

UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA: INGENIERÍA ELECTRÓNICA

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO
DE INGENIERO ELECTRÓNICO**

PROYECTO TÉCNICO:

**ANÁLISIS DE LA TECNOLOGÍA ARDUPILOT ORIENTADA A LA
IMPLEMENTACIÓN DE SISTEMAS DE CONTROL AUTOMÁTICO.**

Autores:

Luis Miguel Aguilar Castro

Christian Paul Ortiz Astudillo

Tutor:

Ing. Julio Cesar Zambrano Abad

CUENCA-ECUADOR

CESIÓN DE DERECHOS DE AUTOR

Yo, Luis Miguel Aguilar Castro, con documento de identificación N° 0104842604, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de grado intitulado: “**ANÁLISIS DE LA TECNOLOGÍA ARDUPILOT ORIENTADA A LA IMPLEMENTACIÓN DE SISTEMAS DE CONTROL AUTOMÁTICO**”, mismo que ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, Enero del 2017



Luis Miguel Aguilar Castro

Cedula: 0104842604

CESIÓN DE DERECHOS DE AUTOR

Yo, Christian Paul Ortiz Astudillo, con documento de identificación N° 0106436314, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de grado intitulado: “*ANÁLISIS DE LA TECNOLOGÍA ARDUPILOT ORIENTADA A LA IMPLEMENTACIÓN DE SISTEMAS DE CONTROL AUTOMÁTICO*”, mismo que ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, Enero del 2017



Christian Paul Ortiz Astudillo

Cedula: 0106436314

CERTIFICACIÓN

Yo declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: ***“ANÁLISIS DE LA TECNOLOGÍA ARDUPILOT ORIENTADA A LA IMPLEMENTACIÓN DE SISTEMAS DE CONTROL AUTOMÁTICO”***, elaborada por Luis Miguel Aguilar Castro y Christian Paul Ortiz Astudillo, obteniendo el Proyecto Técnico que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana

Cuenca, Enero del 2017



Ing. Julio Cesar Zambrano Abad

Cédula: 0301489696

TUTOR DEL TRABAJO DE TITULACIÓN

DECLARATORIA DE RESPONSABILIDAD

Nosotros, Luis Miguel Aguilar Castro Cédula: 0104842604 y Christian Paul Ortiz Astudillo Cédula: 0106436314 autores del Trabajo de Titulación *“Análisis de la tecnología ArduPilot orientada a la implementación de sistemas de control automático”* certificamos que el total contenido del Proyecto Técnico son de nuestra exclusiva responsabilidad y autoría.

Cuenca, Enero del 2017



Luis Miguel Aguilar Castro

Cédula: 0104842604



Christian Paul Ortiz Astudillo

Cédula: 0106436314

ÍNDICE GENERAL

INTRODUCCIÓN

ANTECEDENTES

JUSTIFICACIÓN

Índice General VI

Índice de Tablas..... XI

Agradecimientos..... XII

Dedicatoria..... XIII

Agradecimientos..... XIV

Dedicatoria..... XV

ANTECEDENTES..... XVI

INTRODUCCIÓN..... XVII

1. INTRODUCCIÓN..... 1

1.1. Presentación 1

1.2. Análisis del Estado del Arte de Plataformas de Control. 1

1.2.1. ARDUINO 1

1.2.2. BEAGLEBOARD 2

1.2.3. LEGO MINDSTORMS 3

1.2.4. RASPBERRY PI..... 3

1.2.5. STM32F4 DISCOVERY 4

1.3. Alcance y Objetivos de esta tesis 4

2. ANÁLISIS DE LA PLATAFORMA ARDUPILOT 5

2.1. Análisis de los componentes del sistema embebido..... 5

2.1.1. ARDUPILOT 5

2.1.2. PARTES DEL ARDUPILOT 6

2.1.3. CARACTERÍSTICAS DEL ARDUPILOT 7

2.1.4. CONEXIÓN DE UN RECEPTOR PPM 8

2.1.5. TELEMETRIA. 9

2.1.6. MÓDULO GPS..... 10

2.2. Análisis de los pines y la alimentación del dispositivo 10

2.2.1. Pines Analógicos y Digitales. 10

2.2.2. POWER-RAIL..... 11

2.2.3. Puerto de Alimentación opcional para la Plataforma 12

2.3. Comunicación para la transmisión de datos del dispositivo hacia la PC..... 12

2.4. Análisis de la Compatibilidad entre el sistema embebido y Simulink®. 13

3. INTERFAZ DE COMUNICACIÓN (SIMULINK®) 14

3.1. Configuración del Software Simulink para compatibilidad con ArduPilot..... 14

3.2. Comunicación entre el Software Simulink® y el dispositivo ArduPilot..... 18

3.3.	Análisis de la Librería APM Block 2.0	21
3.4.	Identificación y forma de Comunicación de los pines analógicos y digitales para la comunicación entre el sistema embebido y Simulink®	26
4.	SISTEMAS DE CONTROL AUTOMÁTICO.....	28
4.1.	Identificación del Sistema.	28
4.1.1.	Planta B del laboratorio de Control y Automatización Totalmente Integrado de la Universidad Politécnica Salesiana Sede Cuenca.	28
4.1.2.	Identificación del proceso de flujo.....	30
4.2.	Análisis de estrategias de control.....	34
4.2.1.	Controladores PID.	34
4.2.2.	Controladores en espacio de estados en tiempo discreto.	35
4.2.3.	Control predictivo basado en modelo (MPC).	40
4.2.4.	Control PID adaptativo.	42
4.3.	Diseño y Simulación de Estrategias de Control.....	44
4.3.1.	Diseño de un controlador PID.....	44
4.3.2.	Diseño del controlador por retroalimentación de estado.....	47
4.3.3.	Diseño del controlador Predictivo Basado en Modelo (MPC).....	52
4.3.4.	Diseño del controlador PID Adaptativo.....	54
5.	IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.	57
5.1.	Implementación de Controladores.	57
5.1.1.	Esquema de conexión entre la plataforma ArduPilot y proceso de flujo para la implementación de estrategias de control.....	57
5.1.2.	Implementación del Controlador PID.....	57
5.1.3.	Implementación del Controlador por Realimentación de Estados.	60
5.1.4.	Implementación del Controlador predictivo basado en modelo (MPC).....	62
5.1.5.	Implementación del Controlador PID Adaptativo.	63
5.2.	Análisis del Funcionamiento de los Controladores.....	65
5.2.1.	Comparación entre el controlador PID y MPC.	65
5.2.1.	Comparación entre el controlador en Espacio de Estados y el controlador PID Adaptativo.	65
5.3.	Beneficios del uso de la plataforma ArduPilot en sistemas de control	66
6.	CONCLUSIONES Y RECOMENDACIONES.....	67
	APÉNDICE A	69
	APÉNDICE B.....	70
	APÉNDICE C.....	72
	APÉNDICE D	73
	REFERENCIAS BIBLIOGRAFICAS	74

ÍNDICE DE FIGURAS

Figura 1	Plataforma embebida Arduino [7]	2
Figura 2	Plataforma embebida Arduino Mega 2560 [7]	2
Figura 3	Plataforma embebida BEAGLEBOARD [8]	2
Figura 4	Plataforma embebida LEGO MINDSTORMS [9]	3
Figura 5	Plataforma Embebida Raspberry PI [10]	3
Figura 6	Plataforma embebida STM32 [11]	4
Figura 7	Dispositivo ArduPilot 2.8 [2]	5
Figura 8	Placa del ArduPilot 2.8 [2]	5
Figura 9	Partes del ArduPilot 2.8 [2]	6
Figura 10	Led's indicadores de comportamiento del ArduPilot 2.8 [2]	7
Figura 11	Conexión de un receptor PPM [2]	8
Figura 12	Led's de comportamiento de los receptores PPM [2]	8
Figura 13	Puerto Telemetría del ArduPilot [2]	9
Figura 14	Definir UART2 como puerto de Telemetría [2]	9
Figura 15	Conexión del módulo Telemetría con ArduPilot [2]	10
Figura 16	Módulo GPS [2]	10
Figura 17	Conexión del Módulo GPS con el dispositivo ArduPilot [2]	10
Figura 18	Entradas/salidas analógicas [2]	11
Figura 19	Entradas Digitales [2]	11
Figura 20	Distribución de energía eléctrica en el dispositivo ArduPilot [2]	11
Figura 21	Auxiliar de alimentación 3DR Power Module [2]	12
Figura 22	Puerto del dispositivo ArduPilot	12
Figura 23	Librería APM 2.0 Simulink Blockset [3]	14
Figura 24	Instalar un nueva Plataforma Embebida.	15
Figura 25	Asistente de Paquetes	15
Figura 26	Seleccionar una Plataforma Embebida.	16
Figura 27	Verificación de Detalles previo a la instalación	16
Figura 28	Reemplazar carpetas "cores"	17
Figura 29	Archivo io_wrappers.cpp	17
Figura 30	Primera línea de código en los archivos asignados	17
Figura 31	Última línea de código en los archivos asignados	18
Figura 32	Directorio a trabajar de Matlab	18
Figura 33	Agregar al directorio de trabajo	19
Figura 34	Resultado de la ejecución	19
Figura 35	Configuración de Parámetros en Simulink	20
Figura 36	Configuración Hardware Implementation	20
Figura 37	Selección de la Plataforma a utilizar	21
Figura 38	Librería APM Blockset 2.0	21
Figura 39	Analog Input	26
Figura 40	Analog Output	27
Figura 41	Planta B del laboratorio de control y automatización totalmente integrado de la Universidad Politécnica Salesiana	28
Figura 42	Diagrama PI&D	30
Figura 43	Programa para Identificación del Sistema	30
Figura 44	Identificación del Sistema	31
Figura 45	Ventana principal del comando "ident"	31
Figura 46	Datos del sistema ingresados	32
Figura 47	Obtención de la Función de Transferencia	32
Figura 48	Configuración de polos y ceros del modelo de la función de transferencia	32
Figura 49	Proceso de obtención de la función de transferencia terminado	33
Figura 50	Aproximación del modelo de la función de transferencia y el sistema físico	33
Figura 51	Función de transferencia del sistema	33

Figura 52 Diagrama de bloques de un controlador PID [21]	34
Figura 53 Ventana de inicio del Toolbox Sisotool-Matlab®	34
Figura 54 Ventana principal del Toolbox Pidtool-Matlab®	35
Figura 55 Diagrama de bloques de un sistema de control lineal en tiempo discreto invariante en el tiempo representado en el espacio de estado [19].	36
Figura 56 Sistema de regulación con observador de estados [20].....	36
Figura 57 Sistema de control con realimentación de estado [20]].	37
Figura 58 Sistema de control con realimentación de estado observado [20].	38
Figura 59 Sistema de Seguimiento con realimentación de estado y control integral [20].	38
Figura 60 Estrategia de control Predictivo [22]	40
Figura 61 Estructura general de los Controladores Predictivos [17].....	41
Figura 62 Diagrama de un Control Adaptativo [24].	42
Figura 63 Esquema para el modelo de referencia	43
Figura 64 PID Para el proceso de flujo	44
Figura 65 Respuesta del controlador en PIDTOOL a un escalón unitario	45
Figura 66 Constantes obtenidas en PIDTOOL para el controlador PID	45
Figura 67 Constante KD.....	45
Figura 68 Constante KP	46
Figura 69 Constante KI	46
Figura 70 Simulación del controlador en Simulink.....	46
Figura 71 Respuesta de la simulación del sistema	47
Figura 72 Diseño del controlador en Espacio de Estados en tiempo discreto para el proceso de flujo	47
Figura 73 Controlador en Espacio de Estados en tiempo discreto del proceso de flujo.....	48
Figura 74 Ingreso de los datos de la Planta al usar el comando Ident.....	48
Figura 75 Estimación del modelo en espacio de estados en tiempo discreto	49
Figura 76 Matrices de Estado, Entrada, Salida y Transmisión directa.....	49
Figura 77 Matrices del espacio de estados representadas como un modelo de nombre MatricesSS	50
Figura 78 Similitud entre el Modelo Real y el Modelo Discreto en Espacio de Estados.....	50
Figura 79 Factor de Amortiguamiento ζ [24].....	51
Figura 80 Resultado de la Simulación del controlador en espacio de estados en tiempo discreto.	52
Figura 81 Matrices del sistema.....	52
Figura 82 Simulación del MPC del sistema realizado en Simulink	53
Figura 83 Math Script del controlador MPC.....	53
Figura 84 Resultado de la Simulación del controlador predictivo basado en modelo (MPC)	54
Figura 85 Diseño del controlador PID Adaptativo para el sistema.	55
Figura 86 Resultado de la Simulación del controlador PID Adaptativo.	55
Figura 87 Conexión del dispositivo APM ArduPilot 2.8 con el Panel frontal del Sistema.....	57
Figura 88 Controlador PID diseñado en Simulink para la Planta B.....	58
Figura 89 Conversión para el bloque Analog Output.....	58
Figura 90 Conversión para el bloque Analog Input	58
Figura 91 PID implementado	58
Figura 92 Comportamiento a cambios de referencia del sistema.....	59
Figura 93 Comportamiento a cambios de referencia y perturbaciones en el sistema.....	59
Figura 94 Controlador en Espacio de Estados en tiempo discreto diseñado en Simulink para la proceso de flujo.	60
Figura 95 Controlador por realimentación de estados implementado.....	60
Figura 96 Comportamiento a cambios de referencia del sistema.....	61
Figura 97 Comportamiento a cambios de referencia y perturbaciones en el sistema.....	61
Figura 98 Implementación del controlador MPC con la plataforma ArduPilot	62
Figura 99 Comportamiento del sistema con la implementación del controlador predictivo basado en modelo, con los parámetros de sintonización $N_c=2$, $N_p=3$ y $rw=50$	62

Figura 100 Comportamiento del sistema con la implementación del controlador predictivo basado en modelo, con los parámetros de sintonización $N_c=2$, $N_p=3$ y $rw=150$	63
Figura 101 Comportamiento del sistema con la implementación del controlador predictivo basado en modelo, con los parámetros de sintonización $N_c=2$, $N_p=4$ y $rw=80$	63
Figura 102 Controlador PID Adaptativo diseñado en Simulink para el proceso de flujo.	64
Figura 103 Comportamiento a cambios de referencia del sistema.....	64
Figura 104 Comparación entre un controlador PID y un MPC.....	65
Figura 105 Comparación entre un Controlador en Espacios de Estados Discreto y un Controlador PID Adaptativo.....	66

ÍNDICE DE TABLAS

Tabla 1 Posición del JP1 (Traducido al español) [2].....	6
Tabla 2 Posición del JP1 (Traducido al español) [2].....	6
Tabla 3 Estados de los Led's del dispositivo (Traducido al español) [2].....	7
Tabla 4 Comparación de Versiones compatibles con la plataforma ArduPilot.....	13
Tabla 5 Entradas y salidas analógicas	22
Tabla 6 Sensor Barométrico	22
Tabla 7 Conversor de datos	22
Tabla 8 Entradas y salidas digitales	23
Tabla 9 Flash	23
Tabla 10 GPS	23
Tabla 11 IMU	24
Tabla 12 Magnetómetro	24
Tabla 13 Pitot	24
Tabla 14 Serial	25
Tabla 15 Canales RC.....	25
Tabla 16 Sincronización.....	26
Tabla 17 Elementos del proceso de flujo [14] [15]	29

AGRADECIMIENTOS

Una etapa de mi vida ha terminado y no me queda más que agradecer a Dios por haberme brindado unos buenos padres, una buena familia y por cruzar buenas y malas personas en mi vida que a la larga fueron y serán un aprendizaje para mi vida.

Primeramente quiero agradecer a mis padres Miguel y Norma por el apoyo que siempre me brindaron a pesar de quizás no ser el mejor hijo y por haberme inculcado buenos valores y enseñarme a nunca darme por vencido. Por otro lado quiero agradecer a mi hermanita Daniela, quien siempre me ha apoyado a pesar nuestras diferencias personales. A pesar que parezca un chiste y absurdo, quiero agradecer a mi gato Pepelepu por acompañarme en mis largas noches de estudio.

Quiero agradecer a mi enamorada Salomé por apoyarme siempre, por creer en mí, por soportarme y amarme a pesar de ser tan complicado, por ser mi orgullo y especialmente por enseñarme a ser una mejor persona día a día.

Quiero agradecer a mis primos Miguel Ángel y Gabriel por ser mis compañeros de locuras y por ser como mis hermanos.

Quiero agradecer a un amigo muy especial Juan Pablo por ser un compañero de estudios y un gran amigo en mi vida.

Quiero agradecer a Mauricio, quien es el encargado de los laboratorios en el área de mecánica, por ser una persona humilde y ayudarnos con nuestro proyecto a pesar de ser ajeno al mismo.

Quiero agradecer al Ing. Julio Zambrano quien fue el tutor de este proyecto y brindó su ayuda desinteresada para la culminación del mismo.

Quiero agradecer a mis compañeros de la Universidad y a los profesores del área de control Iván Escandón, Marco Carpio y Walter Orozco quienes brindaron su ayuda desinteresada. Finalmente quiero agradecer a mi compañero de tesis Christian por haberme brindado la confianza y hacer un buen equipo de trabajo.

Luis Miguel Aguilar Castro

DEDICATORIA

Quiero dedicar este proyecto a Dios quien ha estado siempre conmigo y me ha ayudado a no desfallecer en ningún momento a pesar de los problemas personales y situaciones que la vida presenta día a día. Quiero agradecer a Dios por hacer que cada día en mi vida sea una nueva experiencia y espero me siga ayudando día a día en las nuevas metas que me he propuesto y en los retos que se presentaran de ahora en adelante

Luis Miguel Aguilar Castro.

AGRADECIMIENTOS

Primero agradezco a Dios por el amor entregado a mi familia, gracias a Él he sido capaz de culminar de manera exitosa una de mis metas.

Agradezco a mis padres, Hermel e Isabel por todo el apoyo incondicional que me han brindado en el transcurso de mi formación profesional.

A mis hermanas, Tatiana y Maritza que me han brindado su amor y todo el apoyo durante esta etapa de mi vida.

Agradezco al encargado de laboratorio de mecánica el Ing. Mauricio Ortiz por la ayuda entregada durante la elaboración de este proyecto.

Agradezco a mi tutor de tesis Ing. Julio Zambrano por la ayuda y el tiempo brindado en el proceso de elaboración de este proyecto de titulación.

Agradezco también a los docentes de la Universidad Politécnica Salesiana Ing. Iván Escandón y Phd. Walter Orozco por facilitarnos material de apoyo para lograr la implementación de este proyecto.

Finalmente agradezco a mis amigos, de manera muy especial a Luis Aguilar por su apoyo y su dedicación en el transcurso de esta tesis.

Christian Ortiz Astudillo.

DEDICATORIA

A Dios por brindarme la salud para lograr cumplir una meta más en mi vida. A mis padres quienes con su apoyo, sus consejos, su ejemplo y su amor han logrado que mi formación profesional y moral crezca. A mis hermanas quienes siempre me han brindado su apoyo para lograr cumplir cada una de mis metas.

Christian Ortiz Astudillo.

ANTECEDENTES.

Simulink® es un software de programación mediante bloques, lo cual facilita la programación de algoritmos muy complejos en comparación con la programación tradicional que es de línea a línea. Una de las grandes aplicaciones de este software es el desarrollo de esquemas de control avanzado [1].

El dispositivo Ardupilot está basado en la plataforma Arduino Mega [2], el cual tiene la posibilidad de manejar programas realizados en Simulink® de Matlab, esto es posible mediante el bloque de Simulink® AMP Block2.0 [3].

Esta librería se encuentra disponible en Mathworks, su única versión ha sido creada en el 2012, y funciona con la tarjeta ATMega 2560, como lo es el Ardupilot 2.8 [3]. La ventaja del dispositivo Ardupilot es su costo relativamente bajo en comparación con otras plataformas embebidas, al tener el dispositivo accesibilidad de comunicación mediante software con una PC y Simulink®, es posible realizar sistemas de control sobre procesos reales [3].

INTRODUCCIÓN

En este proyecto se realizará un análisis sobre la tecnología ArduPilot, principalmente se analizará la factibilidad de esta plataforma para desarrollar y ejecutar esquemas de control avanzado como es el caso del control por retroalimentación de estados, control adaptativo y el control predictivo basado en modelo.

Con el fin de analizar la versatilidad de la plataforma ArduPilot se implementarán los esquemas de control antes mencionados para controlar el flujo de ingreso a un depósito. Este sistema cuenta con posicionador electroneumático burkert 8792 [4] como elemento de actuación, mientras que la señal de retroalimentación será proporcionada por un transmisor de flujo burkert 8035 [5].

La intencionalidad de este proyecto radica en evaluar esquemas de control sobre una plataforma embebida de bajo costo. La implementación de estos controladores involucran cálculos complejos los mismos que hasta la actualidad en su mayoría han sido desarrollados en computadores personales debido a su alto coste computacional.

En primera instancia se hará una investigación sobre la plataforma Ardupilot para conocer su estructura de hardware y modos de configuración y programación. Posterior a esto se analizará la compatibilidad de las operaciones matemáticas que involucra el desarrollo de los cuatro esquemas de control con la plataforma. A continuación previa a la implementación de los controladores se hará una identificación del proceso de manera que los sistemas de control puedan ser simulados.

CAPÍTULO 1

1. INTRODUCCIÓN

1.1. Presentación

En la actualidad los esquemas de control para la industria generalmente utilizan módulos de control de costos elevados y que son específicos para cada una de las plantas o sistemas. El poder reemplazar estos módulos con plataformas embebidas que cumplan su objetivo de manera eficiente podría ser ventajoso pues los recursos económicos que se invertirían serían mucho menores.

Existen diferentes tipos de plataformas embebidas que trabajan con el software Matlab®-Simulink, en donde el diseño de los controladores es realizado y descargado a dicha plataforma a través de algún sistema de comunicaciones. Cabe recalcar que estas plataformas trabajan en tiempo real por lo cual su funcionamiento puede mostrar mejor desempeño que un controlador que se encuentre operando bajo un ordenador con sistema operativo estándar.

Actualmente existen herramientas informáticas que permiten interactuar con estas plataformas y brindan una facilidad de programación utilizando lenguajes de alto nivel. Una de las herramientas informáticas que presta mayor versatilidad para este tipo de interacción es Simulink de Matlab®.

El software Matlab® ha generado versiones desde 2012 que permiten la comunicación con plataformas embebidas, las mismas que tienen soporte con Simulink con lo cual se facilita la programación. Algunas de las plataformas que soporta el software Matlab® son [6]:

- Arduino [7]
- Beagleboard [8]
- Lego Mindstorms [9]
- Raspberry PI [10]
- STM32F4 Discovery [11]

1.2. Análisis del Estado del Arte de Plataformas de Control.

Debido a que se pretende utilizar el lenguaje de programación gráfico que brinda Simulink a continuación se detalla algunas características de las plataformas que tienen compatibilidad con esta herramienta de software.

1.2.1. ARDUINO

Las versiones de Arduino compatibles con el software Matlab®-Simulink son Arduino uno (véase Figura 1) y Arduino Mega 2560 (véase Figura 2). Arduino Uno se encuentra basado en un procesador ATmega328P contando con 14 pines de entrada y salida digital entre los cuales 6 de ellos pueden ser utilizados como PWM, cuenta con 6 entradas analógicas, dispone de un reloj de cristal de 16 MHz [7].

Arduino Mega 2560 cuenta con un procesador ATmega2560, dispone de 54 pines de entrada y salida digital de los cuales 15 pueden ser utilizados para aplicaciones PWM, dispone de 16 entradas analógicas, 4 UART'S, utiliza un oscilador de 16MHz [7].



Figura 1 Plataforma embebida Arduino [7]



Figura 2 Plataforma embebida Arduino Mega 2560 [7]

Estos dos procesadores ATmega328P y ATmega2560 son soportados por el software de Matlab®-Simulink.

1.2.2. BEAGLEBOARD

Es una plataforma de bajo costo (véase Figura 3) diseñada específicamente para el procesamiento de señales, esta plataforma cuenta con procesador ARM Cortex-A8 de 1 GHz de TI, dispone audio estéreo y conectividad de vídeo digital, y es compatible con USB, RS-232 y Ethernet. Esta plataforma es compatible con el software Simulink® [8].



Figura 3 Plataforma embebida BEAGLEBOARD [8]

CAPITULO 1. INTRODUCCIÓN

Simulink® Support Package for BeagleBoard Hardware permite crear y ejecutar modelos de Simulink® en hardware BeagleBoard. El paquete de soporte incluye una biblioteca de bloques de Simulink para configurar y acceder a los periféricos de E / S e interfaces de comunicación. También le permite controlar de forma interactiva y ajustar algoritmos desarrollado en Simulink que circulen por BeagleBoard [8].

1.2.3. LEGO MINDSTORMS

LEGO MINDSTORMS NXT (véase Figura 4) es una plataforma robusta, de muy bajo costo y flexible para el desarrollo de aplicaciones robóticas. LEGO MINDSTORMS NXT cuenta con un microcontrolador ARM de 32 bits (256 KB de flash y 64 MB de RAM), proporciona puertos para la conexión de algunos tipos de sensores, como ultrasonido, color / luz, y tacto, y es compatible con USB y Bluetooth. Para la comunicación entre la plataforma y el software Matlab®-Simulink se requiere instalar Simulink® Support Package for LEGO® MINDSTORMS® NXT [9].



Figura 4 Plataforma embebida LEGO MINDSTORMS [9]

1.2.4. RASPBERRY PI

Raspberry Pi (véase Figura 5) es una plataforma embebida de bajo costo que soporta el sistema operativo Linux (Raspbian). Raspberry Pi dispone de un procesador ARM ® Cortex® y proporciona conectividad de periféricos de audio estéreo, vídeo digital (1080p), USB, Ethernet y dispone de la tarjeta de la cámara opcional y placa de sensores complementarios [10].

Esta plataforma embebida tiene la capacidad de interactuar con el software Matlab®-Simulink, al igual que las plataformas embebidas mencionadas anteriormente el Raspberry PI dispone de una librería que cuenta con bloques para manejar de maneras más fácil las entradas y salidas digitales o analógicas. Luego de crear el modelo en Simulink, es posible simular y descargar el algoritmo sobre la plataforma Raspberry PI. Esta plataforma ha sido mejorada para hacer posible que sea compatible con la última versión del software Matlab® (r2015b) [10].



Figura 5 Plataforma Embebida Raspberry PI [10]

1.2.5. STM32F4 DISCOVERY

MATLAB Coder™ y Simulink Coder™ generan código C y C++ que puede ser compilado y ejecutado en la plataforma STM32F4-Discovery (véase Figura 6). Esta plataforma dispone de un núcleo de 168 MHz ARM Cortex-M, también cuenta con una memoria flash de 1024 KB y una memoria RAM de 192KB. Adicionalmente la plataforma STM32F4 DISCOVERY dispone de un puerto serial, USB, Ethernet, CAN y I2C [11].



Figura 6 Plataforma embebida STM32 [11]

1.3. Alcance y Objetivos de esta tesis

Esta tesis tiene como alcance realizar un análisis de la tecnología ArduPilot orientada a conocer la disponibilidad de esta plataforma para implementar esquemas de control clásico y moderno, en aras de disponer de un recurso tecnológico para el control de procesos disponibles en el laboratorio de control de la Universidad Politécnica Salesiana.

Con esta tesis se pretende lograr los siguientes objetivos:

- Conocer y analizar la plataforma ArduPilot.
- Analizar el alcance de aplicación del instrumento utilizando Simulink®.
- Desarrollar una plataforma de comunicación entre el dispositivo y Simulink®.
- Desarrollar aplicaciones en procesos de control reales para la evaluación de la plataforma.
- Realizar estrategias de control como PID, espacio de estados, control PID Adaptativo y control predictivo basado en modelo, evaluando de esta manera la capacidad de la plataforma para implementar este tipo de sistemas de control.

CAPÍTULO 2

2. ANÁLISIS DE LA PLATAFORMA ARDUPILOT

2.1. Análisis de los componentes del sistema embebido.

2.1.1. ARDUPILOT

El ArduPilot (véase Figura 7) es una plataforma embebida diseñada para el control de dispositivos aéreos. Este dispositivo pertenece a la familia de Arduino, sin embargo su capacidad de poder descargar programas realizados en el software Matlab-Simulink en su memoria lo hace novedoso. La descarga en el dispositivo se logra gracias a la librería de Simulink APM, con el uso del bloque APM Block2.0. [2].

Las principales aplicaciones de la plataforma debido a su diseño compacto (véase Figura 8) están ligadas al control de helicópteros y Drones. La ventaja que posee el dispositivo es la comunicación entre Software y Hardware, al ser una plataforma de costo relativamente bajo, es posible reemplazar módulos de control costosos, ya que en el dispositivo se pueden realizar controladores modernos y aplicarlos a procesos reales [2].

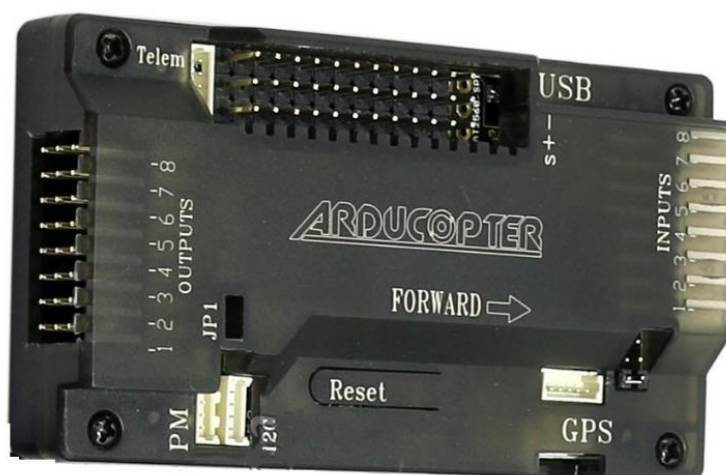


Figura 7 Dispositivo ArduPilot 2.8 [2]

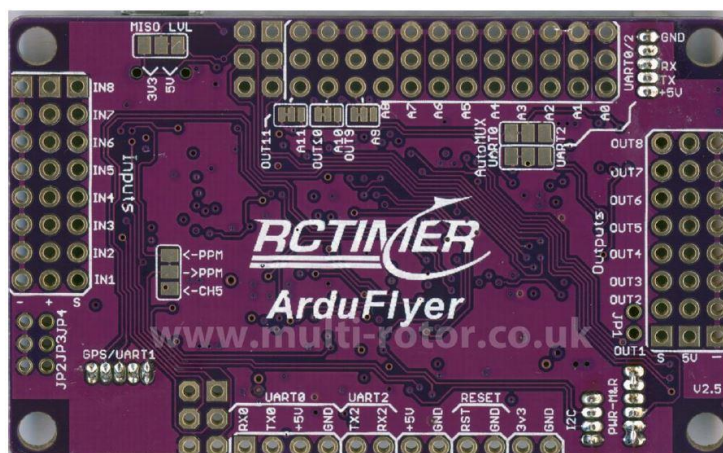


Figura 8 Placa del ArduPilot 2.8 [2]

2.1.2. PARTES DEL ARDUPILOT

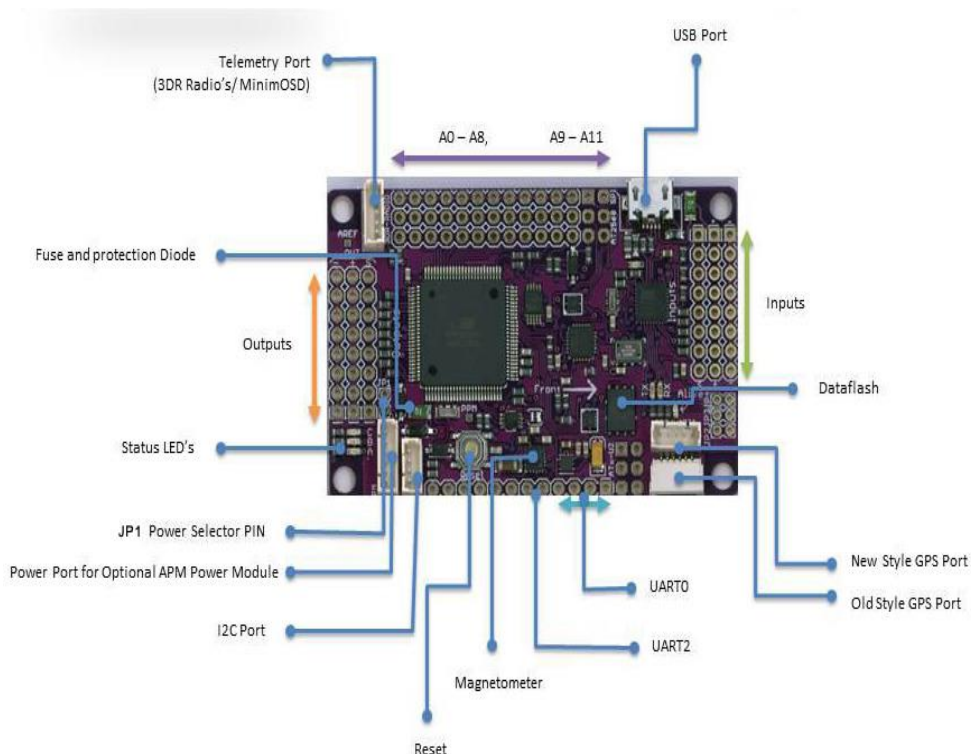


Figura 9 Partes del ArduPilot 2.8 [2]

La plataforma Ardupilot 2.8 dispone de múltiples periféricos de comunicación (véase Figura 9) como entradas/salidas analógicas, entradas/salidas digitales, puerto I2C, GPS, magnetómetro, puerto USB, 3DR Radio. Además dispone de una protección a cortocircuito mediante un fusible y un selector de potencia (JP1), mediante esta medida de protección se permite que todos los bordes dispongan de voltaje o que simplemente el voltaje se encuentre en las salidas del dispositivo. Finalmente como todo dispositivo electrónico dispone de un reset [2].

