# 2º TRABALHO
## INSTRUMENTAÇÃO E AQUISIÇÃO DE DADOS


| **Ana Sofia Sousa** | **Duarte Marques** | **João Chaves** |
|---|---|---|
| 96508 | 96523 | 96540 |


## Objetivos

Este trabalho tem como objetivo a execução de um sistema de controlo de rega, com recurso a um Arduino UNO e a um Raspberry PI Model B, com as seguintes características propostas:

- 12 canais de rega (correspondentes a 12 eletroválvulas);
- até 20 programas;
- escolha entre duas fontes (torneira ou bomba);
- leitura e interpretação de dados de um sensor de humidade;
- funcionamento independente do Arduino;
- permitir a implementação de todos os *drivers* de correntes e relés necessários;
- programação de *software* para o Raspberry PI que permita configurar e ler a configuração do Arduino.

## Trabalho realizado

Foram desenvolvidos um programa e duas bibliotecas em Python3 - `main.py`, `source_arduino.py` e `source_window.py`, para executar no Raspberry PI - e um programa em C++ - `REGA.ino`, para ser executado no Arduino.

- **main.py**: No respetivo código, é importado o programa `source_window.py` (no qual, por sua vez, se importa `source_arduino.py`). É também aplicado o comando `run()` à classe `Master` (definida no ficheiro `source_window.py`), de forma a que todo o trabalho possa ser executado ao correr o programa `main.py`.
- **source_window.py**: Com este programa, é criada a interface com o utilizador, nomeadamemte, janelas, botões, *tabs* e *labels* com diferentes caraterísticas, dependendo das condições de operação do sistema de rega que se pretende selecionar.
- **source_arduino.py**: Neste programa, encontra-se definida a função `start_serial`, na qual é iniciada a comunicação serial entre o Arduino e o Raspberry PI e é controlado o envio do tempo atual para o primeiro (recorrendo ao parâmetro `do_time`). Existem também as funções que controlam o cessar da comunicação serial (`stop_serial`) e a escrita no *buffer* sem ou com mensagem de retorno (`write` e `request`, respetivamente). Associadas à monitorização do **tempo**, foram definidas as funções `get_time` (recebe uma *string* com os dados do tempo decorrido e retorna os respetivos valores) e `set_time` (envia para o Arduino o tempo atual). Com `get_source` e `set_source`, recebe-se e envia-se (respetivamente) qual a **fonte** (0 para *water tank* e 1 para *well pump*). Através de `get_mode`, recebe-se do Arduino o **modo de operação** - para o controlo da humidade (modo 0) ou dos horários pré-estabelecidos (modo diário 1 e semanal 2). O primeiro é estabelecido no Arduino graças a `set_humidity`; o valor da humidade num dado instante é obtido por `get_humidity` (retorna o valor em %, estando o sensor ligado a 5V). Para os modos de funcionamento semanal e diário, existem as funções `set_weekly` e `set_daily`, respetivamente, sendo possível receber o modo e parâmetros de funcionamento com a função `get_prescheduled`. As condições de funcionamento do sistema de rega associadas às funções anteriores são introduzidas pelo utilizador na interface.

- **REGA.ino**: Neste *script*, começa-se por definir variáveis globais (utilizadas em diferentes funções), nomeadamente as variáveis relativas à fonte selecionada (torneira ou bomba), aos modos de operação do sistema de rega e ao tempo decorrido. Na função principal (`loop`), é controlada toda a operação de rega e a comunicação com os programas do Raspberry PI. A função `update_time` é chamada uma vez em cada *loop* e mantém um registo temporal, calibrado incialmente pelo Raspberry PI. A verificação se as torneiras devem ser abertas/fechadas, seja consoante o tempo atual ou consoante a humidade, é realizada apenas de 5 em 5 segundos, de forma a poupar recursos e não atrasar a contagem do tempo através da função interna `millis()`.

Todas as funções e definições encontram-se comentadas no próprio código, mas de forma a sistematizar o seu uso (nos programas Python), foi criado um documento Doxygen, onde existe um manual de referência detalhado. Este pode ser acedido através do ficheiro `index.html`, acessível na pasta `html`.

De forma a testar os códigos desenvolvidos, foi montado, com recurso a uma *breadboard*, o circuito representado na Figura 1. Foi ligado um LED (associado a uma resistência) ao pino A5 de forma a testar visualmente a abertura e fecho da válvula do relé. Por sua vez, o condensador foi ligado entre os pinos GND e RESET do Arduino para desativar o *reset* automático quando é iniciada comunicação *serial*, preservando assim os parâmetros de funcionamento enquanto o Arduino está ligado.

## Guia de utilização

De forma a inicializar a interface, deve executar-se o programa `main.py`.

Na janela inicial, deve-se selecionar um dos modos de operação do sistema de rega: **Humidity control mode** ou **Pre-scheduled mode**. No primeiro caso, a válvula será aberta ou fechada mediante a leitura da humidade pelo sensor e as configurações impostas. No segundo caso, definir-se-á um horário para a rega. Uma vez selecionado o modo de operação, deverá clicar-se no botão **Start**. Quando o Arduino faz *reset*, o modo por defeito é o primeiro. Caso contrário, o modo que aparece na caixa é o atual. Em ambos os modos de operação, a fonte que aparece inicialmente na caixa é também a que está selecionada nesse momento, no Arduino (por defeito, é a primeira). Quaisquer alterações são enviadas conjuntamente com o resto das configurações (e por conseguinte, apenas se estas forem válidas).

