Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Paulo Jorge Craveiro Ferreira

# Arduino Science Kits
# Plataforma Open-Hardware para práticas laboratoriais no ensino das Ciências Experimentais

**Tese de Mestrado**
em Sistemas e Tecnologias de Informação para Organizações

Professor Doutor Jorge Alexandre de Albuquerque Loureiro

Outubro de 2015

## Agradecimentos

Um grande bem-haja a toda a minha família, sem a qual este esforço não faria sentido e cujo apoio incondicional permitiu manter o ânimo e a força ao longo desta caminhada.

É meu dever deixar aqui registado, também, um agradecimento ao meu orientador, Professor Doutor Jorge Alexandre Loureiro, pela sua disponibilidade, pelos seus conselhos, pela sua competência e orientação que foram um contributo fundamental para este projeto.

Por fim, quero agradecer a todos que contribuíram, direta ou indiretamente, para a realização deste trabalho.

## RESUMO

Este trabalho propõe uma plataforma de *hardware* e software baseada no *Arduino* para suporte a atividades laboratoriais da disciplina de Física.

No ensino das disciplinas das ciências experimentais a demonstração prática é fundamental e legalmente obrigatória. Estas demonstrações implicam a aquisição de *kits* de experiências por parte das Escolas para utilização nos laboratórios das disciplinas de Física, Química e Biologia.

Os *kits* existentes no mercado são caros e pouco flexíveis, normalmente funcionam para uma só experiência e não é fácil extrair os dados para posterior análise de forma flexível, como por exemplo, numa folha de cálculo. Além disto, o preço destes *kits* dificulta a execução das tarefas pelos próprios alunos, uma vez que limita o número de experiências que podem ser executadas em simultâneo.

Com este projeto, pretende-se desenvolver uma plataforma de captura e tratamento de dados recolhidos por diversos sensores, com recurso a uma ou várias plataformas *Arduino* que possam ser utilizadas de forma flexível pelos docentes, em sala de aula, com os seus alunos.

O *Arduino* é uma plataforma eletrónica *open-source* que inclui um microcontrolador programável com uma linguagem baseada em C/C++ e que dispõe de portas digitais e analógicas onde podem ser ligados diversos sensores para interagir com o mundo.

Uma das motivações para este projeto é a necessidade existente nas Escolas de disporem de um modo mais acessível de executar as experiências. Mais acessível do ponto de vista económico e do ponto de vista de operacionalização. Não é de desprezar, também, o valor pedagógico de serem os próprios alunos a construírem os *kits* a partir dos componentes, podendo, deste modo, desfrutar de um ensino mais participativo e aliciante.

Existe também uma motivação pessoal que justifica por si só o projeto, mas que, aliada à necessidade existente no sistema de ensino e sentida pelos docentes, permite considerar uma vertente prática e financeira do mesmo.

A arquitetura geral da plataforma é composta por três níveis, ou camadas:

- No nível mais baixo estará o Arduino e os sensores. Neste nível é implementado um protocolo de comunicação que se pretende que seja universal, de modo a suportar todas as experiências. Este funcionará como um sistema operativo para a plataforma, gerindo os sensores e controlando a comunicação.

- Num nível intermédio foi desenvolvido o mesmo protocolo de comunicação, mas do lado das aplicações. Este componente foi implementado para as plataformas Windows .NET e Android e dá suporte às aplicações de nível superior. Estes componentes poderão, no futuro, ser reutilizados na execução de outros projetos, daí a sua separação na arquitetura geral da plataforma.
- No nível superior encontram-se as aplicações com as quais os docentes e os discentes vão interagir. Foi implementado sob a forma de uma aplicação para Windows que permite ligar um ou vários Arduinos em simultâneo e outra aplicação para Android que se liga ao Arduino através de Bluetooth. Em geral estas aplicações permitem capturar e apresentar os dados bem como partilhá-los com outras aplicações.

Durante a fase de desenvolvimento o projeto foi testado com um grupo de alunos em sala de aula. O teste foi realizado numa fase em que as aplicações ainda se encontravam em desenvolvimento devido aos constrangimentos de tempo, os alunos estavam já na última semana de aulas do ano letivo. O objetivo deste teste foi realizar uma avaliação do trabalho realizado.

Foi interessante ver a reação de curiosidade dos discentes quando lhes foi apresentado o Arduino e os sensores, tendo os alunos participado na construção do aparato a utilizar na experiência, instalando os componentes no Arduino.

Nesta sessão foram realizadas duas atividades laborais: Queda Livre e Bola Saltitona. A aplicação de captura de dados utilizada foi a versão Windows, ainda em versão inacabada, mas que não foi de difícil utilização, tendo os alunos compreendido com facilidade os comandos disponíveis bem como os dados que foram sendo apresentados.

Os resultados obtidos foram precisos e dentro das margens de erro previsíveis para as atividades laboratoriais em sala de aula. Para além disso a plataforma despertou nos alunos o interesse em conhecer melhor os seus componentes. É, por isso, possível dizer que a plataforma produziu resultados não só ao nível das atividades laboratoriais, mas também ao nível pedagógico.

O trabalho não termina aqui. De futuro podem ser adicionadas novas funcionalidades que permitam abranger outras disciplinas e outras atividades laboratoriais.

## ABSTRACT

This work proposes a hardware and software solution based in the Arduino platform for teaching Physics laboratory activities. In science teaching the experimental activities are legally mandatory in Portugal and highly recommend as a tool to help students comprehend some abstract concepts.

This demonstrations require the acquisition of expensive kits for the science labs. The market has offers of kits for the different science disciplines but they aren't always flexible to allow using in multiple experiments.

The high price of the kits make it difficult for schools to have enough units that allow teachers to engage the entire class in the experiments simultaneously.

With this project was created a platform for capturing and treating data from multiple sensors connect to one or more Arduino boards.

The Arduino is an open-source prototyping platform that uses a microcontroller programmable in a language similar to C/C++. It includes digital and analogical pins that can be used to connect sensors and modules to interact with de physical world.

The need for an accessible way to make the laboratorial activities was one of the main motivations for this project, it have to be easy to use and the cost shouldn't be an obstacle for the acquisition of multiple kits by the schools.

The educational value of involving the students in the construction of the lab apparatus is something to consider. By allowing students to build the physical part of the platform, choosing and installing the sensors and teaching them how everything works the students are engaged in a higher level.

The architecture of the platform is composed of three levels:

- The lower level includes the Arduino and the sensors. It's at this level that a communication protocol was implemented to allow the execution of all the experimental activities.
- In the middle level resides the same protocol inside the applications. This were implemented in such way that they can be reused in other projects.
- The top level are the visible components of the platform, the apps. One desktop application for Windows that will be connecting to the Arduino, one or more simultaneously and one mobile app for Android. They are responsible for capturing the data, presenting them to the users and sharing them.

During the development of the project was executed a usability test with students. The Arduino platform was able to engage the students at a new level. They were curious at how it all worked and what where the components that were been used. The students even contributed to the lab apparatus whit great ease.

In this test two lab experiments were executed: Free Fall and Bouncing Ball. The application used to capture the data was the Windows version, although not finished. The users felt that the application was easy to use and the options were self-explanatory.

The data were presented in two forms: numerical and graphical. The results obtained were precise and within the margin error that were expected for a class room. The platform produced results on the laboratorial level but also at the pedagogical level.

The work doesn't end here. In the future, new functionalities, for other sciences like chemistry or biology, as new lab activities for physics can be added bringing new value to this project.

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABELAS

## GLOSSÁRIO

| | |
|---|---|
| **Activity** | Uma classe que implementa uma determinada função num programa para Android |
| **Android** | Sistema operativo para dispositivos móveis |
| **Baud rate** | Número de bits enviados por segundo |
| **Bluetooth** | Protocolo de comunicação sem fios para curta distância |
| **Breadboard** | Placa que permite fazer protótipos de dispositivos eletrónicos sem utilizar solda |
| **DLL – Dynamic Link Library** | Ficheiro que contém código que pode ser utilizado por vários programas em simultâneo. |
| **Firmware** | Programa residente em memória não volátil |
| **Framework** | Conjunto de objetos e métodos desenvolvidos por terceiros |
| **Full duplex** | Modo de transmissão bidirecional em simultâneo e simétrico |
| **IDE** | Ambiente de desenvolvimento de programas que integra todas as diferentes ferramentas necessárias à produção de aplicações |
| **Internet Of Things** | Tendência recente baseada na ligação de todos os dispositivos à Internet |
| **LDR** | *Light Dependent Resistor*, resistência cujo comportamento varia de acordo com a intensidade de luz |
| **LED** | *Light Emitting Diode*, díodo emissor de luz |
| **Pulse Width Modulation (PWM)** | Técnica que permite obter resultados analógicos em pinos digitais, através da ativação e desativação do pino a uma determinada frequência, que possibilita simular voltagens intermédias entre 0 e 5 volts. |
| **RAM** | Read Only Memory, memória volátil que é utilizada para armazenar código e dados de programas |
| **SI** | Sistema Internacional de Unidades, determina as unidades de medida a utilizar |
| **Thread** | Porção de código executado independentemente de outras partes da aplicação |
| **USART** | Componente físico de um sistema responsável pela comunicação síncrona ou assíncrona |
| **Buffer** | Memória intermédia, normalmente utilizada no envio e receção de dados |

# CAPÍTULO 1 – INTRODUÇÃO

Este capítulo faz o enquadramento do projeto, apresenta as motivações e uma visão geral do mesmo, incluindo um resumo dos resultados obtidos.

## 1.1 – CONTEXTO

No ensino das disciplinas de ciências experimentais, as atividades laboratoriais têm um papel importante. Estas possibilitam a participação ativa dos alunos no processo ensino-aprendizagem.

Nas atividades laboratoriais procura-se passar da teoria à prática. Para isso, não basta mostrar as experiências aos alunos. Sem uma verdadeira atividade laboratorial executada pelos discentes, os resultados escolares obtidos não são os mesmos das práticas laboratoriais em que os alunos colocam "as mãos na massa" (Oliveira et al., 2011).

As atividades laboratoriais completam a formação dos alunos, estimulando-os para a exploração de teorias ou hipóteses, a execução de experiências e recolha de dados, a validação dos

resultados e a dedução de conclusões que concordam ou não com as hipóteses inicialmente propostas.

São inúmeros os autores que defendem a utilização destas atividades por forma a facilitar a assimilação de conceitos abstratos (Fiolhais & Trindade, 2003),(Veit & Teodoro, n.d.). Estes preconizam a utilização de diferentes métodos de ensino, para além dos tradicionais, de modo a tornar o processo ensino-aprendizagem eficaz.

Os professores, em geral, estão motivados para o uso das tecnologias de informação e comunicação, personificadas pelo computador e pela Internet nas suas práticas letivas, mas procuram uma utilização que vá para além da simples facilitação do conhecimento, pretendem uma ferramenta que não se limite a fornecer informação (Veit & Teodoro, n.d.). Estes procuram no computador uma ferramenta de construção de conhecimento, que motive o aluno a pensar e a formar, por essa via, o seu conhecimento de modo mais eficaz do que quando se limita a ser um ouvinte passivo nas aulas.

No panorama atual das Escolas Portuguesas, com as restrições orçamentais em vigor, torna-se mais difícil dispor dos recursos necessários para a execução destas atividades.

As soluções existentes no mercado, de empresas como a Pasco, têm alguns aspetos menos positivos. O custo associado à sua aquisição é um desses aspetos. Esse custo limita o número de dispositivos disponíveis na sala de aula para estas atividades e consequentemente não permite que sejam os próprios alunos a executar as experiências, relegando-se, estes, a um papel menos participativo, assistindo a demonstrações executadas pelo docente, ao contrário do que é indicado pelos programas das disciplinas (Ferreira et al., 2014).

O segundo aspeto menos positivo destas soluções é a sua orientação, quase exclusiva, para uma só experiência. Esta limitação obriga a um investimento financeiro maior de modo a possibilitar a execução das diferentes atividades laboratoriais previstas nos currículos das disciplinas (Rocha & Marranghello, 2014).

Por fim, o facto de os dispositivos serem apresentados como uma solução fechada, que não permite que os alunos possam satisfazer a sua curiosidade quanto ao seu funcionamento ou à sua constituição, limitando, assim, o valor educativo (Laudares et al., 2014) associado ao conhecimento intrínseco dos componentes utilizados para a recolha dos dados (Rocha & Marranghello, 2014).

## 1.2 – MOTIVAÇÃO

É no contexto apresentado na secção anterior que nasce a ideia de utilizar a plataforma *Arduino* na execução de experiências da disciplina de Física e Química A do Ensino Secundário Português.

O *Arduino* (Figura 1) é uma plataforma eletrónica aberta que inclui um microcontrolador programável com recurso a uma linguagem de programação baseada em C/C++.


(Fonte: www.arduino.cc)

**Figura 1 - Arduino UNO R3.**

Já existem diversas propostas de utilização desta plataforma para o ensino da Física (Souza et al., 2011), (Rocha & Marranghello, 2014), (Zachariadou, et al., 2012), mas este projeto é um pouco diferente. Desenvolveu-se uma plataforma que permite a execução de várias experiências com diferentes sensores e diferentes aplicações.

Uma aplicação para computadores com sistema operativo *Windows* que permite capturar em tempo real dados de um ou mais dispositivos, permitindo, assim, centralizar os dados das experiências executadas pelos alunos, possibilitando a comparação dos resultados entre diferentes grupos de trabalho. Esta aplicação controla todo o processo laboratorial, inicia e termina a captura dos dados, permite ou não o envio para os dispositivos móveis, dando, assim, o controlo total ao docente sobre a execução dos trabalhos.

Uma aplicação para dispositivos móveis com sistema operativo Android que captura os dados através de uma ligação *Bluetooth*, possibilitando que cada grupo possa visualizar autonomamente os resultados das suas experiências nos seus próprios dispositivos.

O trabalho realizado vai para além das atividades específicas implementadas, disponibiliza uma ferramenta de utilização aberta e flexível que permitirá ao docente conceber outras experiências com outros sensores sem que para isso tenha de recorrer a outra aplicação ou à reprogramação do Arduino. Esta funcionalidade permite que o professor escolha quais os pinos a utilizar na captura dos dados e que, posteriormente, este faça o tratamento que desejar sobre esses dados, traçando gráficos ou executando cálculos com regressões ou inserindo as suas próprias fórmulas matemáticas.

## 1.3 – OBJETIVOS

Os objetivos aqui apresentados são enquadrados nas oportunidades identificadas aquando da pesquisa bibliográfica (ver Capítulo 2 – ).

Em termos genéricos podemos enunciar como objetivos deste projeto:

- Conceber uma plataforma de *software* e *hardware* flexível e de baixo custo para uso no ensino da disciplina de Física;
- Construir um kit, baseado na plataforma *Arduino*, pré-programado e equipado com diversos sensores para a execução de práticas laboratoriais de Física;
- Desenvolver uma aplicação desktop para Windows que permita a captura, apresentação e gestão de dados de um ou vários Arduinos, via USB;
- Conceber uma aplicação móvel para Android que permita a captura e apresentação de dados de um Arduino, via Bluetooth.

As aplicações têm objetivos específicos que a seguir se apresentam. Em relação à aplicação para o *Arduino*:

- Simplificar a utilização da plataforma *Arduino* para que o Docente não tenha de alterar o código instalado;
- Permitir a captura dos dados de vários sensores em simultâneo;
- Comunicar os valores dos sensores por USB e Bluetooth;
- Suportar sensores analógicos e digitais.

Quanto à aplicação móvel:

- Permitir a captura dos dados em dispositivos móveis Android;
- Implementar mecanismos de tratamento e partilha dos dados.

A aplicação *desktop*:

- Permitir que o professor controle num só computador diversas atividades laboratoriais a decorrer em simultâneo;
- Possibilitar a captura e comparação de dados em várias execuções da mesma experiência por forma a comprovar os resultados obtidos;
- Apresentar os resultados de modo a facilitar a sua leitura e utilização em relatórios (valores numéricos e gráficos de dados);
- Partilhar os dados obtidos com outras aplicações (exportação para formatos de partilha de dados);
- Guardar, de forma persistente, os resultados obtidos com referência à turma e à atividade laboratorial;
- Permitir a atribuição de comentários relativos à avaliação da atividade desenvolvida pelos alunos;
- Promover a integração da aplicação com as plataformas de e-learning;
- Implementar um modo de funcionamento simplificado com os sensores previstos;
- Permitir, para utilizadores mais avançados, um modo de funcionamento aberto com outros sensores para além dos previstos.

## 1.4 – RESUMO DOS RESULTADOS

Os resultados são fundamentais num projeto orientado para a ciência, a sua qualidade deve ser inequívoca e os dados de fácil interpretação e consistentes.

Durante a execução do trabalho realizaram-se testes de captura de dados que foram fundamentais na validação das aplicações e permitiram melhorar o resultado final.

Para uma primeira avaliação da aplicação Windows foi realizada uma aula prática com um grupo de alunos do décimo ano da Escola Secundária de Santa Comba Dão. Com esta atividade não se pretendeu tirar conclusões definitivas, mas antes recolher novos dados da observação e da troca de ideias com os utilizadores.

Os resultados obtidos nas diferentes atividades laboratoriais, quer através da aplicação Windows, quer através da aplicação Android, comprovam e concordam com as leis e os princípios da física em estudo.

Na atividade Queda Livre obtém-se um valor próximo do valor esperado para a força gravítica. Na atividade Bola Saltitona, é possível comparar resultados com bolas de diferentes materiais e verificar quais as que têm um coeficiente de elasticidade maior. Também na atividade do Plano

Inclinado se obtêm valores para a velocidade de descida e a energia cinética dentro dos expectáveis, ainda que nesta atividade o material tenha uma influência nos resultados finais, pois a qualidade do carrinho utilizado e o material do próprio plano inclinado podem contribuir com um valor apreciável de atrito.

De igual modo a atividade com o acelerómetro comprova o valor da força gravítica e das forças que interagem com o dispositivo.

## 1.5 – ESTRUTURA DA DISSERTAÇÃO

No primeiro capítulo foram expostos os objetivos deste projeto, bem como, a motivação e um resumo dos resultados obtidos.

O capítulo dois apresenta o resultado da pesquisa bibliográfica, relativamente ao estado da arte no âmbito do domínio deste projeto, realizada aquando da preparação do trabalho.

O terceiro capítulo descreve o processo de conceção e planeamento da plataforma, começando pelos conceitos de matemática e de física, fundamentais nesta empreitada. Entre eles o método científico, que orientou a estrutura das aplicações e o tratamento estatístico de grandes volumes de dados, por forma a obter tendências que permitem apurar parâmetros de equações. Foi com base nestes princípios que foi definida a arquitetura da plataforma.

O quarto capítulo aborda o desenvolvimento das aplicações. O programa do *Arduino*, a aplicação *Windows* e a aplicação Android. Neste capítulo são apresentadas as estruturas de dados utilizadas e as funcionalidades implementadas.

O capítulo cinco apresenta os resultados obtidos com a aplicação durante a fase de desenvolvimento e com um grupo de alunos em ambiente de sala de aula.

O último capítulo, o sexto, expõe as conclusões finais do trabalho realizado e possíveis desenvolvimentos futuros para este projeto.

# CAPÍTULO 2 – ESTADO DA ARTE

Neste capítulo apresenta-se uma resenha dos meios disponíveis para as práticas laboratoriais no âmbito do ensino da Física.

É sempre útil saber o que existe numa determinada área antes de avançar para o desenvolvimento de produtos que se podem mostrar obsoletos ou desnecessários e não ter espaço ou razão para existirem.

Este conhecimento prévio permite, também, que o trabalho possa ser uma evolução de outros já existentes.

## 2.1 – PROPOSTAS COMERCIAIS

### 2.1.1 – PASCO

A Pasco é uma empresa Americana com mais de 50 anos de experiência no desenvolvimento de soluções para a educação (Pasco, 2014). Está presente em mais de 100 países com produtos de qualidade e um catálogo completo de sensores para diversas disciplinas na área das ciências

experimentais. Os seus produtos são vendidos em Portugal através da J.Roma (J. Roma, Lda, n.d.).

No ensino da disciplina de Física, a Pasco dispõe de propostas que cobrem as mais diferentes áreas como sejam: ótica, mecânica, ondas e som, termodinâmica, eletromagnetismo, entre muitas outras.

Ao nível de *software* a Pasco fornece uma aplicação para desktop, para Windows e Mac, com suporte para os seus sensores que permite apresentar os resultados dos valores capturados.

A captura dos dados pode ser via USB ou via Bluetooth. Para interligar os sensores ao computador é necessário utilizar uma interface extra que deve ser adquirida em separado com os sensores. Está disponível uma interface para interligar via USB e outra para fazer a ligação sem fios, via Bluetooth.

As versões mais recentes das aplicações são bastante completas e permitem visualizar os dados das experiências de diferentes formas, com gráficos, com tabelas e com valores atuais em tempo real.

Dispõem de ferramentas de análise automática dos dados que apresentam valores totais, médias bem como valores mínimos e máximos e a respetiva amplitude de valores. A interface é personalizável permitindo ao utilizador dispor os elementos presentes nas posições que achar mais adequadas às suas preferências.

Incorporado nas aplicações existe ainda a possibilidade de guardar imagens do ecrã (vulgo *printscreens*) por forma a serem utilizadas pelos alunos para a construção dos relatórios das experiências, permitindo criar gráficos com valores introduzidos manualmente, para além dos valores capturados pelos sensores.

As aplicações podem guardar os dados obtidos num formato proprietário ou como página HTML com as imagens dos ecrãs geradas durante as experiências. Em termos didáticos, a aplicação dispõe de alguns questionários que os professores podem apresentar aos alunos enquanto executam as experiências.

Quanto à aplicação para dispositivos móveis, permite a captura dos dados através dos sensores do próprio dispositivo ou via Bluetooth dos sensores ligados à interface específica. Esta aplicação é gratuita e funciona em tablets Android e iPad.

Desta análise, podemos comprovar que a Pasco fornece bons produtos mas, ainda assim, continua a existir espaço para inovar. Por um lado, os sensores dependem de um segundo

dispositivo para comunicar e, por outro, a aplicação desktop só permite a captura de um dispositivo de cada vez.

### 2.1.2 – ALTAY

A Altay é uma empresa com mais de 60 anos, com representantes nos cinco continentes, que produz produtos para o ensino das ciências (Altay Scientific Group, n.d.). Os seus produtos são importados para Portugal através da Dismel (Dismel, distribuidor de material electrónico, lda, n.d.).

Para o ensino da Física propõe produtos na área da mecânica, termodinâmica, ótica, electroestática, magnetismo, eletricidade e eletrónica, radioatividade, entre outros.

Além de produtos de medida, como termómetros, cronómetros, multímetros fornece também kits para atividades laboratoriais.

Os seus sensores têm uma interface específica para um dispositivo de recolha dos dados. Os dispositivos podem ser ligados a diferentes sensores, como sensores de luz, sensores de temperatura e sensores de movimento.

O dispositivo de captura dos dados suporta o armazenamento destes num Flash Drive para posterior importação para aplicações como o Microsoft Excel.

A aplicação de captura de dados da Altay, denominada LoggerPro 3, permite guardar e apresentar dados organizados em tabelas ou sob a forma de gráficos. Está disponível para plataforma Windows e Mac OS X. Este programa permite a captura de dados de diferentes sensores. O aspeto menos positivo da aplicação é a sua interface que não está disponível em português.

### 2.1.3 – VERNIER

A Vernier é uma empresa Americana fundada em 1981 (Vernier, n.d.). A sua atividade comercial é baseada na produção de interfaces, sensores e software para uso no ensino das ciências.

O seu catálogo de produtos é semelhante ao das empresas já apresentadas neste trabalho. Um kit típico é composto por um sensor ligado a uma interface que, por sua vez, é ligado a um computador.

Os produtos são bastante completos e suportam uma grande variedade de experiências. O sítio da empresa inclui vídeos demonstrativos da sua utilização bem como manuais de utilizador com as especificações técnicas dos sensores. Existem interfaces para ligação via USB e interfaces para

ligação via Bluetooth que suportam o envio dos dados para dispositivos móveis proprietários e *tablets* ou *smartphones*.

A empresa propõe uma aplicação para a plataforma Android e outra para os dispositivos móveis Apple. O *software* proposto para a análise dos dados capturados é o mesmo da Altay, o LoggerPro 3. Além do equipamento e das aplicações, a empresa vende um manual de apoio com várias experiências que podem ser realizadas, utilizando os seus produtos.

São, em todo o caso, produtos caros. Os preços disponíveis no sítio da empresa (Verney, 2014) são de 203 dólares para a interface USB e de 108 dólares para o sensor de movimento, comparativamente os preços de um *Arduino* e de um sensor de movimento que não ultrapassa os 50 euros, isto sem contar com custos do importador, que é o mesmo da Altay.

A Tabela 1 apresenta uma análise comparativa das propostas comerciais.

**Tabela 1 - Análise comparativa das propostas comerciais**

| | Pasco | Altay | Vernier |
|---|---|---|---|
| Aplicação desktop | Sim | Sim | Sim |
| Aplicação móvel | Sim | Não | Sim |
| Aplicação em Português | Sim | Não | Não |
| Apresenta dados em gráficos | Sim | Sim | Sim |
| Permite exportação dos dados | Sim | Sim | Sim |
| Exportação para folhas de cálculo | Sim | Sim | Sim |
| Integração com plataformas e-learning | Não | Não | Não |
| Permite elaboração de relatórios | Sim | Sim | Sim |
| Captura em simultâneo de múltiplas experiências | Não | Não | Não |
| Permite comparação de dados a partir de repetição de experiências | Sim | Sim | Sim |
| Permite a atribuição de classificação dos resultados obtidos pelo docente | Não | Não | Não |
| Implica utilização de interface externa | Sim | Sim | Sim |
| Ligação a computador | Sim | Sim | Sim |
| Sensores em caixas fechadas | Sim | Sim | Sim |
| Dispositivos proprietários para captura de dados | Sim | Sim | Sim |

## 2.2 – PROPOSTAS NÃO COMERCIAIS

Do ponto de vista da utilização do *Arduino* para as práticas laboratoriais, existem vários autores que apresentam propostas para a implementação de experiências com sensores específicos, que a seguir se resumem.

### 2.2.1 – A LOW-COST COMPUTER-CONTROLLED ARDUINO-BASED EDUCATIONAL LABORATORY SYSTEM FOR TEACHING THE FUNDAMENTALS OF PHOTOVOLTAIC CELLS

Neste artigo os autores propõe a utilização do Arduino como um laboratório educacional para o estudo de células fotovoltaicas (Zachariadou et al., 2012), sendo o sistema proposto direcionado ao ensino universitário.

Dos vários objetivos propostos no artigo destacam-se o baixo custo do projeto comparativamente com soluções comerciais e permitir aos alunos aplicar o método científico de formulação de hipóteses, execução de experiências e registo dos resultados, para os poder analisar e interpretar com o propósito de deduzir conclusões.

Esta proposta suporta quatro atividades experimentais, todas relacionadas com células fotovoltaicas, e uma simulação.

Para a análise dos dados é proposta uma aplicação para sistema operativo Windows, desenvolvida em C#. Esta permite visualizar os resultados obtidos sob a forma de uma tabela e de um gráfico.

A arquitetura da solução inclui uma base de suporte para três placas com diversos componentes e ainda um Arduino UNO R3. Entre os componentes existe um painel fotovoltaico que será o objeto do estudo.

As atividades que são possíveis de realizar são descritas em grande detalhe, incluindo orientações quanto às observações mais importantes e as fórmulas matemáticas associadas às leis estudadas nas aulas. Para a apresentação gráfica dos dados é utilizada uma *framework open-source* desenvolvida no CERN denominada ROOT.

### 2.2.2 – ACELERÔMETRO ELETRÔNICO E A PLACA ARDUÍNO PARA ENSINO DE FÍSICA EM TEMPO REAL

A proposta deste artigo não se resume à utilização do *Arduino* com um acelerómetro para o estudo da cinemática (Rocha & Marranghello, 2014). Os autores do artigo defendem a utilização das atividades laboratoriais, mais do que para confirmar o que é aprendido, como um modo

diferente de aprender. Os proponentes deste trabalho apresentam dois argumentos contrários às soluções comerciais: o custo e a apresentação do tipo "caixa preta". Assim é defendido que os docentes criem as suas próprias soluções de acordo com as suas necessidades.

Após a apresentação do sensor escolhido para o caso em estudo é feita uma breve descrição do seu funcionamento. De seguida o artigo apresenta o Arduíno, a sua estrutura e os seus componentes.

Como o objetivo é demonstrar a utilização do Arduino no ensino, uma das principais secções apresenta duas aplicações da plataforma em experiências.

A primeira permite registar as variações de aceleração sofridas por um corpo num salto de *Bungee Jump*. Os responsáveis pelo artigo incluem uma explicação detalhada dos cálculos envolvidos na experiência, desde as expressões matemáticas que validam os resultados obtidos até um conjunto de razões que explicam os valores.

Na segunda demonstração, a plataforma é utilizada para estudar a variação da aceleração sofrida por um objeto preso a uma haste em diferentes pontos desta. A explicação desta experiência segue a mesma estrutura da primeira, com a apresentação dos valores registados, das fórmulas e dos conceitos de Física envolvidos.

Nas duas experiências, o código utilizado no *Arduino* é apresentado num anexo do artigo. Os dados são recolhidos no computador através de um *Add-In* para o Microsoft Excel ("PLX-DAQ - Parallax Inc," 2014).

