Faculdade de Engenharia da Universidade do Porto

SCADA em Android

Rafael Pisco

V1

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Automação

Orientador: Mário Jorge Rodrigues de Sousa (Professor)

29-06-2015

Resumo

Actualmente com a grande expansão do mercado dos smartphones/tablets quases todos as pessoas possuem um. Isto permite que os smartphones/tablets sejam usados para realizar muitas tarefas que antigamente eram apenas realizadas por computadores ou dispositivos de tamanho médio/grande.
Ao mesmo tempo que o mercado dos smartphones/tablets tem evoluído também a tecnologia têm evoluído como por exemplo ao nível das melhorias do uso de wifi para comunicações modbus TCP/IP.
Esta melhoria aliada ao elevado crescimento do mercado de smartphones/tablets permitiu que estes pudessem ser usados para controlar mecanismos através de comunicações modbus.
No entanto há ainda um grande mercado que pode ser explorado por estas melhorias, o mercado das consolas industriais.
Substituindo a compra de consolas por tablets/smartphones iria baixar substancialmente o custo de aquisição bem como o custo de manutenção. Iria também permitir ao cliente ter uma maior possibilidade de escolha em termos de componentes que podem afectar a performance e em termos de tamanho de ecrãs para o dispositivo, isto pois existe uma maior variabilidade de dispositivos android quando comparado com consolas industriais. Isto iria permitir ao cliente escolher um dispositivo completamente adaptado às suas necessidades, em vez de ter de escolher uma consola de uma lista pré-definida.
No entanto para isto tudo ser possível é necessário adaptar os sistemas SCADA existentes nas consolas industriais para os dispositivos android.
Esta dissertação assenta então sobre essa necessidade, visando a criação de um sistema runtime SCADA semelhante aos usados nas consolas, mas para dispositivos android. Este sistema tem de ter uma performance semelhante à das consolas industriais e tem de conseguir manter essa performance sobre stress, para poder ser viável a substituição das consolas industriais.

Abstract

Nowadays with the big expansion of the smartphones'/tablets' market almost every person was one. This allows that smartphones/tablets to be used to accomplish many tasks that were formerly accomplished by computers or medium/large size devices.
Side by side with the smartphones'/tablets' market expansion, technology has evolving to like at the level of improvements in the use of wifi to communicate via Modbus TCP/IP.
This improvement ally with the growth of the devices' market allowed that these devices could be used to control mechanisms through Modbus' communications. Nonetheless there is still a market to be explored by these improvements, the industrial devices' market.
Replacing the bought of industrial devices with smartphones/tablets would lower significantly the cost of acquisition and the cost of maintenance. Would to, allow the client to have a larger choice in terms of performance components and device screens' sizes, because there is a larger choice of android devices when compared with industrial devices. This large choice would allow the client to choose a device fully adapted to is necessities instead of choosing from a predefined list of industrial devices.
Nonetheless for this to be possible it's necessary to adapt the SCADA systems present on industrial devices to android devices.
This dissertation is based on that necessity, aiming to create a SCADA's runtime system similar of that used in industrial devices but for android devices. This system must have a similar performance of that found in industrial devices and has to maintain that performance in stress situations to be viable to substitute industrial devices.

Agradecimentos

Gostava de agradecer a toda a minha família e a minha namorada pela paciência que tiveram comigo nos momentos de maior stress.

Quero também agradecer ao meu orientador pelo tempo que disponibilizou para discutir comigo o trabalho e pelas ideias que foi dando.

Índice

Lista de figuras

Lista de tabelas

Abreviaturas e Símbolos

Lista de abreviaturas (ordenadas por ordem alfabética)

ADT	*Android Development Tools*
AOSP	*Android Open Source Project*
API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
CPU	*Central Processing Unit*
DDE	*Dynamic Data Exchange*
DLL	*Dynamic-link library*
GNU	*Gnu is Not Unix*
IDE	*Integrated Development Environment*
IP	*Internet Protocol*
LAN	*Local Area Network*
LTS	*Long-Term Support*
ODBC	*Open Data Base Connectivity*
OLE	*Object Linking and Embedding*
OOM	*Out-Of-Memory*
PLC	*Programmable Logic Controllers*
QEMU	*Quick Emulator*
RTDB	*Real-Time DataBase*
RTU	*Remote Terminal Unit*
SCADA	*Supervisory Control and Data Acquisition*
SDK	Software Development Kit
WLAN	Wireless Local Area Network

Capítulo 1

1 Introdução

Neste capítulo faço uma pequena introdução ao tema e explico também o porque de ser um bom tema para ser desenvolvido. Apresento também neste capítulo os objectivos da dissertação bem com a metodologia usado para os atingir. Por fim apresento a estrutura da dissertação.

1.1 Enquadramento

O uso dos smartphones/tablets pelas pessoas no seu dia-a-dia está a sofrer um elevado crescimento, levando a que a maioria das pessoas possua pelo menos um. Este enorme crescimento levou à aumento do número de dispositivos diferentes no mercado.

Por outro lado a evolução tecnológica levou também a uma grande evolução nas comunicações modbus TCP/IP por wifi. Esta evolução tecnológica permitiu que aparelhos equipados com wifi pudessem controlar aplicações modbus.

Este aumento do mercado de smartphones e do avanço nas comunicações modbus podem ser aproveitados em conjunto possibilitando a substituição das consolas em ambientes industriais por smartphones/tablets. Esta substituição tem a enorme vantagem de os smatphones/tablets terem um custo de aquisição muito mais reduzido que o custo

de aquisição de consolas industriais. Por outro lado os smartphones/tablets tem também um custo baixo de manutenção, enquanto as consolas industriais apresentam um elevado custo de manutenção e por vezes essa manutenção só pode ser realizada pelo vendedor da consola. Por outro lado os smartphones/tablets apresentam uma variedade de tamanhos diferentes com características de processamento diferentes quando comparados às consolas industriais existentes no mercado. Esta enorme variedade permite ao consumidor comprar um dispositivo que melhor se adapte as suas necessidades e não uma consola que não cumpre as suas necessidades ou que ultrapassa as suas necessidades.

Apesar de ser uma boa ideia utilizar estas evoluções em conjunto é necessário para isso criar um sistema SCADA idêntico aos presentes nas consolas industriais de maneira a ser possível controlar e monitorizar o ambiente industrial.

1.2 Objetivos

Para este projecto visto que existe um limite de tempo imposto, o objectivo será apenas criar a parte runtime do sistema SCADA deixando para trabalho futuro a criação de um configurador gráfico.

Este runtime deve incluir os elementos básicos e alguns elementos extra dos sistemas SCADA bem como deve conseguir comunicar usando o protocolo modbus TCP/IP via wifi.

Além disto deve também ser possível configurar o sistema através de uma ficheiro XML. Este ficheiro será usado no futuro pelo configurador gráfico para guardar as configurações.

1.3 Metodologia

Para este projecto como existe pouco tempo disponível é necessário usar uma metodologia que permita realizar o trabalho de maneira estruturada e concisa. Por esse motivo a metodologia adoptada foi a seguinte:

- Aprender a programar em ambiente Java e Android;
- Interpretar o problema e pensar em soluções possíveis;
- Estruturar uma arquitectura do sistema usando a melhor solução possível para o problema;

- Programar o sistema tendo por base a arquitectura já realizada, mantendo uma capacidade critica e resolutiva aquando do aparecimento de erros e problemas;
- Testar o sistema em conjunto com o simulador da fábrica e corrigir erros que possam surgir;
- Realizar testes de performance e robustez para ver como o sistema reage sobre stress;

1.4 Estrutura da tese

Esta dissertação está estruturada em cinco capítulos.
No primeiro capítulo é dada uma pequena introdução ao tema, é explicado o motivo deste ser um bom tema. É explicado também quais são os objectivos e quais as metodologias a usar para atingir esses objectivos.

No segundo capítulo é apresentado o estado de arte dos conceitos necessários a desenvolver o tema. Este capítulo dividisse nos temas Android, Modbus e Sistemas SCADA.

No terceiro capítulo é descrito o problema a tratar e apresentada uma solução bem estruturada para resolve-lo. Neste capitulo é apresentado também uma planificação da distribuição do tempo por cada parte do projecto.

No quarto capitulo é demonstrado como foi implementada a solução bem como o software e hardware usado para chegar à mesma. É também demonstrado neste capítulo testes de performance e de robustez realizados ao sistema e os resultados desses testes. Por fim no ultimo capítulo são apresentadas as conclusões retiradas do projecto bem como o trabalho que pode ser realizado futuramente para melhorar e completar o projecto.

Capítulo 2

2 Tecnologias e conceitos associados

Neste capítulo faço uma introdução e apresento os conceitos mais importantes das tecnologias a serem usadas.

2.1 Android

2.1.1 Introdução

Android é um sistema operativo de telemóveis baseado no *kernel* do *Linux* e em desenvolvimento pela Google.

A sua interface do utilizador é baseada em manipulação directa, fazendo com que este seja desenhado principalmente para dispositivos com ecrãs tácteis. O sistema operativo reconhece gestos usados no dia-a-dia para manipular objectos presentes no ecrã e apresenta também um teclado virtual. Apesar de ser desenhado principalmente para dispositivos com ecrãs tácteis também é usado em alguns casos em dispositivos sem ecrãs tácteis. O sistema operativo *Android* é sistemas operativo mais usado para telemóveis, vendendo mais do que os seus competidores juntos.

O código fonte do *Android* é disponibilizado pela Google sobre licenças gratuitas, no entanto maior parte dos dispositivos *Android* acaba por ser vendido como uma combinação de *software* de licença gratuita e de software proprietário, incluindo software proprietário desenvolvido pela Google. Inicialmente desenvolvido pela *"Android, Inc."* e financiado pela Google, acabou por ser comprado pela Google em 2005 e revelado ao público em 2007.

O sistema *Android* é popular nas empresas de tecnologia que requerem um *software* já feito, com baixo preço e facilmente alterado para dispositivos de alta tecnologia. O facto de o *Android* ser de licença aberta encorajou uma enorme comunidade de desenvolvedores e entusiastas a usar o *software* para realizar projectos pedidos pela comunidade que adicionam novas funcionalidades para utilizadores experientes ou a exportar o sistema *Android* para dispositivos que foram oficialmente lançados para o mercado com outros sistemas operativos [1].

2.1.2 Funcionalidades

2.1.2.1 Interface

A interface predefinida do utilizador é baseada na manipulação directa usando toques no ecrã que usamos normalmente no dia-a-dia e um teclado virtual. A resposta aos cliques do utilizador é desenhada para ser imediata e fluida, muitas vezes acompanhada da vibração do telemóvel para dar algum feedback ao utilizador. *Hardware* interno como acelerómetros, giroscópios, sensores de proximidade são usados por algumas aplicações para responder a outras acções do utilizador como por exemplo o esticar do ecrã quando o dispositivo está na horizontal.

Os dispositivos *Android* iniciam no homescreen, o principal ponto de navegação e informação do dispositivo. Este *homescreen* é normalmente constituído por ícones de aplicações e por *widgets*. Os ícones das aplicações abrem essa determinada aplicação quando clicados, enquanto que os *widgets* apresentam informação que se vai alterando automaticamente, como a meteorologia ou o email do utilizador. O *homescreen* pode ser composto por várias páginas bastando apenas um gesto com a mão para alternar entre elas. No entanto este *homescreen* é totalmente alterável, o que possibilitou um grande número de aplicações na *Google Play* para alterar o aspecto do *homescreen* e que várias empresas se diferenciem de outras pelas alterações que realizam ao dispositivo.

