UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:

INGENIEROS ELECTRÓNICOS

TEMA:

IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL ÓPTIMO ESTOCÁSTICO EN LA NUBE MEDIANTE HERRAMIENTAS DE PROGRAMACIÓN IOT PARA UN MOTOR DC

AUTORES:

JUAN PAULO ESCOBAR CALDERÓN

PATRICIO GEOVANNY SULCA CHIGUANO

TUTOR:

CARLOS GERMÁN PILLAJO ANGOS

Quito, septiembre de 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Juan Paulo Escobar Calderón con documento de identificación N° 172560079-3 y Patricio Geovanny Sulca Chiguano con documento de identificación N° 172295893-9, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación intitulado: **IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL ÓPTIMO ESTOCÁSTICO EN LA NUBE MEDIANTE HERRAMIENTAS DE PROGRAMACIÓN IOT PARA UN MOTOR DC**, mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, 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 nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

.....

Juan Paulo Escobar Calderón

Cédula: 172560079-3

.....

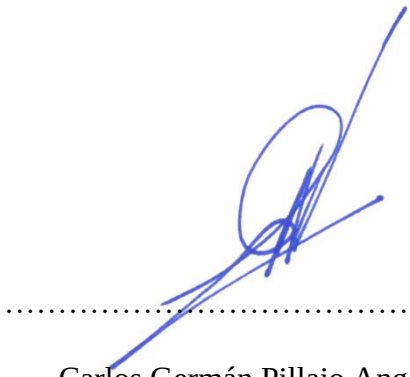
Patricio Geovanny Sulca Chiguano

Cédula: 172295893-9

Quito, septiembre de 2021

DECLARACIÓN DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación, IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL ÓPTIMO ESTOCÁSTICO EN LA NUBE MEDIANTE HERRAMIENTAS DE PROGRAMACIÓN IOT PARA UN MOTOR DC, realizado por Juan Paulo Escobar Calderón y Patricio Geovanny Sulca Chiguano, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerado como trabajo final de titulación.



…….

Carlos Germán Pillajo Angos

Cédula: 170925511-9

Quito, septiembre de 2021

DEDICATORIA

Por guiarme y demostrarme que con esfuerzo se puede lograr cada una de las metas que me proponga, este trabajo de titulación se dedicó a mis padres que siempre me brindan su compresión, cariño y amor, por ustedes puedo decir hoy en día que lo logramos y seguiré avanzando para que sigan sintiéndose orgullosos.

Los amo padres

Juan

A mis padres por brindarme todo el apoyo incondicional durante toda mi vida, siempre presentes en mis logros, guiándome con el ejemplo, brindándome su paciencia, cariño, amor, sus sabios consejos que me forjaron la persona que soy hoy en día, bendiciéndome y motivándome diariamente para alcanzar mis sueños, anhelos, y enseñándome a ser perseverante, por eso dedico este trabajo a mi madre amada Marianita y mi padre amado Teofilito.

Los amo con todo mi corazón.

Patricio

AGRADECIMIENTOS

Le doy gracias a Dios, al esfuerzo y apoyo que cada día hicieron mis padres para que cumpla mi sueño de ser profesional, mi hermano que con sus palabras me motivo para seguir adelante a pesar de todas las dificultades que se presentaban, a cada uno de mis familiares que con su apoyo y aliento me daban fuerzas para continuar con este camino, enamorada desde los inicios de esta aspiración y todas las personas que me acompañaron a lo largo de estos 5 años de carrera, puedo manifestar con mucha felicidad que esta meta esta cumplida.

Juan

Primero a Dios, por darme salud, vida y permitir disfrutar de mi familia, a mis padres que sin el apoyo de ellos no pudiera estar escribiendo estas líneas, siendo ellos la fuerza para continuar y en los momentos más difíciles ayudarme a no desmayar y seguir adelante, a mis hermanas que de una u otra manera me apoyaron a culminar con este proyecto, a mi amada que siempre tuvo el tiempo para apoyarme, ayudarme en el día a día, con consejos, palabras de aliento y su apoyo incondicional, a mis familiares y amigos que a lo largo de este tiempo estuvieron apoyándome ya se con una palabra de ánimo. Les agradezco a todo de corazón por ayudarme a concluir con esta meta, que me llena de mucho orgullo y felicidad.

Patricio

INDICE DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

En este trabajo de titulación se presenta el estudio relacionado con la industria 4.0, implica nuevas técnicas de operación y producción mediante el uso de tecnología inteligente. Estas pueden solucionar problemas que presentan las industrias, por lo que se propone desarrollar algoritmos de control óptimos estocásticos en la nube para el control de un motor DC, mediante herramientas de programación IoT. Para establecer los parámetros de funcionamiento de la planta, se obtuvo la ecuación del sistema mediante un muestreo y estimación de datos con la ayuda del software Matlab**,** para posteriormente desarrollar los algoritmos de control partiendo de las matrices de estado que se originan de la función de transferencia; esto proporciona las características para la programación de los controles en la herramienta Node-Red con el análisis previo de configuración del protocolo MQQT y la herramienta Remoteit. Finalizando con la comparativa de operación de cada uno de los algoritmos de control aplicados, por medio de pruebas de comunicación y tiempos de respuesta que se puede apreciar en cualquier dispositivo inteligente mediante el uso de la aplicación RemoteRED, validando el control a través de la nube y permitiendo establecer la operatividad de los controladores PID, LQG y LQI. Se destaca que el controlador que tiene el mejor desempeño en este tipo de planta es el LQI, puesto que presenta un alto grado de fiabilidad al momento de someter la planta a perturbaciones. Debido a que muestra un rápido tiempo de establecimiento y disminuye los sobreimpulsos en la señal de control del sistema.

ABSTRACT

In this degree work, the study related to industry 4.0 is presented, it involves new operation and production techniques through the use of intelligent technology. These can solve problems presented by industries, so it is proposed to develop optimal stochastic control algorithms in the cloud for the control of a DC motor, using IoT programming tools. To establish the operating parameters of the plant, the equation of the system was obtained by sampling and estimating data with the help of Matlab software, to later develop the control algorithms starting from the state matrices that originate from the function of transfer; This provides the features for programming the controls in the Node-Red tool with the previous analysis of the MQQT protocol configuration and the Remoteit tool. Ending with the operation comparison of each of the applied control algorithms, through communication tests and response times that can be seen in any smart device through the use of the RemoteRED application, validating the control through the cloud. and allowing to establish the operability of the PID, LQG and LQI controllers. It is highlighted that the controller that has the best performance in this type of plant is the LQI, since it presents a high degree of reliability when subjecting the plant to disturbances. Because it shows a fast-settling time and decreases overshoots in the system control signal.

INTRODUCCIÓN

En la actualidad las WNCS están solucionando varios problemas de comunicación en los procesos manufactureros de las compañías, por la forma de la transmisión de datos y las diferentes ventajas que estas proporcionan a los usuarios (Yuan, Xia, Yang, & Yuan, 2018). Por lo que el estudio de los algoritmos de control óptimos representa una programación dinámica con respecto al espacio de estados, ya que estos ejecutan técnicas dentro de su estructura para ajustar los parámetros de control (Díaz, Armesto, & Sala, 2019). En el Ecuador la tendencia de utilizar algoritmos de control básicos para las empresas manufactureras ha sido un estilo que se ha llevado por muchos años, donde en la mayoría de los casos se ha empleado la transmisión de datos por cables, sin tomar en cuenta la eficiencia que se puede obtener al implementar controles inalámbricos en los procesos. Para el presente documento se aborda la evaluación de los algoritmos de control en la nube, los cuales nos permiten conocer de una mejor manera la fiabilidad que presentan en la red, para lo cual se realiza un estudio comparativo en la aplicación y susceptibilidad ante variaciones de los parámetros del sistema, para conseguir conclusiones que permitan disponer de criterios adecuados para la elección del control, estos se encuentran en dependencia de los requerimientos dinámicos que presente la planta, así como de los medios técnicos de que se disponga.

CAPÍTULO 1

ANTECEDENTES

1.1 Planteamiento del problema

En la actualidad existen aplicaciones de investigación en redes locales dadas desde hace pocos años tales como: “Sistema de control en red inalámbrica WNCS implementado al control de voltaje para un generador DC didáctico”, la cual posee un control PID en un servidor local alojado en la placa de desarrollo raspberry pi 3 (Terán & Narváez, 2018). Por otra parte, en la tesis: “Control robusto de un motor AC mediante WNCS”, se presentan dos tipos de control como LQG y Fuzzy que manejan una base de datos en un servidor aplicado en la raspberry pi (Balseca & Durán, 2019).

De acuerdo con las investigaciones antes mencionadas es necesario avanzar en las aplicaciones de control en la nube por las diferentes ventajas que proporcionan al implementar tanto en software como en hardware, por lo cual surge la pregunta: ¿Cuál de los algoritmos presenta mejor fiabilidad para el control del motor DC a través de la nube?

1.2 Tema

IMPLEMENTACIÓN DE ALGORITMOS DE CONTROL ÓPTIMO ESTOCÁSTICO EN LA NUBE MEDIANTE HERRAMIENTAS DE PROGRAMACIÓN IOT PARA UN MOTOR DC.

1.3 Justificación

Con la evolución de la tecnología es importante brindar un control más eficiente en los procesos industriales, debido a que la manipulación y monitoreo de las diferentes actividades que se cumplen día a día se las puede optimizar mediante la aplicación de controles estocásticos en la nube mediante el uso de WNCS que facilitan el control y optimiza el uso de recursos (Chung, Ibrahim, Asirvadam, Saad, & Hassan, 2015).

Con esta aplicación de algoritmos de control en la nube, se busca una solución para simplificar el modelo de control actual que proporciona una alternativa al hardware utilizado y la forma de operación de cualquier tipo de planta, acceso a la información de manera remota desde cualquier dispositivo móvil, también un incremento en el control de inventarios y minimizar los costos de aplicación.

Este proyecto es importante por los diferentes resultados que se presentan en el cuadro comparativo que determine el control con la más alta fiabilidad, entre el algoritmo de control PID y variantes del control LQG, como LQG/LTR y LQI aplicados en la arquitectura de la nube.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar algoritmos de control óptimos estocásticos en la nube para el control de un motor DC, mediante herramientas de programación IoT.

1.4.2 Objetivos específicos

- Estudiar el funcionamiento de los algoritmos de control básico y óptimo, mediante la lectura de artículos científicos, tesis y libros de investigación de campo, para la implementación de los controladores en la nube.
- Categorizar el motor DC y su conexión a la nube, mediante la adquisición de datos en Matlab, determinando las funciones matemáticas que describen el sistema dinámico.
- Definir los algoritmos de control tanto básico como óptimos estocásticos y robustos, por medio de herramientas de programación IoT, para el control del motor DC.
- Comparar el funcionamiento de los algoritmos, por medio de pruebas de comunicación y tiempos de respuesta, para la validación del control a través de la nube.

1.5 Metodología

- El método exploratorio se aplica para el estudio del funcionamiento de los algoritmos de control óptimo, las diferentes herramientas IoT y de la aplicación de máquinas virtuales en Google Cloud.
- Para adquirir datos mediante el software Matlab se utilizará el método experimental, lo cual determinará las funciones matemáticas que describen el sistema dinámico.
- Mediante el método inductivo, se desarrollarán los algoritmos de control en la herramienta Node-Red con el análisis previo de configuración del protocolo MQQT, sirviendo de apoyo para la eficiencia y seguridad del diseño de control inalámbrico.
- Se podrá apreciar los resultados de la comparativa que se darán de los controles aplicados, probando el rendimiento y eficiencia del sistema, discriminando el que tenga un mejor control y funcionamiento en la nube con la ayuda del método explicativo.

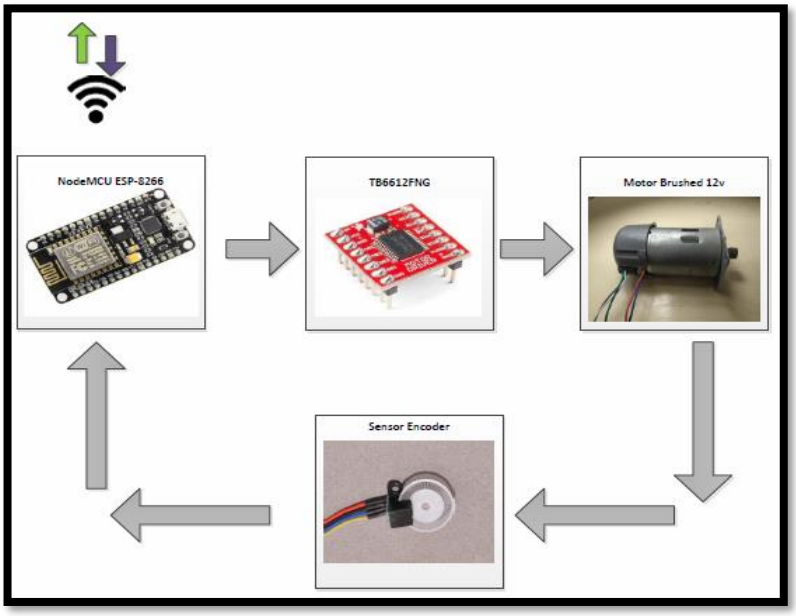
CAPÍTULO 2

MARCO TEÓRICO

2.1 Elementos que conforman la planta

En esta sección se presenta cada uno de los componentes más representativos que se utilizaron para el acoplamiento y funcionamiento del motor DC.

Figura 0.1 Diagrama de bloques de los componentes de la planta

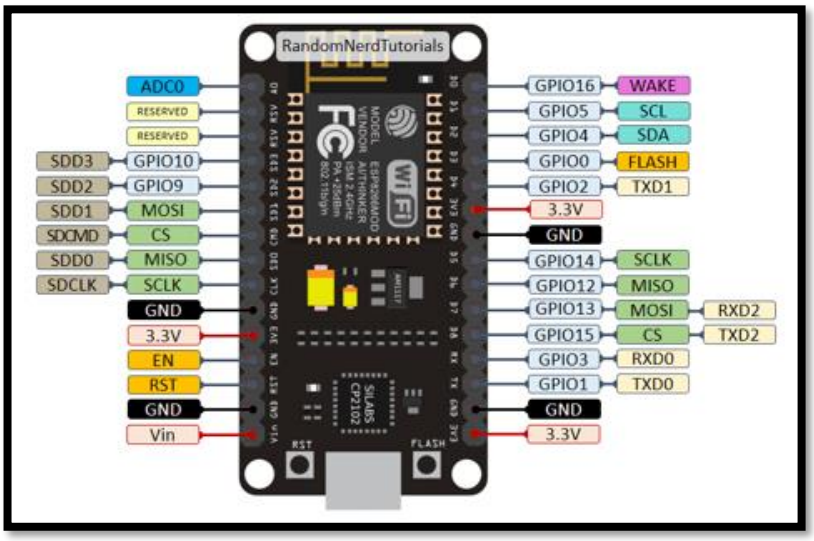


Dispositivos que funcionan conjuntamente para el requerimiento de estudio de este proyecto, Juan Escobar & Patricio Sulca

2.1.1 Microcontrolador NodeMCU

Es un kit de desarrollo de código abierto que ayuda a crear productos de IoT, desarrollado para un uso fácil manejando API avanzada para entradas y salidas de hardware (Saputra & Lukito, 2017). La ventaja más importante que ofrece es que está compuesto por el microcontrolador ESP-12 o ESP-12E que integra el módulo ESP-8266, el cual tiene la función de establecer conexiones Wifi y permitir el desarrollo de proyectos con sistemas inalámbricos (Muñoz T. C., 2018).

Figura 0.2 Módulo Wi-fi NodeMCU ESP8266



Microcontrolador inalámbrico para el motor DC, Juan Escobar & Patricio Sulca

Cabe recalcar que el NodeMCU que se visualiza en la Figura 2.2, es una plataforma de desarrollo más no un microcontrolador, este es un medio para ofrecer un mayor manejo para la programación, donde se consideran las entradas y salidas ya sean análogas o digitales, teniendo en cuenta las diferentes versiones de número de pines y dimensiones. Además de que la codificación se la puede establecer en el entorno del IDE de Arduino, para fijar la conexión del módulo a la red como también la recepción y envío de tópicos con los que trabaja en forma conjunta mediante el protocolo MQTT.

2.1.2 Motor DC Brushed

Este tipo de motores están constituidos por un imán permanente, carcasa de metal, escobillas de carbón y eje de acero, son ideales para aplicaciones de bajo costo y para los equipos que son alimentados por baterías ya que presenta ventajas como baja fricción y voltaje de arranque, de poseer una alta eficiencia como también una alta velocidad de par lineal (Chotai & Narwekar, 2017).

De hecho, este motor se ocupa para evaluar el desempeño de los controladores aplicados en la nube, ya que ofrece diferentes tipos de ventajas al momento de ponerlo en marcha considerando las condiciones de desempeño pretendidos en este estudio.

2.1.3 TB6612fng módulo driver

Este módulo controlador permite manipular dos motores de corriente continua o uno de paso, ya que está construido en base al circuito integrado TB6612fng que dispone en su interior de dos puentes H independientes con capacidad de conducir 1.2 amperios constantes a 3.2 amperios en picos (Liuliu, y otros, 2016).

Una de las principales ventajas que proporciona este módulo de admitir entradas de señal PWM para manipular la velocidad en el motor propuesto para este tema y que trabaje en forma contigua con el motor DC de 12 voltios.

2.1.4 Sensor encoder

Silva (2016) señala que: "Son dispositivos de medición acoplados a un objeto giratorio, este consta de la etapa del transductor que es la conversión del movimiento del disco en señales de pulso que se pueden codificar de forma digital" (págs. 432-433).

Este disco codificador es por lo tanto incremental que requiere solo una pista primaria en regiones igualmente espaciadas e idénticas, las cuales sirven para iniciar el conteo de pulsos para la medición de posición angular y detectar las revoluciones completas en relación a la velocidad del motor.

2.2 Softwares dedicados para la programación y simulación

En relación con lo expuesto se utilizan los siguientes programas para la obtención de la función dinámica, codificación del microcontrolador y simulaciones pertinentes de cada uno de los controladores que se detallan en la sección 2.3.