### 2.2.3 – A PLACA ARDUINO: UMA OPÇÃO DE BAIXO CUSTO PARA EXPERIÊNCIAS DE FÍSICA ASSISTIDAS PELO PC

Este artigo, tal como os anteriores, defende a utilização do Arduino como uma solução viável de baixo custo para a aquisição de dados de atividades laboratoriais na disciplina de Física (Souza et al., 2011).

O artigo começa por apresentar uma introdução ao Arduino para, de seguida, descrever o objeto de estudo, neste caso, as oscilações amortecidas.

A arquitetura da solução proposta é descrita de modo a permitir a sua repetição incluindo os resultados obtidos quando os autores a realizaram. A proposta é composta por uma placa Arduino e um sensor de luminosidade (LDR - *Light Dependent Resistor*). Com base na mesma estrutura, os autores propõem a utilização de um termístor NTC (NTC – *Negative Temperature*

*Coeficient*) para comparar a variação da temperatura de duas superfícies com cores diferentes, quando iluminadas por uma lâmpada de 150 W.

O código utilizado no Arduino não é apresentado nem tão pouco a aplicação para captura e tratamento dos dados.

### 2.2.4 – FÍSICA COM ARDUINO PARA INICIANTES

Este artigo apresenta diferentes modos de utilizar o *Arduino* para aquisição de dados de atividades laboratoriais no ensino de Física (Cavalcante, Tavolaro, & Molisani, 2011).

Em termos gerais o artigo apresenta as razões que justificam a utilização do computador como ferramenta de ensino nas ciências experimentais, passando para a apresentação da plataforma *Arduino* e a proposta de utilização experimental.

A atividade laboratorial proposta pelos autores é o estudo dos tempos de carga e descarga de condensadores. Para enquadrar esse estudo é feita uma introdução do que é um condensador e das leis físicas associadas. O código fonte utilizado bem como os esquemas de ligação dos componentes são apresentados no artigo.

Em resumo as propostas não comerciais comungam das seguintes caraterísticas:

- Estão normalmente limitadas a um sensor;
- Implicam a programação do *Arduino;*
- Comunicação do *Arduino* com o computador via USB;

A Tabela 2 apresenta uma análise comparativa das propostas não comerciais analisadas.

**Tabela 2 - Análise comparativa das propostas não comerciais**

| | **2.2.1** | **2.2.2** | **2.2.3** | **2.2.4** |
|---|---|---|---|---|
| Número de atividades possíveis | 4 | 2 | 2 | 1 |
| Necessário programar o Arduino | Sim | Sim | Sim | Sim |
| Aplicação Windows | Sim | Não | Não | Não |
| Aplicação Android | Não | Não | Não | Não |
| Análise gráfica dos dados | Sim | Não | Não | Não |
| Necessário outro software | Não | Sim (MS Excel) | Não | Não |

# CAPÍTULO 3 – CONCEITOS TEÓRICOS E CONCEÇÃO DA PLATAFORMA

Este capítulo apresenta um conjunto de conceitos e etapas fundamentais no processo de desenho e planificação dos trabalhos que conduziram à produção do sistema proposto. A primeira secção expõe os princípios teóricos que determinaram a estrutura geral das aplicações. Também, antes de se iniciar o desenvolvimento das aplicações, foi necessário avaliar algumas *frameworks* por formar a escolher as mais adequadas. De seguida desenvolve-se a arquitetura do sistema, o modelo concetual do protocolo e os comandos implementados.

## 3.1 – PRINCÍPIOS TEÓRICOS

Para o desenvolvimento deste projeto foi necessário observar aulas práticas de Física e Química A. Esta observação inspirou algumas ideias importantes para o desenho das aplicações, não só

em termos da apresentação dos resultados como também ao nível da interface. A esse nível destaca-se, a título de exemplo, a opção de esconder os resultados calculados pelas aplicações permitindo, assim, aos discentes executar os cálculos manualmente para no final comprovarem a qualidade dos seus resultados com os calculados automaticamente pelas aplicações.

Um papel importante também foi desempenhado pelos docentes do grupo disciplinar de Física e Química, particularmente os docentes que forneceram conhecimentos fundamentais da disciplina através de pequenas entrevistas informais e que validaram os resultados e as atividades criadas, contribuindo com novas ideias e sugestões.

### 3.1.1 – MÉTODO CIENTÍFICO

A organização geral da aplicação é inspirada no método científico (Wilson, 1952): após a observação ou descrição de um fenómeno é formulada uma hipótese que deve ser testada.

O que se pretende numa aula prática de Física é aplicar o método científico para comprovar ou refutar os conceitos expostos nas aulas teóricas.

O professor orienta os alunos no estudo da hipótese para a qual é necessário realizar experiências que a comprovem ou invalidem (Seixas, 2005). As experiências devem permitir capturar dados reais que são trabalhados, por forma a validar os modelos teóricos.

Não se pretende com estas aplicações retirar aos alunos a hipótese de alcançar as suas próprias soluções. Para isso, as atividades têm de passar sempre por duas fases: a primeira, de captura dos dados, com a execução da experiência e, uma segunda, em que os discentes trabalham os dados recolhidos aplicando os seus conhecimentos. Esta última fase é fundamental, pois permite ao professor avaliar as competências dos seus alunos. Para isso, estes devem produzir um relatório com as conclusões da atividade. Estas duas fases foram fundamentais na organização da interface das aplicações, existindo dois separadores, em cada atividade, um que apresenta os dados capturados e outro que permite trabalhar estes dados.

Assim, as aplicações foram desenhadas com o princípio pedagógico de permitir aos alunos percorrer o seu próprio caminho, trabalhando os dados manualmente. No final da atividade, o docente pode utilizar as aplicações para verificar se os resultados a que os alunos chegaram são aproximados aos esperados, comparando com os resultados calculados pelos programas que são revelados, quando este assim entender.

Os discentes podem, também, ser chamados a participar na criação do aparato da experiência. O *Arduino* permite a utilização de componentes simples que existem em equipamentos vulgares e para os quais os alunos podem contribuir de duas formas: reciclando brinquedos ou

dispositivos danificados e utilizando os conhecimentos de eletricidade e eletrónica na ligação dos sensores ao *Arduino*, tornando a experiência mais enriquecedora.

A ferramenta matemática mais comumente utilizada nas atividades laboratoriais de física é a análise visual dos dados através da representação gráfica das séries de dados capturados, selecionando pontos de interesse nesses dados e realizando uma análise dos valores, com o objetivo de definir um modelo matemático.

As aplicações desenvolvidas incluem algoritmos de análise automática dos dados que permitem obter os pontos de interesse e os resultados sem esforço por parte do utilizador. Por exemplo na atividade de queda livre o algoritmo consegue detetar a passagem das linhas que obstruem o sensor e a partir destes calcular a aceleração do objeto.

Estas funcionalidades têm limitações associadas à qualidade dos dados obtidos. No caso da atividade queda livre a iluminação ambiente pode impedir a obtenção inequívoca da passagem das linhas. Outras limitações advêm dos próprios sensores, por exemplo o sensor de distância por ultrassons limita a distância mínima e máxima a que os objetos são detetados.

O método de execução de uma atividade laboratorial pode-se resumir em algumas etapas:

- Preparação do equipamento a utilizar, para produzir e medir os resultados;
- Executar a atividade recolhendo os dados experimentais;
- Analisar os valores recolhidos;
- Avaliar a credibilidade dos resultados obtidos comparando-os com os previstos pelo modelo teórico;
- Elaborar um relatório com as conclusões a retirar da atividade realizada.

As aplicações desenvolvidas para este projeto suportam estas etapas. Por um lado, na captura dos dados quando a atividade é executada, por outro lado, fornecem ferramentas de análise dos resultados obtidos, através de gráficos e equações.

Para a análise dos dados foram desenvolvidas ferramentas específicas que permitem deteção automática de picos em séries de valores facilitando a manipulação de listas extensas de valores.

Para uma utilização pedagógica, os dados fornecidos pela aplicação podem ser manipulados pelos discentes que devem ser incentivados a elaborar, a partir deles, os seus próprios resultados podendo, posteriormente, comparar as suas soluções com os resultados produzidos pela aplicação.

Todos os dados capturados ou produzidos, para além de respeitarem o sistema internacional de unidades, são sempre apresentados com a respetiva unidade de medida (mais detalhes na secção 3.1.4 – Sistema de Internacional de Unidades).

### 3.1.2 – REGRESSÃO LINEAR SIMPLES

A Ciência procura propor modelos matemáticos que representem a realidade (Veit & Teodoro, n.d.). Esses modelos devem ser produzidos a partir da observação dos fenómenos naturais, da recolha de dados e posterior análise destes. Os modelos matemáticos não são mais do que fórmulas, equações, que explicam os fenómenos naturais, através da linguagem matemática.

Para a definição destas fórmulas, e dos seus componentes, a análise dos dados através de gráficos é uma das ferramentas mais importantes, ao permitir o tratamento dos dados e a definição de tendências de comportamentos. A partir dos dados utilizados para gerar o gráfico, também conhecido por diagrama de dispersão, procede-se ao cálculo da regressão dos dados. A análise de regressão permite determinar os valores constantes da equação.

Dependendo do tipo de equação podemos ter de utilizar um modelo de regressão linear ou um de regressão não linear. No modelo linear, quando existe uma variável dependente e uma variável independente trata-se de um modelo linear simples. Quando existem várias variáveis independentes temos um modelo de regressão linear múltipla.

Para a equação da velocidade (1) utiliza-se uma regressão linear simples que permite determinar o valor de uma variável (v - velocidade) cujo comportamento é linear em função do valor de outra variável (t - tempo) e de duas constantes ($a$-aceleração e $v_0$– velocidade inicial) (James, et al., 2013).

$$v = at + v_0 \qquad (1)$$

Para a equação da velocidade a variável t, dita variável independente, representa o tempo, a velocidade do objeto, representada pela variável v, é a variável dependente.

No cálculo de uma regressão linear é necessário obter as duas constantes, ou parâmetros, que definem o declive da reta (a) e o ponto de interseção da reta com o eixo dos y ($v_0$).

O método utilizado para estimar os parâmetros foi o método dos mínimos quadrados (James et al., 2013). Neste método o declive da reta é obtido através da equação do declive da reta (2),

enquanto a interseção resulta da equação que determina o ponto de interseção da reta com o eixo dos y (3).

$$a = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} \quad (2)$$

$$v_0 = \bar{y} - a\bar{x} \quad (3)$$

Nestas fórmulas $\bar{x}$ e $\bar{y}$ representam a média das séries de valores em cada eixo do gráfico de dispersão.

Após o cálculo destes dois parâmetros é fundamental determinar a qualidade da solução. Há que mostrar que a equação produz melhores resultados do que uma simples média dos valores existentes de y.

A equação é avaliada pela correlação entre os resultados obtidos pela experiência e os resultados da aplicação da equação. Para isso é necessário utilizar a equação que apura o coeficiente de correlação (4).

$$r^2 = 1 - \frac{SSE}{SST} \quad (4)$$

Para o cálculo desta equação é necessário determinar três valores:

- SSE, soma dos quadrados dos resíduos, corresponde à variação dos resultados existentes e os obtidos pela equação (5);

$$SSE = \sum_{i=0}^{n} (y_i - \bar{y})^2 \quad (5)$$

- SSR, soma dos quadrados da regressão, corresponde à diferença dos resultados obtidos pela equação e a média dos valores originais de y (6);

$$SSR = \sum_{i=0}^{n} (\hat{y}_i - \bar{y})^2 \quad (6)$$

- SST, soma dos quadrados totais, ou seja, soma de SSE com SSR.

O valor do coeficiente de correlação permite determinar o grau de proximidade entre os valores obtidos pela equação da reta e os valores reais. Um valor próximo de um (1) indica que o coeficiente de correlação é ótimo, permitindo prever o comportamento do sistema com um elevado grau de confiança a partir da equação da reta.

### 3.1.3 – REGRESSÃO NÃO LINEAR

Tal como referido, quando o estudo incidir numa equação quadrática é necessário utilizar outro método de regressão, esse é caso do cálculo da aceleração de um objeto em queda livre, permitindo validar o princípio da força gravítica.

A fórmula a utilizar é a da equação do movimento (Caldeira & Bello, 2008). Neste modelo, a posição de um objeto é dada pela aceleração (a) ao longo do tempo (t), adicionando a posição inicial ($y_0$) e o produto do tempo pela velocidade inicial ($v_0$).

$$\mathrm{y} = \frac{1}{2}at^2 + v_0 t + \ y_0 \qquad (7)$$

Esta fórmula aplica-se aos objetos uniformemente acelerados, ou seja, a aceleração dos objetos é constante ao longo do tempo (Caldeira & Bello, 2008).

A regressão linear não pode ser aplicada com funções quadráticas, exponencial, logarítmicas e todas outras que produzam um gráfico cuja curva de ajuste não é uma reta.

Neste caso, temos um polinómio de grau superior a um e a determinação dos parâmetros é feita calculando as $n + 1$ derivadas parciais (Monteiro, 2013).

Para a resolução deste problema foi utilizada uma *framework* denominada CurveFunctions (Stephens, 2014) que foi implementa em C#. Procedeu-se à sua conversão para Java, sendo utilizada na aplicação Android.

### 3.1.4 – SISTEMA DE INTERNACIONAL DE UNIDADES

As unidades de medida são fundamentais para uma correta interpretação dos resultados. Portugal adotou o Sistema Internacional de Unidades (SI).

O SI foi definido pelo International Bureau of Weights and Measures (BIPM) criado em 1875 pela Metre Convention, assinada em Paris (BIPM, 2008). Fazem parte desta convenção 51 países, entre os quais Portugal.

Esta convenção determina, não só as unidades de medida a utilizar, mas também um conjunto de equações que permitem calcular a relação entre as diferentes escalas.

Neste trabalho são utilizadas as seguintes unidades e os respetivos símbolos:

- Metro como unidade de comprimento, símbolo m;
- Quilograma como unidade de massa, símbolo kg;
- Segundo como unidade de tempo, símbolo s;
- Microssegundo como unidade de tempo, símbolo µs (1 µs = 0,000001 s);
- Joule como unidade de energia, símbolo J;
- Metro por segundo como unidade de velocidade, símbolo m/s;
- Hertz como unidade da frequência de repetição de um fenómeno, símbolo Hz;
- Kelvin e Celsius para temperaturas, cujos símbolos são K e °C, respetivamente.

## 3.2 – *FRAMEWORKS*

Como é natural em qualquer projeto de desenvolvimento de *software*, também neste trabalho foi necessário recorrer a código criado por outros. Esta secção apresenta as *frameworks* adotadas bem como as que foram testadas e descartadas.

### 3.2.1 – NO PROGRAMA DO ARDUINO

**SoftwareSerial**

Desde o início do projeto foi definido o objetivo de interligar o Arduino ao computador via USB e ao Android, via Bluetooth. Para isso, o Arduino necessita de duas portas de comunicação. A versão UNO do Arduino só tem uma. Assim, foi necessário procurar um modo de contornar esta limitação, sendo que a utilização de outra versão do Arduino não era uma solução possível, por esta ser a versão mais popular e uma das mais acessíveis.

Não existindo uma segunda porta de comunicação física, a solução passou por implementar uma porta de comunicação lógica, por *software*. Assim foram testadas duas soluções disponíveis: a SoftwareSerial (Arduino SA, 2015d) e a AltSoftSerial. A livraria de funções SoftwareSerial tem uma limitação que impediu a sua utilização que se prende com o facto de não permitir

comunicação *full duplex*, ou seja, quando está a enviar dados não consegue receber. Neste projeto essa possibilidade é uma necessidade.

**AltSoftSerial**

Esta biblioteca de funções implementa, tal como a SoftwareSerial, uma porta série por *software*, para suprimir a falta de uma segunda porta série no *hardware* do *Arduino*. Esta *framework* permite a comunicação em simultâneo, ao contrário da SoftwareSerial, condição que era fundamental neste projeto.

Os aspetos menos positivos desta biblioteca de funções são: inutilizar a função de PWM – Pulse Width Modulation- (Stoffregen, 2015) no pino digital número 10, não permitindo a sua utilização para controlo de LEDs ou motores, e estar limitada a uma velocidade de 57.600 bps (bits por segundo). Estas limitações, ainda assim, não são impeditivas da sua utilização neste projeto, pois o Arduino continua a ter outros pinos digitais com função PWM disponíveis e a velocidade máxima imposta só é um problema para a atividade da Queda Livre.

Foi esta a framework escolhida para implementar a comunicação em simultâneo neste projeto.

**NewPing**

Criada por Tim Eckel, esta biblioteca de funções permite utilizar diferentes tipos de sensores de distância, ao contrário das funções disponíveis por defeito com o Arduino (Eckel, 2012).

Sendo a flexibilidade de escolha de diferentes sensores a principal vantagem da sua utilização, não é de desprezar os melhores resultados obtidos com a sua utilização por oposição, com as funções padrão de ultrassons do Arduino. Os testes realizados com um sensor de ultrassons, modelo HC-SR04, mostraram a obtenção de dados com menos ruído com a *framework* NewPing, sendo, por isso, escolhida para ser utilizada neste trabalho.

### 3.2.2 – NA APLICAÇÃO PARA O WINDOWS

**Microsoft Chart Extension**

O controlo Chart da Microsoft é utilizado para criar representações gráficas dos dados. No entanto, neste projeto, existe a necessidade de, para além de apresentar os dados, permitir a sua manipulação. Nesse aspeto este controlo é muito limitado. Para contornar estas limitações, foi adicionada de uma extensão a este controlo.

A MS Chart Extension (Code Artist, 2012) adiciona as seguintes funcionalidades:

- Selecionar quais as séries de valores a visualizar em simultâneo;
- Selecionar um ponto do gráfico e visualizar as coordenadas correspondentes;

- Fazer *zoom* sobre parte do gráfico;
- Deslocar horizontalmente o gráfico, para a esquerda e para a direita.

Esta extensão foi disponibilizada com código fonte e tendo por base a licença MIT (Open Source Initiative, 1998) que permite, para além do seu uso, a sua alteração. Assim foram implementadas as seguintes alterações:

- Tradução para português das opções;
- Uma vez que a interface se resumia à utilização do botão direito do rato sobre o gráfico foram adicionados botões que permitem a seleção da funcionalidade a utilizar de modo mais intuitivo;
- Implementação de uma nova funcionalidade que permite calcular a distância entre dois pontos no eixo dos x. Esta opção permite, por exemplo, calcular quanto tempo passou entre dois eventos selecionados a partir do gráfico dos dados.

Em todos os gráficos de dados tratados são colocadas marcas que assinalam com maior precisão a posição dos valores. Associada a cada ponto existe uma pequena mensagem que mostra os valores correspondentes ao ponto. A mensagem só aparece quando a seta do rato é deixada parada em cima de um dos pontos.

**CurveFunctions**

Esta *framework* foi escolhida para este projeto pois permite executar uma regressão não linear, com base no método dos quadrados mínimos, sobre uma lista de pares de valores. Da pesquisa realizada não se encontrou outra opção disponível nas mesmas condições de utilização.

Trata-se de uma implementação para Windows em C# que, para este projeto, teve de ser convertida para Java para utilização na aplicação Android.

A mesma *framework* foi editada para suportar regressões lineares, igualmente necessárias para o presente trabalho. A implementação das regressões lineares seguiu os princípios enunciados na secção 3.1.2 – Regressão Linear Simples.

Para comprovar a qualidade dos resultados das funções de regressão, linear e não linear, foram executados testes com os mesmos dados nas aplicações deste projeto e no Microsoft Excel. Os resultados foram exatamente iguais, tanto nos valores calculados para os parâmetros como para o valor da taxa de correlação (ver Figura 2 e Figura 3).

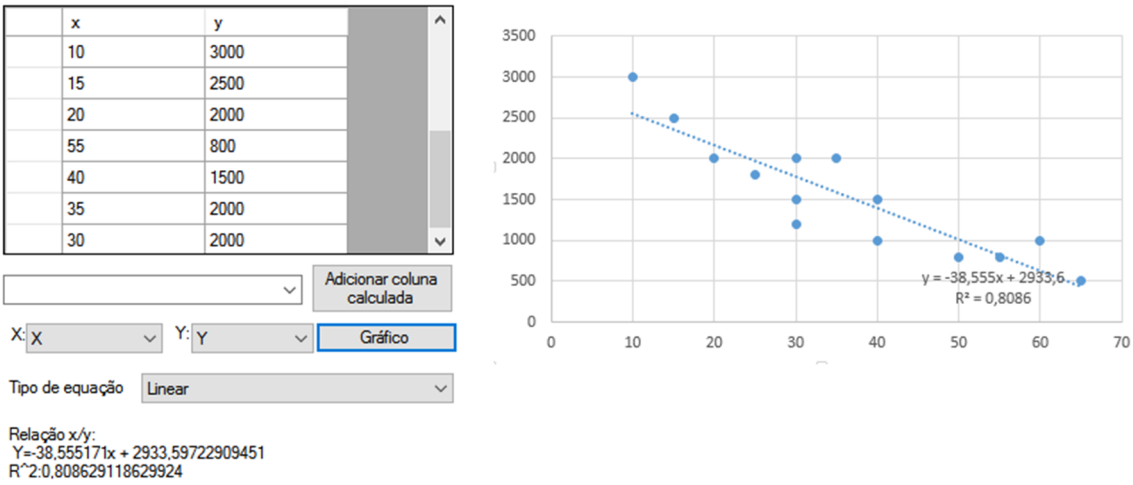



Figura 2 - Teste de validação de resultados de regressão linear.

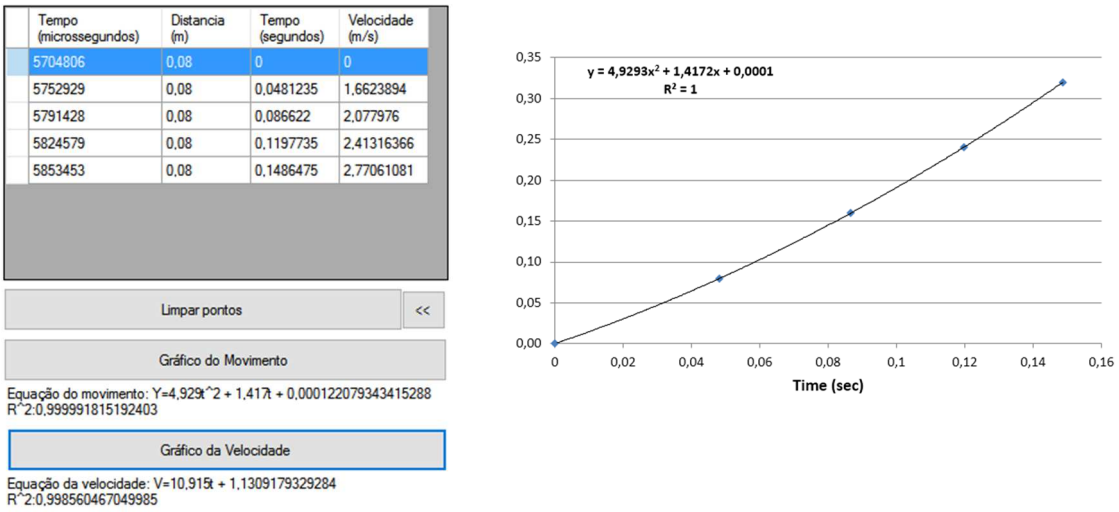



Figura 3 - Teste de validação de resultados de regressão não linear.

Os resultados obtidos nas duas aplicações, Windows e Android, também foram comparados para garantir que não existissem diferenças nas soluções produzidas. O que, a existir, implicava a existência de erros no código das aplicações.

### 3.2.3 – NA APLICAÇÃO ANDROID

**Android Plot**

Tal como na aplicação Windows foi necessário procurar uma solução para gerar os gráficos.

Android Plot foi a *framework* selecionada para esse fim, pois permite gerar os diferentes tipos de gráficos necessários para este trabalho. Apesar disso, foi necessário desenvolver algumas funcionalidades que não existiam, entre elas:

- Possibilidade de fazer *zoom* sobre um ponto do gráfico;
- Fazer deslizar o gráfico horizontalmente, para a direita ou para a esquerda;
- Selecionar um ponto do gráfico a adicionar a uma lista de pontos a analisar matematicamente.

**Android File Explore**

No sistema Android não existe nenhuma classe que permita a pesquisa de um documento no sistema de ficheiros do dispositivo.

Ao contrário do .NET, que disponibiliza uma classe que permite ao utilizador percorrer os dispositivos de armazenamento, as respetivas pastas e escolher um ficheiro, o sistema operativo Android não inclui nenhuma classe que implemente essa funcionalidade e que possa ser utilizada na produção de aplicações.

A *framework* Android File Explore é disponibilizada em inglês. Assim foi necessário proceder à sua tradução para português e também foi preciso alterar os dados que eram devolvidos, pois, por omissão, o criador da framework não definiu qual a informação que era devolvida, quando a *Activity* de seleção do ficheiro terminasse (Burman, 2011).

Esta framework foi a escolhida para utilizar neste projeto, devido à simplicidade de utilização.

## 3.3 – ARQUITETURA DO SISTEMA

A arquitetura do sistema foi pensada para ser modular e reutilizável. Para isso o sistema foi dividido em três níveis que interagem para produzir os resultados desejados (Figura 4).

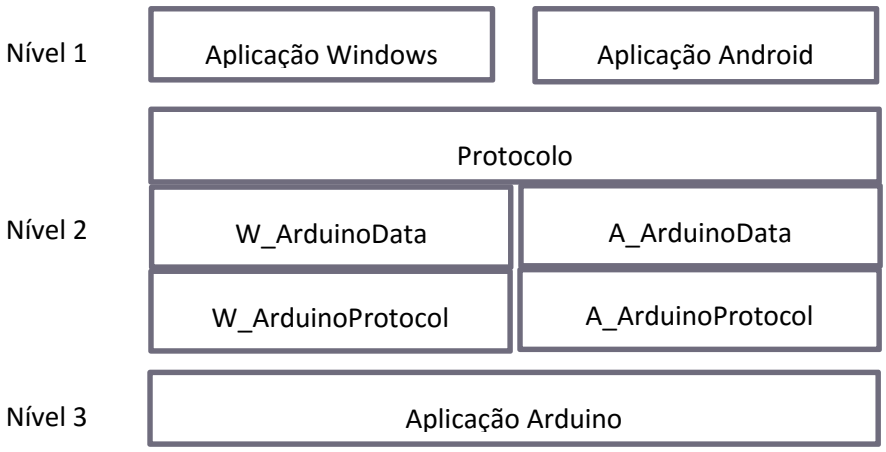



**Figura 4 - Arquitetura da plataforma.**

O primeiro nível, o mais visível, são as aplicações que os utilizadores vão utilizar. Estas têm previstas atividades laboratoriais específicas com interfaces orientadas às suas necessidades. Para além disso, existe uma interface de uso genérico que pode ser utilizada para a captura, tratamento e análise de dados por utilizadores mais avançados (o P*layground*).

Em geral as aplicações permitem a execução das tarefas presentes em qualquer atividade laboratorial: captura dos dados, tratamento e análise dos dados na forma de valores numéricos e de gráficos, cálculo dos parâmetros de equações de leis da Física e guardar os dados e os gráficos.

É este o princípio de todas as atividades laboratoriais. Primeiro executar a experiência e capturar dados da mesma para, de seguida, analisar os mesmos, por forma a comprovar a teoria modelada numa ou em mais fórmulas matemáticas.

Assim, as aplicações incluem ferramentas de análise de gráficos de resultados que permitem a seleção de pontos diretamente no gráfico bem como o cálculo de regressões de equações de primeiro e segundo grau, para calcular os coeficientes que melhor se adequam ao gráfico produzido.

Todas as atividades laboratoriais incluídas na plataforma permitem guardar os dados obtidos na experimentação e o respetivo gráfico produzido em formatos abertos que podem facilmente ser importados para outras aplicações, de modo a que possam ser utilizados na produção de relatórios pelos alunos. Os dados guardados podem, mais tarde, ser lidos e reutilizados para a continuação dos trabalhos ou para reanálise e eventual correção de erros.

Na aplicação para Windows foram implementadas as atividades de Queda Livre, Bola Saltitona e Plano Inclinado. Todas estas atividades estão previstas no programa da disciplina de Física e Química para o 10º e 11º ano de escolaridade.

Para além das atividades específicas existe uma atividade de utilização aberta designada de Playground. Esta funcionalidade permite que o utilizador crie as suas próprias atividades laboratoriais utilizando os seus sensores analógicos, executando a captura dos dados e procedendo à análise dos resultados com recurso a fórmulas que o próprio pode inserir ou analisar os resultados de forma visual com uma ferramenta gráfica.

A aplicação Windows detém o controlo total do processo de experimentação laboratorial, é esta que inicia e termina a captura dos dados no Arduino, bem como o envio dos dados do Arduino para o dispositivo Android.

Na aplicação para Android foram implementadas as atividades Queda Livre, Bola Saltitona, Plano Inclinado e ainda uma atividade de captura de dados a partir do acelerómetro. Esta aplicação foi desenhada para ser de utilização pelos alunos sem que se percam os objetivos educativos deste projeto, assim, alguns dos resultados obtidos na aplicação Windows não são apresentados na aplicação Android para que sejam os próprios alunos a tratar os dados manualmente valorizando e comprovando as suas aprendizagens.

O protocolo é o segundo nível da arquitetura e dele dependem os comandos enviados das aplicações para o *Arduino* e as respostas dadas por este em retorno a esses pedidos. O princípio deste protocolo é um conjunto de mensagens padrão que representam as solicitações das aplicações e as respostas devolvidas.

