Wattering from Twitter

José Sousa
up201503443
MI:ERSI

Guilherme Amado
up201608528
M:SI

June 16, 2020

Word count: 1967

Abstract

Hoje em dia, com a popularização da Internet e dos Smartphones há cada vez uma maior ligação ao mundo online, onde recebemos muitas vezes notícias e actualizações ao minuto. Neste projeto tiramos vantagem desta ligação e, tirando uso de hardware como sensores, Arduino e Raspberry Pi, enviamos atualizações do estado do meio ambiente da planta, tirando vantagem também da popular rede social Twitter. Com o nosso projeto torna-se possível vigiar a planta à distância e até "personalizar" a sua presença na rede social.

1 Introdução

Ao longo dos anos tem sido cada vez mais presente nas nossas vidas as "actualizações de estado", em grande parte com a popularização das redes sociais. De facto, hoje em dia, conseguimos obter informação sobre quase tudo minuto a minuto.

Nesta linha de pensamento decidimos implementar um sistema que, recorrendo a hardware como sensores e microcontroladores, fosse capaz de actualizar o estado do meio ambiente duma planta. Decidimos que seria relevante ler dados como a temperatura, a humidade no ar, a humidade do solo e a luz, factores chave para a vida saudável da planta.

Na secção 2 clarificamos quais os objectivos que tivemos em conta na realização deste trabalho, em 3 listamos os requisitos necessários para a concretização do mesmo, na secção 4 falamos da arquitectura do sistema e na secção 5 descrevemos a implementação do sistema. Finalmente vemos os resultados finais na 6 e tiramos algumas conclusões sobre os mesmos na secção 7.

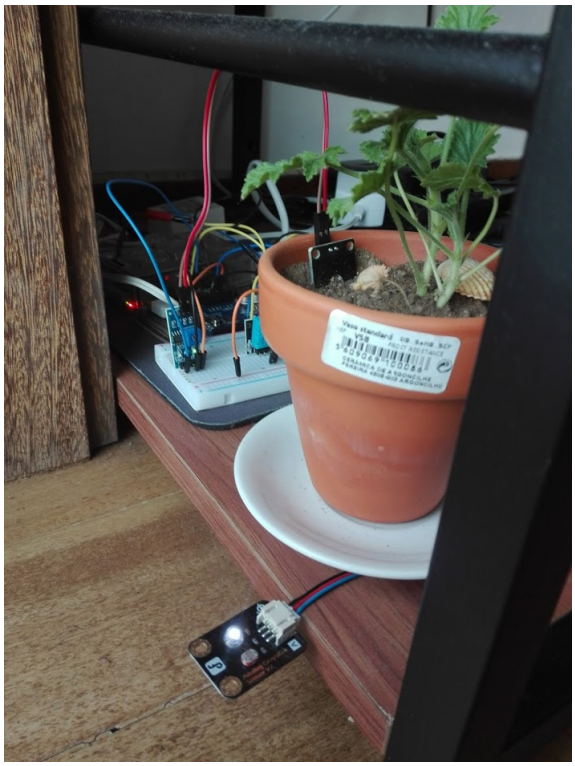

Figure 1: Planta com o sistema.

2 Objectivos

O principal objectivo deste projecto seria ter alguma forma de receber actualizações do estado do ambiente duma dada planta. Para isso, empregaram-se sensores ligados a um Raspberry Pi, que periodicamente, lança estas actualizações do estado para uma conta Twitter sob a forma de post.

Outro objectivo era ao mesmo tempo criar soluções para que ambos os membros do grupo pudessem trabalhar com os equipamentos, ainda que apenas um tivesse acesso físico ao mesmo.

Finalmente se ainda tivéssemos tempo queríamos também criar uma aplicação para poder ver em tempo real as condições da planta, o que acabou por não acontecer.

3 Requisitos

Os requisitos dividem-se em requisitos de software e de hardware. Para já adiantamos apenas as diferentes peças que vão ser precisas, e nas secções de arquitectura (4) e implementação (5) falaremos mais em concreto do papel de cada.

3.1 Hardware

Sumariamente, o hardware a utilizar foi:

- Raspberry Pi, no nosso caso o modelo 3 Model B+;
- Arduino Uno;
- Sensor de temperatura e humidade do meio ambiente, modelo DHT-11;
- Sensor de humidade do solo, modelo B077PW1VW5, HUABAN;
- Sensor de luz solar, modelo Gray Scale V2, GROVE.

