Instituto Politécnico de Coimbra

Instituto Superior de Engenharia

*i*SEC
Intelligent and Sustainable Educational Campus

Samuel Caetano Augusto Martins Pereira

Trabalho de Projeto para obtenção do Grau de Mestre em

Automação e Comunicações em Sistemas de Energia

COIMBRA

2013

Instituto Politécnico de Coimbra

Instituto Superior de Engenharia

*i*SEC
Intelligent and Sustainable Educational Campus

Orientador:
Nuno Miguel Fonseca Ferreira
Professor Adjunto do Departamento
de Engenharia Electrotécnica, ISEC

Coorientador:
Micael Santos Couceiro
Investigador
Instituto de Sistemas e Robótica
Universidade de Coimbra

Samuel Caetano Augusto Martins Pereira
a9602112@alunos.isec.pt

Trabalho de Projeto para obtenção do Grau de Mestre em
Automação e Comunicações em Sistemas de Energia

COIMBRA
2013

Agradecimentos

Primeiramente, quero agradecer a Deus por estar presente na minha vida e por me dar ânimo e sabedoria para eu concretizar os meus sonhos, conseguindo chegar ao fim de mais uma meta com os objetivos cumpridos ao finalizar esta dissertação.

Quero deixar expresso o meu agradecimento para a minha família que prontamente me tem dado apoio, animado e incentivado para que a minha formação académica vá além, suportando muitas vezes a minha ausência, nomeadamente os meus "filhotes" e a minha esposa.

Quero agradecer também à Escola Profissional Vasconcellos Lebre, onde atualmente leciono, Dr.ª Maria Manuela Coleta Alves, na qualidade da Diretora Pedagógica, Eng.º João Fernandes Duarte Pega, na qualidade de antigo Diretor Geral, e Dr. Nuno Castela Canilho, na qualidade de Diretor Geral, por prontamente me cederem o tempo necessário nesta reta final, que muito útil foi e que, de outro modo, não teria sido possível.

Ao Orientador Dr. Nuno Miguel Fonseca Ferreira que esteve sempre disponível. Ao Coorientador Eng.º Micael Santos Couceiro que muitas vezes me ouviu, que teve algum trabalho acrescido, acreditando sempre que seria possível, gerando motivação muitas vezes onde nem eu próprio via devido aos muitos afazeres onde me envolvo no dia-a-dia.

A todos os meus amigos e a todos quantos tornaram este trabalho possível, o meu sincero agradecimento.

Resumo

Ao longo dos últimos anos, os sistemas de controlo de acessos têm-se tornado cada vez mais sofisticados e diversas medidas de segurança têm vindo a ser implementadas para combater as ameaças de insegurança a vidas humanas e às propriedades. O controlo de entradas não autorizadas em edifícios tem tido um papel fundamental no setor público e privado, usufruindo das mais diversas tecnologias, desde os simples cartões de acesso até aos métodos biométricos, como as impressões digitais, íris e reconhecimento facial. Hoje em dia, os requisitos de segurança nos sistemas de informação têm aumentado significativamente. Além dos requisitos de segurança, a identificação pessoal desempenha um papel muito importante na autenticação e no controlo de acessos.

No contexto educativo, torna-se fundamental o uso destes sistemas, focando-se essencialmente em três níveis: *i*) segurança; *ii*) gestão e controlo de acessos; e *iii*) supervisão e controlo de equipamentos. Ao nível da segurança, torna-se pertinente limitar o acesso ao *campus* da instituição apenas a pessoal autorizado. A gestão e controlo de acessos permite controlar a assiduidade e o acesso dos alunos e funcionários a determinadas áreas do *campus* da instituição, (*e.g.*, departamentos, laboratórios e/ou salas), e em determinado horário (*e.g.*, durante o fim-de-semana). Finalmente, a supervisão e controlo de equipamento torna possível monitorizar os inúmeros sistemas inerentes à instituição (*e.g.*, equipamentos laboratoriais).

Deste modo, em 2011, o Instituto Superior de Engenharia de Coimbra (*ISEC*) iniciou um projeto, denominado de *Intelligent and Sustainable Educational Campus* (*i*SEC), com a perspectiva de desenvolver e integrar um sistema de controlo e gestão de acessos nos diversos polos do Instituto Politécnico de Coimbra (*IPC*). Adicionalmente, este sistema teria de ser sustentável, de baixo custo, de fácil implementação e extensão futura. Para o efeito, optou-se pela identificação por rádio frequência (*RFID – Radio Frequency Identification*), dado esta ser uma das tecnologias mais em voga nos dias de hoje. O número de soluções, investigação de desenvolvimento em torno desta tecnologia ultrapassam largamente as alternativas biométricas, tirando o máximo partido da identificação única a custo significativamente reduzido.

O presente Projeto de Mestrado tem como objetivo aprimorar a fiabilidade do *software* e do *hardware* da solução *iSEC*, reduzindo ainda mais o custo da solução. Adicionalmente, o

projeto visa ainda o desenvolvimento de um interface com o utilizador e respetiva base de dados, validando o mesmo nos polos do *IPC*.

Palavras-chave: *Arduino*; *Display LCD*; Leitor *RFID*; Trinco Eléctrico; Base de Dados; Autenticação; Controlo de Acessos.

Abstract

Over the past few years, access control systems have become increasingly sophisticated and a broad variety of security measures have been implemented to combat the threats against the security of mankind and facilities. The control of unauthorized access has played a key role in both public and private sectors, benefiting from different technologies, from a simple access card, to biometric methods, such as fingerprint, iris and facial recognition. Nowadays, the security requirements in information systems have significantly increased. In addition to the security requirements, personal identification also plays a crucial role in authentication and access control.

In the educational context, it becomes essential to use these systems, focusing mainly on three distinct levels: *i*) security; *ii*) management and access control; and *iii*) equipment monitoring and control. At the security level, it is pertinent to limit the access to the campus only to authorized personnel. The management and access control allows to control the attendance and access from students and staff to certain areas of the campus (*e.g.*, departments, laboratories and/or offices), and at a certain period (*e.g.*, on weekend). Finally, the equipment supervision and control makes it possible to monitor numerous systems inherent to the institution (*e.g.*, laboratory equipment).

In spite of this, in 2011, the Engineering Institute of Coimbra (*ISEC*) has initiated a project called the *Intelligent and Sustainable Educational Campus* (*i*SEC), with the prospect of developing and integrating a control and access management system in the different poles of the Polytechnic Institute of Coimbra (IPC). Additionally, this system should be sustainable, low cost, of easy implementation and future extension. To that end, the radio frequency identification (*RFID*) was the technology considered, being currently widely used. The number of solutions and research developments around this type of technology far outweigh the biometric alternatives, taking full advantage of a unique identification mechanism for a significantly reduced cost.

This Master's Project aims to enhance the reliability of the software and hardware from the *i*SEC solution so far developed, further reducing its cost. Additionally, the project also aims to develop a user interface and database, validating the full solution on same poles of the IPC.

Keywords: *Arduino*; *LCD*; *RFID* Reader; Electric Latch; Database; Authentication; Access Control.

Índice

Índice de Figuras

Índice de Tabelas

Nomenclatura

Abreviaturas

μC	"Microcontrolador"
APT	"Advanced Packaging Tool"
Arduino UNO	"Microcontrolador Arduino Uno"
DLL	"Dynamic-Link Library"
GNU	"GNU is Not Unix"
GNU/Linux	"Sistema Operativo com núcleo Linux"
GPL	"Generic Public Licence"
HTTPS	"HyperText Transfer Protocol Secure"
IIS	"Internet Information Services"
KHz	"Kilo Hertz"
LCD	"Liquid Crystal Display"
Middleware	"Mediador entre aplicações"
MySQL	"My SQL"
PHP	"PHP: Hypertext Preprocessor"
PIC	"Programmable Interface Controller"
RFID	"*Radio Frequency Identification*"
RS-232	"Protocolo de comunicação série"
RTC	"Real Time Clock"
SD	"Secure Digital"
SGBD	"Sistema de Gestão de Bases de Dados"
SQL	"Structured Query Language"
suPHP	"Substitute User on PHP Script"
TCP/IP	"Transmission Control Protocol / Internet Protocol"
UHF	"Ultra High Frequency"
Web	"World Wide Web"
Wireless	"Comunicação sem fio"

1 Introdução

A importância dos sistemas de controlo e gestão é inquestionável, tanto no domínio público como no privado. Entre as várias vantagens provenientes da implementação destes sistemas, destacam-se: *i*) a redução do risco de intrusão; *ii*) o planeamento para ações futuras; *iii*) a monitorização da dimensão da instituição; *iv*) a prevenção de falhas de gestão e coordenação; *v*) a supervisão de funcionários; e *vi*) a capacidade de descentralização da instituição (Otley *et al.*, 1995).

De acordo com especialistas na área da educação, a gestão das escolas e das salas de aula visa incentivar e estabelecer um maior autocontrolo estudantil, através de um processo de promoção do desempenho e do comportamento positivista nos alunos. Deste modo, a concretização académica, a eficácia do professor e o comportamento dentro das salas de aula, estão diretamente relacionados com o conceito de gestão institucional (Froyen & Iverson, 1999).

Foi com estes critérios em mente que, em 2011, o Instituto Superior de Engenharia de Coimbra (ISEC) do Instituto Politécnico de Coimbra (IPC), iniciou um projeto, denominado de *Intelligent and Sustainable Educational Campus* (*i*SEC), com o intuito em desenvolver um sistema de controlo e gestão de acessos no meio académico (Costa *et al.*, 2012). Desde então, o projeto tem vindo a ser alvo de novas soluções e atualizações constantes, ao nível do *software* e *hardware*, no qual este trabalho de Mestrado se identifica como sendo a versão mais atual, contando com a implementação prática, não só no ISEC, como também na Escola Superior de Educação de Coimbra (ESEC).

1.1 Objetivos

O projeto *i*SEC tem como principal objetivo desenvolver e implementar um sistema de controlo e gestão de acessos. Esta necessidade visa colmatar as falhas inerentes à solução alternativa atual, não só na perspetiva de tolerância a falhas, mas também de facilidade de manutenção, reprogramação/reconfiguração, mantendo a perspetiva de baixo custo. Para o efeito, a solução beneficia da tecnologia *RFID* integrada nos cartões bancários fornecidos pela Caixa Geral de Depósitos (*CGD*) e atualmente utilizados pela comunidade do *IPC*.

Este Projeto de Mestrado consiste numa segunda fase no desenvolvimento do *i*SEC, melhorando não só a solução previamente proposta do ponto de vista de fiabilidade do *software* e do *hardware*, como também reduzindo ainda mais o custo da solução. Adicionalmente, o projeto visa ainda o desenvolvimento de um interface com o utilizador e respetiva base de dados.

A validação deste Projeto de Mestrado passou pela implementação de algumas soluções propostas nos *campus* ISEC e ESEC.

1.2 Organização da dissertação

Esta dissertação é estruturada da seguinte forma:

O segundo capítulo consiste em motivar o leitor, fazendo referência aos trabalhos previamente desenvolvidos no contexto dos sistemas de controlo e gestão inteligentes e a necessidade dos mesmos no meio educativo e organizacional. De forma a validar o estudo de mercado, este capítulo compara ainda a solução proposta com as soluções comerciais já existentes no mercado nacional.

O terceiro capítulo explica o desenvolvimento do servidor, focando-se no desenvolvimento do interface gráfico com o utilizador e na base de dados que agrega toda a informação necessária para o registo e controlo de acessos dos utilizadores.

O quarto capítulo apresenta a solução desenvolvida do lado do cliente, considerando as diversas modalidades em que essa é aplicada. Para o efeito, este capítulo discute a solução na perspetiva do *software* desenvolvido e *hardware* utilizado.

O sexto capítulo apresenta alguns resultados experimentais obtidos, *in loco*, de forma a validar o sistema proposto como uma solução comercial passível de ser implementada em contexto educativo e industrial.

O quinto capítulo descreve a implementação prática do sistema implementado no ISEC e na ESEC, descrevendo a planificação e organização espacial dos diversos tipos de equipamentos nas plantas das instituições. Para além disso, discute o nível de implementação já atingido até à data da entrega deste relatório.

O sétimo e oitavo capítulos apresentam as conclusões e perspetivas de futuro desenvolvimento do projeto, respetivamente.

2 Estado da Arte

A investigação atual demonstra que uma alta incidência de problemas disciplinares na sala de aula tem um impacto significativamente negativo sobre a eficácia do ensino e da aprendizagem. Neste contexto, verificou-se que os professores que enfrentam tais problemas não conseguem planear as tarefas escolares de forma adequada, prejudicando, inevitavelmente, o processo de aprendizagem do aluno. Para além disso, a monitorização desse mesmo processo de aprendizagem acaba por ser negligenciada na maioria das situações.

De forma a combater este tipo de situação, as instituições de ensino têm vindo a aplicar estratégias de controlo e gestão cada vez mais eficientes, reduzindo a necessidade da típica, e falível, "folha de presenças", para sistemas mais autónomos e tolerantes a falhas.

Este capítulo tem como objetivo mostrar o trabalho desenvolvido anteriormente, comparando com as vantagens do uso deste sistema

2.1 Sistemas de Controlo e Gestão Inteligente

O artigo apresentado pelos autores em (Farooq et al, 2010) apresenta o desenvolvimento de um sistema de controlo de assiduidade por *wireless*. Neste projeto os autores desenvolveram um sistema que tem, uma estação/servidor central para receber e processar toda a informação, um módulo para controlo de acesso a lugares restritos em cada porta ou laboratório, e depois o mais interessante é o desenvolvimento de um módulo para cada utilizador, no caso de se ter uma empresa com milhares de colaboradores torna-se impraticável. Não sendo um sistema com sensores biométricos existe a possibilidade de troca de módulos entre utilizadores, por isso a solução ideal não será o desenvolvimento de um módulo por utilizador, mas sim como este projeto utiliza, um cartão *RFID*, por utilizador, que no caso de perda como é referido pelos autores, o valor da perda é muito menor, e também poderá ter na mesma um sistema de notificação por *SMS* para informar o sistema que algum utilizador tentou aceder com o cartão perdido. Além do mais o uso da tecnologia *wireless* em todos os módulos obriga a utilização de baterias que terão de ser substituídas ou carregadas ao final de algum tempo.

Os autores em (Khan, S. R. 2012) apresentam o desenvolvimento de um controlo de acessos de baixo custo para um escritório. Este projeto foi desenvolvido utilizando um *μC PIC* da *MicroChip*[1], *PIC16F876A*, um display *Liquid Crystal Display (LCD)* de 20x4 caracteres, e um teclado de 4x4 caracteres. A versão apresentada pelos autores, apesar do reduzido custo, é demasiado simplista e limitada. Por exemplo, a escolha face ao uso opcional de um cartão *Secure Digital* (*SD)* como uma possível solução *offline* não é viável. Na realidade, de forma a garantir tolerância a falhas no sistema, é necessário o uso obrigatório de um cartão *SD* ou equivalente. A alternativa apresentada pelos autores, de utilizar o *μC* para o efeito não é viável, dado que este não possui memória suficiente para ler, guardar e validar uma elevada quantidade de registos, pois isso significaria a introdução da base de dados integral no código do programa. Para além disso, isto obrigaria à reprogramação do *μC* sempre que fosse necessário adicionar ou alterar utilizadores. Deste modo, a solução sem capacidade de armazenamento apresentada em (Khan, S. R. 2012) é inviável pois impossibilita os registos de entradas e/ou saídas. Dito de outro modo, o sistema informa que houve uma tentativa de entrada, mas não consegue informar quais os códigos que foram inseridos, nem a que horas.

Silva *et al.* (2008) apresenta o desenvolvimento de um controlo automático de assiduidade em sala de aula utilizando *RFID*. Este projeto foi desenvolvido utilizando um leitor de proximidade *RFID* da empresa *ZKSoftware*[2], que efetua a leitura de cartões *RFID* a uma frequência de 125Khz. Os autores optaram por uma solução comercial com capacidade para realizar a leitura do cartão *RFID* e enviar a devida informação por *TCP/IP*. No entanto, este tipo de solução não é a indicada pois, tratando-se de uma solução comercial, a empresa disponibiliza apenas uma *Dynamic-link Library (DLL)*. Deste modo, a solução proposta por (Silva *et al.* 2008) torna-se exclusiva para ambiente *Microsoft Windows*, obrigando a instituição a possuir uma licença *Windows* para o servidor e para o desenvolvimento da aplicação. Em termos de estrutura, a solução proposta pelos autores também não é a mais robusta, dependendo de três servidores dedicados: *i)* Servidor de *RFID*; *ii)* Servidor de Base de Dados de *RFID*; e *iii*) Servidor *Web*. Adicionalmente, a solução depende ainda de um *Middleware* (mediador entre aplicações) para passar os dados do Servidor *RFID* para o Servidor Web. No entanto, tudo isto poderia ser simplificado com um Servidor *GNU/Linux* devidamente configurado, tal como este trabalho apresenta.

[1] http://www.microchip.com
[2] http://www.zksoftware.com.pt

O artigo apresentado pelos autores em (Patel *et al.*, 2012) propõe o desenvolvimento de um sistema *online* de monitorização da assiduidade de estudantes em sala de aula usando tecnologia *RFID*. Este projeto tem como objetivo efetuar a leitura automática de todos os cartões que estejam dentro de determinada sala, sem que os alunos tenham de aproximar o cartão de um leitor de cartões. Apesar da inovação inerente a esta solução, a mesma poderá não funcionar como previsto na atualidade. Note-se que, hoje em dia, quase todas as instituições bancárias disponibilizam cartões com tecnologia *RFID*, sendo que o mesmo indivíduo pode possuir múltiplos emissores *RFID*. Assim sendo, quando se efetua uma leitura na sala de aula com múltiplos indivíduos, o leitor pode não ter a capacidade de ler os diversos cartões em simultâneo, mas sim apenas um deles, ou até mesmo nenhum. Infelizmente, os autores não apresentam qualquer tipo de resultados experimentais com este tipo de constrangimento de forma a validar a eficácia da estratégia adoptada. Adicionalmente, como citado pelo autor em (Khan, S. R. 2012), ao utilizarmos uma estratégia de *broadcast Ultra High Frequency (UHF)* deste tipo, passamos de um sistema de controlo de assiduidade de alunos, para um sistema de monitorização constante dos mesmos, podendo originar invasão de privacidade.

Similarmente, o artigo em (Arulogun *et al.*, 2013) apresenta o desenvolvimento de um sistema de controlo de assiduidade por *RFID*. O projeto usufrui de um leitor *RFID* da *Intersoft RFID DemoKit-1*[3], composto por um leitor *RFID*, um interface *RS-232 (Protocolo de comunicação série)*, uma placa de leitura *TR-R01-OEM Board* e antena. Esta solução apresenta essencialmente duas lacunas. A primeira é o facto de necessitar de uma constante ligação a um computador via *RS-232*. Tal torna-se inadequado para ambientes fora de contexto laboratorial ou em pontos específicos já equipados de computadores, dado ser impensável a alocação específica de um computador para cada controlo de acesso. Para solucionar este problema, bastaria recorrer ao uso de μC (*e.g.*, *Arduino*[4]). A segunda acaba por estar relacionada com o constrangimento imposto pela primeira lacuna. Dado ser necessária uma constante ligação a um computador, poderia fazer-se uso da alimentação fornecida pelo mesmo (*e.g.*, *USB*) em vez de se optar por uma alimentação independente do módulo.

Contrariamente aos trabalhos anteriores, os autores em (Kishore *et al.*, 2013) apresentam o desenvolvimento de um sistema de controlo de assiduidade por biometria. Este projeto foi desenvolvido utilizando um *Arduino UNO* que realiza o interface entre todos os outros

[3] http://www.intersoft-us.com
[4] http://arduino.cc

componentes, um sensor biométrico da *Adafruit*[5]. Além dos componentes inerentes a um projeto deste tipo, é usado um teclado de 4x4 caracteres para passar informações para o servidor, um *GSM/GPRS Shield* [6] para enviar um aviso para os encarregados de educação que o aluno não está presente, e um processador *ARM Raspberry PI* com a dimensão de um cartão de crédito. Apesar do elevado nível de funcionalidades da solução proposta, os autores abordam este projeto de uma forma que poderá gerar alguns problemas no funcionamento do sistema. Por exemplo, com este sistema torna-se necessário que o docente utilize o teclado para identificar manualmente o dia, podendo desta forma originar diversas falhas. O sistema proposto nesta tese de Mestrado permite que essa informação seja atualizada automaticamente, mesmo que o sistema se encontre *offline*. Adicionalmente, e tratando-se de uma solução biométrica, é necessário manter uma base de dados de impressões digitais atualizada. Por outro lado, a velocidade de processamento do *Raspberry PI* poderá não ser a melhor para este tipo de operação e, quanto maior for a base de dados, maior será o tempo de resposta. Relativamente ao leitor biométrico, e como se pode verificar na documentação e vídeos facultados pela *Adafruit*[7], o leitor nem sempre consegue efetuar a leitura da impressão digital dado que o sistema se baseia em processamento de imagem. Finalmente, o sistema de alerta proposto pelos autores também não é o mais eficiente. O envio das notificações de faltas por *SMS*, apesar de inovador, pode facilmente provocar uma sobrecarga elevada no sistema. Supondo que um aluno não compareça durante todo o dia, o seu encarregado de educação iria receber uma *SMS* por cada falta cometida ao longo desse dia. Para evitar este tipo de sobrecarga do sistema, bastaria enviar um único *SMS* no final do dia contendo o número de faltas do aluno em causa.

