Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2016

Dinis Hugo
Lima Brás

Sistema de Baixo Custo para Medição de Higrotermia

Dinis Hugo Lima Brás

Sistema de Baixo Custo para Medição de Higrotermia

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Automação Industrial, realizada sob a orientação científica do Doutor Arnaldo Silva Rodrigues de Oliveira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e co-orientação científica da Doutora Inês Osório de Castro Meireles, Professora Auxiliar do Departamento de Engenharia Civil da Universidade de Aveiro.

o júri / the jury

presidente / president	**Prof. Doutor José Paulo Oliveira Santos** Professor Auxiliar da Universidade de Aveiro
vogais / examiners committee	**Prof. Doutor Valter Filipe Miranda Castelão da Silva** Professor Adjunto da Universidade de Aveiro (arguente)
	Prof. Doutor Arnaldo Silva Rodrigues de Oliveira Professor Auxiliar da Universidade de Aveiro (orientador)
	Prof. Doutora Inês Osório de Castro Meireles Professora Auxiliar da Universidade de Aveiro (co-orientadora)

agradecimentos / acknowledgements

A realização desta dissertação contou com importantes apoios e incentivos de várias pessoas, a quem quero expressar o meu agradecimento.

Em primeiro lugar gostaria de agradecer à minha família, por estar presente em todos os momentos e me encorajarem a seguir em frente perante os obstáculos diários.

Ao meu orientador, Professor Doutor Arnaldo Oliveira agradeço a sua supervisão e orientação ao longo do desenvolvimento da dissertação.

Agradeço também à minha co-orientadora, Professora Doutora Inês Meireles pela disponibilidade e conselhos.

O meu agradecimento ao Professor Vítor Sousa pela sua participação e colaboração nas questões relacionadas com a higrotermia.

Por último, gostaria de deixar um especial agradecimento aos meus amigos e colegas de curso, pelo apoio e incentivo na concretização desta dissertação.

Palavras chave

Domótica, protocolos de comunicação, higrotermia, arduino, android, temperatura, humidade relativa, dispositivos remotos, sensores.

Resumo

O presente trabalho demonstra o desenvolvimento de um sistema de baixo custo para medição de Higrotermia. A Higrotermia consiste na medição da humidade e da temperatura, sendo um contributo ativo para a domótica. A domótica caracteriza-se por automatizar tarefas domésticas que proporcionam conforto, segurança e eficiência na racionalização dos recursos energéticos.

Fez-se uma abordagem à Higrotermia nomeadamente, os conceitos fundamentais, os instrumentos de medida e a apresentação de algumas soluções comerciais. Apresentam-se também os protocolos de comunicação mais representativos na domótica.
Demonstra-se a solução higrotérmica desenvolvida, cuja aplicação principal é a monitorização da humidade e da temperatura em edifícios no âmbito da Engenharia Civil, através de dispositivos de sensores remotos (plataforma Arduino).

Através de uma aplicação Android é feita a interação entre utilizador e o dispositivo de sensores remoto, utilizando a rede WiFi local (LAN - *Local Area Network*). O utilizador conecta-se a dispositivos de sensores remoto para enviar comandos nomeadamente, o tempo que o dispositivo remoto deve estar em *standby*, desligar o dispositivo e obter o valor das variáveis humidade e temperatura no instante atual.

Por intermédio de um programa desenvolvido em MatLab, é possível analisar a evolução (através de gráficos) de cada uma das variáveis em função do tempo, que pode ser ao dia, ao mês ou ao ano. São ainda apresentados o valor máximo, o valor mínimo e a média dos resultados.

Keywords Home automation, communication protocols, hygrothermal, arduino, android, temperature, relative humidity, remote devices, sensors.

Abstract

The present work aims at demonstrating the development of a low cost system for measuring higrothermy. Higrothermy consists in the measurement of humidity and of temperature, therefore having an active contribution for domotics. The latter is characterized by automating domestic chores to provide comfort, safety and efficiency in the rationalization of energy resources.

An approach to higrothermy was made, namely its fundamental concepts, the measuring equipment and some commercial solutions. Domotics most representative communication protocols are also shown.
The developed hygrothermal solution, whose main application is the monitoring of humidity and temperature in building for civil engineering purposes through remote sensor devices (Arduino platform).

Through an Android application, the interaction between the user and the remote sensor devices is established, using a Local Area Network. The user connects to remote sensor devices to send commands, namely, how long should the device standby, turn off the device and obtain the humidity and temperature variables instantly.

By means of a Matlab developed program, it's possible to analyze the evolution (through graphs) of each variable as a function of time, which can be daily, monthly or yearly. Maximum, minimum and mean results are also shown.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ANSI	American National Standards Institute
AVAC	Aquecimento, Ventilação e Ar Condicionado
CSMA	Carrier Sense Multiple Access
DCR	Deterministic Collision Resolution
EIA	Electronic Industries Alliance
FFD	Full Function Devices
HR	Humidade Relativa
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
IR	Infra Red
ISM	Industrial, Scientific and Medical
OSI	Open Systems Interconnection
PLC	PowerLine Carrier
PL	Power Line
P2P	Peer-to-Peer
QAI	Qualidade do Ar Interior
RCCTE	Regulamento das Caraterísticas de Comportamento Térmico dos Edifícios
RECS	Regulamento de desempenho Energético dos edifícios de Comércio e Serviços
RF	Radio Frequency
RFD	Reduced Function Devices
REH	Regulamento de desempenho Energético dos edifícios de Habitação

SCE	Sistema de Certificação Energética dos edifícios
SD	Secure Digital
SDK	Software Development Kit
SI	Sistema Internacional de Unidades
UDP	User Datagram Protocol
UML	Unified Modeling Language
WHANs	Wireless Home Automation Networks
XML	eXtensive Markup Language

Capítulo 1

Introdução

1.1 Enquadramento

Nos dias de hoje, as pessoas passam a maior parte do seu tempo em meios de transporte ou no interior de edifícios. Verifica-se que, a qualidade dos edifícios não está apenas relacionada com a sua qualidade estrutural e características arquitetónicas, é indispensável que existam condições de conforto, quer por questões de saúde, quer por questões de desempenho laboral/profissional, sendo uma exigência imposta pelos utilizadores e organismos competentes, nomeadamente através de legislação e regulamentos [Per14].
O Decreto-Lei nº 118/2013, de 20 de Agosto, reúne num único diploma o SCE (Sistema de Certificação Energética dos edifícios) constituído por dois regulamentos, o REH (Regulamento de desempenho Energético dos edifícios de Habitação) e o RECS (Regulamento de desempenho Energético dos edifícios de Comércio e Serviços). Estes regulamentos impõem requisitos ao projeto de novos edifícios e grandes remodelações de forma a salvaguardar a satisfação das condições de conforto térmico nesses edifícios, nomeadamente a QAI (Qualidade do Ar Interior). A QAI é um fator determinante, sendo um conceito bastante abrangente, dependendo de um grande número de fatores, como a temperatura, a humidade relativa, a velocidade do ar, a existência de odores, a concentração de micro-organismos ou poeiras em suspensão no ar, o nível de ruído, a iluminação, entre outros. Estes fatores, são agrupados em quatro grandes áreas: a qualidade do ar, a qualidade acústica, a qualidade da iluminação e a qualidade higrotérmica [Cat10]. A qualidade higrotérmica é uma das preocupações da higrotermia (medição de humidade e temperatura), e tem como objetivos a satisfação higrotérmica habitacional (sensação de bem-estar relativamente à humidade e à temperatura ambiente) e ser também um contributo ativo no incremento da eficiência energética.

Constata-se que o Homem sempre procurou automatizar as suas tarefas e rotinas domesticas com o propósito de aumentar a sua comodidade e a eficiência de recursos. Inicialmente, com recurso a elementos elétricos básicos (e.g. relés, termostatos, etc.), criaram-se automatismos caraterizados por apenas reagirem a estímulos locais e não executarem tarefas em função do processamento de informação. A evolução da eletrónica (em particular dos sistemas baseados em microprocessadores), permitiu que os automatismos iniciais convergissem para sistemas inteligentes, caraterizados por serem programavéis e agirem em função de informação recebida de sensores e estágios de controlo, dando assim origem ao conceito da domótica [Pat09]. A domótica, advém da junção da palavra *"domus"* (casa), com a palavra "robótica", sendo associada à automatização de rotinas e tarefas domésticas [DPG06].

A domótica carateriza-se por ser uma tecnologia ou combinação de tecnologias que permitem a gestão automática de recursos habitacionais, tendo como objetivos e benefícios fundamentais oferecer maior conforto, segurança e racionalização de recursos energéticos [DP04], sendo a higrotermia, uma das componentes com elevada relevância.
As funcionalidades da domótica são frequentemente agrupadas da seguinte forma [KA12]:

- gestão de equipamentos - atividades como apagar/acender, abrir/fechar, ajustar/regular aplicações e dispositivos domésticos como: iluminação, aspiração central, estores, sistemas de rega e atuação remota de eletrodomésticos.

- vigilância e segurança - monitorização e controlo de acessos, vídeo-vigilância, deteção e atuação em situações de emergência como: incêndios, fugas de gás, de água e cenários de intrusão.

- utilização eficaz recursos - racionalização de recursos de sistemas AVAC (Aquecimento, Ventilação e Ar Condicionado), higrotermia, monitorização de consumos de eletricidade, gás e água, com vista a eliminar gastos desnecessários, promovendo a eficiência de recursos energéticos.

O desenvolvimento de soluções domóticas acentuou-se a partir da década de 70. Foram desenvolvidos vários protocolos de comunicação, sendo o X10 o mais amplamente difundido e com elevado sucesso comercial [JN02]. Desde então, a domótica tem acompanhado a evolução e diversas soluções têm surgido. Em geral, as soluções de domótica seguem um de dois paradigmas: o centralizado, e o descentralizado (também designado de distribuído) [DP04].
No paradigma centralizado os sistemas dispõem de uma unidade central de controlo que recebe e processa toda a informação. È normalmente a opção mais económica, pois retira a capacidade de processamento dos diversos dispositivos e centraliza tudo num único dispositivo. A principal desvantagem, é o facto de, se o elemento central se danificar, comprometer a instalação na sua globalidade [DP04].
Por outro lado, o paradigma descentralizado é constituído por diversos dispositivos com processamento inteligente próprio, sendo desta forma flexível e mais imune a falhas, já que, numa eventual falha de um dos dispositivos, apenas se comprometer o funcionamento desse, continuando assim o resto do sistema operacional. Em geral é uma opção mais completa e robusta, mas mais dispendiosa [DP04].
Para a troca de informação entre dispositivos são necessários meios de transmissão, como a rede elétrica, a fibra ótica, os infravermelhos, a radiofrequência, dependendo da(s) tecnologia(s) a usar. Para a comunicação entre vários dispositivos não é apenas necessário interliga-los através de um meio de comunicação, é também necessário que ambos usem o mesmo protocolo de comunicação (indica o formato das mensagens, o conjunto de regras sobre o modo de comunicação entre todos os dispositivos). Esses protocolos são designados de protocolos normalizados e protocolos proprietários [JC 16].
Os protocolos normalizados caracterizam-se pelo facto de serem uma tecnologia aberta a qualquer empresa, assegurando o futuro da tecnologia e a existência de uma maior variedade de produtos, de diferentes fabricantes, que comunicam e interagem entre si.
Por sua vez, os protocolos proprietários caracterizam-se por serem aqueles que são desenvolvidos por uma única empresa e apenas essa empresa, ou outras devidamente licenciadas, fabricam produtos capazes de comunicar entre si. A aposta nestas soluções implica um enorme

risco, pois caso a empresa decida deixar de apostar nesta tecnologia, o cliente fica sem suporte técnico.

Consta-se que, a domótica tem tido um crescimento acentuado muito devido a razões de dimensão tecnológica (maior oferta, escalabilidade, portabilidade, simplicidade de manutenção e utilização das soluções), de dimensão económica (custos mais apelativos) e de dimensão ecológica (incremento da eficiência energética) contudo, a maioria das habitações ainda não estão equipada com estas mais valias.

1.2 Motivação

A crescente preocupação em proteger o meio ambiente tem reunido consensos a nível global. Têm sido aprovados protocolos e tratados por órgãos de soberania com o objetivo de reduzir as emissões de gases de efeito de estufa, promovendo incentivos ao uso de energias alternativas amigas do ambiente. Estas medidas são muito importantes contudo, é necessário mais ambição nos diversos sectores da economia. No caso particular do sector imobiliário, verifica-se que este sector é responsável pelo consumo de uma fatia significativa de recursos nomeadamente, a água, o gás e a eletricidade. É necessário utilizar os recursos de forma mais eficiente e para isso é preciso ter em conta duas situações: desenvolver equipamentos mais eficientes e melhorar a eficiência energética dos edifícios. O desenvolvimento de equipamentos eficientes visa em primeiro lugar que estes consumam menos energia e que contemplem também funções de controlo inteligente do consumo de recursos habitacionais.

Relativamente à eficiência energética dos edifícios verifica-se que as construções mais antigas possuem pontes térmicas (ausência ou isolamento térmico insuficiente), que provocam perdas térmicas e por consequente desconforto térmico (sensação de mau-estar relativamente à humidade e temperatura ambiente). É necessário que se quantifique a humidade e a temperatura para que se apliquem medidas para melhorar o conforto térmico. Esse papel cabe à higrotermia. A higrotermia consiste na medição da humidade e da temperatura e é um importante contributo para o desenvolvimento de soluções para incrementar a eficiência energética. A higrotermia é um suporte muito importante para a domótica. Apesar da domótica constituir um vasto leque de soluções, verifica-se que ainda existem algumas limitações a nível da interligação de diversos dispositivos com as diferentes tecnologias e fabricantes, principalmente nos sistemas tradicionais. De forma a solucionar algumas das limitações anteriormente mencionadas na domótica e em particular nos sistemas higrotérmicos, optou-se como tema de dissertação o desenvolvimento de um sistema higrotérmico de baixo custo com uma rede de sensores distribuídos capazes de operarem nos diversos espaços dos edifícios de forma flexível, utilizando tecnologias *WiFi* e baixo consumo energético.

1.3 Objetivos

Este trabalho visa desenvolver um sistema Higrotérmico para monitorização de Temperatura e Humidade relativa em edifícios no âmbito da Engenharia Civil.
O sistema deverá contemplar uma aplicação desenvolvida numa plataforma móvel (e.g. plataforma Android), e também a utilização de componentes de uma plataforma *open source* (e.g. plataforma Arduino). Através da plataforma *open source* contruir-se-á um dispositivo remoto que hospedará todos os dispositivos necessários nomeadamente:

- *Sensores* - para obtenção das variáveis a adquirir: Temperatura e Humidade relativa.

- *Data Logger* - memória não volátil para guardar os parâmetros de interesse num ficheiro do tipo .csv[1], nomeadamente o *timestamp*[2] e o valor das variáveis adquiridas seguindo o seguinte formato: Data;Hora;Temperatura;Humidade relativa.

- Interface para comunicação *wireless* - para interação entre a plataforma móvel e o dispositivo remoto.

O dispositivo remoto deverá ser capaz de adquirir e registar durante o tempo de aquisição (valor máximo de 1 minuto e amostragens com cadência de um segundo) o *timestamp* e o valor das variáveis a adquirir para o ficheiro .csv. Após a aquisição e registo das variáveis, o dispositivo remoto deve entrar em modo *Standby* (não adquire nem regista as variáveis) durante o tempo *Standby* (em minutos) pré-estabelecido pelo utilizador, que poderá ser 60, 30, 20 ou 10. Decorrido o tempo *Standby*, o dispositivo remoto deve retomar a aquisição e registo da Temperatura e Humidade relativa durante o tempo de aquisição, e repetir este ciclo até ser interrompido pelo utilizador. O dispositivo remoto deve também fornecer (sempre que o utilizador solicite) o valor em tempo real da Temperatura e Humidade relativa (sem interferir no processo de registo das variáveis ou período *Standby*).

A aplicação móvel terá como função permitir ao utilizador interagir com o dispositivo remoto em redes de área local *(LAN - Local Area Network*, via *WiFi)* enviando comandos nomeadamente, o tempo de *Standby* pretendido, solicitar o valor em tempo real da Temperatura e Humidade relativa e suspender as atividades do sistema remoto (suspensão da aquisição e registo das variáveis ou o tempo de *Standby*).

Pretende-se também um programa que permita a análise das variáveis adquiridas, devendo ser desenvolvido num *software* voltado para o cálculo numérico (e.g. MatLab). Este programa permitirá ao utilizador visualizar as variáveis (guardadas no ficheiro .csv) em função do tempo, que poderá ser dia, mês ou ano. O programa deverá também estar preparado para lidar com vários tipos de erros, como a tentativa de executar um ficheiro inadequado, ausência de parâmetros no ficheiro, entre outros, indicando ao utilizador qual a razão desse mesmo erro.

[1] CSV - *Comma Separated Values* - o separador utilizado entre variáveis será o ';'
[2] *Timestamp* - marca temporal (data e hora) da ocorrência de evento

1.4 Estrutura da dissertação

Além deste Capítulo, esta dissertação encontra-se organizada em mais cinco capítulos e quatro apêndices.

No Capítulo 2 (Conceitos fundamentais), é feita uma abordagem é higrotermia, em particular aos conceitos de temperatura, humidade relativa e equipamentos de medição. São apresentadas também algumas soluções existentes em higrotermia. Para terminar o capítulo, são apresentados e caracterizados os protocolos de comunicação mais representativos, utilizados na domótica.

No Capítulo 3 (Arquitetura do sistema), é feita a descrição global da arquitetura adotada.

No Capítulo 4 (Implementação), é demonstrado e explicado como foi implementada a solução desenvolvida para cumprir os objetivos propostos, apresentando algoritmos, diagramas de blocos e fluxogramas, de acordo com as particularidades de cada situação.

No Capítulo 5 (Demonstração do Sistema e Resultados), apresentam-se os resultados do trabalho concebido e a descrição (na ótica de utilizador) da utilização da solução global desenvolvida.

Por sua vez no Capítulo 6, (Conclusão), é feita uma avaliação dos resultados obtidos, apresentando-se também sugestões de possíveis melhorias futuras.

No Apêndice A (Sinopse - Plataforma Android) apresenta-se uma sinopse da plataforma Android e demonstram-se os componentes utilizados para desenvolver a aplicação.

No Apêndice B (Ferramentas, compilação, execução e transferência de programas para *Smartphones Android*) são apresentadas ferramentas para desenvolver aplicações para a plataforma móvel Android, assim como a compilação, transferência e execução de programas.

No Apêndice C (Compilação, execução e transferência de programas para a plataforma Arduino) é demonstrado como se compila, transfere e executam programas na plataforma *open source* Arduino.

No Apêndice D (Considerações antes da execução do programa MatLab) são mencionadas recomendações a ter em conta antes de executar o programa MatLab.

Capítulo 2

Conceitos fundamentais

Neste capítulo, procurar-se-á fazer a revisão de aspetos fundamentais em higrotermia. Na secção 2.1, é feita uma abordagem ao conceito de higrotermia, sendo o foco principal a temperatura e a humidade relativa, apresentando conceitos fundamentais, instrumentos de medida e algumas soluções comerciais. Na secção 2.2, apresentam-se os protocolos de comunicação mais representativos, utilizados na domótica.

2.1 Higrotermia

A higrotermia é o conceito geral para medição de humidade e temperatura sendo esta dividida em higrometria e termometria. A higrometria consiste na medição de humidade e a termometria na medição de temperatura.

São apresentados os conceitos e equipamentos de medição mais importantes sobre humidade relativa (subsecção 2.1.1) e temperatura (subsecção 2.1.2). Na subsecção 2.1.3 são anunciadas algumas soluções para obtenção dessas grandezas.

2.1.1 Humidade Relativa

O conceito de Hr (Humidade relativa), expressa o conteúdo de vapor de água na atmosfera [WW15] e traduz-se de acordo com a equação 2.1

$$Hr(\%) = \frac{P(H_2O)}{P"(H_2O)} * 100\% \tag{2.1}$$

onde:

- $P(\mathrm{H_2O})$ - pressão parcial de vapor de água numa mistura de gases.
  Como o vapor de água é um gás, a lei de Dalton relativa ás pressões parciais[1] [HWB07] é aplicável, sendo então o valor normal da pressão atmosférica 1.013bar (varia com a altitude e temperatura) a soma das pressões parciais dos constituintes do ar[2].

[1] A pressão total de um sistema é a soma das pressões parciais de todas as partes, ou seja, as pressões exercidas por cada um dos componentes de uma mistura gasosa à mesma temperatura e volume.

[2] Composto maioritariamente por azoto, oxigénio, árgon e por outros em quantidade residual.

- $P"(H_2O)$ - pressão de vapor de saturação da temperatura da mistura de gases. Quanto mais alta a temperatura, mais alta a pressão do vapor de saturação e mais vapor de água o ar consegue conter. Assim, o ar quente tem maior capacidade de conter vapor de água do que o ar frio.

Conforto higrotérmico implica que a Hr deva estar compreendida entre 40% a 70%.

2.1.1.1 Higrometria

Como referido anteriormente, a higrometria consiste na medição da humidade, sendo os equipamentos de medição denominados de higrómetros. Os mais comuns são os seguintes [WW15] [Bel11] :

- Sensor de humidade relativa - higrómetro baseado num componente eletrónico que absorve o vapor de água de acordo com a humidade do ar, alterando a sua impedância.

- Higrómetro de condensação - fundamenta-se no facto de o vapor de água condensar sobre superfícies frias. Mantendo fria uma cápsula e fazendo passar o vapor de água por esta, o vapor condensa-se. Efetuando a medição da temperatura a que ocorre condensação e conhecendo a temperatura ambiente, é possível determinar a humidade.

- Higrómetro mecânico - utiliza materiais orgânicos, tais como cabelo humano que se dilata ou contrai de acordo com as variações da humidade atmosférica. Essas dilatações ou contrações são usadas para mover um ponteiro, em que a variação na resistência de uma substância higroscópica é usada como indicação de humidade.

- Psicrómetro - aparelho que contém dois termómetros idênticos colocados um ao lado do outro, que irão servir para avaliar a quantidade de vapor de água encontrada no ar. Enquanto um deles trabalha com o bulbo seco, o outro trabalha com o bulbo húmido. Esse aparelho é muito utilizado para a determinação do ponto de orvalho e da humidade relativa do ar.

2.1.2 Temperatura

È uma grandeza física que mede a energia cinética associada ao movimento desordenado dos átomos e moléculas. As unidades mais utilizadas são o grau Celsius (^oC), o grau Fahrenheit (^oF) e o grau Kelvin (^oK), sendo esta última a unidade SI (Sistema Internacional de unidades). A temperatura está relacionada com o calor contudo, são conceitos diferentes [SE08]. O calor é a energia térmica (medida em Joule) em movimento, ou seja, a energia cinética proveniente da movimentação dos átomos ou moléculas de um corpo para o outro por diferença de temperatura. As formas de transferência de calor são as seguintes: condução (sólidos), convecção (líquidos e gases) e radiação (sem contacto físico) [SE08].

2.1.2.1 Termometria

A termometria como referido anteriormente, consiste na medição de temperatura. Contudo, para altas temperaturas designa-se de pirometria, para baixas temperaturas designa-se de criometria.

Os equipamentos de medição chamam-se termómetros, sendo os mais comuns os seguintes [Ana07]:

- Termómetro clínico - utilizado para medir a temperatura corporal. Existem os tradicionais que usam o princípio de dilatação de líquidos (mercúrio) e os digitais.
- Termómetro de máxima e de mínima - utilizado na meteorologia, indica a máxima e a mínima temperatura atingida no dia.
- Termómetro a gás - são muito precisos, geralmente são usados para calibrar outros termómetros.
- Pirómetro de infravermelhos - mede a intensidade da radiação proveniente do objeto. È utilizado para medir a temperatura de metais incandescentes, fornalhas e até estrelas.
- Termómetro de lámina bimetálica - constituído por duas lâminas de metais diferentes soldadas que, ao serem aquecidas, dilatam-se. Por serem metais diferentes, um dilata mais que o outro e encurva a lâmina.
- Termopar - composto por dois fios de metais diferentes soldados nas extremidades e, quando aquecidos, produzem uma corrente elétrica que depende da temperatura.
- Termístor - semicondutor, carateriza-se pela variação da sua resistência elétrica com a temperatura. Existem 2 tipos, o NTC (*Negative Temperature Coeficient*, a resistência diminui com o aumento da temperatura) e PTC (*Positive Temperature Coeficient*, a resistência aumenta com o aumento da temperatura).

Como referido anteriormente, o conforto higrotérmico é cada vez mais uma exigência imposta pelos utilizadores e organismos competentes. Mediante estudos de higrotermia, evidenciam-se vantagens para incrementar a presença da higrotermia em sistemas domóticos. Diversas soluções tem surgido no mercado, umas mais abrangentes, outras menos abrangentes, dependendo da particularidade de cada situação. Contudo, todas as soluções visam o aumento do conforto higrotérmico e a eficiência de recursos energéticos.

Na secção seguinte, são apresentadas soluções de higrotermia em sistemas domóticos.

2.1.3 Higrotermia em sistemas domóticos

Nesta subsecção, são apresentadas algumas soluções para domótica que permitem (entre outras possibilidades) a medição de humidade e/ou temperatura, tendo sido realizadas no âmbito de trabalhos de investigação.

Design and Implementation of a WiFi Based Home Automation System

Neste artigo [EH12], os autores apresentam uma solução ampla, pois permite monitorizar e controlar uma série de parâmetros, nomeadamente a temperatura e humidade, deteção de movimentos, deteção de fumo, controlo de iluminação e vídeo-vigilância. A figura 2.1 demonstra o sistema proposto.

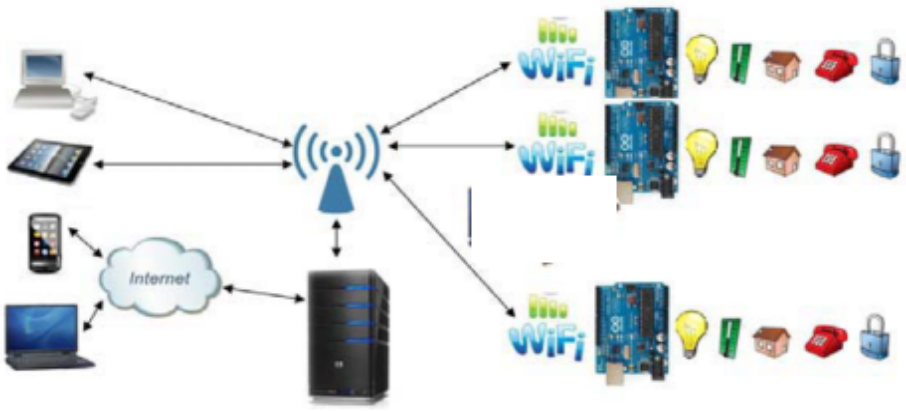


Figura 2.1: *WiFi Based Home Automation System* [EH12]

O sistema é constituído por um computador comum que age como *web server*. O utilizador, interage com o sistema através de dispositivos fixos ou móveis, desde que o *web browser*, suporte tecnologia asp.net. O micro-controlador utilizado foi o Arduino e contém os sensores necessários que satisfazem o problema. A solução carateriza-se essencialmente por ter uma interface *user friendly*, baixo custo e escalável.