Caso se tenha escolhido **Humidity control mode**, existem dois modos separados de funcionamento: **Calibrate** e **Manual**. Para o primeiro, deverá clicar-se no botão **Calibrate**, de forma a obter um valor da humidade atual (lida pelo sensor). Deverá ainda selecionar-se um valor em **Set margin (%)**, de forma a indicar ao programa para abrir a válvula quando a humidade for inferior ao limite estipulado por esta margem e fechar a válvula quando se ultrapassar o limite superior de humidade. Para o segundo, podem selecionar-se os intervalos manualmente, tendo o programa duas sugestões.

É de notar que podem ocorrer pequenos erros de arredondamento (o Arduino requer inteiros de 0 a 1023 e não percentagens), pelo que os valores apresentados como definidos podem divergir ligeiramente dos selecionados. Tal como no outro modo, deve também selecionar-se a fonte em **Select source** - poderá escolher-se **Water tank** ou **Well pump**. Em seguida, clica-se em **Send**, surgindo uma mensagem de erro caso os parâmetros de humidade e/ou a fonte não tenham sido comunicados corretamente.

No **Pre-scheduled mode**, para além da seleção da fonte, existem duas *tabs*, **Daily** e **Weekly**, que permitem, respetivamente, selecionar o horário de funcionamento por intervalos de dias entre regas ou em determinados dias da semana. No modo Daily, em **Daily interval**, o utilizador insere o intervalo de dias entre regas (1 para regas diárias, 2 para regas a cada dois dias, etc.). Na coluna mais à direita, deverá escolher as horas e minutos de início e fim de cada programa de um dia; após esta escolha, o utilizador deverá clicar em **Add**, surgindo o intervalo imposto na coluna central. Surge uma mensagem de erro caso se tenha selecionado um tempo de início superior ao de fim ou se tenha introduzido uma entrada duplicada. Por outro lado, podem utilizar-se os botões **Undo** e **Clear** para eliminar a última entrada adicionada ou todas as entradas, respetivamente. Por fim, deve clicar-se em **Send** para enviar os parâmetros para o Arduino.

Finalmente, na *tab* **Weekly**, selecionam-se à esquerda os dias da semana desejados, em *checkboxes*, podendo o utilizador escolher qualquer combinação de dias. De forma análoga ao modo **Daily**, o utilizador seleciona na coluna da direita os tempos de funcionamento do sistema de rega em cada dia, devendo clicar em **Send** no final da sua escolha. Existe também validação dos dias e tempos selecionados antes do envio.

É de notar que, tanto no modo **Weekly** como **Daily**, os intervalos são otimizados *antes* do seu envio para o Arduino, juntando intervalos sobrepostos, se necessário. Os intervalos efetivamente enviados surgem no ecrã após o envio, em vez de os originalmente selecionados. Existe um limite de 50 intervalos temporais, aparecendo um aviso caso este seja ultrapassado.

## Conclusões

Foi programado com sucesso um sistema de rega com as caraterísticas desejadas, recorrendo à comunicação entre o Arduino UNO e o Raspberry PI, utilizando o sensor de humidade Grove e podendo o Arduino funcionar em *stand-alone*. Por um lado, é possível selecionar entre duas fontes - *water tank* e *well pump*. Além disso, pode controlar-se a ativação da rega tendo em conta a humidade lida no sensor ou um programa pré-definido - com um dado intervalo de dias entre regas ou em determinados dias da semana (em cada caso, pode ser ativada a rega em diferentes períodos de um dia). Considera-se assim que os objetivos propostos foram alcançados.

## Anexo - esquema da montagem

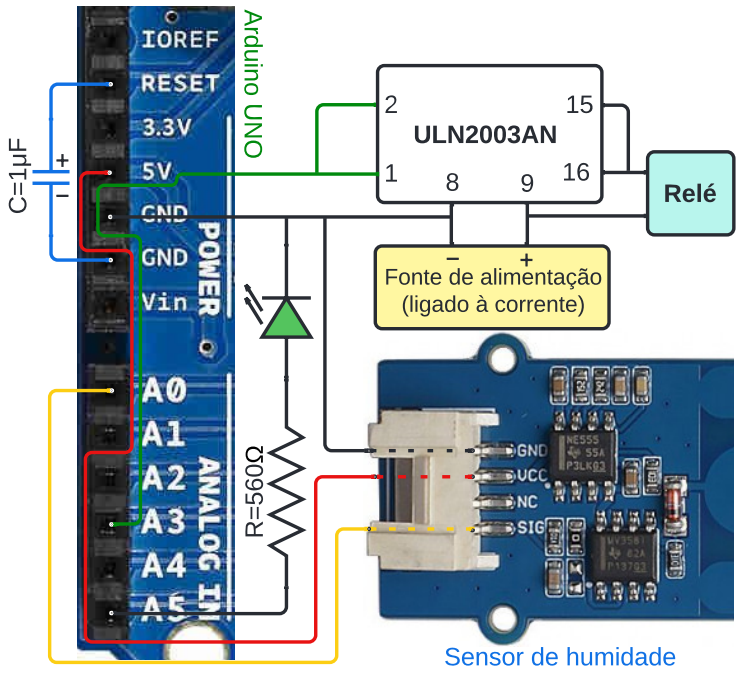


Figura 1: Circuito montado na *breadboard* para testar os códigos desenvolvidos.