2.2 Comparação

Tendo sido efetuada uma análise de mercado, foram consultados quatro fornecedores nacionais, que apresentam uma solução em sistemas de assiduidade e controlo de acessos. Apenas um dos fornecedores apresenta uma solução também para controlo de parques.

[5] http://www.adafruit.com
[6] http://arduino.cc/en/Main/ArduinoGSMShield
[7] http://www.youtube.com/watch?v=1diFaa5OsFg

As empresas *Smarstep*[8], *LogicPulse*[9] e *Idonic*[10] apresentam uma solução para cada um dos sistemas a controlar, ou seja, se uma empresa pretende implementar um sistema de assiduidade, adquire o *software* e os equipamentos de assiduidade. O mesmo acontece para os sistemas de controlo de acessos e de parque. No entanto, para uma solução de controlo de parque, a *Idonic* é a única empresa que consegue fornecer os três tipos diferentes de sistemas. Para além de ser dispendioso, pois cada licença de *software* tem um custo entre os 300€ e os 900€, não existe qualquer tipo de interligação entre os vários sistemas, pois são soluções isoladas e incompatíveis.

A empresa com *software* e equipamento mais versátil é a *Cifial*[11]. Esta desenvolveu um *software* único, e todos os seus equipamentos podem ser configurados como sendo um sistema de assiduidade ou um sistema de controlo de acessos. Este tipo de arquitetura flexível é comparável ao apresentado neste projeto, como será apresentado ao longo deste documento. No entanto, o custo do equipamento apresentado pela empresa é aproximadamente duas vezes superior ao custo da nossa solução.

Em termos de número de equipamentos de controlo de acessos que o sistema consegue suportar, a *Smartstep* apresenta um controlador com um custo de 1400€ que suporta até 16 portas, tendo um custo por equipamento de aproximadamente 250€. A *LogicPulse* apresenta um controlador com um custo de 670€ que suporta até 4 portas, sendo que cada equipamento atinge um custo de 90€. As empresas *Cifial* e *Idonic* não apresentam limitações quanto ao número de portas que suporta, sendo que a primeira apresenta um custo por equipamento de 298€, e a segunda apresenta um custo por equipamento de 375€.

Relativamente ao número de clientes, a *Cifial* e a *Idonic,* não apresentam qualquer limite face às soluções apresentadas. O equipamento de assiduidade *Smartstep* tem um limite máximo de 100 utilizadores, sendo que o número de utilizadores máximo permitido pelo equipamento de controlo varia de acordo com a licença adquirida, *i.e.*, quanto maior for o número de utilizadores, mais elevado será o valor da licença. Já a *LogicPulse* é o oposto, sendo que o número de utilizadores por equipamento de assiduidade varia de acordo com a licença adquirida, e o equipamento de controlo de acessos tem um limite máximo de 500 utilizadores.

Comparando este projeto com as soluções comercias existentes no mercado, consideramos que a solução proposta é uma mais-valia, aos diversos níveis, seja pelo

[8] http://www.smartstep.pt
[9] http://www.logicpulse.pt
[10] http://www.idonic.pt
[11] http://www.cifial.pt

software, seja pela gestão do sistema ser centralizada do lado do servidor, *i.e.*, o operador/administrador apenas necessita de ter um navegador, que poderá ser o *Chrome*, *Firefox*, *Safari*, entre outros. Dito de outro modo, não existe limitação em termos de sistema operativo, sendo compatível com qualquer plataforma. No mesmo *software*, conseguimos gerir o controlo de acessos, a assiduidade, o controlo de parques (acesso ao campus).

Em termos de *hardware*, o mesmo equipamento utilizado para a assiduidade é utilizado para o controlo de acessos, apenas sendo necessário adicionar um relé para a abertura de uma porta. Relativamente ao custo, este projeto apresenta duas versões distintas para o controlo de acessos e assiduidade, uma versão sem Display LCD (*i*SEC Basic) e outra com Display LCD (*i*SEC Display), tendo um valor compreendido entre os 138€ e os 155€, respetivamente. Para o sistema de controlo de parque apresenta um valor de 222€ (consultar a seção 4.5).

Equipamento	***Software***	***Formação***	***Controladora***	***Preço***	***N.º Equip.***	***N.º Clientes***	***Resposta***
Cifial	500 €	500 €	-	298.00 €	9999	9999	±1000ms
SmartStep Assiduidade	899 €	-	-		1	100	±1000ms
SmartStep Controlo	300 €	1,590 €	1,400 €	250.00 €	16	Licença	±1000ms
LogicPulse Assiduidade	500 €	250 €	-	300.00 €	1	Licença	±1000ms
LogicPulse Controlo	500 €	100 €	670 €	90.00 €	4	500	±1000ms
Idonic Assiduidade	545 €	145 €	-	55.00 €	1		±1000ms
Idonic Controlo	665 €	345 €	-	375.00 €	9999		±1000ms
iSEC Basic	-	-	-	138.00€	2^32	2^32	±600ms
iSEC Display	-	-	-	155.00€	2^32	2^32	±600ms
iSEC Campus	-	-	-	222.00 €	2^32	2^32	±600ms

2.3 Sumário

Em suma, pode-se concluir que este projeto tem capacidade para competir com soluções comercias já existentes no mercado, tanto ao nível de funcionalidades e potencialidades como ao nível do custo.

Os valores de custo apresentados correspondem ao desenvolvimento dos protótipos. No entanto, mesmo com uma margem de 30% a 40% de lucro, consegue-se uma solução bastante competitiva.

3 Servidor

O servidor é um dos principais elementos do projeto, pois é nesse que toda a informação necessária é registada e arquivada para o funcionamento do mesmo. É no servidor que a página *Web* e a base de dados são alojadas.

No servidor será armazenada toda a informação necessária para o funcionamento do projeto, desde a validação da ligação do *Arduino*, aos registos diários, semanais e mensais a todos os relatórios gerados.

3.1 Ubuntu Server

"*Ubuntu*" é uma antiga palavra Africana, cujo significado é "*Humanidade para todos*" e "*Eu sou o que sou devido ao que todos nós somos*[12]" . A distribuição *Ubuntu Linux* traz o espírito do *Ubuntu* ao mundo do *software*. O *Ubuntu* é um sistema operativo completo, baseado em Linux, livremente disponível, com suporte proveniente da comunidade.

O *Ubuntu Server* é uma distribuição de *GNU/Linux* para ser utilizada em servidores para a Web.

A escolha do servidor face à distribuição *Ubuntu*, e não outra alternativa (e.g., *Windows Server*), deve-se essencialmente a três fatores: *i*) porque é um sistema operativo baseado em *Debian* tendo como principal vantagem o uso do *Advanced Packaging Tool (APT)*; *ii*) porque tem uma das maiores comunidades que dão um excelente suporte sempre que necessário; e *iii*) porque é completamente *open source* e gratuito. A escolha de uma sistema operativo *GNU/Linux* vs *Windows Server* encontra-se assente na necessidade em manter o baixo custo e a flexibilidade do projeto, sendo que se optou por uma solução que não exigisse a aquisição de licenças por parte da empresa/instituição. Por outro lado, pode-se argumentar que 65,4% [13]dos servidores na internet utilizam *GNU/Linux*. Adicionalmente, a versatilidade e eficiência de um servidor *GNU/Linux* para o desenvolvimento do interface de gestão do projeto ser *PHP: Hypertext Preprocessor (PHP)* e *My S-Q-L (MySQL)*, tornou-se numa mais-valia para este projeto.

[12] http://www.ubuntu.com
[13] http://w3techs.com/technologies/overview/web_server/all

No entanto, se a instituição assim o pretender, é possível implementar a solução em funcionamento num *Windows Server*, bastando para o efeito substituir os serviços pelo *Apache HTTP Server for Windows* [14]e pelo *MySQL for Windows*[15].

3.2 Servidor Apache

O *Apache HTTP Server* consiste num servidor de páginas Web da *The Apache Software Fundation*[16] tendo sido desenvolvido em 1995 por Ron McCool no *National Center for Supercomputing Applications*[17] na Universidade de Illinois. O *Apache HTTP Server* tem sido o servidor mais popular na internet desde Abril de 1996. Segundo a *NETCRAFT*,[18] em Agosto de 2013 o *Apache* ainda lidera com 47%, seguido da Microsoft com o *Internet Information Services (IIS)* com 23%.

Existem algumas alternativas relativamente ao uso do *Apache* mas, devido às potencialidades do mesmo, optámos por este serviço. Tal como foi referido no ponto anterior, é um servidor de páginas Web que tanto pode ser instalado num servidor *GNU/Linux* como num servidor *Windows*. O *Apache* permite a instalação de inúmeros módulos adicionais, tais como o *suPHP*. Em servidores dedicados para a Web, este módulo permite que cada página seja executada com permissões de utilizador, fornecendo assim uma maior segurança no servidor.

[14] http://httpd.apache.org/docs/2.2/platform/windows.html
[15] http://www.mysql.com/why-mysql/windows
[16] http://www.apache.org
[17] http://www.ncsa.illinois.edu
[18] http://news.netcraft.com/archives/2013/08/09/august-2013-web-server-survey.html

3.3 Base de Dados

De forma simplista, pode-se definir uma base de dados como um conjunto/coleção de dados organizados. Os sistemas de gestão de bases de dados (SGBD) são especialmente desenhados para gerir, manipular e organizar dados necessários para determinada aplicação com o utilizador. Este projeto usufruiu das características do *MySQL Server 5.1*[19], sendo as que mais se destacam a facilidade de uso, elevada segurança, elevada velocidade, escalabilidade, gratuito, entre outros.

Na Figura 3.1 pode-se observar o diagrama da base de dados completa deste projeto. É uma base de dados com um total de 18 tabelas, todas elas respeitando as 3 normas formais de normalização[20]. Sempre que necessário, são efetuadas as respetivas ligações entre tabelas para garantir a integridade da base de dados.

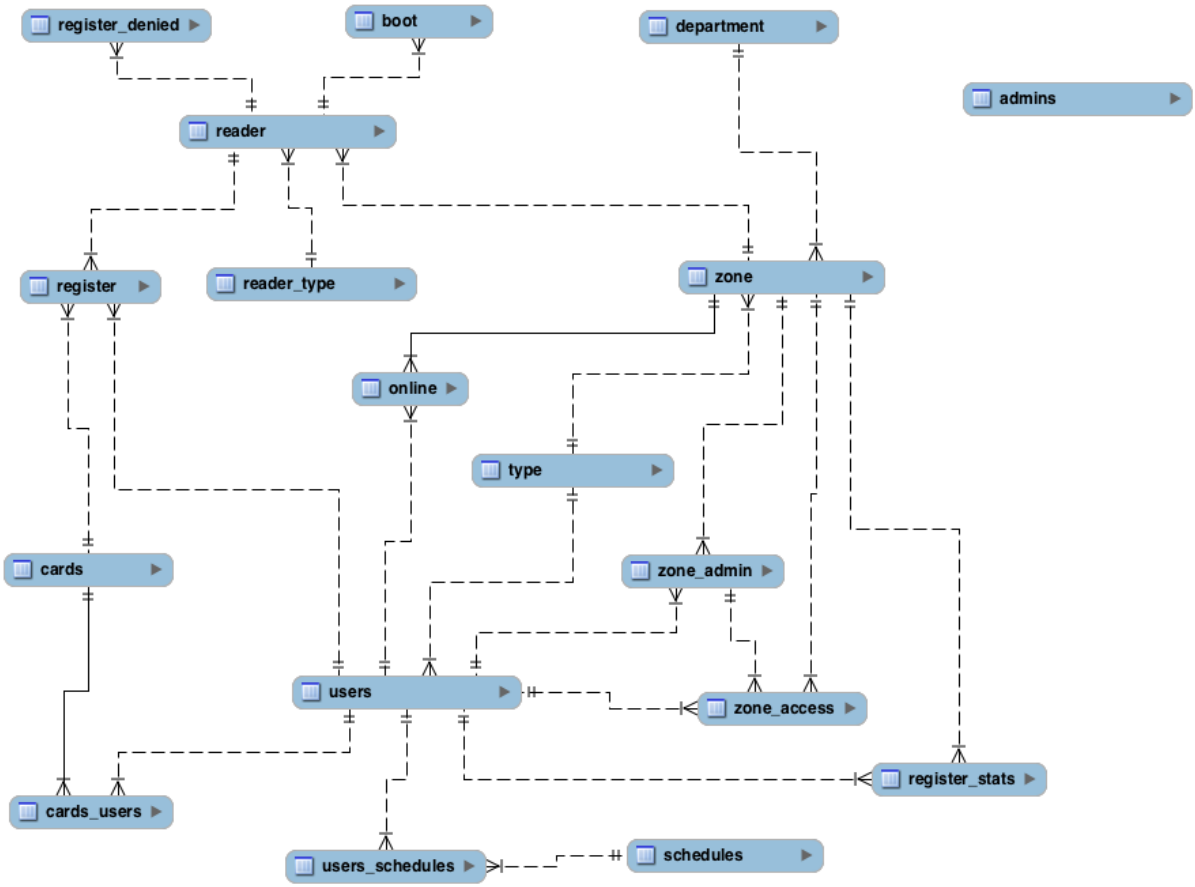


Figura 3.1. Diagrama da BD.

As tabelas encontram-se organizadas de forma a que não exista duplicação de dados.

[19] http://dev.mysql.com/downloads/mysql/5.1.html
[20] http://www.devshed.com/c/a/MySQL/An-Introduction-to-Database-Normalization/3

De uma forma genérica, existem elementos fundamentais na base dados, tais como os *Leitores* (tabela *reader*), os *Cartões* (tabela *cards*) e os *Utilizadores* (tabela *users*).

Deste modo, para se poder adicionar um leitor, é necessário previamente inserir um *Departamento* (tabela *department*), uma *Zona* (tabela *zone*) e saber que tipo de leitor se pretende adicionar (*i.e.*, *Assiduidade*, *Controlo de Acessos* ou *Controlo de Parque*). É de notar que as *Zonas* existem para que se torne possível colocar vários controlos de acesso na mesma sala, *i.e.*, um departamento/sala pode conter duas ou mais entradas/saídas.

Para se poder adicionar um utilizador, basta definir os campos necessários (*e.g.*, *Nome*, *Palavra-passe* e *E-mail*) e dizer que tipo de utilizador é (*e.g.*, *Externo*, *Aluno*, *Docente* ou *Funcionário*). Apesar de ser possível adicionar cartões sem qualquer tipo de constrangimento, estes só são válidos para o sistema quando associados a um utilizador.

Sempre que um utilizador tenta aceder a alguma zona, quer por motivos de assiduidade, quer por controlo de acessos ou controlo de parque, esses dados são enviados por *Hypertext Transfer Protocol Secure* (*HTTPS*) para o servidor. Se o cartão *RFID* for válido (*i.e.*, registado na base de dados e/ou com acesso à zona) o registo/acesso é aceite e a identificação do mesmo é colocada na tabela de registos (*register*). Caso contrário, se o cartão não for aceite, o acesso é negado e é criado um registo na tabela de registos negados (*register_denied*).

3.3.1 Leitores RFID

Como já foi referido anteriormente, de forma a adicionar um leitor é necessário criar primeiramente um *Departamento* e uma *Zona*.

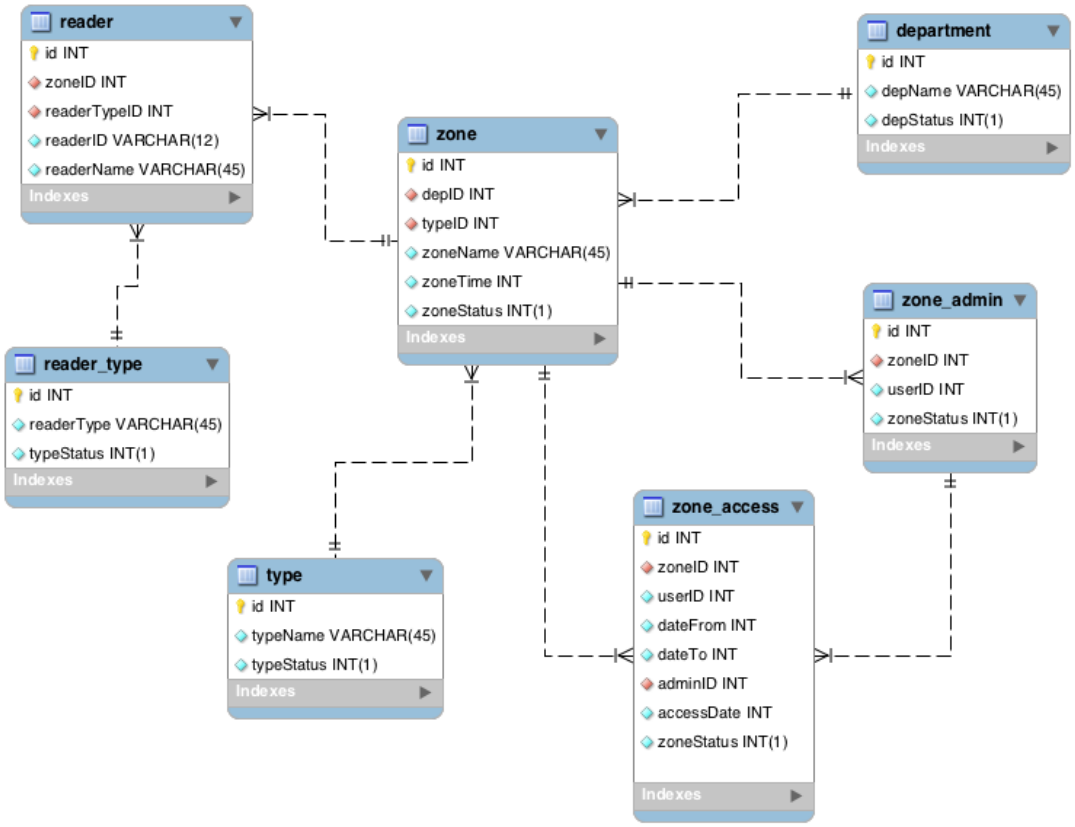


Figura 3.2. Relacionamentos na BD para a tabela de leitores (*reader*).

Para adicionar um *Departamento*, é necessário introduzir o nome do departamento (*depName*), e o estado (*depStatus*) que, por defeito, é colocado a um (1), representando *Departamento Ativo*.

Analogamente, para adicionar uma *Zona*, é necessário escolher o *Departamento* (*depID*) a que se refere, e o tipo de *Zona* (*typeID*), *i.e.*, se é preferencialmente para *Estudantes* (1), *Docentes* (2), *Funcionários* (3) ou *Custom* (4). O campo *Custom* serve para uma utilização específica, como é caso de um centro de investigação, onde se pretende contabilizar o tempo que cada membro despende no centro. E, por último, o estado (*zoneStatus*) que, de forma similar ao *Departamento*, é colocado por defeito a um (1), representando *Zona Ativa*.

Tendo sido criado um *Departamento* e uma *Zona*, é possível adicionar um *Leitor*. Para o efeito, basta selecionar o *Departamento* (*depID*), e a *Zona* (*zoneID*), bem como o tipo de *Leitor* (reader*TypeID*): *Assiduidade*; *Controlo de Acessos* (Entrada / Saída); *Controlo de*

Parque (Entrada / Saída); o nome da *Zona* (*zoneName*). Finalmente, deve de igual modo definir-se o número de série do leitor (*readerID*) e o nome do mesmo (*readerName*).

3.3.2 Cartões RFID

Os cartões podem ser adicionados à tabela sem qualquer tipo de dependência (Figura 3.3). A única limitação reside no facto da mesma não aceitar a adição de dois cartões iguais, *i.e.*, com a mesma identificação.

Neste projeto, é possível adicionar cartões com o estatuto *VIP* que permite aceder a todas as zonas, sobrepondo todas as restrições. No entanto, por motivos de segurança, essa definição é colocada diretamente no cartão e não no utilizador.

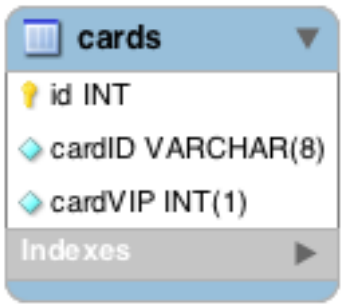


Figura 3.3. Tabela de cartões (*cards*).

3.3.3 Utilizadores

Para adicionarmos um utilizador é necessário introduzir o *Nome*, a *Palavra-passe* e o *E-mail* do mesmo (e.g., *userName*, *userPassword* e *userEmail*) como pode-se ver na Figura 3.4. O *E-mail* é fundamental, pois é através do mesmo que o utilizador é identificado no sistema. De seguida, se a instituição utilizar *Ligthweight Directory Access Protocol* (*LDAP*)[21], o sistema irá preencher automaticamente o nome do utilizador e utilizar a palavra-passe do *LDAP* para entrar no interface do projeto. Também deverá ser definido o tipo de utilizador, *i.e.*, *Externo*, *Aluno*, *Docente* ou *Funcionário*, e o nível de acesso (possibilidade de aceder às permissões de zona).

