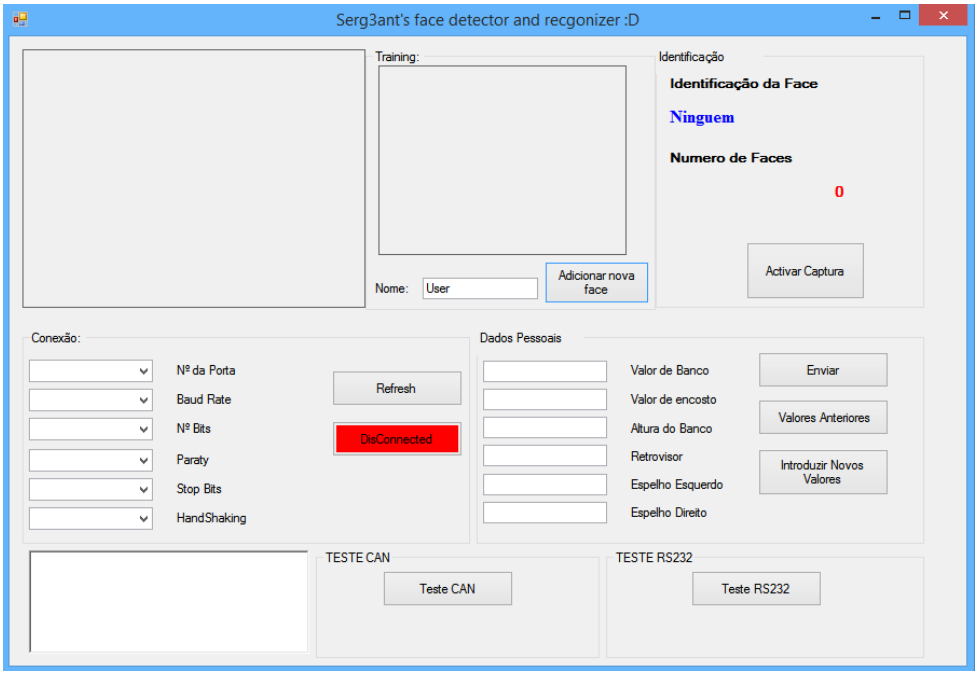

Figura 17 - Interface com o Utilizador

Podemos ver que existem vários botões para selecionar diversas funções, para que o programa inicie temos que carregar em ativar captura esse botão ativa a camara, bem conforme figura 18

