Facultad de Ingeniería Universidad de Buenos Aires

75.99 - Trabajo Profesional

MLC version 3 (Machine Learning Control)

Tutor: Adriana Echeverría
Co-Tutor: Julia Garibaldi

Integrantes:

Padrón	Nombre	Email
89579	Torres Feyuk, Nicolás R. Ezequiel	ezequiel.torresfeyuk@gmail.com
86882	Germano Zbrun, Marco	marco.germano@intraway.com
88430	Lopez Skuba, Raúl	raullopez0@gmail.com

Índice

1. Introducción Teórica

El presente trabajo se propone el análisis, diseño e implementación de un conjunto de mejoras requeridas por Thomas Duriéz, investigador del laboratorio de fluodinámica de la facultad de ingeniería, a aplicarse sobre el sistema MLC, desarrollado por él, y utilizado como herramienta en su trabajo de investigación. El mismo es un sistema que permite encontrar leyes de control de sistemas complejos de forma automática, dentro del cual es de especial interés la estabilización de flujos turbulentos. El siguiente capítulo fue realizado basado en la información obtenida en [1].

1.1. Control a Lazo Cerrado optimizado a través de Función de Costo

La teoría de control clásica categoriza los sistemas de control en **Open-Loop Control Systems** y **Close-Loop Control Systems** [2]. Se definen los mismos a continuación:

- **Open-Loop Control Systems:** Son aquellos sistemas en donde la salida del sistema no es comparada contra una referencia de entrada. Estos sistemas funcionan a base de calibración. Ejemplos de los mismos son pavas eléctricas, balanzas, etc.. Un esquema de este modelo es mostrado en la figura 1.
- **Closed-Loop Control Systems:** También llamados sistemas retroalimentados (Feedback Control Systems). Esta clase de sistema compara la salida retroalimentada contra una referencia de entrada, obteniendo así una función de error. La señal retroalimentada (feedback signal) puede ser la salida del sistema o bien una función derivada de la misma. La función de error es utilizada para especificar la actuación a ingresar en el sistema físico. Ejemplos de los mismos son el sistema de frenos ABS o la termorregulación corporal, entre otros. Un esquema de este modelo es mostrado en la figura 2.

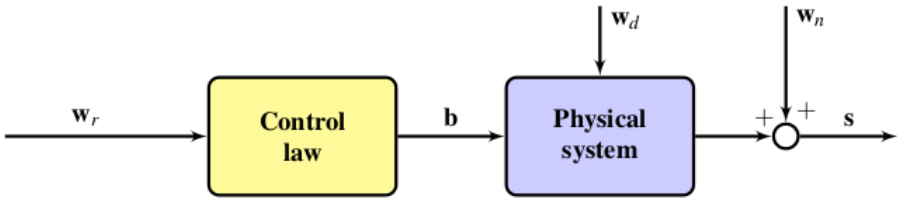

Fig. 1 Open-Loop Control System, siendo $\mathbf{w_r}$ la señal de referencia, **b** la señal de actuación, $\mathbf{w_d}$ perturbaciones externas, $\mathbf{w_n}$ ruido introducido por los sensores involucrados y **s** la señal de salida. Imagen tomada de [1]

Si bien los sistemas de control a lazo cerrado son complejos de diseñar y mantener, son los únicos que permiten estabilizar aquellos sistemas físicos que se ven afectados por estímulos externos. Esto último no es posible de realizar en un sistema de lazo abierto. Es por esta razón que este tipo de modelo es

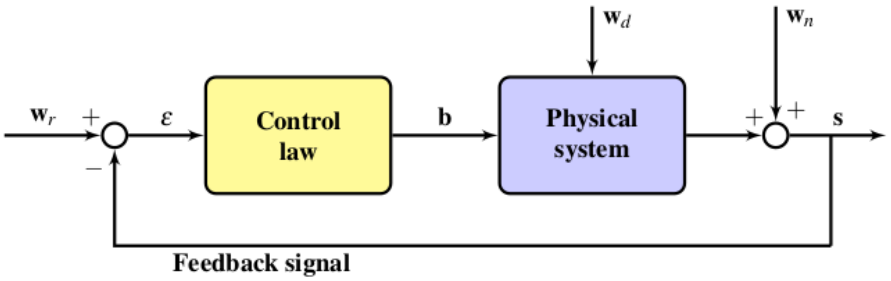

Fig. 2 Closed-Loop Control System, siendo $\mathbf{w_r}$ la señal de referencia, **b** la señal de actuación, $\boldsymbol{\varepsilon}$ la función de error, $\mathbf{w_r}$ perturbaciones externas, $\mathbf{w_n}$ ruido introducido por los sensores involucrados y **s** la señal de salida. Imagen tomada de [1]

el mayormente usado para resolver problemas de naturaleza compleja (no lineales, caóticos, etc.) con un alto grado de interacción con variables externas.

Diseñar la ley de control que logre estabilizar el sistema deseado es un reto. Esto se debe a que, además de tener una comprensión matemática del sistema físico, se deben tener en cuenta todas las variables externas que modifican al mismo. Por esta razón, es importante tener una medida de la calidad de la ley de control propuesta. En la figura se muestra un sistema a lazo cerrado con la adición de una función de costo **J**. Esta función de costo es la clave para lograr definir luego un método efectivo para encontrar leyes de control a través del MLC.

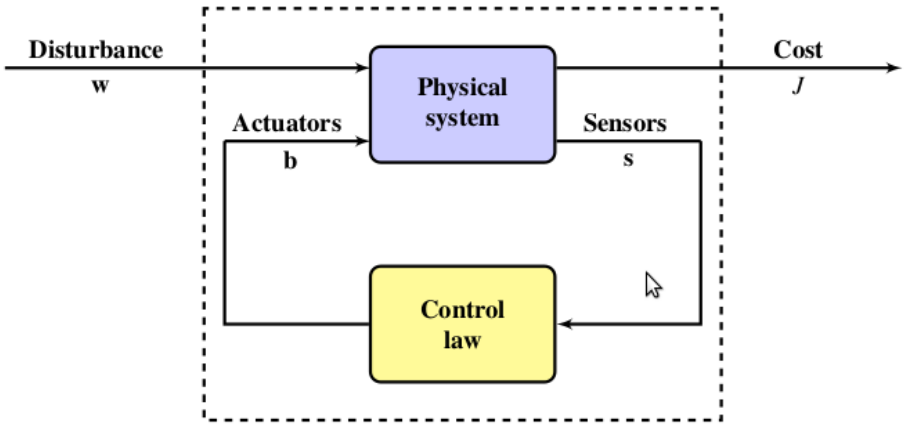

Fig. 3 Sistema de Control a Lazo Cerrado de Costo **J**. Imagen tomada de [1]