[21] http://msdn.microsoft.com/en-us/library/windows/desktop/aa367008(v=vs.85).aspx

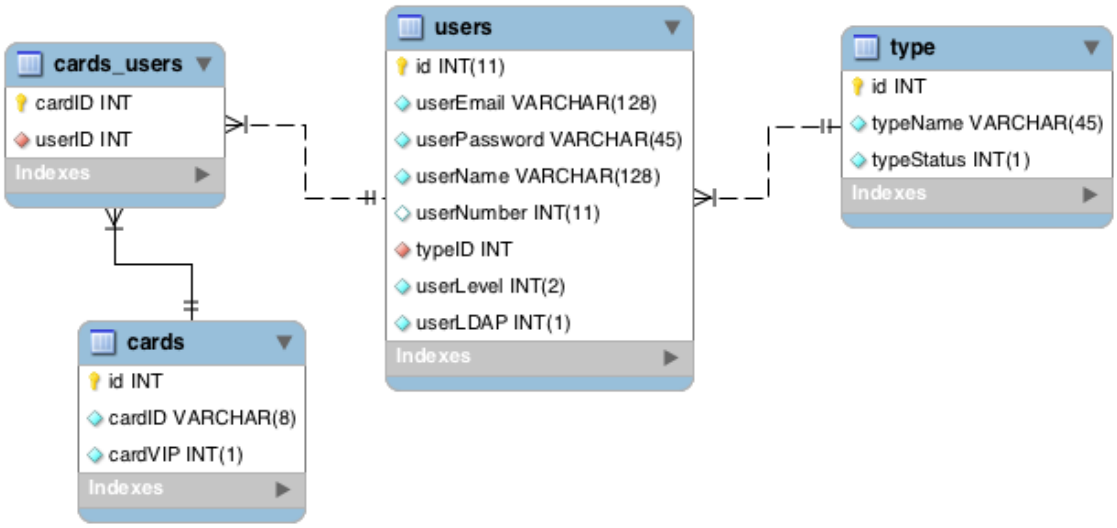


Figura 3.4. Relacionamentos na BD para a tabela utilizadores (*users*).

3.4 Página PHP

PHP é um acrónimo recursivo para "*PHP: Hypertext Preprocessor*". O *PHP* é uma linguagem interpretada desenvolvida no âmbito do *PHP Group*[22] em 1995 por Rasmus Lerdorf. O sucesso e aplicabilidade associados ao mesmo, do ponto de vista de desenvolvimento *Web*, tornaram esta ferramenta, embebida diretamente em *HTML*, na escolha adequada para a representação gráfica do projeto.

A Figura 3.5 demonstra a página de autenticação do interface *Web*. Apenas utilizadores registados podem aceder a esta página. O interface *Web* utiliza as potencialidades das sessões do *PHP*[23], para garantir a segurança dos dados e restringir o acesso a pessoas não autorizadas.

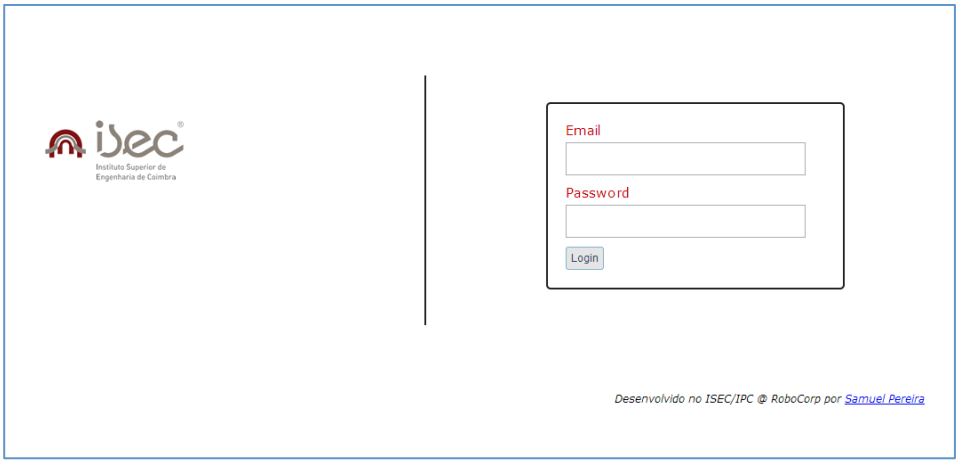


Figura 3.5. Interface Web, página de autenticação.

[22] http://www.phpgroupbd.com
[23] http://www.php.net/manual/en/book.session.php

Todo o interface Web, foi desenvolvido utilizando *PHP* e *HTML*. De seguida pode-se ver o conteúdo de dois ficheiros, primeiramente um ficheiro *PHP* (*index.php*) e em seguida um ficheiro *HTML* (*templates/index.html*). Isto pretende ilustrar algum do desenvolvimento realizado no contexto deste projeto.

```php
<?php

/*
 * ISEC @ Robocorp
 * By Samuel Pereira (spereira@isec.pt)
*/

include ("include/config.inc.php");
include ("include/funcoes.inc.php");
include ("include/session.inc.php");

$i_erro = 0;

if (isSet($_SESSION['ADMIN'])):
        header('Location: admin_index.php');
endif;

if (isSet($_GET['erro'])):
        switch ($_GET['erro']):
                default:
                        $arr_erros[$i_erro] = "O utilizador/password não estão corretos.";
                        $i_erro++;
                        break;
        endswitch;
endif;

include ("templates/index.html");

?>
```

```html
<html>
 <head>
  <title>Controlo de Acessos </title>
  <link rel="stylesheet" type="text/css" href="templates/css/login.css" />
 </head>
 <body>

 <div id="floater"></div>
 <div id="container">
  <div id="areas">
   <div id="area_left"></div>
   <div id="area_right">
    <div id="login">
     <form action="login.php" method="POST">
      <div class="campo">Email</div>
      <div class="campoinput"><input name="email" type="text" /></div>
      <div class="campo">Password </div>
      <div class="campoinput"><input name="password" type="password"  /></div>
      <div class="submit">
       <input type="submit" value="Login" id="bt_login" /><span id="msg"></span>
      </div>
      <div><?php if ($i_erro>0) mostrar_erros ($arr_erros, $i_erro); ?></div>
     </form>
    </div>
   </div>
  </div>
 </div>
 <div id="floater2">Desenvolvido no ISEC/IPC @ RoboCorp por
  <a href="mailto:spereira@isec.pt">Samuel Pereira</a>
 </div>
</body>
</html>
```

De seguida, ir-se-á analisar todos os itens disponíveis do *Menu* do *Interface* que pode ser observado na Figura 3.6.

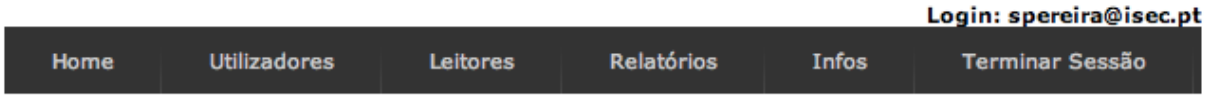


Figura 3.6. Menu do Interface Web.

3.4.1 Utilizadores

No menu Utilizadores é possível adicionar, editar ou listar *Utilizadores*, *Administradores* e *Cartões*.

Ao adicionar ou editar um utilizador (Figura 3.7), a tabela de utilizadores da base de dados será alterada (*cf.*, Figura 3.4). Como já foi referido anteriormente, as instituições que tiverem disponíveis o *LDAP* poderão usufruir do mesmo, utilizando deste modo a mesma palavra-passe de acesso para todo o sistema.

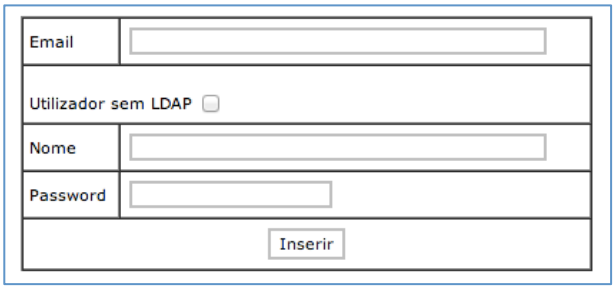


Figura 3.7. Formulário para adicionar um utilizador.

Posteriormente, após a inserção de um novo utilizador, é possível editar os dados introduzidos, como pode-se verificar na Figura 3.8. Neste formulário é possível alterar os dados pretendidos, bem como ativar/desativar o uso do *LDAP*.

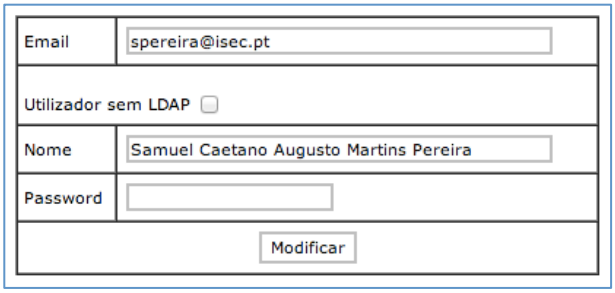


Figura 3.8. Formulário para adicionar editar um utilizador.

É possível ainda visualizar a listagem de *Utilizadores* como pode-se verificar na Figura 3.9. Através desta listagem é possível aceder à edição dos dados de um utilizador como vimos na figura anterior. (*cf.*, Figura 3.8)

Adicionar Novo Utilizador

Selecionar Tipo de Utilizador Funcionário mostrar

N.º	E-mail	Name	LDAP	
1	ines.duarte@isec.pt	Inês Mafalda Correia Duarte	LDAP	edit
2	camorim@isec.pt	Carlos Manuel Amorim Jorge	LDAP	edit
3	teresa.jorge@isec.pt	Teresa da Conceição Costa Jorge	LDAP	edit
4	paula@isec.pt	Paula Cristina Rega Moura Lamas	LDAP	edit
5	ricardo84@isec.pt	Ricardo Manuel Jorge Pereira	LDAP	edit
6	ssimoes@isec.pt	Sandra Isabel Gonçalves do Amaral Simões	LDAP	edit
7	apinheiro@isec.pt	Ana Cristina Almeida Pinheiro	LDAP	edit
8	mgomes@isec.pt	Maria Margarida dos Santos Gomes	LDAP	edit
9	sribeiro@isec.pt	Sandra Cristina Almeida Ribeiro	LDAP	edit
10	claudia@isec.pt	Cláudia Sofia Simões Forte	LDAP	edit
11	maria.torres@isec.pt	Maria Goreti da Silva Duarte Torres	LDAP	edit
12	amafalda@isec.pt	Ana Mafalda Mesquita Carvalho de Sousa Machado	LDAP	edit
13	libaniar@isec.pt	Libânia Maria Batista Rasteiro	LDAP	edit
14	jpalves@isec.pt	João Pedro Antunes Pereira Rodrigues Alves	LDAP	edit
15	cristiano.leal@isec.pt	Cristiano da Silva Leal	LDAP	edit

Figura 3.9. Listagem de utilizadores.

Da mesma maneira, é também possível visualizar a listagem de administradores dos sistema.

Adicionar Novo Administrador

N.º	E-mail	Name	LDAP	
1	spereira@isec.pt	Samuel Caetano Augusto Martins Pereira	LDAP	edit
2	nunomig@isec.pt	Nuno Miguel Fonseca Ferreira	LDAP	edit
3	micael@isec.pt	Micael Couceiro	LDAP	edit
4	a21180221@alunos.isec.pt	Pedro Miguel Dias Vieira Resende	LDAP	edit
5	a21180401@alunos.isec.pt	Elca Tatiana Carvalho Gonçalves	LDAP	edit
6	ana.rosmaninho@isec.pt	Ana Nogueira Rosmaninho	LDAP	edit
7	ines.duarte@isec.pt	Inês Mafalda Correia Duarte	LDAP	edit
8	editemar@isec.pt	Edite Maria Simoes Martins	LDAP	edit
9	carlasus@isec.pt	Carla Susana Fernandes de Oliveira Teixeira	LDAP	edit
10	luis.cantarinhas@isec.pt	Luis Pedro Ramalho Teixeira Cantarinhas	LDAP	edit
11	a21170133@alunos.isec.pt	Fabio Jorge Gomes da Costa	LDAP	edit

Figura 3.10. Listagem de administradores.

Relativamente aos cartões, é possível adicionar um cartão à base de dados de duas maneiras: primeiramente inserindo o número de identificação de cada cartão utilizando o formulário da Figura 3.11, ou submetendo um ficheiro *Comma-separated Values* (*CSV*) como pode-se observar na mesma figura. O ficheiro *CSV* deverá ter apenas uma coluna, contendo o número de identificação de todos os cartões que se pretendem inserir. Este tipo de inserção permite inserir milhares de cartões na base de dados em poucos segundos.

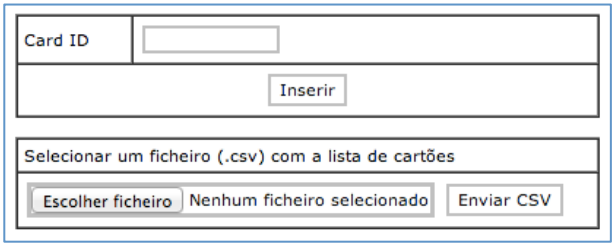


Figura 3.11. Formulário para adicionar um cartão.

A listagem de cartões e respetivo utilizador é efetuada utilizando os relacionamentos existentes entre as diversas tabelas da base de dados. Para este caso específico, como pode-se ver na caixa de texto seguinte, a procura na base de dados é efetuada através dos relacionamentos entre a tabela de *Cartões* (*cards*), a tabela que relaciona os *Cartões* com os *Utilizadores* (*cards_users*) e, obviamente, a tabela de *Utilizadores* (*users*).

```
SELECT c.id, c.cardID, cu.userID, u.userName FROM cards c
LEFT JOIN cards_users cu ON c.id=cu.cardID
LEFT JOIN users u ON u.id=cu.userID
```

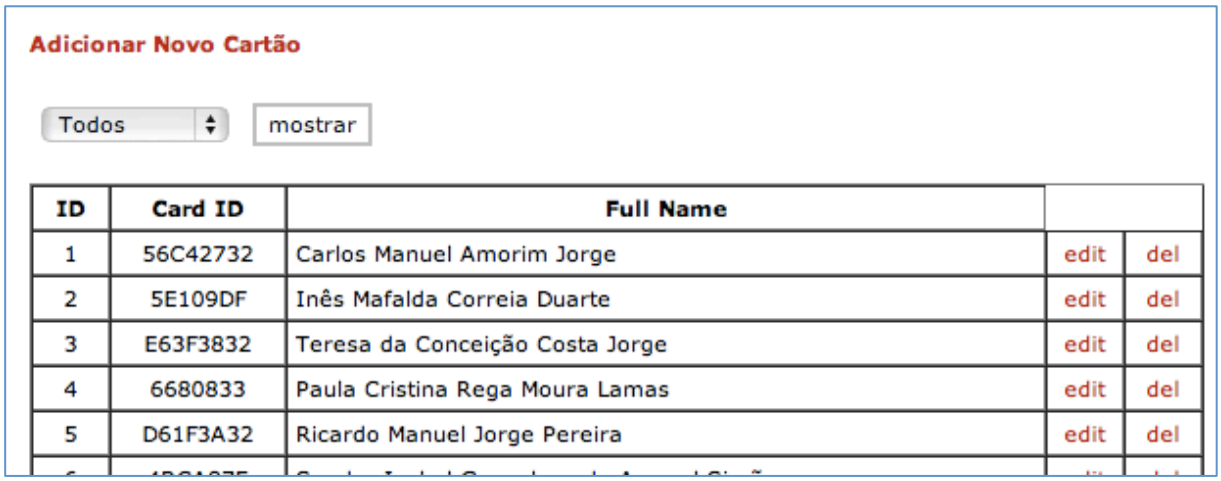


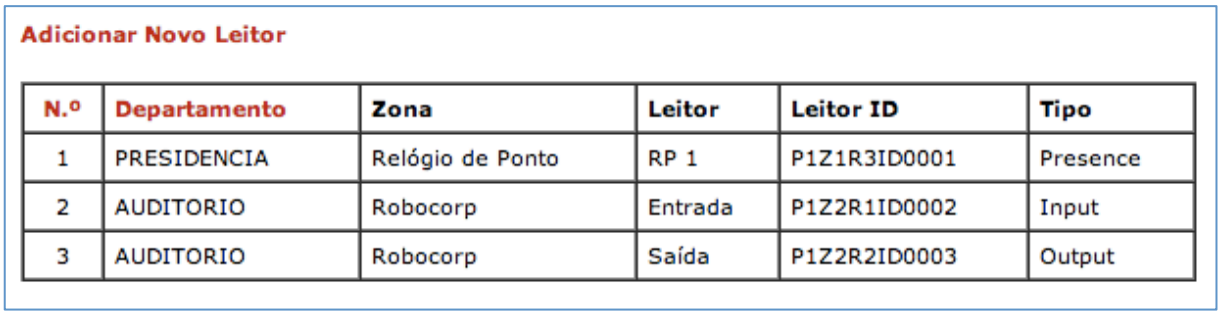
Figura 3.12. Listagem de cartões.

3.4.2 Leitores

Conforme visto no ponto anterior, nesta opção do menu também é possível adicionar, editar ou listar: *Departamentos*, *Zonas* e *Leitores*.

Ir-se-á apenas visualizar a listagem de leitores (*cf.*, Figura 3.13). Na caixa de texto seguinte, pode-se ver o relacionamento existente entre as tabelas dos leitores, a tabela dos tipos de *Leitores*, as *Zonas* e *Departamentos* (*cf.*, *readers*, reader_type, *zone* e *department*)

```
SELECT r.id, depName, zoneName, readerName, readerID, readerType FROM
reader r
INNER JOIN reader_type rt ON rt.id=r.readerTypeID
INNER JOIN zone z ON z.id=r.zoneID
INNER JOIN department d ON d.id=z.depID
ORDER BY $VAR
```



Adicionar Novo Leitor

N.º	Departamento	Zona	Leitor	Leitor ID	Tipo
1	PRESIDENCIA	Relógio de Ponto	RP 1	P1Z1R3ID0001	Presence
2	AUDITORIO	Robocorp	Entrada	P1Z2R1ID0002	Input
3	AUDITORIO	Robocorp	Saída	P1Z2R2ID0003	Output

Figura 3.13. Listagem de leitores.

3.4.3 Relatórios

No menu Relatórios pode-se ver alguns relatórios dos dados inseridos, tais como: *Relatório de Gantt*[24] *(Gráfico de barras popular que ilustra um planeamento de um projeto), Registos Globais, Registos de Horas por Semana, Registos de Horas por Mês e Registos do Arduino*.

O *Relatório de Gantt* utiliza exclusivamente os dados inseridos na tabela de dados registados e processados (*register_stats*). Esta tabela tem duas dependências, uma com a tabela de *Zonas* (*zones*) e outra com a tabela de *Utilizadores* (*users*). Apenas registos válidos poderão ser introduzidos nesta tabela.

Sempre que um utilizador acede ao sistema, quer para dar entrada num controlo de acessos, quer no registo de assiduidade ou mesmo no controlo de parque, será inserida uma linha de dados (ficando esta em aberto) para esse utilizador nesta tabela. Essa linha de dados

[24] http://en.wikipedia.org/wiki/Gantt_chart

será concluída quando o utilizador sair do sistema. Se, por alguma razão, o utilizador não der saída no sistema mas voltar a dar entrada, a linha de dados será concluída e uma nova linha será inserida para registo da nova entrada no sistema.

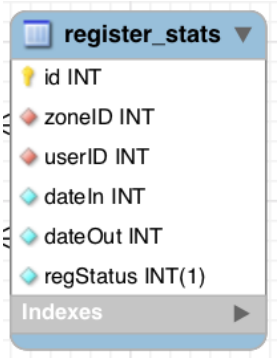


Figura 3.14. Tabela de registos processados (*register_stats*).

Para gerar o Relatório de Gantt foi utilizado uma biblioteca de *PHP* (*PHP Graph Plotting library*[25]).