2.1.2.2 Aplicações

As aplicações, normalmente designadas *apps,* aumentam as funcionalidades dos dispositivos usando um maior potencial do seu *hardware* e software. As aplicações são principalmente escritas na linguagem de programação Java usando o *Android SDK*. Este *SDK* inclui um número variado de ferramentas, nomeadamente um *debugger*, bibliotecas,

um emulador de dispositivos baseado em *QEMU*, documentação, tutoriais e código exemplo. O *IDE* oficial é o *Eclipse* usando o plugin das ferramentas de desenvolvimento de *Android* (*ADT*). Outras ferramentas de desenvolvimento são suportadas, algumas permitindo o uso de outra linguagem de programação e outras mais visuais, permitindo às pessoas sem quase nenhum conhecimento programar.

2.1.2.3 Gestão de memória

Como todos os dispositivos *Android* são baseados no uso de bateria, o sistema *Android* foi desenhado para gerir a sua memória *RAM* de maneira a manter o consumo de energia no mínimo. Quando uma aplicação não está a ser usada, o sistema suspende-a automaticamente na memória, desta maneira apesar de a aplicação se encontrar carregada não consome recursos. Isto tem um duplo beneficio pois assim, as aplicações não tem de estar sempre a ser abertas do zero sempre que são precisas e garante que as aplicações que não estão a ser usadas não consumam energia desnecessária [2][3].

2.2 Modbus

2.2.1 Introdução

O Modbus define uma arquitectura cliente/servidor em que o cliente é responsável pela organização da rede, pelo fluxo das mensagens e também pelo envio de mensagens aos servidores. Um servidor numa rede Modbus é, usualmente, um módulo de entradas e saídas. A estrutura básica de uma mensagem de Modbus apresenta três campos distintos: o endereço de destino, o "function code" pretendido, uma gama de informação adicional que está intrinsecamente ligado com o "function code" da mensagem [4].

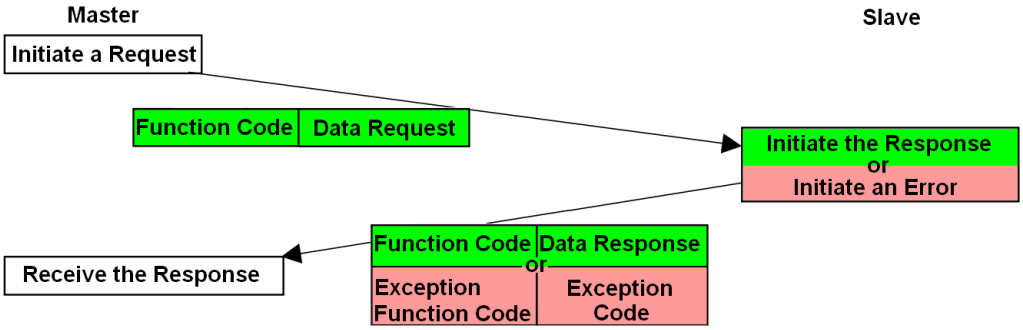


Figura 1 - Arquitectura cliente/servidor modbus

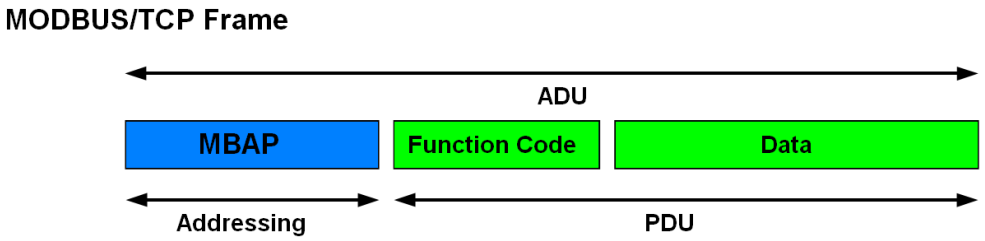


Figura 2 - Trama típica modbus TCP/IP

2.2.2 Function codes

O "function code" é um código de dois bytes que identifica uma função que o destinatário deverá reconhecer. Este "function code" é o que permite ler/escrever diferentes tipos de dados através de modbus [5]. Os "function code" possíveis são:

Function code	Action	Table name
01 (01 hex)	Read	Discrete output coils
05 (05 hex)	Write single	Discrete output coil
15 (0F hex)	Write multiple	Discrete output coils
02 (02 hex)	Read	Discrete output contacts
04 (04 hex)	Read	Analog input contacts
03 (03 hex)	Read	Analog output holding registers
06 (06 hex)	Write single	Analog output holding register
16 (10 hex)	Write multiple	Analog output holding registers

Figura 3 - Function Codes possíveis para mensagens modbus

2.2.3 Error codes

Quando não é possível criar uma mensagem modbus é enviado uma trama especial com o sétimo byte igual a 0x80 e o oitavo byte igual ao código da excepção. As excepções possíveis são as presentes na tabela seguinte.

Exception Code	Nome
0x01	Illegal Function
0x02	Illegal Data Address
0x03	Illegal Data Value
0x04	Slave Device Failure
0x05	Acknowledge
0x06	Slave Device Busy
0x07	Negative Acknowledge
0x08	Memory Parity Error
0x0A	Gateway Path Unavailable
0x0B	Gateway Target Device Failed to Respond

Tabela 1 - Códigos de excepção possíveis

2.3 Sistema SCADA

2.3.1 Introdução

SCADA é um acrónimo para *Supervisory Control and Data Acquisition*. Estes sistemas são usados para monitorizar e controlar a planta ou equipamentos nas indústrias como as telecomunicações, água e controlo de desperdícios, energia, refinarias de óleo e gás e transporte [6]. Estes tipos de sistemas envolvem a transferência de dados entre o computador central e um número RTUs e/ou PLCs, e também a transferência de data entre o computador central e os terminais dos operadores. Podem ser sistemas relativamente simples, como sistemas para monitorizar as condições ambientais de um pequeno escritório, ou muito complexos, como sistemas para monitorizar todas as actividades numa central nuclear ou as actividades do sistema de águas municipal [7][8][9].

Estes sistemas são constituídos por:

- Interfaces Homem/Máquina – Interface que apresenta os dados dos processos para um operador humano, e através disto, o operador monitoriza e controla o processo;
- Computador central (supervisor) – Computador que recolhe todos os dados dos processos e envia comandos para os processos;
- RTUs – Conecta-se aos sensores nos processos, e converte os sinais dos sensores em dados digitais e envia esses dados para o sistema supervisor;
- PLCs – Usam-se como dispositivos de campo porque são mais económicos, versáteis, flexíveis e configuráveis que os RTUs especializados;

- Infra-estrutura de comunicação – Usam-se para efectuar a ligação entre o sistema supervisor e os RTUs;

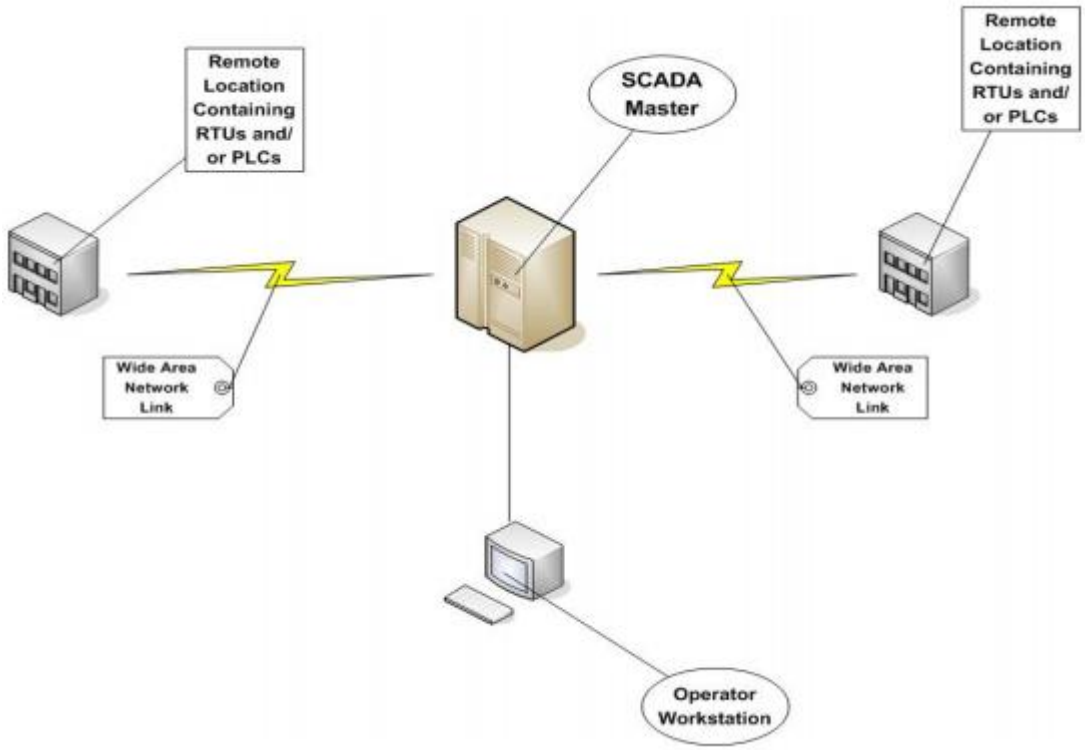


Figura 4- Sistema SCADA básico

2.3.2 Componentes do sistema SCADA

2.3.2.1 Dispositivos de campo

Os dispositivos de campo são os "olhos e ouvidos" de um sistema *SCADA*. Dispositivos que meçam a posição, temperatura, potência, pressão, fornecem informação que indica ao utilizador como um certo sistema se está a comportar, daí a serem os olhos e ouvidos. Por outro lado também são as "mãos" de um sistema *SCADA*, isto pois os actuadores permitem ao utilizador actuar sobre o sistema. No entanto esta informação de/para os dispositivos de campo necessita ser compatível com a linguagem do sistema *SCADA*. Para atingir esse objectivo são usados *Remote Terminal Units* que transformam os sinais electrónicos provenientes dos dispositivos de campo na linguagem usada para transmitir os dados através de um canal de comunicação. As instruções para a automação dos dispositivos de campo, como a lógica de controlo de determinado dispositivo são armazenadas localmente em *PLCs*, devido ao limitação da largura de banda na comunicação entre o computador central e os dispositivos de campo. À medida que as aplicações começaram a exigir maior automação, os *PLCs* começaram a ser mais usados o

que levou a um maior uso da telemetria nos *PLCs* em ambientes remotos. Tornou-se desejável influenciar o programa dentro do *PLC* através do uso de um sinal remoto. Isto é na verdade a parte do acrónimo *SCADA* que se refere ao *"Supervisory Control"*. Isto pois a partir do momento em que apenas um simples programa local era necessário para controlar o sistema, tornou-se possível armazenar esse mesmo programa num *RTU* e efectuar o controlo a partir desse mesmo dispositivo. Ao mesmo tempo os *PLCs* incluíam os módulos de comunicação que permitiam ao *PLC* reportar o estado do programa de controlo ao um computador conectado fisicamente ou remotamente ao *PLC* [10][11][12].

2.3.2.2 Redes de comunicação

A rede de comunicação é usada para transmitir os dados entre os servidores do computador central e os dispositivos de campo. Esta rede pode ser cabo, telefone ou rádio. Os cabos são mais usados em fábricas, mas não são usados em sistema com larga cobertura geográfica devido ao custo elevado dos cabos e da sua instalação. O uso de linhas telefónicas é mais económico para soluções de sistemas com grande cobertura. Locais remotos não são normalmente acessíveis através de cabos telefónicos. No caso de ser necessário aceder a locais remotos a solução mais económica e normalmente usada são o uso de modens de radio.

Apesar de historicamente as redes de *SCADA* serem redes dedicadas, com o aumento do uso das redes *LAN* e *WLAN* para comunicação entre escritórios, existe a possibilidade de integrar as redes *LAN* de *SCADA* nas redes do dia-a-dia dos escritórios. Trazendo isto uma enorme vantagem, quer ao nível da facilidade da integração com as aplicações já existentes nos escritórios, como na remoção da necessidade de um computador apenas como terminal para os operadores [10][11][12].