1.2. Motivación

La estabilización de flujos turbulentos es una de las ramas más estudiadas dentro del control de fluidos. Las dificultades presentes en la resolución de este tipo de problemas, sumado a la gran cantidad de campos de aplicación, hacen del mismo un tema atractivo para científicos alrededor del mundo. Se listan a continuación algunas de las características más importantes:

- La cantidad de grados de libertad (debido a la naturaleza no lineal y caótica de los mismos) **dificulta la modelización del sistema físico**.
- La naturaleza no lineal de esta clase de flujos impide aplicar el principio de superposición sobre los efectos producidos por cada actuador.
- Perturbaciones externas **impiden encontrar una ley de control que lleve a la convergencia del sistema**, aún cuando se haya realizado una correcta modelización del mismo.
- Los flujos turbulentos poseen **estabilidad estacionaria** (existen valores definidos para variables estadísticas como la media y varianza). Esto permite conocer el tiempo tarda el sistema en volver a su estado original luego de ser estimulado, haciendo posible la reproducibilidad de los experimentos a realizar.

Teniendo en cuenta las características enumeradas, el campo de los fluidos turbulentos permite encontrar leyes de control válidas basadas en prueba y error. Esto fue lo que motivó el desarrollo del framework MLC, el cual se describe a continuación.

1.3. MLC

1.3.1. Arquitectura

MLC es un framework utilizado para descubrir leyes de control. Tiene como fin generar modelos de sistemas de forma automática, los cuales llamaremos controladores, a través de información de la calidad de la solución generada. Esta información es procesada a través de métodos basados en *Machine Learning*, los cuales están diseñados para buscar nuevos controladores dentro del espacio de soluciones existente. La arquitectura básica del mismo se exhibe en la figura 4.

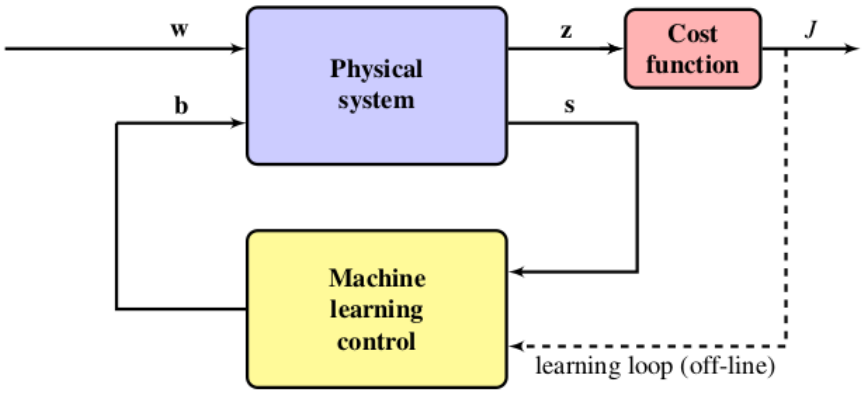

Fig. 4 Arquitectura del MLC. Imagen tomada de [1]

Comparando las figuras 3 y 4, se puede observar como la ley de control es reemplazada por el MLC, el cual recibe como entrada los valores sensados **s** y un costo asociado a la ley de control, resultante de la aplicación de la función de costo **J** ante la salida del sistema físico excitado. En función de estas muestras,

MLC genera una nueva ley de control con el objetivo de *minimizar/maximizar* el resultado de la evaluación de la función de costo **J**.

La *función de costo* debe ser definida por el usuario del MLC. La misma debe adecuarse al tipo de problema que se desea resolver. Se puede pensar como ejemplo la estabilización de un flujo turbulento: se puede considerar que el sistema converge si la velocidad del mismo en diferentes puntos es semejante (deja de ser turbulento para pasar a ser laminar). Una *función de costo* válida en este contexto sería aquella que arroje un número que tienda a cero en cuanto las velocidades tienden a ser homogéneas.

1.3.2. Machine Learning y Programación Genética

Machine Learning es una subrama de las ciencias de la computación basada en las campos de *Pattern Recognition* y *Computational Learning Theory*. La misma es utilizada, entre otros usos, para identificar sistemas dentro de un espacio de soluciones de alta dimensionalidad (*high dimensional search space*) a través de técnicas de aprendizaje. Existe una gran cantidad de técnicas a utilizar dentro del campo de **Machine Learning**, tales como *Gradient Search* o *Monte-Carlo Statistical Analysis*, pero las mismas poseen como característica principal la obtención de soluciones subóptimas o asociadas con un costo alto cuando el espacio de soluciones es extremadamente grande [1]. Es aquí donde entran en juego los algoritmos evolucionarios, los cuales han demostrado ser óptimos a la hora de resolver este tipo de problemas.

La *Programación Genética* y los *Algoritmos Genéticos* se basan en la propagación de generaciones de individuos, llamadas poblaciones, los cuales son seleccionados en función del resultado de la evaluación de una *función de costo* **J**, también llamada *función de fitness*. Dicho resultado es conocido como *costo* o *fitness*. Un individuo en este ámbito **tiene una correspondencia directa con la estructura y parámetros que definen a una ley de control**. En la figura 5 se exhibe un diagrama detallado del uso de estos algoritmos dentro de la solución implementada en MLC. Un individuo $\mathbf{b} = \mathbf{K}(\mathbf{s})$ es evaluado en el sistema dinámico. De dicha evaluación, se obtiene su *fitness*. Al terminar de evaluar una población entera, MLC genera una nueva población a través del uso de diferentes técnicas de *Programación Genética*. Este proceso se exhibe en detalle en la sección 1.3.3.

Un individuo se encuentra modelizado dentro del MLC como un conjunto de funciones y parámetros que poseen la forma de un *recursive function tree*. Las funciones matemáticas utilizadas son las operaciones +, -, ×, / con el agregado de cualquier otro tipo de función, como puede llegar a ser un coseno, una exponencial o una tangente hiperbólica por nombrar algunas. Los nodos hojas poseen constantes o bien parámetros del sistema (por lo general sensores), mientras que los nodos no hoja poseen las funciones matemáticas anteriormente nombradas. La figura 6 muestra la modelización de un individuo. La composición de los individuos a través de *trees* hacen de los mismos candidatos ideales para ser utilizados como input dentro de *algoritmos genéticos*.