2.2.1 Matlab

En el estudio que presenta Chávez (2020), argumenta que "Es un software que permite un alto rendimiento en cálculo numérico y además de poseer varias herramientas de desarrollo que permiten la visualización y simulación de sistemas dinámicos como Simulink" (págs. 23-24).

Este programa proporciona varias herramientas de desarrollo para proyectos como pruebas de funcionamiento, integración de controles, como también para análisis y rendimientos, a su vez agilitando el cálculo de los parámetros de funcionamiento de del controlador PID, LQG y LQI obtenidos a través de programación.

2.2.2 Simulink

Es una opción que brinda Matlab para los sistemas que se programan mediante diagramas de bloques donde los modelos se editan a través de los comandos manipulados por el mouse, esta herramienta permite la simulación de modelos de sistemas lineales en tiempo continuo o discreto, con la facilidad de apreciar los resultados antes de ser implementados (Mora, 2020).

Con esta herramienta se simulan los controladores en red y sin red, esto se obtiene cargando la librería true time en Matlab, permitiendo conocer las características de las señales de salida de cada uno de los controladores, como son el tiempo de estabilización y sobreimpulso.

2.2.3 IDE de Arduino

Es un entorno de programación que permite al usuario tener una serie de componentes para de una manera eficiente optimizar el tiempo de codificación, ya que el IDE está compuesto por un editor de código para la programación, además de un intérprete, compilador, depurador y fabricador de interfaz gráfica denominada GUI. Este entorno integrado de Arduino es flexible por lo que facilita la creación de proyectos y aplicaciones complejas, dentro de este entorno existen una gran diversidad de librerías que permite la manipulación de motores, la comunicación serial e inalámbrica (Gomez & Reina, 2015).

Este programa facilita la codificación del microcontrolador con Wi-fi incorporado, validando así la red a cuál va a estar conectado el módulo, como también sus diferentes tópicos con los que se asociara el protocolo MQTT, mediante la selección de la placa y el puerto al que esté conectado para aprovechar las librerías de Arduino en la compilación y despliegue.

2.2.4 Software Python

Como mencionan Molina, Loja, Zea y Loaiza (2016) señalan que: “Es un lenguaje de programación interactivo e interpretado, con la capacidad de ejecutarse en una gran cantidad de plataformas” (págs. 201-202).

Este tipo de lenguaje es interpretado por lo que no necesita compilar el código fuente para su ejecución, lo que otorga diversas ventajas como la rapidez de desarrollo, una alta cantidad de librerías y funciones para programar con una mayor facilidad, libre y de fuente abierta, este nos permite validar la programación de los controladores que serán ejecutados dentro de un nodo de función Python en la interfaz de NodeRed.

2.2.5 Protocolo MQTT

Analizando los diferentes tipos de protocolos existentes, Light (2017) define que: “MQTT utiliza un modelo de publicación - suscripción, tiene una sobrecarga de red baja y se puede implementar en dispositivos de baja potencia, como microcontroladores que podrían usarse en sensores remotos de Internet de las cosas” (pág. 265).

Este protocolo permite un intercambio de información en la comunicación establecida entre los dispositivos que conforman una red local o global, los clientes receptan la información en milisegundos emanadas del publicador, facilitando la información en tiempo real de las variables que interfieren en el sistema según los topics establecidos en la programación de nodos originados en NodeRed.

2.2.6 Node-Red

Se basa en una interfaz web que se puede utilizar en cualquier navegador de internet, este posibilita la creación de eventos mediante la interconexión de nodos presentando un ligero entorno de trabajo y desarrollo, la flexibilidad que ofrece este software es en parte por su fundamentación en Node.js que le permite operar al borde de la red o en la nube (Suárez & Rodríguez, 2018).

Por su entorno y la cantidad de librerías que ofrece al programador agilita el proceso de programación, este puede trabajar con mensajes preestablecidos cuando se usa el

protocolo MQTT, solucionando el problema de conexión de cualquier tipo de proyecto.

2.2.7 Firebase

Es una plataforma que fomenta el desarrollo de aplicaciones móviles y web creada en 2011 que fue adquirida por Google, además se encuentra integrada en la nube en Google Cloud Platform, lo que aporta una gran cantidad de ventajas a los desarrolladores que la utilizan como escalar el tamaño de la aplicación, sincronización de los datos, conjunto de herramientas multiplataforma en web como en aplicaciones móviles, servicio BaaS y entre otras funciones. También cuenta con una gran cantidad de documentación gratuita y disponible para los usuarios, gran actividad en GitHub y StackOverflow (Barrena, 2019).

Esta plataforma facilita la programación para distintas aplicaciones y fomenta el desarrollo en diferentes lenguajes de codificación como Python, Java Script, Node JS, C++, Unity, Android, iOS y entre otras.

2.2.8 Remote.it

Es un software que posibilita la gestión de dispositivos por medio de un control remoto, ofreciendo conexiones seguras en red privada tipo VPN para evitar accesos no solicitados ya que elimina la exposición a ataques de puertos abiertos (Nuñez, 2021).

Es para muchos una solución al conectar dispositivos tanto por internet como por conexión inalámbrica o a través de redes móviles, facilitando él envió de disposiciones de control e información sin conectarte por cable al dispositivo.

2.2.9 TightVNC Viewer

Como describe Tapia (2020) en su estudio que: “Es un software que permite el control remoto de forma gratuita con la ventaja de administrar dispositivos de forma ágil y de ser compatible con el software estándar VNC, este se basa a las especificaciones relacionadas del protocolo RFB” (págs. 87-88).

Este software proporciona en sus características principales ser multiplataforma, para uso personal o comercial y fácil al momento de administrar la máquina virtual alojada en Google Cloud permitiendo manejarla de forma visual por su interfaz de usuario.

2.2.10 Google Cloud SDK Shell

Es una herramienta de línea de comandos con un conjunto de instrucciones de la CLI de gcloud, con este se autoriza a las herramientas del SDK de Cloud para validar las credenciales de cuenta que posee el usuario con el fin de acceder a Google Cloud o a su vez facilite la selección de una cuenta que posee una autorización de acceso previa (Ortega, 2017).

Por su fácil instalación brinda de manera accesible la gestión de ingreso por medio líneas de comando, a través de una dirección validada en Google Cloud.

2.3 Nociones sobre los controladores aplicados en la nube

Se detalla cada uno de los controladores aplicados en la nube, donde se destaca sus características y estructura simulada mediante bloques con la herramienta simulink.

2.3.1 Control óptimo estocástico

Durante los últimos años se ha incrementado el estudio de los controles estocásticos por la importancia en las múltiples aplicaciones en los que se los puede utilizar, para hallar una solución al problema de control optimo estocástico se utiliza la estrategia de Richard Bellman que es: “Un camino óptimo tiene la propiedad de que cualesquiera que sean las condiciones iniciales y los valores de control sobre un cierto periodo inicial, las variables de control sobre el periodo restante tienen que ser óptimas para el problema restante” (Fernández, 2019, págs. 7-8). Se considera que los algoritmos de control en WNCS presentan un problema en el tiempo de muestro de la variable, ya que la planta actúa como cliente en la red y el tiempo puede ser fijo, por otro lado el controlador tiene la función de servidor presenta un tiempo no determinista debido a que la red actualiza la información procesada de forma aleatoria, teniendo en cuenta

que en los procesos de control el periodo de muestreo es primordial para tener un óptimo control del algoritmo (Pillajo & Hincapie, 2018).

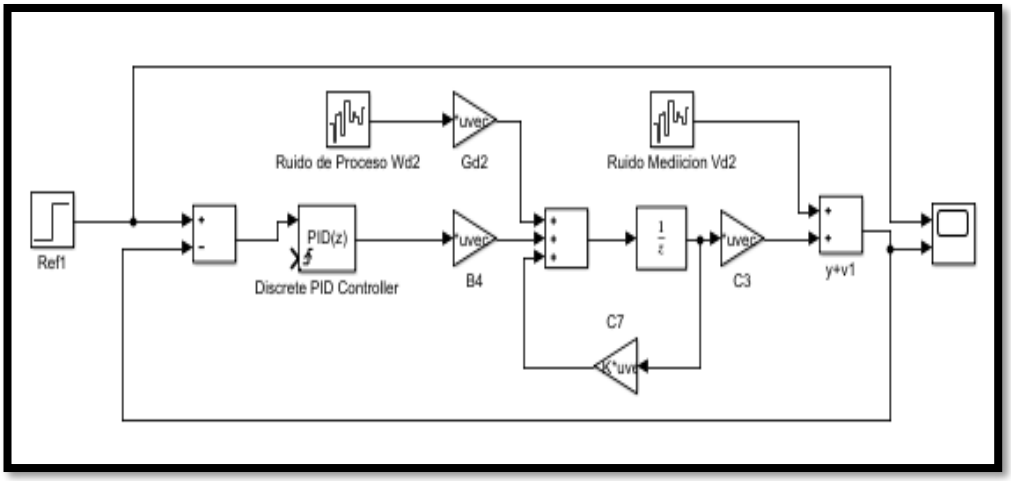
Los sistemas dinámicos sujetos a perturbaciones pueden ser controlados con el objetivo de optimizar algún criterio de desempeño, ya que los modelos de toma de decisiones se consideran por las variables que intervienen directamente en el sistema, para lo cual se establece un intervalo fijo según la interpretación del proceso maximizando la resolución del problema.

2.3.2 Control PID

Es un control utilizado en la industria debido a su confiabilidad, rapidez ante el cambio de la señal de referencia, bajo costo de implementación y sencillez en el manejo de la calibración, debido a esto este tipo de controlador es eficaz alrededor de una variable de operación ya que mantiene su punto de operación en valores establecidos según la referencia deseada por el usuario (Zambrano, 2020).

Este tipo de control es uno de los métodos que más se usan en la regulación automática, permite obtener resultados positivos en las diferentes aplicaciones en las que es utilizado debido a su flexibilidad, su estructura se define en la Figura 2.3.

Figura 0.3 Control PID simulado en Simulink sin red



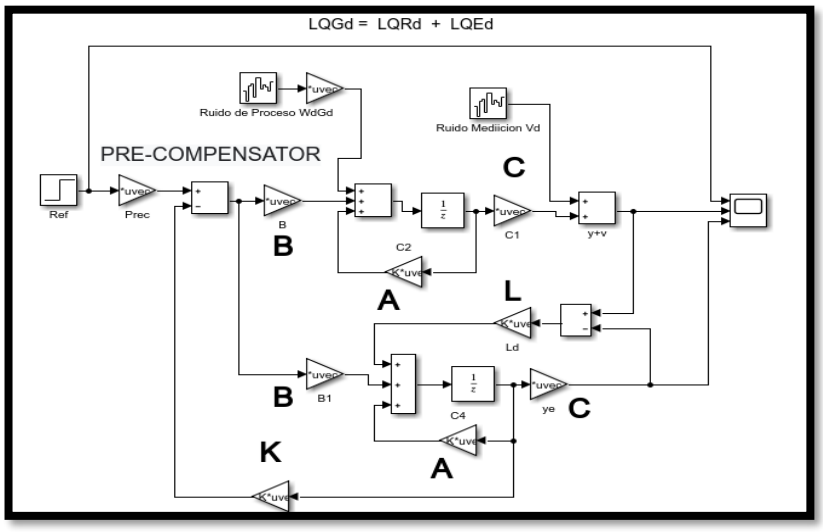
Simulación del control PID realizada en simulink sin conexión a la red, Juan Escobar & Patricio Sulca

2.3.3 Control LQG (Linear-Quadratic-Gaussian)

El control LQG es una variante del controlador LQR donde la realimentación de estados se realiza con el vector de estimación de Kalman, es capaz de proporcionar un rendimiento estable en condiciones de ruido e incertidumbre del sistema, en un problema de regulación se debe cerrar el lazo de control entre los valores estimados de velocidad y los valores de referencia (Carlucho, Menna, Paula, & Acosta, 2019).

Enfatiza que en este tipo de control se debe identificar la función de transferencia, la cual acerca el comportamiento dinámico presente en la velocidad del motor, la cual puede ser afectada por perturbaciones externas que afectan al sistema, obteniendo un modelo en variables de estado las cuales conforman el diseño del regulador LQG identificado en la Figura 2.4.

Figura 0.4 Control LQG en simulink sin red



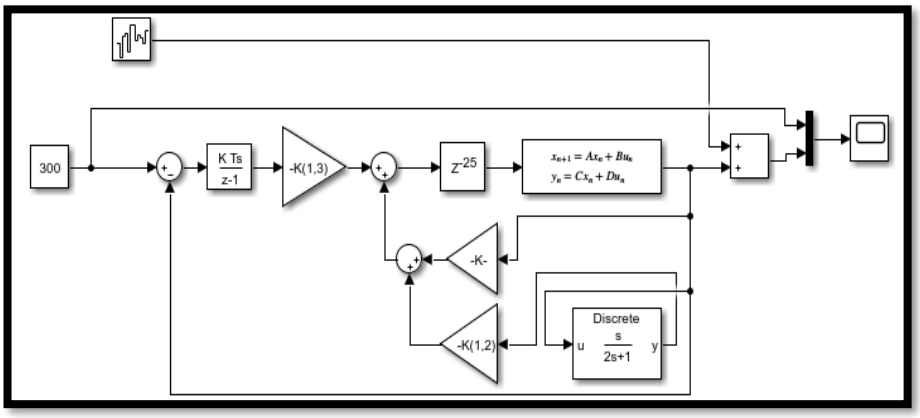
Control LQG sin red simulado con la herramienta simulink de Matlab, Juan Escobar & Patricio Sulca

2.3.4 Control LQI (Linear-Quadratic-Integral)

En el estudio realizo por Pérez (2011), se refiere al controlador LQI como: “Una forma útil de mejorar la robustez del sistema y asegurar que el error sea cero en estado estable es adicionar un integrador a la dinámica del sistema” (pág. 53).

Este control se basa en el control LQR, se adiciona a esta estructura la realimentación integral de estados para llevar el error a cero esto se aprecia en la Figura 2.5, ya que con estas consideraciones se obtiene los nuevos valores de la matriz de ganancia.

Figura 0.5 Control LQI en simulink sin red



Simulación del control LQI sin red mediante el uso de la herramienta simulink de Matlab, Juan Escobar & Patricio Sulca

CAPÍTULO 3

DISEÑO DEL SISTEMA DE CONTROL EN LA NUBE

3.1 Desarrollo del sistema a implementar

En el sistema que se presenta en la figura 3.1 se aprecia cada uno de los dispositivos utilizados que conforman el sistema en red inalámbrica, en principio se obtiene la ecuación del sistema dinámico para definir las matrices de estado, ya que las mismas sirven de referencia para la aplicación de los controladores PID, LQG y LQI en la nube mediante una máquina virtual alojada en Google Cloud para el control del motor DC, el cual se adapta a una placa desarrollada para que en forma conjunta funcione con el módulo ESP-8266, módulo LM2396S y driver TB6612fng, para establecer la comunicación se define el protocolo MQTT que agilita el intercambio de mensajes entre el broker y la planta.

Figura 0.1 Diagrama WNCS para la aplicación de los controladores

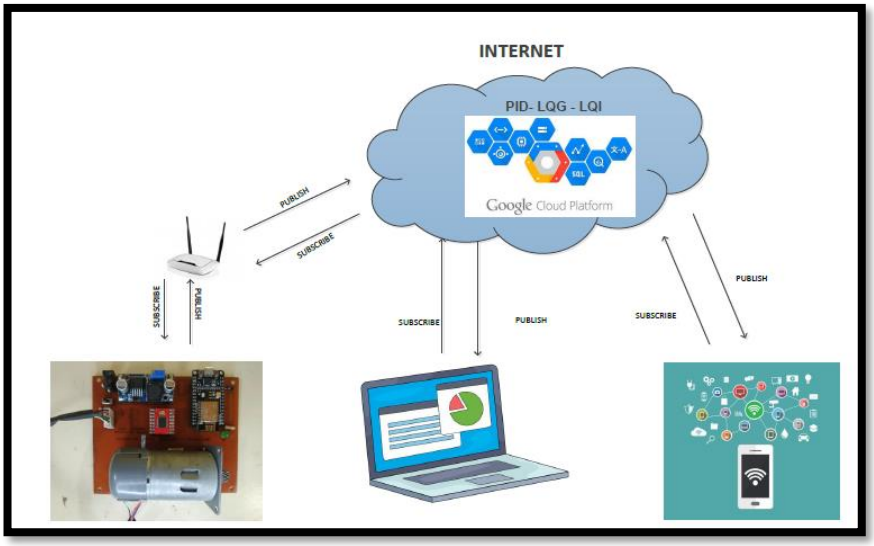


Gráfico para la implementación del proyecto de grado, Juan Escobar & Patricio Sulca

3.2 Estructura del sistema

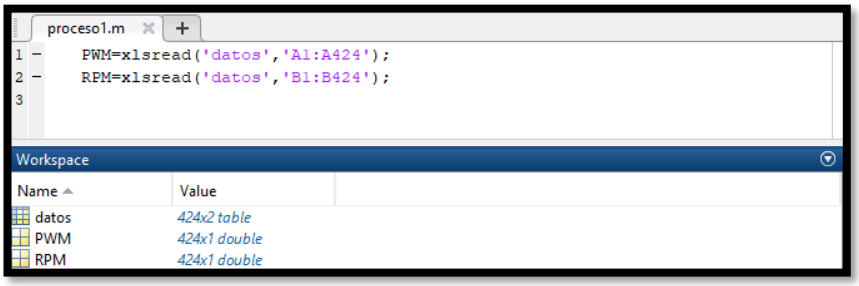
Para el progreso de esta sección se detalla la adquisición de la ecuación del sistema dinámico, matrices de estado, implementación de la máquina virtual en la nube,

habilitación del software Remoteit y diseño de la placa para la operación del motor DC.

