Relatório de projecto

Dispositivo Anti-Carjacking

João Paulo Pigarro Mesquita Spranger

Agosto de 2010

IPCA
INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

Dispositivo Anti-Carjacking

Relatório de projecto de final de curso submetido, como parte dos requisitos, para a obtenção do grau de licenciado em Informática – Ramo Industrial

Realizado por

João Paulo Pigarro Mesquita Spranger

No âmbito da disciplina de projecto leccionada por

Dr. João Luís Araújo Martins Vilaça

Sob orientação na Escola Superior de Tecnologia de

Mestre José Henrique de Araújo Silveira de Brito

Barcelos, Agosto de 2010

"Just when I feel there's no excuse for what happens, things fall into place."

Chuck Schuldiner

Dedicatória

Dedico este projecto à minha família, amigos, e colegas de curso
pela motivação que me deram ao longo do percurso.

Agradecimentos

Ao meu orientador, Mestre José Henrique de Araújo Silveira de Brito, por confiar nas minhas aptidões ao me atribuir este projecto, e também pela disponibilidade demonstrada e apoio prestado ao longo do projecto.

Ao Dr. Luis Gonzaga Martins Ferreira, pelo apoio prestado em situações pontuais no decorrer do projecto.

Resumo

Este projecto implica a criação de um sistema Anti-carjacking.

O sistema Anti-carjacking é um sistema distribuído, composto por um dispositivo físico Anti-carjacking e um programa para visualização da posição da viatura.

O projecto será, portanto, dividido em duas partes em que uma complementa a construção do dispositivo físico capaz de devolver a posição do carro e desligar a viatura, e a segunda parte refere-se ao desenvolvimento do software de visualização dos mapas.

O dispositivo Anti-Carjacking será essencialmente constituído por um microcontrolador e por dois módulos que irão comunicar com ele através do protocolo RS-232. Um deles será um módulo GSM e o outro um módulo GPS. O módulo GSM será responsável pela recepção e envio de mensagens SMS, e o módulo GPS será responsável pela recepção das coordenadas.

No decorrer do projecto será efectuado um estudo do funcionamento do protocolo RS-232, do microcontrolador e de ambos os módulos, será também esquematizado um circuito plausível de solucionar o objectivo pretendido de acordo com a programação que será implementada no microcontrolador.

Efectuar-se-à um estudo relativo ao software para visualização das coordenadas recebidas pelo dispositivo e respectivo desenvolvimento

Índice

Lista de Tabelas

Lista de Figuras

Siglas

GSM	Global Systems for MobileCommunications
GPS	Global Positioning System
SMS	Short Message Service
IVA	Imposto sobre o Valor Acrescentado
ITU	International Telecommunication Union
CCITT	Comit Consultatif International Tlphonique et Tlgraphique
EIA	Electronic Industries Alliance
BPS	Bits Per Second
ETSI	European Telecommunications Standards Institute
DTE	Data Terminal Equipment
DCE	Data Circuit-Terminating Equipment
PDU	Protocol Data Unit
NMEA	National Marine Electronics Association
USART	Universal Synchronous/Asynchronous Receiver/Transmitter
PCB	Printed Circuit Board
TTL	Transistor-Transistor Logic
API	Application Programming Interface
FIFO	First In First Out
CR	Carriage Return
LF	Line Feed
JSON	JavaScript Object Notation
RAM	Random Access Memory
ISP	In-System Programming
LED	Light-Emmiting Diode

Introdução

O carjacking é um atentado grave à segurança pública que nos últimos anos cresceu exponencialmente, sofrendo um pico de 70% casos registados em 2008 relativamente ao ano 2007. Em 2008 foram adoptadas medidas para combater o carjacking, sendo uma dessas medidas sensibilizar a população para adquirir dispositivos capazes de localizar remotamente as suas respectivas viaturas.(iol.pt, 2008)

As soluções actualmente disponíveis no mercado, de uma forma geral, são dispendiosas, quer seja pelo preço do dispositivo ou pela necessidade de adesão a um serviço contractual, implicando uma modalidade de pagamento contínua.

Este projecto pretende satisfazer a necessidade de existir um sistema distribuído, de baixo custo, que permita ao utilizador conhecer e visualizar a posição global da sua viatura a um dado instante e, caso seja pretendido, desligá-la.

Para o desenvolvimento de tal sistema, está implicita a criação de um dispositivo físico que permita retornar as coordenadas actuais da viatura, ou desligá-la, mediante códigos enviados para o dispositivo através de mensagens SMS, e também o desenvolvimento de um software que permita mostrar ao utilizador as coordenadas retornadas pelo dispositivo num mapa.

Nessa medida o projecto está dividido em duas partes, sendo que a primeira se refere ao desenvolvimento do dispositivo de geolocalização, nomeadamente o dispositivo Anti-Carjacking, e a segunda aborda o desenvolvimento do Software de visualização dos mapas.

Segue-se uma breve descrição dos objectivos para cada uma dessas partes.

Dispositivo Anti-Carjacking

Pretende-se utilizar um microcontrolador Atmel®, que será programado em linguagem C, para estabelecer comunicação com um módulo GSM e com um módulo GPS através do protocolo RS-232. Todo o controlo do dispositivo será feito através de mensagens SMS.

Ao receber uma mensagem SMS com o código “2716carLocation”, o microcontrolador terá de comunicar com o módulo GPS por forma a obter as coordenadas actuais da sua localização e reenviá-las, também através de mensagem SMS, para o número de telefone que enviou a mensagem SMS para o dispositivo.

Também será possível desligar o automóvel ao enviar para o módulo GSM ligado ao microcontrolador uma mensagem SMS com o código “2716carShutdown”. Este procedimento será implementado através do controlo de um relé, que deverá estar ligado à ignição do automóvel e deverá ser actuado directamente pelo microcontrolador.

Na figura 1, no quadrado marcado a azul, é possível ver o diagrama de blocos para o dispositivo pretendido.

Software de visualização dos mapas

Pretende-se que o software seja capaz de comunicar com um módulo GSM através do protocolo RS-232, permitindo a leitura das mensagens SMS e a selecção e extracção das coordenadas retornadas pelo dispositivo. Na figura 1, no quadrado marcado a vermelho, é possível ver o diagrama de blocos para a interface entre o software e o módulo GSM.

Deverá ser possível escolher as coordenadas que se pretendem visualizar no mapa e mostrar uma rota provável entre a sequência das coordenadas seleccionadas. Nessa medida, terá de ser mantido um histórico de coordenadas que respeite uma ordem cronológica.

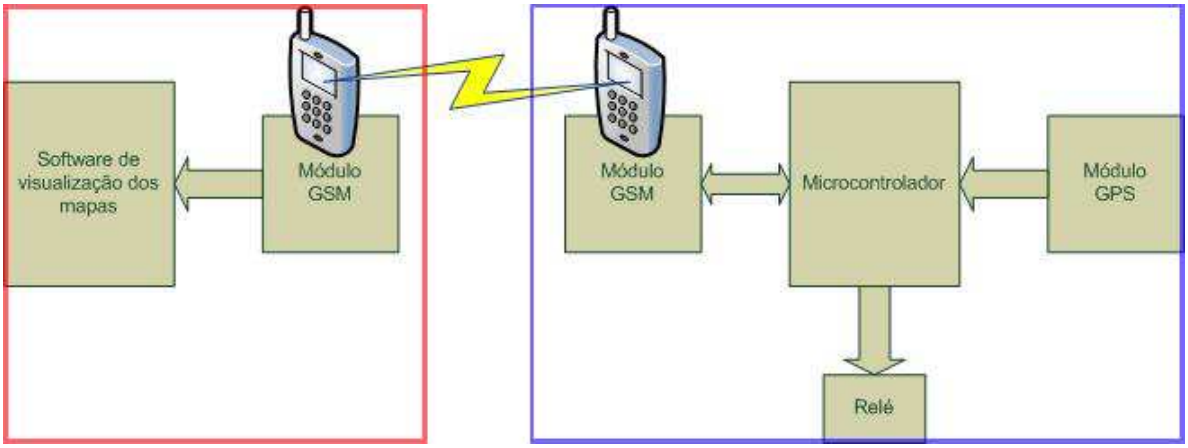

Figura 1 — Diagrama de blocos do sistema distribuído pretendido

Estado da arte

Hoje em dia existe uma quantidade considerável de dispositivos no mercado capazes de detectar e transmitir as suas coordenadas actuais. O dilema da situação passa pelo custo envolvido na obtenção de um equipamento geolocalizador, considerando a relação custo-qualidade a nível de funcionalidades.

Um exemplo de um dispositivo anti-carjacking é o GeoTrack da Geo24. Este dispositivo serve o propósito de geolocalizador e as suas funções mais relevantes são a função de enviar por mensagem SMS as coordenadas e admitir baixo consumo. O preço do equipamento é 369€. ("GeoTrack,")

Outro exemplo de um dispositivo deste género é o NV-auto. Este dispositivo complementa as funções referidas do GeoTrack (à excepção do baixo consumo) e possibilita também desligar a viatura remotamente, entre outras funções não tão relevantes. O preço do dispositivo é 350 €. ("Sistema de Localização anti-Carjacking NV-Auto e NV-Moto,")

Com base nos dispositivos descritos, estima-se que os preços de sistemas Anti-Carjacking com estas funcionalidades, sem necessidade de provedores, rondem um custo, pelo menos, apartir dos 300 €. O benefício na aquisição de um sistema deste tipo é não ser necessária intervenção de terceiros, pelo que o custo associado, à partida, resume-se ao preço do dispositivo. Uma desvantagem consiste nos recursos limitados que estes dispositivos oferecem em comparação a um serviço. A título de exemplo, os dispositivos não incluem um software para visualização das coordenadas recebidas e se esse software fosse incluído, o custo do dispositivo seria potencialmente mais elevado.

Um exemplo de um serviço providenciado por terceiros é o Wireless SafeCar da Vodafone, as funcionalidades oferecidas por este serviço são a geolocalização através da utilização de módulos GPS e GSM, retornando uma mensagem SMS com as coordenadas ao utilizador através de um código, a possibilidade de desligar a viatura e monitorização constante da viatura por parte do provedor do serviço. Para um cliente particular o serviço custa 512 € e tem uma mensalidade de 7 € (Preços sem IVA). ("Wireless SafeCar,")

Relativamente aos dispositivos anteriores, o custo de instalação é consideravelmente mais elevado e implica uma mensalidade. A vantagem deste serviço é contemplar o funcionamento dos dispositivos referidos acima e incluir um serviço de monitorização.

Enquadramento teórico

No Mundo da Electrónica é comum a necessidade de existirem dispositivos moduladores e/ou codificadores que sirvam o propósito de estabelecer interface entre um dispositivo terminal e uma rede exterior que contenha um sinal eléctrico com o qual o dispositivo terminal não esteja apto a interagir. Ao dispositivo modulador/codificador intitula-se de DCE, e ao dispositivo terminal dá-se o nome DTE. (Strangio, 2006)

O exemplo, provavelmente mais comum, desta aplicação seria um modem (dispositivo DCE) que efectua a conversão entre o sinal lógico/digital provindo de um computador (dispositivo DTE) e o sinal analógico da rede telefónica.

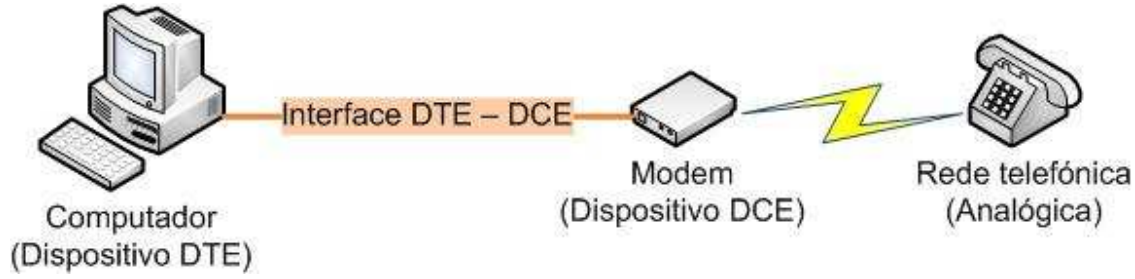

Figura 2 — Interface DTE-DCE

Conforme disposto na figura 2, a interface DTE-DCE corresponde à conexão física existente entre o dispositivo DTE e o dispositivo DCE.

Existem alguns protocolos standard que facultam lógica de conectividade a esta interface, de entre eles o X.25 estabelecido pela empresa ITU (anteriormente conhecida como CCITT) ou o RS-232-C (mais conhecido como RS-232) estabelecido pela empresa EIA.

No âmbito deste projecto, toda a comunicação entre os seus dispositivos constituintes será efectuada seguindo as especificações do protocolo standard RS-232, o qual será descrito em seguida.

Protocolo RS-232

Especificações

O protocolo RS-232 visa assegurar a coerência da comunicação da interface DTE-DCE através das seguintes especificações:

1. A mesma tensão deve ser considerada para os terminais de ambos os dispositivos DTE e DCE, sendo que os sinais de tensão para o protocolo RS-232 variam entre +5V e +15V para um sinal alto (corresnpondente a um 0 lógico) e entre -5V e -15V para um sinal baixo (correspondente a 1 lógico), existindo uma margem de erro de 2V para o dispositivo receptor (ver figura 3). ("Fundamentals of RS-232 Serial Communications," 1998)

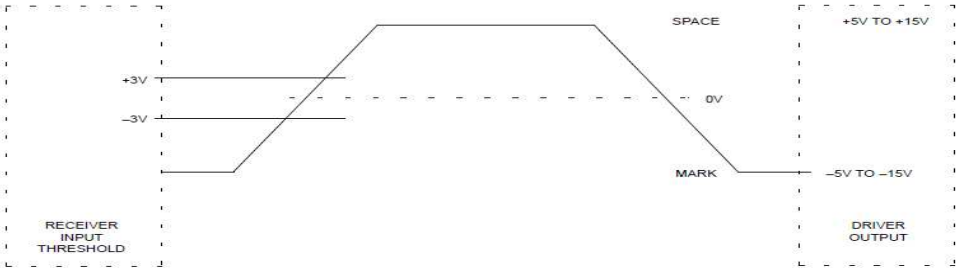

Figura 3 — Níveis lógicos RS-232 ("Fundamentals of RS-232 Serial Communications," 1998)

2. Estabelecer ligações físicas em comum entre os dispositivos DTE e DCE. Os conectores físicos standard que são utilizados em comum com o protocolo RS-232 são o conector DB-25 (Porta paralela) e DB-9 (Porta série) conforme disposto na figura 4. ("Fundamentals of RS-232 Serial Communications," 1998)

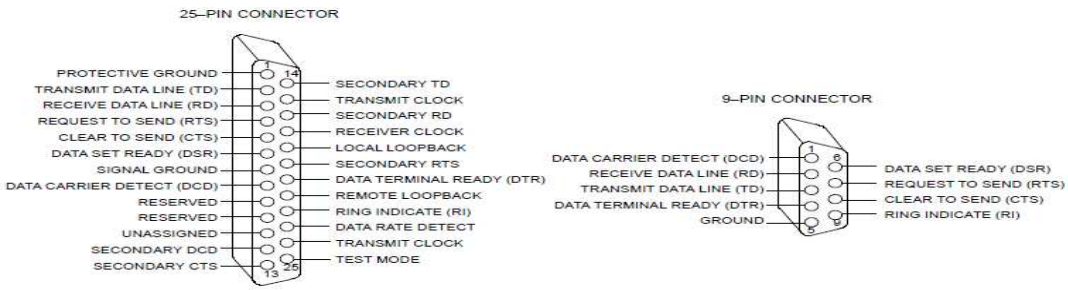

Figura 4 — Conector DB-25 e DB-9 ("Fundamentals of RS-232 Serial Communications," 1998)

3. Minimizar a quantidade de informação de controlo processada pelos dispositivos DTE e DCE, tarefa que é distribuída por pinos de controlo presentes no conector DB-25.). ("Fundamentals of RS-232 Serial Communications," 1998)

Na sequência deste projecto será implementada uma versão minimalista do protocolo RS-232 em que somente serão utilizados os pinos TD, RD e Ground, não respeitando o ponto 3 referido anteriormente.

A função dos pinos TD e RD são nomeadamente a de transmissão e recepção de dados, estando geralmente presentes em ambos os dispositivos DTE e DCE, potenciando (não exlusivamente) a transmissão de dados em Full-Duplex dependentemente da funcionalidade dos dispositivos envolvidos na troca de dados. (Strangio, 2006)

Formato da mensagem

As mensagens (frames) trocadas através de RS-232 contêm valores lógicos (binários). Uma particularidade da comunicação por série é o facto da transferência do dispositivo emissor para o dispositivo receptor ocorrer sequencialmente de bit a bit, conforme é possível verificar na figura 5. (Hill, 2008)

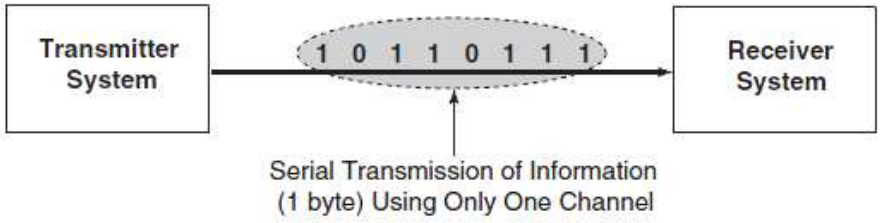

Figura 5 — Transmissão série de valores lógicos (Hill, 2008)

A comunicação por série pode ser síncrona ou assíncrona, sendo que a comunicação síncrona está dependente de um temporizador externo que funciona através dos pinos Transmit clock e Receive clock dispostos no conector DB-25 conforme está presente na figura 4. (Hill, 2008)

Porém, como já foi referido, o projecto em questão utiliza uma versão minimalista do protocolo RS-232 em que os pinos referidos no parágrafo anterior não serão utilizados, nessa medida será descrito apenas o funcionamento do protocolo em modo assíncrono.

O modo assíncrono implica a necessidade de estruturar uma sequência de valores lógicos (bits) que sejam compreendidos pelo receptor, ou seja, existe a necessidade de o receptor conhecer, no mínimo, em que bit a mensagem começa e em que bit acaba. Para esse efeito existem bits específicos que têm funções particulares no conteúdo da frame:

- Start bit – Somente pode existir um Start bit por cada frame, e serve para notificar o receptor que os bits seguintes compõem os dados da frame.(*RS-232: Serial Ports*, 2002)
- Stop bit – Podem existir um ou dois Stop bits em cada frame. A sua função é a de dar a conhecer ao receptor que a frame terminou.(*RS-232: Serial Ports*, 2002)
- Handshake – (Opcional) Sequência de bits emitida antes do Start bit que visa preparar o receptor para a recepção de uma nova frame. (Hill, 2008)

- Parity bit – (Opcional) Bit adicionado imediatamente antes do(s) Stop bit(s) que permite determinar se a frame foi recebida com ou sem erros. Esta verificação é feita através de 3 modos:

    1. Even: Se a contagem dos bits a serem transmitidos for par, o parity bit será enviado com o valor lógico 1, caso contrário terá o valor lógico 0.

    2. Odd: Se a contagem dos bits a serem transmitidos for ímpar o parity bit será enviado com o valor 1, caso contrário terá o valor lógico 0.

    3. Mark: O valor lógico do parity bit enviado será 1, e o receptor irá verificar se ao ser recebido o valor se mantém igual.

(*RS-232: Serial Ports*, 2002)

Listados os bits funcionais contidos na frame RS-232, somente sobram os Data bits, que são os bits de dados da frame. Em teoria podem existir 4, 5, 6, 7 ou 8 Data bits numa frame RS-232, podendo variar dependentemente dos dispositivos. (*RS-232: Serial Ports*, 2002)

Apesar de este tipo de comunicação assíncrono não estar dependente de um temporizador externo, os dispositivos que se pretende que comuniquem necessitam estar configurados para comunicar na mesma frequência, intitulada Baud Rate (medido em BPS). O intuito dessa configuração é estabelecer uma frequência pela qual o receptor possa diferenciar os bits que são transmitidos. ("RS232 Specifications and Standard," 2010)

A título de exemplo, se fossem transmitidos algures na frame dois bits que correspondessem a 0 lógico, o receptor poderia considerar como tendo recebido somente um bit se não existisse uma frequência que determinasse a duração que deve ser considerada para cada bit recebido, já que entre os dois bits não existiria diferenciação no sinal lógico.

Na seguinte figura é possível observar a representação de uma frame RS-232 composta por um Start bit, um Stop bit e 8 Data bits (1 byte de dados). É comum os data bits de uma frame serem interpretados pelos dispositivos como um caracter da tabela ascii[1].

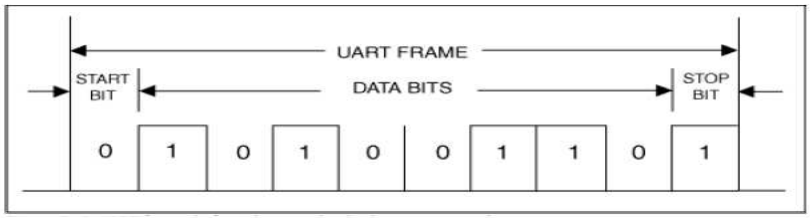

Figura 6 — Exemplo de uma frame RS-232 ("IrDA and RS-232: A Match Made in Silicon," 2004)

[1] Tabela que associa valores binários ou hexadecimais a um respectivo caracter.

Dispositivo Anti-Carjacking

Conforme visto anteriormente na figura 1, é possível verificar que o dispositivo Anti-Carjacking em sí é um sistema distribuído, e nessa medida, para compreender como o sistema deverá comunicar entre sí, é necessário conhecer os protocolos e normas que respeitam cada um dos módulos constituintes.

Módulo GSM - Início

O módulo GSM utilizado no decorrer deste projecto será um telemóvel Siemens M50.

O módulo GSM é um dispositivo DCE (modulador/codificador) que irá estabelecer conectividade com o microcontrolador através de uma interface DTE-DCE utilizando o protocolo RS-232 para a troca de mensagens (ver figura 7).