Ubiquitous Smart Home System Using Android Application

Nesta publicação [Kum14], apresenta-se um sistema *low cost* de um *standalone* flexível. Baseia-se numa aplicação Android e num Arduino com *Ethernet shield*. Com a *Ethernet shield* e o Arduino obtiveram um *web server*, permitindo que remotamente o utilizador através da aplicação Android controle e monitorize este *setup*. A arquitetura da solução, demonstra-se na figura 2.2.

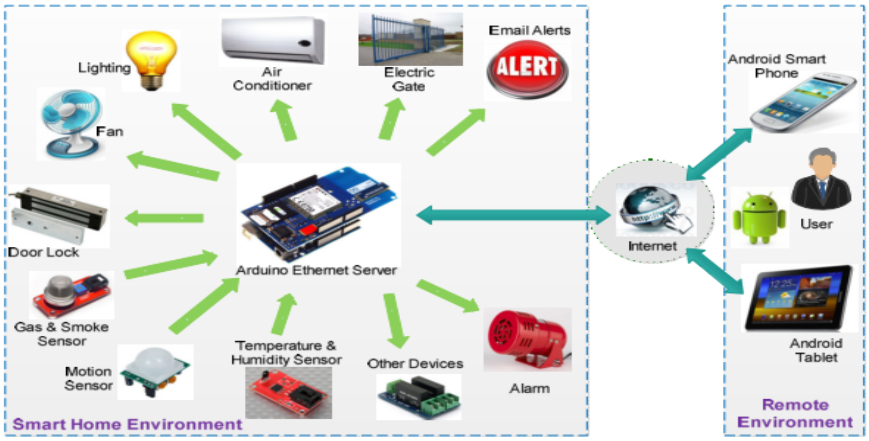


Figura 2.2: *Ubiquitous Smart Home System* [Kum14]

Permite o controlo de iluminação, monitorização de humidade, temperatura, deteção de intrusão e fumo, portas/janelas, ar-condicionado, entre outros. Em caso de intrusão, o sistema ativa sirenes e notifica o proprietário através de *email*.

Na secção seguinte são apresentados e analisados alguns dos mais preponderantes protocolos de comunicação utilizados no mercado da domótica.

2.2 Protocolos de comunicação

Para haver comunicação, é necessário a existência de uma fonte (emissor) de uma mensagem e de um ou vários recetores. Um protocolo de comunicação descreve formalmente o formato da mensagem a ser transmitida e as respetivas regras de transmissão relativamente à sintaxe, semântica e sincronização da comunicação. Dessa forma, uma aplicação domótica requer a instalação de sistemas e protocolos apropriados, que permitam garantir a conetividade e integração entre as múltiplas funcionalidades dos dispositivos. Verifica-se que, qualquer sistema domótico é constituído por uma ou várias redes de comunicação, com fios *(wired)*, sem fios *(wireless)* ou ambas (híbrida), que interliga os diversos dispositivos que se pretendem controlar e monitorizar.

Nesta secção, apresentam-se os protocolos X10 (subsecção 2.2.1), o IEEE 802.11 (subsecção 2.2.2), o IEEE 8002.15.1 (subsecção 2.2.3), o Z-Wave (subsecção 2.2.4), o CEBus (subsecção 2.2.6), o KNX (subsecção 2.2.7), o Insteon (subsecção 2.2.8) e o LonWorks (subsecção 2.2.9).

2.2.1 X10

È um protocolo normalizado que possibilita a sua instalação em edificações já existentes, sem necessidade de refazer a instalação elétrica, sendo este um aspeto chave desta tecnologia e a sua maior vantagem face a outras. Os preços baixos e a facilidade na instalação contribuíram para o seu rápido sucesso comercial nos EUA e UE [JN02] [DP04].

Utiliza a rede elétrica existente monofásica ou trifásica, como meio de transmissão de sinais/comandos, principalmente para ligar/desligar equipamentos, aumentar/diminuir brilhos em iluminação, podendo também recorrer a telecomandos de radiofrequência [JN02]. Utiliza dispositivos emissores e recetores maioritariamente *plug&play*.

2.2.2 IEEE 802.11

Também designado vulgarmente por *Wi-Fi (Wireless Fidelity)*, designa o padrão para redes de área local sem fio WLANs *(Wireless Local Area Network)* [Far12]. As WLANs tornaram-se muito populares no fornecimento de conectividade IP em pequenos ambientes, e desde aí tem tido um crescimento enorme. Hoje em dia tem um papel crucial nas redes de computadores. Existem duas normas dominantes a IEEE 802.11 e a HiperLAN.

2.2.3 IEEE 802.15.1

O padrão IEEE 802.15.1 é vulgarmente designado por *Bluetooth*. Permite troca de dados entre dispositivos que estão a uma curta distância (máximo até 100 metros), sendo que esta varia com a potência de transmissão. A transmissão é feita através de ondas curtas de rádio na banda ISM[3] desde 2400 a 2480 MHz. Esta tecnologia veio substituir os cabos elétricos dos periféricos dos computadores tais como ratos, teclados, entre outros [Far12]. Hoje em dia, esta tecnologia devido ao seu sucesso conseguiu expandir-se estando disponível em grande parte dos dispositivos eletrónicos, como *smartphones* e presente em diversas aplicações, nomeadamente a domótica.

2.2.4 Z-Wave

Nesta tecnologia a comunicação é feita via RF *(Radio Frequency)*. O sinal RF usado na UE é de 868 MHz e 908 MHz nos USA [OAeR+14]. A transmissão de dados é feita à taxa de 9,6 kb/s e 40 kb/s. Atualmente 2,4 GHz e 200 kb/s são já uma realidade. Não foi concebido para transmitir grandes quantidades de dados ou qualquer tipo de *streaming*. O principal objetivo é a transmissão de mensagens de forma fiável a partir de uma unidade de controlo para um ou mais nós na rede [JFR06]. Os dispositivos Z-Wave classificam-se essencialmente em dois tipos: unidades de controlo e nós escravos [GP10].
A(s) unidade(s) de controlo são os nós da rede que controlam e enviam comandos para os outros nós designados de nós escravos, que respondem e executam os comandos. Os nós escravos nóo podem transmitir informação diretamente para outros nós, a menos que tenha sido autorizado pela unidade de controlo [JFR06]. A unidade de controlo contém uma tabela de encaminhamento *(routing table)* de todos os nós da rede.

[3]Industrial, Scientific and Medical - bandas de rádio-frequência reservadas internacionalmente para fins industriais, médicos e científicos (isentas de licenciamento).

2.2.5 ZigBee

Assenta no padrão IEEE 802.15.45, para redes de área pessoal sem fio WPAN *(Wireless Personal Area network)*. É apropriado para aplicações que envolvam dispositivos remotos alimentados por baterias e com capacidade de hospedar milhares de dispositivos (teoricamente 65.536). Simples e barato, de baixo consumo de energia e com taxas de transferência de dados de 20 kb/s, 40 kb/s e 250 kbps. [GP10]. Três topologias de rede diferentes podem ser implementadas, designando-se por estrela, árvore e malha.
Os dispositivos ZigBee são caracterizados em dois tipos: lógicos e físicos [MDS13]. Os dispositivos lógicos classificam-se em coordenadores, *routers* e dispositivos finais *(end devices)*. Por sua vez, os dispositivos do tipo físico são classificados em FFD *(full function devices)* e RFD *(reduced function devices)* [OAeR$^+$14]. Os dispositivos FFD podem comunicar com qualquer dispositivo da rede podendo assumir o papel de coordenador, *router* ou mesmo de dispositivo final. São dispositivos complexos de implementar. Já os dispositivos RFD só se podem associar com dispositivos FFD e geralmente assumem o papel de dispositivo final podendo ser interruptores de iluminação, *dimmers*, controlo de relés, sensores, entre outros. São dispositivos com uma implementação mais simples [MDS13]. Os dispositivos tem alcance entre os 10 e os 100 (metros) e operam na banda ISM.

2.2.6 CEBus

Consumer Electronics Bus é atualmente um protocolo aberto e normalizado. Opera em redes P2P[4], sendo cada dispositivo considerado um nó. Foi desenvolvido para dar resposta à inexistência de uma forma padronizada de comunicação entre os diversos dispositivos (principalmente os de fabricantes diferentes), designando-se de norma ANSI/EIA-600. São vários os meios de transmissão que disponibiliza [JN02]:

- linha de energia eléctrica *(Power Line)*.
- cabo par entrançado *(Twisted Pair)*.
- cabo coaxial (preferencialmente para áudio e vídeo).
- rádio-frequência (RF).
- infra-vermelhos (IR).
- fibra ótica.

Através destes meios, é possível criar uma solução domótica, para infraestruturas já existentes sem necessidade de refazer a instalação elétrica. A interligação entre os diferentes meios de comunicação é feita com recurso a *routers* e *bridges*. Para evitar colisões de dados, CEBus utiliza o CSMA/CDCR *(Carrier Sense Multiple Access /Collision Detection and Resolution)*, impondo que o nó aguarde que o meio de comunicação esteja livre [JN02].

[4]Peer-to-peer (ponto-a-ponto) - cada um dos pontos ou nós da rede funciona tanto como cliente, ou como servidor.

2.2.7 KNX

O protocolo é uma iniciativa baseada em três protocolos precedentes do EIB *(European Installation Bus)* o BatiBUS e o EHS *(European Home System)*, com o objetivo de conceber um protocolo padrão que respondesse aos requisitos das instalações de domótica. É uma tecnologia aberta e descentralizada que garante o funcionamento de produtos de diferentes fabricantes. O KNX possibilita a construção de um sistema modular reconfigurável, que permite em qualquer altura do desenvolvimento do sistema ou até depois de este estar implementado, acrescentar mais dispositivos e funções ao sistema [Ass09]. Suporta quatro meios de comunicação distintos:

- linha de energia eléctrica *(Power Line)*.
- cabo par entrançado *(Twisted Pair)*.
- cabo coaxial.
- IP/Ethernet.

2.2.8 Insteon

Protocolo versátil a nível da comunicação entre dispositivos, sendo possível via radio-frequência (Insteon RF) ou via rede de energia elétrica (Insteon PL) ou ambos conjuntamente, designado-se de tecnologia de rede em dupla malha [GP10]. É robusto, todos os dispositivos podem transmitir, receber ou agir como repetidores. Se o(s) dispositivo(s) que se pretende(m) comunicar não se encontrar(em) na área de alcance do emissor, é adotada uma estratégia de multi-salto [OAeR+14]. O alcance do sinal é de aproximadamente 45 metros, sendo este significativamente reduzido pela presença de paredes ou outros bloqueadores físicos. Usa comandos simples, é muito fiável, pois qualquer mensagem é confirmada quando recebida com sucesso e caso haja algum erro esta é novamente reenviada. Os dispositivos RF e PL comunicam através do envio de mensagens com comprimento fixo. Existem dois tipos de mensagem Insteon: *Standard Message*, e *Extended*. As *Standard Message* são usadas para comando e controlo direto dos dispositivos. As *Extended Message* podem ser usadas para *downloads, uploads*, encriptação e aplicações avançadas [Tec07].

2.2.9 LonWorks

Tecnologia de barramento de campo *(fieldbus)* desenvolvida pela *Echelon Corporation*. Configura-se como uma rede de dispositivos inteligentes de controlo (nós), que estabelecem a comunicação por meio do protocolo comum denominado Lontalk [DP04] [YB13]. A comunicação é feita de forma distribuída, entre pares (peer-to-peer) que possibilita que os nós individuais da rede comuniquem com os sensores, atuadores e demais dispositivos, sem necessidade de possuir um controle central, pois cada nó da rede contém inteligência. Os transcetores providenciam interface elétrica sobre o canal de comunicação e estão disponíveis para uma variedade de meios de comunicação, incluindo um único cabo de par trançado, linha de energia (PL), de rádio frequência (RF), infra-vermelho, fibra ótica e cabo coaxial [DP04].

Apresenta-se de seguida o capítulo relativo à arquitetura da solução desenvolvida que vai ao encontro dos objetivos propostos 1.3.

Capítulo 3

Arquitetura do Sistema

Neste capítulo é apresentada a especificação do sistema e a sua arquitetura, sendo esta caracterizada por ser baseada no modelo Cliente/Servidor.

Na secção 3.1 caracteriza-se de forma geral o modelo Cliente/Servidor, apresentando particularidades do Cliente e características do Servidor.

A secção 3.2 especifica o sistema e ilustra os seus constituintes descrevendo, a composição do Servidor (subsecção 3.2.1) e as funcionalidades do Cliente (subsecção 3.2.2).

Para concluir o capítulo, explicita-se a interação entre Cliente/Servidor (subsecção 3.2.3).

3.1 Modelo Cliente/Servidor

O modelo Cliente/Servidor consiste numa estrutura de aplicação distribuída, sendo constituída por 3 entidades distintas fundamentais:

- Servidor - *host(s)* que compartilha(m) serviços com Clientes *(Response)*.
- Cliente - *software* instalado em dispositivo(s) que permite ao utilizador solicitar serviços ao(s) Servidor(es) *(Request)*.
- Meio de comunicação - define qual o meio de comunicação entre Cliente e Servidor (*wired, wireless*, ou híbrido).

A Figura 3.1 exemplifica a arquitetura do modelo Cliente/Servidor. Vários Clientes podem aceder aos serviços do Servidor. O Servidor aguarda sempre por *Requests* vindos do Cliente, e fornece o *Response* em conformidade. O Cliente pode ser qualquer dispositivo, computador, *smartphone, tablet*, etc.
Geralmente os Clientes e os Servidores comunicam através de uma rede de computadores em computadores/máquinas distintas, mas em certas aplicações, tanto o Cliente como o Servidor podem residir na mesma máquina.

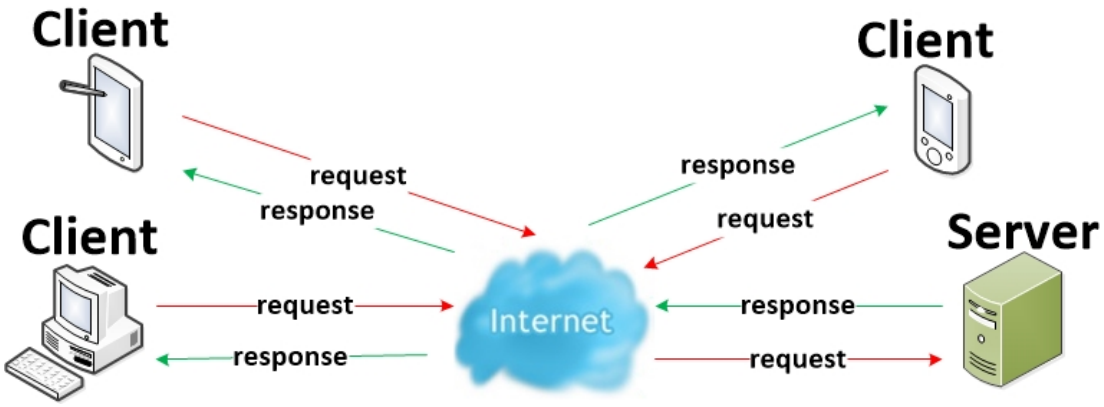


Figura 3.1: Arquitetura - Modelo Cliente/Servidor

3.2 Especificação do sistema

Como referido anteriormente, o modelo Cliente/Servidor permite que Servidores providenciem serviços *(Responses)* sempre que Clientes solicitem *(Requests)*. O Cliente é uma aplicação desenvolvida numa plataforma móvel e o Servidor um dispositivo construído com recursos duma plataforma *open source*. A comunicação entre Servidor e Cliente é feita exclusivamente através da rede *WiFi*, padrão IEEE 802.11 via UDP *(User Datagram Protocol)*, devido ao facto desta rede ser omnipresente e de proporcionar ao utilizador mobilidade. Tanto o Cliente como o Servidor comunicam unicamente dentro da mesma LAN *(Local Area Network)*, podendo o sistema ser constituído por "n" Servidores, assim como "m" Clientes. A Figura 3.2, ilustra a arquitetura da solução.

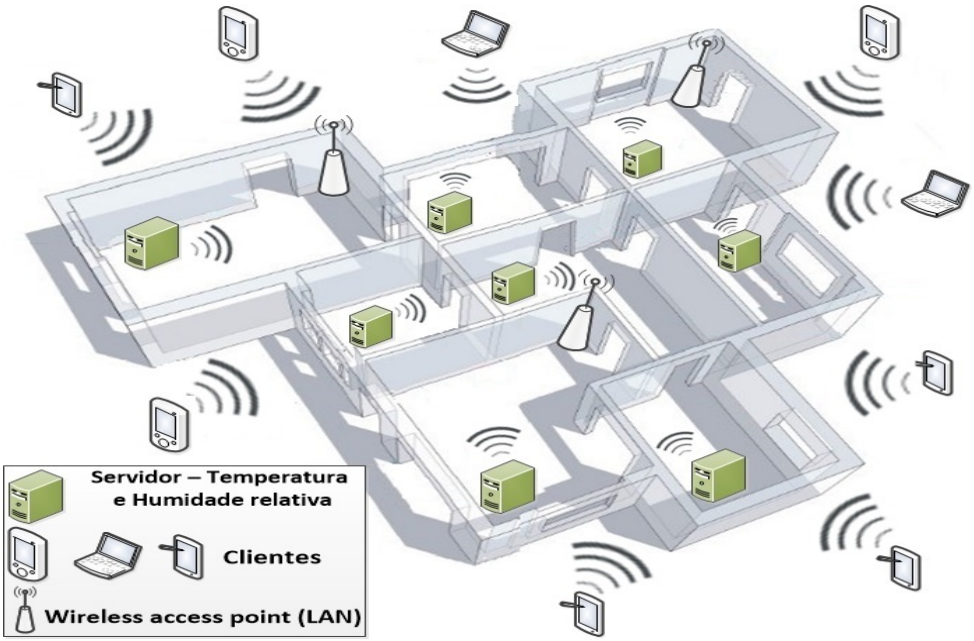


Figura 3.2: Arquitetura do sistema - modelo Cliente/Servidor

3.2.1 Servidor

Este elemento do sistema, designado de Servidor tem como propósito adquirir e registar os valores associados às variáveis Temperatura e Humidade relativa, onde quer que este esteja instalado, bem como envia-las ao Cliente sempre que sejam requeridas. Na Figura 3.3, ilustram-se os seus constituintes.

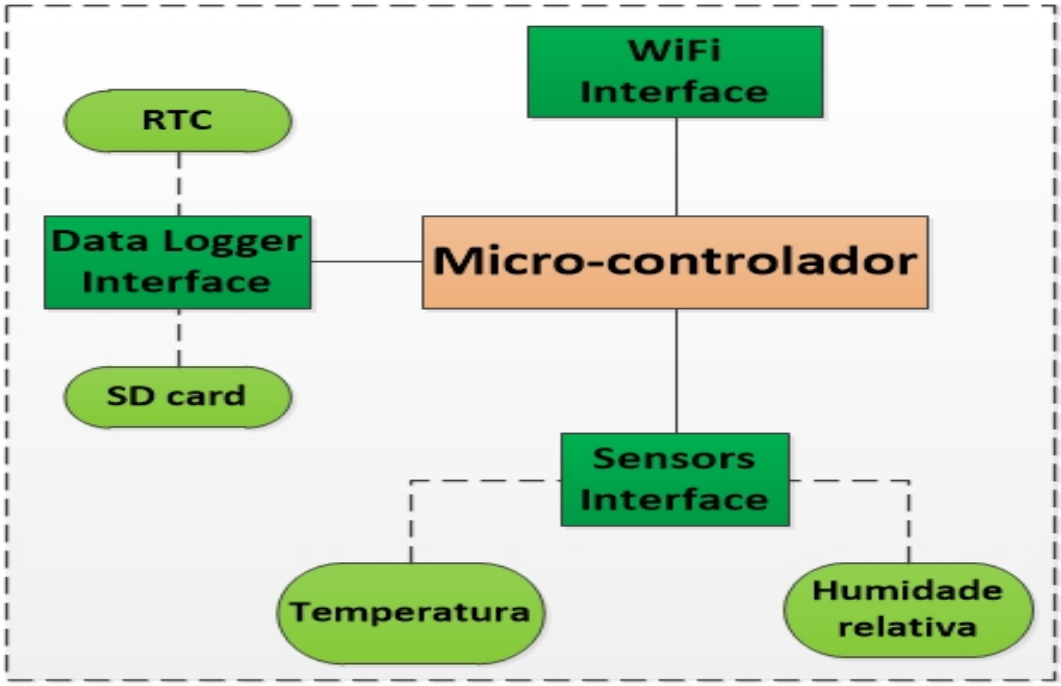


Figura 3.3: Constituintes do Servidor

Conforme se verifica na Figura 3.3, o Servidor é composto por um micro-controlador (responsável por realizar o processamento de dados), e por várias interfaces externas, nomeadamente:

- Interface *WiFi* - para conexão com o(s) *access point* da rede *WiFi*.
- Interface Sensors - para a aquisição das variáveis Temperatura e Humidade relativa.
- Interface *Data Logger* - constituída por um RTC (*Real Time Clock*, para indicação do *timestamp*, formato `DD/MM/YYYY;HH:MM:SS`) e por um cartão SD.

A função do *Data Logger* é registar o *timestamp* e o valor das variáveis num ficheiro denominado de *"file_base.csv"*, seguindo o formato: `Timestamp;Temperatura;Humidade relativa`. O Servidor iniciará as suas funções após indicação do Cliente. Cada Servidor identifica-se e caracteriza-se por um ***IP Address***, um ***Port number*** e ***Place*** (local onde este está a operar).

3.2.2 Cliente

O outro componente do sistema designado de Cliente, dispõe de um conjunto de características e funcionalidades que permitem ao utilizador (através do envio de comandos) interagir com Servidores. Na Figura 3.4 demonstra-se os seus principais constituintes.

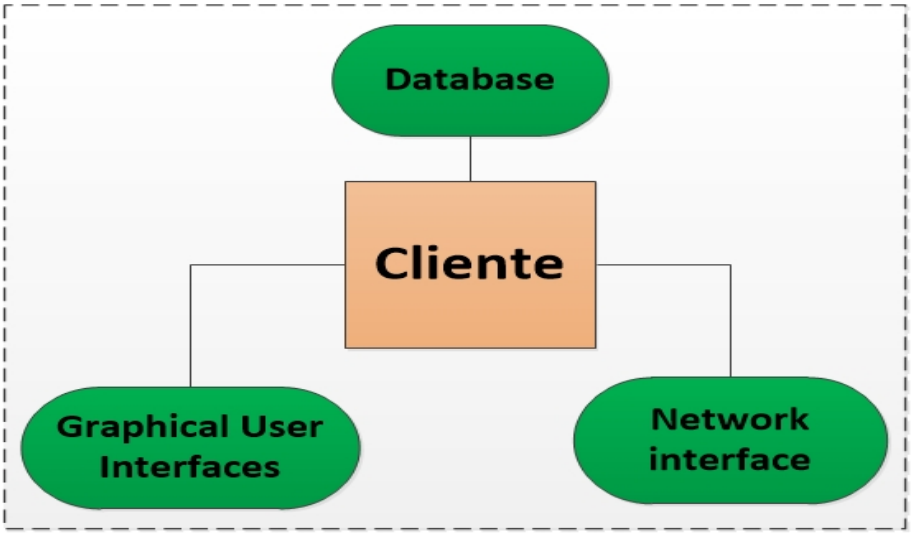


Figura 3.4: Constituintes do Cliente

O Cliente é constituído por várias GUIs *(Graphical User Interfaces)*, que permitem ao utilizador navegar nas várias opções da aplicação desenvolvida na plataforma móvel.
Além disso, e como já referido, o Cliente encontra-se habilitado a comunicar (através de *WiFi*) com Servidores recorrendo a uma *network interface*.
Por outro lado, o Cliente poderá interagir com vários Servidores. Para isso, o utilizador deve conhecer o ***IP Address*** e o ***Port number*** do Servidor para o qual deseja que o Cliente se conecte.
Para tornar a tarefa mais automática e cómoda ao utilizador (tendo em conta que poderão existir vários servidores), capacitou-se o Cliente com uma base de dados SQLite. Esta base de dados permite guardar vários Servidores, sendo que cada um contém:

- ***IP Address*** - identifica o Servidor na rede *WiFi* onde este esteja conectado.
- ***Port number*** - identificador local da porta usada pelo Servidor na comunicação.
- ***Place*** - tem como única função relembrar o utilizador da localização de cada Servidor (não é importante para o processo de comunicação).

Desta forma, o Cliente fica com um grupo de servidores guardado na base de dados, permitindo ao utilizador selecionar o Servidor para o qual se pretende conectar de uma forma prática.

3.2.3 Interação Cliente/Servidor

Numa fase inicial, o Servidor encontra-se em espera, e quando o Cliente solicitar o Servidor responde em conformidade.
Como se verifica na Figura 3.5, o Cliente pode solicitar (por qualquer ordem) ao Servidor o valor em tempo real das variáveis Temperatura e Humidade (*Get Sensors Values*), enviar o tempo (em minutos) que o Servidor vai ficar em *Standby* (60, 30, 20, 10) e suspender todas as suas atividades (*Stop Server*). O Servidor recebe o pedido e responde em concordância.