2.3.2.3 Computador central

O computador central é normalmente um computador ou uma rede de computadores que fornece ao operador a interface com o sistema *SCADA*. O computador processa todos os dados recebidos e enviados pelos *RTUs* e apresenta-os ao operador de uma maneira que ele os perceba e possa trabalhar com eles.

Antigamente os vendedores de *SCADA* ofereciam *hardware* proprietário, sistemas operativos e software que muitas vezes eram incompatíveis com outros sistemas *SCADA*. No entanto com o grande aumento do uso do computador pessoal, as redes de internet passaram a ser usuais nos escritórios, o que levou a que os sistemas *SCADA* pudessem passar a utilizar essa mesma rede. Por esse motivo hoje em dia a maioria dos sistemas *SCADA* pode ser instalado e usado em servidores idênticos aos usados nos escritórios normalmente, o que abriu um leque de possibilidades em termos das combinações que se poderiam fazer entre o sistema *SCADA* e as aplicações do dia-a-dia dos escritórios [10][11].

2.3.2.4 Estação de trabalho

As estações de trabalho do operador são normalmente terminais que estão ligados ao computador central. Nesta ligação o computador central funciona como um servidor e os terminais funcionam como clientes. Estes clientes pedem/enviam dados para o computador central consoante os comandos do operador [10][11][13].

2.3.2.5 Componentes de software

Dependendo da grandeza e da natureza da aplicação *SCADA*, o *software* pode ter um custo significante ao ser desenvolvido, mantido e expandido. Isto pois para que um sistema *SCADA* seja produzido é necessário o uso de um *software* bem definido, bem desenhado, bem documentado, bem verificado e bem testado. Maior parte dos sistemas *SCADA* emprega *software* comercial proprietário com o qual o sistema é desenvolvido, o que leva a que maior parte das vezes este software não seja compatível com o *hardware* e software dos seus competidores no mercado [14][15]. Logo é necessário planear bem qual o *software* a se usar, e para isso deve-se ter em conta:

- Sistema operativo do computador central;
- Sistema operativo dos terminais do operador;
- Aplicações que tratam da transmissão e recepção dos dados de e para os *RTUs* e o computador central;
- Aplicações que permitem aos operadores acederem a informação disponível no computador central;
- *Drivers* de protocolos de comunicação;
- *Software* de controlo das redes de comunicação;
- *Software* de automação dos *RTUs*;

2.3.3 Evolução das arquitecturas do sistema SCADA

Os sistemas SCADA têm vindo a evoluir em paralelo com a evolução da tecnologia. Durante a sua evolução apareceram três grandes gerações de arquitecturas *SCADA*:

- Primeira Geração – Monolítica;
- Segunda Geração – Distribuída;
- Terceira Geração – *Networked*;

2.3.3.1 Sistemas SCADA monolíticos

Quando os sistemas SCADA foram desenvolvidos o conceito de computação estava geralmente centrado em sistemas "*mainframe*" e as redes geralmente não existiam, o que levava a que cada sistema centralizado fosse autónomo sem quase nenhuma ligação a outros sistemas.

As redes WAN que eram implementadas para comunicar com os *RTUs* serviam apenas para comunicar com os RTUs no campo e nada mais, isto pois os protocolos *WAN* usados hoje em dia eram desconhecidos na altura.

Os protocolos de comunicação destes sistemas eram desenvolvidos pelos vendedores dos *RTUs* e eram maioritariamente proprietários. Essas características, levavam a que esses protocolos fossem muito rígidos não suportando qualquer funcionalidade para além de pesquisar e controlar os dispositivos remotos e impossibilitando outros tipos de tráfego de dados com comunicação *RTU* nessa rede.

A conectividade ao computador central era também muito limitada pelos vendedores, isto pois geralmente essa ligação só podia ser efectuada ligando um adaptador próprio ao *CPU*.

A redundância nesta geração era feita usando dois sistemas mainframe, o principal e o de *backup*. A função do sistema de backup era apenas a de monitorizar o sistema primário e detetar um evento de falha, o que significava que quase nenhum processamento era realizado neste sistema de backup [15][16].

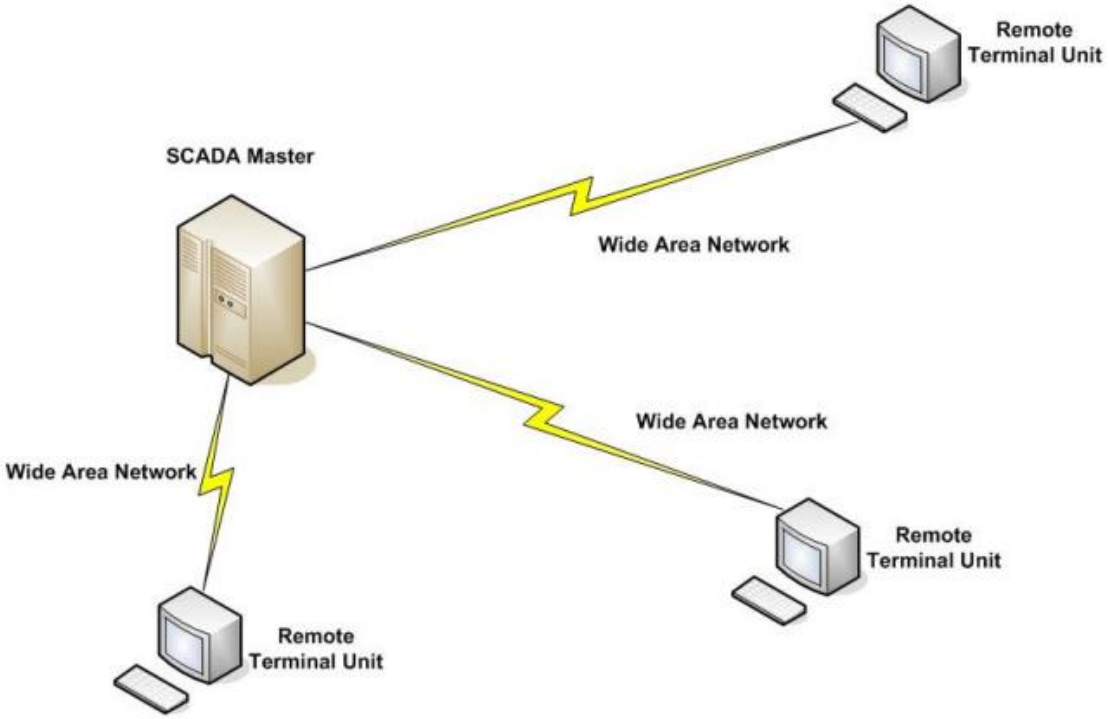


Figura 5 - Arquitectura SCADA de primeira geração

2.3.3.2 Sistemas SCADA distribuídos

Esta geração tirou partido dos avanços tecnológicos ao nível da miniaturização dos sistema e ao nível da tecnologia das redes *LAN* para distribuir o processamento por vários sistemas. Várias estações, cada uma com a sua função, eram ligadas a uma *LAN* e partilhavam informação umas com as outras em tempo real. Como cada uma tinha a sua função as estações passaram a ser tipicamente mini computadores, muito mais baratos e pequenos que os da primeira geração.

Esta distribuição das funcionalidades do sistema por sistema ligados a rede *LAN* serviu não apenas para aumentar a capacidade de processamento (vários processadores com simples funcionalidades vs um processador), mas também para aumentar a redundância e a confiança no sistema como um todo. Isto permite a que ao contrário do sistema primário/backup usado nos sistemas de primeira geração, se tivesse uma arquitectura distribuída que mantinha todas as estações num estado online, levando a substituição de uma estação por outra em caso de falha, sem necessidade de esperar da transição do sistema primário para o de backup.

No entanto a comunicação entre estações tinha de ser realizada localmente na rede *LAN* e a rede *WLAN* era apenas usada unicamente para comunicação entre os *RTUs* e um servidor de comunicações na rede *LAN*, não havendo qualquer estação remota.

Outro problema era que os protocolos usados na rede *LAN* eram normalmente proprietárias, por um lado permitindo optimizar o tráfego em tempo real da rede, mas por outro lado limitando ou até mesmo eliminando a possibilidade de conexão com redes *LAN* de outros vendedores. Logo a segunda geração de sistemas *SCADA* continuava a ser limitada ao nível do hardware, software e dispositivos periféricos pelos vendedores [16][17][18].

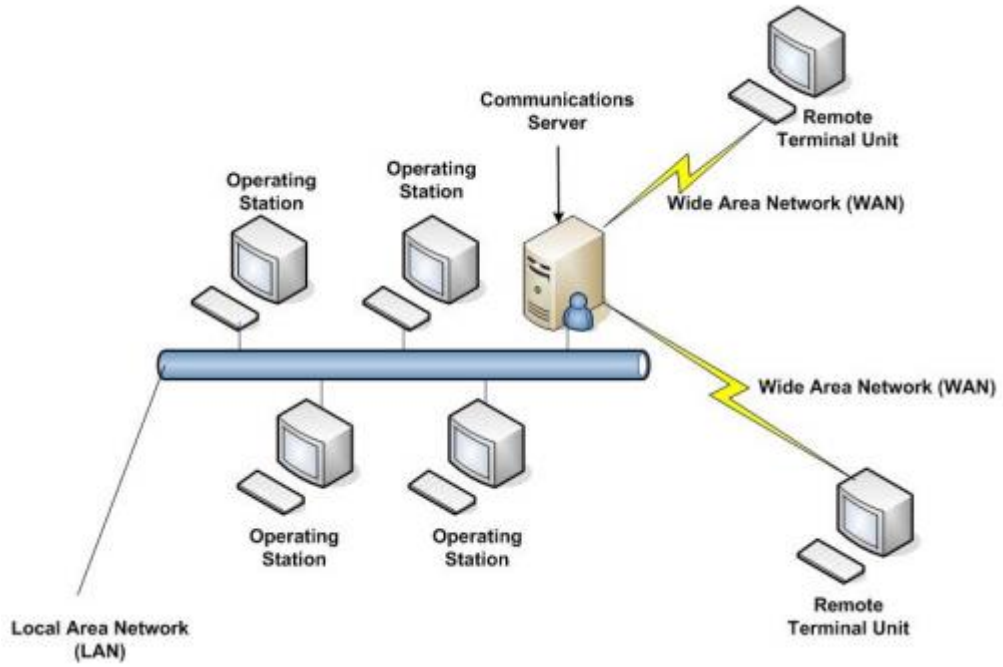


Figura 6 - Arquitectura SCADA de segunda geração

2.3.3.3 Sistemas SCADA Networked

A actual geração dos sistemas SCADA apresenta a maior diferença de usar protocolos abertos e não proprietários. Ainda existem múltiplos sistemas partilhando as funcionalidades do computador central, e RTUs que usam protocolos proprietários. O maior avanço desta geração foi tornar a arquitectura do sistema aberta, utilizando *standarts* e protocolos abertos o que possibilitou distribuir as funcionalidades através da

rede *WLAN* e não apenas da rede *LAN*. Tornar o sistema aberto facilitou também a conexão de dispositivos periféricos ao sistema e/ou a rede.

As maiores vantagens destas geração advém da possibilidade do uso de protocolos *WLAN* como o *Internet Protocol* (IP) para comunicar entre estação principal e os equipamentos de comunicação. Isto permitiu a que uma parte da estação principal responsável pela comunicação com os dispositivos de campo pudesse estar dispersa por toda a rede *WLAN* [16][17][18].