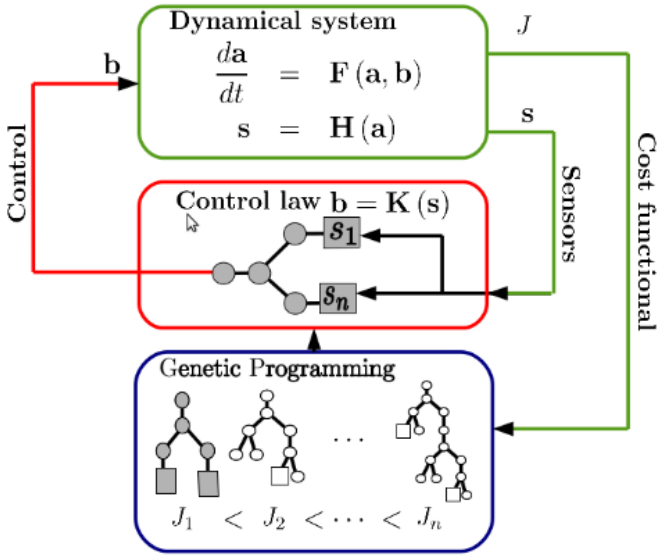

Fig. 5 Generación de nuevas leyes de control. Imagen tomada de [1]

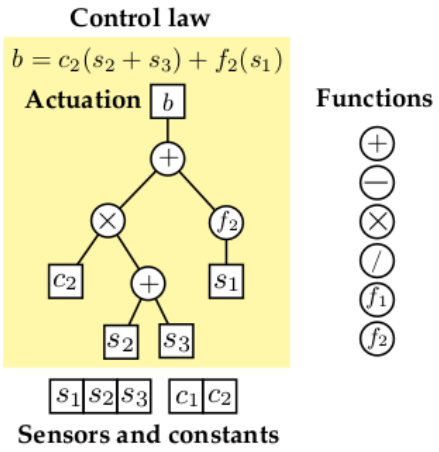

Fig. 6 Individuo $\mathbf{b} = \mathbf{K}(\mathbf{s_1}, \mathbf{s_2}, \mathbf{s_3})$ visto como un *recursive funcion tree*. Imagen tomada de [1]

1.3.3. Generación de nuevas Leyes de Control

La primera población de individuos es generada de forma aleatoria en función de los sensores, constantes y funciones matemáticas a disposición y la misma no posee un costo asociado a priori. La población luego es evaluada. Este proceso consiste en tomar muestras de los sensores involucrados en la ley de control y reemplazarlas en la función matemática que define a un individuo. De dicha evaluación se obtiene el *fitness* asociado a cada individuo. A partir de este punto se procede a evolucionar la población. Dicha evolución se lleva a cabo a través de los siguientes algoritmos [1]:

- **Elitismo:** Se eligen a los mejores individuos de la población evaluada y se agregan directamente en la próxima generación.
- **Replicación:** De forma aleatoria, se eligen algunos individuos de la población evaluada y se hace avanzar a los mismos a la próxima población.
- **Crossover:** Se toman a dos individuos que posean un *fitness* similar y se intercambian de forma aleatoria algunas ramas del *recursive tree* asociado a cada uno.
- **Mutación:** Se modifican de forma aleatoria los parámetros o funciones de ciertos individuos.

Luego de realizar la evolución de la población se procede nuevamente a evaluar a la misma. Este proceso se repite hasta que se llega a algún punto de corte. Por lo general, el punto de corte del algoritmo utilizado por MLC consiste en alcanzar cierto número de generaciones evaluadas. Otro punto de corte que se puede utilizar es el alcance de un valor de *fitness* deseado en el mejor individuo. En la figura 7 se muestra un diagrama de flujo explicando el proceso de generación de nuevas poblaciones.

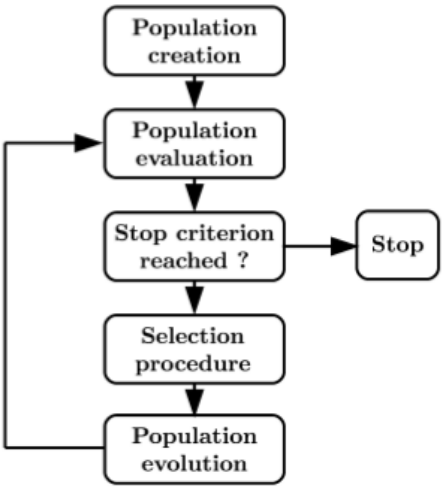

Fig. 7 Diagrama de Flujo del proceso de Generación de Poblaciones. Imagen tomada de [1]

2. Objetivo

Durante los años que Thomas ha estado trabajando en MLC se ha encontrado con una serie de dificultades técnicas y funcionales. Durante el segundo cuatrimestre del 2015, como parte de un trabajo práctico de la asignatura *75.61 Taller de Programación III* se trabajó sobre el sistema MLC, relevando requerimientos e implementando en forma parcial alguno de ellos. Debido al corto período tiempo dispuesto en la asignatura y la cantidad de tareas a realizar, se decidió continuar con el mismo como parte del presente Trabajo Profesional. Con más recursos a disposición y luego de diferentes reuniones que hemos tenido con Thomas, se llegó a una lista de objetivos en común. La misma se exhibe a continuación:

- **Migrar el sistema de MATLAB a Python:** MLC se encuentra desarrollado enteramente en MATLAB, con excepción del código que es utilizado en los dipositivos *Arduino*. Las principales razones por las cuales se desea realizar esta migración son:
  - Gracias a la cantidad de bibliotecas científicas robustas que dispone el lenguaje (*numpy, matplotlib, SciPy Stack, etc.*), Python permite desarrollar aplicaciones de procesamiento numérico y matemático 100 % Open Source
  - Si bien MATLAB es el lenguaje más utilizado en la comunidad científica, no es un buen lenguaje para diseñar aplicaciones de software complejas. Entre las principales desventajas que posee, se pueden listar el limitado soporte a OOP (*Object Oriented Programming*) y estructuras de datos y sintaxis a medida para el procesamiento de matrices.
- **Pruebas unitarias, de integración y funcionales:** El software no posee ningún tipo de testing. Hoy en día el correcto funcionamiento del software es validado de forma manual por Thomas. Es de interés agregar tests que permitan obtener una medida de la cobertura de código alcanzada, como así definir métricas automáticas que permitan a futuro ponderar la calidad del sistema.
- **Refactorización de Código:** Gran cantidad del proyecto se encuentra realizado con las herramientas que MATLAB brinda a nuestra disposición. La migración a Python abre la posibilidad de cambiar el diseño actual del sistema por uno más flexible y robusto.
- **Cuellos de Botella:** Actualmente la forma en la que se evalúan los individuos (o su equivalente, *Control Laws*) es poco flexible. El procedimiento implica reprogramar los dispositivos *Arduino* ante cada población a ser evaluada. Esto último es llevado a cabo por temas de *performance*, debido a que ha sido imposible para Thomas evaluar cada individuo por separado. Se desea identificar y reproducir las limitaciones encontradas por Thomas y encontrar soluciones a las mismas, siempre que estas existan.
- **Interfaz Gráfica:** MLC no posee interfaz gráfica alguna. Se desea crear una interfaz de escritorio con una correcta usabilidad que permita a usuarios de la comunidad científica aprovechar al máximo las virtudes del presente framework.

3. Alcance

3.1. Requerimientos Funcionales

- Migrar el sistema completo de *MATLAB* a *Python* a través de un modo de trabajo vertical. Esto implica ir reemplazando código *MATLAB* a *Python* de forma gradual, verificando en cada paso el correcto funcionamiento del sistema.
- Utilizar el MLC desde Python con *Simulink* de forma opcional. Debido a que gran cantidad de los experimentos realizados por *Thomas* son diseñados en *Simulink*, y dado que no existe una reemplazo a esta herramienta, se debe respetar esta compatibilidad.
- Diseñar e implementar un modelo de persistencia de la aplicación basado en Base de Datos. En estos momentos los datos de las simulaciones realizadas se almacenan en formato binario. Se debe permitir manipular varios experimentos a la vez sin la necesidad de estar cambiando los archivos de configuración como se está realizando actualmente.
- Rediseñar el procesamiento de las leyes de control. Esto es lo que actualmente se está realizando de forma desprolija en el *Arduino*.
- Diseñar e implementar un protocolo de comunicación que permita configurar y evaluar las leyes de control. Esto implica transformar los individuos de un *recursive function tree* a un formato que pueda interpretado por los dispositivos *Arduino* u otro dipositivo que respete el protocolo en cuestión. Se debe evitar, siempre que se pueda lograr, tener que quemar el *Arduino* en cada iteración.
- Desarrollar una API genérica que permita comunicar al sistema MLC con el sistema embebido. De esta forma se busca desacoplar al sistema en su totalidad de algún dispositivo de hardware específico, dejando abierta la posibilidad de poder cambiar el mismo por alguna otra plaqueta (Raspberry, beagleboard, etc.) en un futuro.
- Implementar una interfaz gráfica que permita configurar los parámetros de entrada del sistema.
- Implementar una interfaz gráfica para la visualización y el análisis de los resultados de las simulaciones realizadas.

3.2. Requerimientos No Funcionales

- El sistema debe ser multiplataforma y debe estar implementado con tecnologías *Open Source*. Una excepción a esto es el uso de *Simulink*, como se explicó en la sección 3.1.
- Se debe evaluar que la velocidad de procesamiento de cada individuo sea menor al tiempo de actuación del sistema físico luego de la refactorización. En caso de no cumplirse este objetivo, se debe utilizar el esquema de trabajo actual y pensar en futuras implementaciones y/o tecnologías que permitan resolver este problema.

- Generar un set de pruebas de integración que permitan validar el funcionamiento del sistema con el cliente.
- Implementar un conjunto de test unitarios que garanticen la cobertura de código de los módulos del sistema de acuerdo a las métricas de calidad definidas.
- Relevar qué métricas se están usando en el laboratorio para medir la performance y qué números está manejando Thomas en este momento.
- Realizar un estudio de mercado sobre las tecnologías embebidas comerciales existentes, de forma de encontrar el producto óptimo que cumpla con la relación velocidad de comunicación / precio. Uno de los objetivos que se busca a través del MLC, es lograr que los dispositivos físicos a utilizar para el sensado de parámetros y actuación sea barato y fácil de conseguir en cualquier parte del mundo.

4. Especificaciones del Sistema

4.1. Hardware

4.1.1. MLC

MLC no posee limitaciones de hardware conocidas. Por esta razón, los requerimientos mínimos que se piden para ejecutar la aplicación son los mismos pedidos por un sistema Linux. Los requerimientos mínimos se listan a continuación:

- Procesador x86 de 1 GHZ
- 512 MB RAM (1 GB recomendado)
- 20 GB de espacio en disco (la mayoría de este espacio está destinado a la instalación del MATLAB. En un futuro cuando se remueva la dependencia con MATLAB podrá reducirse este número, siempre y cuando no se utilice *Simulink*)

4.1.2. Sensado y Actuación

Los dispositivos utilizados para realizar el sensado de magnitudes físicas y la actuación de dispositivos físicos son *Arduinos*. El modelo utilizado del mismo no ha sido definido aún, pero se están realizando pruebas con el modelo *Due*. Es posible que este modelo se cambie por otro *Arduino* o por otro dispositivo embebido que cumpla con los requerimientos necesarios.

4.2. Software

4.2.1. MLC

En su versión terminada, el sistema correrá enteramente en **Python 2.7**. Se utilizarán todas las bibliotecas matemáticas necesarias para reemplazar la funcionalidad de MATLAB en Python. A priori, se supone que con el conjunto provisto por el *Scipy Stack* será suficiente, pero no se descarta tener que agregar alguna otra dependencia al proyecto.

Para realizar la migración a Python, se utilizará el módulo *Python Engine*. El mismo permite ejecutar código MATLAB en Python. Este módulo es provisto por MATLAB a partir de la versión **R2014 B**. Como bien se ha comentado, en su versión final MATLAB no será necesario para utilizar MLC, pero existen algunas herramientas provistas por *Mathworks* como *Simulink* que no poseen un reemplazo. Por esta razón es que se agrega como una de las dependencias de software a MATLAB.