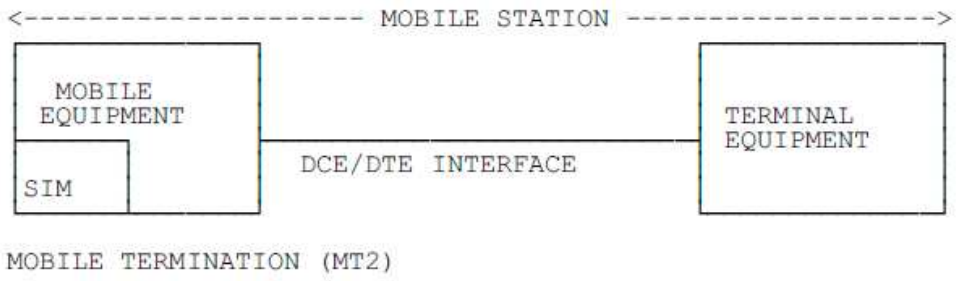

Figura 7 — Interface DCE/DTE entre módulo GSM e terminal ("GSM 07.05," 1997)

A troca de mensagens SMS em módulos GSM respeita o standard GSM 07.05 emitido pela ETSI. Nessa medida irá ser feito um estudo desse mesmo standard. ("AT command Set: Siemens Cellular Engines," 2002)

Módulo GSM - GSM 07.05

O standard GSM 07.05 distingue 3 protocolos para efectuar a troca de mensagens entre o módulo e o dispositivo terminal, sendo estes o “Block Mode”, “Text Mode” e o “PDU Mode”. No decorrer do projecto somente serão utilizados o “Text Mode” e o “PDU Mode”, portanto, o “Block Mode” não será abordado, inclusive porque só serão utilizados os pinos para recepção, transmissão e ground e tal abordagem só pode ser conseguida através dos dois modos referidos. ("GSM 07.05," 1997)

O “Text Mode” e o “PDU Mode” funcionam através de uma linha de comandos que segue a recomendação V.25ter da ITU. A síntaxe utilizada para ambos os modos intitula-se “comandos AT”, em que cada comando é dado pelo prefixo AT seguido do comando pretendido, e em teoria termina em <CR>, podendo variar conforme o dispositivo. No caso do módulo utilizado neste

projecto, através de tentativa e erro denotou-se que o término dos comandos é dado por <CR><LF>. ("AT command Set: Siemens Cellular Engines," 2002)

A resposta obtida em teoria apresenta o formato <CR><LF><resposta><CR><LF>, porém, para os módulos utilizados no projecto (inclusive porque não é removido o eco dos caracteres enviados) por observação experimental foi determinado, no geral, o formato AT<comando><CR><LF><CR><CR><LF><resposta><CR><LF>. ("AT command Set: Siemens Cellular Engines," 2002)

Durante o projecto somente serão utilizados alguns comandos AT para estabelecer comunicação com os dispositivos, esses comandos passarão a ser descritos em seguida:

- AT: (Somente o prefixo) Serve para testar a sincronização entre o dispositivo DTE e DCE, as respostas esperadas são <CR><LF>OK<CR><LF> ou <CR><LF>ERROR<CR><LF>. ("AT command Set: Siemens Cellular Engines," 2002)

- AT+CMGF=[<mode>]: [<mode>] geralmente é 0 ou 1, que correspondem respectivamente a PDU Mode ou Text Mode. Este comando permite escolher entre o modo PDU ou ou modo de texto. As respostas esperadas são <CR><LF>OK<CR><LF> ou <CR><LF>ERROR<CR><LF>. ("AT command Set: Siemens Cellular Engines," 2002)

- AT+CMGL=<status>: <status> representa o estado da mensagem (lida, por ler, etc), para este projecto será utilizada a opção “ALL” que efectua uma listagem de todas as mensagens SMS no dispositivo, as respostas esperadas são uma sequência de respostas semelhantes ao comando AT+CMGR para cada mensagem SMS, podendo encontrar um <CR><LF>ERROR<CR><LF> no processo ou um <CR><LF>OK<CR><LF> no final. ("AT command Set: Siemens Cellular Engines," 2002)

- AT+CNMI=[<mode>], [<mt>], [<bm>], [<ds>], [<bfr>]: Este comando serve para indicar a chegada de uma nova mensagem SMS através de comunicação série ao dispositivo terminal. Para o projecto em questão a indicação de nova mensagem SMS será <CR><LF>+CMTI: <mem>, <index> <CR><LF> onde <mem> corresponde ao sítio na memória onde a mensagem está guardada e <index> corresponde ao índice da mensagem recebida.("AT command Set: Siemens Cellular Engines," 2002)

- AT+CMGR=<index>: <index> neste comando têm o mesmo significado que o referido no ponto anterior. O comando serve para efectuar a leitura da mensagem SMS com o índice disposto em <index>, transferindo-a do dispositivo DCE para o dispositivo DTE através de RS-232. No modo de texto o conteúdo da mensagem é recebido em caracteres ascii não codificados. A resposta esperada neste caso será +CMGR: <status>, <emmiter>, <datetime><CR><LF>Conteúdo da Mensagem SMS<CR><LF>OK<CR><LF>, em que <status> corresponde ao estado da mensagem, <emmiter> corresponde ao número de telefone de onde proveio a mensagem e datetime corresponde à data e hora a que foi recebida a mensagem. No modo PDU os dados da mensagem são recebidos em caracteres que representam código hexadecimal, esses caracteres contém os dados da mensagem de forma codificada. A descodificação da mensagem será abrangida na sequência deste projecto. A resposta esperada para o dispositivo utilizado no modo PDU é +CMGR:

<status>,<emmiter>,<length><CR><LF>Mensagem no formato PDU<CR><LF>. ("AT command Set: Siemens Cellular Engines," 2002)

- AT+CMGS=<length><CR><PDU>0x1A: Este comando somente será utilizado no modo PDU, <length> corresponde ao comprimento da mensagem pdu menos o Centro de Serviço em octetos. <pdu> é o conteúdo da mensagem codificada e 0x1A é o código hexadecimal que corresponde ao atalho Ctrl+Z, que serve para enviar a mensagem. As respostas esperadas são +CMGS: <index><CR><LF>OK<CR><LF> ou <CR><LF>ERROR<CR><LF> no caso de erro. ("AT command Set: Siemens Cellular Engines," 2002)

Módulo GSM - Codificação PDU

As mensagens SMS compreendidas pelos módulos GSM respeitam o alfabeto GSM de 7 bits de acordo com o standard GSM 03.38, porém a recepção e envio são efectuados com caracteres de 8 bits. ("GSM 03.38,")

A mensagem PDU é uma mensagem constituída por caracteres que dois a dois representam octetos. Cada um desses octetos tem um significado específico no conteúdo da mensagem. Existem dois formatos PDU, que são o formato da mensagem recebida e o da mensagem enviada. Nas tabelas 1 e 2 são representados exemplos práticos destes formatos. A mensagem PDU segue o standard GSM 03.40. ("GSM 03.40,")

Um exemplo prático da mensagem 2716carShutdown recebida no formato PDU é 07915391260100002 40C9153914612339100000180132272754 00FB25BCC360ECBA7E83A9DFCBEBB01 e o seu significado encontra-se descrita na tabela 1.

Tabela 1 — Formato de uma mensagem PDU recebida

Octeto	Descrição
07	Comprimento da informação do Centro de Serviço
91	Tipo de número do Centro de Serviço (Internacional)
539126010000	Número do Centro de Serviço (codificado)
24	Primeiro octeto da mensagem recebida
0C	Comprimento do número do emissor
91	Tipo de número do emissor (Internacional)
539146123391	Número do emissor
00	Identificador de protocolo
00	Esquema de codificação de dados
01801322727540	Data e hora
0F	Comprimento do conteúdo
B25BCC360ECBA7E83A9DFCBEBB01	Conteúdo (codificado em octetos)

A mensagem .outputGPS.ShutdownSuccess enviada no formato PDU é 0001000C915391461233910000 1BAE779D0EAFD38FD0A96B8AAED3C9EFBB1B34AD8FC7E5F91C, e o seu significado é descrito na tabela 2.

Tabela 2 — Formato de uma mensagem PDU enviada

Octeto	Descrição
00	Comprimento da informação sobre o Centro de Serviço
01	Primeiro octeto da mensagem a enviar
00	TP-Message-Reference (Irrelevante)
0C	Comprimento do número do emissor
91	Tipo de número do receptor (Internacional)
539146123391	Número do receptor
00	Identificador de protocolo
00	Esquema de codificação de dados
1B	Comprimento do conteúdo
AE779D0EAFD38FD0A96B8AAED3C9EFBB1B34AD8FC7E5F91C	Conteúdo (Codificado em octetos)

Conforme se pode verificar, o formato para comprimento do número do emissor e o próprio número do emissor não se altera independentemente da mensagem ter sido enviada ou recebida, e por isso não existe interesse em descodificar/codificar qualquer um desses campos.

O conteúdo, por outro lado, tem de ser interpretado pelo microcontrolador para poder ser efectuada uma tomada de decisões em relação à mensagem recebida. Nessa medida o conteúdo necessita ser descodificado no microcontrolador para ser compreendido, e posteriormente um novo conteúdo tem de ser codificado para ser enviado de volta ao emissor. A figura seguinte ilustra parte da conversão entre 7 bits e 8 bits do conteúdo da mensagem descrita na tabela 2. A conversão presente na figura servirá como base para estruturar os algoritmos que serão utilizados no programa do microcontrolador.

Hex	AE	77	9D	0E	AF	D3	8F	+++++++	D0
Octets	10101110	01110111	10011101	00001110	10101111	11010011	10001111		11010000
septets	0101110	1101111	1110101	1110100	1110000	1110101	1110100	1000111	1010000
Character	.	o	u	t	p	u	t	G	P

Figura 8 — Conversão entre 7 e 8 bits (Ekkebus)

Módulo GPS - Início

O módulo GPS utilizado no decorrer do projecto será uma antena GPS Navilock 303p.

O módulo em questão também é um dispositivo DCE que estabelecerá comunicação com o microcontrolador através de uma interface DTE-DCE controlada por RS-232.

O protocolo standard de comunicação utilizado pelo módulo GPS é o NMEA 0183.

Módulo GPS - NMEA 0183

O protocolo NMEA 0183 é responsável pelas características eléctricas que permitem estabelecer comunicação com o dispositivo DTE e pelo formato das mensagens transferidas pelos módulos GPS que respeitem esse protocolo. A nível eléctrico, os dispositivos que seguem este standard assumem um baud rate igual a 4800 BPS, 8 Data bits, 1 Start bit e 1 Stop bit, e não utilizam bits de paridade ou handshake. (Betke, 2000)

Relativamente à comunicação, conforme disposto na figura 1, o módulo GPS somente irá transmitir dados para o microcontrolador, não se verificando o oposto. Nessa medida será necessário compreender o formato da mensagem transmitida pelo módulo GPS que será recebida pelo microcontrolador.

As mensagens transmitidas de acordo com o protocolo NMEA 0183 são frases sequenciais em que cada frase contém um identificador e dados. Cada frase começa com o caracter $ e acaba com a sequencia <CR><LF>.(Betke, 2000)

A frase que será utilizada para determinar a posição global do módulo será a frase que contem o identificador GGA, que contém a latitude, longitude, hora e outros dados não relevantes para o projecto. Na figura seguinte está representado o formato da frase GGA. ("NMEA Reference Manual," 2008)

```
                                                      11
        1         2       3 4        5 6 7  8   9  10 |  12 13  14   15
        |         |       | |        | | |  |   |   | |   |  |   |    |
$--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh

 1) Time (UTC)
 2) Latitude
 3) N or S (North or South)
 4) Longitude
 5) E or W (East or West)
 6) GPS Quality Indicator,
    0 - fix not available,
    1 - GPS fix,
    2 - Differential GPS fix
 7) Number of satellites in view, 00 - 12
 8) Horizontal Dilution of precision
 9) Antenna Altitude above/below mean-sea-level (geoid)
10) Units of antenna altitude, meters
11) Geoidal separation, the difference between the WGS-84 earth
    ellipsoid and mean-sea-level (geoid), "-" means mean-sea-level below ellipsoid
12) Units of geoidal separation, meters
13) Age of differential GPS data, time in seconds since last SC104
    type 1 or 9 update, null field when DGPS is not used
14) Differential reference station ID, 0000-1023
15) Checksum
```

Figura 9 — Formato da frase GGA do protocolo NMEA-0183 (Betke, 2000)

Microcontrolador - Início

Um microcontrolador é um circuito integrado com capacidade de cálculo e endereçamento de variáveis binárias. (Crisp, 2004)

A estrutura básica de um microcontrolador é tipicamente composta por unidade de memória RAM, unidade central de processamento, unidade(s) de Input/Output (Portas), unidade(s) de comunicação em série, oscilador, watchdog, conversor analógico-digital (ver figura 9). Naturalmente, diferentes controladores podem contemplar outras unidades. (Crisp, 2004)

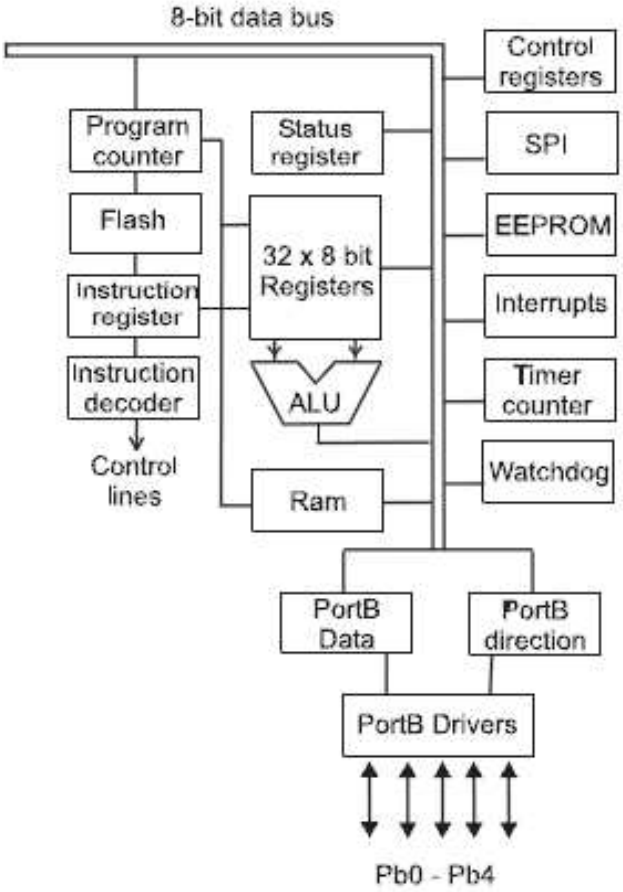

Figura 10 — Estrutura básica de um microcontrolador AVR (Crisp, 2004)

Para este projecto será utilizado o microcontrolador AVR Atmega88 da empresa Atmel, que é um microcontrolador de 8 bits[2] com 1 Kilobyte de ram. Segue-se uma descrição das funcionalidades do microcontrolador que serão utilizadas no decorrer do projecto.

[2] O processamento dos dados é efectuado através da manipulação de 8 bits (1 byte) de cada vez

Microcontrolador - USART

A unidade USART serve o propósito de estabelecer comunicação série com outros dispositivos. Fisicamente, a unidade USART utiliza três pinos do microcontrolador, sendo estes o Rx, Tx e Ground, respectivamente para o intuito de receber e transmitir dados, e estabelecer um Ground comum com o dispositivo com o qual se pretende comunicar. ("ATmega88," 2009)

Em termos do formato das mensagens enviadas/recebidas, a unidade USART funciona de acordo com o protocolo RS-232 descrito anteriormente, porém, é uma implementação minimalista em termos de quantidade de pinos e difere nos valores de tensão. Sendo que a transmissão do valor lógico 1 corresponde a +5V e o valor lógico 0 corresponde a 0V. ("ATmega88," 2009)

Relembrando o que foi discutido anteriormente acerca dos Data bits, que são os bits de dados da frame, o microcontrolador dispõe de um buffer de 8 bits que é utilizado para guardar provisoriamente os Data bits enviados ou recebidos (o mesmo buffer é utilizado para ambas as funções). Este buffer adopta uma técnica FIFO relativamente aos bits, ou seja, o primeiro bit a entrar no buffer é também o primeiro bit a sair do buffer, independentemente de se tratar de uma recepção ou transmissão.("ATmega88," 2009)

Para o ATmega88 o nome dado ao buffer é UDR0, e está representado na figura seguinte.

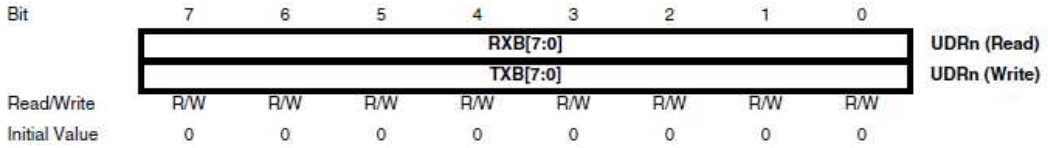

Figura 11 — Ilustração do registo UDR0 ("ATmega88," 2009)

Da mesma forma que o formato da frame pode ser configurado através de bits com funções especiais, o mesmo acontece com a USART, sendo que essa configuração é feita através da atribuição de valores lógicos em bits específicos nos registos de controlo da unidade USART.

Para o ATmega88 existem três registos de 8 bits que controlam as configurações da transferência de informação por série, que são:

- UCSR0A:

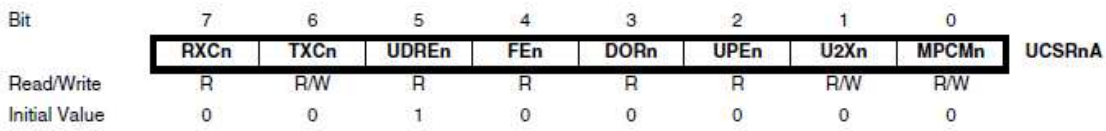

Figura 12 — Ilustração do registo UCSR0A ("ATmega88," 2009)

Dos bits da figura anterior só existe necessidade de abordar o RXC0 e o UDRE0. Esses bits são flag bits cuja função é a de controlo e podem originar interrupções (que serão estudadas no decorrer do projecto).

- RXC0: É o bit 7 do registo UCSR0A. A particularidade deste bit é adoptar o valor lógico 1 quando existem Data bits no buffer por ler, e tomar o valor 0 quando o buffer está vazio. ("ATmega88," 2009)

- UDR0: É o quinto bit do registo UCSR0A e a sua função é a de adoptar o valor lógico 1 quando o buffer UDR0 está vazio e o valor 0 quando existem dados no buffer. ("ATmega88," 2009)

Estes dois bits serão responsáveis pelo controlo da leitura e escrita por USART durante o projecto.

- UCSR0B:

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 13 — Ilustração do registo UCSR0B ("ATmega88," 2009)

Os bits presentes na figura 12 relevantes para o projecto são o RXCIE0, RXEN0 e TXEN0.

- RXCIE0: É o bit 7 do registo UCSR0B e a sua função é a de habilitar a ocorrência de uma interrupção quando são recebidos dados no buffer UDR0, caso o seu valor lógico seja escrito a 1. ("ATmega88," 2009)

- RXEN0: É o bit 4 do registo UCSR0B e a sua função é ligar o receptor quando lhe é atribuido o valor lógico 1 e desligá-lo quando atribuido o valor 0. ("ATmega88," 2009)

- TXEN0: É o terceiro bit do registo UCSR0B e permite ligar e desligar o emissor consoante lhe seja atribuído o valor lógico 1 ou 0. ("ATmega88," 2009)

- UCSR0C:

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Figura 14 — Ilustração do registo UCSR0C ("ATmega88," 2009)

- UMSEL01 e UMSEL00: São os bits 7 e 6, respectivamente, do registo UCSR0C. Permitem seleccionar o modo de operação síncrono ou assíncrono, por defeito ambos os bits têm o valor lógico 0 e a comunicação é efectuada de modo assíncrono, se o bit UMSEL00 assumir o valor lógico 1, a comunicação passa a ser efectuada de modo síncrono. ("ATmega88," 2009)

- UPM01 e UPM00: São o quinto e quarto bit do registo UCSR0C e permitem habilitar e definir o tipo de paridade da frame. Por defeito ambos os bits têm o valor 0 e a

paridade está desabilitada. Atribuir o valor 1 ao bit UPM01 habilita a paridade e torna-a do tipo Even. Se a ambos os bits for atribuído o valor 1 a paridade é habilitada com o tipo Odd. ("ATmega88," 2009)

- USBS0: O terceiro bit do registo UCSR0C selecciona a quantidade de Stop bits na frame, por defeito o valor deste bit é 0 implicando a existência de somente 1 Stop bit na frame, se o valor do bit for alterado para 1, a frame passará a ter 2 Stop bits. ("ATmega88," 2009)
- UCSZ01 e UCSZ00: Os bits 2 e 1 do registo UCSR0C dizem respeito à quantidade de Data bits que serão transportados na frame. Por defeito ambos os bits têm o valor lógico 1, estabelecendo que serão transportados 8 bits de dados na frame. Na sequência do projecto esse será o valor utilizado. ("ATmega88," 2009)
- UCPOL0: O bit 0 do registo UCSR0C é relevante somente para a comunicação assíncrona. Sendo que irá ser utilizada exclusivamente comunicação assíncrona, este bit deve tomar o valor 0, valor esse já atribuído por defeito. ("ATmega88," 2009)

Conforme foi esclarecido na secção que retrata o formato da mensagem RS-232, os dispositivos que estabelecem comunicação por série necessitam ter a mesma frequência (Baud rate) para que consigam compreender a mensagem trocada entre eles.

No modo assíncrono, as funções responsáveis por calcular os valores necessários para estabelecer o Baud rate pretendido são as dispostas na figura 14.

Antes de abordar o cálculo, é necessário esclarecer que o oscilador é uma fonte de relógio (externa ou interna) que o microcontrolador utiliza para temporizar as instruções que são executadas. (Matic, 2000)

Existe um registo responsável por estabelecer a frequência com a qual o microcontrolador irá comunicar com restantes dispositivos, que é o registo UBRR0. ("ATmega88," 2009)

O registo UBRR0 é composto por 16 bits, porém, os 4 bits mais significativos não são utilizados, restando 12 bits para serem atribuídos. Para o modo assíncrono normal, deve ser atribuído a este registo um valor que quando relacionado com a frequência do oscilador na função representada à esquerda na figura 14 origine um valor igual à frequência pretendida para a transferência dos dados. Conhecendo o valor pretendido para o Baud rate, é possível utilizar a função inversa representada à direita na figura 14 para calcular o UBRR. ("ATmega88," 2009)

$$BAUD = \frac{f_{OSC}}{16(UBRRn+1)} \qquad UBRRn = \frac{f_{OSC}}{16BAUD} - 1$$

Figura 15 — Funções de cálculo do Baud e UBRR0 (modo assíncrono) ("ATmega88," 2009)

Microcontrolador - Portas

As portas do microcontrolador são unidades de Input/Output que permitem a interface com o exterior através de pinos aos quais são atribuídos valores lógicos com tensões de 0V ou +5V. Os pinos podem ser configurados para assumirem a função de entrada ou saída de dados. ("ATmega88," 2009)

Podem existir várias portas num microcontrolador, com diferente quantidade de pinos associados, mas o modo de configurá-las é sempre o mesmo. Os pinos das portas são configurados através de um registo DDRn, em que n é o identificador da porta em questão (exemplo: Em PORTB, B é o identificador da porta) e o tamanho do registo é igual à quantidade de bits da porta correspondente. O procedimento é atribuir o valor 1 aos bits correspondentes aos índices dos pinos que se pretendem configurar como pinos de saída, e efectuar o mesmo procedimento com o valor 0 para os pinos que se pretendem de entrada. ("ATmega88," 2009)

Feita a configuração, para enviar um valor lógico através de um pino de saída de uma porta, é necessário apenas atribuir o valor lógico pretendido ao respectivo pino. ("ATmega88," 2009)

Para este projecto não é necessário abordar a leitura dos pinos.

Microcontrolador - Interrupções

Geralmente, num microcontrolador, os programas são executados num espaço da memória intitulado “Program Space”, porém, se forem configurados eventos para aos quais o microcontrolador terá de dar uma resposta imediata executando um conjunto separado de instruções, quando o evento ocorrer, a execução irá saltar para um espaço distinto da memória conhecido como Vector de interrupção. ("ATmega88," 2009)

Os microcontroladores possuem muitas fontes de interrupções, a única utilizada neste projecto será a interrupção Usart RX.