Opciones de Energía(Voltaje)	Nominal	Max Abs	Estado JP1
Voltaje en la salida del conector PWM	5.37V+-0.5	6V	JP1 conectado
Voltaje en la salida del conector PWM	5.00V+-0.25	5.5V	JP1 conectado

Tabla 1 Posición del JP1 (Traducido al español) [2]

Opciones de Energía(Voltaje)	Nominal	Max Abs	Estado JP1
Voltaje en la salida del conector PWM	5.00V+-0.5	6V	JP1 abierto
Voltaje en la salida del conector PWM	5.00V+-0.25	5.5V	JP1 abierto

Tabla 2 Posición del JP1 (Traducido al español) [2]

En la Tabla 1 y Tabla 2 se encuentran los voltajes manejados del dispositivo según la posición en la que se encuentre el Jumper 1.

2.1.3. CARACTERÍSTICAS DEL ARDUPILOT

De acuerdo a información presentada en [2], a continuación se detallan características de esta plataforma.

- Es Compatible con arduino.
- Giroscopio de 3 ejes, acelerómetro, magnetómetro y barómetro de alto rendimiento.
- Chip de 4 MP Dataflash para el registro automático de datos.
- Brújula digital mediante el chip HMC5883L-TR de Honeywell.
- Sensor de presión barométrica MS5611-01BA03.
- Chip Atmega2560 y ATMEGA32U2 para funciones de procesamiento y USB.
- Indicador led para la fuente de alimentación e indicadores LED A, B y C (véase Figura 10), de colores verde, amarillo y rojo, respectivamente.

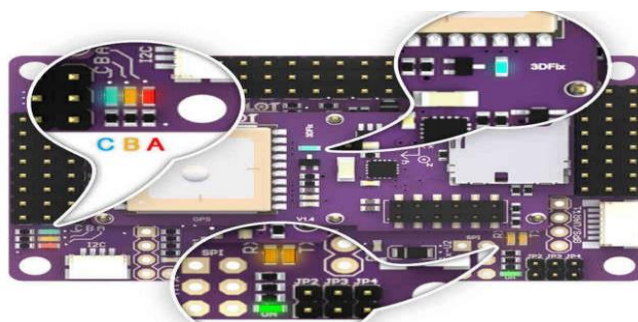


Figura 10 Led's indicadores de comportamiento del ArduPilot 2.8 [2]

LEDs	Comportamiento
Energía	Cuando se alimenta el dispositivo
TX	Visualiza los datos recibidos en el puerto USB/FTDI
RX	Visualiza los datos enviados en el puerto USB/FTDI
A C	Intermitencia del led más rápida mientras se calibra el giroscopio, entonces pausa, entonces rotación intermitente (Alrededor de un total de 45 segundos)
B	Intermitencia = Calibrando. Encendido Fijo= Listo para volar sin el bloqueo del GPS
C	GPS bloqueado – C Intermitente mientras se espera el bloqueo del GPS, Fijo con el GPS bloqueado
B C	Fijos B y C = listo para volar con el GPS bloqueado

Tabla 3 Estados de los Led's del dispositivo (Traducido al español) [2]

CAPITULO 2. ANÁLISIS DE LA PLATAFORMA ARDUPILOT

El estado de funcionamiento del dispositivo puede interpretarse mediante la Tabla 3, ya que esta indica el comportamiento de los indicadores luminosos o Led's [2].

- Consumo de corriente bajo, generalmente entre 200mA.
- Compatibilidad con el sensor ultrasónico XL MaxSonar EZO [12], el cual posee características superiores que el sensor ultrasónico HC-SR04 y con el que de hecho el dispositivo no es compatible.

2.1.4. CONEXIÓN DE UN RECEPTOR PPM

Los receptores PPM son un tipo de modulación capaz de codificar un dato de N bits mediante la transmisión de un solo pulso. La entrada de un receptor PPM (véase Figura 11) en el dispositivo Ardupilot 2.8 se logra al realizar un puente en las entradas RCA 2 y 3, quedando el pin 1 como el receptor de señal [2].

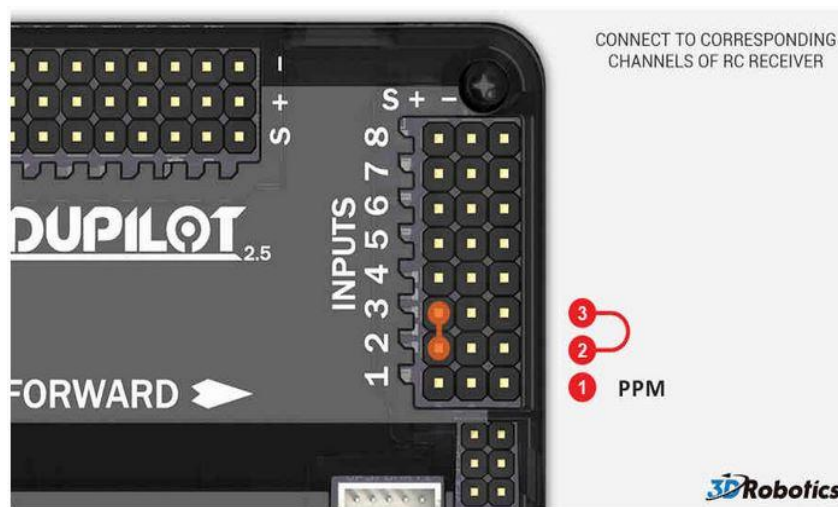


Figura 11 Conexión de un receptor PPM [2]

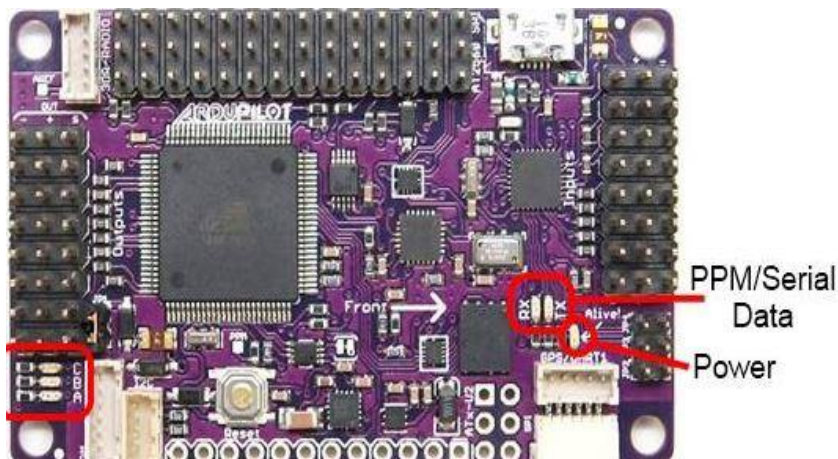


Figura 12 Led's de comportamiento de los receptores PPM [2]

Los Led's de comportamiento (véase Figura 12) para el PPM indican la transmisión (Tx) y recepción (Rx) de datos, además también existe un led que indica si la comunicación está en proceso. Con la ayuda de los Led's Tx y Rx, se podrá conocer si la información se está enviando o recibiendo según sea el caso requerido.

2.1.5. TELEMETRIA.

Es una tecnología que permite la medición remota de magnitudes físicas y el posterior envío de la información hacia el operador. La transmisión de información es realizada de manera inalámbrica, pero también se lo puede realizar mediante teléfono, redes de ordenadores, enlace de fibra óptica, entre otros. En los sistemas de telemetría las instrucciones son enviadas desde el sistema de control que se encuentre utilizando [2].



Figura 13 Puerto Telemetría del ArduPilot [2]

El puerto telemetría (véase Figura 13) se usa con el módulo Xbee para permitir la transmisión de datos a distancias, una de sus principales aplicaciones es en dispositivos aéreos mediante un determinado software, como por ejemplo Misión Planner el cual permitirá conocer la ubicación del dispositivo [2].

Si se utiliza el módulo Xbee Pro 900, debe ser acoplado con la antena tipo alambre al ArduPilot (véase Figura 15). El alcance de los módulos Xbee es aproximadamente de 2km esto depende de la antena utilizada [2].

En la Figura 15 se puede apreciar la conexión del módulo Xbee, que está conectado en el puerto telemetría del ArduPilot, el cual por defecto está conectado en el UART0 (Transmisor-Receptor Asíncrono Universal), es decir en el Serial de Arduino o Serial 0. Se puede usar el UART2 (véase Figura 14) para la comunicación por medio del Serial 3 al desoldar unas uniones en la placa del dispositivo ArduPilot. Mediante el UART0 y el puerto Serial 0 se usa el Puerto USB para la comunicación y alimentación del dispositivo mediante una computadora. Mediante cuatro cables se alimenta el módulo Xbee y mediante otros dos se adquiere la señal para procesos futuros [2].

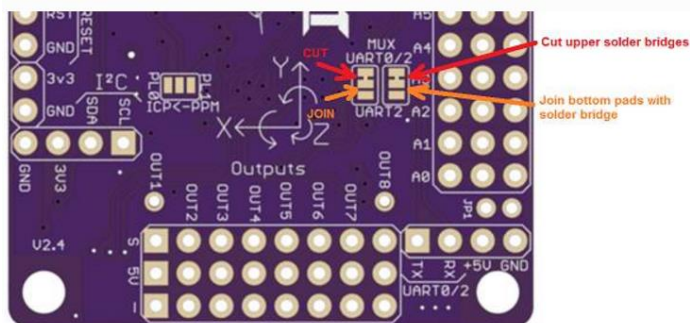


Figura 14 Definir UART2 como puerto de Telemetría [2]

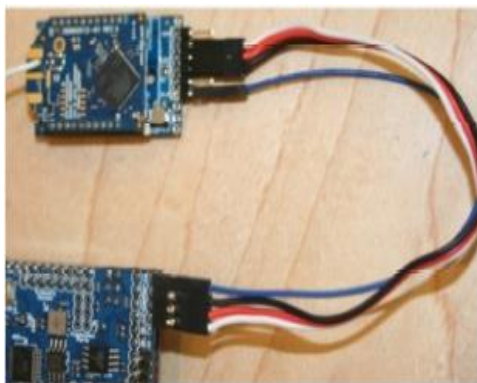


Figura 15 Conexión del módulo Telemetría con ArduPilot [2]

2.1.6. MÓDULO GPS



Figura 16 Módulo GPS [2]

El modulo GPS (véase Figura 16), dispone de dos conectores MAG y GPS, la conexión entre el modulo y la plataforma (véase Figura 17) se realiza mediante un cable dividido en dos conectores, el primer conector se asociará al puerto I2C usando 4 cables (alimentación y señales adquiridas), mientras que el segundo conector mediante 6 cables deberá asociarse al llamado puerto GPS en el dispositivo [2].

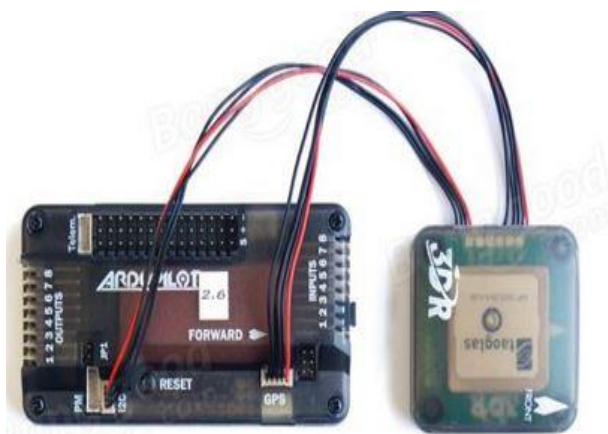


Figura 17 Conexión del Módulo GPS con el dispositivo ArduPilot [2]

2.2. Análisis de los pines y la alimentación del dispositivo

2.2.1. Pines Analógicos y Digitales.

Las entradas/salidas analógicas (véase Figura 18) se denominan desde A0 hasta A13, de igual manera serán colocadas las entradas/salidas analógicas en el software de programación de la plataforma Ardupilot 2.8 [2].

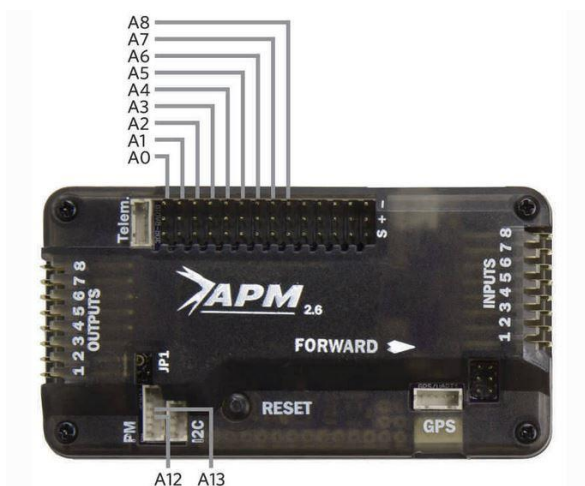


Figura 18 Entradas/salidas analógicas [2]

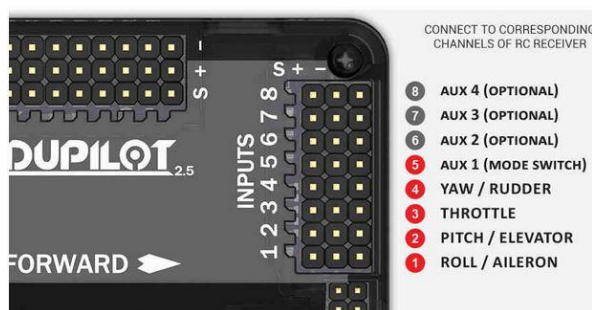


Figura 19 Entradas Digitales [2]

Las entradas digitales (véase Figura 19) del dispositivo son ocho, estas entradas digitales pueden ser manejadas desde Simulink utilizando los bloques RC de la librería del ArduPilot [2].

También la plataforma Ardupilot 2.8 utiliza el mismo conjunto de 9 pines de entrada analógicas como pines digitales de salida (Pin 54 a 62). Para utilizar las entradas analógicas como salidas digitales, es necesario agregar el número “54” al pin analógico. Por ejemplo Pin 54 es A0 [2].

2.2.2. POWER-RAIL

En este dispositivo se dispone de tres conectores dispuestos en 3 carriles o filas (véase Figura 20). Estos carriles están colocados en las entradas analógicas, en las salidas digitales y en las entradas digitales, la primera fila es la señal del ArduPilot, la del centro es la señal positiva en donde se obtiene 5 voltios y la última es tierra. Se debe conocer que para proteger el dispositivo se dispone de un fusible de 500 mA y de un diodo de protección [2].

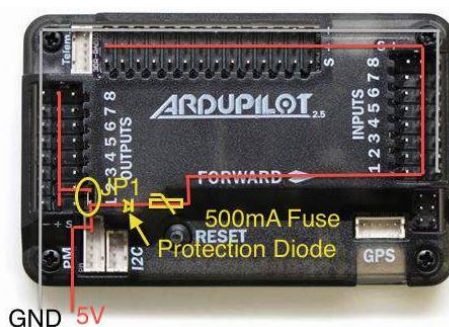


Figura 20 Distribución de energía eléctrica en el dispositivo ArduPilot [2]

2.2.3. Puerto de Alimentación opcional para la Plataforma

Mediante el dispositivo 3DR Power Module (PM) (véase Figura 21), se obtiene una alimentación fija de 5.37V y 2.25A, al conectar una batería que tenga un voltaje máximo de 18V y una corriente máxima de 90 A, el dispositivo mencionado nos permite la monitorización del voltaje y la corriente en la batería [2].

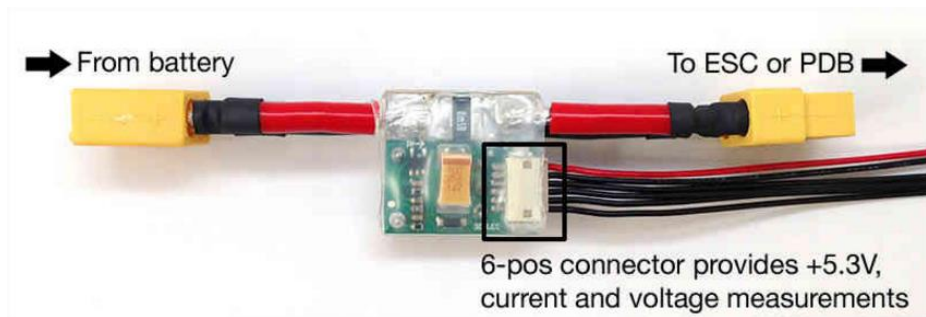


Figura 21 Auxiliar de alimentación 3DR Power Module [2]

2.3. Comunicación para la transmisión de datos del dispositivo hacia la PC

Para establecer la comunicación entre el sistema embebido y el ordenador, se utiliza el puerto serial UART0 mediante un conector Micro USB. Mediante este puerto serial se envía y adquiere los datos, trabajando a una velocidad de 115200 baud. Este puerto se encuentra al lado lateral de la plataforma ArduPilot [3].

Una vez que se conecte la plataforma al ordenador es necesario conocer el “COM” en el que se estableció la conexión, para conocer el puerto se debe proceder a “dispositivos e impresoras” ubicado en el “panel de control” del ordenador y verificar el puerto en el que se ha enlazado la plataforma. Esta plataforma es reconocida como Arduino Mega 2560 debido a que dispone del procesador ATmega2560 [3].

Es importante conocer cuál es el “COM” ya que este deberá ser configurado en Simulink® para proceder a descargar el programa sobre la plataforma de manera satisfactoria [3].

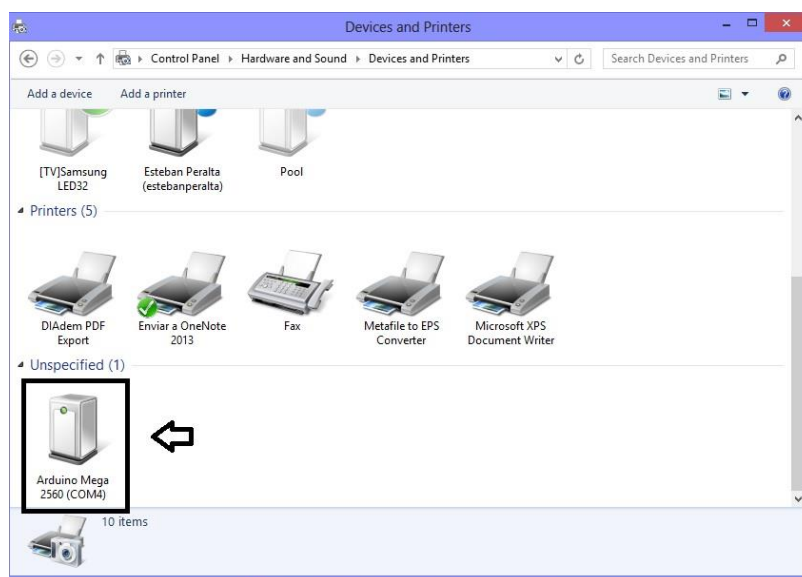


Figura 22 Puerto del dispositivo ArduPilot

Como se observa en la Figura 22 se encuentra marcado el dispositivo cuando ya ha sido reconocido por el ordenador y brinda un puerto, para nuestro caso será “COM4”. Esto debe ser configurado en el software Simulink® antes de proceder a descargar el diseño realizado sobre la plataforma embebida [3].

2.4. Análisis de la Compatibilidad entre el sistema embebido y Simulink®.

Como puede observarse en la Tabla 4 el sistema embebido ArduPilot no es compatible con las últimas versiones de Matlab®, esto se debe a que la librería “APM 2.0 Simulink Blockset” fue creada para la versión 2012 y la misma no ha sido actualizada. Debido a esto cuando se instala esta librería en las últimas versiones de Matlab®, esta no genera el package que dispone de los bloques de cada periférico con el que cuenta el sistema embebido [3].

Versión de Matlab®	Librería a Implementar	Observación
r2012a	APM 2.0 Simulink Blockset	Funcionando Correctamente. Genera un nuevo package dentro de la librería de Simulink®, la cual contiene bloques que conectan directamente con la plataforma embebida.
r2012b	APM 2.0 Simulink Blockset	
r2013a	APM 2.0 Simulink Blockset	
r2013b	APM 2.0 Simulink Blockset	No se genera el package dentro la librería de Simulink®, por lo tanto no es posible utilizar los periféricos con los que dispone la plataforma ArduPilot.
r2014a	APM 2.0 Simulink Blockset	
r2014b	APM 2.0 Simulink Blockset	
r2015a	APM 2.0 Simulink Blockset	
r2015b	APM 2.0 Simulink Blockset	

Tabla 4 Comparación de Versiones compatibles con la plataforma ArduPilot.

Cabe recalcar que esta librería ha logrado funcionar de manera correcta en la versión de MatLab® 2013^a, ya que esta no muestra una significativa diferencia con la versión de MatLab® 2012^a [3]. A pesar que el sistema embebido cuenta con un procesador ATmega 2560, esta librería no funciona con el arduino ya que se ha diseñado específicamente para el uso de la plataforma ArduPilot, la cual cuenta con diferentes periféricos que el Arduino Mega [3].

CAPÍTULO 3

3. INTERFAZ DE COMUNICACIÓN (SIMULINK®)

3.1. Configuración del Software Simulink para compatibilidad con ArduPilot.

Simulink® es un software de programación mediante bloques, lo cual facilita la programación de algoritmos muy complejos en comparación con la programación tradicional que es de línea a línea. Una de las grandes aplicaciones de este software es en el control avanzado [3].

La plataforma ArduPilot 2.8 tiene la capacidad de poder conectarse con el software Simulink® mediante la librería APM 2.0 Simulink Blockset, esta librería se encuentra disponible en Mathworks, su única versión ha sido creada en el 2012, y funciona con la tarjeta ATmega 2560 embebida en la plataforma Ardupilot 2.8 [3].

Esta librería puede ser descargada en una carpeta comprimida desde la siguiente página: <http://www.mathworks.com/matlabcentral/fileexchange/39037-apm2-simulink-blockset>, dentro de esta carpeta se encontrará también sus instrucciones para su instalación y además contiene un ejemplo de un modelo en Simulink® [3].

Siendo capaz de comunicar programación en el software Simulink® y la plataforma ArduPilot 2.8 se puede implementar control avanzado en diferentes plantas reales, con la ventaja de que la plataforma ArduPilot tiene un costo mucho menor que un módulo de control comercial [3].



Figura 23 Librería APM 2.0 Simulink Blockset [3]

En la Figura 23 se puede observar la página de donde se puede obtener la librería APM2 Simulink Blockset, la misma que sirve para realizar una comunicación con la tarjeta ATmega 2560, de la cual dispone ArduPilot.

Enlace de la Plataforma Embebida ArduPilot con Simulink de Matlab® 2012ª.

Para lograr enlazar Simulink de Matlab® 2012a con la plataforma ArduPilot se requiere haber descargado los archivos [13]:

- Librería APM 2.0 APM Simulink® Blockset.
- Compilador .mex compatible con Matlab® 2012a.

Pasos a Seguir para el enlace:

1. Abrir el software Matlab® 2012a como administrador [13].

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

2. Abrir un nuevo modelo de Simulink® y luego dirigirse a “TOOLS>RUN_ON_TARGET_HARDWARE>INSTALL/UPDATE+SUPPORT_PACKAGE”, como se observa en la Figura 24 [13].

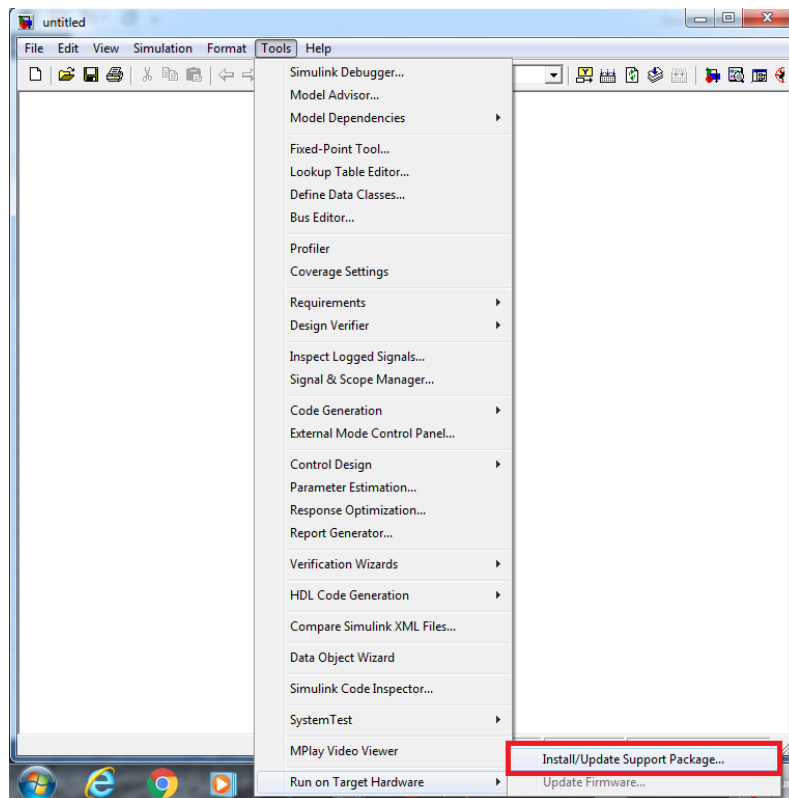


Figura 24 Instalar un nueva Plataforma Embebida.

3. En la ventana de dialogo Target Installer, se ha abierto el asistente de paquetes, seleccionar la opción Internet (recommended), como se observa en la Figura 25 [13].

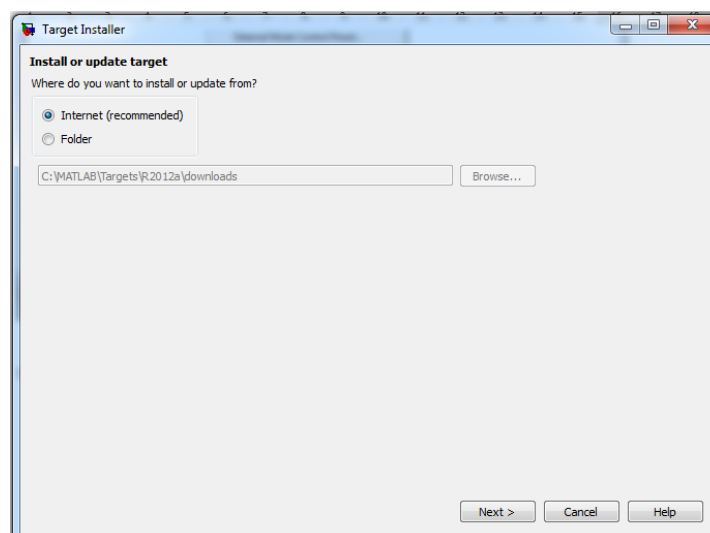


Figura 25 Asistente de Paquetes.

4. Hacer clic en Next, a lo cual se abrirá una nueva ventana de dialogo en la que se debe seleccionar la plataforma a utilizar, en este caso seleccionar la tarjeta Target Arduino Mega 2560 como se observa en la Figura 26 [13].

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

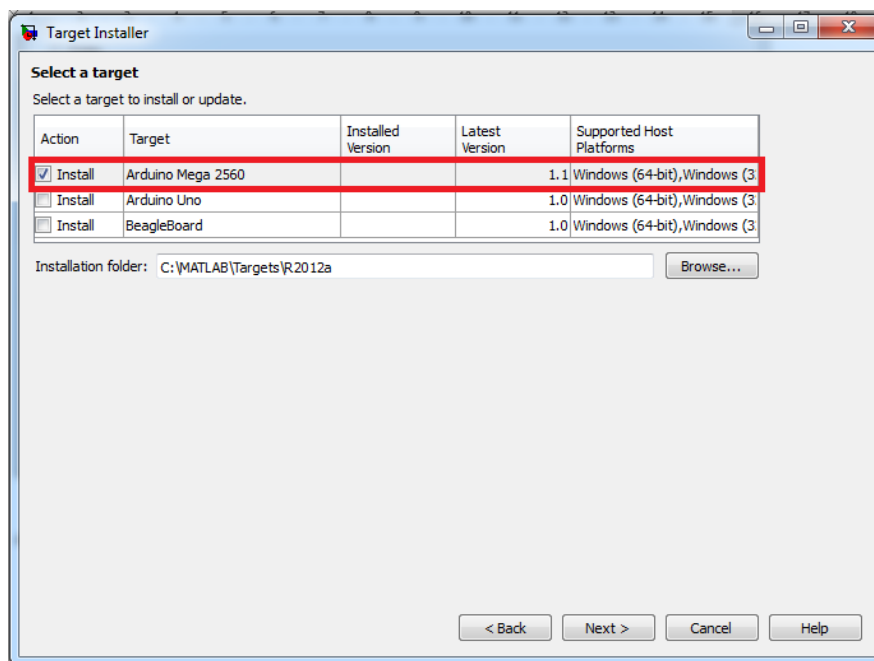


Figura 26 Seleccionar una Plataforma Embebida.

5. A continuación se abrirá una ventana de diálogo para verificar los detalles antes de comenzar la instalación, para iniciar la instalación se debe dar clic en Install como se muestra en la Figura 27 [13].