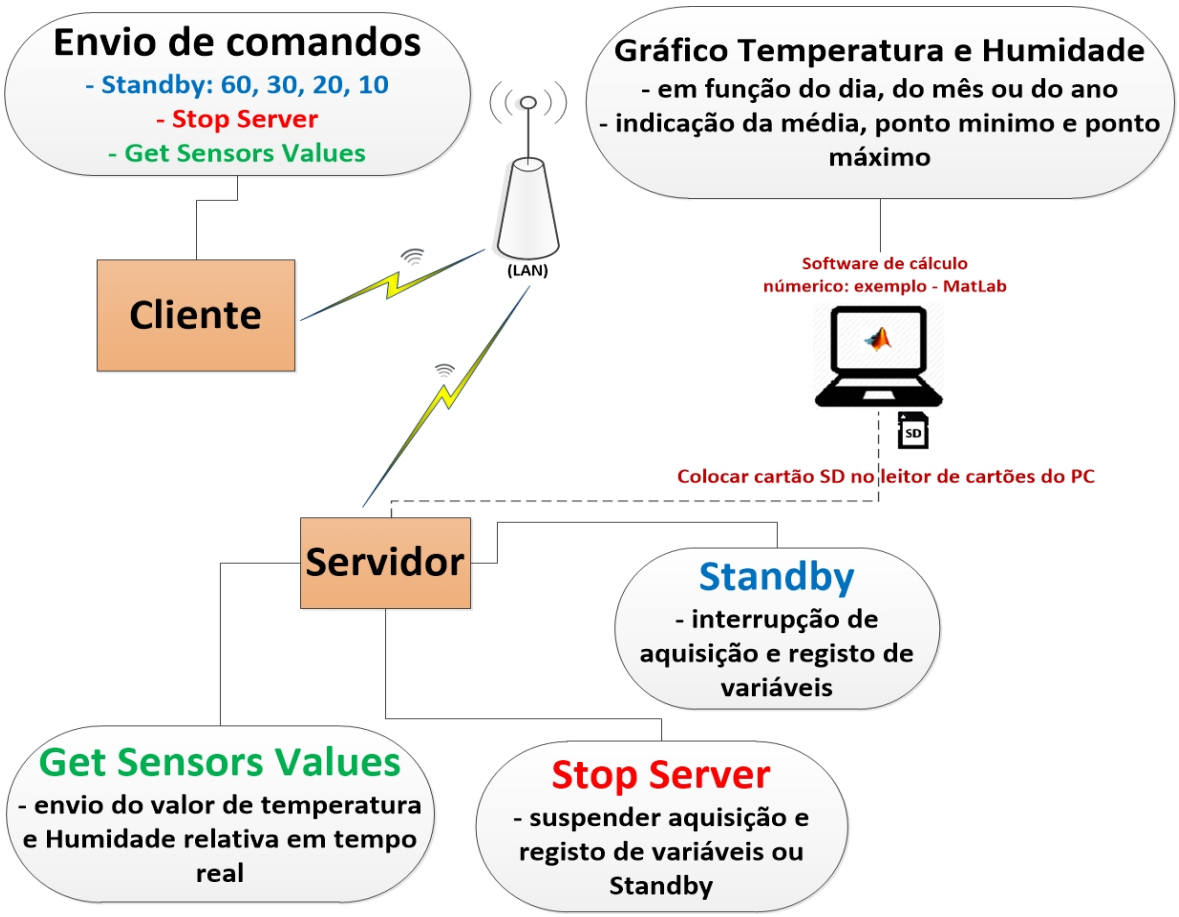


Figura 3.5: Arquitetura do sistema - interação Cliente/Servidor

Para qualquer comando é sempre enviado do Servidor para o Cliente um *acknowledgement*, indicando que o Servidor recebeu o comando corretamente. Caso não haja *acknowledgement* significa que houve algum problema (e.g: Servidor inexistente, problemas de ligação à *internet*, etc.) e nesse caso o Cliente informa o utilizador que o envio do comando não foi bem sucedido. Esta situação verifica-se após um *timeout* de 3 segundos (tempo por omissão, mas configurável), ou seja, após o Cliente enviar comando, se decorridos 3 segundos não receber *acknowledgement* significa que há alguma anomalia.

De modo a complementar a descrição anterior, ilustra-se nas Figuras 3.6, 3.7 e 3.8 cada um dos *Requests* do Cliente, assim como *acknowledgements* e fluxo de dados *(Responses)* do Servidor, durante uma sessão de comunicação UDP.
A Figura 3.6, demonstra o processo para o *Request* *Get Sensors Values*.

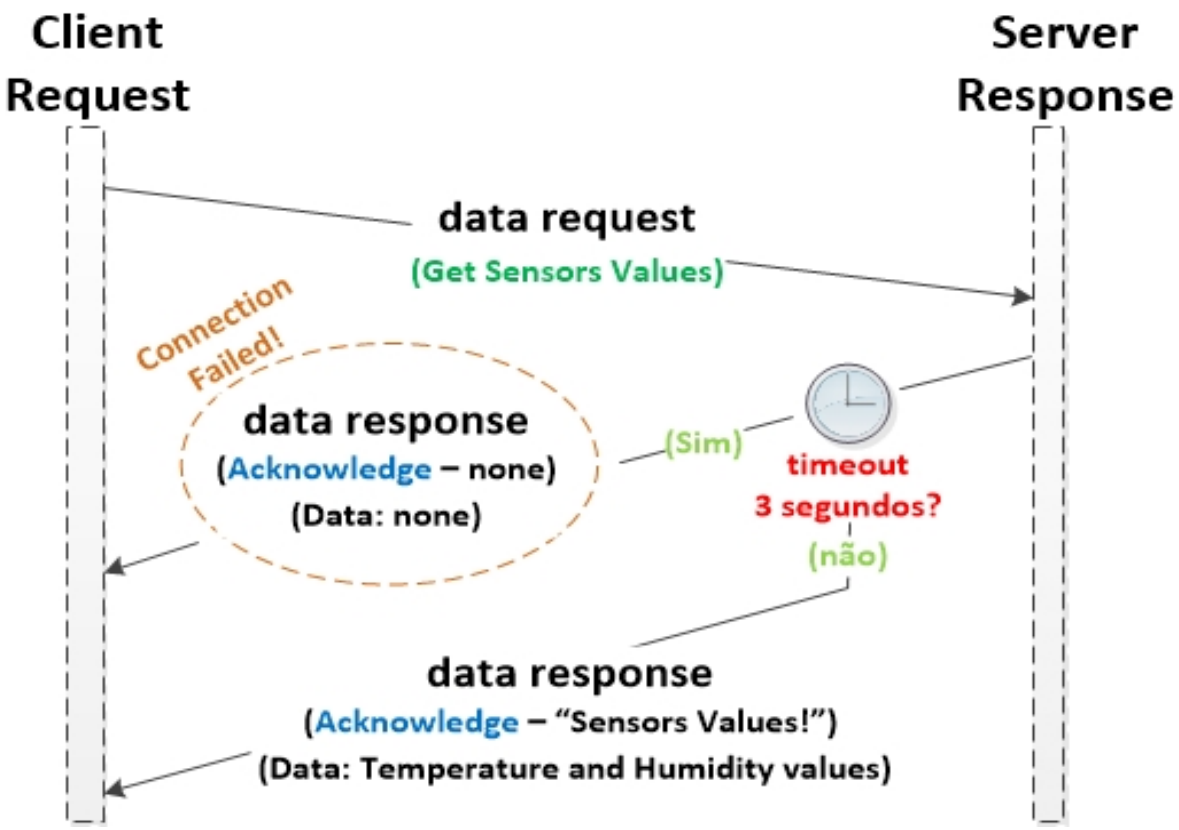


Figura 3.6: *Client Request (Get Sensors Values)* e *Server Responses*

Conforme se verifica na Figura 3.6, o Cliente faz *Request* ao Server. Nesse instante, o Cliente inicia o processo de *timeout*. Se o *timeout* estipulado for atingido (neste caso 3 segundos), significa que a comunicação não teve sucesso, sendo o utilizador notificado (na respetiva GUI) com a seguinte mensagem: **"Connection Failed!"**. Por outro lado, não ocorrendo o *timeout* o Server responde em conformidade, através do envio do *data response (Acknowledge e Data)*. A descrição anterior é também válida para as Figuras 3.7 e 3.8, pois estas são muito semelhantes contudo, as únicas diferenças residem no *data request*, e no *data response*.

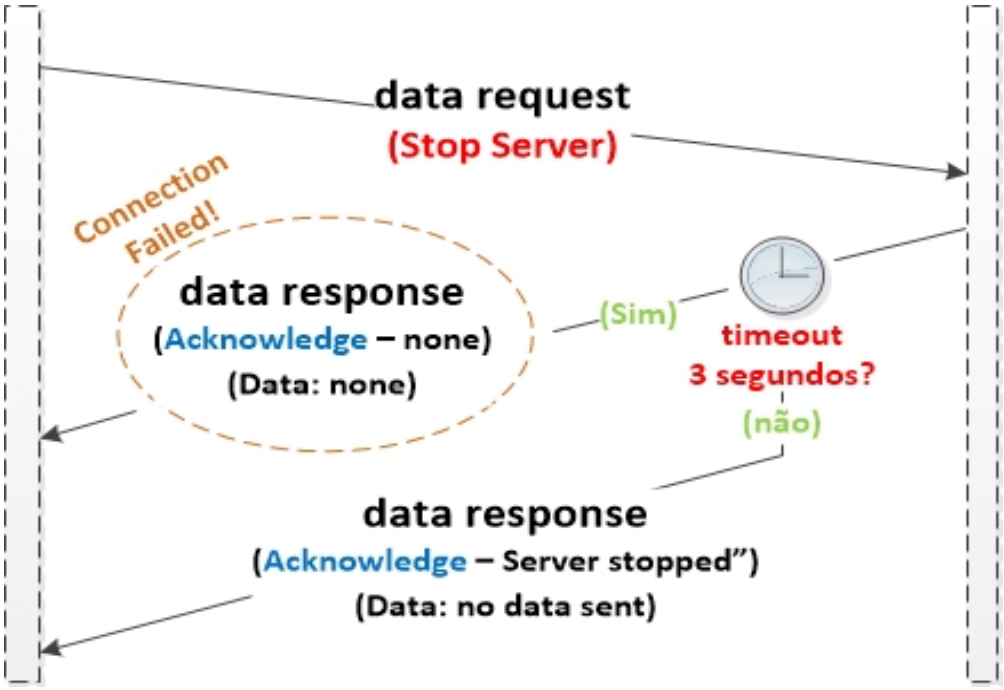


Figura 3.7: *Client Request (Stop Server)* e *Server Responses*

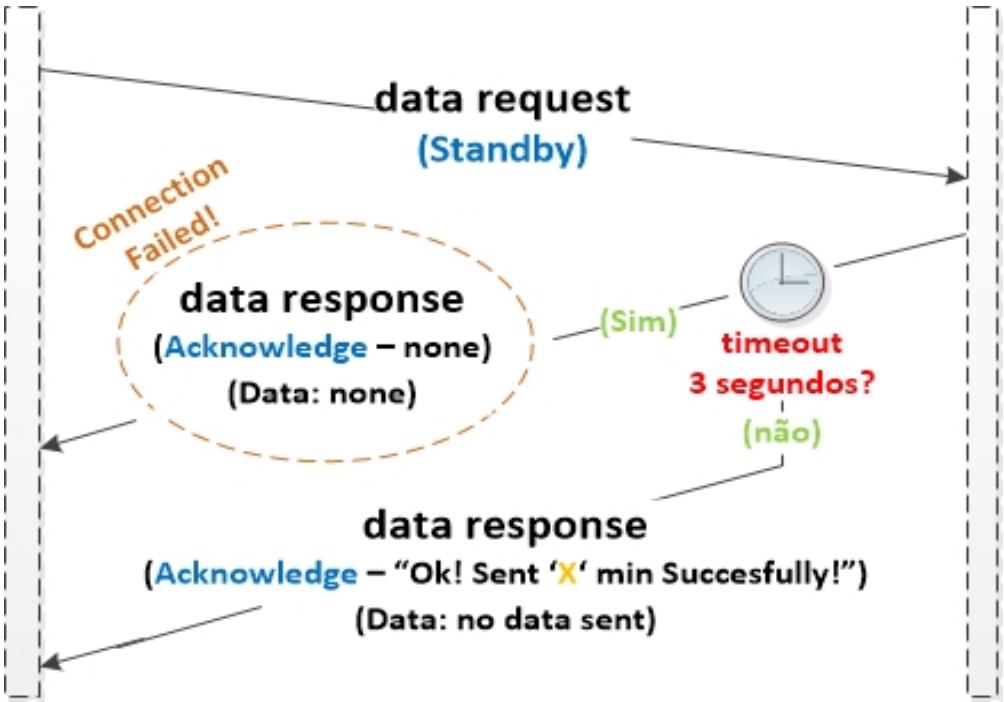


Figura 3.8: *Client Request (Standby)* e *Server Responses*

Para o *acknowledge* da Figura 3.8, o caractere 'X' toma um de quatro valores: 60, 20, 30 ou 10.

De seguida, é demonstrado na Figura 3.9 o modo de operação geral do Servidor, de acordo com tipo de *Request* do Cliente.

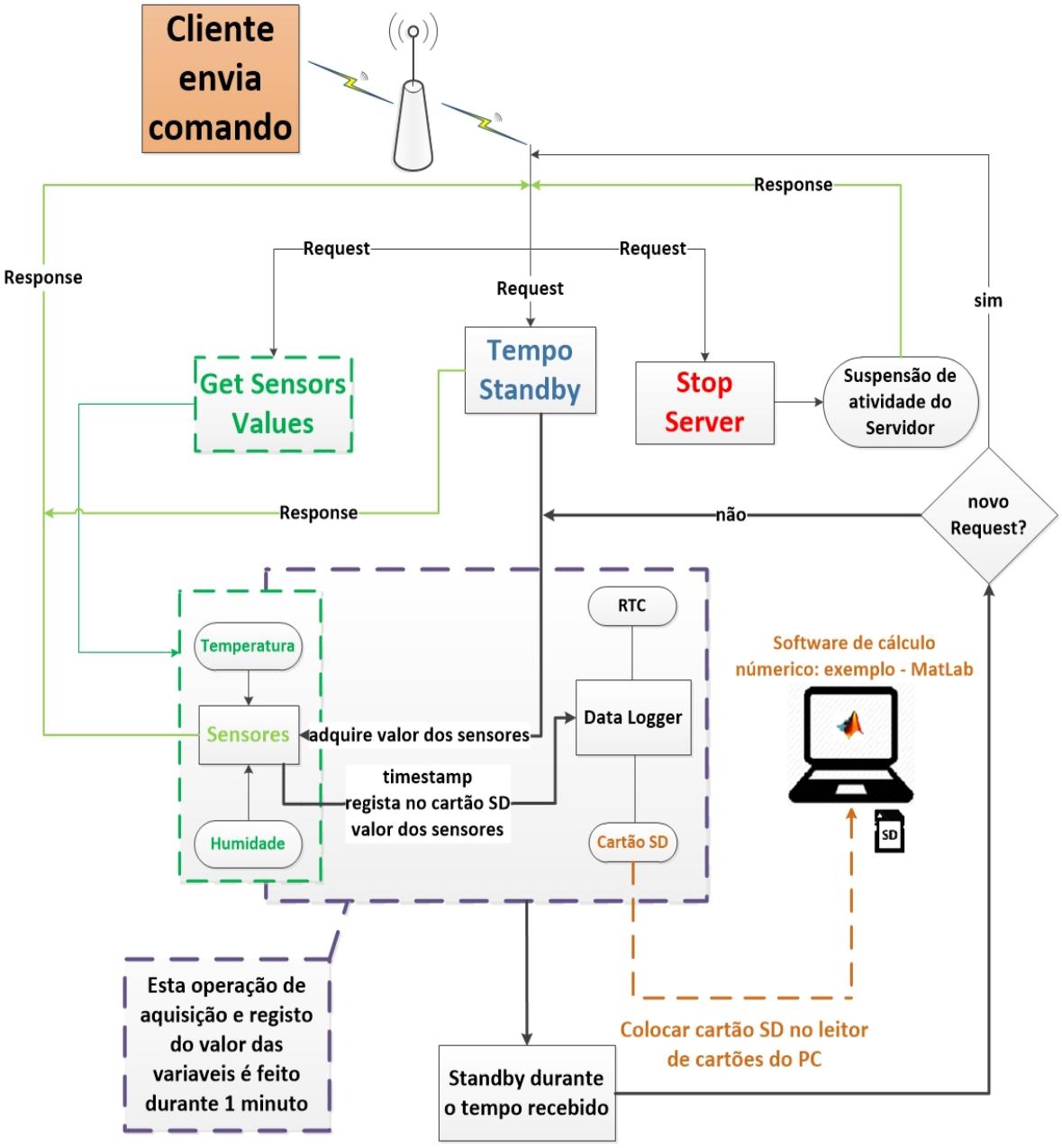


Figura 3.9: Modo de operação do Servidor a Requests do Cliente

Como se verifica na Figura 3.9, o Servidor poderá receber três tipos de *Request*, e atuará em conformidade em cada situação. Se o Cliente enviar um *Tempo Standby* (60, 30, 20 ou 10), o Servidor iniciará durante 1 minuto a aquisição e registo das variáveis no *Data Logger*, ficando

de seguida em *Standby* durante o tempo recebido (repete este ciclo enquanto não receber novo comando).

Por outro lado, o *Request Get Sensors Values* permite ao Cliente solicitar o valor das variáveis de aquisição ao Servidor, que as fornecerá em tempo real. Esta etapa é feita sem interromper a sua operação, quer esteja em aquisição e registo, quer durante o período de *Standby* (repete este ciclo enquanto não receber novo comando).

O *Request Stop Server*, permite interromper as atividades do Servidor, ou seja, a operação de aquisição e registo das variáveis ou o período *Standby*. Nesta situação se o Cliente quiser ativar novamente o Servidor, basta enviar o tempo *Standby* desejado para o Servidor, e este começara as suas operações.

Quando o utilizador entender que já tem dados suficientes guardados (no *"file_base.csv"* existente no cartão SD) ou por outra razão decidir interromper a atividade do Servidor, pode recorrer a outro recurso associado à solução, que consiste num programa desenvolvido num *software* de cálculo numérico (e.g. MatLab) para analisar as variáveis adquiridas.

Apresenta-se de seguida o capítulo que demonstra a implementação da arquitetura anteriormente mencionada.

Capítulo 4

Implementação

Este capítulo tem como objetivo demonstrar a implementação da arquitetura proposta. Encontra-se dividido em duas secções. A primeira secção (secção 4.1), começa com a descrição do *Hardware* utilizado para a construção do Servidor apresentando em várias subsecções as características técnicas de cada um dos seus componentes.

Por sua vez, a secção seguinte (secção 4.2) aborda o *Software* desenvolvido demonstrando-se o algoritmo do Servidor, assim como o algoritmo do Cliente.

4.1 Hardware

O *hardware* utilizado para desenvolver o Servidor, consiste em componentes da plataforma Arduino. Esta opção deve-se ao facto de a plataforma disponibilizar um vasto leque de *shields*, caracterizadas por serem compactas, robustas, com custos de aquisição baixos e modulares. Devido a estes fatores, é assim possível fazer montagens em "cascata" de acordo com a necessidade da solução. Existe uma enorme compatibilidade no acoplamento de *shields*, não sendo na maioria dos casos necessário adicionar *hardware* externo.
Utilizou-se como micro-controlador o Arduino Uno (subsecção 4.1.2), para conexão com a *web* uma *WiFi shield* (subsecção 4.1.3), para guardar os valores dos sensores uma *Data Logger shield* (subsecção 4.1.4) e por fim os sensores necessários uma *Weather shield* (subsecção 4.1.5). A comunicação entre os constituintes do Servidor é do tipo *Master/Slave* (subsecção 4.1.1).

4.1.1 Comunicação Master/Slave

Neste tipo de comunicação o *Master* gere as comunicações entre *Slaves*, e apenas o *Master* pode enviar dados para os *Slaves* sempre que "quiser". Os *Slaves* s podem enviar dados como resposta a pedidos precedentes do *Master*. O *Master* pode enviar para o *Slave* mensagens do tipo escrita ou leitura. Se a mensagem for do tipo "leitura"o *Slave* processa e responde ao *Master* com os itens pedidos (e.g. valor de Temperatura e Humidade relativa). Se a mensagem for do tipo "escrita"não pressupõem resposta do *Slave*, este limita-se unicamente a atualizar as posições de memória ou as suas saídas com os valores indicados pelo *Master* (e.g. ativação/desativação da interface *Data Logger-SD card* para escrita das variáveis no *"file_base.csv"*). A Figura 4.1 ilustra os dispositivos *Slave* e o dispositivo *Master*, assim como os protocolos de comunicação envolvidos nomeadamente, o I2C e o SPI.

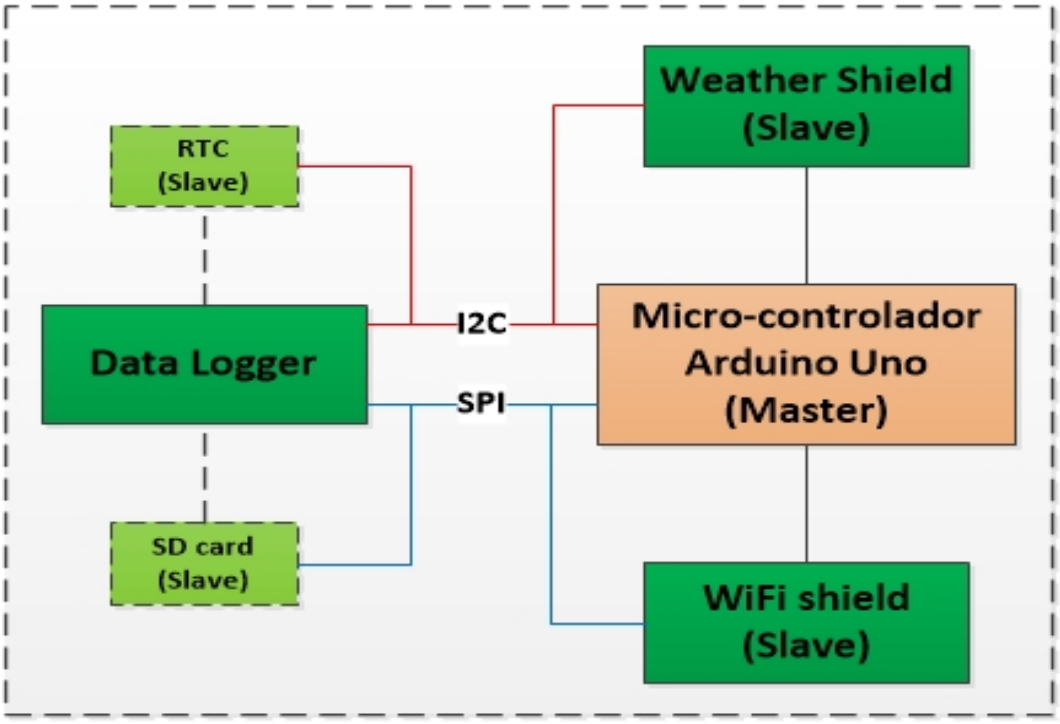


Figura 4.1: Servidor - constituintes *Master/Slave* e protocolos de comunicação (SPI e I2C)

Como se verifica na Figura 4.1, existe um *Master* (Arduino Uno) e os seguintes *Slaves*: uma *WiFi shield*, uma *Weather shield* e uma *Data Logger shield*. A *Data Logger shield*, "contém" dois dispositivos *Slaves* que utilizam protocolos distintos.

4.1.2 Arduino Uno

O micro-controlador Arduino Uno (Figura 4.2), foi escolhido para ser o elemento principal do Servidor, pois tem os recursos necessários e elevada compatibilidade com várias *shields*. Consiste num micro-controlador Atmel AVR de 8 bits, contém 14 portos que podem ser usados como entradas ou saídas digitais. Estes portos operam a 5V, onde cada porto pode fornecer ou receber uma corrente máxima de 40mA. Alguns desses portos têm funções especiais: portos 3, 5, 6, 9, 10 e 11 podem ser usados como saídas PWM *(Pulse-Width Modulation)* de 8 bits. Para comunicação série utiliza os portos 0 (Rx) e 1 (Tx).
Os portos 2 e 3 podem ser configurados para gerar interrupções externas.
Para interface analógica existem 6 entradas, onde cada uma tem a resolução de 10 bits.
Possui ainda um regulador de tensão de 5V, um botão de *reset*, um conector de alimentação externa e uma entrada de ligação USB. A ligação USB é utilizada para a transferência de programas contudo, também serve para o alimentar.

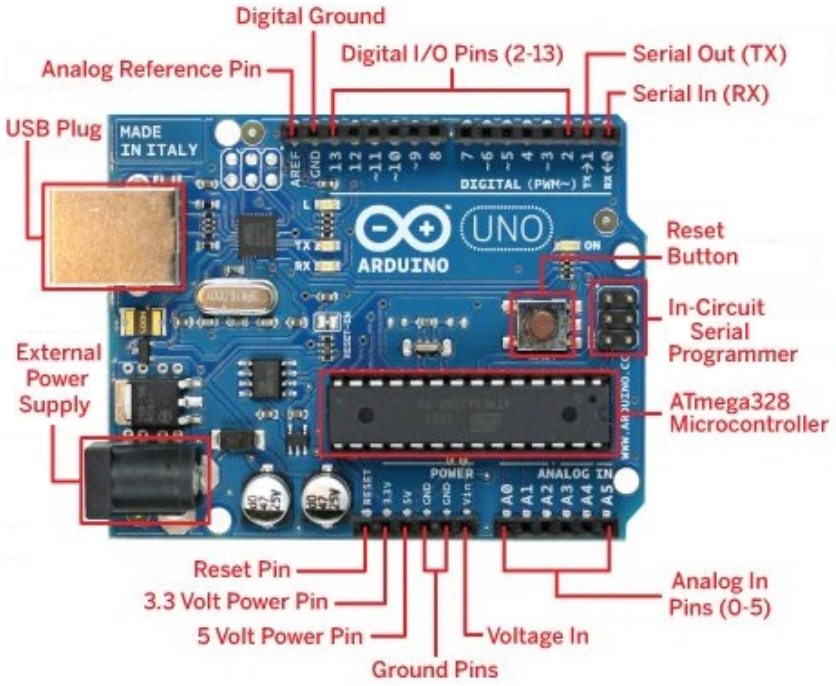


Figura 4.2: Arduino Uno [Htt16b]

O resumo das especificações técnicas são apresentadas na tabela 4.1 [Ard16].

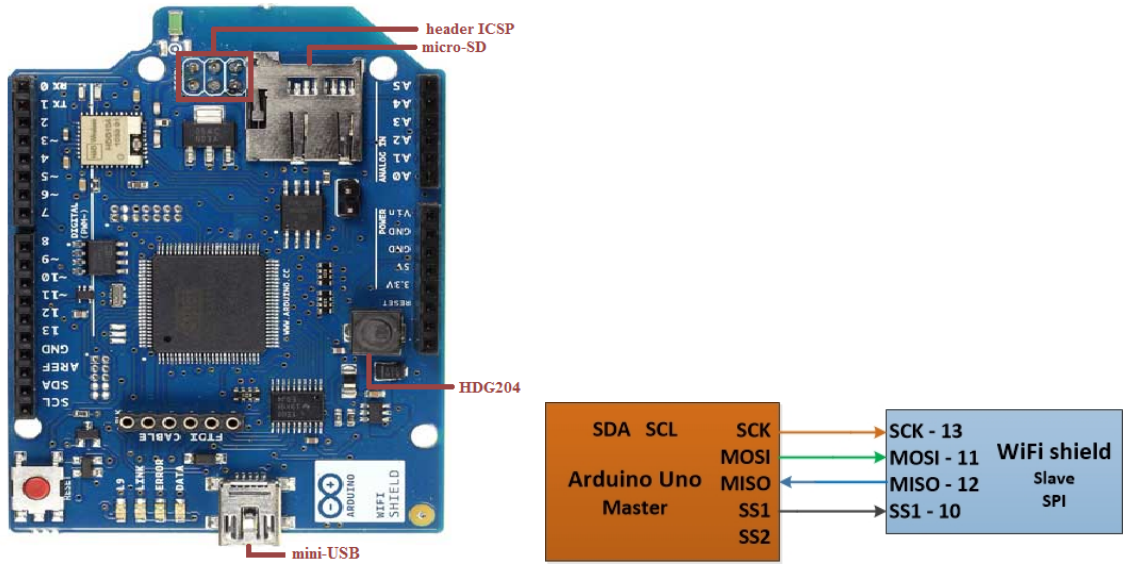
Tabela 4.1: Especificações Técnicas - Arduino Uno

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage	7-12V
Digital I/O Pins	14
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V	50 mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Apresenta-se de seguida as características da *shield* que permite a conexão com a *web*, a *WiFi shield*.

