Faculdade de Engenharia da Universidade do Porto

Desenvolvimento de uma Solução de Aquisição de Dados para Laboratório

Diogo Nogueira de Moura

Dissertação em Automação

Mestrado Integrado em Engenharia Mecânica

Orientador: Prof. Joaquim Gabriel Magalhães Mendes

novembro de 2014

Diogo Nogueira de Moura

diogo.moura@fe.up.pt

http://diogonmoura.com

Secção de Automação, Instrumentação e Controlo (**SAIC**)

Departmento de Engenharia Mecânica (**DEMec**)

Faculdade de Engenharia da Universidade do Porto (**FEUP**)

Porto

Portugal

Resumo

Os sistemas *SCADA* são comummente utilizados na industria na monitorização e controlo de processos, com grandes vantagens demonstradas. Em contrapartida, esta tecnologia não é ainda muito utilizada no controlo de laboratórios, não se estando assim a aproveitar as melhorias que estes sistemas podem oferecer para tornar estes locais mais seguros e "inteligentes".

Este projeto foi realizado no âmbito de uma colaboração entre a Laborial, uma empresa portuguesa que fabrica mobiliário de laboratório e a Faculdade de Engenharia da Universidade do Porto. O objetivo que esta colaboração quer alcançar é redefinir os *standards* no mobiliário de laboratório, pela incorporação de tecnologia em bancadas. Assim, o desafio desta dissertação é fazer um sistema de aquisição de dados remoto de baixo custo baseado num microcontrolador, que interage diretamente com o ambiente que o rodeia.

Neste projeto foi desenvolvido um sistema que permite adquirir dados de vários tipos de sensores, tanto analógicos como digitais, maioritariamente termómetros (sensores infravermelhos, circuitos integrados, termopar, Pt100), sensor de humidade relativa, sensor de iluminação ambiente e *RFID*. Para conectar os sensores digitais são usados os protocolos I^2C e 1-Wire. O processamento da informação é feito em placas de desenvolvimento baseadas na plataforma Arduino. Para minimizar o consumo elétrico, foi implementado um *sleep code* que desativa algumas funcionalidades da placa.

A arquitetura deste sistema incorpora uma rede de aquisição de dados cujos nós são constituídos pelos módulos Arduino referidos que comunicam com o sistema *SCADA* (InduSoft Web Studio) por Zigbee (baseado na norma IEEE 802.15.4) ou comunicação série, tanto USB como RS-232, através de protocolo Modbus.

O protótipo desenvolvido foi interligado no advanLab® (*SCADA* desenvolvido pela Laborial, em InduSoft Web Studio v7.0), encontrando-se totalmente funcional.

Abstract

SCADA systems are commonly used in industry to monitor and control processes, with great proven advantages. On the other hand, this technology is not very used to control laboratories, not taking advantage of the improvements that these systems can offer to make these sites safer and more "intelligent"

This project was accomplished in the scope of a collaboration between Laborial, a Portuguese company that manufactures laboratory furniture, and Faculdade de Engenharia da Universidade do Porto. The objective which this collaboration aims to achieve is to redefine the standard of laboratory furniture, by incorporating technology in stands. Therefore, the challenge is to make a low cost remote data acquisition system based on a microcontroller, which can interact directly with the environment.

In this project was developed a system that can acquire data from several types of sensors, both analog and digital, mainly thermometers (IR sensor, integrated circuits, thermocouple, Pt100), relative humidity sensor, environment light sensors and RFID. To connect the digital sensors the protocols used were I^2C and 1-Wire. Information processing is done on development boards, based on the Arduino platform. To minimize the power consumption, was implemented a sleeping mode that deactivates some features of the board.

The architecture of the system encloses a data acquisition network whose nodes consist in Arduino modules that communicate with the *SCADA* system (Indusoft Web Studio) over Zigbee (based on the standard IEEE 802.15.4) or over serial communication, both USB and RS-232, through Modbus.

The prototype was integrated in advanLab®, a commercial solution offered by Laborial, developed in InduSoft Web Studio v7.0, being totally functional.

Agradecimentos

Antes de tudo, gostaria de expressar a minha gratidão aos meus mentores e exemplos como pessoa, que estiveram sempre presentes em todos os momentos, mesmo nos menos bons. Mãe e Pai, um enorme obrigado pela oportunidade que me ofereceram.

Gostaria ainda de agradecer a todo o suporte e orientação dados pelo Professor Doutor Joaquim Gabriel como supervisor e mentor deste projeto, pela oportunidade de manter o foco da minha dissertação num assunto que tanto me interessa.

A todos os colaboradores do "L003", pela ajuda e conselhos dados durante todo o semestre, principalmente à Ana, ao André, António, João, Marco, Prof. Ricardo e Rui.

A todas as pessoas que trabalharam comigo na Laborial, Luciano, Carlos e Emanuel, muito obrigado pelas ideias, pela ajuda na ligação de todos os elementos ao advanLab e principalmente pelo "empurrãozinho"a ultrapassar os obstáculos que se interpuseram no caminho. Sem esta precisa ajuda, possivelmente não teria conseguido chegar tão longe no projeto.

A todos os meus colegas e amigos, pelos 5 anos de companheirismo, partilha e ajuda, a minha sincera gratidão!

Por fim, mas não menos importante, um obrigado enorme à minha namorada, à minha irmã e a toda a minha família por todo o apoio, suporte e paciência durante os dias menos animados!

Este trabalho foi apoiado pelo projeto de I&D INTELLAB II, "Inteligência em Laboratórios", FCOMP-01-0202-FEDER-033877 - financiado pelo Fundo da Comissão Europeia (FEDER) através da COMPETE - Programa Operacional Factores de Competitividade (POFC).

"Education is an admirable thing,
but it is well to remember from time to time
that nothing that is worth knowing can be taught."

Oscar Wilde

em memória do meu *Avô Quim*

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas

BBB	BeagleBone Black
E/S	Entradas e Saídas
HDMI	*High-Definition Multimedia Interface*
HMI	*Human-Machine Interface* - Interface Homem-Máquina
IDE	*Integrated Development Environment*
IR	*InfraRed* - Infravermelho
IWS	InduSoft Web Studio
I/O	*Input/Output*
I^2C	*Inter-Integrated Communication*
Kbps	*Kilobits per second* (1024 bits *per second*)
NTC	*Negative Temperature Coefficient*
PWM	*Pulse-Width Modulation*
RAM	*Random-Access Memory*
RFID	*Radio-Frequency IDentification*
SCADA	*Supervisory, Control And Data Acquisition* - Supervisão, Controlo e Aquisição de Dados
SPI	*Serial Peripheral Interface*
UART	*Universal Asynchronous Receiver / Transmitter*
USB	*Universal Serial Bus*

Capítulo 1

Introdução

Na indústria os sistemas *SCADA* são uma ferramenta extremamente útil na monitorização e gestão de processos de produção, porém ainda não são muito utilizados em laboratórios. Neste âmbito, a Laborial, uma empresa de fabrico de mobiliário de laboratório, desafiou a Faculdade de Engenharia da Universidade do Porto no sentido de desenvolver soluções de *hardware* de baixo custo. Este *hardware* deverá completar o leque de soluções oferecidas pela empresa e integrar o *software SCADA* já desenvolvido - advanLab®. Este sistema *SCADA* visa automatizar as tarefas mais comuns desempenhadas em laboratórios, tornando-os mais "inteligentes" e seguros.

Ao longo de todo o projeto foi tido em consideração o custo dos componentes utilizados, pois esta é uma área onde os componentes associados devem ter um preço acessível de forma a tornar estas soluções desejáveis e competitivas.

Serão de seguida apresentadas a Motivação desta dissertação e a Laborial, empresa onde o projeto está a ser desenvolvido. Os Objetivos deste projeto, os Resultados alcançados e a sua Estrutura também estão presentes neste local.

1.1 Motivação

Para dar resposta à falta de soluções *open source* para a automatização de laboratórios, a Laborial, em parceria com a Faculdade de Engenharia da Universidade do Porto, está a desenvolver o Projeto INTELLAB II - Inteligência em Laboratórios. Este projeto surge como continuação de um projeto anterior, denominado INTELLAB, onde foi criada uma solução para monitorização de laboratórios, tendo por base o desenvolvimento de um sistema *SCADA* - o advanLab® - baseado no *software* InduSoft Web Studio 7.0. Estas inovações levaram a Laborial a atingir uma posição de reconhecimento entre os produtores de laboratórios, resultando em dois 1°s prémios em feiras internacionais da especialidade.

Para melhorar a solução prévia, constatou-se a necessidade de implementar um sistema que adquirira e guarde a informação de forma automática e independente, um *data logger*. Esta funcionalidade é bastante requisitada por muitos clientes, pois permite manter um *backup* dos dados adquiridos no caso de não ser possível efetuar a comunicação com o servidor. Para além disso, em algumas aplicações menos exigentes, a resolução dos conversores analógico-digitais (ADC) existentes em microcontroladores é suficiente para responder a essas necessidades, permitindo assim diminuir os custos.

1.1.1 Laborial

A Laborial tem como atividade principal a produção, comercialização e instalação de laboratórios e mobiliário técnico hospitalar [1].

Ao longo do tempo, a empresa adquiriu conhecimento e ganhou cada vez mais interesse na criação de soluções para fins específicos, como laboratórios para a área da saúde e educação, a nível nacional e internacional.

Situada na Zona Industrial da Maia, esta empresa encontra-se a redefinir os limites daqueles que são considerados como laboratórios de alta tecnologia. Este grande

salto deu-se quando foi introduzida a *Blautouch*[1] e o advanLab®.

O INTELLAB II pretende explorar novas características de monitorização e controlo que até hoje ainda não foram desenvolvidas, aproveitando as capacidades da "Computação Ubíqua"[2] e da "Internet das coisas". Estas características farão parte de uma nova geração de laboratórios que estão a ser desenvolvidos em áreas como anatomia patológica, segurança biológica ou veterinária [3].

1.2 Objetivos

O objetivo principal definido para este trabalho é o desenvolvimento de um *data logger* de baixo custo para integrar o advanLab®. A solução desenvolvida deve estar focada principalmente na aquisição de temperatura e humidade, uma vez que são as grandezas com mais necessidade de monitorização nos projetos que a empresa desenvolve.

No desenvolvimento do *data logger* está implícito que a solução proposta deverá cumprir requisitos básicos, como por exemplo, estar preparada para lidar com sensores de diferentes fabricantes, tanto analógicos como digitais (I^2C e 1-Wire). A comunicação entre o sistema a desenvolver e o sistema *SCADA* deverá ser estabelecida recorrendo ao protocolo Modbus, atualmente utilizado na empresa.

Numa fase final, o sistema desenvolvido deverá ser otimizado de modo a ter um consumo energético reduzido e estar conectado ao advanLab® recorrendo a tecnologia sem fios, sendo que a solução cablada deverá ser uma redundância do sistema, se assim for justificável.

[1] *Blautouch* - Bancada de Laboratório com um computador integrado, passível de ser utilizada em salas brancas.

[2] Ubíqua – Presente em todo o lado [2].

1.3 Resultados Obtidos

Neste projeto utilizou-se a plataforma de desenvolvimento Arduino, que no máximo consegue adquirir a uma taxa de 10 kHz (para um canal), com um conversor analógico-digital (ADC) de 10 bit. Para além da aquisição de dados, este nó permite gravar os dados num cartão de memória, funcionando como um *data logger*. De modo a assegurar que o instante registado é o da aquisição e não da transmissão, foi introduzido uma placa de *Real-Time Clock* no sistema. Para reduzir o consumo energético da solução foram aproveitadas as potencialidades de *sleep* do microcontrolador, desligando os periféricos quando estes não se encontram em utilização.

A solução final deste projeto inclui vários sensores de temperatura analógicos (sensores integrados, termístor, termopar e Pt-100) e digitais (sensores digitais integrados e sensores infravermelhos via 1-Wire e I^2C). Para comunicação com o servidor, o nó de aquisição de dados utiliza o protocolo Modbus que pode ser usado por cima de comunicação série (USB e RS-232) ou via *wireless*, utilizando o protocolo Zigbee (baseado na norma IEEE 802.15.4).

O protótipo foi implementado na empresa, numa bancada de operação veterinária, cumprindo todos os requisitos e objetivos propostos.

1.4 Estrutura

Este documento está dividido em 5 capítulos que irão descrever o trabalho que foi executado ao longo deste projeto.

No Capítulo 2, Estado da Arte, são apresentadas e discutidas algumas soluções de *hardware* e *software* existentes no mercado.

No Capítulo 3, é apresentada a arquitetura implementada para o desenvolvimento do projeto, desde a solução proposta até à solução final, sendo justificadas as opções relevantes que foram tomadas.

No Capítulo 4 são discutidos os resultados obtidos, bem como alguns testes que foram efetuados durante a dissertação.

As conclusões deste trabalho são apresentadas no Capítulo 5, bem como uma proposta de Trabalhos Futuros.

Extratos do código implementado para programar a solução serão apresentados nos Anexos, assim como alguns esquemas relevantes de montagem dos sensores.

Capítulo 2

Revisão do Estado da Arte

No seio das tecnologias para laboratórios muito já foi desenvolvido, porém ainda há espaço para novas criações. Com este objetivo em mente, o desenvolvimento de sistemas de baixo custo é uma forma de disponibilizar o estado da arte da tecnologia a um maior número de pessoas.

Por consequência, o desafio é escolher e desenvolver tanto *hardware* como *software* que, sem gastos desnecessários, acrescente valor a um laboratório. A fim de cumprir este objetivo, é considerada uma solução baseada num microcontrolador e em *software* de supervisão.

No decorrer deste Capítulo, serão introduzidos os aspetos gerais sobre a questão em estudo, bem como a pesquisa que foi feita de forma a tornar este trabalho possível.

2.1 Introdução

Atualmente os laboratórios têm necessidade de monitorizar e controlar diversas variáveis, tais como a temperatura, humidade relativa, pressão atmosférica, acessos, entre outras. Para isso existem várias soluções no mercado, umas mais rudimentares, outras mais complexas e avançadas. As variáveis que são controladas com maior frequência são a temperatura (tanto do ambiente como de

um local específico) e a humidade relativa. Para além disso, existe também a necessidade de controlar acessos, tanto a locais como ao *software*.

A Figura 2.1 mostra uma imagem do advanLab®, referido anteriormente.

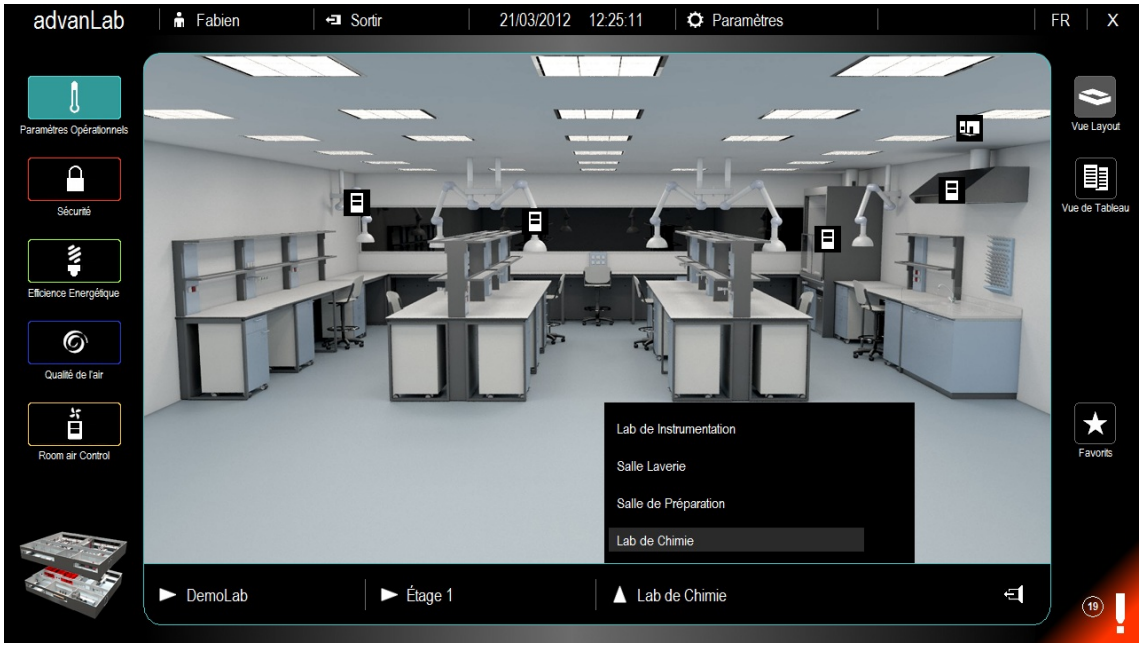


Figura 2.1: Exemplo do advanLab®

2.2 *Hardware*

Nesta Secção será introduzido o *hardware* e apresentadas possíveis soluções para responder à necessidade exposta anteriormente. Por conseguinte, foram estabelecidas algumas especificações chave a fim de determinar qual seria a solução mais apropriada para o projeto.