A interrupção Usart Rx é controlada através do bit RXCIE0 do registo UCSR0B mencionado anteriormente, quando este bit tem o valor 1 a interrupção por Usart Rx é considerada ligada, porém, isso não é o suficiente para habilitar a ocorrência de interrupções. Para habilitar todas as interrupções é necessário atribuir o valor 1 ao bit I (Global Interrupt Enable) do registo de Status (SREG). ("ATmega88," 2009)

Habilitada a interrupção por Usart Rx, será a função do bit RXC0 do registo UCSR0A iniciar a execução da rotina de interrupção cada vez que assumir o valor lógico 1 (quando existirem dados para ler no buffer UDR0). ("ATmega88," 2009)

Microcontrolador – Sleep modes

Um interesse neste projecto é o de evitar um consumo alto de energia quando não for necessário, pois propondo que o dispositivo irá ser alimentado pela bateria do veículo, quando o veículo está parado e o seu alternador[3] não está em funcionamento, este não é capaz de recarregar a bateria do veículo, e se o dispositivo consumir muita energia, poderá o veículo acabar por não ter energia suficiente para ligar.

Os sleep modes do microcontrolador existem para permitir o consumo mínimo de energia quando o microcontrolador não está a ser utilizado para nenhuma função. Esta redução de consumo é conseguida desligando módulos constituintes do microcontrolador que não estejam a ser utilizados. ("ATmega88," 2009)

Existem 5 sleep modes disponíveis para o ATmega88, que são o Idle mode, ADC Noise Reduction mode, Power-Down mode, Power-Save mode e Standby Mode. ("ATmega88," 2009)

Para este projecto foi utilizado o Idle mode porque, apesar de ser menos eficiente que os restantes, permite ao microcontrolador acordar do sleep mode ao ocorrer uma interrupção do tipo Usart Rx, o que vai de encontro com as expectativas do projecto.

Para seleccionar o sleep mode pretendido para o microcontrolador, é necessário alterar os bits SM2, SM1 e SM0 do registo SMCR, a cada sequência de valores lógicos nestes bits está atribuído um modo de sleep. Para o modo de sleep pretendido, o valor lógico para os três bits têm de ser igual a 0. O bit SE desse mesmo registo, quando escrito o valor lógico 1 habilita o sleep mode e o microcontrolador entra em modo de consumo reduzido de energia. ("ATmega88," 2009)

Microcontrolador – Reset e Watchdog timer

Para este projecto existe o interesse de a execução do programa ser ininterrupta (à excepção da interrupção Usart Rx), dessa forma, serão descritas as potenciais fontes que podem interromper a sequência normal do programa, e como desligá-las.

O Reset é considerado uma interrupção. Quando ocorre, as portas assumem os seus valores por defeito e é executado o código que tiver sido criado para a rotina de interrupção deste Vector. Caso não tenha sido atribuído nenhum código, então será executado novamente o início do programa. ("ATmega88," 2009)

O ATmega88 tem quatro fontes de Reset, que são:

- Power-On Reset: Ocorre se a alimentação do microcontrolador estiver abaixo de um nível (Vpot) determinado através da configuração dos fusíveis. Esta verificação é feita antes do início da execução do programa. Como alterar os fusíveis pode causar dano ao microcontrolador, esta fonte de interrupção não foi alterada. ("ATmega88," 2009)

[3]Dispositivo que permite recarregar a bateria quando o veículo está em movimento.

- External Reset: Na sua configuração de pinos, o microcontrolador tem um pino intitulado $\overline{\text{RESET}}$ que deve manter sempre um nível lógico alto durante a execução do programa. No caso de ser induzido um nível lógico baixo, será efectuado Reset ao microcontrolador. ("ATmega88," 2009)
- Watchdog system reset: O watchdog timer é um contador que gera uma interrupção quando o contador expira, ou seja, ocorre overflow na contagem até um valor predeterminado. Uma das interrupções possíveis é do tipo Reset (controlado pelo fusível WTDON). A forma mais simples de desabilitar todas as possíveis interrupções que possam ser geradas pelo watchdog timer é simplesmente desligá-lo (ou não ligá-lo). ("ATmega88," 2009)
- Brown-Out Reset: À semelhança do Power-On Reset, o Brown-Out Reset ocorre se a alimentação do microcontrolador for inferior ao valor Vbot, esta fonte de reset existe para, contrariamente ao Power-On Reset, ser utilizada durante a execução do programa, através de uma comparação constante entre o valor de alimentação e o valor Vbot predeterminado nos fusíveis, comparação efectuada pelo Brown-Out Detector. ("ATmega88," 2009)

Software de visualização dos mapas

A comunicação entre o software de visualização de mapas e o módulo GSM respectivo será efectuada através do standard RS-232, com um cabo proprietário da marca do módulo GSM, logo, na sequência desta parte do projecto somente será necessário configurar o formato da frame trocada entre o computador e o módulo.

O módulo em questão será um telemóvel Vodafone V975.

Tendo em conta o funcionamento proposto do programa, foi elaborado um diagrama de actividades (disposto na figura 15) para estudar a sequência básica de acontecimendos que deverá ocorrer na execução do software.

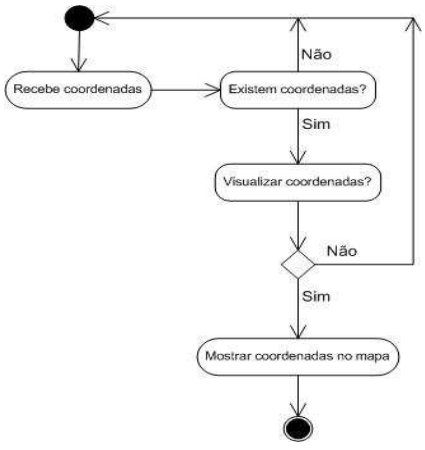

Figura 16 — Diagrama de actividades proposto para o software

Denota-se apartir do diagrama a necessidade de implementar uma interface para comunicação com o módulo GSM e obtenção das coordenadas, já que o resto do programa estará dependente dessa ocorrência. Mas em relação aos mapas não existe ainda conhecimento plausível, e nessa medida foi construído um diagrama de casos de uso com relevo nas principais funções que serão abordadas no tratamento dos mapas.

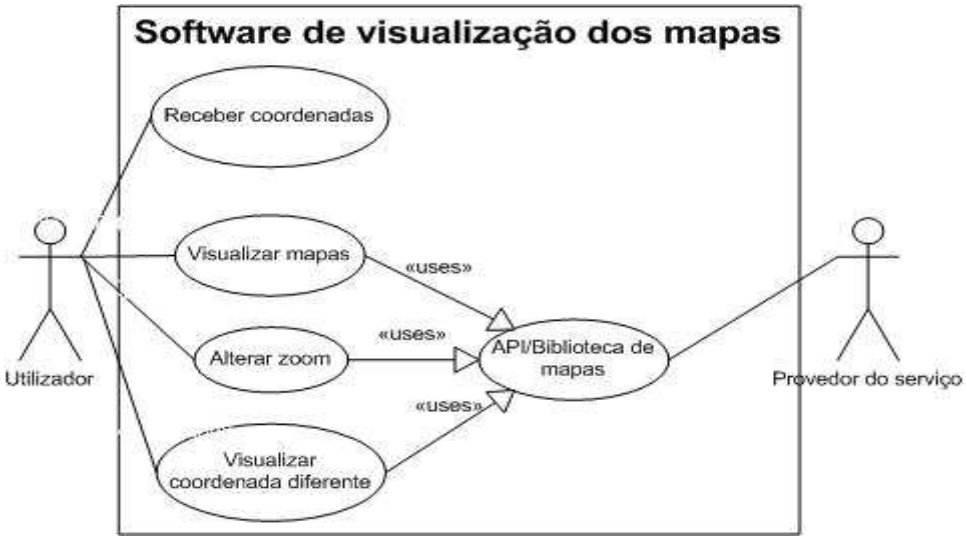

Figura 17 — Diagrama de casos de uso proposto para o software

Em teoria, a biblioteca de mapas em si não deveria fazer parte do sistema do software, ao que deveria ser providenciada remotamente por um provedor e acedida também através de um método de comunicação remoto. Porém, na sua maioria, as APIs dos provedores são estruturadas para serem utilizadas em páginas web, através da utilização de linguagens orientadas para a web (Javascript, php, entre outras), e o programa de visualização será uma aplicação cliente, ocorrendo uma contradição. Existiria a possibilidade de utilizar JSON[4] para estabelecer comunicação com as respectivas APIs, mas tal revelou-se desnecessário porque no entretanto foi tomado conhecimento de um conjunto de bibliotecas opensource[5] que foram desenvolvidas sob o nome Great Maps, e contêm APIs que permitem interacção com mapas de vários provedores.("GMap.NET - Great Maps for Windows Forms & Presentation,")

Deduz-se claramente que a biblioteca representada no sistema do diagrama da figura 14 é, portanto, a biblioteca Great Maps.

Infelizmente, não existe (ainda) documentação acerca da API desta biblioteca, pelo que a sua utilização no decorrer do projecto será maioritariamente intuitiva e através da observação de exemplos de código providenciados pelos seus criadores.

[4] Formato de troca de dados
[5] Software de código aberto

Desenvolvimento

Preparação

Alguns factores têm de ser considerados antes de ser estabelecida conectividade entre os diferentes módulos, tendo em conta os tópicos abordados no capítulo dos conceitos teóricos. Entre estes factores estão a frequência de comunicação dos dispositivos, a tensão de alimentação do microcontrolador, inclusive a nivelação de tensões para permitir retro-compatibilidade entre os níveis lógicos standard do protocolo RS-232 e os níveis lógicos utilizados pelo microcontrolador, entre outros.

Neste capítulo serão abordados tais factores e a forma como serão resolvidos.

Baud rate

Para o sistema distribuído Anti-Carjacking, a comunicação por série será efectuada entre 2 módulos (dispositivos DCE) e o microcontrolador (Dispositivo DTE). Ambos os módulos irão comunicar directamente com o microcontrolador. Não é essencial, mas por uma questão de simplificar o programa, será utilizado sempre o baud rate menor de entre todos os dispositivos.

Conforme foi dito anteriormente, o microcontrolador possibilita a configuração do baud rate através de um cálculo que relaciona o valor do registo UBRR0 com a frequência do oscilador.

No caso do módulo GSM, este está apto a reconhecer a frequência da comunicação automaticamente. Tal funcionamento deve-se ao código do seu firmware[6] proprietário, ao qual não existe acesso público. Esta dedução foi obtida através de realização experimental.

O módulo GSM, conforme foi esclarecido anteriormente, funciona de acordo com o protocolo NMEA 0183 que define a frequência de comunicação como sendo igual a 4800 BPS, portanto, essa será a frequência utilizada para a comunicação entre os dispositivos do sistema ao longo do projecto. Será apenas necessário obter o valor que deverá ser atribuído ao registo UBRR0.

É conhecido o valor a utilizar para o Baud rate, e o valor que será utilizado para o oscilador do ATmega88 é 1.000,000 Hz (1 MHz), que é a frequência obtida através de um circuito RC que se encontra no interior do microcontrolador. Este valor é a sua frequência de oscilação por defeito. ("ATmega88," 2009)

[6] Conjunto de instruções programadas em hardware

Cálculo do UBRR:

$$UBRR0 = \frac{1000000}{16(4800)} - 1 \cong 12,02$$

Ao registo UBRR0 terá de ser atribuído um valor inteiro, que será o 12, e os números que se apresentam após a vírgula são assumidos como um erro. É possível calcular a percentagem do erro através da fórmula presente na figura 17, e se a percentagem de erro obtida for inferior a 0,5%, então o valor para o UBRR0 é aceitável. ("ATmega88," 2009)

$$\text{Error}[\%] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \bullet 100\%$$

Figura 18 — Cálculo da percentagem de erro do Baud rate ("ATmega88," 2009)

Para calcular a percentagem de erro, é necessário primeiro calcular o valor para o Baud rate utilizando o valor o inteiro que será atribuído ao registo UBRR0, neste caso, 12.

Cálculo do BAUD:

$$BAUD = \frac{1000000}{16.(12+1)} \cong 4807,69$$

Cálculo da percentagem de erro:

$$Erro[\%] = \left(\frac{4807,69}{4800} - 1\right).100 \cong 0,16\%$$

É possível verificar que o valor obtido se encontra abaixo dos 0,5%, e portanto, é aceitável para ser utilizado como frequência de comunicação pelo microcontrolador.

Tensões de alimentação

A tensão de alimentação necessária para alimentar o microcontrolador reside entre os valores 2.7V e 5.5V. ("ATmega88," 2009)

Para o módulo GPS a tensão pretendida para funcionamento tem de estar entre 4.5V e 6.5V. ("Product Specification RS232 GPS Receiver NL-303P,")

O problema está na tensão que provém originalmente da bateria do carro, tensão essa que tem o valor teórico de 12 V. Para resolver essa situação será utilizado um circuito integrado regulador de tensão conhecido como 7805. (Ofria, 2006)

Este regulador permite transformar um sinal de entrada que se situe entre os 8V e 20V num sinal de saída constante de 5V, o que vai de encontro com as especificações dos dispositivos aos quais se pretendem alimentar. ("L7800 Series - Positive Voltage Regulators," 2003)

Será utilizado o circuito da figura seguinte, que é a montagem recomendada para regulação de tensão com a série de integrados 78XX.

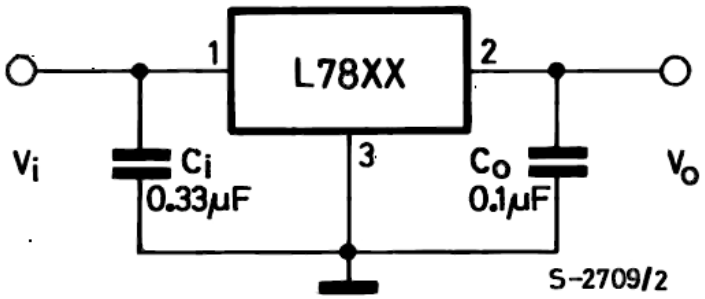

Figura 19 — Circuito regulador de tensão ("L7800 Series - Positive Voltage Regulators," 2003)

A questão da tensão de alimentação do circuito está resolvida, porém, existe outra questão relevante no intuito do projecto. O módulo GSM em questão (Siemens M50) é um dispositivo que funciona com uma bateria. O problema das baterias nesta situação é, eventualmente, descarregarem.

O dispositivo deve estar operacional a 100% do tempo, pois é impossível estimar o momento em que a viatura poderá ser desviada. Para isso é necessário que o módulo não esteja dependente da bateria, mas sim que seja alimentado pelo próprio circuito.

O transformador utilizado para alimentar o aparelho em questão têm a informação acerca da alimentação apagada devido ao tempo e desgaste do equipamento. Nessa medida, a tensão de alimentação foi obtida através de um multímetro[7], medindo os terminais do carregador. Obteve-se uma tensão constante de 8.6V.

Para não exceder e potencialmente danificar o dispositivo, foi utilizada uma tensão inferior à tensão obtida, nomeadamente 8V, através de um regulador de tensão 7808 respeitando a montagem representada na figura 18. Este circuito integrado pode ser utilizado porque as tensões de entrada podem variar entre 11.5V e 23V. ("L7800 Series - Positive Voltage Regulators," 2003)

Ambos os integrados limitam a saída de corrente a 1.5A, presume-se que essa intensidade de corrente será suficiente para o circuito, ou esta abordagem não poderia ser utilizada. ("L7800 Series - Positive Voltage Regulators," 2003)

Interfaces físicas

O protocolo de comunicação que será utilizado pelos dispositivos já está estabelecido, pelo que resta conhecer os adaptadores físicos que permitirão estabelecer as conexões.

[7] Aparelho utilizado para medições de grandezas em circuitos eléctricos.

Conforme foi explicado anteriormente, todos os dispositivos integrantes do sistema terão de partilhar 3 pinos em comum, que são o RD, TD e Ground. Porque se pretende efectuar a comunicação nas duas vias, o TD de um equipamento terá de ser conectado ao RD do outro, assim como o inverso, para que os transmissores de ambos os dispositivos estejam conectados aos receptores dos dispositivos contrários, não esquecendo que os dispositivos têm de partilhar o Ground em comum.

Para o microcontrolador, os pinos que correspondem ao receptor e transmissor são o RXD e TXD, e relativamente ao seu esquema de pinos, são os pinos 2 e 3, respectivamente. O ground situa-se no pino 8. ("ATmega88," 2009)

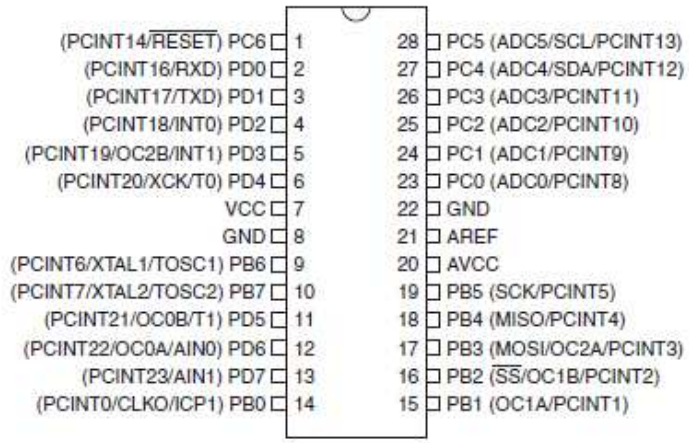

Figura 20 — Estrutura de pinos do ATmega88 ("ATmega88," 2009)

O módulo GPS tem uma interface física mini-DIN[8] macho de 6 pinos que funciona de acordo com o protocolo RS-232, pelo que existirá necessidade de construir um adaptador fêmea para poder conectá-lo ao circuito. O esquema de pinos do conector do módulo está representado na figura 20.

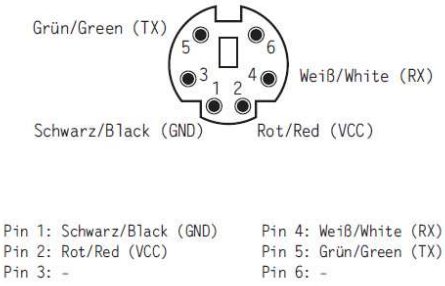

Figura 21 — Conector do módulo GPS ("Product Specification RS232 GPS Receiver NL-303P,")

[8] Semelhante ao utilizado por teclados e ratos antigos (PS\2)

O módulo GSM utiliza um conector proprietário da marca, pelo que será necessário adquirir um conector igual e alterá-lo conforme o esquema de pinos utilizado de acordo com o pretendido para poder estabelecer conectividade com o circuito. No caso deste módulo, os pinos relevantes são o Ground (pino 1), Alimentação (pino 3), Transmissor (pino 5) e Receptor (pino 6). ("C45/M50/MT50 Level2.5e Repair Document v1.0," 2002)

Níveis de tensão

Como foi dito anteriormente, o microcontrolador utiliza níveis lógicos de tensão entre os 0V e +5V, tal como o módulo GSM. A esses níveis são chamados níveis TTL. Porém, o módulo GPS respeita os níveis de tensão do protocolo RS-232, o que implica a necessidade de modelar os sinais dos restantes dispositivos para respeitar as tensões do protocolo.

Para este projecto será utilizado um circuito integrado conhecido como MAX232 cuja função é a de converter os sinais com níveis TTL para níveis RS-232, e vice-versa. Uma vantagem da utilização deste integrado é a sua tensão de alimentação variar entre os 4.5V e 5.5V, pois poderá ser alimentado pelo circuito integrado 7805 referido anteriormente.

Na figura seguinte está representado o circuito básico com o qual o dispositivo é utilizado, e esse será o circuito implementado no decorrer do projecto. ("MAX232, MAX232I Dual Eia-232 Drivers/Receivers," 2002)

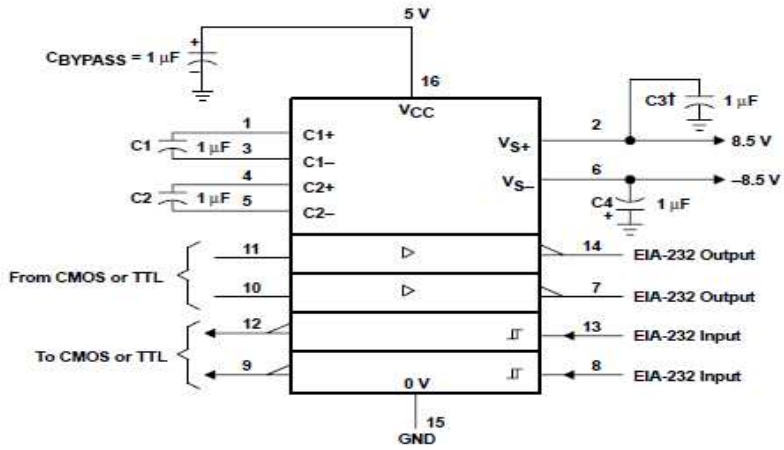

Figura 22 — Montagem do MAX232 ("MAX232, MAX232I Dual Eia-232 Drivers/Receivers," 2002)

Multiplexador

No processo de comunicação, é esperado que o microcontrolador comunique com ambos os dispositivos, mas somente um de cada vez. Para isso ocorrer será necessário existir algum tipo de selecção de linha de comunicação, supondo que os módulos comunicam por diferentes linhas. Para isso, foi considerada a utilização de um circuito integrado multiplexador.

O circuito multiplexador considerado é conhecido como 74157, e a sua tensão de alimentação é 5V. De uma forma simplificada, cada secção do multiplexador é constituída por 2 inputs e 1 output. A cada momento um dos inputs está directamente conectado ao output, a escolha de qual input está conectado ao output é efectuada com um pino de selecção, através da atribuição de um valor lógico em nível TTL. ("74HC/HCT157 Quad 2-input Multiplexer," 1990)

Em relação ao integrado em questão, está representado na próxima figura o esquema de relação dos pinos de input e output, e também as equações booleanas para o cálculo do input escolhido consoante o valor lógico do pino S (Select) e E (Enable).

O pino Enable está invertido, ou seja, só existe conectividade entre um dos inputs e o output quando o valor lógico do pino Enable for 0. Posto isto, é simples compreender o resto da equação. Tomaremos como exemplo a equação 1Y, quando E for 0, se o pino S for 1 estará activo o input $1I_1$, porque o input $1I_0$ será multiplicado por 0 ($\overline{S}$), não esquecendo que se trata de uma operação na base 2. ("74HC/HCT157 Quad 2-input Multiplexer," 1990)

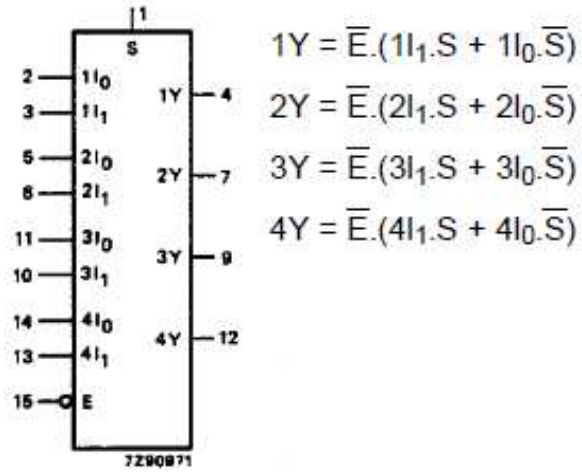

Figura 23 — Esquema lógico e equações ("74HC/HCT157 Quad 2-input Multiplexer," 1990)

Para o efeito do projecto, o pino Enable será ligado directamente ao Ground (0 lógico), portanto, a comunicação input/output estará sempre activa, e porque o microcontrolador é responsável pelo controlo de toda a comunicação por RS-232, o pino Select será controlado por ele.

Relé

Um relé é um interruptor (switch) mecânico controlado electronicamente.