Aún no se ha decidido que framework gráfico será utilizado para realizar la interfaz gráfica de la aplicación. Se debe evaluar entre las diferentes versiones disponibles para Python (PyQT4, PyQT5, Pyside, PyGTK, wxWidgets, etc.) y elegir la que sea más conveniente.

En lo que respecta a base de datos, tampoco se ha decidido aún que motor se utilizará. Debido a que el volumen de datos manejados por experimento es baja (cota superior de 500 MB por experimento) se analizará utilizar algún motor ligero como SQLite. En caso que se necesite un motor más robusto, se

analizará el uso de otras opciones como pueden llegar a ser MySQL, Postgres, MariaDB o algún tipo de motor No SQL.

4.2.2. Sensado y Actuación

El software utilizado para el desarrollo de los dispositivos embebidos está fuertemente atado al SDK liberado por cada fabricante. En el caso de *Arduino*, los mismos proveen un IDE que viene integrado con un cross-compiler compatible con el hardware disponible en cada una de las placas a disposición (*Arduino Uno, Due, Mega, etc.*). Habiendo dicho esto, todo el desarrollo con *Arduino* se realizará en C y C++.

5. Metodología

La metodología de trabajo elegida para trabajar en el proyecto es *Scrum*. El uso de esta metodología es ideal para el tipo de desarrollo a realizar, debido a que se posee un cliente real con el cual trabajar. La duración de cada uno de los sprints variará entre dos y tres semanas. En la finalización de cada sprint, se realizará una demo con el cliente mostrando los avances realizados.

Se utilizará **Trello** como herramienta de gestión de proyecto. En el mismo se encontrará el *backlog* y cada uno de los *sprints* realizados durante el transcurso del proyecto. A través de los plugins **Ollert** y **BurndownChartForTrello** se mantendrá el registro de las horas trabajadas en cada tarea, así como también se tomarán estadísticas de la eficiencia del equipo de trabajo.

El hosting del proyecto se encontrará en un repositorio privado por pedido expreso de Thomas. Al finalizar el presente trabajo se liberará el código a la comunidad. El hosting utilizado actualmente es **www.github.com**, y el mismo se utilizará además para mantener trazabilidad entre las tareas definidas en **Trello** y las modificaciones de código realizadas.

6. Cronograma

La asignatura *75.99 Trabajo Profesional* posee asociados 12 créditos en el plan de estudios de la carrera Ingeniería en Informática. Esto implica que, según el estatuto de la facultad, la misma posee una carga horaria de 24 horas semanales (12 de cursada + 12 horas de trabajo por cuenta del alumno). Teniendo en cuenta que la duración de un cuatrimestre es de 16 semanas, la cantidad de horas asignadas a cada alumno equivale a 384 horas. Este número por 3 (la cantidad de integrantes del presente proyecto), da como resultado el número total de horas del proyecto: 1152. Se redondea este número de horas a 1200 para realizar la planificación del proyecto. A continuación se define el cronograma tentativo de tareas:

Tarea	**Descripción**	**Horas**
Gestión de Proyecto	Seguimiento de tareas, análisis de métricas, planeamiento de sprints y desarrollo del proyecto	75
Reuniones de Avance	Tiempo dedicado a las reuniones a tener tanto con el cliente como con los tutores. Esto último incluye las demos a realizar en la finalización de cada *sprint*	50
Migración algoritmos genéticos	Migrar de MATLAB a Python la generación y evolución de individuos	200
Diseño e Implementación del modelo de persistencia	Se deben modelar los datos, estructuras y relaciones presentes en el MLC a fin de persistir los experimentos en algún esquema de base de datos a definir	45
Análisis e Implementación de Gráficos	Se debe analizar que herramientas provistas por Python se amoldan a los requerimientos gráficos de la aplicación, que incluye la implementación de los gráficos existentes	40
Obtención de métricas y análisis de disp. *Arduino*	Se realizarán pruebas de desempeño de hardware de comunicación para dispositivos *Arduino*, a fin de determinar la viabilidad del procesamiento de las leyes de control en Python	30
Diseño e Implementación de Protocolo de Comunicación	El protocolo debe ser capaz de configurar los sensores y actuadores a utilizar en un experimento dado. También debe ser capaz de obtener lecturas de los sensores previamente configurados y manipular los actuadores	100
Confección de Pruebas de Integración	Se debe generar un conjunto de pruebas que permitan validar el correcto funcionamiento entre las diferentes partes que componen al MLC	30
Diseño y Armado de Entorno de Prueba Real	Se debe implementar un experimento de laboratorio que permita evaluar el correcto funcionamiento y desempeño del MLC	60

Análisis y Diseño de Mejoras de Arquitectura	Una vez que se haya migrado el MLC a Python, se procederá a evaluar mejoras en el diseño del sistema	90
Implementación de Mejoras de Arquitectura	Con la nueva arquitectura y diseño en mente, se procederá a implementar la nueva solución propuesta	270
Documentación Interna de diseño	Documentación que describa el diseño y arquitectura del sistema. La misma debe incluir diagramas de clases, secuencia, etc., como así también el detalle del protocolo de comunicación implementado	60
Diseño e Implementación de Interfaz Gráfica	Se debe desarrollar una interfaz gráfica agradable para el usuario que permita la administración y configuración del sistema. La misma debe permitir configurar y visualizar los experimentos a realizar	30
Implementación de Instaladores	Se deben crear instaladores para los sistemas operativos más conocidos (Windows, MAC, Linux flavors) con el fin de facilitar el uso del MLC en la comunidad científica	60
Elaboración de Manual de Usuario	Se deben confeccionar los manuales de usuario para el uso y extensión de la versión final del sistema	60
Total Horas:		**1200**

7. Datos Integrantes

7.1. Ezequiel Torres Feyuk

7.1.1. CV

Nicolás Rodrigo Ezequiel Torres Feyuk | Curriculum Vitae

Adolfo Alsina 2550 2B, CABA

+54 911 6940 7659 • ezequiel.torresfeyuk@gmail.com
Fecha de Nacimiento: 17 Mayo 1989 - DNI: 34.650.445

Estudiante de Ingeniería Informática actualmente completando el último año de la carrera de grado. Perseverante y proactivo, es de mi interés trabajar en equipos interdisciplinarios que permitan aprovechar los conocimientos adquiridos durante el transcurso de mi carrera profesional.

