



INSA de Rennes
Département INFORMATIQUE
Rapport de stage

Maître de stage : Marie BABEL

Développement d'une application de traitement d'image

Sacha PERCOT-TÉTU

Diffusion du rapport (y compris en version électronique) :

- ☐ Non autorisée (le rapport ne sera pas diffusé, mais archivé par obligation légale)
- ☐ Autorisée en interne à l'INSA
- ☒ Autorisée en interne et en externe

Rennes, le 10 octobre 2012

Table des matières

Introduction	3
1 Spécifications de l'application	4
1.1 Un logiciel image	4
1.1.1 Différents types d'images	4
1.1.2 Manipulation des images	4
1.1.3 Informations sur les images	5
1.2 Des algorithmes de traitement	5
1.2.1 Architecture orientée algorithme	5
1.2.2 Une bibliothèque d'algorithmes	5
1.3 Des modules d'extension	6
1.3.1 Utilisation des modules d'extension	6
1.3.2 Développement des modules d'extension	6
2 Analyse de l'existant	7
2.1 Un logiciel : <i>Image EII</i>	7
2.1.1 Une version initiale <i>Visual C++ 6</i>	7
2.1.2 Une première version <i>Qt</i>	7
2.1.3 Une seconde version <i>Qt</i>	7
2.2 Le projet <i>Detiq-T</i>	8
2.2.1 Une bibliothèque : <i>ImageIn</i>	8
2.2.2 Une interface générique	8
3 Application finale	9
3.1 Architecture globale	9
3.2 Intégration du projet <i>Detiq-T</i>	10
3.2.1 La bibliothèque <i>ImageIn</i>	10
3.2.2 L'interface générique	10
3.2.3 Vision globale des modifications	10
3.3 Le logiciel <i>eiimage</i>	10
3.3.1 Le coeur de l'application : <i>eiimage core</i>	10
3.3.2 L'application <i>eiimage</i>	11
3.3.3 Les modules d'extension	11
Conclusion	13
Annexe A : Planning du stage	15
Annexe B : Informations technique	16

Introduction

Remerciements Je tiens tout d'abord à remercier le département EII de l'INSA de Rennes pour m'avoir offert l'opportunité d'effectuer ce stage. Je tiens à remercier Marie BABEL, mon maître de stage, enseignant-chercheur au département EII, pour sa confiance, son attention et l'aide qu'elle m'a apporté tout au long de mon stage. Je tiens également à remercier Muriel PRESSIGOUT et Luce MORIN, enseignants-chercheurs au départements EII, pour le temps qu'elles ont pu m'accorder lors de ce stage. Enfin, je tiens à remercier le département Informatique de l'INSA pour m'avoir accueillis dans leurs locaux pour la durée de ce stage.

Ce rapport décrit succinctement le travail réalisé lors de mon stage d'été de second cycle.

Ce stage s'est déroulé au département EII¹ de l'INSA de Rennes du 01/06/2012 au 31/07/2012 puis du 27/08/2012 au 27/09/2012.

L'objectif de ce stage est le portage d'une ancienne application de traitement d'image développée sous une ancienne technologie Visual C++ vers les technologies Qt². L'objectif principal est de développer une application maintenable et multiplateforme comportant l'ensemble des fonctionnalités de l'application existante.

Nous commencerons donc pas décrire les spécifications de l'application finale, qui sont pour la plupart commune à l'application d'origine. Nous détaillerons par la suite l'ensemble des éléments existants liées à l'application. Enfin, nous décriront l'architecture globale de l'application finale ainsi que les bases qui ont servi au développement de cette application.

1. Electronique et Informatique Industriel

2. Qt est un framework C++ open source offrant des fonctionnalités d'interface graphique

1. Spécifications de l'application

1.1 Un logiciel image

L'application en tant que logiciel de traitement d'image doit offrir un certain nombre de fonctionnalités relatives à la manipulation d'images matricielles mais également d'autres fonctionnalités spécifiques au traitement et à l'analyse d'image.