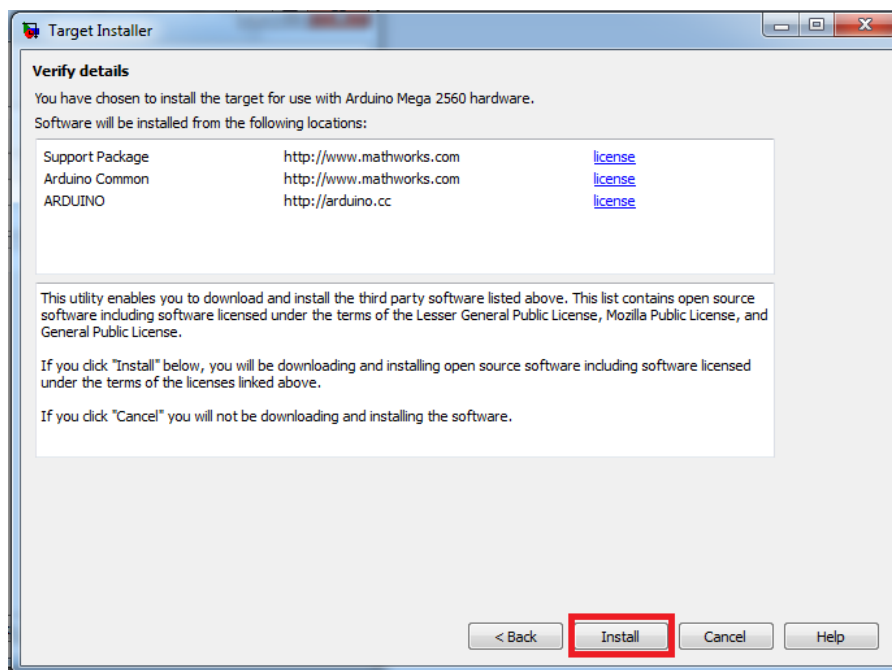


Figura 27 Verificación de Detalles previo a la instalación.

6. Hacer clic en Finish.
7. Posteriormente se debe modificar el directorio en el que se encuentran ubicados los núcleos (cores) de arduino, para ello dirigirse a (C:\MATLAB\Targets\R2012a-arduino-1.0\hardware\arduino\cores /arduino). A continuación se debe reemplazar la carpeta "cores" por la que se encuentra dentro del archivo .Zip descargado previamente. Esto se puede observar en la Figura 28 [13].

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

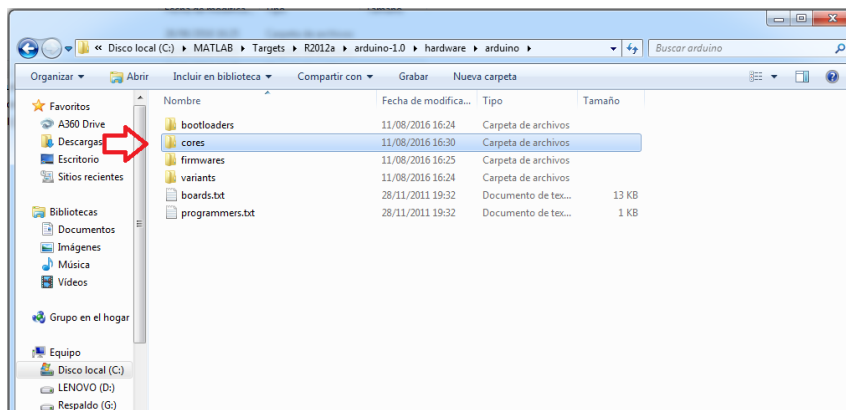


Figura 28 Reemplazar carpetas "cores"

8. Posterior se debe modificar los archivos IOWrappers.cpp (véase Figura 29), para ello se debe abrir los archivos ubicados en la dirección C:/MATLAB/Targets/R2012a/arduino/include/io_wrappers.h y los archivos ubicados en C:/MATLAB/Targets/R2012b/arduino/src/io_wrappers.cpp. Se debe modificar el código (véase Figura 30 y Figura 31) agregando `#if 0` como primera línea de código y `#endif` como última línea de código, finalmente se debe guardar y sobrescribir los archivos [13].

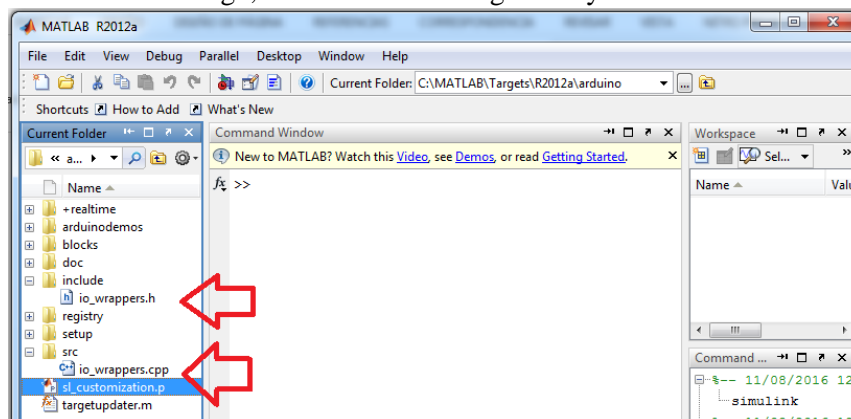


Figura 29 Archivo io_wrappers.cpp

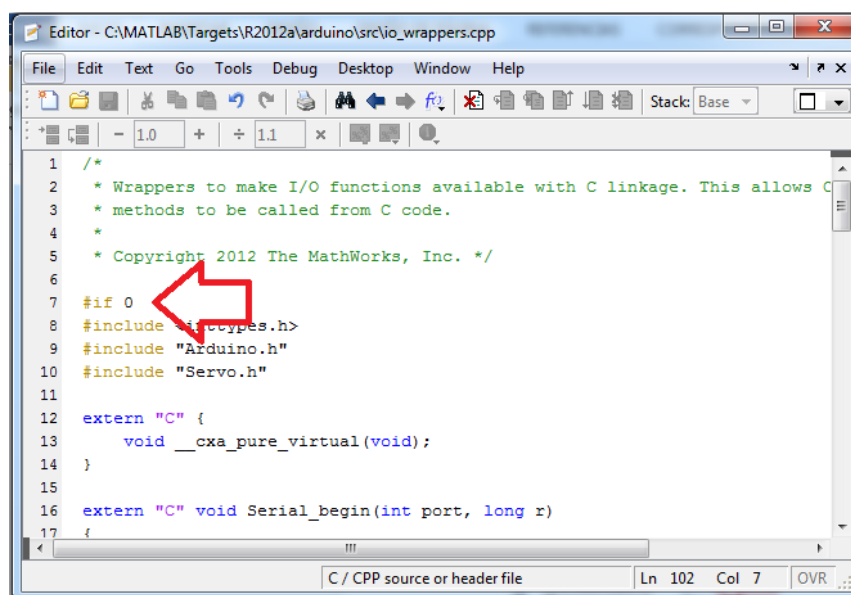


Figura 30 Primera línea de código en los archivos asignados.

```

87
88 extern "C" void MW_servoAttach(uint8_t nServo, uint8_t pinNumber)
89 {
90     myservos[nServo].attach(pinNumber);
91 }
92
93 extern "C" void MW_servoWrite(uint8_t nServo, uint8_t inValue)
94 {
95     myservos[nServo].write(inValue);
96 }
97
98 extern "C" int MW_servoRead(uint8_t nServo)
99 {
100     return myservos[nServo].read();
101 }
102 #endif
103 #endif
    
```

Figura 31 Última línea de código en los archivos asignados

3.2. Comunicación entre el Software Simulink® y el dispositivo ArduPilot

Para tener una comunicación entre el software Simulink y la plataforma embebida ArduPilot, se debe seguir los siguientes pasos:

1. Se debe crear un nuevo directorio para el proyecto (véase Figura 32), la misma que deberá contener los archivos “arduino” y “blocks”, estos se encontraran en el ZIP que se descargó anteriormente.
 - El primer archivo se mostrará como “ArduinobuildMexFiles.m”
 - El segundo archivo se mostrará como “IMU_Serial_Example.slx”

Solo para comprobar que la librería se instaló correctamente y que compila de manera correcta., esta dirección debe ser establecida como el directorio a trabajar en Matlab® [13].

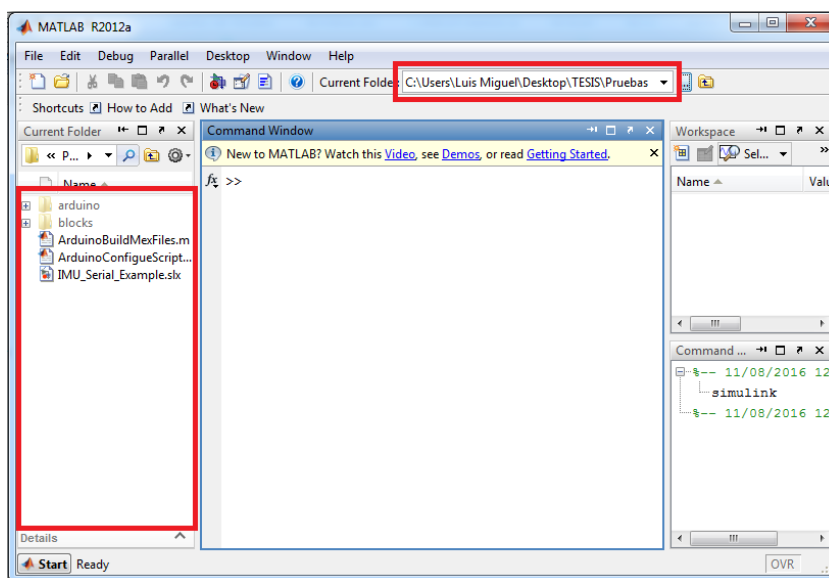


Figura 32 Directorio a trabajar de Matlab.

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

2. Una vez generada la dirección se debe añadir las carpetas y sub carpetas, esto se indica en la Figura 33 [13].

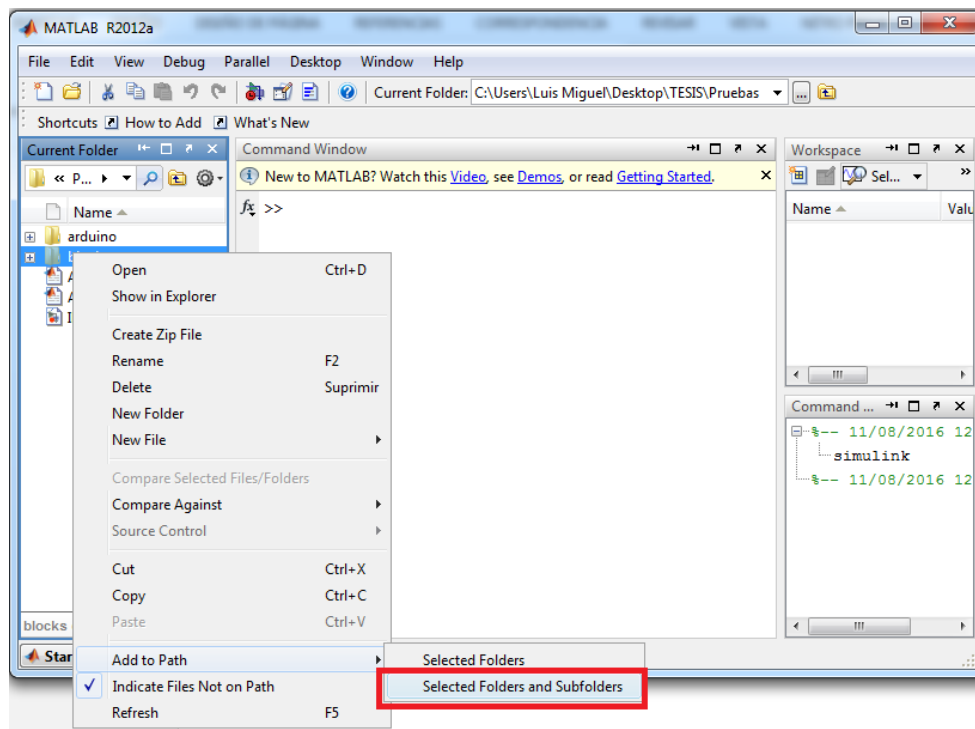


Figura 33 Agregar al directorio de trabajo.

3. Posteriormente se ejecuta el script “ArduinoBuildMexFiles.m” (véase Figura 34) y se debe esperar a que la ejecución termine [13].

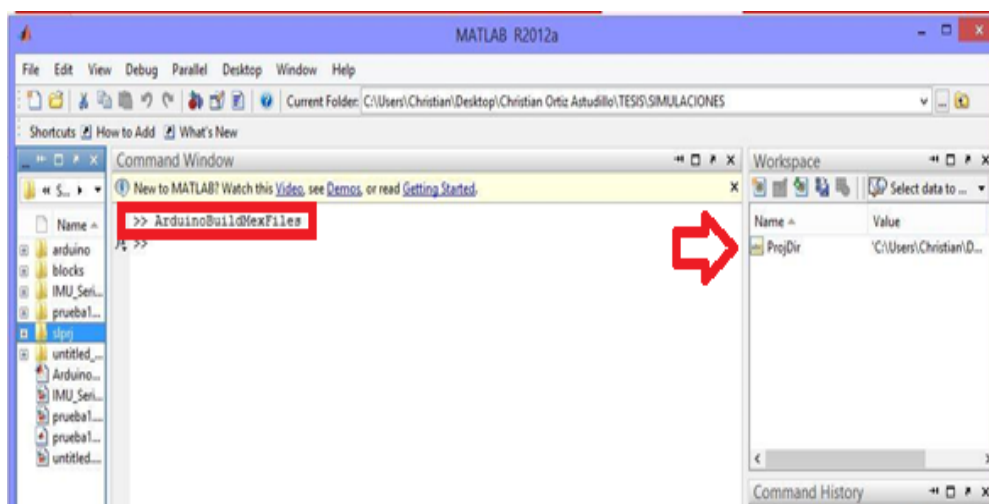


Figura 34 Resultado de la ejecución.

4. Desde un nuevo modelo en Simulink® dirigirse a la barra de herramientas en donde se debe seleccionar Simulation>Configuration_Parameters, en donde se abrirá una nueva ventana. En la Figura 35 se muestra la ventana de la configuración de parámetros en donde se cambiarán los parámetros que se encuentran dentro del recuadro rojo, para Type el parámetro será Fixed-step y para Solver el parámetro será discrete. Finalmente se procede a dar clic en el botón Apply [13].

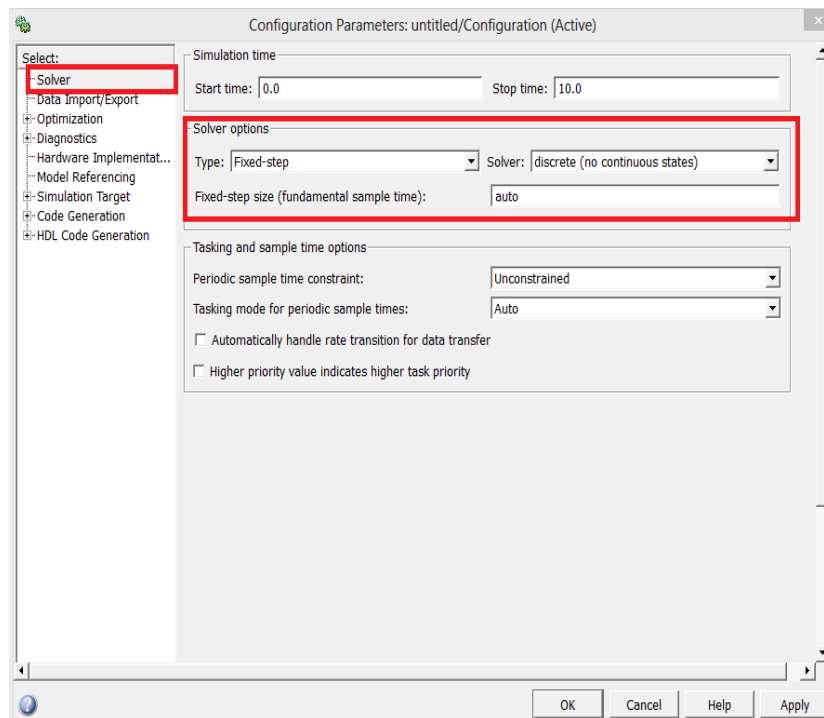


Figura 35 Configuración de Parámetros en Simulink

5. Posteriormente en la pestaña Hardware Implementation, en la opción Device vendor se debe seleccionar Atmel. Tal como se muestra en la Figura 36 [13].

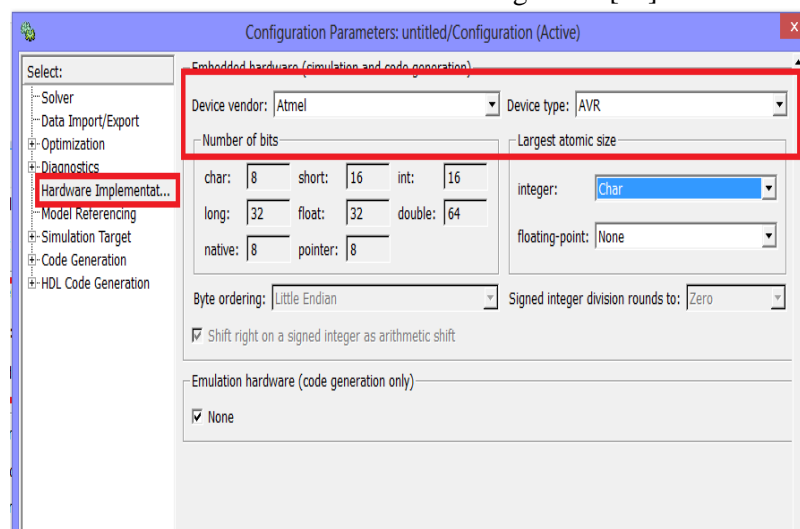


Figura 36 Configuración Hardware Implementation

6. Luego en la barra de herramientas se debe dirigir hacia la siguiente dirección: Tools>Run_on_Target_Hardware>Prepare_to_Run y la ventana que se presentará se muestra en la Figura 37 [13].

En la Figura 37 se presenta la configuración del parámetro externo que se utilizara, es decir la tarjeta con la que se deberá trabajar que para este caso es el ArduPilot, pero este dispositivo contiene una tarjeta de comunicación denominada Arduino Mega 2560. Aquí el único cambio es la tarjeta Hardware, el resto de parámetros serán por defecto. Cabe destacar que la opción Signal monitoring and parameter tuning, no es posible usarla con esta librería [13].

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

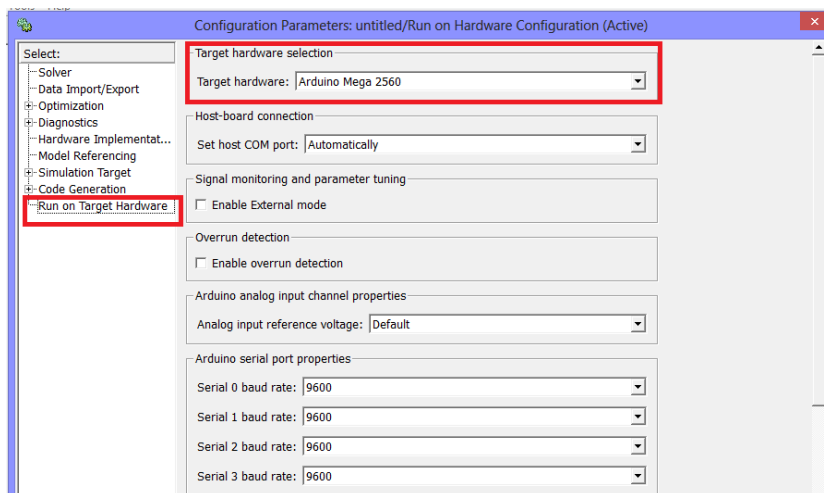


Figura 37 Selección de la Plataforma a utilizar.

- Finalmente se debe dirigir a la barra de herramientas en TOOLS>RUN y el modelo creado en Simulink® se descargará directamente en el ArduPilot [13].

3.3. Análisis de la Librería APM Block 2.0

La Librería APM Block 2.0 (véase Figura 38) disponible para Simulink®, se muestra como una librería con el nombre ArduPilot2 Target

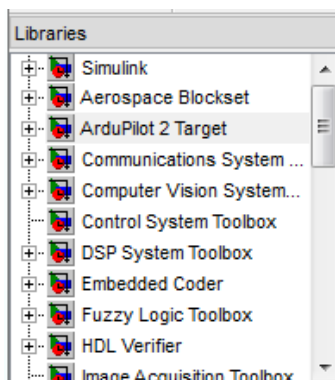


Figura 38 Librería APM Blockset 2.0

Esta librea posee los siguientes componentes:

- Analog IO.

BLOQUE	DESCRIPCION
	<p>Este bloque lee un voltaje de entrada analógico entre 0 y 5 V de un pin específico y da como salida un valor de 0 a 1023 [3].</p>

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

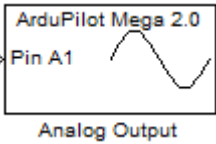
	<p>Este bloque aproxima una salida analógica en un pin específico, usando una señal PWM, con una frecuencia de 490 Hz. La entrada de este bloque es una señal entera entre 0 y 255 correspondiente a valores entre 0 y 5 V [3].</p>
---	---

Tabla 5 Entradas y salidas analógicas

- Baro.

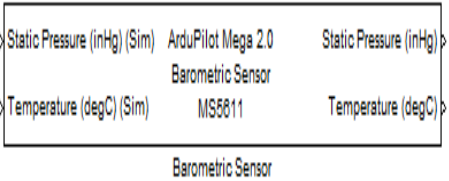
BLOQUE	DESCRIPCION
	<p>Este bloque lee desde el sensor barométrico APM2. Las entradas son usadas solamente con fines de simulación y se pasan directamente a través de las salidas cuando se utiliza en este modo. Cuando se leen los datos desde el hardware, este bloque leerá los datos desde el dispositivo actual [3].</p>

Tabla 6 Sensor Barométrico

- Data Type Conversions.

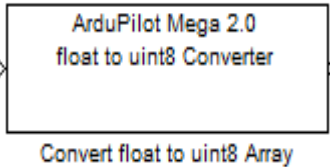
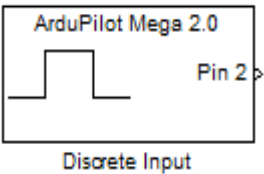
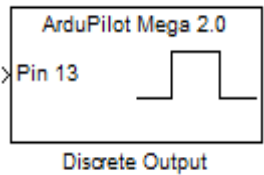
BLOQUE	DESCRIPCION
	<p>Este bloque toma un valor de tipo float y lo convierte en un array de valores uint8, para usarlo con el Serial Write block [3].</p>

Tabla 7 Conversor de datos

- Discrete IO.

BLOQUE	DESCRIPCION
	<p>Este bloque lee el valor digital de un pin específico en el dispositivo APM2. Indica en la salida 1 para alto y 0 para bajo [3].</p>
	<p>Este bloque envía un valor digital a un pin específico en el dispositivo APM2. Una entrada de 1 pondrá el pin alto y una de 0 en bajo [3].</p>

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

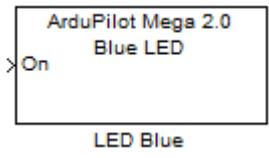
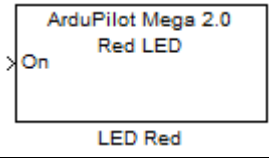
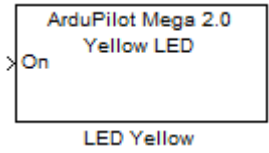
	<p>Este bloque controla el LED Azul de la tarjeta. Al recibir en el bloque un valor de 1 el LED se encenderá y con 0 se apagará [3].</p>
	<p>Este bloque controla el LED Rojo de la tarjeta. Al recibir en el bloque un valor de 1 el LED se encenderá y con 0 se apagará [3].</p>
	<p>Este bloque controla el LED Amarillo de la tarjeta. Al recibir en el bloque un valor de 1 el LED se encenderá y con 0 se apagará [3].</p>

Tabla 8 Entradas y salidas digitales

- Flash.

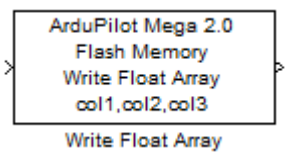
BLOQUE	DESCRIPCION
	<p>Este bloque escribe un array de tipo floating point en una memoria flash [3].</p>

Tabla 9 Flash

- GPS.

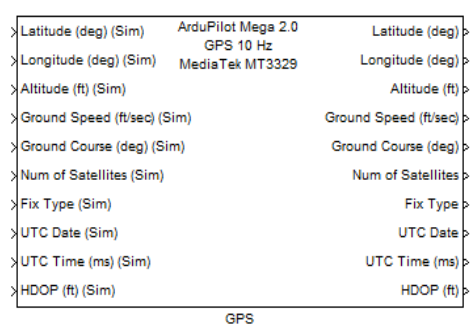
BLOQUE	DESCRIPCION
	<p>Este bloque tiene las salidas etiquetadas. El tiempo UTC (Universal Time Cordinated) está en milisegundos para la media noche. La Fecha UTC es un número de formato DDMMYY [3].</p> <p>Fix Type:</p> <p>0 = No se encuentra instalada una unidad GPS (Failure!) 1 = Unidad GPS funcionando. Posición no detectada. 2 = Unidad GPS funcionando. Posición detectada.</p>

Tabla 10 GPS

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

- IMU.

BLOQUE	DESCRIPCION
	<p>Este bloque toma datos desde el ArduPilot 2.0. Los rangos máximos para los giroscopios y los acelerómetros pueden ser seleccionados por el usuario, pero se debe tener en cuenta que incrementar el rango decrece la resolución. La resolución se puede aproximar de la siguiente manera [3]:</p> $\text{Resolution} = (2 * \text{Max Range}) / (2^{16})$

Tabla 11 IMU

- Magnetometer.

BLOQUE	DESCRIPCION
	<p>Este bloque nos proporcionara datos del campo magnético censado por el magnetometro en su nanotesla [3].</p>

Tabla 12 Magnetómetro

- Pitot.

BLOQUE	DESCRIPCION
	<p>Es un sensor analógico que tiene baja velocidad. Las entradas pasan directamente a las salidas cuando se encuentra en modo de simulación. Los datos son reemplazados con los datos medidos en el dispositivo cuando este está en funcionamiento [3].</p>

Tabla 13 Pitot

- Serial.

BLOQUE	DESCRIPCION
	<p>Este bloque escribe un vector de formato reconocible para el humano, los datos (ASCII) floating point se escriben en el puerto serial del dispositivo APM. Cada elemento en la</p>

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

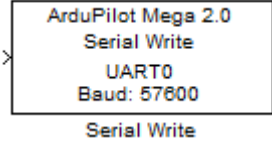
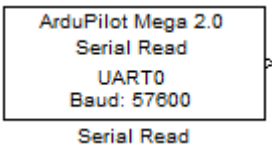
	entrada del array se podrá sacar separándolo con un espacio [3].
	Este bloque escribe un vector de bytes en el puerto serial del dispositivo ArduPilot 2.0. El bus definido por defecto es el UART0, sin embargo ya que este se comparte con el bus USB, se puede usar el UART2 al alterar el puente de solda como se indica en las instrucciones de uso del UART2 como bus de Telemetría [3].
	Este bloque lee un byte de la interfaz serial específico. Devuelve -1 si los datos no están disponibles para leer [3].

Tabla 14 Serial

- RC Channels.

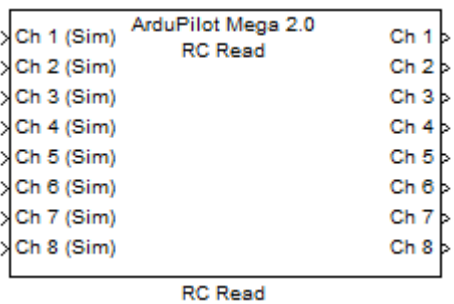
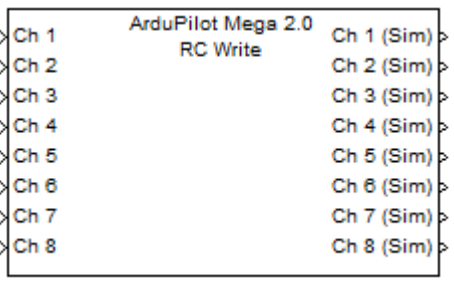
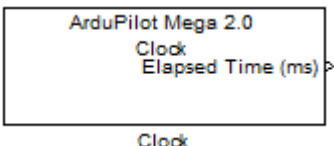
BLOQUE	DESCRIPCION
	Este bloque recibe los comandos RC desde un canal específico. En simulación el bloque pasa los puertos de entrada directamente a la salida. Las entradas no se usan cuando se está ejecutando el dispositivo mientras, el bloque lee desde el dispositivo actual [3].
	Este bloque envía los comandos RC a un canal específico. En simulación este bloque pasa los puertos de entrada directo a la salida. Las salidas no se usan cuando se está ejecutando el dispositivo mientras, el bloque envía datos al dispositivo actual [3].

Tabla 15 Canales RC

- Timing.

BLOQUE	DESCRIPCION
	Este bloque nos indica el número de milisegundos desde que la alimentación fue aplicada al dispositivo [3].

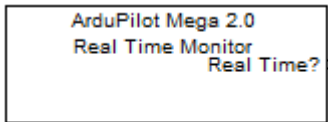
	<p>Este bloque determina si el dispositivo está funcionando en tiempo real. Nos indica verdadero si el programa está funcionando a la velocidad específica de simulación, o falso si está ejecutándose en velocidad distinta a la de simulación. Si está funcionando en una velocidad distinta el usuario deberá reducir la complejidad del código. Este bloque depende del tiempo de muestro definido por el usuario. Definir un tiempo incorrecto puede ocasionar errores [3].</p>
---	--

Tabla 16 Sincronización

3.4. Identificación y forma de Comunicación de los pines analógicos y digitales para la comunicación entre el sistema embebido y Simulink®

Con la librería que se instaló para poder establecer una comunicación entre el sistema embebido y Simulink®, se dispone de bloques que permiten el fácil manejo de los periféricos con los que cuenta dicha plataforma. Dentro de esta librería se dispone de bloques para el manejo de las entradas y salidas analógicas, digitales, Led's que dispone la plataforma, el manejo del puerto serial y el manejo de diferentes módulos como el GPS, IMU, Timers, entre otros [13].

Con la característica de Simulink® llamada ejecutarse sobre el hardware es posible descargar programas realizados en Simulink® sobre la plataforma ArduPilot, de tal manera que facilita la programación en dicha plataforma, como se conoce la programación en Simulink® es mediante bloques [13].

Una de las ventajas de descargar el diagrama realizado sobre la plataforma es que evita hacer trabajar al procesador del ordenador lo que permite que el ordenador realice otras tareas [13].

Los bloques que se utilizan para realizar diseños de estrategias de control en Simulink con la plataforma ArduPilot son:

- Analog Input (véase Figura 39).
- Analog Output (véase Figura 40).

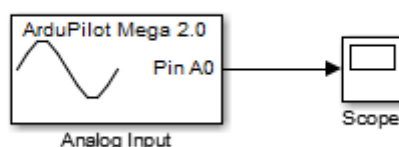


Figura 39 Analog Input

En la Figura 39 se aprecia el bloque “Analog Input” de la librería ArduPilot, en donde este bloque permite la adquisición de datos de un proceso y permite visualizar estos datos en Simulink

CAPITULO 3. INTERFAZ DE COMUNICACIÓN (SIMULINK)

mediante un scope. Al bloque se debe configurar el Pin con el que se va a trabajar (A0). Las características de este bloque se mencionaron anteriormente.

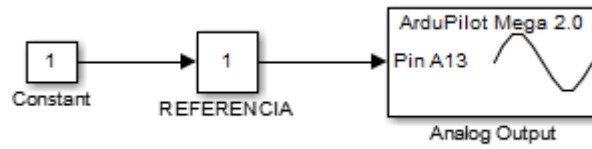


Figura 40 Analog Output

En la Figura 40 se presenta el bloque “Analog Output”, este bloque permitirá enviar datos desde la plataforma ArduPilot al proceso con el que se trabaje. En este bloque se debe configurar el Pin que se utilizara (A13). Las características de este bloque se mencionaron anteriormente

CAPÍTULO 4

4. SISTEMAS DE CONTROL AUTOMÁTICO

4.1. Identificación del Sistema.

4.1.1. Planta B del laboratorio de Control y Automatización Totalmente Integrado de la Universidad Politécnica Salesiana Sede Cuenca.