Para a implementação deste protocolo foi estudado um protocolo desenvolvido para interligar dispositivos musicais e computadores, o MIDI.

Este segundo nível é materializado nas aplicações através de uma classe que implementa as funcionalidades do protocolo e que pode ser reutilizada para outros projetos que envolvam a utilização do *Arduino*, como por exemplo em trabalhos no setor hoje denominado de *Internet Of Things* (IoT). Esta classe foi destacada do projeto ao ser compilada para uma DLL independente da aplicação, facilitando, assim, a sua reutilização em outros projetos.

O nível do protocolo implementa todo o conjunto de mensagens definidas para a comunicação entre os dispositivos físicos da plataforma. O mesmo conjunto de mensagens foi implementado para Windows e para Android, permitindo a comunicação destes com o Arduino.

Este nível inclui, ainda, uma classe dedicada à gestão dos dados recebidos. Esta classe implementa as funções de armazenamento dos dados na memória dos equipamentos, materialização dos dados em ficheiros e métodos de tratamento estatístico, como a execução de cálculos matemáticos como a média e a moda, entre outros.

Uma vez que o projeto inclui duas aplicações de plataformas diferentes foi necessário implementar duas versões das classes ArduinoProtocol e ArduinoData. Uma versão para Windows designadas de W_ArduinoProtocol e W_ArduinoData e outra para Android com as designações de A_ArduinoProcotol e A_ArduinoData. Estas são conceptualmente iguais, tendo o mesmo comportamento e produzindo os mesmos resultados, mas tecnicamente foram desenvolvidas com os recursos disponíveis em cada um dos sistemas operativos (mais detalhes sobre estas classes no Capítulo 4 – Implementação).

No terceiro nível do sistema está o *firmware* desenvolvido para residir no *Arduino*. Este representa o sistema operativo do microcontrolador no sentido que faz a gestão das mensagens que são recebidas, abrindo e fechando pinos analógicos e digitais, efetuando leituras e gerindo sensores e, ainda, enviando respostas para os dispositivos com os quais está ligado.

O Arduino é uma parte fundamental da plataforma. É a este microcontrolador que estão ligados os sensores que vão ler os fenómenos físicos em estudo, bem como o computador e os dispositivos móveis para os quais estes dados são enviados.

Como a versão em uso só contempla 2 quilobytes de memória RAM para dados, não é possível capturar e guardar os valores dos sensores para enviar mais tarde. Assim, sempre que se procede à captura de dados, estes são enviados para o computador ou dispositivo móvel tão rápido quanto possível.

Este ciclo de leitura e envio é crucial pois da sua velocidade depende a viabilidade de experiências com dados em mutação rápida, como o estudo de objetos em queda.

A próxima seção é dedicada aos sensores e aos módulos utilizados no Arduino.

## 3.4 – SENSORES E COMPONENTES ELETRÓNICOS

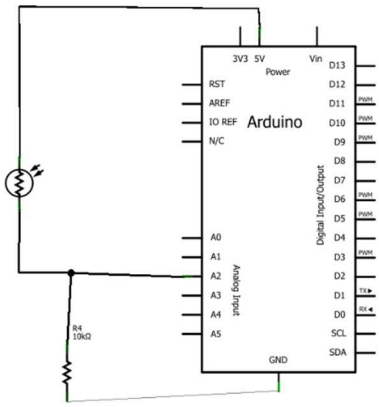
Parte fundamental neste projeto, os sensores, são a fonte dos dados que permitem estudar os fenómenos.

### 3.4.1 – LDR

Este sensor permite fazer variar a corrente elétrica em função da intensidade da luz que lhe incide. É um componente bastante barato e fiável, não estando propenso a avarias. É de fácil utilização, pois não tem polarização.

Neste projeto é utilizado para detetar a passagem de objetos em atividades como a Queda Livre ou o Plano Inclinado.

A utilização deste componente implica a utilização de uma resistência de 10K Ohm, que tem um custo igualmente baixo. A Figura 5 mostra o circuito de como ligar um LDR a uma porta analógica do *Arduino*.



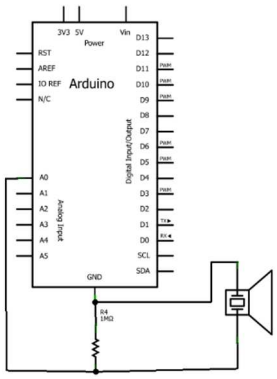

(Imagem gerada com Fritzing)

**Figura 5 - Utilização de um LDR.**

### 3.4.2 – PIEZO

O Piezo é um componente eletrónico que pode ser utilizado para emitir e detetar tons. A página web do Arduino refere: “A piezo is an electronic device that generates a voltage when it's physically deformed by a vibration, sound wave, or mechanical strain. Similarly, when you put a voltage across a piezo, it vibrates and creates a tone. Piezos can be used both to play tones and to detect tones.” (Arduino SA, n.d.-a).

No esquema de ligação do piezo é possível verificar que é necessário utilizar uma resistência 1 M Ohm (Figura 6). Trata-se de um componente polarizado por isso deve-se respeitar os polos na ligação ao Arduino.



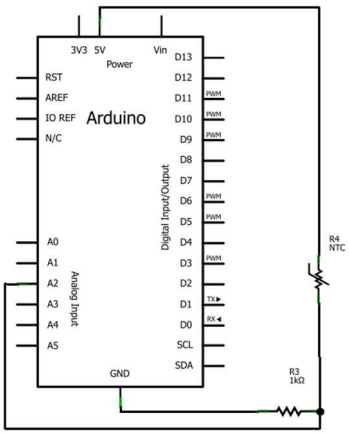

(Imagem gerada com Fritzing)

**Figura 6 - Utilização de um Piezo.**

Este componente é utilizado para detetar o impacto provocado pela queda de objetos, como por exemplo, uma bola.

### 3.4.3 – THERMISTOR

Um thermistor é uma resistência variável em função da temperatura. É um componente de baixo custo e utilização simples, como se pode ver pelo esquema da Figura 7.



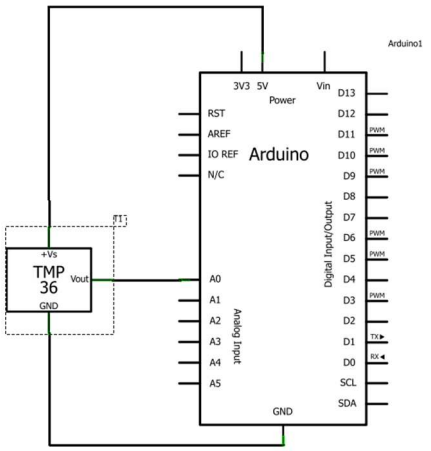

(Imagem gerada com Fritzing)

**Figura 7 - Utilização de um thermistor.**

Este componente pode ser utilizado para atividades laboratoriais que envolvam o estudo do comportamento térmico de diferentes materiais.

### 3.4.4 – TMP36

O TMP36 é um sensor de temperatura analógico. Existem ainda mais duas versões deste sensor o TMP35 e o TMP37. Basicamente, este componente gera um sinal analógico proporcional à temperatura em Celsius (Analog Devices, 2008).

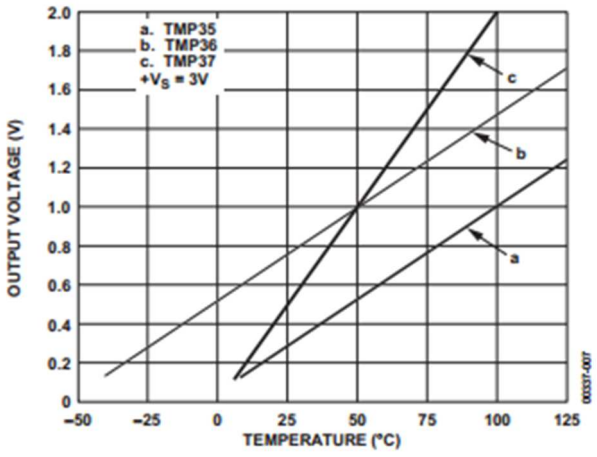



(Imagem gerada com Fritzing)

**Figura 8 - Utilização de um TMP36.**

O sensor TMP36 não carece de calibração. Pode ser utilizado nas mais variadas condições ambientais, tendo uma precisão de até ±2°C em temperaturas de -40°C até +125°C.

A voltagem de saída varia numa escala de 10 mV/°C. Sendo um sensor analógico os valores lidos estão compreendidos entre 0 e 1023, graças ao conversor analógico-digital, de 10 bits, que o Arduino tem associado a cada porta analógica. A conversão deste valor para volts realiza-se aplicando uma regra de proporcionalidade direta, uma vez que o seu comportamento é, como se pode ver pela Figura 9, linear. A figura apresenta o comportamento de outros sensores de temperatura cujo comportamento, sendo linear, é diferente do TMP36.




(Fonte: *Datasheet* do produto)

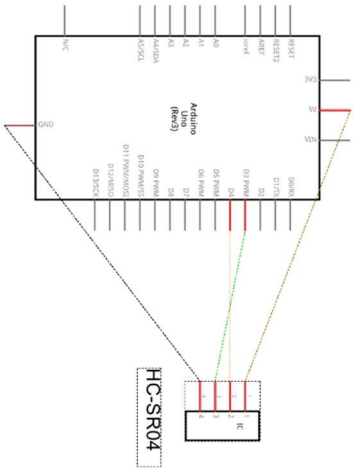
**Figura 9 – Voltagem de saída em função da temperatura.**

### 3.4.5 – HC-SR04

Este sensor digital utiliza ultrassons para determinar a distância a objetos, calculando a distância pelo tempo que o eco do ultrassom emitido demora a ser detetado (Cytron Tecnologies, 2013).

Este componente pode ser utilizado em diferentes atividades laboratoriais como a bola saltitona (ver 5.2 – Atividade Laboratorial Bola Saltitona). O manual de utilizador indica que pode ser utilizado para detetar objetos a distâncias entre 2cm a 400cm, com uma resolução de 0,3cm e um angulo de amplitude de 30º.

O HC-SR04 tem quatro pinos, dois são a alimentação do sensor, os outros dois servem para provocar a emissão do ultrassom e acusar a recessão do eco.



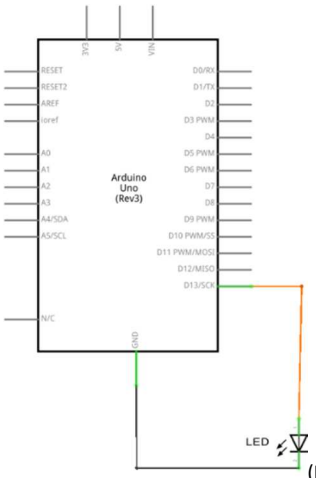

(Imagem gerada com Fritzing)

**Figura 10 – Esquema de ligação do HC-SR04 ao Arduino.**

A ligação ao Arduino é feita através dos pinos digitais. Para o protocolo criado deve-se ligar o pino Trigger do sensor ao pino 3 do Arduino e o pino 4 do Arduino ao pino Echo do sensor (ver Figura 10).

### 3.4.6 – LED

Light-Emitting Diode, ou díodo emissor de luz tem uma presença universal em equipamentos de eletrónica. É um componente polarizado, isto é, ao instalar deve-se respeitar a posição dos pinos, um positivo, o maior, e outro negativo, o mais pequeno.




(Imagem gerada com Fritzing)

**Figura 11 – Esquema de ligação de um LED.**

Como se pode ver na Figura 11, neste projeto, o LED ligado ao pino digital 13 é utilizado para mostrar o estado do Arduino, ligado ou desligado da aplicação Windows. Ao selecionar um Arduino este LED liga e ao fechar a atividade o LED desliga. Isto permite também ao docente saber qual o Arduino que está a controlar.

Os LED são utilizados para que seja possível utilizar o LDR independentemente das condições de iluminação ambiente ao fazer incidir a luz emitida no LDR.

### 3.4.7 – MÓDULO BLUETOOTH HC-06

Bluetooth é uma tecnologia de comunicação sem fios cujo objetivo é substituir os cabos de comunicação que interligam os equipamentos. É uma tecnologia praticamente universal que permite a ligação entre inúmeros dispositivos.

A comunicação é iniciada com o emparelhamento de dois dispositivos, permitindo a transmissão de dados entre os equipamentos em curtas distâncias, no máximo até 10 metros variando com as especificações dos equipamentos em uso (Bluetooth SIG, n.d.). A especificação da norma define três classes de dispositivos:

- Classe 3 – distância de transmissão de até 1 metro;
- Classe 2 – distância de transmissão de até 10 metros;
- Classe 1 – distância de transmissão de até 100 metros.

Para que o envio de dados do *Arduino* para o dispositivo Android se processe via comunicação Bluetooth é necessário um módulo de comunicação, como o HC-06, que desempenha o papel de emissor dos dados.

O módulo HC-06 permite comunicação até 20 metros, utiliza o protocolo Bluetooth 2.0 necessitando de 3.3V de alimentação (Lee, 2013). Por omissão está configurado para uma taxa de transmissão de 9600 bps (bits por segundo) e para o emparelhamento é necessário inserir o código 1234.

A ligação ao Arduino é efetuada com 4 pinos, dois para alimentação, um para enviar e outro para receber dados. Para além dos pinos, o módulo inclui um led que indica o estado da ligação.

Neste projeto os pinos de transmissão devem ser ligados assim:

- Pino DO (envio) do HC-06 ao pino 8 do Arduino (receção);
- Pino DI (receção) do HC-06 ao pino 9 do Arduino (envio).

A razão da ligação se efetuar nestes pinos está associada à *framework* de comunicação utilizada no Arduino.

A configuração do módulo pode ser alterada utilizando um conjunto de comandos envidados através do Arduino (AT Command). Com estes comandos é possível alterar o nome dos dispositivo, alterar o código de emparelhamento e a velocidade de comunicação.

A Figura 12 ilustra como se podem ligar os diferentes sensores e módulos ao Arduino utilizando uma *breadboard*.

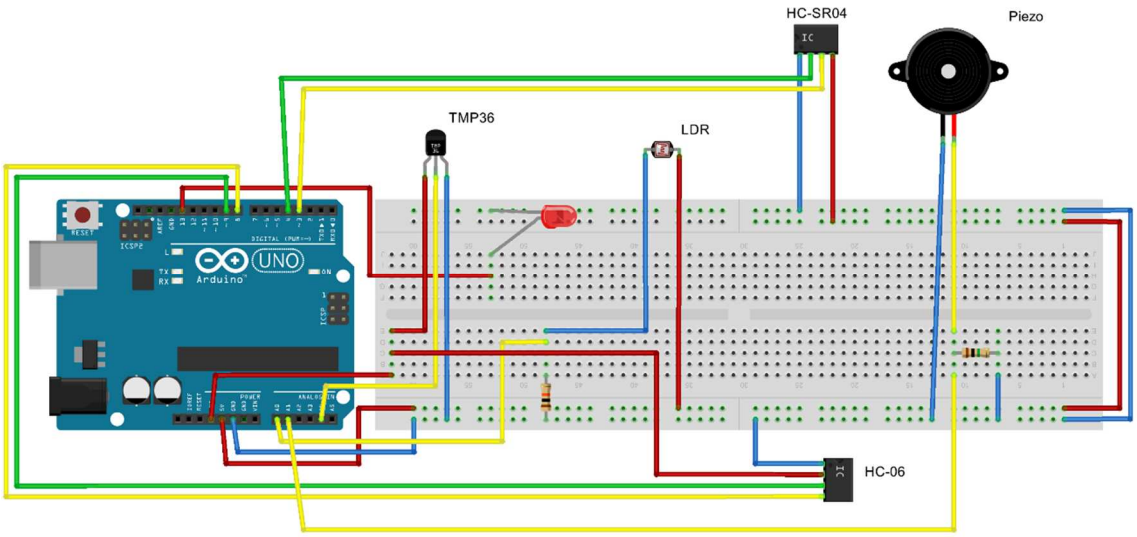



Figura 12 - Ligação dos sensores e módulos ao Arduino.

O capítulo seguinte aborda o desenvolvimento da plataforma em detalhe.

# CAPÍTULO 4 – IMPLEMENTAÇÃO

Este capítulo descreve a fase de desenvolvimento das diferentes aplicações da plataforma. A primeira secção é dedicada ao Arduino, a segunda secção aborda o programa criado para o Windows e a última secção é dedicada ao Android.

A arquitetura definida para o projeto, apresentada no capítulo anterior e repetida aqui por conveniência, contempla três níveis (Figura 13).

O primeiro nível conta com duas aplicações, uma para o Windows outra para Android. Estas aplicações partilham funcionalidades comuns entre si, mas também implementam opções que tiram partido das especificidades dos dispositivos em que são executadas.

No segundo nível reside o protocolo criado para a comunicação entre o Arduino e os restantes dispositivos, computador e dispositivo móvel. Este protocolo foi implementado como um componente independente do resto do projeto, ainda que integrado neste, por forma a facilitar a sua reutilização.

O terceiro nível é materializado pela aplicação que corre no Arduino, gerindo o microcontrolador e os sensores que lhe estão ligados. Esta aplicação procede à leitura dos valores detetados pelos sensores e envia, em resposta a mensagens recebidas, estes resultados para as aplicações do nível superior utilizando o protocolo de comunicação estabelecido.

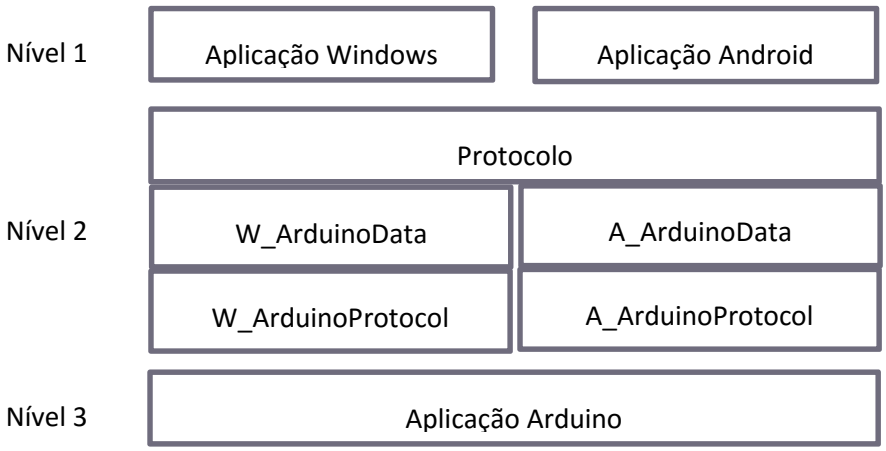



**Figura 13 - Arquitetura da plataforma.**

De seguida são apresentadas as seções que descrevem o desenvolvimento de cada um destes níveis.

## 4.1 – A APLICAÇÃO ARDUINO

Esta aplicação corresponde ao terceiro nível da arquitetura da plataforma. Trata-se do "sistema operativo" do Arduino que se encarrega de receber as mensagens das aplicações do primeiro nível, gere o estado dos pinos, analógicos e digitais, e envia os valores capturados de volta.

A ferramenta utilizada para desenvolver a aplicação para o Arduino foi o IDE do Arduino versão 1.6.1, esta pode ser retirada do site do Arduino e existem versões para Windows, Mac OSX e Linux. Esta ferramenta é utilizada como editor do código, compilador e permite ainda enviar o programa compilado para o Arduino.

A codificação do protocolo no Arduino foi orientada pela limitada quantidade de recursos da plataforma, nomeadamente ao nível da memória dinâmica disponível para as variáveis.

O programa atualmente consome 26% da memória do Arduino, correspondendo a 543 bytes. Para este valor contribuem não só as variáveis do protocolo mas também as variáveis do código das *frameworks* utilizadas.

Também foi devido aos limitados recursos de memória, no total são dois quilobytes, que não foi possível optar por capturar e armazenar os dados para enviar somente no final do processo de amostragem.

A aplicação no Arduino implementa todas as funcionalidades do protocolo. Ao longo do desenvolvimento foi essencial assegurar que o código desta não esgotava as limitadas capacidades do *hardware*, especialmente ao nível da memória disponível para armazenar o código e as variáveis.

### 4.1.1 – ESTRUTURA GERAL

Esta aplicação está dividida em três blocos de código (ver algoritmo). O primeiro que verifica a existência de dados a receber, via cabo USB ou Bluetooth. O segundo que processa os comandos recebidos e a serem executados. Por fim, o bloco que faz o envio dos dados, mais uma vez, via cabo USB e/ou Bluetooth.

Um programa no Arduino, também conhecido por *sketch* (Arduino SA, 2015a), tem sempre duas funções base, aqui designadas por Iniciar (função setup) e Repetir para sempre (função loop).

**Estrutura do algoritmo de processamento do protocolo no Arduino**

```
1. Iniciar
        (a) Definição do estado inicial do Arduino

2. Repetir para sempre
        (a) Se (existem dados para receber)
                        leitura_dados()
        (b) Se (bluetooth ligado e existem dados para receber no bluetooth)
                        leitura_dados_bluetooth()
        (c) Se (comando<>-1 e parâmetro<>-1)
                processaComando()
                comando=-1
                parâmetro1=-1
        (d) LerEnviarDados()
```

A função Iniciar é executada uma vez quando o Arduino é ligado e deve ser utilizada para definir o estado inicial do dispositivo, definindo taxas de transmissão de dados ou iniciando variáveis.

A função Repetir para sempre é chamada num ciclo infinito, enquanto o Arduino está ligado. É nesta função que ocorre o processamento de todo o código que implementa o protocolo.

Inicialmente, verifica-se a existência de mensagens recebidas na porta série, USB e Bluetooth. Caso existam, estas são lidas e posteriormente processadas. Após processamento, o comando e os respetivos parâmetros são limpos.

Por fim executa-se a função que verifica a necessidade de enviar dados, de acordo com os comandos recebidos e o estado atual do Arduino, para, de seguida, se repetir o procedimento.

O estado interno dos pinos é mantido num vetor. Cada pino pode ter um de três estados:

- Estado fechado – não existe leitura nem envio de dados;
- Estado aberto somente para uma leitura – o Arduino lê uma vez o valor e procede ao fecho do pino e, claro, ao envio do valor lido;
- Estado aberto – o Arduino em cada iteração faz a leitura e o envio do valor do pino, até ser recebido o comando para fechar o pino.

### 4.1.2 – COMANDOS IMPLEMENTADOS

O protocolo é baseado num conjunto de mensagens pré-definidas com uma estrutura básica comum. Para o desenvolvimento deste protocolo foi estudado o protocolo MIDI. Tal como neste último, cada mensagem pode ter um número variável de parâmetros (Stanford, 2014).

O formato das mensagens é o sempre o mesmo independentemente de serem recebidas via USB ou Bluetooth. Neste projeto só a aplicação Windows é que envia comandos para o Arduino, via USB, mas nada impede que noutro projeto as mensagens sejam recebidas através do Bluetooth.

As mensagens enviadas para o Arduino têm a estrutura apresentada na Tabela 3.

**Tabela 3 - Formato das mensagens enviadas para o Arduino**

| Comando | Parâmetro 1 | Parâmetro 2 |
|---|---|---|
| 1 byte - obrigatório | 1 byte - obrigatório | 1 byte - opcional |

As mensagens enviadas pelo Arduino para as aplicações têm uma estrutura fixa composta por três partes. A primeira indica o tempo, em microssegundos, de quando os dados foram recolhidos, a segunda contém o valor lido e, por fim, a origem dos dados, número do pino ou do sensor.

Para a conceção do protocolo foi necessário considerar a utilização a dar ao Arduino. A principal função desempenhada pelo Arduino é servir de interface entre as aplicações e os sensores. Uma vez que as suas capacidades de armazenamento são muito limitadas o principal trabalho

realizado é a leitura dos valores fornecidos pelos sensores e o envio para as portas de comunicação.

O Arduino possui dois tipos de pinos: analógicos e digitais. Assim, os comandos mais utilizados são os que fazem a leitura direta destes pinos. Os pinos digitais limitam-se a valores binários do tipo ligado ou desligado (1 ou 0). Os pinos analógicos estão conectados a um conversor analógico-digital com uma resolução de 10 bits por isso devolvem valores na escala de 0 a 1023.

Para a leitura dos valores foram definidos dois comandos: um que faz uma única leitura do pino e outro que abre o pino para leituras sucessivas, até que seja recebido um comando para fechar o pino.

Para terminar a leitura dos pinos também foram implementados dois comandos. Um que fecha um pino indicado e outro que fecha todos os pinos.

**Tabela 4 - Lista de comandos para leitura de valores de pinos**

| Comando | Descrição |
|---|---|
| **ABRIR_PINO** | Inicia a leitura permanente do pino indicado |
| **LER_ESTADO_PINO** | Lê o estado de um pino |
| **FECHAR_PINO** | Termina a leitura do pino indicado |
| **FECHAR_TODOS** | Termina a leitura de todos os pinos |

Com estes comandos é possível fazer a leitura do estado de qualquer porta analógica ou digital do Arduino. Para este projeto fazemos uso destes comandos para ler sensores como LDRs, Piezos, NTCs, TMP36 ou outros (ver secção 5.6 para mais pormenores sobres estes sensores).

Estes comandos e o formato das mensagens foram evoluindo com os testes realizados. A primeira atividade laboratorial considerada para o projeto foi a queda livre de um objeto que consiste na medição da aceleração da velocidade da queda, em função da gravidade. Esta atividade necessita de medir a queda de uma régua transparente de aproximadamente 40 cm cuja duração da queda não chega a meio segundo (500 milissegundos).

Esta necessidade, em termos de tempo de resposta, não era satisfeita nas primeiras versões do protocolo. A primeira versão conseguia capturar dados de um pino analógico de 15 em 15 milissegundos, o formato das mensagens era o apresentado na Tabela 5.

Tabela 5 – Formato inicial das mensagens enviadas pelo Arduino

| Letra P seguida de : | Número do pino lido | Separador | Letra V seguida de : | Valor lido |
|---|---|---|---|---|
| P: (2 bytes) | Inteiro (2 bytes) | ; (1 byte) | V: (2 bytes) | Inteiro (2 bytes) |

Este formato tinha alguns problemas:

- Não tinha o tempo em que o valor foi capturado, o que implicava utilizar o tempo do dispositivo que recebia a mensagem criando disparidades quando a transmissão era simultânea, para o computador e para o Android;
- Era complicado de descodificar, implicava separar os valores numéricos dos restantes componentes da mensagem;
- Era desnecessariamente grande (9 bytes).

Inicialmente, para melhorar os tempos de resposta, foram realizados testes com a concatenação dos dados numa *string*, por forma a verificar se o atraso se devia ao envio dos dados em várias instruções, uma por cada parte da mensagem, em vez de uma só mensagem.

Esta alteração, ao invés de melhorar os tempos de envio, pioraram-nos, para além de promover uma maior utilização da memória *RAM* do Arduino, pois as *strings* estão implementadas num *header file* que necessitava ser incluído no código do projeto.

Dados estes resultados, enveredou-se pela simplificação do protocolo, passando cada mensagem enviada a partir do Arduino a ter duas partes. A primeira parte da mensagem contendo o tempo de captura em microssegundos (um microssegundo divide o segundo em um milhão de partes ou seja $10^{-6}$ = 0,000001) e a segunda parte passou a conter o pino ou sensor e o valor lido promovendo-se a sua junção matematicamente tirando partido do facto do valor nunca ultrapassar 4 dígitos (no máximo o valor atinge 1023). Assim o formato das mensagens enviadas passou a ser:

Tabela 6 - Formato das mensagens enviadas pelo Arduino

| Tempo de captura em microssegundos | Separador | Pino*10000+valor |
|---|---|---|
| Unsigned long (4 bytes) | ; (1 byte) | Unsigned long (4 bytes) |

Com base na documentação consultada, o Arduino necessita de 100 microssegundos (0,0001 segundos) para fazer a leitura de um pino analógico (Arduino SA, 2015b). Com estes valores deveria ser possível ler 10,000 valores por segundo (10.000Hz).

Sendo necessário atingir maiores velocidades foram realizados testes com a velocidade de ligação, ou *baud rate*, definida pelo número de bits enviados por segundo. Apesar da documentação do Arduino sugerir velocidades de transmissão de até 115.200 bits por segundo (bps) é possível atingir velocidades superiores. Após alguns testes realizados verificou-se que o protocolo funciona em modo *full duplex* (bidirecional simétrico) a uma velocidade de 500000 bps.

Uma última otimização implementada, ao nível da função de leitura dos pinos analógicos, foi conseguida através da manipulação dos tempos de resposta do conversor de analógico para digital (Margolis, 2011).

Após estas atualizações foram realizados novos testes que permitiram obter uma taxa de captura na ordem dos 1.000 Hz (1.000 valores lidos por segundo) possibilitando, assim, a realização de experiências envolvendo queda de objetos.

Com uma taxa de captura de dados tão elevada, facilmente se verificou que grande parte dos dados enviados são iguais. Com o objetivo de minimizar o envio de dados repetidos foram definidos comandos que ativam e desativam a filtragem dos valores lidos antes de serem enviados, só ocorrendo o envio quando a diferença dos valores a enviar seja superior a um determinado valor de ruído estabelecido em relação ao último valor enviado a partir do mesmo pino. Também foi criado um comando para definir o valor do ruído (ver Tabela 7).

**Tabela 7 - Comandos relativos à filtragem dos dados enviados**

| Comando | Descrição |
|---|---|
| **FILTRAR_ON** | Ativa a filtragem dos valores antes de serem enviados. Opcionalmente pode ser passado o valor do ruído. |
| **FILTRAR_OFF** | Desativa a filtragem dos valores |
| **DEFINE_RUIDO** | Define o valor de ruído, a partir do qual os valores são enviados |

Desde o início deste trabalho que um dos objetivos é permitir a comunicação com um computador, via USB, e, em simultâneo, um dispositivo móvel Android, via Bluetooth.