3.2.1 Ecuación del sistema dinámico

Para iniciar con el proceso de obtención de la ecuación se delimita un conjunto de datos PWM de entrada y RPM de salida, obtenidas con el funcionamiento en conjunto del sensor encoder y el motor DC. Cada muestra consta de 9 pruebas, ya que a mayor número de pruebas se disminuye el error al momento de obtener el modelado de la función dinámica del sistema. La variación inicia desde los 300 PWM, de modo que el motor arranca de su inercia y se va incrementando con pasos de 15 hasta el valor final de 1023 PWM. Estas columnas de datos deben constar en una hoja de Excel para posterior ser cargados al Workspace de Matlab, se especifican su título para definir la información a utilizar por medio de código en un nuevo script como se define en la Figura 3.2.

Figura 0.2 Validación de variables en matlab



Se define cada una de las variables a utilizar para la obtención de la función de transferencia, Juan Escobar & Patricio Sulca

Se continúa con la herramienta Ident de Matlab para subir los datos obtenidos de las variables de entrada y salida, cabe considerar que se trabaja con el teorema de muestreo de Nyquist, este define que se puede reconstruir una señal analógica muestreada a partir de las muestras adquiridas, siendo la frecuencia de muestreo 2 veces mayor al ancho de banda de la señal (Maleh, Fudge, Boyle, & Pace, 2012). Es por ello que para evitar el aliasing, la señal debe estar perfectamente limitada a menos o igual de 1/2 de la frecuencia de muestreo, por lo que es sustancial precisar el tiempo de muestro de 0.5 segundos, este permite apreciar el comportamiento del motor cada cierto periodo, facilitando el procesamiento de la señal para la obtención de la función dinámica del

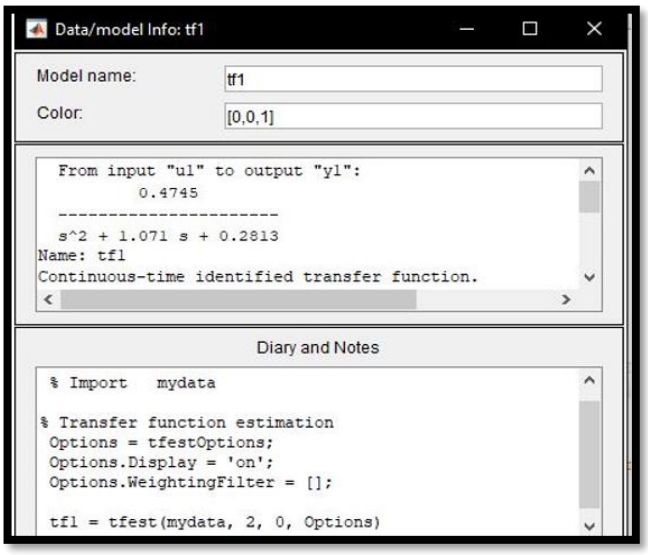
sistema. Por otra parte, se procede a la estimación de los datos con la opción Transfer Function Models fijando en sus opciones dos polos y ningún cero en función del tiempo continuo como se identifica en la Figura 3.3.

Figura 0.3 Herramienta Ident

Con esta herramienta facilita la estimación para encontrar la función de transferencia del sistema dinámico, Juan Escobar & Patricio Sulca

Una vez finalizado el testeo se obtiene una función de transferencia con una precisión mayor al 70%, este porcentaje de estimación indica la utilidad que brinda la función de transferencia para posteriores implementaciones, se refleja en la Figura 3.4.

Figura 0.4 Estimación en tiempo continuo de los datos PWM vs RPM

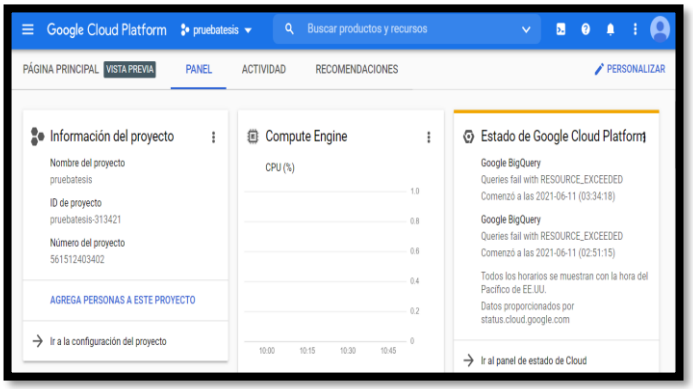


Finalización de la valoración de los datos de la función de transferencia por medio del testeo de la herramienta Ident, Juan Escobar & Patricio Sulca

3.2.2 Máquina virtual

Los controladores equipados en la nube deben estar alojados directamente en la máquina virtual creada en Google Cloud, para comenzar se debe generar un usuario llenando todos los campos requeridos, de modo que se pueda acceder a cada una de las herramientas que proporciona esta plataforma, en principio se debe dirigir a la opción de Compute Engine como se aprecia en la Figura 3.5, para la creación de una instancia.

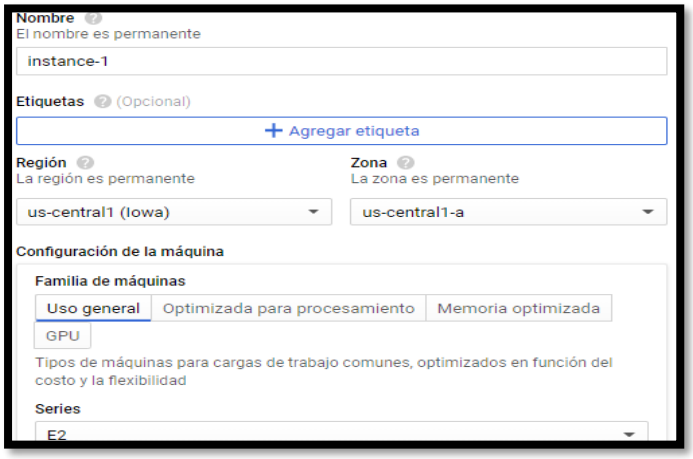
Figura 0.5 Escritorio de inicio de la plataforma Google Cloud



Plataforma de Google para la implementación del proyecto, Juan Escobar & Patricio Sulca.

Por otra parte, se debe conducir a la opción de crear nueva instancia para configurar la capacidad de la máquina, teniendo en cuenta la región, zona, serie y el tipo de máquina como se puede observar en la Figura 3.6.

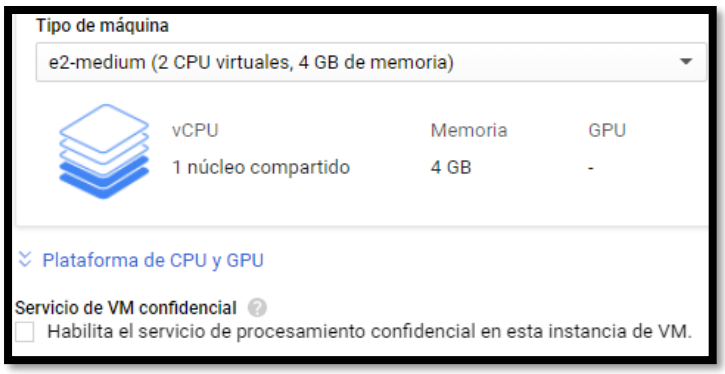
Figura 0.6 Bloque básico de información de la máquina virtual



Parámetros de configuración previos de la máquina virtual en la plataforma de Google Cloud, Juan Escobar & Patricio Sulca

En esta sección cabe resaltar la capacidad que le queremos dar a nuestra máquina, la capacidad está relacionada con el costo estimado mensual, a mayor capacidad el costo aumenta, por lo que en la opción tipo de máquina se selecciona 2 CPU virtuales con 4 GB de memoria, además de esto se habilita el servicio VM para la plataforma CPU y GPU, como se indica en la Figura 3.7.

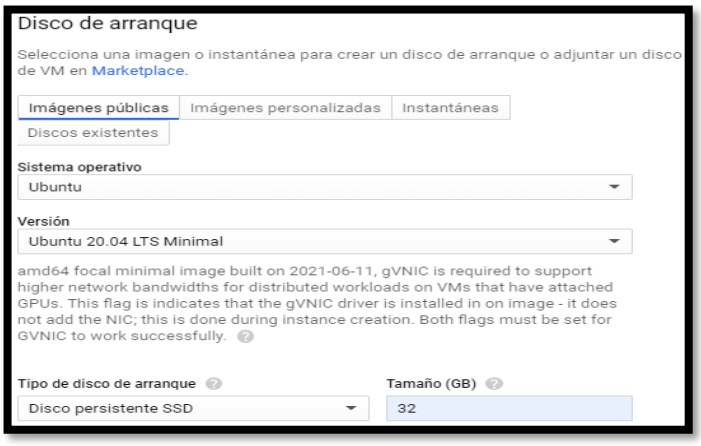
Figura 0.7 Elección del procesador de la computadora virtual



Variables físicas de creación en la máquina virtual desde la Google Cloud, Juan Escobar & Patricio Sulca

De la misma forma se debe configurar el disco de arranque, en este se debe instalar el sistema operativo Ubuntu con la versión 20.04 LTS Minimal, además de esto se toma como tipo de arranque a un disco persistente SSD de un tamaño de 32 GB, esta configuración se presenta en la Figura 3.8.

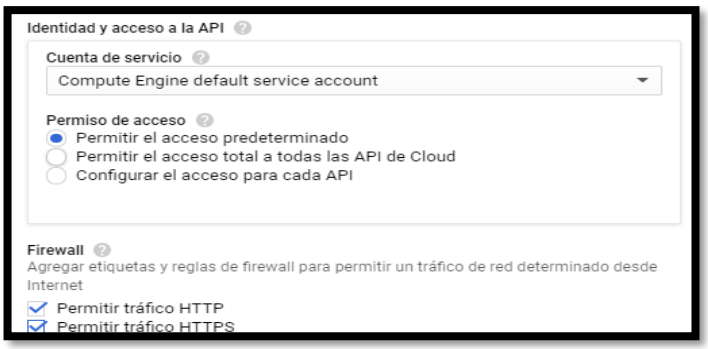
Figura 0.8 Sistema operativo a integrar en la máquina



Opciones de disposición de sistema operativo, versión y disco de arranque, Juan Escobar & Patricio Sulca

Al final se tiene que activar las reglas de firewall para autorizar el tráfico en red determinado desde internet que se muestra en la Figura 3.9, con este último paso se complementa la creación de la instancia donde será necesario la instalación del sistema operativo por medio de líneas de comandos.

Figura 0.9 Permisos de funcionamiento en red

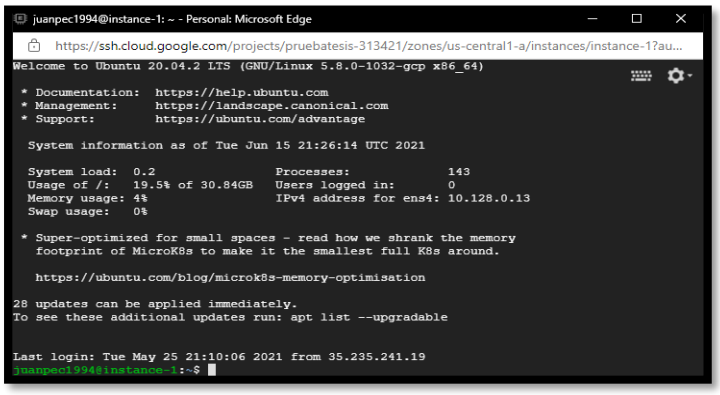


Permisos de seguridad para habilitar el funcionamiento en internet de la máquina virtual, Juan Escobar & Patricio Sulca

3.2.3 Características del sistema operativo

Una vez en la instancia se procede a iniciar para poder habilitar la conexión con la nube, en esta se dirige a la opción SSH en Google Cloud habilitando una ventana emergente para la implementación del sistema operativo Ubuntu que se indica en la Figura 3.10.

Figura 0.10 Ventana emergente de la instancia creada en Google Cloud

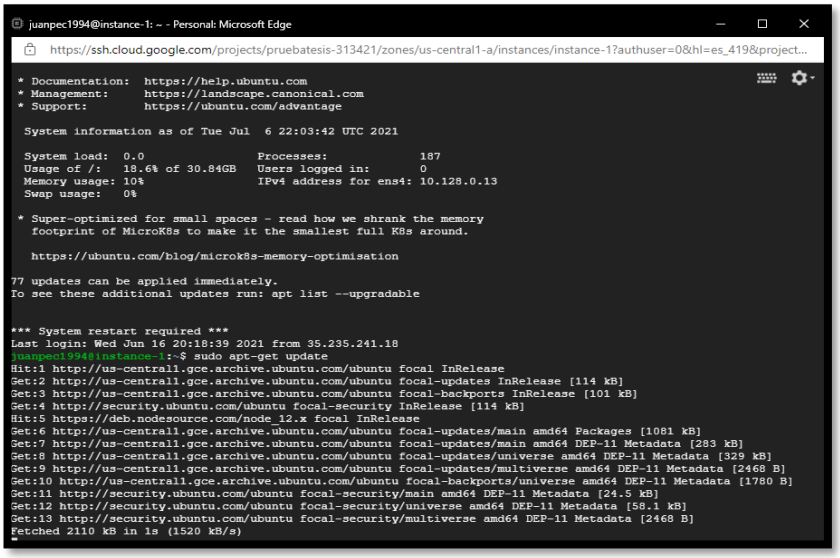


Inicio de sesión mediante la habilitación SSH de la instancia creada en Google Cloud, Juan Escobar & Patricio Sulca

Se ingresa los comandos que se comparten a continuación en la ventana emergente que se demuestra en la Figura 3.11.

- sudo apt-get update
- sudo apt-get upgrade
- sudo apt-get install gnome-shell
- sudo apt-get install gnome-core
- sudo apt-get install gnome-panel
- sudo apt-get install gnome-themes-standard
- sudo apt-get install autocutsel
- sudo apt-get install tightvncserver
- touch ~/.Xresources

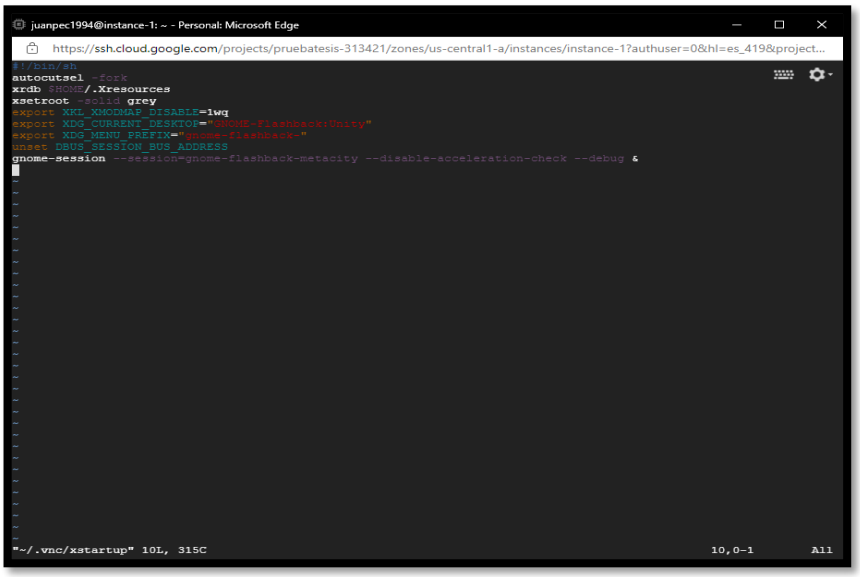
Figura 0.11 Digitalización de comandos para Ubuntu



Ingreso de los comandos para la instalación del sistema Ubuntu en la instancia creada, Juan Escobar & Patricio Sulca

Luego de digitar estos comandos se procede a inicializar el servidor VNC mediante el comando “vncserver”, donde es necesario crear una contraseña con mínimo ocho caracteres. De igual modo se abre el archivo de configuración con el comando “vim /home/usuario/.vnc/xstartup”, cabe señalar que se debe colocar el correo con el que se generó la cuenta de Google Cloud que se aprecia en la Figura 3.12.

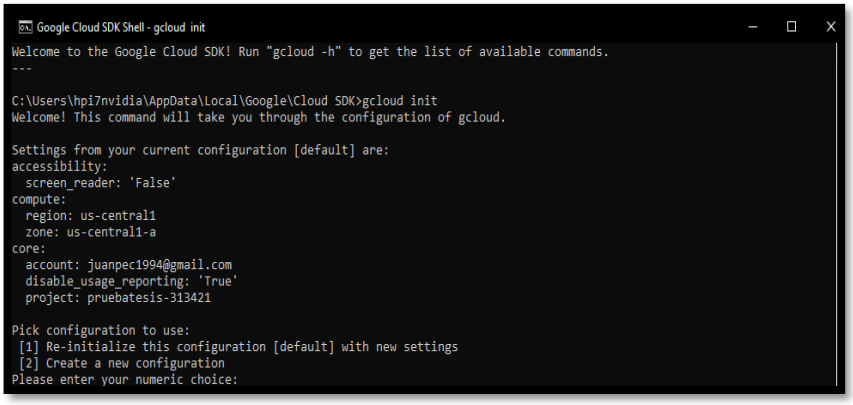
Figura 0.12 Opción emergente después de inicializar el servidor



Archivo de configuración, Juan Escobar & Patricio Sulca

Se instala el SDK de Google Cloud para poder tener el acceso a la cuenta que está directamente vinculada al servicio de esta plataforma, se tipea el comando gcloud init para desplegar las opciones de configuración como se señala en la Figura 3.13, del mismo modo se escoge la cuenta, nombre de proyecto que se creó o instancia y zona horaria con la que se creó la máquina virtual.