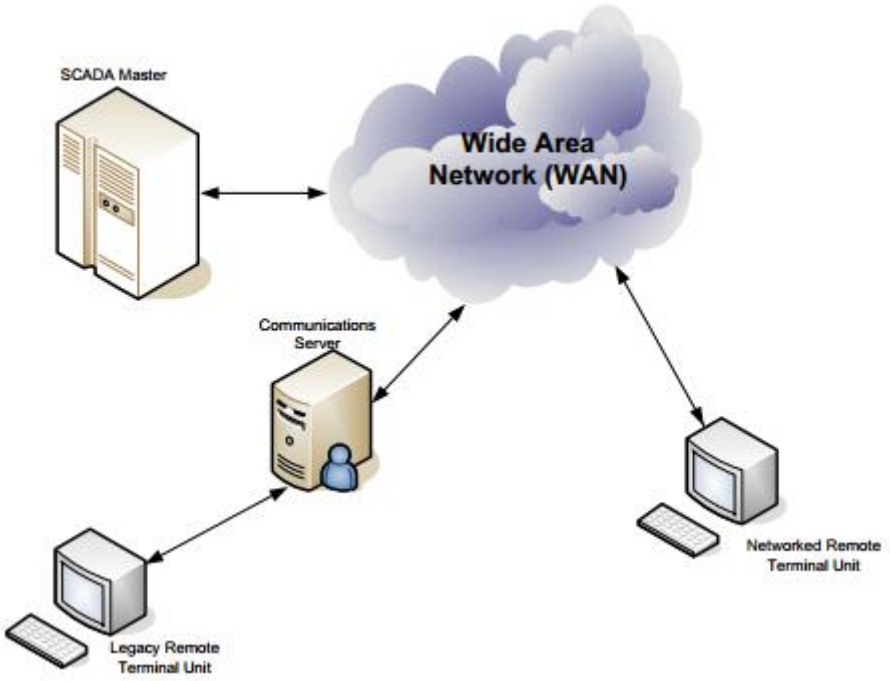


Figura 7 - Arquitectura SCADA de terceira geração

Capítulo 3

3 Descrição do problema

Neste capitulo vou descrever o problema e elaborar soluções estruturadas para o mesmo, bem como elaborar um planeamento do tempo que tenho disponível.

3.1 Requisitos do sistema

- O sistema deve incluir os elementos essenciais de um sistema SCADA (botão, luz);
- O sistema deve incluir os elementos extra de um sistema SCADA (imagem, forma e texto, etc);
- O sistema deve incluir dois tipos de tags (digital, numérica);
- O sistema deve conseguir usar quatro funções modbus (leitura de coils – 0x01, leitura de entradas discretas – 0x02, leitura de registos de entrada – 0x04, escrita de uma coil – 0x05);
- Os estados dos elementos botão e luz devem poder depender de uma tag digital;
- Os gradientes dos elementos forma devem poder depender de uma tag numérica;
- Os valores dos elementos texto devem poder depender de uma tag numérica;
- As visibilidades dos elementos devem poder depender de uma tag digital;
- A posição no eixo x e a posição no eixo y devem poder depender de uma tag numérica;

3.2 Arquitectura do sistema

De maneira a desenvolver uma solução de maneira estruturada que cumprisse todos os requisitos, criei um diagrama de classes UML que mostra a solução dentro das possíveis que achei mais adequada ao problema.

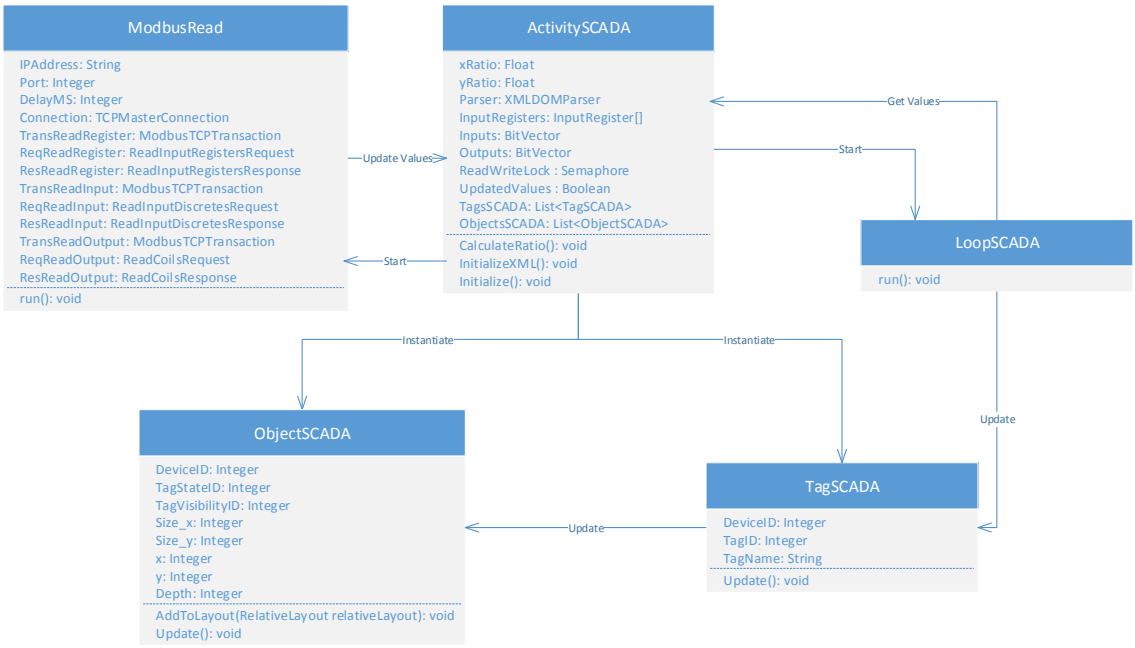


Figura 8 - Diagrama de classes do sistema

Como se pode observar pela Figura 8 a classe ActivitySCADA é a classe principal que gera todas as outras. Esta classe é responsável por calcular a razão usada para manter a proporcionalidade entre vários ecrãs, por ler o ficheiro XML e inicializar os objectos e por adicionar os objectos ao layout. Esta classe contém também os valores de entrada, saídas e registos lidos pelo modbus, bem como um semáforo que impede que a escrita e leitura desses mesmos valores lidos sejam simultâneas.

Podemos também observar que as classes ModbusRead e LoopSCADA são threads que correm separadas da main thread. Estas classes foram projectadas desta maneira com o propósito de não causarem atrasos na thread principal, sendo que a classe ModbusRead é responsável pela alteração dos valores das variáveis lidas e pelas comunicações modbus e a classe LoopSCADA é a responsável por usar os valores lidos para actualizar as tags e objectos correspondentes.

Criei também um diagrama de classes que mostra que tipos de objectos existem e um diagrama de classes que mostra que tipo de tags existem, como se pode observar pela Figura 9 e Figura 10 respectivamente.

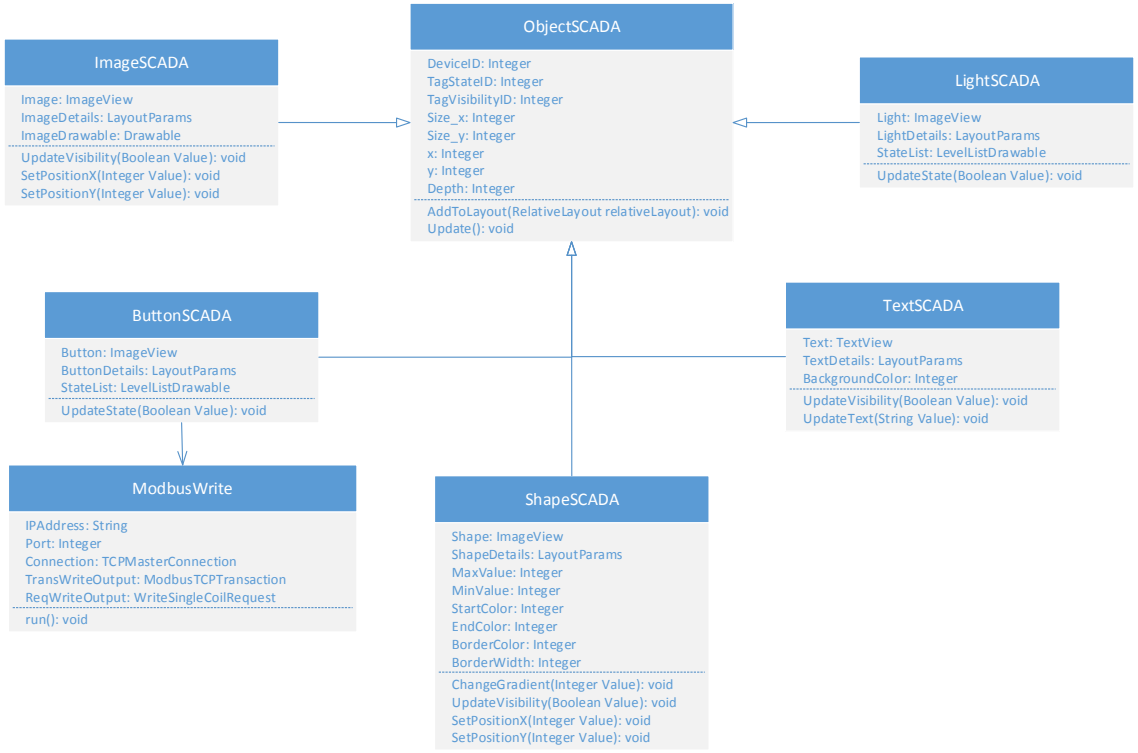


Figura 9 - Diagrama de classes dos objectos

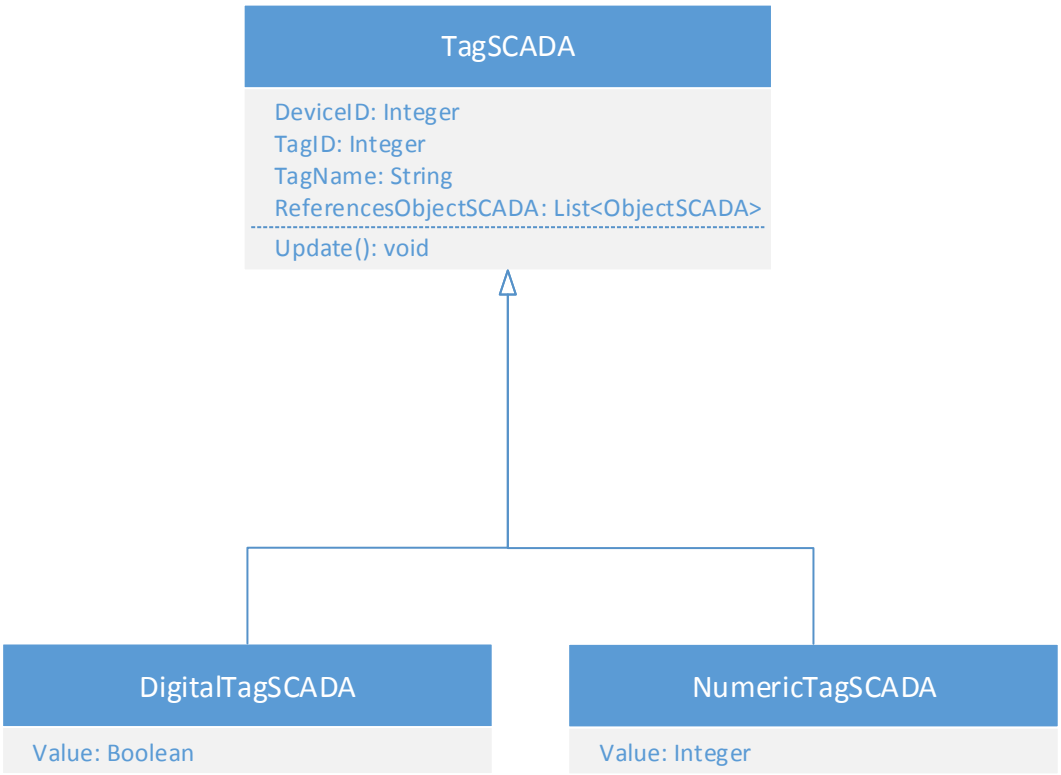
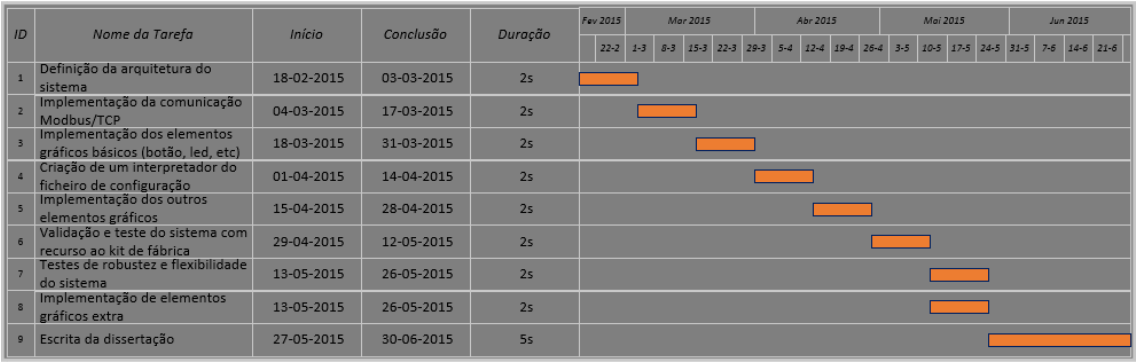


Figura 10 - Diagrama de classes das tags

3.3 Planificação

De maneira a distribuir correctamente a quantidade de trabalho pelo tempo disponível criei um diagrama de Gantt. Este diagrama permite-me uma melhor gestão do tempo, mantendo assim uma carga de trabalho idêntica durante todas as semanas disponíveis.