1.1.1 Différents types d'images

Une image matricielle I de largeur w , de hauteur h et contenant n composantes peut être définie sous la forme d'une matrice de n -uplet :

$$I \in (M_{h,w}(K))^c \text{ i.e. } I = (c_1, c_2, \dots, c_n), \quad c_i = (p_{i,j})_{1 \leq i \leq w, 1 \leq j \leq h}, \quad p_{i,j} \in K$$

Dans le cas d'une image standard, $n \in \llbracket 1, 4 \rrbracket$. Ce qui signifie qu'une image peut comporter de une composante (noir) à quatre composantes (rouge, vert, bleu, transparence). La valeurs des pixels de chaque composante varie quand à elle entre 0 et 255 ($K = \llbracket 0, 256 \rrbracket$).

Le traitement d'image peut amener à manipuler d'autres type d'image :

- Des images de profondeur $n > 8 \text{ bits}$, auquel cas $K = \llbracket 0, 2^n \rrbracket$.
- Des images d'entiers relatifs, i.e. $K \subset \mathbb{Z}$.
- Des images de réels, i.e. $K \subset \mathbb{R}$.

L'application doit donc être capable de manipuler tous ces types d'images. De plus, les images autres que celles du cas général ne sont pas affichables, l'application doit être en mesure d'effectuer un certain nombre de transformations sur ces images afin d'en offrir à l'utilisateur une représentation affichable. Ces transformations ne doivent être utilisées que pour l'affichage final de l'image, et ne doivent donc en aucun cas altérer les données réelles associées à l'image.

L'application doit permettre à l'utilisateur de visualiser la valeur numérique de chaque pixel de l'image. Lors de l'affichage d'une image, l'utilisateur doit avoir la possibilité de visualiser la valeur du pixel se situant sous le curseur. L'application doit également proposer une visualisation de l'image sous forme de *grille de pixel* sur laquelle l'utilisateur pourra directement visualiser la valeur de chacun des pixels constituant l'image.

1.1.2 Manipulation des images

L'application doit offrir à l'utilisateur un certain nombre d'opérations de pré-traitement des images. Ces opérations incluent :

- Recadrage
- Ré-échantillonnage (sur-échantillonnage ou sous-échantillonnage)
- Opérations avec une valeur donnée (opérations arithmétiques, opérations booléennes)
- Opérations entre images (opérations arithmétiques, opérations booléennes)
- Transformations géométrique (translation, rotation, symétries axiales)

Chacune de ces opérations doit pouvoir être appliquée sur tous les types d'images supportés par l'application.

1.1.3 Informations sur les images

L'application doit permettre à l'utilisateur d'afficher la distribution des valeurs de l'image sous forme d'histogramme. Sur cet histogramme, l'utilisateur doit pouvoir visualiser avec précision la valeur de chaque point de l'histogramme. L'utilisateur doit également pouvoir visualiser simultanément ou séparément l'histogramme de chacune des composante de l'image.

L'application doit permettre à l'utilisateur d'obtenir un certain nombre d'informations statistiques relative à une ou plusieurs images. Ces informations incluent :

- Valeurs minimum, maximum et moyenne d'une image
- Variance et écart-type d'une image
- Entropie d'une image
- Rapport signal-bruit entre deux images
- Erreur quadratique moyenne entre deux images

1.2 Des algorithmes de traitement

1.2.1 Architecture orientée algorithme

L'objectif principal de l'application est la mise en oeuvre d'opérations de traitement d'image. L'architecture du logiciel doit donc être adaptée pour le développement et la manipulation d'algorithmes de traitement ainsi que la mise en oeuvre d'interfaces utilisateurs nécessaire avant et après l'application d'un algorithme.

L'architecture du logiciel doit permettre l'ajout et la suppression d'opérations de traitement de manière simple et efficace afin de faciliter la maintenance de l'application.

1.2.2 Une bibliothèque d'algorithmes

L'application doit mettre en oeuvre un certain nombre d'algorithmes nécessaire à l'enseignement du traitement du signal au département EII.

Ces algorithmes incluent notamment :

- Des transformées mathématiques (Fourier, cosinus, Haar, Hough, etc.)
- Des algorithmes de codage (Huffman, MICD, etc.)
- Des algorithmes de classification

- Des algorithmes de morphologie mathématique.
- Des algorithmes de filtrage linéaire.
- etc.

Certains de ces algorithmes peuvent n'être applicable que sur un sous-ensemble des types d'image manipulés par l'application.