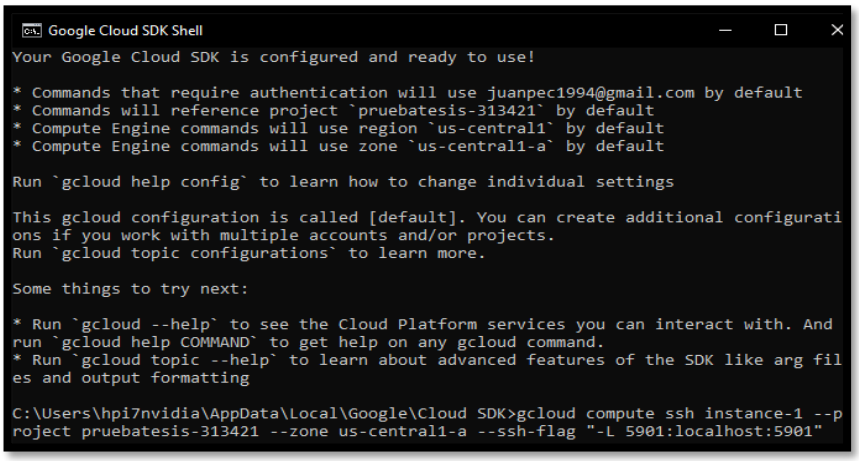
Figura 0.13 Opciones de inicio en Google Shell



Inicialización de la herramienta Google Cloud SDK Shell por comandos, Juan Escobar & Patricio Sulca

Para concluir la configuración se ingresa el comando “gcloud compute” que va seguido del nombre de la instancia, nombre del proyecto, zona horaria y el host prestablecido, esto genera un puente para el acceso remoto entre el computador y la máquina virtual como se visualiza en la Figura 3.14.

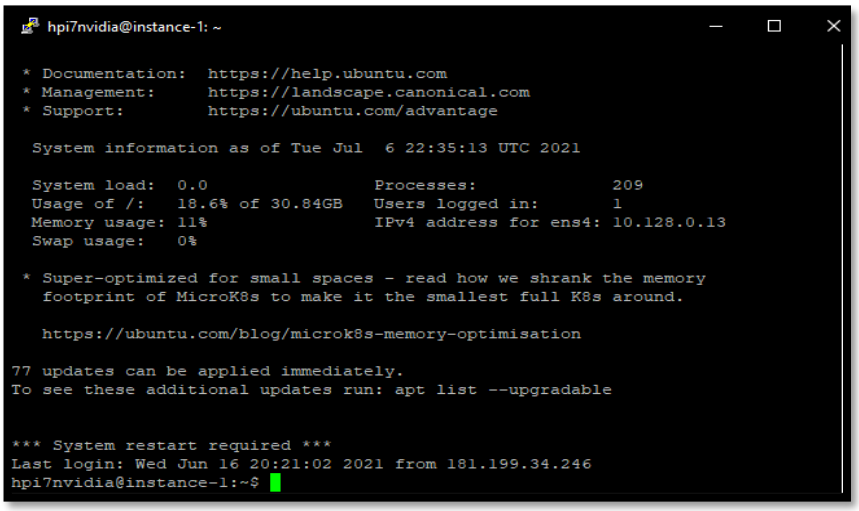
Figura 0.14 Validación del funcionamiento de la máquina



Habilitación por comandos de la instancia para el inicio de la máquina virtual, Juan Escobar & Patricio Sulca

De esta forma se habilita una ventana emergente del programa Putty, el cual permite la autentificación de la configuración preestablecida en Google Cloud SDK Shell, habilita a la máquina virtual para su funcionamiento que se aprecia en la Figura 3.15.

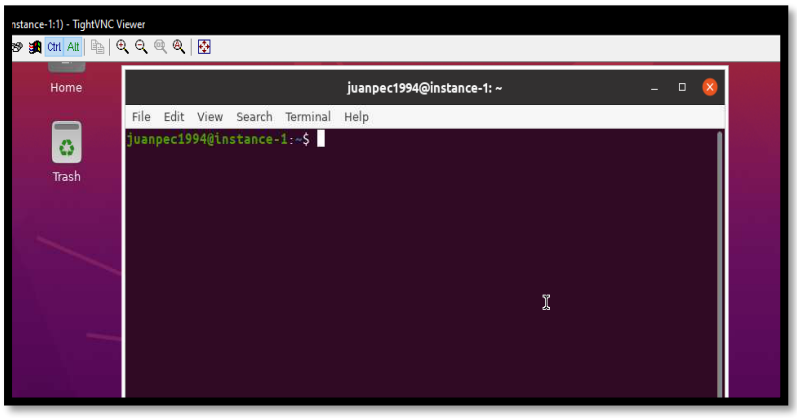
Figura 0.15 Habilitación para la conexión al servidor



Preparación de conexión con la herramienta Putty para arranque del servicio, Juan Escobar & Patricio Sulca

A continuación, se procede abrir el programa TightVNC Viewer para la autentificación de usuario y contraseña que nos apruebe el ingreso como se presenta en la Figura 3.16, una vez cumplido este requisito nos arroja a la ventana principal, se apertura una ventada de terminal para continuar con la instalación de Node.js y el bróker Mosquitto.

Figura 0.16 Ventana de comandos de Ubuntu

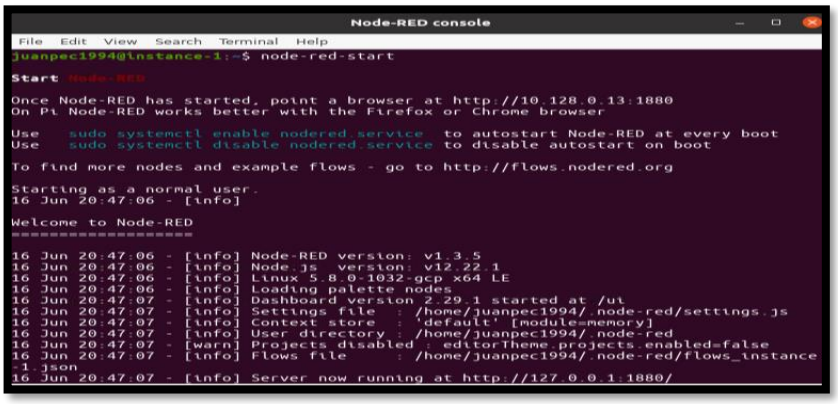


Interfaz gráfica de la máquina previo a la instalación del Bróker Mosquitto, Juan Escobar & Patricio Sulca

Para la instalación se realiza con las siguientes líneas de comandos:

- bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
- sudo apt-get install mosquitto mosquitto-clients
- sudo apt-get update

Figura 0.17 Instalación por comandos para protocolo MQTT



Instalación de programas complementarios para el funcionamiento del protocolo MQTT, Juan Escobar & Patricio Sulca

Con respecto a la inicialización de Node-Red se debe digitar el comando “node-red-start”, de esta manera se obtiene la ip que nos permite abrirla en un cualquier navegador como se visualiza en la Figura 3.17, para habilitar la interfaz de software y promanar por diagrama de flujos los controladores PID, LQG y LQI.

3.2.4 Instalación de Remoteit en la máquina virtual

En la instalación de este software se debe habilitar un terminal de comando, en este caso el sistema operativo instalado en la máquina virtual es Ubuntu, por lo que se procede a instalar la aplicación Node.js para Debian con los comandos que se detalla a continuación:

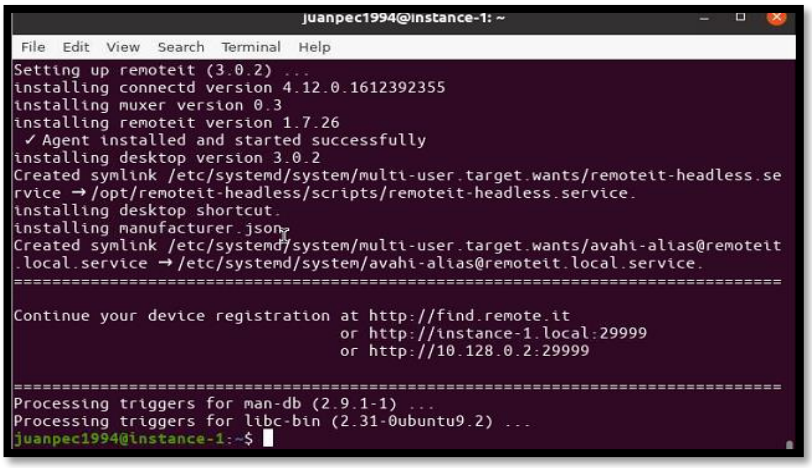
- curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash –
- sudo apt-get install -y nodejs

Dentro de este marco se debe verificar el paquete Remoteit de Debian amd64 con la instalación mediante las líneas de comandos que se especifican en los siguientes puntos.

- curl -LO https://downloads.remoteit /debian/latest/remoteit_amd64.deb
- md5sum remoteit_amd64.deb

Por último, se instala el paquete Remoteit por medio del comando “sudo apt install -y ./remoteit_amd64.deb”, el cual finaliza el establecimiento del programa presenta un mensaje para continuar con el registro de la cuenta generada en la página oficial como se considera en la Figura 3.18.

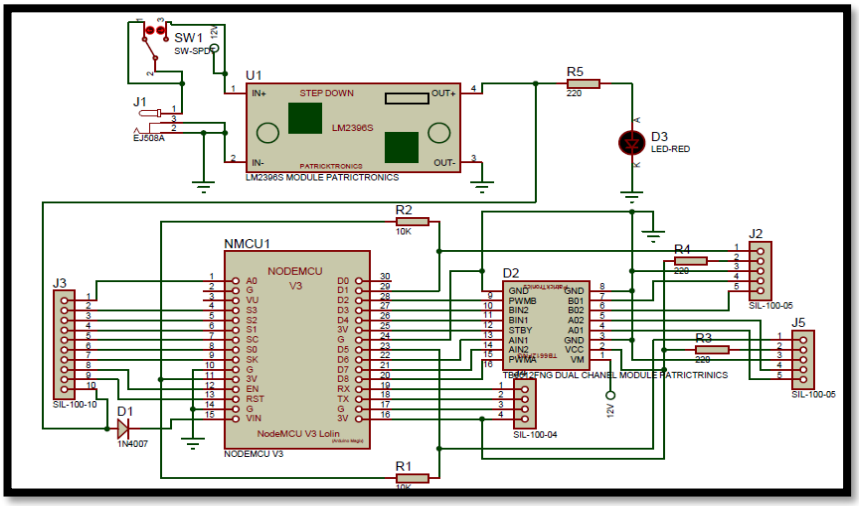
Figura 0.18 Terminal de comando con el proceso de instalación



Se detalla el progreso de instalación del software Remoteit en la máquina virtual alojada en Google Cloud, Juan Escobar & Patricio Sulca

3.2.5 Elementos y conexiones de la placa desarrollada para el funcionamiento del motor DC.

Figura 0.19 Simulación de dispositivos



Conexiones de los diferentes dispositivos que conforman la placa para el funcionamiento del motor DC, Juan Escobar & Patricio Sulca

Por otra parte, las conexiones simuladas son necesarias para el diseño de la placa PCB. Con esto se busca tener en cuenta cada uno de los dispositivos utilizados, donde se emplea una fuente de 12V a 3A que suministra la energía a todo el circuito tanto para la parte de control y de fuerza, seguido de esto se coloca un switch que establece el encendido y apagado del circuito, el mismo soporta una corriente máxima de 4A a 35V.

En este caso se señala también que el módulo step down LM2596 trabaja con un voltaje de entrada de 4.5V a 40V y administrando un voltaje de salida de 1.5V a 35V ajustables con una corriente máxima de 3A, este “Reduce al mínimo el uso de componentes externos para simplificar el diseño de fuentes de alimentación” (González & Castro, 2020, pág. 180), la configuración utilizada en la placa desarrollada recibe 12V de entrada directamente de la fuente y se adapta 5V en la salida para alimentar al circuito de control, constituido por la placa NodeMCU ESP-8266, a su vez energiza al sensor encoder y driver TB6612fng.

Dentro de este marco, la placa NodeMCU se encuentra asociado con el módulo ESP-8266, facilita la elaboración del proyecto con relación a WNCS, ya que administra las

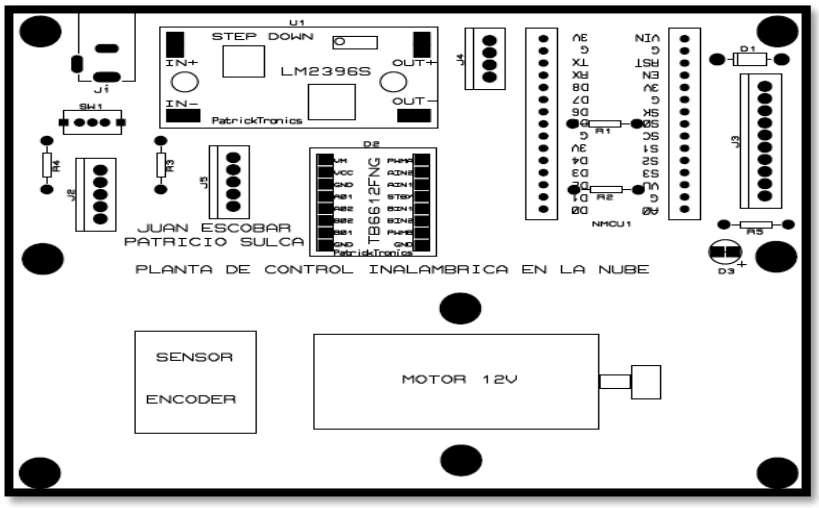
entradas, salidas y procesamiento de datos que requiera el programa en cada instante (Muñoz G. R., 2018).

En este orden se tiene que el motor DC Brushed de 12V el cual opera a 400mA sin carga. Con relación a lo anterior se plantea el uso de un módulo TB6612fng, funciona en un rango de alimentación para motores de 4.5 a 13.5 voltios, con voltaje de control de 2.7 a 5.5 voltios, una corriente continua por canal de 1A y máxima de 3A pico. Este regula el ancho de pulso, con esto se ajusta la potencia entregada al motor, ya que es un intermediario entre el dispositivo de control y el actuador (Vega, Acuña, Acuña, & Velázquez, 2019), este driver se alimenta con 12V para el motor y voltaje de control de 3.3V.

Por otro lado, el sensor encoder óptico infrarrojo Tcst2103 con voltaje de alimentación de 3 a 5 voltios, es el componente que satisface la medición de velocidad del motor, usando un disco ranurado de pulsos (Becerra, 2016), el diseño en la placa se complementa con resistencias de 220 Ω y 10k, como también un diodo led de indicador que se presenta en el circuito de la Figura 3.19.

Teniendo en cuenta cada uno de elementos, se debe proceder al ajuste y colocación donde se prioriza la orientación, ubicación y que estos estén organizados dentro del espacio delimitado para la placa como se puede ver en la Figura 3.20.

Figura 0.20 Serigrafia de superior de la placa PCB

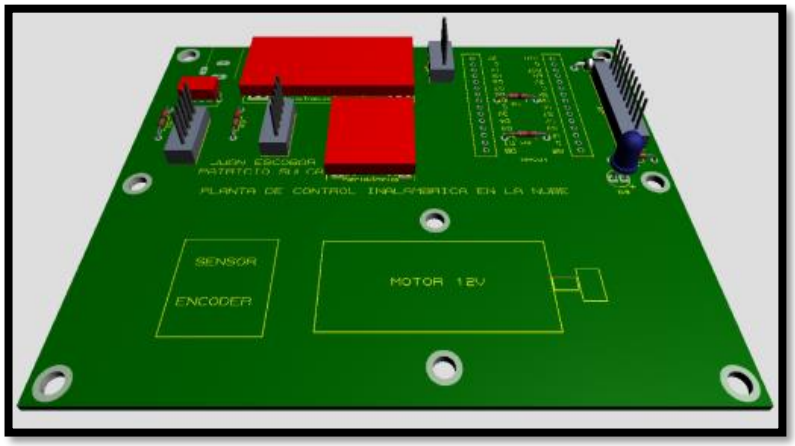


Serigrafia de la parte superior e inferior de la placa desarrollada para el funcionamiento del motor DC, Juan Escobar & Patricio Sulca

Además de esto se debe definir el ancho de las pistas para un PCB de baquelita con espesor 0.035 mm de cobre a una temperatura aproximadamente 25°C, se establece pistas en el circuito de potencia de 80th, que soporta un máximo de 2A para corrientes picos del motor, mientras que en el circuito de control se maneja pistas de 20th con capacidad máxima de 500 mA que alimenta el resto de elementos de la placa.

Con esto se busca la simetría para enrutar las conexiones de señal, energía y tierra en la placa. Esto previene los encadenamientos de líneas que producen cortos circuitos, ya que con la separación se busca un camino de alimentación funcional. Retomando la expresión anterior, se debe situar la serigrafía en la cara superior de la placa para reforzar la identificación de cada componente, priorizando que los elementos ocupen el sitio designado al momento de preparar la placa como se aprecia en la Figura 3.21.

Figura 0.21 Vista de la placa en 3D



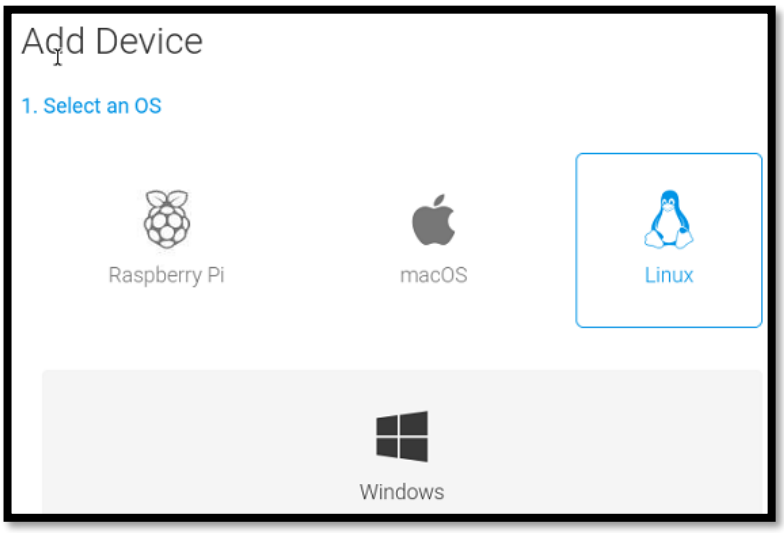
Placa acondicionada para la colocación de los elementos que solventan el funcionamiento del motor DC, Juan Escobar & Patricio Sulca

CAPITULO 4

APLICACIÓN DE LOS CONTROLADORES

Para vincular la conexión de la planta a la nube con el uso del protocolo MQTT, se utiliza el programa Remoteit, el cual permite compartir recursos a diferentes dispositivos IoT, para agregar un nuevo dispositivo en Remoteit se debe elegir el sistema operativo en el cual se está operando, ya que los comandos de instalación son diferentes para cada plataforma.