ID	Nome da Tarefa	Início	Conclusão	Duração
1	Definição da arquitetura do sistema	18-02-2015	03-03-2015	2s
2	Implementação da comunicação Modbus/TCP	04-03-2015	17-03-2015	2s
3	Implementação dos elementos gráficos básicos (botão, led, etc)	18-03-2015	31-03-2015	2s
4	Criação de um interpretador do ficheiro de configuração	01-04-2015	14-04-2015	2s
5	Implementação dos outros elementos gráficos	15-04-2015	28-04-2015	2s
6	Validação e teste do sistema com recurso ao kit de fábrica	29-04-2015	12-05-2015	2s
7	Testes de robustez e flexibilidade do sistema	13-05-2015	26-05-2015	2s
8	Implementação de elementos gráficos extra	13-05-2015	26-05-2015	2s
9	Escrita da dissertação	27-05-2015	30-06-2015	5s

Figura 11 - Diagrama de Gantt

Capítulo 4

4 Implementação da solução

Neste capítulo vou descrever como procedi para implementar e verificar a solução estruturada no capítulo anterior.

4.1 Software usado

4.1.1 Android Studio

IDE escolhido para desenvolver o código e compilar a aplicação.

Dentro do enorme número de opções disponíveis as duas opções mais completas e robustas são o Eclipse e o Android Studio.

Apesar de serem ambas completas o Android Studio tem as vantagens de permitir *debugging* em tempo real quando conectado por um cabo usb, de compilar o código em Android Package (.apk) pronto a ser instalado por dispositivos físicos e de ter embutido um sistema de criação e emulação de dispositivos android. Apesar de ser possível obter todas estas funcionalidades no Eclipse seria necessário instalar plugins e aplicações externas ao IDE, tendo por isso escolhido o Android Studio.

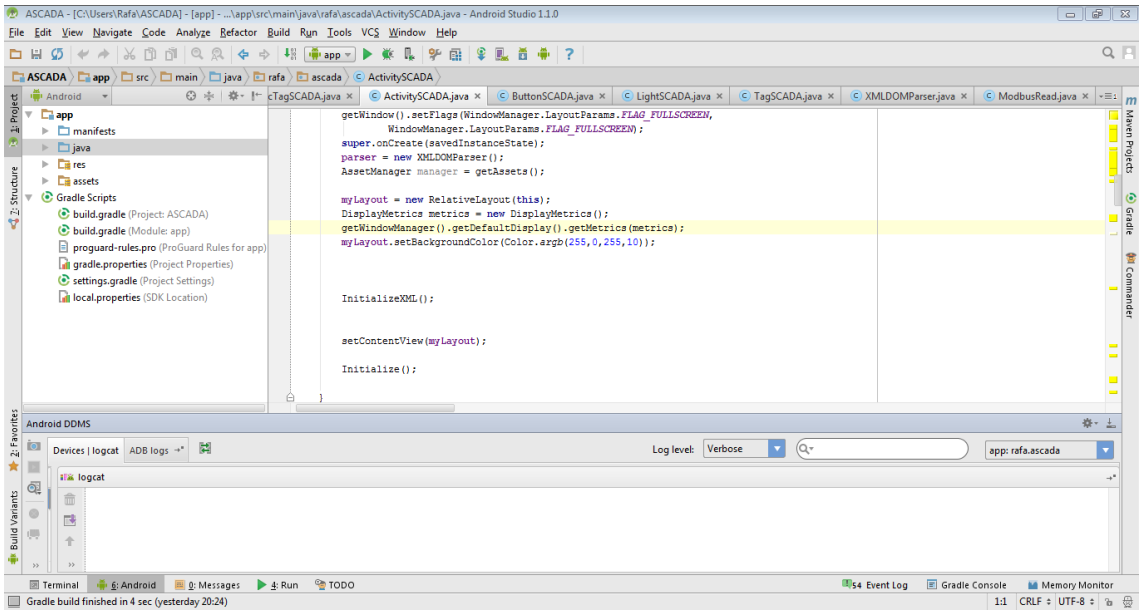


Figura 12 - Android Studio

4.1.2 Simulador

Simulador de um kit fábrica presente na sala de automação I005.

Todas as entradas e saídas do simulador são iguais às entradas e saídas do kit, permitindo que todo o código feito para o simulador possa ser usado directamente no kit.

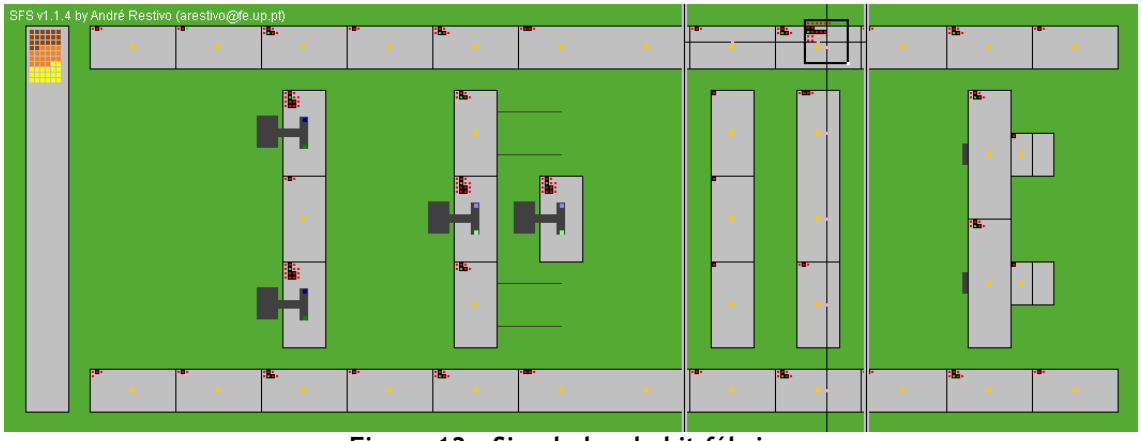


Figura 13 - Simulador do kit fábrica

4.1.3 Unity Pro XL

Software que permite programar em usando línguas IEC61131-3, bem como simulador um PLC no computador. Usei este software em conjunto com o simulador, de maneira a automatizar a fábrica. A vantagem de automatizar a fábrica é poder apresentar no ecrã do dispositivo android, simulações de movimentos das peças e não apenas dos sensores da fábrica.

A razão para ter escolhido este software e não outro qualquer, foi por já ter trabalhado com ele noutras cadeiras, não tendo assim de aprender a trabalhar com ele.

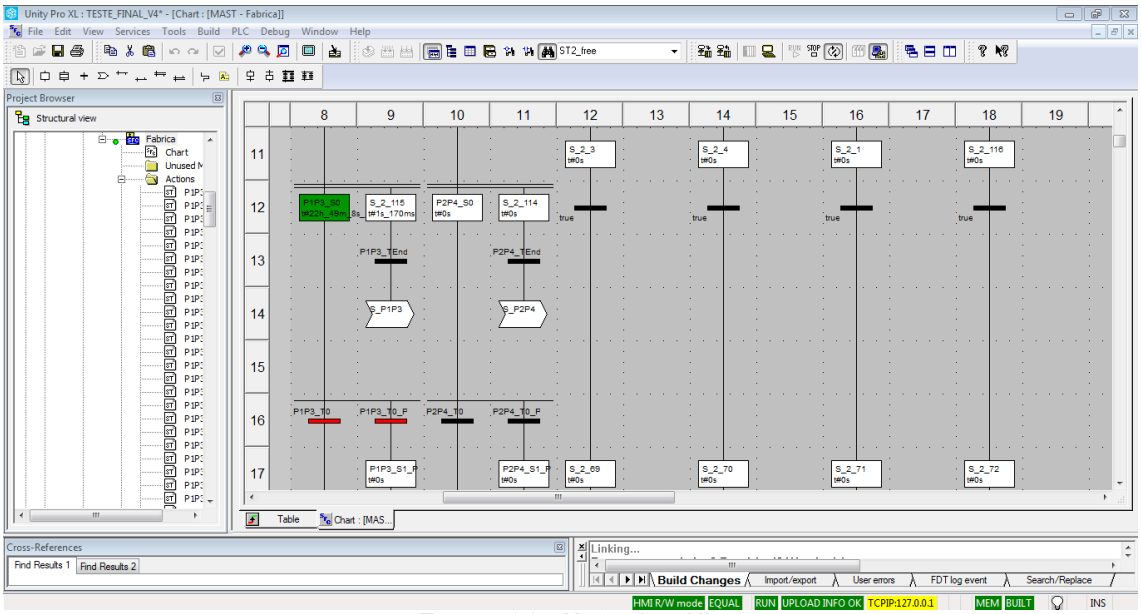


Figura 14 - Unity Pro XL

4.2 Hardware usado

4.2.1 Computador portátil

Computador usado para desenvolver o código do projecto. Usei também o computador para correr o programa do Unity Pro XL e o simulador da fábrica. No inicio usei-o para emular vários tipos de dispositivos android de gama baixa até alta, e com vários tamanhos de ecrã. Após vários testes em dispositivos emulados, usei o computador para compilar e transmitir o código para os dispositivos físicos de maneira a testar o código sem qualquer ligação física ao computador.

4.2.2 Smartphone E-Star x35

Dispositivo físico usado para testar o sistema. Este dispositivo foi escolhido por mim para testar o sistema pois é um dispositivo de gama baixa, apresentando fracas características como se pode ver pela folha de características presente nos anexos. O interesse de usar um dispositivo de gama baixa é verificar como é que o sistema se comporta em dispositivos de baixo custo, já que no emulador foram criados dispositivos de gama média/alta para testar o sistema.

4.2.3 Tablet E-Star Beauty Dual-Core

Dispositivo físico também usado para testar o sistema. Este dispositivo foi escolhido por mim não só por ser também de baixa gama, como se pode verificar pela folha de

características presente nos anexos, mas também para testar como se comporta a conversão do tamanho dos objectos consoante o tamanho do ecrã em dispositivos físicos.

4.3 Trabalho desenvolvido

4.3.1 Comunicação modbus

Ao nível das comunicações modbus foi primeiramente integrado com o ambiente android uma libraria modbus chamada jamod. Após integrada a libraria, realizei pedidos modbus através dela para ver como se comporta a libraria em ambiente android. A libraria apresentou bons resultados e por ser open source permitiu a alteração de código quando necessário. O único problema da libraria foi um erro que só acontece em ambiente android, em que as respostas se misturam umas com as outras, mas após pesquisar pela internet consegui encontrar umas alterações a serem realizadas no código da libraria que resolveram o problema.