A especificação mais importante do projeto é basear-se em *hardware* de baixo custo que irá complementar aquele que já se encontra em funcionamento. Este projeto visa responder à necessidade de gravação dos dados adquiridos numa plataforma externa (*data logger*) e a possibilidade de se ligar ao sistema principal por cabo e por tecnologia *wireless*.

Inicialmente foram analisadas soluções já existentes no mercado para monitorização de laboratórios. De seguida estas foram contrapostas com versões baseadas em *hardware* de desenvolvimento de projetos genéricos (Subsecção 2.2.4).

2.2.1 Sistemas de Aquisição de Dados

Um sistema de aquisição de dados é composto geralmente por sensores, um módulo de aquisição de sinal e um computador. O dispositivo de aquisição de sinal faz o condicionamento de sinal e integra o conversor analógico-digital (ADC). No computador está instalado o *software* necessário à comunicação deste com o dispositivo de aquisição de sinal e a aplicação que irá apresentar os dados recolhidos, Figura 2.2 [4].

2.2.2 *Data logger*

Um *data logger*[1] é um dispositivo eletrónico que guarda informação ao longo do tempo, com uma taxa de amostragem pré-definida. A vantagem destes sistemas

[1] Também pode ser escrito sob a forma de *datalogger*.

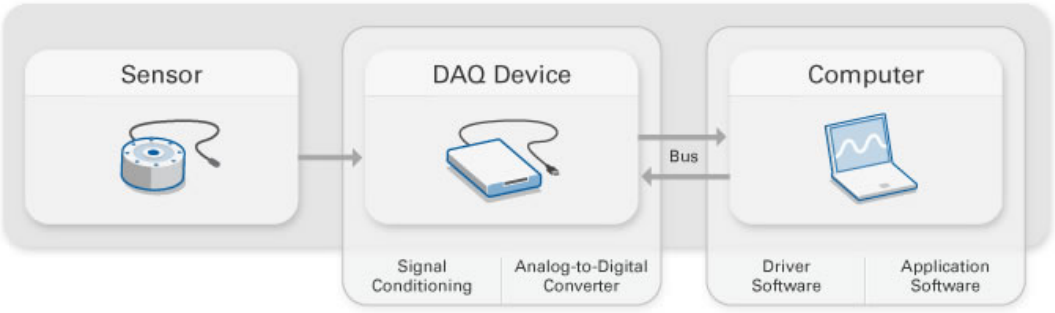


Figura 2.2: Esquema de um sistema de aquisição de dados

face a outros dispositivos de aquisição de dados, é o facto de operarem independentemente de um computador.

Existem várias configurações e formas disponíveis. Desde dispositivos simples de um único canal com funções bem definidas, até soluções complexas e completas, capazes de lidar com centenas de entradas mesmo tempo [5, 6].

Nas Figuras 2.3 [6], 2.4 [7] e 2.5 [8] são apresentados alguns exemplos presentes no mercado.

Figura 2.3: Um *data logger* com o formato de um ovo

Figura 2.4: Um *data logger* da Fourtec

Figura 2.5: Um sistema de *data logger* completo com display da Omega

2.2.3 *Hardware* específico de laboratório

As soluções específicas de laboratório podem trazer consigo algumas vantagens à sua implementação, uma vez que todo o *hardware* está preparado para as

necessidades que lhe serão impostas. Em geral, os produtores de *hardware* desenvolvem também o *software* para comunicação com os seus produtos.

Como as necessidades mais comuns dos projetos que a Laborial desenvolve são a aquisição e gravação de dados de temperatura e humidade, as soluções apresentadas terão como foco estas características. Assim, serão expostas 2 possíveis implementações de fabricantes distintos. Outras soluções seriam possíveis mas afastar-se-iam dos objetivos da empresa, pois recorrem a *software* proprietário. Na Tabela 2.3 estão sumarizados alguns fabricantes deste tipo de soluções encontrados na literatura.

Todas as soluções analisadas consistem numa estação base (*gateway*), dispositivo que estabelece a ligação entre os módulos de aquisição e o computador onde estará instalado o sistema de supervisão.

2.2.3.1 WiSensys

Uma possível resposta à necessidade da automação laboratorial é a solução desenvolvida pela WiSensys. A estação base analisada, Figura 2.6a [9], comunica com os módulos de aquisição de dados por tecnologia sem fios Zigbee. Esta estação base tem a capacidade de ligar até 100 sensores na sua rede.

Os módulos fazem a aquisição de dados, como por exemplo, o modelo WS-DLTc que mede temperatura e humidade relativa do ar. Para além dos sensores, existe a hipótese de acoplar um display externo para facilitar a leitura das variáveis. A Figura 2.6b [10] representa o referido módulo sendo que o aspeto exterior dos demais é semelhante [9–11].

A título de exemplo, o módulo WS-DLTc tem as seguintes características:

- Guarda até 10000 dados;
- Gama de medição: 10 a 90 % (Humidade Relativa),
  -20 a 80 °C (Temperatura);

- Resolução: 0,1 % (Humidade Relativa),
  0,1 °C (Temperatura);
- Intervalo de medição: de 1 a 20 segundos;
- Amostras por transmissão: 1 a 5 amostras;
- Duração da bateria: 3 anos (com 1 amostra enviada por transmissão e intervalos de medição de 20 segundos);
- Comunicação por Zigbee (alcance de 1000 m em linha de vista);

(a) Estação base WS-BU

(b) Sensor de temperatura e humidade WS-DLTc

Figura 2.6: Solução laboratorial da WiSensys

2.2.3.2 VAISALA

O *hardware* desenvolvido pela VAISALA é semelhante ao da WiSensys: existe uma estação base que comunica com o servidor e com os módulos de aquisição.

As estações base têm a capacidade de ligar até 16 módulos e podem guardar até 250 registos por cada sensor. A VAISALA tem disponíveis vários tipos de sensores, de destacar os de temperatura (Termopar ou termístor) e os de humidade relativa.

Relativamente aos sensores de temperatura, por exemplo, o modelo HMT140, Figura 2.7 [12] tem as seguintes características:

- Gama de medição: 0 a 100 % (Humidade Relativa),
  -40 a 80 °C (Temperatura);
- Resolução: ±0,4 °C (Temperatura);
- Comunicação por Wi-Fi;

Este fabricante oferece ainda a possibilidade de ligar os seus equipamentos através de *ethernet* ou via *Wi-Fi*, tendo por base a norma IEEE 802.11 [12].

Figura 2.7: Módulo HTM140 da VAISALA

2.2.4 *Hardware* genérico

A característica principal definida pela empresa é tornar o *hardware* o mais versátil possível para poder responder às necessidades atuais da empresa. Por exemplo, situações em que é necessário adquirir sinais apenas de um ou dois sensores, ou ter uma única saída.

Na Tabela 2.2 [13] estão listadas algumas características das soluções mais exploradas.

2.2.4.1 Advantech

Os módulos da Advantech, Figura 2.8 [14], são aqueles que são considerados como um dos *standards* industriais pelas suas características. São dispositivos de aquisição de dados robustos e certificados.

A solução deste fabricante que foi estudada consiste numa estação base que interliga o servidor e os módulos de aquisição de dados. Neste caso foi analisada uma solução sem fios, da série ADAM-2000.

O fabricante dispõe de um módulo específico para aquisição de temperatura e humidade e módulos genéricos, tanto analógicos como digitais. Estes módulos suportam RS-232 e RS-485, no entanto não têm disponíveis comunicações como I^2C e 1-Wire. Os custos desta solução encontram-se explícitos na Tabela 2.1 [14].

Uma das desvantagens desta solução é a necessidade de comprar um módulo para cada função e de estes não serem programáveis [14].

Tabela 2.1: Custo da solução da Advantech

Modelo	Função	Custo
ADAM-2031Z-AE	Nó de aquisição de temperatura e humidade relativa	196,00 €
ADAM-2520Z-AE	*Wireless Modbus RTU Gateway*	216,00 €
ADAM-2051Z-AE	Nó de aquisição de sinais digitais (8 canais)	162,00 €
ADAM-2017PZ	Nó de aquisição de sinais analógicos (6 canais)	n.d.

Figura 2.8: Módulo de aquisição de dados Advantech

2.2.4.2 BeagleBone Black

O BeagleBone Black, Figura 2.9 [15], é um mini computador que pode ser adquirido a baixo custo, tem bom desempenho, interage com diversos periféricos, no entanto é focado em projetos pessoais. As suas características podem ser consultadas na Tabela 2.2 [13]. Com 7 entradas analógicas e 65 entradas e saídas digitais esta poderia ser uma solução bastante interessante, todavia o fabricante não autoriza projetos com propósitos comerciais [16, 17].

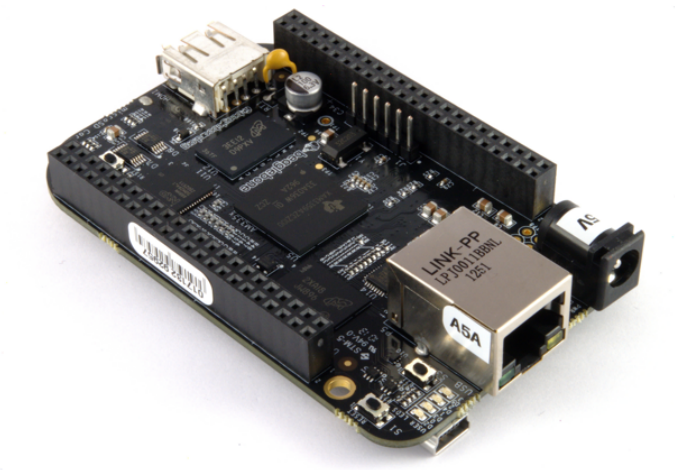


Figura 2.9: Exemplo do BeagleBone Black

2.2.4.3 Arduino UNO

A placa de desenvolvimento Arduino UNO, Figura 2.10 [18], tem por base um microcontrolador ATmega 328, que tem 32KB de memória interna, funciona com uma velocidade de relógio de 16 MHz, tem um ADC de 10 bits e é capaz de adquirir amostras, no máximo, a uma frequência de 10 kHz (apenas num canal).

Devido ao seu baixo custo e facilidade de uso, existe bastante informação disponível e tem uma comunidade de *developers* considerável.

Com esta solução é possível realizar soluções com fins comercias, no entanto o fabricante pede que se informe o cliente que a sua solução é baseada na plataforma Arduino.

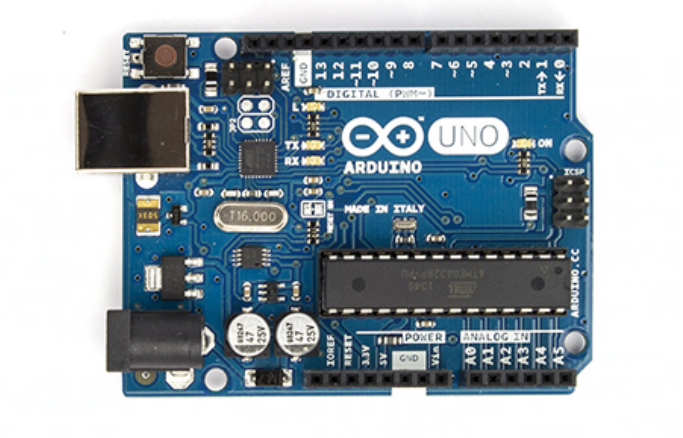


Figura 2.10: Fotografia do Arduino Uno

Tabela 2.2: Comparação de *Hardware*

Controlador	Série ADAM-2000	BBB	Arduino Uno
Processador		TI Sitara AM3359 ARM Cortex A8	ATmega 328
Velocidade do Relógio		1GHz	16MHz
RAM		512 MB	2 KB
Memória		2 GB μSD	32 KB
USB	0	1	1
Saída de Audio **Saída de Vídeo**	N.D.	HDMI	N.D.
Ethernet	N.D.	10 / 100 MBps	N.A.
I/O	6 Entradas Analógicas (Tensão e Corrente) 8 Entradas Digitais Temperatura e Humidade Relativa 8 Entradas Digitais (com amplificador de potência)	3 I^2C 2 CAN 2 SPI 5 Portas Serie 65 GPIO 8 PWM 7 Entradas Analógicas (ADC com 12bit)	14 GPIO (6 PWM ou I^2C ou SPI) 6 Entradas Analógicas (ADC com 10bit)
Tamanho		8,63 x 5,33 cm	7,49 x 5,33 cm
OS	N.D.	Linux Android MS Windows	N.D.
Ferramentas **Desenvolvimento**		Python Scratch Linux Eclipse	Arduino IDE
Preço	Total $\approx$ 900 €	55 €	22 €

2.3 *Software*

No desenvolvimento desta solução foram tidos em consideração dois tipos de *software*; os que foram criados especificamente para laboratórios e os que foram projetados para aplicações industriais. Os desenvolvidos para laboratórios normalmente são criados pelos fabricantes de *hardware* para comunicação cos seus produtos.

O *software* pensado exclusivamente para laboratório será apresentado na Subsecção 2.3.1, enquanto que o *software* para fins industriais será introduzido na Subsecção 2.3.2.

2.3.1 *Software* específico para laboratório

O *software* criado para satisfazer as necessidades de um laboratório apresenta vantagens no que toca à manipulação da informação obtida pelas variáveis para o qual está programado. No entanto, uma grande desvantagem deste tipo de programas é a falta de flexibilidade que eles apresentam. Se for necessário adquirir uma variável para o qual o programa não está pré-programado, por norma estas soluções não o permitem. Para além disso, usualmente quem desenvolve estas aplicações são os fabricantes de *hardware*.

Esta última característica inviabiliza um dos pontos fulcrais do projeto, que é a capacidade de interagir com uma vasta gama de produtos, de diversos fabricantes. Ainda assim, são listadas na Tabela 2.3 [11] algumas das soluções encontradas no mercado.

Tabela 2.3: Tabela comparativa entre *software* específico para laboratório

Empresa	Variáveis	Comunicações disponíveis	Características Gerais
Rmoni	Temperatura Humidade Relativa CO_2 Outras, quando encomendado	IEEE 802.15.4 / Zigbee	Respeita a norma FDA 21 part 11, Sistema local ou Web
accsense	Temperatura Humidade Relativa CO_2 Porta Aberta Vibração Som Luminosidade	IEEE 802.15.4	Totalmente baseado na Web; Alarme e sistema de notificação Exportação de dados
VAISALA	Temperatura Humidade Relativa CO_2 Porta Aberta Pressão pH Outros	RS-232 USB Ethernet Wifi	Cumpre FDA 21 part 11; Sistema baseado na Web
CheckPoint	Temperatura Porta Aberta CO_2	N.A.	Sistema baseado na Web ; Usa base de dados SQL Capacidade de alarmes
testo	Temperatura Humidade Relativa Entradas Analógicas(4 - 20 mA)	Ethernet Wireless 868 MHz Wireless 2.4 GHz	Sistema baseado localmente Assistente de Configuração Diferentes níveis de alarme
WiSensys	Temperatura Humidade Relativa CO_2 Consumo Energético Entradas Analógicas(0 - 25 mA, 0 - 30 V, 0 - 4 V)	RS-232 RS-485 GPRS Ethernet	Sistema baseado no local e na Web Capacidade de Login e de Alarmes Exportação de dados;
labguard	Temperatura Humidade Relativa CO_2 Pressão Entradas Analógicas (0 - 5V, 0 - 10V and 4 - 20mA) Outros	Ethernet RS-232 RS-485 Wireless 868 MHz GPRS Wavenis	Interface gráfica intuitiva Deteção automática de *Hardware*; Capacidade de conexão à Web Diferentes níveis de acessos
HANNA instruments	Temperatura	N.A.	N.A.
Aginova	Temperatura Humidade Relativa Energia Luminosidade Corrosão	Low power IEEE 802.11	Sistema Baseado na Web Alarmes e exportação de dados Diferentes níveis de acesso Respeita FDA21 part 11 Suporte de base de dados SQL
Rees Scientific	Temperatura Humidade Relativa CO_2 Pressão Atmosférica Luminosidade O_2 Tensão e corrente Velocidade do Ar Deteção de fugas	Zigbee	N.A.
IsatecLtd	Temperatura caudal CO_2 pH Pressão Condutividade Consumo energético	IEEE 802.11 b/g	Sistema localmente baseado, com ligação à internet Aquisição de dados automática Capacidade de alarmes exportação de dados
E-Control Systems	Temperatura Humidade Relativa Entradas analógicas (0 - 5 V, 4 - 20 mA) Fuga de liquido refrigerante Pressão Compressor Runtime	IEEE 802.15.4 Zigbee	Sistema baseado na Web exportação de dados, diferentes níveis de acessos
Comark Instruments	Temperatura Humidade Relativa Pressão Caudal CO_2 Porta Aberta Outros	IEEE 802.15.4	Sistema Baseado na Web
SENSONET	Temperatura Humidade Relativa Luminosidade Raios Ultravioleta Pressão Atmosférica	IEEE 802.15.4	Sistema baseado localmente, com ligação a internet Exportação de dados Capacidade de Alarmes má qualidade gráfica

2.3.2 *SCADA*

O *SCADA* (Supervisão, Controlo e Aquisição de Dados[2]) é ,como o próprio nome sugere, um sistema de monitorização e controlo industrial. É uma tecnologia que permite ao utilizador adquirir dados de processos produtivo que se encontram em diferentes locais e interagir com estes em tempo real, quer local, quer remotamente.