Figura 4.1 Adición de un nuevo dispositivo en Remoteit



Escritorio del software Remoteit con las opciones para la adición de un nuevo dispositivo, Juan Escobar & Patricio Sulca

Cabe considerar en este procedimiento, se debe ingresar las siguientes líneas de código en una ventana de comando:

- curl -LkO https://raw.githubusercontent.com/remoteit/installer/master/scripts/auto-install.sh
- chmod +x ./auto-install.sh
- sudo ./auto-install.sh

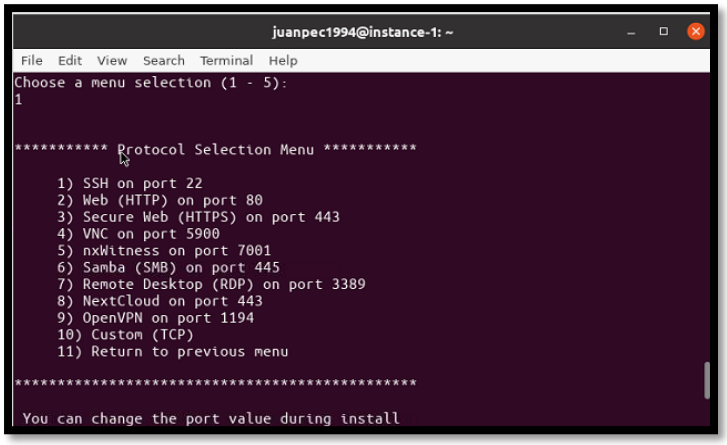
Para habilitar el menú interactivo de creación de dispositivo en la ventana de comando se debe agregar el comando “sudo connectd_installer”, como se presenta en la Figura 4.2.

Figura 4.2 Menú de servicios de Remoteit

En la ventana se despliegan las opciones de creación de un nuevo dispositivo en Remoteit, Juan Escobar & Patricio Sulca

Por otra parte, se digita la opción uno para poder ingresar el correo con el que se registró en la página oficial de Remoteit, la cual nos permite ingresar a las opciones de protocolos que maneja este software. Se observa en la Figura 4.3 once diferentes preferencias en protocolos para escoger.

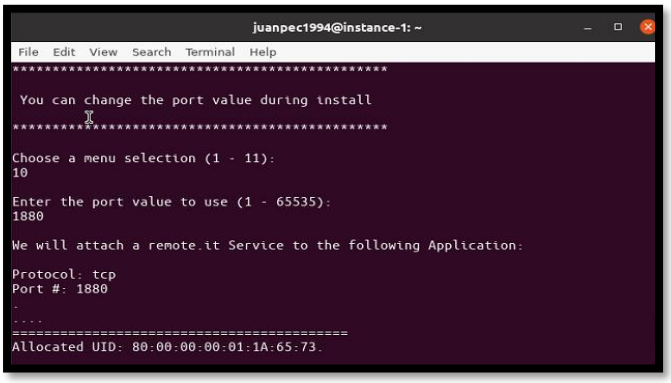
Figura 4.3 Selección de protocolos



Un primer aspecto a considerar es el protocolo en el que se va a trabajar dentro de los parámetros de comunicación, Juan Escobar & Patricio Sulca

De la misma manera se selecciona la opción diez, esta indica que se debe configurar el protocolo TCP con el puerto 1880 como medio transmisión de datos entre diferentes colaboradores dentro de una red informática, como se identifica en la Figura 3.5.

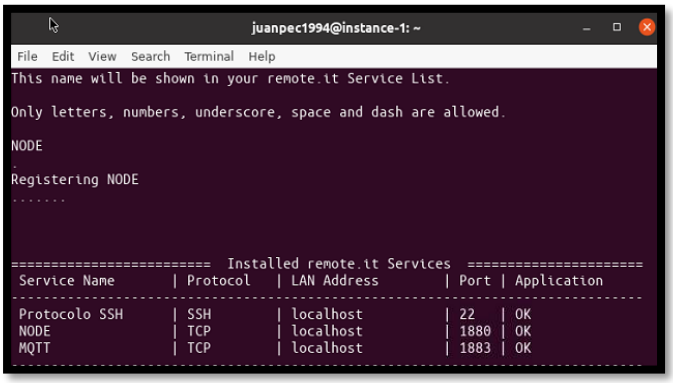
Figura 4.4 Configuración del protocolo en la red



Se ingresa la definición de puerto para validar la creación del dispositivo en el software Remoteit, Juan Escobar & Patricio Sulca

Para concluir con el registro del dispositivo se debe definir el nombre, el cual debe contener una sintaxis corta para poder identificarlo de una manera ágil en el escritorio de Remoteit, esto se puede notar en la Figura 4.5 que valida el registro del dispositivo.

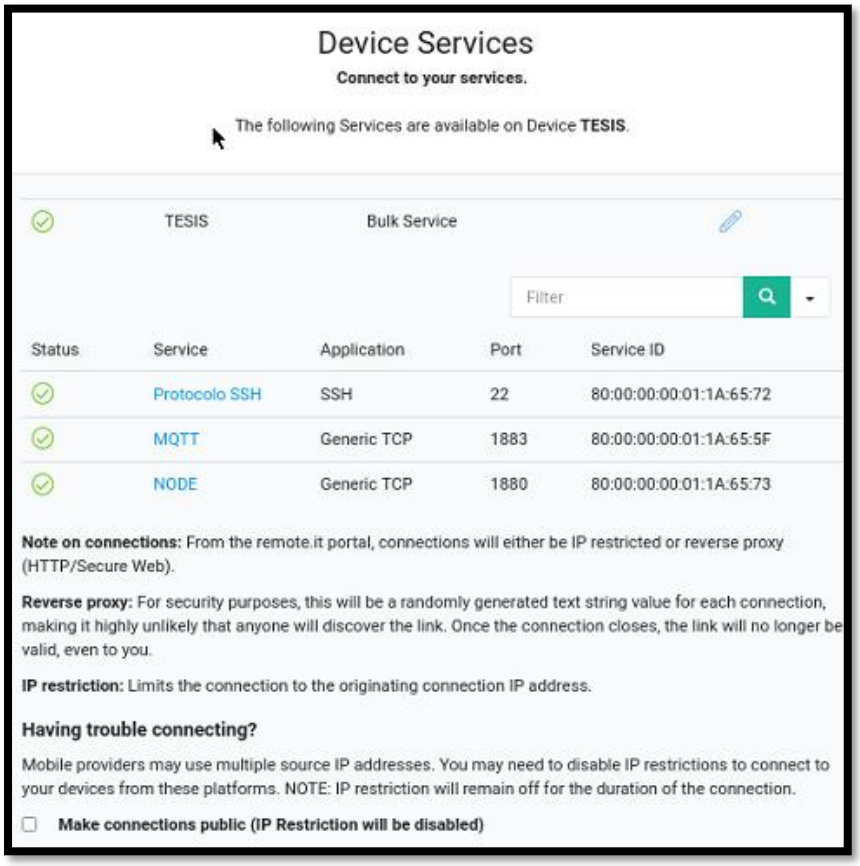
Figura 4.5 Conclusión del registro



El registro del dispositivo NODE es exitoso por lo que se habilita en la aplicación, Juan Escobar & Patricio Sulca

Se expone el registro del dispositivo en la página de Remoteit en la Figura 4.6. Una vez creado se establece una conexión pública para obtener una dirección de proxy, la misma que debe constar en la programación que se efectúa en el IDE de Arduino para el funcionamiento y posterior carga al módulo-wifi Esp-8266.

Figura 4.6 Servicios disponibles en Remoteit



Remoteit permite la comunicación que se establece entre la planta y la máquina virtual, Juan Escobar & Patricio Sulca.

4.1 Programación de los controladores

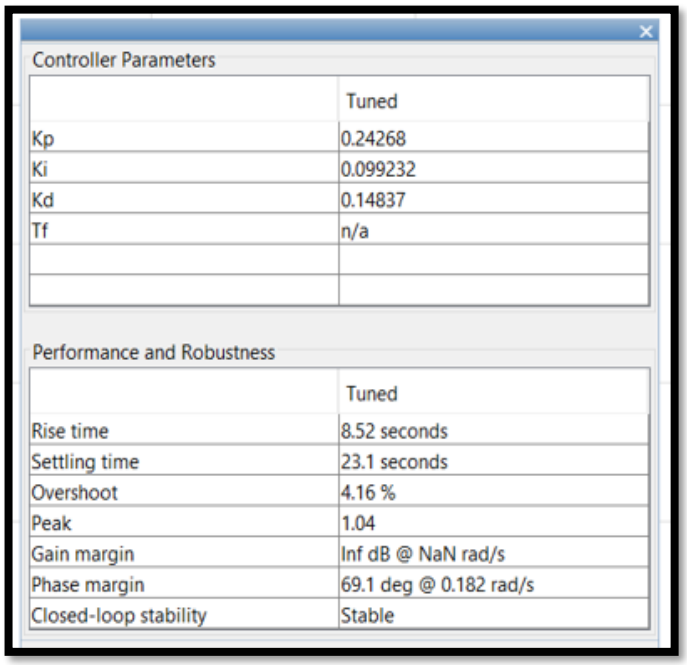
Para comenzar con este proceso se define la función de transferencia del sistema dinámico como se muestra en la Ecuación 4.1, el proceso de obtención se detalla y justifica en la sección 3.2.1, la cual es obtenida mediante el procesamiento de los datos experimentales adquiridos del sensor encoder, que trabaja simultáneamente con el motor. Esta función del sistema es implementada en cada uno de los controladores que se enuncian a continuación: PID, LQG y LQI.

$$f_T = \frac{0.4745}{s^2 + 1.071s + 0.2813} \qquad \text{Ec. (4.1)}$$

4.1.1 Control PID en WNCS

Para el caso del control PID se calcula las ganancias con la herramienta PID Tuner de Matlab que se aprecia en la Figura 4.7.

Figura 4.7 Ganancias para el controlador PID



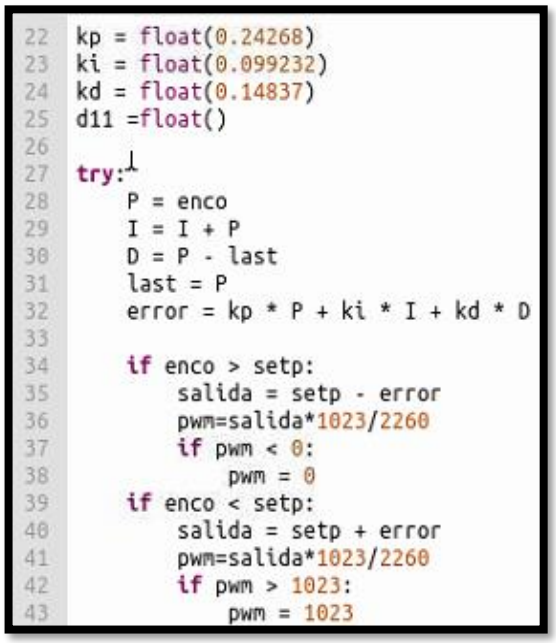
Estas ganancias se obtienen mediante el procesamiento de las muestras de la variación PWM en el motor mediante la herramienta PID Tuner, Juan Escobar & Patricio Sulca.

Se procede a la codificación en Python, se considera que este control está constituido por una fracción PI enlazada con otra fracción PD, las cuales permiten una acción combinada que favorecen al control por lo que se interpreta la Ecuación 4.2 en la codificación.

$$\frac{U(s)}{E(s)} = k_P\left(1 + \frac{k_I}{S} + k_D\right) \qquad \text{Ec. (4.2)}$$

En función de lo planteado se tiene en la Figura 4.8 la sección de programación del algoritmo PID, con sus correspondientes cantidades de ganancias con la que se establece un control funcional para la planta.

Figura 4.8 Definición del control PID en Python



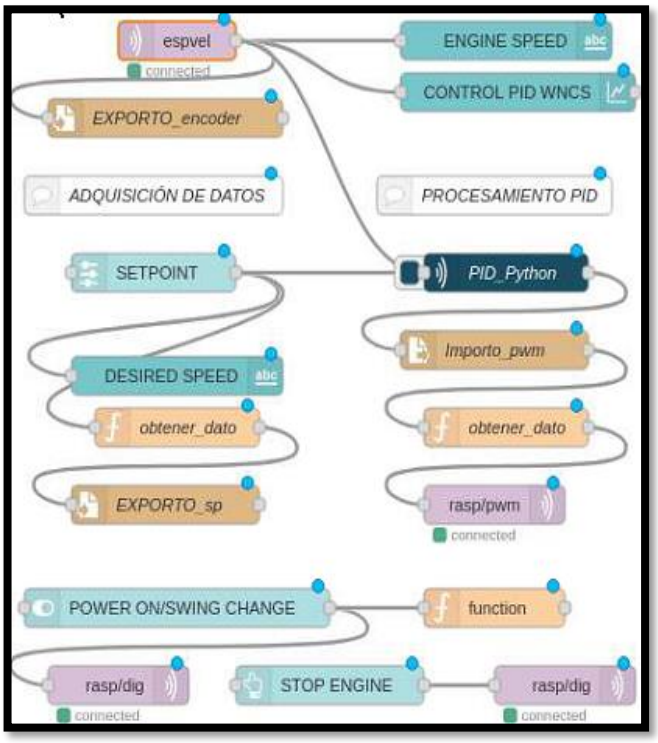
```
kp = float(0.24268)
ki = float(0.099232)
kd = float(0.14837)
d11 =float()

try:
    P = enco
    I = I + P
    D = P - last
    last = P
    error = kp * P + ki * I + kd * D

    if enco > setp:
        salida = setp - error
        pwm=salida*1023/2260
        if pwm < 0:
            pwm = 0
    if enco < setp:
        salida = setp + error
        pwm=salida*1023/2260
        if pwm > 1023:
            pwm = 1023
```

Porción de código en Python que permiten definir el control PID, Juan Escobar & Patricio Sulca.

Retomando lo antes mencionado se contempla la programación en NodeRed, se arrastran los nodos del protocoló MQTT que permiten el estableciendo del broker y topic, estos agilitan el intercambio de mensajes en los controladores. En secuencia se colocan los nodos pertenecientes a las opciones de dashboard que son; slider, text, chart, button como también del nodo switch, estos admiten la manipulación y visualización de las constantes de entrada y salida con relación al funcionamiento del control.

Se añade también los nodos correspondientes a función que otorgan el llamado del archivo en Python y conceden la exportación e importación de datos en un archivo.txt que tienen relación directa con la variación de setpoint o referencia. Todos los nodos expuestos anteriormente se visualizan en la Figura 4.9, correspondiente al control PID.

Figura 4.9 Control PID en NodeRed



Nodos utilizados para el funcionamiento y manejo del control PID en WNCS, Juan Escobar & Patricio Sulca.

Para establecer que la comunicación sea bidireccional entre la planta y la máquina virtual se programa los tópicos usados en NodeRed, similarmente se define la red a la cual se va a conectar el módulo MCU como también del proxy que proporciona Remoteit, teniendo en cuenta que la programación se la realiza en el IDE de Arduino para posterior ser cargada al módulo ESP-8266 como se puede evidenciar en la Figura 4.10.

Figura 4.10 Definición de variables en IDE de Arduino

```
// Update these with values suitable for your network.
volatile int contador=0, vel=0;
const char* ssid = "LUMBIBOTS";
const char* password =    "1722958939";
//const char* mqtt_server = "192.168.0.112";   /// example 192.168.0.19
const char* mqtt_server = "proxy21.rt3.io";
const char* topic_analog = "rasp/p";
const char* topic_dig = "rasp/dig";
```

Porción encargada de establecer la comunicación entre el broker y la planta, Juan Escobar & Patricio Sulca

4.1.2 Control LQG en WNCS

En relación con el anterior controlador, adopta cambios relacionados con el modelamiento matemático ya que está compuesto por un filtro de Kalman, que actúa como predictor reduciendo el error, al mismo asume como observador a un control LQR, lo cual permite ofrecer una solución lineal, esta definición de su estructura se centra en los espacios de estado que representan como "x" los cuales se denotan en la Ecuación 3, 4 y 5.

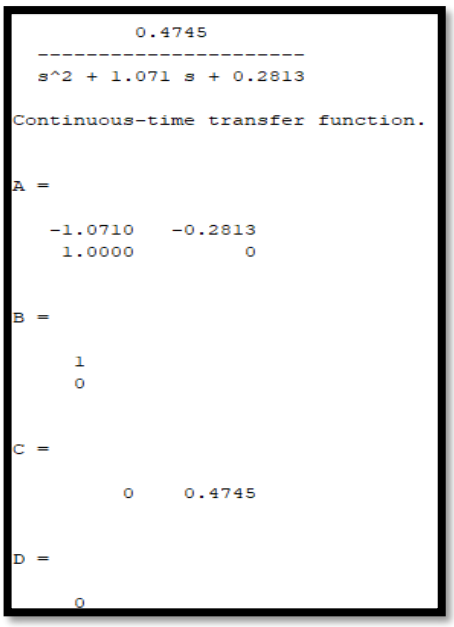
$$J = \int_{t0}^{tf} (x^T Q x + u^T R u) dt \qquad \text{Ec. (4.3)}$$

$$x_{k+1} = Ax_k + Bu_k + W_k \qquad \text{Ec. (4.4)}$$

$$y_k = Cx_k + V_k \qquad \text{Ec. (4.5)}$$

Se considera el procesamiento de la función de transferencia para la obtención de las matrices de estado en Matlab partiendo de su función de transferencia, obteniendo Figura 4.11.

Figura 4.11 Matrices de estado originadas de la función de transferencia



Matrices de estados definidas en el código de Python para ser ejecutado desde el nodo de función en NodeRed, Juan Escobar & Patricio Sulca

La determinación de la ganancia para el controlador LQG se la realiza por medio de programación en Matlab en base a su fundamento matemático detallado en las ecuaciones de la sección 4.1.2, teniendo como resultado la ganancia K que se presenta en la Figura 4.12.