A primeira versão do protocolo implementada no Arduino fazia com que essa comunicação acontecesse sempre em simultâneo.

Tendo em conta que o Arduino *Uno R3* só tem uma USART – *Universal Synchronous and Asynchronous Serial Receiver and Transmitter* (Atmel, 2011) - o primeiro problema passou pela pesquisa de um modo de replicar a comunicação para um segundo dispositivo. A solução

encontrada foi utilizar uma *framework* que implementa uma porta de comunicação por *software*. Como é natural, esta não consegue o mesmo desempenho da porta física.

Nos testes realizados com e sem envio em simultâneo foi possível comprovar o custo desta comunicação dupla dos dados lidos. Como se pode observar na Tabela 8 a taxa de leitura é muito menor quando se procede ao envio em simultâneo.

**Tabela 8 - Taxa de captura de dados**

| Tempo de captura de dados do teste (segundos) | Com envio simultâneo (USB e Bluetooth) | | Só envio para USB | |
|---|---|---|---|---|
| | Número de leituras de um LDR | Taxa de leitura | Número de leituras de um LDR | Taxa de leitura |
| 3,43466 | 217 | 61,17 Hz | 4111 | 1196,91 HZ |

Com estes valores de transmissão a qualidade dos resultados em atividades cuja taxa de leitura é fundamental, como por exemplo na atividade Queda Livre, não é aconselhável utilizar a comunicação Bluetooth, sendo preferível capturar os dados no computador e partilhá-los, no final, com os dispositivos móveis.

Para permitir um maior controlo sobre a comunicação foram implementados mais dois comandos que permitem ativar e desativar o envio dos dados e a receção de comandos via *Bluetooth*.

**Tabela 9 - Comandos para comunicação via Bluetooth**

| **Comando** | **Descrição** |
|---|---|
| **LIGAR_BT** | Ativa a comunicação através de Bluetooth |
| **DESLIGAR_BT** | Desativa a comunicação através de Bluetooth |

Estes comandos permitem também que seja a aplicação desktop a controlar o uso ou não dos dispositivos móveis, permitindo que seja o docente a decidir se deixa ou não que os seus alunos os utilizem.

O controlo dos pinos digitais passa por três comandos, um para ligar o pino colocando em estado de ativado (HIGH) e outro para desligar o pino desativando-o (estado LOW). Nos pinos digitais,

que permitem utilizar o Pulse With Modulation (PWM), é possível utilizar um comando para enviar um valor intermédio.

Tabela 10 - Comandos para pinos digitais

| Comando | Descrição |
|---|---|
| **LIGAR_PINO** | Pino digital passa para o modo de OUTPUT e fica em estado HIGH |
| **DESLIGAR_PINO** | Pino digital passa para o modo de INPUT e fica em estado LOW |
| **VALOR_PWM** | Pino digital passa para modo de OUTPUT e é enviado um valor para PWM |

Com estes comandos é possível ligar e desligar LEDs ou controlar motores DC incluindo definir uma velocidade variável entre parado ou velocidade máxima.

Para além destes comandos, foi desenvolvido um comando específico para um sensor de distância (ver Tabela 11).

Uma vez que, ao contrário dos pinos analógicos que são lidos com uma única função independentemente do tipo de sensor instalado, a leitura de dados de sensores digitais depende de código específico a cada caso. Assim, foi necessário implementar funções tendo em conta o tipo de sensor, ainda que tenham sido utilizadas bibliotecas de funções, sempre que disponíveis, que permitem a utilização de diferentes modelos de sensores.

De futuro, a adição de outros sensores digitais implica a atualização do protocolo.

A consulta de valores neste tipo de sensores tem um custo implícito no tempo de resposta. No caso do sensor de distância, que depende de um mecanismo baseado no retorno de um som emitido, o melhor que é possível obter com alguma segurança são 30 leituras por segundo (Eckel, 2012). Ainda assim, na versão atual do protocolo o número de leituras ronda aproximadamente as 60 por segundo.

Tabela 11 - Comandos para ligar e desligar sensor distância

| Comando | Descrição |
|---|---|
| **SENSOR_DISTANCIA_ON** | Ativa a leitura e envio a partir do sensor de distância |
| **SENSOR_DISTANCIA_OFF** | Desativa o sensor de distância |

Sempre que disponíveis, devem-se privilegiar os sensores analógicos aos sensores digitais, pois são mais baratos, mais simples de ligar e a leitura de dados é mais rápida.

Para o suporte a futuras versões do protocolo e das aplicações, existe um comando que permite consultar a versão do protocolo instalada no Arduino (ver Tabela 12). Deste modo, em futuras versões que requeiram uma versão diferente da atual, será possível informar o utilizador da necessidade de atualizar o protocolo do Arduino.

Tabela 12 - Comando para consultar versão do protocolo

| Comando | Descrição |
|---|---|
| **VERSAO** | Devolve a versão do protocolo instalada no Arduino |

Todos estes comandos foram implementados com base em constantes simbólicas em hexadecimal. Cada mensagem é terminada com uma quebra de linha que serve como carater terminador para detetar o fim de cada comando.

A seção seguinte descreve em detalhe a aplicação Windows que envia os comandos do protocolo e recebe as mensagens com os dados recolhidos.

## 4.2 – A APLICAÇÃO WINDOWS

A ferramenta utilizada para desenvolver o programa para o Windows foi o Visual Studio, inicialmente na versão 2013 e posteriormente na versão 2015.

### 4.2.1 – ESTRUTURA GERAL

A aplicação Windows é, para o utilizador, a parte principal deste projeto. Será com esta que este interage e é também responsável por enviar comandos para o Arduino, bem como por receber e tratar os dados capturados pelos sensores.

A aplicação está organizada em três camadas (ver Figura 14). A camada superior é a que interage com os utilizadores. Esta é constituída pelo conjunto de formulários e controlos que constituem a interface. A interação com o Arduino, fundamental no envio de comandos e na receção dos dados, está noutra camada implementada pela classe W_ArduinoProtocol que implementa o protocolo de comunicação. Por fim existe uma camada intermédia que armazena os dados (classe W_ArduinoData), que implementa funções de tratamento estatístico e que incorpora a capacidade de guardar e ler os dados materializando estes no sistema de ficheiros, utilizando o formato XML.

As classes W_ArduinoProtocol e W_ArduinoData foram compiladas para uma DLL independente da aplicação, permitindo assim uma reutilização das funções implementadas noutros projetos.

Estas classes são fundamentais para a camada superior, a da interface, pois a mecânica da comunicação e armazenamento dos dados foi aqui implementada.

A opção por utilizar uma interface baseada em Windows Forms ao invés da mais moderna Windows Presentation Foundation (WPF) prende-se com o atual parque informático das escolas portuguesas. Tratando-se na maior parte de computadores com 2 gigabytes de memória RAM e com Windows 7, existindo ainda muitas máquinas com Windows Vista e Windows XP, nos quais a execução da aplicação seria mais lenta com a interface WPF.

No entanto uma atualização da interface não implicará uma reescrita total da aplicação, uma vez que das três camadas desenvolvida só o nível superior terá de ser substituído.

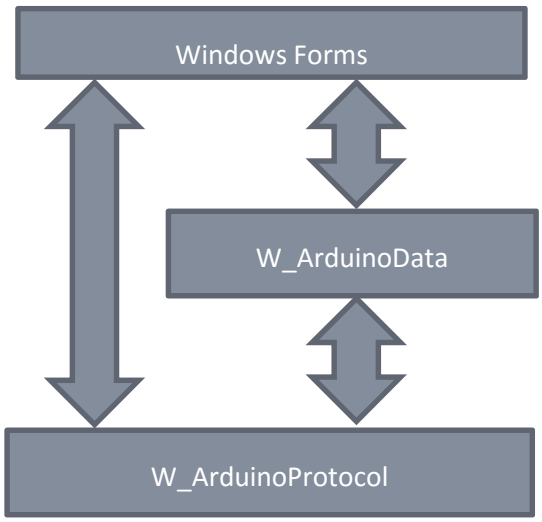



**Figura 14 - Arquitetura da aplicação Windows.**

Quanto à materialização dos dados optou-se por utilizar um formato aberto que permite, não só a partilha entre os dispositivos fixos e móveis, como também a importação para outras aplicações como folhas de cálculo ou até sistemas de gestão de bases de dados.

Não sendo possível garantir que todos os dados capturados estão livres de erros ou interferências foram incluídas opções que permitem a remoção dos dados considerados indesejáveis, sendo possível ao utilizador selecionar várias linhas das tabelas de dados e escolher pela sua remoção.

Um objetivo definido para esta aplicação foi a possibilidade de capturar dados de vários Arduinos em simultâneo, de modo a permitir a execução de várias experiências numa aula prática. Esta capacidade foi conseguida (Figura 15), cada atividade instancia um objeto do tipo

W_ArduinoProtocol que utiliza uma *thread* independente para a captura. Assim, desde que existam vários Arduinos ligados ao computador, é possível ao docente ter vários grupos de alunos a implementar as suas próprias experiências, mantendo o controlo sobre o processo.

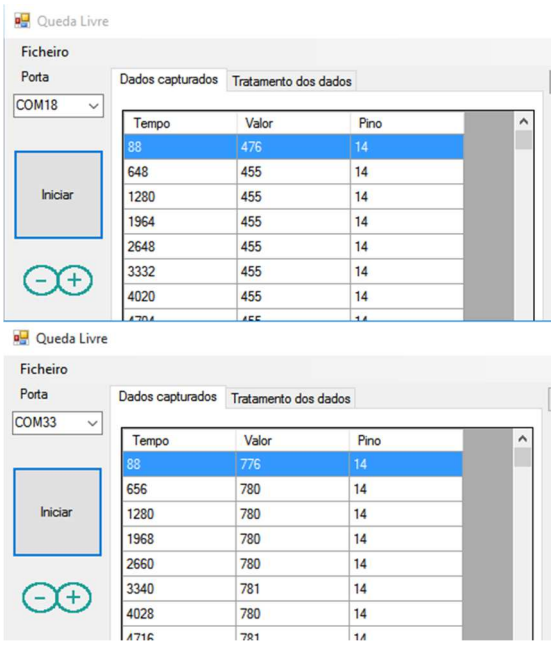



Figura 15 - Captura de dados em simultâneo.

**Classe W_ArduinoProtocol**

Esta classe é responsável por toda a comunicação com o Arduino através do cabo USB. É nesta classe que estão definidas as constantes simbólicas utilizadas como comandos nas mensagens enviadas. Também é aqui que se encontra definida a velocidade de comunicação e o objeto do tipo SerialPort que faz a ligação, ao nível mais baixo, com o Arduino.

A classe SerialPort permite o controlo de uma porta série através de eventos de receção e envio de dados, bem como a definição das propriedades de comunicação, como seja a velocidade de envio e receção. Esta classe existe desde a versão 2 do .Net framework.

Com o objetivo de avaliar a solução, durante a fase de desenvolvimento, foram implementadas duas propriedades que indicam a quantidade de valores lidos corretamente e a quantidade de valores considerados erros de comunicação por não ser possível descodificar o seu conteúdo. Com esta funcionalidade foi possível verificar a qualidade das mensagens recebidas, nomeadamente se a leitura destas no computador ocorria sem que se perdessem dados devido a erros de comunicação ou mensagens mal formadas, ou ainda devido ao *buffer* de comunicação se encontrar cheio.

É na função de leitura das mensagens que são atualizadas as propriedades que permitem avaliar a qualidade das mensagens recebidas. Sempre que uma mensagem recebida é corretamente

descodificada o contador de mensagens corretamente recebidas é incrementado por oposição quando uma mensagem recebida não apresenta o formato previsto, nesse caso o contador de mensagens com erros é atualizado.

Nas versões iniciais da aplicação Windows, em que a leitura ainda não era feita numa thread separada, as grelhas (DataGridView) com os dados recebidos eram atualizadas em tempo real durante a captura dos dados. Nesta fase foi possível verificar a existência de erros de leitura das mensagens, graças às funcionalidades de verificação da qualidade das mensagens. Estes erros ocorriam devido a partes das mensagens serem perdidas quando o *buffer* de memória da porta de comunicação se encontrava cheio.

Para resolver este problema optou-se por não atualizar a grelha dos dados durante a captura, na atividade Queda Livre, e por fazer a captura dos dados numa thread separada.

Também foi graças a estas funcionalidades de controlo de qualidade das mensagens que foi possível detetar erros no código da aplicação do Arduino, nomeadamente quando vários pinos eram abertos para leitura que implicava o envio de dados em simultâneo verificou-se que o programa gerava mensagens com um formato errado.

Na atividade Playground o utilizador pode consultar o número total de mensagens recebidas e o número de mensagens recebidas que não foi possível descodificar devido a erros.

A função mais importante da classe W_ArduinoProtocol é a que processa a leitura das mensagens recebidas. Esta é executada numa *thread* separada para garantir que o processo de receção dos dados é independente do resto da aplicação.

A *thread* de captura dos dados procede à leitura das mensagens recebidas, retira os dados destas e armazena-os para tratamento e apresentação ao utilizador. O processo de leitura dos dados e armazenamento decorre em simultâneo com a leitura e apresentação ao utilizador, sendo executada na *thread* da interface e podendo, por isso, ocorrerem erros de concorrência no acesso aos dados. Estes erros acontecem quando um mecanismo procede à alteração dos dados e em simultâneo outra função tenta fazer a leitura dos mesmos.

Para evitar erros de concorrência foi necessário proceder a um mecanismo de bloqueio (Microsoft, 2015b) das estruturas de dados por forma a armazenar os valores recebidos em segurança, situação fundamental quando em simultâneo a aplicação recebe dados e apresenta-os ao utilizador. Sem este mecanismo a consulta dos dados em simultâneo com a captura gerava exceções, pois a classe DataTable é segura numa utilização com várias *threads* mas somente nas operações de leitura não nas operações de escrita (Microsoft, 2015a). Durante as operações de

adição, alteração e remoção de dados da DataTable é necessário bloquear o acesso até estas operações terminarem.

A classe W_ArduinoProtocol mantém um vetor com os últimos valores lidos de cada pino e sensor instalado no Arduino. Este vetor facilita uma consulta imediata não sendo necessário recorrer à tabela que contém todos os dados recebidos.

Para além das propriedades referidas a classe implementa uma função para cada tipo de mensagem prevista no protocolo, estas funções são utilizadas pelas funcionalidades criadas para as atividades laboratoriais facilitando, assim, a utilização da classe e a programação do algoritmo de comunicação com o Arduino.

A estrutura completa das propriedades e dos métodos da classe pode ser consultada nos anexos deste documento (ver Anexo C – Estrutura das Classes).

É nesta classe que existe um objeto criado a partir da classe W_ArduinoData cuja função é armazenar os dados e implementar funções de tratamento e análise dos valores capturados.

**Classe W_ArduinoData**

Esta classe utiliza um objeto do tipo DataTable para armazenar todos os dados recebidos durante uma sessão de comunicação com o Arduino. A DataTable é uma classe de objetos existente no ADO.NET desde as primeiras versões (Microsoft, 2015a), sendo considerada *thread* safe durante a leitura dos dados enquanto a escrita deve ser sincronizada.

Para além do objeto para armazenar os dados, existe uma propriedade que permite filtrar os dados em função de um determinado valor mínimo, dentro do qual as variações são consideradas negligenciáveis, tratando-se de ruído. Esta propriedade é particularmente útil em algumas funções de tratamento e análise de dados em que se procuram determinados picos de valores.

A classe W_ArduinoData é utilizada, não só para os dados que são capturados através do protocolo de comunicação (implementado na classe W_ArduinoProtocol), mas também para guardar os dados que são tratados pelo utilizador. Para isso, o DataTable é reconfigurado em função das necessidades de cada atividade experimental.

A classe DataTable é uma classe fundamental no ADO.NET (Microsoft, 2015a) e neste projeto é a base de todo o armazenamento dos dados recebidos e calculados. Desde a receção dos dados através das mensagens do Arduino até aos cálculos realizados sobre estes em cada uma das atividades laboratoriais e ainda na atividade Playground onde se tira partida da funcionalidade

que permite adicionar colunas calculadas que executam operações aritméticas sobre dados de outras colunas.

Em relação às funções da classe existe um conjunto de funções de utilização transversal ao projeto e outras mais específicas a cada atividade.

Quanto às funções mais gerais desenvolveram-se métodos para:

- Adicionar valores ao DataTable, com os diferentes formatos associados a cada atividade laboratorial;
- Devolver valores de pinos ou sensores, é possível obter diferentes valores como o primeiro valor recebido ou o último;
- Devolver a lista dos pinos de onde os dados foram recebidos;
- Devolver o número total de dados recebidos;
- Devolver a média dos valores armazenados;
- Devolver a moda dos valores;
- Devolver o valor menor e o maior valor;
- Devolver a amplitude dos dados (diferença entre o maior e o menor valor);
- Guardar e ler dados utilizando um ficheiro XML.

Quanto às funções específicas destaca-se:

- Na atividade de queda livre de um objeto, uma função que permite analisar todos os dados armazenados em busca de picos nos valores que evidenciem o obscurecer do sensor quando o objeto cai;
- Na atividade da bola Saltitona é utilizada um função que analisa os valores procurando encontrar os pontos de queda e ressalto da bola ao saltar após colidir com o solo. Este método foi desenvolvido com diversos mecanismos de limpeza dos dados, uma vez que o sensor de distância devolve valores com algum ruído que devem ser ignorados.

O capítulo relativo aos resultados obtidos em cada uma das atividades experimentais descreve em mais pormenor algumas das funções criadas para o tratamento de dados.

## 4.3 – A APLICAÇÃO ANDROID

A aplicação Android foi criada com o Android Studio versão 1.3. Esta ferramenta realiza o trabalho típico de um IDE, edita e compila o código, mas também procede ao envio da aplicação para um dispositivo Android ou para um emulador.

Sendo esta aplicação de utilização complementar com o programa principal do Windows, não se desenvolveram funcionalidades de envio de mensagens de controlo para o Arduino. Assim, a captura dos dados só é iniciada pelo microcontrolador quando a aplicação Windows o determina. Esta opção permite um maior controlo sobre a execução das atividades laboratoriais.

Os dados guardados na aplicação Windows podem ser lidos pela aplicação Android e vice-versa. Deste modo o docente controlando a aplicação principal, pode executar a captura dos dados e posteriormente partilhar os dados com os dispositivos móveis via correio eletrónico ou utilizando um qualquer serviço de partilha de ficheiros.

O desenvolvimento desta aplicação implicou um trabalho suplementar, em relação à aplicação Windows, pois no sistema operativo Android não estão disponíveis classes de objetos que existem no .NET, como por exemplo o DataTable para armazenar os dados, para além de a comunicação ser via Bluetooth, cujo processo de ligação é mais complicado do que a ligação USB.

### 4.3.1 – ESTRUTURA GERAL

A aplicação Android é de uso complementar ao programa Windows. O objetivo é que os discentes tenham acesso a esta aplicação em dispositivos móveis, onde podem receber os dados das atividades laboratoriais, diretamente ou através dos ficheiros com os dados gravados na aplicação Windows.

O uso da mesma é controlado pelo docente que através da aplicação principal permite ou não o envio dos dados do Arduino para esta.

Tratando-se de uma aplicação para uso pelos alunos, alguns cálculos executados nas atividades laboratoriais são deliberadamente escondidos, uma vez que se espera que os próprios discentes os realizem.

O objetivo desta aplicação é essencialmente permitir a captura dos dados ou abertura dos ficheiros com os dados capturados e a execução das operações normalmente realizadas numa calculadora gráfica, como sejam os gráficos dos dados ou as regressões para obtenção das componentes das equações.

O código da aplicação encontra-se estruturado em três camadas:

- No nível mais próximo do utilizador encontram-se as classes das *activities* que apresentam a interface ao utilizador (*view*), controlam os dados inseridos e mostram os resultados das operações realizadas;

- No nível mais próximo do Arduino encontra-se a classe que suporta o protocolo de comunicação, denominada A_ArduinoProtocol;
- Num nível intermédio encontra-se uma classe responsável pela gestão dos dados capturados, denominada A_ArduinoData.

A Figura 16 apresenta um diagrama esquemático da organização da aplicação.

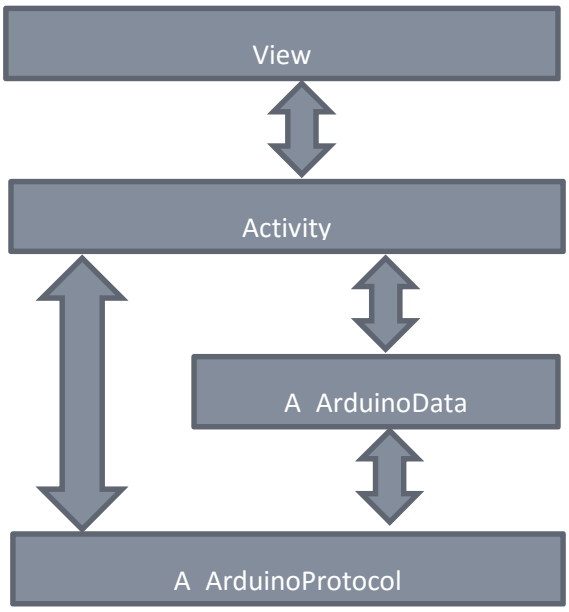



**Figura 16 - Estrutura da aplicação Android.**

Uma preocupação transversal no desenvolvimento desta aplicação foi a tentativa de otimizar a utilização da memória primária do dispositivo, uma vez que este pode ter uma quantidade limitada de memória, especialmente se for um dispositivo mais antigo.

Tendo presente este requisito, foram implementadas algumas medidas que libertam memória assim que esta deixa de ser necessária, como por exemplo, quando o utilizador fecha uma atividade laboratorial para voltar ao menu principal a *activity* é terminada e tira-se partido do evento onDestroy desta, para libertar toda a memória que foi reservada dinamicamente, incluindo uma chamada à função do sistema operativo que sinaliza que este é um bom momento para executar o Garbage Collector (Google, n.d.-c).

A interface de uma aplicação móvel é um desafio relativamente novo, no desenvolvimento de programas. Não só pelas reduzidas dimensões do ecrã, mas também pelo método do utilizador interagir com o dispositivo, não existindo teclado físico nem rato.

No desenho da aplicação optou-se por manter a consistência, dentro da aplicação Android, mas também entre esta e a aplicação Windows. As duas aplicações dispõem de uma tabela que lista os dados capturados e um segundo separador com os dados tratados. A estrutura dos menus é a mesma para as atividades laboratoriais.

Ao contrário da aplicação Windows, os gráficos são apresentados ocupando totalmente o ecrã não sendo, por isso, possível apresentar fórmulas em simultâneo com os gráficos.

A seleção de pontos no gráfico para execução de regressões também foi um desafio, sendo necessário permitir ao utilizador escolher os pontos de interesse e posteriormente passando estes dados para a atividade laboratorial. Para cada ponto selecionado é apresentada uma linha que marca o ponto no gráfico e são apresentadas as coordenadas ao utilizador podendo este escolhê-lo, se, de facto, aquele ponto for para adicionar à lista ou não.

Os gráficos desenvolvidos suportam um conjunto de gestos padrão que permitem executar tarefas de tratamento e análise dos dados:

- Com dois dedos afastar para aumentar ou diminuir o detalhe (zoom in e zoom out);
- Com um dedo deslocar o gráfico para a esquerda e para a direita;
- Com um dedo em cima de um ponto, para o selecionar.

Com o objetivo de validar a exatidão das aplicações os resultados obtidos na aplicação Windows têm de ser exatamente iguais aos da aplicação Android, para os mesmos dados. Assim, durante o desenvolvimento destas foram feitos testes comparativos patentes nas Figura 17 e Figura 18.

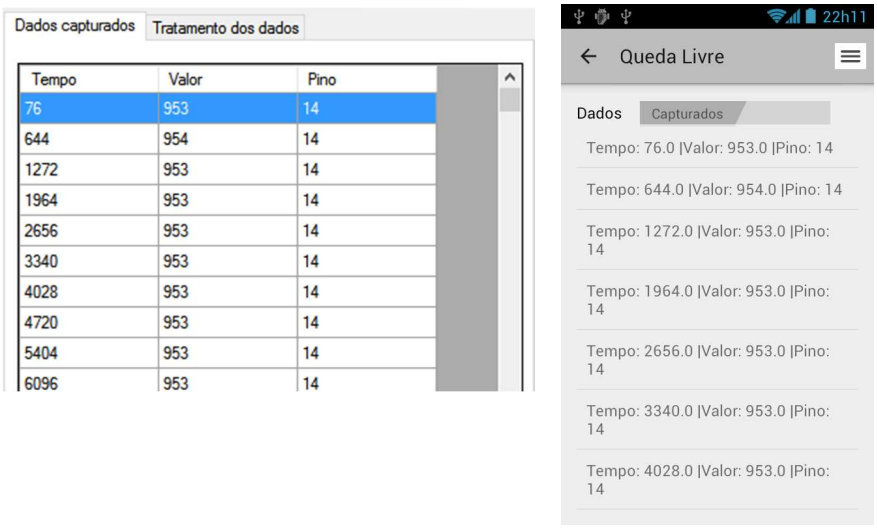



Figura 17 - Dados capturados no Windows e no Android.

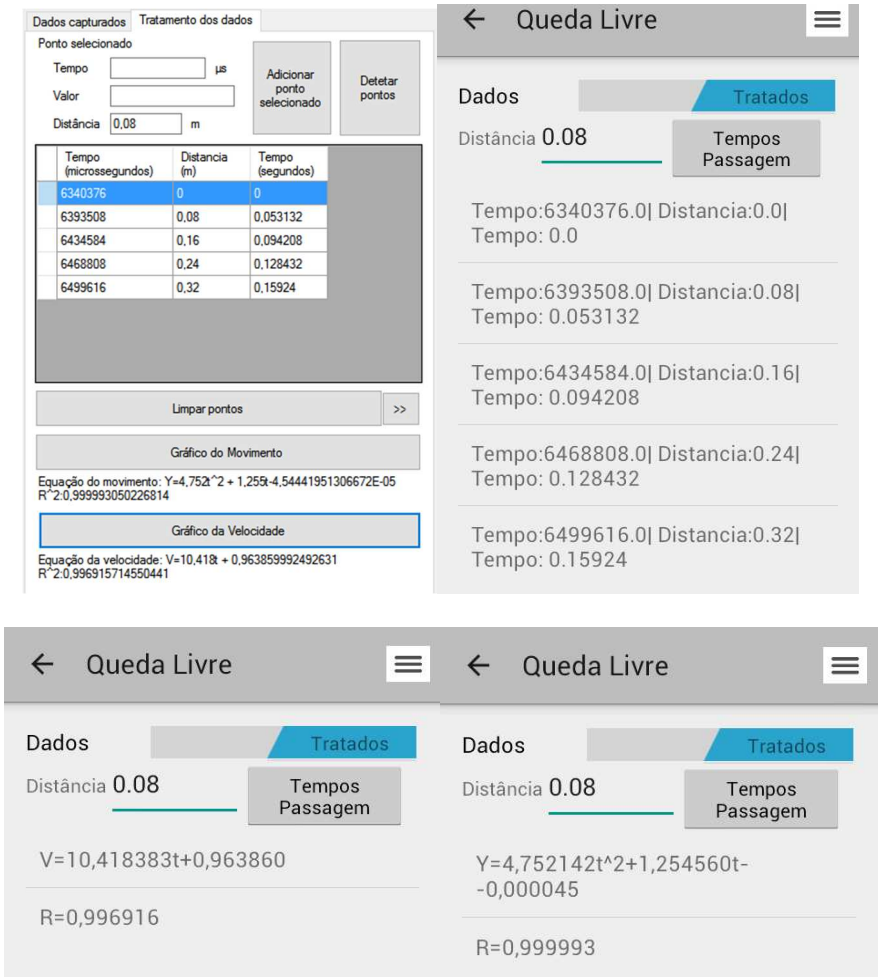



**Figura 18 - Comparação dos resultados obtidos no Windows e no Android.**

Com estes testes garante-se a implementação correta do protocolo nas duas aplicações e ainda as classes que executam as regressões lineares e não lineares.

**A_ArduinoProtocol**

Esta classe é responsável pela implementação do protocolo na aplicação. A classe define as constantes simbólicas representativas das mensagens bem como as funções que procedem ao seu envio.

Uma vez que a comunicação ocorre via Bluetooth as propriedades da classe também incluem três objetos de suporte à transmissão:

- BluetoothDevice, classe que representa o dispositivo, neste caso o Arduino, com o qual a aplicação vai comunicar;
- BluetoothAdapter, classe do objeto que representa o adaptador de comunicação no próprio dispositivo Android;
- BluetoothSocket, classe que suporta a comunicação, desde a ligação entre os dispositivos até ao envio e à receção.

O construtor da classe recebe o dispositivo previamente emparelhado com o qual se pretende comunicar. O emparelhamento ocorre na *activity* inicial da aplicação sendo terminado quando esta é destruída. Esta classe mantém dois vetores com os últimos dados recebidos, um para os sensores e outro para os módulos digitais.

Para utilização na fase de desenvolvimento foram incluídas duas propriedades que contabilizam o número total de dados recebidos corretamente e o número total de mensagens recebidas com erros, as mensagens que não foi possível decifrar.