O relé é controlado através de um circuito de baixa voltagem que permite ligar ou desligar uma bobina magnética que é utilizada para movimentar mecanicamente um braço metálico que permite estabelecer a conexão entre input(s) e output(s), conforme representado na figura 23. (Surtell, 1998)

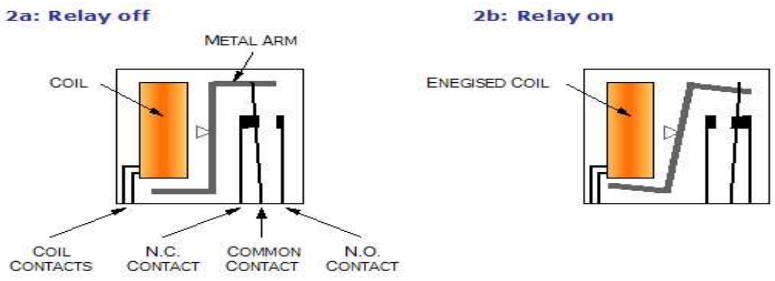

Figura 24 — Modo de funcionamento de um relé (Surtell, 1998)

A grande vantagem da utilização dos relés é a independência do circuito de controlo relativamente aos inputs e outputs. Ao não existir necessidade de incorporar o circuito de tensão alta em comum com o circuito de controlo (tensão baixa), o risco de danificar algum componente do circuito de tensão baixa é inferior.

O relé utilizado para este projecto é um relé de 5V com 2 inputs e 1 output como o que está na figura 23, e vai de encontro com a tensão de alimentação do circuito. Claramente se deduz que os 5V servem para alimentar/ligar a bobina. ("PCB Relay G6E,")

Os pinos do relé em questão são representados na figura seguinte. Os pinos marcados como 1 e 6 são pinos de tensão baixa para ligar ou desligar a bobina, os pinos marcados como 10 e 12 são pinos de input, e o pino marcado como 7 é o pino de output.

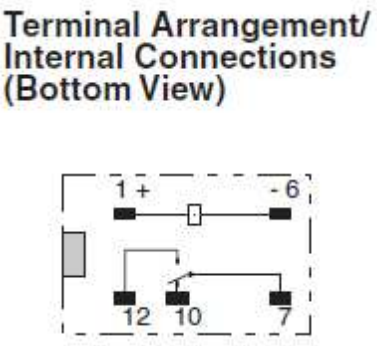

Figura 25 — Esquema de pinos do relé (vista de baixo) ("PCB Relay G6E,")

O esquema adoptado para conectar o relé ao circuito será uma adaptação do circuito da figura 25.

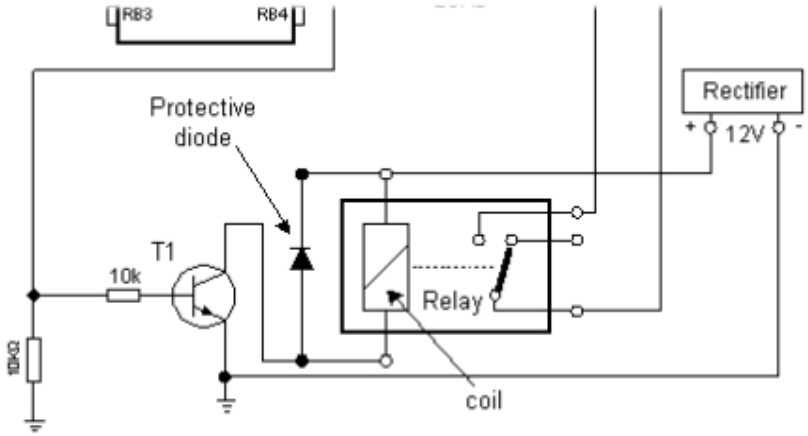

Figura 26 — Exemplo de circuito de ligação do relé (Matic, 2000)

Desenho e construção

De acordo com o estudo formulado na fase de preparação, terá sido estipulado um esquema do circuito que será implementado. O esquema está repartido ao longo das próximas 4 figuras.

Para o circuito só existiu uma abordagem que terá sido levada desde o início e baseada no estudo descrito acima, naturalmente ocorreram complicações durante o processo que levaram a alterações no esquema do circuito. O esquema representado nas próximas figuras está na sua versão final, mas no decorrer deste documento serão descritas as alterações que foram efectuadas relativamente ao conceito original, e as respectivas razões.

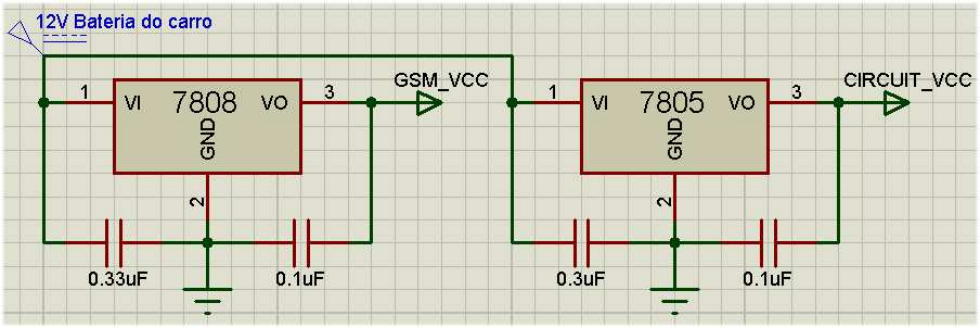

Figura 27 — Integrados 78XX para alimentação do módulo GSM e circuito.

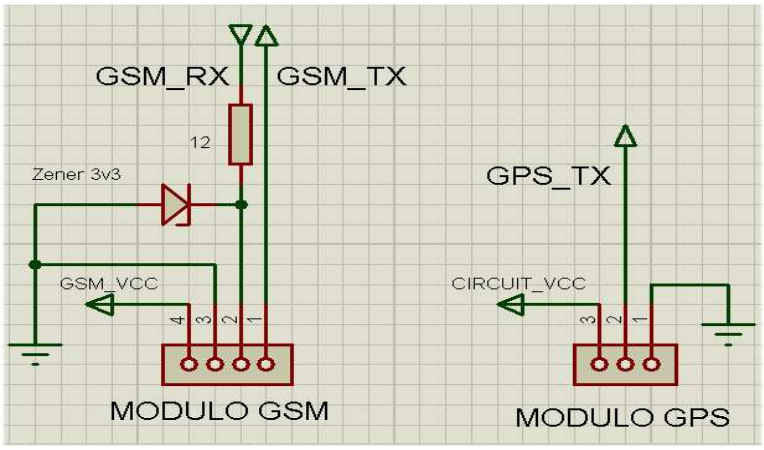

Figura 28 — Configuração de pinos dos módulos GSM e GPS

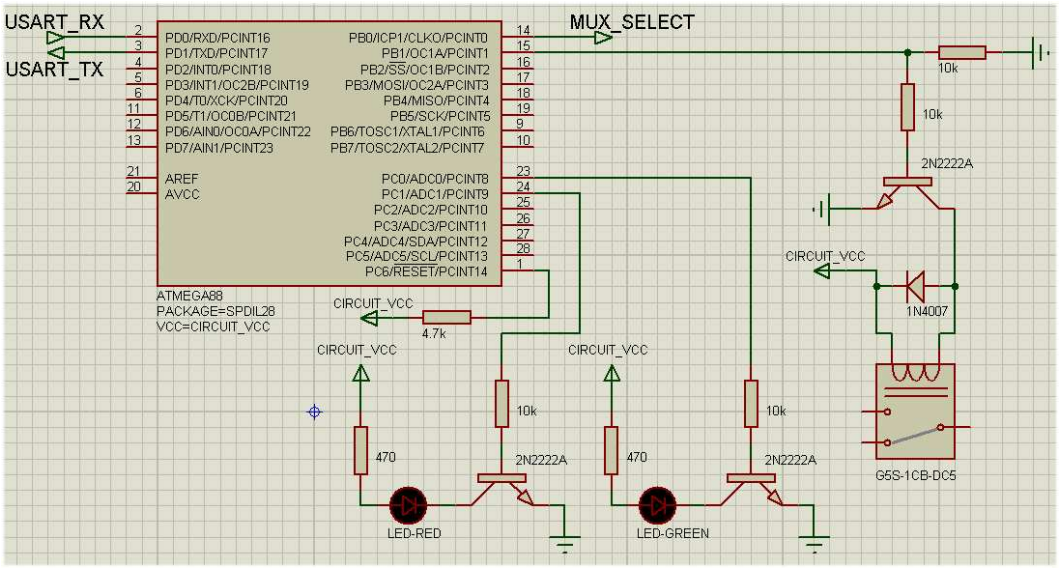

Figura 29 — Microcontrolador e periféricos

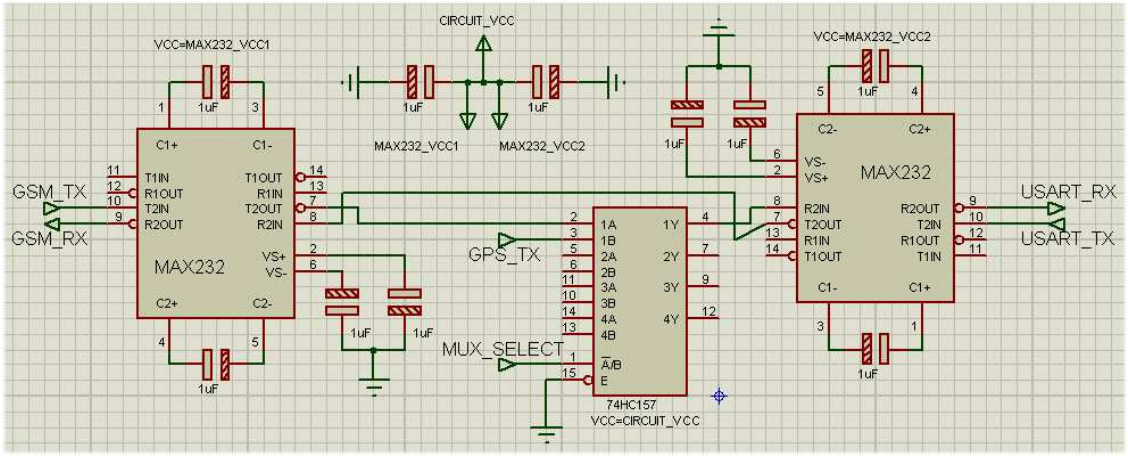

Figura 30 — MAX232 e multiplexador

Após ter sido esquematizado o circuito a implementar, foi efectuada a sua montagem em breadboard e testado. Neste momento não é relevante a discussão dos testes, pois não foi abordada a programação implementada no microcontrolador. Na figura seguinte é possível observar a montagem do circuito na breadboard.

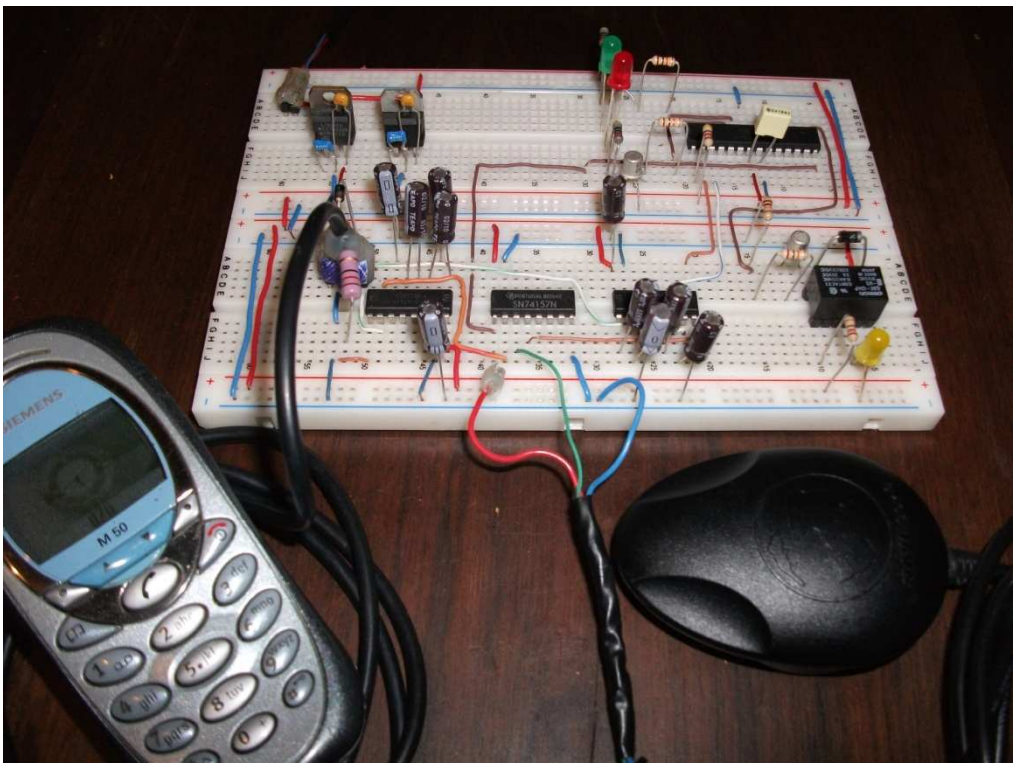

Figura 31 — Montagem em breadboard

Segue-se uma tabela com os componentes eléctricos utilizados no dispositivo:

Tabela 3 — Lista de componentes eléctricos do dispositivo

Componente	Valor	Quantidade
Integrado 7805	-	1
Integrado 7808	-	1
Integrado MAX232	-	2
Integrado 74HC157	-	1
Integrado ATmega88	-	1
Relé OMRON G6E-134P	-	1
Condensadores	0.33 µF	2
	0.1 µF	3
Condensadores Electrolíticos	1 µF - 63V	10
Transistores	2N2222A	3
Resistências	10 KΩ - ¼ W	4

Resistências	4.7 KΩ - ¼ W	1
	470 Ω - ¼ W	2
	12 Ω - 1 W	1
Díodos	1N4007	1
	Zener - 3V3	1
	LED - Vermelho	1
	LED - Verde	1

Foi também projectado o layout pretendido para a implementação em placa de circuito impresso. Terá sido considerado um método de layout laminado de duas faces, em que a continuidade é estabelecida entre os pinos através de linhas de cobre presentes nas duas faces do circuito. ("The printed circuit board (PCB) layout,")

Seguem-se figuras que contêm o layout construído para o dispositivo Anti-Carjacking.

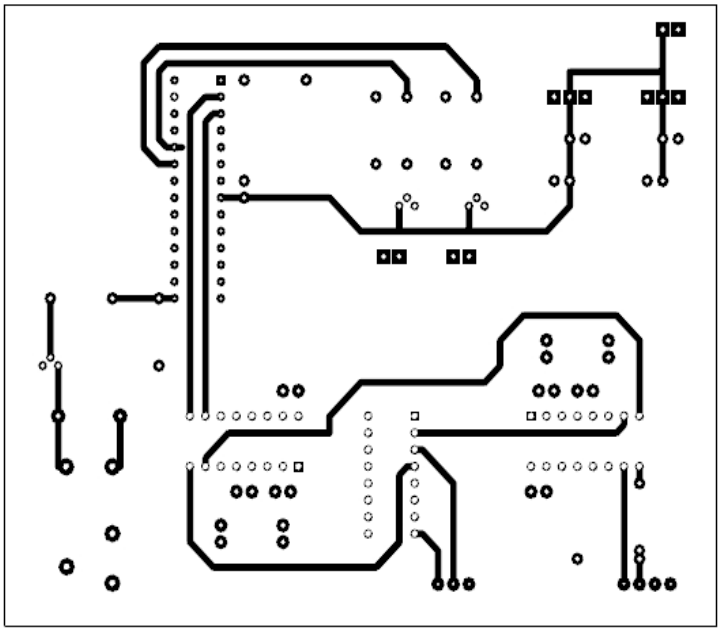

Figura 32 — Layout para a parte de baixo da placa PCB

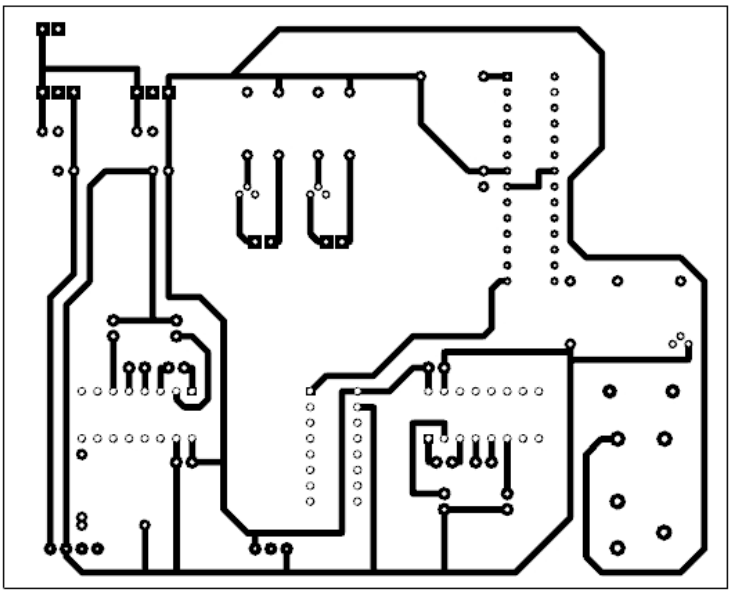

Figura 33 — Layout para a parte de cima da placa PCB

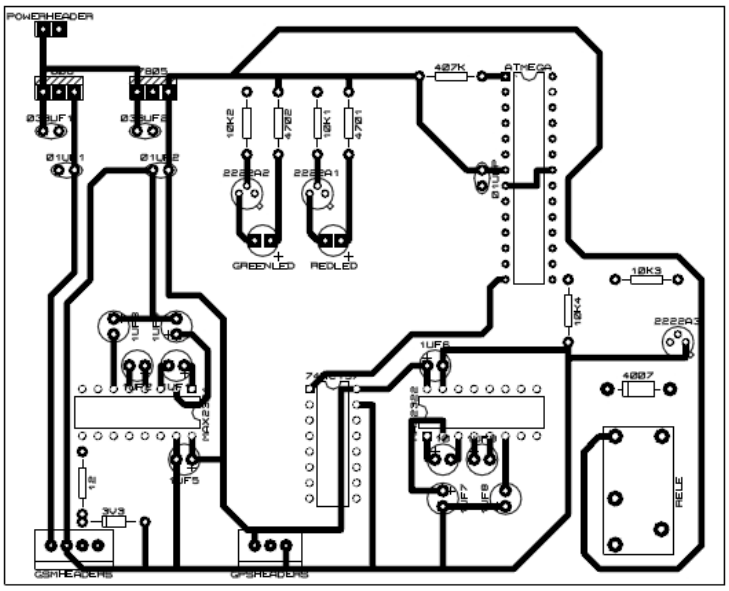

Figura 34 — Layout do topo da placa com as formas dos componentes

O protótipo deste projecto não chegou a ser implementado em placa de circuito impresso. As duas figuras seguintes ilustram uma modelação tridimensional da implementação do circuito em PCB e permitem visualizar o seu aspecto final.

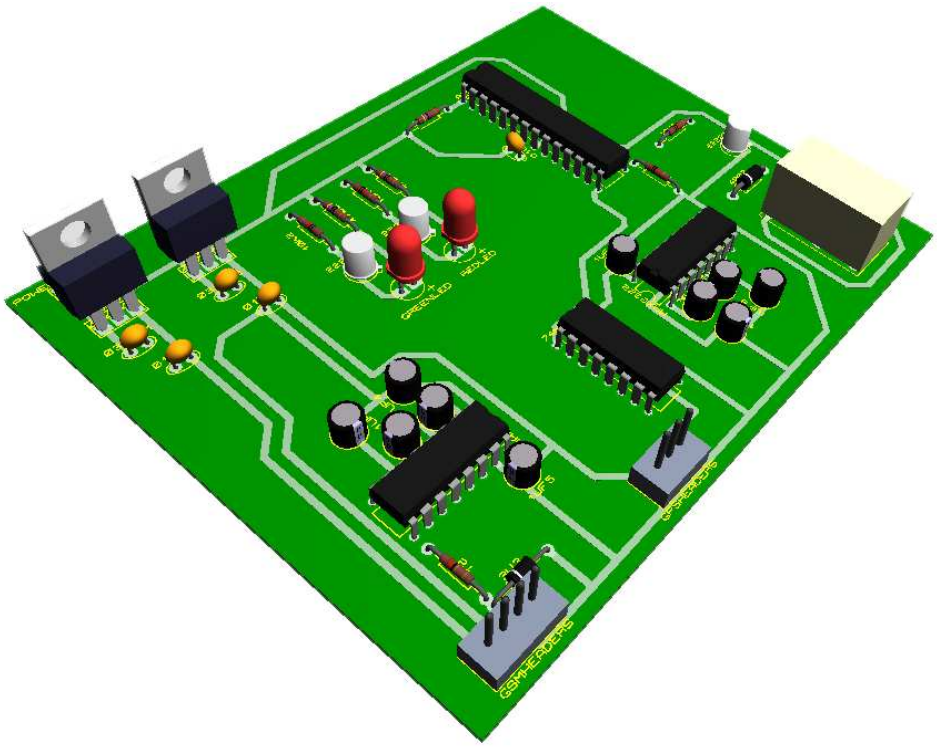

Figura 35 — Ilustração tridimensional do topo do circuito

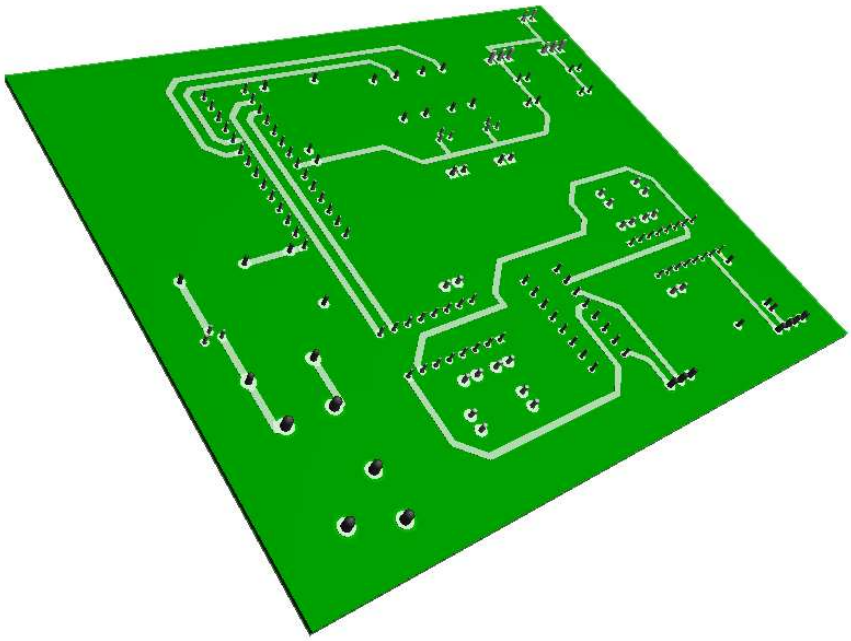

Figura 36 — Ilustração tridimensional do fundo do circuito

Programação

Microcontrolador - Código

A programação do microcontrolador foi efectuada na linguagem C.

A IDE utilizada foi o ATMEL AVR Studio 4 em conjunto com o compilador AVR-GCC, que é um módulo constituinte do projecto open-source WinAVR. O AVR-GCC contém bibliotecas com definições específicas dos microcontroladores ATMEL que são necessárias para programá-los. ("WinAVR,")

Foram construídas 5 classes, as classes gps.c e gsm.c contêm métodos relativos à interacção com cada um dos respectivos módulos do dispositivo, a classe usart.c inclui métodos de configuração e comunicação por série, a classe procedures.c agrega a utilização dos métodos das classes referidas anteriormente e contém os métodos principais do programa, e finalmente, a classe anticarjacking.c é a classe principal (main) do programa, e é responsável pela inicialização do dispositivo, controlo do funcionamento e tratamento das interrupções.