A implementação de sistemas *SCADA* é vastamente utilizada quando há a necessidade de controlar e monitorizar processos que requerem interação em tempo real e armazenamento de dados.

Na Tabela 2.4 [19] são apresentados alguns exemplos de *software SCADA* e os seus fabricantes. De todas as soluções enumeradas, a Laborial selecionou o *software* o InduSoft Web Studio.

Tabela 2.4: Algumas soluções de *software SCADA* disponíveis no mercado

Software	**Fabricante**
ArcInfo	PcVue
Citect	Schneider Electric
ControlMaestro	Elutions
DAQFactory	AzeoTech
DSC (Datalogging and & Supervisory Control)	National Instruments
Ez Data Logger	ICP DAS
Fast/Tools	Yokogawa
General Electric	iFIX
Geo Scada	Iconics
InduSoft Web Studio	Schneider Electric
iX Software	Beijer Electronics
Lookout	National Instruments
RsView	Rockwell Automation
Satec	eXperManager
Simatic	Siemens
Winlog Pro	Sielco Sistemi

[2]Supervisory, Control And Data Acquisition

2.3.2.1 InduSoft Web Studio

O InduSoft Web Studio (IWS) cumpre todos os requisitos que foram pré-estabelecidos, disponibilizando as seguintes características:

Funcionalidades

Permite a criação de aplicações onde é possível identificar o utilizador recorrendo ao sistema *Login / Password*, permite ainda a visualização gráfica da evolução de variáveis, registo temporal de dados, visualização *online* de alarmes e a criação automática de relatórios. Esta aplicação é programável utilizando a linguagem *Visual Script*, o que permite aumentar as funcionalidades que vêm pré-programadas.

Interface Gráfica

Como o aspeto gráfico da aplicação final é bastante importante numa solução comercial, bem como o facto desta ser *user friendly*, esta característica teve um peso importante na escolha do *software* para desenvolver a aplicação final.

O *IWS* permite flexibilidade no ambiente gráfico, já que é dotado de funções de manipulação gráfica. nomeadamente alterar cores ou formas de acordo com o estado das variáveis.

Protocolos de Comunicação

Um *software SCADA*, por definição, suporta uma grande variedade de protocolos de comunicação, bem como *drivers* distintos que lhe permite interagir com o *hardware* disponível no mercado. Esta característica valoriza os *SCADA* perante os *softwares* específicos.

Este *software* tem ao seu dispor um vasto leque de *drivers* que suporta a maioria do *hardware* utilizado na industria. Para além disso é possível desenvolver um *driver* específico para um determinado componente. Oferece também a possibilidade de configurar o acesso à aplicação via *Web*, assim como via dispositivo móvel.

Licenças

Para os software *SCADA* existem dois tipos de licenças: uma para desenvolver o aplicativo, denominada de licença de desenvolvimento e outra para utilizar o *software*, que é adquirida pelo cliente final, a licença de *Run-Time*.

O InduSoft oferece uma licença de desenvolvimento com 1500 tags[3] e uma utilização de três *drivers* diferentes. Disponibiliza também uma licença de *Run-Time* que oferece suporte para as funções de *Web Thin Client*, *Secure Viewer Thin Client* e *SMA Thin Client*[4].

A Figura 2.11 exibe o ambiente de programação do InduSoft do lado esquerdo, e a aplicação em funcionamento do lado direito.

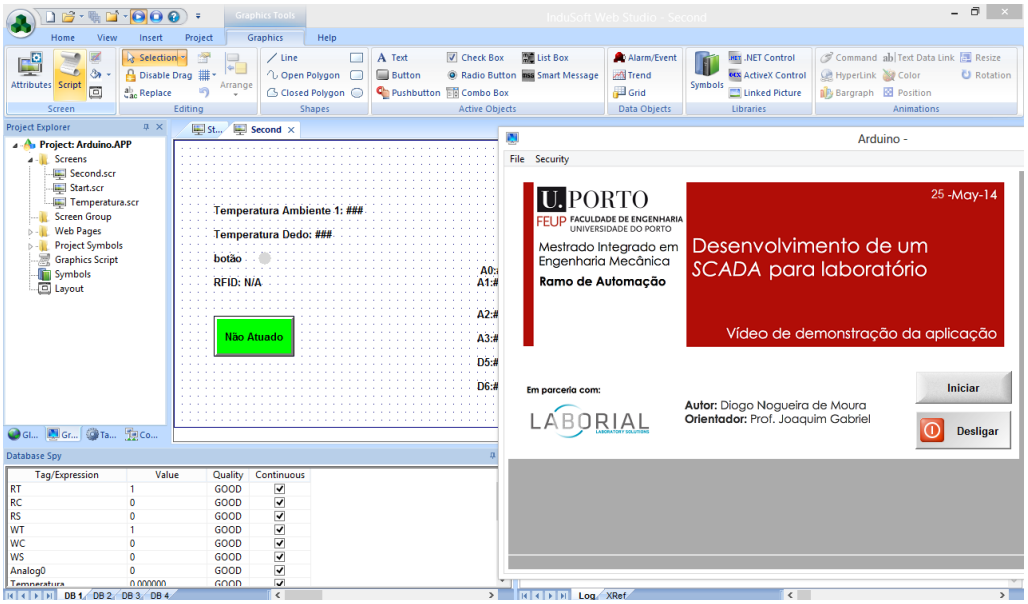


Figura 2.11: Ambiente de desenvolvimento do InduSoft Web Studio e a aplicação em funcionamento

[3]Tags – Variáveis de entradas, saídas ou internas
[4]Funções para acesso à aplicação via Web e dispositivos móveis

2.4 Conclusões

Após a análise do *hardware* apresentado, conclui-se que o mais adequado ao desenvolvimento do presente projeto é utilizar um dispositivo que seja versátil e que possa ler diferentes sensores de variados fabricantes. Deste modo, a solução será baseada na plataforma Arduino, uma vez que o seu custo de aquisição é relativamente baixo, é um *hardware* fiável e flexível. Para além das razões já referidas, esta ferramenta pode adquirir sinais analógicos.

Relativamente ao *software*, o *SCADA* é mais aconselhável para este projeto uma vez que é mais flexível, personalizável e fiável do que um aplicativo específico de laboratório. Assim sendo, foi selecionado o *software* InduSoft Web Studio pela sua facilidade e flexibilidade de programação, e pelas suas características gráficas. Outra vantagem deste *software* é o apoio que presta ao cliente, disponibilizando ajuda online em tempo real, em Inglês e em Português.

Capítulo 3

Arquitetura

Ao longo deste Capítulo será apresentada a arquitetura desenvolvida neste trabalho para criar uma solução baseada num microcontrolador; que disponha de entradas digitais e analógicas, com um ADC, no mínimo, de 10 bits; que comunique com o sistema *SCADA* recorrendo tanto a soluções cabladas como a tecnologia sem-fios e que disponha da funcionalidade de *data logger* através da um cartão de memória μSD, de acordo com o esquema apresentado na Figura 3.1.

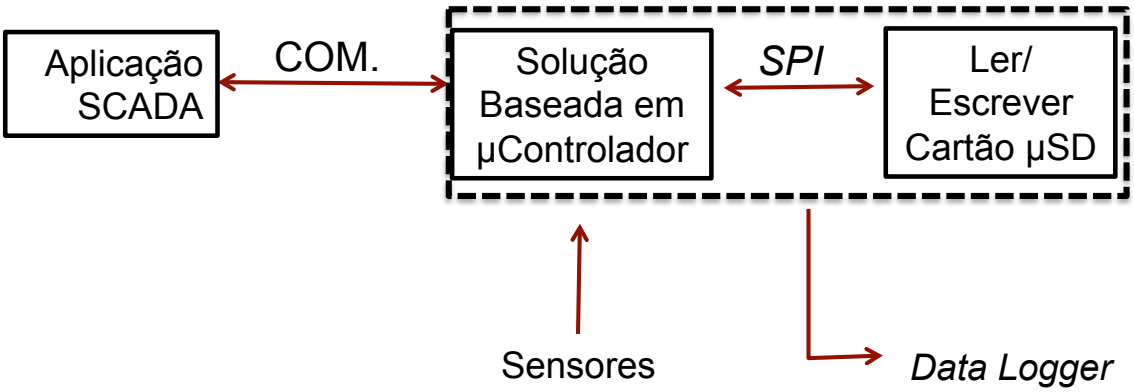


Figura 3.1: Esquema ilustrativo da arquitetura proposta

3.1 Arduino

O dispositivo Arduino é uma placa de desenvolvimento com um microcontrolador que integra toda a eletrónica necessária ao seu funcionamento,

sendo apenas necessário transferir o programa desejado. Esta característica significa que não é necessário ter conhecimentos avançados de eletrónica para se desenvolver um protótipo com esta ferramenta.

A facilidade de adicionar placas de expansão, conhecidas como *shields* são uma vantagem face a outras opções. Sendo esta plataforma *open source*, o *software* e os esquemas elétricos estão disponíveis, permitindo a sua personalização.

Como já for referido anteriormente, é possível criar projetos para fins comerciais baseados na tecnologia Arduino.

Neste projeto foi utilizado o Arduino UNO com um *shield Wireless SD* que incorpora um leitor de cartões μSD. A razão da utilização deste *hardware* foi a sua disponibilidade imediata de funcionamento, uma vez que é o modelo base de toda a plataforma Arduino. Quanto ao *shield*, a razão da sua escolha é a junção de duas funcionalidades num só componente: a possibilidade de efetuar registo de dados e de os transmitir via *wireless*, pois este periférico é dotado de um *socket* para ligação de uma antena Zigbee.

3.2 Comunicações

Nesta Secção serão introduzidos os protocolos de comunicação utilizados.

A comunicação entre o Arduino e o sistema de supervisão foi feita recorrendo ao protocolo Modbus, pois é o protocolo utilizado pela Laborial nos seus projetos.

Os protocolos utilizados para interface com os sensores digitais e o Arduino foram o I^2C e o 1-Wire. Estes serão explicados mais detalhadamente na Subsecção 3.2.2 e no Anexo C.

3.2.1 Arduino e *SCADA*

3.2.1.1 Modbus

Em 1979 a Modicon, hoje propriedade da Schneider Electric, desenvolveu um protocolo de troca de mensagens chamado Modbus. Este protocolo é utilizado para estabelecer comunicação entre dispositivos eletrónicos usando o modelo de Servidor-Cliente ou *Master-Slave*. Nos dias que correm, é um dos padrões das comunicações industriais, uma vez que é o protocolo mais utilizado neste ambiente. O Modbus atingiu esta posição de reconhecimento pois é totalmente aberto e funciona como uma "língua universal" entre dispositivos de diferentes fabricantes [20].

O Modbus é um protocolo de pedido/resposta, que pode ser implementado em diferentes barramentos ou redes, como por exemplo: RS-232, USB ou Ethernet para soluções cabladas. Para situações onde seja necessário a implementação sem-fios, este funciona por cima de Zigbee, Wi-Fi ou mesmo Bluetooth.

Existem vários tipos de Modbus, no entanto os mais frequentes são o Modbus ASCII, o Modbus RTU e o Modbus/TCP. As mensagens são enviadas todas da mesma forma, sendo que a diferença entre os formatos é a forma como estas estão codificadas. Destes três tipos, o mais utilizado é o Modbus RTU.

Quando um servidor Modbus deseja saber alguma informação sobre qualquer dispositivo, este propaga uma mensagem por toda a rede com o endereço do periférico com que pretende comunicar. Embora todos os aparelhos ligados na rede recebam a mensagem, apenas o selecionado irá responder. Neste tipo de protocolo apenas o Servidor pode iniciar uma comunicação, os clientes apenas se limitam a responder [21].

Uma aplicação comum para o Modbus é a monitorização de dispositivos ou máquinas recorrendo a computadores ou HMIs[1]. Para além destas tarefas, também estabelece a ligação entre os sensores e o servidor [20].

Na Figura 3.2 é mostrada a janela de configuração de Modbus do InduSoft Web Studio para o projeto desenvolvido.

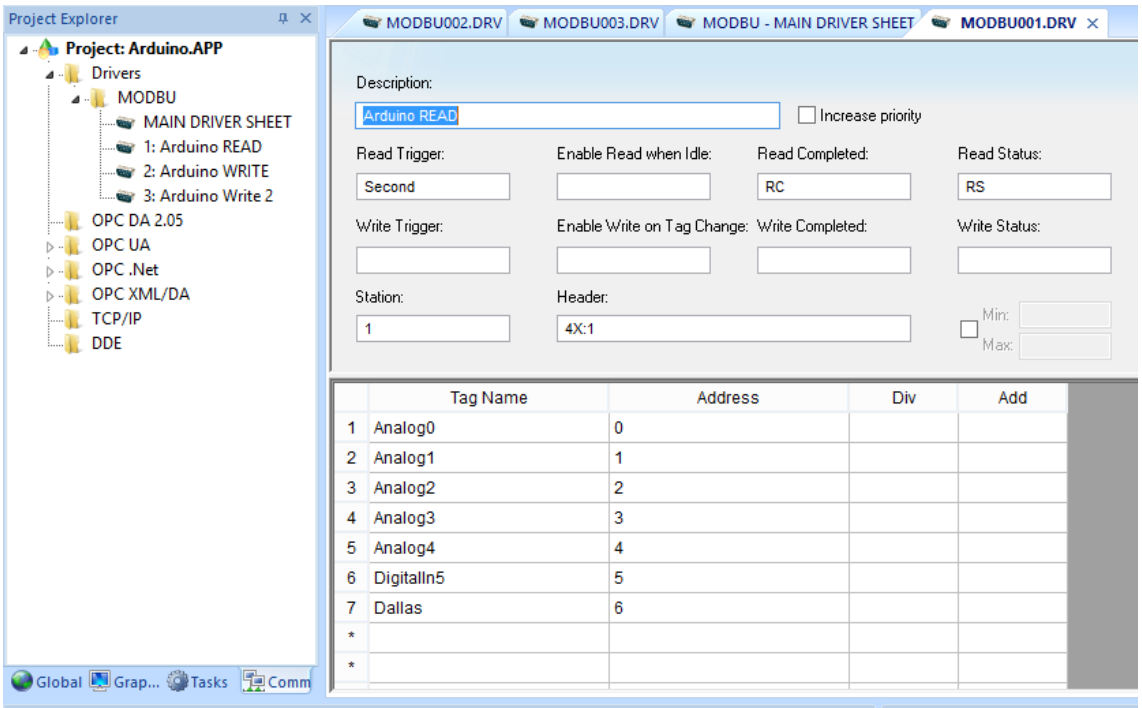


Figura 3.2: Captura de ecrã do IWS, na janela de configuração do Modbus

3.2.1.2 Zigbee

As comunicações sem fios foram estabelecias recorrendo a Zigbee, protocolo baseado na norma IEEE 802.15.4. Esta é uma alternativa viável para

[1] *Human-Machine Interface*

comunicações desta natureza uma vez que é um protocolo sem custos de utilização, que consome menos energia do que Wi-Fi e é excelente para aplicações em que seja necessário a utilização de baterias como fonte de energia.