4.1.3 WiFi shield

A *WiFi shield* (Figura 4.3a) acoplada no Arduino Uno permite que este se conecte à *internet* através de *WiFi* (padrão 802.11b/g, via UDP). A comunicação entre Arduino e a *shield* (Figura 4.3b), é realizada através do protocolo SPI (por intermédio do *header* ICSP existente em ambas as placas), recorrendo aos portos digitais 11, 12 e 13.
Disponibiliza interface para cartão micro-SD contudo, este não e utilizado nesta solução devido à inexistência de RTC.
Salienta-se que, os portos digitais 10, 4 e 7 estão reservados, sendo os portos 10 e 4 usados como SS (*Slave Select*, protocolo SPI). O porto 10 é usado para selecionar o componente HDG204[1] e dessa forma habilitar a comunicação WiFi.
O porto 4 é utilizado para habilitar a interface micro-SD (não utilizado) e por fim o porto 7 é usado para fazer o *handshake* entre a placa Arduino e a *WiFi shield*.
A *shield* pode-se conectar a redes abertas e também a redes encriptadas seguindo a *WPA2 Personal* ou a *WEP Personal* [sH16b].



(a) **(b)**

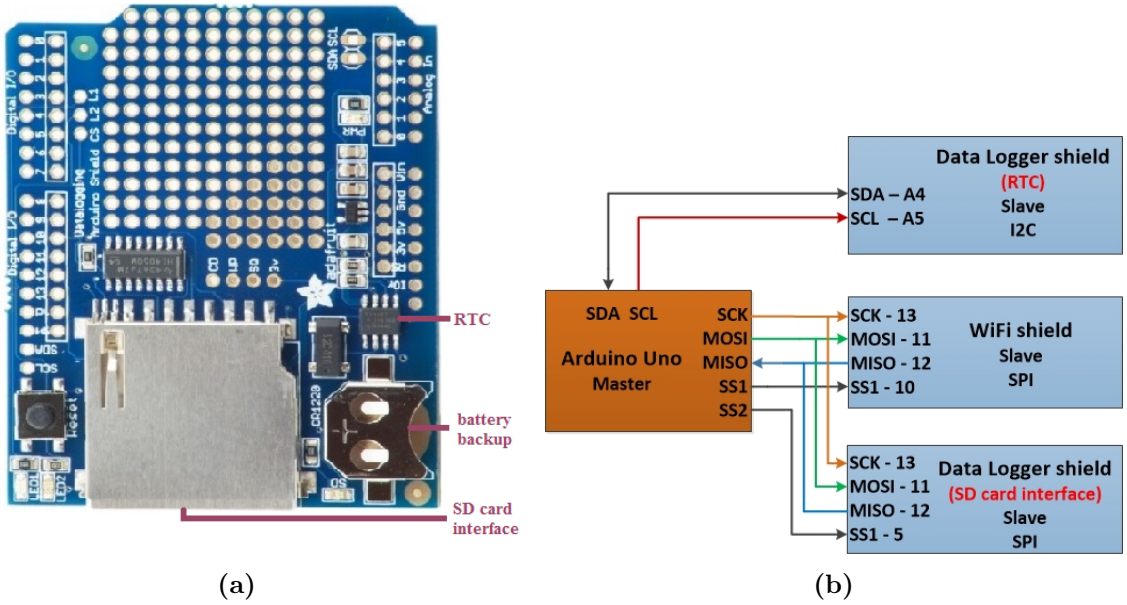
Figura 4.3: **(a)** *WiFi shield* [sH16b], **(b)** Integração com Arduino Uno - protocolo SPI *(WiFi shield)*

Apresenta-se de seguida as características da *shield* utilizada para guardar os valores das variáveis adquiridas, a *Data Logger shield*.

[1] Tecnologia *WiFi embedded*.

4.1.4 Data Logger shield

Para guardar o *"file_base.csv"* (contém as variáveis `Timestamp;Temperatura;Humidade relativa`) recorreu-se a uma *Data Logger shield* (Figura 4.4a), que contém uma interface para cartão SD (formatação FAT16 ou FAT32) e também um RTC.
Na ausência de energia ou paragem de operação do Arduino Uno, o RTC mantém a "contagem da data e da hora" atualizados em memória devido ao facto de estar a ser alimentado por uma bateria externa (pilha de 3V com autonomia para vários anos) [sH16a].
Esta *shield* (Figura 4.4b) comunica com o Arduino Uno através do protocolo SPI (portos digitais 11, 12 e 13) para utilizar a interface SD *card* e o protocolo I2C (portos analógicos A4, A5) para a utilização do RTC.



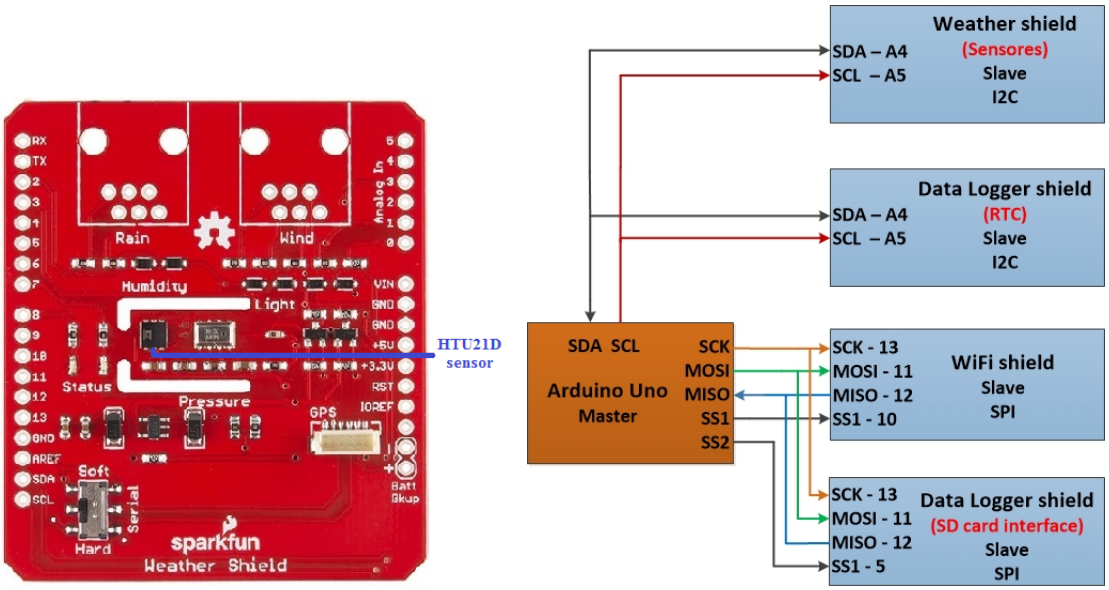
(a) **(b)**

Figura 4.4: **(a)** *Data Logger shield* [sH16a], **(b)** Integração com Arduino Uno e *WiFi shield*

Na subsecção seguinte apresentam-se as características da *shield* que contém os sensores necessários para obter o valor das variáveis a adquirir (Temperatura e Humidade relativa), a *Weather shield*.

4.1.5 Weather shield

Como referido anteriormente, é necessário obter o valor das variáveis Temperatura e Humidade relativa. Para isso, é necessário recorrer a sensores.
A *Weather shield* (Figura 4.5a) inclui sensores de temperatura, humidade relativa, luminosidade e pressão barométrica. Contém também a possibilidade de conectar outros sensores externos nomeadamente, para velocidade/direção do vento, pluviómetro e GPS. Por ser bastante compacta e completa a nível de sensores, é muito utilizada para fazer estações meteorológicas [sH16c].
Contudo, para esta solução só foram utilizados os sensores de Temperatura e Humidade relativa, mais concretamente o sensor HTU21D. De acordo com o *datasheet* [spe13], este sensor caracteriza-se por ser um sensor digital de Humidade relativa com saída de Temperatura, opera a 3.3V (VCC) e 0V (GND). Apresenta baixo consumo de energia e precisão típica de ±2% para Humidade relativa e de ±0.3% para Temperatura. As faixas de operação são [0 a 100]% para Humidade relativa e [-40 a +125]°C para Temperatura.
A comunicação com o Arduino Uno é feita através do protocolo I2C, utilizando os portos analógicos A4 e A5 (Figura 4.5b).



(a) **(b)**

Figura 4.5: **(a)** *Weather shield* [sH16c], **(b)** Integração com Arduino Uno, *WiFi shield* e *Data Logger shield*

Após a análise individual de cada *shield* (verificação de portos reservados, alimentações, sinais, protocolos, etc.), encaixou-se o Arduino Uno com as três *shields* de acordo com a ordem seguinte: Arduino Uno → *WiFi shield* → *Data Logger shield* → *Weather shield* ficando assim um *setup* (dispositivo de sensores remoto - Servidor) compacto conforme se verifica na Figura 4.6.

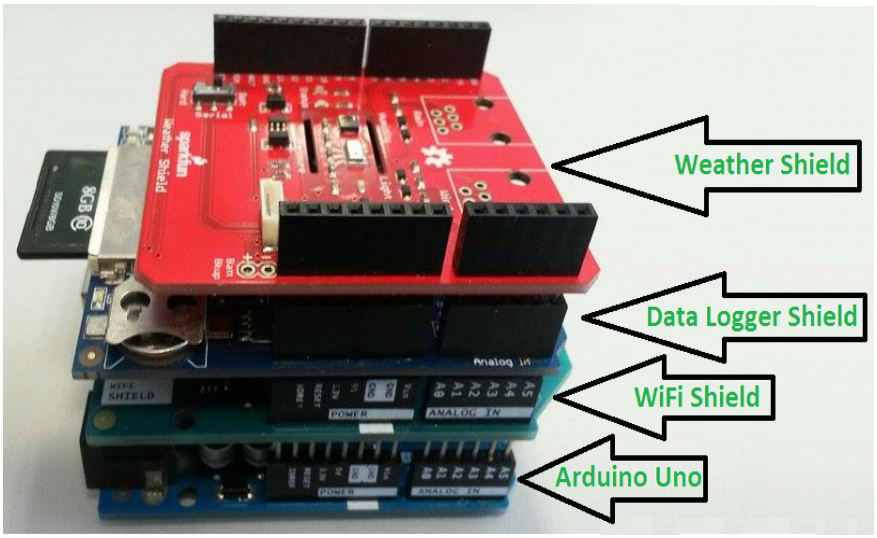


Figura 4.6: Servidor (dispositivo de sensores remoto)

4.2 Software

Nesta secção, é feita a descrição do *software* desenvolvido (recorrendo a diagramas e fluxogramas) que permite ir ao encontro dos objetivos propostos 1.3.

Descreve-se na subsecção 4.2.1 o algoritmo Arduino (Servidor - dispositivo de sensores remoto), na subsecção 4.2.2 o algoritmo Android (Cliente), e por fim na (subsecção 4.2.3 o algoritmo MatLab (programa para análise das variáveis adquiridas).

4.2.1 Arduino

Desenvolveu-se um programa denominado de `HygrothermalAppUA_Arduino.ino` no IDE oficial da plataforma *open source* Arduino, de modo a poder conjugar as três *shields*. No Apêndice B, demonstra-se como se compila, transfere e executam programas na plataforma.

Apresenta-se na Figura 4.7, o fluxograma do programa Arduino, que ilustra todo o funcionamento da aplicação Servidor.

Para concluir a subsecção, procura-se explicar a arquitetura do Servidor (Figura 4.1) e quais os protocolos de comunicação utilizados em cada um dos seus intervenientes.

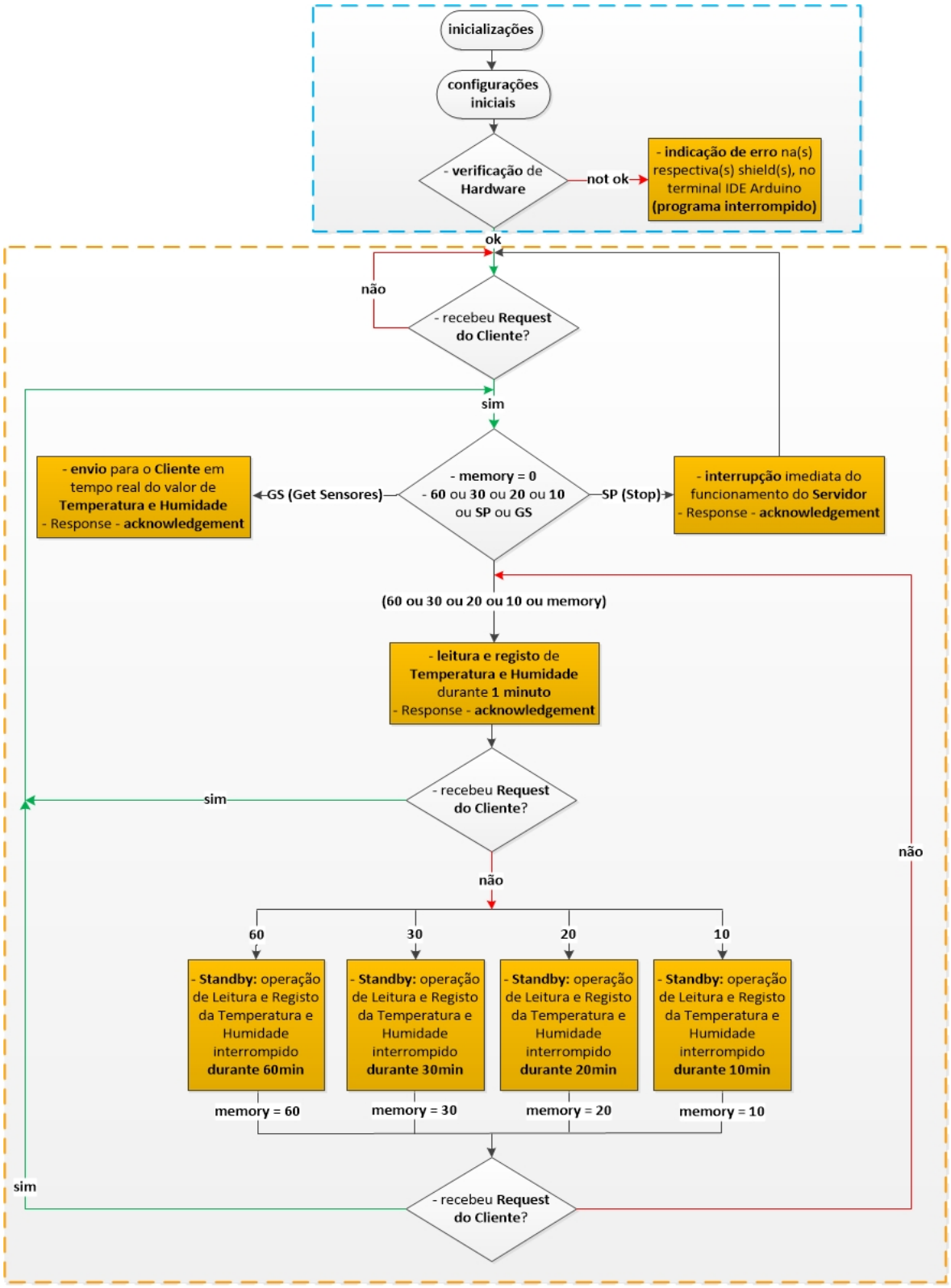


Figura 4.7: Fluxograma - algoritmo Arduino

A caixa com linha azul tracejado do fluxograma da Figura 4.7, é referente a testes que se realizam quando se coloca o Servidor em funcionamento pela primeira vez. Consiste na verificação do *Hardware* e é visualizado no IDE com o Servidor ligado por USB a um computador com o código desenvolvido em execução. Por sua vez, a caixa com linha laranja (da mesma Figura) demonstra o algoritmo do Servidor, quando este faz *Response* aos *Requests* do Cliente. O Servidor encontra-se sempre à espera de *Request* de algum Cliente e quando recebe responde em conformidade. Pode receber tempos *Standby*(60 ou 30 ou 20 ou 10), *GS*(Get Sensors) ou *SP*(Stop). Como referido anteriormente, o protocolo de comunicação para troca de comandos usado entre Cliente e Servidor é o protocolo IEEE 802.11 via UDP. Este protocolo não dá garantias que o comando chegue ao destino. Por essa razão dotou-se o Servidor com a capacidade de quando receber um comando enviar um *acknowledgement* para o respetivo Cliente, e dessa forma o Cliente ter uma garantia se a operação foi bem sucedida. Por outro lado, o Cliente sempre que envia um comando, contém um *timeout* de 3 segundos, e sempre que esse tempo seja alcançado sem obter um *Response* significa que o envio do comando não teve sucesso.

4.2.2 Android

A aplicação Cliente desenvolveu-se em Android Studio (IDE oficial da Google) e chama-se `HygrothermalAppUA`. No Apêndice C, são apresentadas as ferramentas necessárias para desenvolver aplicações, assim como a demonstração da compilação, da transferência e da execução de programas. A Figura 4.8 apresenta o diagrama UML *(Unified Modeling Language)* de classes da aplicação e tem como principal função evidenciar os relacionamentos entre as *Activities*/classes (setas representam herança e os losangos representam agregação). Apresentado na Figura 4.9) um fluxograma que demonstra o algoritmo completo da aplicação Cliente. No Apéndice A caracterizam-se os componentes utilizados no desenvolvimento da aplicação. A subsecção é concluída apresentam-do-se as características principais da aplicação.

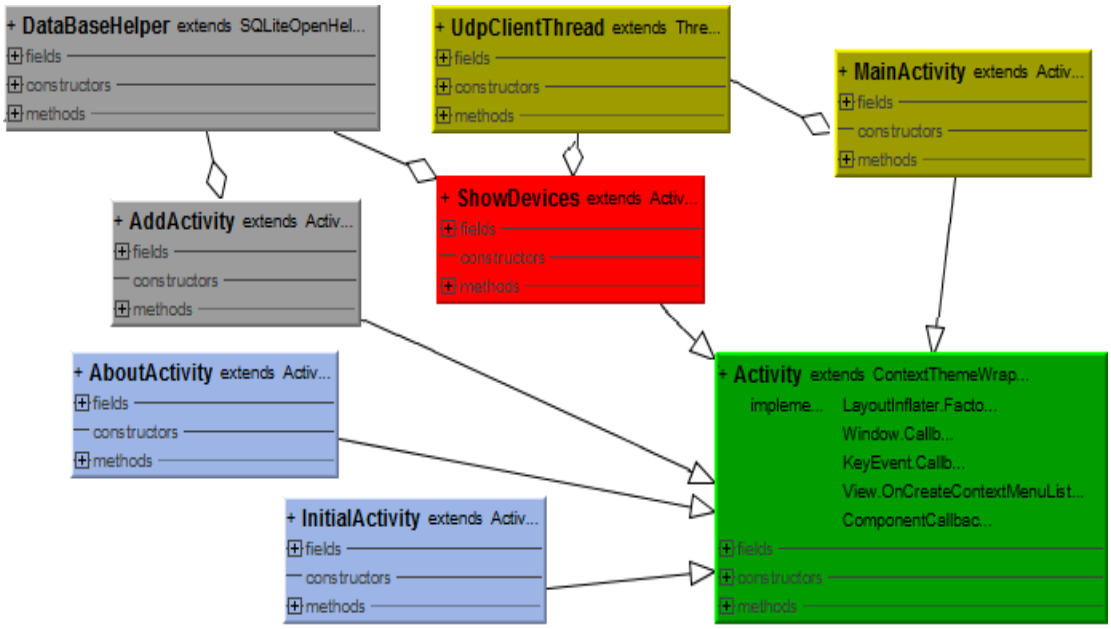


Figura 4.8: Diagrama UML - Aplicação HygrothermalAppUA

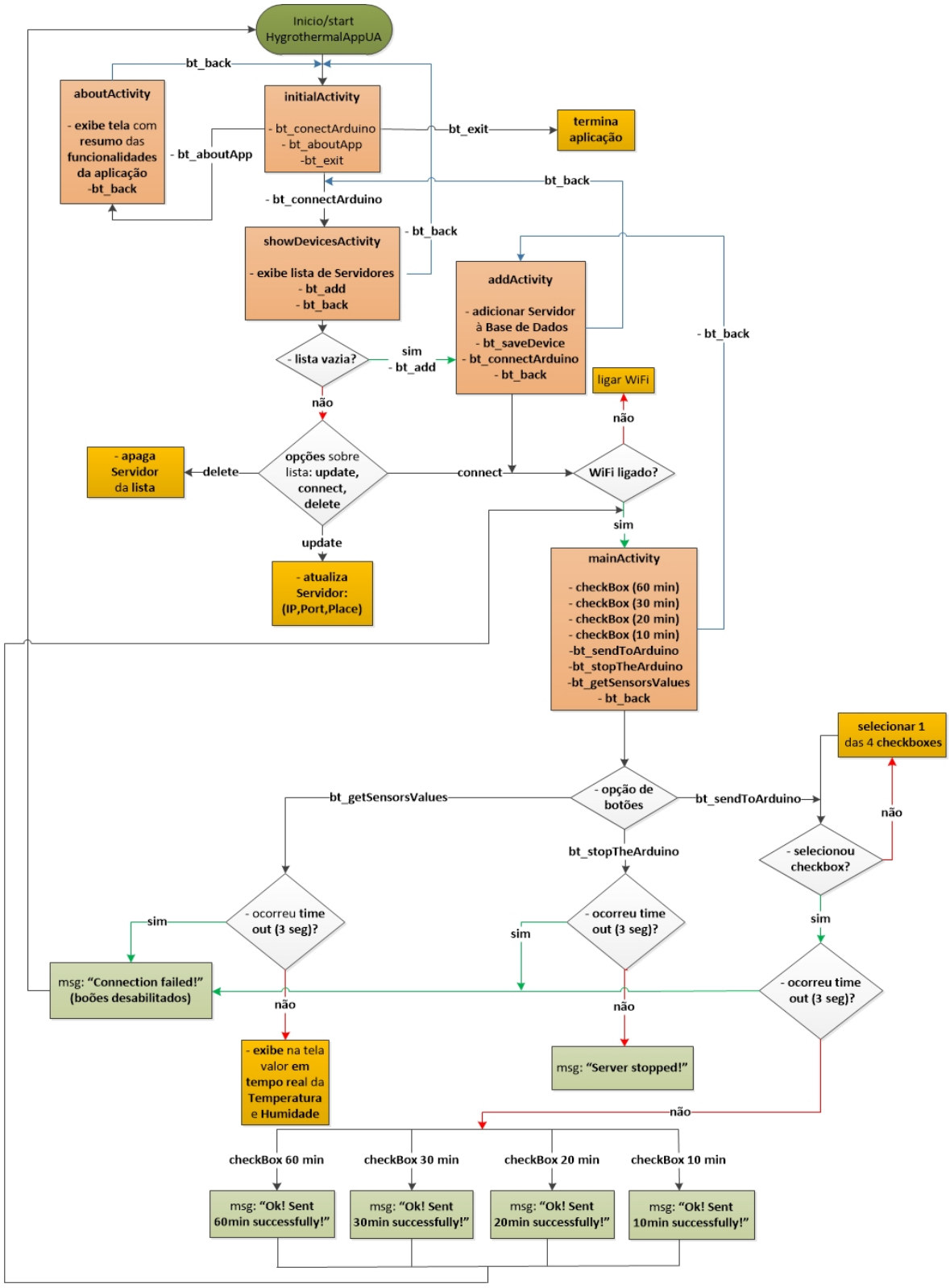


Figura 4.9: Fluxograma - algoritmo Android

O fluxograma da Figura 4.9 demonstra de forma exaustiva o algoritmo e funcionalidades da aplicação Android (Cliente). Estão presentes todas as *Activities* (blocos de cor rosa), sendo que em cada *Activity* estão mencionados os seus respetivos botões, necessários para o utilizador "navegar" pela aplicação, assim como também estão presentes as mensagens, alertas de todos os cenários possíveis que a aplicação oferece ao utilizador.

Apresenta-se de seguida a descrição dos elementos chave da aplicação: a base de dados SQLite e a comunicação UDP.

Base de dados SQLite

A base de dados é composta por uma tabela (*arduino_table*) sendo esta constituída por quatro atributos, conforme se verifica na Tabela 4.2.

Tabela 4.2: Atributos de dispositivo de sensores remoto (Servidor) a guardar na base de dados

arduino_table			
ID	**IP address**	**Port number**	**Place**
1	192.168.1.98	8081	Deti
2	192.168.1.91	8080	Dem
(...)	(...)	(...)	(...)

A chave primária é o ID e esta é incrementada automaticamente sempre que seja adicionado um novo dispositivo. Quando se remove dispositivo, os IDs não são atualizados. O ID em causa não é decrementado, mas sim eliminado (esse ID não surge mais na *ListView* até que todos os dispositivos sejam eliminados, e o incremento comece novamente).

Envio de mensagens Cliente/Servidor

O Cliente e o Servidor trocam informação através do IEEE 802.11, via UDP, utilizando *sockets*. O UDP é um serviço sem conexão, nenhuma sessão é estabelecida entre Cliente e Servidor. Caracteriza-se por ser um protocolo simples, não ser fiável, pois não possui mecanismos de controlo que verifiquem se o envio e receção do comando foi bem sucedido. O objetivo dessa opção é acelerar o processo de envio, visto que todas as etapas de comunicação necessárias para verificar a integridade de um comando (e para o reenviar, se necessário) contribuiriam para o deixar lento. Por essa razão é muito utilizado para transmitir dados pouco sensíveis, como fluxos de áudio e vídeo. As aplicações que usam o protocolo são responsáveis por oferecer a confiabilidade necessária ao transporte dos comandos.
Como referido anteriormente, para garantir a integridade do envio, foi necessário implementar estruturas de controle. Para o Cliente implementou-se mecanismo de *timeout* e para o Servidor mecanismo de envio de *acknowlegment*. Sempre que o Cliente envie um comando para o Servidor, este além de executar o serviço, deve também enviar um *acknowlegment* ao Cliente, indicando a receção do comando. Por sua vez, se decorridos 3 segundos após o Cliente enviar o comando ao Servidor não obtiver um *acknowlegment*, significa que houve falha na comunicação.