4.3.2 Runtime SCADA

Ao nível do runtime SCADA e tendo por base o diagrama de classes do capitulo anterior, criei um runtime orientado a objectos, modular e que mantém uma razão entre o tamanho dos objectos em todos os dispositivos.

O runtime foi orientado a objectos para que as tags que sofrem alterações de valores apenas actualizem os objectos dependentes delas, sem terem qualquer conhecimento dos métodos que os objectos usam para actualizar.

Este tipo de programação orientada a objectos permitiu que o sistema fosse modular, podendo-se adicionar novas classes apenas criando a classe e mudando a função InitializeXML.

Este runtime mantém também a razão entre o tamanho dos objectos em todos os dispositivos, permitindo uma experiência similar aquela do dispositivo para onde foi primeiramente desenvolvida em qualquer outro dispositivo, como se pode observar pelos exemplos das Figuras 15 e 16.

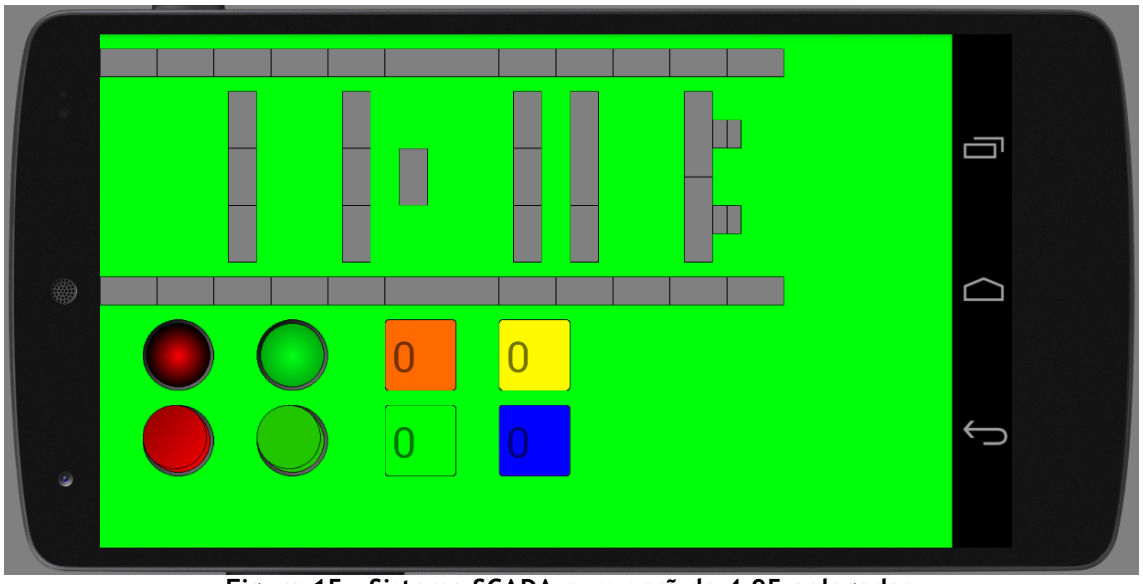


Figura 15 - Sistema SCADA num ecrã de 4,95 polegadas

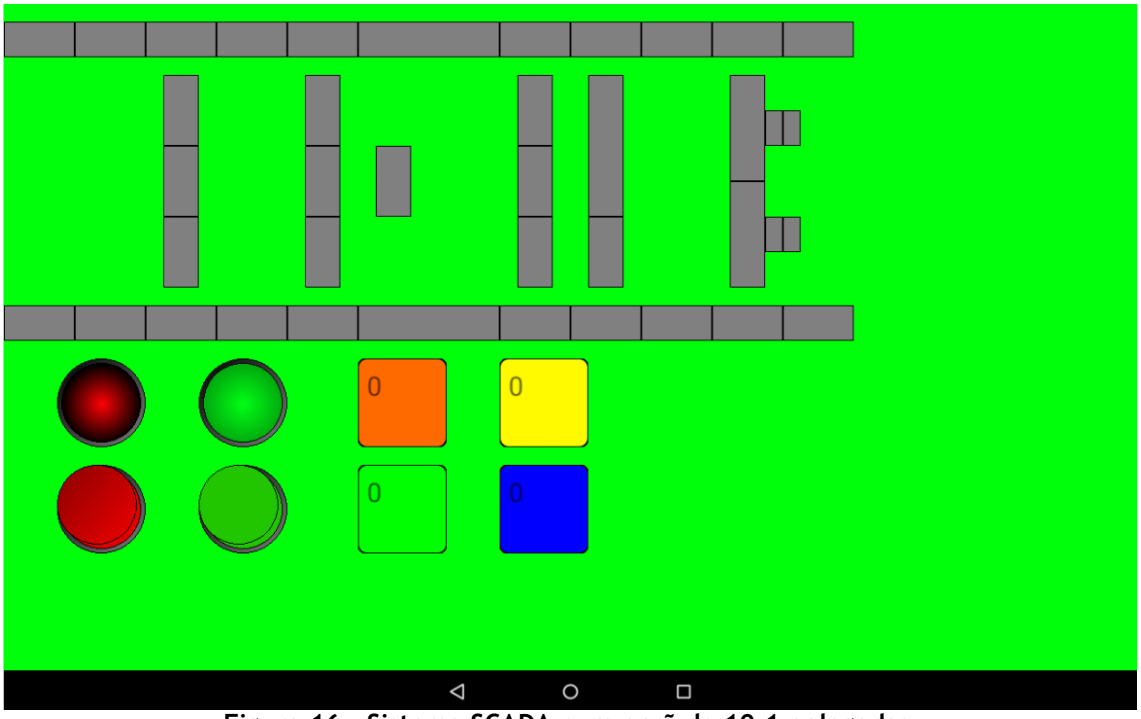


Figura 16 - Sistema SCADA num ecrã de 10,1 polegadas

Este runtime permite também o uso de cinco tipos de objectos diferentes que derivam da classe ObjectSCADA. Estes cinco objectos são:

- ButtonSCADA (implementação do objecto botão dos sistemas SCADA)
    - Dois estados possíveis (Ligado/Desligado) – tag digital;
    - Dois tipos possíveis (Switch/Pressão) - byte;
    - Escrita de coil – tag digital;
    - Posição no eixo dos Z – int;
- LightSCADA (implementação do objecto led dos sistemas SCADA)
    - Dois estados possíveis (Ligado/Desligado) – tag digital;

    - Posição no eixo dos Z – int;
- ImageSCADA (implementação do objecto imagem dos sistemas SCADA)
    - Um estado possível;
    - Dois estados de visibilidade (Visível/Invisível) – tag digital;
    - Movimento pelo eixo dos x – tag numérica;
    - Movimento pelo eixo dos y – tag numérica;
    - Posição no eixo dos Z – int;
- ShapeSCADA (implementação do objecto forma dos sistemas SCADA)
    - Vários estados possíveis (gradiente de duas cores) – tag numérica;
    - Três tipos possíveis (Rectângulo/Rectângulo redondo/Oval) - byte;
    - Dois estados de visibilidade (Visível/Invisível) – tag digital;
    - Movimento pelo eixo dos x – tag numérica;
    - Movimento pelo eixo dos y – tag numérica;
    - Posição no eixo dos Z – int
- TextSCADA (implementação do objecto texto dos sistemas SCADA)
    - Um ou vários estados possíveis (estático/variável) – tag numérica;
    - Dois estados de visibilidade (Visível/Invisível) – tag digital;
    - Movimento pelo eixo dos x – tag numérica;
    - Movimento pelo eixo dos y – tag numérica;
    - Posição no eixo dos Z – int;

Permite também o uso de dois tipos de tags diferentes que derivam da classe TagSCADA. Essas tags são:

- DigitalTagSCADA (implementação da tag digital dos sistemas SCADA)
    - Dois estados possíveis (True/False);
    - Nome da tag – string;
    - ID da coil – int;
- NumericTagSCADA (implementação da tag digital dos sistemas SCADA)
    - Duzentos e cinquenta e seis estados possíveis (0…255);
    - Nome da tag – string;
    - ID do registo – int;

De maneira a se entender melhor como funciona o runtime criei um diagrama de actividades que mostra como é que o sistema se comporta desde que é ligado até ser desligado.

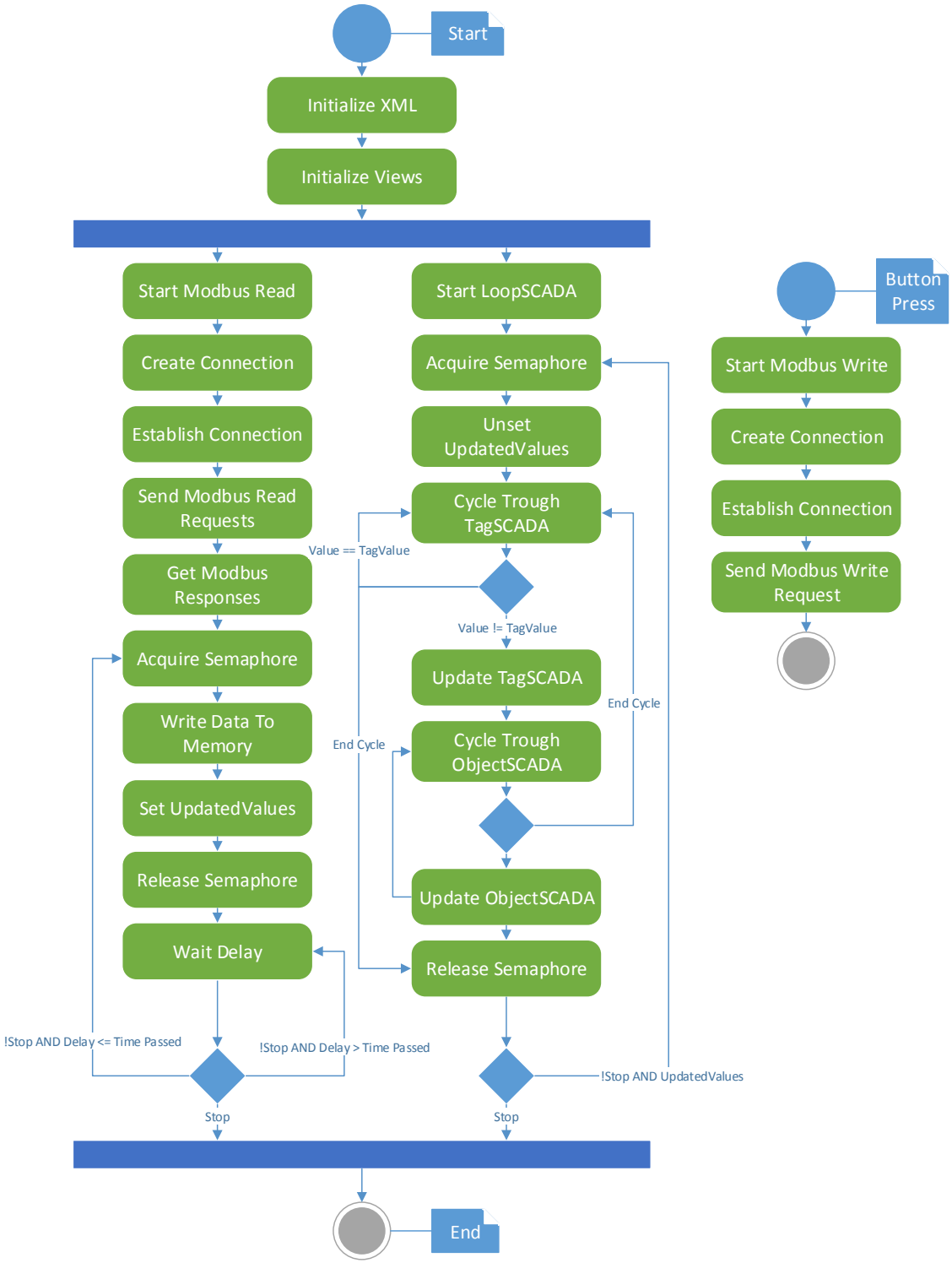


Figura 17 - Diagrama de actividades do sistema

Como se pode observar pelo diagrama da Figura 17, quando o sistema é iniciado calcula a razão nos eixos x e y, lê o ficheiro de configuração e cria os objectos e por fim adiciona-os ao layout.