A conexão é estabelecia muito rapidamente, na ordem de alguns milissegundos, no entanto tem baixo débito de informação. Neste caso isto não é problemático, pois as mensagens enviadas são curtas.

A norma IEEE 802.15 estabelece 3 frequências que podem ser usadas para transferir dados: 868 MHz, 915 MHz e 2.4 GHz. De acordo com a frequência, a velocidade da transferência altera-se: para a primeira, é possível utilizar velocidades na ordem dos 20 Kbps, para segunda de 40 Kbps e para a última de 250 Kbps. Embora o débito de informação seja relativamente reduzido, é normalmente suficiente para estas aplicações.

A norma base define dois tipos de dispositivos, os *Full Function Devices* (FFD) e os *Reduced-Function Devices* (RFD), porém o Zigbee distingue três tipos de dispositivos: o Coordenador (ZC), o Router (ZR) e o dispositivo de fim de linha (ZED). As topologias de rede disponíveis neste tipo de comunicação são três: em estrela, em *cluster* e em *mesh*.

Para cumprir os requisitos da norma, atribui-se aos coordenadores e aos routers a função FFD, enquanto que os dispositivos de fim de linha têm atribuídas as funções de FFD e RFD, consoante o que for necessário para a rede em questão [22].

3.2.2 Arduino e sensores

3.2.2.1 *1-Wire*

Originalmente conhecido como MicroLAN, o protocolo 1-Wire é um sistema de comunicação entre dispositivos eletrónicos. É composto por três componentes principais: o *master*, o *slave* e o meio físico de comunicação entre eles.

Para existir comunicação, o *master* envia uma mensagem a toda a rede com o endereço do cliente que deseja, e à semelhança do Modbus, todos os dispositivos

recebem a mensagem, mas só o destinatário responde. Este protocolo de comunicação funciona em *half-duplex*, uma vez que transmite informação em ambos os sentidos, apenas um de cada vez.

Um *master* controla todos os elementos que estejam disponíveis no barramento a uma taxa de transferência de dados de 14,4 Kbps, 100 nós e um cumprimento máximo de 500 m. Os *slaves* não estão autorizados a comunicar entre si.

Na Figura 3.3 está representado um esquema deste tipo de instalação [23–25].

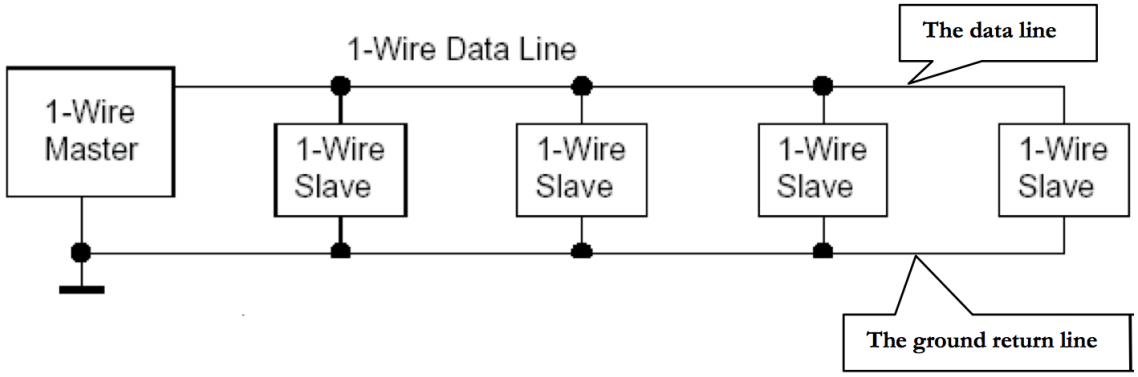


Figura 3.3: Esquema para a instalação de 1-Wire

3.2.2.2 I^2C

Inter-Integrated Communication (I^2C) significa comunicação entre integrados e é um protocolo usado na transferência de informação entre dispositivos. Foi projetado pela Philips Semiconductor (hoje NXP) no início da década de 80. Algumas das características do protocolo estão listadas em baixo [26, 27]:

- Necessidade apenas de dois fios de ligação *serial data line* (SDA) e *serial clock line* (SCL);
- Dispositivos com endereços que podem estar no mesmo barramento, com um máximo de capacitância de 400 pF;
- Velocidade até 400 Kbps (em modo rápido) e 100 Kbps modo normal;
- Barramento com capacidade de *multi-master*, havendo deteção de colisões;

- Consumo energético extremamente baixo;
- Imunidade ao ruído elétrico.

3.3 Sensores

Esta Secção apresenta os sensores que foram utilizados para o desenvolvimento da solução. Alguns extratos do código implementado podem ser consultados no Anexo A e os esquemas de montagem encontram-se disponíveis no Anexo C.

Na Figura 3.4 são apresentados alguns sensores, bem como a placa de desenvolvimento Arduino. Também está presente um Transmissor Universal que transforma o sinal elétrico de um Termopar e de um Pt100 em saídas em tensão de 0 a 5 Volt. Na Figura 3.5 é mostrado o esquema elétrico da ligação de alguns dos sensores utilizados ao Arduino.

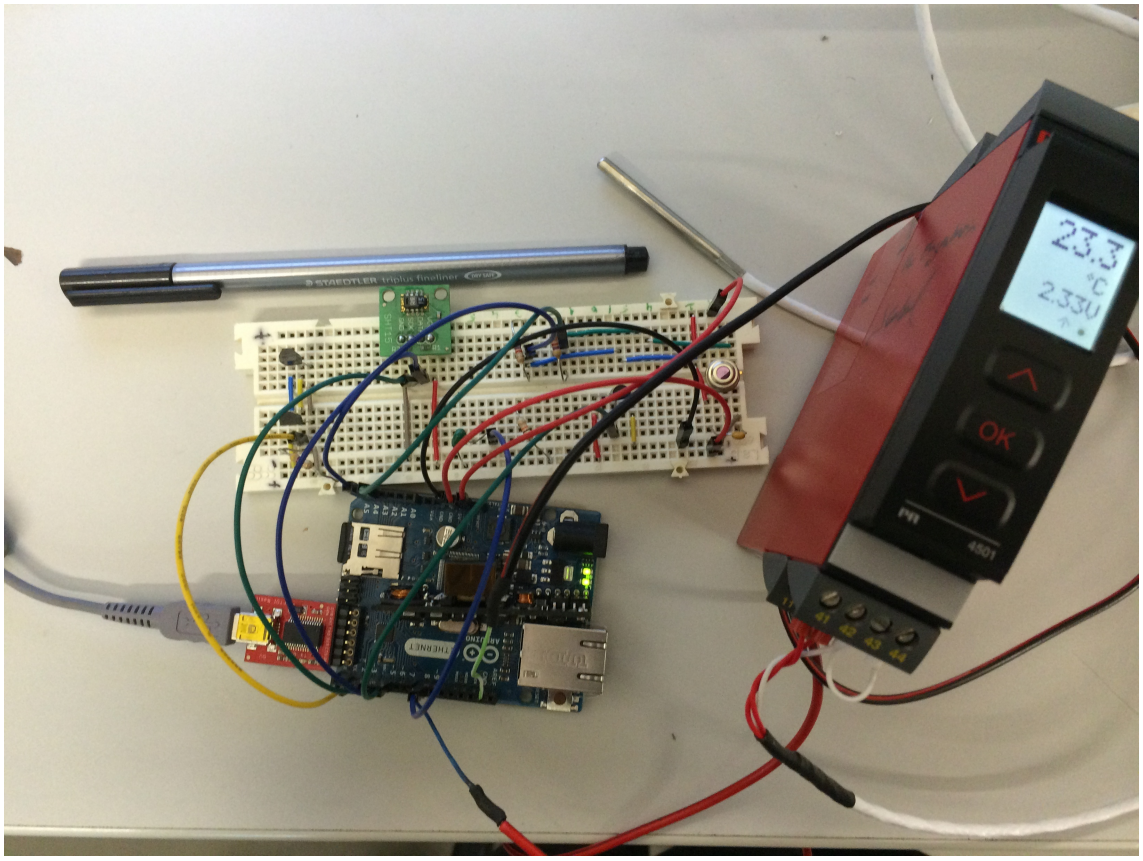


Figura 3.4: Aspeto da ligação de alguns sensores ao Arduino

A Tabela 3.1 [28–32] resume os sensores mais importantes do projeto, bem como algumas das suas características.

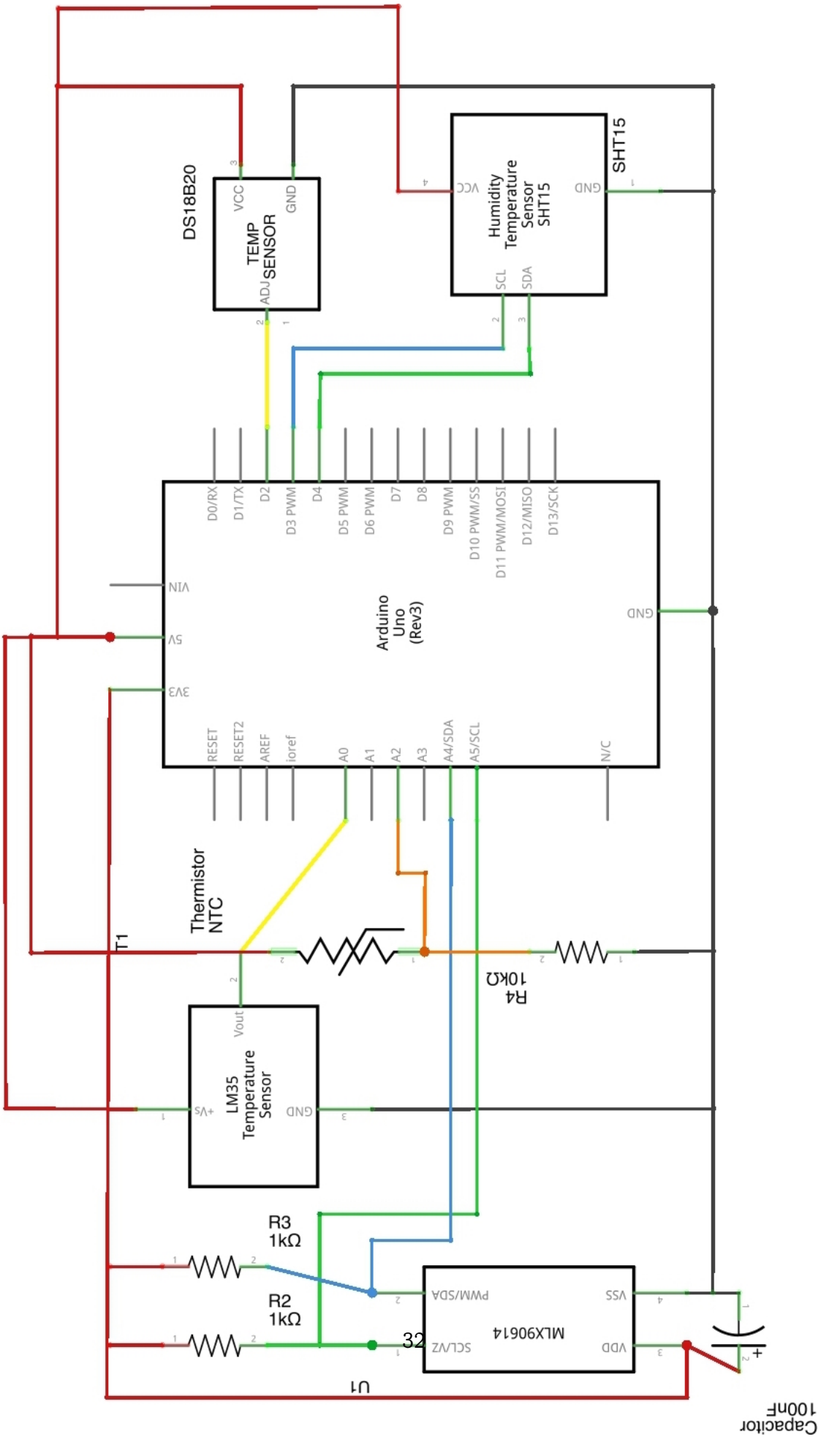
Nesta dissertação recorreu-se à implementação de tão variados sensores para se criar um vasto leque de soluções, uma vez que a aquisição de temperatura é a maior necessidade dos projetos da Laborial.

Tabela 3.1: Lista de sensores e algumas especificações

	Sensor	Fabricante	Comunicação	Resolução	Alcance
Analog	**LM35**	Texas Instruments	Analog (0-5V)	0,49 °C (10 bit)	-55 a 150 °C
	Termístor		Analog (0-5V)		-1 a 130 °C
	Termopar tipo T	A. J. Antunes & CO.	Analog (0-5V)	0,0075 × Temperatura	-40 to 180 °C
	Pt100		Analog (0-5V)		-30 to 300 °C
Digital	**DS18B20**	Maxim Integrated	1-Wire	0,05 °C	-55 a 125 °C
	SHT 15 (Temperatura)	Sensorion	I^2C	0,01 °C	-40 a 123,8 °C
	SHT 15 (Humidade)	Sensorion	I^2C	± 0,1% RH[2]	0 - 100 % RH
	MLX90614 (IR) - Ambiente	Malexis	I^2C	0,05 °C	-40 a 85 °C
	MLX90614 (IR) - Objeto	Malexis	I^2C	0,02 °C	-30 a 382,2 °C

[2]RH - Relative Humidity- Humidade Relativa

Arquitetura



3.4 RFID

RFID, ou *radio-frequency identification*[3] é a tecnologia que é utilizada para identificar pessoas, objetos e animais.

Existem diversas formas para fazer esta identificação, embora seja recorrente guardar-se um número de identificação numa cartão ou chip, que posteriormente será associado ao individuo.

Este sistema é composto por uma *tag* e por um leitor. A *tag* é a junção de um chip que conserva a informação e uma antena que a transmite.

As *tags* podem ser ativas ou passivas estando esta última limitada a distâncias de leitura menores [33, 34].

[3]Identificação por frequência rádio

Capítulo 4

Resultados

Este Capítulo apresenta os resultados alcançados no âmbito deste projeto, bem como os resultados dos testes mais relevantes.

4.1 *Data Logger*

Para implementar a funcionalidade de *data logger*, foi selecionado um modelo de Arduino que já possui um leitor de cartões, Arduino Ethernet. Mais tarde utilizou-se o Arduino UNO com um *shield* que tivesse esta funcionalidade.

A placa foi programada para escrever os dados adquiridos no cartão de memória, recorrendo à biblioteca "<SD.H>".

4.1.1 *Real-Time Clock*

Para tornar esta solução mais exata, foi incorporada na solução um RTC - *Real-Time Clock*[1], Figura 4.1. Este dispositivo irá fornecer à solução desenvolvida a hora e data real, uma vez que o Arduino não é capaz de efetuar essa contagem. O RTC necessita apenas de ser configurado uma vez, e a partir desse momento conta o tempo desde o dia 1 de janeiro de 1970 (ou o chamado *UNIX Time*).

[1] Relógio de Tempo Real

A necessidade da junção deste pequeno dispositivo foi detetada quando se verificou que a hora que estava anexa aos dados era a hora de transmissão dos mesmos e não o momento de aquisição, podendo esta questão comprometer a exatidão temporal da solução.

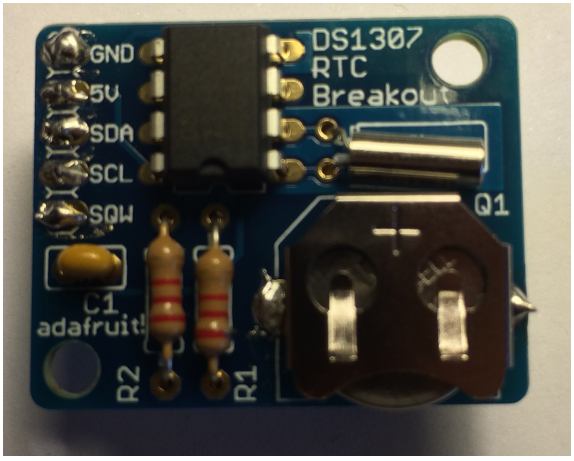


Figura 4.1: *Real-Time Clock* DS1307

4.2 Implementação do Protocolo Modbus

Como foi referido no Capítulo 3, o protocolo Modbus foi implementado de forma a estabelecer a ligação entre a solução desenvolvida e o advanLab®.

Foi importada uma biblioteca de Modbus [35] e reconfigurada para cumprir os requisitos do projeto. Uma vez que esta já tinha sido desenvolvida há bastante tempo e encontrava-se um pouco desatualizada, foi revista e foram implementadas algumas melhorias. Após esta fase de análise inicial, foram programadas as funções necessárias para suprir as necessidades da empresa. Na Figura 4.2 é apresentado o Arduino Uno conectado ao *software SCADA* por RS-232 usando Modbus.