1.3 Des modules d'extension

L'application doit mettre en oeuvre un système de module d'extension permettant d'ajouter à l'application de nouveaux algorithmes de traitement sans modification du programme.

1.3.1 Utilisation des modules d'extension

Chaque module d'extension doit pouvoir contenir un ou plusieurs algorithmes de traitements d'image.

Un module d'extension doit pouvoir être chargé dans l'application à l'exécution. L'utilisateur doit pouvoir charger plusieurs modules d'extension simultanément.

1.3.2 Développement des modules d'extension

Lors du développement d'un module d'extension, l'utilisateur-développeur doit pouvoir faire abstraction de toute notion d'interface avec l'utilisateur final afin de se focaliser sur son algorithme de traitement.

Ainsi, l'architecture du logiciel doit offrir un certain nombre d'outils permettant au développeur de mettre en oeuvre un module d'extension le plus simplement possible. Ces outils doivent inclure un ensemble de composantes logicielles permettant au développeur de communiquer avec l'utilisateur final, sans pour autant avoir à manipuler d'interface graphique.

2. Analyse de l'existant

Le temps imparti pour le projet étant relativement court, il est alors impératif de faire une analyse rapide mais approfondie des logiciels existants, afin d'en tirer un maximum de profit.

2.1 Un logiciel : *Image EII*

2.1.1 Une version initiale *Visual C++ 6*

Le logiciel Image EII est un logiciel de traitement d'image développé en interne par le département EII en 2003. Ce logiciel est fortement basé sur l'environnement de développement Microsoft Visual C++ 6 et sa bibliothèque MFC 6.0. Il est donc dès lors inenvisageable de déployer ce logiciel sur d'autres systèmes que Microsoft Windows. De plus, ces technologies datent de 1998 et Microsoft les rends obsolète dès 2002 par le lancement des technologies Visual C++ .NET. Ce logiciel devient donc rapidement difficile à maintenir, ce qui rend son déploiement de plus en plus problématique.

Dès lors, le besoin de porter ce projet vers des technologies plus récentes devient indispensable. Les technologies retenues sont les suivantes :

- Langage C++ natif, sans surcouche propriétaire pour être plus maintenable et multi-plateforme.
- Le framework Qt pour une interface simple, performante et multi-plateforme.

Ces technologies devraient permettre d'obtenir une application multiplateforme et plus facile à maintenir.

2.1.2 Une première version *Qt*

La première tentative de portage de l'application Image EII vers les technologies C++/Qt date de 2009. Un groupe d'étudiant est alors chargé d'effectuer ce portage dans le cadre des projets logiciel de 5ème année EII. La mission n'est pas chose facile car l'intégralité du code source du logiciel Image EII est basé sur les technologies Microsoft. Tout d'abord l'interface graphique se base sur la technologie MFC, le système de plugin et la lecture de fichier exploitent des outils Microsoft et chaque implémentation de chaque algorithme de traitement jusqu'à la modélisation des images matricielle : tout est basé sur les structures de données Microsoft Visual C++.

Ce projet donnera naissance à un logiciel inachevé, doté d'une partie des fonctionnalités du logiciel d'origine, mais où la plupart des algorithmes de traitements n'ont pas été implémentés.

2.1.3 Une seconde version *Qt*

La seconde tentative de portage de l'application Image EII vers les technologies C++/Qt date de 2011. C'est dans le cadre d'un stage de fin d'étude qu'un

étudiant du département EII est chargé d'achever le portage de cette application, initié deux ans plus tôt.

L'architecture très bancale (voire inexistante sur certain points) du premier projet est relativement améliorée, un certain nombre d'algorithmes jusqu'alors absents sont implémenté et une grande partie des bugs présent est alors corrigée.

Le résultat final est un logiciel fonctionnel mais amputé d'une partie des fonctionnalités et des algorithmes du logiciel d'origine. De plus, l'architecture de ce logiciel comporte beaucoup de faiblesses qui rende l'exploitation de ce projet problématique.