Podemos adiantar que o Raspberry será quem agendará os tweets e as verificações de estado, enquanto o Arduino fará a ligação Sensores-Raspberry.

3.2 Software

O software diz tanto respeito ao software que usamos para o serviço de tweeting como também o software usado para ultrapassar os desafios impostos pelo COVID-19 para a realização do trabalho.

No lado do Raspberry, iremos usar NodeJS e Java. Usamos duas APIs com o NodeJS. A primeira é a Johnny-Five, que facilita o controlo do microcontrolador. A segunda é a Twit - uma API que simplifica a demanda de enviar Tweets para o Twitter. Também usamos Java para a implementação do *servidor*, do qual falaremos mais na parte de implementação (5).

Relativamente ao Arduino, utilizou-se uma biblioteca para estabelecer uma ligação entre o NodeJS, a correr no Raspberry Pi, e o Arduino. Esta biblioteca, *StandardFirmataPlus*, segue o protocolo Firmata para estabelecer esta comunicação entre microcontrolador e host.

Para além disto usamos o software ZeroTier para conseguir estabelecer uma VPN para o Raspberry Pi. Uma vez este configurado com os sensores e Arduino permitiu a ambos os membros do grupo interagir com o sistema.

4 Arquitectura

Quanto à arquitectura, os vários sensores irão estar conectados às várias portas do Arduino, que irá fornecer a energia para a execução correta destes sensores. É visível na figura 4 o sistema. O Arduino estabelecerá uma conexão com o Raspberry Pi e este irá transmitir os dados para a Internet, através do Twitter.

Os sensores que foram utilizados são todos de sinal analógico, à excepção do de humidade e temperatura (uma vez que este estava directamente ligado ao Raspberry Pi). O Arduino tem um ADC(Analog Digital Converter) embutido na própria placa que faz a conversão de sinal analógico para sinal digital.

Os sensores ligados ao Arduino estão a usar todos a mesma voltagem de 3.3 volts, ou seja, estão a usar a intensidade mínima requerida pelos sensores para maximizar o tempo de vida dos mesmos.

A informação depois de ser obtida pelos sensores através do Arduino, é enviada para o Raspberry Pi onde serão processados pela aplicação. Esta ligação entre o Arduino e o Raspberry Pi é feita por uma ligação BUS.

De seguida este computador envia a informação já processada para a conta Twitter da planta.

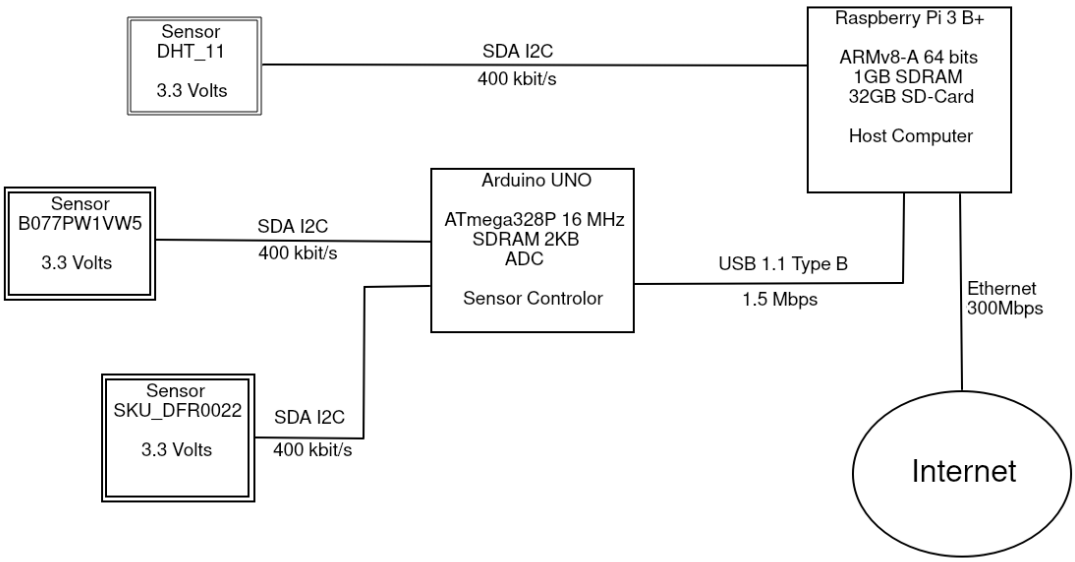

Figure 2: Arquitectura do sistema.