Figura 4.12 Ganancia establecida por programación del controlador LQG

```
%% Definición de la planta
clc; clear all; close all;
num = [0.4745];
den = [1,1.071,0.2813];
Hs=tf(num,den) % Función de transaferencia continua
[A,B,C,D] = tf2ss(num,den)
B2 =[0;1];
%% controlabilidad
n=size(A,1)
c_cont=ctrb(A,B);
if (rank(c_cont) ~=n)
error('planta no contolable');
end
%% Observabilidad
o_cont=obsv(A,C);
if(rank(o_cont )~=n)
error('planta no observable');
end
%% Inicializaciones
```

```
Command Window

K =

    0.2274    0.2673
```

Ganancia K perteneciente al controlador LQG que define se define como una matriz, Juan Escobar & Patricio Sulca

En función de esto, se plantea la programación del algoritmo de control en Python, resaltando la definición de las matrices como de su ganancia K en la Figura 4.13.

Figura 4.13 Precisando las matrices y ganancias del LQG

```
d1 = open("/home/juanpec1994/Desktop/TESIS/Prueba1","r")
d11= d1.read()

dato =float()
referencia=float()


## MATRICES DE ESTADO A,B,C,G
A = np.array([[-1.0710,-0.2813],[1,0]])
B = np.array([[0],[1]])
C = np.array([0,0.4745])
G = np.array([[0],[1]])

# GANANCIA K (LQR)
K = np.array([0.2274,0.2671])
x = np.array([[0],[0]])
y = np.array([0.0,0.0])
z = float()
```

Matrices de estados definidas en el código de Python para ser ejecutado desde el nodo de función en NodeRed, Juan Escobar & Patricio Sulca

De esta manera la codificación en Python debe resaltar el modelamiento del control LQG con la ecuación de Riccati que se puede identificar en la Ecuación 4.11 y el filtro Kalman establecido por la Ecuación 4.9

$$u = -K_c\hat{x} \qquad \text{Ec. (4.6)}$$

$$K_c = R_c^{-1}B^T P_c \qquad \text{Ec. (4.7)}$$

$$A^T P_c + P_c A - P_c B R^{-1} B^T P_c + Q_c = 0 \qquad \text{Ec. (4.8)}$$

$$\frac{d\hat{x}}{dt} = A\hat{x} + B\hat{u} + K_o(y - C\hat{x}) \qquad \text{Ec. (4.9)}$$

$$K_o = P_o C^T R_o^{-1} \qquad \text{Ec. (4.10)}$$

$$AP_o + P_o A^T + Q_o - P_o C^T R_o^{-1} C P_o = 0 \qquad \text{Ec. (4.11)}$$

Tanto como resulta la adaptación en lenguaje Python del control antes mencionado, el análisis procede eventualmente con el número de repeticiones establecidas habituando el funcionamiento propuesto mediante la manipulación del setpoint, retomando la idea anterior se evidencia en la Figura 4.14.

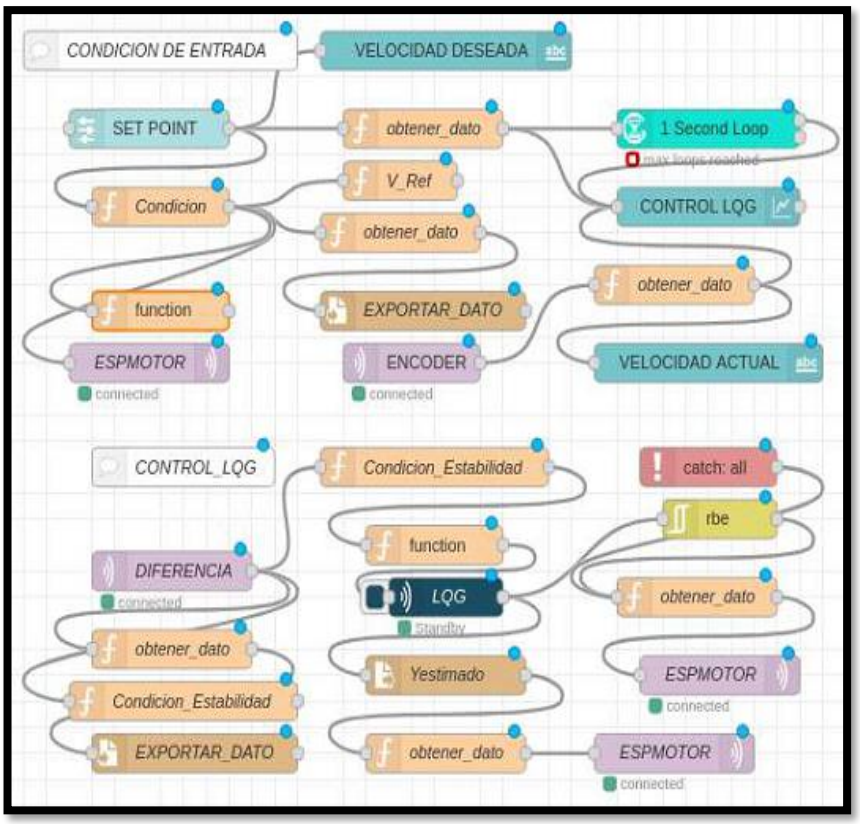
Figura 4.14 Adecuación de la algebra del control LQG

```
while c1< N :
        randn=random.uniform (-1.2,1.2)
        Kxe=np.dot(K,xe) ## MULTIPLICACION MATRICES K*xe
        u = -Kxe+prec*ref;
        #////////////////x = A*x+B*u+G*sqrt(W(k))*randn;//////////////////////////
        Ax=np.dot(A,x)              ##MULTIPLICACION Matriz A* Matriz x
        Bu=B*u                      ##MULTIPLICACION Matriz B*u
        AxBu= Ax+Bu                 #SUMA MATRIZ A*x + B*u
        #////////////////////////////RUIDO (G*sqrt(W(k))*randn)//////////////////////////
        Gr= G* sqWk
        GrsqWk = Gr * randn
        #////////////////////////////////////////////////////////////////
        x = AxBu + GrsqWk
        #////////////////////////xp = A*xe + B*u;///////////////////////
        Axe = np.dot(A, xe)
        xp= Axe+Bu
        #////////////////////xe =xp + L*(z-C*xp);////////////////////////////////
        Cxp = np.dot(C, xp)
        zCxp= z-Cxp;
        LzCxp = L*zCxp
        xe= xp+LzCxp
       #/////////////// y =C*x/////////////////////////////
        y1 = np.dot(C, x)
```

Adaptación de la ecuacion de Riccati y del filtro de Kalman pertenecientes al control LQG, Juan Escobar & Patricio Sulca

Al igual que en el control PID, se identifica los nodos a utilizar, teniendo en cuenta que el control LQG es llamado dentro del nodo función python-shell, los datos de salida y entrada se exportan en archivos.txt para ser procesados dentro de funciones como se plantea en la Figura 4.15.

Figura 4.15 Estructura del control LQG en NodeRed



Estructura basada en el llamado del control en Python mediante una función en NodeRed, Juan Escobar & Patricio Sulca.

Si bien es cierto que el proceso en los tres controles se repite para establecer la comunicación mediante la definición de los tópicos y red dentro de la programación en el IDE de Arduino que se carga al módulo ESP-8266.

El control LQG difiere en los parámetros de recepción de datos ya que recibe la salida estimada y salida actual debido al cálculo que está establecido en su modelado matemático. Por lo que se establece topic para cada salida y esto se demuestra en la Figura 4.16, en la programación para ser cargado en modulo ESP-8266.

Figura 4.16 Condición de recepción de topics

```
void callback(char* topic, byte* payload, unsigned int length) {

  String mensaje= topic;

  Serial.print("Mensaje Recibido del topico: ");
  Serial.println(mensaje);

if (mensaje == "entrada" )  {
    referen ="" ;
    for (int i = 0; i < length; i++) {
        referen+=((char)payload[i]);
    }
    //Serial.println(referen);
 }
else if (mensaje == "salida_estimada") {
  pestimado ="" ;
  for (int i = 0; i < length; i++) {
        pestimado+=((char)payload[i]);
  }
  //Serial.println(pestimado);
 }

 else if (mensaje == "salida_LQG")    {
  sLQG ="" ;
  for (int i = 0; i < length; i++) {
        sLQG+=((char)payload[i]);
        PWM = sLQG.toInt();
  }
  Serial.print("lqg");
```

Estas condiciones permiten al control efectuar un control predictivo en base al error, Juan Escobar & Patricio Sulca

4.1.3 Control LQI en WNCS

En los controles anteriores se establecieron claras diferencias en su modelado por lo que el control LQI no está absorto de hacerlo, este control tiene como base la realimentación de estados por lo que define su modelado matemático en la siguiente Ecuación 4.12.

$$J = \int_{t_1}^{t_2} \left[\left((x(t))^T Q x(t) \right) + m(t)^T + Rm(t) \right] dt \qquad \text{EC (4.12)}$$

Cabe considerar, por otra parte, que las matrices Q y R establecidas en la Ecuación 4.12, determina el error relativo y el consumo de energía para las señales de control que se expone con la Ecuación 4.13.

$$u(t) = -K * x(t) \qquad \text{EC (4.13)}$$

Retomando la expresión anterior, la matriz de ganancia K se presenta a continuación en la ecuación 4.14, permite optimizar el control en todo el sistema.

$$K = R^{-1} * B^T * P \qquad \text{EC (4.14)}$$

Donde P se precisa como una matriz positiva que debe compensar la ecuación reducida de Riccatti en la Ecuación 4.15.

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \qquad \text{EC (4.15)}$$

En la Ecuación 4.16, se aumenta el término para inhabilitar el error de régimen permanente, donde $\xi(t)$ pertenece a la señal de error del sistema.

$$u(t) = -Kx(t) + K_i\xi(t) \qquad \text{EC (4.16)}$$

Obteniendo las ganancias del controlador LQI, donde se estima la matriz optima aumentada especificada como Kt en la Ecuación 4.17.

$$Kt = [K \quad -K_i] \qquad \text{EC (4.17)}$$

Estableciendo la matriz aumentada conformada por las matrices del espacio de estado A, B, K, C y Ki, la parte integral es una matriz identidad y $\xi(t)$ es un vector de error que esta denotada en la Ecuación 4.18.

$$\begin{bmatrix}\dot{x}(t)\\ \dot{\xi}(t)\end{bmatrix} = \begin{bmatrix}A - BK & BK_i\\ -C & 0\end{bmatrix}\begin{bmatrix}x(t)\\ \xi(t)\end{bmatrix} + \begin{bmatrix}0\\ I\end{bmatrix}u(t) \qquad \text{EC (4.18)}$$

Con base a lo expuesto en las ecuaciones pertenecientes a la sección 4.1.3, se programa en java el control LQI que se observa en la Figura 4.18, teniendo en cuenta las ganancias que se obtuvieron mediante programación en Matlab, fundamentada en la estructura matemática del controlador que se puede apreciar en la Figura 4.17.

Figura 4.17 Procesamiento de la función de transferencia para control LQI

```
num = [0.4745];
den=[1,1.071,0.2813];
tf2 = tf(num,den)

% Transfer function to space states
ssl = ss(tf2);
sysd = c2d(ssl,0.102);
[A,B,C,D] = ssdata(sysd);

% Q and R matrix
Q = [6 0 0;
     0 0.05 0;
     0 0 0.08];
R = 0.001;
```

```
Command Window
            0.4745
  ----------------------
  s^2 + 1.071 s + 0.2813

Continuous-time transfer function.


K =

    9.2824    6.3032   -1.1571
```

Matriz de ganancias correspondientes al controlador LQI obtenidas en Matlab, Juan Escobar & Patricio Sulca.

Retomando lo mencionado sobre el control LQI este asume una referencia como en los anteriores controles, ya que aporta el establecimiento de un punto de referencia para la velocidad en la cual va a trabajar la planta, facilitando la optimización de la entrada y regulando de forma eficiente el comportamiento del sistema.

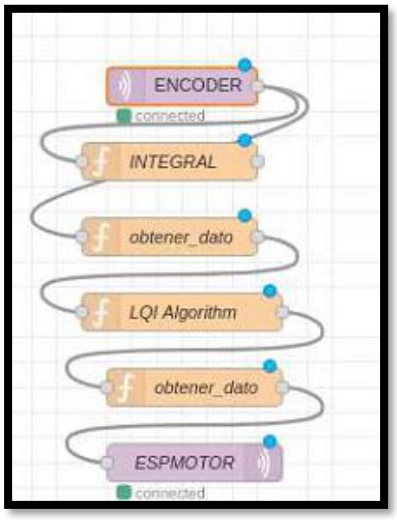
Figura 4.18 Programación en java del control LQI

```
flow.set('velocidad', msg.payload);
flow.set('integralError' , 0);
var lastvelocidadLocal ;
const K = [0.419608186659955, 0.570023806908869, -10.115289242421617];

var error;
var pwm;
var integralErrorLocal;

var stateVector = [];
    error = flow.get('referencia') -flow.get('velocidad');

    integralErrorLocal= (flow.get('integralError') || 0 )+ (error * 0.05);
    flow.set('integralError',integralErrorLocal);


    //velocidad
    stateVector[0] = flow.get('velocidad');

    lastvelocidadLocal=flow.get('lastvelocidad')||flow.get('velocidad');

    //derivative of pressure
    stateVector[1] = (stateVector[0] - lastvelocidadLocal ) / 0.05;
    flow.set('lastvelocidad',stateVector[0]);

    //Control law
    pwm = (((flow.get('integralError')||0)*(-K[2])))+ (stateVector[0] * (K[0])) + (stateVector[1] * (K[1]));


return { payload: pwm};
```

Proceso del control LQI adaptado a java en el nodo perteneciente al nodo de función en NodeRed, Juan Escobar & Patricio Sulca

Para la programación visual del control LQI se vincula al proyecto los nodos de función y protocolo MQTT, como con los dashboard para su interacción al momento de manipular la variable de entrada y apreciar el funcionamiento del control como resulta en la Figura 4.19.

Figura 4.19 Programación del control LQI en NodeRed



Control LQI propuesto en NodeRed, Juan Escobar & Patricio Sulca.

CAPITULO 5

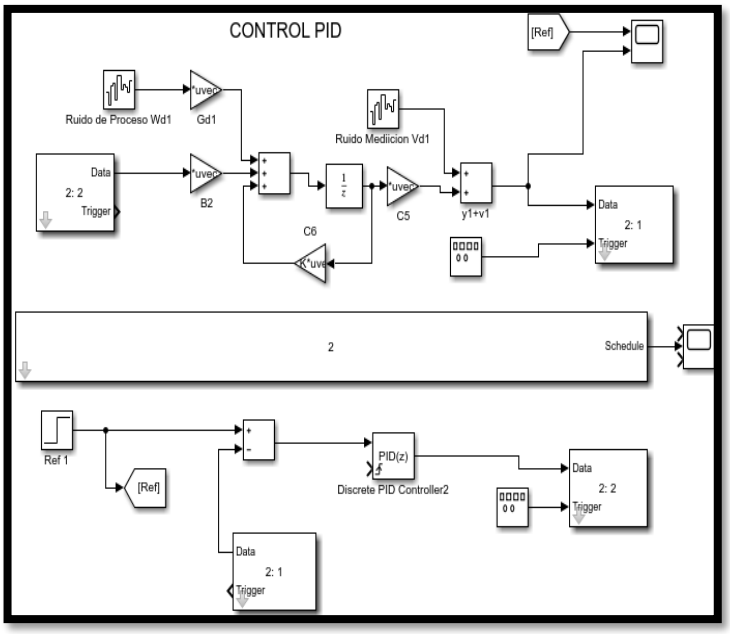
PRUEBAS Y RESULTADOS DE FUNCIONAMIENTO

En este capítulo, se somete a prueba cada uno de los controles; estas se realizan mediante simulación y prácticas reales. Para las simulaciones aplica el uso de la librería true time en simulink, esta permite simular el control en la red y tomar las diferentes respuestas de estabilización que tienen al momento de ser sometidos a una referencia o set point. De igual manera, el entorno visual de la herramienta NodeRed permite representar el control en tiempo real efectuado por cada uno de los algoritmos de control.

5.1 Resultados del control PID

En la Figura 5.1, se aprecia el bloque de control PID, los bloques de la conexión inalámbrica en los cuales se ingresa la referencia o setpoint y como también el ruido, este sirve tanto como para las perturbaciones y ocupación del canal en la red.

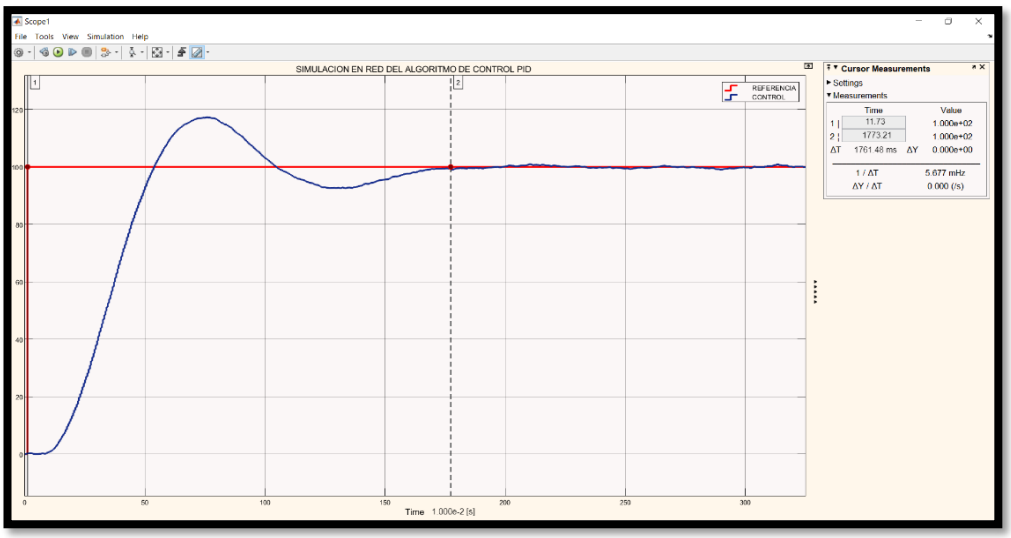
Figura 5.1 Control PID en red mediante la herramienta simulink



Simulación en red del control PID para obtener las pruebas de funcionamiento, Juan Escobar & Patricio Sulca

Por lo tanto, en la Figura 5.2 se aprecia el comportamiento del control PID y tiempo de estabilización para la planta en red simulada, presentando un sobreimpulso al momento de alcanzar el punto de referencia y manteniendo mínimas oscilaciones en la señal afianzada.