Socket UDP

Sockets são uma abstração para endereços de comunicação através dos quais processos comunicam, ou seja, *Socket* é a combinação do ***IP address*** (identifica a máquina na rede) e do ***Port number*** (identificador local da porta usado pelo Servidor). O Cliente deve saber previamente qual o ***IP address*** do Servidor (recorre à *ListView* da base de dados) e o respetivo ***Port number*** para o qual pretende comunicar. A Figura 4.10 ilustra um Request/Response através do protocolo UDP, indicando os mecanismos de controlo de cada dispositivo.

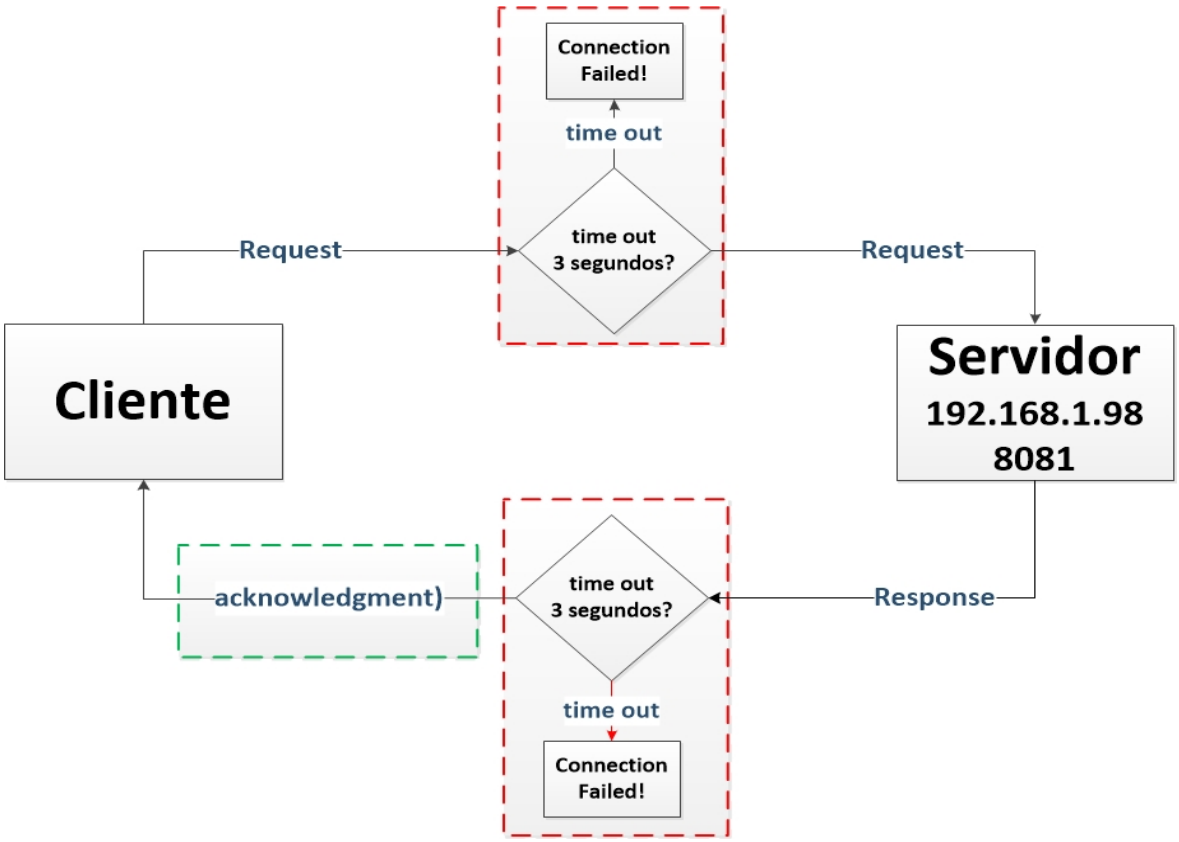


Figura 4.10: Protocolo UDP - *Request/Response* (Cliente/Servidor)

Verifica-se que o Cliente fez um *Request* para o Servidor identificado pelo ***IP address*** 192.168.1.98 e pelo ***Port number*** 8081 (exemplo). Caso não haja problemas (e.g. Servidor inativo, problemas na rede, etc.), o Servidor receberá o *Request*. Se o Cliente não receber um *acknowledgement* do Servidor antes de ter ocorrido *timeout*, significa que o comando não foi recebido (ver retângulo picotado vermelho, Figura 4.10), caso contrário recebe um *acknowledgement* (ver retângulo picotado verde, da mesma Figura) que indica que o comando foi bem recebido pelo Servidor.

4.2.3 MatLab

Para a análise e representação gráfica dos dados do ficheiro *"file_base.csv"* foi necessário desenvolver um programa voltado para o cálculo numérico. Optou-se pelo MatLab[2] por ser uma ferramenta de engenharia otimizada para fazer cálculos matemáticos e para a representação gráfica de dados. Contém um conjunto alargado de funções pré-definidas, normalmente agrupadas em *toolboxes*, que tornam o desenvolvimento do algoritmo mais rápido e eficiente.

O programa desenvolvido chama-se `HygrothermalAppUAMatLab.m` e começa por verificar se o *"file_base.csv"* existe. Se não existir, o programa termina. Caso exista, todos os dados serão guardados numa matriz (à exceção da primeira linha, que indica o tipo e a ordem das variáveis). Através dessa matriz, é desenvolvido o programa.

Dado que o dispositivo de sensores remoto (Servidor) monitoriza e guarda os valores dos sensores durante um tempo máximo de 1 minuto (tempo de aquisição), a primeira ação a ser feita é a média dos valores de cada variável em cada tempo de aquisição, obtendo assim um único valor. Estes novos dados são guardados num novo ficheiro denominado de *"file_dados.csv"*, sendo estes os dados utilizados para gerar os gráficos pretendidos.

A Figura 4.11 ilustra o algoritmo para obtenção do ficheiro *"file_base.csv"* a partir do *"file_dados.csv"*.

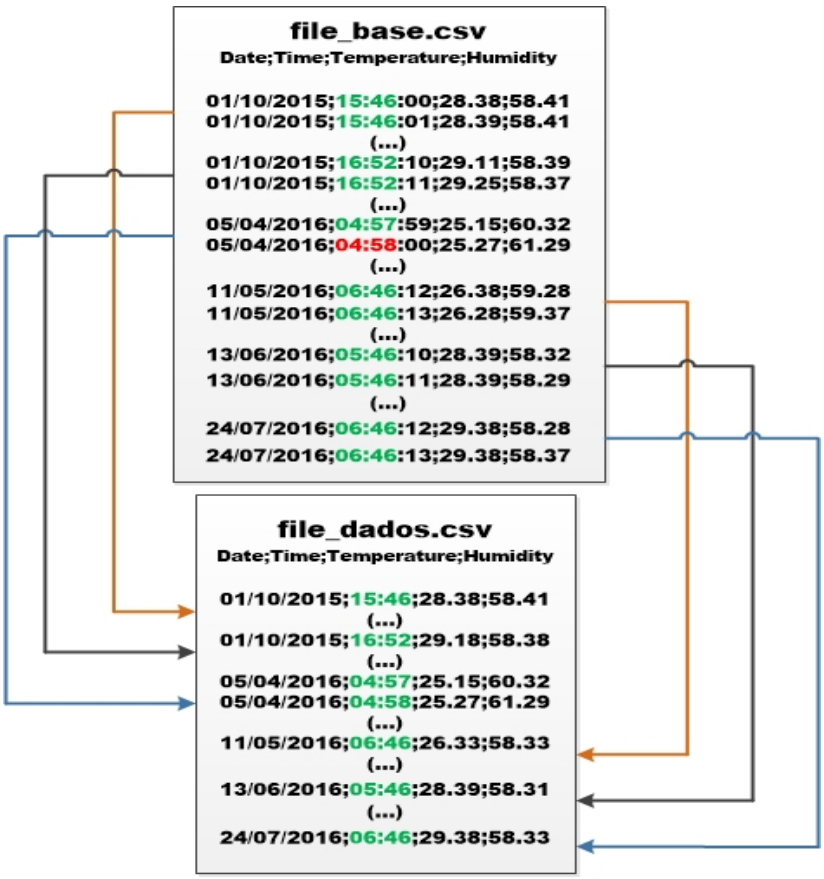


Figura 4.11: Obtenção do *file_dados.csv* a partir do *file_base.csv*

[2]MATrix LABoratory - software interativo cujo elemento básico de informação são matrizes.

A Figura 4.11 contém um trecho de dados, que ajudam a perceber qual a técnica usada para se obter o novo ficheiro *("file_dados.csv"* usado para fazer os gráficos). É feita a média do valor de cada variável adquirida, durante o tempo de aquisição (valor máximo de 1 minuto). Em cada minuto obtêm-se 60 amostras de cada variável (exceto o utilizador tenha interrompido o funcionamento do dispositivo, ou outra razão externa). O programa começa por comparar minutos de cada hora, e sempre que forem iguais faz a média, avançando até ao fim do ficheiro *"file_base.csv"*. Após ter sido feita a média o programa escreve no novo ficheiro a data, a hora (formato `HH:MM`, os segundos são descartados) e um único valor para cada variável adquirida, ficando com o seguinte formato: `DD/MM/YYYY;HH:MM;Temperatura;Humidade`, conforme se verifica no *"file_dados.csv"* da Figura 4.11. Através dos dados deste novo ficheiro são feitos os gráficos pretendidos. A Figura 4.12 indica como é feita a abordagem para fazer gráficos para dia, mês ou ano, sendo que neste exemplo só são feitos gráficos para Temperaturas (dados do ficheiro *"file_dados.csv"* da mesma Figura).

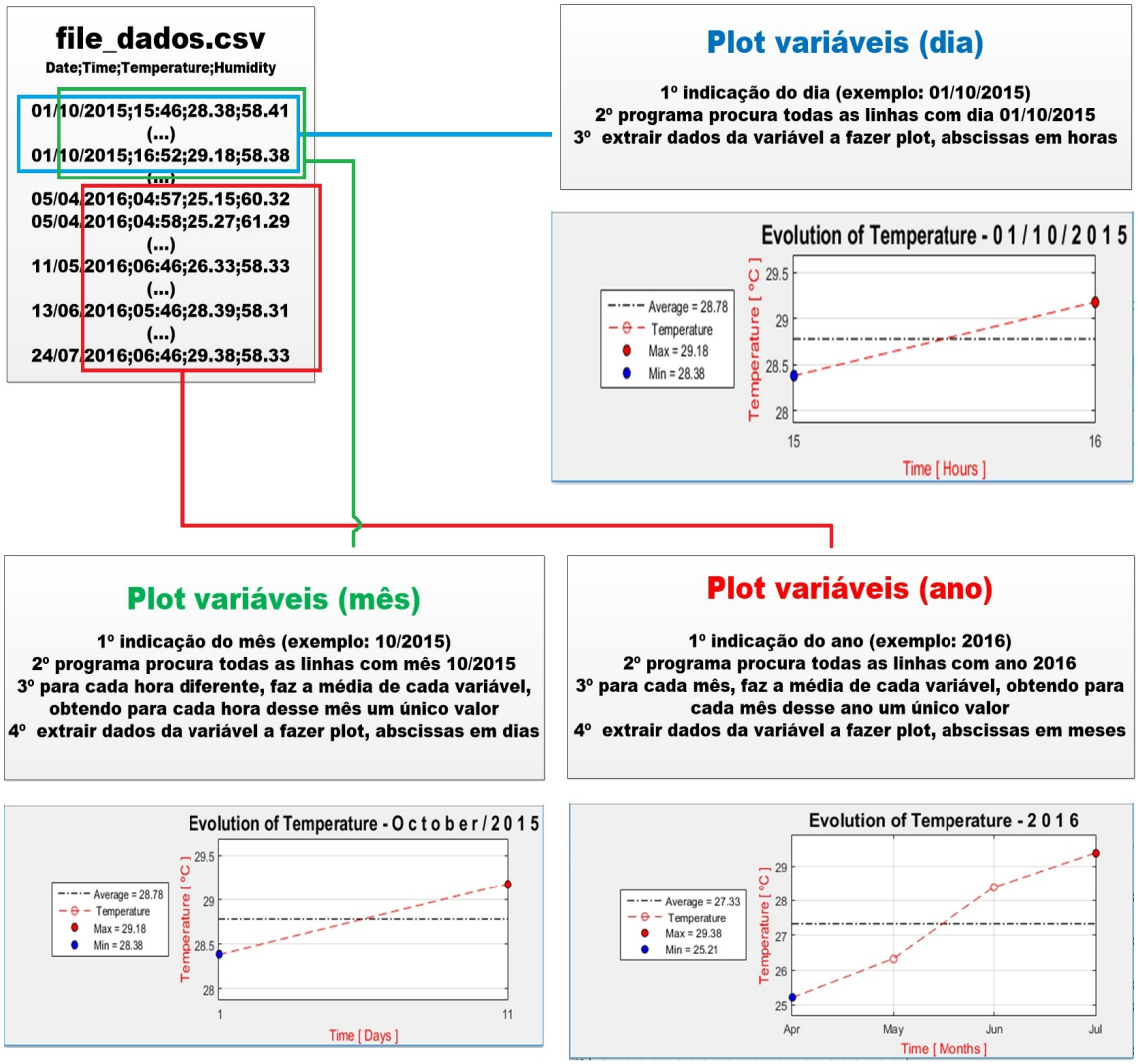


Figura 4.12: Demonstração de gráficos, para Temperatura (dia, mês, ano)

Apresenta-se na Figura 4.13 o fluxograma que demonstra o programa MatLab.

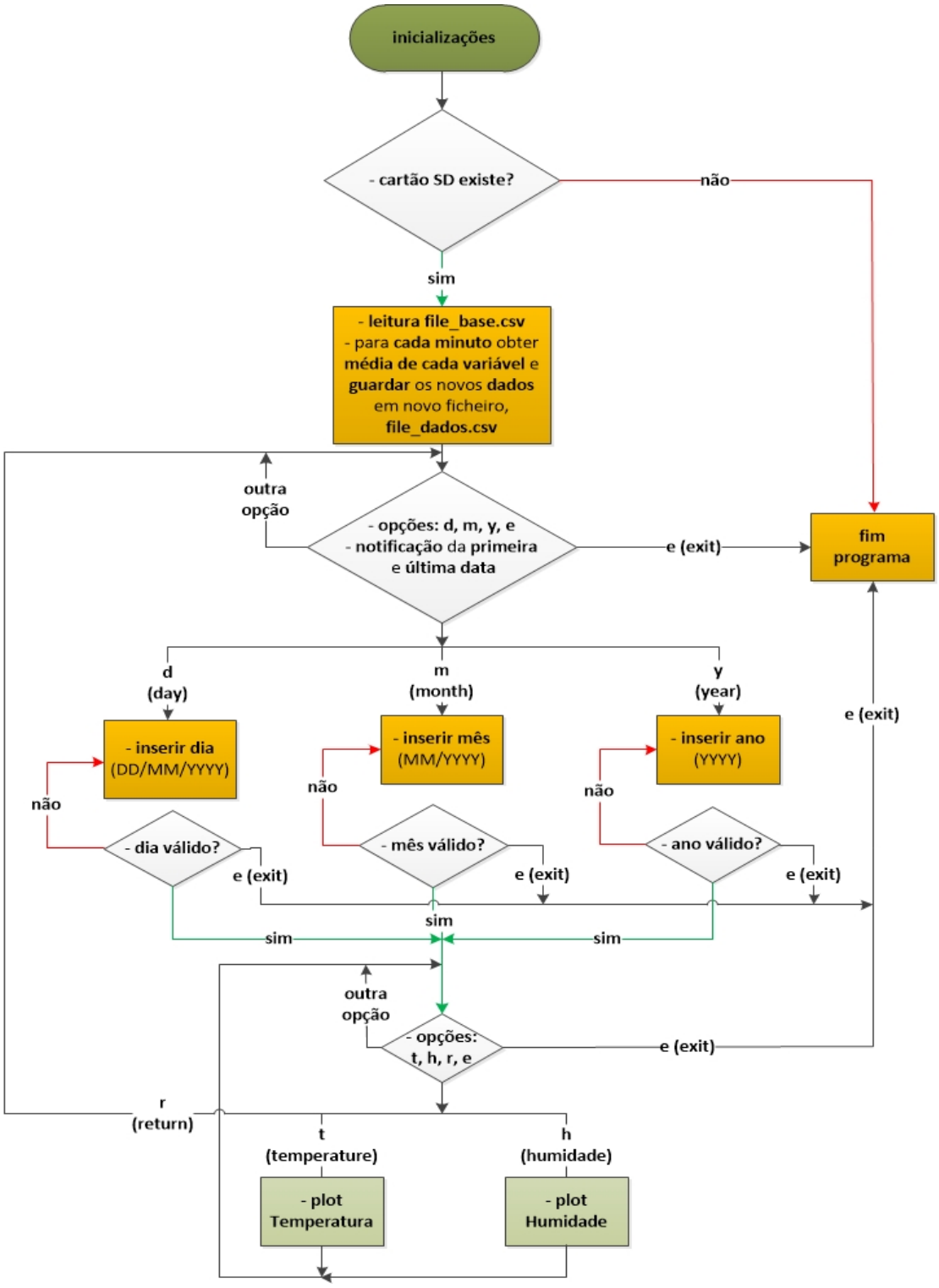


Figura 4.13: Fluxograma - algoritmo MatLab

Apresenta-se de seguida o capítulo que demonstra o sistema e resultados obtidos.

Capítulo 5

Demonstração do Sistema e Resultados

Neste capítulo são apresentados os resultados deste trabalho. Como referido anteriormente foi desenvolvida uma solução que utiliza o modelo Cliente/Servidor. O Cliente é uma aplicação Android e o Servidor é um dispositivo de sensores remoto construído com componentes da plataforma Arduino, que adquire e regista num ficheiro denominado *"file_base.csv"* dados no seguinte formato: `Timestamp;Temperatura;Humidade relativa`, tendo o *Timestamp* o formato `DD/MM/YYYY;HH:MM:SS`.

Apresenta-se a interface gráfica da aplicação Cliente (secção 5.1), cujo objetivo consiste em demonstrar as funcionalidades da aplicação e também a interface gráfica do programa MatLab (secção 5.2), cuja finalidade é permitir ao utilizador analisar as variáveis Temperatura e Humidade relativa do *"file_dados.csv"* em função do tempo, que pode ser ao dia, ao mês ou ao ano.

5.1 Interface Gráfica e Utilização da Aplicação Android

A descrição que se apresenta tem como propósito a ótica de utilizador e é feita com recurso a imagens a ilustrar os vários cenários e funcionalidades da aplicação `HygrothermallAppUA` utilizando como *smartphone* de testes o modelo **Samsung S4 mini**, Figura 5.1a na qual é visível o ícone da aplicação (círculo vermelho). Pressionando o ícone da aplicação, o utilizador tem acesso à *Activity* inicial, sendo esta mostrada na Figura 5.1b.

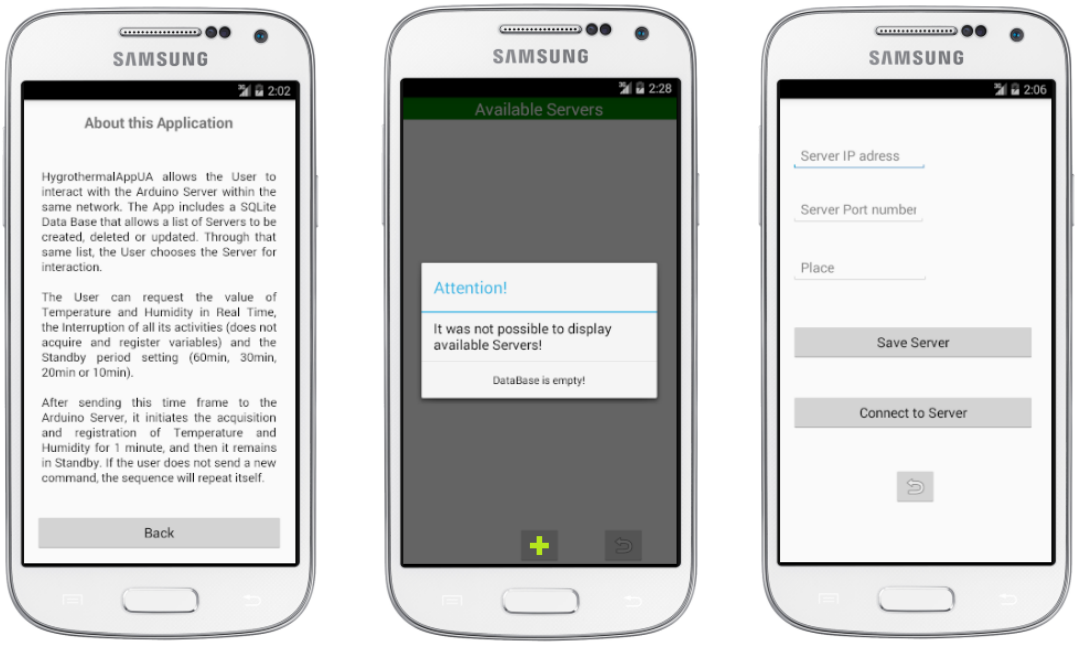
(a) **(b)**

Figura 5.1: **(a)** Ìcone da aplicação HygrothermalAppUA, **(b)** *Activity* inicial da aplicação

Tendo em conta a Figura 5.1b, verifica-se que a *Activity* inicial é constituída pelos seguintes botões:

- Connect to Server
- About the App
- Exit

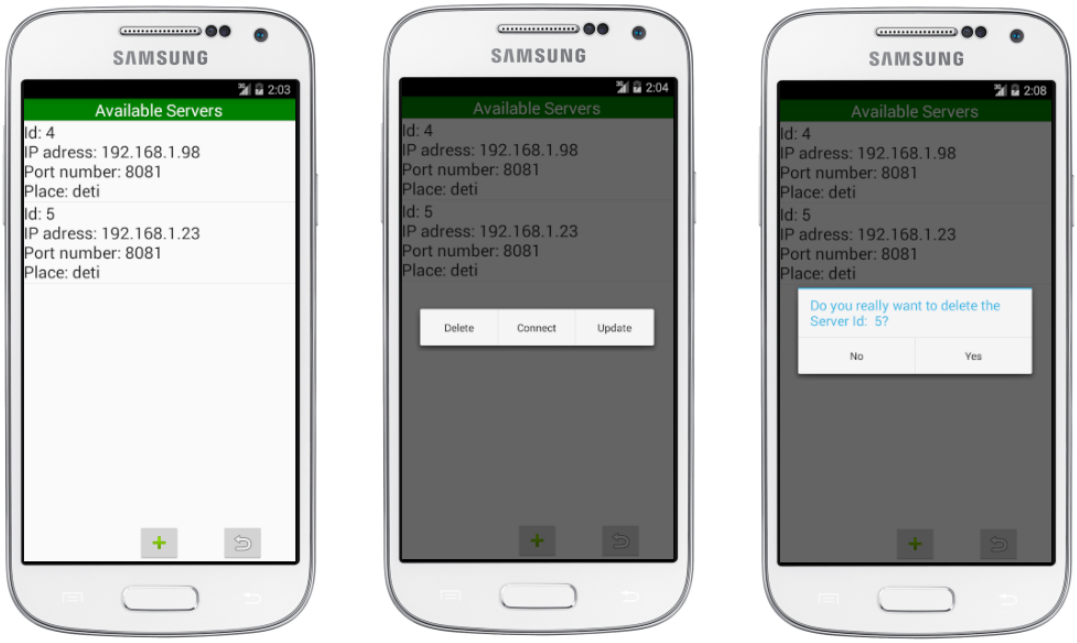
O botão Exit tem como única função terminar a aplicação.
A função do botão About the App é permitir exibir outra *Activity*, que contém um resumo das funcionalidades da aplicação (Figura 5.2a). Por sua vez, o botão Connect to Server (elemento mais importante nesta *Activity*), quando pressionado duas situações podem ocorrer: existir ou não existir dispositivos (Servidores) guardados. Se não existirem, um *AlertDialog* é exibido (Figura 5.2b), informado que não há dispositivos.
Nesse caso é necessário adicionar à base de dados um novo dispositivo. Para isso, nessa mesma *Activity* (Figura 5.2b) existe o botão [+] que dá acesso a uma nova *Activity* (Figura 5.2c) para adicionar novos dispositivos.



(a) **(b)** **(c)**

Figura 5.2: **(a)** *Activity* - Sobre a aplicação, **(b)** *AlertDialog* - Base de dados vazia, **(c)** *Activity* - Adicionar dispositivos

Cada dispositivo é constituído por um ***Id*** (valor incrementado automaticamente, utilizado como chave primária na base de dados), um ***IP Address*** (identifica o dispositivo na rede onde este esteja conectado), por um ***Port number*** (porto para a conexão *socket*) e por fim pelo ***Place*** (indicação do local onde o dispositivo está em operação).
Todos os parâmetros introduzidos pelo utilizador são validados e caso estejam incorretos, o utilizador é notificado (através de *AlertView*) para fazer a respetiva correção.
Se estiverem corretos, o dispositivo é adicionado com sucesso à base de dados (após pressionar o botão Save Server).
Por outro lado, se já existirem dispositivos o utilizador tem acesso a uma *ListView* (Figura 5.3a), que permite visualizar quantos dispositivos existem no "terreno". Dessa forma, selecionando o dispositivo pretendido o utilizador fica habilitado a três opções: **Delete**, **Update** e **Connect** (Figura 5.3b).