2.2 Le projet *Detiq-T*

Le projet Detiq-T¹ est un projet de 4ème année Informatique dans le cadre duquel six étudiants sont chargés de développer un environnement de développement permettant de faciliter la mise en oeuvre d'application de traitement d'image en C++. Cet environnement de développement a été utilisé dans le cadre du projet pour le portage d'ancienne application de traitement d'image basées sur des technologies Microsoft Visual C++ 6. Ce projet se décompose en deux partie :

- Une bibliothèque de traitement d'image : **ImageIn**
- Une interface générique pour des application de traitement

2.2.1 Une bibliothèque : *ImageIn*

La bibliothèque ImageIn rassemble toutes les fonctionnalités nécessaire pour la manipulation de tout type d'image matricielle. Elle a été conçu de manière générique et extensible afin de permettre tout type d'usage et d'offrir la possibilité d'enrichir facilement la bibliothèque de nouvelles fonctionnalités. Elle permet la lecture et l'enregistrement dans les formats BMP, JPEG et PNG mais son architecture permet l'ajout de nouveaux formats de manière très simple. Elle offre également un certain nombre de fonctionnalités de traitement d'image (calcul d'histogramme, filtrage, morphologie mathématique, etc.)

2.2.2 Une interface générique

L'interface générique quand à elle consiste à rassembler dans une bibliothèque toute les fonctionnalités d'une interface graphique destinée au traitement d'image. Elle offre des fenêtres d'affichage d'image, d'histogramme, de grille de pixel. Ainsi qu'un certain nombre de fonctionnalités ergonomiques et une architecture complète pour le développement d'application. Pour développer une application il suffit alors simplement d'hériter de cette interface générique et d'y ajouter les fonctionnalités voulues.

1. Développement d'un Environnement de Traitement d'Image sous Qt

3. Application finale

3.1 Architecture globale

Les spécification de l'application font ressortir quatre composantes clés :

- Un modèle de représentation d'image
- Des algorithmes de traitements
- Une interface graphique
- Des modules d'extension

L'analyse de l'existant nous permet de conclure qu'aucune des versions du logiciel Image EII ne comporte de réel modèle pour la représentation d'image et d'algorithme. En revanche, la bibliothèque *ImageIn* du projet *Detiq-T*, de par sa conception très générique, semble pouvoir répondre de manière simple et structurée aux exigences de l'application finale. Cette application se basera donc sur cette bibliothèque pour toutes les représentations et manipulations d'images matricielles.

L'ensemble des algorithmes de traitement mentionnés dans les spécifications de l'application sont déjà implémentés dans la version d'origine du logiciel Image EII. Ces algorithmes étant pour la plupart pleinement fonctionnels, le portage de l'ensemble de ces algorithmes vers la nouvelle application peut donc s'effectuer pour le coût d'une simple réécriture de l'implémentation sans considération nécessaire pour l'algorithme sous-jacente. La bibliothèque *ImageIn* offre également quelques fonctionnalités de traitement d'image qui pourront être utilisées par l'application.

Le développement d'une interface graphique étant très coûteux, il est indispensable de réutiliser une des interfaces existantes. Deux interfaces sont potentiellement réutilisables : l'interface du portage le plus récent du logiciel Image EII et l'interface générique du projet *detiq-t*. L'interface du logiciel Image EII offre toutes les fonctionnalités nécessaire à l'application, mais elle n'est cependant pas réutilisable directement. En effet, cette interface étant fortement basée sur une représentation des images qui ne sera pas réutilisée par l'application, son utilisation nécessiterait donc de l'adapter pour fonctionner avec la bibliothèque *ImageIn*. L'interface générique du projet *detiq-t* est déjà basée sur la bibliothèque *ImageIn* mais elle n'offre pas toutes les fonctionnalités nécessaire à l'application. Elle n'est notamment pas conçue pour exploiter d'autres type d'images que des images affichables *standards*. La réutilisation de cette interface générique serait donc la solution la plus coûteuse en terme de temps de développement. Cependant, l'architecture de l'interface générique étant plus robuste que celle du logiciel Image EII, elle permettrait de construire un logiciel plus stable et dont la maintenance serait beaucoup plus aisée. Le choix a donc été

de baser l'application sur cette interface générique, malgré les coûts nécessaires pour y intégrer les fonctionnalités manquantes.

Le principe de module d'extension est une fonctionnalité qui est également intégré dans le logiciel Image EII d'origine. Son implémentation, basée sur la plateforme Visual C++, n'est pas exploitable. Son architecture en revanche, bien que relativement limitée, permet de poser une base de réflexion pour la conception de ce point clé des spécifications du logiciel.