La Planta B es un sistema industrial disponible en el Laboratorio de Control y Automatización Totalmente Integrado de la Universidad Politécnica Salesiana Sede Cuenca, la misma que está compuesta por una posicionador electroneumático, una válvula Y de 2/2 Vías, un depósito y por sensores de nivel y flujo (véase Figura 41). Además esta planta cuenta con un sistema de alimentación el cual proporciona fluido desde un reservorio a través de una bomba controlada por un variador de velocidad.

Esta planta es ideal para realizar lazos de control tanto de flujo como de nivel. Para evaluar el desempeño de la plataforma ArduPilot se ha seleccionado el lazo de flujo el cual contiene los instrumentos detallados en la Tabla 17.



Figura 41 Planta B del laboratorio de control y automatización totalmente integrado de la Universidad Politécnica Salesiana

Los elementos de la Planta B se presentan a continuación, la información ha sido obtenida de [14] y [15].

Nombre del elemento	Descripción
Válvula Y de 2/2 Vías 	Modelo: Burkert Flow 2000 Valve Variable controlada: Agua, alcoholes, aceites, combustibles, líquidos hidráulicos. Máxima presión: 150 PSI Máxima viscosidad: 600 mm ² /s Temperatura: -10°C a 180°C

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO




<p align="center">Posicionador Electroneumático Digital</p> 	<p>Modelo: Burkert 8792 Alimentación: 24 VDC +/- 10% Presión: 1.4 a 7 bar Salida: 4-20 mA, 0-20 mA, 0-10 V, 0-5 V</p>
<p align="center">Transmisor Digital de Flujo</p> 	<p>Modelo: Burkert SE35/8035 Alimentación: 12-30 V DC (V+) ± 10% Variable medida: Velocidad de flujo Rango de medida: 0.5 a 10 m/s Precisión: ≤ ±0.5% Salida: 4-20 mA Temperatura: 0 hasta 50°C</p>
<p align="center">Variador de frecuencia</p> 	<p>Modelo: Danfoss VLT 2800 Comunicación: Mediante protocolos de comunicación (hart o profibus). Tensión de alimentación: 200 a 220 Voltios. Frecuencia de alimentación: 50/60 Hz ±3Hz. Factor de potencia real: 0.9 a carga nominal. Factor de potencia de desplazamiento: >0.98.</p>

Tabla 17 Elementos del proceso de flujo [14] [15]

Se ha desarrollado un Diagrama PI&D (véase Figura 42 o APENDICE D) en donde se puede observar los elementos actuadores y sensores que conforman la Planta B.

Donde:

- LI-101=Indicador de Nivel.
- FI-101=Indicador de Flujo.
- LIC-201=Indicador Controlador de Nivel.
- FC-201=Controlador de Flujo.
- LAH-90=Alarma de Nivel Alto.
- LAL-25 Alarma de Nivel Bajo.
- P-101=Bomba de Agua.
- TK-101=Tanque de Reserva
- Variador-101=Variador de Frecuencia

Los elementos que se han usado en nuestro proceso de control son:

- FI-101 (Transmisor Digital de Flujo Burkert SE35/8035) [14].
- FC-201 (Válvula Y de 2/2 Vías Burkert Flow 2000 Valve) [14].
- LIC-201 (Posicionador Electroneumático Digital Burkert 8792) [14].
- P-101=Bomba de Agua.
- TK-101=Tanque de Reserva.
- Variador-101=Variador de Frecuencia.

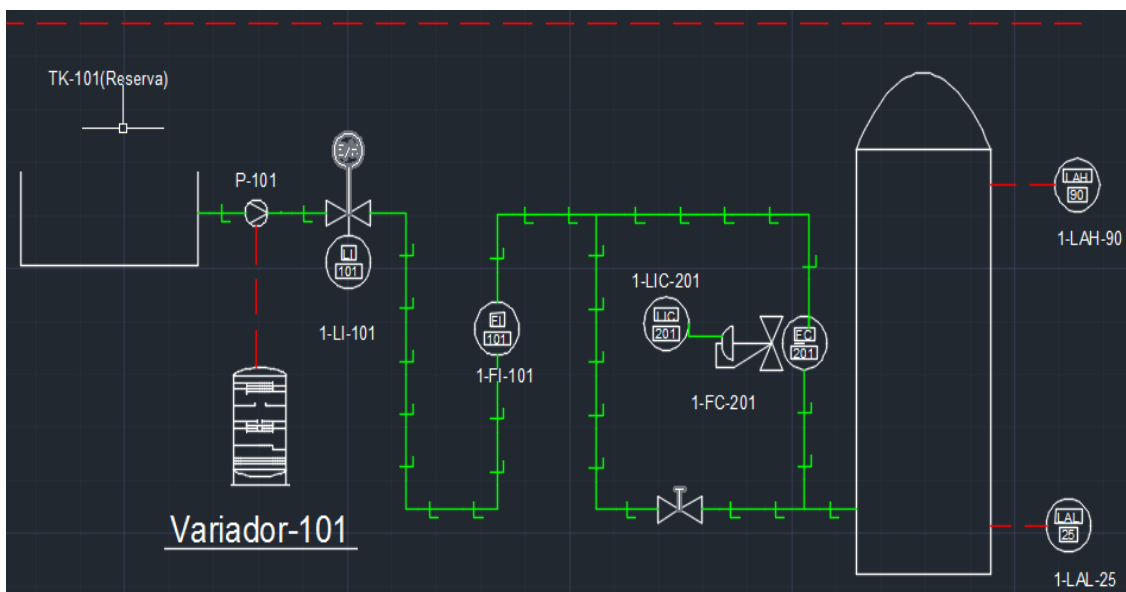


Figura 42 Diagrama PI&D.

4.1.2. Identificación del proceso de flujo

Para la identificación del sistema se ha generado escalones con diferentes valores de amplitud, para ello se ha realizado un diseño en el software Simulink (véase Figura 43) el mismo que permitirá enviar señales desde la plataforma hacia la Planta B y también permitirá adquirir las señales de comportamiento del sistema a los cambios de referencia que se generen.

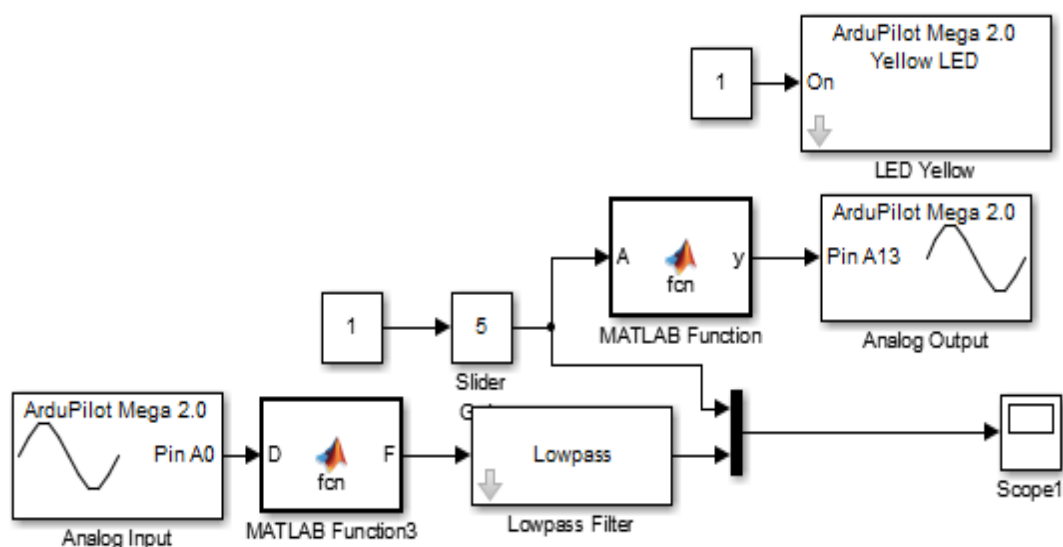


Figura 43 Programa para Identificación del Sistema

En la Figura 43 se puede observar un programa realizado en Simulink, el cual sirve para identificar el proceso. Este programa dispone de una entrada analógica y una salida analógica para la lectura y escritura de las variables respectivamente. Los bloques Matlab Function sirven para hacer un escalado de dichas variables de manera que puedan ser escritas y leídas de manera correcta, es decir la entrada analógica lee la señal en un rango de 0-1023 y esta debe ser convertida a una señal de voltaje de 0-5V, mientras que la salida analógica debe ser convertida de 0-5V a 0-255 que es el rango con el que trabaja la plataforma ArduPilot.

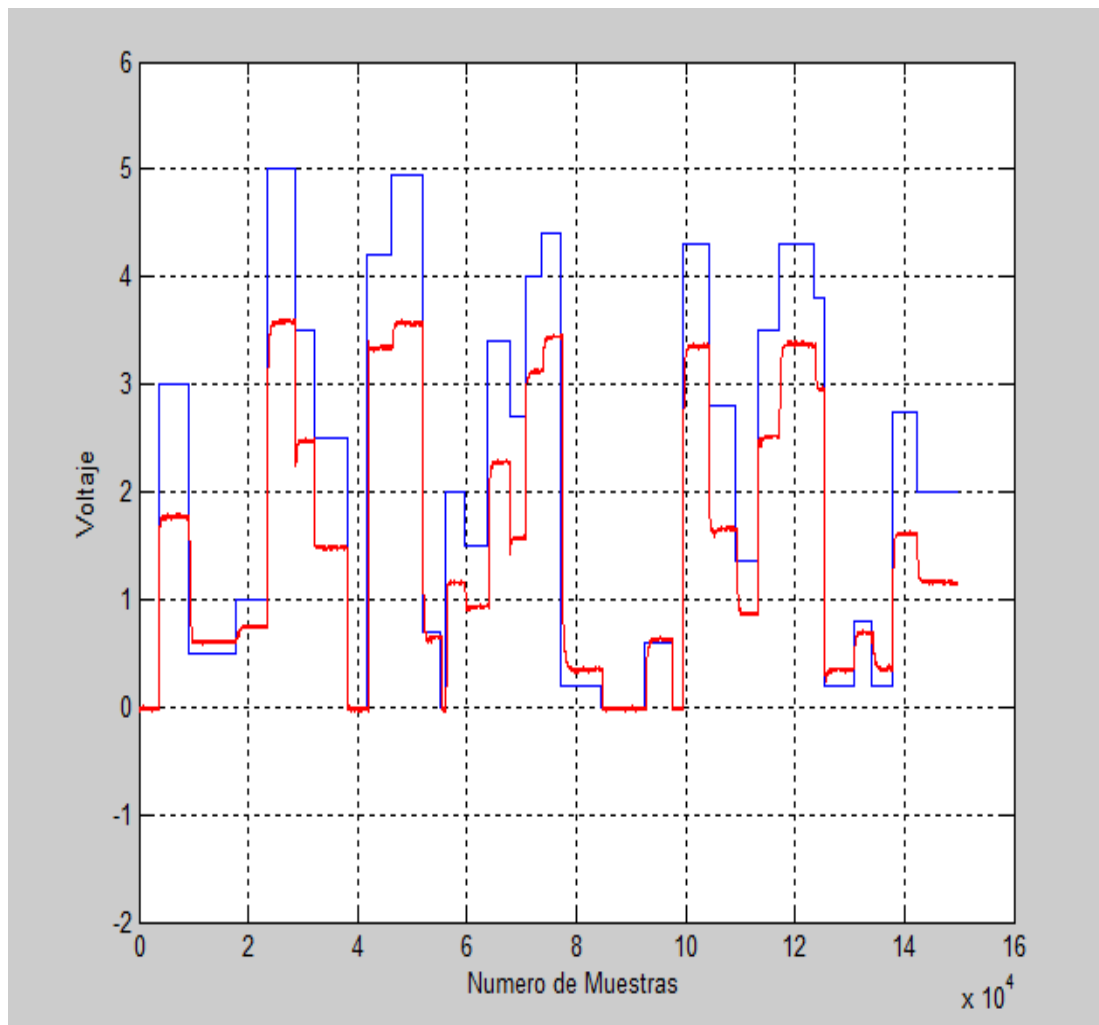


Figura 44 Identificación del Sistema

En la Figura 44 se muestra la respuesta del sistema a los escalones generados, en donde la señal azul es el escalón generado y la señal de color rojo es la respuesta del sistema. En el eje 'x' se encuentra el número de muestras y en el eje 'y' se encuentra el valor en voltaje.

Mediante el comando “ident” en el software Matlab®, se obtiene la función de transferencia del proceso de flujo.

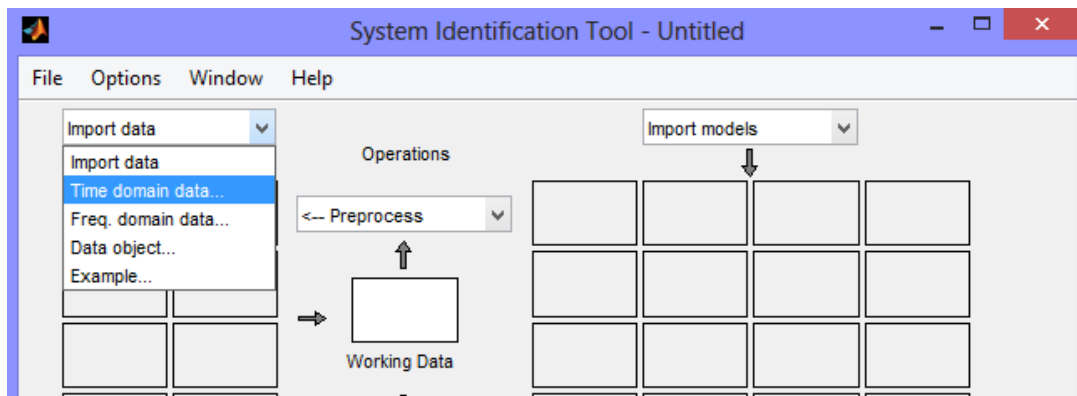


Figura 45 Ventana principal del comando "ident"

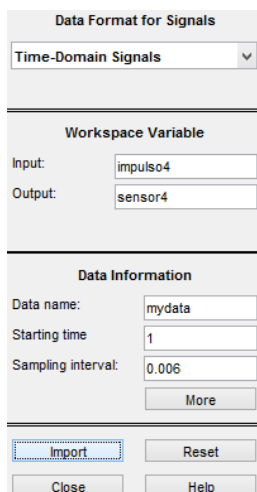


Figura 46 Datos del sistema ingresados

En la ventana principal del comando “ident” (véase Figura 45) se ingresa el nombre de las variables que contienen los datos de entrada y salida del proceso, los cuales fueron obtenidos con el diagrama mostrado en la Figura 44. Otro parámetro que se debe configurar es el tiempo de muestreo con el que se realizó la identificación del sistema (6 ms). Una vez configurado estos parámetros se presiona “Import” y aparecerá otra ventana (véase Figura 47).

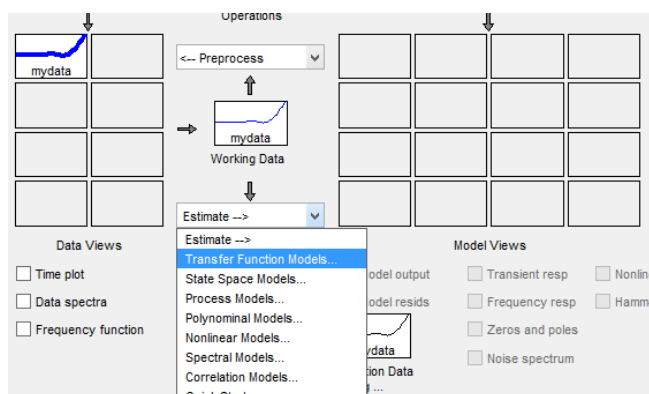


Figura 47 Obtención de la Función de Transferencia

En la Figura 47 se muestra como obtener la estimación del modelo de la función de transferencia. Una vez ingresado los datos (DatosPlanta) se debe configurar el número de polos y ceros que se desee en la función de transferencia (véase Figura 48).

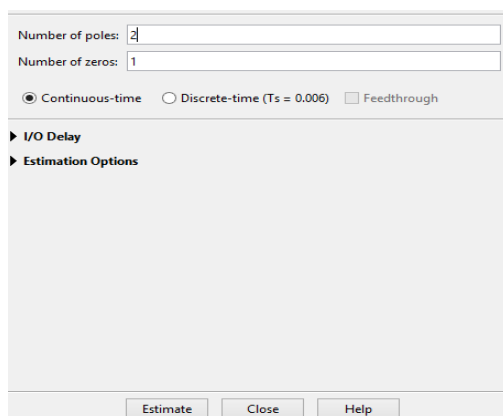


Figura 48 Configuración de polos y ceros del modelo de la función de transferencia

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

Una vez indicado el número de polos y ceros se presiona “Estimate” y la herramienta presentará en la cuadrícula derecha el modelo estimado (véase Figura 49).

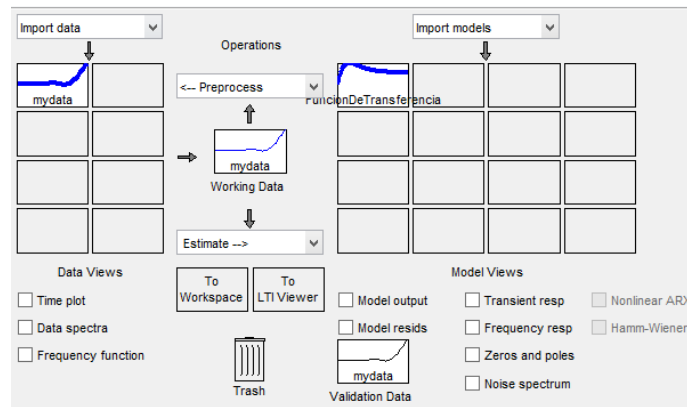


Figura 49 Proceso de obtención de la función de transferencia terminado

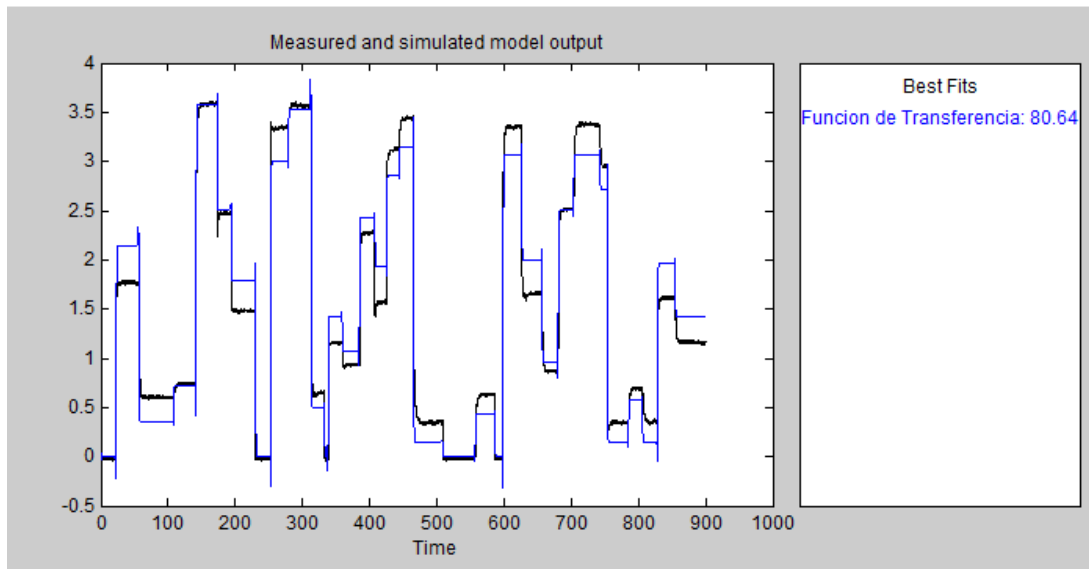


Figura 50 Aproximación del modelo de la función de transferencia y el sistema físico

En la Figura 50 se muestra el porcentaje de aproximación del modelo de la función de transferencia (80.54%) obtenido con un número de polos igual a 2 y un número de ceros igual a 1. La función de transferencia identificada se muestra en la Figura 51.

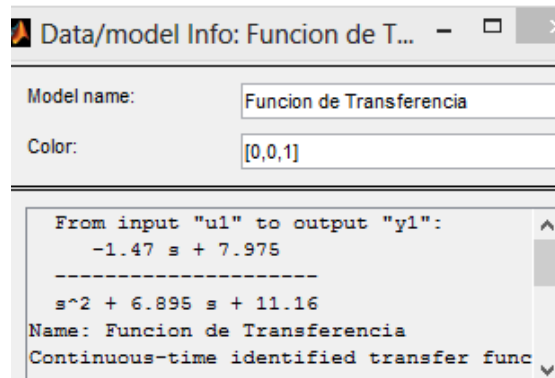


Figura 51 Función de transferencia del sistema

4.2. Análisis de estrategias de control.

4.2.1. Controladores PID.

Un controlador PID (Proporcional-Integrador-Derivativo), es un controlador con muy buenas prestaciones y de fácil implementación. Este controlador genera una acción de control en función del error producido entre la entrada deseada y la señal de retroalimentación. La acción proporcional permite mejorar la ganancia del sistema sin embargo esta debe ser regulada bajo ciertos criterios porque un exceso podría provocar que el sistema sea demasiado oscilatorio. La acción derivativa como su nombre lo indica permite tener información respecto a la derivada o tasa de cambio del error. Con la ganancia derivativa se puede regular la velocidad de respuesta del sistema. Por otra parte la acción integral permite corregir el error en estado estacionario [16].

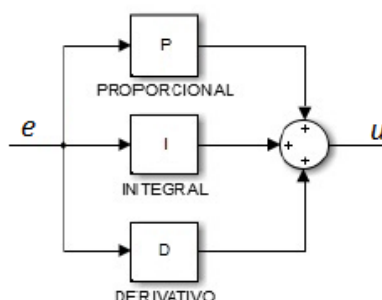


Figura 52 Diagrama de bloques de un controlador PID [21]

En la Figura 52 se aprecia la estructura de un controlador PID, en donde “e” representa la señal de error y “u” representa la señal de acción de control. “P”, “I” y “D” son las constantes encontradas para que el controlador funcione correctamente [17].

La acción de control designada para un controlador PID en tiempo continuo esta descrita por la ecuación 1.

$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int e(\tau) d\tau + T_d \frac{de(t)}{dt} \right] \quad (1)$$

Mediante el software Matlab® es posible realizar la sintonización del controlador PID con la ayuda del comando “Sisotool” (véase Figura 53), se puede sintonizar de manera automática o de manera gráfica. La forma gráfica consiste en un método interactivo que permite la variación de polos y ceros dentro del lugar geométrico de las raíces [18].

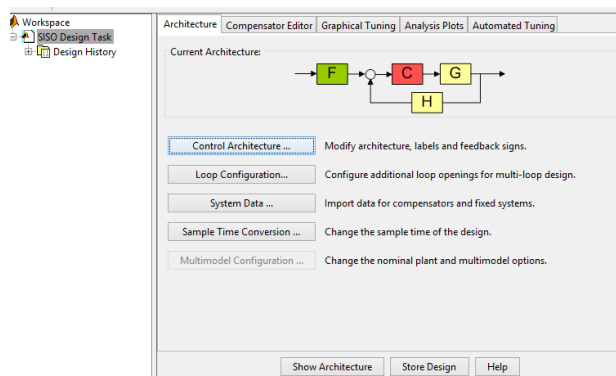


Figura 53 Ventana de inicio del Toolbox Sisotool-Matlab®

Otra forma de sintonizar controladores PID es mediante comando “Pidtool” (véase Figura 54) de **Matlab**, en donde las constantes de sintonización se encuentran gráficamente. En “Pidtool” se

observa la respuesta al escalón del controlador dependiendo del valor de las tres constantes “Kp”, “Ki” y “Kd”.

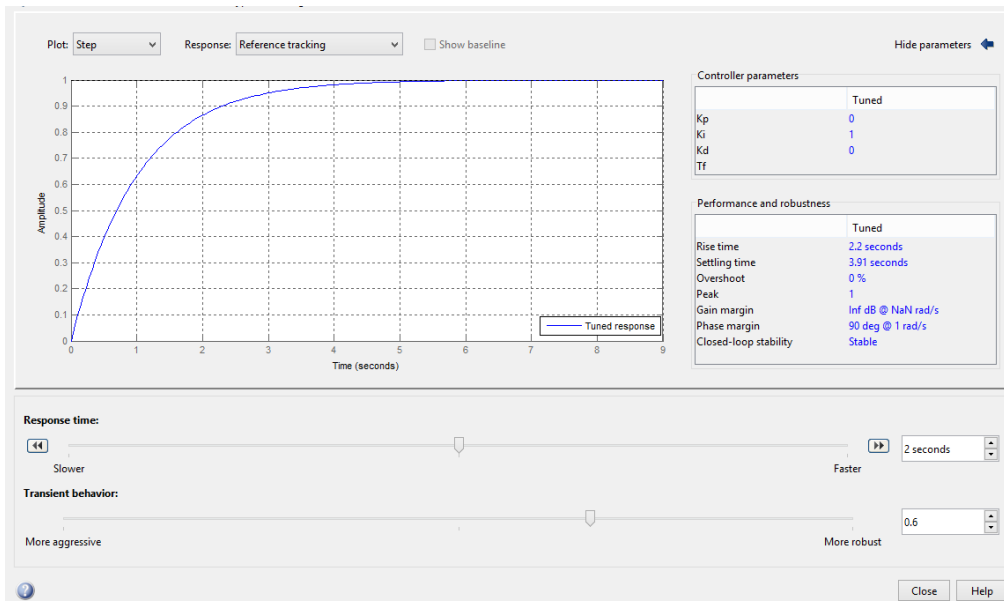


Figura 54 Ventana principal del Toolbox Pidtool-Matlab®

4.2.2. Controladores en espacio de estados en tiempo discreto.

El método de espacio de estados en tiempo discreto permite visualizar un sistema en términos de n ecuaciones en diferencias. Al representar el sistema en forma de matrices se sintetiza la forma matemática del sistema. La Ecuación de estado en sistemas lineales de tiempo discreto invariantes en el tiempo, se puede definir como [19]:

$$x(k + 1) = Gx(k) + Hu(k) \quad (2)$$

La Ecuación de salida en sistemas lineales de tiempo discreto invariantes en el tiempo, se puede definir como [19]:

$$y(k) = Cx(k) + Du(k) \quad (3)$$

Donde:

- Vector de estado: $x(k) =$ vector n
- Vector de salida: $y(k) =$ vector m
- Vector de entrada: $u(k) =$ vector r
- Matriz de Estado: $G(k) =$ matriz $n \times n$
- Matriz de Entrada: $H(k) =$ matriz $n \times r$
- Matriz de Salida: $C(k) =$ matriz $m \times n$
- Matriz de transmisión directa: $D(k) =$ matriz $m \times r$

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

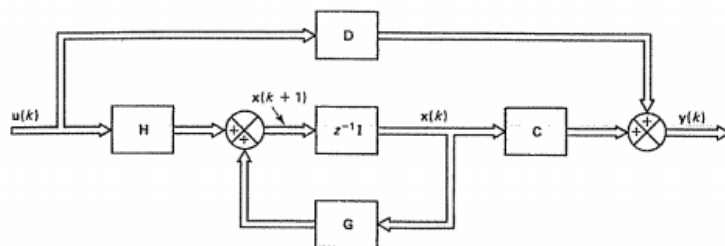


Figura 55 Diagrama de bloques de un sistema de control lineal en tiempo discreto invariante en el tiempo representado en el espacio de estado [19].

Las Ecuaciones 2 y 3 están representadas en la Figura 55.

Existen casos en los que no todas las variables de estado están disponibles para su medición directa, es por ello que mediante un observador se pueden estimar las variables que no se miden directamente, esto se logra tomando en cuenta las variables de salida y de control. Un observador solo se puede diseñar si se cumple la condición de observabilidad. La condición de observabilidad se puede verificar mediante las ecuaciones 4 y 5, siendo estas las ecuaciones de estado y de salida respectivamente [20].

$$x(k + 1) = Gx(k) + Hu(k) \quad (4)$$

$$y(k) = Cx(k) \quad (5)$$

Donde

- Vector de estado: $x(k) = \text{vector } n$
- Vector de salida: $y(k) = \text{vector } m$
- Vector de control: $u(k) = \text{vector } r$
- Matriz no singular: $G(k) = \text{matriz } nxn$
- Matriz: $H(k) = \text{matriz } nxr$
- Matriz: $C(k) = \text{matriz } mxn$

En la Figura 56 se puede observar el sistema de control lineal en tiempo discreto invariante en el tiempo con un observador de estados.

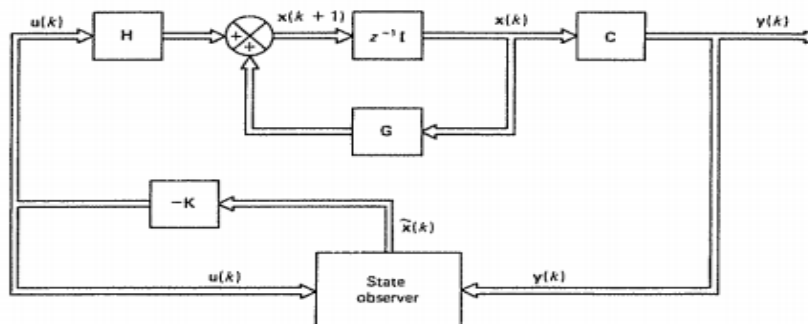


Figura 56 Sistema de regulación con observador de estados [20].

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

La ecuación 6 sirve para saber si un sistema es completamente observable:

$$\text{rango}[C^* \ G^*C^* \ \dots \ (G^*)^{n-1}C^*] = n \quad (6)$$

En un observador de estado de orden completo aparece una matriz de Ganancia de realimentación de estado “K” [20], finalmente las ecuaciones 7, 8 y 9 describen el sistema de control con realimentación de estados [20].

$$x(k + 1) = Gx(k) + Hu(k) \quad (7)$$

$$y(k) = Cx(k) \quad (8)$$

$$u(k) = -Kx(k) \quad (9)$$

Donde

- Vector de estado: $x(k) = \text{vector } n$
- Vector de salida: $y(k) = \text{vector } m$
- Vector de control: $u(k) = \text{vector } r$
- Matriz no singular: $G(k) = \text{matriz } nxn$
- Matriz: $H(k) = \text{matriz } nxr$
- Matriz: $C(k) = \text{matriz } mxn$
- Matriz de Ganancia de realimentación de estado: $K(k) = \text{matriz } mxr$

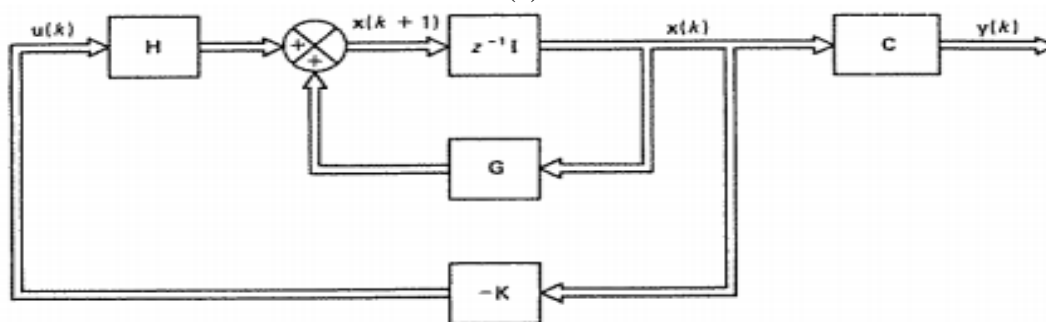


Figura 57 Sistema de control con realimentación de estado [20].

En la Figura 57 se puede observar el diagrama de bloques del sistema de control con realimentación de estados, descrito por las ecuaciones 7, 8 y 9. Al incorporar una matriz de ganancia de realimentación del observador “Ke” y mediante el estado observado se forma el vector de control [20].