Trabajos Previos

- **Facultad de Ingeniería** — **San Telmo**
  Colaborador Universitario 66.02 Laboratorio — *Agosto 2009–Septiembre 2012*
  Como colaborador, mi rol consistió en dar apoyo a los estudiantes a la hora de adquirir la teoría básica en la rama de la metrología así como en la manipulación de los dispositivos de medición utilizados en la materia (Multímetro, Amperímetro, Osciloscopio, Contador, etc.). En los últimos años tuve la posibilidad de preparar clases y parciales con la supervisión del coordinador de la materia.

- **Veccsa S.A.** — **V. Urquiza**
  Firmware and Software Developer — *July 2012–June 2013*
  Mi trabajo consistió en mantener aplicaciones para la creación de informes médicos y procesamiento de señales médicas de dispositivos tales como Holters Cardíacos y Ergometrías. Durante este tiempo, también tuve como responsabilidad el mantenimiento del firmware de los dispositivos médicos, los cuales consistían en el sensado y almacenamiento de las señales obtenidas de los pacientes.

- **Intraway S.R.L.** — **V. Urquiza**
  Software Developer — *Septiembre 2013–Actualidad*
  Mi trabajo en Intraway consiste en el mantenimiento de diferentes aplicaciones de software dentro del estandar **DOCSIS**. En particular, poseo ownership del componente TFTP, el cual se encarga de confeccionar los archivos de configuración dentro de la red de los *ISPs (Internet Service Providers)* requeridos por Cablemodems y MTAs.

Education

Academic Qualifications

- **Facultad de Ingeniería** — **San Telmo**
  Ingeniería en Informática, Principios de 2017 — *2007-Actualidad*
- **Colegio Parroquial Juan Beat XXIII** — **Ramos Mejía**
  Título Técnico en Informática Profesional y Personal, Polimodal — *2003-2006*
- **Colegio María Mazzarello** — **San Justo**
  Nivel Primario — *1994–2002*

Proyectos

- **DOCSIS Device Simulator:** *'Desarrollo de Simulador de Dispositivos'*

1/2

Estuve trabajando durante más de un año en el diseño e implementación de un simulador masivo de dispositivos. El mismo permite emular el flujo de diferentes dispositivos dentro del estandar **DOCSIS** enviando paquetes a través de la red mediante el uso de *Raw Sockets* para simular el uso de IPs ficticias. Algunos de los protocolos que el mismo soporta son *DHCP, TFTP, SNMP, ToD, MGCP y COPS*. El flujo de cada dispositivo es configurable, y se han realizado pruebas exitosas con más de 1 millón de dispositivos.

Conocimientos Técnicos

- **Lenguajes de Programación:** Semi-Senior en: C, C++, Python
  Conocimientos en otros lenguajes: Java, JavaScript, PL/SQL, Matlab, LaTeX, XML, bash scripting.
- **Habilidades en Software:** Conocimientos avanzado en IPCs y programación con concurrente (Threads y Sockets). Conocimientos intermedios en Sistemas Distribuídos.
- **Other:** Conocimientos básicos en Procesamiento de Señales Determínisticas y Estocásticas, diseño y armado de circuitos integrados. Experiencia en programación con microntroladores (Arduino, PICs, etc.)

7.1.2. Listado de Materias

Cód.	Asignatura	Nota	Acta	Fecha	Créd.
75.40	Algoritmos y Programación I	10	17-098-175	04/07/08	6
61.08	Álgebra II	6	01-150-248	10/07/08	8
61.03	Análisis Matemático II A	7	01-151-234	06/08/08	8
63.01	Química	6	03-073-207	12/12/08	6
61.09	Probabilidad y Estadística B	6	01-155-032	11/02/09	6
62.01	Física I A	7	02-107-089	12/02/09	8
75.41	Algoritmos y Programación II	8	17-101-086	04/08/09	6
62.03	Física II A	5	02-107-156	05/08/09	8
62.15	Física III D	6	02-108-001	23/12/09	4
75.07	Algoritmos y Programación III	8	17-102-210	29/12/09	6
61.10	Análisis Matemático III A	7	01-149-225	10/02/10	6
66.70	Estructura del Computador	8	06-138-111	11/02/10	6
66.02	Laboratorio	4	06-138-157	23/02/10	6
75.12	Análisis Numérico I	8	17-104-058	14/07/10	6
66.06	Análisis de Circuitos	7	06-140-063	23/02/11	10
75.06	Organización de Datos	8	17-106-086	03/03/11	6
75.42	Taller de Programación I	9	17-106-199	06/07/11	4
66.20	Organización de Computadoras	5	06-141-016	08/08/11	6
75.08	Sistemas Operativos	10	17-107-158	11/08/11	6
71.14	Modelos y Optimización I	8	11-153-164	13/02/12	6
66.74	Señales y Sistemas	7	06-141-187	17/02/12	6
66.09	Laboratorio de Microcomputadoras	9	06-142-046	13/07/12	6
66.74	Procesos Estocásticos	6	06-142-131	06/08/12	6
75.59	Técnicas de Programación Concurrente I	6	17-110-020	08/08/12	6
75.09	Análisis de la Información	5	17-110-135	10/12/12	6
75.43	Introducción a los Sistemas Distribuidos	8	17-111-028	28/12/12	6
75.26	Simulación	7	17-111-048	06/02/13	6
75.10	Técnicas de Diseño	6	95-0001028	08/07/13	6
66.08	Circuitos Electrónicos I	7	86-0001011	10/07/13	8
75.15	Base de Datos	7	95-0001331	07/08/13	6
75.52	Taller de Programación II	5	95-0002383	04/02/14	4
75.74	Sistemas Distribuídos I	5	96-0001462	06/02/14	6
71.40	Leg. y Ej. Prof. de la Inf. en Informat.	5	71-0001509	21/02/14	4
71.12	Estructura de las Organizaciones	6	71-0001490	26/02/14	6
75.67	Sist. Autom. De Diag. y Detec. de Fallas I	7	95-0002261	04/08/14	6
75.65	Manufactura Integrada por Comp. (CIM) I	7	95-0002365	14/08/14	6
75.68	Sist. De Soporte P/Celdas de Prod. Flexib.	10	95-0002440	10/12/14	4
75.66	Manufactura Integrada por Comp. (CIM) II	7	95-0002462	11/12/14	6
64.05	Estática y Resistencia de los Materiales B	8	64-0001643	09/02/15	6
75.61	Taller de Programación III	8	95-0003741	25/02/16	6
Total Créditos					**244**

7.2. Raúl Lopez Skuba

7.2.1. CV

Raúl López

Datos Personales

Lugar y fecha de nacimiento:	Buenos Aires, 29 de Mayo de 1987
Dirección:	calle Alberti nº 1764 piso 8 depto. 2 , Capital Federal
Teléfono:	15-30324843
email:	raulopez0@gmail.com

Experiencia Laboral

Octubre 2013-Julio 2015 | Backend Solution Developer C/C++,"Intraway corp."
Diseño, desarrollo y soporte de aplicaciones en lenguajes C/C++ y Python para control, mensajería y provisioning de Set-Top Boxes de TV digital. Implementación de protocolos para comunicación con sistemas de acceso condicional (CAS) pertenecientes a diversas tecnologías.