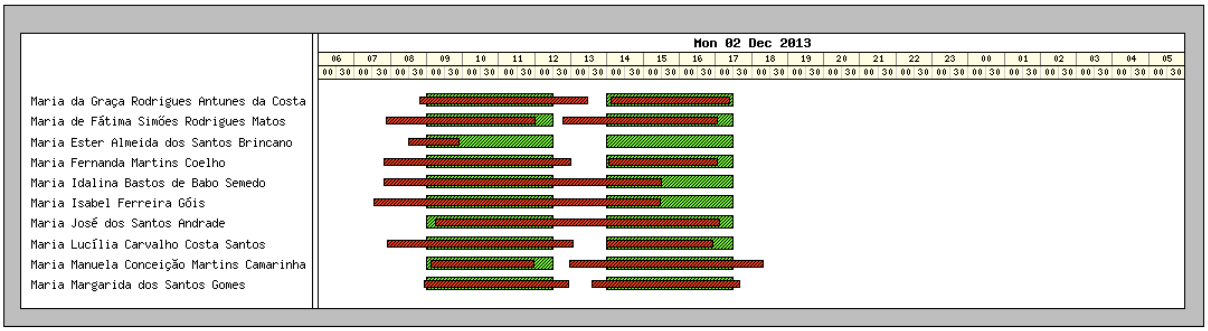
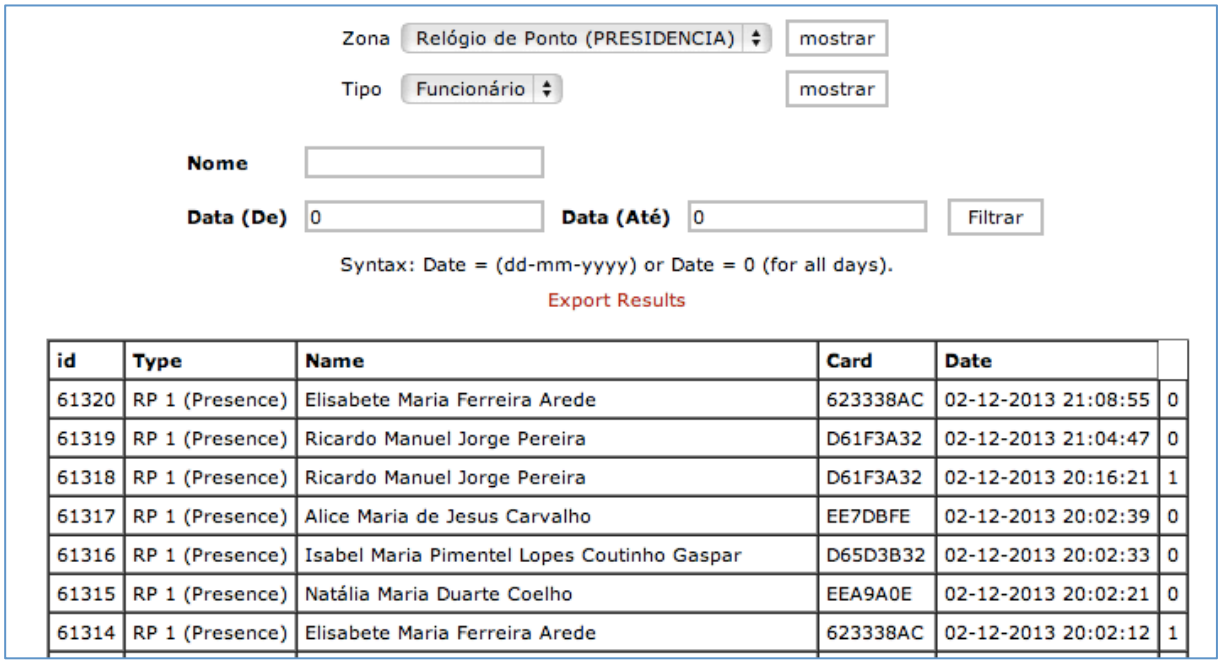


Figura 3.15. Relatório de Gantt.

Pode-se ainda visualizar a listagem de registos globais (Figura 3.16), onde ser poderá fazer diversas filtragens, desde o tipo de utilizador, passando pelo nome, até ao filtro de data, onde se poderá definir a data de início e a data de fim.

[25] http://jpgraph.net



id	Type	Name	Card	Date	
61320	RP 1 (Presence)	Elisabete Maria Ferreira Arede	623338AC	02-12-2013 21:08:55	0
61319	RP 1 (Presence)	Ricardo Manuel Jorge Pereira	D61F3A32	02-12-2013 21:04:47	0
61318	RP 1 (Presence)	Ricardo Manuel Jorge Pereira	D61F3A32	02-12-2013 20:16:21	1
61317	RP 1 (Presence)	Alice Maria de Jesus Carvalho	EE7DBFE	02-12-2013 20:02:39	0
61316	RP 1 (Presence)	Isabel Maria Pimentel Lopes Coutinho Gaspar	D65D3B32	02-12-2013 20:02:33	0
61315	RP 1 (Presence)	Natália Maria Duarte Coelho	EEA9A0E	02-12-2013 20:02:21	0
61314	RP 1 (Presence)	Elisabete Maria Ferreira Arede	623338AC	02-12-2013 20:02:12	1

Figura 3.16. Listagem de Registos Globais.

Os relatórios da contagem do número de horas, quer por semana (*cf.*, Figura 3.17), quer por mês (*cf.*, Figura 3.18), apresentam um somatório das horas registadas no sistema de assiduidade.

	Sem 45	Sem 46	Sem 47	Sem 48	Sem 49
Seg	09h04	--	07h07	08h10	07h45
Ter	03h00	--	09h01	08h08	--
Qua	03h34	--	08h02	08h48	--
Qui	--	02h59	09h08	09h56	--
Sex	07h53	08h13	08h01	08h26	--
Sáb	--	--	--	--	--
Dom	--	--	--	--	--

Figura 3.17. Visualização do número de horas de trabalho por dia.

	Ago	Set	Out	Nov	Dez
Abílio Nuno Falcão Teixeira	38h15	135h39	187h31	169h17	08h08
Adilia Maria Rasteiro Batista	--	118h43	161h05	105h55	--
Adriana Maria Cordeiro de Lima Pinto	86h50	161h04	169h53	149h03	08h18
Alice Maria de Jesus Carvalho	--	71h32	171h47	173h36	03h53
Ana Cristina Almeida Pinheiro	66h56	118h50	191h34	156h59	07h50
Ana Cristina Alves Pereira da Silva Machado	--	78h15	177h04	163h13	08h09
Ana Cristina Pinto da Fonseca Alves Mercier	52h49	96h51	148h38	145h18	07h46
Ana Luísa Rodrigues Henriques Pereira	30h57	104h56	176h20	168h32	06h16
Ana Mafalda Mesquita Carvalho de Sousa Machado	46h35	124h25	142h09	122h13	07h41

Figura 3.18. Visualização do número de horas de trabalho por mês.

Por último, os registos do *Arduino* servem para monitorizar o sistema, de forma a poder, verificar se os clientes estão *online* e, caso não estejam, quando foi a última vez que estiveram *online*.

Registo do Arduino		
Relógio de Ponto (RP 1)	02-12-2013 22:15:28	Online
Relógio de Ponto (RP 1)	02-12-2013 21:45:30	Online
Relógio de Ponto (RP 1)	02-12-2013 21:15:31	Online
Relógio de Ponto (RP 1)	02-12-2013 20:45:32	Online
Relógio de Ponto (RP 1)	02-12-2013 20:15:33	Online

Figura 3.19. Registos de ligação do *Arduino* ao servidor.

3.5 Cópias de Segurança

As cópias de segurança são cópias secundárias de dados. São essenciais e imprescindíveis em todos os sistemas onde os dados registados são uma prioridade. Neste projeto as cópias de segurança são efetuadas com uma periocidade diária. As cópias de segurança estão a ser efetuadas através do c*rontab* (*Cron Table* - é um ficheiro que contêm comandos que são executados de acordo com uma agendamento) do *GNU/Linux*.

3.5.1 Crontab

O *crontab* irá executar um *Shell Script* (linguagem de programação executada através da linha de comandos do *GNU/Linux*). Este *Shell Script*, executa uma série de comandos, comandos estes que exportam todas as base de dados do sistema para um ficheiro, que posteriormente cria uma ficheiro comprimido utilizando o *Tape ARchive (TAR)* e o *GNU Zip* (*GZIP*). (*e.g.*, *backup.tar.gz*).

De seguida, será utilizado o *RSYNC* (ferramenta *open source* que permite a transferência incremental de ficheiros entre servidores).

```
2 0 * * * /home/spereira/backup-mysql.sh > /dev/null 2>&1
4 0 * * * rsync -rv --ignore-existing /home/spereira/backup
  spereira@datahost.pt: /home/spereira/backup > /dev/null 2>&1
```

3.5.2 Shell Script

Como já foi referido anteriormente, este *Shell Script* irá efetuar a exportação de todas as bases de dados do *MySQL* para o ficheiro temporário (*dump.sql*). De seguida, irá comprimir o ficheiro, utilizando o *TAR* e o *GZIP*.

```
#!/bin/sh

bkdir='/home/spereira/backup'
bkfile='mysql-backup-'`date +"%y.%m.%d"`

echo '# Backup Automático MySQL'
echo '# A criar a cópia da bd'
mysqldump -u root -pPASSWORD --all-databases > $bkdir/dump.sql

echo '# A comprimir o ficheiro'
tar zcvf $bkdir/$bkfile.tar.gz $bkdir/dump.sql

echo '# A limpar ficheiro temporario'
rm -v $bkdir/dump.sql
```

3.5.3 RSYNC

O *RSYNC* permite a transferência incremental de ficheiros entre servidores. Sempre que um ficheiro é colocado na pasta das cópias de segurança (*/home/spereira/backup*), este é transferido para outro servidor.

```
rsync -rv --ignore-existing /home/spereira/backup  spereira@
datahost.pt:/home/spereira/backup

sending incremental file list
backup/mysql-backup-13.12.03.zip
backup/mysql-backup-13.12.04.zip

sent 2880382 bytes  received 51 bytes  523715.09 bytes/sec
total size is 47504943  speedup is 16.49
```

3.6 Sumário

Pode-se concluir que o servidor é um dos elementos mais importantes deste projeto, pois todo o processo é baseado no mesmo. Os clientes ligam-se ao servidor, este verifica os dados inseridos e, conforme a resposta enviada pelo servidor, os clientes executam ou não determinada tarefa. A centralização de todo os procedimentos no servidor é uma mais-valia para este projeto.

4 Cliente

Como referido anteriormente, neste projeto foram implementadas correções a nível do *hardware* e *software* face ao trabalho desenvolvido anteriormente.

A nível de *hardware* houveram diversas alterações do ponto de vista tecnológico. A primeira foi a substituição da alimentação 5V de toda a solução cliente por intermédio da placa de controlo *Arduino* por uma fonte externa (*e.g.*, transformador AC-DC 9V). Foram também retirados 3 componentes, nomeadamente, o controlador e carregador da bateria *Lipo Rider Pro*[26], a bateria de lítio de 3,7V / 4,4 A e o *Real Time Clock* (*RTC*). Os primeiros dois componentes foram removidos devido à alteração efetuada na alimentação do *Arduino*, pois o mesmo necessita de ser alimentado com uma tensão entre 7-12V DC, sendo que o *Lipo Rider Pro* permite apenas uma alimentação de 5V, causando instabilidade. O *RTC* foi retirado para reduzir o custo, tendo sido substituído pelo relógio interno do *Arduino*. Com a remoção dos 3 componentes consegue-se uma redução no valor final do cliente de 27%. De forma a comprovar a fiabilidade do relógio interno do *Arduino*, foi efetuado um estudo preliminar através da análise detalhada de 13 dias sucessivos, semanais e fim-de-semana. Verificou-se com o mesmo que esse apresenta um desvio de 109 segundos a cada hora. Por esse motivo, o sistema foi programado para atualizar o relógio interno de 30 em 30 minutos recorrendo ao relógio do servidor, para que assim seja possível garantir a fiabilidade do sistema (ver fluxograma da Figura 4.6 apresentado mais à frente neste relatório).

Devido à remoção do *RTC*, uma das condições necessárias para o arranque do sistema é existência de uma ligação inicial ao servidor, de modo a atualizar o relógio interno do *Arduino*. Se essa ligação não for assegurada no momento do arranque do sistema cliente, esse ficará em modo de espera até a mesma ser garantida.

De forma sucinta, pode-se classificar o cliente como sendo o interface entre o utilizador e o sistema. Neste projeto são utilizados dois tipos de controladores *Arduino*, nomeadamente o *Arduino Mega 2560* (*cf.*, Fig. 4.1) e o *Arduino Ethernet* (*cf.*, Fig. 4.3). Existem também algumas variações adicionais consoante as necessidades da aplicação como será detalhado de seguida.

[26] http://www.seeedstudio.com/depot/lipo-rider-pro-p-992.html?cPath=155

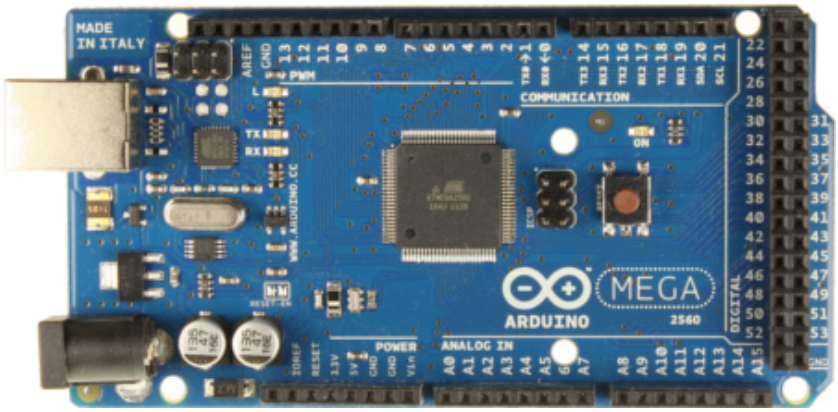


Figura 4.1. Arduino Mega 2560 R3.

Primeiramente, foi desenvolvido um cliente com os seguintes componentes: *Arduino Mega 2560 (Fig. 4.1), Arduino Ethernet Shield, Leitor MIFARE RFID (Fig. 4.2)* e um *Display LCD 20x4 (Fig. 4.5)*. Este leitor serve para realizar o registo de assiduidade e para controlo de acessos. A versatilidade do sistema permite definir se o leitor é um registo de assiduidade ou um controlo de acessos pela configuração do mesmo no servidor. No caso do controlo de acessos de determinada sala, tipicamente com dois ou mais clientes localizados geograficamente próximos (*e.g.*, um no interior da sala e outro no exterior), pode-se simplesmente adicionar um *Leitor MIFARE RFID*, sem necessidade de um cliente adicional.

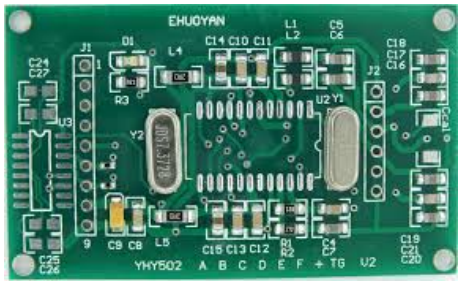


Figura 4.2. Leitor MIFARE RFID YHY502CTG da EHUOYAN.

Posteriormente, foi desenvolvido outro cliente para controlo de acessos, ficando assim uma solução ainda mais económica, tanto a nível de *Arduino* como de todos os restantes componentes. Pode-se observar o esquema elétrico do mesmo na Figura 4.4. Este segundo cliente tem apenas os seguintes componentes: *Arduino Ethernet (Fig. 4.3) e 2 Leitores MIFARE RFID*. Embora não seja aconselhável, também é possível configurar este equipamento como registo de assiduidade. No entanto, dado o mesmo não possuir *Display*

LCD, é necessário criar um código de cores para que os utilizadores (*e.g.*, funcionários) possam verificar se o registo foi efetuado ou não com sucesso.

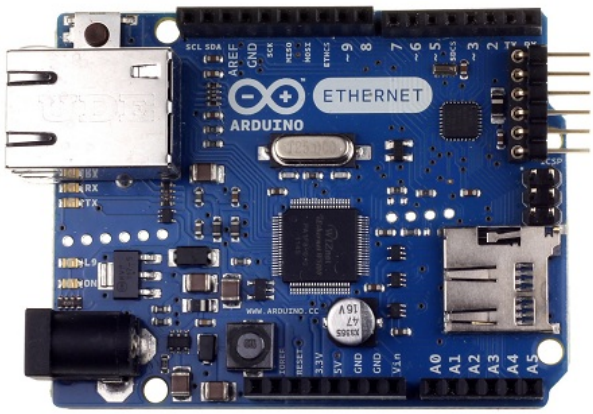


Figura 4.3. Arduino Ethernet.

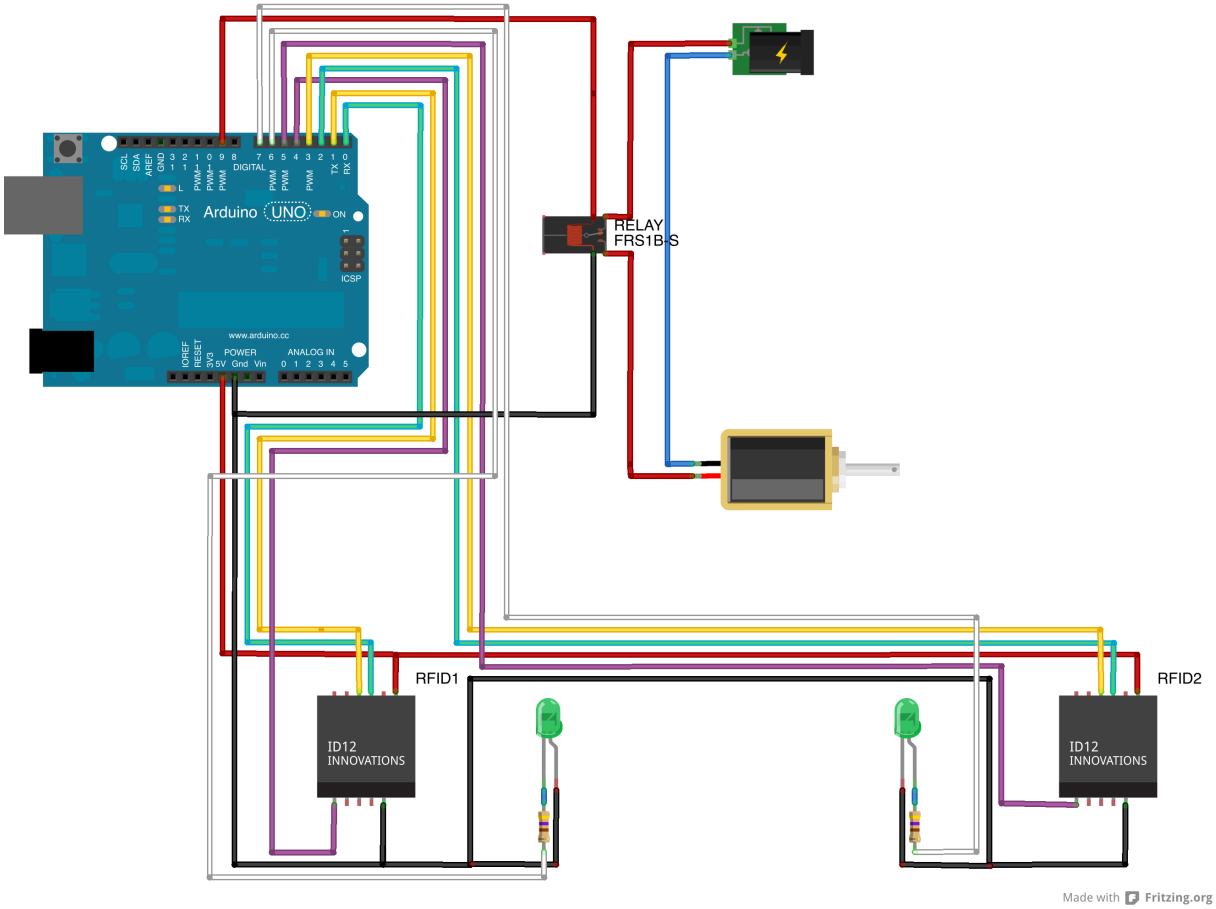


Figura 4.4. Esquema eléctrico da solução cliente que utiliza o *Arduino Ethernet*.

Os cartões utlizados no sistema possuem tecnologia *RFID* de 13,56Mhz, sendo utilizados como identificação de estudante/funcionário da instituição. No caso do IPC, os cartões são fornecidos pela Caixa Geral de Depósitos (*CGD*). No entanto, esta solução é compatível com cartões provenientes de qualquer outra instituição bancária que utilize tecnologia *Visa PayWave* ou *Maestro PayPass*. Para utilizadores que não tenham cartões que possuam esta tecnologia (*e.g.*, visitantes externos), é possível providenciar uma *tag* ou cartão *RFID* básico, com um custo inferior a 1,00 € cada.

Relativamente ao controlo de acesso ao campus, foram desenvolvidas duas placas específicas. A primeira placa providencia o interface entre o *Arduino* e o sistema de controlo das cancelas de acesso ao parque. As placas existentes fazem a deteção automática dos veículos, não sendo deste modo necessário implementar o controlo das cancelas. A segunda placa desenvolvida consiste no interface entre o utilizador e o sistema. Os leitores existentes foram substituídos e apenas a estrutura exterior foi reutilizada. Para o efeito, foi desenvolvida uma placa com as dimensões adequadas a essa estrutura. O próprio *Display LCD* utilizado teve de ser modificado para que assim pudesse utilizar a estrutura existente.

A comunicação entre o cliente e o servidor é efetuada por *HTTPS*, utilizando o método *GET*; um método que recebe toda a informação enviada por um formulário que estiver disponível no endereço. Sempre que um cliente comunica com o servidor, é verificada a *String* enviada pelo *GET*. Assim, se houver algum campo que não seja válido, a *String* não é aceite.

A *String* é composta pelos seguintes dados: *palavra-passe*, *número do leitor*, *número do cartão* e a *data* em formato *time_t* (*unix timestamp*).

A *palavra-passe* está colocada no código como método de autenticação do cliente. Em determinada instituição, todos os clientes podem possuir a mesma *palavra-passe* de forma a evitar a reprogramação dos clientes. Isso deve ser algo realizado durante a planificação da instalação da solução proposta. Outros tipos de medidas poderiam ser implementadas, tais como a validação do endereço de *Internet Protocol* (*IP*) do cliente. No entanto, devido às configurações de rede existentes, nomeadamente do ISEC, e das várias redes disponíveis, o acesso ao endereço *IP* do cliente não é assegurado, sendo apenas possível aceder ao *IP* do *gateway*. No entanto, em outros polos, como é o caso da ESEC, esta configuração pode ser implementada devido à existência de uma *Virtual Private Network* (*VPN*) dedicada para o projeto.

Através do *número do leitor*, é possível saber se o registo procede de um equipamento de assiduidade, de um controlo de acessos ou do controlo de parque. Através dessa informação, os dados são processados de forma distinta como referido no capítulo anterior.