Se forma un sistema de control con realimentación de estado observado, el cual se puede apreciar en la Figura 58 y está descrito por la ecuación 10. A este sistema se le llama comúnmente observador de predicción [20].

$$\tilde{x}(k + 1) = (G - K_e C)\tilde{x}(k) + Hu(k) + K_e y(k) \quad (10)$$

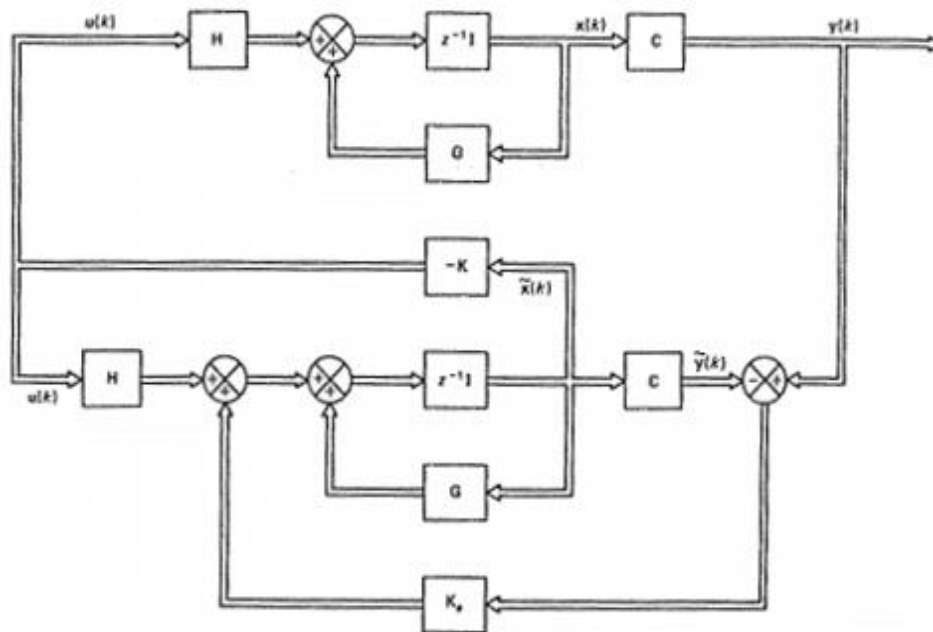


Figura 58 Sistema de control con realimentación de estado observado [20].

Un sistema de seguimiento añade uno o más integradores dentro del lazo cerrado de control. El controlador integral que se observa en la Figura 59 está formado por m elementos integradores, determinados por las m entradas que dirigen al sistema o también llamadas m referencias [20].

Las matrices K_1 y K_2 que se observan en la Figura 59, sirven como un parámetro de diseño [20]. Estas matrices se pueden determinar mediante la ubicación de polos en lazo cerrado [20]. Estas matrices se pueden encontrar con distintos métodos como la fórmula de Ackerman y el Filtro de Kalman, a su vez las matrices K_1 y K_2 deseadas se pueden calcular mediante la ecuación 11.

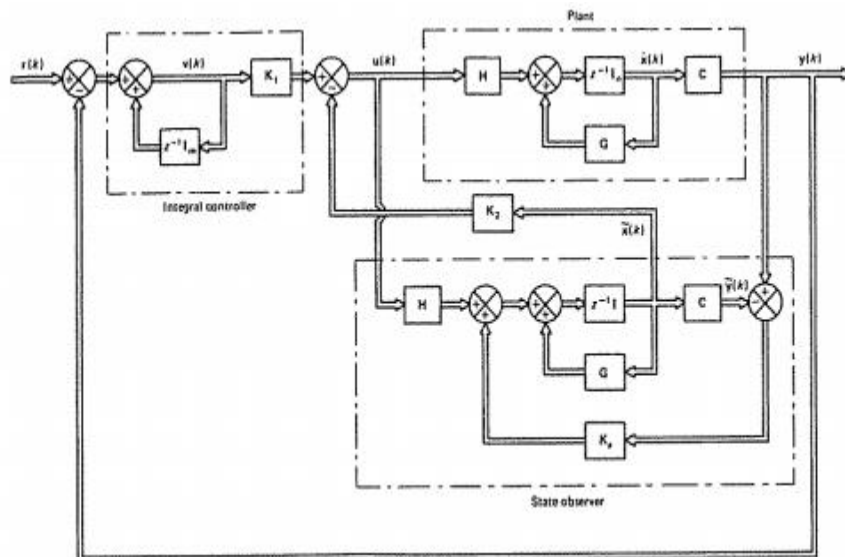


Figura 59 Sistema de Seguimiento con realimentación de estado y control integral [20].

$$[K_2:K_1] \begin{bmatrix} G - I_n & H \\ CG & CH \end{bmatrix} = \tilde{R} + [0 : I_m] \quad (11)$$

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

Donde \hat{K} será $m \times (n + m)$ matriz.

$$\hat{K} = [K_2G - K_2 + K_1CG : I_m + K_2H + K_1CH] \quad (12)$$

La fórmula de Ackerman permite encontrar la matriz de ganancia de realimentación del observador o también llamado K_e [21]. Esta fórmula se presenta en la ecuación 13:

$$K_e = \phi(A) \cdot \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad (13)$$

Donde el polinomio característico esperado para el observador de estado es:

$$\phi(A) = \phi(S) = \prod_{i=1}^n (s - u_i) = (s - u_1)(s - u_2) \dots (s - u_n) \quad (14)$$

Otra alternativa para observar los estados de un sistema es el Filtro de Kalman, el cual mediante el uso de ecuaciones matemáticas desarrolla un estimador predictor-corrector, para disminuir la covarianza del error estimado [21].

El Filtro de Kalman se puede aplicar en un sistema discreto como se muestra a continuación:

$$\begin{aligned} x_k &= A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1} \\ y_k &= C_kx_k + v_k \end{aligned}$$

Donde:

- Ruido del proceso: w_k
- Ruido en la medida: v_k

Las variables w_k y v_k se pueden representar con una distribución normal de probabilidad como se muestra en las ecuaciones 15 y 16.

$$p(w_{k-1}) \sim N(0, Q) \quad (15)$$

$$p(v_k) \sim N(0, R) \quad (16)$$

El filtro divide sus ecuaciones en dos procesos: actualización en el tiempo o predicción y actualización en las medidas o corrección [21].

- Actualización en el Tiempo o predicción.

En esta fase se estima el estado actual mediante la ecuación 17 y además se calcula la covarianza del error P mediante la ecuación 18, partiendo del error en el instante anterior [21].

$$\hat{x}_k |_{k-1} = A\hat{x}_{k-1} |_{k-1} + Bu_k \quad (17)$$

$$P_k |_{k-1} = AP_{k-1} |_{k-1} A^T + Q \quad (18)$$

- Actualización en las medidas o corrección.

En esta fase se calcula la ganancia de Kalman K , usando el valor de P , calculado en la fase anterior, esto se describe en la ecuación 19 [21]. En la fase de corrección también se obtiene los nuevos valores de la estimación del estado a través de las medidas mediante la ecuación 20, [21]. Finalmente se calcula la nueva covarianza del error usando la ganancia de Kalman y la covarianza del error en el instante anterior, esto se realiza mediante la ecuación 21 [21].

$$K_k = P_k|_{k-1} C^T (C P_k|_{k-1} C^T + R)^{-1} \quad (19)$$

$$\hat{x}_k|_k = \hat{x}_k|_{k-1} + K_k (y_k - C \hat{x}_k|_{k-1}) \quad (20)$$

$$P_k|_k = (I - K_k C) P_k|_{k-1} \quad (21)$$

4.2.3. Control predictivo basado en modelo (MPC).

El control predictivo es la estrategia más común de control por computador. Este tipo de controlador consiste en predecir el comportamiento de las salidas del sistema. En el control predictivo basado en modelo se dispone de un horizonte de predicción y un horizonte de control. Se denomina horizonte de predicción al intervalo de tiempo que se va a predecir considerando que este intervalo está dado por instantes de muestreo. El horizonte de control deberá ser menor al horizonte de predicción, de tal manera que para comportamientos que se encuentren más allá del horizonte de control la acción de control deberá ser constante [22].

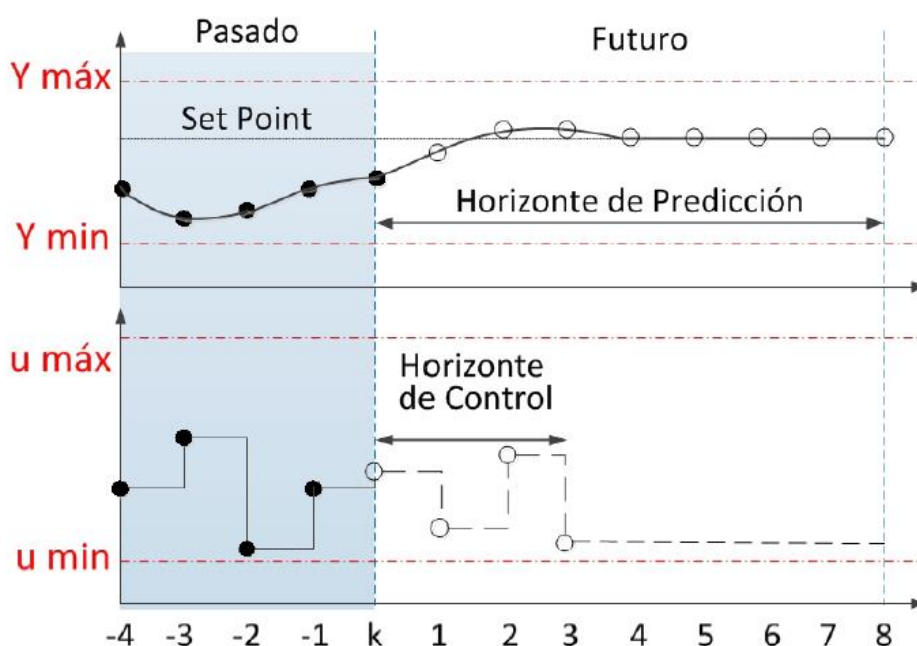


Figura 60 Estrategia de control Predictivo [22]

La estrategia MPC está basada en tres parámetros fundamentales: el modelo de predicción, la función de coste y la optimización [17].

- **Modelo de predicción:** Este parámetro define el modelo matemático del comportamiento que se espera en el sistema.
- **Función de coste:** Este parámetro expresa el criterio que se tendrá para optimizar.
- **Optimización:** Este parámetro intenta aproximar la señal que se predice con la señal de referencia, se lo realiza en cada instante considerando los comportamientos en el horizonte de control.

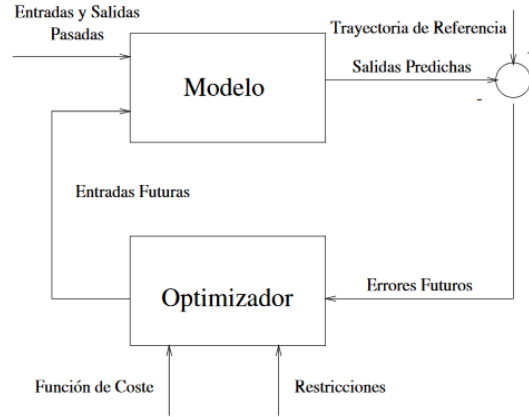


Figura 61 Estructura general de los Controladores Predictivos [17]

De acuerdo a [22] la formulación matemática de un controlador predictivo basado en modelo utilizando espacio de estados se presenta a continuación:

La formulación matemática para el MPC utilizando espacio de estados parte de un modelo con integrador embebido el cual asegura tener un error nulo en el estado estacionario (véase las ecuaciones 23 y 24).

$$x(k+1) = Ax(k) + Bu(k) \quad (23)$$

$$y(k) = Cx(k) + Du(k) \quad (24)$$

El modelo descrito se ve representados por las ecuaciones 25 y 26

$$\begin{bmatrix} x(k+1) \\ \Delta x(k+1) \\ y(k) \end{bmatrix} = \begin{bmatrix} A_e \\ A & o_m^T \\ CA & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ \Delta x(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_e \\ B \\ C.B \end{bmatrix} \Delta u(k) \quad (25)$$

$$y(k) = \begin{bmatrix} C_e \\ o_m & 1 \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix} \quad (26)$$

Del modelo aumentado se calcula las predicciones de estado y salida el instante presente k_i (véase las ecuaciones 27, 28 y 29).

$$x(k_i + 1 | k_i) = A_e x(k_i) + B_e \Delta u(k_i) \quad (27)$$

$$x(k_i + 2 | k_i) = A_e^2 x(k_i) + A_e B_e \Delta u(k_i) + B_e \Delta u(k_i + 1) \quad (28)$$

$$x(k_i + N | k_i) = A_e^N x(k_i) + A_e^{N-1} B_e \Delta u(k_i) + A_e^{N-M} B_e \Delta u(k_i + M - 1) \quad (29)$$

Donde N representa el horizonte de predicción y M el horizonte de control.

Para las predicciones de salida se utiliza el mismo criterio (véase las ecuaciones 30, 31 y 32).

$$y(k_i + 1 | k_i) = C_e A_e x(k_i) + C_e B_e \Delta u(k_i) \quad (30)$$

$$y(k_i + 2 | k_i) = C_e A_e^2 x(k_i) + C_e A_e B_e \Delta u(k_i) + B_e \Delta u(k_i + 1) \quad (31)$$

$$y(k_i + N | k_i) = C_e A_e^N x(k_i) + C_e A_e^{N-1} B_e \Delta u(k_i) + C_e A_e^{N-2} B_e \Delta u(k_i + 1) + \dots + C_e A_e^{N-M} B_e \Delta u(k_i + M - 1) \quad (32)$$

El conjunto de predicciones está representado por la ecuación 33

$$Y = Fx(k_i) + \Phi \Delta U \quad (33)$$

Donde F y Φ están basados en las matrices del modelo aumentado (véase las ecuaciones 34 y 35).

$$F = \begin{bmatrix} C_e A_e \\ C_e A_e^2 \\ C_e A_e^3 \\ \vdots \\ C_e A_e^N \end{bmatrix} \quad (34)$$

$$\Phi = \begin{bmatrix} C_e B_e & 0 & \dots & 0 \\ C_e A_e B_e & C_e B_e & \dots & 0 \\ C_e A_e^2 B_e & C_e A_e B_e & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ C_e A_e^{N-1} B_e & C_e A_e^{N-2} B_e & \dots & C_e A_e^{N-M} B_e \end{bmatrix} \quad (35)$$

La función de coste (véase ecuación 36) minimiza el error entre la salida predicha y el punto de ajuste R_s considerando los parámetros de la señal ΔU (véase ecuación 37 y 38).

$$J(y, u) = (R_s - Y)^T (R_s - Y) + \Delta U^T R \Delta U \quad (36)$$

Donde:

$$\Delta U = H \Phi^T [R_s - r(k_i) - Fx(k_i)] \quad (37)$$

$$H = (\Phi^T \Phi + R)^{-1} \quad (38)$$

Una vez encontrado el vector ΔU únicamente se considera el primer elemento de este vector, mientras que los restantes son omitidos. Con el primer elemento de ΔU se puede calcular la acción de control U, que será enviada a la planta en cada instante de muestreo.

4.2.4. Control PID adaptativo.

Un controlador adaptativo permite realizar un ajuste automático en el control del sistema cuando existan perturbaciones externas que puedan cambiar el funcionamiento del sistema. Este controlador es robusto y sencillo [23].

En un sistema de control adaptativo se tiene un modelo de referencia para el comportamiento del sistema [24]. Este modelo ayuda a ajustar el error de acuerdo al algoritmo de control del sistema [24]. El esquema de control se puede observar en la Figura 62.

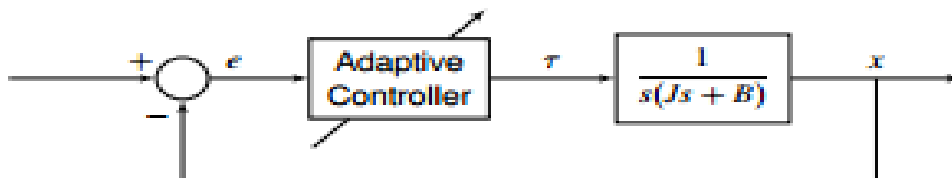


Figura 62 Diagrama de un Control Adaptativo [24].

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

Donde:

- Algoritmo de Control PID Adaptativo [24].

$$\tau = kp(t)e(t) + ki(t) \int e(\tau) d\tau + kd(t)\dot{e}(t) \quad (39)$$

- Planta [24].

$$J\ddot{x} + B\dot{x} = \tau \quad J > 0 \quad (40)$$

De acuerdo a [25] se presenta la formulación matemática del controlador PID adaptativo.

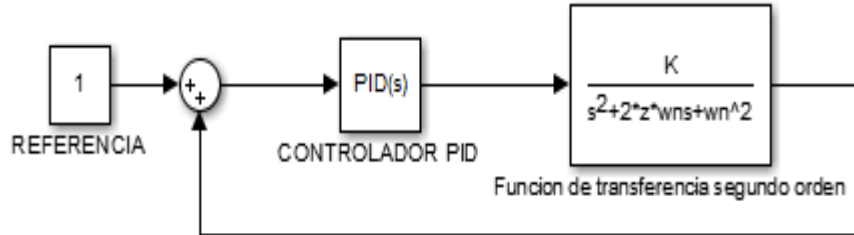


Figura 63 Esquema para el modelo de referencia

Para hallar el modelo de referencia necesario para el controlador Adaptativo se debe partir de un proceso de lazo cerrado como se aprecia en la Figura 63.

Donde la acción de control del controlador PID está descrita por la ecuación 41.

$$U = kp * E(t) + \frac{ki * E(t)}{s} + kd * s * E(t) \quad (41)$$

Factorizando la ecuación 41 la acción de control será (véase ecuación 42):

$$U = E(t) \left(\frac{kp * s + ki + kd * s^2}{s} \right) \quad (42)$$

El modelo de referencia está representado por la ecuación 43:

$$M(s) = \frac{G(s) * C(s)}{1 + G(s)C(s)H(s)} \quad (43)$$

Donde

$$G(s) = \frac{k}{s^2 + A_1s + A_2}$$

$$C(s) = \left(\frac{kp * s + ki + kd * s^2}{s} \right)$$

$$H(s) = 1$$

Reemplazando los datos en la ecuación 43 se tiene el modelo de referencia de tercer orden para el controlador PID Adaptativo (véase ecuación 44).

$$M(s) = \frac{k * (kd * s^2 + kp * s + ki)}{s^3 + s^2(A_1 + kd) + s(A_2 + k * kp) + k * ki} \quad (44)$$

Para hallar las constantes del controlador PID adaptativo (véase ecuaciones 45, 46 y 47) se utiliza la regla de MIT.

$$kp = -\frac{\gamma'}{s} e \left(\frac{s}{s^3 + A_1 s^2 + A_2 s + A_3} \right) E(s) \quad (45)$$

$$ki = -\frac{\gamma'}{s} e \left(\frac{1}{s^3 + A_1 s^2 + A_2 s + A_3} \right) E(s) \quad (46)$$

$$kd = -\frac{\gamma'}{s} e \left(\frac{s^2}{s^3 + A_1 s^2 + A_2 s + A_3} \right) E(s) \quad (47)$$

Donde:

$E(s)$ representa la señal de error, A_1 , A_2 y A_3 son parámetros del sistema a controlar y gamma prima es una constante de sintonía del controlador.

4.3. Diseño y Simulación de Estrategias de Control.

4.3.1. Diseño de un controlador PID.

Una vez encontrada la función de transferencia del proceso de flujo se debe diseñar las estrategias de control para lo cual se utiliza Simulink de Matlab®. La primera estrategia de control a diseñar es el controlador PID (véase Figura 64) para lo cual se utiliza el comando “pidtool” de Matlab® el mismo que permite encontrar los parámetros de sintonización del controlador. El comando “pidtool” necesita conocer la función de transferencia para generar un respuesta al escalón (véase Figura 65) y de tal manera obtener las constantes (KD, KP, KI) necesarias para el controlador PID.

Las constantes obtenidas con la ayuda del comando “pidtool” y con las que la respuesta al escalón responde de la mejor manera se las puede apreciar en la Figura 66. Posteriormente estas constantes deben ser ingresadas en el diseño del controlador PID (véase Figura 64). Los valores de las constantes ingresadas en el diseño del controlador en Simulink se los puede apreciar en la Figura 67, Figura 68 y Figura 69.

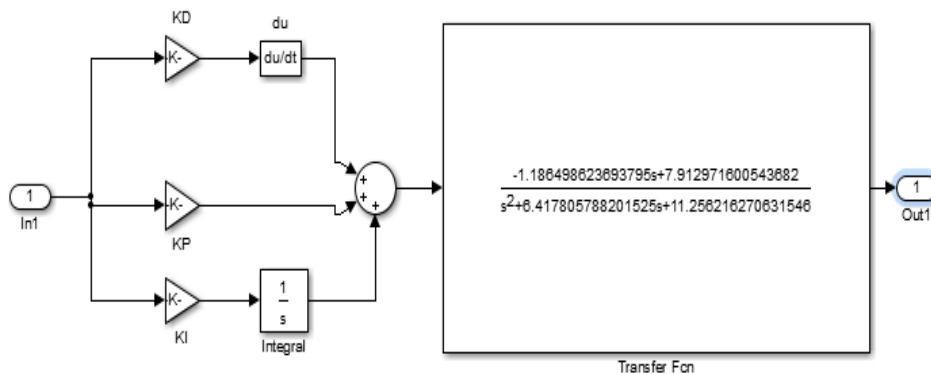


Figura 64 PID Para el proceso de flujo

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

En la Figura 64 se puede apreciar el diagrama del PID y la función de transferencia del sistema a controlar. Aquí se debe configurar las constantes del controlador y también se configura el tiempo de muestreo.

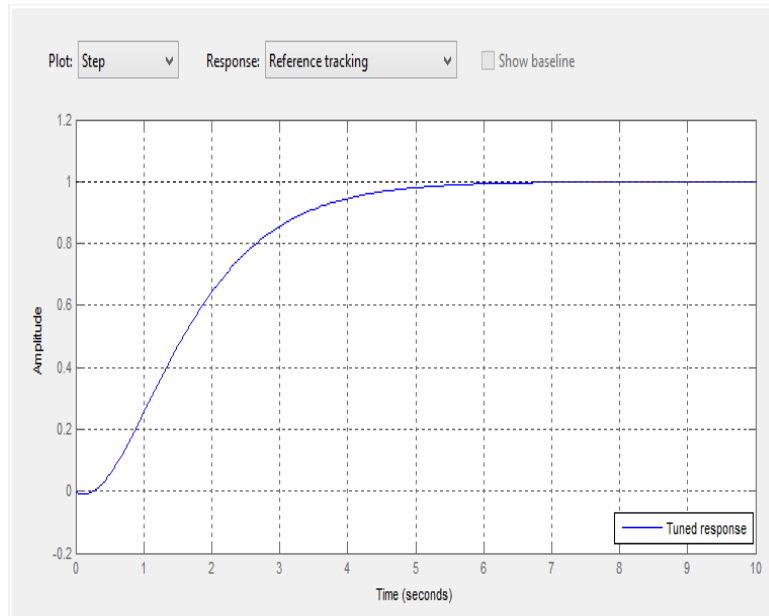


Figura 65 Respuesta del controlador en PIDTOOL a un escalón unitario

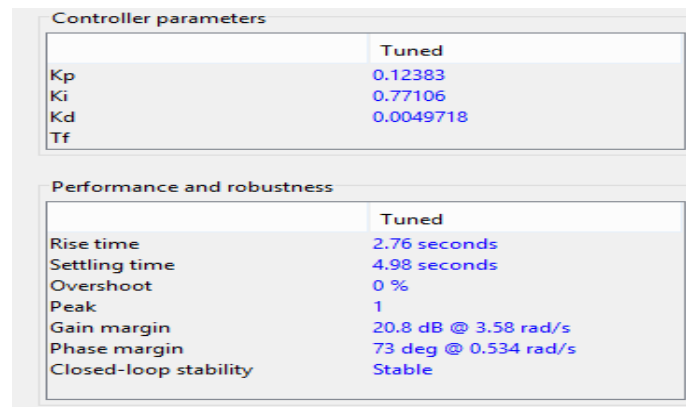


Figura 66 Constantes obtenidas en PIDTOOL para el controlador PID

En la Figura 66 se puede observar además de las constantes de sintonización las características con las que responde el controlador como por ejemplo el tiempo de establecimiento (4.98s), el sobrepaso (0%), el tiempo de levantamiento (2.76s), entre otros.

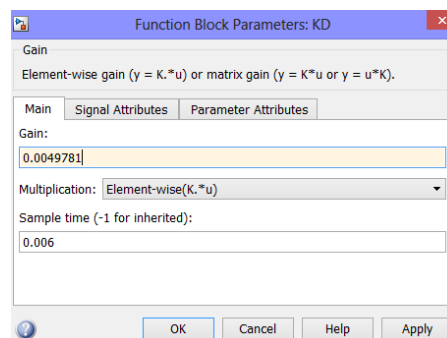


Figura 67 Constante KD

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

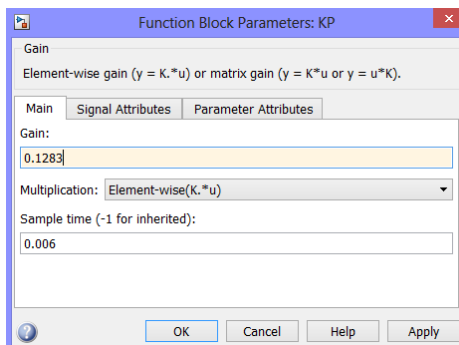


Figura 68 Constante KP

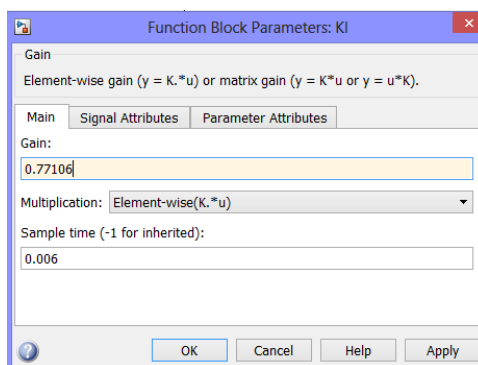


Figura 69 Constante KI

Se realizó la simulación del controlador PID para el proceso de flujo, en donde se analiza el comportamiento que muestra el sistema. En la Figura 70 se encuentra el diseño realizado para la simulación.

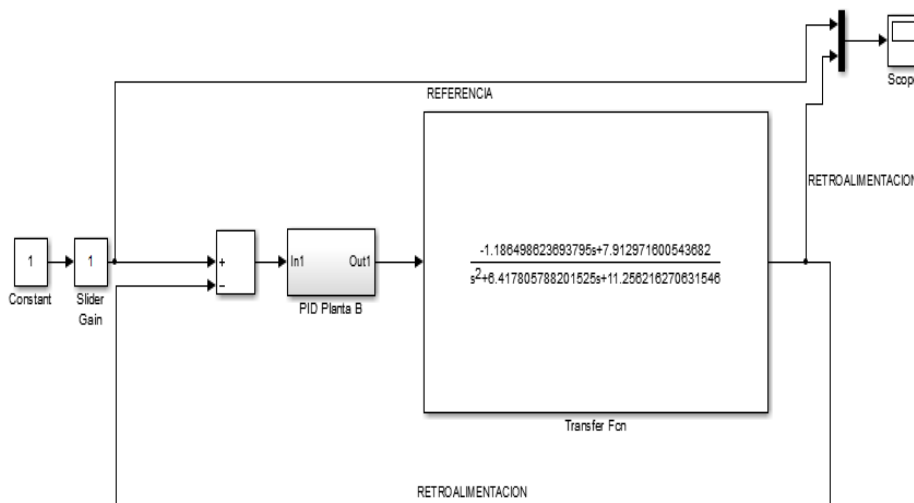


Figura 70 Simulación del controlador en Simulink

En el subsistema “PID Planta B” que se observa en la Figura 70 es en donde se ha ingresado el diseño del controlador PID al cual recibe la señal de error y tiene como salida la señal de acción de control. En la Figura 71 se aprecia la respuesta de la simulación, en donde la señal de color rojo representa la referencia y la de color azul representa la señal controlada, observando que su tiempo de establecimiento es aproximadamente de 5 segundos. En la simulación se genera cambios de referencia como lo que se evalúa el controlador PID para el proceso de flujo.

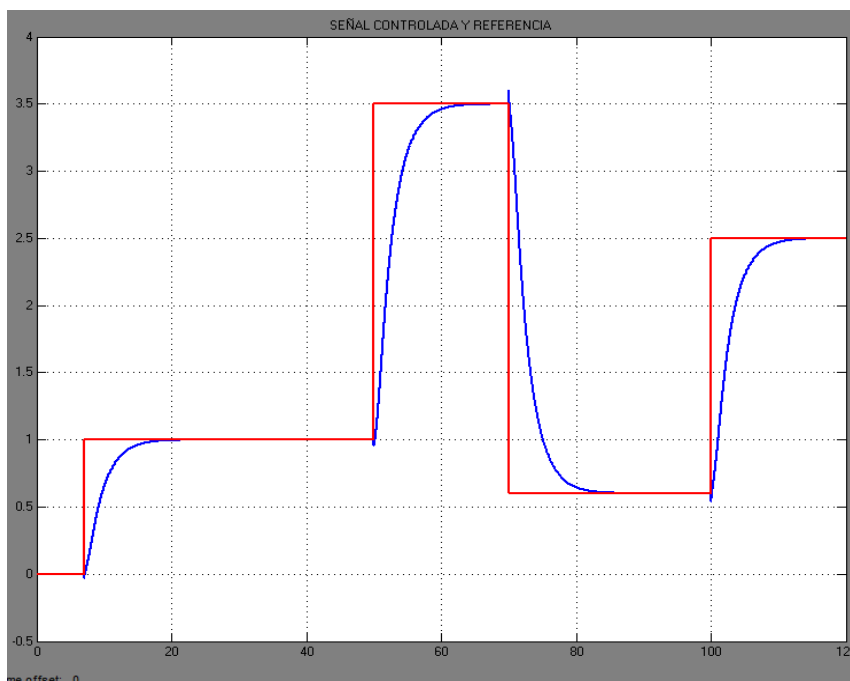


Figura 71 Respuesta de la simulación del sistema

4.3.2. Diseño del controlador por retroalimentación de estado.

El diseño del controlador se lo ha realizado en Simulink de Matlab®. En la Figura 72 se aprecia el diseño de simulación del controlador en Espacio de Estados en tiempo discreto para el proceso de flujo. En la gráfica se observa un subsistema “Espacio de Estados” en este se encuentra el diseño del controlador por realimentación de estados (Figura 73) al cual la señal de entrada es la diferencia entre la referencia deseada y la retroalimentación, mientras que la señal de salida del subsistema es la acción de control enviada a la función de transferencia del proceso de flujo.

Se debe primero comprobar que el sistema sea observable y controlable, para lo cual se utiliza los comandos “obsv” “ctrb” y rank” de MatLab®, estos comandos se los aprecia en el apéndice A. En el diseño de un controlador por retroalimentación de estados se requiere de la matriz de estados G, la matriz de entrada H y la matriz de salida C, los cuales son calculados con un código realizado en Matlab® (véase Apéndice “A”).

En la Figura 73 se puede apreciar el diagrama completo de un controlador en espacio de estados en tiempo discreto con sus correspondientes matrices que comprenden al diseño del controlador.