De seguida são iniciadas duas threads que correm em paralelo com a thread principal.

A thread ModbusRead cria uma conexão ao servidor ao qual vai realizar pedidos, usando os valores de IP e porta presentes no ficheiro de configuração e por fim conecta-se. Esta conexão é então usada para enviar pedidos e receber respostas através do protocolo modbus TCP/IP. Após serem recebidas todas as respostas a thread espera até o semáforo de leitura e escrita estar livre e bloqueia-o. Depois de bloqueado o semáforo a thread escreve o valor das respostas nas variáveis respectivas, avisa que já são valores actuais através dum parâmetro booleano (UpdatedValues) e desbloqueia o semáforo. Após tudo isto a thread espera passar um delay especificado no ficheiro de configuração e repete o ciclo.

A thread LoopSCADA espera até o semáforo de leitura e escrita estar livre e bloqueia-o. De seguida percorre a lista de todos os objectos do tipo TagSCADA e verifica se o valor guardado é igual ao valor actual lido pela thread ModbusRead. No caso de serem diferentes a thread LoopSCADA chama a função Update dessa TagSCADA. Esta função Update altera o valor guardado pela TagSCADA para o seu valor actual e percorre a lista de todos os ObjectSCADA que dependem da mesma. Por cada ObjectSCADA da lista é chamada a sua função Update aonde a TagSCADA apenas passa o valor da tag, o tipo de tag e ID correspondente. Esta função Update verifica depois, consoante os valores passados, que método deve chamar dentro da função Update de cada ObjectSCADA. Depois de percorridos todos os objectos da lista que contém os objectos TagSCADA a thread avisa que os valores já não são actuais através dum parâmetro booleano (UpdatedValues) e liberta o semáforo. Quando o parâmetro booleano informa que já existem valores actuais a thread repete o ciclo.

Existe também uma thread chamada ModbusWrite que é criada e executada sempre que um objecto ButtonSCADA é pressionado. Esta thread cria e estabelece uma conexão com o servidor e efectua um pedido modbus para escrever uma coil (função 0x05).

4.3.3 Configuração XML

Ao nível da configuração XML criei um ficheiro XML do qual o runtime lê e inicializa os objectos correspondentes. Este ficheiro contém as configurações do ip, porta e delay usados para a comunicação modbus, bem como a quantidade de entradas, saídas e registos a serem lidos pela thread ModbusRead.

Além das configurações das comunicações, o ficheiro configura ainda as tags digitais e numéricas das quais os objectos dependem e todos os parâmetros gráficos dos objectos.

Através do ficheiro XML é possível configurar as imagens para os diferentes estados dos objectos, as cores de fundo e da linha exterior dos objectos forma, a cor do texto dos objectos texto, a cor de fundo do sistema, bem como a posição, tamanho e posição no eixo z de todos os objectos.

4.3.4 Início padrão

Ao nível do início padrão do sistema o sistema cria um directório de pastas público, exemplificado na Figura 18, onde serão colocados o ficheiro de configuração XML e as imagens usadas pelo programa.

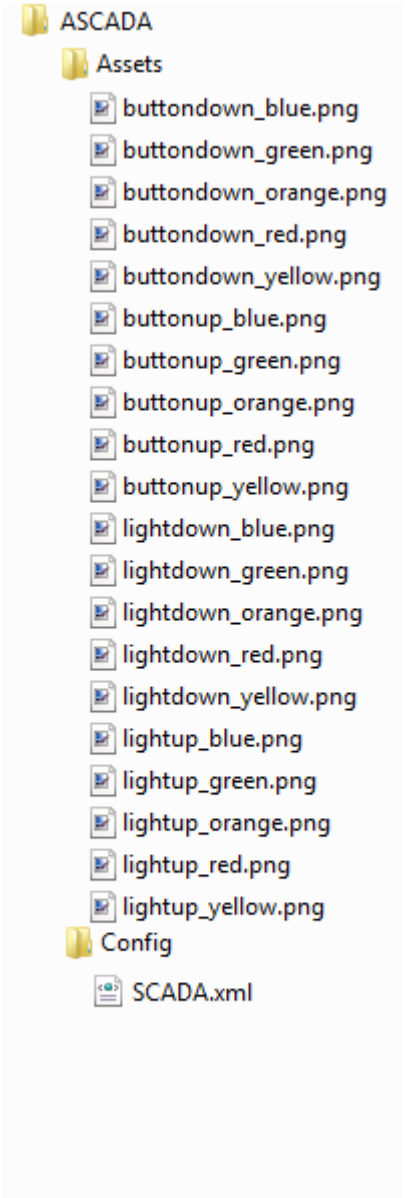


Figura 18 – Directório de pastas

O ficheiro de configuração padrão contém exemplos de como definir cada objecto, bem como uma explicação para cada um dos exemplos, como se pode observar pela Figura 19.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SCADA>
    <!-- Configure Modbus Comunication Here -->
    <!-- Change IP, delay, and number of inputs/coils/register to read -->
    <SCADAModbus>
        <ip>10.0.2.2</ip>
        <port>502</port>
        <delay>1000</delay>
        <inputstart>0</inputstart>
        <inputnumber>1</inputnumber>
        <coilstart>0</coilstart>
        <coilnumber>1</coilnumber>
        <registerstart>0</registerstart>
        <registernumber>1</registernumber>
    </SCADAModbus>

    <!-- Configure Background Color Here -->
    <!-- Change Color for color HEX value -->
    <SCADABackground>
        <Color></Color>
    </SCADABackground>

    <!-- Configure Tags SCADA Here -->
    <!-- Change DeviceID, ID of input/coil/register (name and description optional) -->
    <SCADADigitalTag>
        <Device>0</Device> <ID>0</ID> <Tag>TagDigName</Tag> <Description>TagDigDescription</Description>
    </SCADADigitalTag>
    <SCADANumericTag>
        <Device>0</Device> <ID>0</ID> <Tag>TagDigName</Tag> <Description>TagDigDescription</Description>
    </SCADANumericTag>
    <!-- Configure Button SCADA Here -->
    <!-- Change TagID for ID of coil to read/write, change imageup and imagedown for images with the state of the button  and value for
    size, position and depth -->
    <SCADAButton>
        <TagID>0</TagID> <imageup>buttonup.png</imageup> <imagedown>buttondown.png</imagedown> <x>0</x> <y>0</y> <size_x>50</size_x>
        <size_y>50</size_y> <depth>0</depth>
    </SCADAButton>

    <!-- Configure Light SCADA Here -->
    <!-- Change TagID for ID of coil to read, change imageup and imagedown for images with the state of the light and value for size,
    position and depth -->
    <SCADALight>
        <TagID>0</TagID> <imageup>lightup.png</imageup> <imagedown>lightdown.png</imagedown> <x>50</x> <y>0</y> <size_x>50</size_x> <size_y>
        50</size_y> <depth>0</depth>
    </SCADALight>

    <!-- Configure Image SCADA Here -->
    <!-- Change TagVisibilityID for ID of coil to read, change TagXID and TagYID for ID of register to read for dynamic movement, change
    Image for the name of the image you want to display and value for size, position and depth -->
    <SCADAImage>
        <TagID></TagID> <TagVisibilityID></TagVisibilityID> <TagXID></TagXID> <TagYID></TagYID> <Image></Image> <x>0</x> <y>100</y> <size_x>
        50</size_x> <size_y>50</size_y> <depth>0</depth>
    </SCADAImage>

    <!-- Configure Shape SCADA Here -->
    <!-- Change TagID for ID of register to read value of gradiente, TagVisibilityID for ID of coil to read, change TagXID and TagYID for
    ID of register to read for dynamic movement, change StartColor and EndColor with the HEX value of gradient colors, change StrokeColor
    with the HEX value of stroke color, MaxValue and MinValue to calculate the percentage and value for size, position and depth -->
    <SCADAShape>
        <TagID></TagID> <TagVisibilityID></TagVisibilityID> <TagX></TagX> <TagY></TagY> <StartColor></StartColor> <EndColor></EndColor>
        <StrokeColor></StrokeColor> <StrokeWidth></StrokeWidth> <x>150</x> <y>0</y> <size_x>50</size_x> <size_y>50</size_y> <depth>0</depth>
         <MinValue>0</MinValue> <MaxValue>0</MaxValue>
    </SCADAShape>