O processo de receção das mensagens é executado numa *thread* separada da *thread* principal. Este processo foi desenvolvido criando uma classe derivada da class Thread. Esta classe cria uma unidade de execução concorrente (Google, n.d.-d).

Na implementação da classe de captura de dados foi criado um construtor que acede ao objeto BluetoothSocket e com este é aberto um InputStream para leitura dos dados recebidos. Para além do construtor é criado um método run que contém o código que é executado na *thread* nova.

Esta *thread* funciona como um serviço que é executado continuamente e que vai retirando as mensagens do *buffer* do dispositivo. Caso a opção de capturar dados tenha sido ativada, os dados recebidos são armazenados num objeto do tipo A_ArduinoData.

De seguida descrevem-se as duas classes mais importantes. A estrutura completa destas pode ser consultada no Anexo C – Estrutura das Classes.

**A_ArduinoData**

Esta classe é responsável pela gestão dos dados capturados. Para o armazenamento dos dados utiliza-se uma lista de dados. Uma lista é uma coleção de elementos (Google, n.d.-a), que, neste caso, cada elemento é um objeto do tipo SerieDados. Cada elemento desta classe tem três propriedades: o tempo de captura, o valor capturado e o pino de origem do valor.

Os dados podem ser materializados em ficheiros em formato XML. Para isso foi necessário desenvolver um conjunto de funções que procedem ao registo dos elementos existentes na lista no ficheiro, bem como funções de leitura destes para reconstituição da lista.

Para além de armazenar os dados, a classe A_ArduinoData implementa um conjunto de métodos que executam cálculos matemáticos como: a média, a moda, o valor máximo, o valor mínimo, para além de variações destas funções que executam os cálculos limitados aos dados de um só pino ou a um conjunto de amostras de dados.

As atividades de queda livre e bola saltitona dispõem de funções específicas que permitem apurar os pontos de interesse nos dados capturados. A atividade queda livre utiliza uma função para encontrar a passagem das linhas marcadas na régua. Já a atividade bola saltitona executa uma função que permite apurar os pontos de queda e ressalto da bola.

Antes de ser possível capturar os dados, o utilizador tem de estabelecer uma ligação entre o equipamento Android com o Arduino. A aplicação apresenta uma activity para ativar o Bluetooth e listar os dispositivos previamente emparelhados.

Ao escolher o dispositivo a ligar, é efetuada uma tentativa de ligação e o utilizador é informado do sucesso ou não desta operação. Após estabelecer uma ligação com sucesso, o utilizador pode executar um teste que faz ligar/desligar um LED no Arduino sinalizando a receção dos comandos.

Para a que o Arduino faça o envio dos dados para o dispositivo é necessário que na aplicação Windows essa opção seja ativada.

O capítulo seguinte apresenta os resultados obtidos com este projeto.

# CAPÍTULO 5 – RESULTADOS

Este capítulo apresenta os resultados obtidos, nas duas aplicações criadas, para cada uma das atividades laboratoriais e, ainda, uma breve descrição de um teste de utilização realizado com um grupo de alunos.

Desde o início deste projeto que a qualidade dos resultados tem sido uma preocupação: não basta conseguir capturar os valores dos sensores, é preciso que estes valores passem pelo crivo da realidade científica.

Uma vantagem de desenvolver *software* para a ciência é que tudo pode ser quantificado, sendo mais transparente a sua avaliação, não existe lugar a opiniões ou dúvidas. Se for pretendido obter a aceleração de um objeto a cair e as leis da física determinam esse valor, então é esse o valor que a plataforma tem de obter, caso contrário, algo está fundamentalmente mal concebido ou executado.

Para testar a qualidade dos resultados obtidos foram resolvidos exercícios de manuais escolares e comparados os resultados das soluções apresentadas no livro e os apurados pelas aplicações.

As secções seguintes descrevem cada uma das atividades especificamente implementadas de acordo com as orientações previstas do programa da disciplina de Física e Química A (Ferreira et al., 2014) e as recomendações dos docentes que apoiaram a execução deste projeto.

## 5.1 – ATIVIDADE LABORATORIAL QUEDA LIVRE

A primeira atividade desenvolvida foi a da Queda Livre, devido ao grau de dificuldade que esta apresentava. O facto de ser necessário medir a velocidade de queda de um objeto obrigou a um trabalho apurado de otimização, testes e revisão dos resultados.

As primeiras versões do protocolo somente conseguiam obter aproximadamente noventa leituras por segundo. O gráfico da Figura 19 mostra que esta cadência de captura de dados não era suficiente para estudar um objeto a acelerar a aproximadamente dez metros por segundo a cada segundo.

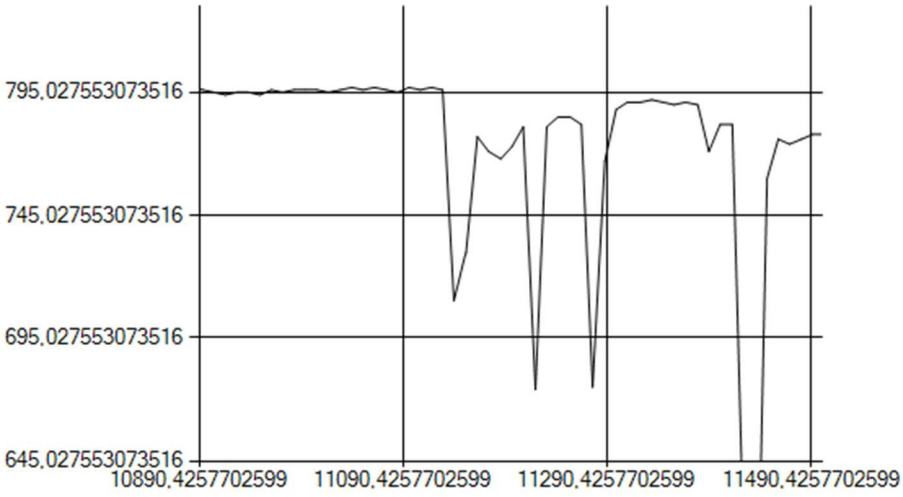



Figura 19 - Dados capturados na versão beta.

A versão atual do protocolo consegue uma taxa de amostragem superior a mil leituras por segundo. Para obter estes resultados foi necessário simplificar as mensagens enviadas pelo Arduino por forma a reduzir o volume de dados. A leitura dos valores analógicos também foi otimizada através da manipulação dos registos do Arduino que determinam a frequência de amostragem (Margolis, 2011).

A velocidade de envio dos dados pela porta USB também foi estudada, tendo-se optado por não utilizar uma velocidade padrão, mais limitada. Ao invés selecionou-se a velocidade de transmissão de quinhentos mil bits por segundo (Arduino SA, 2015c).

Na atividade de queda livre os dados não são apresentados durante a captura. Só quando o utilizador para o processo de aquisição dos dados é que estes são inseridos na tabela e apresentados no gráfico.

Foram realizados testes para que os dados fossem atualizados em tempo real, mas os tempos de refrescamento, especialmente da tabela, são muito demorados o que provocava um atraso da recolha dos dados do *buffer* de comunicação. Quando se terminava a captura dos dados o processo de leitura continuava durante vinte a trinta segundos até parar.

Esta atividade realiza-se no 11º ano da disciplina de Física e Química A, no domínio da mecânica e é designada por AL 1.1. O objetivo geral é:

> "Determinar a aceleração da gravidade num movimento de queda livre e verificar se depende da massa dos corpos." (Ferreira et al., 2014)

Para a execução desta atividade é necessário um LDR e uma régua transparente. A régua deve ter algumas linhas igualmente espaçadas e suficientemente largas para que o LDR consiga detetar a sua passagem. Aconselha-se cinco linhas espaçadas com seis centímetros entre si e cada linha com uma largura de dois centímetros. Para um funcionamento ótimo do LDR, independentemente das condições de luminosidade ambiente, deve-se adicionar um LED direcionado ao LDR.

### 5.1.1 – VERSÃO WINDOWS

O processo de captura inicia-se com um clique no botão para iniciar, de seguida deve-se deixar cair a régua de modo a que passe entre o LED e o LDR, interrompendo a luz emitida pelo LED e captada pelo LDR. Por fim clica-se no botão para parar e o programa apresenta os dados capturados.

O programa apresenta os dados numa tabela, Figura 20, e na forma gráfica, Figura 21.

Dados capturados | Tratamento dos dados

| Tempo | Valor | Pino |
|---|---|---|
| 88 | 718 | 14 |
| 648 | 721 | 14 |
| 1280 | 721 | 14 |
| 1968 | 720 | 14 |
| 2648 | 721 | 14 |
| 3336 | 721 | 14 |
| 4024 | 721 | 14 |
| 4704 | 721 | 14 |
| 5392 | 721 | 14 |
| 6080 | 720 | 14 |
| 6764 | 721 | 14 |
| 7440 | 720 | 14 |
| 8128 | 721 | 14 |
| 8812 | 721 | 14 |
| 9488 | 721 | 14 |
| 10180 | 719 | 14 |
| 10916 | 721 | 14 |
| 11660 | 721 | 14 |
| 12400 | 719 | 14 |
| 13144 | 721 | 14 |

**Figura 20 - Tabela com os dados capturados.**

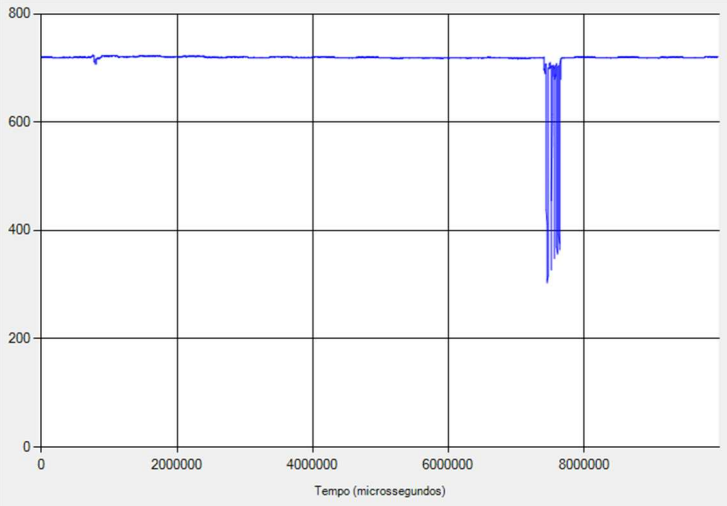



**Figura 21 - Gráfico dos dados capturados na atividade queda livre.**

O volume de dados capturados é de aproximadamente 1KHz, isto é 1000 amostras por segundo. No exemplo representado, em menos de dez segundos foram capturadas mais de onze mil amostras.

A parte interessante dos dados capturados são as linhas que assinalam a passagem da parte da régua que interrompe o LDR. Fazendo uma aproximação podemos observar melhor o resultado obtido na Figura 22.

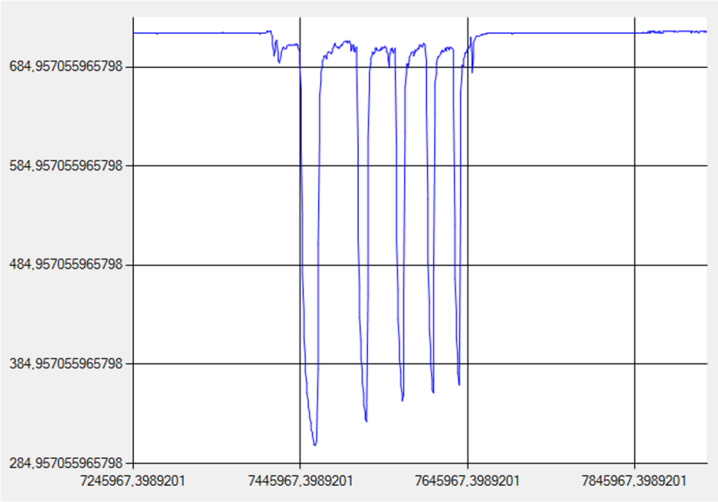



Figura 22 - Aproximação da parte dos dados que assinalam a passagem do objeto.

Cada curva descendente corresponde à parte em que o LDR fica obscurecido por uma linha. A fase ascendente refere-se à recuperação do LDR após estar obscurecido. É de realçar que a distância na régua entre cada um dos picos do gráfico inclui no só o espaço transparente entre as linhas mas também a largura da própria linha. No exemplo as linhas estão espaçadas de 6 em 6 cm e cada linha tem 2 cm, assim os cálculos são efetuados considerando a distância percorrida de 8 cm entre cada pico.

Como é possível observar no gráfico o espaço entre cada um dos picos é cada vez menor, indicando que a régua está a cair cada vez mais rápido.

Na fase de tratamento dos dados (ver Figura 23) procede-se à marcação de cada um dos pontos onde a luminosidade atingiu o valor mais baixo. A aplicação permite a deteção automática ou manual destes pontos.

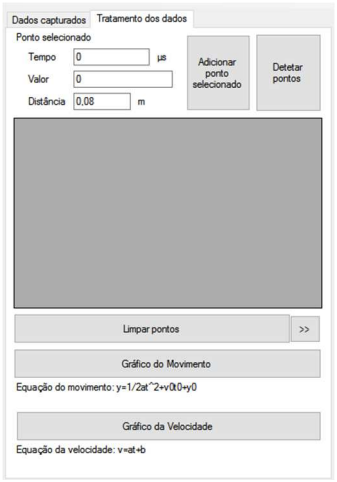



Figura 23 - Tratamento dos dados.

Manualmente o utilizador seleciona cada um dos pontos e adiciona à lista dos pontos escolhidos. Para selecionar cada ponto deve-se utilizar a ferramenta de seleção criada para a extensão dos gráficos (ver Figura 24). É também possível inserir os valores correspondentes a cada um dos pontos manualmente, mas trata-se de um processo fastidioso e propenso a inserção de erros, pois os valores têm vários dígitos.

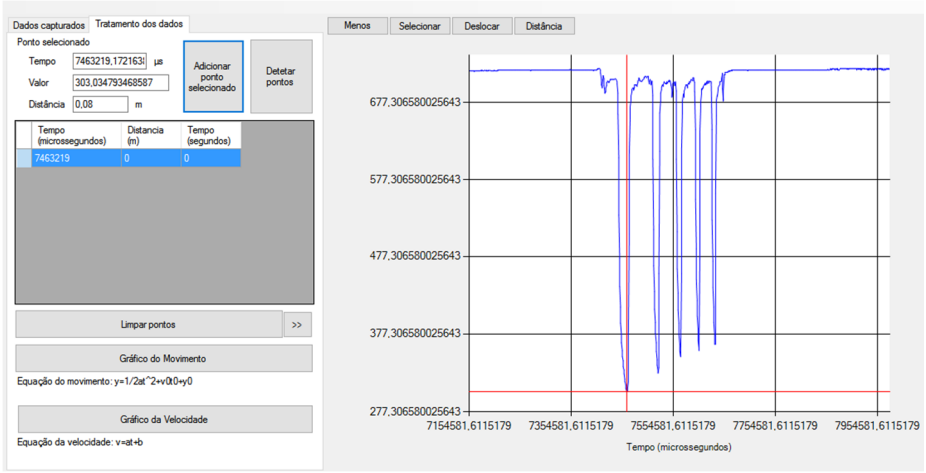



Figura 24 - Seleção manual dos pontos.

O processo de seleção automática (ver Figura 25) funciona corretamente desde que os dados capturados tenham qualidade e para isso o sensor não deve sofrer outras obstruções para além das marcas da régua.

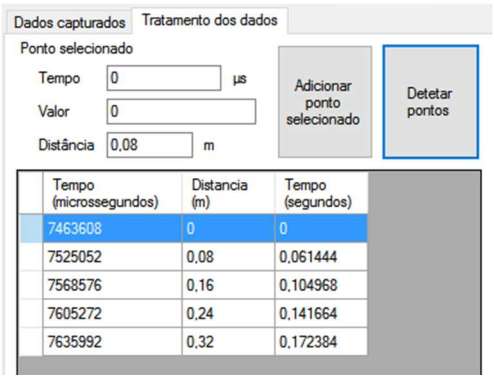



Figura 25 - Resultado da deteção automática.

O algoritmo que deteta os pontos de passagem percorre todos os dados em busca de cada um dos pontos onde a luminosidade baixou, para além de um valor considerado ruído, e recomeça a subir. Sendo selecionado o ponto imediatamente anterior à subida. Esse ponto corresponde à passagem de uma linha.

Para cada ponto escolhido, manualmente ou não, o programa procede do seguinte modo:

- O primeiro ponto é considerado como o ponto de partida, servindo de referência para os restantes;
- Para os restantes pontos é calculada a distância percorrida a partir do ponto inicial e o tempo decorrido;
- Para cada um dos pontos é também calculada a velocidade, mas este cálculo é escondido por omissão para que os discentes possam ser desafiados a executar o cálculo por si próprios.

No passo seguinte deve-se pedir aos alunos que calculem a velocidade e a aceleração da queda da régua em cada uma das marcas, considerando-se a primeira marca como o ponto inicial.

Os cálculos realizados pelos alunos podem ser confrontados com os resultados da aplicação.

Com os valores das colunas da distância e do tempo traça-se um gráfico que representa a distância percorrida em função do tempo. Este gráfico mostra que se trata de uma progressão em forma de uma curva, polinomial e não uma reta, linear. Assim os alunos devem introduzir os dados nas suas calculadoras e executar uma regressão não linear para obterem os componentes da equação do movimento.

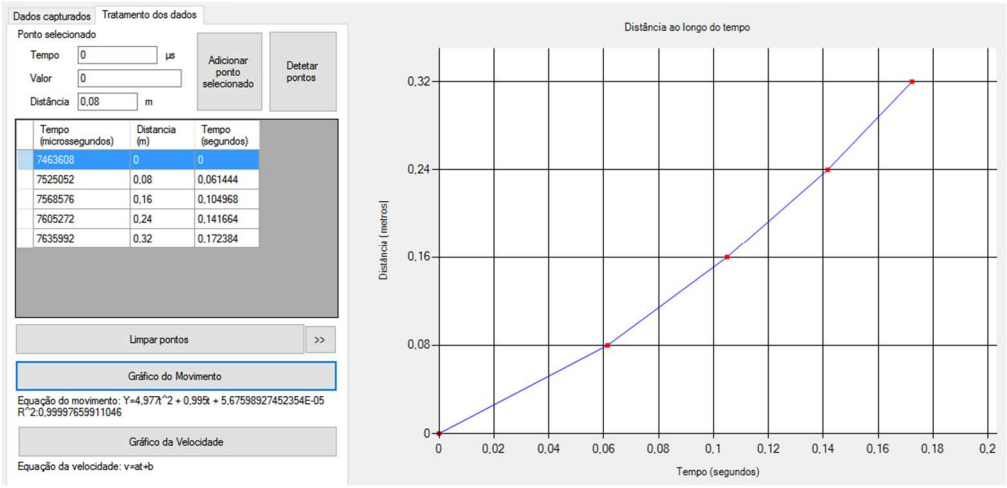



**Figura 26 - Equação e gráfico do movimento.**

O que se pretende é que os alunos obtenham, com a regressão, os componentes da equação do movimento.

A validade dos resultados obtidos pelos alunos é apurada comparando estes com os resultados esperados, ou seja, a aceleração deve ser constante e aproximada ao valor da aceleração gravítica (9,8 metros por segundo$^2$).

Por fim, executam-se os procedimentos para gerar o gráfico e a equação da velocidade. Os alunos devem utilizar, agora, as colunas do tempo e da velocidade calculada anteriormente para cada ponto. O resultado obtido será semelhante ao da Figura 27.

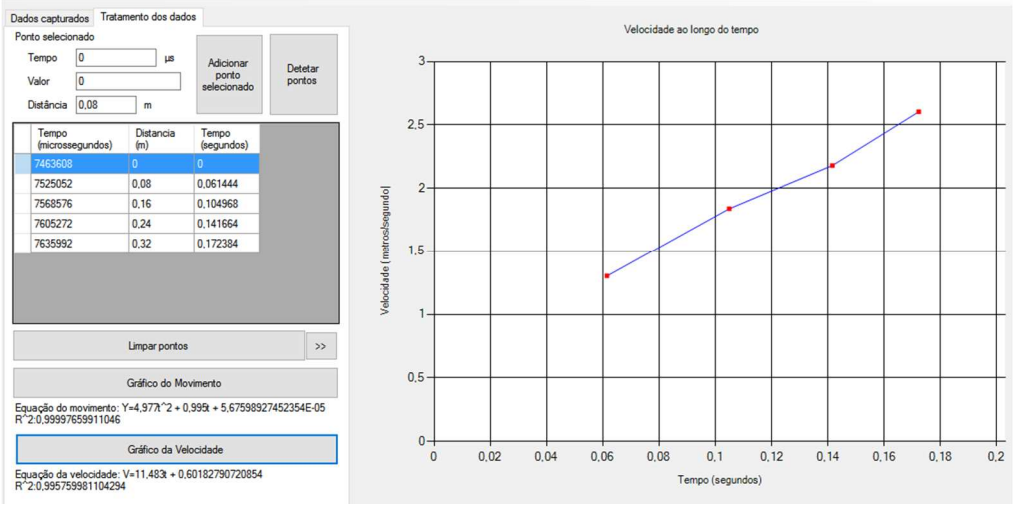



**Figura 27 - Gráfico e equação da velocidade.**

Os valores capturados e tratados podem agora ser guardados. Os gráficos gerados podem, igualmente, ser guardados. Em ambos os casos os formatos escolhidos permitem a reutilização dos documentos noutros programas. Os dados podem ser importados para uma folha de cálculo enquanto os gráficos podem ser inseridos num processador de texto.

### 5.1.2 – VERSÃO ANDROID

Na aplicação Android pode-se abrir o ficheiro dos dados capturados ou tratados pela aplicação Windows. Os dados capturados são apresentados numa lista (ver Figura 28) e existe um botão de deslizar que permite ao utilizador selecionar se pretende visualizar os dados capturados ou os dados tratados.


**Figura 28 - Dados capturados da queda.**

Com os dados capturados é possível gerar um gráfico, apresentado na Figura 29.

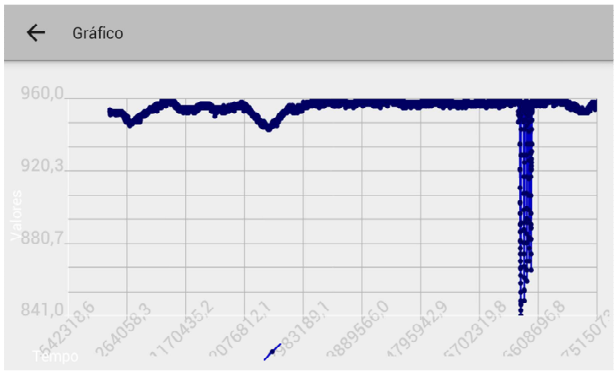



**Figura 29 - Gráfico no Android dos dados capturados na queda.**

No separador dos dados tratados deve ser introduza a distância, em metros, entre as linhas da régua (ver Figura 30). O utilizador deve por fim utilizar o botão que deteta os tempos de passagem.

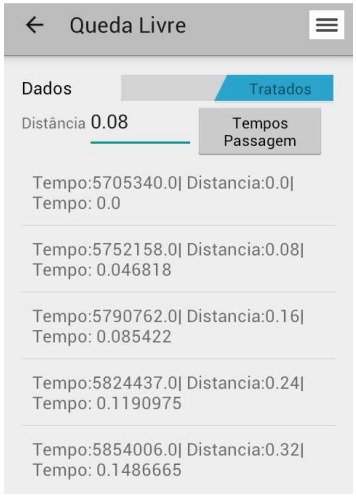



**Figura 30 - Dados tratados da queda livre no Android.**

Os dados apresentados devem permitir aos discentes aplicar os seus conhecimentos para calcular a velocidade instantânea de passagem da régua em cada um dos pontos. Estes cálculos não são apresentados.

Com os resultados obtidos os alunos devem poder calcular a regressão dos dados para obter os parâmetros das equações dos movimentos (ver Figura 31) e da velocidade (ver Figura 33) bem como gráficos representativos da velocidade e da deslocação (ver Figura 32).


**Figura 31 - Equação do movimento no Android.**

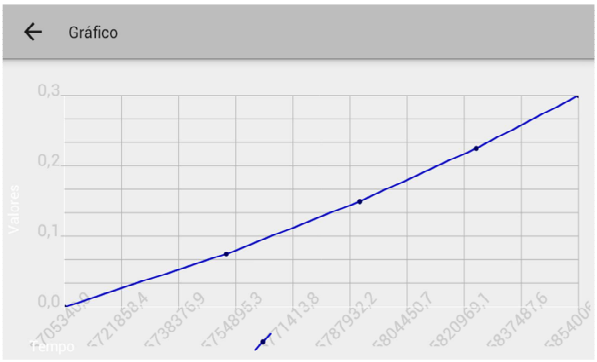



**Figura 32 - Gráfico do movimento ao longo do tempo no Android.**


**Figura 33 - Equação da velocidade no Android.**

Por fim o utilizador pode optar por limpar os dados por forma a repetir as operações sem ter de fechar a actitivy.

## 5.2 – ATIVIDADE LABORATORIAL BOLA SALTITONA

A atividade Bola Saltitona foi testada com recurso a bolas com diferentes caraterísticas obtendo-se assim resultados díspares. Como era de se esperar a bola com maior elasticidade, a bola de borracha obteve um coeficiente de restituição superior do que a bola com menor elasticidade, a bola de futebol (Figura 34).

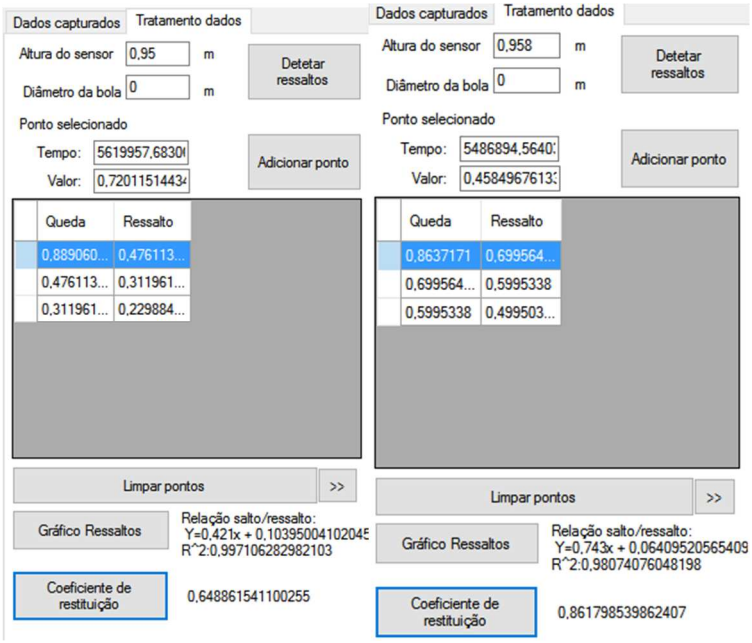



**Figura 34 - Comparação dos resultados obtidos (bola futebol e bola de borracha).**

Os testes realizados com esta atividade mostraram valores corretos com o sensor instalado a uma distância do chão entre 60 centímetros até uma altura de 1,20 metros, sendo óbvio que com alturas superiores a bola fará mais ressaltos.

Esta atividade, designada de AL 1.2 no programa da disciplina de Física e Química A, deve ser realizada no 10º ano e o seu objetivo geral é:

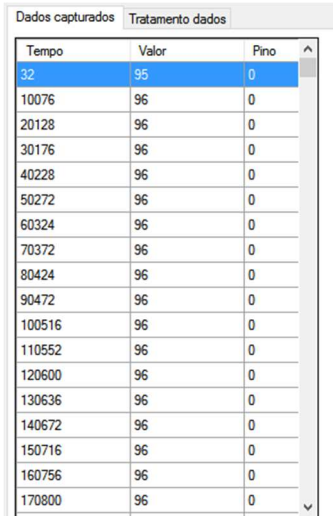
> "Investigar, com base em considerações energéticas (transformações e transferências de energia), o movimento vertical de queda e de ressalto de uma bola." (Ferreira et al., 2014).

Nesta atividade utiliza-se um sensor de distância, como por exemplo o HC-SR04. Este deve ser colocado numa posição suficientemente elevada que permita que a bola ao cair ressalte pelo menos três vezes.

### 5.2.1 – VERSÃO WINDOWS

O processo de captura dos dados é semelhante ao da atividade queda livre, um clique no botão para iniciar e o programa começa a apresentar os dados capturados pelo sensor em tempo real no gráfico. De seguida aproxima-se a bola do sensor, sem tocar neste, e deixa-se cair. A bola deve ressaltar dentro da área de captura do sensor sem nenhuma interferência externa.

Ao clicar no botão para parar os dados são apresentados, mais uma vez, numa tabela (ver Figura 35) e num gráfico (ver Figura 36).




| Tempo | Valor | Pino |
|---|---|---|
| 32 | 95 | 0 |
| 10076 | 96 | 0 |
| 20128 | 96 | 0 |
| 30176 | 96 | 0 |
| 40228 | 96 | 0 |
| 50272 | 96 | 0 |
| 60324 | 96 | 0 |
| 70372 | 96 | 0 |
| 80424 | 96 | 0 |
| 90472 | 96 | 0 |
| 100516 | 96 | 0 |
| 110552 | 96 | 0 |
| 120600 | 96 | 0 |
| 130636 | 96 | 0 |
| 140672 | 96 | 0 |
| 150716 | 96 | 0 |
| 160756 | 96 | 0 |
| 170800 | 96 | 0 |

Figura 35 - Dados capturados pelo sensor HC-SR04.