Através do *número do cartão*, e em conjunto com o *número do leitor*, é efetuada a verificação do utilizador de forma a aferir se este tem acesso autorizado a efetuar o registo de assiduidade, aceder ao controlo de acessos ou aceder ao parque.

A *data* também é de extrema relevância e deve ser enviada para cada registo. Através da *data* é possível realizar-se todo o processamento de validação. Por exemplo, no caso da assiduidade é possível efetuar registos em modo *offline* (sem ligação ao servidor). Neste caso, os registos serão processados quando o sistema voltar a ficar *online*, sendo que o servidor receberá todos os dados de forma transparente independentemente do modo de funcionamento (i.e., *online* ou *offline*).

4.1 Registos de assiduidade

O registo de assiduidade poderá ser feito com apenas um cliente onde será efetuada o registo de entrada e o registo de saída de cada funcionário. O funcionário será informado que está a efetuar um determinado registo (*Registo de Entrada; Registo de Saída; Registo Efetuado*). A mensagem com *Registo Efetuado*, aparece sempre que são efetuados dois registos seguidos em menos de 60 segundos. Como confirmação adicional, é mostrado o nome do funcionário no Display LCD.

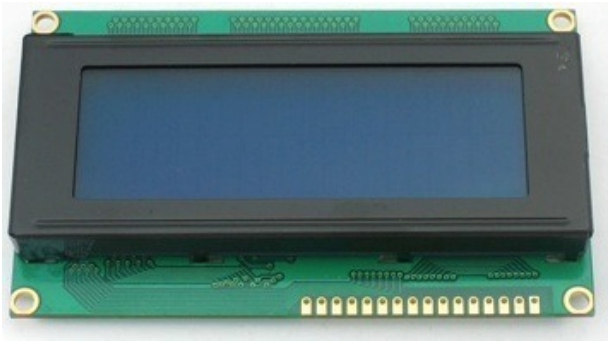


Figura 4.5. Display LCD 20x4.

Se forem utilizados dois clientes, será um para os registos de entradas e outro para os registos de saída.

Como já foi referido anteriormente, o sistema de assiduidade poderá funcionar tanto em *online*, como em *offline*, só existe um requisito para o funcionamento do sistema, que obrigatoriamente deverá iniciar com acesso ao servidor, para que o *Arduino*, consiga sincronizar o relógio interno com a hora atual, após essa verificação, poderá funcionar continuamente em offline, podendo armazenar milhares de registos. Logo que o sistema volte a *online*, todos os registos processados serão enviado para o servidor.

4.2 Acesso a departamentos e salas

O controlo de acessos, também poderá ser efetuado com um ou com dois clientes, em modo *online* ou em modo *offline*. O acesso a departamentos, obrigatoriamente necessidade de um cliente com dois leitores *MIFARE RFID*, para que assim se possa dar permissão e registar todas as entradas e saídas de cada departamento. O acesso a salas de aula, ou a laboratórios, poderá ser feita utilizando um cliente com um ou dois leitores. Se pretende apenas abrir a porta de uma sala de aula, bastará apenas um leitor. Se pretende controlar os acessos, as

entradas e saídas e o tempo que cada utilizador permanece dentro de uma sala ou laboratório, obrigatoriamente serão necessários dois leitores, uma para os registos de entrada e outro para os registos de saída, um colocado no interior e outro no exterior de uma sala.

No caso de se querer controlar apenas os acessos a uma sala, existe também a possibilidade do funcionamento ser em modo *offline*. Para se poder dar este acesso autorizado em modo *offline*, o cliente terá uma listagem de todos os cartões autorizados (que será descarregada automaticamente pelo cliente através de um aviso no servidor), sempre que haja uma falha na comunicação com o servidor, o cartão será verificado nesta listagem.

4.3 Acesso ao campus

O controlo de acesso ao campus exige uma política muito mais restrita. Neste caso, garante-se o funcionamento em modo *offline* apenas para determinados cartões *Very Important Person* (*VIP*), para que no caso de uma falha na rede ou no servidor esses permitam validar uma entrada ou uma saída. A política de utilização poderá ser ajustada de acordo com a instituição mas, se nada for solicitado, estas serão as regras de utilização:

- Os alunos só conseguem entrar no parque se esse não estiver lotado;
- O pessoal docente e não docente consegue entrar mesmo se este, estiver lotado, dado que a contagem é suscetível a erros nesta fase;
- Um veículo só consegue validar a entrada se não estiver dentro do *campus*, de forma a evitar partilha de cartões entre utilizadores;
- Se o sistema estiver a bloquear as entradas de acesso ao campus, o cartão *VIP* consegue entrar no *campus*.

4.4 Fluxogramas

Neste ponto ir-se-á analisar os fluxogramas de controlo de acessos, registo de assiduidade e de acesso ao campus.

4.4.1 Fluxograma da função *setup()*

A função *setup()* inicia com algumas verificações. Primeiramente, verifica a existência do cartão *SD*. Se este estiver presente, verifica se existem registos *offline* para serem processados. De seguida, irá verificar a ligação com o servidor. Se esta existir, atualiza o relógio do cliente e coloca o sistema em modo *online*. Se não existir comunicação com o servidor, o sistema fica bloqueado, como já foi referido anteriormente.

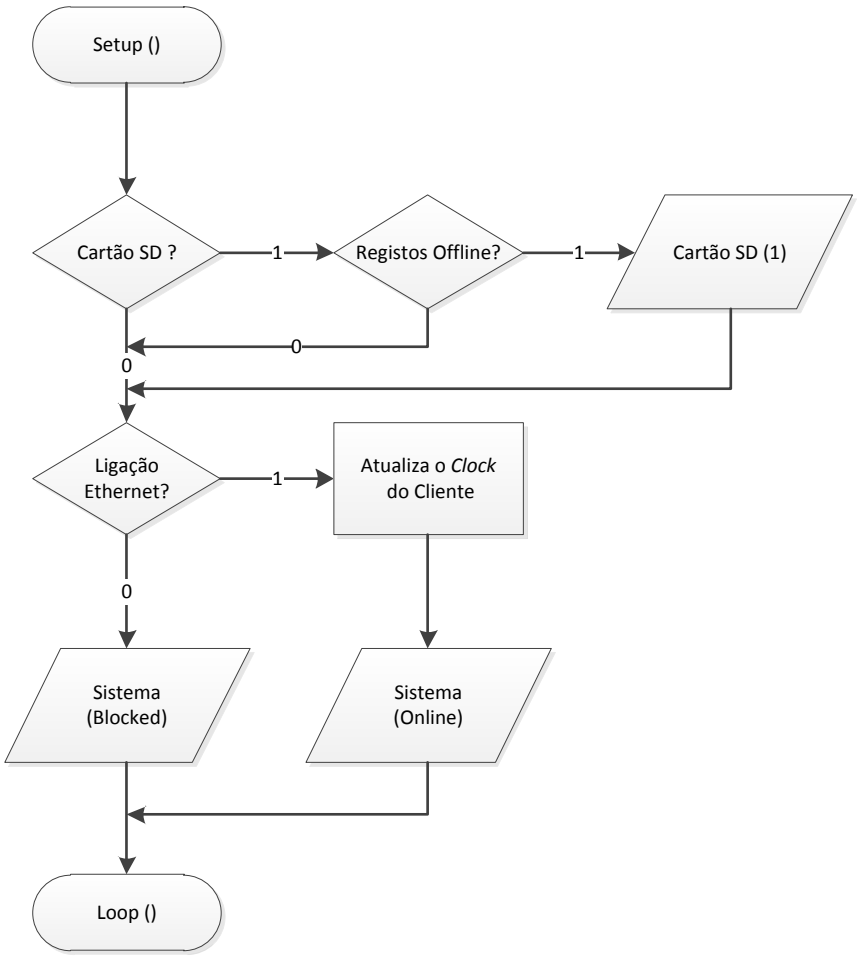


Figura 4.6. Fluxograma da função *setup()* do *Arduino*.

4.4.2 Fluxograma do *loop ()*

A função *loop()* também inicia com algumas verificações. Se o sistema estiver bloqueado, a cada 60 segundos será verificada a ligação com o servidor. Quando a ligação for estabelecida, será atualizado o estado do sistema. Quando o sistema está modo *online*, ou *onlineSD* (preferencialmente apenas para o registo de assiduidade), o *Display LCD* é atualizado de segundo a segundo, para apresentar a data e hora atual. O sistema em modo *onlineSD* permite que os registos sejam efetuados para o cartão *SD*. Neste modo, o sistema verifica a cada 5 minutos a ligação ao servidor. Quando a ligação com o servidor for reposta, os registos armazenados no cartão *SD* serão processados e enviados para o servidor.

Como já foi referido previamente, a cada 30 minutos, o relógio interno do *Arduino* será atualizado.

Após a conclusão de todas as verificações o sistema aguarda a detecção de algum cartão junto do leitor. Sempre que é efetuada uma leitura, a consistência dos dados é verificada através de um *binary term (byte)* de *checksum* (soma de verificação para verificar a integridade dos dados recebidos). Se os dados forem válidos e se o sistema estiver em modo *online*, estes serão enviados para o servidor. Por outro lado, se estiver em modo *onlineSD*, os dados serão armazenados no cartão *SD*.

Após o envio dos dados para o servidor, o sistema recebe uma resposta. Se a mesma for válida, ativará o relé, de forma a ativar um aviso sonoro ou abrir um trinco elétrico.

No fim de todo o processo, o programa só avança se não existir nenhum cartão junto aos leitores *RFID*.

No caso do fluxograma para o acesso ao campus, este é ligeiramente diferente, pois necessita de contemplar a deteção do carro e o estado da respectiva barreira. O restante fluxograma mantém-se.

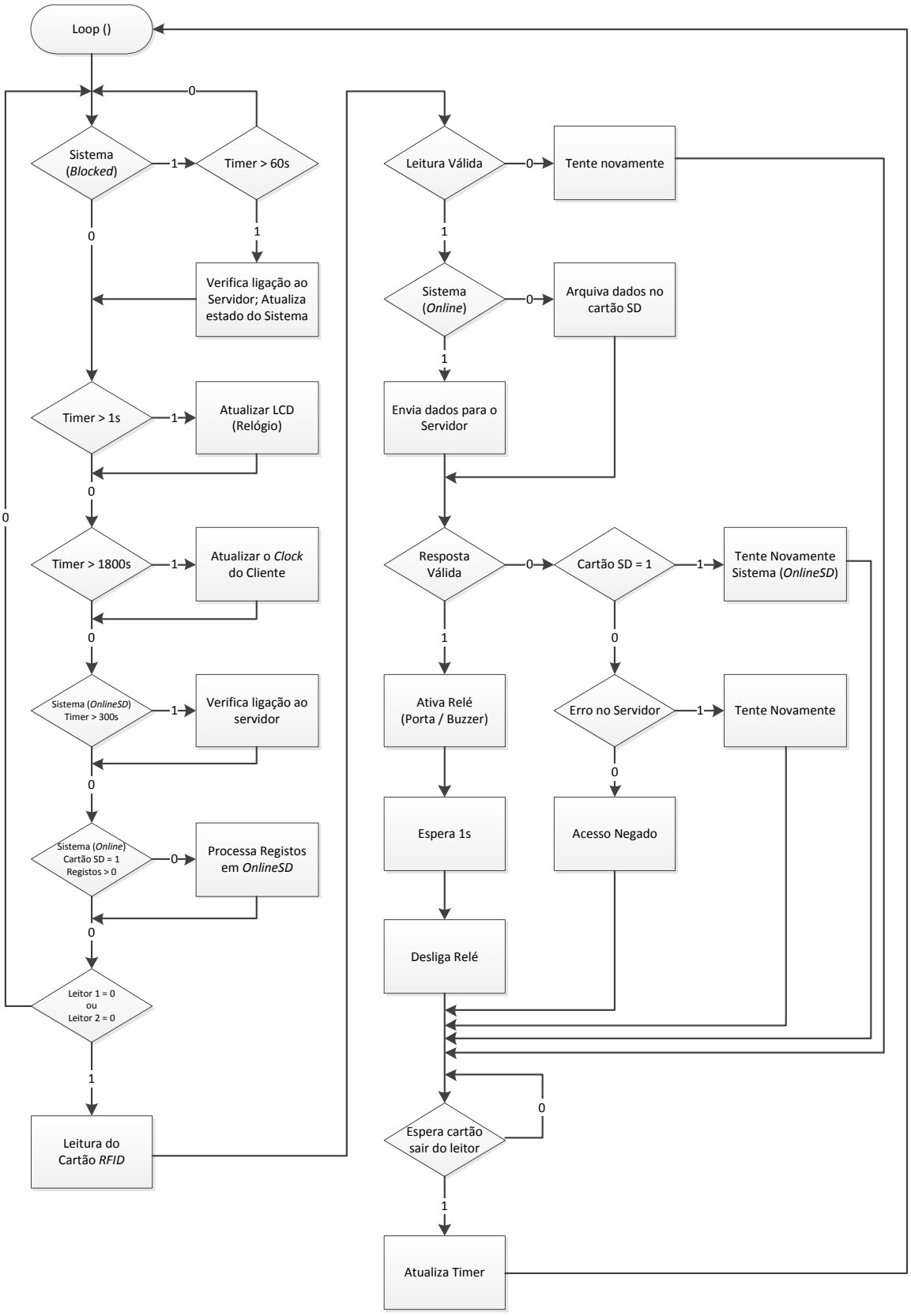


Figura 4.7. Fluxograma da função *loop()* do *Arduino.*

4.5 Listas de Material / Orçamento

Neste ponto, serão apresentadas as listas de material para os diversos tipos de equipamentos que poderão ser implementados. Não existem diferenças significativas nas soluções e, por esse mesmo motivo, os preços oscilam entre os 138,04€ e os 221,20€.

Na tabela 4.1, pode-se observar a lista de material e o respetivo valor de orçamento para um sistema de controlo de acessos.

Descrição	Quantidade	Preço Unitário	Preço Total
Arduino Ethernet	1	52,77 €	52,77 €
Header macho de 9 pinos	1	0,12 €	0,12 €
Cabo Ethernet CAT 5E 1m	2	0,49 €	0,98 €
Caixa inferior	2	3,08 €	6,16 €
Caixa superior	1	11,07 €	11,07 €
Conector Fêmea 9 pinos 2,54 mm	2	0,74 €	1,48 €
Díodo 1N4001	1	0,18 €	0,18 €
Leitor MIFARE RFID	2	25,34 €	50,68 €
Manga termo-retráctil 9,5mm 30cm	1	0,39 €	0,39 €
Passa Fios	5	0,26 €	1,30 €
Relé de 5V	1	4,24 €	4,24 €
Resistência 1K 0,25W	1	0,05 €	0,05 €
Terminais	20	0,06 €	1,20 €
Terminal Macho 9 pinos 90º 2,54mm	1	0,52 €	0,52 €
Transformador 12V, 2A	1	6,15 €	6,15 €
Transístor 2N2222	1	0,75 €	0,75 €
		TOTAL c/ IVA	**138,04 €**

Tabela 4.1. Lista de material do controlo de acessos.

Na tabela 4.2, pode-se observar a lista de material e o respetivo valor de orçamento para um sistema registo de assiduidade com apenas um leitor e opção para um segundo leitor.

Descrição	Quantidade	Preço Unitário	Preço Total
Arduino Mega 2560	1	39,00 €	39,00 €
Arduino Ethernet Shield	1	35,67 €	35,67 €
Cabo Ethernet CAT 5E 1m	2	0,49 €	0,98 €
Caixa	2	8,00 €	16,00 €
Díodo 1N4001	1	0,18 €	0,18 €
Display LCD 20x4 c/ *BackLight*	1	16,77 €	16,77 €
Leitor MIFARE RFID	1	25,34 €	25,34 €
Manga termo-retráctil 9,5mm 30 cm	1	0,39 €	0,39 €
Passa Fios	5	0,26 €	1,30 €
Relé de 5V	1	4,24 €	4,24 €
Resistência 1K 0,25W	1	0,05 €	0,05 €
Terminais	20	0,06 €	1,20 €
Transformador 12V, 2A	1	6,15 €	6,15 €
Transístor 2N2222	1	0,75 €	0,75 €
		TOTAL c/ IVA	**148,02 €**
Leitor MIFARE RFID	1	25,34 €	25,34 €
		TOTAL c/ IVA	**173,36 €**

Tabela 4.2. Lista de material do registo de assiduidade.

Na tabela 4.3, pode-se observar a lista de material e o respetivo valor de orçamento para o sistema de acesso ao campus.

Descrição	Quantidade	Preço Unitário	Preço Total
Arduino Mega 2560	1	39,00 €	39,00 €
Arduino Ethernet Shield	1	35,67 €	35,67 €
Cabo Ethernet CAT 5E 1m	2	0,49 €	0,98 €
Condensadores 10uF	4	0,06 €	0,24 €
Díodo 1N4001	3	0,18 €	0,54 €
Display LCD 20x2 c/ *BackLight*	2	18,30 €	36,60 €
Leitor MIFARE RFID	2	25,34 €	50,68 €
Manga termo-retráctil 9,5mm 1m	1	1,54 €	1,54 €
Placa PCB 0,4 m^2	1	42,57 €	42,57 €
Regulador de Tensão 5V/1A	1	1,05 €	1,05 €
Regulador de Tensão 12V/1A	1	1,15 €	1,15 €
Relé de 5V	2	4,24 €	8,48 €
Resistência 1K 0,25W	15	0,05 €	0,75 €
Terminais	9	0,06 €	0,54 €
Transístor 2N2222	2	0,75 €	1,50 €
		TOTAL c/ IVA	**221,29 €**

Tabela 4.3. Lista de material do controlo de parque.

4.6 Sumário

Pode-se concluir que os equipamentos desenvolvidos neste projeto são versáteis, apresentando-se como soluções de baixo custo e elevada fiabilidade.

5 Resultados Experimentais

Neste capitulo ir-se-á abordar os resultados experimentais observados.

Numa fase inicial, foram efetuados testes ao tempo de resposta do leitor *MIFARE RFID*, de forma a se poder verificar qual o tempo necessário para que o leitor possa efetuar a leitura de cada cartão *RFID*. Posteriormente, foram efetuados testes ao tempo de resposta do servidor, perante um pedido de um cliente. Como existe uma diferença na ordem dos 100 milissegundos para uma resposta de acesso válido e de um acesso negado, foram efetuados testes aos dois tipos de análises.

Para se ter uma maior fiabilidade nos resultados obtidos, foi desenvolvida uma aplicação em *Arduino* dedicada à análise dos dados.

Registos	**Leitura de RFID [milissegundos]**	**Acesso válido [milissegundos]**	**Acesso negado [milissegundos]**
1	77	579	494
2	72	584	504
3	76	601	493
4	76	601	496
5	75	669	507
6	72	593	495
7	73	599	518
8	76	594	494
9	74	605	496
10	71	594	494
11	75	585	494
12	75	591	497
13	73	582	496
14	73	583	493
15	75	585	495
Média ± Desvio Padrão	**74.2 ± 1.8**	**596.3 ± 21.6**	**497.7 ± 6.9**

Tabela 5.4. Resultados experimentais dos tempos de resposta obtidos.

Todo o código desenvolvido para o *Interface Web*[27] e para o *Arduino*[28] encontra-se disponível para *download* nos *links* mencionados.

[27] http://datahost.pt/isec/InterfaceWeb.tar.gz
[28] http://datahost.pt/isec/CodigoArduino.zip

5.1 Sumário

Em suma, face à comparação com as soluções comerciais existentes, previamente efetuada na seção 2.2, considera-se que os dados obtidos são competitivos. Ao nível de resposta do leitor, esse consegue efetuar a leitura de um cartão em aproximadamente 74 milissegundos. Por outro lado, o servidor consegue garantir um tempo de resposta entre os 493 e os 669 milissegundos. Esta diferença reside não só no facto de que um acesso negado possui um processamento inferior por parte do servidor, mas também devido às variações de carga da própria infraestrutura de rede.