Cada uma das classes anteriores (à excepção da classe anticarjacking.c) contêm um respectivo ficheiro header, e nesse ficheiro são declarados os métodos globais que estarão visíveis e poderão ser utilizados pelas outras classes. Para além desses headers, foi incluído um ficheiro header, em todas as classes, com inclusões das bibliotecas necessárias para controlo do microcontrolador.

Para os métodos públicos mais extensos das diferentes classes serão dispostos fluxogramas ilustrativos aos seus respectivos funcionamentos. Os restantes serão explicados textualmente.

Para a classe usart.c:

- void initUSART(int baudrate): Este método permite atribuir ao registo UBRR0 o valor inteiro pretendido para a baudrate, que conforme calculado anteriormente, será 12. Tembém neste método são efectuadas as configurações da comunicação por série e são habilitados o transmissor e receptor.
- void sendUSART(unsigned char *input): Permite enviar uma string através do transmissor, um caracter de cada vez.
- unsigned char *retrieveUSART(int size, int minimum): Retorna uma string de dados que termine com o caracter <LF>, com tamanho máximo size e tamanho mínimo minimum. Essa string é obtida caracter a caracter através do receptor.
- void sendChar(unsigned char c): Envia o caracter c através do transmissor.

Para a classe gps.c:

- unsigned char *retrieveGPS(void):

Figura 37 — Fluxograma de funcionamento do método retrieveGPS

Para a classe gsm.c:

- void initGSM(void): Envia comandos para inicializar a indicação de novas mensagens pelo módulo GSM através de comunicação em série.
- SMS *retrieveSMS(unsigned char *cmti): Adquire uma string (a mensagem SMS em formato PFU) com o código contido em CMTI retornada pelo telemóvel, seguidamente selecciona o comprimento do número do emissor, o número do emissor e o conteúdo da mensagem. Após descodificar o conteúdo da mensagem retorna uma Struct com o comprimento do número do emissor, emissor e mensagem descodificada, cada um com tipo string.
- void sendSMS(unsigned char *address_length, unsigned char *address, unsigned char *content): Após codificar o content, envia para o modulo GSM o comando para enviar uma mensagem SMS seguido de uma string estruturada no formato PDU que contém a mensagem codificada.

Para a classe procedures.c:

- void startup(void): Efectua a inicialização de portas, módulos e comunicação por série.
- void processRoutine(void):

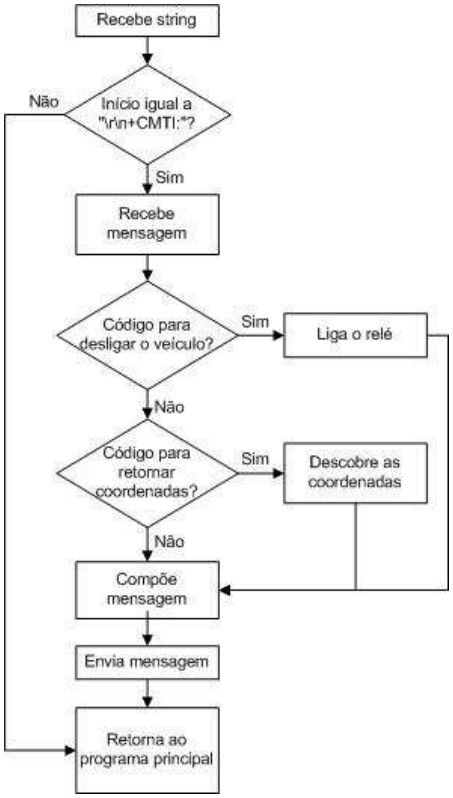

Figura 38 — Fluxograma de funcionamento do método processRoutine

- unsigned char *returnSubStr(int istart, int istop, unsigned char *input): Este método permite retornar uma substring da string input, substring essa que é constituída pelos caracteres apartir do índice istart, até ao índice istop (inclusive).

A classe anticarjacking.c não dispõe de métodos globais por ser a classe que contém o programa principal e cumprir a função de controlo do dispositivo. Na figura seguinte é representado um fluxograma do funcionamento desta classe.

Figura 39 — Fluxograma de funcionamento do microcontrolador

Microcontrolador – Escrita do firmware

Para escrever o código no microcontrolador foi utilizado o dispositivo programador AVR STK500 da Atmel que é controlado através do AVR Studio. O código do programa compilado gera um ficheiro de extensão hex[9] que é escrito directamente na memória flash do microcontrolador. ("AVR Stk500 User Guide," 2003)

A memória flash do microcontrolador é um espaço de memória reservado para as instruções contidas no ficheiro hex. Essas instruções providenciam ao microcontrolador o funcionamento que respeita o intuito do código programado na IDE. ("ATmega88," 2009)

O dispositivo STK500 comunica com o computador através de RS-232 por interveniência da porta série. O AVR Studio permite, através do dispositivo, não só escrever dados na memória flash como também alterar configurações nos fusíveis para, a título de exemplo, selecção da fonte de oscilação ou alterar as definições de reset. ("AVR Stk500 User Guide," 2003)

Na próxima figura encontra-se disposto um esquema dos componentes do STK500.

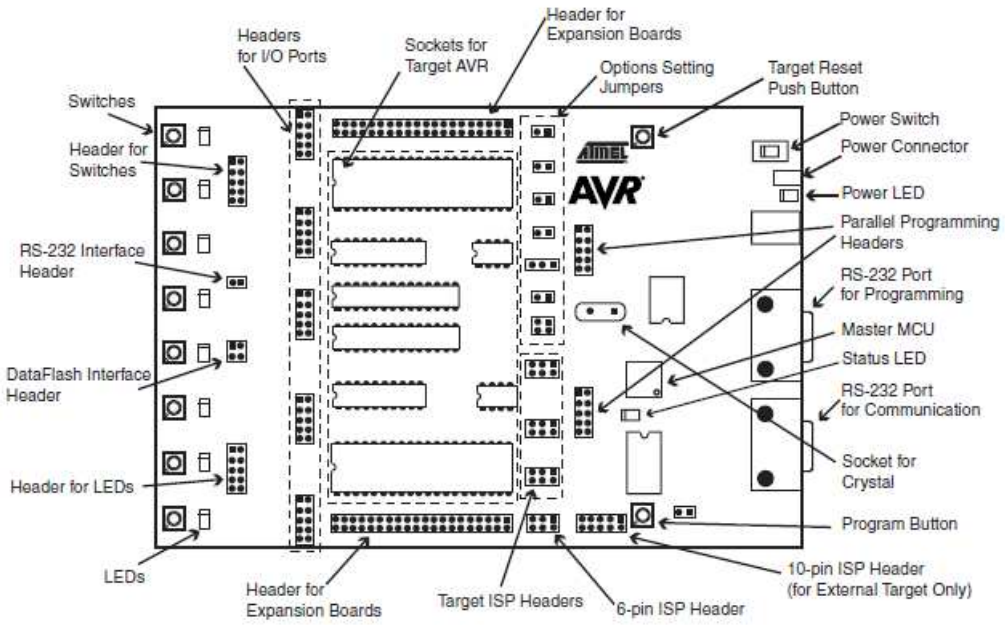

Figura 40 — Esquema de componentes do Avr STK500 ("AVR Stk500 User Guide," 2003)

De entre alguns modos disponíveis para a escrita, o modo escolhido para este projecto foi o ISP. Existem 3 target ISP headers disponíveis. Para a programação no modo ISP, o header de 6 pinos é ligado ao target ISP header correspondente ao socket do dispositivo que se pretende programar. Para o ATmega 88 esse socket é o SCKT3200A2, que corresponde ao header central. ("AVR Stk500 User Guide," 2003)

[9] Contém as instruções em linguagem compreendida pelo microcontrolador.

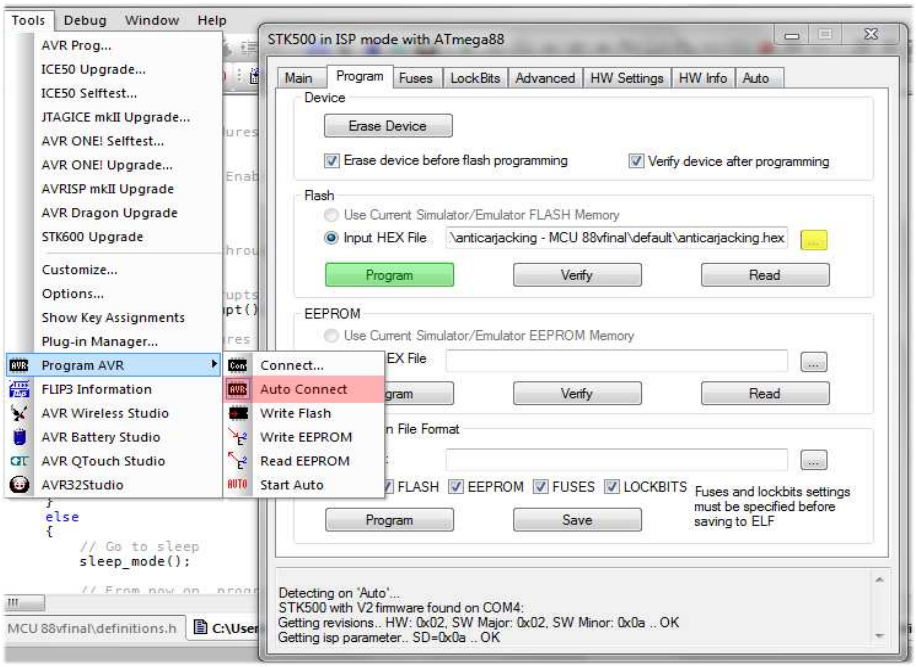

Figura 41 — Programação através do AVR STK500 no AVR Studio 4

Na figura anterior está representado o interface de escrita no microcontrolador utilizado pelo AVR Studio 4. Após o AVR STK500 estar conectado ao computador e ligado, é possível abrir a interface através do controlo que está salientado a vermelho na figura. Após abrir o interface, é necessário estipular o modelo do microcontrolador no separador Main, de seguida utiliza-se o controlo salientado a amarelo para seleccionar o ficheiro com extensão hex. Após seleccionado, clica-se no controlo Program salientado a verde. ("AVR Stk500 User Guide," 2003)

Software de visualização dos mapas

O código foi elaborado na linguagem C# através da IDE Visual Studio 2008.

Primariamente, conhecendo a necessidade de manter um histórico de coordenadas, foi elaborada uma classe Information com um construtor de uma DataTable com as colunas referentes ao número de telefone, data e hora, latitude e longitude. Para permitir a manipulação desta Datatable de acordo com as necessidades do programa, foram criados métodos públicos para as funções de atribuição directa na datatable da classe Information e também retorno da própria datatable. Foram também criadas as funções para adicionar e apagar linhas da datatable. As linhas são excluídas conforme o seu índice.

No que diz respeito à adição de linhas, pretende-se que o histórico respeite uma sequência cronológica. Isso pode ser obtido percorrendo a Datatable após a adição de cada nova linha e alterando a ordem das linhas através da comparação entre os valores da coluna da data e hora. Para esta função utiliza-se um algoritmo Selection Sort, que cada vez que percorre o conjunto das linhas coloca o valor máximo encontrado no fundo da lista. (Bezroukov, 2009)

Durante o programa foram implementados métodos de escrita e leitura dos dados da Datatable em ficheiro xml, para que os dados não sejam voláteis.

A primeira prioridade é estabelecer o interface com o módulo GSM através da porta série e ler as coordenadas. Na biblioteca .NET Framework existe a classe Ports que permite tal abordagem. Para tal é criado um novo objecto do tipo SerialPort, depois é configurado o formato da mensagem e finalmente os métodos Open e Close da classe permitem abrir e fechar, respectivamente, a interface com o dispositivo. A API para esta classe dispõe de métodos de leitura e escrita que serão utilizados para comunicar com o módulo GSM. Os métodos básicos para comunicação encontram-se na classe SerialInterface, que são utilizados pela classe que contém o programa principal em funções compostas. ("SerialPort Class,")

A classe principal do programa contém as variáveis estáticas que serão manipuladas no decorrer do funcionamento do programa. A função mais relevante desta classe é a função que permite retornar uma Datatable com os valores das coordenadas. Nesse método é efectuada a comunicação com o módulo GSM através de comandos AT para leitura de todas as mensagens no modo de texto (AT+CMGL="ALL") e são seleccionados os índices das mensagens que contêm os códigos das coordenadas recebidos pelo dispositivo Anti-Carjacking. Através do comando AT+CMGR=<index> são lidas as mensagens individualmente e seleccionados os valores referentes ao número de telefone do emissor, data e hora, latitude e longitude.

Toda a selecção de conteúdo é efectuada por expressões regulares, através da comparação e extracção dos dados de strings que respeitem a estrutura da expressão. (Goyvaerts, 2009)

A estrutura da mensagem SMS em modo de texto foi abordada anteriormente, agora será relacionada essa estrutura com as expressões regulares utilizadas.

- `\+CMGL:\s(\d+):` Percorre a string e retorna os dígitos encontrados após a palavra +CMGL: seguida de um espaço. É utilizada para retornar o índice das mensagens SMS.
- `^.+"\+(\d{3})(\d{9})".+"(\d{4})/(\d{1,2})/(\d{1,2}),(\d{1,2}):(\d{1,2}):(\d{1,2})\"$:` Percorre a string e retorna os 12 dígitos encontrados após um caracter + (telefone do emissor), seguidamente retorna cada um dos dígitos que compõem a data e a hora, que estão no formato dd/MM/aa,hh:mm:ss.
- `^\.outputGPS\.(N|S)-(\d{2}),(\d{4})\.(E|W)-(\d{2}),(\d{4})\.$:` Retorna de forma independente os dígitos e orientação da latitude e longitude. A string têm de começar por .outputGPS. e conter depois os dados da latitude e longitude conforme representado na expressão.

O controlo do programa reside todo na sua interface gráfica, pois esta contém os botões que permitem invocar as funções implementadas na classe principal do programa. Para o caso da comunicação com o telemóvel, a interface é a disposta na próxima figura.

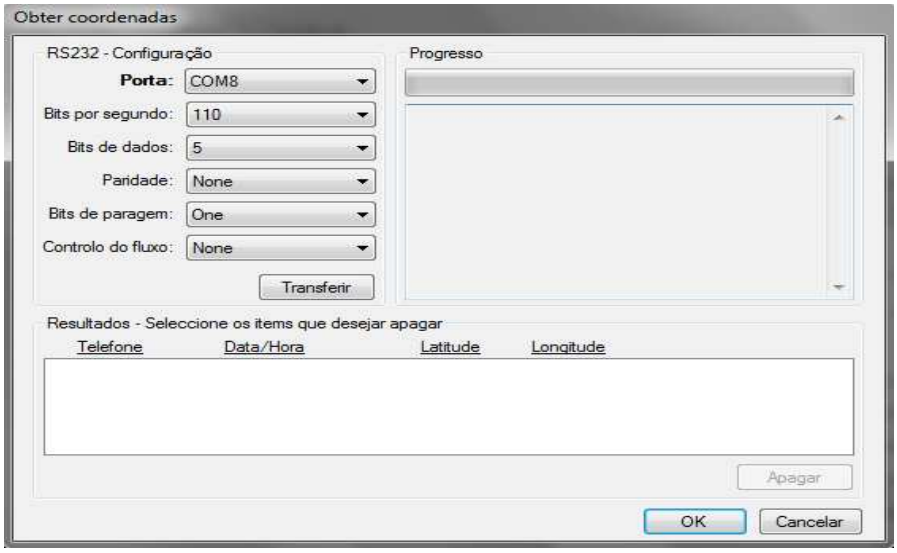

Figura 42 — Interface gráfica do modo de obtenção de coordenadas

Nesta interface é possível seleccionar as definições desejadas para o formato da frame RS-232, e ao clicar no botão Transferir, é executado o método de obtenção das coordenadas que está no programa principal, retornando uma datatable com os valores obtidos. Cada linha da datatable é apresentada na CheckedListBox de acordo com a ordem representada na imagem. O botão Apagar permite apagar as coordenadas seleccionadas que não se pretendem passar para a janela de visualização dos mapas.

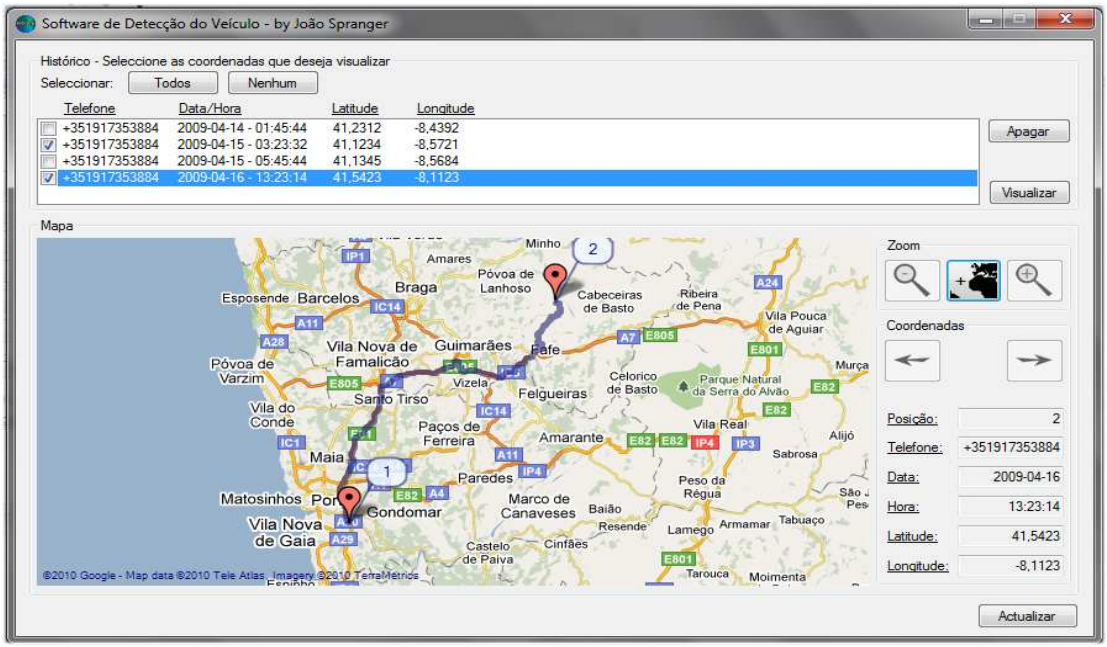

Figura 43 — Interface gráfica do modo de visualização de mapas

Na figura 42 é possível visualizar o botão Actualizar, a sua função é invocar a janela da figura 41 de forma a obter novas coordenadas. Na CheckedListBox em cima é possível observar o formato das coordenadas dispostas. A disponibilidade dos botões Todos, Nenhum Apagar e Visualizar está dependente de eventos consoante a selecção efectuada às coordenadas da lista. O botão Apagar permite remover coordenadas da lista e o controlo Visualizar mostra as coordenadas seleccionadas num mapa e habilita os controlos do mapa presentes à direita na figura 42.

Todo o código de controlo do mapa é efectuado na interface da figura 42. Para poder ser utilizada a API da biblioteca Gmaps, foi necessário referenciar as bibliotecas BSE.Windows.Forms, Gmap.NET.Core e Gmap.NET.WindowsForms. Foi também necessário criar uma classe de controlo parcial da classe GmapControl. Esse controlo foi adicionado à interface da figura 42 e utilizado como um painel para display dos mapas, conforme se pode ver na figura. ("GMap.NET - Great Maps for Windows Forms & Presentation,")

Quando se pretende visualizar as coordenadas, é criada uma lista com a informação dessas coordenadas. A coordenada imediatamente disponível para visualização é a mais recente, sendo possível navegar por entre as coordenadas e alterar o zoom de visualização do mapa. ("GMap.NET - Great Maps for Windows Forms & Presentation,")

Numa forma muito simples, para mostrar o mapa basta atribuir ao método CurrentPosition da API Gmaps as coordenadas pretendidas e invocar o método ReloadMap. Para percorrer as diferentes coordenadas seleccionadas, procede-se à subtracção ou incrementação do índice da lista e repetem-se os dois métodos referidos. ("GMap.NET - Great Maps for Windows Forms & Presentation,")

Existem algumas configurações que podem ser editadas, como o zoom máximo, mínimo e actual. Para aumentar ou diminuir o zoom, procedeu-se à soma e subtracção respectivamente do zoom actual em uma unidade até atingir um valor igual ao valor mínimo ou máximo. ("GMap.NET - Great Maps for Windows Forms & Presentation,")

Foram também adicionados marcadores nos respectivos sítios das coordenadas e traçada uma rota provável efectuada pelo veículo, tal foi obtido através da utilização de objectos do tipo GMapMarker e GMapRoute. É possível obter uma rota entre duas coordenadas através do método GMaps.Instance.GetRouteBetweenPoints. Para calcular a rota provável entre as várias coordenadas executa-se esse método a cada duas coordenadas cronologicamente sequênciais. Os marcadores e rotas encontram-se em layers dispostos por cima da representação do mapa. ("GMap.NET - Great Maps for Windows Forms & Presentation,")

Discussão

Resultados

Actualmente, o dispositivo Anti-carjacking demonstra um funcionamento proveitoso. Porém, por vezes a mensagem SMS que deveria ser retornada ao emissor falha. Desconhece-se a origem do problema pois após falhar a mensagem o dispositivo retorna ao seu funcionamento normal. Presumem-se possíveis erros na troca de dados entre os módulos, mas a situação mais provável é uma falha de alocação de memória no microcontrolador, em que apesar de o programa ser executado normalmente, é impossível enviar a mensagem pois algures faltam dados que não puderam ser escritos na memória.

Se esse for o caso, tal situação poderia, em princípio, ser facilmente resolvida através da utilização de um microcontrolador com maior capacidade de RAM, ou interface com um dispositivo externo que permita alocação provisória dos dados.

Em relação ao software dos mapas, funciona em pleno.

De acordo com o propósito do baixo custo, aparte dos módulos GSM e GPS, estima-se que o custo total do circuito deverá residir entre 20 € e 30 €.

Constrangimentos e abordagens

Ao longo do projecto verificaram-se alguns contratempos que tiveram de ser resolvidos.

Em relação ao circuito, um exemplo disso foi a necessidade de inclusão do integrado 7808, pois a montagem inicial não continha esse circuito.

Quando era efectuada comunicação através de RS-232 com o módulo GSM, este perdia a rede. Admitiu-se que o facto de o módulo não estar a ser alimentado pelo circuito poderia constituir um problema no seu funcionamento, e essa foi a razão original pela qual o integrado 7808 foi incluído no circuito. Efectivamente, o módulo deixou de perder rede.

Também relacionado com esse módulo, durante a execução do programa o funcionamento demonstrou-se proveitoso até ao momento de enviar a mensagem SMS, porém, o envio da mensagem falhava sempre. Após alguma pesquisa descobriu-se que a tensão máxima do pino RX do módulo teria de ser 3V, caso contrário o módulo perderia rede ao comunicar por mensagem SMS. ("cellphone cable and connector wiring scheme," 2008)

A abordagem utilizada para essa situação foi a inclusão de um circuito nivelador de tensão através de um díodo zener e uma resistência, conforme representado na figura 27. Foi calculada a queda de tensão originada pelo díodo Zener e seguidamente calculada a resistência a utilizar apartir da lei de ohm. ("5V and 3V Level Translators," 2004)

$$5V - 3.3V = 1.7V\ (Queda\ de\ tensão)$$

$$V = R.i,\ i = 0{,}4\ (obtido\ através\ de\ medição)$$

$$R = \frac{5}{0.4} = 12{,}5\ \Omega \text{ e } P = V.i = 1.7 \times 0.4 = 0{,}68\ W$$

Foi utilizada uma resistência de 12 Ω com potência de 1 W, e obteve-se uma tensão igual a 2.6V, a diferença não é significativa e a abordagem em questão resolveu o problema.