5 Implementação

5.1 Configurações iniciais

Relativamente às configurações iniciais, usamos o RasDebian, um sistema operativo Linux baseado na distribuição Debian, próprio para o sistema Raspberry pi. Depois da instalação do sistema operativo, iniciamos a instalação das dependências do NodeJS.

O Arduino tem como objetivo criar a comunicação entre os sensores e o computador. Uma vez que o Arduino usa uma linguagem própria (baseado na linguagem C) e o nosso objectivo era usar a linguagem NodeJS optamos por dar uso ao protocolo Firmata, mais precisamente o *StanderFirmataPlus* [1] já existente no Arduino. Este protocolo tem como objectivo criar a comunicação com o microcontrolador (Arduino) a partir de software em um computador host (Raspberry Pi).

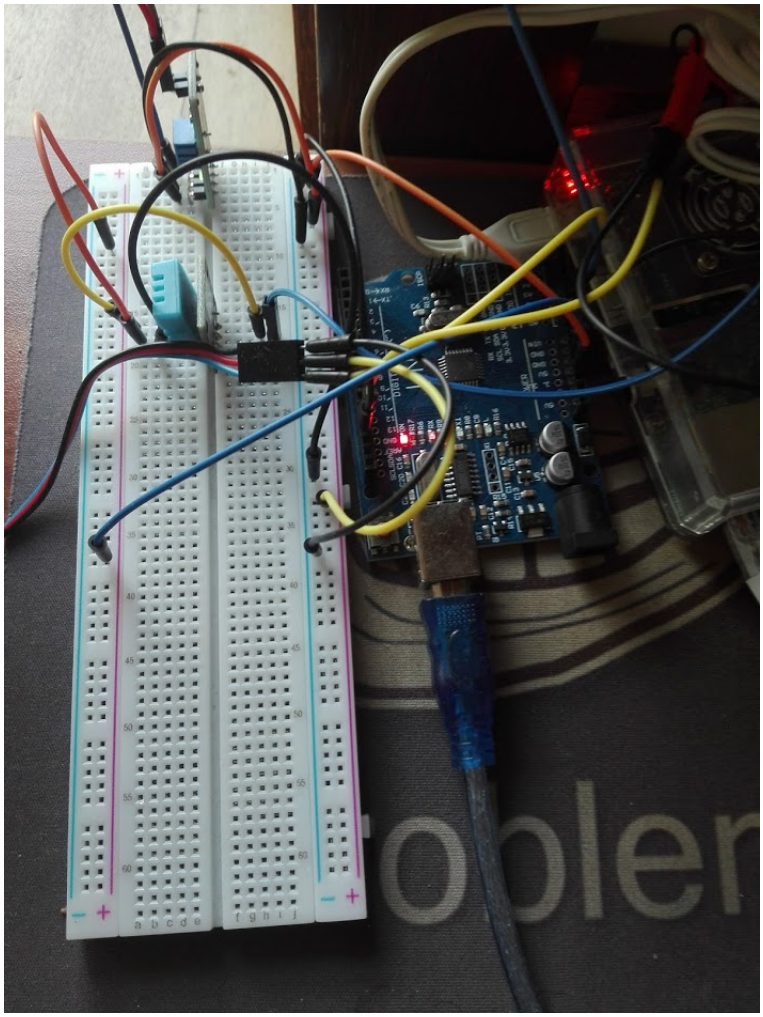

Figure 3: Sistema.

Uma vez que o Raspberry Pi é o computador host, onde irá estar a implementação do sistema e um de nós não podia estar presente devido ao afastamento social imposto pelo COVID-19, tivemos de criar uma conexão via ssh para que ambos pudéssemos trabalhar. Para isso usamos a ferramenta ZeroTier, que cria redes virtuais descentralizadas via software. Estas redes funcionam como fossem redes privadas de uma empresa, em que as maquinas associadas nessa rede possam comunicar entre si, por exemplo por via ssh - que foi o que utilizamos.