L'architecture globale de l'application finale est représentée sur la figure 3.1.

3.2 Intégration du projet *Detiq-T*

3.2.1 La bibliothèque *ImageIn*

La conception de la bibliothèque ImageIn étant très générique, aucun modification n'étaient nécessaire pour son intégrations dans l'application finale. Quelques fonctionnalités ont cependant été ajouté afin de faciliter son utilisation. Certains algorithmes ont également été optimisées et quelques bugs ont été corrigés.

3.2.2 L'interface générique

De nombreuses fonctionnalités ont été ajouté à l'interface générique afin de répondre aux spécifications de l'application. L'architecture extensible de cette interface générique a grandement facilité l'intégration de ces nouvelles fonctionnalités. De nombreuses améliorations et optimisations des fonctionnalités pré-existantes a été cependant nécessaire afin d'obtenir une interface robuste et performante.

3.2.3 Vision globale des modifications

La figure 3.2 permet de visualiser graphiquement les modifications apportées au projet detiq-t. Les modifications précédant la barre verticale correspondent à toutes les modifications effectuées depuis la création du projet. Les modifications suivant la barre verticale correspondent à toutes les modifications effectuées dans le cadre de ce stage. Le projet detiq-t n'a donc pas seulement été réutilisé, mais bel et bien remanié et finalisé avant de pouvoir être intégré dans ce projet.

3.3 Le logiciel *eiimage*

3.3.1 Le coeur de l'application : *eiimage core*

Le coeur de l'application est une bibliothèque dynamique nommé *eiimagecore*. Cette bibliothèque peut être vue comme un modèle exploité à la fois par l'application finale et par ses modules d'extensions. Cette architecture garantie qu'il n'y ait aucune dépendance entre les modules d'extension et l'application finale. Cette absence de dépendance facilite le développement et le déploiement

de ces modules tout en offrant une grande souplesse dans la maintenance de l'application.

3.3.2 L'application *eiimage*

L'application *eiimage* correspond à l'exécutable final.

L'interface de cette application est une spécialisation de l'interface générique, elle y intègre donc toutes les fonctionnalités tout en proposant une organisation propre au logiciel *eiimage*.

L'ensemble des algorithmes décrits dans les spécifications ont été intégrés dans cette application en se basant sur les implémentations présentes dans le logiciel Image EII. Si certains algorithmes ont pu facilement être adaptés pour utiliser la bibliothèque ImageIn et ainsi être intégrés dans l'application, une grande partie d'entre eux ont cependant dû être au moins partiellement réécrits.

Cette application met également en oeuvre la gestion des modules d'extension, du chargement d'un module à l'organisation et l'exploitation des algorithmes qu'il contient.

3.3.3 Les modules d'extension

Les modules ne sont que des implémentations d'un modèle défini dans *eiimagecore*. Il peuvent ainsi être développés et déployés sans connaissance de l'application dans laquelle ils seront intégrés.

FIGURE 3.1 – Architecture globale de l’application finale.

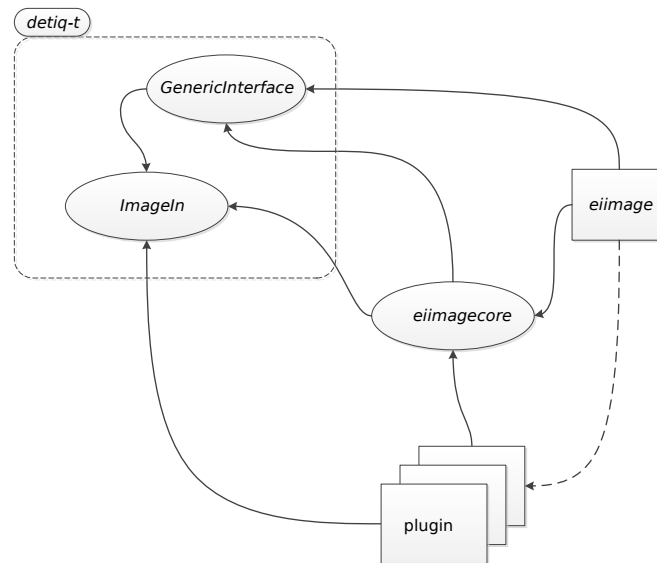
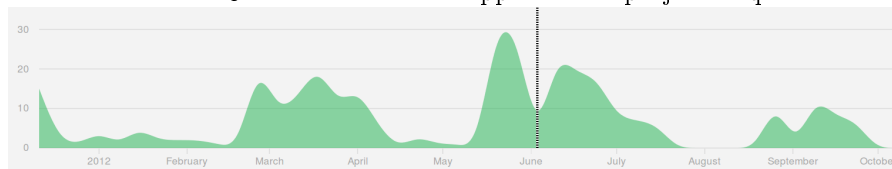