Em princípio não seria necessário o circuito integrado 7808 a alimentar o módulo GSM apartir deste ponto, mas pela vantagem de manter a bateria carregada o circuito foi mantido.

O desenho original do circuito continha um microcontrolador ATmega8515 com 512 bytes de RAM ao invés do ATmega88. Ambos estes microcontroladores são semelhantes em termos de funcionalidade mas com o ATmega8515, o programa apresentava um comportamento errático na execução. Apesar de a mensagem SMS ser lida, durante o processo o microcontrolador era inexplicavelmente reiniciado.

Foram implementados leds nos pinos do microcontrolador para efeitos de debug[10] conforme o circuito presente na próxima figura:

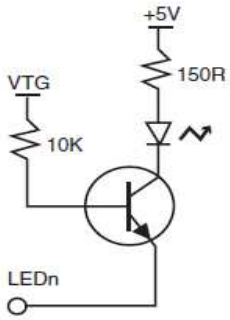

Figura 44 — Circuito de ligação dos LEDs ("AVR Stk500 User Guide," 2003)

Após ter sido efectuado um debug extensivo compreendeu-se que o problema era derivado a fugas de memória no código e falta de RAM por parte do microcontrolador ATmega8515. Após um longo processo de debug ao código, o problema persistiu.

Experimentou-se o microcontrolador ATmega88 (com 1K de RAM) e o código efectuou pelo menos um ciclo completo à partida, demonstrando que o programa estava a funcionar, porém ainda existiam algumas fugas de memória que levaram ao eventual reset ocasional.

Foi adoptada uma estratégia para reduzir a memória RAM ocupada, que consistiu na escrita de variáveis globais constantes na memória flash do programa, liberando-as da RAM.Tal é possível através da biblioteca progmem.h incluída no AVR-GCC. ("WinAVR,")

[10] Procura e reparação de erros de lógica no código do programa.

A memória libertada possibilitou o funcionamento por mais alguns ciclos, acabando por ocorrer overflow da RAM e conduzir a reset. Após mais algum debug as fugas de memória foram extintas e deixou de ocorrer a situação de reset.

Em relação ao software de visualização dos mapas, o único constrangimento ocorrido foi a situação referida acerca das APIs dos provedores, que foi resolvida com a utilização da biblioteca Gmaps, conforme explicado nos conceitos teóricos.

Conclusão e trabalho futuro

No decorrer deste projecto foi desenvolvido o protótipo de um dispositivo Anti-Carjacking e o respectivo software para visualização das coordenadas num mapa. Foi um processo longo e moroso, mas extremamente cativante na medida em que complementou o meu conhecimento em vários aspectos.

Foi possível observar a nível prático de que forma a utilização de protocolos standard potencia o funcionamento de sistemas distribuídos, especialmente em implementações de baixo nível onde é possível verificar todas as mensagens trocadas entre os dispositivos.

Os microcontroladores possuem uma memória RAM extremamente limitada, pelo que deve ser considerada uma abordagem que evite o uso excessivo da memória. Aparte desse constrangimento são dispositivos muito versáteis que permitem efectuar operações complexas e controlar um circuito conforme desejado. A possibilidade de estabelecer comunicação com outros dispositivos em baixo nível providencia uma excelente estratégia para aumentar cada vez mais a portabilidade do circuito, reduzindo o seu tamanho

A utilização, no Visual Studio, de bibliotecas de sistema capazes de interagir directamente com o hardware é também muito interessante porque dá relevo ao hardware enquanto objecto de programação. É uma perspectiva agradável ter a noção que o programador não está limitado ao software quando programa em linguagens de alto nível, utilizando APIs com métodos relativamente simples de compreender e utilizar.

De uma forma geral, foi uma experiência interessante e reveladora.

Em termos de trabalho futuro, seria considerável encontrar uma solução para a situação das mensagens que não são retornadas pelo microcontrolador, implementar o envio de mensagens SMS para o microcontrolador através do próprio programa de visualização dos mapas e guardar, de modo não volátil, as preferências referentes à frame RS-232 no programa.

Poderiam também ser incluídos parâmetros adicionais no funcionamento do microcontrolador, tal como acionar um alarme e trancar o carro quando se desliga a ignição.

Bibliografia

5V and 3V Level Translators. (2004). Retrieved 21-08, 2010, from http://www.daycounter.com/Circuits/Level-Translators/Level-Translators.phtml

74HC/HCT157 Quad 2-input Multiplexer [Electronic. (1990). Version]. Retrieved 28-05-2009, from http://www.nxp.com/documents/data_sheet/74HC_HCT157_CNV.pdf

AT command Set: Siemens Cellular Engines [Electronic. (2002). Version]. Retrieved 23-06-2009,

ATmega88 [Electronic. (2009). Version], from **http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf**

AVR Stk500 User Guide [Electronic. (2003). Version]. Retrieved 28-01-2009, from http://www.atmel.com/dyn/resources/prod_documents/doc1925.pdf

Betke, K. (2000). The NMEA 0183 Protocol [Electronic Version]. Retrieved May, from http://www.cs.put.poznan.pl/wswitala/download/pdf/NMEAdescription.pdf

Bezroukov, N. (2009). Selection Sort. from http://www.softpanorama.org/Algorithms/Sorting/selection_sort.shtml

C45/M50/MT50 Level2.5e Repair Document v1.0 [Electronic. (2002). Version], from http://www.eserviceinfo.com/download.php?fileid=7624

cellphone cable and connector wiring scheme. (2008). from http://pinouts.ru/CellularPhonesCables/siemens_cable_pinout.shtml

Crisp, J. (2004). *Introduction to Microprocessors and Microcontrollers* (2 ed.): Elsevier.

Ekkebus, S.-P. Javascript PDU Converter. from http://www.rednaxela.net/pdu.php

Fundamentals of RS-232 Serial Communications [Electronic. (1998). Version], from http://www.lammertbies.nl/download/dallas-appl-83.pdf

GeoTrack. from http://www.geo24.pt/upload/index.php?act=viewProd&productId=52

GMap.NET - Great Maps for Windows Forms & Presentation. from http://greatmaps.codeplex.com/

Goyvaerts, J. (2009). Regular-Expressions.info. from http://www.regular-expressions.info/index.html

GSM 03.38 [Electronic. Version]. Retrieved 02-06-2009, from http://www.mobilecity.cz/doc/GSM_03.38_5.3.0.pdf

GSM 03.40 [Electronic. Version], from http://ftp.3gpp.org/ftp/specs/2000-09/Ph1/03_serie/0340-370.PDF

GSM 07.05 [Electronic. (1997). Version], from http://pda.etsi.org/exchangefolder/ets_300585e05p.pdf

Hill, G. (2008). *The Cable and Telecommunications Professionals' Reference* (Vol. 2): Elsevier Inc.

iol.pt, R. (2008). Carjacking aumentou 70%, mas vai baixar [Electronic Version]. *iol.pt*, from http://diario.iol.pt/sociedade/carjacking-psp-gnr-administracao-interna-ministerio-prevencao/972750-4071.html

IrDA and RS-232: A Match Made in Silicon [Electronic. (2004). Version], from http://pdfserv.maxim-ic.com/en/an/AN3024.pdf

L7800 Series - Positive Voltage Regulators [Electronic. (2003). Version], from http://www.datasheetcatalog.org/datasheet/stmicroelectronics/2143.pdf

Matic, N. (2000). *The PIC Microcontroller Book*. Retrieved 09-10-2008, from http://elektra-ku.lt/attachments/File/Studentams/PIC_Microcontrollers_for_Beginners.pdf

MAX232, MAX232I Dual Eia-232 Drivers/Receivers [Electronic. (2002). Version]. Retrieved 20-05-2009, from http://www.datasheetcatalog.org/datasheet/texasinstruments/max232.pdf

NMEA Reference Manual [Electronic. (2008). Version]. Retrieved 04-04-2009, from http://www.usglobalsat.com/downloads/NMEA_commands.pdf

Ofria, C. (2006). A short course on charging systems. from http://www.familycar.com/Classroom/charging.htm

PCB Relay G6E [Electronic. Version]. Retrieved 13-02-2009, from http://www.omron.com/ecb/products/pdf/en-g6e.pdf

The printed circuit board (PCB) layout. from http://airborn.com.au/method/layout.html

Product Specification RS232 GPS Receiver NL-303P [Electronic. Version]. Retrieved 06-05-2009, from http://ezkits.illumicon.nl/gps/NL-303P-specification%20ver1.04i.pdf

RS232 Specifications and Standard. (2010). from http://www.lammertbies.nl/comm/info/RS-232_specs.html

RS-232: Serial Ports. (2002). from http://www.lavalink.com/dev/fileadmin/dos/support/white_papers/rs_232_serial_ports.pdf

SerialPort Class. from http://msdn.microsoft.com/en-us/library/30swa673.aspx

Sistema de Localização anti-Carjacking NV-Auto e NV-Moto. from http://www.lojagps.com/catalogo/product_info.php?cPath=26_34&products_id=290

Strangio, C. E. (2006). The RS232 Standard [Electronic Version], from http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

Surtell, T. (1998). Relays. from http://www.eleinmec.com/article.asp?24

WinAVR. from http://winavr.sourceforge.net/index.html

Wireless SafeCar. from http://www.empresas.vodafone.pt/solucoesmobilidade/solucoesparceria/wirelesssafecar.htm

Anexos

João Paulo Pigarro Mesquita Spranger

Datasheet 7808/7805

João Paulo Pigarro Mesquita Spranger

Datasheet MAX232

João Paulo Pigarro Mesquita Spranger

Datasheet 74157

João Paulo Pigarro Mesquita Spranger

Datasheet Relé G6E-134P

João Paulo Pigarro Mesquita Spranger

Datasheet ATmega88

Licenciatura em Informática – Ramo Industrial

João Paulo Pigarro Mesquita Spranger

User Guide STK500

Código-Fonte - Microcontrolador

Anticarjacking.c

```
// Include Project Libraries
#include "definitions.h"
#include "procedures.h"

// Volatile global variable interchageable between Main Program & ISR
// (1 if data waiting to be read at USART - Rx, 0 if not)
volatile uint8_t rxflag = 0;

int main(void)
{
	// Perform startup procedures
	startup();
	__enable_interrupt(); // Enable Global Interrupts

	// Loop forever
	for (;;)
	{
		// If data received through USART - Rx
		if (rxflag == 1)
		{
			// Disable interrupts
			__disable_interrupt();

			// Manage procedures
			processRoutine();

			// Update rxflag variable
			rxflag = 0;

			// Enable RXCIE interruption
			UCSR0B ^= 0x80;

			// Enable interrupts
			__enable_interrupt();
		}
		else
		{
			// Go to sleep
			sleep_mode();

			// From now on, program will be oriented to interrupt @ RXC Flag bit
		}
	}
	return 0;
```

```
}

/* --------------------------- INTERRUPT VECTOR --------------------------- */
ISR(USART_RX_vect)
{
	// Disable Global Interrupts
	__disable_interrupt();

	// Update rxflag variable
	rxflag = 1;

	// Disable RXCIE interruption
	UCSR0B ^= 0x80;

	// Enable Global Interrupts
	__enable_interrupt();
}
```

Usart.c

```
// Include Project Libraries
#include "../definitions.h"
#include "usart.h"

/* ------------------------------- PROTOTYPES ------------------------------- */
unsigned char getChar(void);
/* -------------------------------------------------------------------------- */

/* ---------------------------- PUBLIC METHODS ---------------------------- */
// Method to set Baud Rate
void initUSART(int baudrate)
{
	// Set baud rate
	UBRR0H = (uint8_t)(baudrate>>8);
	UBRR0L = (uint8_t)baudrate;

	// Enable receiver & transmitter
	UCSR0B |= (1<<RXEN0)|(1<<TXEN0);
}

// Method to send string (char *) through transmitter
void sendUSART(unsigned char *input)
{
	uint8_t i;
	for (i = 0; i < strlen(input); i++) sendChar(input[i]);
}
```

```
// Method to retrieve string (char *) from receiver
unsigned char *retrieveUSART(int size, int minimum)
{
	unsigned char *string = calloc(size, sizeof(unsigned char));

	unsigned char test = FALSE;
	int i;
	while (test == FALSE)
	{
		string[i] = getChar();
		if ((string[i] == '\n') & ((i + 1) > minimum))
		{
			test = TRUE;
			string[i + 1] = '\0';
		}
		i++;
	}
	return string;
}

// Method to send char through USART
void sendChar(unsigned char c)
{
   while (!(UCSR0A & (1<<UDRE0))); // While UDR not empty, do nothing
	UDR0 = c; // Send char
}

/* ------------------------------------------------------------------------ */

/* ---------------------------- PRIVATE METHODS ---------------------------- */
// Method to retrieve char through USART
unsigned char getChar()
{
	while (!(UCSR0A & (1<<RXC0))); // While RXC == 0, do nothing
   return UDR0; // Get char
}
```

Gps.c

```
// Include Project Libraries
#include "../definitions.h"
#include "../procedures.h"
#include "../usart/usart.h"
#include "gps.h"
```

```
// Global variables
const unsigned char gpgga[] = "$GPGGA";

/* ------------------------------ PROTOTYPES ------------------------------ */
void toggleInput(void);
/* ------------------------------------------------------------------------ */

/* ----------------------------- PUBLIC METHODS ----------------------------- */
unsigned char *retrieveGPS()
{
	unsigned char *gps_string = calloc(21, sizeof(unsigned char));

	uint8_t test = FALSE;
	while (test == FALSE)
	{
		toggleInput();
		_delay_ms(5);
		unsigned char *gps_string_temp = retrieveUSART(100, 0);
		toggleInput();
		_delay_ms(5);

		if (strlen(gps_string_temp) > 38)
		{
			unsigned char *gps_test = returnSubStr(0, 5, gps_string_temp);

			if ((strcmp(gps_test, gpgga) == 0))
			{
				unsigned char *gps_temp = calloc(21, sizeof(unsigned char));

				gps_temp[0] = gps_string_temp[28];
				gps_temp[1] = '-';
				gps_temp[2] = gps_string_temp[18];
				gps_temp[3] = gps_string_temp[19];
				gps_temp[4] = ',';
				gps_temp[5] = gps_string_temp[20];
				gps_temp[6] = gps_string_temp[21];
				gps_temp[7] = gps_string_temp[23];
				gps_temp[8] = gps_string_temp[24];
				gps_temp[9] = '.';
				gps_temp[10] = gps_string_temp[41];
				gps_temp[11] = '-';
				gps_temp[12] = gps_string_temp[31];
				gps_temp[13] = gps_string_temp[32];
				gps_temp[14] = ',';
```

```
                                gps_temp[15] = gps_string_temp[33];
                                gps_temp[16] = gps_string_temp[34];
                                gps_temp[17] = gps_string_temp[36];
                                gps_temp[18] = gps_string_temp[37];
                                gps_temp[19] = '.';
                                gps_temp[20] = '\0';

                                strcpy(gps_string, gps_temp);
                                free(gps_temp);
                                test = TRUE;
                        }
                        free(gps_test);
                }
                free(gps_string_temp);
        }
        return gps_string;
}
/* ------------------------------------------------------------------------ */

void toggleInput() { PORTB ^= 0x01; }
```

Gsm.c

```
// Include Project Libraries
#include "../definitions.h"
#include "../procedures.h"
#include "../usart/usart.h"
#include "gsm.h"

// Program memory global variables
const unsigned char at[] PROGMEM = "AT\r\n";
const unsigned char atcmgf[] PROGMEM = "AT+CMGF=0\r\n";
const unsigned char atcnmi[] PROGMEM = "AT+CNMI=1,1,0,0,1\r\n";

const unsigned char atcmgr[] PROGMEM = "AT+CMGR=";
const unsigned char atcmgs[] PROGMEM = "AT+CMGS=";

const unsigned char smsbeginning[] PROGMEM = "000100";
const unsigned char numbertype[] PROGMEM = "91";
const unsigned char smsmiddle[] PROGMEM = "0000";

const unsigned char crlf[] PROGMEM = "\r\n";

const unsigned char error[] PROGMEM = "ERROR\r\n";
```

```
PGM_P gsmstaticdata[] PROGMEM =
{       at,                             // 0
        atcmgf,                 // 1
        atcnmi,                 // 2
        atcmgr,                 // 3
        atcmgs,                 // 4
        smsbeginning,   // 5
        numbertype,             // 6
        smsmiddle,              // 7
        crlf,                   // 8
        error                   // 9
};

/* ------------------------------ PROTOTYPES ----------------------------- */
void sendMODEM(unsigned char *);
unsigned char *decodePDU(unsigned char *, int);
unsigned char returnHexfromChar(unsigned char *);
unsigned char assignChar(unsigned char, unsigned char, uint8_t);
unsigned char *convert8to7bit(unsigned char *, int);
unsigned char *encodePDU(unsigned char *);
unsigned char returnCharfromHex(unsigned char, int);
unsigned char *convert7to8bit(unsigned char *);
/* ----------------------------------------------------------------------- */

/* --------------------------- PUBLIC FUNCTIONS --------------------------- */
void initGSM()
// Prepare GSM Modem to send RS232 message when SMS arrives
{
        // Send command to test if GSM modem is fully functional ("AT\r\n")
        unsigned char command1[strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[0])))];
        strcpy_P(command1, (PGM_P)pgm_read_word(&(gsmstaticdata[0])));
        sendMODEM(command1);

        // Send command to set messages on PDU format ("AT+CMGF=0\r\n")
        unsigned char command2[strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[1])))];
        strcpy_P(command2, (PGM_P)pgm_read_word(&(gsmstaticdata[1])));
        sendMODEM(command2);

        // Send command to make phone alert on new sms received ("AT+CNMI=1,1,0,0,1\r\n")
        unsigned char command3[strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[2])))];
        strcpy_P(command3, (PGM_P)pgm_read_word(&(gsmstaticdata[2])));
        sendMODEM(command3);
}

SMS *retrieveSMS(unsigned char *cmti)
```

```
// Retrieve the SMS content
{
	// Retrieve sms index number
	unsigned char *index = returnSubStr(14, strlen(cmti) - 3, cmti);

	// Send code (with index) to retrieve sms
	unsigned char *cmgr = calloc((10 + strlen(index)) + 1, sizeof(unsigned char));
	strcpy_P(cmgr, (PGM_P)pgm_read_word(&(gsmstaticdata[3])));
	strcat(cmgr, index);
	strcat_P(cmgr, (PGM_P)pgm_read_word(&(gsmstaticdata[8])));
	sendUSART(cmgr);

	// Read echo (to clear UDR receive buffer of echo) = "AT+CMGR=code\r\r\n"
	// Size = strlen(cmgr) + 1
	free(retrieveUSART((strlen(cmgr) + 1) + 1, 0));
	free(cmgr);
	free(index);

	// Retrieve sms message ----------------------------------------------------
	// Receive (header) -> +CMGR: 0,,20\r\n
	free(retrieveUSART(15, 0));

	//IMPORTANT - CONTENT OF SMS MESSAGE
	unsigned char *pdu_received = retrieveUSART(100, 0);

	// Minimum = (\r\n)OK\r\n -> 2 char
	free(retrieveUSART(10, 2));
	// ------------------------------------------------------------------------

	// DECODE SMS --------------------------------------------------------------
	// SMSC INFORMATION HEADER
	// Acquire SMSC number length (in pseudo-octets represented by 2 char)
	unsigned char *smsc_length = returnSubStr(0, 1, pdu_received); // get substring with selection
	int smsc_nbr_length = returnHexfromChar(smsc_length) * 2;// get int value with number of chars
	free(smsc_length); // Discard information

	// PDU-String --------------------
	// Get number of chars representing the emmiter phone number
	unsigned char *emmiter_length = returnSubStr(4 + smsc_nbr_length, (6 + smsc_nbr_length) - 1, pdu_received); // get substring with selection
	int emmiter_nbr_length = returnHexfromChar(emmiter_length);// get int value with number of chars
```

```
	if (emmiter_nbr_length % 2 != 0) emmiter_nbr_length++; // Case length != pair, char before last is 'F' (just for filling)

	// Get emmiter phone number (coded) into unsigned char *
	unsigned char *emmiter = returnSubStr(8 + smsc_nbr_length, (8 + smsc_nbr_length + emmiter_nbr_length) - 1, pdu_received); // get substring with selection

	// SMS (decoded) length (number of chars)
	unsigned char *sms_length = returnSubStr(26 + smsc_nbr_length + emmiter_nbr_length, 27 + smsc_nbr_length + emmiter_nbr_length, pdu_received);
	int sms_nbr_length = returnHexfromChar(sms_length);
	free(sms_length);

	// Get PDU sms (coded)
	unsigned char *sms_coded = returnSubStr(28 + smsc_nbr_length + emmiter_nbr_length, strlen(pdu_received) - 3, pdu_received);

	// Get PDU sms after decoding
	unsigned char *sms_decoded = decodePDU(sms_coded, sms_nbr_length + 1);

	free(sms_coded);
	free(pdu_received);

	SMS *sms = calloc(1, sizeof(SMS));
	sms -> address_length = calloc(strlen(emmiter_length) + 1, sizeof(unsigned char));
	strcpy(sms -> address_length, emmiter_length);
	sms -> address = calloc(strlen(emmiter) + 1, sizeof(unsigned char));
	strcpy(sms -> address, emmiter);
	sms -> content = calloc(strlen(sms_decoded) + 1, sizeof(unsigned char));
	strcpy(sms -> content, sms_decoded);

	free(emmiter_length);
	free(emmiter);
	free(sms_decoded);

	return sms;
}

void sendSMS(unsigned char *address_length, unsigned char *address, unsigned char *content)
// Send SMS with response
{
	unsigned char *content_size = calloc(3, sizeof(unsigned char));
	sprintf(content_size, "%02X", strlen(content));
```

```
        unsigned char *encoded_content = encodePDU(content);
        unsigned char *temp = returnSubStr(0, 12, content);
        if ((strcmp(temp, ".outputGPS.N-") == 0) || (strcmp(temp, ".outputGPS.S-") == 0))
strcat(encoded_content, "00"); // encodePDU() fix

// ------------------------------------------------------------------------------------
        unsigned char *startofsms =
calloc(strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[4]))) + 1, sizeof(unsigned char));
        strcpy_P(startofsms, (PGM_P)pgm_read_word(&(gsmstaticdata[4])));

        sendUSART(startofsms);
        free(startofsms);

        int octets_int =
        strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[5])))     +
        strlen(address_length) +
        strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[6]))) +
        strlen(address) +
        strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[7]))) +
        strlen(content_size) +
        strlen(encoded_content);

        unsigned char *octets = calloc(3, sizeof(unsigned char));
        sprintf(octets, "%d", (octets_int/2) - 1);

        sendUSART(octets);
        free(octets);

        unsigned char *endofstart =
calloc(strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[8]))) + 1, sizeof(unsigned char));
        strcpy_P(endofstart, (PGM_P)pgm_read_word(&(gsmstaticdata[8])));

        sendUSART(endofstart);
        free(endofstart);

        free(retrieveUSART(14, 0)); // AT+CMGS=(Octets -> 2 chars) + \r\r\n

  unsigned char *begin = calloc(strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[5]))) + 1,
sizeof(unsigned char));
        strcpy_P(begin, (PGM_P)pgm_read_word(&(gsmstaticdata[5])));

        sendUSART(begin);
        free(begin);

        sendUSART(address_length);
```

```
  unsigned char *tp = calloc(strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[6]))) + 1, sizeof(unsigned char));
        strcpy_P(tp, (PGM_P)pgm_read_word(&(gsmstaticdata[6])));

        sendUSART(tp);
        free(tp);

        sendUSART(address);

  unsigned char *mid = calloc(strlen_P((PGM_P)pgm_read_word(&(gsmstaticdata[7]))) + 1, sizeof(unsigned char));
        strcpy_P(mid, (PGM_P)pgm_read_word(&(gsmstaticdata[7])));

        sendUSART(mid);
        free(mid);

        sendUSART(content_size);
        free(content_size);

        sendUSART(encoded_content);
        free(encoded_content);

        // Enviar char 0x1A       para enviar a sms.
        sendChar(0x1A);

        // RETRIEVE USART RX ECHO FROM COMMAND SENT + RESPONSE
        free(retrieveUSART(100, 0)); // > PDU(ctrl<z>) + \r\n
        unsigned char *status = retrieveUSART(20,0); // +CMGS: (INDEX)\r\n or ERROR\r\n
        if (strcmp_P(status,(PGM_P)pgm_read_word(&(gsmstaticdata[9]))) == 0) ;
        else free(retrieveUSART(7, 2)); // \r\nOK\r\n
        free(status);
}
/* ------------------------------------------------------------------------ */

/* --------------------------- PRIVATE FUNCTIONS --------------------------- */
void sendMODEM(unsigned char *cmd)
{
        // Maximum Response Size = strlen(cmd) + 8 [ERROR\r\r\n]
        unsigned char *response = calloc((strlen(cmd) + 9) + 1, sizeof(unsigned char));

        uint8_t test = FALSE;
        while (test == FALSE)
        {
                sendUSART(cmd);
```

```
		response = retrieveUSART((strlen(cmd) + 9) + 1, (strlen(cmd) + 1)); // First response is echo
		uint8_t i;
		for (i = (strlen(cmd)); i < (strlen(response) - 2); i++) // VERIFICAR SE ALTERAÇÃO ESTÁ CORRECTA!!!!!!!!!
		{
			if ((response[i] == 'O') & (response[i + 1] == 'K')) test = TRUE;
			// else repeat
		}
	}
	free(response);
}

// Método privado para concatenar hexadecimal
unsigned char assignChar(unsigned char value, unsigned char hex, unsigned char test)
{
	if (test == FALSE) value = hex;
	else
	{
		value = value << 4;
		value = value ^ hex;
	}
	return value;
}

// Método para converter para hex partindo de array de char com hex
unsigned char returnHexfromChar(unsigned char *input)
{
	uint8_t test = FALSE;
	unsigned char c = 0;

	uint8_t i;
	for (i = 0; i < strlen(input); i++)
	{
		switch(input[i])
		{
			case '0':
				c = assignChar(c, 0x00, test);
				test = TRUE;
				break;
			case '1':
				c = assignChar(c, 0x01, test);
				test = TRUE;
				break;
			case '2':
```

```
            c = assignChar(c, 0x02, test);
            test = TRUE;
            break;
        case '3':
            c = assignChar(c, 0x03, test);
            test = TRUE;
            break;
        case '4':
            c = assignChar(c, 0x04, test);
            test = TRUE;
            break;
        case '5':
            c = assignChar(c, 0x05, test);
            test = TRUE;
            break;
        case '6':
            c = assignChar(c, 0x06, test);
            test = TRUE;
            break;
        case '7':
            c = assignChar(c, 0x07, test);
            test = TRUE;
            break;
        case '8':
            c = assignChar(c, 0x08, test);
            test = TRUE;
            break;
        case '9':
            c = assignChar(c, 0x09, test);
            test = TRUE;
            break;
        case 'A':
            c = assignChar(c, 0x0A, test);
            test = TRUE;
            break;
        case 'B':
            c = assignChar(c, 0x0B, test);
            test = TRUE;
            break;
        case 'C':
            c = assignChar(c, 0x0C, test);
            test = TRUE;
            break;
        case 'D':
            c = assignChar(c, 0x0D, test);
```

```
                                test = TRUE;
                                break;
                        case 'E':
                                c = assignChar(c, 0x0E, test);
                                test = TRUE;
                                break;
                        case 'F':
                                c = assignChar(c, 0x0F, test);
                                test = TRUE;
                                break;
                }
        }
        return c;
}

unsigned char *convert8to7bit(unsigned char *input, int strsize)
{
        unsigned char *converted = calloc(strsize, sizeof(unsigned char));

        unsigned char hibits = 0;
        unsigned char lobits = 0;
        unsigned char bits = 7;
        unsigned char rest = 0;

        uint8_t counter = 0;
        uint8_t i;
        for (i = 0; i < strlen(input); i++)
        {
                if (bits == 0)
                {
                        converted[counter] = hibits;
                        counter++;
                        bits = 7;
                        rest = 0;
                        hibits = 0;
                }
                lobits = input[i] << (byte - bits);
                lobits = lobits >> (byte - bits - rest);
                converted[counter] = lobits | hibits;
                counter++;
                hibits = input[i] >> bits;
                bits--;
                rest++;
        }
        converted[counter] = '\0';
```

```
	return converted;
}

// Método para descodificar a mensagem SMS
unsigned char *decodePDU(unsigned char *input, int strsize)
{
	unsigned char *smsbytes = calloc(((strlen(input)/2) + 1), sizeof(unsigned char));
	unsigned char *temp = calloc(3, sizeof(unsigned char));

	uint8_t counter = 0;
	uint8_t i;
	for (i = 0; i < strlen(input); i++)
	{
		temp[0] = input[i];
		temp[1] = input[i + 1];
		temp[2] = '\0';
		smsbytes[counter] = returnHexfromChar(temp);
		counter++;
		i++;
	}
	smsbytes[counter] = '\0';
	free(temp);

	unsigned char *decoded = convert8to7bit(smsbytes, strsize);

	free(smsbytes);

	return decoded;
}

// Método para converter para array de char com hex partindo de int
unsigned char returnCharfromHex(unsigned char input, int index)
{
	unsigned char *temp = calloc(3, sizeof(unsigned char));
	sprintf(temp, "%02X", input);

   unsigned char c = temp[index];

	free(temp);

	return c;
}

unsigned char *convert7to8bit(unsigned char *input)
```

```
{
        unsigned char *converted = calloc(60, sizeof(unsigned char));

        unsigned char hibits = 0;
        unsigned char lobits = 0;
        unsigned char bits = 0;

        uint8_t counter = 0;
        uint8_t i;
        for (i = 0; i < strlen(input); i++)
        {
                if (bits == 7) bits = 0;
                else
                {
      lobits = input[i] >> bits;
                        hibits = input[i + 1] << (byte - bits - 1);
                        converted[counter] = lobits | hibits;
                        counter++;
                        bits++;
                }
        }
        converted[counter] = '\0';
        return converted;
}

// Método para codificar a mensagem SMS
unsigned char *encodePDU(unsigned char *input)
{
        unsigned char *converted = convert7to8bit(input);
        unsigned char *result = calloc((strlen(converted) * 2 ) + 1, sizeof(unsigned char));

        uint8_t counter = 0;
        uint8_t i;
        for (i = 0; i < strlen(converted); i++)
  {
    result[counter] = returnCharfromHex(converted[i], 0); counter++;
    result[counter] = returnCharfromHex(converted[i], 1); counter++;
  }
  result[counter] = '\0';
        free(converted);

        return result;
}
/* ------------------------------------------------------------------------ */
```