    <!-- Configure Text SCADA Here -->
    <!-- Change TagID for ID of register to read value, TagVisibilityID for ID of coil to read, change TagXID and TagYID for ID of register
    to read for dynamic movement, change Text for the default text you want to display and value for size, position and depth -->
    <SCADAText>
        <TagID></TagID> <TagVisibility></TagVisibility> <TagX></TagX> <TagY></TagY> <TextDefault></TextDefault> <TextColor></TextColor> <x>0
        </x> <y>100</y> <size_x>50</size_x> <size_y>50</size_y> <depth>0</depth>
    </SCADAText>
</SCADA>
```

Figura 19 - Ficheiro de configuração padrão

Ao nível das imagens usadas pelo programa, criei um conjunto de botões e luzes padrão demonstrados na Figura 20.

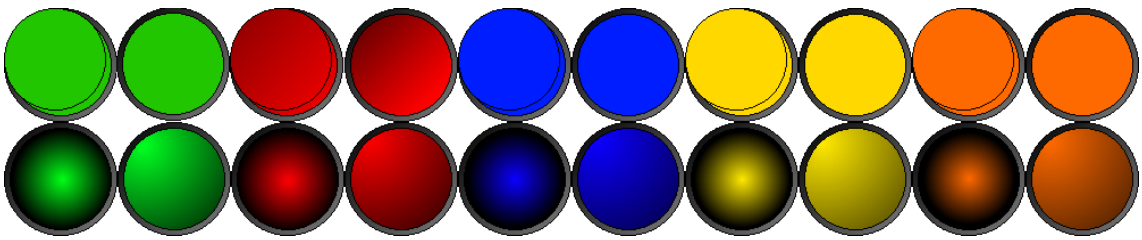


Figura 20 - Botões e luzes padrão presentes no sistema de origem

4.3.5 Teste do sistema

Ao nível do teste do sistema criei um ficheiro XML que inicializa:

- Formas rectangulares para todos os tapetes;
- Imagens para cada estado possível dos tapetes (Frente/Trás/Roda para a direita/Roda para a esquerda);
- Imagens para cada estado possível das máquinas(Trabalhar/Desligada/Mover para cima/Mover para baixo/Mover para a direita/Mover para a esquerda);
- Formas para cada posição possível das peças;
- Botões para paragem de emergência e inicio do sistema;
- Luzes para indicar o estado do sistema (A funcionar/Parado);
- Formas e texto para cada uma das quatro peças que podem ser produzidas, para indicar os números de peças produzidas;

Criei também um programa do Unity Pro XL, usando Sequencial Function Charts (SFC) e Structured Text (ST), que controla a célula de produção C1. Este programa permite assim a produção de peças do tipo P3 e P4 e o seu armazenamento nos tapetes PM1 e PM2.

Para criar este programa de teste foi necessário usar mais de quatrocentas tags digitais e quatro registos. Os objectos iniciados pelo runtime, usam os valores destas tags para alterar os seus estados e a sua visibilidade, como se pode observar pelas Figuras 20 e 21.

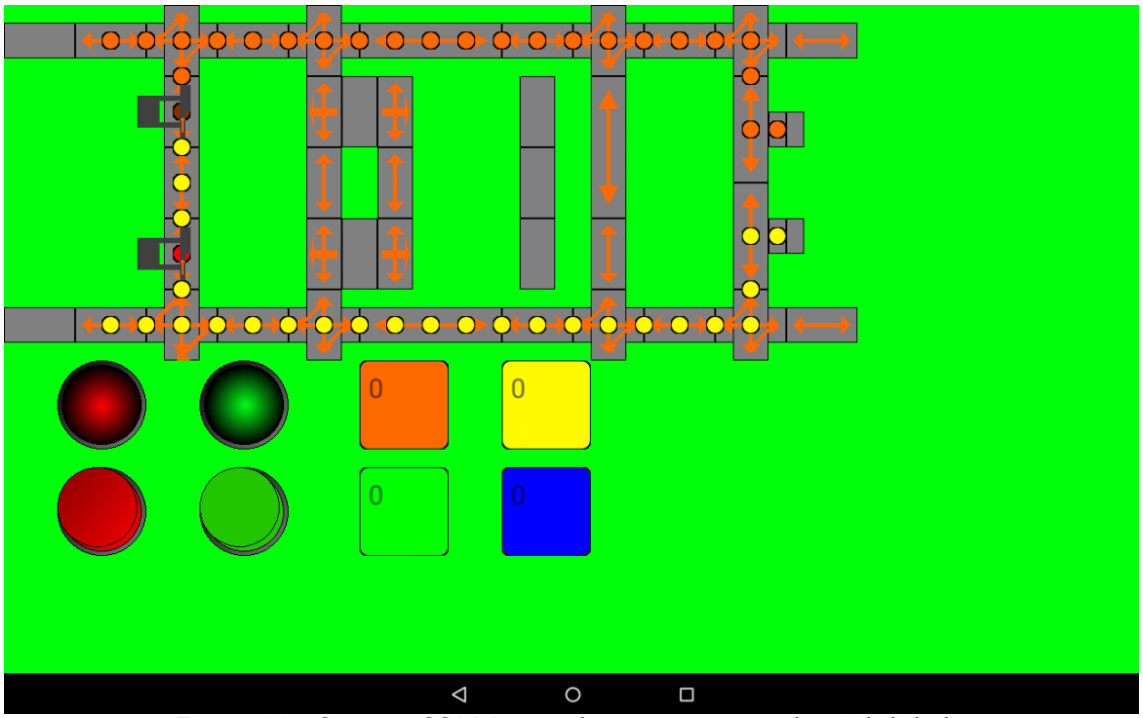


Figura 20 - Sistema SCADA completo sem as tags de visibilidade

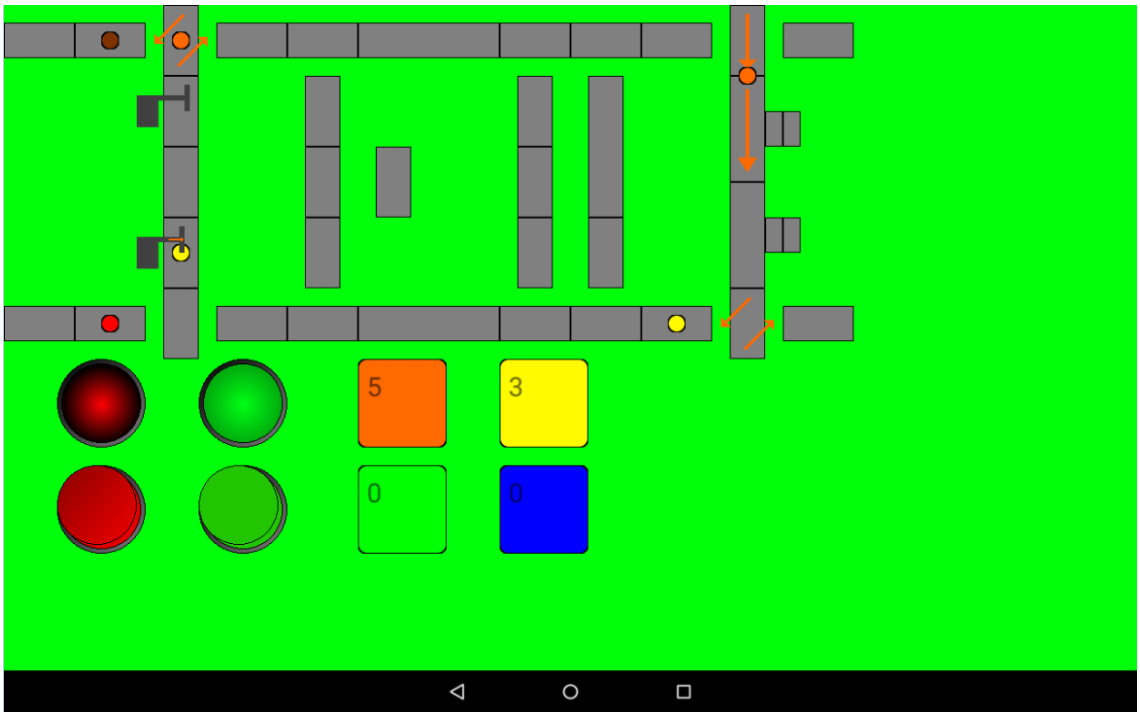


Figura 21 - Sistema SCADA completo com as tags de visibilidade e a funcionar

4.4 Testes de performance e robustez

4.4.1 Performance

Para testar a performance do sistema usei o dispositivo físico E-star X35 e a app Cool Tool - system stats para registar o valor de processador e RAM livre com e sem o uso do sistema SCADA e obtive os seguintes resultados:

	S/ SCADA			C/ SCADA		
	Min	Média	Max	Min	Média	Max
CPU (%)	0	2	6	0	4	8
RAM Livre (MB)	106	109	117	81	82	85

Tabela 2 - Performance do dispositivo com e sem o sistema SCADA

Medi também o tempo que demora o programa a executar o código com e sem o tempo da comunicação modbus (120 entradas/320 saídas/4 registos) em milissegundos e obtive os seguintes resultados:

	S/ modbus			C/ modbus		
	Min	Média	Max	Min	Média	Max
Tempo de execução (ms)	0	0	1	63	70	103

Tabela 3 - Tempo de execução do sistema com e sem modbus

4.4.2 Robustez ("stress test")

Para testar a robustez do sistema usei o mesmo dispositivo e a mesma app dos testes anteriores, mas desta vez usei um ficheiro de configuração com três mil objectos LightSCADA dependentes todos da mesma tag. Depois de iniciado o sistema, medi a performance com o uso do sistema SCADA e depois medi a performance aquando da alteração do valor lógico da tag de qual dependem os três mil objectos e obtive os seguintes resultados:

	C/ SCADA e S/tag			C/ SCADA e C/tag		
	Min	Média	Max	Min	Média	Max
CPU (%)	0	3	8	26	35	40
RAM Livre (MB)	77	78	80	51	53	60

Tabela 4 - Performance do dispositivo com sistema SCADA e com ou sem alteração do valor da tag

Medi também para este teste o tempo que demora o programa a executar o código com e sem o tempo da comunicação modbus (1 entrada) aquando da alteração do valor da tag e obtive os seguintes resultados:

	S/ modbus			C/ modbus		
	Min	Média	Max	Min	Média	Max
Tempo de execução (ms)	0	0	2	12	16	23

Tabela 5 - Tempo de execução do sistema com e sem modbus aquando da alteração do valor da tag

Capítulo 5

5 Conclusão e trabalho futuro

Neste capítulo apresento as conclusão retiradas deste projecto e apresento qual o trabalho que pode ser realizado futuramente para melhorar e completar a minha solução.

5.1 Conclusão

Depois de terminado o sistema consegui tirar algumas conclusões do resultado.

A primeira conclusão foi que como tinha o trabalho bem estruturado e planificado consegui acabar o sistema no tempo previsto e atingir todos os requisitos impostos pelo orientador.

Outra conclusão a que cheguei olhando para a Tabela 1 foi que o sistema se comporta como esperado e apresenta uma boa performance quando comparado ao telemóvel sem nenhuma aplicação aberta, mesmo quando usado num dispositivo de gama baixa. Observando a Tabela 3 concluí também que mesmo sobrecarregando o sistema com uma enorme quantidade de objectos ele mantém uma boa performance, diminuindo apenas quando os três mil objectos são actualizados ao mesmo tempo. No entanto, observando a Tabela 2 e a Tabela 4 na coluna sem as comunicações modbus, verificamos que apesar do uso do CPU ter aumentado muito aquando da alteração do valor da tag o tempo de execução do código se manteve idêntico. Observando as mesmas tabelas, mas

agora comparando os valores com e sem o uso do modbus verificamos que o onde o programa demora mais a executar é aquando das comunicações modbus. Usando o Wireshark verifiquei que os tempos que o código demoraria a correr aquando das comunicações modbus seriam o tempo que o servidor demoraria a responder, ou seja o tempo que demoraria entre o computador receber o pedido e enviar a resposta. Isto permitiu-me concluir que o tempo de execução elevado se devia a processamento no computador e não o tempo de transmissão do pedido ou recepção da resposta.

A última conclusão a que cheguei foi que apesar da programação em android ter algumas limitações e alguns truques e não ter qualquer experiência passada a programar em java ou em android consegui resolver os problemas à medida que foram aparecendo e concluir o programa apenas com o meu gosto por programar, uma capacidade critica e alguma ajuda do meu orientador.

5.2 Trabalho futuro

Apesar do trabalho se encontrar completo e com uma óptima performance é sempre possível completar mais o trabalho.

Para mim o que falta para este ser um trabalho completamente finalizado é a adição de um sistema de configuração, que permita configurar o sistema sem ter de escrever à mão num ficheiro XML.

Por outro lado apesar de não ser algo que seja preciso adicionar para o trabalho estar finalizado seria de valorizar a adição de outros objectos que existem noutros sistemas SCADA como gráficos de barras ou de linhas.

Referências

[1] http://en.wikipedia.org/wiki/Android_%28operating_system%29 - Acesso em 16/2/2015
[2] https://dspace.mah.se/bitstream/handle/2043/10721/AndroidApplicationDevelopment.pdf?sequence=1- Acesso em 16/2/2015
[3] http://developer.android.com/sdk/index.html - Acesso em 16/2/2015
[4] http://repositorio-aberto.up.pt/bitstream/10216/59515/1/000135118.pdf - Acesso 16/2/2015
[5] http://www.automation.com/automation-news/article/profibus-and-modbus-a-comparison - 16/2/2015
[6] https://scadahacker.com/library/Documents/ICS_Basics/SCADA%20Basics%20-%20NCS%20TIB%2004-1.pdf – Acesso em 10/2/2015
[7] http://www.scada.com/Software - Acesso em 10/2/2015
[8] https://www.inductiveautomation.com/what-is-scada - Acesso em 10/2/2015
[9] http://electrical-engineering-portal.com/an-introduction-to-scada-for-electrical-engineers-beginners - Acesso em 10/2/2015
[10]WHAT_IS_SCADA.pdf – Conteúdos 2014/2015, unidade curricular Automação - Acesso em 10/2/2015
[11]Operator_Interface.pdf – Conteúdos 2014/2015, unidade curricular Automação - Acesso em 10/2/2015
[12]http://www.scadalink.com/support/knowledge-base/an-introduction-to-scada/
[13]http://www.engineersgarage.com/articles/scada-systems - Acesso em 10/2/2015
[14]http://www.science.smith.edu/~jcardell/Readings/TRUST%20US/2005_09_15_Jeff_Dagle.pdf - Acesso em 10/2/2015
[15]http://futuronix.in/download/scada.pdf - Acesso em 10/2/2015
[16]http://scadahistory.com/~ - Acesso em 10/2/2015
[17]http://ourinstrumentationgroup.com/SCADA_Primer.pdf - Acesso em 10/2/2015
[18]http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/Introduction_to_SCADA_Security_for_Managers_and_Operators.pdf - Acesso em 10/2/2015