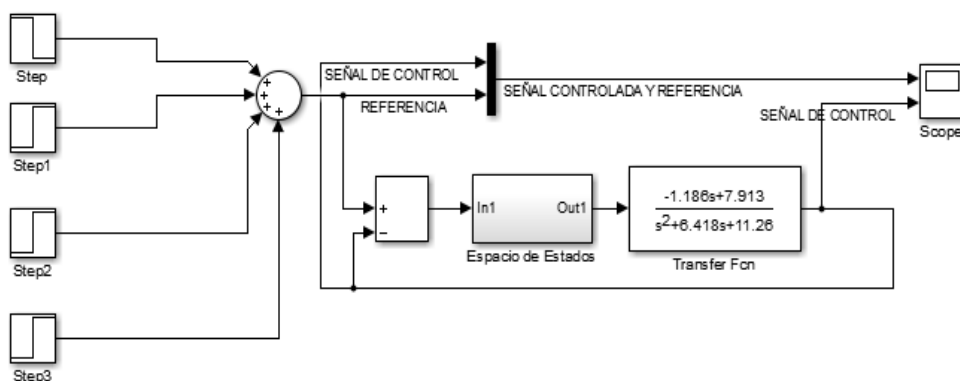


Figura 72 Diseño del controlador en Espacio de Estados en tiempo discreto para el proceso de flujo

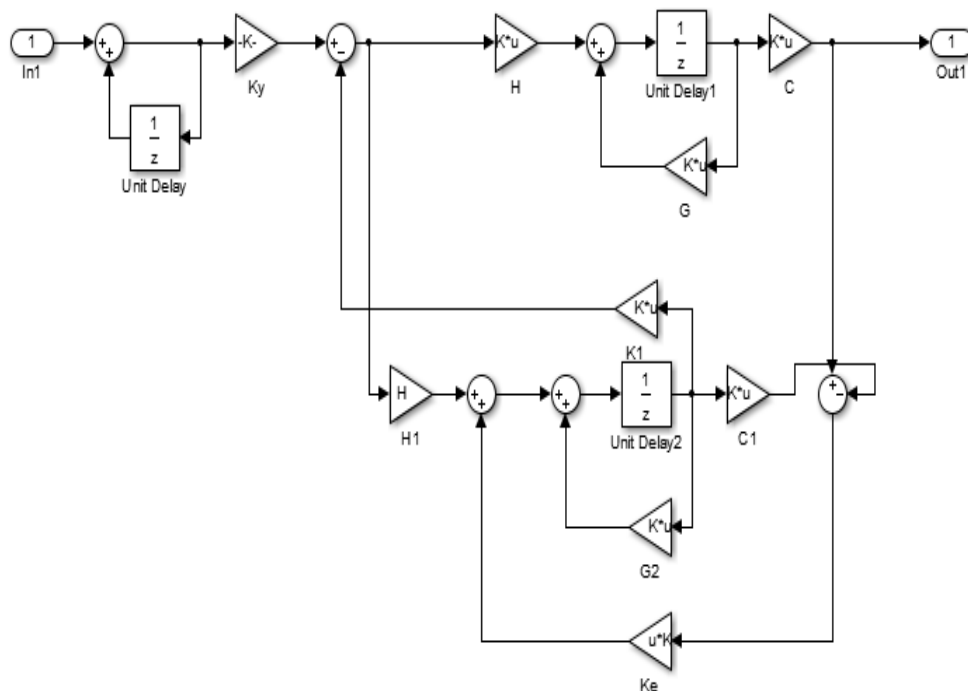


Figura 73 Controlador en Espacio de Estados en tiempo discreto del proceso de flujo

Para hallar las matrices expuestas anteriormente se utiliza el comando “ident” de Matlab®, en donde los valores que se utilizan son los de la señal de entrada y de salida que se obtuvieron anteriormente en la identificación del proceso de flujo esto se puede observar en la Figura 74.

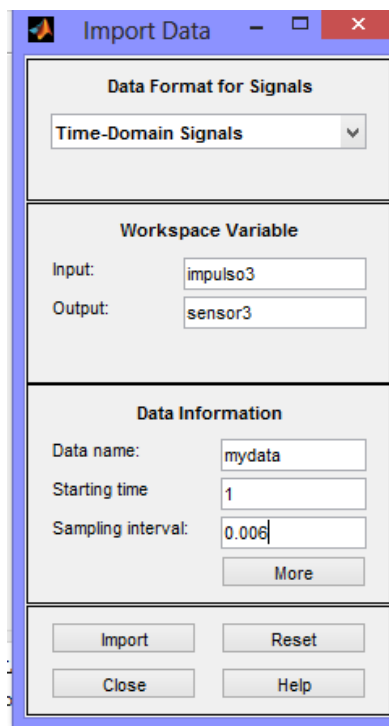


Figura 74 Ingreso de los datos de la Planta al usar el comando Ident.

Ahora se deben estimar los datos de entrada y salida a un modelo en espacio de estados en tiempo discreto de segundo orden, esto se puede observar en la Figura 75.

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

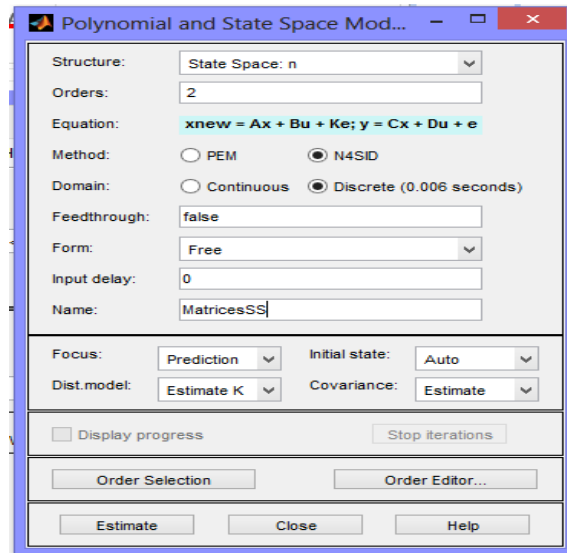


Figura 75 Estimación del modelo en espacio de estados en tiempo discreto

Finalmente se debe presionar en “Estimate” y se presentará una nueva ventana que contiene las matrices solicitadas (véase Figura 76).

Estas matrices se guardaran en un archivo .m, para ser usadas posteriormente en el código generado para encontrar los parámetros de esta estrategia de control.

Donde:

- $G = A$
- $H = B$
- $C = C$
- $D = D$

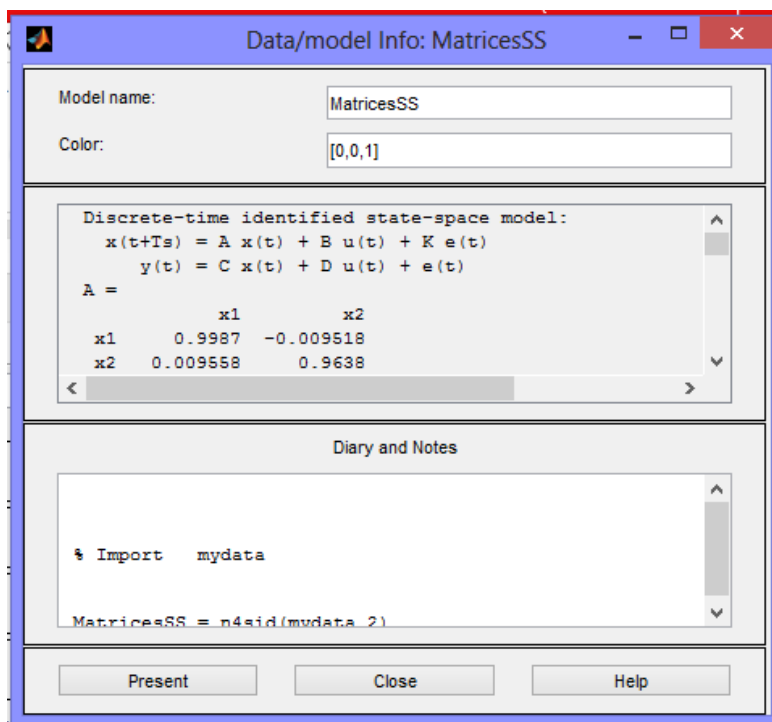


Figura 76 Matrices de Estado, Entrada, Salida y Transmisión directa

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

El modelo en espacio de estados se muestra en la Figura 77 con el nombre de MatricesSS. El archivo MatricesSS contiene los valores de las matrices G , H , C y D .

$$G = \begin{bmatrix} 0.9987 & -0.0095 \\ 0.0096 & 0.9638 \end{bmatrix}$$

$$H = \begin{bmatrix} -1.7726e - 06 \\ -1.0795e - 04 \end{bmatrix}$$

$$C = [100.8985 \quad -0.4497]$$

$$D = 0$$

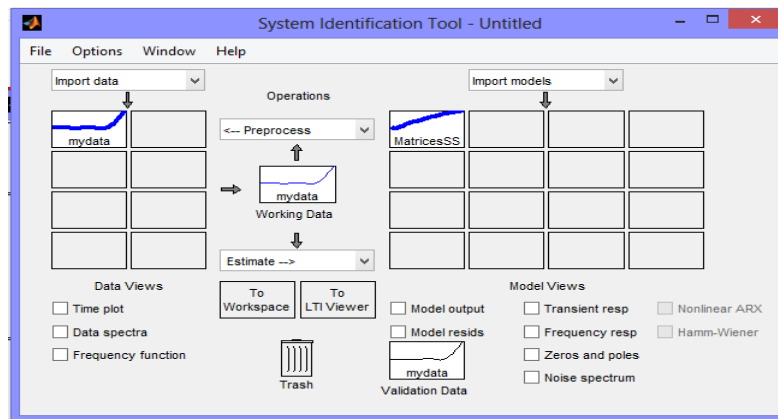


Figura 77 Matrices del espacio de estados representadas como un modelo de nombre MatricesSS

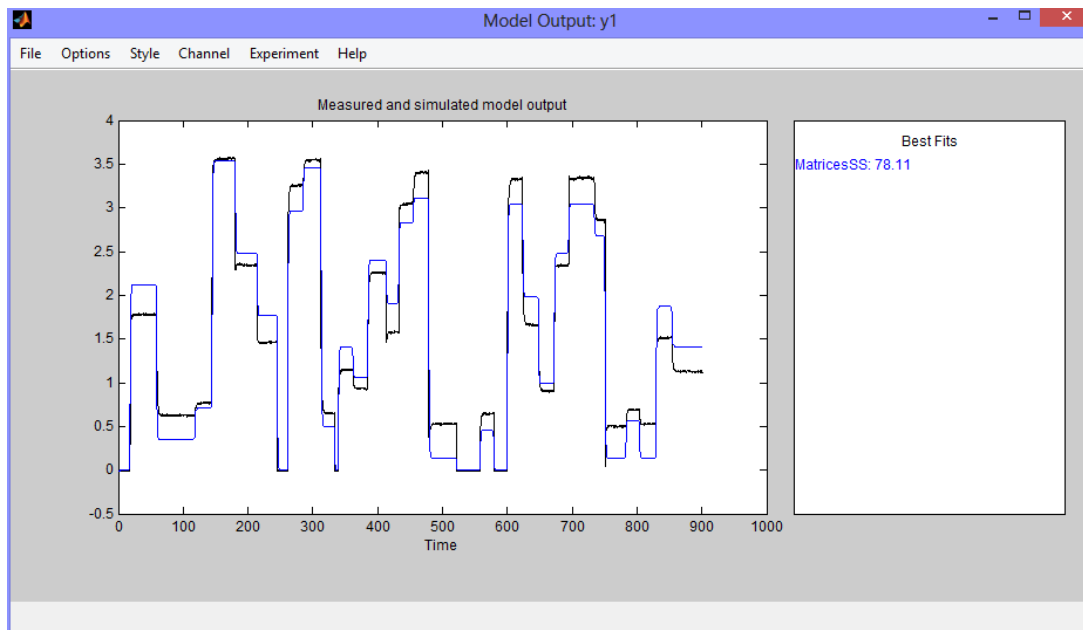


Figura 78 Similitud entre el Modelo Real y el Modelo Discreto en Espacio de Estados.

Se puede saber la aproximación del modelo identificado en espacio de estados con los datos de entrada y salida ingresados. En la Figura 78 se puede observar que el modelo discreto en Espacio de Estados tienen una aproximación de 78.11%.

- Cálculo de la Matriz de ganancia de realimentación de estado K mediante la Fórmula de Ackerman

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

Para encontrar las matrices K1 y K2 se ha utilizado la fórmula de Ackerman. Matlab® posee un comando que entrega la matriz de ganancia de realimentación de estado, el comando “acker” requiere las matrices aumentadas A y C, además de los polos del diseño para el comportamiento del controlador.

Para el diseño de los polos se tomó en cuenta el tiempo de asentamiento del sistema t_s y el factor de amortiguamiento ζ . Una vez obtenido estos parámetros se formó el polinomio característico de un sistema de segundo orden, el cual se muestra en la ecuación 19.

$$s^2 + 2\zeta\omega_n s + \omega_n^2 \quad (19)$$

Donde:

$$\zeta\omega_n = \sigma \quad (20)$$

$$\sigma = \frac{4}{t_s} \quad (21)$$

El factor de amortiguamiento ζ se puede escoger según el tiempo que se desee la estabilización y el sobre paso de la señal controlada, esto se puede observar en la Figura 79.

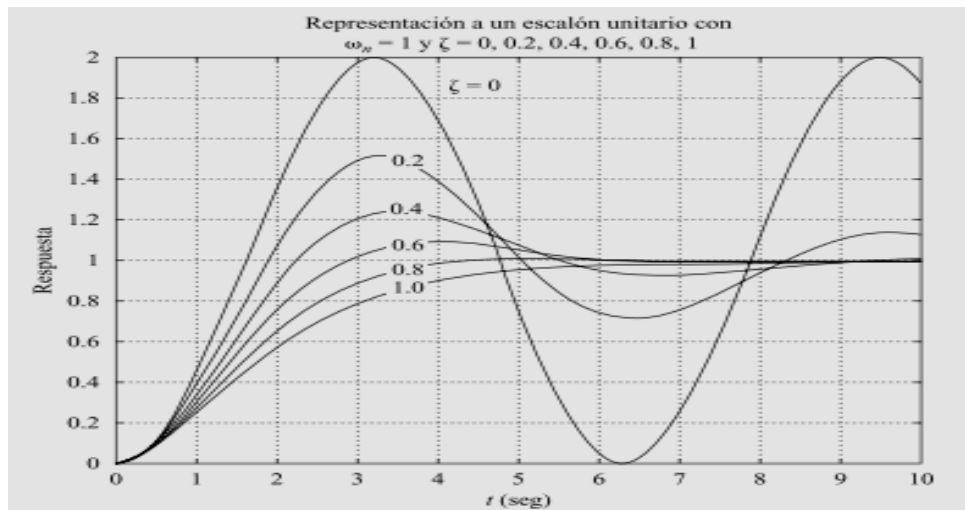


Figura 79 Factor de Amortiguamiento ζ [24].

Se procede a obtener las raíces del polinomio, estos factores son los polos que necesita el comando “acker”, para ello se usa otro comando llamado roots, en donde se coloca los coeficientes del polinomio y nos devuelve los polos del sistema.

- Cálculo de la Matriz de Ganancia de realimentación del observador Ke.

Para el cálculo de la matriz Ke se usó el filtro de Kalman. Para ello se debe tener las matrices en forma discreta.

Las Condiciones iniciales que se han colocado fueron las constantes de covarianza de la perturbación del proceso Q y la matriz de la perturbación de la medida R, ya que estas no deben cambiar en el tiempo.

En la Figura 80 se puede observar la simulación del controlador al dar diferentes cambios de referencia. En la Figura 80 se observan 3 señales, en la gráfica superior se encuentra la señal

controlada (color azul) con la referencia (color rojo) y en la gráfica inferior se observa la señal de control (color azul).

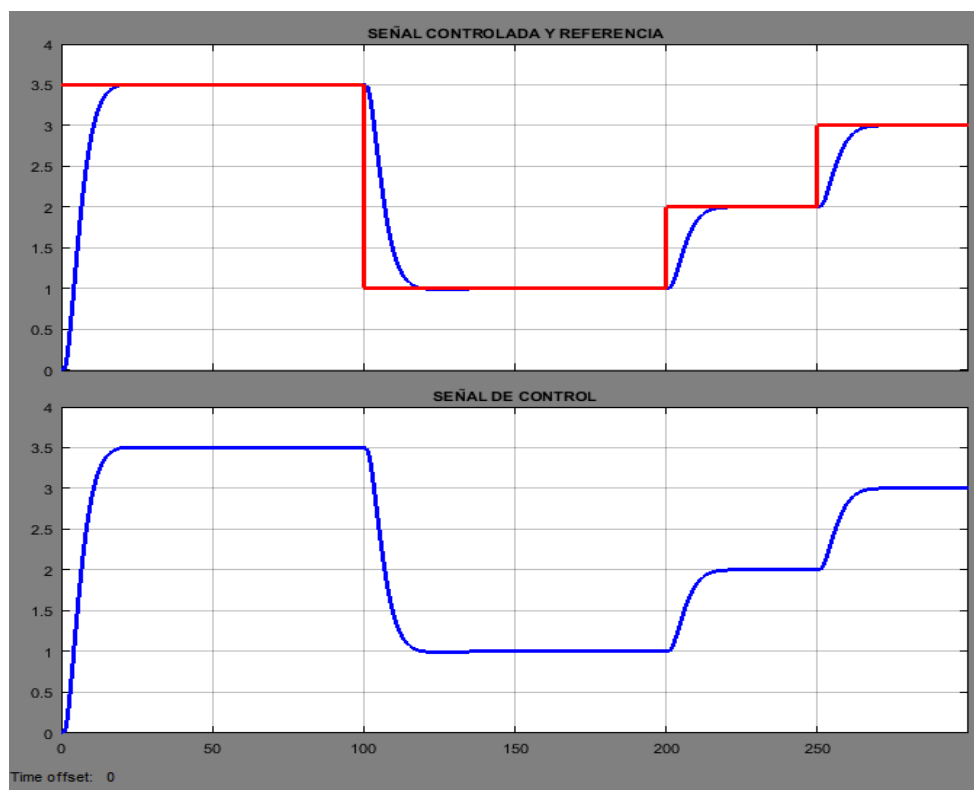


Figura 80 Resultado de la Simulación del controlador en espacio de estados en tiempo discreto.

4.3.3. Diseño del controlador Predictivo Basado en Modelo (MPC).

Para la implementación del controlador predictivo basado en modelo (MPC) se ha utilizado el software Simulink de Matlab®, en cual se ha realizado una interfaz que permitirá visualizar el comportamiento del controlador sobre el sistema.

Con la función de transferencia obtenida en la identificación del proceso de flujo se debe obtener las matrices A, B, C y D (véase Figura 81), que pueden ser calculadas con el comando “tf2ss” de Matlab® el mismo que está encargado de convertir una función de transferencia a espacio de estados.

Posteriormente las matrices A, B, C y D deben ser convertidas a matrices discretas Adis, Bdis, Cdis y Ddis (véase Figura 81) para lo cual se usa el comando “c2d” de Matlab®. También se calcula las matrices Φ y F lo cual se lo puede apreciar en el Apéndice B.

```

23     % Matrices del sistema en funcion de "s"
24     A=[-6.417805788201525,-11.256216270631;1,0];
25     B=[1;0];
26     C=[-1.1864986236,7.91297160054];
27     D=0;
28     % Matrices discretas del sistema
29     Adis=[0.991658,-3.03e-05;0.005974,0.9999]
30     Bdis=[0.0059596;1.761550e-05];
31     Cdis=[0.9714,0.00375];
32     Ddis=0;
    
```

Figura 81 Matrices del sistema

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

Se procede a generar la acción de control para la cual se requiere obtener las matrices Φ y F , las matrices del sistema en espacio de estados, la señal de entrada y la retroalimentación. Todo este proceso está realizado en un código de Matlab® [18] que se muestra en el Apéndice B.

Además esta estrategia de control requiere de la matriz de ganancia del observador, la cual es calculada mediante el filtro de Kalman.

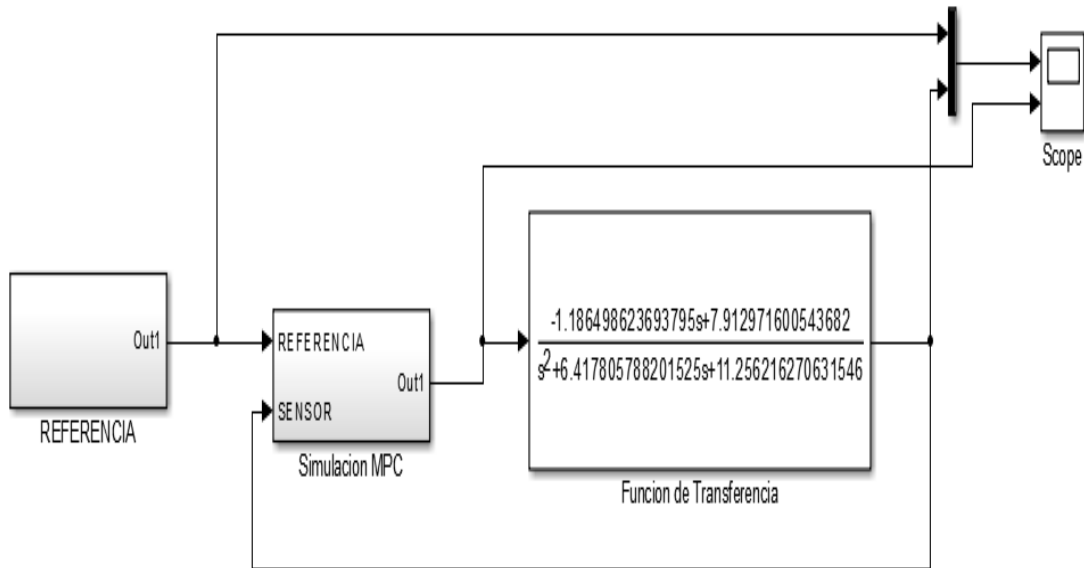


Figura 82 Simulación del MPC del sistema realizado en Simulink

En la Figura 82 se muestra el diseño de simulación del controlador predictivo basado en modelo para el proceso de flujo.

En el diseño que se realizó se encuentra un math script (véase Figura 83) el cual realizara el proceso matemático necesario para obtener la acción de control. Dentro del math script se encuentra el código que se ha explicado anteriormente. El código dentro del math script se ejecuta durante toda la simulación, en donde se realiza todos los cálculos matriciales para obtener la acción de control.

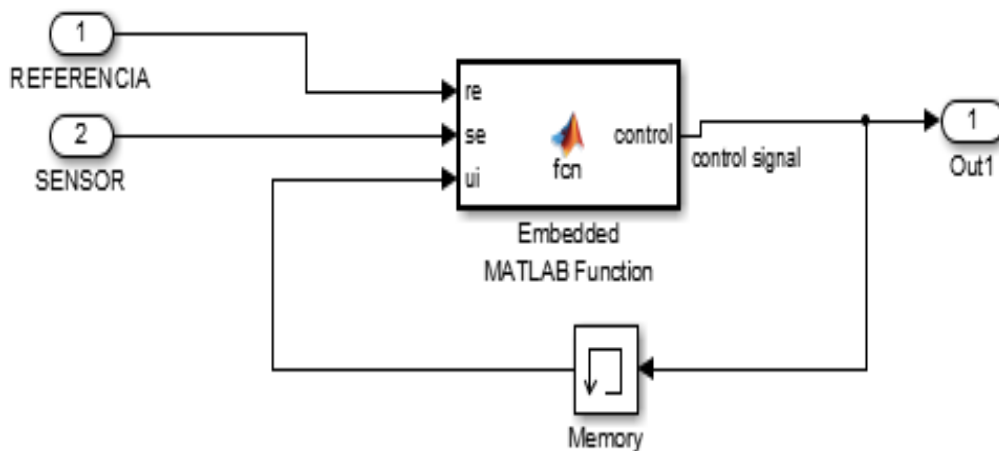


Figura 83 Math Script del controlador MPC

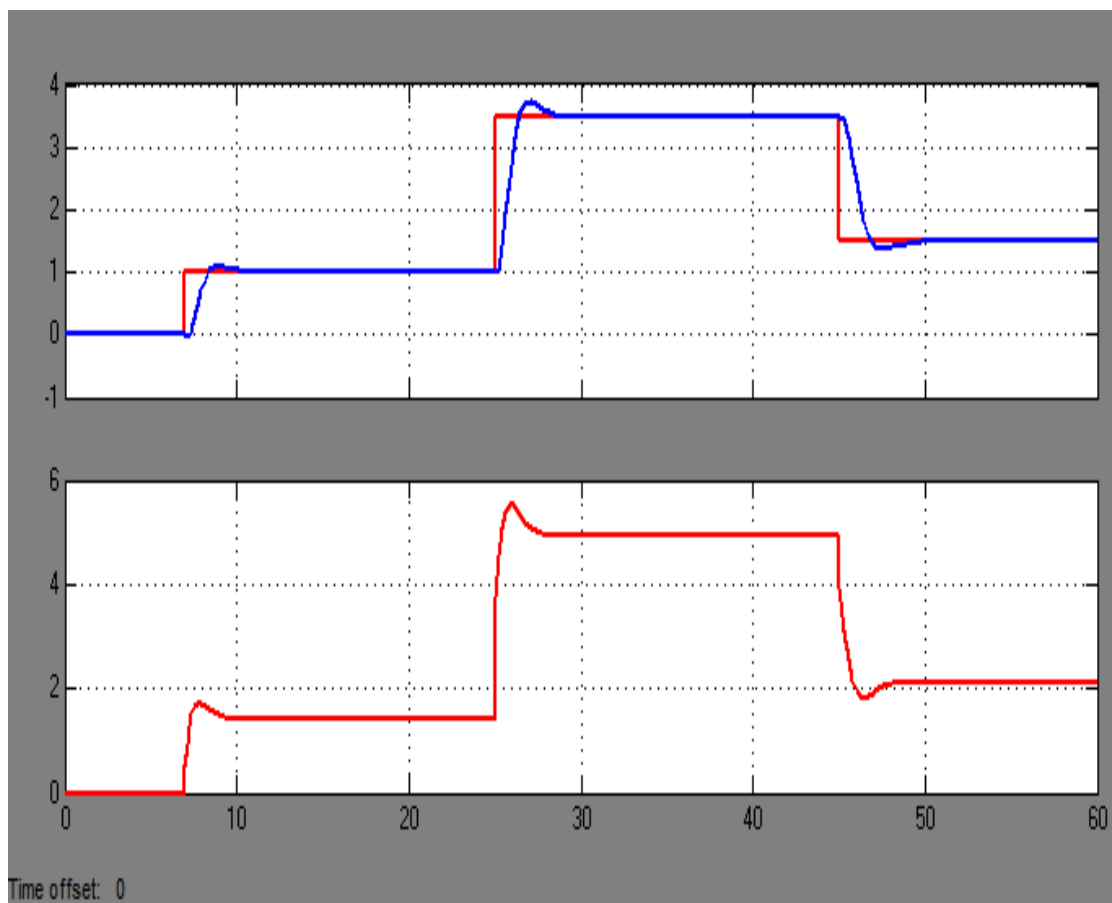


Figura 84 Resultado de la Simulación del controlador predictivo basado en modelo (MPC)

En la Figura 84 se muestra el resultado de la simulación realizada del control MPC, se presenta el comportamiento de dicho controlador a cambios de referencia generados. En donde la señal de color rojo (parte superior) representa la referencia y la señal de color rojo (parte inferior) indica la acción de control, mientras que la señal de color azul (parte superior) representa la respuesta del controlador.

4.3.4. Diseño del controlador PID Adaptativo.

El diseño del controlador se lo ha realizado en Simulink. Para realizar la simulación de la estrategia de control de un PID adaptativo se requiere la función de transferencia del proceso de flujo. Además se necesita un modelo de referencia el cual debe ser de tercer orden predefinido por condiciones iniciales del sistema. El modelo de referencia de tercer orden se presenta a continuación.

$$\frac{Km}{s^3 + A1s^2 + A2s + A3}$$

Donde:

$$Km = \frac{K \cdot \omega_2^2}{\tau}$$

En la Figura 85 se aprecia el diseño de simulación del controlador PID Adaptativo. Para el cálculo de los parámetros del modelo de referencia presentado anteriormente se debe considerar el tiempo de asentamiento del sistema t_s , el factor de amortiguamiento ζ , y el valor real de los polos según el tiempo de asentamiento por el criterio del 2% σ , como condiciones iniciales del sistema. Una

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

vez obtenido estos parámetros se establece el polinomio característico de un sistema de tercer orden. El código para obtener los parámetros de sintonía necesarios para implementar la estrategia de control PID Adaptativo se lo puede apreciar en el Apéndice C.

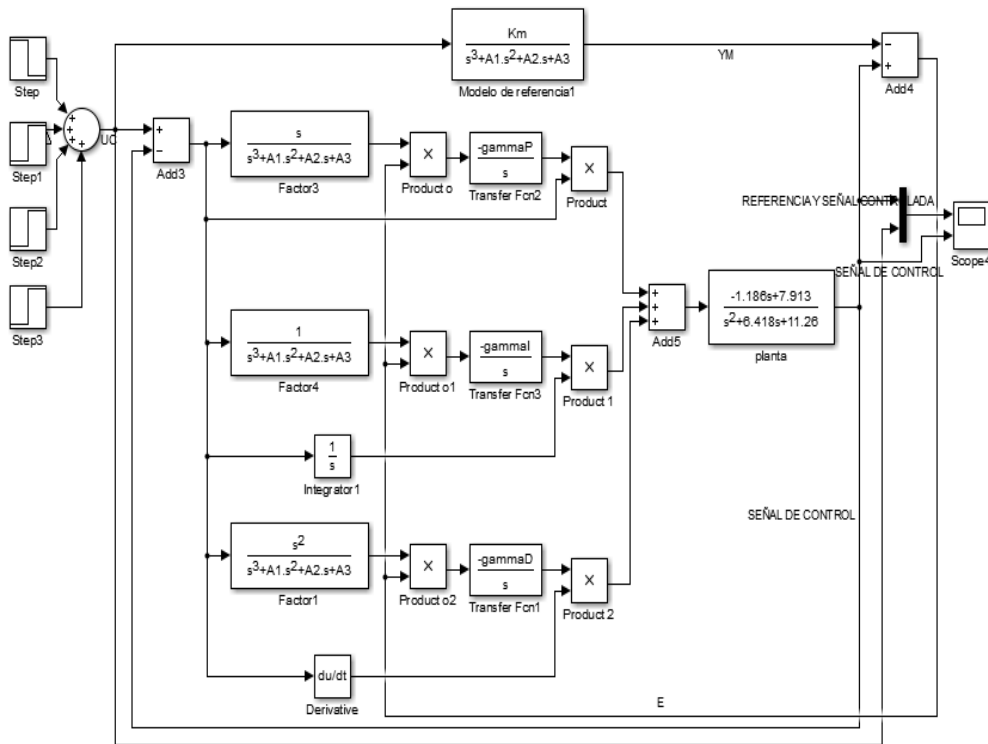


Figura 85 Diseño del controlador PID Adaptativo para el sistema.

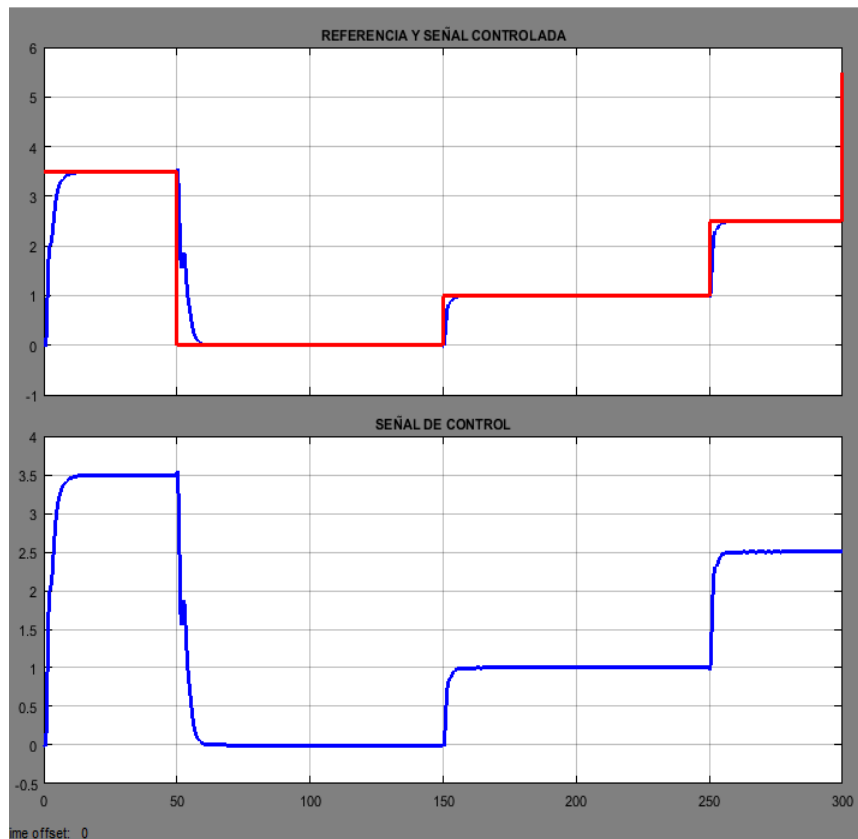


Figura 86 Resultado de la Simulación del controlador PID Adaptativo.