No extrato de código do Anexo A.3, podem ser vistas as configurações a serem implementadas.

Se as definições corresponderem às do *Modbus Master*, a ligação estabelecer-se-á e os dois dispositivos irão comunicar. Na Figura 4.3 pode ser vista uma captura

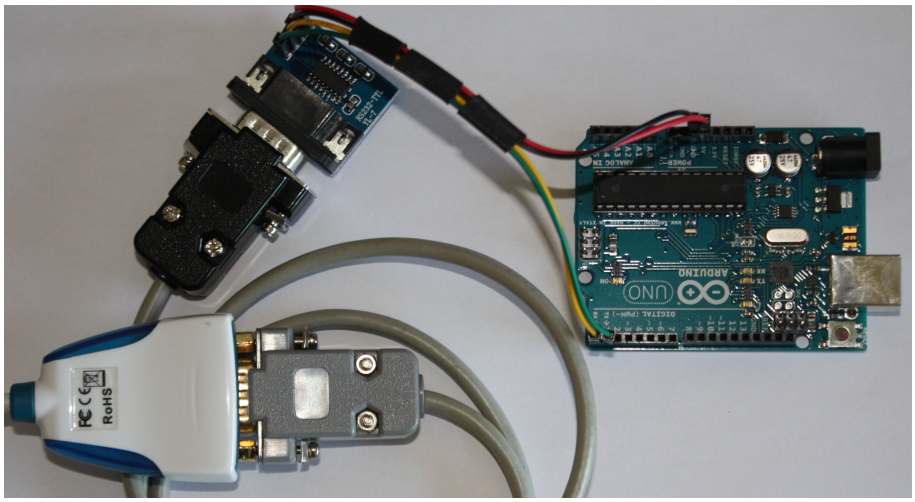


Figura 4.2: Arduino Uno conectado por Modbus sobre RS-232

de ecrã do painel do *software* utilizado no projeto para testar as configurações aplicadas.

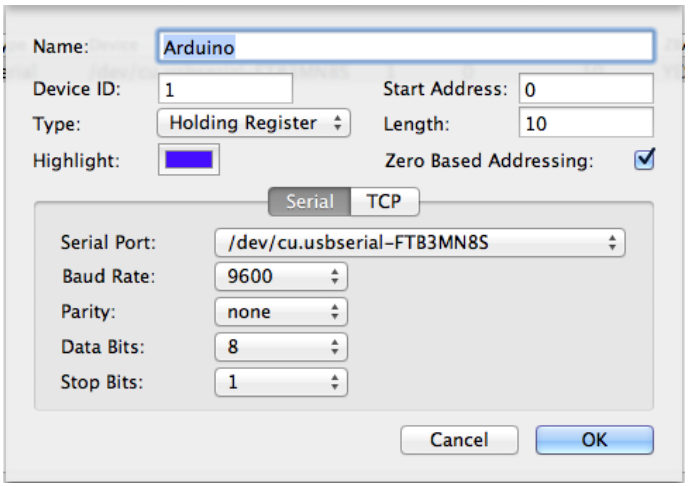


Figura 4.3: Definições do Modbus Probe

A Figura 4.4 mostra o ambiente de trabalho do Modbus Probe[2] desenvolvido pela R Engineering, Inc and Volitans Software.

[2] *Software* para testar o protocolo Modbus.

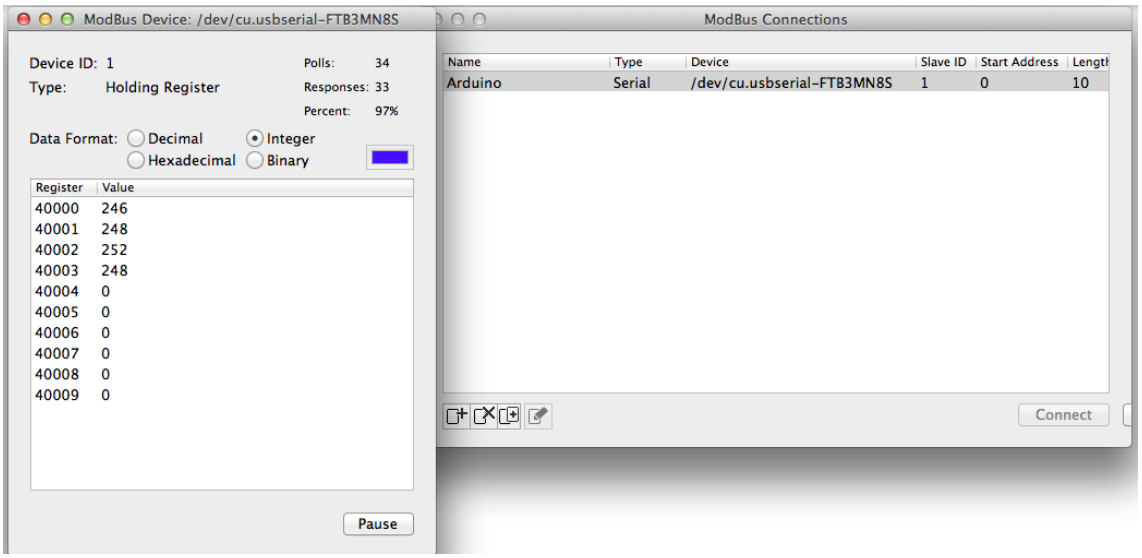


Figura 4.4: *Software* para os testes de Modbus

4.3 Redução do Consumo Energético

A implementação do *Sleep Code* tem um grande impacto na solução final, uma vez que permite uma redução do consumo energético. O Arduino disponibiliza cinco diferentes modos de *Sleep*, que se encontram listados em baixo, sendo o "SLEEP_MODE_PWR_DOWN"o que consome menos energia.

- SLEEP_MODE_IDLE
- SLEEP_MODE_ADC
- SLEEP_MODE_PWR_SAVE
- SLEEP_MODE_STANDBY
- SLEEP_MODE_PWR_DOWN

O modo implementado neste projeto foi o "SLEEP_MODE_IDLE", que é o modo que consome mais energia, embora na placa que foi utilizada para o desenvolvimento, as diferenças entre este modo e o modo mais económico não sejam muito percetíveis.

Este modo é também o único modo que deixa a porta USART ativa, porta que é utilizada para a comunicação com o *SCADA*.

Caso fosse necessário implementar qualquer outro modo de *Sleep Code* teria de ser criada uma interrupção no normal funcionamento do dispositivo, ligar o pino RX ao pino do *interrupt0*. Esta ligação garante que cada vez que o Arduino recebe informação na porta USART a interrupção acontece, provocando que o Arduino interrompa a sua rotina normal, execute o processo de comunicação e volte à sua rotina.

A implementação desta etapa demorou bastante mais tempo do que o que estava inicialmente previsto, uma vez que existiam conflitos tanto com a funcionalidade de Modbus como com a funcionalidade do *data logger*.

Em suma, a implementação desta função é um avanço em direção à solução sem fios, uma vez que com uma placa de desenvolvimento mais económica, poderá ser implementado o uso de baterias.

Na Figura 4.5 está presente uma fotografias do Arduino em funcionamento, com a funcionalidade de *Sleep Code* ativa.

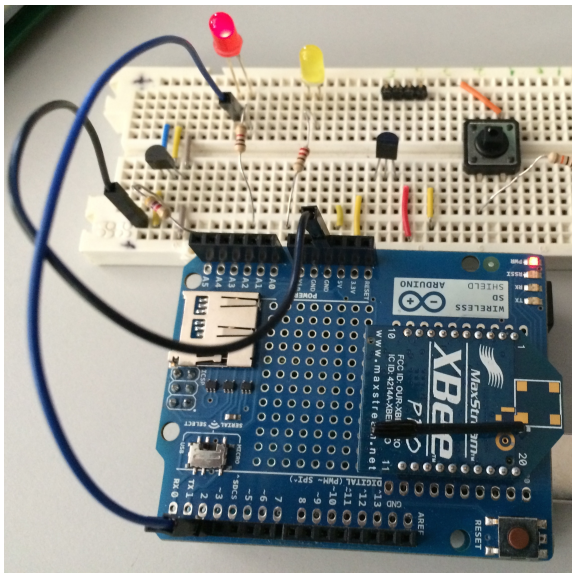


Figura 4.5: Placa de desenvolvimento com um LED sinalizador

Os dois momentos em que esta nova funcionalidade não está ativa são quando a placa se encontra a adquirir dados dos sensores e quando recebe pedidos do *SCADA*.

Estes dois ciclos correm independentemente, permitindo assim a leitura dos sensores mesmo que não haja comunicação Modbus.

Um pequeno extrato do código de implementação do *Sleep Code* é mostrado de seguida:

```
1
2  void sleepNow()
3 {
4    attachInterrupt(0, pinInterrupt, LOW);
5    set_sleep_mode(SLEEP_PWR_DOWN);    // Modo de dormir e definido aqui
6
7    sleep_enable();           /* ativa o modo de sleep mode*/
```

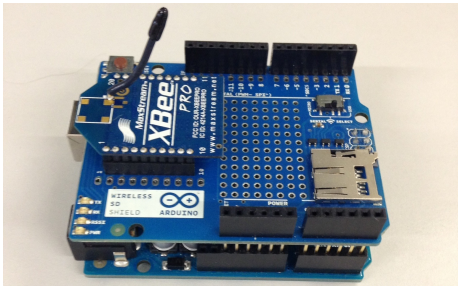
```
8
9    power_adc_disable();
10   power_spi_disable();
11   power_timer0_disable();
12   power_timer1_disable();
13   power_timer2_disable();
14   power_twi_disable();
15
16
17   sleep_mode();             // Fica realmente a dormir!
18
19                               // dps de acordar continua daqui
20   sleep_disable();
21
22   power_all_enable();
23
24 }
```

4.4 Comunicação Sem Fios Zigbee

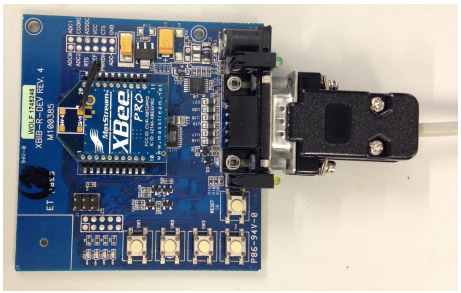
Para a solução se tornar totalmente desprovida de fios é necessário que as comunicações entre o Arduino e o *SCADA* deixem de ser cabladas. Assim foi projetada uma solução que comunica utilizando o protocolo Zigbee (baseado na norma IEEE 802.15.4) já que é de simples implementação e consome pouca energia. Para além destas duas vantagem, esta forma de comunicação é dotada da tecnologia de "Plug'n'Play".

Para implementar a comunicação Zigbee foi utilizada a antena XBee Pro da Digi (anteriormente MaxStream) e adicionado ao Arduino o *shield* "Wireless SD", Figura 4.6a. Uma vez que este *shield* já tem um leitor de cartões μSD, não é necessário mais nenhum acessório.

Para que a conexão entre os dispositivos e o *SCADA* se efetue de forma apropriada, o coordenador, Figura 4.6b, deverá estar ligado ao computador onde o sistema *SCADA* está instalado.



(a) *Shield* "Wireless SD" montado no Arduino UNO, com a antena de XBee



(b) Coordenador que deverá estar ligado ao computador

Figura 4.6: Solução com Zigbee

4.5 Implementação de RFID

A identificação RFID é usada para auxiliar o controlo do acesso ou a identificação de materiais. Para que esta funcionalidade funcione na plenitude é necessário criar uma base de dados que faça corresponder a cada número de série uma pessoa,

animal ou objeto desejado. Com esta tecnologia não é necessário haver toque, permitindo assim a sua utilização em salas limpas.

4.6 Sensor de Infravermelhos

Durante o decorrer desta dissertação foi ainda adicionado um sensor de infravermelhos que permite a medição de temperaturas de objetos sem contacto (útil para minimizar o efeito de contaminações). Desta forma, foi utilizado um sensor MLX90614 da Melexis que comunica com o Arduino via I^2C. Este sensor tem uma resolução máxima de 0,02°C e consegue fazer leituras de temperatura de objetos desde os -70 até aos 382,2 °C e na gama de -40 a 125 °C para temperatura ambiente.Este foi montado num oximentro de pulso, Figura 4.7, sendo assim possível, por exemplo, efetuar leituras de temperatura não invasivas na ponta do dedo de um paciente.

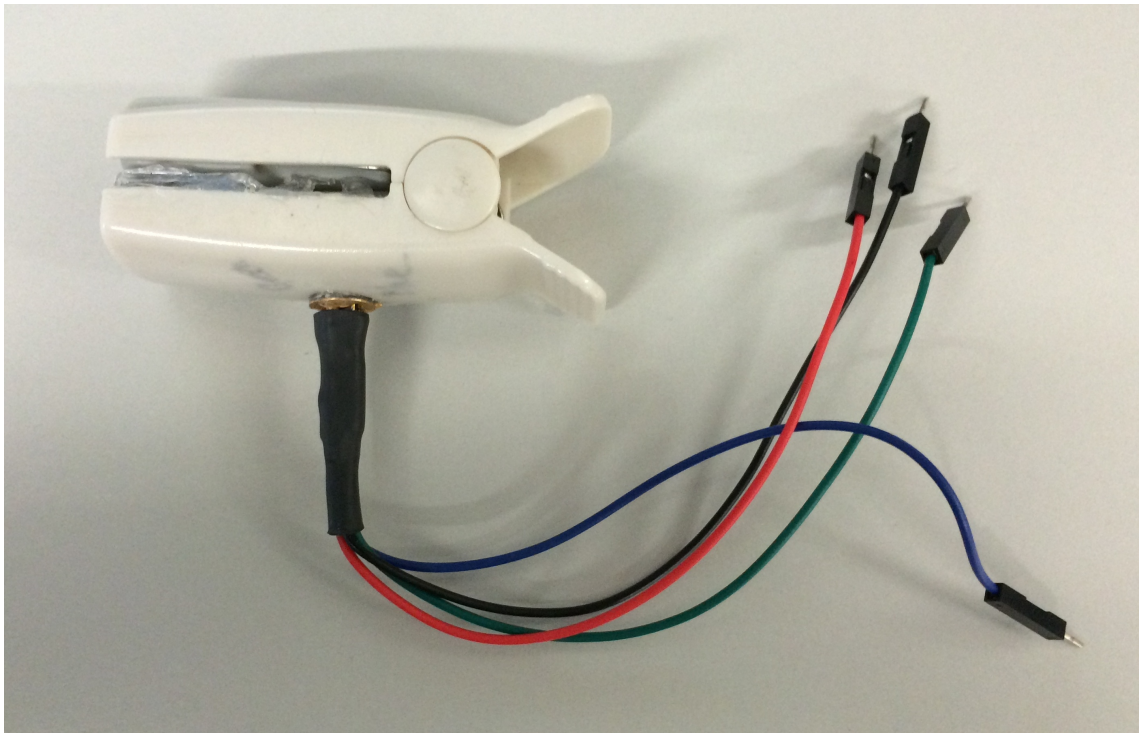


Figura 4.7: Sensor de infravermelhos montado num oximetro de pulso

4.7 Testes de Desempenho

Durante todas as fases do desenvolvimento do projeto foram efetuados vários testes com o intuito de confirmar as funcionalidades implementadas, corrigir alguns erros e antecipar alguns problemas que poderão vir a surgir.

Um dos testes realizados inicialmente foi a comparação das leituras dos sensores entre si. Após a verificação de os valores se encontrarem dentro das gamas de leitura dos mesmos, foram comparados com um pirómetro.

4.7.1 Consumo Energético

Foi feita uma análise ao consumo energético da solução desenvolvida. Para tal efetuaram-se várias medidas da corrente consumida da seguinte forma: utilizou-se uma fonte de tensão constante de 9 Volt e foi analisada a corrente consumida pelo equipamento.

A corrente consumida pela Arduino quando este se encontra com um programa que apenas o mantém ligado são 50 mA. Na Tabela 4.1 são apresentados os valores durante os diferentes estádios do programa implementado. A fase do ciclo de funcionamento onde é consumia a maior quantidade de energia é quando existe comunicação dos dados com o *SCADA*, uma vez que é necessário fornecer energia à antena de Zigbee.