6 Implementação Prática

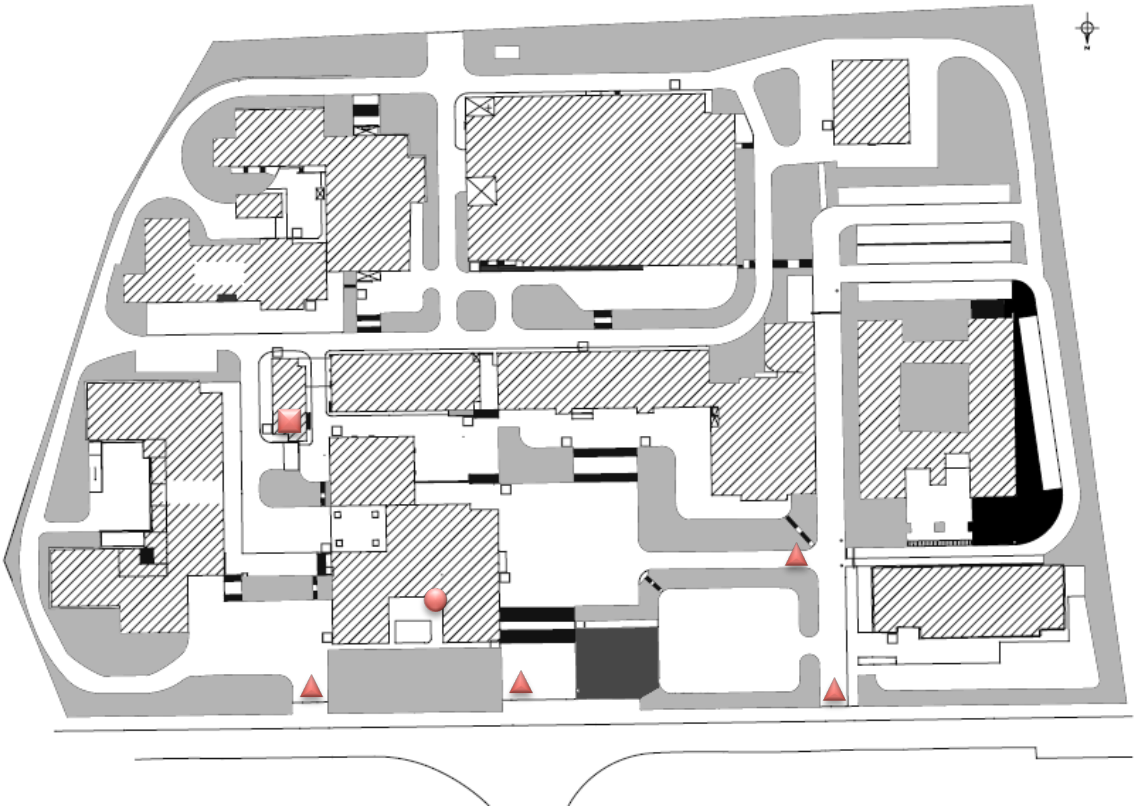
Este projeto foi implemento em duas instituições do *IPC*, nomeadamente no *ISEC* e na *ESEC*. Toda a implementação prática já desenvolvida encontra-se descrita neste capítulo.

De seguida podemos observar a Figura 6.8, onde pode-se ver a implementação de um controlo de acessos no *Auditório* para acesso à *Robocorp*.

Figura 6.1. Controlo de acessos.

6.1 ISEC

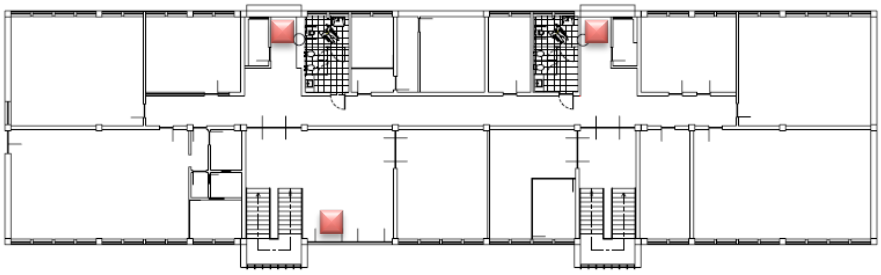
A Figura 6.2. mostra a planificação da implementação no campus *ISEC* realizada no âmbito do *i*SEC. Atualmente, encontra-se implementado um sistema de assiduidade junto à *Presidência do ISEC*, um controlo de acessos no *Auditório*, para acesso à *RoboCorp*, e um controlo de acesso ao campus na cancela de acesso à parte exclusivamente para pessoal docente e não docente.



Legenda:
Controlo de Acessos ; Registo de Assiduidade ; Acesso ao campus ;

Figura 6.2. Planta do campus do ISEC.

O que se pretende, de futuro, será implementar o controlo de acessos em todos os departamentos. Na figura seguinte pode-se visualizar a planta do edifício dos Gerais no *ISEC*. Sendo possível aceder ao mesmo por três portas, sendo necessário implementar três sistemas de controlo de acesso com um leitor *RFID* suplementar cada.



Legenda:
Controlo de Acessos ;

Figura 6.3. Planta do Departamento dos Gerais do ISEC.

6.1.1 Orçamento

Sistema	Quantidades	Preço Unitário	Preço Total
Controlo de Acessos	2	138,04 €	276,08 €
Assiduidade	1	148,02 €	148,02 €
Acesso ao Campus	1	221,29 €	221,29 €
		TOTAL c/ IVA	**1087,96 €**

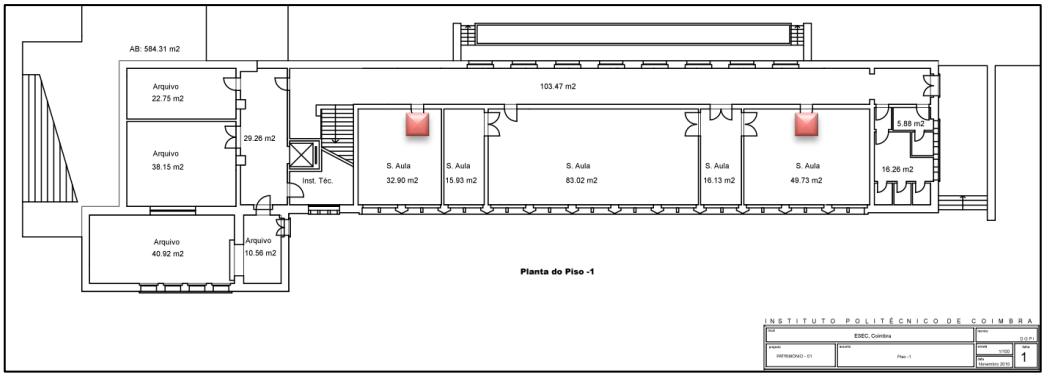
Tabela 6.1. Orçamento do ISEC.

6.2 ESEC

Para implementação da solução proposta na *ESEC*, foi efetuado um estudo inicial, tendo sido feito o levantamento das necessidades da instituição ao nível de controlo de acessos, assiduidade e acesso ao campus. Para o efeito, aferiu-se a necessidade de implementação de 30 sistemas de controlo de acesso, 1 sistema de assiduidade e 2 sistemas de acesso ao campus.

De seguida, apresenta-se a planificação do projeto *i*SEC na *ESEC*, identificando os respectivos pontos onde serão implementadas as soluções estudadas.

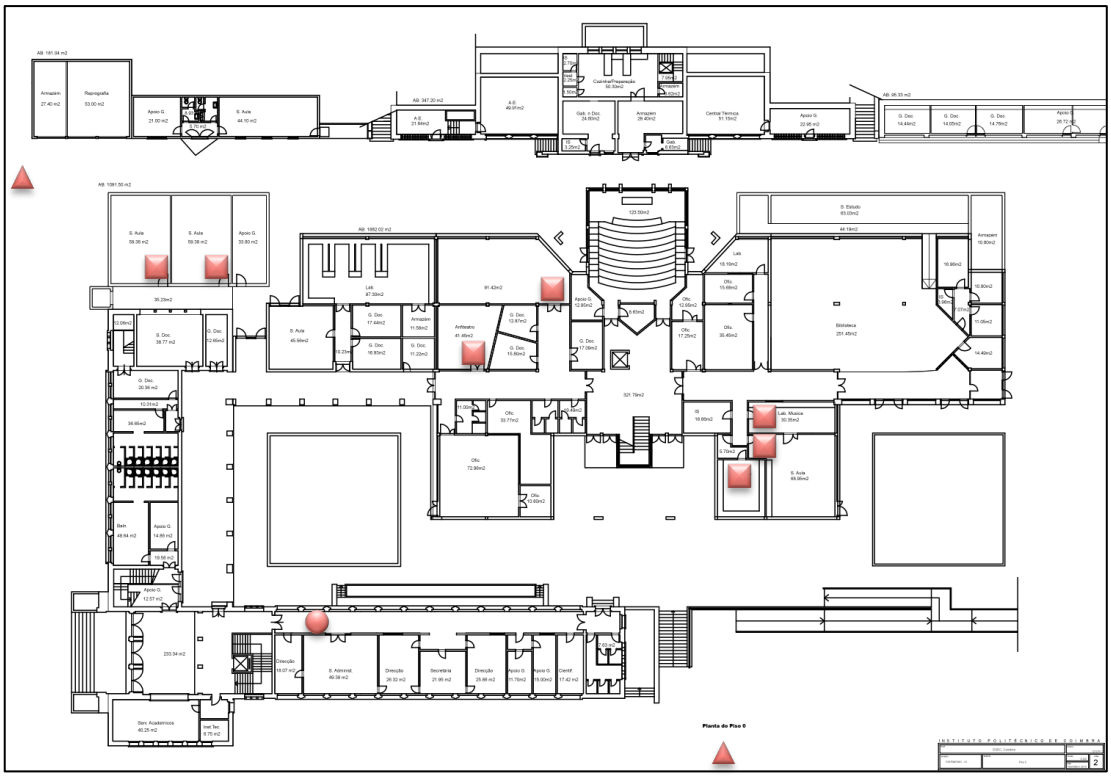
Na Figura 6.4, referente à planta do piso -1, serão implementados 2 sistemas de controlo de acesso.



Legenda:
Controlo de Acessos ;

Figura 6.4. Planta da ESEC (*Piso -1*).

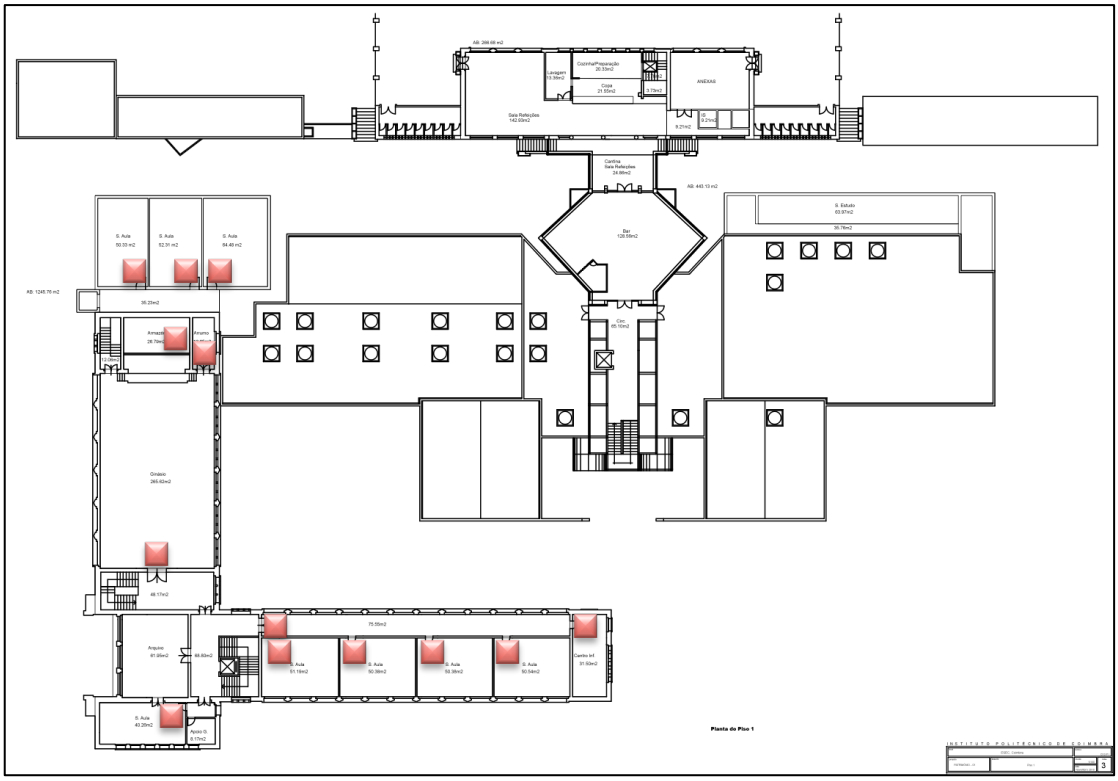
Na Figura 6.5, referente à planta do piso 0, serão implementados 7 sistemas de controlo de acesso, 1 sistema de assiduidade e 2 sistemas de acesso ao campus.



Legenda:
Controlo de Acessos ; Registo de Assiduidade ; Acesso ao campus ;

Figura 6.5. Planta da ESEC (*Piso 0*).

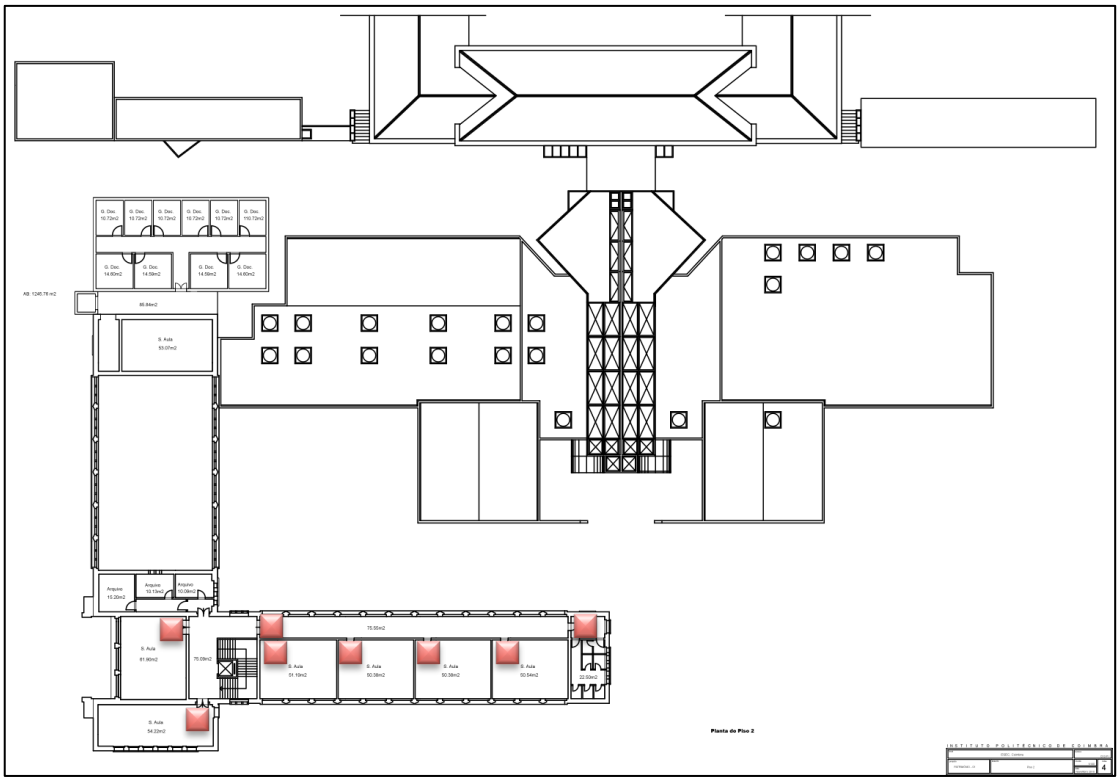
Na Figura 6.6, referente à planta do piso 1, serão implementados 13 sistemas de controlo de acesso.



Legenda:
Controlo de Acessos ;

Figura 6.6. Planta da ESEC (*Piso 1*).

Na Figura 6.7, referente à planta do piso 1, serão implementados 8 sistemas de controlo de acesso.



Legenda:
Controlo de Acessos ; Registo de Assiduidade ; Acesso ao campus ;

Figura 6.7. Planta da ESEC (*Piso 2*).

Até ao momento encontram-se implementados apenas 2 sistemas de controlo de acesso numa zona de gabinetes de professores. O resto da implementação ainda não foi efetuada devido à reestruturação atual das instalações da *ESEC*.

6.2.1 Orçamento

Sistema	Quantidades	Preço Unitário	Preço Total
Controlo de Acessos	30	138,04 €	4.141,13 €
Assiduidade	1	148,02 €	148,02 €
Acesso ao Campus	2	221,29 €	442,58 €
		TOTAL c/ IVA	**4.731,72 €**

Tabela 6.2. Orçamento da ESEC.

6.3 Sumário

Em suma, este capítulo apresentou o estado inicial do projeto de implementação prática das soluções propostas. Torna-se agora necessário implementar as restantes componentes do sistema para que este projeto seja se torne completamente operacional em todos os restantes polos do *IPC*.

7 Conclusão

O presente trabalho propôs-se a implementar uma solução inteligente e sustentável, quer para instituições públicas, quer para privadas.

Foi desenvolvida uma solução integrada e compatível entre todos os módulos do projeto. Considerou-se essencial a capacidade da versatilidade e flexibilidade dos equipamentos, sendo que a solução pode ser atribuídos para um sistema de controlo de acessos ou para um sistema de assiduidade, bastando para o efeito uma simples alteração na configuração da base de dados. Essa capacidade de versatilidade e flexibilidade pode-se de igual modo observar ao nível do próprio *software*, tanto do *Arduino* como do *Interface Web*, sendo que ambas são soluções gratuitas, *open source*, e com uma comunidade ativa.

Anteriormente a este projeto, a programação *C* para *Arduino* era diferente de solução para solução, existindo várias versões consoante a aplicação do cliente. Por exemplo, se fosse necessário um ou dois leitores *RFID* por cliente, também seria necessária uma nova versão de *software*. Neste momento, foi possível desenvolver uma única versão de *software* que contempla o sistema de controlo de acessos e o sistema de registo de assiduidade. Apenas o *software* para o controlo de parques (*i.e.*, acesso ao campus) é distinto dado que o mesmo requer algumas especificidades únicas (*e.g.*, variáveis de entrada).

O *Interface Web* foi desenvolvido de forma a contemplar as três áreas deste projeto. Deste modo, o mesmo é escalável independentemente do tipo e número de clientes existentes na instituição. Por outro lado, a base de dados possibilita de igual modo um número mais elevado de clientes (2^{32} clientes) do que as soluções comerciais atuais no mercado.

A título de conclusão, assume-se ser de extrema importância a concretização de todos os objetivos propostos inicialmente no projeto *i*SEC. Contudo, este Projeto de Mestrado detalha as dificuldades, desafios e soluções associadas ao controlo e gestão de acessos.

8 Trabalho Futuro

Relativamente ao trabalho futuro, é necessário continuar a investir massa crítica humana para concretizar todos os objetivos inerentes ao projeto *i*SEC. Considerando a necessidade em desenvolver um campus educacional inteligente e sustentável, este é um domínio de aplicação em constante desenvolvimento.

A aplicabilidade da tecnologia *RFID* é inquestionável e, seguramente, a quantidade de projetos a dependerem deste tipo de solução continuará a aumentar ao longo dos próximos anos. Como foi referido anteriormente, as instituições bancárias têm desenvolvido novos cartão bancários com a tecnologia *RFID*, o que permite não só o controlo de acessos, mas também efetuar pagamentos até 20€ a 25€ (*e.g.*, *PayWave*). Por esta razão, novos produtos e potencialidades irão emergir num futuro próximo, recorrendo a este tipo de tecnologia como forma de pagamento. A título de exemplo, está previsto no projeto *i*SEC o pagamento das refeições nas cantinas do *IPC* beneficiando deste tipo de tecnologia. Para além das cantinas, poderão interligar-se outras componentes, nomeadamente, o cartão da biblioteca, de forma a permitir um maior controlo e o levantamento de estatísticas do número de utilizadores, e o controlo do sistema de aquecimento central, garantindo não só o controlo de comando do sistema mas também a monitorização de todo o campus da instituição.

A nível do servidor, torna-se necessário implementar uma descentralização do mesmo por intermédio de uma solução em *Cloud*, promovendo a partilha de informação / computação em nuvem, com vários servidores interligados (Catteddu, 2010). Isto permitirá introduzir um sistema de redundância de partilha de dados e processamento dos mesmos podendo, deste modo, ultrapassar potenciais falhas no sistema. A redundância de dados e a interruptabilidade do sistema é algo essencial pois, existindo uma falha no servidor, todo o sistema pode ficar comprometido (*e.g.*, perda das bases de dados de acesso ao campus). Para além disso, ao adicionar uma nova camada de processamento distribuído de dados, conhecido como *Cloud Computing*, pode-se integrar um maior número de soluções, nomeadamente agentes robóticos para vigilância e supervisão das instituições (Waibel *et al.*, 2011).

Referências

Catteddu, D. (2010). Cloud Computing: benefits, risks and recommendations for information security (pp. 17-17). Springer Berlin Heidelberg.

Costa, F., Pereira, S., Rosmaninho, A., Couceiro, M., Figueiredo, C., Santos, V., & Ferreira, N. (2012). "Low-Cost Access Management System in an Educational Environment", EDUTE'12 - 8th WSEAS International Conference on Educational Technologies, July 1-3, Porto, Portugal.

Farooq, U., Amar, M., Ibrahim, H. R., Khalid, O., Nazir, S., & Asad, M. U. (2010). Cost effective wireless attendence and access control system. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on* (Vol. 9, pp. 475-479). IEEE.

Froyen, L. A., & Iverson, A. M. (1999). *Schoolwide and classroom management: The reflective educator-leader*. Upper Saddle River, NJ: Merrill.

Khan, S. R. (2012). Development of Low Cost Private Office Access Control System (OACS). *arXiv preprint arXiv:1212.6196*.

Kishore, S., & Karthikeyan, P. A. (2013). Foolproof Biometric Attendance Management System. International Journal of Information and Computation Technology, ISSN 0974-2239 Volume 3, Number 5, pp. 433-438.

Otley, D., Broadbent, J., & Berry, A. (1995). Research in management control: an overview of its development. *British Journal of management*, *6*(s1), S31-S44.

Patel, R., Patel, N., & Gajjar, M. (2012). Online Students Attendance Monitoring System in Classroom Using Radio Frequency Identification Technology: A Proposed System Framework. International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 2, Issue 2.

Rao, S., & Satoa, K. J. (2013). An Attendance Monitoring System Using Biometrics Authentication. *International Journal*, *3*(4).