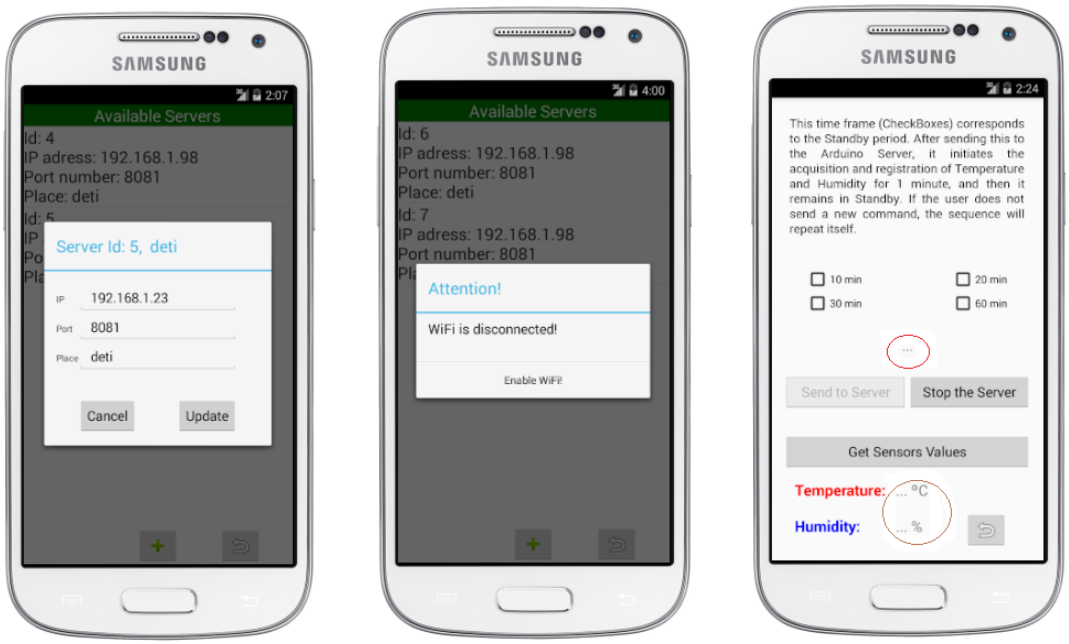


(a) **(b)** **(c)**

Figura 5.3: **(a)** Dispositivos existentes, **(b)** Opção para dispositivo, **(c)** Apagar dispositivo

Estas três opções permitem a gestão da base de dados e a interação com os dispositivos remotos (Servidores).
A opção **Delete** (Figura 5.3c) invoca um *AlertView* que permite apagar o dispositivo selecionado.
A opção **Update** (Figura 5.4a) permite atualizar o dispositivo selecionado, por exemplo devido ao facto de este ter sido mudado de local, e nesse caso pelo menos o ***IP address*** e o ***Place*** devem ser alterados. Os novos dados introduzidos são também validados, e caso estejam incorretos o utilizador é notificado (através de *AlertView*) para fazer a respetiva correção.
A opção **Connect** quando pressionada permite ao utilizador a possibilidade de interagir com o dispositivo remoto. A primeira situação a ser verificada nesta etapa, é se o *Smartphone* Android está ligado ao *WiFi*. Caso não esteja, o utilizador é notificado para o fazer (Figura 5.4b). Estando esta verificação ultrapassada surge uma nova *Activity* (Figura 5.4c), onde esta apresenta quatro *CheckBox* e os seguintes botões:

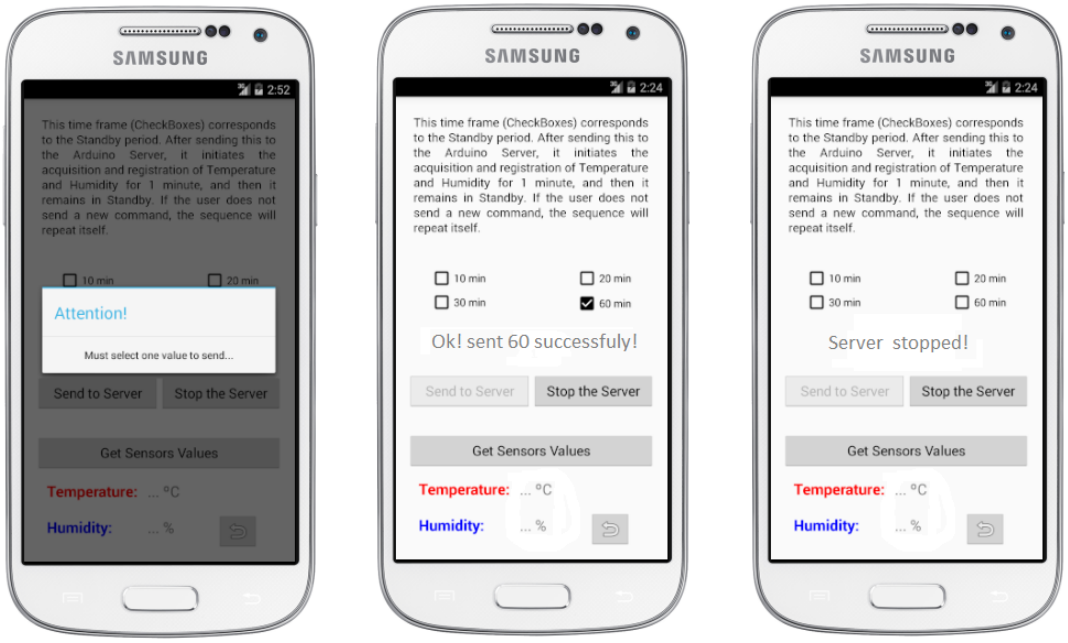
- Send to Server
- Stop the Server
- Get Sensors Values



(a) **(b)** **(c)**

Figura 5.4: **(a)** Atualizar dispositivo, **(b)** Alerta *WiFi* desligado, **(c)** Interação dispositivo

Nesta etapa, o botão Send to Server encontra-se desabilitado. Cada *CheckBox* corresponde a um tempo (em minutos) que o utilizador escolhe para pôr o dispositivo remoto em *standby*. Deve ser selecionado uma *CheckBox* (não é possível selecionar duas ou mais em simultâneo) e de imediato o botão Send to Server fica habilitado. Caso o utilizador clique no botão Send to Server envia esse tempo para o dispositivo remoto. Se por ventura não tenha sido selecionado nenhuma *CheckBox* e o botão Send to Server seja pressionado, surge um *AlertView*, (Figura 5.5a) indicando que deve ser selecionado uma *CheckBox*.
Caso o envio seja bem sucedido, o campo **'...'** (identificado pelo circulo vermelho) da Figura 5.4c são substituídos pela mensagem **'Ok! sent** ck **successfully!'**, sendo que ck toma o valor da *CheckBox* selecionada (Figura 5.5b).
Por sua vez, o botão Stop the Server tem como função interromper o funcionamento do dispositivo (deixa de haver obtenção e registo de Temperatura e Humidade relativa ou interrompe o período de *Standby*) e surge a mensagem **'Server stopped!'** (substituindo o campo **'...'** da Figura 5.4c), como se verifica na Figura 5.5c.
Por sua vez, o botão Get Sensors Values, permite ao utilizador obter o valor (nesse instante) da Temperatura e Humidade relativa, sendo estes indicados nos campos **'...°C** e **...%** (identificados pelo circulo castanho) e substituindo o campo **'...'** (identificado pelo circulo vermelho) da Figura 5.4c, pela mensagem **'Sensors values'**, conforme se verifica na Figura 5.6a.
Contudo, se o dispositivo ao qual o utilizador pretenda conectar-se não esteja disponível (e.g. problemas de rede, dispositivo inexistente, etc.), e pressione qualquer um dos três botões, o utilizador é notificado com a seguinte mensagem: **'Connection Failed!'** (substituindo o campo **'...'** da Figura 5.4c, identificado pelo circulo vermelho). Nesta situação (ocorreu *timeout* de 3 segundos) os três botões deixam de estar habilitados, conforme se verifica na Figura 5.6b.



(a) **(b)** **(c)**

Figura 5.5: **(a)** *Activity* - Selecionar *CheckBox* (tempo de *Standby*), **(b)** *Activity* - Interação com dispositivo remoto (Servidor), **(c)** *Activity* - Servidor em Standby

(a) **(b)**

Figura 5.6: **(a)** *Activity* - Valor de Teperatura e Humidade dos sensores do Servidor, **(b)** *Activity* - Indicação de falha na conexão com Servidor

5.2 Utilização do programa MatLab

Antes de o utilizador executar o programa MatLab, é necessário que as considerações do Apêndice D sejam verificadas. Tendo em conta que essas considerações foram verificadas, basta fazer *Run* ou F5 no *script* `HygrothermallAppUAMatLab.m` para o programa executar. Começa por verificar se o *"file_base.csv"*, existe. Se não existir, o programa termina indicando a seguinte mensagem: **'ERROR! Cannot open the file... Check the current path, or file name...'**, como se verifica na Figura 5.7. Contudo, se existir é exibido o menu inicial (Figura 5.8), que contém três possibilidades para visualizar a variáveis adquiridas (Temperatura e Humidade relativa).

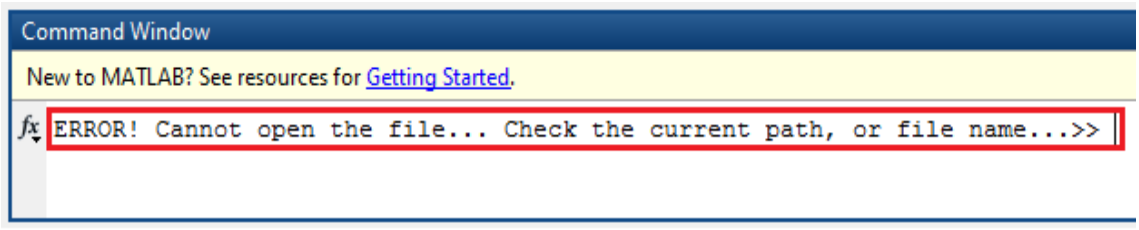


Figura 5.7: Mensagem de erro sobre *file_base.csv*

O utilizador deve indicar como pretende visualizar as variáveis adquiridas, tendo três possibilidades: d(day/dia), m(month/mês) ou y(year/ano). Se pretender terminar o programa, basta em qualquer circunstância indicar e(exit).

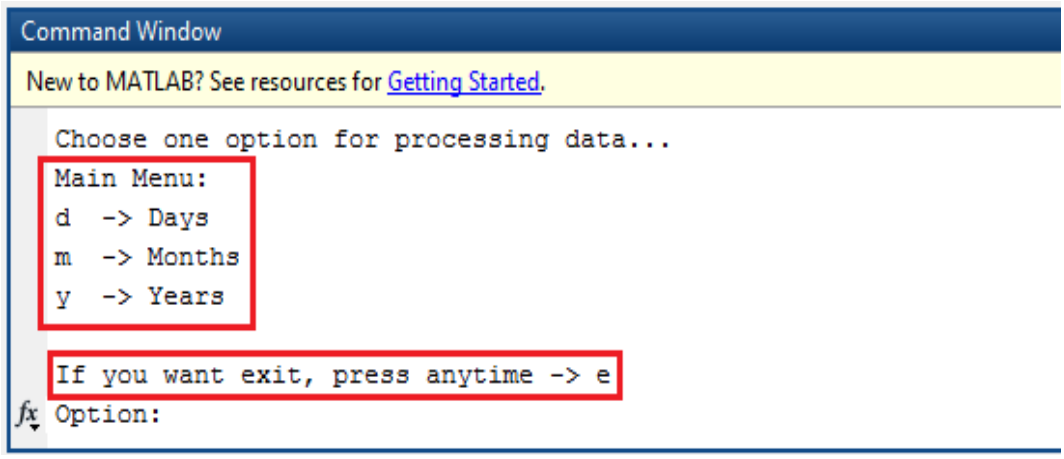


Figura 5.8: Menu inicial

Se porventura em qualquer situação, indicar outra opção diferente das anteriores, o utilizador é notificado que houve erro, sendo novamente solicitado a indicar uma opção correta (Figura 5.9).

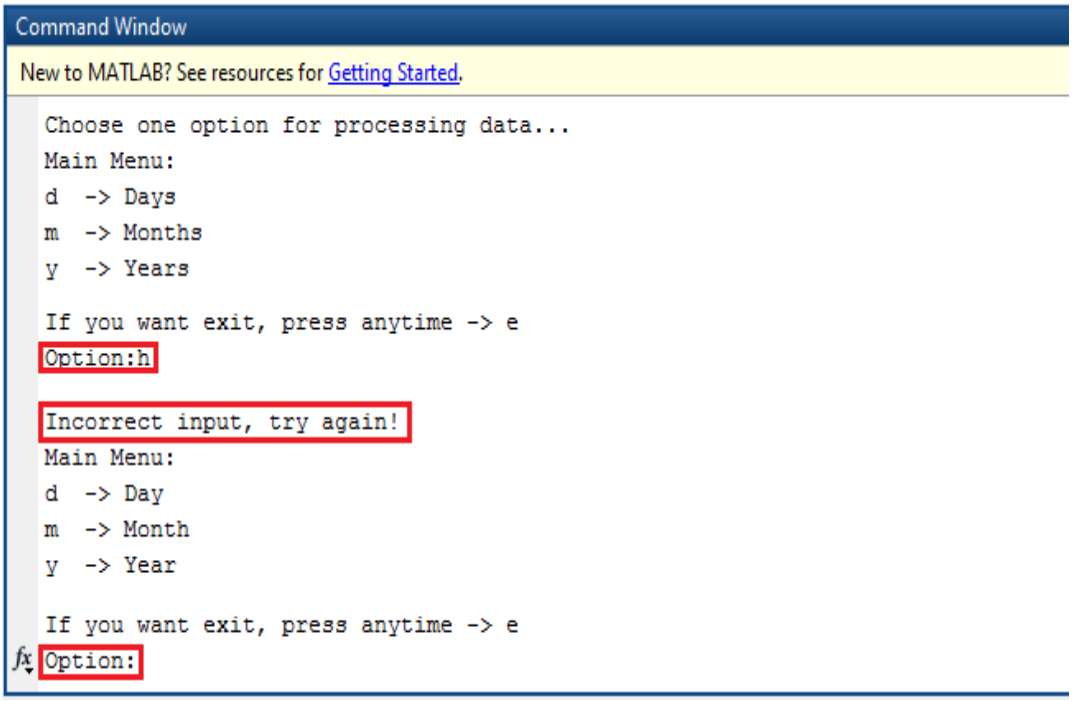


Figura 5.9: Indicação de erro no *input* inicial

Após indicar um *input* válido (neste exemplo 'd', Figura 5.10), é indicado ao utilizador a data de início e fim dos dados existentes no ficheiro, assim como o formato do novo *input*. Se a opção for 'd' o *input* terá de ser do tipo 'DD/MM/YYYY', se for 'm' será 'MM/YYYY', e por fim se for 'y' o *input* será 'YYYY'.

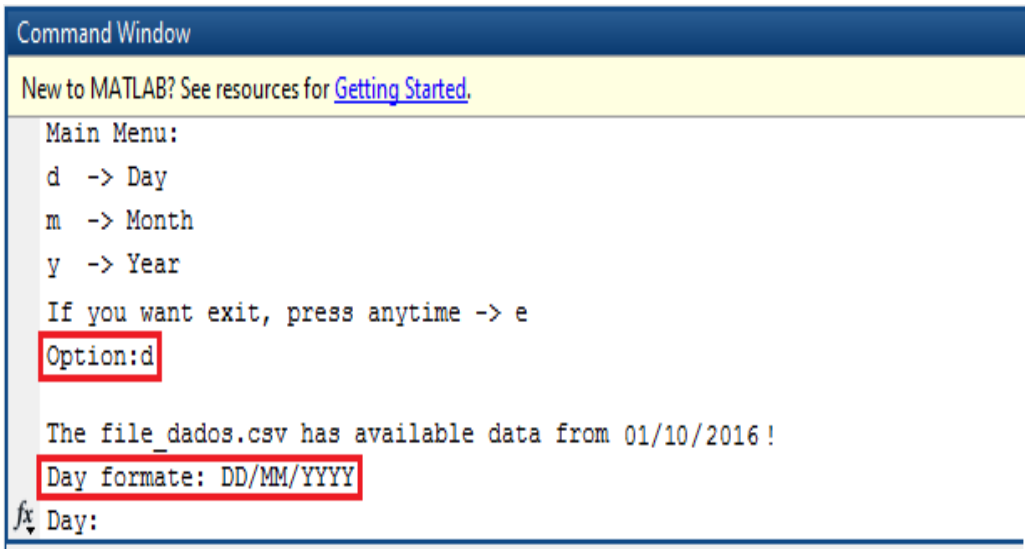


Figura 5.10: Intervalo de datas existentes em cartão e indicação de formato de *input* correto

Após indicar um *input* válido (neste exemplo '01/10/2016'), surge o menu (Figura 5.11), onde estão presentes as opções para visualizar as variáveis adquiridas.

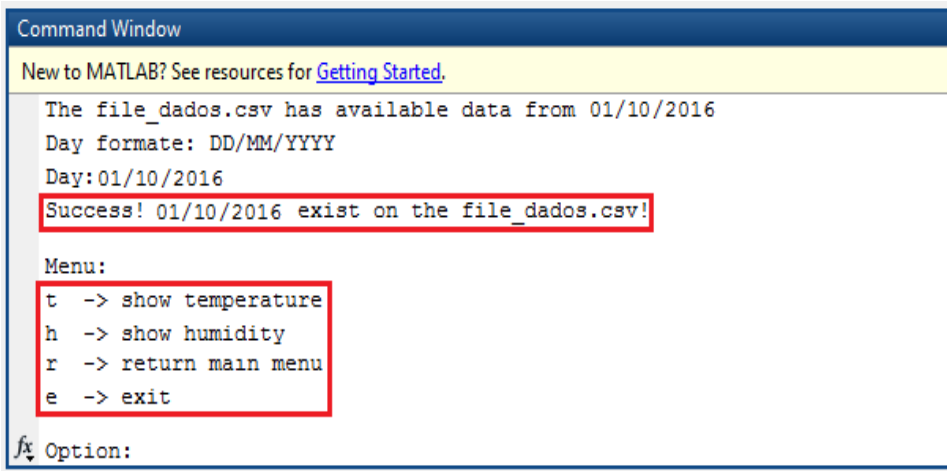


Figura 5.11: Opções para visualizar variaváis adquiridas

Através deste menu, o utilizador pode visualizar as variáveis adquiridas, no intervalo temporal selecionado anteriormente. Se não desejar terminar o programa, pode a qualquer momento regressar ao menu inicial através da opção 'r' e escolher outra opção, estando o programa sempre em *runtime*. À semelhança do menu inicial (Figura 5.9), também neste menu existe a verificação se a opção escolhida está correta. Caso não esteja, o utilizador é notificado que houve erro, e deve optar por uma opção válida, conforme se verifica na Figura 5.12.

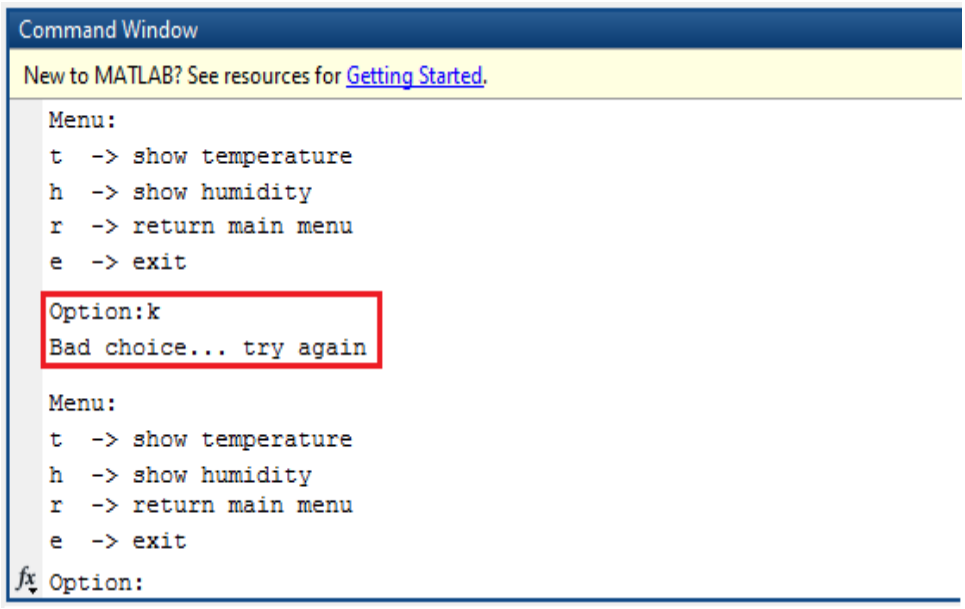


Figura 5.12: Opções para visualizar variaváis adquiridas

Caso a opção esteja correta, o utilizador visualiza a variável adquirida através dum gráfico que apresenta na legenda, a média, o ponto máximo e o ponto mínimo da variável.

São de seguida apresentados exemplos de gráficos de Temperatura em função do dia, (valores reais), do mês e do ano (dados experimentais, devido à falta de valores reais para longos períodos de aquisição).

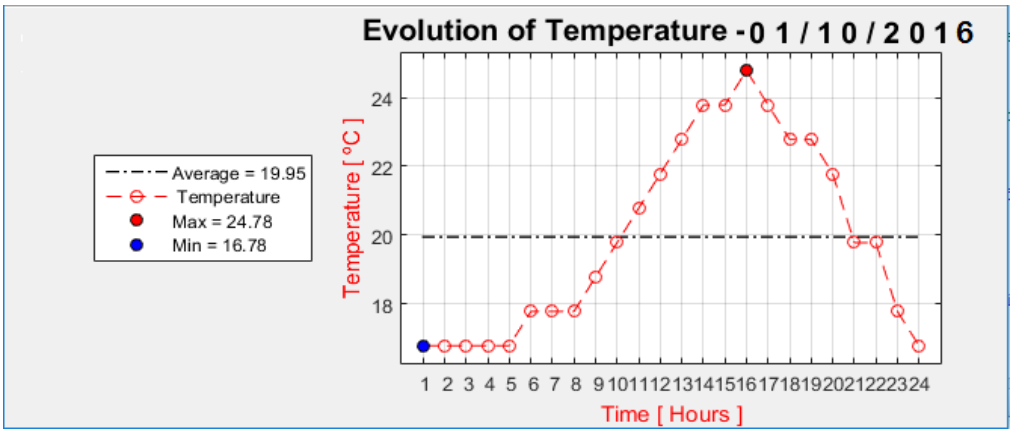


Figura 5.13: Gráfico - Evolução da Temperatura (dia)

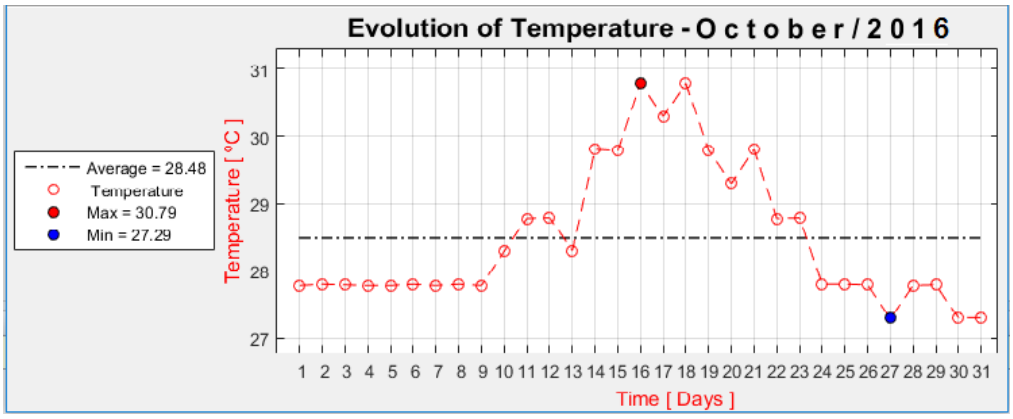


Figura 5.14: Gráfico - Evolução da Temperatura (mês)

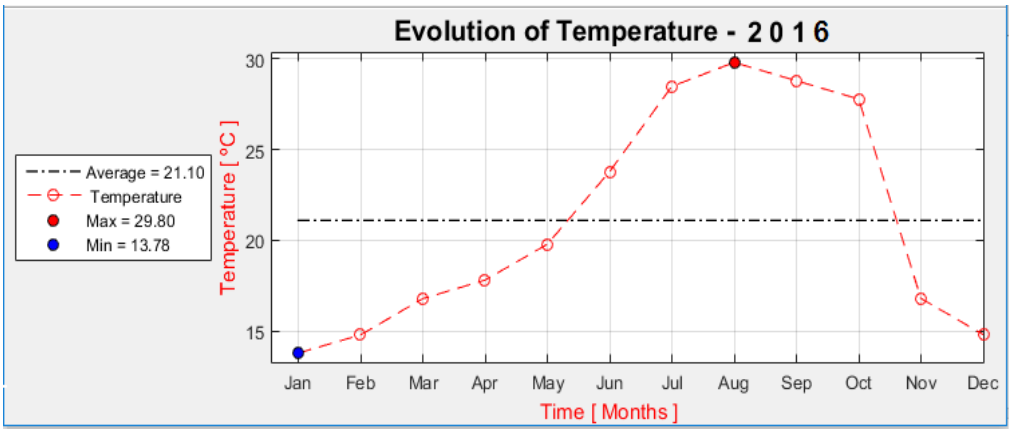


Figura 5.15: Gráfico - Evolução da Temperatura (ano)

Apresenta-se também exemplos de gráficos de Humidade (com dados experimentais, para mês e ano) em função do dia, do mês e do ano.

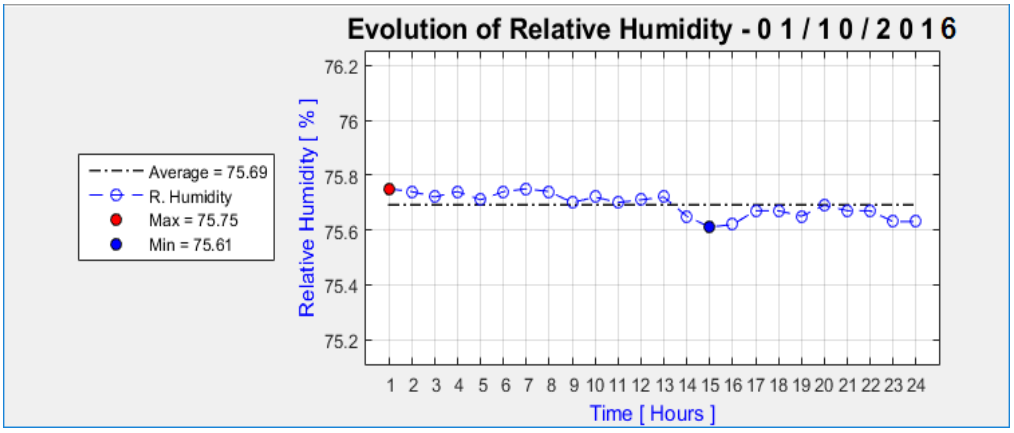


Figura 5.16: Gráfico - Evolução da Humidade (dia)

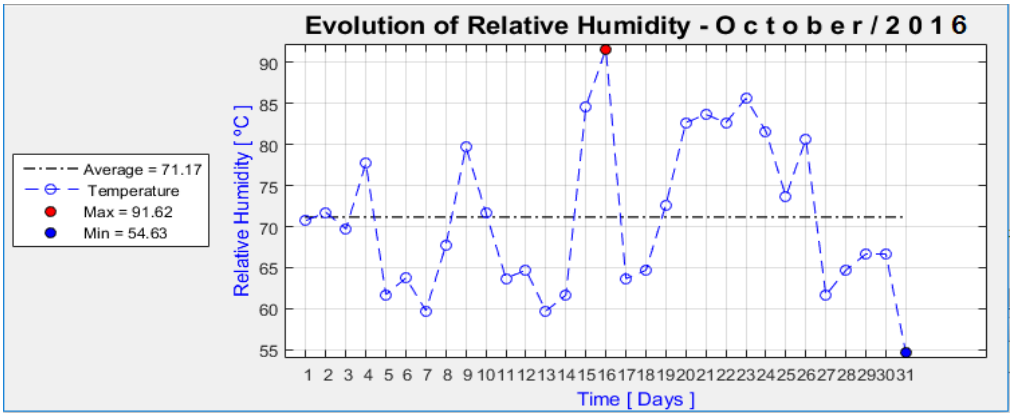


Figura 5.17: Gráfico - Evolução da Humidade (mês)

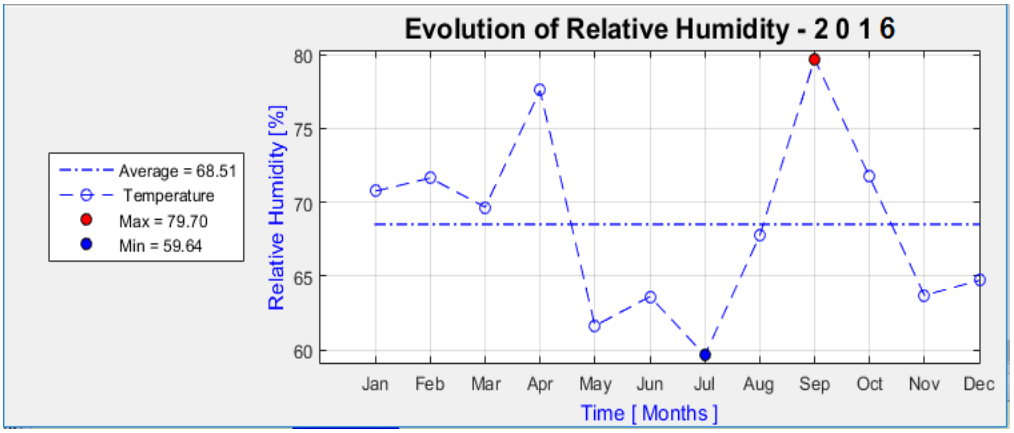


Figura 5.18: Gráfico - Evolução da Humidade (ano)

As abcissas de todos os gráficos ajustam-se consoante os dados existentes. Apresenta-se de seguida o capítulo com as conclusões do trabalho e aspetos para trabalho futuro.