Tabela 4.1: Comparação entre os diferentes estados

Corrente	**Estado**	**OBS**
36 mA	Modo *Slepping*	
50 mA	Em funcionamento	Programa "Bare Minimum"
66 mA	Adquirindo dados	
123 mA	Transmitindo dados	

4.8 Solução Final

A solução final é composta por um Arduino UNO e por um *Shield* "Wireless SD". Esta solução permite a ligação de uma antena Zigbee para comunicação sem fios, mas também a comunicação cablada, tendo sido testadas as comunicações recorrendo a USB e a RS-232. Para além desta funcionalidade básica, é possível incorporar todas as funcionalidades descritas em baixo:

- Comunicação Série (tanto USB como RS-232);
- Comunicação sem fios (Zigbee);
- *Data logger* com *RTC*;
- Sensores de temperatura;
- Detetor;
- Funcionalidade RFID
- Sensores de Luminosidade Ambiente (*LDR*);

Na Figura 4.8 é possível ver a arquitetura da solução final.

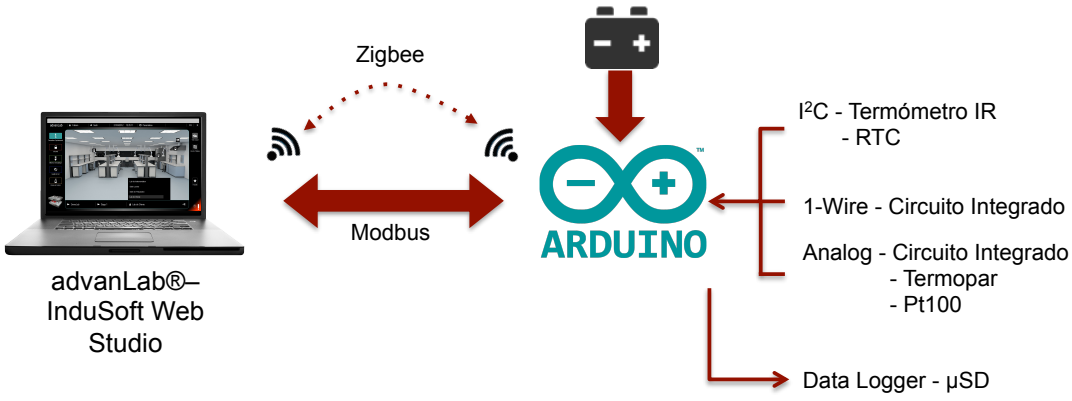


Figura 4.8: Arquitetura da solução final

O código implementado contém algumas funções base para o funcionamento do dispositivo, que são a função *setup* e *loop*.

Na primeira função são definidos os valores de algumas variáveis como as de funcionamento do Modbus, nomeadamente o *baud rate* e a paridade da ligação. É também inicializada a verificação da existência de um cartão na ranhura devida. Esta função apenas é executada no início do funcionamento do Arduino ou sempre que este for reiniciado.

De seguida está definida a função de poupança de energia, o *sleep code*. Nesta função todos os equipamentos não necessários são desligados.

A função que se encontra definida após o *sleep code* é a função de *loop*. Esta é a rotina que se repete sempre que o Arduino se encontrar em funcionamento. É aqui que são definidas as aquisições dos dados e a gravação destes para o *data logger*.

Sempre que existe um pedido por parte do servidor, o programa é interrompido e inicia-se a comunicação ao *SCADA*. Após esta comunicação ter sido terminada o programa retorna ao seu funcionamento normal.

4.9 Integração da Solução Desenvolvida no advanLab®

Para terminar os testes ao produto, a solução foi submetida a um teste em "situação real". Assim, o *hardware* desenvolvido foi ligado com sucesso ao advanLab®. Para tal, recorreu-se à comunicação sem-fios, uma vez que é a que, à partida, poderia incorrer em mais problemas. Após a configuração da base de dados do *SCADA*, que consistiu na adição de mais um módulo para ser possível a troca de informação, a comunicação foi estabelecida sem qualquer percalço, verificando-se que estavam todas as funcionalidade operacionais. Na Figura 4.9 é possível ver uma captura do ecrã do servidor, onde se destaca a atenção para o nome do módulo em funcionamento (no canto inferior esquerdo da imagem, a verde - Arduino).

Assim, conclui-se o ciclo de desenvolvimento do protótipo da solução que se almeja alcançar.

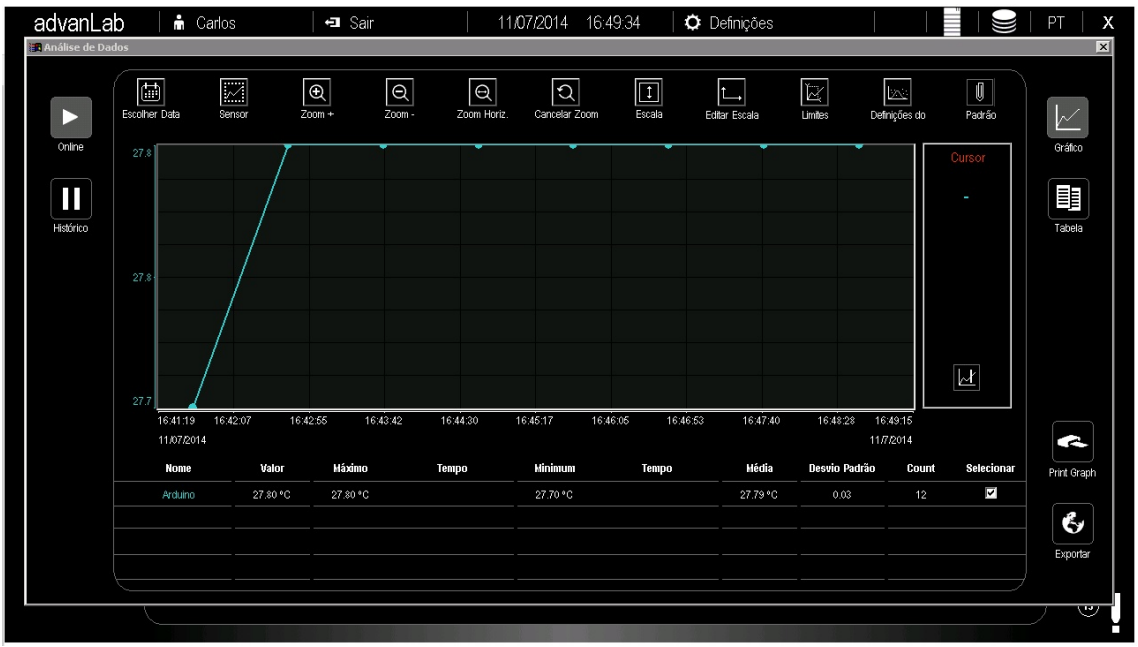


Figura 4.9: Captura do ecrã do servidor do advanLab®, com a solução desenvolvida

4.9.1 Controlo do Sistema de Anestesia de uma Bancada de Operação Veterinária

Após a integração do produto desenvolvido no advanLab® o protótipo foi integrado numa bancada de operação veterinária, para o controlo do sistema de anestesia. O esquema da Figura 4.10 monstra esquematicamente essa implementação.

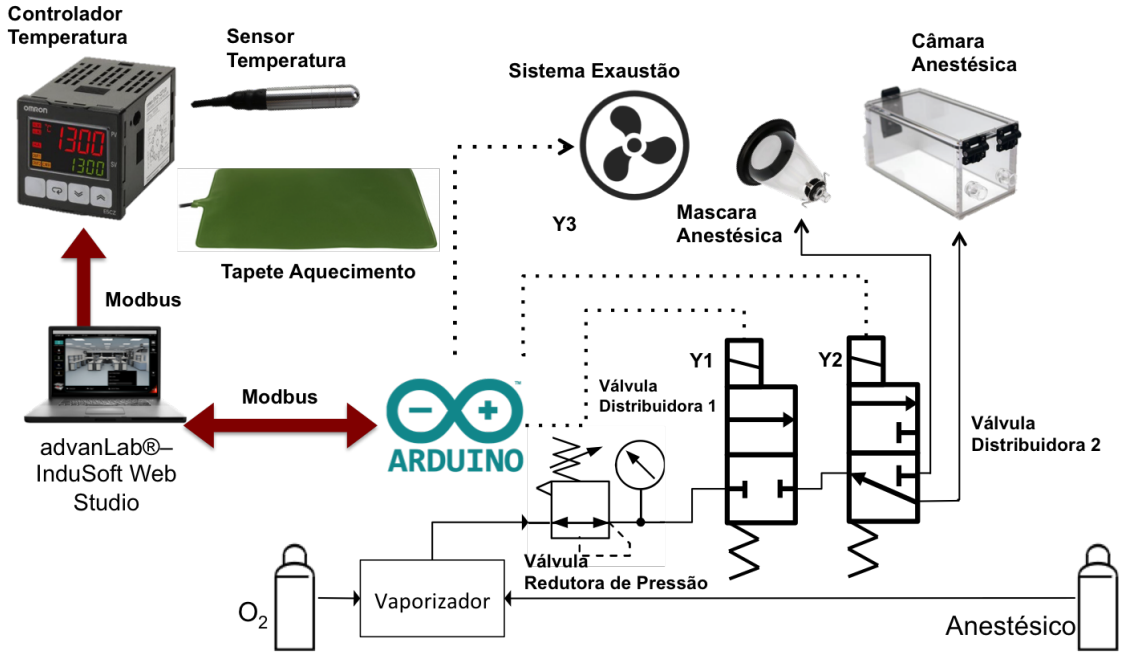


Figura 4.10: Esquema Explicativo do Controlo do Sistema de Anestesia

O sistema de exaustão do gás anestésico é controlado através do Arduino, tanto o *on/off*, como a velocidade de rotação da ventoinha, aumentando ou diminuindo o volume de ar extraído. Para controlar a administração do gás anestésico, o Arduino faz a comutação das válvulas distribuidoras. A válvula distribuidora 1 permite ou não a passagem do gás, enquanto que a válvula distribuidora 2 comanda o local para onde este se deverá dirigir, ou para a máscara ou para a câmara anestésica.

Para ligar os 2 sistemas (exaustão e administração) ao Arduino, construi-se uma placa de interface de potência para ser possível comandar a comutação das válvulas e do sistema de exaustão.

A velocidade de rotação da ventoinha do sistema de exaustão é controlada recorrendo a uma saída PWM do Arduino.

O controlo da temperatura do tapete de aquecimento é realizado pelo Controlador Omron E5CZ Q2MT, mediante a informação do sensor Termopar tipo K. Este controlador é ligado ao advanLab®, via Modbus, onde é possível definir e ler todos os parâmetros.

Um excerto do código utilizado para a implementação do Sistema Anestésico e uma ilustração da montagem do mesmo podem ser consultados nos Anexos A.4 e C.6, respetivamente. Na Figura 4.11 está presente o esquema elétrico do Sistema de Anestesia.

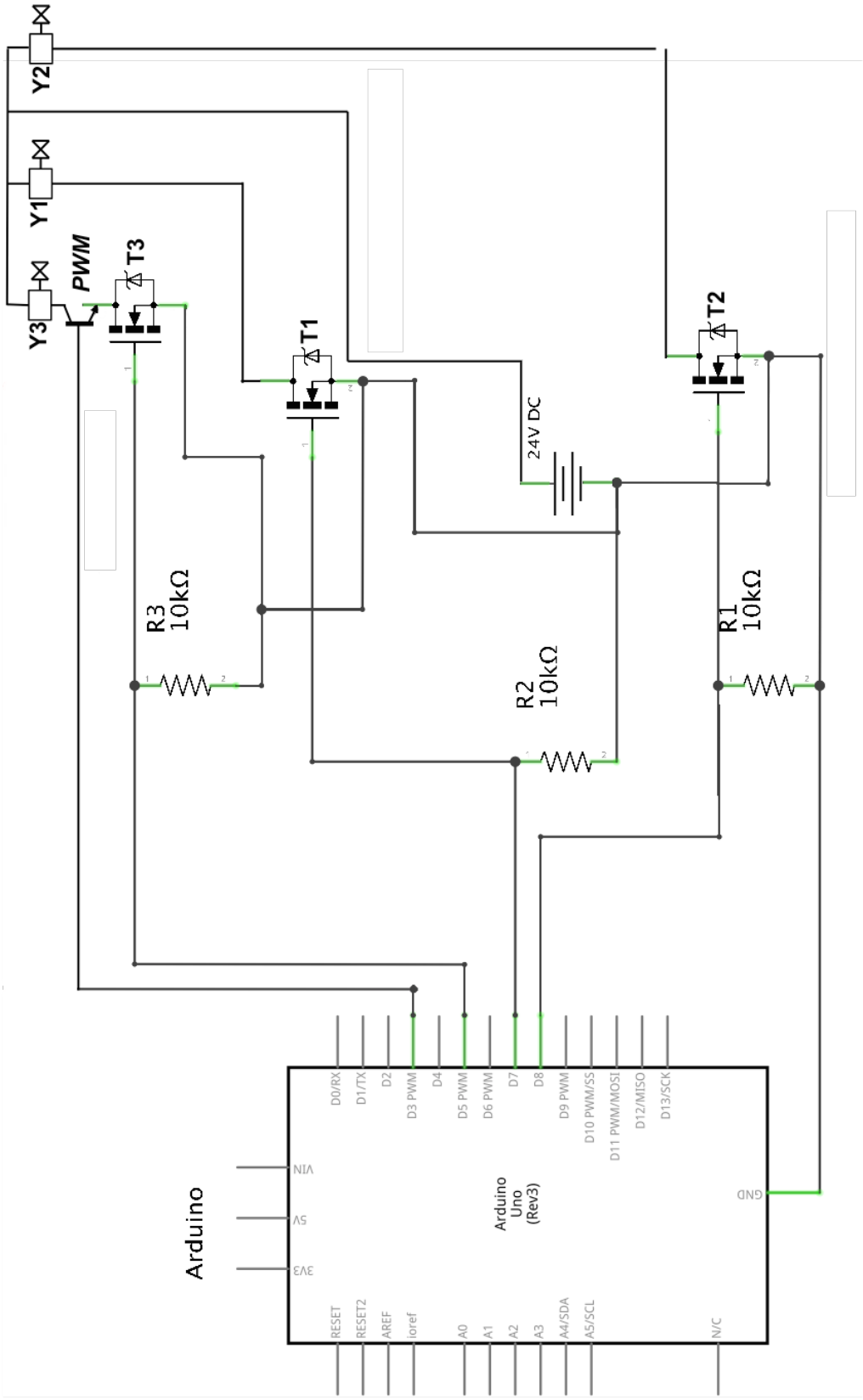


Figura 4.11: Esquema elétrico do sistema de controlo de anestesia da bancada veterinária

4.10 Discussão dos Resultados

Após a análise dos resultados e dos testes executados é possível concluir que a solução desenvolvida está implementada e a funcionar conforme previsto.

O custo desta solução, tanto do *hardware*, como do *software* pode ser analisado na Tabela 4.2, assim como um local onde é possível adquirir este material em Portugal.

O *sleep code* é uma função de extrema importância caso se queira implementar o uso de baterias, pois reduz o consumo energético do equipamento. No entanto, o *hardware* de desenvolvimento não é o melhor para este tipo de soluções. Para tal recomenda-se a utilização de um dispositivo chamado *LilyPad* (baseado na plataforma Arduino), que já é projetado para consumos reduzidos.

Tabela 4.2: Tabela sumária com o preço da solução final

	Componentes	Quantidade	Preço sem IVA	loja	sítio
Software	**Licença de desenvolvimento IWS**	1	340,00 €	InduSoft Web Site	**URL**
	Licença de *Run-Time* IWS	1	190,00 €	InduSoft Web Site	**URL**
	Arduino IDE		free	Arduino Web Site	**URL**
	XCTU		free	Digi Web Site	**URL**
Hardware	**Arduino UNO**	1	20,00 €	InMotion	**URL**
	WiFi SD Shield	1	19,90 €	InMotion	**URL**
	XBee Pro	2	2 x 34,50 €	InMotion	**URL**
	USB XBee BreakOut	1	14,95 €	InMotion	**URL**
	Cartão μSD	1	9,90 €	InMotion	**URL**
	Suporte de Bateria	1	0,40 €	InMotion	**URL**
	Bateria 9 Volt carregável	2			
	Carregador de Baterias	1			

Capítulo 5

Conclusões e Trabalhos Futuros

Os sistemas *SCADA* são ferramentas que têm sido bastante utilizadas em ambiente industrial, com provas dadas das vantagens da sua utilização. Estes sistemas também têm vindo a ser implementados em locais que até há relativamente pouco tempo a sua presença não era comum. Desta forma, estas soluções têm tido uma utilização crescente em ambientes com necessidades muito específicas como os laboratórios. Deste modo, os *SCADA* são utilizados para tornar estes locais mais "inteligentes", eficientes e seguros.