FIGURE 3.2 – Modifications apportées au projet detiq-t



Conclusion

L'application finale eiimage réponds à toutes les spécifications logicielles décrites dans la première partie de ce rapport. Elle comporte l'ensemble des fonctionnalités du logiciel d'origine Image EII tout en y ajoutant un certain nombre de fonctionnalités supplémentaires.

Une version de développement du logiciel eiimage a déjà été utilisée en cours de stage dans le cadre des modules de sensibilisation du département EII, la version finale devrait être utilisée dès cette année pour les travaux pratiques de traitement d'image.

Le développement de cette application m'a permis d'approfondir de manière significative mes connaissances en traitement d'image tout en me confrontant aux différentes problématiques liées à la reprise d'application. J'ai également pu me familiariser encore un peu plus avec les technologies Qt et aiguïser mes connaissances du langage C++.

Malgré l'absence de composante industrielle, la grande diversité thématique de ce stage m'aura permis d'approcher des domaines de connaissance que je n'avais pour l'instant pas eu l'occasion d'approcher lors de ma formation informatique à l'INSA.

Annexe A : Planning du stage

Juin 2012							
	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
22					1	2	3
23	4	5	6	7	8	9	10
24	11	12	13	14	15	16	17
25	18	19	20	21	22	23	24
26	25	26	27	28	29	30	
} Étude de l'existant et analyse fonctionnelle } Conception de l'application } Développement ¹ de l'application							
Juillet 2012							
	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
26							1
27	2	3	4	5	6	7	8
28	9	10	11	12	13	14	15
29	16	17	18	19	20	21	22
30	23	24	25	26	27	28	29
31	30	31					
} Développement ¹ de l'application } Développement des algorithmes							
Août 2012							
	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
31			1	2	3	4	5
32	6	7	8	9	10	11	12
33	13	14	15	16	17	18	19
34	20	21	22	23	24	25	26
35	27	28	29	30	31		
} Développement des algorithmes							
Septembre 2012							
	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
35						1	2
36	3	4	5	6	7	8	9
37	10	11	12	13	14	15	16
38	17	18	19	20	21	22	23
39	24	25	26	27	28	29	30
} Développement des algorithmes } Finalisation et déploiement							

1. Développement = Code + Tests + Documentation

Annexe B : Documentation technique

L'ensemble du projet est disponible sur les dépôts publics à l'adresse <http://github.com/eiimage>. Un projet Qt prêt à l'emploi est également disponible à l'adresse http://github.com/downloads/eiimage/eiimage/eiimage_project.zip.

La compilation du projet nécessite GCC 4.4 ou supérieur et Qt 4.8 ou supérieur. Le déploiement a été testé avec succès sur les systèmes Windows XP, Windows 7, Windows 8, Ubuntu Linux 12.04 et Linux Fedora 16. Le déploiement de l'application sous plateforme Mac est théoriquement possible, mais n'a cependant pas été testé.

L'ensemble du projet est documenté selon la méthode *Doxygen*². Une documentation peut donc être générée pour chaque module à l'aide de l'utilitaire *Doxygen*.

2. <http://www.doxygen.org/>

Ce stage d'été de 4ème année a été effectué au département EII de l'INSA de Rennes du 1er juin 2012 au 27 septembre 2012. Le sujet de ce stage était le portage d'un logiciel de traitement d'image de technologies Microsoft Visual C++ vers Qt.

This 4th year summer internship took place at the EII³ department of the INSA-Rennes from June 1st 2012 to September 27 2012. The intership's subject was to port an image processing application from Microsoft Visual C++ technology to Qt.

3. Electronic and Computer Engineering