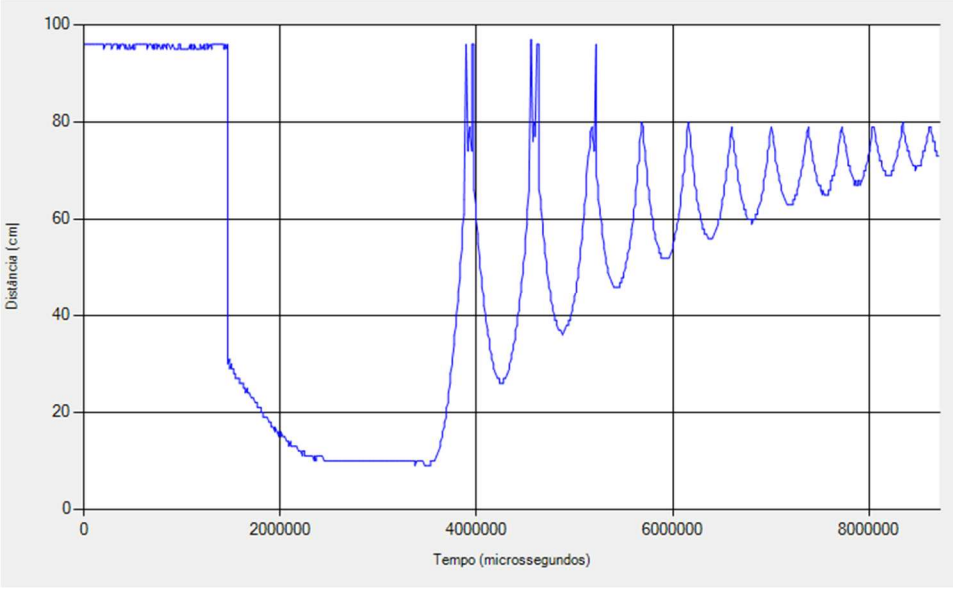



Figura 36 - Gráfico da queda e dos ressaltos.

O próximo passo é tratar os dados capturados. No separador criado para o efeito, deve-se preencher os campos referentes à altura do sensor e o diâmetro da bola. Caso se opte por preencher a altura do sensor até à bola no chão, em vez da altura ao chão, o diâmetro da bola

deve ser zero. Ao selecionar o separador de tratamento de dados o programa calcula a média dos cinco primeiros valores capturados assumindo esta como a altura do sensor ao chão (ver Figura 37).


Figura 37 - Interface para tratamento de dados.

O tratamento dos dados prossegue com a indicação da altura da queda e o respetivo ressalto. Para isso é possível utilizar a opção automática, em que o programa tenta encontrar esses pontos, ou manualmente, selecionando cada um no gráfico e adicionando à lista.

O processo de tratamento de inserção dos pontos selecionados é tratado da seguinte forma:

- O primeiro ponto selecionado é definido como sendo a altura da queda (Figura 38);
- O ponto seguinte é a altura do ressalto do ponto anterior e a altura da queda do ponto seguinte (Figura 39);

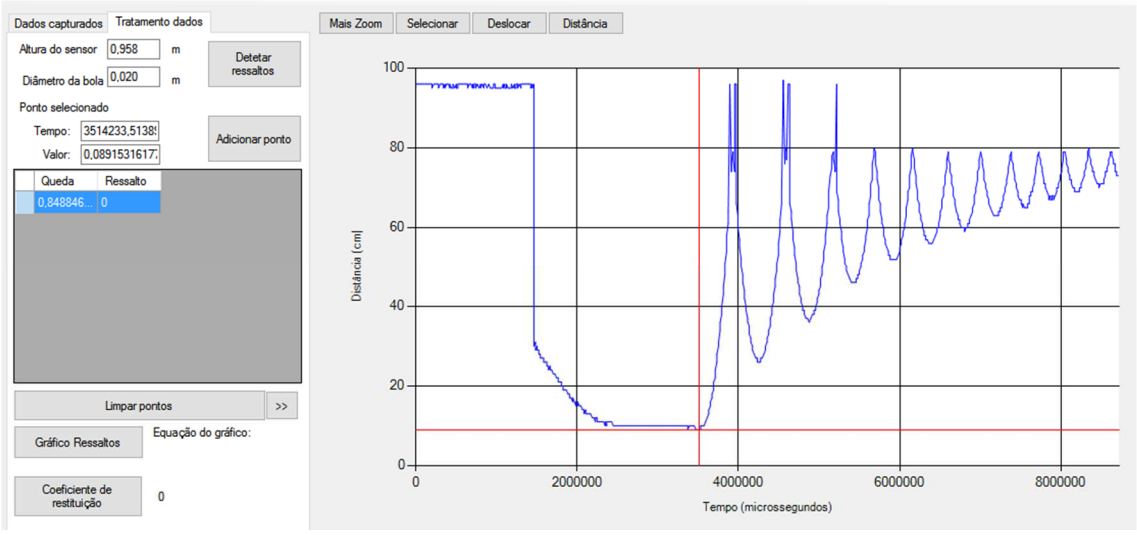



Figura 38 - Ponto de queda.

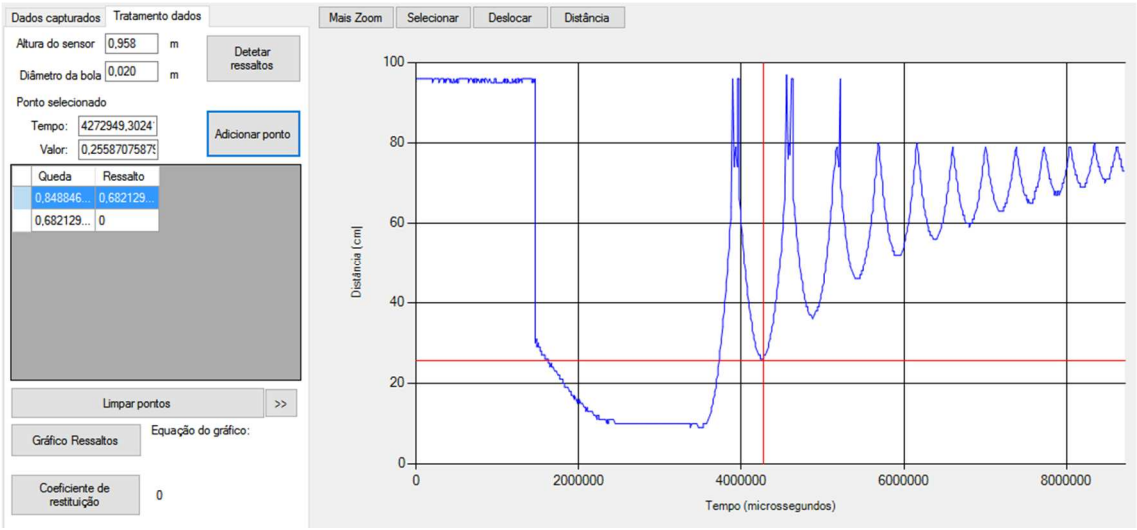



Figura 39 - Ponto de ressalto e queda.

No final do processo de definição dos pontos de queda e ressalto é possível calcular o coeficiente de restituição de cada queda e ressalto e ainda a percentagem de energia dissipada (Caldeira, Quadros, & Machado, 2015).

Com uma regressão linear dos dados é possível obter o declive da reta (8) com o qual calcula-se o valor do coeficiente de restituição.

$$m = \frac{\Delta h_{ressalto}}{\Delta h_{queda}} \quad (8)$$

O cálculo do valor do coeficiente de restituição é obtido com a equação (9).

$$e = \sqrt{\frac{\Delta h_{ressalto}}{\Delta h_{queda}}} \quad (9)$$

O resultado obtido será semelhante ao da Figura 40.

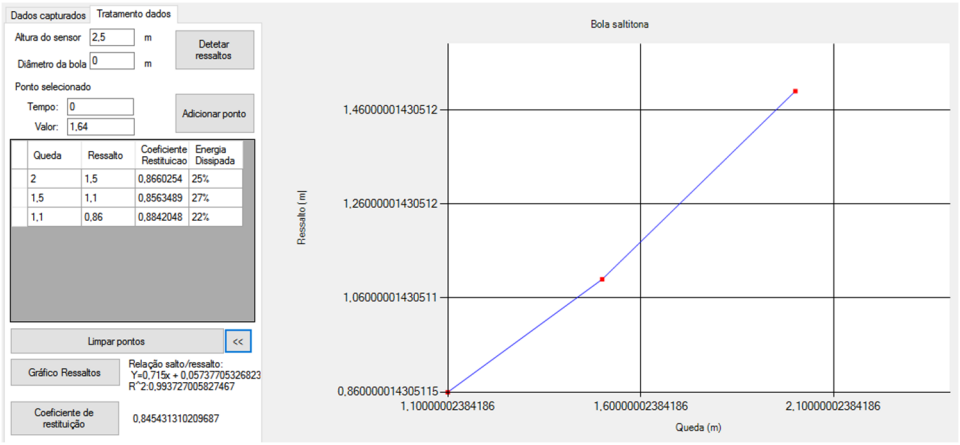



**Figura 40 - Gráfico e equação da queda e ressalto.**

A experiência deve ser realizada com diversos tipos de bolas e, se possível, com diferentes pisos, com propriedades distintas, por forma a se obterem diferentes valores de coeficientes de restituição.

### 5.2.2 – VERSÃO ANDROID

Para receber os dados a aplicação Windows deve ativar a opção de envio via Bluetooth.

Os dados capturados são apresentados numa lista (Figura 41), a partir da qual o utilizador pode gerar um gráfico (Figura 44).

Ao deslocar o botão para tratar os dados a aplicação calcula a média dos primeiros cinco valores capturados e utiliza esta como a altura do sensor, sendo possível o utilizador alterar este valor. De seguida deve ser introduzido o diâmetro da bola, caso a altura introduzida seja do sensor até à bola no chão o diâmetro deve ser zero (Figura 42).

O tratamento dos dados passa por selecionar os pontos de queda e ressalto (Figura 43). Este processo pode ser automaticamente calculado pela aplicação, com o clique no respetivo botão, ou realizado manualmente pelo utilizador a partir do gráfico dos dados (Figura 44).


**Figura 41 - Dados capturados na atividade Bola Saltitona no Android.**

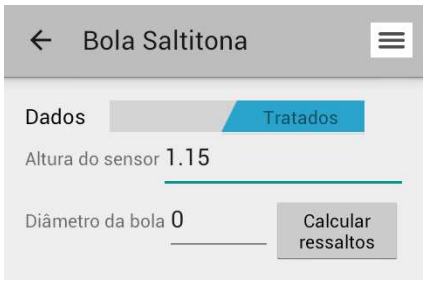



**Figura 42 - Tratamento dos dados da Bola Saltitona no Android.**

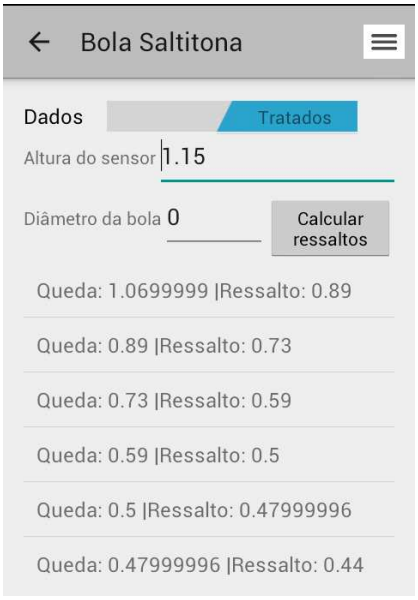



**Figura 43 - Dados tratados automaticamente no Android.**

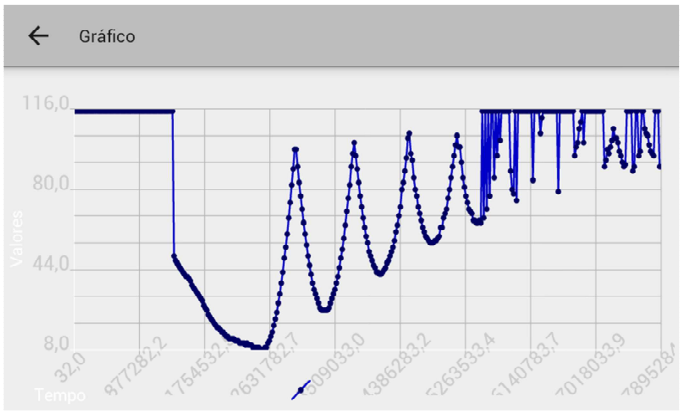



Figura 44 - Gráfico da queda e ressaltos no Android.

A seleção manual dos pontos de queda e ressalto apresenta os dados selecionados com uma linha a realçar o ponto escolhido no gráfico. O utilizador deve confirmar estes valores a partir de uma janela (Figura 45).

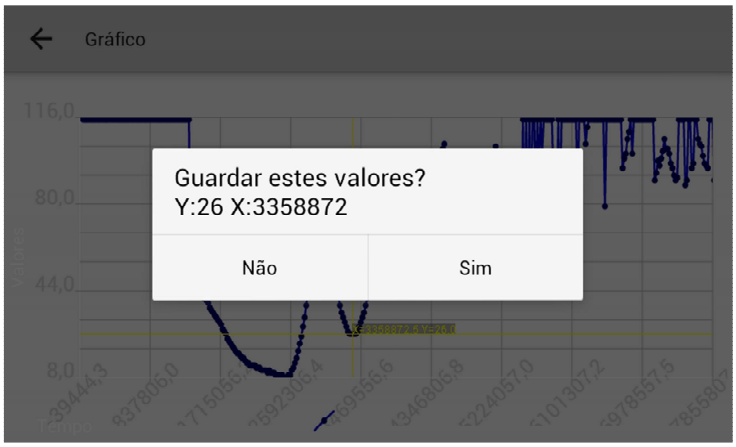



Figura 45 - Seleção de ponto de queda e ressalto no Android.

Os valores selecionados são apresentados ao utilizador quando este retorna à atividade anterior (Figura 46).


Figura 46 - Ponto de queda e ressalto selecionado manualmente no Android.

Com os pontos de queda e ressalto selecionados é possível traçar o gráfico dos ressaltos, que mostra a relação entre a altura da queda e o respetivo ressalto (Figura 47).

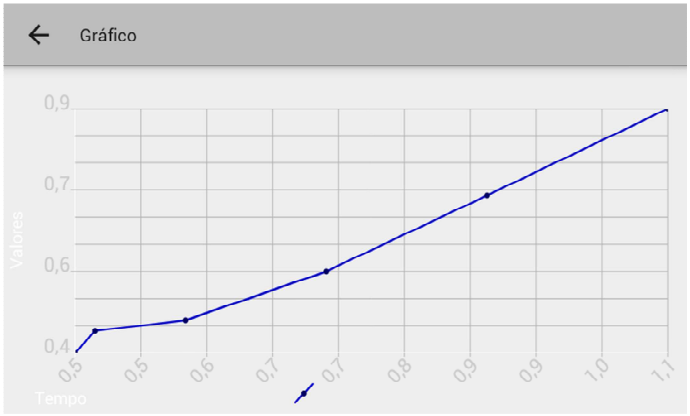



Figura 47 - Gráfico da queda e ressaltos no Android.

O próximo passo passa por calcular o declive da reta, para isso é necessário fazer a regressão linear simples dos dados e determinar a equação dos ressaltos (Figura 48).


Figura 48 - Equação dos ressaltos no Android.

Com o declive da reta calcula-se o coeficiente de restituição da bola (Figura 49).


Figura 49 - Coeficiente de restituição no Android.

Os alunos devem ser incentivados a repetir a experiência com bolas de diferente elasticidade e comparar os resultados.

## 5.3 – ATIVIDADE LABORATORIAL PLANO INCLINADO

A atividade Plano Inclinado é a atividade que requer mais sensores e componentes. A atividade foi executada com quatro sensores LDR, mas podem ser utilizados até seis.

Como plano foi utilizada uma pista de carrinhos com as resistências fotossensíveis inseridas diretamente no plano de modo a que o carrinho, ao descer o plano, obscurece os sensores ao passar-lhes por cima.

O programa de Física e Química A designa esta atividade de AL 1.1 e prevê que se realiza no 10º ano de escolaridade. O objetivo geral é:

> "Estabelecer a relação entre variação de energia cinética e distância percorrida num plano inclinado e utilizar processos de medição e tratamento estatístico de dados." (Ferreira et al., 2014)

O conhecimento que a humanidade tinha dos movimentos era baseado no senso comum, desenvolvido por Aristóteles, e ainda hoje é tido como verdade, por grande parte da população.

Aristóteles defendia que a força que impelia um objeto dependia da sua composição, isto é, um objeto descia um plano inclinado com uma velocidade maior ou menor em função da sua massa (Costa et al., 2008).

Galileu contestou esta conceção com as suas famosas experiências realizadas na Torre de Pisa e com diversos planos inclinados (Costa et al., 2008). Segundo Galileu, o espaço percorrido na queda livre é proporcional ao quadrado dos tempos e a constante de proporcionalidade é a aceleração gravítica.

Nesta atividade pretende-se determinar a energia cinética (Rodrigues & Dias, 2007) associada a um carrinho em movimento pelo plano inclinado com base na fórmula apresentada na equação (10).

$$E_c = \frac{1}{2}mv^2 \qquad \textbf{(10)}$$

Nesta equação $m$ representa a massa e $v$ a velocidade.

Um corpo que se move possui energia cinética cujo valor depende da sua massa e da velocidade a que se desloca (Caldeira et al., 2015).

Para esta atividade são necessários vários sensores LDR para detetar a passagem de um carrinho a descer um plano inclinado. Assim é necessário receber dados de vários pinos em simultâneo apresentando os valores recebidos num gráfico com várias séries de dados, uma por cada pino (Figura 50).

A atividade foi criada de modo a que o utilizador escolha quantos LDR pretende utilizar. Estes devem ser ligados nos pinos analógicos de forma sequencial começando sempre pelo pino A0.

Ao distribuir os LDR pelo plano inclinado é muito importante que estes sejam colocados todos à mesma distância uns dos outros, isto é, o espaçamento entre os sensores deve ser exatamente igual. Este espaçamento é inserido no programa para que seja calculada a velocidade de deslocamento do carro ao descer o plano inclinado.

### 5.3.1 – VERSÃO WINDOWS

Não sendo necessária uma velocidade de captura tão elevada como na atividade de queda livre, pois o carrinho não atinge uma velocidade tão alta como o objeto em queda. Ainda assim, existe um elevado número de dados a serem enviados pelo Arduino devido ao número de sensores em uso. Para minimizar o número de dados optou-se por tirar partido da funcionalidade de filtragem incluída no protocolo (ver secção 4.1.2 – Comandos Implementados).

A opção de filtragem de dados utilizada, para além de reduzir o volume de dados trocados entre o Arduino e o computador, também reduz o ruído nos dados recebidos.

Existe também uma opção que, se selecionada, emite um som assinalando a passagem do carrinho por cada sensor no plano inclinado. Esta opção é inspirada no plano inclinado de Galileu (Costa et al., 2008).

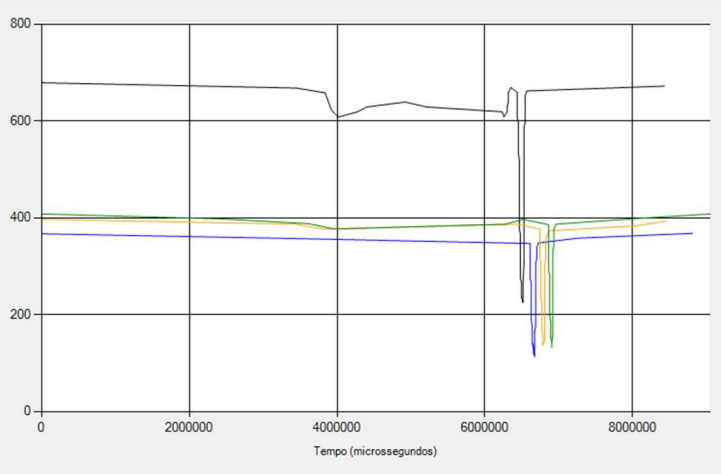



**Figura 50 - Dados capturados no plano inclinado.**

É possível executar esta atividade só com um LDR, mas implica a utilização da régua transparente com as linhas instalada em cima do carro.

Para o tratamento dos dados desta atividade laboratorial é, agora, necessário inserir a massa do carrinho (ver Figura 51). Ainda em relação ao carrinho a utilizar é muito importante que este tenha o mínimo de atrito possível para que os cálculos não sejam influenciados por fatores externos.

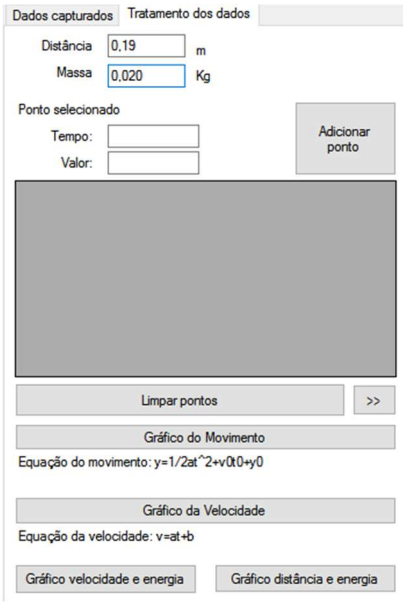



Figura 51 - Tratamento dos dados do plano inclinado.

Por fim utiliza-se a ferramenta de seleção do gráfico para marcar e adicionar os pontos de passagem do carrinho pelo sensor. É muito importante que esta seleção seja consistente, optando-se por assinalar o ponto inicial, quando o carro começa a tapar o sensor, ou o ponto final, quando o carro já passou sobre o sensor.

O tratamento dado aos pontos selecionados é baseado nos seguintes princípios:

- O primeiro ponto é o ponto inicial a partir do qual são executados os restantes cálculos;
- Para cada um dos restantes pontos adicionados calcula-se a distância percorrida, o tempo decorrido em segundos e a velocidade e energia cinética.

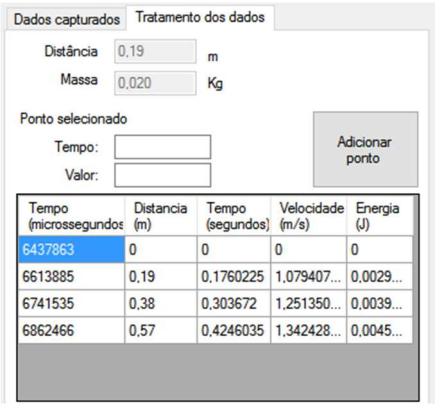



Figura 52 - Dados do plano inclinado tratados.

As colunas da velocidade e energia estão escondidas por omissão para que os alunos possam fazer os cálculos, podendo no final comparar os seus resultados com as soluções da aplicação.

Com estes dados é possível traçar quatro gráficos:

- Gráfico do movimento, com as colunas do tempo em segundos e da distância;
- Gráfico da velocidade, com as colunas do tempo em segundos e da velocidade;
- Gráfico da energia em função da velocidade;
- Gráfico da energia em função da distância.

Todos os gráficos podem ser guardados bem como os dados capturados e tratados.

### 5.3.2 – VERSÃO ANDROID

A captura dos dados depende da ativação do envio via Bluetooth através da aplicação Windows.

Ao ativar o modo de captura na atividade os dados recebidos são apresentados numa lista a qual apresenta o tempo de captura, em microssegundos, o valor recebido e o pino de origem dos dados (Figura 53).


**Figura 53 - Dados capturados no plano inclinado no Android.**

Os dados capturados podem, agora, ser apresentados num gráfico. Os dados recebidos provém de vários pinos, sendo cada pino apresentado com uma cor diferente (Figura 54).

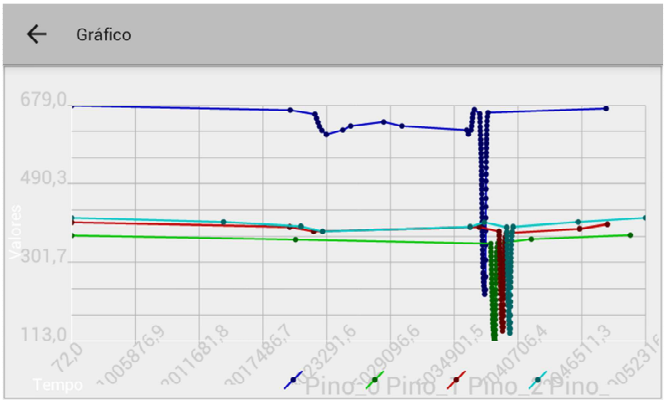



**Figura 54 - Gráfico dos dados capturados no plano inclinado.**

Para iniciar o processo de tratamento dos dados é necessário introduzir a distância entre cada sensor no plano inclinado bem como o valor da massa do carrinho a percorrer o plano inclinado (Figura 55).


**Figura 55 - Tratamento dos dados do plano inclinado no Android.**

Para apurar a velocidade do carrinho a descer o plano inclinado é necessário selecionar o ponto de passagem em cada LDR, para isso é útil que o utilizador faça *zoom* sobre os dados do gráfico (Figura 56).

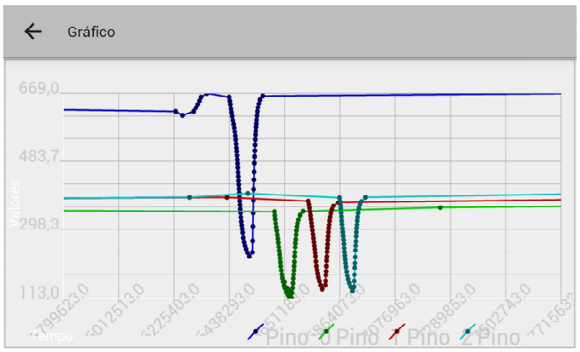



Figura 56 - Zoom sobre os dados capturados.

A seleção dos pontos no gráfico apresenta ao utilizar uma linha a indicar a posição precisa do ponto selecionado e uma janela onde o utilizador deve confirmar se o ponto é de facto para guardar (Figura 57).

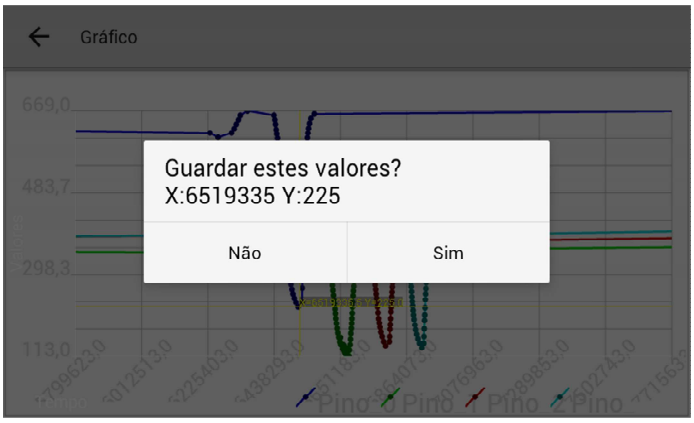



Figura 57 - Ponto selecionado nos dados do plano inclinado.

Terminado o processo de seleção dos pontos no gráfico, regressa-se à atividade anterior que agora lista os pontos selecionados, tratados com dados que devem permitir aos alunos calcular a velocidade de passagem em cada ponto (Figura 58).


Figura 58 - Dados tratados no plano inclinado.

Com os resultados da velocidade instantânea calculados os alunos podem determinar a energia cinética. Estes cálculos não são apresentados na aplicação móvel de modo que os alunos tenham de executar as operações matemáticas envolvidas. Os resultados podem ser validados pela aplicação principal, no Windows.

A utilização de carrinhos com diferentes massas, por exemplo adicionando pesos, é possível comprovar, utilizando a equação do movimento, se existem ou não diferentes velocidades (Figura 59).


Figura 59 - Equação do movimento no plano inclinado.

Existe, também, uma opção que apresenta a equação da velocidade (Figura 60).


Figura 60 - Equação da velocidade no plano inclinado.

## 5.4 – PLAYGROUND

O Playground é uma funcionalidade exclusiva da aplicação Windows sendo também a única que não dispõe da opção que ativa o envio dos dados do Arduino para o dispositivo Android via Bluetooth.

Esta opção da aplicação permite a execução de outras atividades laboratoriais, para além das previstas nas opções enunciadas anteriormente.

Nesta funcionalidade é possível selecionar os pinos analógicos a capturar, bem como o sensor de distância. É ainda possível indicar se a captura dos dados será filtrada e em caso afirmativo qual o valor do filtro. A Figura 61 apresenta todas as opções disponíveis.