CAPITULO 4. SISTEMAS DE CONTROL AUTOMÁTICO

En la Figura 86 se puede observar la simulación del controlador al dar diferentes cambios de referencia. Se observan que existen 3 señales, en la parte superior se encuentra la señal controlada (azul) y la señal de referencia (rojo), mientras que en la parte inferior se observa la señal de control (azul).

CAPÍTULO 5

5. IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.

5.1. Implementación de Controladores.

5.1.1. Esquema de conexión entre la plataforma ArduPilot y proceso de flujo para la implementación de estrategias de control.

Para los controladores implementados en el proceso de flujo del laboratorio de control y automatización totalmente integrado de la Universidad Politécnica Salesiana se debe realizar la conexión que se muestra en la Figura 87.

En la Figura 87 se observa la forma que debe ser conectada la plataforma ArduPilot sobre el sistema para poder implementar las estrategias de control mencionadas anteriormente. Se configurara el posicionador electroneumático para que funcione con una señal de voltaje en un rango de 0 a 5 voltios, mientras que para la adquisición de los datos se debe ubicar una resistencia de 250Ω en serie con el sensor de flujo para así convertir la señal de corriente (4-20mA) a una señal de voltaje en un rango de 1 a 5 voltios.

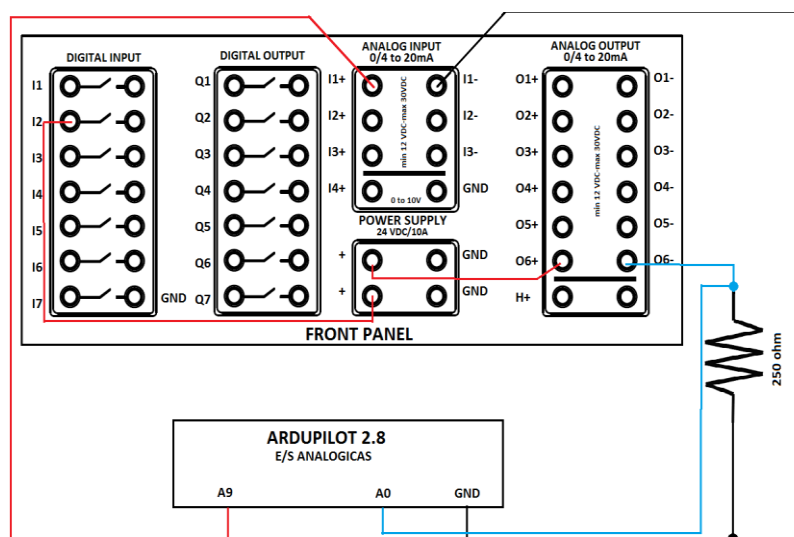


Figura 87 Conexión del dispositivo APM ArduPilot 2.8 con el Panel frontal del Sistema.

5.1.2. Implementación del Controlador PID

En la Figura 88 se presenta el diseño realizado en el software Matlab®-Simulink, el mismo diseño que es descargado sobre la plataforma ArduPilot para poder realizar el control. Para el diseño se utiliza de la librería ArduPilot los bloques “Analog Input”, “Analog Output” y “Led Blue”. Además en el diseño se ha implementado un filtro pasa bajo el cual permitirá eliminar el ruido que el propio sistema genera. En el MATLAB Function 1 se encuentra la conversión de la señal de 0-5 a una señal de 0-255 (véase Figura 89). Mientras que en el MATLAB Function 2 se encuentra la conversión de la señal de 0-1023 a una señal de 0-5 (véase Figura 90). Dentro del subsistema con el nombre “Diseño del PID” se encuentra el diseño del controlador PID, el mismo ha sido configurado con los parámetros encontrados anteriormente (véase Figura 91). Los datos del comportamiento del sistema con la implementación del controlador PID se visualizan en

CAPITULO 5. IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.

Simulink mediante un “Scope”. El bloque “LED Blue” sirve para avisar al operario que el diseño se ha cargado sobre la plataforma y esta puede empezar a funcionar.

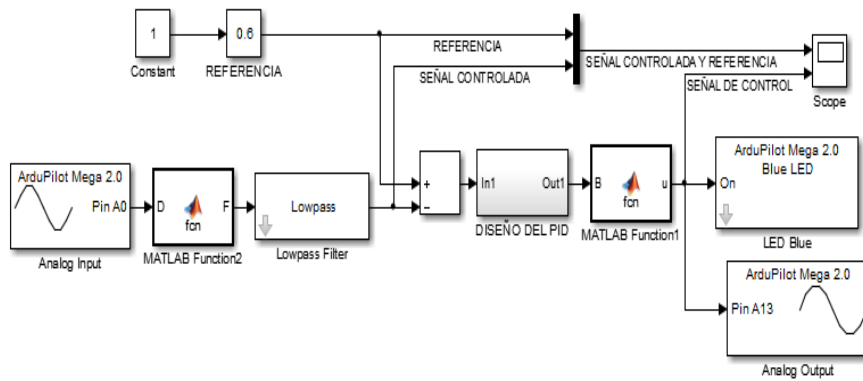


Figura 88 Controlador PID diseñado en Simulink para la Planta B

```

MATLAB Function1
1 function u = fcn(B)
2 - P=double(B);
3 - if P >= 5
4 -     u=int16(255);
5 - else
6 -     u=int16((P*255)/5);
7 - end
8 - if u <= 0
9 -     u=int16(0);
10 - end
    
```

Figura 89 Conversión para el bloque Analog Output

```

MATLAB Function2
1 function F = fcn(D)
2 - E=double(D);
3 -
4 - if E >= 1023
5 -     F=5;
6 - else
7 -     F=(E*5)/1023;
8 -     F=F-1;
9 - end
    
```

Figura 90 Conversión para el bloque Analog Input

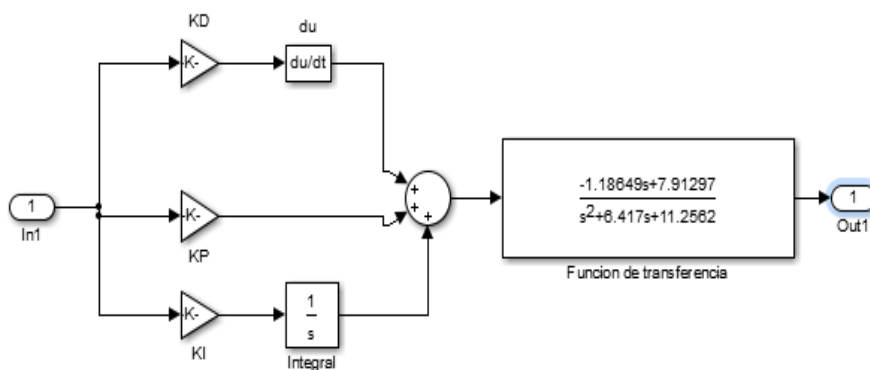


Figura 91 PID implementado

CAPITULO 5. IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.

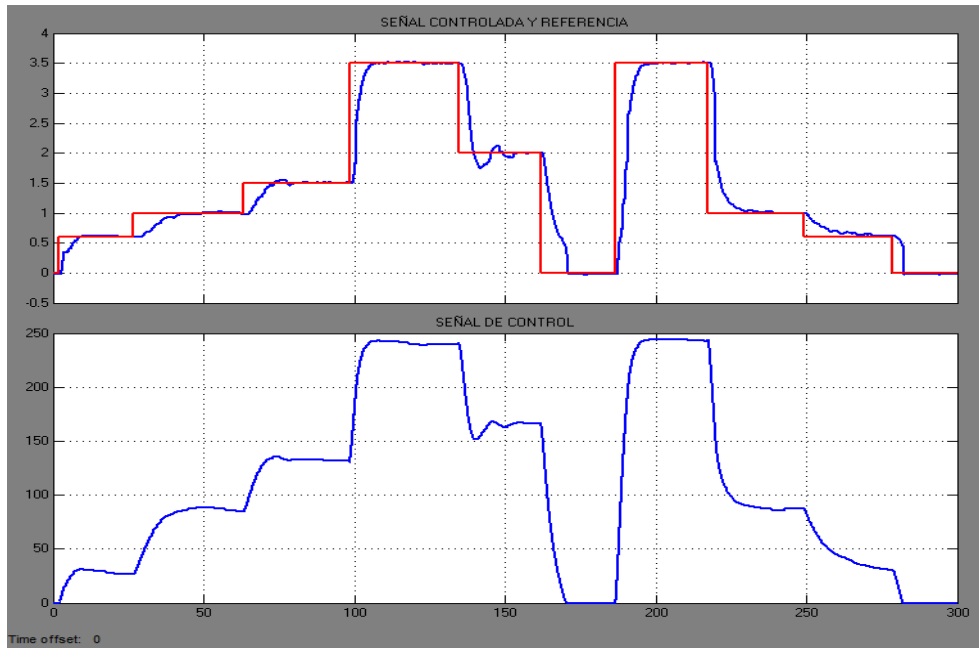


Figura 92 Comportamiento a cambios de referencia del sistema

Los resultados obtenidos de la implementación del controlador PID con la plataforma ArduPilot se puede observar en la Figura 92, Figura 93.

En la Figura 92 se presenta el comportamiento del sistema a cambios de referencia realizados. La señal de color rojo representa la referencia y la de color azul representa la señal controlada, se observa que no existe un gran sobrepaso de la señal controlada al momento de realizar un cambio de referencia. También se puede visualizar la acción de control del sistema en cada uno de los cambios de referencia que se han generado.

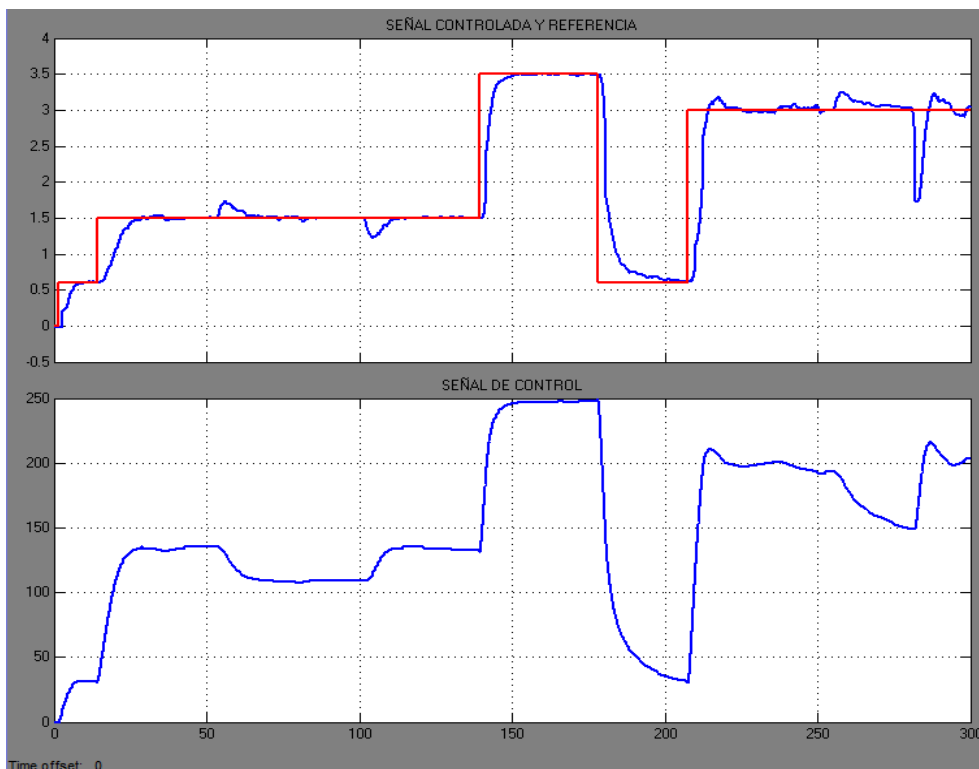


Figura 93 Comportamiento a cambios de referencia y perturbaciones en el sistema

CAPITULO 5. IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.

En la Figura 93 se presenta unos cambios de referencia, pero a más de ellos también se ha insertado una perturbación en el rango entre los 50 y 100 segundos de simulación, y otra perturbación mayor en el rango entre 250 y 300 segundos. En ambos casos se puede observar como la acción de control actúa al momento de que se ha generado la perturbación y al momento que se retire la perturbación.

5.1.3. Implementación del Controlador por Realimentación de Estados.

En la Figura 94 se presenta el diseño del controlador en espacio de estados realizado en el software Matlab®-Simulink. Este diseño debe ser descargado sobre la plataforma como se había explicado en la sección anterior. Para la implementación de esta estrategia de control al igual que en el controlador PID se utiliza los mismos bloques con la diferencia que dentro del subsistema llamado “Espacio de Estados” se encuentra el diseño del controlador por realimentación de estados (véase Figura 95).

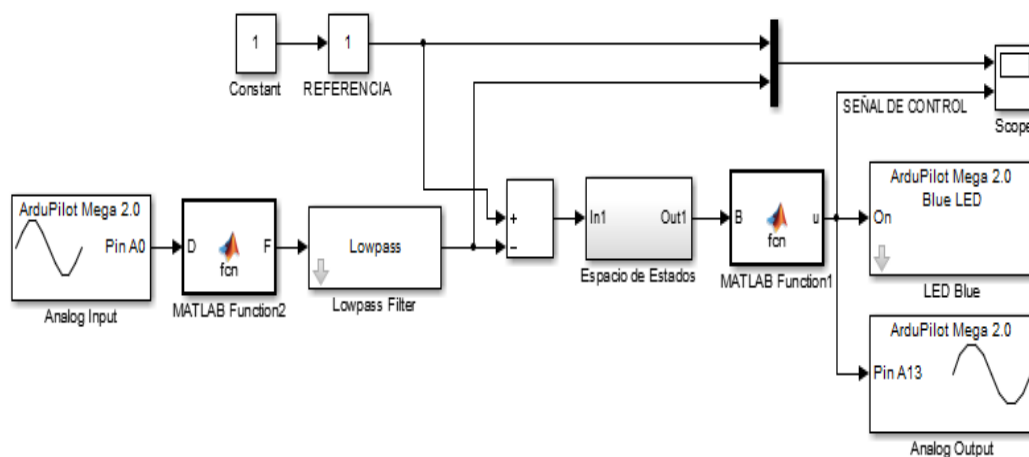


Figura 94 Controlador en Espacio de Estados en tiempo discreto diseñado en Simulink para la proceso de flujo.

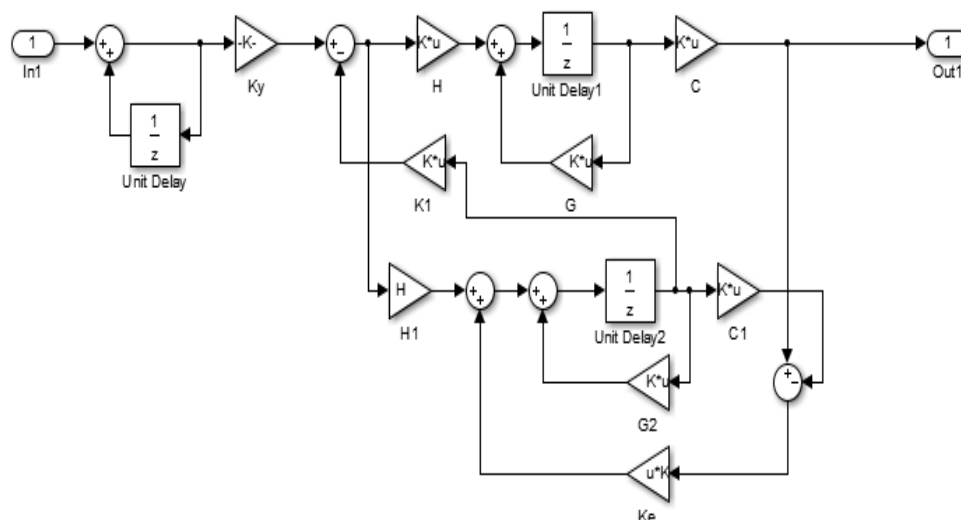


Figura 95 Controlador por realimentación de estados implementado

Los resultados obtenidos de la implementación del controlador en Espacio de Estados en el proceso de flujo con la plataforma ArduPilot se puede observar en la Figura 96 y Figura 97.

CAPITULO 5. IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.

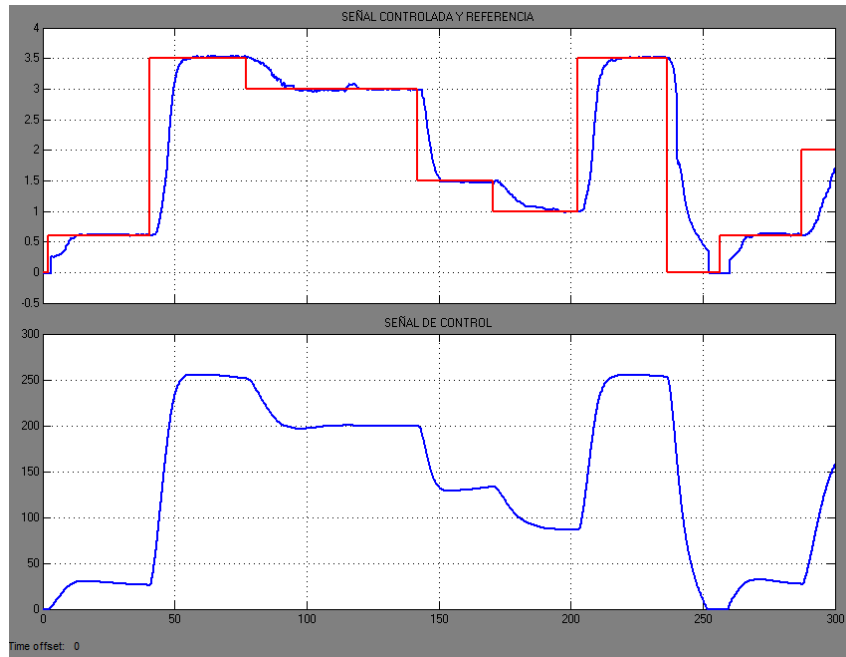


Figura 96 Comportamiento a cambios de referencia del sistema

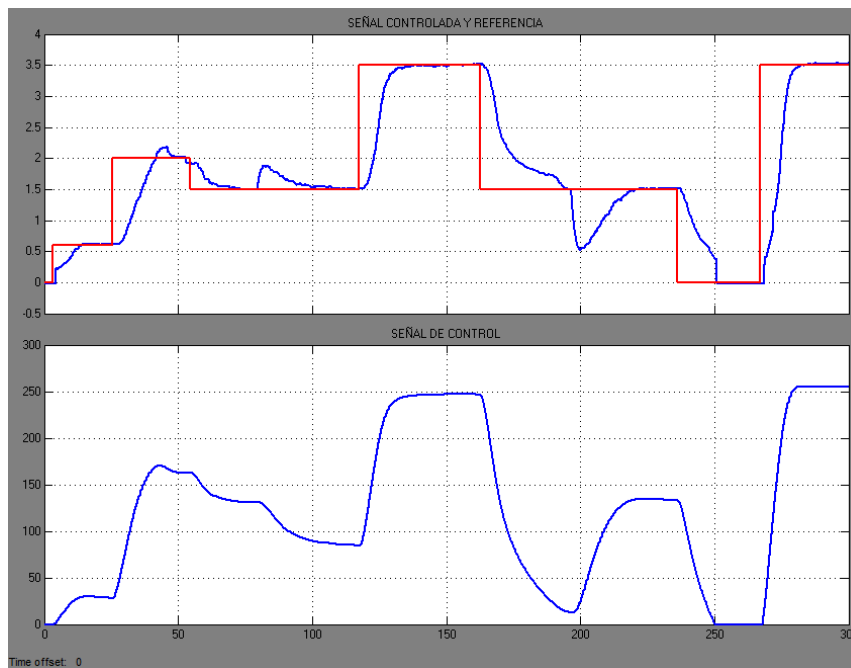


Figura 97 Comportamiento a cambios de referencia y perturbaciones en el sistema

En la Figura 96 se presenta el comportamiento del sistema a cambios de referencia realizados. La señal de color rojo representa la referencia y la de color azul representa la señal controlada, se observa que no existe un gran sobrepaso de la señal controlada al momento de cambiar de referencia. También se puede visualizar la acción de control del sistema en cada uno de los cambios de referencia.

En la Figura 97 se ha realizado cambios en la señal de referencia, también se ha inyectado perturbaciones entre los 75 y 150 segundos, mientras que otra perturbación mayor se realizó en el rango entre 150 y 250 segundos. En ambos casos se puede observar como la acción de control actúa al momento de que se ha generado la perturbación y al momento que se retire la misma.

5.1.4. Implementación del Controlador predictivo basado en modelo (MPC).

En la Figura 98 se muestra el diseño del controlador MPC realizado en Simulink el cual será descargado sobre la plataforma ArduPilot, este diseño usa la librería propia de la plataforma y además dispone del math script que realizara la formulación matemática que ejecutara la plataforma para obtener la señal de control.

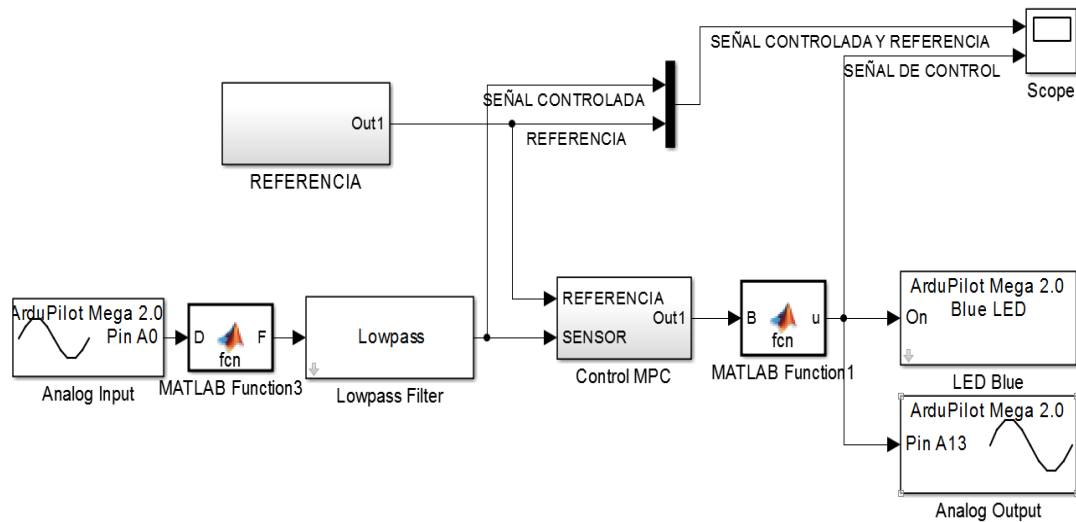


Figura 98 Implementación del controlador MPC con la plataforma ArduPilot

En la Figura 99, Figura 100 y Figura 101 se muestra los resultados obtenidos de la implementación del controlador MPC aplicados sobre el sistema con diferentes parámetros de sintonía, la Figura 99, Figura 100 y Figura 101 muestran cambios de referencia generados y se visualiza el comportamiento del sistema a estos cambios de referencia. La señal de color rojo representa la referencia que se ha generado y la señal azul de la parte superior de la gráfica representa la señal controlada, mientras que la señal azul de la parte inferior de la gráfica representa la acción de control.

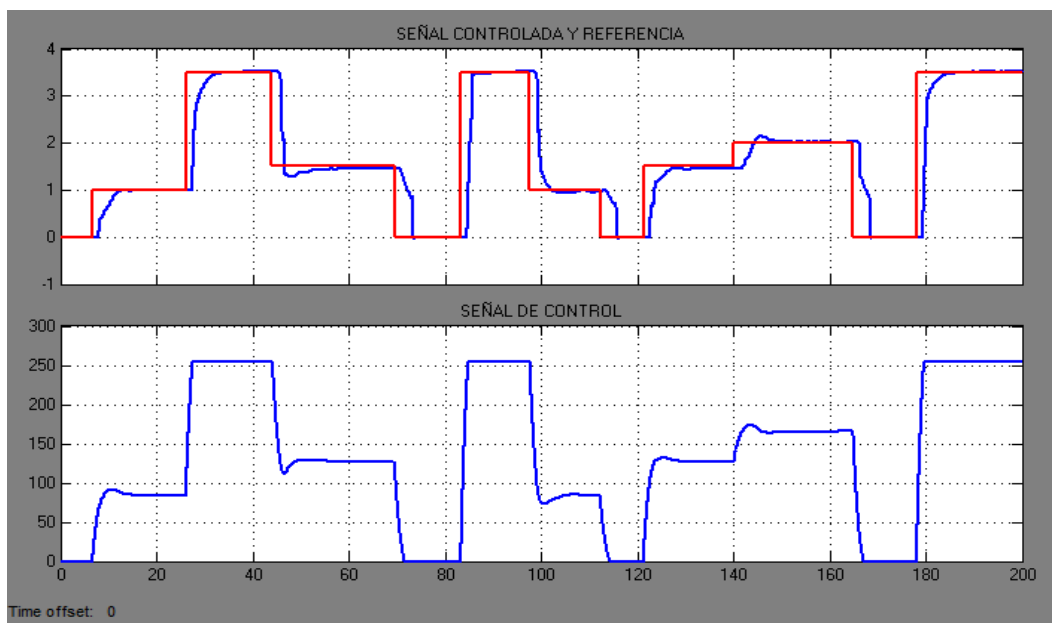


Figura 99 Comportamiento del sistema con la implementación del controlador predictivo basado en modelo, con los parámetros de sintonización $N_c=2$, $N_p=3$ y $rw=50$

CAPITULO 5. IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.

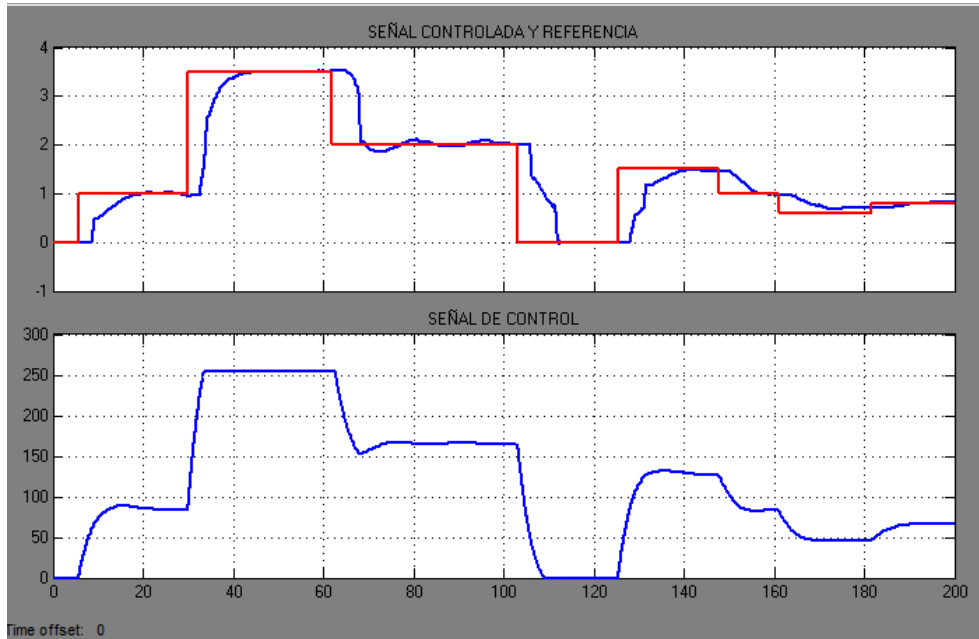


Figura 100 Comportamiento del sistema con la implementación del controlador predictivo basado en modelo, con los parámetros de sintonización $N_c=2$, $N_p=3$ y $rw=150$

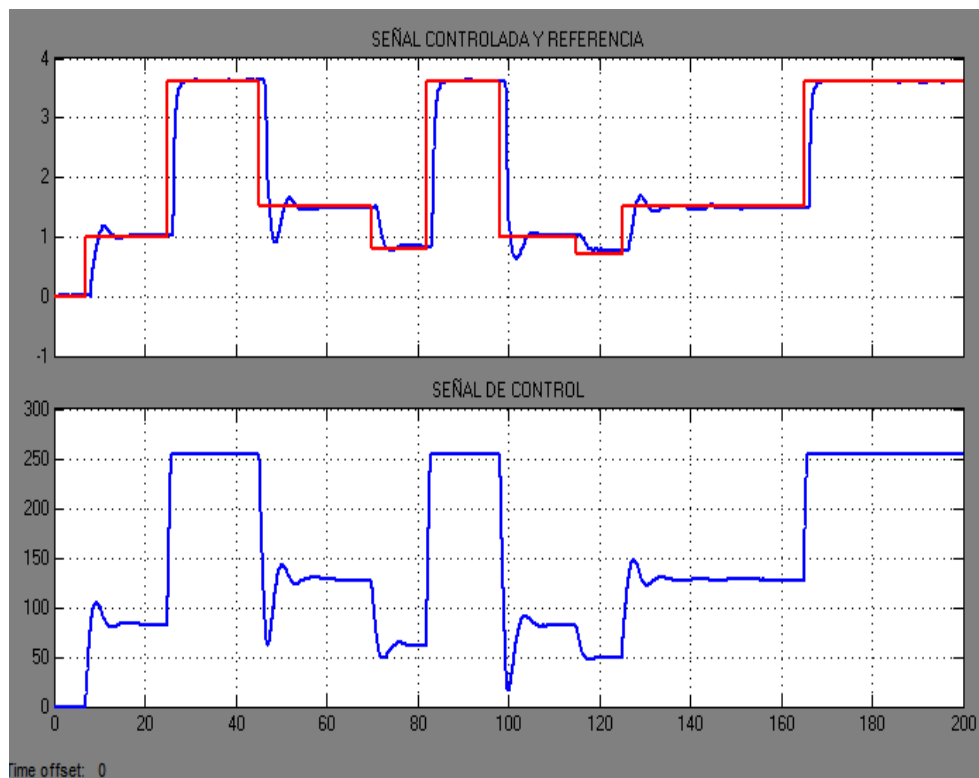


Figura 101 Comportamiento del sistema con la implementación del controlador predictivo basado en modelo, con los parámetros de sintonización $N_c=2$, $N_p=4$ y $rw=80$

5.1.5. Implementación del Controlador PID Adaptativo.

En la Figura 102 se presenta el diseño realizado en el software Matlab®-Simulink, el mismo diseño que es descargado sobre la plataforma ArduPilot. Las constantes necesarias para esta estrategia de control se han calculado anteriormente con la ayuda del programa que se había explicado en la sección anterior.

CAPITULO 5. IMPLEMENTACIÓN DE ESTRATEGIAS Y EVALUACIÓN DE LA PLATAFORMA.

Los resultados obtenidos de la implementación del controlador PID con la plataforma ArduPilot se puede observar en la Figura 103.

En la Figura 103 se presenta el comportamiento del sistema a cambios de referencia realizados. La señal de color rojo representa la referencia y la de color azul representa la señal controlada, se observa que no existe un gran sobrepaso de la señal controlada al momento de cambiar de referencia. También se puede observar que el controlador se va adaptando ya que la primera vez que tiene un escalón de subida o de bajada responde de manera lenta, pero a los siguientes cambios tienen un tiempo de establecimiento más rápido que el inicial.