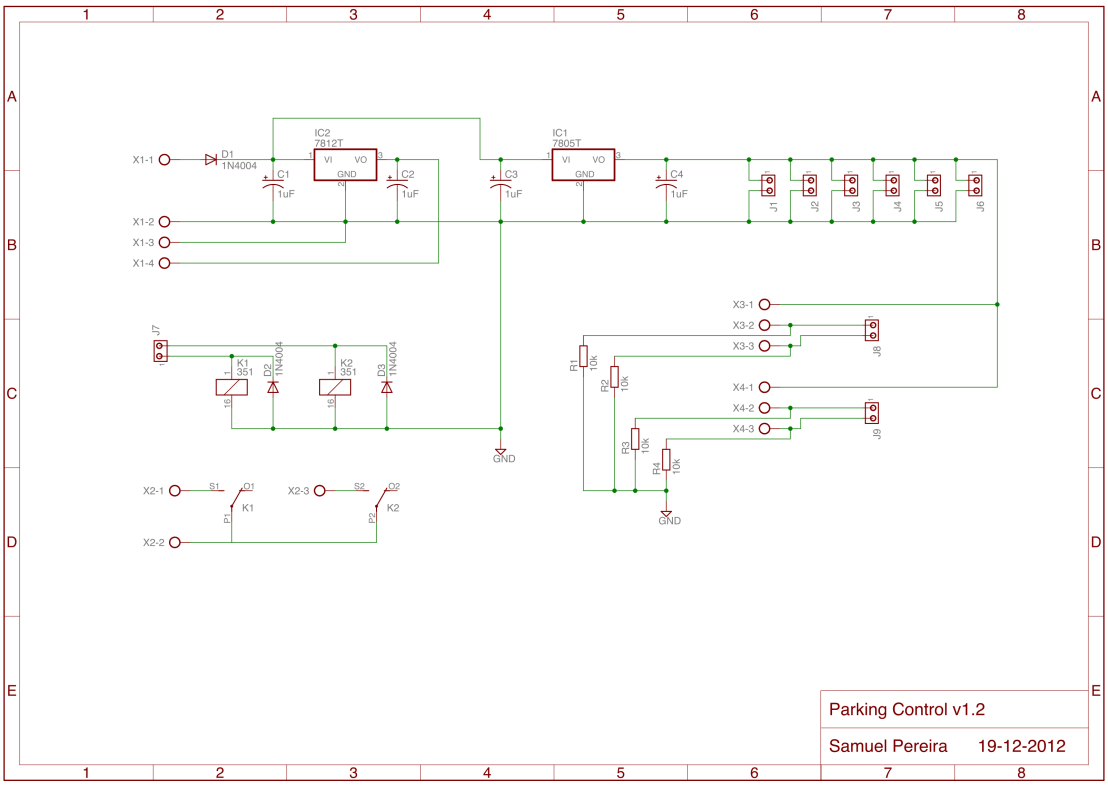
Silva, F., Filipe, V., & Pereira, A. (2008). Automatic Control of Students' Attendance in Classrooms Using RFID. In *Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on* (pp. 384-389). IEEE.

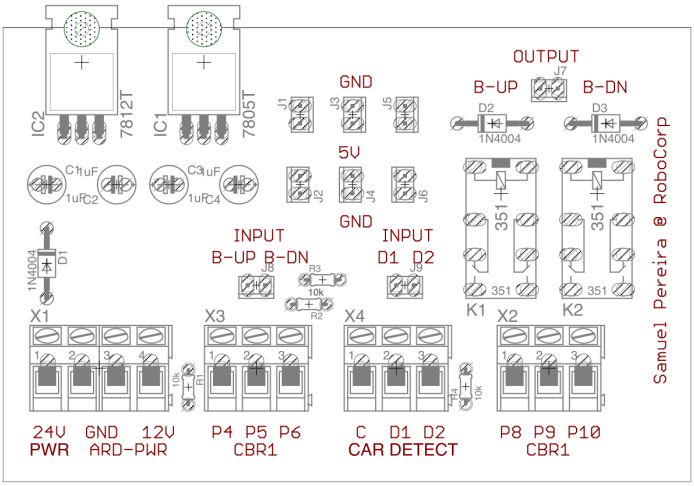
Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., & van de Molengraft, R. (2011). Roboearth. Robotics & Automation Magazine, IEEE, 18(2), 69-82.

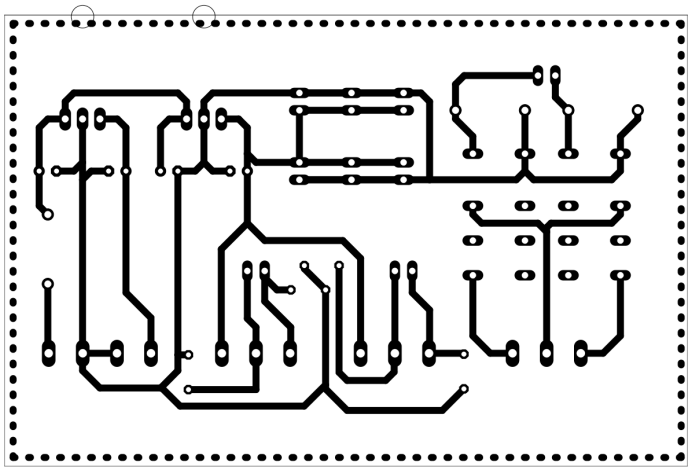
Anexos

A. Schematic e Boards do Eagle

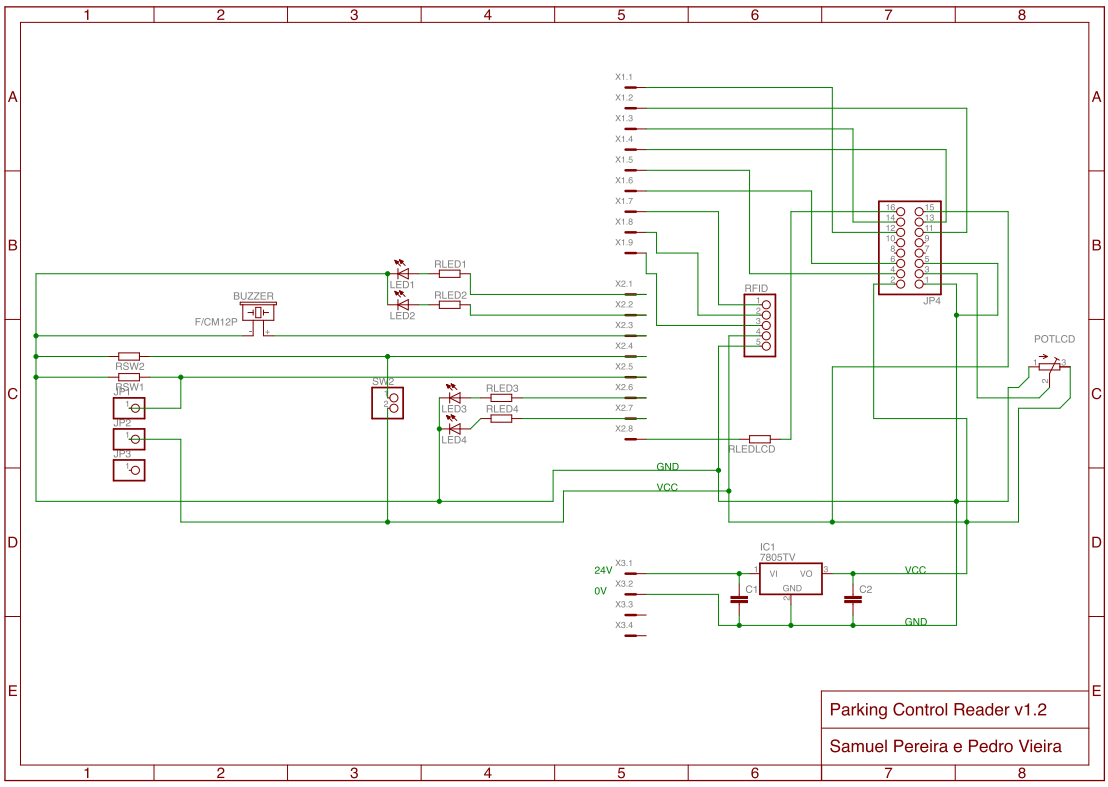
Placa *Parking Control Interface*, interface entre o este projeto atual e as cancelas existentes.

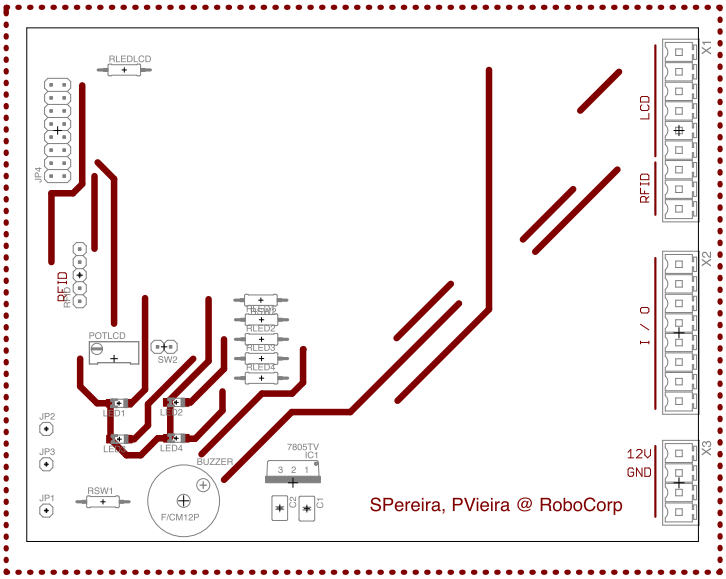


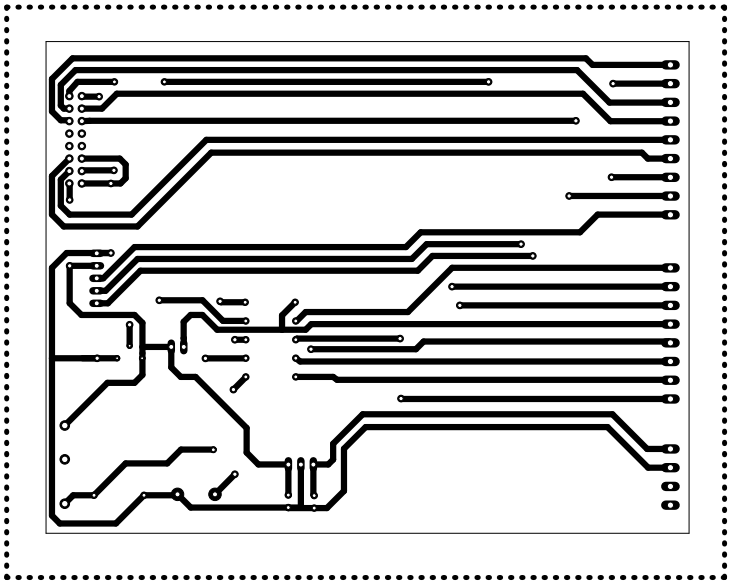




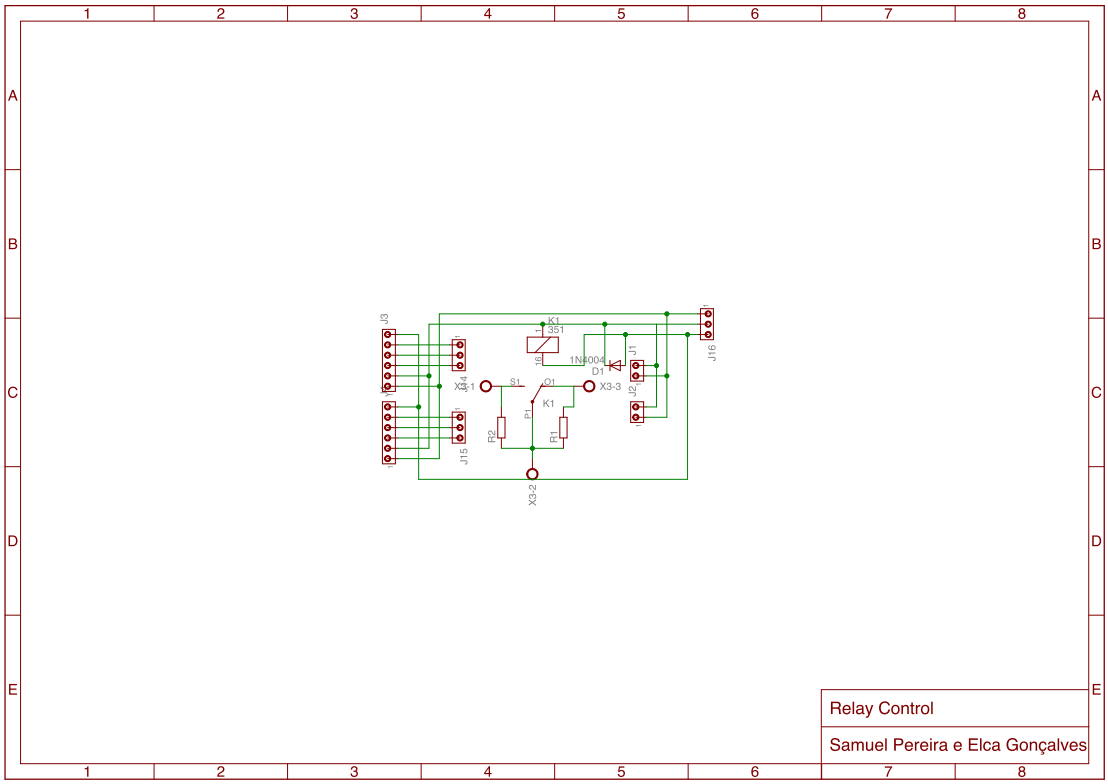
Placa *Parking Control Reader*, interface entre o cliente e o sistema.

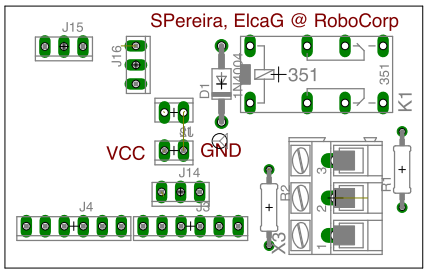


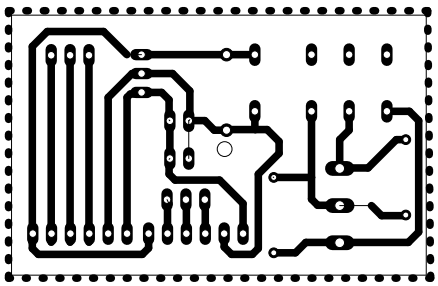

RLEDLCD
JP4
RFID
POTLCD
SW2
RLED1
RLED2
RLED3
RLED4
LED1
LED2
LED3
LED4
JP2
JP3
JP1
RSW1
BUZZER
F/CM12P
7805TV
IC1
3 2 1
C2
C1
X1
LCD
RFID
X2
I / O
X3
12V
GND
SPereira, PVieira @ RoboCorp



Placa *Relay Control*, interface o *Arduino* e a bobina de controlo de um porta.