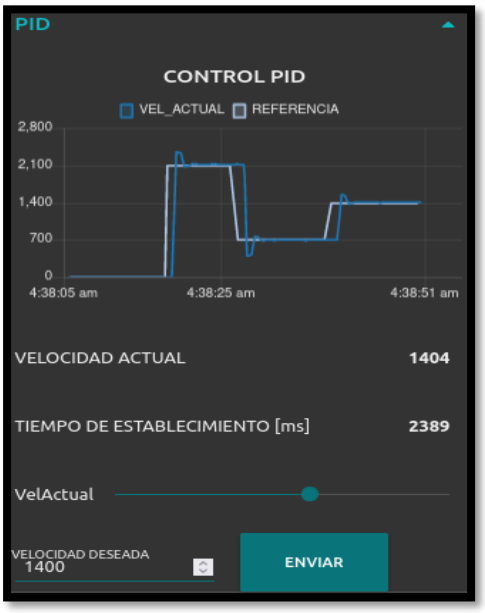
Figura 5.2 Salida de control PID en red simulada



Respuesta de estabilidad del control PID en red simulada mediante la herramienta simulink, Juan Escobar & Patricio Sulca

En este punto se evidencia la gráfica del control PID real, en el cual se fijan referencias iniciando desde cero, llevándola a 2100 RPM, posterior que el control este estabilizado se reduce el setpoint a 700 RPM, para finalizar con otra prueba fijada en 1400 RPM, se compara con la velocidad actual obteniendo la diferencia como se observa en la Figura 5.3, esta fluctúa dentro del rango de 0 a 4 RPM.

Figura 5.3 Variación de la referencia del control PID



Cambios de referencia pertenecientes al controlador PID, Juan Escobar & Patricio Sulca

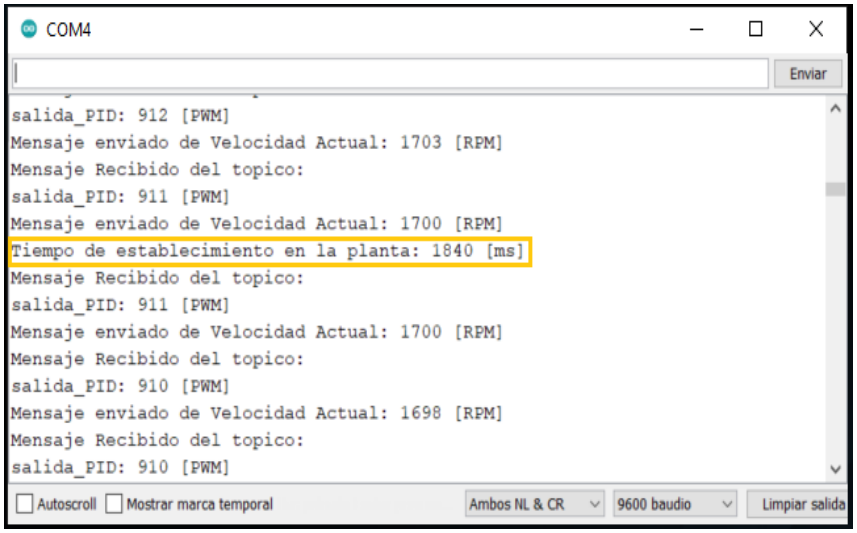
Al someter la planta a cambio de referencia se observa que en la Figura 5.4 y al igual que en la Figura 5.2, evidencian la presencia de sobreimpulso, teniendo un tiempo de establecimiento para el setpoint deseado de 2393 milisegundos.

Figura 5.4 Control PID sometido a perturbación y cambio de referencia

En el dashboard se pueden apreciar la velocidad actual y el tiempo de establecimiento del controlador PID, Juan Escobar & Patricio Sulca

Se aprecia la ventana serial del software Arduino en la Figura 5.5, el valor de 1840 ms como tiempo de establecimiento perteneciente al controlador PID, teniendo en cuenta la diferencia existente de 553 ms menos en relación a la apreciación del valor de la Figura 5.4.

Figura 5.5 Identificación del tiempo de establecimiento del controlador PID

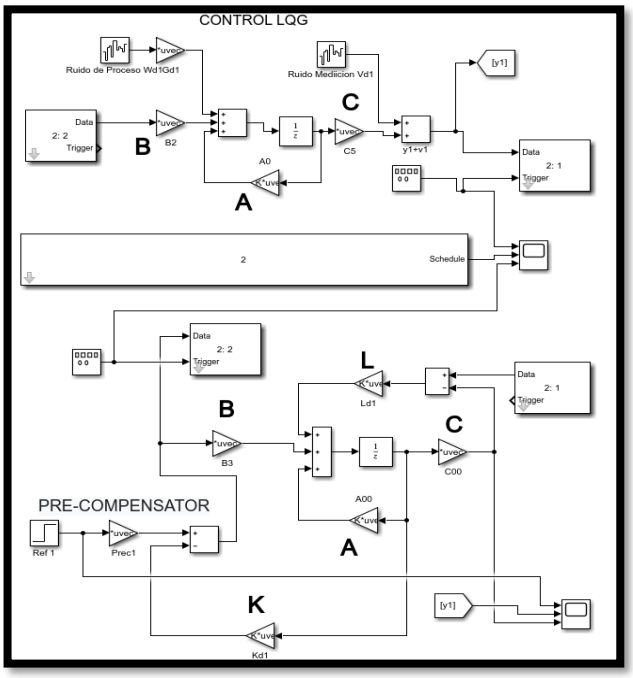


En esta Figura se presenta valores de suscripción y publicación entre el broker y la planta, Juan Escobar & Patricio Sulca

5.2 Resultados del control LQG

En esta perspectiva que maneja el control LQG en red simulada se tiene como bloques funcionales a las matrices de estado, ganancias que maneja el controlador, precompensador como también el observador, ingreso de la referencia y ruido para las perturbaciones en el sistema que se presenta en la Figura 5.6.

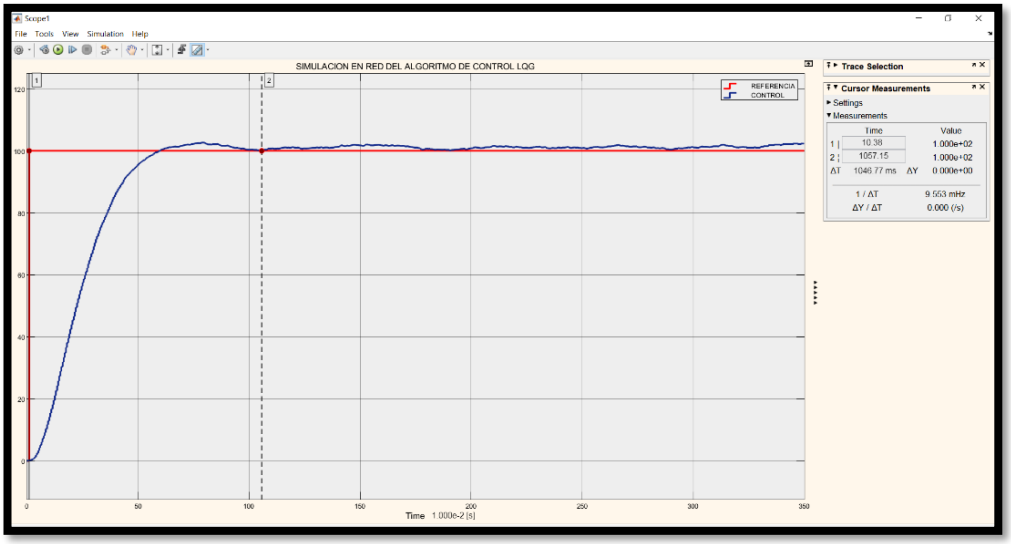
Figura 5.6 Control LQG en red mediante la herramienta simulink



Simulación en red del control LQG para pruebas de funcionamiento, Juan Escobar & Patricio Sulca.

En función de lo planteado para simulación en red del control LQG se obtiene la salida, esta se indica en la Figura 5.7, mostrando un control de estabilidad en la señal requerida con bajo sobreimpulso, pero con ciertas oscilaciones.

Figura 5.7 Salida del controlador LQG en red simulada



Simulación en red del control LQG para someter a pruebas de funcionamiento, Juan Escobar & Patricio Sulca.

El control LQG adopta la variación de setpoint con los mismos valores de prueba del control PID, siendo estos; 0 a 2100 RPM, de 2100 RPM a 700 RPM y 700 RPM a 1400 RPM, mostrando un desfase al momento de estabilizar la señal en la velocidad deseada, debido al error en estado estable por lo que en la velocidad actual existe una diferencia de 0 a 13 RPM con relación al setpoint, esta se muestra en la Figura 5.8.

Figura 5.8 Variación de la referencia en control LQG

Generación de señal establecida por la variación de la referencia en el control LQG, Juan Escobar & Patricio Sulca.

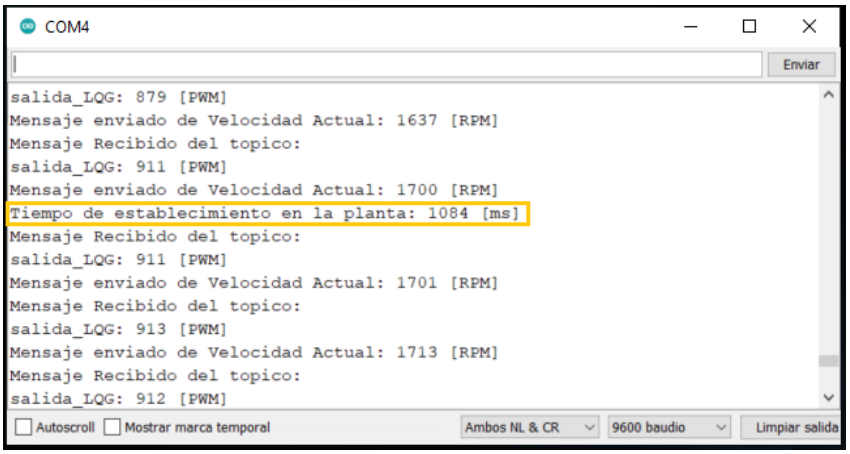
Teniendo en cuenta que el control LQG presenta eliminación del sobreimpulso, ya que minimiza el error porque su dinámica de trabajo es basada en el espacio de estados a diferencia del controlador PID, que realiza el cálculo de la diferencia que existe entre la velocidad actual y la velocidad deseada, asumiendo un tiempo de establecimiento de 1682 milisegundos, por consecuencia de las variaciones externas que se le aplican a la planta, dando los valores que se muestran en la Figura 5.9.

Figura 5.9 Cambio de referencia y aplicación de perturbaciones en el control LQG

Gráfica de perturbación y cambio de referencia en el control LQG, Juan Escobar & Patricio Sulca

En la ventana serial de la Figura 5.10, se resalta como principal el tiempo de establecimiento de 1084 ms del controlador LQG, con una diferencia menor de 598 ms en comparación al tiempo mostrado en la Figura 5.9.

Figura 5.10 Tiempo de establecimiento del controlador LQG

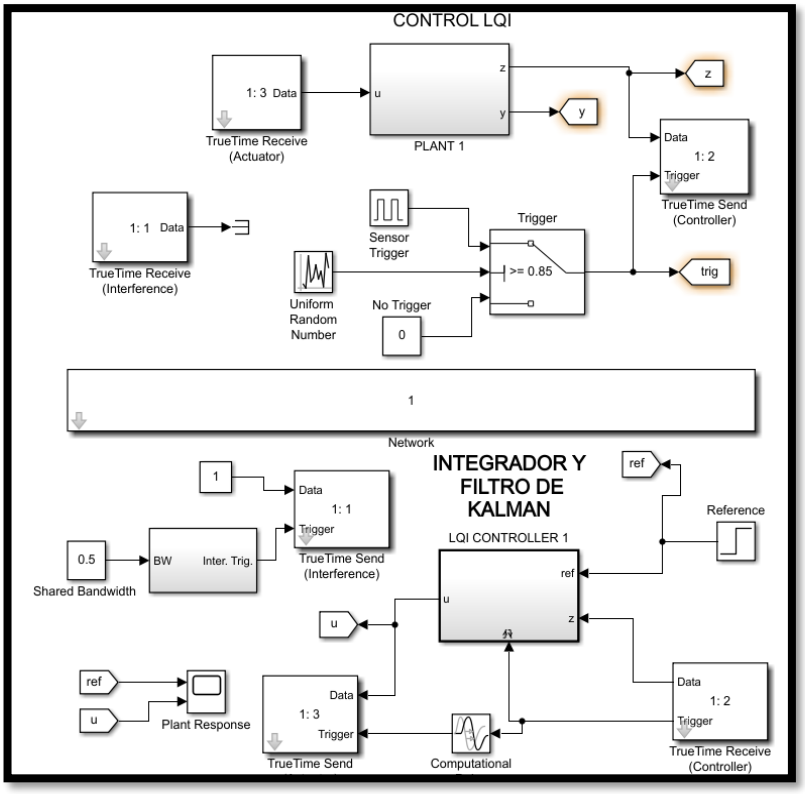


Se detalla los valores de velocidad actual y salida pertinentes al controlador LQG, Juan Escobar & Patricio

5.3 Resultados del control LQI

Dentro del marco de simulación de la Figura 5.11, el control LQI evidencia el bloque que contiene internamente la parte del integrador y el filtro de Kalman, adicional funciona con las matrices de estado, ganancias y el ruido para ocupación del canal.

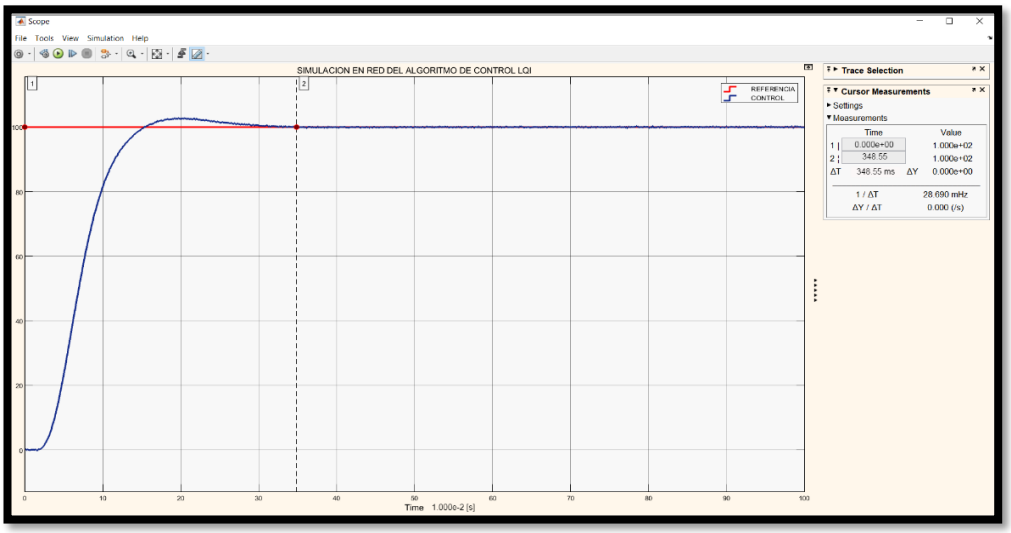
Figura 5.11 Control LQI en red mediante la herramienta simulink



Simulación en red del control LQI sometida a pruebas de funcionamiento, Juan Escobar & Patricio Sulca.

En lo esencial se debe señalar que la salida LQI muestra bajo sobre impulso y no presenta fluctuaciones en su control, se pueden observar en la Figura 5.12.

Figura 5.12 Salida del controlador LQI en red simulada



Simulación en red del control LQI para someter a pruebas de funcionamiento, Juan Escobar & Patricio Sulca

Por otro lado, en la Figura 5.13 perteneciente al control LQI el cual posee una alteración de la salida actual en un rango de 0 a 3 RPM, con respecto a la velocidad deseada de la última prueba, para lo cual se han tomado diversos puntos de referencia, como se han establecido en los anteriores controladores antes mencionados.

Figura 5.13 Cambio de punto de referencia en control LQI

Interfaz en NodeRed pertenecientes al control LQI sometido a cambios de referencia, Juan Escobar & Patricio Sulca

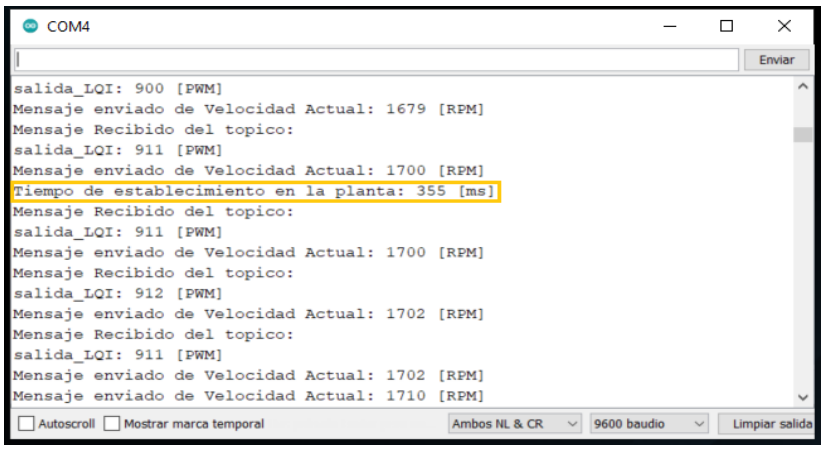
En la Figura 5.14 se aprecia la gráfica de respuesta del control LQI ante la presencia de alteración del punto de referencia, observando un tiempo de 1003 milisegundos para la estabilidad del control.