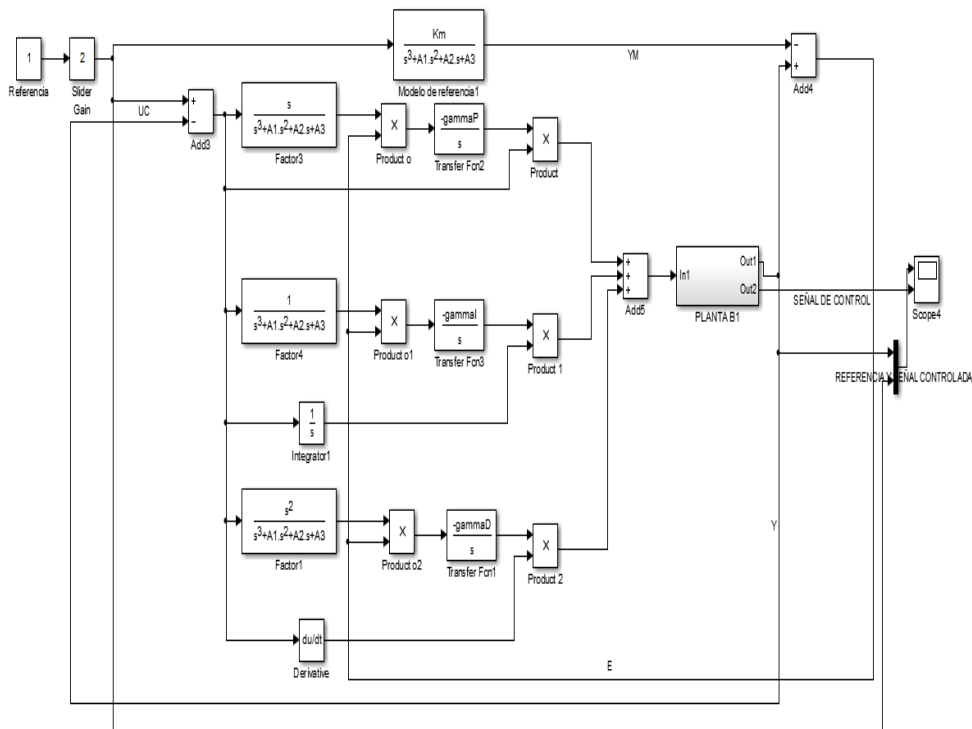


Figura 102 Controlador PID Adaptativo diseñado en Simulink para el proceso de flujo.

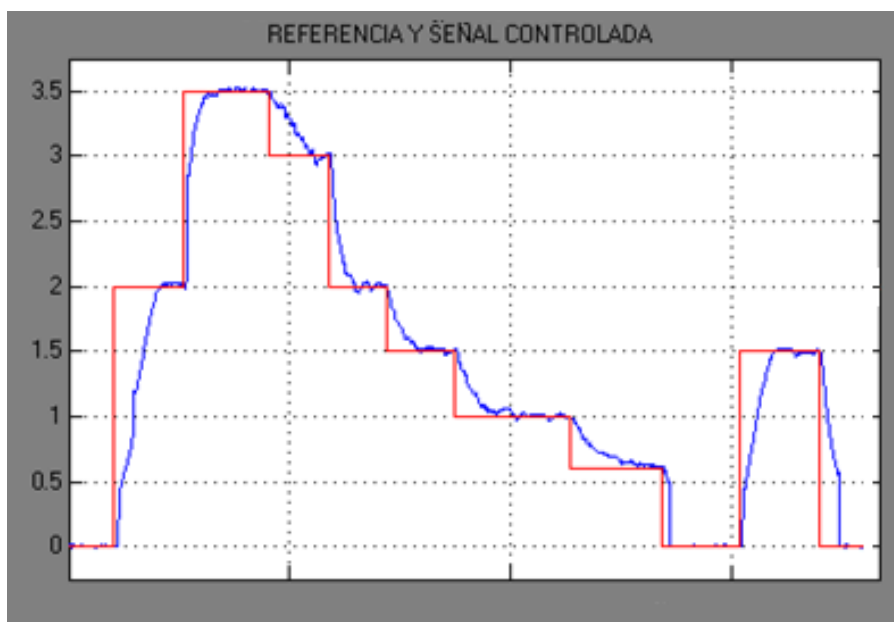


Figura 103 Comportamiento a cambios de referencia del sistema

5.2. Análisis del Funcionamiento de los Controladores

A lo largo de esta tesis se ha desarrollado cuatro diferentes estrategias de control las cuales son:

- Controlador PID
- Controlador en Espacio de Estados en tiempo discreto
- Controlador PID Adaptativo
- Controlador Predictivo Basado en Modelo (MPC)

Para un mejor análisis se ha realizado una comparación entre distintas estrategias de control:

- Controlador PID y Controlador Predictivo Basado en Modelo (MPC).
- Controlador en Espacio de Estados y Controlador PID Adaptativo.

5.2.1. Comparación entre el controlador PID y MPC.

Se ha enviado una misma referencia a las dos estrategias de control en un tiempo de 200 segundos de simulación.

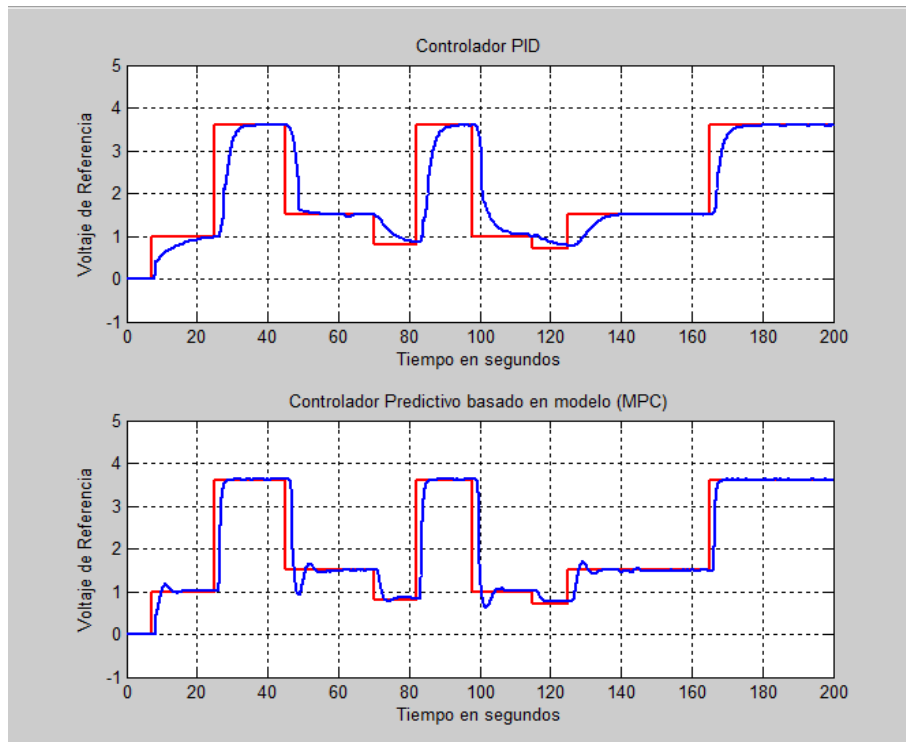


Figura 104 Comparación entre un controlador PID y un MPC

En la Figura 104 se puede apreciar que el controlador predictivo basado en modelo a pesar de tener un sobrepaso (M_p) mayor que el controlador PID, tiende a estabilizarse en un tiempo menor que el controlador PID.

El controlador MPC es más rápido debido a que mediante el horizonte de predicción es capaz de predecir muestras futuras en su salida.

5.2.1. Comparación entre el controlador en Espacio de Estados y el controlador PID Adaptativo.

Se ha enviado una misma referencia a las dos estrategias de control en un tiempo de 200 segundos.

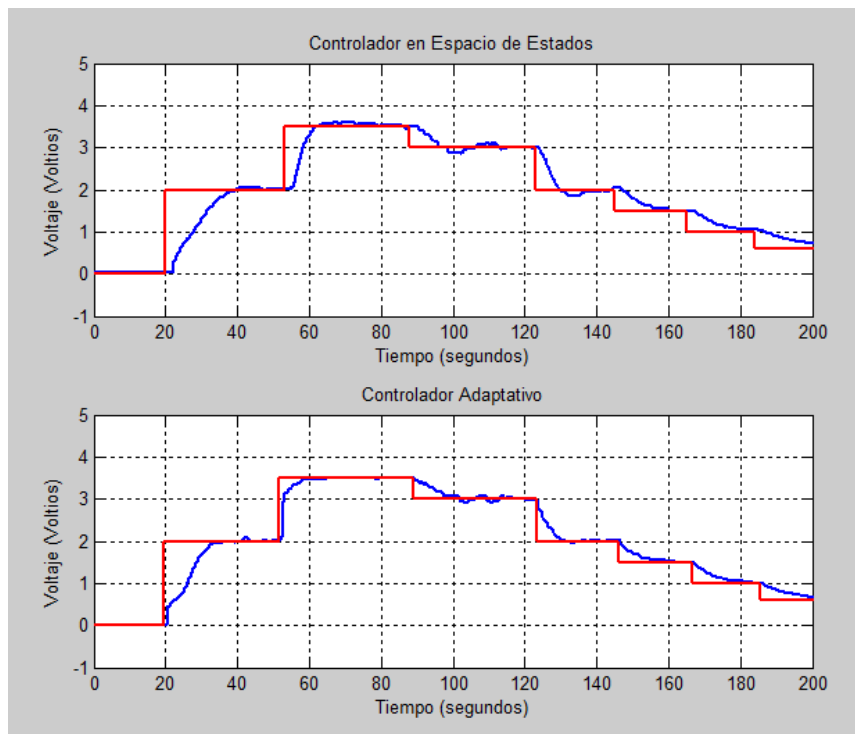


Figura 105 Comparación entre un Controlador en Espacios de Estados Discreto y un Controlador PID Adaptativo.

En la Figura 105 se puede apreciar que el controlador PID Adaptativo tiende a estabilizarse en un tiempo menor que el controlador discreto en Espacio de Estados.

El controlador PID Adaptativo es más rápido luego de adaptarse al primer cambio de referencia en subida y al primer cambio de referencia de bajada, esto se debe a que el controlador se adapta a las condiciones externas y ajusta el modelo conforme este funciona.

5.3. Beneficios del uso de la plataforma ArduPilot en sistemas de control

Los beneficios que dispone la plataforma ArduPilot en el uso de sistemas de control se los presenta a continuación:

- Uno de los grandes beneficios de esta plataforma es la capacidad de trabajar en tiempo real, lo que implica que esta plataforma puede ser usada en cualquier de sistema que se requiera controlar.
- Esta plataforma ArduPilot es compatible con los sistemas operativos Windows XP, Windows 7 y Windows 8.
- Esta plataforma tiene como beneficio que dispone de su propia librería la misma que debe ser instalada y configurada sobre el software Matlab®-Simulink para utilizarlos en las diferentes estrategias de control que se pretenda realizar. Además esta plataforma es capaz de soportar cualquier bloque de Simulink como por ejemplo bloques de filtros, lo que en otras plataformas es difícil de lograr.
- Otro de los beneficios de utilizar la plataforma ArduPilot en sistemas de control es que esta plataforma dispone de un procesador en el cual una vez que se haya descargado el diseño realizado en Simulink no utiliza el procesador de un ordenador para realizar la formulación matemática que requiera la estrategia de control que se haya implementado.
- Con la ayuda de esta plataforma se puede reemplazar módulos de control de tal manera que se podría reducir el costo, ya que los módulos de control son muy costosos y a más de esto los módulos necesitan de un software específico para ser utilizados.

CAPÍTULO 6

6. CONCLUSIONES Y RECOMENDACIONES

En la actualidad existen diversas plataformas embebidas capaces de soportar diseños realizados en Matlab®-Simulink. Una de estas plataformas es ArduPilot, que es una tecnología de bajo costo y también es compatible con software de computadora como Windows XP, Windows 7 y Windows 8. Una ventaja de trabajar con ArduPilot es que esta plataforma genera su propia librería lo que facilita la comunicación entre el ordenador y la plataforma.

La plataforma analizada en este proyecto de titulación dispone de diversos puertos de comunicación como I2C, USB, telemetría y GPS, estos puertos pueden ser controlados desde Simulink con la ayuda de su librería. La plataforma es capaz de trabajar en tiempo real con lo que se podrá implementar esta plataforma embebida en cualquier sistema de control.

La identificación del sistema de control con el que se trabajó en este proyecto de titulación fue mediante “ident” de Matlab®, en donde primero se generó una señal con escalones de diferentes amplitudes y se adquirió los datos del comportamiento del sistema frente a esta señal generada. El comando “ident” permite obtener el modelo matemático del sistema de una manera eficaz.

Matlab®-Simulink es un software que permite diseñar estrategias de control avanzado como por ejemplo el controlador por realimentación de estados, el controlador PID Adaptativo y el controlador predictivo basado en modelo, este tipo de controladores requiere de un proceso matemático complejo el cual puede ser desarrollado en Simulink mediante el uso del bloque Math Script .

Para la implementación de las estrategias de control de este proyecto de titulación se realiza una comunicación USB entre el ordenador y la plataforma ArduPilot. Con la ayuda de este tipo de comunicación se logra descargar los diseños realizados en el software Simulink sobre la plataforma embebida.

La primera estrategia de control que se probó mediante la plataforma Ardupilot fue el controlador PID, al ser Matlab ® un software que usa lenguaje de cálculo técnico y posee algoritmos matemáticos para desarrollo e implementación de estrategias de control, se han usado dos herramientas conocidas y sencillas para la sintonización de controladores PID. La primera estrategia que se uso fue el comando “sisotool”, el cual nos entrega la ecuación del controlador PID, es importante mencionar que partiendo de esta ecuación se encontraran las constantes KP, KI y KD necesarias para la sintonización del mismo, el segundo comando que se uso fue “pidtool” esta herramienta podría ser más sencilla y eficaz que la primera, ya que esta nos entrega según los requerimientos de diseño del controlador los parámetros de sintonización KP, KI y KD. En las dos herramientas o comando utilizados es necesario tener la función de transferencia en “s” ya que se implementó un controlador PID en tiempo continuo.

El controlador en Espacio de Estados en tiempo discreto fue realizado identificando sus matrices en espacio de estados mediante el comando “ident”, para encontrar la matriz de ganancia de realimentación de estados K se usó la fórmula de Ackerman partiendo del cálculos de los polos del sistema, utilizando parámetros de diseño como el tiempo de asentamiento t_s y el factor de amortiguamiento ζ , para encontrar la matriz de ganancia de realimentación del observador K_e se usó el Filtro de Kalman, tomando como constantes invariantes en el tiempo la covarianza de la perturbación del proceso Q y la matriz de la perturbación de la medida R. El controlador obtenido respondió acorde a las condiciones de diseño, por cual fue posible obtener su respuesta en régimen permanente con distintos cambios de referencias y se pudo observar su acción de control al introducir perturbaciones en el sistema.

CONCLUSIONES Y RECOMENDACIONES

El controlador PID adaptativo fue diseñado partiendo de un modelo de referencia de segundo orden controlado por un PID normal, las especificaciones del controlador se diseñaron según un tiempo de asentamiento t_s y un factor de amortiguamiento ζ . Se obtuvo como resultado un controlador que se adapta al entorno de funcionamiento, es decir cuando se le presenta el primer cambio de referencia en subida y en bajada tarda en llegar al régimen permanente o estabilización, mientras que para los siguientes cambios de referencia reduce su tiempo de estabilización. La plataforma respondió de manera correcta a pesar de la complejidad de cálculos matemáticos que este controlador poseía.

El controlador MPC requiere de cálculos matemáticos matriciales complejos, estos cálculos se deben estar ejecutando a cada instante de la simulación, los parámetros que requiere el MPC para los cálculos son las matrices del sistema, el horizonte de predicción, el horizonte de control y el parámetro de sintonía r_w . Con ello si se implementa la estrategia de control con un tiempo de muestreo de 6ms estos cálculos deben realizarse en un tiempo menor para obtener la acción de control correspondiente., de tal manera se puede evaluar la eficiencia de la plataforma con los resultados que se obtienen de la implementación del controlador. Debido a que la plataforma ArduPilot dispone de dos procesadores ATmega2560 y ATMEGA32U2 puede realizar los cálculos matriciales complejos que presenta esta estrategia de control.

Como trabajo futuro se sugiere la utilización de esta plataforma para el control de vehículos aéreos, los mismos que pueden ser controlados con estrategias de control avanzado como las que se detallan en este proyecto. Otro desafío importante sería evaluar el desempeño del controlador MPC con restricciones, donde para cada iteración se debe resolver un problema de optimización complejo.

APÉNDICE A

```

clear all;
clc;
T=0.006; %tiempo de simulación
load ('Coeficientes.mat')
TF = ss(m.A,m.B,m.C,m.D);
TFd = c2d(TF,T,'zoh');
G = m.A;
H = m.B;
C = m.C;
D = m.D;
%CONTROLABLE
M=ctrb(G,H);
cont=rank(M) %el sistema es de estado completamente controlable
%OBSERVABLE
N=obsv(G,C);
obs=rank(N) %el sistema es de estado completamente observable
%Modelo aumentado
tamC = size(C);
Am = [G (zeros(1,tamC(2)))';C*G 1];
Bm = [H;C*H];
Cm = [zeros(1,tamC(2)) 1];
%Calculo Polos
ts=5; %Tiempo de asentamiento del sistema
sigma=4/ts; % Valor real de los polos según el criterio de 2%
z=0.9; % Factor de amortiguamiento
wn=sigma/z; %Frecuencia natural
ecuacion=[1 2*(z)*(wn) wn^2]; %Ecuación característica
polos=roots(ecuacion);% Obtención de los polos
P2=[polos(1),polos(2),0];
P22=[polos(1),polos(2)];
%GANAN K1 y Ky
K2 = acker(Am',Cm',P2)
K22 = acker(G',C',P22)
K1 = [K2(1) K2(2)];
Ky=K2(3);
%OBSERVADOR CON FILTRO DE KALMAN
Qkal=0.000001;
Rkal=1;
[kalmf,L,v,M,Z] = kalman (TFd,Qkal,Rkal);
kalmf = kalmf(1,:);
Ke=M';

```

APÉNDICE B

```

%MPC SIN RESTRICCIONES PARA PLANTA SISO.
%SERVOMECANISMO PARA CONTROL DE POSICIÓN
%ALGORITMO CON INTEGRADOR EMBEBIDO (MODELO AUMENTADO)
%MODELO DE LA PLANTA IGUAL AL MODELO DE PREDICCIÓN
%AUTOR:
%
%           ING. JULIO ZAMBRANO A.
%*****
%PARAMETROS DE SINTONIA
%*****
Kmpc = [0.90964;-2.047]; %Vector de ganancia del observador
Np=4; Nc=2;
N_sim=120;
rw=150;
re=re;
BarRs=ones(Np,1)*re;
u1=zeros(1,N_sim);
y1=zeros(1,N_sim);
u=ui; % u(k-1) =0
y=se;
%MODELO ORIGINAL
%CAMBIO DE CONTINUO A DISCRETO.
%PARAMETROS DEL SISTEMA.
A=[-6.417805788201525,-11.256216270631;1,0];
B=[1;0];
C=[-1.1864986236,7.91297160054];
D=0;
Acont=A;
Bcont=B;
Ccont=C;
Dcont=D;
Adis=[0.991658890022048,-3.035270448926868e-
05;0.005974941828596,0.999999908814768]
Bdis=[0.005974941828596;1.794984878661550e-05];
Cdis=[0.971400000000000,0.003750000000000];
Ddis=0;
% *****
% *****
% PARAMETROS DEL SISTEMA.
Acontc=A;
Bcontc=B;
Ccontc=C;
Dcontc=D;
Delta_t=0.1;
Adisc=[0.991658890022048,-3.035270448926868e-
05;0.005974941828596,0.999999908814768];
Bdisc=[0.005974941828596;1.794984878661550e-05];
Cdisc=[0.971400000000000,0.003750000000000];
Ddisc=0;
%MODELO AUMENTADO CON INTEGRADOR EMBEBIDO.
[m1,n1]=size(Cdisc);
[n1,n_in]=size(Bdisc);
A_e=eye(n1+m1,n1+m1);
A_e(1:n1,1:n1)=Adisc;
A_e(n1+1:n1+m1,1:n1)=Cdisc*Adisc;
B_e=zeros(n1+m1,n_in);
B_e(1:n1,:)=Bdisc;
B_e(n1+1:n1+m1,:)=Cdisc*Bdisc;
C_e=zeros(m1,n1+m1);

```

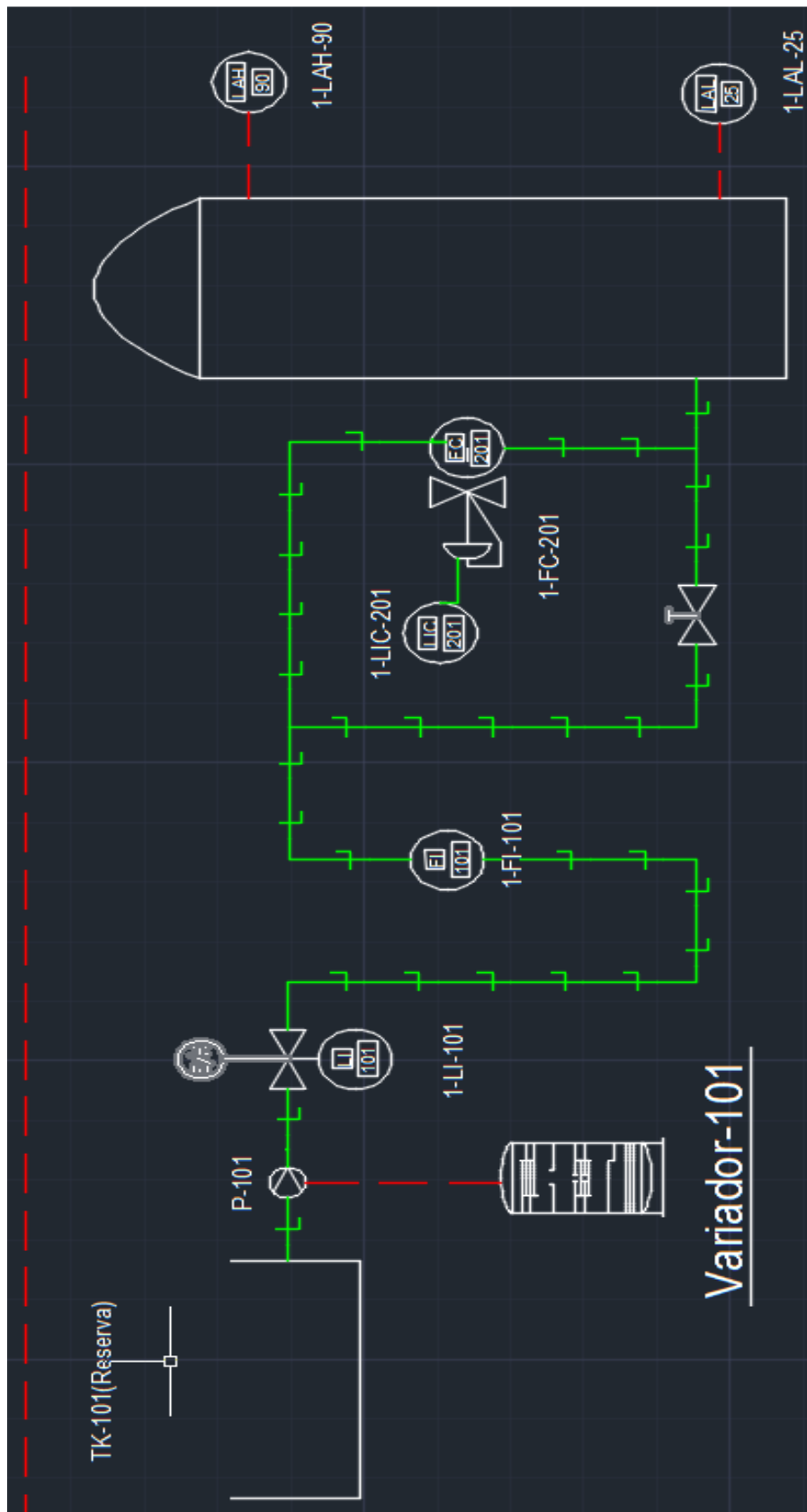
APÉNDICES

```
C_e(:,n1+1:n1+m1)=eye(m1,m1);
%*****
%CALCULO DE LAS MATRICES FI Y F
n=n1+m1;
h=zeros(Np,3);
F=zeros(Np,3);
h(1,:)=C_e;
F(1,:)=C_e*A_e;
for kk=2:Np
h(kk,:)=h(kk-1,:)*A_e;
F(kk,:)= F(kk-1,:)*A_e;
end
v=h*B_e;
Phi=zeros(Np,Nc); %declare the dimension of Phi
Phi(:,1)=v; % first column of Phi
for i=2:Nc
Phi(:,i)=[zeros(i-1,1);v(1:Np-i+1,1)]; %Toeplitz matrix
end
%*****
R=rw*eye(Nc,Nc);
xm=zeros(n-1,1);
Xf=zeros(n,1);
for kk=1:N_sim;
DeltaU=inv(Phi'*Phi+R)*Phi'*(BarRs-F*Xf);
deltau=DeltaU(1,1);
u=u+deltau;
u1(kk)=u;
y1(kk)=y;
xm_old=xm;
xm=Adis*xm+Bdis*u + Kmpc*(y - C*xm);
y=Cdis*xm;
Xf=[xm-xm_old;y];
control = u1(kk);
end
control=u1(kk);
```


APÉNDICE C

```
clc;
%Condiciones Iniciales
ts=1; %Tiempo de asentamiento del sistema
sigma=4/ts; %valor real de los polos según el criterio de 2%
z=0.9; %Factor de Amortiguamiento
wn=sigma/z;
tao=1/(z*wn);
k=1;
syms s;
%Ecuacion del Sistema de Tercer Orden
ecuacion=(s+(1/tao))*((s^2)+2*z*wn*s+(wn^2));
polinomio=expand(ecuacion);
vector=sym2poly(polinomio);
A1=vector(2);
A2=vector(3);
A3=vector(4);
Km=(k*wn^2)/tao;
gammaP=12.2;
gammaI=2.75;
gammaD=1
```

APÉNDICE D



REFERENCIAS BIBLIOGRAFICAS

- [1] H. Halvorsen, «Introduction to Simulink,» University College of southeast Norway, 2016.
- [2] A. D. Team, «Archived: APM 2.5 and 2.6 Overview,» 2016. [En línea]. Available: <http://www.ardupilot.org/copter/docs/common-apm25-and-26-overview.html>. [Último acceso: Agosto 2016].
- [3] R. Hartley, «APM2 Simulink Blockset,» MATLAB Central, 13 Noviembre 2012. [En línea]. Available: <http://www.mathworks.com/MATLABcentral/fileexchange/39037-apm2simulink-blockset>. [Último acceso: Agosto 2016].
- [4] burkert FLUID CONTROL SYSTEMS, «Type 8792, 8793 Electropneumatic positioner and process controller,» 2009. [En línea]. Available: <https://www.burkert.com/en/Media/plm/MAN/MA/MA8792-Standard-EU-EN.pdf>. [Último acceso: Noviembre 2016].
- [5] burkert FLUID CONTROL SYSTEMS, «FLOW TRANSMITTER 8035,» 2009. [En línea]. Available: http://www.iprocessmart.com/images/Burkert/US_M_8035_Flow_Transmitter.pdf. [Último acceso: Noviembre 2016].
- [6] J. R. Orellana, «LABORATORIO DE SISTEMAS DE CONTROL DISCRETO. Soporte de MATLAB a plataformas embebidas.Robert Hartley, APM2 Simulink Blockset,» 2012.
- [7] D. C. Fernández, «Programación de Arduino mediante MATLAB/Simulink. Aplicación al control de velocidad de motores BLDC,» 2016.
- [8] E. Granados, «BeagleBoard Support from Simulink,» 2015. [En línea]. Available: <http://es.mathworks.com/hardware-support/beagleboard.html>. [Último acceso: Septiembre 2016].
- [9] M. Kim, S. Malack, N. Gautier y K. Nunna, «LEGO MINDSTORMS NXT Support from Simulink,» 2015. [En línea]. Available: <http://es.mathworks.com/hardware-support/lego-mindstorms-simulink.html>. [Último acceso: Septiembre 2016].
- [10] B. McKay, «Raspberry PI Programming using Simulink,» 2015. [En línea]. Available: <http://es.mathworks.com/hardware-support/raspberry-pi-simulink.html>. [Último acceso: Septiembre 2016].
- [11] STMicroelectronics, «STM32 Embedded Target for MATLAB and Simulink,» 2015. [En línea]. Available: http://es.mathworks.com/products/connections/product_detail/product_75144.html.
- [12] LV-MaxSonar, «High Performance Sonar Range Finder MB1000, MB1010, MB1020, MB1030, MB1040,» 2015. [En línea]. Available:

REFERENCIAS BIBLIOGRÁFICAS

- http://www.maxbotix.com/documents/Lv-MaxSonar-EZ_Darasheet.pdf. [Último acceso: Septiembre 2016].
- [13] J. Martínez Ruiz, «ArduPilot y Simulink,» 2015. [En línea]. Available: <http://jujomaruiit.blogspot.com/2015/08/ardupilot-y-simulink.html>. [Último acceso: Septiembre 2015].
- [14] J. S. Escobar Hinojosa y G. E. Dominguez Crespo, «Estudio y elaboración de un manual de prácticas de la planta de control de procesos “A” del laboratorio de control y automatización totalmente integrado de la Universidad Politécnica Salesiana Cuenca,» Cuenca, 2015.
- [15] Danfoss, «Guía de Diseño VTL® 2800.,» [En línea]. Available: <http://www.danfoss.com/Spain>. [Último acceso: 20 Enero 2017].
- [16] B. C. Kuo, "Sistemas de Control Automático", Prentice Hall, 1996.
- [17] C. Jara Ortiz y J. Maldonado Achig, «Diseño e implementación de un control PID y MPC aplicado al posicionamiento lineal de un sistema neumático en bucle cerrado,» Cuenca, 2015.
- [18] X. Ochoa Parra, «Elaboración de un módulo guía para el desarrollo de prácticas en el laboratorio de control de la Universidad Politécnica Salesiana,» Cuenca, 2016.
- [19] K. Ogata, «Análisis en el espacio de estado,» de *Sistemas de Control en Tiempo Discreto*, Mexico, L. Griffiths, pp. 290-376.
- [20] K. Ogata, «Ubicación de Polos y diseño de Observadores,» de *Sistemas de Control en Tiempo Discreto*, 2da Ed. Mexico, L. Griffiths, pp. 377-516..
- [21] TIT. Anibal, «Capítulo 5 Observadores de Estado,» 2007.
- [22] J. Zambrano y A. Gonzáles, «Implementación de un algoritmo de control predictivo en espacio de estados sobre una plataforma de simulación desarrollada en Matlab,» *Ingenius*, p. 10, 2013.
- [23] S. Aguilar, A. Fuentes, F. Arteaga, J. Castellanos y Joel Rivas, «IMPLEMENTACIÓN DEL ALGORITMO PID ADAPTATIVO PARA CONTROLAR TEMPERATURA A TRAVÉS DEL COMPUTADOR USANDO ALGORITMOS GENÉTICOS».
- [24] A. Anuradha, «Adaptive Control Lecture 7 Adaptive PID Control,» [En línea]. Available: <http://aaclab.mit.edu/material/lect/lecture7.pdf>. [Último acceso: Diciembre 2016].
- [25] R. Duraisamy y S. Dakshinamurthy, «An Adaptive Optimisation scheme for controlling air flow process with satisfactory transient performance,» *Maejo International Journal of Science and Technology*, p. 14, 2010.