Procedures.c

```
// Include Project Libraries
#include "definitions.h"
#include "procedures.h"
#include "usart/usart.h"
#include "gsm/gsm.h"
#include "gps/gps.h"

// Program memory variables
const unsigned char cmtistr[] PROGMEM = "\r\n+CMTI:";
const unsigned char beginofsms[] PROGMEM = ".outputGPS.";
const unsigned char location[] PROGMEM = "2716carLocation";
const unsigned char locationfailure[] PROGMEM =  "Unavailable";
const unsigned char shutdown[] PROGMEM = "2716carShutdown";
const unsigned char shutdownsuccess[] PROGMEM = "Shutdown Success";
const unsigned char shutdownfailure[] PROGMEM = "Shutdown Failure";
const unsigned char wrong_code[] PROGMEM = "Wrong Code";

PGM_P proceduresstaticdata[] PROGMEM =
{
        cmtistr,                // 0
        beginofsms,                     // 1
        location,                       // 2
        locationfailure, // 3
        shutdown,                       // 4
        shutdownsuccess,        // 5
        shutdownfailure,        // 6
        wrong_code                      // 7
};

/* ------------------------------ PROTOTYPES ------------------------------ */
void toggleReady(void);
void toggleBusy(void);
/* ------------------------------------------------------------------------ */

/* ---------------------------- PUBLIC METHODS ---------------------------- */
void startup()
{
        // Disable Watchdog
        wdt_disable();

        // Set sleep mode (idle)
        set_sleep_mode(SLEEP_MODE_IDLE);
```

```
	// Setup PORTA
	DDRC = 0x03;    // Set first 2 pins of PORTA to output
	PORTC = 0x00;  // Set all pins of PORTA as low
	// Setup PORTB
	DDRB = 0x03;    // Set first 2 pins of PORTB to output
	PORTB = 0x00;  // Set all pins of PORTB as low

	// Test Leds for 1 second
	toggleReady(); toggleBusy();
	_delay_ms(1000);
	toggleReady(); toggleBusy();
	_delay_ms(1000);

	// Turn on System Busy Led
	toggleBusy();
	_delay_ms(100);

	// Initialize USART
	uint16_t baudrate = 12; // UBRR value for Baud = 4800 @ 1 MHz
	initUSART(baudrate);
	// Initialize GSM
	initGSM();

	// Enable RXCIE Interrupt
	UCSR0B ^= 0x80;

	// Turn on System Ready Led && off System Busy Led
	toggleReady(); toggleBusy();
}

void processRoutine()
{
	// Turn on System Busy Led
	toggleBusy();

	// Common message = \r\n+CMTI: "SM",(%d+)\r\n (minimum size = 17)
	unsigned char *cmti = retrieveUSART(25, 2);
	unsigned char *teststring = returnSubStr(0, 7, cmti);

	// Compare start of received string to "\r\n+CMTI:"
	if (strcmp_P(teststring, (PGM_P)pgm_read_word(&(proceduresstaticdata[0]))) == 0)
	{
		SMS *receive = retrieveSMS(cmti);

		// Maximum size = 11 + 20 = 31 -> ".outputGPS." + gps_coords
```

```
		unsigned char *send = calloc(40, sizeof(unsigned char));
		strcpy_P(send, (PGM_P)pgm_read_word(&(proceduresstaticdata[1])));

		unsigned char *gps_coords = calloc(21, sizeof(unsigned char));

		uint8_t code = 0;
		if (strcmp_P(receive -> content, (PGM_P)pgm_read_word(&(proceduresstaticdata[2]))) == 0) code = 1;
		if (strcmp_P(receive -> content, (PGM_P)pgm_read_word(&(proceduresstaticdata[4]))) == 0) code = 2;

		switch (code)
		{
			case 0:
				strcat_P(send, (PGM_P)pgm_read_word(&(proceduresstaticdata[7])));
				break;

			case 1:
				; // <- C fix
				unsigned char *gps_coords_temp = retrieveGPS();
				strcpy(gps_coords, gps_coords_temp);
				free(gps_coords_temp);

				if ((gps_coords[0] == 'N') || (gps_coords[0] == 'S')) strcat(send, gps_coords);
				else strcat_P(send, (PGM_P)pgm_read_word(&(proceduresstaticdata[3])));
				break;

			case 2:
				PORTB = 0x02;
				if(PORTB == 0x02) strcat_P(send, (PGM_P)pgm_read_word(&(proceduresstaticdata[5])));
				else strcat_P(send, (PGM_P)pgm_read_word(&(proceduresstaticdata[6])));
				break;
		}
		// SEND SMS
		sendSMS(receive -> address_length, receive -> address, send);

		free(send);
		free(gps_coords);
		free(receive -> content);
		free(receive -> address_length);
```

```
                free(receive -> address);
                free(receive);

        }
        free(teststring);
        free(cmti);
        // Turn on System Ready Led
        toggleBusy();
}

// Método para retornar array de char composto pelos índices seleccionados
unsigned char *returnSubStr(int istart, int istop, unsigned char *input)
{
        unsigned char *substr = calloc(((istop - istart) + 1) + 1, sizeof(unsigned char));

        uint8_t counter = 0;
        uint8_t i;
        for (i = istart; i <= istop; i++)
        {
                substr[counter] = input[i];
                counter++;
        }
        substr[counter] = '\0';
        return substr;
}
/* ------------------------------------------------------------------------ */

/* ---------------------------- PRIVATE METHODS ---------------------------- */
void toggleReady(void) { PORTC ^= 0x01; }

void toggleBusy(void) { PORTC ^= 0x02; }
/* ------------------------------------------------------------------------ */
```

Definitions.h

```
#define F_CPU 1000000UL  // 8 MHz

#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>
```

```
#include <util/delay.h>
#include <avr/sleep.h>

#define __disable_interrupt cli
#define __enable_interrupt sei

#define FALSE   0
#define TRUE    1
```

Usart.h

```
void initUSART(int);
void sendUSART(unsigned char *);
unsigned char *retrieveUSART(int, int);
void sendChar(unsigned char); // Only public because of ctrl-z
```

Gps.h

```
unsigned char *retrieveGPS(void);
```

Gsm.h

```
typedef struct
{
        unsigned char *address_length;
        unsigned char *address;
        unsigned char *content;
} SMS;

void initGSM(void);

SMS *retrieveSMS(unsigned char *);

void sendSMS(unsigned char *, unsigned char *, unsigned char *);

#define byte 8
```

Procedures.h

```
void startup(void);
void processRoutine(void);
unsigned char *returnSubStr(int, int, unsigned char *);
```

Código-Fonte – Programa de visualização dos mapas

Information.cs

```
using System;
using System.Data;
using System.Collections.Generic;
using System.Text;
using System.IO;

namespace VehicleDetection
{
    class Information
    {
        #region Variables
        private DataTable info = new DataTable();
        private DataColumn telefone = new DataColumn();
        private DataColumn datahora = new DataColumn();
        private DataColumn latitude = new DataColumn();
        private DataColumn longitude = new DataColumn();
        #endregion

        public Information()
        {
            #region DataColumns configuration
            info.TableName = "Information";

            this.telefone.ColumnName = "Telefone";
            this.telefone.DataType =
System.Type.GetType("System.String");
            info.Columns.Add(telefone);

            this.datahora.ColumnName = "DataHora";
            this.datahora.DataType =
System.Type.GetType("System.DateTime");
            info.Columns.Add(datahora);

            this.latitude.ColumnName = "Latitude";
            this.latitude.DataType =
System.Type.GetType("System.Double");
            info.Columns.Add(latitude);

            this.longitude.ColumnName = "Longitude";
            this.longitude.DataType =
System.Type.GetType("System.Double");
            info.Columns.Add(longitude);
            #endregion
        }

        #region Methods
        // Method to add row to DataTable
        public void addRow(String tel, DateTime datetime, Double lat,
Double lon)
        {
            bool exists = false;
            for (int i = 0; i < info.Rows.Count; i++)
            {
```

```
                if (((String)info.Rows[i]["Telefone"] != tel) ||
((DateTime)info.Rows[i]["DataHora"] != datetime)) exists = true;
            }

            if (exists == false)
            {
                DataRow row = info.NewRow();
                // Fill row
                row["Telefone"] = tel;
                row["DataHora"] = datetime;
                row["Latitude"] = lat;
                row["Longitude"] = lon;
                // Add row to Temporary DataTable
                info.Rows.Add(row);
            }

            // Sort DataTable
            info = this.sortDataTable(info);
        }

        // Method to sort DataTable - Private
        private DataTable sortDataTable(DataTable dt)
        {
            DataTable sorted = dt.Clone();

            if (dt.Rows.Count > 1)
            {
                List<DataRow> list = new List<DataRow>();
                foreach (DataRow dr in dt.Rows) list.Add(dr);

                #region Selection Sort Algorithm
                int counter = 0;
                while (counter < list.Count)
                {
                    DataRow max = list[0];
                    int index = 0;
                    for (int i = 0; i < list.Count - counter; i++)
                    {
                        DateTime maxDate = (DateTime)max["DataHora"];
                        DateTime auxDate = (DateTime)list[i]["DataHora"];

                        if (maxDate < auxDate)
                        {
                            max = list[i];
                            index = i;
                        }
                    }
                    DataRow temp = list[list.Count - 1 - counter];
                    list[list.Count - 1 - counter] = list[index];
                    list[index] = temp;
                    counter++;
                }
                #endregion

                sorted.Rows.Clear();
                foreach (DataRow dr in list)
                {
                    DataRow row = sorted.NewRow();
                    row["Telefone"] = dr["Telefone"];
                    row["DataHora"] = dr["DataHora"];
                    row["Latitude"] = dr["Latitude"];
```

```
                    row["Longitude"] = dr["Longitude"];
                    sorted.Rows.Add(row);
                }
            }
            return sorted;
        }

        // Method to remove row from DataTable
        public void delRow(int index)
        {
            info.Rows.RemoveAt(index);
        }

        // Method to set DataTable
        public void set(DataTable dt)
        {
            info = this.sortDataTable(dt);
        }

        // Method to get DataTable
        public DataTable get()
        {
            return info;
        }
        #endregion
    }
}
```

SerialInterface.cs

```
using System;
using System.Data;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.IO.Ports;

namespace VehicleDetection
{
    class SerialInterface
    {
        // Global Variables
        private SerialPort port = new SerialPort();

        // Constructor
        public SerialInterface()
        {
            this.port.NewLine = Environment.NewLine;
            this.port.ReadTimeout = 500;
        }

        // Public methods
        public void setPortName(String pName) { this.port.PortName =
pName; }
        public void setBaudRate(int baud) { this.port.BaudRate = baud; }
        public void setDataBits(int datab) { this.port.DataBits = datab;
}
        public void setParity(Parity prt) { this.port.Parity = prt; }
        public void setStopBits(StopBits stp) { this.port.StopBits = stp;
}
```

```
        public void setHandshake(Handshake hsk) { this.port.Handshake =
hsk; }

        public List<string> transfer(String cmd)
        {
            List<string> data = new List<string>();
            try
            {
                port.Open();
                data = communicate(cmd);
                port.Close();
            }
            catch (Exception exp)
            {
                throw exp;
            }
            return data;
        }

        // Private methods
        private List<string> communicate(String command)
        {
            List<string> lst = new List<string>();
            bool timeout = false;

            port.WriteLine(command);
            // Because method BytesToRead malfunctions on virtual ports,
returning allways 0
            // Used ReadTimeout as condition to while cycle through
TimeoutException
            while (timeout == false)
            {
                try
                {
                    lst.Add(port.ReadLine());
                }
                catch (TimeoutException)
                {
                    timeout = true;
                }
            }
            return lst;
        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using System.IO;
using System.IO.Ports;
using System.Text.RegularExpressions;

namespace VehicleDetection
{
    static class Program
    {
```

```
        /// <summary>
        /// The main entry point for the application.
        /// </summary>

        #region Program Control Section
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            fillDataTable();
            Application.Run(new mainWindow());
        }

        public static void endProgram()
        {
            Application.Exit();
        }
        #endregion

        #region Static variables
        private static Information information = new Information();
        private static SerialInterface serial = new SerialInterface();
        private static propertiesWindow properties;

        private static int[] preferences = new int[6];

        private static Regex smsVars = new Regex("^\\.outputGPS\\.(N|S)-
(\\d{2}),(\\d{4})\\.(E|W)-(\\d{2}),(\\d{4})\\.$");
        private static Regex smsID = new Regex("\\+CMGL:\\s(\\d+)");
        private static Regex smsNbrDateTime = new
Regex("^.+\"\\+(\\d{3})(\\d{9})\".+\"(\\d{4})/(\\d{1,2})/(\\d{1,2}),(\\d{
1,2}):(\\d{1,2}):(\\d{1,2})\"$");
        #endregion

        #region DataTable operations

        private static void fillDataTable()
        {
            try
            {
                DataTable table = new DataTable();
                table.ReadXmlSchema("InformationSchema.xml");
                table.ReadXml("InformationData.xml");
                information.set(table);
            }
            catch (IOException) { }
        }

        public static void upgradeInformation(DataTable dt)
        {
            foreach (DataRow dr in dt.Rows)
            {
                String telephone = (String)dr["Telefone"];
                DateTime datetime = (DateTime)dr["DataHora"];
                Double latitude = (Double)dr["Latitude"];
                Double longitude = (Double)dr["Longitude"];
                information.addRow(telephone, datetime, latitude,
longitude);
            }
        }
```

```
        public static void downgradeInformation(int i)
        {
            information.delRow(i);
        }

        public static DataTable getInformation()
        {
            return information.get();
        }

        public static void saveDataTable(DataTable dt)
        {
            try
            {
                dt.WriteXmlSchema("InformationSchema.xml");
                dt.WriteXml("InformationData.xml");
            }
            catch (IOException) { }
        }

        #endregion

        #region Properties window Operations
        [STAThread]
        public static void getProperties(mainWindow mw)
        {
            properties = new propertiesWindow(mw);
            properties.ShowDialog();
        }

        public static void setPreferences(int[] prefs) { preferences =
prefs; }

        public static int[] getPreferences() { return preferences; }

        public static DataTable getCoordinates(String portname, int
baudrate, int databits, Parity prt, StopBits stopbits, Handshake
handshake)
        {
            #region Serial Interface configuration
            serial.setPortName(portname);
            serial.setBaudRate(baudrate);
            serial.setDataBits(databits);
            serial.setParity(prt);
            serial.setStopBits(stopbits);
            serial.setHandshake(handshake);
            // Progress bar + 1
            updateProgressBar(1);
            #endregion

            #region Communication & Data manipulation
            DataTable result = information.get().Clone();
            result.Rows.Clear();
            //AT Command
            String at = "AT";
            if (testCmd(at, getSerialData(at, 0)))
            {
                // Progress bar + 2
                updateProgressBar(2);
```

```
                // AT+CMGF=1 Command
                String atcmgf = "AT+CMGF=1";
                if (testCmd(atcmgf, getSerialData(atcmgf, 0)))
                {
                    // Progress bar + 2
                    updateProgressBar(2);

                    // AT+CMGL="ALL" Command
                    String atcmgl = "AT+CMGL=\"ALL\"";
                    if (getSMS(atcmgl).Count != 0)
                    {
                        updateMessageLog(atcmgl, getSMS(atcmgl));
                        // Progress bar + 2
                        updateProgressBar(2);

                        List<int> codes = codesRegExp(getSMS(atcmgl));
                        // Progress bar + 1
                        updateProgressBar(1);

                        foreach (int code in codes)
                        {
                            List<string> sms = new List<string>();
                            Object[] obj = new Object[4];
                            // AT+CMGF=(code) Command
                            String atcmfg = "AT+CMGR=" + code.ToString();
                            sms = getSMS(atcmfg);
                            obj = infoRegExp(sms);
                            updateMessageLog(atcmfg, getSMS(atcmfg));

                            DataRow row = result.NewRow();
                            row["Telefone"] = (String)obj[0];
                            row["DataHora"] = (DateTime)obj[1];
                            row["Latitude"] = (Double)obj[2];
                            row["Longitude"] = (Double)obj[3];
                            result.Rows.Add(row);
                        }
                        // Progress bar + 2
                        updateProgressBar(2);
                    }
                }
            }
            return result;
            #endregion
        }

        #region Private communication & data manipulation support methods
        private static List<string> getSerialData(String cmd, int error)
        {
            List<string> data = new List<string>();
            if (error < 10)
            {
                try
                {
                    data = serial.transfer(cmd);
                }
                catch (Exception exp)
                {
                    data.Clear();
                    data.Add(exp.Message);
                    properties.setProgressBar(0);
                }
```

```
                if (data.Count != 0 && data[data.Count -
1].Equals("ERROR"))
                {
                    getSerialData(cmd, error + 1);
                }
            }
            else
            {
                data.Clear();
            }
            return data;
        }

        private static bool testCmd(String cmd, List<string> list)
        {
            bool test = false;
            switch (list.Count)
            {
                case 0:
                    MessageBox.Show("Erro contínuo ao transmitir o
comando " + cmd + ", por favor verifique a ligação e tente novamente!",
"Erro");
                    break;
                case 1:
                    MessageBox.Show(list[list.Count - 1]);
                    break;
            }
            if (list.Count > 1)
            {
                updateMessageLog(cmd, list);
                test = true;
            }
            return test;
        }

        private static List<string> getSMS(String cmd)
        {
            List<string> data = new List<string>();
            try
            {
                data = serial.transfer(cmd);
            }
            catch (Exception exp) { MessageBox.Show(exp.Message); }
            if (data[data.Count - 1].Equals("ERROR"))
            {
                MessageBox.Show("Obtido " + data[data.Count -
1].ToString() + " na leitura das mensagens." + Environment.NewLine +
                                "Alguns dados podem não ter sido obtidos
com sucesso." + Environment.NewLine + Environment.NewLine +
                                "O programa continuará a sua execução
regular. Porém, é recomendado que verifique" + Environment.NewLine +
                                "a integridade das mensagens SMS
presentes no telemóvel, e que tente novamente.", "ATENÇÃO");
            }

            return data;
        }

        private static List<int> codesRegExp(List<string> messages)
        {
```

```
            List<int> msgcode = new List<int>();
            for (int i = 0; i < messages.Count; i++)
            {
                if (smsVars.IsMatch(messages[i]))
                {
                    String[] id = smsID.Split(messages[i - 1]);
                    msgcode.Add(int.Parse(id[1]));
                }
            }
            return msgcode;
        }

        private static Object[] infoRegExp(List<string> message)
        {
            Object[] obj = new Object[4];

            for (int i = 1; i < message.Count; i++)
            {
                if (smsVars.IsMatch(message[i]))
                {
                    String temp1 = message[i - 1];
                    String temp2 = message[i];

                    String[] generalData = smsNbrDateTime.Split(temp1);
                    String[] locationData = smsVars.Split(temp2);

                    obj[0] = "+" + generalData[1] + generalData[2];
                    obj[1] = new DateTime(int.Parse(generalData[3]),
                        int.Parse(generalData[4]),
int.Parse(generalData[5]),
                        int.Parse(generalData[6]),
int.Parse(generalData[7]),
                        int.Parse(generalData[8]));
                    String latitude = String.Empty;
                    if (locationData[1].Equals("S")) latitude += "-";
                    latitude += locationData[2] + "," + locationData[3];
                    obj[2] = double.Parse(latitude);
                    String longitude = String.Empty;
                    if (locationData[4].Equals("W")) longitude += "-";
                    longitude += locationData[5] + "," + locationData[6];
                    obj[3] = double.Parse(longitude);
                }
            }
            return obj;
        }
        #endregion

        #region Message Log Update Methods
        private static void updateMessageLog(String cmd, List<string>
msgs)
        {
            List<string> log = new List<string>();
            if (cmd.Equals("AT") || cmd.Equals("AT+CMGF=1"))
            {
                log.Add("Enviado comando: " + cmd + Environment.NewLine);
                log.Add("Resposta obtida: " + msgs[1] +
Environment.NewLine);
                log.Add("-" + Environment.NewLine);
            }
            else if (cmd.Equals("CMGL=\"ALL\""))
            {
```

```
                log.Add("Enviado comando: " + cmd + Environment.NewLine);
                for (int i = 1; i < log.Count - 2; i++)
                {
                    log.Add(msgs[i] + Environment.NewLine);
                }
                if (msgs.Count == 3) log.Add("Resposta obtida: " +
msgs[msgs.Count - 2] + Environment.NewLine);
                else log.Add("Resposta obtida: " + msgs[msgs.Count - 1] +
Environment.NewLine);
                log.Add("-" + Environment.NewLine);
            }
            else
            {
                int cnt = 1;
                log.Add("Enviado comando: " + cmd + Environment.NewLine);
                while (cnt < msgs.Count - 1)
                {
                    log.Add(msgs[cnt] + Environment.NewLine);
                    cnt++;
                }
                log.Add("Resposta obtida: " + msgs[msgs.Count - 1] +
Environment.NewLine);
                log.Add("-" + Environment.NewLine);
            }
            updateText(log);
        }

        [STAThread]
        private static void updateText(List<string> list)
        {
            properties.writeTextBox(list);
        }

        [STAThread]
        private static void updateProgressBar(int i)
        {
            properties.incrementProgressBar(i);
        }
        #endregion

        #endregion
    }
}
```