Abril 2012-Agosto 2013 | Analista Programador Java Freelance
Desarrollo de aplicaciones Multithreading, acceso a base de datos SQL-SERVER y Web-Servers sobre HTTPS.

Febrero 2011-Marzo 2012 | Analista Programador C,"ENOFIR Systems sa"
Diseño y programación de aplicaciones cliente-servidor para diversas plataformas POS (Point of Sale).

Educación

2007 - Presente | Ingeniería en Informática (orientación en sistemas distribuidos).
Actualmente cursando el último año de la carrera.
Facultad de Ingeniería, Universidad de Buenos Aires.

2004 | Bachiller en economía y gestión de las organizaciones.
Instituto J.M. Estrada.

Conocimientos Técnicos

Lenguajes de programación (avanzado):	C, C++, Java, Python
Otros Lenguajes de programación:	nodejs, php, C#, Javascript, Shell scripting, Perl
Frameworks/Libraries:	ACE, Boost, Googletest, Axis
Bases de Datos:	SQLServer, MySQL, OracleDB
GUI:	Swing, gtk, gtkmm
Plataformas:	Linux, Windows
SCM:	Svn, Git
Otros:	Eclipse, POO, Patrones de diseño, Valgrind, GDB UML, Unit Testing, Programación concurrente, RPM build.

Idiomas

Inglés:	Intermedio
Italiano:	Básico

7.2.2. Listado de Materias

Cód.	Asignatura	Nota	Acta	Fecha	Créd.
75.40	Algoritmos y Programación I	6	17-096-221	24/07/2007	6
62.01	Física I A	7	02-105-200	24/07/2007	8
61.03	Análisis Matemático II A	5	01-151-171	09/08/2007	8
62.03	Física II A	8	02-106-064	15/02/2008	8
75.41	Algoritmos y Programación II	8	17-098-095	05/03/2008	6
61.08	Álgebra II	7	01-150-238	07/03/2008	8
63.01	Química	8	03-073-178	31/07/2008	6
66.70	Estructura del Computador	5	06-136-172	18/12/2008	6
62.15	Física III D	10	02-107-085	22/12/2008	4
66.02	Laboratorio	8	06-136-199	22/12/2008	6
75.07	Algoritmos y Programación III	7	17-100-160	18/02/2009	6
61.10	Análisis Matemático III A	8	01-149-193	11/08/2009	6
66.20	Organización de Computadoras	8	06-137-165	13/08/2009	6
75.12	Análisis Numérico I	6	17-101-239	25/08/2009	6
75.06	Organización de Datos	9	17-105-167	20/12/2010	6
75.42	Taller de Programación I	9	17-105-177	21/12/2010	4
71.12	Estructura de las Organizaciones	8	11-151-180	22/12/2010	6
61.09	Probabilidad y Estadística B	9	01-159-001	10/02/2011	6
75.08	Sistemas Operativos	8	17-107-012	14/07/2011	6
71.14	Modelos y Optimización I	6	11-152-220	27/07/2011	6
75.52	Taller de Programación II	9	17-108-167	17/02/2012	4
75.09	Análisis de la Información	6	17-108-183	22/02/2012	6
75.59	Técnicas de Programación Concurrente I	8	17-109-072	06/07/2012	6
75.43	Introducción a los Sistemas Distribuidos	6	17-110-009	07/08/2012	6
75.10	Técnicas de Diseño	8	17-110-055	13/08/2012	6
66.69	Criptografía y Seguridad Informática	9	06-143-011	17/12/2012	6
75.74	Sistemas Distribuídos I	10	17-111-154	27/02/2013	6
66.06	Análisis de Circuitos	5	86-0001172	31/07/2013	10
75.15	Base de Datos	5	95-0001832	12/02/2014	6
66.74	Señales y Sistemas	8	86-0001948	04/08/2014	6
75.45	Taller De Desarrollo De Proyectos I	7	95-0002377	07/08/2014	6
75.61	Taller de Programación III	8	95-0002939	02/07/2015	6
75.26	Simulación	6	95-0003075	15/07/2015	6
67.61	Fundam. Matem. De La Visión Robótica	9	67-0002447	30/07/2015	6
71.40	Leg. y Ej. Prof. de la Inf. en Informat.	5	71-0002162	06/08/2015	4
75.62	Técnicas de Programación Concurrente II	10	95-0003389	10/12/2015	4
75.67	Sist. Autom. De Diag. y Detec. de Fallas I	8	95-0003442	16/12/2015	6
75.50	Introducción A Los Sistemas Inteligentes	6	95-0003475	21/12/2015	6
66.26	Arquitecturas Paralelas	7	86-0002857	14/07/2016	6
Total Créditos					**236**

7.3. Marco Germano Zbrun

7.3.1. CV

Marco Guido Germano Zbrun

Montevideo 1141 2°C, Buenos Aires (C.P. 1019), Capital Federal, Argentina
(011) 4811-5520 • (15) 6464-8194 • sw.xwing.alliance@gmail.com

Datos personales

Fecha de nacimiento: 11 de septiembre de 1986
Lugar de nacimiento: Santa Rosa, Pcia de La Pampa.
Nacionalidad: Argentina
Edad: 29
DNI: 32.046.595
Estado civil: Soltero

Educación

Universitaria
Universidad de Buenos Aires, Facultad de Ingeniería
Actualmente cursando el 4to año de la carrera de Ingeniería en Informática, 2005 – Presente
Materias aprobadas: 37

Terciario
Instituto Tecnológico Argentino
Técnico en Hardware de PC, 2004 – 2005.
Promedio: 9,40