Finalmente tivemos de criar uma conta "Twitter developer". O objetivo destas contas é precisamente para que quem está a desenvolver aplicações possa interagir com a plataforma. Para este fim tivemos de ir buscar tanto os *Consumer tokens* como os *Access Tokens*.

A diferença entre os dois é que os primeiros servem para usar o Twitter no sentido lato: por exemplo, se tivéssemos uma aplicação que interagisse com o Twitter mas cada pessoa entrava com a sua conta. Os tokens de acesso por outro lado são tokens que permitem realmente autenticar na conta da nossa planta (@plantamarada).

5.2 Medir dados dos sensores

5.2.1 Scripts

A ideia inicial era implementar tudo em NodeJS, já que ambas as API's que estávamos a usar eram em NodeJS. No entanto mais tarde tomamos a decisão de usar apenas pequenos programas em NodeJS (scripts) para obter os dados, dando antes uso a Java para estabelecer uma plataforma que agendaria as actualizações de estado.

A razão para isto é que nos sentíamos mais confortáveis com java para além de que podíamos separar trabalho criando um "layer de abstracção": enquanto um de nós fazia estes pequenos scripts para obter os dados o outro podia ir indo desenvolvendo o servidor em java.

No total tínhamos três pequenos destes scripts em NodeJS, dois para retirar medições dos sensores e um para fazer os tweets:

- Script para efetuar medições a partir do Raspberry;
  - O Raspberry Pi lê o sensor de humidade e temperatura através de uma função do module "node-dht-sensor". Esta função lê o output do sensor (dht modelo 11) e envia os valores através do pin GPIO 4 do Raspberry Pi. As informações são enviadas usando o protocolo I2C [2].
- Script para efectuar medições a partir do Arduino;

  - No Arduino foi preciso importar o módulo "johnny-five", uma API própria para os diferentes sensores do Arduino e outras placas. Antes de obter os valores, o Arduino precisa de iniciar a placa através do método "board.on" que estará pronta a receber os valores assim que estiver disponível. Uma vez que os valores sejam obtidos (são enviados através dos fios pelo protocolo I2C [2]) procedeu-se a uma chamada do metodo "process.exit(1)" que termina a execução do programa.

- Script para enviar tweets.

  - No momento em que os valores estão reunidos, procede-se ao envio dos mesmos para o Twitter. Neste script é lido um ficheiro com esses valores guardados e envia-os em formato próprio da API "twit" através do método "Twit.post()".

Quanto ao script do Twitter usamos a API *twit*. O que esta API faz essencialmente é simplificar a API do Twitter, nomeadamente a parte da autenticação, que requer o cumprimento de múltiplos passos no protocolo de autenticação. Para além disto a API também implementa interações no Twitter (envio de mensagens, responder a ações como ganhar um "follower", etc). Infelizmente estas interações foram descontinuadas por parte do Twitter, pelo que arranjamos outra forma de personalizar um pouco mais a atividade da planta na rede social. Assim sendo fizemos apenas um pequeno script que dado um ficheiro de texto "tweetava" o conteúdo do mesmo. Para além disto foi necessário pôr os tokens num ficheiro de configuração à parte.

5.3 Backend

Tendo resolvido implementar o servidor em java, a forma como decidimos fazer o backend foi a seguinte:

- Criação de um servidor para fazer testes ao ambiente
- Criação de um servidor de para enviar Tweets

A razão inicial que motivou esta divisão foi o facto de um dos scripts ser problemático, pois o sensor tem um pequeno *cooldown* e quando o script corria durante esse período não retornava resultados de todo. A ideia do servidor de testing passou a ser então continuamente testar o ambiente da planta, guardando esta informação em ficheiros. Para além disto, assim é possível usar os testes ao ambiente para outras aplicações ao mesmo tempo.

Enquanto isto decorre no servidor de Testing, o servidor de tweeting irá buscar estes dados periodicamente (neste momento configurado para duas horas) e fará um tweet. Noutra periodicidade

mais frequente, também o TweetingServer irá verificar estes dados que vão sendo retirados constantemente (20 em 20 minutos). Caso os dados por alguma razão pareçam estranhos, calor a mais, o solo está muito seco, etc, também será enviado um tweet, chamado *tweet de emergência*. Numa tentativa de personalizar um pouco a planta demos um pouco de personalidade aos tweets de emergência.