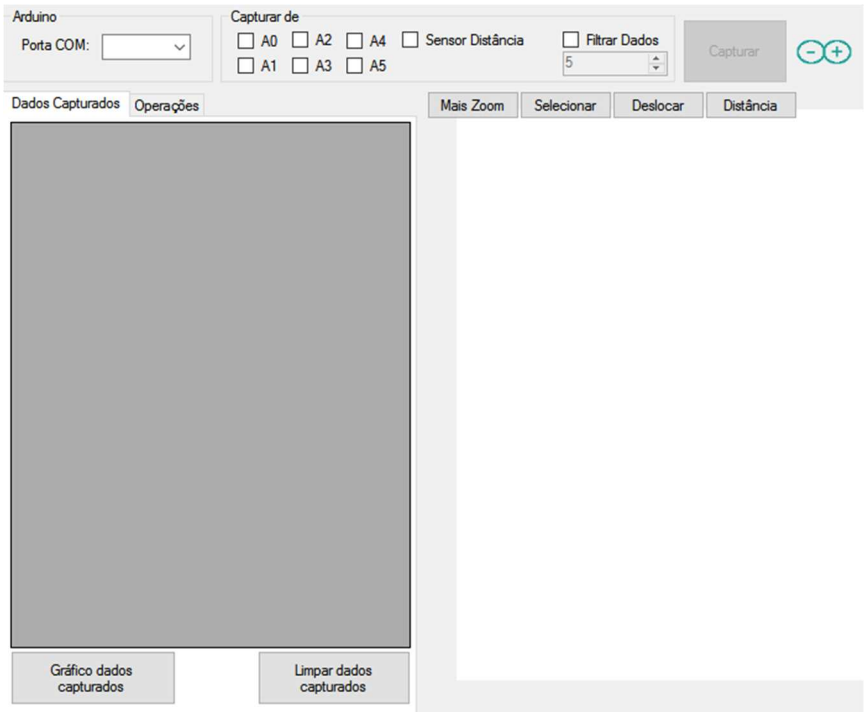



Figura 61 - Interface Playground.

Com os dados capturados é possível fazer um conjunto de operações simples como calcular a média, moda ou a amplitude dos dados, mas também adicionar colunas calculadas à tabela de dados. Na Figura 62 apresenta-se um exemplo em que é adicionada uma coluna que permite converter o valor recebido de um sensor de temperatura analógico, neste caso o TMP36, no correspondente valor da temperatura em Celcius.

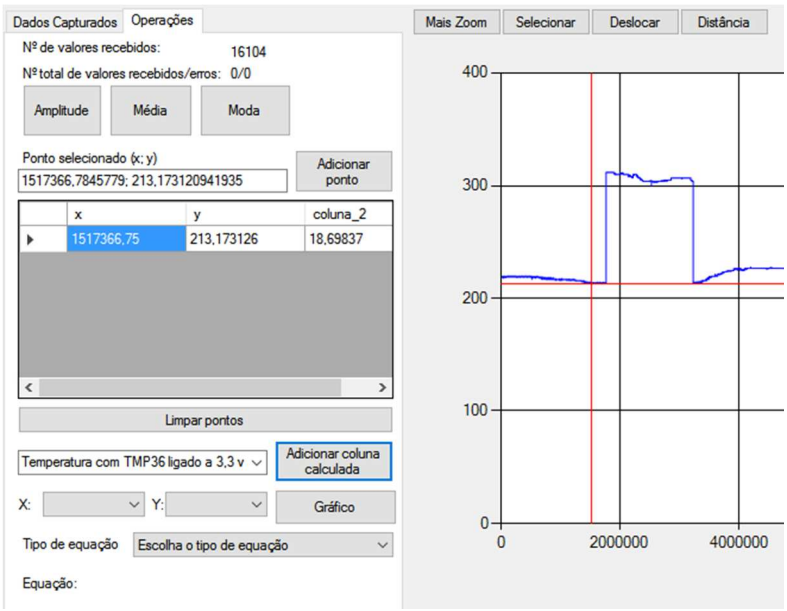



Figura 62 - Coluna da temperatura calculada.

Estas colunas são essenciais uma vez que os valores devolvidos pelo Arduino não são tratados, estando compreendidos entre 0 e 1023. É através destas colunas que se pode atribuir algum significado a estes para que possam ser interpretados pelo utilizador em função do sensor utilizado.

Na aplicação existem fórmulas pré-definidas para:

- Conversão do valor em volts numa escala de 5 volts;
- Conversão do valor em volts numa escala de 3,3 volts;
- Conversão do valor recebido de um sensor de temperatura TMP36 ligado a uma fonte de 3,3 volts;
- Conversão do valor recebido de um sensor de temperatura TMP36 ligado a uma fonte de 5 volts;
- Conversão da coluna de tempo em segundos.

Com estas fórmulas é possível utilizar sensores de temperatura TMP36 em experiências como por exemplo na atividade laboratorial AL 3.2, prevista para o 10º ano de escolaridade em que se pretende “Determinar a capacidade térmica mássica de um material” (Ferreira et al., 2014).

Podendo o Arduino receber até 5 volts nos pinos analógicos, esta opção permite estudar as caraterísticas de uma pilha (atividade laboratorial 2.1, programada para o 10º ano) ou estudar a energia voltaica fornecida por um painel fotovoltaico (atividade laboratorial 3.1 do 10º ano), desde que não se ultrapassa o limite máximo permitido pelo Arduino.

Para além destas fórmulas incluídas, o utilizador pode inserir uma à sua escolha desde que utilize uma notação matematicamente correta, podendo incluir nas fórmulas dos cálculos os nomes das colunas da tabela como variáveis.

Todos os cálculos podem ser efetuados sobre valores inseridos manualmente pelo utilizador, permitindo que este execute regressões lineares ou não lineares sobres quaisquer dados que possam ser úteis ao utilizador, sendo o tipo de regressão a realizar escolhido pelo utilizador.

Para a execução do cálculo da regressão bem como para traçar o gráfico o utilizador deve associar duas colunas da tabela com os eixos x e y, por omissão são selecionadas as colunas x para o eixo dos x e y para o eixo dos y.

Na Figura 63 apresenta-se uma experiência que foi realizada com dois sensores de temperatura (TMP36) com uma coluna calculada que converte o valor devolvido pelo Arduino em temperatura, em graus Celsius. Para a realização desta experiência envolveu-se um dos sensores com uma pelicula de alumínio e comparou-se as temperaturas dos dois sensores ao longo do tempo, tendo sido utilizado um candeeiro como fonte de calor.

Com os dados capturados podem-se retirar conclusões quanto ao comportamento térmico do da pelicula de alumínio.

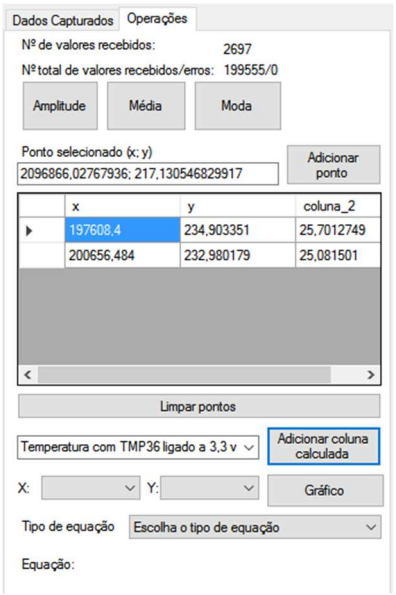



Figura 63 - Playground com dados de dois TMP36.

## 5.5 – ATIVIDADE ACELERÓMETRO

Esta atividade, exclusiva da aplicação móvel, tira partido do acelerómetro do dispositivo.

O acelerómetro mede a aceleração aplicada ao dispositivo, incluindo a força gravítica (Google, n.d.-b). A maior parte dos dispositivos Android dispõe de um módulo de acelerómetro e a sua utilização é simples, pois estes utilizam o sistema de coordenadas cartesianas.

Caso o dispositivo não disponha de um acelerómetro, é apresentada uma mensagem ao utilizador.

A escala utilizada pelo sensor é também uma escala padrão do sistema internacional de unidades, em $m/s^2$.

Nesta funcionalidade é possível visualizar o valor da aceleração em tempo real, sem capturar os valores (Figura 64).

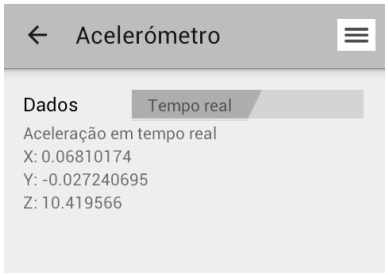



Figura 64 - Dados do acelerómetro em tempo real.

Ao escolher a opção de captura dos dados é apresentada uma mensagem ao utilizador que indica que os dados estão agora a ser guardados (Figura 65).

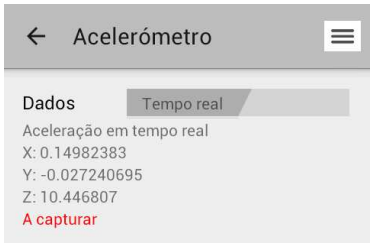



Figura 65 - A capturar dados do acelerómetro.

Durante a fase de captura dos dados é a altura certa para executar a experiência em causa, pode-se por exemplo deixar cair o dispositivo ou fazer um estudo da aceleração num *Bungee Jump*, prendendo o dispositivo a elásticos.

Esta atividade depende da capacidade do dispositivo, caso este seja limitado serão capturados poucos pontos durante a queda, sendo mais difícil realizar o estudo. No exemplo apresentado na Figura 66 o dispositivo só conseguiu capturar dois pontos durante a queda.

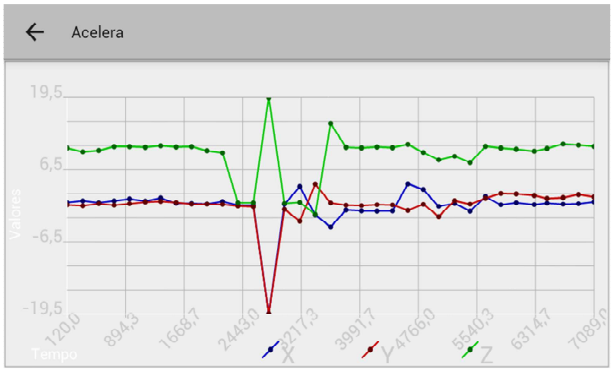



Figura 66 - Dados capturados durante a queda.

Os dados capturados são apresentados ao utilizador numa lista (Figura 67) que inclui os valores da aceleração em cada um dos eixos (x, y e z) e o tempo (em microssegundos).

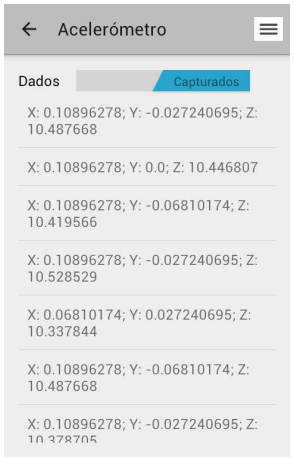



Figura 67 - Dados capturados do acelerómetro.

Com o dispositivo em repouso é possível verificar o valor da gravidade (Figura 68).

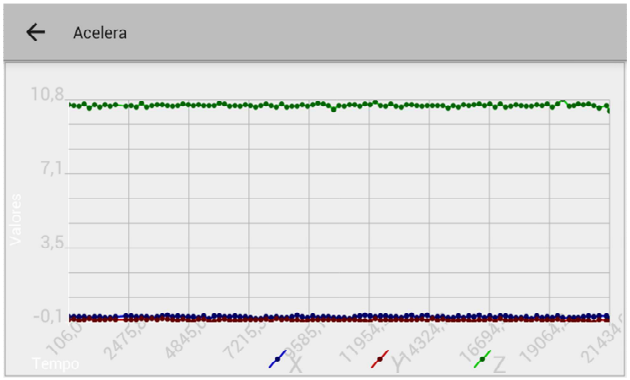



Figura 68 - Gráfico dos dados capturados do acelerómetro.

Com esta atividade podem-se fazer variações à experiência utilizando um elástico com mais ou menos comprimento ou elasticidade, utilizando um peso ou associando à captura destes dados com um sensor de distância por debaixo do dispositivo preso ao elástico de forma a poder comprar a distância percorrida com as acelerações sofridas.

Outra utilização pode ser o estudo das acelerações de um pendulo ou de um carrinho de choque.

A força centrípeta também pode ser estudada sendo, no entanto, necessário um aparato a que se possa prender o dispositivo móvel e que promova um movimento circular.

## 5.6 – TESTE DE UTILIZAÇÃO

Qualquer projeto de desenvolvimento de software deve incluir testes de usabilidade. Este projeto não foi exceção a esta regra, tendo-se realizado um teste com um grupo de alunos em sala de aula.

O teste decorreu com as limitações impostas pelo calendário. O ano escolar estava prestes a terminar e não era possível protelar o teste para que fossem utilizadas versões mais maduras das aplicações.

Nesta sessão não foi possível testar a aplicação móvel, pois que, à data da sessão, o estado de desenvolvimento desta era ainda muito frágil e depois, não foi possível repetir a sessão devido ao términus das atividades letivas.

A atividade foi realizada com um turno de quinze alunos, de uma turma do 10º ano composta por 29 alunos. O projeto foi apresentado aos alunos e foi feito um enquadramento para que estes compreendessem que o que se pretendia não era avalia-los a eles, mas sim avaliar a plataforma. Os alunos foram deixados à vontade para que não se sentissem constrangidos em criticar ou apresentar sugestões.

Em termos pedagógicos não foi possível fazer uma verdadeira avaliação da plataforma. Apesar de se ter realizado uma sessão de utilização com um grupo de quinze alunos, e de esta ter decorrido com sucesso, não se pode daí retirar conclusões definitivas quanto à validade das aplicações sem que sejam realizados mais testes.

Ainda assim, foi interessante assistir à reação muito positiva dos alunos ao facto do aparato de captura ser aberto, tendo estes participado na instalação dos sensores e ficando assim a conhecer melhor cada um dos componentes.

Inicialmente foi feita uma apresentação do Arduino e dos sensores que iriam ser utilizados, bem como os cuidados a ter na sua manipulação. Depois foi instalado o programa no computador da sala de aula e feita a instalação do driver do Arduino. De seguida foram apresentados os objetivos da primeira atividade a realizar, a Queda Livre.

### 5.6.1 ATIVIDADE QUEDA LIVRE

Os alunos conseguiram instalar o LDR necessário para a captura do dados bem como fazer a ligação entre a aplicação Windows e o Arduino. O processo de captura dos dados foi simples e decorreu bem, tendo os alunos repetido o processo diversas vezes, para comprovar os resultados obtidos.

Os resultados obtidos na atividade Queda Livre foram os esperados para a experiência (Figura 69). Os alunos repetiram os procedimentos diversas vezes por forma a comprovar que os dados capturados podiam variar em função do modo como deixavam cair a régua. Como se pode comprovar pela Figura 69 é possível verificar a passagem das linhas da régua durante a queda.

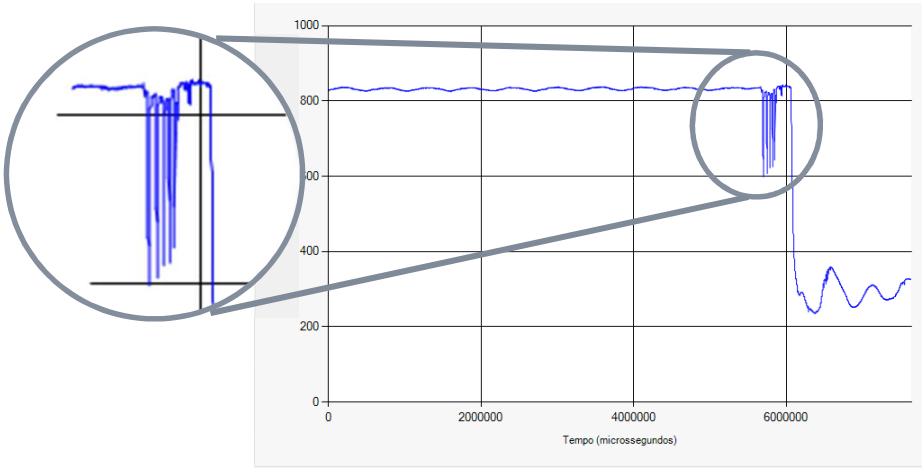



**Figura 69 - Dados capturados durante um teste.**

Com os estes dados foi possível obter resultados dentro dos valores aceitáveis para a gravidade e para a velocidade da queda (Figura 70).


**Figura 70 - Resultados produzidos durante um teste.**

Como se pode verificar pela Figura 70, o valor obtido para a aceleração da gravidade foi de 10,2 metros por segundo$^2$, sendo que o valor esperado seria de 9,8 metros por segundo$^2$, ou seja conseguiu-se um valor com uma margem de erro de 4%. A atividade foi repetida por vários grupos de alunos que, sempre que executaram os procedimentos da atividade corretamente, obtiveram dados consistentes.

### 5.6.2 BOLA SALTITONA

Por fim, procedeu-se à execução da atividade Bola Saltitona. Desta vez os alunos utilizaram um sensor de ultrassons. A utilização da aplicação foi simplificada pela consistência da interface e dos procedimentos entre as diferentes atividades laboratoriais.

A experiência foi realizada duas vezes com diferentes alturas de queda da bola.

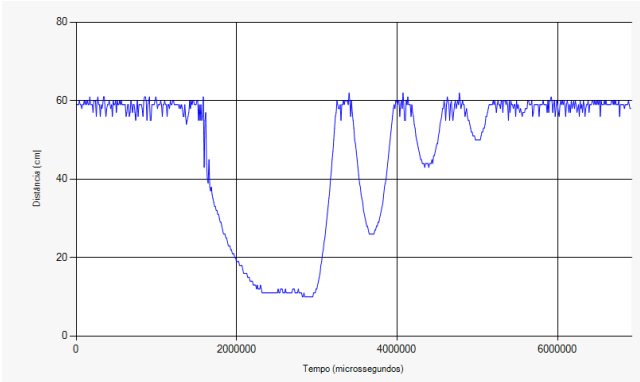



**Figura 71 - Dados capturados com a bola a cair de 60 cm de altura.**

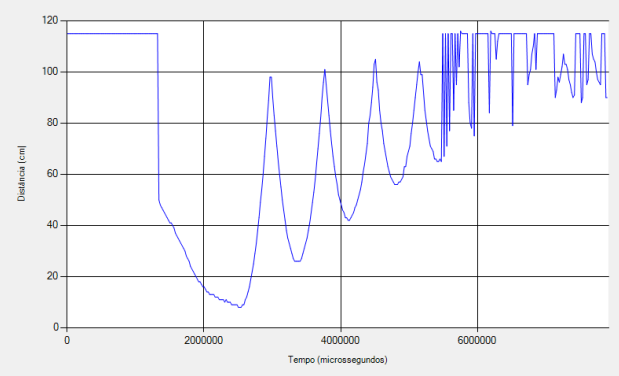



**Figura 72 - Dados capturados com a bola a cair de 120 cm de altura.**

Em ambas as execuções, os alunos conseguiram selecionar os pontos de queda e ressalto da bola, tendo sido fácil para estes produzir um gráfico dos ressaltos, em função da altura da queda, bem como obter o coeficiente de restituição (Figura 73).

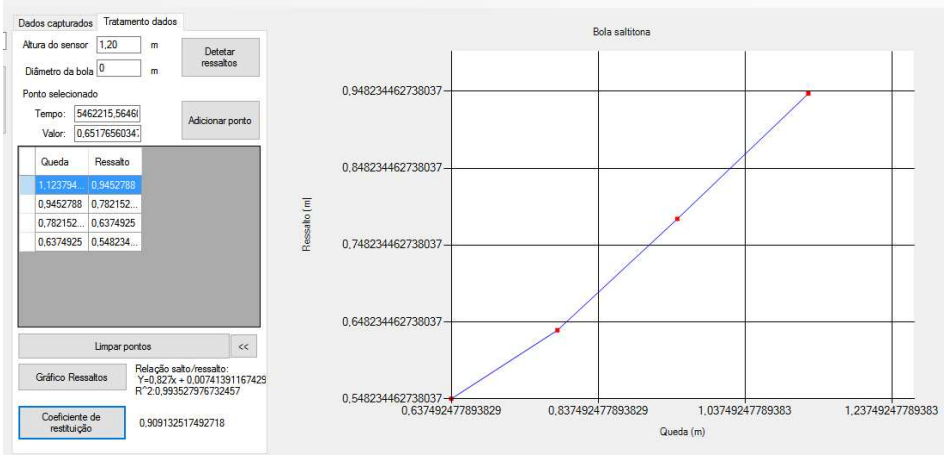



**Figura 73 - Coeficiente de restituição e gráfico dos ressaltos.**

No final, os alunos conseguiram guardar os dados e os gráficos gerados, podendo posteriormente utilizar estes no relatório da atividade.

### 5.6.3 AVALIAÇÃO CRÍTICA DOS TESTES DE UTILIZAÇÃO

Tendo os alunos executado previamente atividades com o equipamento da Pasco foi possível recolher opiniões comparativas entre este equipamento e a plataforma Askit. Em geral estes consideraram mais interessante o dispositivo apresentado neste projeto, por duas razões fundamentais:

- Tratando-se de uma plataforma aberta, sentiram maior curiosidade não só nos resultados que foram conseguidos, mas também no próprio dispositivo e nos seus componentes;

- Ao contrário do equipamento da Pasco, a nova plataforma é consistente e integra numa só interface as diferentes atividades. O equipamento da Pasco utilizado para a Bola Saltitona tem um *software* específico, enquanto que para a Queda Livre, foi utilizada uma photogate ligada diretamente a um dispositivo físico, com uma interface baseada num simples ecrã de cristais líquidos.

Nesta atividade não foi possível recolher a opinião do docente da disciplina, uma vez que decorreu na sua ausência, pois que este encontrava-se numa reunião de avaliação de uma turma do 12º ano. No entanto, os resultados foram-lhe apresentados e a participação deste no projeto permitiu-lhe acompanhar o desenvolvimento da aplicação e os seus contributos foram importantes.

O capítulo seguinte apresenta as conclusões e possíveis trabalhos para dar continuidade a este projeto.

# CAPÍTULO 6 – CONCLUSÃO E TRABALHOS FUTUROS

Para este projeto definiu-se um conjunto de objetivos genéricos para a plataforma e específicos para cada um dos seus componentes.

Em termos genéricos pretendia-se conceber uma plataforma baseada no Arduino capaz de permitir a execução de várias atividades laboratoriais de modo flexível, com um baixo custo e ligação em simultâneo a um computador e a um dispositivo Android.

No lado do Arduino o objetivo era permitir a criação de diferentes atividades laboratoriais sem que fosse necessário recorrer à sua reprogramação, garantindo a comunicação com dois tipos de dispositivos, o computador e o dispositivo móvel.

Para as aplicações móvel e desktop pretendia-se que capturassem os dados e que disponibilizassem ferramentas para o seu tratamento.

Por fim um objetivo específico da aplicação desktop era que a ferramenta a criar permitisse a execução de atividades laboratoriais de forma aberta, possibilitando que o professor criasse as suas próprias experiências.

Em geral, estes objetivos foram atingidos. O projeto criado permite que o Arduino não tenha de ser reprogramado graças ao protocolo de comunicação que se definiu, facultando o acesso aos sensores instalados com a troca de mensagens entre as aplicações desktop e móvel e o próprio Arduino.

O equipamento necessário para executar as atividades tem um custo reduzido, atualmente por cerca de 50 € é possível comprar o Arduino e os sensores necessários.

As aplicações criadas permitem a captura e o tratamento dos dados, facultando ferramentas de tratamento estatístico, como as regressões, e ferramentas de tratamento gráfico.

Todos os dados capturados e gerados pelas aplicações podem ser guardados e reutilizados noutros programas, como por exemplo o Microsoft Excel e o Microsoft Word, graças ao uso de formatos de ficheiros comuns como o XML e o PNG.

A aplicação desktop permite o controlo total das atividades por parte do docente, ao permitir ou não o envio em simultâneo dos dados para os dispositivos móveis.

Não se pretende com este trabalho substituir as propostas comerciais existentes. O tempo de desenvolvimento e os meios não são comparáveis. Ainda assim, os resultados obtidos comprovam a qualidade desta plataforma, para além de ser de salientar algumas vantagens, como o facto de todas as atividades serem desenhadas com base no currículo nacional da disciplina, tal como o Ministério da Educação o definiu e de se tratar de uma solução transparente, enriquecendo o valor pedagógico da plataforma.

Comparativamente com às soluções não comerciais, este projeto tem a vantagem de apresentar uma visão global para as atividades laboratoriais, ao contrário das soluções casuísticas propostas teriam.

As atividades específicas criadas permitem a execução de três atividades de modo orientado. No entanto, a opção de captura livre dos dados (denominada *Playground*) permite a utilização do projeto em inúmeras atividades, limitadas somente pelos sensores disponíveis.

Este projeto centra-se no ensino da disciplina de física, ainda que possa ser utilizado no ensino de outras ciências, desde que se possuam os sensores adequados.

O microcontrolador do Arduino impõe algumas limitações associadas às suas caraterísticas técnicas. Tendo uma velocidade de processamento de 16 MHz (Arduino SA, n.d.-b) a função que

devolve o tempo em microssegundos tem uma resolução de 4 (quatro) microssegundos, isto é, devolve sempre múltiplos de 4. Para além disso essa função volta a zero ao fim de aproximadamente 70 minutos de utilização continua, a razão prende-se com o facto de devolver um número de 32 bits permitindo no máximo $2^{32}$ valores diferentes (4,294,967,296) e 70 minutos são 4,200,000,000 microssegundos. Ainda assim com um microcontrolador diferente, por exemplo com um Arduino Mega, é possível obter uma maior capacidade de processamento e armazenamento ultrapassando algumas das limitações que, existindo, não são impeditivas da sua utilização.

A qualidade dos resultados depende dos sensores utilizados e da qualidade destes. Por exemplo, a qualidade do sensor de distância por ultrassons é menor que a qualidade de um sensor por infravermelhos. No entanto, a este nível deve-se procurar um equilíbrio entre a qualidade e o custo.

As aplicações não são exigentes, ao nível de recursos dos dispositivos. O programa para o Windows foi testado e desenvolvido com o Windows 7 SP1 e o Windows 10, ambos na versão de 64 bits. O computador utilizado foi um Toshiba modelo Satellite P750, com um processador i7-2630QM e 4 GB de memória RAM. A aplicação Android foi testada na versão 4.0.3 deste sistema operativo instalado num equipamento relativamente limitado para os padrões atuais, processador *single core* a 1GHz e 512MB de memória RAM, da marca HUAWEI modelo U8815.

O desenvolvimento do projeto foi um desafio do ponto de vista técnico. A plataforma é composta por três programas desenvolvidos com três ferramentas diferentes e três linguagens de programação diferentes. Do ponto de vista dos conhecimentos da área científica, foi necessário mobilizar conhecimentos de estatística como as regressões e, claro, de física.

Com este projeto conseguiu-se desenvolver um protocolo de comunicação entre três plataformas: Windows, Arduino e Android. Este protocolo sendo independente das aplicações pode, facilmente, ser reutilizado noutros projetos, no caso de um projeto Windows, basta a utilização da DLL criada que inclui as duas classes fundamentais de comunicação e armazenamento dos dados.

As atividades podem ser executadas com equipamento monetariamente bastante acessível e de montagem simples. A sua estrutura transparente e aberta possibilita outras oportunidades de aprendizagem e outros níveis de envolvimento dos alunos no processo.

A aplicação móvel completa a plataforma, envolve os alunos ao nível individual. Mesmo quando executam as tarefas em grupo podem analisar os dados nos seus próprios dispositivos móveis e sem onerar o orçamento escolar.

O projeto foi pensado e desenvolvido para uma utilização pedagógica, daí que os resultados a obter por parte dos alunos não são facultados. Ao invés, as aplicações fornecem os dados que os discentes devem utilizar para desenvolver as suas capacidades, podendo no final, comparar os seus resultados com os obtidos pelas aplicações.

De modo algum o projeto se pode considerar terminado. Como em qualquer projeto de desenvolvimento de *software*, foram definidos objetivos e mobilizados os recursos, escassos, para os atingir. Neste momento terminou uma fase que, para além de permitir um balanço do trabalho realizado, também contribuiu para o germinar de novas ideias.

A continuidade do projeto pode passar pela criação de novas atividades laboratoriais para a área de Física, mas também para outras disciplinas das ciências exatas, como a Química ou a Biologia.

A criação de uma plataforma Web que possa ser utilizada para carregar os ficheiros dos dados capturados e permitir a execução do seu tratamento é outra possibilidade de expandir o trabalho realizado. Sendo uma aplicação Web permitiria a partilha de resultados e ideias entre docentes de diferentes Escolas.

O protocolo tem também uma margem para melhoria. A adição de novos sensores digitais, como acelerómetros ou sensores de pressão pode ser outro caminho para a evolução do projeto.

A criação de versões do protocolo adaptadas para outras plataformas, como o novo Arduino Zero ou o Arduino Mega que, além de disponibilizarem mais memória e mais capacidade de processamento, também aumentam o número de pinos analógicos e digitais disponíveis.

A aplicação móvel pode ser melhorada, não só com a adição de novas atividades laboratoriais mas também com a criação de uma versão adaptada a *tablets* cujo ecrã maior permite a inclusão de novas funcionalidades.

A aplicação Windows tem a possibilidade de melhoria com o novo paradigma iniciado com o Windows 8 (WinRT) e continuado com o recente Windows 10 e as aplicações Windows universais (UWP) que para além de refrescarem a interface com o utilizador, com novos controlos, também podem ser executadas num conjunto de dispositivos mais amplo (Microsoft, n.d.).

Quanto ao dispositivo Arduino propriamente dito também pode ser melhorado com o desenvolvimento de uma estrutura, transparente, que inclua todos os sensores previamente instalados e devidamente acondicionados para que não seja tão suscetível de sofrer danos físicos durante a manipulação.

Em termos pedagógicos a colaboração com professores de Física e Química permitirá desenvolver novas atividades laboratoriais especificas, para além de permitir explorar aprofundadamente a atividade Playground.

A tradução das aplicações para inglês, espanhol e francês permitirá também ampliar o possível mercado alvo deste projeto.

O mercado dos dispositivos da Apple, computadores e dispositivos móveis, deve também ser considerado, ainda que não existam muitos equipamentos destes nas escolas secundárias portuguesas, pode ser uma aposta válida para um mercado mais abrangente.