A automação laboratorial terá um papel de bastante relevância num futuro próximo, uma vez que têm sido impostos novos padrões de qualidade que só recorrendo a este tipo de solução serão possíveis de alcançar.

Sentindo uma necessidade de inovar na sua principal atividade, a Laborial aliou-se à Faculdade de Engenharia da Universidade do Porto para desenvolver um sistema *SCADA* que pudesse elevar os *standards* de laboratórios inteligentes, de onde resultou o advanLab®.

O objetivo desta dissertação é desenvolver um produto para complementar o sistema *SCADA* que foi desenvolvido num projeto anterior, denominado INTELLAB.

O projeto desenvolvido visa atender a necessidades de clientes que pretendam uma solução simples e compatível com o sistema existente, de baixo custo, com

funcionalidades flexíveis, estando contudo limitada ao nível da resolução e consumo elétrico. Para isso, optou-se pelo desenvolvimento de uma solução que pudesse integrar algumas das funcionalidades necessárias como a aquisição de dados multiponto, a gravação desses dados, e a posterior transmissão dos mesmos para o sistema central recorrendo a tecnologia sem-fios.

A arquitetura proposta é baseada num microcontrolador que adquire a informação de sensores, analógicos e digitais e comunica com o servidor recorrendo a tecnologia com fios e sem-fios. A solução desenvolvida é baseada na plataforma Arduino e implementa um *data logger* capaz de adquirir dados dos seguintes sensores:

- Termómetro Infravermelho, Sensor de humidade relativa e temperatura SHT 15 e um Relógio *RTC* DS1307 via I^2C;
- Sensor de temperatura integrado digital (Dallas DS18B20) via 1-Wire;
- Termómetro de circuito integrado, termopar, Pt100 e Sensor de iluminação ambiente (LDR) via Entrada analógica.

Foi dada especial atenção aos sensores de temperatura e humidade uma vez que são as grandezas que tem maior necessidade de serem monitorizadas.

Para além destas funcionalidades, a solução também oferece redução de consumo energético com vista à utilização de baterias. Aliando esta última funcionalidade à comunicação Zigbee (baseado na norma IEEE 802.15.4) é possível ter uma solução totalmente sem fios, passível de ser montada em locais onde seja impossível utilizar soluções cabladas. Uma das vantagens da utilização de Zigbee é o seu reduzido consumo energético, pelo que o seu uso se torna proveitoso nesta solução. O protocolo de comunicação estabelecido é o Modbus, que é um protocolo extremamente versátil e já se encontrava presente nos produtos da empresa.

O preço final do sistema desenvolvido (sem ter em conta a sensorização) é de 135,15 €. Este preço está dentro da margem que a empresa tinha idealizado, uma vez que a solução atual custa cerca de 400 € e contempla um *gateway* e um módulo

leitura. Para além disso, a solução que a empresa tem em funcionamento não dispõe da funcionalidade de *data logger*.

Embora o Arduino seja uma solução relativamente completa, este também acarreta algumas desvantagens, como por exemplo, um ADC com baixa resolução (10 bits). Recorrendo a um sensor de temperatura analógico, este tem uma resolução de cerca de 0,5 °C, que para medições de temperaturas ambientes não incorre em grandes problemas. Caso seja necessário uma solução com maior resolução (0,01 °C) é possível utilizar sensores digitais (como SHT 15 utilizado) ou um ADC externo de melhor resolução.

No âmbito deste projeto foi igualmente desenvolvida uma bancada de anestesia animal com controlo de temperatura, administração de anestesia e exaustão do gás, sendo os dois últimos controlados pelo Arduino e o primeiro diretamente pelo advanLab®. A bancada desenvolvida foi apresentada na ExpoQuimia 2014, demonstrando assim o interesse deste tipo de soluções na medicina veterinária.

Para concluir, a solução desenvolvida encontra-se totalmente funcional e implementada na Laborial, estando a funcionar de acordo com as especificações inicialmente propostas, no entanto existe sempre espaço para melhorias e novos desenvolvimentos. Desta forma, na Secção 5.1 algumas dessas melhorias e desenvolvimentos são propostos.

5.1 Trabalhos Futuros

Uma das primeiras melhorias a serem implementadas seria o estudo da construção de uma solução semelhante, mas totalmente dedicada. Quer isto dizer que se deveria estudar o desenvolvimento de uma solução apenas e só com os componentes necessários à aplicação em causa: introdução de um ADC com maior resolução, 16bit por exemplo. Para além desta melhoria seria necessário dotar a nova solução de possibilidade de armazenamento externo e de comunicar sem fios com o sistema. Assim o tamanho final da solução e o seu consumo energético diminuiriam consideravelmente.

Em alternativa a esta construção, poder-se-ia utilizar a placa *LilyPad* (baseado na plataforma Arduino) que é desenvolvida com vista ao baixo consumo energético e à alimentação por baterias.

A implementação de novas funções no protocolo Modbus desenvolvido, para abranger funcionalidades como o estado da bateria, sinal de Zigbee ou o espaço livre do cartão. A implementação destas novas funções acrescentaria valor ao que foi desenvolvido até então.

Mais alguns testes deveriam ser executados, principalmente em ambientes mais hostis para perceber realmente a robustez do produto final.

Referências

[1] Laborial. Laborial - presentation, 2010. Último Acesso: 2014.05.26.

[2] Oxford Dictionaries. "ubiquitous".

[3] Laborial. Relatório técnico-científico intellab2 nº1. Technical report, 2013.

[4] National Instruments. What is data acquisition? Último Acesso: 2014.05.31.

[5] onset. What is a data logger? **URL:** http://www.onsetcomp.com/what-is-a-data-logger, 2014. Último Acesso: 2014.05.26.

[6] Omega. Introduction to data loggers. **URL:** http://www.omega.com/prodinfo/dataloggers.html. Último Acesso: 2014.05.28.

[7] fourtec. Picolite overview. Último Acesso: 2014.05.28.

[8] Omega. Portable paperless recorders and data acquisition stations. Último Acesso: 2014.05.28.

[9] WiSensys. Wisensys base station ws-bu. **URL:** http://www.wisensys.com/sites/default/bestanden/products/sheets/Product%20sheet%20WS-BU_0.pdf, 2011. Último Acesso: 2014.07.15.

[10] WiSensys. Wisensys wireless sensor ws-dltc. **URL:** http://www.wisensys.com/sites/default/bestanden/products/sheets/Product_sheet_WS-DLTc.pdf, 2011. Último Acesso: 2014.07.15.

[11] Emanuel Silva. Software para o intellab. Technical report, Laborial, 2010.

[12] VAISALA. Vaisala wi-fi data logger hmt140 for multiple environmental parameters.

[13] Luciano Silva. Raspberry pi. Technical report, Laborial, 2014.

[14] Advantech. Wireless i/o modules: Adam-2000. **URL:** http://www.advantech.com/products/Wireless-Sensor-Networks/sub_Wireless_Sensor_Networks.aspx. Último Acesso: 2014.07.22.

[15] LinuxUser&Developes. Beaglgebone black review. **URL:** http://www.linuxuser.co.uk/reviews/beaglebone-black-review, 2013. Último Acesso: 2014.06.06.

[16] Luciano Silva. Beaglgebone black. Technical report, Laborial, 2014.

[17] BeagleBone Black. Bleaglebone black. **URL:** http://beagleboard.org/Products/BeagleBone+Black. Último Acesso: 2014.06.06.

[18] Arduino. Arduino board uno. Último Acesso: 2014.06.06.

[19] Carlos Azevedo. Intellab - laboratory solutions. Technical report, Laborial, 2011.

[20] The ModBus Organization. Modbus faq: About the protocol. **URL:** http://www.modbus.org/faq.php. Último Acesso: 2014.04.02.

[21] Control Engineering. Modbus in automation and process control, 2007.

[22] Carlos Azevedo. Comando e monitorização de sistemas de actuação via redes wireless – zigbee. Master's thesis, 2010.

[23] Springbook Digitronics. *Design Guide v1.0 - 1-Wire FAQ*, 2004.

[24] N. Montoya; L. Giraldo; D. Aristizá; A. Montoya. Remote monitoring and control system of physical variables of a greenhouse through a 1-wire network. 2006.

[25] Maxim Integrated™. 1-wire devices. **URL:** http://www.maximintegrated.com/products/1-wire/, 2014. Último Acesso: 2014.04.08.

[26] Philips Semiconductores. The i²c-bus and how to use it. Technical report, Philips Semiconductores, 1995.

[27] I²CBus-Organization. I2c-bus: What's that? **URL:** http://www.i2c-bus.org. Último Acesso: 2014.06.14.

[28] Maxim Integrated™. Ds18s20 - 1-wire® parasite-power digital thermometer. **URL:** http://www.maximintegrated.com/datasheet/index.mvp/id/2815. Último Acesso: 2014.03.10.

[29] Sparkfun. Infrared thermometer - mlx90614. **URL:** https://www.sparkfun.com/products/9570. Último Acesso: 2014.03.10.

[30] Texas Instruments. Lm35 precision centigrade temperature sensors. Technical report, Texas Instruments, 2013.

[31] Sparkfun. Humidity and temperature sensor breakout - sht15. Technical report, Sparkfun, 2010.

[32] MicroChipTechno. **URL:** http://www.microchiptechno.com/ntc_thermistors.php. Último Acesso: 2014.03.07.

[33] Stephen A. Weis. Rfid (radio frequency identification): Principles and applications.

[34] RFIDJournal. Rfid frequently asked questions. **URL** :http://www.rfidjournal.com/faq/. Último Acesso: 2014.06.25.

[35] Samuel Marco Juan Pablo Zometa and Andras Tucsni. Arduino modbus slave. **URL:** https://sites.google.com/site/jpmzometa/, 2010. Último Acesso: 2014.03.30.

[36] Vishal Bhingare. Automation - open source hardware - arduino. **URL:** http://vishalbhingare.wordpress.com/2013/09/23/arduino-temperature-sensor-lm35/, September 2013. Último Acesso: 2014.03.06.

[37] bildr blog. One wire digital temperature. ds18b20 + arduino. **URL:** http://bildr.org/2011/07/ds18b20-arduino/, July 2011. Último Acesso: 2014.03.07.

[38] bildr blog. Sensing humidity with the sht15 + arduino. **URL:** http://bildr.org/2012/11/sht15-arduino/, November 2012. Último Acesso: 2014.03.10.

[39] bildr blog. Is it hot? arduino + mlx90614 ir thermometer. **URL:** http://bildr.org/2011/02/mlx90614-arduino/, February 2011. Último Acesso: 2014.03.10.

REFERÊNCIAS

Anexo A

Código Arduino

Neste Apêndice serão apresentados extratos de códigos relevantes para o projeto.

A.1 Código Usado para Testar os Sensores

Nesta Secção será apresentado o código usado para testar alguns sensores. Todas as licenças de utilização e alteração do código estão presentes no Apêndice B

A.1.1 Sensores Analógicos

Código para leitura de sinais analógicos, nomeadamente o sensor LM35:

A.1.1.1 LM35

```
1 \\The LM35 sensor should be connected to the Analog Input 0
2 void setup() {
3   Serial.begin(9600);
4 }
5
6 void loop() {
7   int LM35 = analogRead(A0);
8
```

```
 float temp = (100.0 * 5.0 / 1023.0) * LM35

  Serial.println(temp);
  delay(100);
}
```

A.1.1.2 Térmistor NTC

```

\\Thermistor should be connected to the Analog Input 1
void setup() {
  Serial.begin(9600);
}

void loop() {
  int thermistor = analogRead(A1);

 float temp = (5.0 / 1023.0) * thermistor

  Serial.println(temp);
  delay(100);
}
```

A.1.2 Sensores Digitais

Será apresentada a forma de leitura de alguns sensores digitais. O primeiro usa o protocolo 1-Wire, o segundo usa I^2C.

A.1.2.1 Sensor DS18B20

```
#include <OneWire.h>

int DS18S20_Pin = 2; //DS18S20 Signal pin on digital 2

//Temperature chip i/o
```

```
6  OneWire ds(DS18S20_Pin);  // on digital pin 2
7
8  void setup(void) {
9    Serial.begin(9600);
10 }
11
12 void loop(void) {
13   float temperature = getTemp();
14   Serial.println(temperature);
15   delay(100);
16 }
17
18
19 float getTemp(){
20   //returns the temperature from one DS18S20 in DEG Celsius
21   byte data[12];
22   byte addr[8];
23
24   if ( !ds.search(addr)) {
25       //no more sensors on chain, reset search
26       ds.reset_search();
27       return -1000;
28   }
29
30   if ( OneWire::crc8( addr, 7) != addr[7]) {
31       Serial.println("CRC is not valid!");
32       return -1000;
33   }
34
35   if ( addr[0] != 0x10 && addr[0] != 0x28) {
36       Serial.print("Device is not recognized");
37       return -1000;
38   }
39
40   ds.reset();
41   ds.select(addr);
42   ds.write(0x44,1); // start conversion, with parasite power on at the
         end
43
```

```
44   byte present = ds.reset();
45   ds.select(addr);
46   ds.write(0xBE); // Read Scratchpad
47
48   for (int i = 0; i < 9; i++) { // we need 9 bytes
49     data[i] = ds.read();
50   }
51
52   ds.reset_search();
53   byte MSB = data[1];
54   byte LSB = data[0];
55
56   float tempRead = ((MSB << 8) | LSB); //using two's compliment
57   float TemperatureSum = tempRead / 16;
58   return TemperatureSum;
59 }
```

A.1.2.2 Sensor InfraRed MLX

```
1
2 #include <i2cmaster.h>
3
4
5 void setup(){
6   Serial.begin(9600);
7   Serial.println("Setup...");
8
9   i2c_init(); //Initialise the i2c bus
10  PORTC = (1 << PORTC4) | (1 << PORTC5);//enable pullups
11 }
12
13 void loop(){
14     int dev = 0x5A<<1;
15     int data_low = 0;
16     int data_high = 0;
17     int pec = 0;
18
19     i2c_start_wait(dev+I2C_WRITE);
```

```
20      i2c_write(0x07);
21
22      // read
23      i2c_rep_start(dev+I2C_READ);
24      data_low = i2c_readAck(); //Read 1 byte and then send ack
25      data_high = i2c_readAck(); //Read 1 byte and then send ack
26      pec = i2c_readNak();
27      i2c_stop();
28
29      //This converts high and low bytes together and processes
            temperature, MSB is a error bit and is ignored for temps
30      double tempFactor = 0.02; // 0.02 degrees per LSB (measurement
            resolution of the MLX90614)
31      double tempData = 0x0000; // zero out the data
32      int frac; // data past the decimal point
33
34      // This masks off the error bit of the high byte, then moves it
            left 8 bits and adds the low byte.
35      tempData = (double)(((data_high & 0x007F) << 8) + data_low);
36      tempData = (tempData * tempFactor)-0.01;
37
38      float celsius = tempData - 273.15;
39
40      Serial.print("");
41      Serial.print("Celsius degree: ");
42      Serial.println(celsius);
43
44
45      delay(1000); // wait a second before printing again
46  }
```

A.2 Data Logger

Nesta Secção é apresentado o código para implementação do *data logger* no Arduino.

```
1 #include <SD.h>
```

```
2
3 // On the Ethernet Shield, CS is pin 4.
4 const int chipSelect = 4;
5
6 void setup()
7 {
8   Serial.begin(9600);
9    while (!Serial) {
10     ;
11   }
12
13
14   Serial.print("Initializing SD card...");
15
16   pinMode(10, OUTPUT);
17
18   // see if the card is present and can be initialized:
19   if (!SD.begin(chipSelect)) {
20     Serial.println("Card failed, or not present");
21     // don't do anything more:
22     return;
23   }
24   Serial.println("card initialized.");
25 }
26
27 void loop()
28 {
29   // make a string for assembling the data to log:
30   String dataString = "";
31
32   // read three sensors and append to the string:
33   for (int analogPin = 0; analogPin < 3; analogPin++) {
34     int sensor = analogRead(analogPin)* (500.0 / 1023.0);
35     dataString += String(sensor);
36     if (analogPin < 2) {
37       dataString += ",";
38     }
39   }
40
```

```
41    // open the file. note that only one file can be open at a time,
42    // so you have to close this one before opening another.
43    File dataFile = SD.open("datalog.txt", FILE_WRITE);
44
45    // if the file is available, write to it:
46    if (dataFile) {
47      dataFile.println(dataString);
48      dataFile.close();
49      // print to the serial port too:
50      Serial.println(dataString);
51    }
52    // if the file isn't open, pop up an error:
53    else {
54      Serial.println("error opening datalog.txt");
55    }
56  }
```