B. Código do Arduino

```
/*
  Access Control v2.3 for Arduino Mega 2560
  Based on RCorp_35_C2 by Fábio Costa

  Created by Samuel Pereira (2012-11-01)
  Modified by Samuel Pereira (2013-08-01)
*/

const String sVersion ="v2.3";

#include <SPI.h>
#include <Ethernet.h>
#include <LiquidCrystal.h>
#include <SD.h>
#include <Time.h>


/***** PASSWORD CONFIGURATION *****/
const String sPasswd ="010101";

/***** RFID CONFIGURATION *****/
const int iReadersNumber = 2;
const String sWelcome ="  Relogio de Ponto  ";
const String sReaderID1 ="P1Z1R3ID0001"; // PRESENCE CONTROL (PRESIDENCIA/Relógio de P

/***** ETHERNET CONFIGURATION *****/
byte mac[] = { 0x90, 0xA2, 0xDA, 0x04, 0x00, 0xA3 };// PRESENCE CONTROL (PRESIDENCIA/

// PRESENCE CONTROL (PRESIDENCIA/Relógio de Ponto)
IPAddress ip(192,168,30,15);
IPAddress mydns(192,168,120,20);
IPAddress gateway(192,168,30,254);
IPAddress subnet(255,255,255,0);

/***** SYSTEM CONFIGURATION *****/
const int inReader1_SIG = 6;
const int inReader2_SIG = 7;
const int outBuzzer = 46;
const int outSDReserved = 53;

// lcd configuration
LiquidCrystal lcd(32, 30, 28, 26, 24, 22);

char *fileCounter   ="counter.txt";
char *fileOnline    ="online.txt";
char *fileRegister  ="register.txt";
char *fileLog       ="logs.txt";

// server configuration
IPAddress  server(193,137,78,147);// ISEC


/* DON'T CHANGE ANYTHING MORE, UNLESS YOU KNOW WHAT YOU'RE DOING */
#define   RESP_HEADER'T'
```

```
int iReader1State = 0;
int iReader2State = 0;
int iReaderActive = 0;
int iRegisterCounter = 0;
int iResp = 0;


// 0 - Offline  - System Disabled
// 1 - Online   - Register to Server
// 2 - OnlineSD - Register to SDCard
int iSystemStatus = 0;

// 0 - SDCard Disabled
// 1 - SDCard Enabled
int iSDCardStatus = 0;

// Library Time.h
#define TIME_MSG_LEN  11
#define   TIME_HEADER'T'
time_t pctime = 0;

unsigned long lTimeRefresh     = 1000;// Clock Refresh - 1 sec
unsigned long lTimeUpdate  =  1800000;// Clock Update  - 30 min
unsigned long lTimeBlocked    =  60000;// Clock Connection Test - 1 min
unsigned long lTimeOffline  =  300000;// Clock Connection Test - 5 min
unsigned long lTimeBreak       = 1000;// Clock Break While - 1 sec

unsigned long lTimer = 0;
unsigned long lTimerAux = 0;
unsigned long lTimer1 = 0;
unsigned long lTimer2 = 0;
unsigned long lTimer3 = 0;

unsigned char readCardNumber[] = { 0xAA, 0xBB, 0x02, 0x20, 0x22 };
unsigned char cardNumber[4];

String sDate;

// create ethernet client
EthernetClient client;
File myFile, logFile;

void setup() {
  int i = 0;

  // Configure I/O
  pinMode(inReader1_SIG, INPUT);
  pinMode(inReader2_SIG, INPUT);
  pinMode(outBuzzer, OUTPUT);
  pinMode(outSDReserved, OUTPUT);

  TurnOffOutputs ();

  // Start lcd
  lcd.begin(20,4);
```

```
  // Start hardware serial port
  Serial.begin(9600);
  Serial1.begin(19200);

  if (iReadersNumber == 2)
    Serial2.begin(19200);

  // LCD message
  lcdBooting ();

  if (iSDCardStatus == 1) {
    // Initialing SD Card
    sdInitialize ();
  }

  if (iSDCardStatus == 1) {
    // Check the counter
    sdReadCounter (fileCounter);
  }

  // Initialing Ethernet
  ethSystemInit(0);
}


void loop() {

  // If system start without eth connection, block the system
  if (iSystemStatus == 0) {

    if (millis()-lTimer > lTimeBlocked) {
      // Initialing Ethernet
            ethSystemInit(3);
    }

  }
  else {

    if(millis()-lTimer1 > lTimeRefresh) {
       lTimer1 =millis();
      lcdShowClock ();
        }
    else if(millis()-lTimer2 > lTimeUpdate) {
       lTimer2 =millis();
      ethGetDate (1);
      lcdShowClock ();
     }


    // If OnlineSD Test The Connection with Server
    if ((iSystemStatus == 2) && (millis()-lTimer > lTimeOffline)) {
      // Initialing Ethernet
      ethSystemInit(2);
    }
```

```
// Process Offline Registers
if ((iSystemStatus == 1) && (iSDCardStatus == 1) && (iRegisterCounter > 0)) {

  // LCD message
  lcdProcessOnlineSDRegister ();

  // Send offline registers to the server
  int iStatus = sdSendRegisterToServer (fileRegister);

  if (iStatus != -1) {
    // Delete processed files
    sdDeleteFile(fileRegister);
    sdDeleteFile(fileCounter);

    // Update register counter
    iRegisterCounter = 0;

    // LCD message
    lcdOnline ();
  }
  else {

    // LCD message
    lcdOnlineSD ();
  }

  // Update timer
   lTimer =millis();
}


// Interrupt output, LOW level indicates card in the field
 iReader1State =digitalRead(inReader1_SIG);
if (iReadersNumber == 2)
   iReader2State =digitalRead(inReader2_SIG);
delay (50);

if ((iReader1State  ==LOW) || ((iReadersNumber == 2) && (iReader2State ==LOW))) {

  if (iReader1State  ==LOW) iReaderActive = 1;
  else if ((iReadersNumber == 2) && (iReader2State ==LOW)) iReaderActive = 2;

  //  LCD  message
  lcdProcessing ();

  // Read card number from reader
  iResp = readCardFromReader();

  if (iResp == -1) {

    // LCD message
    lcdPleaseTryAgain ();
    delay(1000);
```

```
}
else {

  // Create string from bytes
  String sCardNumber =String(cardNumber[0],HEX)  +String(cardNumber[1],HEX)  +St

  iResp = 0;
  if (iSystemStatus == 1) {
         iResp  =  ethSendData  (sCardNumber,  sDate);
  }
  else if (iSystemStatus == 2) {
     iResp = sdWriteOnFile(fileRegister, sCardNumber);
     delay(500);
  }

  if (iResp == 80) {
    digitalWrite(outBuzzer, HIGH);
    delay(1000);
    digitalWrite(outBuzzer, LOW);
    delay(50);
  }
  else if ((iResp == -1) && (iSDCardStatus == 1)) {
    lcdPleaseTryAgain ();
    delay(1000);

    iSystemStatus = 2;
  }
  else if (iResp == -1) {
    lcdPleaseTryAgain ();
    delay(1000);
  }
  /*else if (iResp == -1)  {
    lcdBlockedReboot ();
    delay(10000);

    resetArduino ();
  } */
  else {
    lcdAccessDenied ();
    delay(1000);
      }

}

// dont read the same card on the same reading
while (iReader1State  ==LOW) {

   iReader1State =digitalRead(inReader1_SIG);
  delay (50);
}

if (iSystemStatus == 1) lcdOnline ();
else lcdOnlineSD ();
```

```
      // Update Timer
       lTimer =millis();
    }
  }
}
```

```
/*
 ETH Module
 Based on Web client Sketch by David A. Mellis (created 18 Dec 2009, modified 9 Apr 201

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13

   Modified by Samuel Pereira (2013-06-14)
*/


int ethRespINT (int num) {
  char cResp[num];

  for (int i=0; i<num; i++) {
    if (client.available()) {
      char c = client.read();
      cResp[i] = c;
    }
    else return 0;
  }
  return atol(cResp);
}


String ethResp (int num) {
  int i;
  String sResp ="";

  for (i=0; i<num; i++) {
    if (client.available()) {
      char c = client.read();
       sResp +=String(c);
    }
    else return 0;
  }
  return sResp;
}


void ethSystemInit (int lcdMessage) {
  int i = 0;

  do {

    switch (lcdMessage) {
      case 0:
        // LCD message
        lcdBooting ();
        break;

      default:
        // LCD message
        lcdRebooting ();
          }
```

```
    // Test connection with the server
      iResp  =  ethGetDate  (lcdMessage);

    i++;
  }
  while ((iResp != 1) && (i<2));

  // Status of the connection with the server
  if ((lcdMessage != 2) && (iResp != 1)) {
    lcdBlocked ();
  }
  else if (iResp == 1) {
    lcdOnline ();
  }
  else {
    lcdOnlineSD ();
  }

  // Update timer
   lTimer  =millis();
}


int ethGetDate (int iOption) {
  iResp = 0;
  String sLog;

  // Start the Ethernet connection
  Ethernet.begin(mac, ip, mydns, gateway, subnet);
  delay(50);

  // log system
   sLog  =String("ETH: Get Date");
   sdLog  (fileLog,  sLog);

  if (client.connect(server, 80)) {
    //DEBUG
    client.print("GET /remote/date.php?reader=");
    client.print(sReaderID1);
    client.print("&id=");
    client.println(iOption);

     lTimer3 =millis();
    while(!client.available()) {
      if(millis()-lTimer3 > lTimeBreak) {
         sLog =String("ETH: Get Date - no answer from HTTP server");
        sdLog (fileLog, sLog);
        break;
      }
    }
    // read answer
    iResp = ethRespINT(2);

    if (iResp == 80) {
```

```
      char c = client.read();
      if(  c  ==  TIME_HEADER  )  {
        for(int i=0; i < TIME_MSG_LEN -1; i++) {
          char c = client.read();
          if( c >='0' && c <='9'){
             pctime = (10 * pctime) + (c -'0')  ;//  convert  digits  to  a  number
          }
         }

        setTime(pctime);
        adjustTime(3600);
        pctime=0;
        //setTime(23,59,50,25,05,2013);

        iResp = 1;
       }
    }

   }
  else {
    // log system
     sdLog (fileLog,"ETH: Connection Failed");
    iResp = -1;
  }

  client.flush();
  delay(50);
  client.stop();

  return iResp;
}


int ethSendData (String  sCardNumber,String sDate) {
  int iHTTP = 1;
  String sName, sLog;

  // Start the Ethernet connection
  Ethernet.begin(mac, ip, mydns, gateway, subnet);
  delay(50);

  // log system
   sLog =String("ETH: Online Card ");
  sLog += sCardNumber;
   sdLog  (fileLog,  sLog);

  if (client.connect(server, 80)) {
    // DEBUG
    client.print("GET /remote/do_on.php?pass=");
    client.print(sPasswd);
    client.print("&reader=");

    if (iReaderActive == 1)
      client.print(sReaderID1);
```

```
    else
      client.print(sReaderID2);

    client.print("&card=");
    client.print(sCardNumber);
    client.print("&date=");
    client.println(sDate);

     lTimer3 =millis();
    while(!client.available()) {
      if(millis()-lTimer3 > lTimeBreak) {
         sLog =String("ETH: Online Card - no answer from HTTP server");
        sdLog (fileLog, sLog);
        iHTTP = 0;
        break;
      }
    }
    // read answer
    iResp = ethRespINT(2);

    if (iResp == 80) {

      char c = client.read();
      if( c == RESP_HEADER ) {
        iResp = ethRespINT(2);
        sName = ethResp(iResp);
        iResp = ethRespINT(1);

        lcdShowName (sName, iResp);
        delay(50);

        iResp = 80;
      }
    }

    if ((iHTTP == 0) || (iResp == 0)) iResp = -1;

   }
  else {
    // log system
     sdLog (fileLog,"ETH: Connection Failed");
    return -1;
  }

  client.flush();
  delay(50);
  client.stop();

  return iResp;
}


int ethSendFromSD (String  sCardNumber,String sDate) {
  int iHTTP = 1;
  String sLog;
```

```
// Start the Ethernet connection
Ethernet.begin(mac, ip, mydns, gateway, subnet);
delay(50);

// log system
 sLog =String("ETH: Offline Card ");
sLog += sCardNumber;
 sdLog (fileLog, sLog);

if (client.connect(server, 80)) {
  // DEBUG
  client.print("GET /remote/do_off.php?pass=");
  client.print(sPasswd);
  client.print("&reader=");
  client.print(sReaderID1);
  client.print("&card=");
  client.print(sCardNumber);
  client.print("&date=");
  client.println(sDate);

   lTimer3 =millis();
  while(!client.available()) {
    if(millis()-lTimer3 > lTimeBreak) {
       sLog =String("ETH: Offline Card - no answer from HTTP server");
      sdLog (fileLog, sLog);
      iHTTP = 0;
      break;
    }
  }
  // read answer
  iResp = ethRespINT(2);

  if (iResp == 80) {

    char c = client.read();
    if( c == RESP_HEADER ) {
      iResp = 80;
    }
      }

  if ((iHTTP == 0) || (iResp == 0)) iResp = -1;

 }
else {
  // log system
   sdLog (fileLog,"ETH: Connection Failed");
  return -1;
}

client.flush();
delay(50);
client.stop();

return iResp;
```

```
}
```

```
/*
  IO Module

  Created by Samuel Pereira (2012-11-01)
  Modified by Samuel Pereira (2013-03-22)
*/


void TurnOffOutputs () {

  digitalWrite(outBuzzer, LOW);

}


void resetArduino () {

  void(* resetFunc) (void) = 0;
  resetFunc();

}


void FormatDigits(char  strOut[3],int num)
{
   strOut[0] ='0' + (num / 10);
   strOut[1] ='0' + (num % 10);
   strOut[2] ='\0';
}
```

```
/*
  LCD Module
  Default LCD Messages

  Created by Samuel Pereira (2012-11-01)
  Modified by Samuel Pereira (2013-06-14)
*/


void lcdHeader () {

  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print(sWelcome);
}

void lcdInit () {

  lcd.clear();
  lcd.setCursor(2,0);
  lcd.print("Access Control");
  lcd.setCursor(7,1);
  lcd.print(sVersion);
  lcd.setCursor(4,2);
  lcd.print("@ RoboCorp");
}

void lcdBlocked () {

  // Update System Status
   iSystemStatus = 0;

   sdLog (fileLog,"LCD: System Blocked");
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("SISTEMA BLOQUEADO");
  lcd.setCursor(1,1);
  lcd.print("SEM ACESSO A REDE");
  lcd.setCursor(2,3);
  lcd.print("CONTACTE O G.I.");
}

void lcdBlockedReboot () {

   sdLog (fileLog,"LCD: System Blocked... SD Card Off & Network Off");
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("SISTEMA BLOQUEADO");
  lcd.setCursor(1,1);
  lcd.print("ERRO NO CARTAO SD");
  lcd.setCursor(1,2);
  lcd.print("SEM ACESSO A REDE");
  lcd.setCursor(2,3);
  lcd.print("CONTACTE O G.I.");
}
```

```
void lcdOnline () {

  // Update System Status
  iSystemStatus = 1;

   sdLog (fileLog,"LCD: System Online");
  lcdHeader ();

  if (iSDCardStatus == 1) {
    lcd.setCursor(19,0);
    lcd.print(":");
  }

  lcdShowClock ();
  lcd.setCursor(2,3);
  lcd.print("Apresente Cartao");
}


void lcdOnlineSD () {

  // Update System Status
  iSystemStatus = 2;

   sdLog (fileLog,"LCD: System Online to SD");
  lcdHeader ();
  lcd.setCursor(19,0);
  lcd.print(".");
   lcdShowClock  ();
  lcd.setCursor(2,3);
  lcd.print("Apresente Cartao");
}


void lcdBooting () {

   sdLog (fileLog,"LCD: booting...");
  lcdInit ();
  lcd.setCursor(0,3);
  lcd.print("a iniciar");
  delay(250);

  lcd.setCursor(9,3);
  lcd.print(".");
  delay(250);

  lcd.setCursor(10,3);
  lcd.print(".");
  delay(250);

  lcd.setCursor(11,3);
  lcd.print(".");
  delay(250);
```

```
}


void lcdRebooting () {

   sdLog (fileLog,"LCD: rebooting...");
  lcdInit ();
  lcd.setCursor(0,3);
  lcd.print("a reiniciar");
  delay(250);

  lcd.setCursor(11,3);
  lcd.print(".");
  delay(250);

  lcd.setCursor(12,3);
  lcd.print(".");
  delay(250);

  lcd.setCursor(13,3);
  lcd.print(".");
  delay(250);
}


void lcdShowName (String  sName,int iStatus) {
  String sLog;
  String sRegister;

  // log system
   sLog =String("LCD: Name: ");
  sLog += sName;
   sLog +=String (", ");
   sLog +=String (iStatus);
  sdLog (fileLog, sLog);

  lcdHeader ();
   lcdShowClock  ();
  lcd.setCursor(0,1);

  if (iStatus == 0)
    lcd.print(" Registo de Entrada ");
  else if (iStatus == 1)
    lcd.print("  Registo de Saida  ");
  else
    lcd.print("  Registo Duplicado ");

  lcd.setCursor(0,3);
  lcd.print(sName);
}


void lcdAccessGranted () {

   sdLog (fileLog,"LCD: Access Granted Offline");
```

```
  //lcdHeader ();
  lcd.clear();
  lcdShowClock ();
  lcd.setCursor(0,2);
  lcd.print("  Registo Efetuado  ");
  //lcd.setCursor(0,3);
  //lcd.print("   Sistema  Offline    ");
}


void lcdAccessDenied () {

   sdLog (fileLog,"LCD: Access Denied");
  lcd.setCursor(0,1);
  lcd.print("    Acesso Negado   ");
  lcd.setCursor(0,3);
  lcd.print("                    ");
  lcd.setCursor(5,3);
  lcd.print("Erro 0x00");
  lcd.print(iResp);
}

void lcdPleaseTryAgain () {

   sdLog (fileLog,"LCD: Please, try again");
  lcd.setCursor(0,3);
  lcd.print("  Tente novamente  ");
}


void lcdProcessing () {

   sdLog (fileLog,"LCD: Processing");
  lcd.setCursor(0,3);
  lcd.print("   A Processar...   ");
}


void lcdProcessOnlineSDRegister () {

   sdLog (fileLog,"LCD: Processing Online SD Registers");
  lcd.setCursor(0,2);
  lcd.print("   A Processar...   ");
  lcd.setCursor(0,3);
  lcd.print("Registos de Memoria ");
}


void lcdSDError () {

   sdLog (fileLog,"LCD: Error on SD Card");
  lcd.setCursor(0,3);
  lcd.print(" Erro no cartao SD  ");
}
```

```
void lcdShowClock () {
  char strOut[3];

  lcd.setCursor(1,0);
  FormatDigits(strOut, day());
  sDate=strOut;
  lcd.print(strOut);
  lcd.print("-");
   FormatDigits(strOut,  month());
  sDate=sDate+strOut;
  lcd.print(strOut);
  lcd.print("-");
  sDate=sDate+String(year()-2000);
  lcd.print(year()-2000);

  lcd.setCursor(11,0);
   FormatDigits(strOut,  hour());
  sDate=sDate+strOut;
  lcd.print(strOut);
  lcd.print(":");
   FormatDigits(strOut,minute());
  sDate=sDate+strOut;
  lcd.print(strOut);
  lcd.print(":");
   FormatDigits(strOut,  second());
  sDate=sDate+strOut;
  lcd.print(strOut);
}


void lcdShowIP () {

  lcd.setCursor(1,3);
  lcd.print("IP: ");
  lcd.print(Ethernet.localIP());

}
```

```
/*
  RFID Module
  Read Card and Verify Check SUM

  Created by Fábio Costa
  Modified by Samuel Pereira (2012-11-01)
*/


int readCardFromReader() {
  int iByte = 0;
  int iStatus = 0;
  int iXOR = 0;

  HardwareSerial *point;
  if (iReaderActive == 1)
    point = &Serial1;
  else if (iReaderActive == 2)
    point = &Serial2;

  point->write(readCardNumber, 5);
  delay(100);

  while(true) {

    if (point->available() > 0) {
      iByte = point->read();

      switch (iStatus) {
        case 0:
          if (iByte == 0xAA) iStatus = 1;
          break;
        case 1:
          if (iByte == 0xBB) iStatus = 2;
          else return -1;
          break;
        case 2:
          if (iByte == 0x06) iStatus = 3;
          else return -1;
          iXOR=iByte;
          break;
        case 3:
          if (iByte == 0x20) iStatus = 4;
          else return -1;
          iXOR=iXOR^iByte;
          break;

        // card number - 4 bytes
        case 4:
        case 5:
        case 6:
        case 7:
          cardNumber[iStatus - 4] = iByte;
          iStatus++;
          iXOR=iXOR^iByte;
```

```
          break;
        case 8:
          if (iByte == iXOR)return 0;
          else return -1;
          break;

        default:
          return -1;
          break;
      }
    }
  }
}
```

```
/*
  SD Module

  Created by Fábio Costa
  Modified by Samuel Pereira (2013-06-14)
*/


void sdInitialize () {

  // On the Ethernet Shield, CS is pin 4. It's set as an output by default.
  // Note that even if it's not used as the CS pin, the hardware SS pin
  // (10 on most Arduino boards, 53 on the Mega) must be left as an output
  // or the SD library functions will not work.
  digitalWrite(outSDReserved, HIGH);
  delay(2000);

  if (!SD.begin(4)) {
    lcdSDError ();
    iSDCardStatus = 0;

    delay(2000);
  }
  else {
     sdLog (fileLog,"SD: Ready");

    // check if necessary file exists
    sdCheck(fileOnline);
    if (iSDCardStatus == 0) {
      sdWriteOnline ();
      iSDCardStatus = 1;
    }
   }
}


void sdCheck (char * filename) {

   myFile =SD.open(filename);
  if (myFile) {

    // read from the file until there's nothing else in it:
    while (myFile.available()) {

      char c = myFile.read();
      if (c =='1') {
        iSDCardStatus = 1;
      }
      else {
        iSDCardStatus = 0;
        break;
        }
    }
    // close the file:
    myFile.close();
```

```
    }
}



void sdDeleteFile (char *filename) {
  String sLog;

  if (SD.remove(filename)  ==true) {
    // log system
     sLog =String("SD: filename ");
    sLog += filename;
     sLog +=String(" removed");
    sdLog (fileLog, sLog);
  }
  else {
    // log system
     sLog =String("SD: cannot remove file ");
    sLog += filename;
    sdLog (fileLog, sLog);
  }
}


void sdReadCounter (char * filename) {
  int i, j;
  char cValue[9];
  String sINI, sLog;

   myFile =SD.open(filename);
  if (myFile) {

    // read from the file until there's nothing else in it:
    while (myFile.available()) {

      // Read char from file and convert to string
      char cINI = myFile.read();
       sINI =String(cINI);

      if (sINI  =="$") {

        for (j=0; j<9; j++) {
          char c = myFile.read();
          if (c  =='#') {
            break;
          }
          else {
            Serial.println(c);
            cValue[j] = c;
            }
        }
        iRegisterCounter = atol(cValue);
      }
```

```
    }
    // close the file:
    myFile.close();

    // log system
     sLog =String("SD: registers found on file ");
    sLog += iRegisterCounter;
    sdLog (fileLog, sLog);

   }else     {
    // log system
     sLog =String("SD: cannot open filename ");
    sLog += filename;
    sdLog (fileLog, sLog);
    }
}


int sdSendRegisterToServer (char * filename) {
  int i,j;
  String sFileCard, sFileDate, sINI, sLog;

   myFile =SD.open(filename);
  if (myFile) {

    // read from the file until there's nothing else in it:
    while (myFile.available()) {

      for (i=0; i<iRegisterCounter; i++) {

        // Read char from file and convert to string
        char cINI = myFile.read();
         sINI =String(cINI);

        if (sINI  =="$") {
           sFileCard ="";
           sFileDate ="";

          for (j=0; j<9; j++) {
            char c = myFile.read();

            if (c  =='%')  break;
            else sFileCard  +=  c;
          }

          for (j=0; j<13; j++) {
            char c = myFile.read();

            if (c  =='#') break;
            else sFileDate += c;
                    }

          iResp = ethSendFromSD (sFileCard, sFileDate);

          if (iResp != -1) {
```

```
            if (iResp == 80) {
              // log system
               sLog =String("SD: Card ");
              sLog += sFileCard;
               sLog +=String(" accepted");
              sdLog (fileLog, sLog);
             }
            else if (iResp != 0) {
               sLog =String("SD: Card ");
              sLog += sFileCard;
               sLog +=String(" rejected");
              sdLog (fileLog, sLog);
            }
            else iResp = -1;

          }
        }
      }
      break;
    }
    // close the file:
    myFile.close();

   }else {
    iSDCardStatus = 0;

    // log system
     sLog =String("SD: cannot open file ");
    sLog += filename;
    sdLog (fileLog, sLog);
   }

   return iResp;
}


int sdWriteOnFile (char  *filename,String sValue) {
  char strOut[3];
  String sLog;

  // check if sdcard is present
  sdCheck (fileOnline);

  if (iSDCardStatus == 1) {

     myFile =SD.open(filename, FILE_WRITE);
    if (myFile) {

      myFile.print("$");
      myFile.print(sValue);
      myFile.print("%");
         FormatDigits(strOut,  day());
      myFile.print(strOut);
         FormatDigits(strOut,  month());
```

```
      myFile.print(strOut);
      myFile.print(year()-2000);
         FormatDigits(strOut, hour());
      myFile.print(strOut);
       FormatDigits(strOut,minute());
      myFile.print(strOut);
         FormatDigits(strOut, second());
      myFile.print(strOut);
      myFile.print("#");

      // close the file:
      myFile.close();

      // log system
       sLog =String("SD: Offline Card ");
      sLog += sValue;
      sdLog (fileLog, sLog);

      // Increment Register Counter
      iRegisterCounter++;
    }

    // Delete file to write new value
    sdDeleteFile(fileCounter);

     myFile =SD.open(fileCounter, FILE_WRITE);
    if (myFile) {

      myFile.print("$");
      myFile.print(iRegisterCounter);
      myFile.print("#");

      // close the file:
      myFile.close();

      // log system
       sLog =String("SD: Card Register on File ");
      sLog += iRegisterCounter;
          sdLog  (fileLog,  sLog);
    }

    lcdAccessGranted ();
    delay(50);

    return 80;

   }else {
    // log system
     sLog =String("SD: cannot open files for writing");
    sdLog (fileLog, sLog);
    return -1;
  }
}
```

```
void sdWriteOnline () {
  char strOut[3];
  String sLog;

     myFile =SD.open(fileOnline, FILE_WRITE);
    if (myFile) {

      myFile.print("1");

      // close the file:
      myFile.close();
    }

    // log system
     sLog =String("SD: created file ");
    sLog += fileOnline;
        sdLog  (fileLog,  sLog);
}


void sdLog (char  *filename,String sString) {
  char strOut[3];

    // send log to serial
    Serial.println(sString);

     logFile =SD.open(filename, FILE_WRITE);
    if (logFile) {

      //DD:MM:YY:HH:II:SS:READERID:STRING
         FormatDigits(strOut,  day());
      logFile.print(strOut);
      logFile.print(":");
         FormatDigits(strOut,  month());
      logFile.print(strOut);
      logFile.print(":");
      logFile.print(year()-2000);
      logFile.print(":");
         FormatDigits(strOut,  hour());
      logFile.print(strOut);
      logFile.print(":");
       FormatDigits(strOut,minute());
      logFile.print(strOut);
      logFile.print(":");
         FormatDigits(strOut,  second());
      logFile.print(strOut);
      logFile.print(":");
      logFile.print(sReaderID1);
      logFile.print(":");
      logFile.println(sString);

      // close the file:
      logFile.close();
    }
}
```