5.4 Principais desafios

Quanto aos maiores desafios encontrados ao longo do projeto destaca-se termos demorado a perceber de porque é que o servidor de Tweeting terminava, sem dar output de erros. A razão devia-se ao facto de quando um dos scripts o servidor dava erro e terminava. A solução que arranjamos foi fazer uma pequena triagem. Se os resultados das medições viessem vazios repetia-se a medição passado um curto espaço de tempo, passando apenas para um "ficheiro permanente" medições válidas.

Para além disto também foi desafiante termos de pegar em tecnologias com a quais não tínhamos experiência prévia, nomeadamente a configuração do Arduino com o Raspberry e o uso da biblioteca Johnny-Five do NodeJS.

6 Resultados

Quantos aos resultados do trabalho, conseguimos criar uma "smart plant" com sucesso, capaz de publicar o estado no Twitter com a periodicidade programada.

A documentação do Twitter avisa que developers de aplicações não devem abusar da API. Também por isso escolhemos enviar tweets com um período algo alargado entre tweets. Na figura 4 é possível ver a planta a enviar tweets de estado em períodos de duas horas. O último tweet é o um *tweet de emergência*, enviado neste caso, por a planta ter o solo demasiado seco.

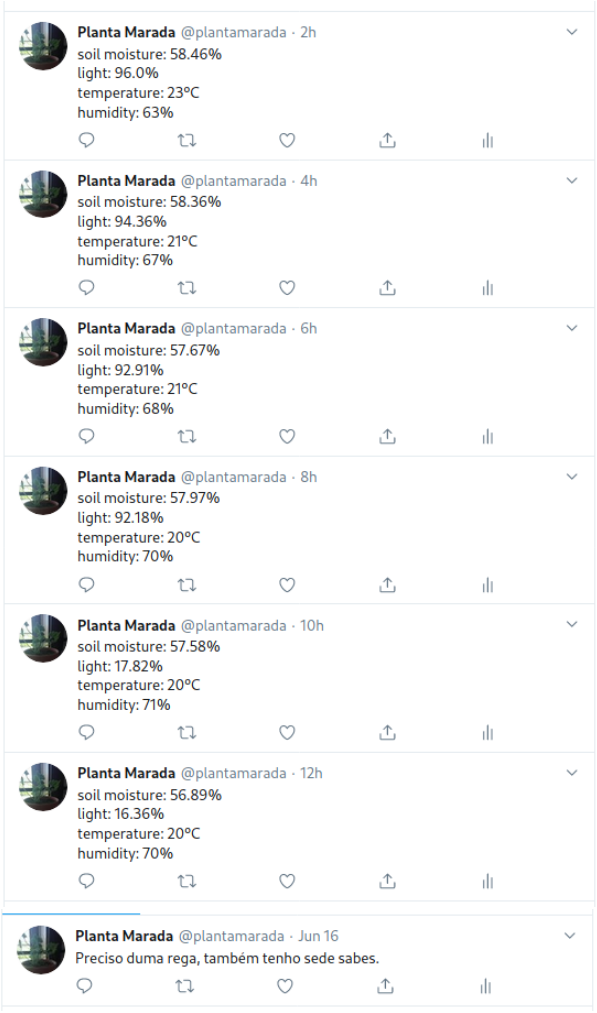

Figure 4: O TwitterServer a trabalhar.

7 Conclusão

Em conclusão, acreditamos que este projecto foi um sucesso, visto termos ultrapassado todos os problemas que tivemos com a implementação, atingindo com êxito a meta final à qual nos tínhamos proposto. Infelizmente não tivemos tempo para fazer uma aplicação mobile mas esse era apenas um objectivo adicional.

Adicionalmente, foi possível obter uma breve e interessante maneira de explorar algumas das tecnologias dos sistemas embutidos, nomeadamente com o Arduino e os sensores. Foi também a primeira vez que trabalhamos com NodeJS.

References

[1] Github, "Firmata protocol for arduino." https://github.com/firmata/arduino. [Online; accessed 09-06-2020].

[2] DroneBot, "Inter-integrated circuit i2c." https://dronebotworkshop.com/i2c-arduino-arduino/. [Online; accessed 15-06-2020].

[3] Twitter, "Build on what's happening." https://developer.twitter.com/en. [Online; accessed 16-06-2020].

[4] J.-F. community, "Build on what's happening." https://johnny-five.io/. [Online; accessed 16-06-2020].