Capítulo 6

Conclusão

O trabalho realizado ao longo desta dissertação permitiu desenvolver um sistema de baixo custo para medição de Higrotermia, cuja aplicação principal é a monitorização de Temperatura e Humidade relativa em edifícios no âmbito da Engenharia Civil. Contudo é também uma solução viável como suporte à domótica. O sistema é baseado numa arquitetura Cliente/Servidor, constituído por um dispositivo de sensores remoto (Servidor), por uma aplicação Android (Cliente) e por um programa MatLab (para processamento das variáveis Temperatura e Humidade relativa).

O Cliente e o Servidor interagem por *WiFi* através do protocolo IEEE 802.11, via UDP (a interação/comunicação só funciona dentro da mesma LAN). Optou-se por este protocolo *WiFi*, devido ao facto da rede estar omnipresente, existe uma estrutura edificada na maioria dos edifícios permitindo flexibilidade na conexão e mobilidade aos utilizadores.

Utilizaram-se componentes da plataforma Arduino para a construção do Servidor, devido ao facto de a plataforma disponibilizar um conjunto alargado de *shields*, não sendo necessário adicionar *hardware* externo, tendo em conta a existência de *shields* muito completas, com elevada compatibilidade na montagem em "cascata" e também por ser *open source*. De modo a reduzir o consumo energético do Servidor, procedeu-se à redução da frequência de *clock* do Arduino Uno dos 16MHz para 8MHz (esta ação não afeta o desempenho pretendido).
A função do Servidor é adquirir e registar Temperatura e Humidade relativa durante o tempo de aquisição, sendo este no máximo 1 minuto. Após o tempo de aquisição, o Servidor fica em *Standby* durante um período escolhido pelo utilizador e enviado através da aplicação Cliente. Esse tempo (em minutos) de *Standby* pode ser um de quatro hipóteses, nomeadamente 60, 30, 20 ou 10. O utilizador opta pelo que entender mais adequado, podendo a qualquer momento modificar o período de *Standby*, enviando novo comando.

Relativamente a aplicação Cliente, foi desenvolvida sobre a plataforma Android, devido ao facto de este sistema operativo móvel obter a maior fatia de utilizadores a nível mundial e dessa forma a aplicação `HygrothermalAppUA` poder estar disponível para milhões de utilizadores. A aplicação centra-se essencialmente em registar Servidores numa base de dados SQLite, para que sempre que o utilizador necessite de interagir com um Servidor o contenha nessa base de dados, bastando selecionar o Servidor pretendido e realizar uma ação sobre o mesmo, que pode ser solicitar nesse instante (em tempo real) o valor de Temperatura e Humidade relativa, alterar o período de *Standby*, ou simplesmente interromper as atividades do Servidor.

As variáveis adquiridas pelo Servidor (existentes no ficheiro *"file_base.csv"*), são depois processadas num computador, através da execução do *script* `HygrothermalAppUAMatLab.m`, programa MatLab que permite ao utilizador fazer gráficos de Temperatura e Humidade relativa em função do tempo, que pode ser ao dia, ao mês ou ao ano. Optou-se por esta ferramenta de engenharia, porque está otimizada para fazer cálculos matemáticos e para a representação gráfica de dados (gráficos).

A solução global caracteriza-se por ser de baixo custo, devido aos componentes utilizados, sendo um sistema fiável, compacto e *user friendly*.

6.1 Trabalho Futuro

Como referido anteriormente, o sistema desenvolvido cumpre os objetivos propostos 1.3, contudo é possível fazer melhoramentos. São de seguida mencionados três aspetos importantes a desenvolver como trabalho futuro, sendo os seguintes:

1. Dotar o Servidor e o Cliente com capacidade de comunicarem em redes WAN (Wide Area Network), tornando o sistema mais IoT *(Internet of Things)*. Explorar as potencialidades das seguintes tecnologias: LPWAN, LoRaWAN, SIGFOX, LoRa e NB-IoT.
2. GUI *(Graphical User Interface)* para o programa MatLab, de forma a o tornar mais apelativo e próximo de uma solução comercial.
3. Construção de protótipo pré-industrial do sistema.

Apêndice A

Sinopse - Plataforma Android

A.1 Plataforma Android

Android é um sistema operativo *open source*[1] baseado em *Kernel Linux 2.6*, projetado principalmente para aparelhos móveis *touchscreen*, como *Smartphones* e *Tablets*. Desenvolvido inicialmente pela Android *Incorporated*, (fundada em Palo Alto, Califírnia), por *Andy Rubin, Rich Miner, Nick Sears e Chris White* em 2003, sendo adquirida pela Google em 2005. Foi apresentado em 2007, juntamente com a fundação da *Open Handset Alliance*[2].

O *Android SDK*[3] fornece as ferramentas e APIs[4] necessárias para começar a desenvolver aplicações na plataforma Android, através da linguagem de programação Java e XML.
Android OS inclui um conjunto de bibliotecas C/C++, destacando-se as seguintes: *System C library, Media library, Surface Manager, LibWebCore, SGL, SQLite, FreeType and 3D libraries* e utiliza o armazenamento de dados numa base de dados SQLite.

Desenvolveram também o seu próprio motor de *Java runtime*, otimizado para os recursos limitados disponíveis numa plataforma móvel chamada de *"Dalvik Virtual Machine"*.

A.2 Componentes da aplicação

O intuito desta secção, consiste em definir os principais componentes da aplicação desenvolvida de forma não exaustiva e o contexto para o qual foram necessários.

Activity

È uma representação gráfica (tela) que permite ao utilizador interagir com a aplicação. A Figura A.1, apresenta o ciclo de vida completo (*entire lifetime)* de uma *Activity*.

[1] *Software* pode ser livremente modificado e distribuído, por fabricantes, operadoras de telefonia, programadores, etc.
[2] Aliança de diversas empresas com a intenção de criar padrões abertos para telefonia móvel.
[3] *Software Development Kit* - ferramenta para criar aplicações para um determinado pacote de *software*.
[4] *Application programming interface* - rotinas, protocolos e ferramentas para a construção de aplicações de *software*.

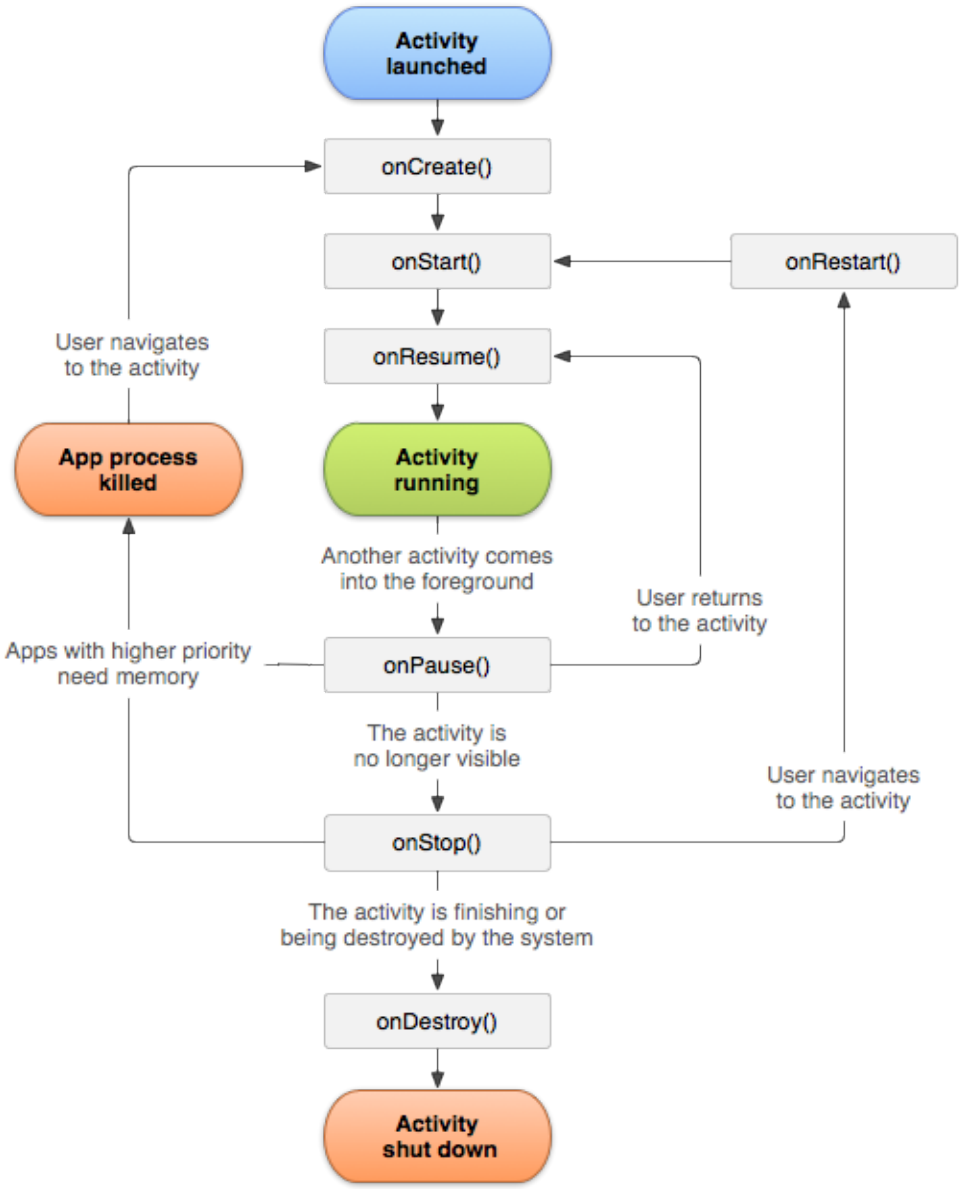


Figura A.1: Android - ciclo de vida de uma *Activity* [Htt16a]

Cada *Activity* pode iniciar outra *Activity*, com o propósito de executar outras ações. São geridas sob a forma de pilha *(stack)* de *Activities*. Quando uma *Activity* é inicializada, passa para o topo da pilha e passa a ser a atividade que está a correr *(running Activity)*. A *Activity* anterior está na posição imediatamente anterior da pilha, e voltará para o topo quando a *Activity* atualmente a correr terminar a sua execução. Uma Activity irá executar tudo o que for "global" em onCreate() e libertar todos os seus recursos em onDestroy().
Conforme se verifica na Figura A.1, uma *Activity* possui os seguintes métodos: [Htt16a]

onCreate()

O primeiro evento utilizado quando iniciamos a *Activity*, no momento da sua criação. Neste método é passado um objeto do tipo "Bundle" contendo o estado anterior duma *Activity*, sendo sempre seguido pelo onStart().

onStart()

Executado quando a *Activity* está visível para o utilizador. Utiliza-se o onResume() se a *Activity* vem para o primeiro plano, ou o onStop() quando não está visível.

onRestart()

Chamado quando uma *Activity* parou por algum motivo. Executa o OnStart() de forma automática.

onResume()

Chamado apenas antes da *Activity* começar a interagir com o utilizador. Neste ponto, a *Activity* está no topo da pilha de *Activities*.

onPause()

Chamado quando o sistema está prestes a começar a retomar outra *Activity*.

onStop()

Este método é chamado quando a *Activity* for encerrada.

onDestroy()

Método responsável por libertar da memória a *Activity*, sendo a última chamada que a mesma irá receber.

Intent

È um objeto de mensagem que pode ser usado para solicitar uma ação de outro componente da aplicação. Embora as *Intents* facilitem a comunicação entre componentes de diversos modos, há três casos de uso fundamentais: a) iniciar uma Activity, b) iniciar um serviço e c)

fornecer uma transmissão. Na aplicação desenvolvida a *Intent* foi usada para iniciar *Activities*, através do click em botões.

Toast

È um recurso que serve para exibir (geralmente no rodapé da *Activity*) mensagens curtas e temporais (ao fim de segundos desaparecem). Foram utilizados para indicar ao utilizador que o dispositivo foi introduzido corretamente na base de dados e para indicar que deve selecionar um dispositivo (quando o utilizador está a percorrer a lista de dispositivos).

AlertDialog

São pequenas janelas que levam o utilizador a tomar uma decisão ou a inserir informações adicionais. Não ocupam a tela toda e são normalmente usadas para situações que exigem que o utilizador realize uma ação antes de continuar.

Para a aplicação foram usadas em várias situações de alerta, nomeadamente: *WiFi* desligado, erros na introdução de dados na base de dados e ausência de seleção de *CheckBox* quando utilizador pretender enviar tempo de *standby*.

Button

Botão pode ter vários estilos, contudo tem maioritariamente a função de executar uma ação ao ser pressionado. Foram utilizados vários botões, tendo como função principal navegar pelas *Activities* e validar opções do utilizador em cada *Activity*.

CheckBox

Widget específico de dois estados, que pode estar habilitado ou desabilitado. Foram usados *CheckBoxs* para o utilizador escolher o tempo que o dispositivo deve estar em *standby*.

ListView

È uma lista de itens. São inseridos automaticamente através de um *ArrayAdapter*. Foi usado para mostrar na tela os dispositivos existentes na base de dados.

Apêndice B

Ferramentas, compilação, execução e transferência de programas para *smartphones Android*

A aplicação `HygrothermalAppUA`, foi desenvolvida no IDE primário/oficial da Google nativo para Android, denominado de Android Studio. Pode ser baixado através do link IDE-Android Studio e está disponível para Windows, MacOS e Linux. Além do IDE oficial, o desenvolvimento de aplicações Android requer um conjunto de ferramentas, nomeadamente:

- JDK - *Java Development Kit* (Máquina virtual Java)
- SDK - *Software Development Kit*

O JDK pode ser baixado através do link JDK.
Por sua vez, o SDK está disponível através do link SDK.
Após a instalação do IDE e das ferramentas de desenvolvimento, estão reunidas as condições para o desenvolvimento de aplicações Android.
Disponibiliza-se o seguinte link pplware que demonstra as etapas necessárias para criar uma aplicação simples em Android Studio.

De seguida na Figura B.1, é demonstrado o panorama do IDE Android Studio, dando ênfase numa primeira instância ao AVD (Android Virtual Device) utilizado na criação desta aplicação, o SamsungS4Mini, assim como a localização dos principais ficheiros, nomeadamente classes Java, *layouts* xml e os princiapis *widgets*.

Na secção B.1 procura-se demonstrar como o projeto se encontra organizado.

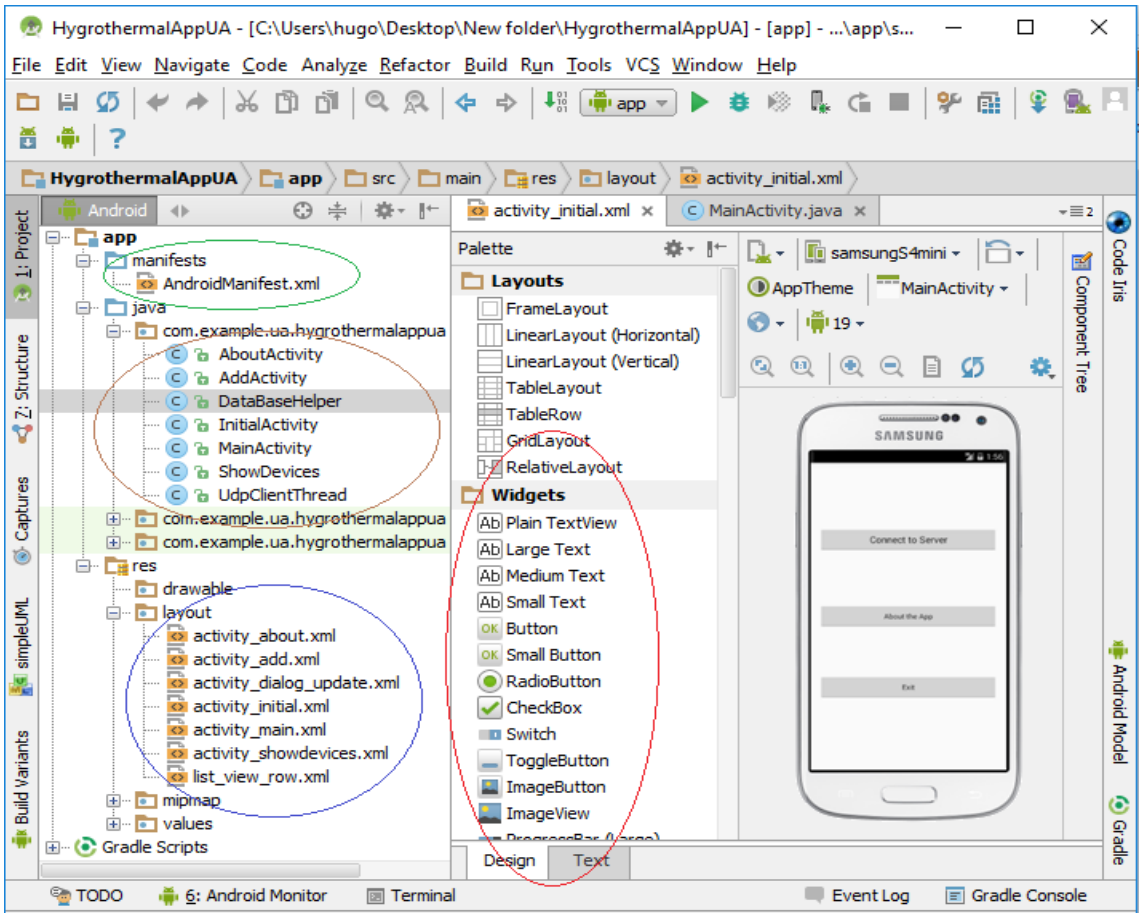


Figura B.1: IDE - Android Studio

B.1 Estrutura do projeto

Os projetos no Android Studio, por padrão estão organizados em módulos para permitir o acesso rápido aos principais arquivos do projeto. Os três módulos com mais utilização são o **manifests**, o **java** e o **res**.

O módulo **manifests** contém um arquivo muito importante, designado de AndroidManifest.xml, pois contém todas as configurações da aplicação, nomeadamente o nome das classes de cada *Activity*, conceder permissões para usar funcionalidades do dispositivo (Internet, GPS, SMS, Bluetooth, etc.), entre outras configurações.

Por sua vez, o módulo **java** contém os arquivos de código fonte do Java, ou seja as classes de cada *Activity*.

Por ultimo, o modulo **res**, organizado em várias pastas, contém todos os recursos que não são código, como *layouts* XML, entre outros.

De seguida na Figura B.2 são identificadas as áreas importantes a ter em conta no desenvolvimento de aplicações.

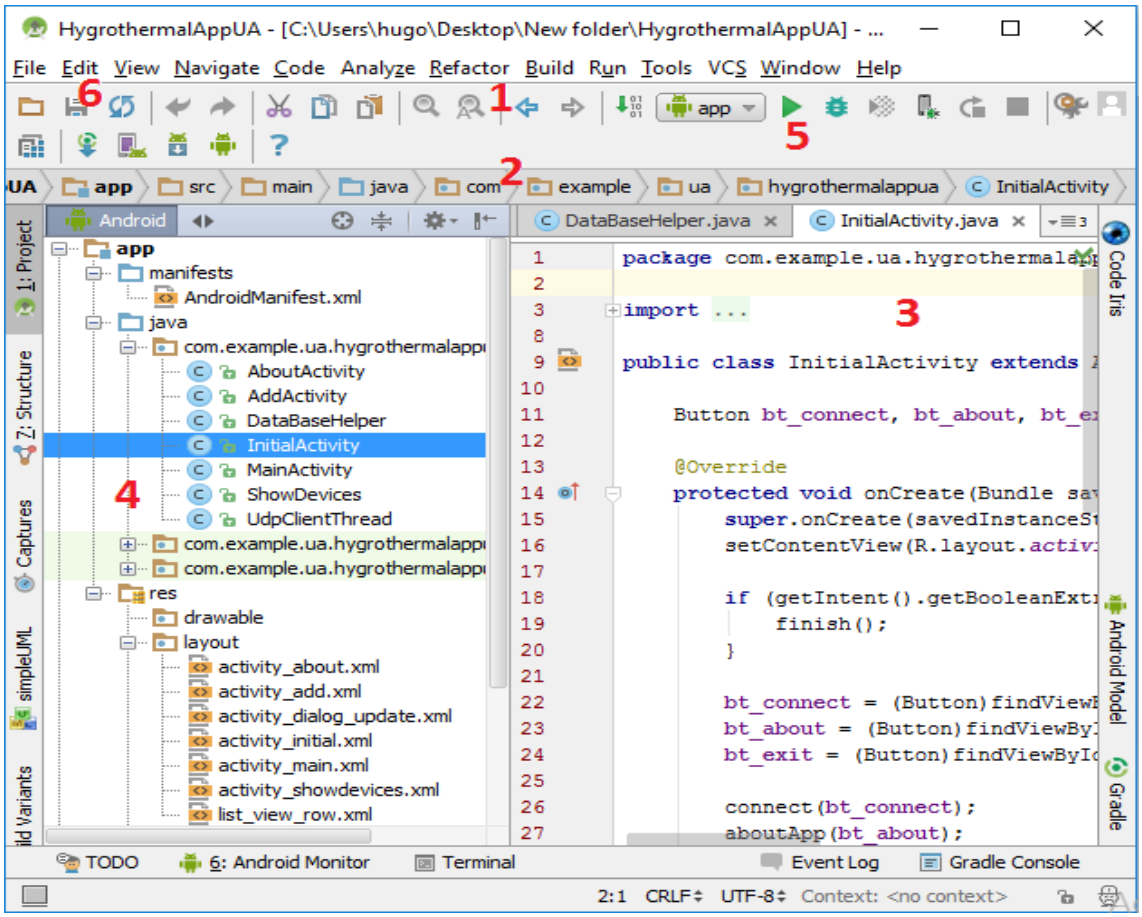


Figura B.2: Identificação de áreas importantes do IDE

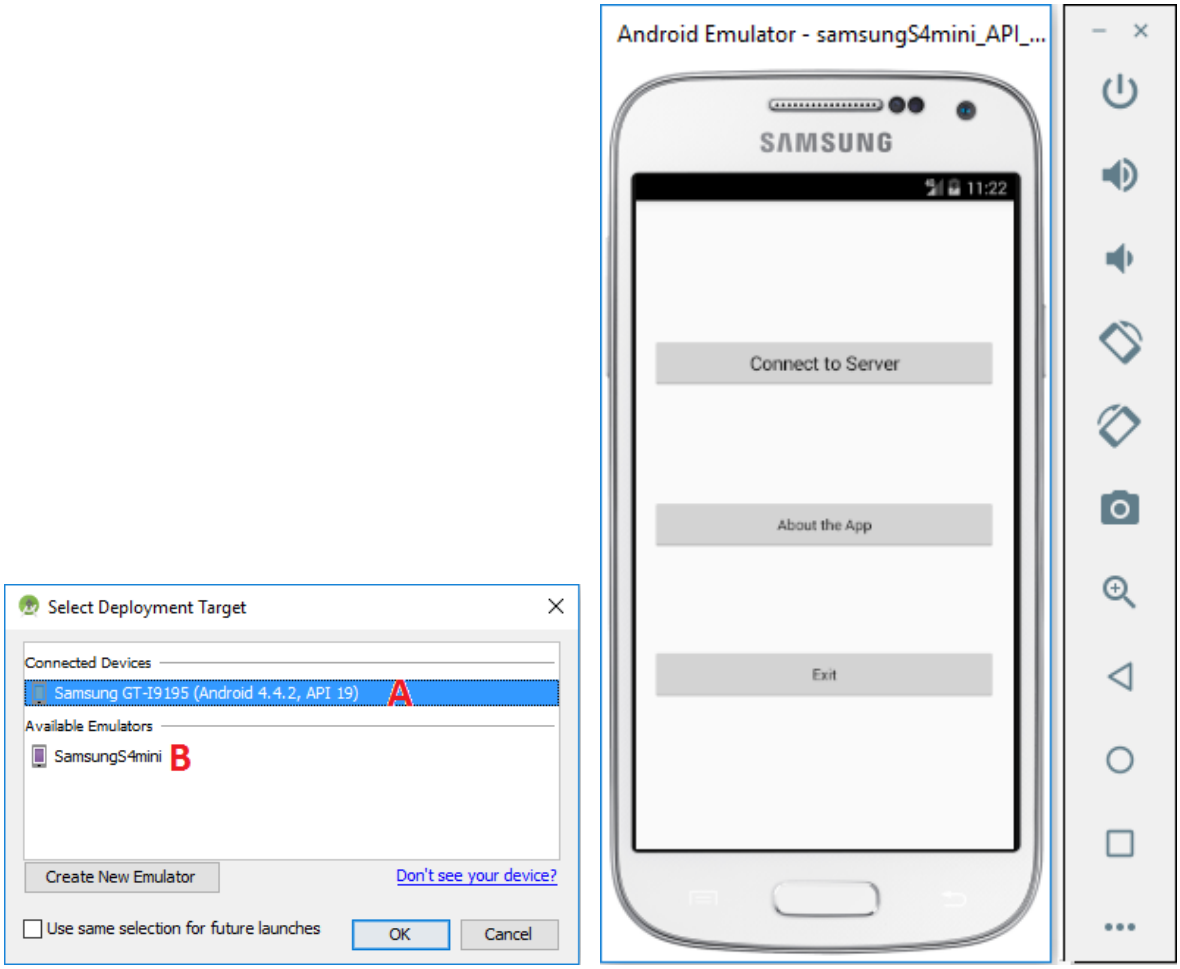
De acordo com a Figura B.2, verifica-se que estão identificadas nove áreas que são necessárias ter em conta ao desenvolver aplicações, sendo as seguintes:

1. **barra de ferramentas** - permite executar uma grande variedade de ações
2. **barra de navegação** - ajuda a navegar pelo projeto e a abrir arquivos para edição
3. **janela do editor** - local onde se escreve ou modifica código
4. **janelas de ferramenta** - local onde estão os módulos anteriormente referidos
5. **botão Run 'app'** - permite correr a aplicação em emulador ou dispositivo real
6. **botão Save all** - permite guardar todos os arquivos do projeto

Mediante os conceitos apresentados anteriormente, já é possível ter uma visão geral do IDE, podemos partir agora para a execução da aplicação. Na secção B.2 ilustra-se o procedimento para executar a aplicação, seja em modo emulador, seja em dispositivo real.