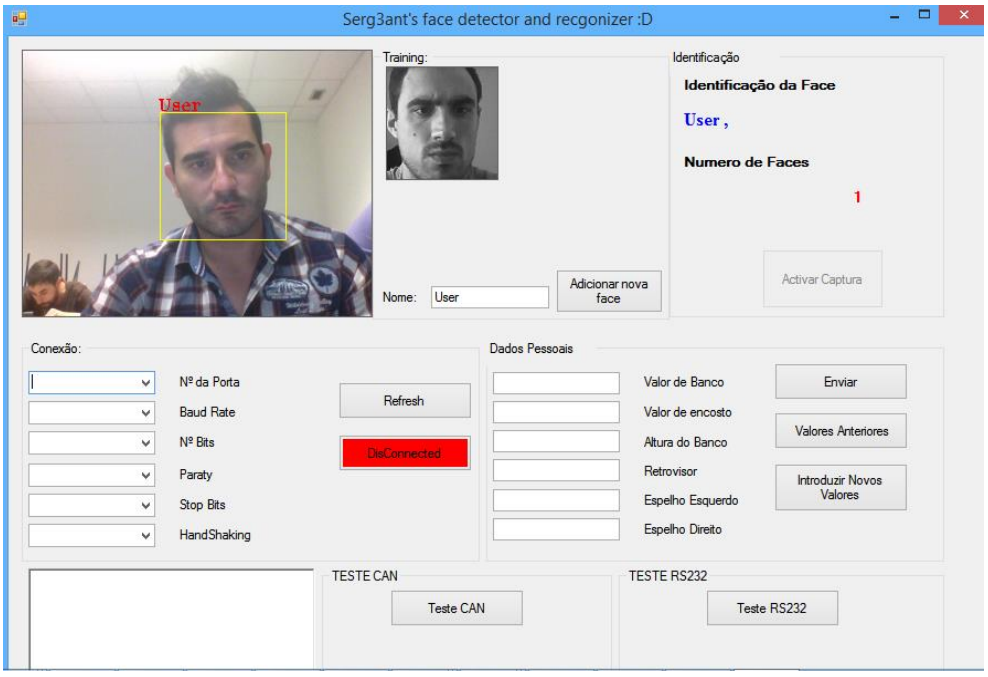

Figura 18 - Captura do utilizador

Colocando o nome de utilizador e carregando no botão adicionar nova face, e tirada uma fotografia que é guardada, o programa pede para tirar 10 fotografias, após estas serem tiradas o programa reconhece automaticamente o utilizador.

Nos dados pessoais é possível colocar os valores referentes ao carro e respetivo a cada condutor, depois de reconhecer o condutor novamente carregamos no botão valores anteriores e coloca já os dados que foram gravados em ficheiro txt.

O programa permite fazer um teste RS232 de comunicação onde testa a ligação com o dispositivo.

Permite ainda configurar a porta serie deforma a garantir a comunicação.

O código usado na captura da imagem é o seguinte:

```
 //Iniciar captura
private void button1_Click(object sender, EventArgs e)
{
    //Initialize the capture device
    grabber = new Capture();
    grabber.QueryFrame();
    //Initialize the FrameGraber event
    Application.Idle += new EventHandler(FrameGrabber);

    button1.Enabled = false;
}
```

O código usado para adicionar novo user é o seguinte:

```
//Adicionar novo user
private void button2_Click(object sender, System.EventArgs e)
{

        try
        {
            //Trained face counter
            ContTrain = ContTrain + 1;

            //Get a gray frame from capture device
            gray = grabber.QueryGrayFrame().Resize(320, 240, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

            //Face Detector
            MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
            face,
            1.2,
            10,
            Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
            new Size(20, 20));

            //Action for each element detected
            foreach (MCvAvgComp f in facesDetected[0])
            {
                TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();
                break;
            }
```

```
//resize face detected image for force to compare the same size with the
//test image with cubic interpolation type method
TrainedFace = result.Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
trainingImages.Add(TrainedFace);
labels.Add(textBox1.Text);

//Show face added in gray scale
imageBox1.Image = TrainedFace;

//Write the number of triained faces in a file text for further load
File.WriteAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", trainingImages.ToArray().Length.ToString()

//Write the labels of triained faces in a file text for further load


    for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
    {

        trainingImages.ToArray()[i - 1].Save(Application.StartupPath + "/TrainedFaces/" + "face" + i + ".bmp");


        File.AppendAllText(Application.StartupPath + "/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");
    }

if (j != 0)
{
    MessageBox.Show("Faltam retirar mais " +(j - 1)+" click Novamente");
    j--;
}
if (j == 0)
{
    MessageBox.Show("Captura concluida");
    MessageBox.Show(textBox1.Text + "´s face detected and added :)", "Training OK", MessageBoxButtons.OK, MessageBoxIcon.Information)
    j = 10;
}



          }

      }
          catch
          {
              MessageBox.Show("Enable the face detection first", "Training Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
          }

          //System.Threading.Thread.Sleep(500);


  }

  /// ///////

  //Porta Serie
  private void button8_Click(object sender, EventArgs e)
  {
```

3 Testes e ensaios finais

Para verificar a funcionalidade do projeto, montamos os componentes na BreadBord deforma a testar o seu funcionamento, no entanto verificamos alguns problemas de comunicação entre o Atmega 328 e o MCP25050, verificamos que só é possível programar uma vez o MCP, o Programa em Arduíno consegue comunicar, mas o MCP não consegue receber os dados.