Olhando para os objetivos traçados para este projeto e os resultados alcançados é possível considerar todo o tempo e trabalho investido proveitoso. Mesmo que este produto nunca venha a ser comercializado, foi possível demonstrar o valor que a plataforma Arduino e mostrar o que as tecnologias de informação em geral podem acrescentar para o ensino, sem que a fatura de aquisição e utilização seja desmotivadora ou impeditiva da sua utilização.

*"A man, as a gerenal rule, owes very little to what he is born with – a man is what he makes of himself."*

*Alexander Graham Bell*

# REFERÊNCIAS BIBLIOGRÁFICAS

Altay Scientific Group. (n.d.). Altay. Retrieved December 16, 2014, from http://www.altayscientific.com/en/home/

Analog Devices. (2008). TMP35/TMP36/TMP37. Retrieved from www.analog.com

Arduino. (2014, December 1). Retrieved December 1, 2014, from Arduino.cc

Arduino SA. (2015a). Arduino - Sketch. Retrieved April 3, 2015, from http://arduino.cc/en/Tutorial/Sketch

Arduino SA. (2015b, February 13). Arduino - AnalogRead. Retrieved from http://arduino.cc/en/Reference/analogRead

Arduino SA. (2015c, April 2). Arduino - Begin. Retrieved from http://arduino.cc/en/Serial/begin

Arduino SA. (2015d, April 3). SoftwareSerial. Retrieved April 3, 2015, from http://arduino.cc/en/Reference/softwareSerial

Arduino SA. (n.d.-a). Arduino - Knock. Retrieved from https://www.arduino.cc/en/Tutorial/Knock

Arduino SA. (n.d.-b). Arduino - Micros. Retrieved January 29, 2015, from https://www.arduino.cc/en/Reference/Micros

Atmel. (2011). Atmel ATmega48/88/168. Retrieved from http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

BIPM. (2008). Bureau International des Poids et Mesures. Retrieved May 28, 2015, from http://www.bipm.org/en/about-us/

Bluetooth SIG. (n.d.). Bluetooth Technology. Retrieved January 22, 2015, from http://www.bluetooth.com/Pages/Basics.aspx

Burman, M. (2011). Android File Explore. Retrieved September 2, 2015, from https://github.com/mburman/Android-File-Explore

Caldeira, H., & Bello, A. (2008). *Ontem e Hoje* (1ª ed.). Porto Editora.

Caldeira, H., Quadros, J., & Machado, C. (2015). *Há Física entre nós* (1ª ed.). Porto Editora.

Cavalcante, M. A., Tavolaro, C. R. C., & Molisani, E. (2011). Física com Arduino para iniciantes. *Revista Brasileira de Ensino de Física*, *33*(4).

Code Artist. (2012, April 30). MSChart Extension. Retrieved May 15, 2015, from http://www.codeproject.com/Articles/376059/MSChart-Extension-Zoom-and-Pan-Controls

Costa, A., Moisão, A., & Caeiro, F. (2008). *Novo Ver+ - Física A 11º ano* (1ª ed.). Plátano Editora.

Cytron Tecnologies. (2013). HC-SR04 User's Manual. Retrieved from http://www.cytron.com.my/p-sn-hc-sr04

Dismel, distribuidor de material electrónico, lda. (n.d.). Dismel. Retrieved December 16, 2014, from http://www.dismel.pt/

DL 139-2012 altera o curriculo dos 1º,2º,3º ciclos e secundário.pdf. (n.d.).

Eckel, T. (2012, August 15). Arduino Playground - NewPing Library. Retrieved April 3, 2015, from http://playground.arduino.cc/Code/NewPing

Ferreira, A. J., Braguez, F., Matos, M. G., Rodrigues, S., Fiolhais, C., Portela, C., … Nogueira, R. (2014). Programa de Física e Química A 10º e 11º anos. Ministério da Educação e Ciência.

Fiolhais, C., & Trindade, J. (2003). Física no computador: o Computador como uma Ferramenta no Ensino e na Aprendizagem das Ciências Físicas. *Revista Brasileira de Ensino de Física*, *25*(3), 259.

Google. (n.d.-a). List | Android Developers. Retrieved February 3, 2015, from http://developer.android.com/reference/java/util/List.html

Google. (n.d.-b). Motion Sensors | Android Developers. Retrieved April 4, 2015, from http://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-accel

Google. (n.d.-c). System - GC. Retrieved March 2, 2015, from https://developer.android.com/reference/java/lang/System.html#gc%28%29

Google. (n.d.-d). Thread. Retrieved February 3, 2015, from http://developer.android.com/reference/java/lang/Thread.html

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.

J. Roma, Lda. (n.d.). J.Roma. Retrieved December 16, 2014, from http://www.jroma.pt/

Zachariadou, Yiasemides, & Trougkakos. (2012, setembro). A low-cost computer-controlled Arduino-based educational laboratory system for teaching the fundamentals of photovoltaic cells. *European Journal of Physics*. Retrieved from stacks.iop.org/EJP/33/1599

Laudares, F. A. L., Cruz, F. A. de O., da Cruz, T. G., & Bigansolli, A. R. (2014). Instrumentação para Ensino de Física da UFRuralRJ: experiências docentes para a introdução tecnológica. *Revista de Formación E Innovación Educativa Universitaria*, *7*(1), 51–58.

Lee, S. (2013, April 19). Electronic brick of HC-06 serial port Bluetooth. Retrieved from iteadstudio.com

Margolis, M. (2011). *Arduino Cookbook*. O'Reilly.

Microsoft. (2015a). DataTable Class. Retrieved December 10, 2014, from https://msdn.microsoft.com/en-us/library/system.data.datatable%28v=vs.110%29.aspx

Microsoft. (2015b, June 18). Lock statement. Retrieved June 18, 2015, from https://msdn.microsoft.com/en-us/library/c5kehkcz.aspx

Microsoft. (n.d.). Guide to Universal Windows Platform (UWP) apps. Retrieved July 7, 2015, from https://msdn.microsoft.com/library/windows/apps/dn894631.aspx

Monteiro, A. (2013). *Otimização Não Linear de Mínimos Quadrados*. Universidade de Aveiro. Retrieved from http://www.portaldoconhecimento.gov.cv/bitstream/10961/3228/1/Agostinho_tese%20%282%29.pdf

Oliveira, F., Nascimento, M. A., Alberto, H. V., & Formosinho, S. (2011). Ensino das Ciências Físico-Químicas: O Papel do Professor face à Diversidade Cultural dos Alunos. *Revista Portuguesa de Pedagogia*, *Extra-Série*, 337–346.

Open Source Initiative. (1998). The MIT License. Retrieved May 15, 2015, from http://opensource.org/licenses/mit-license.php

Pasco. (2014). Pasco. Retrieved December 9, 2014, from www.pasco.com

PLX-DAQ - Parallax Inc. (2014, December 29). Retrieved December 29, 2014, from http://www.parallax.com/downloads/plx-daq

Rocha, F. S., & Marranghello, G. F. (2014). Acelerômetro eletrônico e a placa Arduíno para ensino de Física em tempo real. *Caderno Brasileiro de Ensino de Física*, *31*(1), 98–123.

Rodrigues, M. M. R. D., & Dias, F. M. L. (2007). *Física na Nossa Vida*. Porto Editora.

Seixas, J. (2005). *Introdução à Programação em Ciência e engenharia*. Escolar Editora.

Souza, A. R. de, Paixão, A. C., Uzêda, D. D., Dias, M. A., Duarte, S., & Amorim, H. S. de. (2011). A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC. *Revista Brasileira de Ensino de Física*, *33*(1). Retrieved from www.sbfisica.org.br

Stanford. (2014, December 12). Essentials of the MIDI protocol. Retrieved December 12, 2014, from https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html

Stephens, R. (2014, October 30). Find a polynomial least squares fit for a set of points in C#. Retrieved from http://csharphelper.com/blog/2014/10/find-a-polynomial-least-squares-fit-for-a-set-of-points-in-c/

Stoffregen, P. (2015). AltSoftSerial. Retrieved April 3, 2015, from http://www.pjrc.com/teensy/td_libs_AltSoftSerial.html

Veit, E. A., & Teodoro, V. D. (n.d.). Modelagem no ensino/aprendizagem de Física e os novos parâmetros curriculares nacionais para o ensino médio. *Revista Brasileira de Ensino de Física*. Retrieved from http://www.if.ufrgs.br/cref/ntef/producao/PCEM_modelagem_Veit_Teodoro.PDF

Verney. (2014, December 29). Kit Verney. Retrieved December 29, 2014, from http://www.vernier.com/products/packages/physics/lq-mini/

Vernier. (n.d.). Vernier. Retrieved December 29, 2014, from http://www.vernier.com

Wilson, E. B. (1952). *An Introduction to Scientific Research*. McGraw-Hill.

# ANEXOS

# ANEXO A – FICHAS DE ATIVIDADES ASKIT

## 11º Ano – A.L. 1.1 – Queda Livre

Um objeto está em queda livre quando a única força que atua sobre ele é a força gravítica. Nem mesmo a resistência do ar pode interagir ou é um valor tão pequeno que pode ser ignorado.

O resultado é o objeto acelerar em direção à Terra a uma velocidade constante.

**Objetivos**

- Medir a aceleração de um objeto em queda livre.

**Fórmulas**

Fórmula da velocidade

$$v = at + v_0$$

Em que:

- $v$ é a velocidade
- $a$ é a aceleração
- $t$ é o tempo
- $v0$ é a velocidade inicial

Fórmula do movimento

$$x = \frac{1}{2}at^2 + v_0t + x_0$$

Em que:

- $x$ é a posição
- $a$ é a aceleração
- $t$ é o tempo
- $v0$ é a velocidade inicial
- $x0$ é a posição inicial

**Materiais**

- Um arduino com o ASKIT-Arduino Protocol instalado;
- Um computador com Windows e o ASKIT-Windows instalado;

- Um LDR;

- Uma fonte de luz, por exemplo um LED;

- Uma régua transparente;

- Fita-cola de cor (opaca).

**Procedimentos**

- Na régua utilizar a fita-cola para marcar linhas opacas com distâncias iguais, por exemplo cinco linhas com seis centímetros de intervalo.

- Ligar o LDR e o LED ao Arduino de forma que a luz emitida pelo LED incida no LDR. O LDR deve ser ligado ao pino A0 para captura dos dados.

- Ligue o Arduino ao computador através do cabo USB.

- Abra o ASKit Windows e selecione a atividade Queda Livre.

- Selecione a porta de comunicação do Arduino e clique no botão Iniciar.

- Se pretender enviar para a aplicação Android ative o envio Bluetooth.

- Ligue a aplicação Android ao Arduino e ative a captura de dados nesta aplicação.

- Deixe cair a régua de modo que as linhas opacas desta obstruam a luz do LED no LDR.

- Clique no botão Parar.

- Selecione o separador para tratamento dos dados.

- Para o tratamento dos dados pode utilizar o procedimento automático de seleção dos pontos de passagem das linhas da régua.

- Para a seleção manual dos pontos faça zoom sobre o gráfico de modo que consiga visualizar as curvas que marcam a passagem das linhas da régua. Agora com a ferramenta de seleção do gráfico clique sobre o ponto mais baixo da linha e clique no botão Adicionar Ponto Selecionado.

- Repita o procedimento para os restantes pontos.

- Para obter as componentes de fórmula do movimento é necessário fazer a regressão dos dados selecionados.

**Resultados**

- Os resultados obtidos devem ser o valor da gravidade para a aceleração da queda da régua;

**Análise dos resultados**

- Os discentes devem, agora, fazer uma análise dos resultados obtidos explicando a razão para uma eventual diferença entre o valor obtido para a aceleração e o valor esperado.

- Deve-se também calcular o erro relativo de um valor medido experimentalmente utilizando a seguinte expressão:

$$E = \frac{|X - X_v|}{X_v} \times 100$$

Onde $X$ representa o valor medido e $X_v$ o valor esperado.

Guarde os dados capturados e tratados, bem como os gráficos para incluir nos relatórios a redigir.

**10º Ano – A.L. 1.2 – Bola Saltitona**

Quando ocorre uma colisão é possível determinar o grau de elasticidade a partir da relação entre a distância antes e após a colisão. Esta relação denomina-se coeficiente de restituição.

**Objetivos**

- Determinar o coeficiente de restituição de uma bola.

**Fórmulas**

O valor do coeficiente de restituição da bola é igual à raiz quadrada do declive da reta.

Com base nos dados das alturas da queda e do ressalto obtém-se a reta dos ressaltos em função da altura da queda.

$$y = mx - x0$$

Coeficiente de restituição:

$$e = \sqrt{m}$$

Em que:

- $m$ é o declive da reta obtida

Percentagem de energia dissipada:

$$E_{dissipada}\% = \frac{h_r}{h_q} \times 100$$

Em que:

- $hr$ é a altura do ressalto
- $hq$ é a altura da queda

**Materiais**

- Um arduino com o ASKIT-Arduino instalado;
- Um computador com Windows e o ASKIT-Windows instalado;
- Um sensor ultrassónico de distância;
- Uma, ou várias, bola(s);

**Procedimento**

- Ligue o sensor de distância de modo que o pino que emite o ultrassom (Trigger) seja ligado ao pino digital número 3 e o pino que deteta o retorno do ultrassom (Echo) deve ser ligado ao pino digital número 4.

- Ligue o Arduino ao computador através do cabo USB.

- Abra o ASKit Windows e selecione a atividade Bola Saltitona.

- Selecione a porta de comunicação do Arduino e clique no botão Iniciar.

- Se pretender enviar para a aplicação Android ative o envio Bluetooth.

- Ligue a aplicação Android ao Arduino e ative a captura de dados nesta aplicação.

- Fixe o sensor de distância virado para baixo à altura pretendida.

- Coloque a bola diretamente por baixo do sensor a aproximadamente dois centímetros.

- Deixe cair a bola até que esta pare de ressaltar.

- Clique no botão Parar.

- Selecione o separador para tratamento dos dados.

- Para o tratamento dos dados pode utilizar o procedimento automático de seleção dos ressaltos.

- Antes de iniciar o tratamento dos dados verifique se altura indicada para o sensor está correta e insira o diâmetro da bola. Caso insira a altura do sensor até à bola no chão deve inserir zero no diâmetro da bola.

- Para um tratamento manual dos dados selecione o ponto da queda e dos ressaltos da bola.

- Com estes dados deve solicitar aos alunos que calculem o coeficiente de restituição, podendo este cálculo ser feito para cada queda e ressalto e no final, utilizando uma regressão linear, com base na inclinação da reta do gráfico dos ressaltos.

- Opcionalmente os alunos podem também calcular a percentagem de energia dissipada em cada ressalto.

- Repetir os procedimentos para diferentes bolas e/ou diferentes pisos.

**Resultados**

- Os resultados obtidos para o coeficiente de restituição devem ser mais próximos de zero para as bolas mais duras, com menos elasticidade, e mais próximos de um para as mais que saltam mais.

**Análise dos resultados**

- Os alunos devem no final analisar os resultados, comparando os dados obtidos com diferentes materiais.

- A qualidade dos dados é fundamental, para isso os discentes devem ter o cuidado de largar a bola de modo que os ressaltos ocorram dentro do campo de ação do sensor e devem ter o cuidado não interferir com os ressaltos ou com o sensor.

Guarde os dados capturados e tratados, bem como os gráficos para incluir nos relatórios a redigir.

**10º Ano – A.L. 1.1 – Plano inclinado**

Um carro a rolar por um plano inclinado tem a sua enérgica mecânica transforma de energia potencial em energia cinética.

**Objetivos**

- Proceder ao estudo da variação da energia cinética em função da distância percorrida ao longo da rampa.

**Fórmulas**

Velocidade instantânea, por aproximação ao valor da velocidade média

$$v = \frac{\Delta x}{\Delta t}$$

Em que

- $\Delta x$ é a deslocação no espaço
- $\Delta t$ é o tempo necessário para percorrer o espaço

Energia cinética

$$E_c = \frac{1}{2} m v^2$$

Em que

- $m$ é a massa do objeto
- $v$ é a velocidade

**Materiais**

- Um arduino com o ASKIT-Arduino instalado;
- Um computador com Windows e o ASKIT-Windows instalado;
- Dois, ou mais, LDRs;
- Uma fonte de luz;
- Um carro;
- Um plano inclinado.

**Procedimentos**

- Instale o plano inclinado e fixe os LDR de modo a capturar a passagem do carrinho durante a descida sem que interfira com este. Os LDR devem ficar todos com a mesma distância entre si.

- Ligue os sensores LDR ao Arduino começando pela ligação do primeiro LDR no primeiro pino analógico, o segundo na segunda e assim por diante.

- Ligue o Arduino ao computador através do cabo USB.

- Abra o ASKit Windows e selecione a atividade Plano Inclinado.

- Selecione a porta de comunicação do Arduino e clique no botão Iniciar.

- Se pretender enviar para a aplicação Android ative o envio Bluetooth.

- Ligue a aplicação Android ao Arduino e ative a captura de dados nesta aplicação.

- Largue o carrinho do início do plano inclinado.

- Clique no botão Parar.

- Selecione o separador para tratamento dos dados.

- Registe a distância entre cada sensor LDR e a massa do carrinho.

- No gráfico selecione cada um dos pontos de passagem correspondentes ao momento em que cada um dos LDR ficou obscurecido.

- Com estes dados deve solicitar aos alunos que calculem a velocidade de passagem em cada ponto, utilizando o primeiro como referência.

- Com a velocidade calculada deve ser pedido aos alunos que calculem a energia cinética em cada um dos pontos.

- Com estes dados os alunos podem produzir vários tipos de gráficos, como por exemplo:

    - Gráfico da posição e da velocidade ao longo do tempo;

    - Gráfico da energia cinética em função da velocidade;

    - Gráfico da energia cinética em função da distância.

- Repetir os procedimentos para diferentes carrinhos ou adicionando pesos ao mesmo carro.

**Resultados**

Os alunos devem chegar a conclusões que indicam que a enérgica cinética aumenta em função da velocidade e da massa.

**Análise dos resultados**

Na análise dos resultados os discentes devem indicar possíveis efeitos secundários que afetaram os resultados como a inércia do carro ou o tipo de piso do plano inclinado.

## ANEXO B – ESTRUTURA DOS FICHEIROS XML

**Queda Livre**

```
<?xml version="1.0" standalone="yes"?>
<DocumentElement>
  <QuedaLivre>
    <Tempo></Tempo>
    <Valor></Valor>
    <Pino></Pino>
  </QuedaLivre>
</DocumentElement>
```

**Bola Saltitona**

```
<?xml version="1.0" standalone="yes"?>
<DocumentElement>
  <Bola>
    <Tempo>32</Tempo>
    <Valor>1</Valor>
    <Pino>0</Pino>
  </Bola>
</DocumentElement>
```

**Plano Inclinado**

```
<?xml version="1.0" standalone="yes"?>
<DocumentElement>
  <Plano>
    <Tempo>72</Tempo>
    <Valor>679</Valor>
    <Pino>14</Pino>
  </Plano>
</DocumentElement>
```

**Playground**

```
<?xml version="1.0" standalone="yes"?>
<DocumentElement>
  <ArduinoPlayGround>
    <Tempo>76</Tempo>
    <Valor>720</Valor>
    <Pino>14</Pino>
  </ArduinoPlayGround>
</DocumentElement>
```

## ANEXO C – ESTRUTURA DAS CLASSES

**Aplicação Windows**

```
class W_ArduinoProtocol{
//Propriedades
      public const int SENSOR_DISTANCIA = 0x01;
      public const int SENSOR_TEMP_HUM = 0x02;
      private const int FILTRAR_ON = 0x03;
      private const int FILTRAR_OFF = 0x04;
      private const int FECHAR_TODOS = 0x05;
      private const int DEFINE_RUIDO = 0x06;
      private const int ABRIR_PINO_TEMPO = 0x07;
      private const int ABRIR_PINO = 0x08;
      private const int FECHAR_PINO = 0x09;
      private const int LIGAR_PINO = 0x10;
      private const int DESLIGAR_PINO = 0x11;
      private const int LER_ESTADO_PINO = 0x12;
      private const int VALOR_PWM = 0x13;
      private const int LIGAR_BT = 0x14;
      private const int DESLIGAR_BT = 0x15;
      private const int SENSOR_DISTANCIA_ON = 0xA;
      private const int SENSOR_DISTANCIA_OFF = 0xB;
      private const int SENSOR_TEMP_HUM_ON = 0xC;
      private const int SENSOR_TEMP_HUM_OFF = 0xD;
      private const int VERSAO = 0xE;
      private const int MAX_DATA = 32;
      private const int TEMPO_RESPOSTA = 50;
      private const int VELOCIDADE_COMUNICA =500000;
      private const int DESLIGADO = -1;
      private SerialPort portaSerie;
      private Thread tarefa=null;
      private object bloquear = new object();
      List<string> COMports = new List<string>();
      private volatile float[] dados = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
      private volatile float[] sensores = { 0, 0 };
      public long conta_lidos_corretos, conta_lidos_com_erros;
      public float versaoProtocolo=0;
      string buffer;
      public W_ArduinoData tabela;
      bool guardarDados;
      public bool aCapturar;
      private Stopwatch tempo;
      UInt32 tempoAtual;
   //Métodos
      public W_ArduinoProtocol(string porta,int velocidade)
      public W_ArduinoProtocol()
      public W_ArduinoProtocol(bool iniciar)
      ~W_ArduinoProtocol()
      public void setGuardaDados(bool guardarDados)
      public bool getGuardaDados()
      public void fecharTodos()
```

```
        public List<string> devolvePortas()
        public void start()
        public bool start(string porta)
        public void start(string porta, int velocidade)
        public float qos()
        public void clearQoS()
        public int portaSerieTemDados()
        public void stop()
        public bool estado()
        private void descobrePortas(){
        public void abrirPino(int numero)
        public void fecharPino(int numero)
        public void abrirSensor(int tipo)
        public void abrirSensor(int tipo,int intervaloMilissegundos)
        public void fecharSensor(int tipo)
        public float valorSensor(int tipo)
        private void processaLeituraBinario()
        private void processaLeitura()
        public float valorPorta(int iporta)
        public void ligarPino(int numero)
        public void enviarPWM(int numero,int valor)
        public void desligarPino(int numero)
        public float lerEstadoPino(int numero)
        public void abrirPinoTempo(int pino, int p)
        public void ligarFiltro()
        public void ligarFiltro(int valor)
        public void desligarFiltro()
        public void defineRuido(int valor)
        public void ligarBT()
        public void desligarBT()
        public void preencheCB(ComboBox cb_Porta)
        public float lerVersaoProtocolo()
    }


class W_ArduinoData{
//Propriedades
        private DataTable tabela;
        private float ruido = 100;
//Métodos
        public Boolean adicionaColunaCalculada(string expressao)
        public W_ArduinoData()
        public void preparaTabela()
        public void preparaTabelaTemposPassagem()
        public void preparaTabelaRessaltos()
        public void preparaTabelaTemposPlano()
        public void preparaTabelaPlayGround()
        public void removeLinha(int linha)
        public void removeLinhas(int linha, int count)
        public void removeColuna(int coluna)
        public void setNome(string nome)
```

```
    public string getNome()
    public float getRuido()
    public void defineRuido(float _ruido)
    ~W_ArduinoData()
    public DataTable getTabela()
    public void clearTabela()
    public DataTable getPino(int pino)
    public DataRow getPrimeiroValor()
    public float getPrimeiroValorPino(int pino)
    public int getNrPinos()
    public List<int> listaPinos()
    public void guardarDadosFicheiro(string nome)
    public void lerDadosFicheiro(string nome)
    public void adiciona(UInt32 tempo, float valor, int porta)
    public void adiciona(UInt32 tempo, float valor)
    public void adiciona(long tempo, float valor)
    public void adiciona(double tempo, float valor)
    public void adiciona(double x,double y)
    public void adicionaPontosPassagem(long temp, float distanciacm, float tempoSegundos)
    public void adicionaPontosPassagem(long temp, float distanciacm, float
tempoSegundos,float velocidade)
    public void adicionaRessaltos(float queda,float ressalto)
    public void adicionaPontosPassagemPlano(long temp, float distanciacm, float
tempoSegundos, float velocidade,float energia)
    public long nrRegistos()
    public float media()
    public float media(int nrvalores)
    public float moda()
    public float min()
    public float min(int coluna)
    public float proxMin(float tempo)
    public float max()
    public DataRow devolveUltimaLinha()
    public float amplitude()
    public Dictionary<UInt32, float> picos()
    public Dictionary<UInt32, float> linhas(Dictionary<UInt32, float> picos)
    public Dictionary<UInt32, float> ressaltos()
  }
```

**Aplicação Android**

```
public class A_ArduinoProtocol extends Thread {
  private class ProcessaLeitura extends Thread{
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    byte[] buffer=new byte[1024];

    public ProcessaLeitura(BluetoothSocket socket,float[] dados)
    public void run()
  }
```

```
//Propriedades
   public static final byte SENSOR_DISTANCIA=0x01;
   public static final byte SENSOR_TEMP_HUM=0x02;
   public static final byte FILTRAR_ON=0x03;
   public static final byte FILTRAR_OFF=0x04;
   public static final byte FECHAR_TODOS=0x05;
   public static final byte ABRIR_PINO_TEMPO=0x07;
   private static final byte ABRIR_PINO=0x08;
   private static final byte FECHAR_PINO=0x09;
   private static final byte LIGAR_PINO=0x10;
   private static final byte DESLIGAR_PINO=0x011;
   private static final byte LER_ESTADO_PINO = 0x12;
   private static final byte VALOR_PWM = 0x13;
   private static final byte SENSOR_DISTANCIA_ON = 0xA;
   private static final byte SENSOR_DISTANCIA_OFF = 0xB;
   private static final byte SENSOR_TEMP_HUM_ON = 0xC;
   private static final byte SENSOR_TEMP_HUM_OFF = 0xD;
   private static final int TEMPO_RESPOSTA=75;
   private BluetoothAdapter mBluetoothAdapter;
   private BluetoothSocket mmSocket;
   private BluetoothDevice mmDevice;
   private static final UUID MY_UUID =UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
   private int estado;
   private OutputStream mmOutStream=null;
   private volatile float[] dados={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
   private volatile float[] sensores={0,0};
   private ProcessaLeitura servicoLeitura;
   public long conta_lidos_corretos,conta_lidos_com_erros;
   private boolean aGuardarDados;
   private A_ArduinoData arduinoData =new A_ArduinoData();
   private Date tempo;
   private long tempoCaptura;
   private float versaoProtocolo;
//Métodos
   public ArduinoProtocol(BluetoothDevice device)
   public ArduinoProtocol()
   public void defineAGuardar()
   public void defineAGuardar(boolean aGuardarDados)
   public boolean devolveAGuardar()
   public float devolveVersaoProtocolo()
   public void lerDados(String ficheiro)
   public void lerDados(String ficheiro,String actividade)
   public String  guardarDados()
   public String  guardarDados(String atividade)
   public void enviar(byte x)
   public void enviar(byte[] x)
   public String getEstado()
   public int devolveEstado()
   public void run()
   public void setCaptura(boolean aGuardarDados)
   public boolean getCaptura()
```

```
    public void toogleCaptura()
    public List getListaDados()
    public void limpaListaDados()
    public W_ArduinoData devolveTabela()
    public void cancel()
    public float QoS()
    public void abrirPino(int numero)
    public void ligarPino(int numero)
    public void abrirSensor(int tipo)
    public void fecharSensor(int tipo)
    public float valorSensor(int tipo)
    public void fecharPino(int numero)
    public float valorPino(int numero)
    public float lerEstadoPino(int numero)
}

public class A_ArduinoData {
//Propriedades
    private List<SerieDados> lDados=new ArrayList<SerieDados>();
    private float ruido=150;
//Métodos
    public A_ArduinoData()
    public float getRuido()
    public void defineRuido(float _ruido)
    public void removeLinha(int linha)
    public void removeLinhas(int linha,int count)
    public List<SerieDados> getTabela()
    public void clearTabela()
    public List<SerieDados> getPino(int pino)
    public SerieDados getPrimeiroValor()
    public float getPrimeiroValorPino(int pino)
    public int getNrPinos()
    public SerieDados devolveUltimaLinha()
    public List<Integer> listaPinos()
    public void adiciona(int tempoCaptura,int valor)
    public void adiciona(int tempoCaptura,float valor)
    public void adiciona(SerieDados dados)
    public float max()
    public float max(int pino)
    public float min()
    public float min(int pino)
    public float moda()
    public float media()
    public float media(int pino)
    public float mediaDe(int nramostras)
    public float amplitude()
    public List<SerieDados> picos()
    public void lerDados(String nome)
    public void lerDados(String nome,String actividade)
    public  List<SerieDadosTempos> lerDadosQuedaProc(String nome)
    public  List<SerieDados> lerDadosBolaProc(String nome)
    public  List<SerieDadosTempos> lerDadosPlanoProc(String nome)
```

```
   public void lerDadosGrafico(Context context)
   public String guardarDados()
   public String guardarDadosGrafico(Context context)
   public String guardarDados(String actividade)
   public List<SerieDados> ressaltos()
}

public class SerieDados {
//Propriedades
   float tempo;
   float valor;
   int pino;
//Métodos
   public SerieDados(float tempo,float valor)
   public SerieDados(int tempo,int valor)
   public SerieDados(float tempo,float valor,int pino)
   public SerieDados(float tempo,float valor,float pino)
   public SerieDados(int tempo,int valor,int pino)
   public float devolveTempo()
   public float devolveValor()
}
```