Secundaria
Escuela Normal Superior N°1 en Lenguas Vivas “Presidente Roque Saenz Peña”.
Bachiller especializado en ciencias Fisico-Matemáticas con intensificación en idiomas extranjeros, 2000 – 2004.
Promedio: 7,50

Conocimientos y habilidades

- Sólidos conocimientos de Programación Orientada a Objetos (POO).
- JavaSE. Manejo de *threads*, colecciones, *generics*, etc. Diseño y codificación de interfaces gráficas de usuario con Swing.
- Dominio de C++. Diseño y codificación de interfaces gráficas con GTKmm. Uso de GDB para debugging. Uso de Valgrind para *profiling* y rastreo de *memory leaks*. Conocimientos de bibliotecas Boost.
- Python. Manejo Gral del lenguaje.
- Sólidos conocimientos de C. Utilización del paradigma de TDA. Manejo de *POSIX Threads y BSD sockets*.
- POSIX y System V IPC.
- Algoritmos de compresión de datos (Huffman, LZ77, LZ78, LZP, etc.)
- Conocimientos de compresión/descompresión de video digital (mjpeg, mpeg, mpeg2 y mpeg4)
- Manejo de SVN/Subversion y Git para el control de versiones.
- Manejo de bases de datos con MySQL Oracle DB. Conocimientos sobre normalización.
- XML, XML Schema, Json y Json Schema.
- Windows y Linux.
- Shell scripting en Bash. Expresiones regulares.
- Suites de oficina MS Office y OpenOffice.

Experiencia laboral

Intraway Corporation
Diciembre 2012 – Actualidad, CABA, Argentina
Desarrollador Backend (C/C++)

- Desarrollo y soporte de productos de aprovisionamiento/control de televisión CATV/DTH
- Diseño y mantenimiento de protocolos tipo SI de DVB

Pasantía, Core Security Technologies
Abril 2011 – Octubre 2011, CABA, Argentina

- Adaptación del kernel de Core Impact Pro para dar soporte a IPv6. El desarrollo incluyó la adaptación a bajo nivel de los distintos componentes que funcionan en los sistemas operativos soportados por el producto (Windows, Linux, MacOS X, Aix y Solaris). Esta tarea se realizó utilizando, principalmente, C.
- Desarrollo y mantenimiento del módulo que da soporte a la biblioteca Pcap dentro de la implementación del software. Esta tarea se realizó utilizando los lenguajes de C y C++.
- Desarrollo de scripts en python para la automatización de tareas.

Idiomas

Español
- Nativo

Inglés
- Nivel de lectura avanzado.
- Nivel de escritura intermedio.
- Nivel oral intermedio.

Francés
- Nivel de lectura básico.
- Nivel de escritura básico.
- Nivel oral básico.

7.3.2. Listado de Materias

Cód.	Asignatura	Nota	Acta	Fecha	Créd.
62.01	Física I A	5	02-104-214	18/07/2006	8
61.03	Análisis Matemático II A	8	01-151-103	31/07/2006	8
75.40	Algoritmos y Programación I	8	17-095-208	06/02/2007	6
61.08	Álgebra II A	6	01-150-190	01/03/2007	8
75.41	Algoritmos y Programación II	7	17-096-114	09/03/2007	6
61.07	Matemática Discreta	5	01-152-163	08/08/2007	6
75.07	Algoritmos y Programación III	8	17-097-115	26/09/2007	6
75.12	Análisis Numérico I	9	17-097-198	26/12/2007	6
63.01	Química	7	03-073-081	09/03/2007	6
62.03	Física II A	6	02-105-193	19/07/2007	8
66.02	Laboratorio	7	06-135-040	26/12/2007	6
75.42	Taller de Programación I	9	17-099-096	05/08/2008	4
61.09	Probabilidad y Estadística B	4	01-152-243	06/08/2008	6
66.70	Estructura del Computador	6	06-135-113	29/02/2008	6
66.20	Organización de Computadoras	6	06-136-078	14/08/2008	6
78.01	Idioma Inglés	8	18-022-210	29/06/2009	4
75.06	Organización de Datos	7	17-101-161	12/08/2009	6
75.08	Sistemas Operativos	9	17-103-110	25/02/2010	6
75.09	Análisis de la Información	5	17-103-145	01/03/2010	6
75.10	Técnicas de Diseño	8	17-104-200	04/08/2010	6
75.45	Taller De Desarrollo De Proyectos I	8	17-105-141	16/12/2010	6
71.12	Estructura de las Organizaciones	4	11-151-204	09/02/2011	6
75.15	Base de Datos	7	17-107-110	03/08/2011	6
75.48	Calidad En Desarrollo De Sistemas	7	17-107-094	01/08/2011	4
71.40	Leg. y Ej. Prof. de la Inf. en Informat.	6	11-153-096	16/12/2011	4
62.15	Física III D	6	02-109-188	08/02/2012	4
75.44	Adm. y control de Proy. Informáticos I	6	17-108-214	28/02/2012	6
75.47	Taller De Desarrollo De Proyectos II	9	17-109-052	03/07/2012	6
75.43	Introducción a los Sistemas Distribuidos	9	17-109-203	31/07/2012	6
75.46	Adm. y control de Proy. Informáticos II	6	17-110-008	07/08/2012	6
66.09	Laboratorio de Microcomputadoras	8	86-0001056	12/07/2013	6
75.59	Técnicas de Programación Concurrente I	8	95-0001537	20/12/2013	6
71.13	Información En Las Organizaciones	5	71-0001504	25/02/2014	6
71.14	Modelos y Optimización I	5	71-0001683	30/07/2014	6
75.52	Taller de Programación II	10	95-0002840	27/02/2015	4
66.17	Sistemas Digitales	10	86-0002335	16/07/2015	6
75.50	Introducción a los Sistemas Inteligentes	6	95-0003475	21/12/2015	6
75.62	Técnicas de Programación Concurrente II	10	95-0003605	25/02/2016	4
Total Créditos					**222**

7.3.3. Plan De Cursada

Las materias detalladas a continuación se cursarán en el segundo cuatrimestre del 2016.

Código	**Materia**	**Créditos**
61.10	Análisis Matemático III A	6
66.69	Criptografía y Seguridad Informática	6
71.44	Recursos Humanos	4
Total Créditos		**14**

Referencias

[1] T. Duriez and S. L. Brunton, *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*, vol. 1. Springer International Publishing, 2016.

[2] K. Ogata, *Modern Control Engineering*, vol. 17. 2002.