B.2 Execução de aplicação

O processo para executar a aplicação é simples e consiste em pressionar o botão Run 'app' (identificado na Figura B.2 pelo nº5) ou pressionando as teclas Shift+F10. Após esta operação surge a seguinte janela, apresentada na Figura B.3a.



(a) **(b)**

Figura B.3: **(a)** Escolha de dispositivo para execução da App, **(b)** Emulador Samsung S4 mini

Como se verifica na Figura B.3a, estão disponíveis (neste exemplo) um emulador e um dispositivo real, sendo ambos Samsung S4 mini. Se se pretender executar a aplicação através de emulador, deve-se selecionar a opção *B* indicada nessa figura. Dessa forma visualiza-se o efeito produzido pelo código desenvolvido, tendo a possibilidade de testar a aplicação. A Figura B.3b, ilustra o emulador, que permite ao programador testar a aplicação com um "dispositivo virtual". O emulador permite testar grande parte das funcionalidades da App, como botões, abrir novas *Activities*, contudo esta opção é limitada para, por exemplo (nesta App) testar a comunicação *WiFi*. Devido a essa razão, e também por se necessitar de perceber

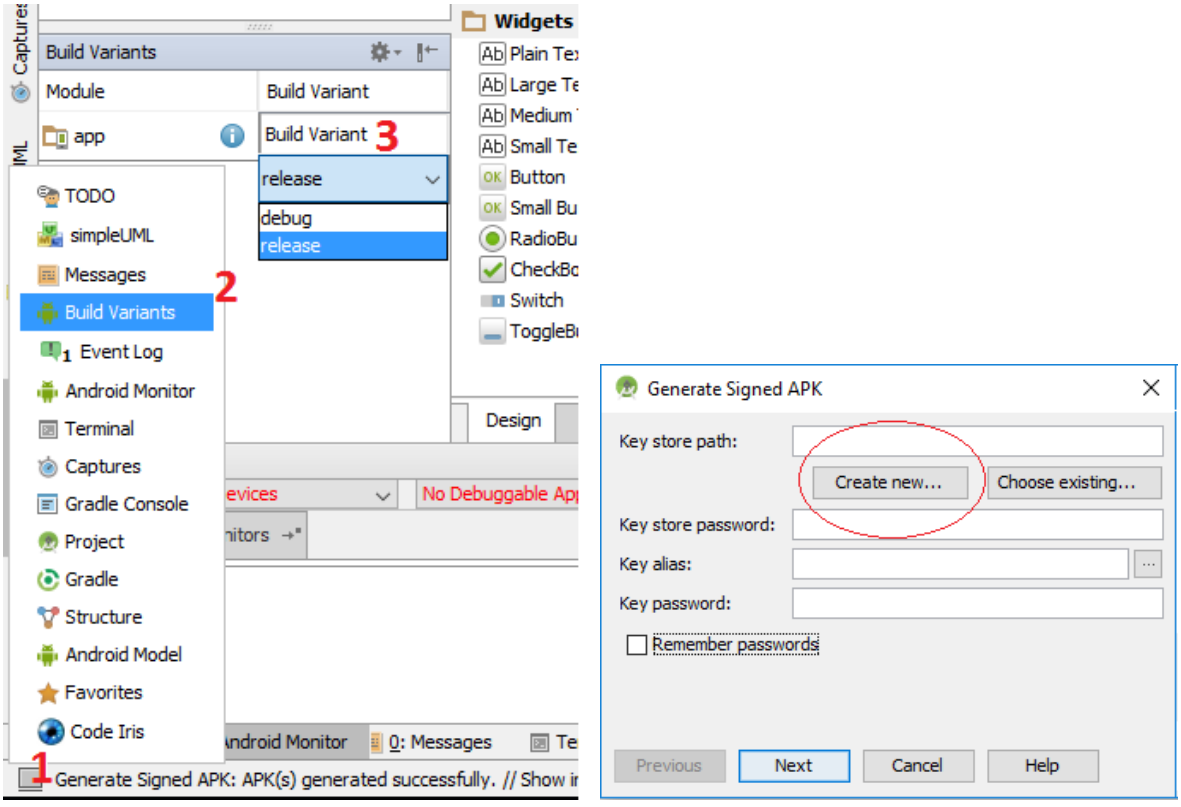
se a aplicação está 100% funcional, recorreu-se a execução da aplicação em dispositivo real (enfatiza-se que o dispositivo real deve ter a opção "USD debugging" ativada). Salienta-se que nesta etapa o dispositivo real Android deve estar conectado por cabo ao computador que contém a aplicação e ao fazer Run 'app', deve ser escolhida a opção *A* da Figura B.3a. Dessa forma, a aplicação será automaticamente instalada no dispositivo real, possibilitando que todas as funcionalidades sejam testadas.

Outra forma de baixar aplicações é utilizando o Google Play. Apresenta-se de seguida na secção B.3, o processo para gerar o ficheiro .apk, uma outra forma de distribuir a aplicação.

B.3 Gerar assinatura do APK

APK é um formato de ficheiro utilizado para instalar aplicações no sistema operativo Android. Pode ser uma App ou um jogo, e é necessário gerar um APK com assinatura para publicar na Google Play.

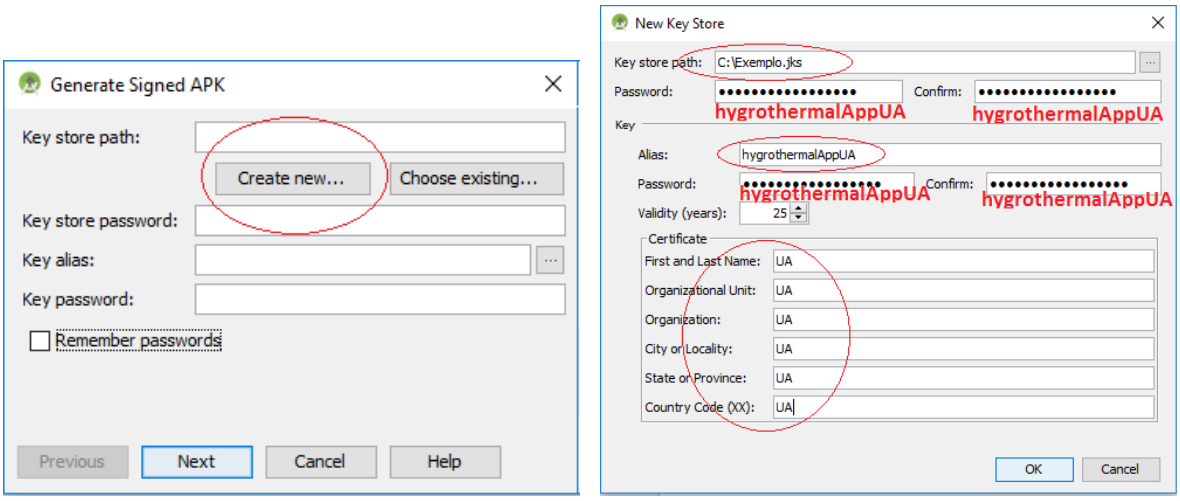
Para gerar esse ficheiro no Android Studio, deve-se alterar em primeiro lugar o *"Build Variants"*, de *debug* para *release*. A Figura B.4a ilustra esta operação.



(a) **(b)**

Figura B.4: **(a)** *debug to release*, **(b)** *Generator Signed APK*

De seguida é necessário ir ao Menu→Build→Generate Signed APK..., abrindo a janela que se ilustra na Figura B.4b.



(a) **(b)**

Figura B.5: **(a)** Criar signed APK, **(b)** *Generator Signed APK*

Clicar em "Create new..." (Figura B.4b) de modo a criar uma Key Store. Surge a janela da Figura B.5b, que ilustra um exemplo de preenchimento dos campos. Para concluir, clicando em "Ok", surge a derradeira janela, visível na Figura B.6.

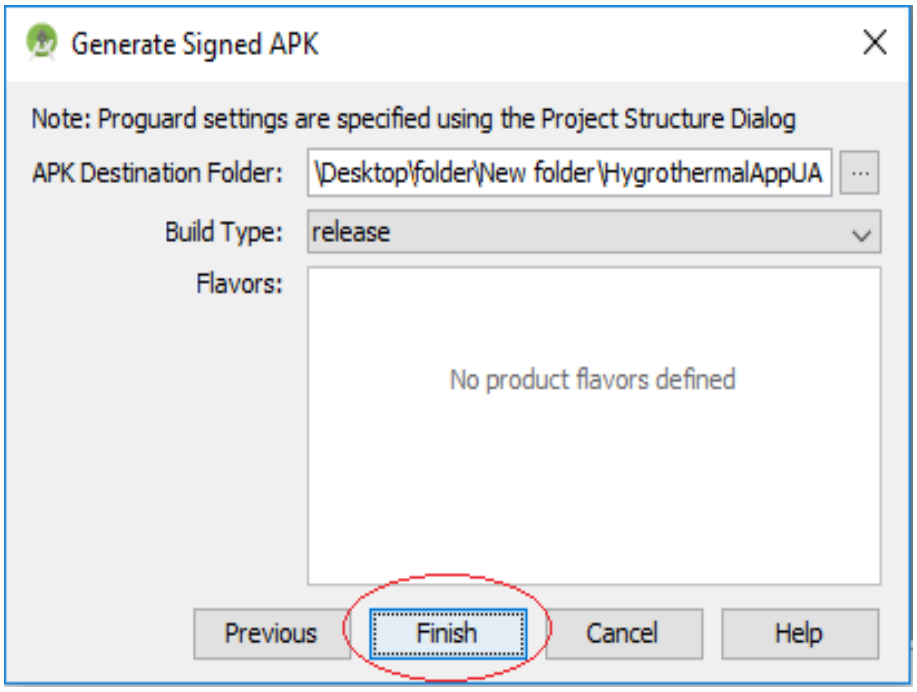


Figura B.6: Gerar .apk

Deve-se ter atenção e escolher o 'APK Destination Folder' de acordo com o pretendido, pois é para este local que o ficheiro .apk será gerado. Para isso é necessário clicar no botão

'Finish' dessa mesma figura. O ficheiro .apk, está agora disponível no 'APK Destination Folder' definido anteriormente, e pronto a ser instalado.

O Android para o qual se pretende instalar o .apk, deve estar configurado para permitir a instalação de ficheiros desconhecidos. Para isso, deve-se ir a Definições→Segurança→ e colocar visto em "fontes desconhecidas" (esta sequência pode variar com a versão do sistema operacional Android).

Caso o *smartphone* não tenha um explorador de arquivos de sistema, é necessário ir ao Google Play e instalar, por exemplo o *Astro File Manager*, que permitirá executar o .apk da aplicação. Dessa forma a aplicação já poderá ser executada a partir do dispositivo real.

Apêndice C

Compilação, execução e transferência de programas para a plataforma Arduino

O programa `HygrothermalAppUA_Arduino.ino` foi desenvolvido no IDE oficial da plataforma Arduino, Figura C.1 podendo este ser baixado através do seguinte link IDE-Arduino.

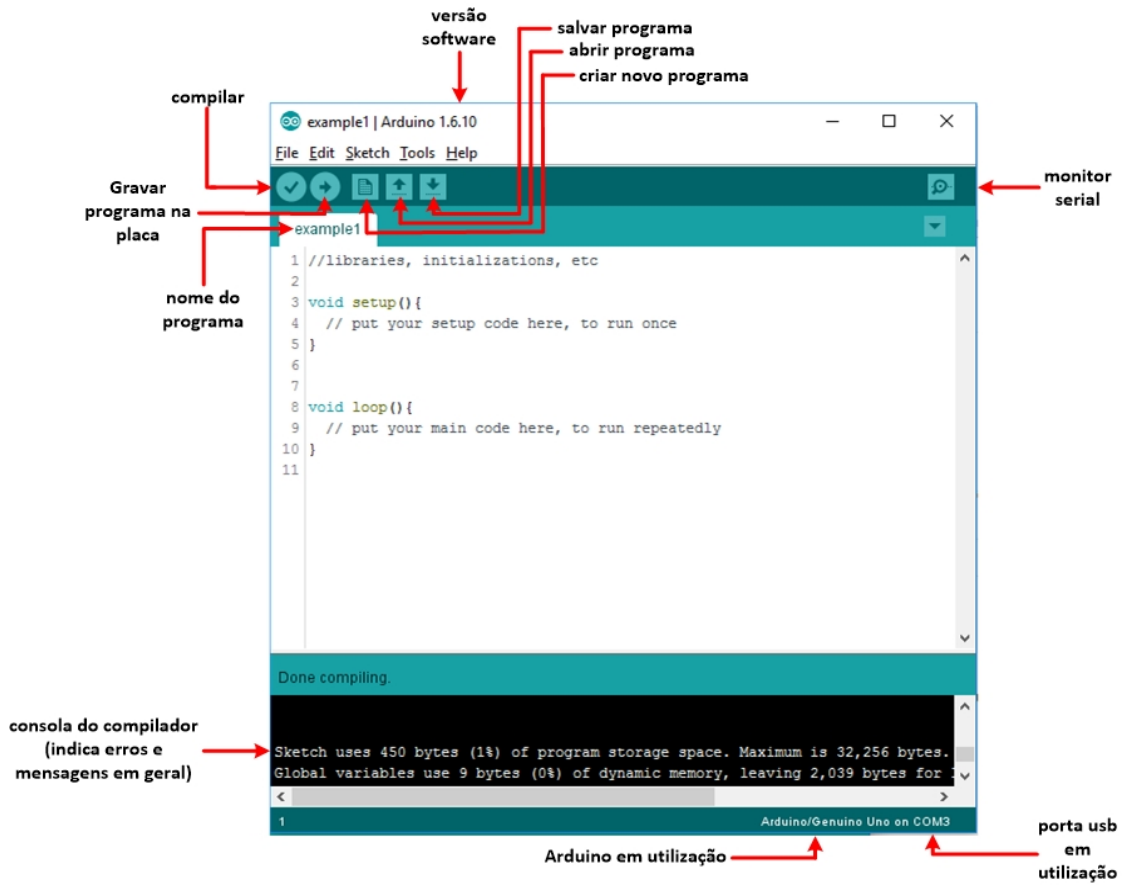


Figura C.1: IDE oficial - plataforma Arduino

A Figura C.1, ilustra a interface genérica do IDE Arduino. Identificam-se os principais comandos necessários a desenvolver programas, assim como o local onde o programa é desenvolvido, tendo inclusivamente na imagem o prototipo das funções principais de qualquer programa Arduino, nomeadamente as funções *void setup()* e *void loop()*.

A primeira recomendação e ação a ser feita pelo programador após abrir o IDE, é guardar o programa com um nome adequado e num local que ache conveniente. Para isso deve executar o atalho Ctrl+Shift+S ou ir ao Menu→File, escolher "Save As..." e definir o local para o guardar.

Antes ou após o programa ser compilado, (ver comando Figura C.1) e se pretenda enviar o programa, deve-se ter em conta a versão do Arduino para o qual se está a desenvolver o programa. A Figura C.2 demonstra como se seleciona a placa pretendida, sendo neste exemplo selecionada a placa Arduino/Genuino Uno.

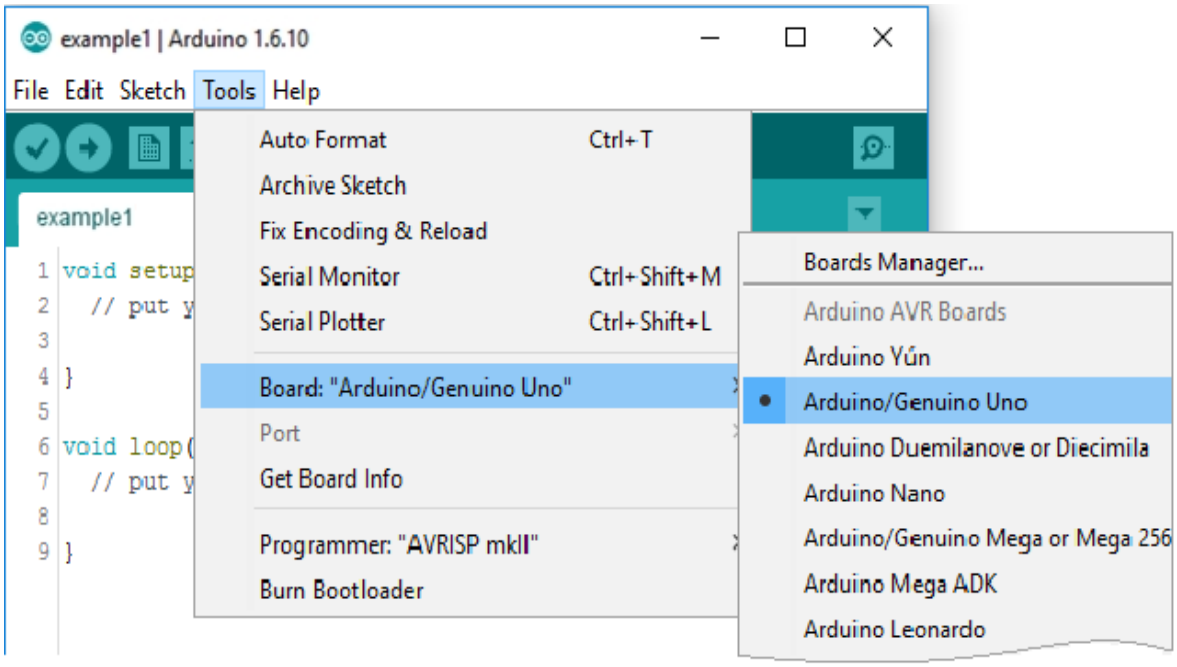


Figura C.2: Selecionar placa de desenvolvimento

Nesta etapa, já estão reunidas as condições para enviar o programa para a versão Arduino selecionada. Deve-se conectar o cabo de programação USB Arduino do computador (onde o código foi desenvolvido) com a placa Arduino. Estando esta operação feita, basta clicar no botão denominado de "Gravar programa na placa" ilustrado na Figura C.1 e o programa será enviado. Após esta operação tenha sido realizada com sucesso, o Arduino está apto a realizar as funções para o qual foi programado.

Apêndice D

Considerações antes da execução do programa MatLab

Quando o utilizador pretender analisar o conteúdo do cartão SD (existente em cada Servidor) deve colocar o cartão no ***SD card slot*** de um computador que tenha o MatLab instalado, e copiar o ficheiro *"file_base.csv"* (contém o valor das variáveis adquiridas) para o *path (Current Folder)* onde se encontra o *script* `HygrothermallAppUAMatLab.m` (programa principal), de modo a conter os ficheiros da Figura D.1, assinalado com circulo vermelho.

O programa principal é constituído por diversas funções auxiliares e por questões de organização foram colocadas numa pasta chamada *functions*. Para que essas funções possam ser invocadas pelo programa principal é necessário que o utilizador adicione essa pasta *functions* ao *path* do MatLab. Para isso deve clicar com o botão direito do rato em *functions* (assinalado com circulo vermelho, Figura D.1) para poder aceder ao menu que permite escolher a opção *Selected Folders* (assinalado com circulo preto, Figura D.1).

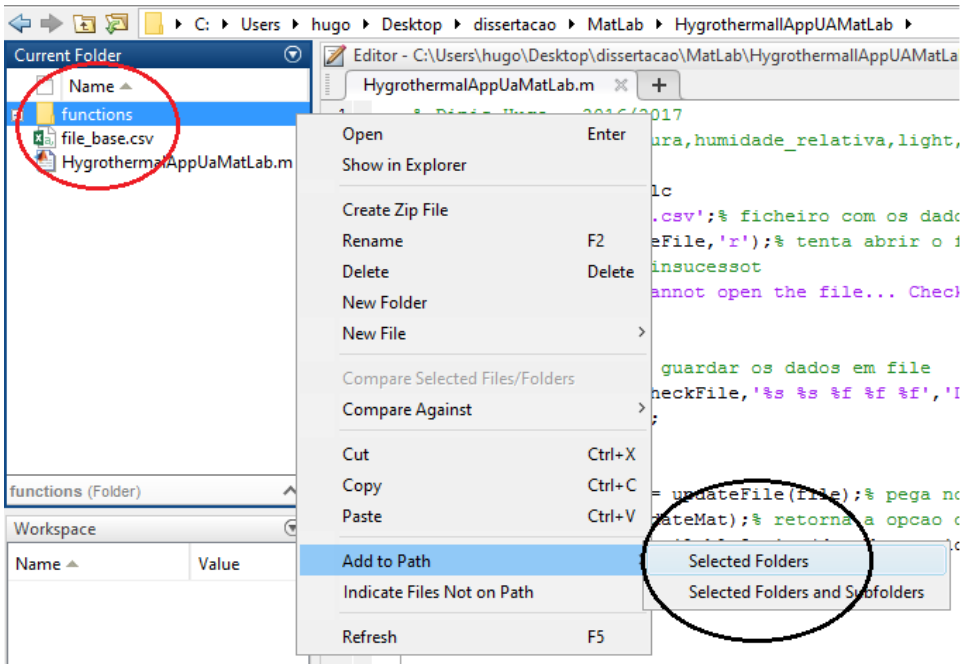


Figura D.1: Adicionar pasta *functions* ao *path* do MatLab

Bibliografia

[Ana07] Alcinda Maria da Costa Anacleto. *Temperatura e sua medição*. Dissertação, Universidade do Porto, 2007.

[Ard16] Arduino Uno. https://www.arduino.cc/en/Main/ArduinoBoardUno. Consultado em 17-05-2016, 2016.

[Ass09] KNX Association. KNX System Specifications. *KNX Standard*, V3.0:1–26, 2009.

[Bel11] Stephanie Bell. Guide to humidity measurement, 2011.

[Cat10] Catarina Abreu. *O Ambiente interior e a saúde dos ocupantes de edifícios de habitação*. Dissertação, Universidade da Beira Inteior, 2010.

[DP04] César Luiz De Azevedo Dias and Nélio Domingues Pizzolato. Domótica: Aplicabilidade e Sistemas de Automação Residencial. *Vértices*, 6(3):10–20, 2004.

[DPG06] A R Delgado, R Picking, and V Grout. Remote-controlled home automation systems with different network technologies. 2006.

[EH12] Ahmed Elshafee and Karim Alaa Hamed. Design and Implementation of a WiFi Based Home Automation System. *World Academy of Science, Engineering and Technology*, 6:1856–1862, 2012.

[Far12] Mohamed El-Habrouk Farouk AbdAllah Salem, Heba kamal sheasha. A Survey of Smart Homes Technologies, 2012.

[GP10] Carles Gomez and Josep Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6):92–101, 2010.

[Htt16a] Android Developer . Https://developer.android.com/index.html. Consultado em 18-05-2016, 2016.

[Htt16b] Arduino Uno Rev3. Https://www.robomart.com/image/catalog/RM0058/02.jpg. Consultado em 18-05-2016, 2016.

[HWB07] Fred R. Hilgeman, Brent Wilson, and Gary Bertrand. Using Dalton's Law of Partial Pressures To Determine the Vapor Pressure of a Volatile Liquid. *Journal of Chemical Education*, 84(3):469–470, 2007.

[JC 16] JC Características da Domótica - http://domoticainteligente.blogspot.pt. Consultado em 19-01-2016, 2016.

[JFR06] JFR. Software Design Specification - Z-Wave Protocol Overview. *Zensys*, pages 1–10, 2006.

[JN02] Renato Jorge and Caleira Nunes. Análise Comparativa de Tecnologias para Domótica - Instituto Superior Técnico. 2002.

[KA12] Sandeep Kumar and Mohammed Abdul Qadeer. Application of AI in Home Automation. *International Journal of Engineering and Technology*, 4(6):803–807, 2012.

[Kum14] Shiu Kumar. Ubiquitous Smart Home System Using Android Application. *International Journal of Computer Networks & Communications*, 6(1):33–43, 2014.

[MDS13] Pooja Malhotra, Parneet Dhillon, and H Sadawarti. ZigBee Technology (802.15. 4): A Survey. *International Journal of Engineering Research & Technology*, 2(11):1939–1944, 2013.

[OAeR+14] Thoraya Obaid, Ali Abou elnour, Muhammad Rehan, Mussab Muhammad Saleh, and Mohammed Tarique. Application in Wireless Home Automation Systems. *International Journal of Computer Networks & Comunications*, 6(4):115–131, 2014.

[Pat09] Gustavo José Henriques Patrício. *Redes Sem Fios de Microcontroladores com Acesso Remoto Aplicada à Domótica*. Dissertação, Universidade Nova de Lisboa, 2009.

[Per14] Bruno Pereira. *A eficiência energética em edifícios - Análise Comparativa da Regulamentação Aplicável na Península Ibérica*. Dissertação, Instituto Politécnico de Viana do Castelo, 2014.

[SE08] Gavin Sullivan and Campbell Edmondson. Heat and temperature. *Continuing Education in Anaesthesia, Critical Care and Pain*, 8(3):104–107, 2008.

[sH16a] Data Logger shield. Https://www.adafruit.com/product/1141. Consultado em 18-05-2016, 2016.

[sH16b] WiFi shield. Https://www.arduino.cc/en/Main/ArduinoWiFiShield. Consultado em 18-05-2016, 2016.

[sH16c] Weather shield. Https://www.sparkfun.com/products/12081. Consultado em 18-05-2016, 2016.

[spe13] Measurement specialties. Datasheet sensor HTU21D. (October):6–16, 2013.

[Tec07] Smartlabs Technology. Insteon - Developers Guide. *SmartLabs Technology*, page 38, 2007.

[WW15] Antony Wilkes and David Williams. Measurement of humidity. *Anaesthesia and Intensive Care Medicine - Elsevier Ltd*, 16(3):128–131, 2015.

[YB13] Xu Yan and Wang Bo. A Security Extension to LonWorks/LonTalk Protocol. *International Journal of Digital Content Technology and its Applications*, 7(6), 2013.