A.3 Modbus e *Sleep Code*

Nesta Secção é apresentado o código para implementação do *sleep code* bem como do Modbus.

```
1  #include <SD.h>
2  #include <avr/interrupt.h>
3  #include <avr/power.h>
4  #include <avr/sleep.h>
5  #include <avr/wdt.h>
6
7  volatile int sleep_count = 0;
8  const int interval=1; //minutes between acquisiions
9  const int sleep_total = (interval*60)/8; // Approximate number
10 // of sleep cycles needed before the interval defined above
11 // elapses. Not that this does integer math.
12
13
14 int sleepStatus = 0 ;
15 int count = 0;
```

```
16 const int chipSelect = 4;
17 void configure_mb_slave(long baud, char parity, char txenpin);
18
19 int update_mb_slave(unsigned char slave, int *regs,
20 unsigned int regs_size);
21
22
23 enum {
24         MB_SLAVE = 1, /* id do slave*/
25 };
26
27
28 enum {  MB_REG1,
29         MB_REG2,
30         MB_REG3,
31         MB_REG4,
32         MB_REG5,
33         MB_REG6,
34         MB_REG7,
35         MB_REG8,
36         MB_REG9,
37         MB_REG10,
38         MB_REGS   /* numero de total de registos no Slave*/
39 };
40
41 int regs[MB_REGS];  /* mapa de registos do slave */
42
43
44 void sleepNow()
45 {
46   set_sleep_mode(SLEEP_MODE_IDLE);   // sleep mode is set here
47   sleep_enable();          // enables the sleep bit in the mcucr
       register
48                            // so sleep is possible. just a safety pin
49   power_adc_disable();
50   power_spi_disable();
51   power_timer0_disable();
52   power_timer1_disable();
53   power_timer2_disable();
```

```
54    power_twi_disable();
55    sleep_cpu();
56
57    sleep_mode();            // here the device is actually put to sleep
          !!
58                             // THE PROGRAM CONTINUES FROM HERE AFTER
                                   WAKING UP
59    sleep_disable();         // first thing after waking from sleep:
60    power_all_enable();
61    digitalWrite(3,LOW);
62    digitalWrite(5,LOW);
63 }
64
65
66 void setup()
67 {
68    watchdogOn();
69   /* ADCSRA = ADCSRA & B01111111; //disables ADC
70    ADCSRA = ADCSRA & B01111111; //disables analog comparator
71    DIDR0 = DIDR0 | B00111111; // disables digital inputs.
72    */
73
74    pinMode(10,OUTPUT);
75    SD.begin(chipSelect);
76    configure_mb_slave(9600, 'n', 0);
77 //Serial.begin(9600);
78 //Serial.println("comecar");
79
80
81 }
82
83
84
85 void loop()
86 {
87     update_mb_slave(MB_SLAVE, regs, MB_REGS);
88
89
90   regs[0]=analogRead(A0);
```

```
91    regs[1]=123;
92    regs[2]=now();
93   // regs[3]=analogRead(A3);
94    //regs[4]=digitalRead(4);
95  delay(100);
96
97  if (sleep_count >= sleep_total) {
98     File dataFile = SD.open("datalog.txt",FILE_WRITE);
99    dataFile.println(sleep_count);
100   dataFile.close();
101
102
103 digitalWrite(5,HIGH);
104 sleep_count=0;
105
106 }
107
108
109
110 if (count >= 50) {
111
112       delay(10);  // this delay is needed, the sleep
113       count = 0;  //function will provoke a Serial error otherwise!!
114       digitalWrite(3,HIGH);
115 //       datalogger();
116       sleepNow();     // sleep function called here
117
118   }
119 //Serial.println(sleep_count);
120 count ++;
121 sleep_count ++;
122 }
123
124 void watchdogOn() {
125
126 // Clear the reset flag, the WDRF bit (bit 3) of MCUSR.
127 MCUSR = MCUSR & B11110111;
128
129 // Set the WDCE bit (bit 4) and the WDE bit (bit 3)
```

```
130 // of WDTCSR. The WDCE bit must be set in order to
131 // change WDE or the watchdog prescalers. Setting the
132 // WDCE bit will allow updtaes to the prescalers and
133 // WDE for 4 clock cycles then it will be reset by
134 // hardware.
135 WDTCSR = WDTCSR | B00011000;
136
137 // Set the watchdog timeout prescaler value to 1024 K
138 // which will yeild a time-out interval of about 8.0 s.
139 WDTCSR = B00100001;
140
141 // Enable the watchdog timer interupt.
142 WDTCSR = WDTCSR | B01000000;
143 MCUSR = MCUSR & B11110111;
144
145 }
146
147 ISR(WDT_vect)
148 {
149 sleep_count ++; // keep track of how many sleep cycles
150 // have been completed.
151 }
152
153 /*
    ********************************************************************************

154  * BEGIN MODBUS RTU SLAVE FUNCTIONS
155  ********************************************************************************
     */
```

A.4 Controlo Sistema de Anestegia

Nesta Secção é apresentado o código para implementação do sistema de controlo de anestesia.

```
1 int update_mb_slave(unsigned char slave, int *regs,
2 unsigned int regs_size);
```

```
3
4 /*  o InduSoft TEM de respeitar estes parametros para conseguir
      comunicar com o arduino*/
5 enum {
6         MB_SLAVE = 1, /* id do slave*/
7 };
8
9
10 enum {   MB_REG1,
11         MB_REG2,
12         MB_REG3,
13         MB_REG4,
14         MB_REG5,
15         MB_REG6,
16         MB_REG7,
17         MB_REG8,
18         MB_REG9,
19         MB_REG10,
20         MB_REGS   /* numero de total de registos no Slave*/
21 };
22
23 int regs[MB_REGS];  /* mapa de registos do slave */
24
25 void setup()
26 {
27         configure_mb_slave(9600, 'n', 0);
28           // initialize serial communication at 9600 bits per second:
29         pinMode(3, OUTPUT);
30         pinMode(11, OUTPUT);
31         TCCR2A = _BV(COM2A0) | _BV(COM2B1) | _BV(WGM21) | _BV(WGM20);
32         TCCR2B = _BV(WGM22) | _BV(CS22);
33         OCR2A = 9;
34         OCR2B = 0
35         ;
36         pinMode(7, OUTPUT);
37         pinMode(8, OUTPUT);
38         pinMode(5, OUTPUT);
39
40
```

```
41 }
42 void loop()
43 {
44
45
46
47
48         /* This is all for the Modbus slave */
49   update_mb_slave(MB_SLAVE, regs, MB_REGS);
50
51         regs[0]= 0;
52   regs[1]=0;
53         regs[2]=0;
54         OCR2B = regs[3];
55         regs[4]=0;
56
57          if (regs[5]!=0)
58        {
59          digitalWrite(5,HIGH);
60        }
61       else
62      {
63        digitalWrite(5,LOW);
64      }
65
66
67         regs[6]=0;
68
69
70
71           if (regs[7]!=0)
72         {
73           digitalWrite(7,HIGH);
74         }
75         else
76         {
77         digitalWrite(7,LOW);
78         }
79
```

```
80
81            if (regs[8]!=0)
82          {
83            digitalWrite(8,HIGH);
84          }
85         else
86        {
87          digitalWrite(8,LOW);
88        }
89
90           regs[9]=0;
91
92
93
94         delay(50);         // delay in between reads for stability
95
96
97 }
```

Anexo B

Licença da Utilização de Código

Licença da Utilização de Código

Anexo C

Ilustração das Ligações dos Sensores ao Arduino

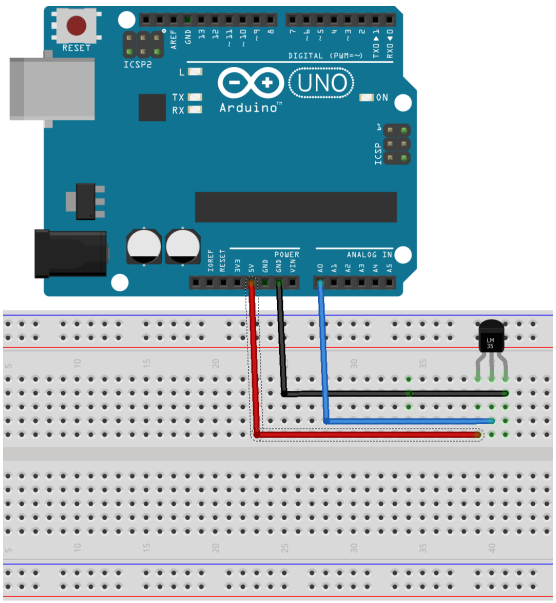


Figura C.1: Esquema de Ligação entre o Arduino e o sensor de temperatura analógico integrado LM35

[36]

Ilustração das Ligações dos Sensores ao Arduino

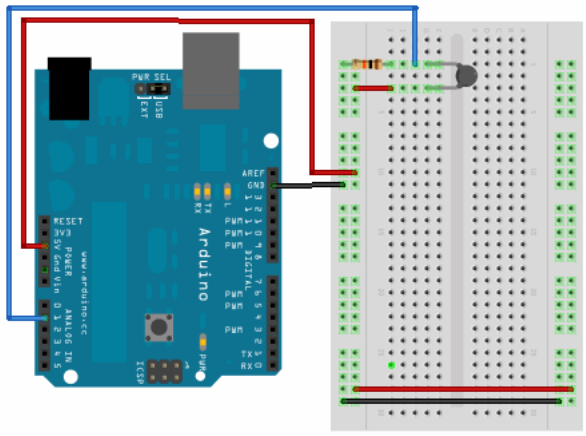


Figura C.2: Esquema de Ligação entre o Arduino e um Termístor

Ilustração das Ligações dos Sensores ao Arduino

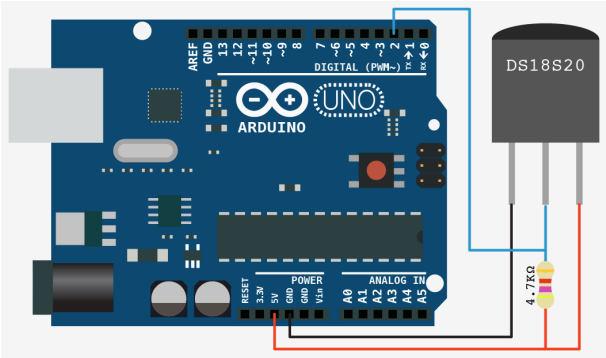


Figura C.3: Esquema de ligação entre o sensor integrado Dallas DS18B20 e o Arduino

[37]

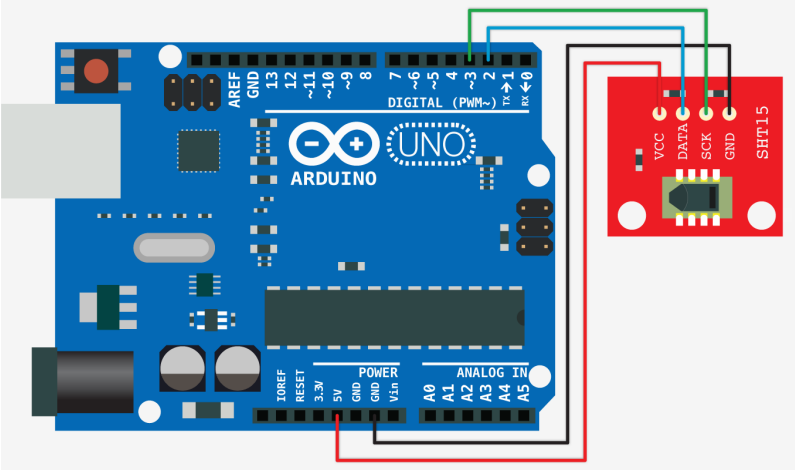


Figura C.4: Ligação Sensor de Humidade Relativa e Temperatura - SHT15 - ao Arduino

[38]

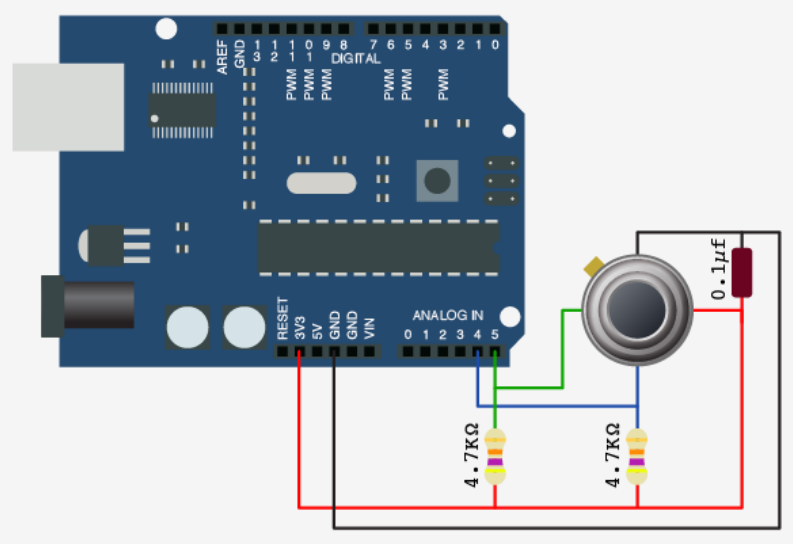


Figura C.5: Esquema de Ligação entre o Arduino e o Sensor de Temperatura Infravermelhos

[39]

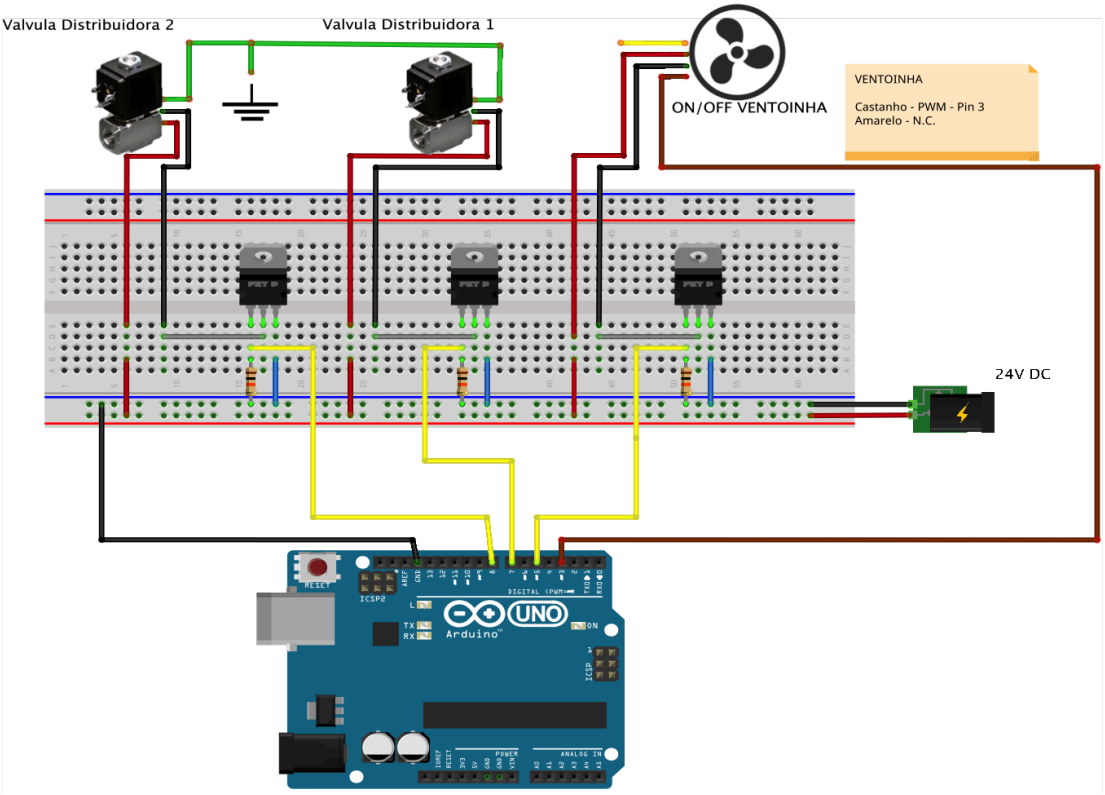


Figura C.6: Ilustração da Montagem do Controlo do Sistema Anestésico