Não foi possível fazer a montagem na PCB, uma vez que só tínhamos componentes para uma montagem, Na figura 19 esta representado a montagem na BreadBord

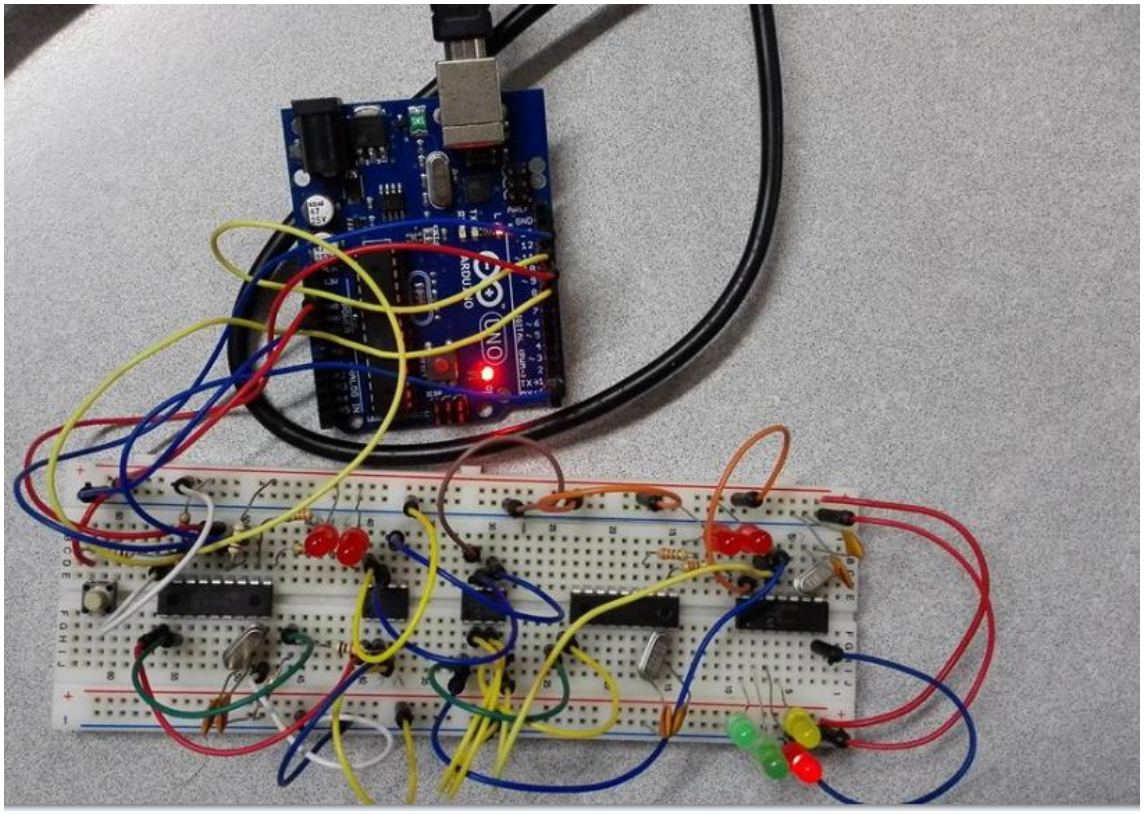

Figura 19 - Montagem na BreadBord

4 Conclusões

O objetivo consistia no desenvolvimento de um sistema para ler as fases do condutor, e outro, baseado em sistemas embebidos, responsável pela comunicação.

A realização deste projeto envolveu diversas áreas entre elas sistemas embebidos, redes, processamento de imagem e concessão de PCB.

Todos os objetivos não foram executados com sucesso. No decorrer do projeto ocorreram alguns problemas, não foi possível montar a PCB, pois os componentes estavam a ser usados na breadbord e as placas foram entregues muito encima da hora.

Quanto ao s*oftware* do módulo master foram executados com sucesso, pois conseguia comunicar perfeitamente com o módulo slave bem como a porta série. O IDE do Arduíno facilitou a programação do mesmo, inclusive as bibliotecas de interação com os dispositivos CAN, também tornou a programação mais intuitiva.

A programação do MCP25050 (módulo slave), também, foram executados com sucesso, cumprindo assim o seu objetivo. A programação por “MCP250xxprogrammer.exe” facilitou bastante o processo de programação dos registos pois tornou o processo fácil e intuitivo.