Controlo Map.cs

```
using System.Windows.Forms;
using GMap.NET.WindowsForms;
//using System.Drawing;

/// <summary>
/// custom map of GMapControl
/// </summary>
public class Map : GMapControl
{
    private void InitializeComponent()
    {
        this.SuspendLayout();
        //
        // Map
        //
```

```
        this.Name = "map";
        this.ResumeLayout(false);
    }
}
```

Janela propertiesWindow.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Windows.Forms;
using System.IO.Ports;

namespace VehicleDetection
{
    public partial class propertiesWindow : Form
    {
        // Variables
        #region String[] content for Comboboxes
        private String[] com = SerialPort.GetPortNames();
        private String[] baud =
{"110","300","1200","2400","4800","9600","19200","38400","57600","115200"
,
                                "230400","460800","921600"};
        private String[] bitdata = { "5", "6", "7", "8" };
        private String[] prt = Enum.GetNames(typeof(Parity));
        private String[] stpb = new String[3];
        private String[] handsk = Enum.GetNames(typeof(Handshake));
        #endregion

        private DataTable info = new DataTable();

        // Constructor
        mainWindow parent;
        public propertiesWindow(mainWindow mw)
        {
            parent = mw;

            InitializeComponent();

            #region Fill StopBits String Array
            int i = 0;
            foreach (String s in Enum.GetNames(typeof(StopBits)))
            {
                if (!s.Equals("None"))
                {
                    stpb[i] = s;
                    i++;
                }
            }
            #endregion

            #region Fill ComboBoxes
            this.fillCombo(this.com, this.portS); this.selDefCombo(0,
this.portS);
            this.fillCombo(this.baud, this.bps); this.selDefCombo(0,
this.bps);
            this.fillCombo(this.bitdata, this.datab); this.selDefCombo(0,
this.datab);
            this.fillCombo(this.prt, this.parity); this.selDefCombo(0,
this.parity);
```

```
            this.fillCombo(this.stpb, this.stopb); this.selDefCombo(0,
this.stopb);
            this.fillCombo(this.handsk, this.hsk); this.selDefCombo(0,
this.hsk);

            if (Program.getPreferences().Length != 0)
            {
                this.selDefCombo(Program.getPreferences()[0],
this.portS);
                this.selDefCombo(Program.getPreferences()[1], this.bps);
                this.selDefCombo(Program.getPreferences()[2],
this.datab);
                this.selDefCombo(Program.getPreferences()[3],
this.parity);
                this.selDefCombo(Program.getPreferences()[4],
this.stopb);
                this.selDefCombo(Program.getPreferences()[5], this.hsk);
            }
            #endregion

            progressBar.Maximum = 10;
            progressBar.Minimum = 0;
        }

        // Public methods
        #region Combobox Manipulation - Private methods
        // Method to automatically fill ComboBoxes
        private void fillCombo(String[] str, ComboBox lst)
        {
            foreach (String s in str) lst.Items.Add(s);
        }

        // Method to select ComboBox Default Item
        private void selDefCombo(int i, ComboBox lst)
        {
            try
            {
                lst.SelectedItem = lst.Items[i];
            }
            catch (NullReferenceException) {}
        }
        #endregion

        #region CheckedListBox Update - Private method
        private void UpdateChkLstBox(DataTable tb)
        {
            // Clear CheckedListBox
            dataBox.Items.Clear();
            // Fill CheckedListBox
            for (int i = 0; i < tb.Rows.Count; i++)
            {
                String s = String.Empty;
                for (int j = 0; j < tb.Rows[i].ItemArray.Length; j++)
                {
                    if (j < 2)
                    {
                        if (j == 0) s += (String)tb.Rows[i].ItemArray[j]
+ "      ";
                        if (j == 1)
                        {
```

```
                            DateTime datetime =
(DateTime)tb.Rows[i].ItemArray[j];
                            s += datetime.ToString("yyyy/MM/dd -
HH:mm:ss") + "      ";
                        }
                    }
                    else
                    {
                        if
(!(tb.Rows[i].ItemArray[j].ToString().StartsWith("-")))
                        {
                            s += " " + tb.Rows[i].ItemArray[j].ToString()
+ "         ";
                        }
                        else s += tb.Rows[i].ItemArray[j].ToString() + "
";
                    }
                }
                dataBox.Items.Add(s);
            }

        }
        #endregion

        #region Transfer button procedures
        private void transfer_Click(object sender, EventArgs e)
        {
            String portname = portS.Text;
            int baudrate = int.Parse(bps.Text);
            int databits = int.Parse(datab.Text);
            Parity prt = (Parity)Enum.Parse(typeof(Parity), parity.Text);
            StopBits stopbits = (StopBits)Enum.Parse(typeof(StopBits),
stopb.Text);
            Handshake handshake =
(Handshake)Enum.Parse(typeof(Handshake), hsk.Text);

            progressBar.Value = 0;
            this.commlog.Text = String.Empty;
            this.info = Program.getCoordinates(portname, baudrate,
databits, prt, stopbits, handshake);
            this.UpdateChkLstBox(info);
            if (this.info.Rows.Count != 0) erase.Enabled = true;
            this.commlog.SelectionStart = this.commlog.Text.Length;
            this.commlog.ScrollToCaret();
        }
        #endregion

        #region Log textbox update method
        public void writeTextBox(List<string> messages)
        {
            foreach (string s in messages) commlog.Text += s;
        }
        #endregion

        #region ProgressBar Operation Methods
        public void incrementProgressBar(int i)
        {
            progressBar.Increment(i);
        }

        public void setProgressBar(int i)
```

```
        {
            progressBar.Value = 0;
        }

        #endregion

        #region OK button Procedures
        private void ok_Click(object sender, EventArgs e)
        {
            int[] prefs = new int[6];

            prefs[0] = portS.SelectedIndex;
            prefs[1] = bps.SelectedIndex;
            prefs[2] = datab.SelectedIndex;
            prefs[3] = parity.SelectedIndex;
            prefs[4] = stopb.SelectedIndex;
            prefs[5] = hsk.SelectedIndex;

            Program.setPreferences(prefs);

            Program.upgradeInformation(info);
            parent.updateInformation();
            this.Close();
        }
        #endregion

        #region Cancel button procedures
        private void cancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        #endregion

        #region Erase button procedures
        private void erase_Click(object sender, EventArgs e)
        {
            if (dataBox.SelectedIndices.Count != 0)
            {
                for (int i = 0; i < dataBox.CheckedIndices.Count; i++)
                {
                    info.Rows[dataBox.CheckedIndices[i] - i].Delete();
                }
                UpdateChkLstBox(info);
                if (info.Rows.Count == 0) erase.Enabled = false;
            }
        }
        #endregion
    }
}
```

Janela mainWindow.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using GMap.NET;
```

```
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.WindowsForms.ToolTips;

namespace VehicleDetection
{
    public partial class mainWindow : Form
    {
        private DataTable information = new DataTable();
        private List<DataRow> checkedInformation = new List<DataRow>();
        private List<PointLatLng> coordinates = new List<PointLatLng>();
        private int coordinateIndex;
        private double currentZoom;
        internal GMapOverlay routes;
        internal GMapOverlay markers;

        public mainWindow()
        {
            InitializeComponent();

            routes = new GMapOverlay(map, "routes");
            map.Overlays.Add(routes);
            markers  = new GMapOverlay(map, "markers");
            map.Overlays.Add(markers);

            #region Load information window (Datatable to checkedlistbox)
            this.information = Program.getInformation();
            this.UpdateChkLstBox(information);
            if (Program.getInformation().Rows.Count != 0)
            {
                selAll.Enabled = true;
                selNone.Enabled = false;
                erase.Enabled = false;
            }
            else
            {
                MessageBox.Show("Não existe ainda informação no sistema!"
+ Environment.NewLine +
                                "Pressione o botão 'Actualizar' para
importar os dados.", "Início");
                selAll.Enabled = false;
                selNone.Enabled = false;
                erase.Enabled = false;
                view.Enabled = false;
            }
            #endregion

            #region Events declaration
            this.FormClosing +=new
FormClosingEventHandler(mainWindow_FormClosing);
            this.checkedListBox1.ItemCheck += new
ItemCheckEventHandler(checkedListBox1_ItemCheck);
            #endregion
        }

        #region Button control
        void checkedListBox1_ItemCheck(object sender, ItemCheckEventArgs
e)
        {
            selAll.Enabled = true;
            selNone.Enabled = true;
```

```
            switch (e.CurrentValue)
            {
                case CheckState.Checked:
                    if (testChkBox(false, false, e.Index))
                    {
                        selAll.Enabled = true;
                        selNone.Enabled = false;
                        erase.Enabled = false;
                        view.Enabled = false;
                    }
                    break;

                case CheckState.Unchecked:
                    erase.Enabled = true;
                    view.Enabled = true;
                    if (testChkBox(true, true, e.Index))
                    {
                        selNone.Enabled = true;
                        selAll.Enabled = false;
                    }
                    break;
            }
        }

        private bool testChkBox(bool b, bool c, int nbr)
        {
            bool test = b;
            int i = 0;
            while (test == b && i < checkedListBox1.Items.Count)
            {
                if (i == nbr) test =
!(checkedListBox1.GetItemChecked(i));
                else test = checkedListBox1.GetItemChecked(i);
                i++;
            }
            return (test == c);
        }

        private void selAll_Click(object sender, EventArgs e)
        {
                for (int i = 0; i < checkedListBox1.Items.Count; i++)
checkedListBox1.SetItemChecked(i, true);
                selAll.Enabled = false;
                selNone.Enabled = true;
                erase.Enabled = true;
                view.Enabled = true;
        }

        private void selNone_Click(object sender, EventArgs e)
        {
            for (int i = 0; i < checkedListBox1.Items.Count; i++)
checkedListBox1.SetItemChecked(i, false);
            selNone.Enabled = false;
            selAll.Enabled = true;
            erase.Enabled = false;
            view.Enabled = false;
        }
        #endregion

        #region On form closing event operations
```

```
        private void mainWindow_FormClosing(Object sender,
FormClosingEventArgs e)
        {
            switch (MessageBox.Show("Deseja guardar as alterações
efectuadas às coordenadas?", "Sair", MessageBoxButtons.YesNoCancel))
            {
                case DialogResult.Yes:
                    Program.saveDataTable(this.information);
                    break;
                case DialogResult.No:
                    break;
                case DialogResult.Cancel:
                    e.Cancel = true;
                    break;
            }
        }
        #endregion

        #region CheckListBox Update
        public void UpdateChkLstBox(DataTable tb)
        {
            // Clear CheckedListBox
            checkedListBox1.Items.Clear();
            // Fill CheckedListBox
            for (int i = 0; i < tb.Rows.Count; i++)
            {
                String s = String.Empty;
                for (int j = 0; j < tb.Rows[i].ItemArray.Length; j++)
                {
                    if (j < 2)
                    {
                        if (j == 0) s += (String)tb.Rows[i].ItemArray[j]
+ "      ";
                        if (j == 1)
                        {
                            DateTime datetime =
(DateTime)tb.Rows[i].ItemArray[j];
                            s += datetime.ToString("yyyy/MM/dd -
HH:mm:ss") + "      ";
                        }
                    }
                    else
                    {
                        if
(!(tb.Rows[i].ItemArray[j].ToString().StartsWith("-")))
                        {
                            s += " " + tb.Rows[i].ItemArray[j].ToString()
+ "          ";
                        }
                        else s += tb.Rows[i].ItemArray[j].ToString() + "
";
                    }
                }
                checkedListBox1.Items.Add(s);
            }

            // Control 'erase' button
            if (this.information.Rows.Count == 0) erase.Enabled = false;
            else erase.Enabled = true;
        }
        #endregion
```

```
        #region Erase checkedlistbox items button operations
        private void erase_Click(object sender, EventArgs e)
        {
            for (int i = 0; i <
this.checkedListBox1.CheckedIndices.Count; i++)
            {

Program.downgradeInformation(checkedListBox1.CheckedIndices[i] - i);
            }

            UpdateChkLstBox(this.information);
            if (this.information.Rows.Count == 0)
            {
                selAll.Enabled = false;
                selNone.Enabled = false;
                view.Enabled = false;
            }
        }
        #endregion

        #region Update Button - Get more coordinates
        private void update_Click(object sender, EventArgs e)
        {
            Program.getProperties(this);
        }

        public void updateInformation()
        {
            this.UpdateChkLstBox(information);
        }

        #endregion

        #region Visualization operations
        private void view_Click(object sender, EventArgs e)
        {
            Boolean connection = true;
            try
            {
                System.Net.IPHostEntry conn =
System.Net.Dns.GetHostEntry("www.google.com");
            }
            catch
            {
                MessageBox.Show("Erro: Ligação à internet indisponível,
por favor verifique a conexão e tente novamente.");
                connection = false;
            }

            map.Enabled = false;
            zoomIn.Enabled = false;
            zoomOut.Enabled = false;
            zoomView.Enabled = false;
            before.Enabled = false;
            after.Enabled = false;

            if (connection == true)
            {
                checkedInformation.Clear();
```

```
                for (int i = 0; i < 
this.checkedListBox1.CheckedIndices.Count; i++)
                {

checkedInformation.Add(this.information.Rows[checkedListBox1.CheckedIndic
es[i]]);
                }

                if (this.checkedInformation.Count == 0)
MessageBox.Show("Erro: Não foram seleccionadas coordenadas!");
                else
                {
                    coordinates.Clear();

                    zoomOut.Enabled = true;
                    zoomView.Enabled = true;

                    for (int i = 0; i < this.checkedInformation.Count; 
i++)
                    {
                        double latitude = 
(Double)this.checkedInformation[i]["Latitude"];
                        double longitude = 
(Double)this.checkedInformation[i]["Longitude"];
                        coordinates.Add(new PointLatLng(latitude, 
longitude));
                    }
                    if (this.coordinates.Count > 1) before.Enabled = 
true;
                    this.coordinateIndex = this.coordinates.Count - 1;

                    showMap();
                    this.updateCoordinate(coordinateIndex);
                }
            }
        }

        private void showMap()
        {
            map.Enabled = true;
            routes.Routes.Clear();
            markers.Markers.Clear();

            GMaps.Instance.Mode = AccessMode.ServerOnly;
            GMaps.Instance.UsePlacemarkCache = false;
            GMaps.Instance.UseRouteCache = false;


            map.MaxZoom = 15;
            map.MinZoom = 7;
            map.Zoom = map.MaxZoom;

            map.OnMarkerEnter += new MarkerEnter(map_OnMarkerEnter);
            map.OnMarkerLeave += new MarkerLeave(map_OnMarkerLeave);

            Font markerfont = new Font(FontFamily.GenericSansSerif, 
(float)10);
            for (int i = 0; i < coordinates.Count; i++)
            {
                GMapMarkerGoogleRed redmarker = new 
GMapMarkerGoogleRed(coordinates[i]);
```

```
                redmarker.ToolTipText = (i + 1).ToString();
                redmarker.ToolTip.Font = markerfont;
                redmarker.ToolTipMode = MarkerTooltipMode.Always;
                markers.Markers.Add(redmarker);
                if (i > 0)
                {
                    MapRoute route =
GMaps.Instance.GetRouteBetweenPoints(coordinates[i - 1], coordinates[i],
false, (int) map.Zoom);
                    if (route != null)
                    {
                        GMapRoute path = new GMapRoute(route.Points,
"Path");
                        routes.Routes.Add(path);
                    }
                }
            }
        }

        void map_OnMarkerEnter(GMapMarker item)
        {
            Placemark p = null;
            p = GMaps.Instance.GetPlacemarkFromGeocoder(item.Position);

            if (p != null)
            {
                char[] str = p.Address.ToCharArray();
                for (int i = 0; i < str.Length; i++)
                {
                    if (str[i] == ',') str[i] = '\n';
                }
                String address = new String(str);
                item.ToolTipText = address;
            }
            else
            {
                item.ToolTipText = item.Position.ToString();
            }
        }

        void map_OnMarkerLeave(GMapMarker item)
        {
            item.ToolTipText = (coordinateIndex + 1).ToString();
        }

        // Visualization controls
        private void before_Click(object sender, EventArgs e)
        {
            after.Enabled = true;
            if (coordinateIndex - 1 == 0) before.Enabled = false;

            this.coordinateIndex--;
            this.updateCoordinate(coordinateIndex);
        }

        private void after_Click(object sender, EventArgs e)
        {
            before.Enabled = true;
            if (coordinateIndex + 1 == this.coordinates.Count - 1)
after.Enabled = false;
```

```
            this.coordinateIndex++;
            this.updateCoordinate(coordinateIndex);
        }

        private void updateCoordinate(int index)
        {
            textPosition.Text = (index + 1).ToString();
            textPhone.Text =
(String)this.checkedInformation[index]["Telefone"];
            DateTime datetime =
(DateTime)this.checkedInformation[index]["DataHora"];
            textDate.Text = datetime.ToString("yyyy/MM/dd");
            textTime.Text = datetime.ToString("HH:mm:ss");

            double latitude =
(Double)this.checkedInformation[index]["Latitude"];
            textLatitude.Text = latitude.ToString();

            double longitude =
(Double)this.checkedInformation[index]["Longitude"];
            textLongitude.Text = longitude.ToString();

            map.CurrentPosition = coordinates[index];
            map.ReloadMap();
        }

        private void zoomOut_Click(object sender, EventArgs e)
        {
            zoomIn.Enabled = true;
            map.Zoom--;
            if (map.Zoom == map.MinZoom) zoomOut.Enabled = false;

        }

        private void zoomIn_Click(object sender, EventArgs e)
        {
            zoomOut.Enabled = true;
            map.Zoom++;
            if (map.Zoom == map.MaxZoom) zoomIn.Enabled = false;
        }

        private void zoomView_Click(object sender, EventArgs e)
        {
            switch (zoomView.Text)
            {
                case "-":
                    zoomView.Text = "+";
                    zoomIn.Enabled = false;
                    zoomOut.Enabled = false;
                    before.Enabled = false;
                    after.Enabled = false;
                    currentZoom = map.Zoom;
                    map.ZoomAndCenterMarkers(markers.Id);
                    break;
                case "+":
                    zoomView.Text = "-";
                    if (currentZoom != map.MinZoom) zoomOut.Enabled =
true;
                    if (currentZoom != map.MaxZoom) zoomIn.Enabled =
true;
                    if (this.coordinates.Count > 1)
```

```
                    {
                        if (coordinateIndex != 0) before.Enabled = true;
                        if (coordinateIndex != coordinates.Count - 1)
after.Enabled = true;
                    }
                    map.Zoom = currentZoom;
                    updateCoordinate(coordinateIndex);
                    break;
                }
            }
            #endregion
        }
}
```