Figura 5.14 Control LQI sometido a perturbaciones

Salida de respuesta a la variación de referencia y fluctuación de la señal requerida, Juan Escobar & Patricio Sulca

En la Figura 5.15, se aprecia como valor primordial el tiempo de establecimiento del controlador LQI, siendo este de 355 ms, presentado una diferencia de 648 ms con respecto al valor presentado de la Figura 5.14.

Figura 5.15 Tiempo de establecimiento del controlador LQI



Valores de comunicación para el control LQI, Juan Escobar & Patricio Sulca

En este estudio los controles óptimos LQI y LQG no presentan sobreimpulso por lo cual se descarta la comparación de los parámetros de tiempo pico, subida y de crecimiento, para enfocarse en el tiempo de establecimiento. Porque en los controladores aplicados en este proyecto, presentan retardos variables debido a los efectos de actualización y lentitudes propios de la red, el tiempo de estabilización es parámetro que nos permitirá optimizar el uso del canal para futuros estudios.

Tabla 5.1 Valores de respuesta obtenidos en simulación y aplicación real

Control	**Tiempo de establecimiento**		
	En simulación [ms]	**En planta real [ms]**	**En dashboard [ms]**
PID	1761,48	1840	2393
LQG	1046,77	1084	1682
LQI	348,55	355	1003

Resultados obtenidos en la aplicación en red simulada y física de los tiempos de establecimiento de cada control para la señal deseada, Juan Escobar & Patricio Sulca

En la Tabla 5.1 se consolidan los resultados de tiempos de establecimiento de los diversos controles aplicados a la planta. Estos son: en simulación con el software Matlab, como también en aplicación real obtenido del serial de Arduino y mediante la apreciación de datos en el dashboard habilitado en Node Red, se procede a calcular el error porcentual de cada controlador mediante la Ecuacion 5.1, teniendo en cuenta que las pruebas realizadas se las hicieron al mismo valor de referencia, para luego establecer las comparaciones respecto a la simulación, aplicación real y dashboard, dando como resultado los datos presentados en la Tabla 5.2.

$$e = \frac{|V_r - V_s|}{V_r} * 100 \qquad \text{Ec. (5.1)}$$

Tabla 5.2 Error entre tiempo de simulaciones y aplicación funcional

Control	**Simulaciones vs Planta real**	**Simulaciones vs Dashboard**
PID	4.27%	26,39%
LQG	3.43%	37,77%
LQI	1.87 %	65,25%

Cálculo del error porcentual con los tiempos de estabilidad en simulación, real aplicado y dashboard, Juan Escobar & Patricio Sulca

Los porcentajes de errores altos en la comparación tiempos de simulación y dashboard pertenecientes a la Tabla 5.2, se debe a la actualización del dato en el dashboard, mas no por el control, para corroborar se procede a utilizar los tiempos de dashboard menos los tiempos de establecimientos de la planta real de la Tabla 5.1, para así obtener la diferencia y el promedio de retardo en la actualización de los datos recibidos en cada uno de los controladores, esto se visualiza en la Tabla 5.3.

Tabla 5.3 Diferencia en los tiempos de establecimiento

Control	Diferencia: Tiempo de dashboard – Tiempo planta real [ms]
PID	553
LQG	598
LQI	648
Promedio	599,67

Cálculo del promedio de retardo en la recepción de datos, Juan Escobar & Patricio Sulca

En la Tabla 5.4 se procede a comparar los tiempos de respuesta de establecimiento con los datos de la Tabla 5.1 de los diferentes controladores, determinando el porcentaje de rapidez del control más eficaz con referencia al controlador que tiene más retardo.

Tabla 5.4 Comparación de los tiempos de respuesta

Controles	Eficacia de respuesta	
	En simulación	En aplicación real
LQI vs PID	80.21 %	57.30 %
LQI vs LQG	66.7 %	47.01 %
LQG vs PID	40.57 %	19.43 %

Comparación de los tiempos de respuesta en los controladores, Juan Escobar & Patricio Sulca

En la Tabla 5.5, se puede verificar las velocidades obtenidas de los controladores a diferentes puntos de referencia en la prueba uno, mientras que en la prueba dos se establece una misma referencia para los tres controladores, para luego deducir el error en estado estable que se considera en la Tabla 5.6.

Tabla 5.5 Velocidades de referencia en los controladores

Control	Velocidad de establecimiento			
	Deseada [RPM]		Alcanzada [RPM]	
	Prueba #1	Prueba #2	R. Prueba #1	R. Prueba #2
PID	1400	1700	1404	1698
LQG	1400	1700	1413	1695
LQI	1400	1700	1397	1702

Adquisición de velocidades con diferente punto de referencia en cada uno de los controladores, Juan Escobar & Patricio Sulca

Tabla 5.6 Error en estado estable

Control	Error: Velocidad deseada vs Velocidad Alcanzada
PID	0,29%
LQG	0,93 %
LQI	0,21 %

Cálculo del error estado estable de los controladores propuestos en este trabajo, Juan Escobar & Patricio Sulca

Al comparar las gráficas de la Figura 5.2, Figura 5.7 y Figura 5. l2, se identifica que el único control que posee un alto sobreimpulso es el PID, mientras que el control LQG carece de sobreimpulso y teniendo al controlador LQI con un mínimo sobreimpulso, que, en comparación con los anteriores controladores, tiene una rápida respuesta ante perturbaciones y señal de consigna, siendo este el mejor controlador para un óptimo funcionamiento en simulación de la planta establecida en la Ecuación 4.1.

Verificando la Figura 5.4, Figura 5.9 y Figura 5.14, debe señalarse que todos los controladores por encontrarse funcionando en la red tienen un retraso al momento del cambio de setpoint. Además, esta demora persiste también al ser sometidos a perturbaciones como se aprecian en la Figura 5.3, Figura 5.8 y Figura 5.13.

Como se vincula en la Tabla 5.2, el error en estado estable de los controladores implementados se origina por los diferentes factores que se pueden presentar durante los periodos transitorios de funcionamiento, como la ocupacional del canal, latencia de red y a las imperfecciones en los componentes de la planta, esto se evidencia en la Tabla 5.3, presentando un retardo en la actualización del dashboard, para cada

controlador con un promedio aproximado de 600 ms. Este error es uno de los principios que se utilizan para medir la calidad y operabilidad de los sistemas de control involucrados en el control de un proceso (Merino, Lino, Ortiz, Gordillo, & Álvarez, 2017).

Los controladores fueron sometidos a varios cambios en el punto de consigna para corroborar el desfase antes mencionado, cabe recalcar que este retardo no disminuye la calidad de los controladores PID, LQG y LQI, sin embargo, si afecta el tiempo de respuesta para el establecimiento de la señal deseada. Donde se determina que el controlador LQI mantiene su operatividad en la red a pesar del error obtenido en la Tabla 5.2, esto en aplicación real y como también en simulación destacando su rapidez con respecto al control PID y LQG, siendo esto apreciado en la Figura 5.2, Figura 5.7 y Figura 5.12.

Del cálculo obtenido en la Tabla 5.4, se tiene un mayor porcentaje de error en estado estable del controlador LQG, teniendo como precedente la variación de referencias en la entrada, manteniéndose el mismo en cualquier punto de consigna dentro del rango de 0 a 14 RPM.

CONCLUSIONES

En virtud de los conocimientos adquiridos sobre los controladores propuestos en este trabajo, se concluye que permiten validar diferentes estrategias de control basados en su modelado matemático, estos modelos de control pueden trabajar directamente en la nube, lo cual facilita su adaptación e interpretación de su estructura con herramientas de programación IoT.

En este trabajo se evidencio que se debe obtener la función dinámica del sistema dentro de los parámetros de muestreo, definiendo un tiempo de muestreo de 0.5, ya que los datos de entrada (PWM) y salida (RPM) en el motor se obtienen de forma experimental, reduciendo de forma considerable el aliasing en la señal y a su vez evitando mediciones de señal incorrecta.

Dentro del análisis de los algoritmos de control propuestos, se valida la programación con la herramienta NodeRed directamente en la máquina virtual en la nube y al analizar el desempeño de los controladores en la nube se comprobó que para mantener la comunicación bidireccional entre el broker y la planta sin tener perdidas de datos se debe crear un canal único y seguro que se origina con un proxy obtenida de la herramienta remote.it.

Se demuestra con los resultados obtenidos que el controlador que posee mejores características de respuesta ante perturbaciones o cambios del punto de referencia es el control LQI, considerando las comparaciones de trabajo de cada uno de los controladores basados en el tiempo de estabilización se tiene que el LQI posee una respuesta de control del 57.30 % más rápida respecto al controlador PID y del 47.01 % con relación al controlador LQG, presentando a su vez particularidades considerables en la disminución del sobreimpulso de la salida de control, ya que invalida las fluctuaciones en la entrada por su parte integral, señalando que el error en estado estable es mínimo con un valor del 0,21%. Sin embargo, cabe recalcar que el controlador LQG presenta un mejor tiempo de estabilidad en la respuesta con un 19.43 % respecto al control PID, pero posee un mayor error en estado estable del 0.93%. Por ultimo el porcentaje de error en el control LQI conseguido en la comparativa de la simulación con red y aplicación real en red, tiene un valor de 65,25 %, esta se debe

por el retraso en la comunicación que oscila aproximadamente entre los 600 ms, pese a esto sigue manteniendo su rapidez de control.

RECOMENDACIONES

Una de las características a tomar en la configuración de la máquina virtual en Google Cloud, es de habilitar un procesador con recursos medios para que al momento de utilizarla en la red no se detenga o presente problemas en su uso recurrente.

Para evitar las oscilaciones que se presentan en la adquisición de datos de velocidad es importante que el sensor encoder utilizado sea preciso y confiable, minimizando así el error del valor de referencia deseado al momento de ejercer los controladores sobre el motor.

Para efectos de simulación de los controles se debe tener en cuenta la librería true time de Matlab, que sirve para efectuar la representación en red al momento de utilizar los bloques de programación en la herramienta simulink teniendo en cuenta el llamado de la librería mediante el comando init_truetime desde el comand window de Matlab.

Precisar una cantidad alta de datos que se obtienen del sensor encoder pertenecientes a la velocidad del motor, ya que desde estas cantidades se precisa la función dinámica del sistema, teniendo en cuenta que si se obtiene una función de transferencia fiable de la planta los controladores tendrán una alta probabilidad de un funcionamiento óptimo.

Desde un inicio se debe realizar la adaptación de los controladores en Matlab para obtener los valores de cada una de las variables que necesitan ser interpretadas en los programas Python y NodeRed, como también de ser simulados con la herramienta simulink verificando si existe el funcionamiento adecuado dada por la función del sistema.

Al momento de agregar un dispositivo en Remoteit se debe tener en cuenta en que protocolo se va a trabajar, ya que se puede escoger diversas opciones que proporciona este programa para establecer la conectividad de dispositivos IoT administrados desde un enlace seguro.

Cuando se configura el broker del controlador en NodeRed utilizando la aplicación Remoteit se debe configurar el server de los tópicos como “localhost:1883”, para evitar el cambio aleatorio de dirección IP, manteniendo siempre la conectividad del broker.

REFERENCIAS

Araújo, L. R. (2018). ANÁLISE DO COMPORTAMENTO DO DFIG EM UMA REDE ELÉTRICA DIANTE DE AFUNDAMENTOS DE TENSÃO E FALHAS INTERNAS UTILIZANDO TÉCNICAS DE CONTROLE PI E LQI. Sobral: UNIVERSIDADE FEDERAL DO CEARÁ . Obtenido de http://www.repositorio.ufc.br/handle/riufc/40137

Balseca, S. L., & Durán, E. P. (julio de 2019). CONTROL ROBUSTO DE UN MOTOR AC MEDIANTE WNCS (WIRELESS NETWORK CONTROL SYSTEM). Quito, Pichincha, Ecuador. Obtenido de https://dspace.ups.edu.ec/handle/123456789/17577

Barrena, S. S. (2019). Aplicación Android para compartir coche basada en la tecnología Firebase. Sevilla, España. Obtenido de https://idus.us.es/handle/11441/91524

Becerra, D. E. (2016). Diseño, construcción, programación y evaluación de un robót móvil omnidireccional. *SENNOVA, 2*(2), 70-91. Obtenido de http://revistas.sena.edu.co/index.php/sennova/article/view/550

Carlucho, I., Menna, B., Paula, M. D., & Acosta, G. G. (2019). Comparación de controlador PID versus control lineal LQG para un móvil autónomo submarino. *IEEE, 15*(28). Recuperado el 19 de febrero de 2021, de http://b-dig.iie.org.mx/BibDig2/P16-0417/papers/IEEE_ARGENCON_2016_paper_217.pdf

Chotai, J., & Narwekar, K. (2017). Modelling and Position control of Brushed DC motor. *IEEE*. doi:10.1109/ICAC3.2017.8318792

Chung, T. D., Ibrahim, R. B., Asirvadam, V. S., Saad, N. B., & Hassan, S. M. (2015). Simulation of WirelessHART Networked Control System with Packet Dropout . *IEEE*. doi:10.1109/ASCC.2015.7244389

Díaz, H., Armesto, L., & Sala, A. (2019). Metodología de programación dinámica aproximada para control óptimo basada en datos. *Revista Iberoamericana de Automática e Informática Industrial*, 273-283. doi:https://doi.org/10.4995/riai.2019.10379

Fernández, A. I. (26 de abril de 2019). Diseño de reguladores para el control estocástico óptimo basados en la solución de la ecuación de Hamilton-Jacobi-Bellman. Cartagena, Colombia. Obtenido de https://repositorio.upct.es/handle/10317/7872

Gomez, H. A., & Reina, J. A. (Diciembre de 2015). Mecanismo de integración de PANDABOARD en el IDE de Arduino en un contexto de IOT. Popayán , Cauca, Colombia. Obtenido de http://repositorio.unicauca.edu.co:8080/handle/123456789/1512

González, J. M., & Castro, J. C. (2020). Bloqueo del sistema de encendido de un vehículo eléctrico, mediante la implementación de un alcoholímetro. *Dominio de las Ciencias*, 177-196. doi:https://doi.org/10.23857/pocaip

Light, R. A. (2017). Mosquitto: server and client implementation of the MQTT protocol. *JOSS*, 265. doi:10.21105/joss.00265

Liuliu, Y., Fang, W., Sen, H., Yuchen, L., Hao, S., Qingjie, L., . . . Quanzhao, W. (2016). Application of Drive Circuit Based on L298N in Direct Current Motor Speed Control System. *SPIE, 10153*. doi:https://doi.org/10.1117/12.2246555

Maleh, R., Fudge, G. L., Boyle, F. A., & Pace, P. E. (2012). Analog-to-Information and the Nyquist Folding Receiver. *IEEE*, 564-578. doi:https://doi.org/10.1109/JETCAS.2012.2223611

Merino, M. J., Lino, E. A., Ortiz, M. M., Gordillo, F. B., & Álvarez, R. A. (2017). *Elementos básicos del control de procesos.* Alicante: 3Ciencias. doi:http://dx.doi.org/10.17993/EcoOrgyCso.2017.25

Mora, W. B. (18 de septiembre de 2020). Desarrollo de aplicaciones prácticas de sistemas dinámicos en MatLab/Simulink y Octave para la asignatura de control y automatismo. Guayaquil, Guayas, Ecuador. Obtenido de http://repositorio.ucsg.edu.ec/handle/3317/15529

Muñoz, G. R. (25 de septiembre de 2018). Sistema de medida de temperatura basado en NodeMCU y Android. Leganés, Madrid, España. Obtenido de https://e-archivo.uc3m.es/handle/10016/29308

Muñoz, T. C. (25 de septiembre de 2018). Sistema de medida de temperatura basado en NodeMCU y Android. Madrid, España. Obtenido de https://e-archivo.uc3m.es/handle/10016/29308

Nuñez, E. M. (Marzo de 2021). Sistema de control de tráfico vehicular aplicando la arquitectura FOG COMPUTING. Ambato, Ecuador. Obtenido de https://repositorio.uta.edu.ec/jspui/handle/123456789/32676

Ortega, A. T. (2017). Ánalisis de un modelo predictivo basado en Google Cloud y Tensor Flow. Madrid, España. Obtenido de https://eprints.ucm.es/id/eprint/46646/

Pillajo, C., & Hincapie, R. (2018). *Wireless Network Control System De la Teoría a la Práctica.* Quito, Ecuador: Abya-Yala.

Ríos, J. R., Mora, N. M., Ordóñez, M. P., & Sojos, E. L. (23 de Septiembre de 2016). Evaluación de los Frameworks en el Desarrollo de Aplicaciones Web con Python. *Revista Latinoamericana de Ingeniería de Software, 4*(4). doi: https://doi.org/10.18294/relais.2016.201-207

Rocha, A. E. (Enero de 2011). Desarrollo de un controlador óptimo para un aeromodelo tipo avioneta. Bogotá, Colombia. Obtenido de https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjrvrTNz4bxAhXhGVkFHeKYAyIQFjAFegQIBhAD&url=https%3A%2F%2Frepositorio.uniandes.edu.co%2Fbitstream%2Fhandle%2F1992%2F14687%2Fu442627.pdf%3Fsequence%3D1&usg=AOvVaw1--qdY

Saputra, L. K., & Lukito, Y. (noviembre de 2017). Implementation of Air Conditioning Control System Using REST Protocol Based on NodeMCU ESP8266. *IEEE*, 126-130. doi:10.1109/ICON-SONICS.2017.8267834

Silva, C. W. (2016). *Sensors and Actuadors - Engeneering System Instrumentation* (2 ed.). Boca Ratón: CRC Press. Obtenido de https://books.google.com.ec/books?id=ukZOCgAAQBAJ&pg=PA433&dq=sensor+encoder&hl=es-419&sa=X&ved=2ahUKEwj0uYnFvYbxAhUyF1kFHU1ZAZoQ6AEwBnoECAUQAg

Suárez, M. Á., & Rodríguez, D. d. (Enero de 2018). Diseño de una plataforma en la nube con NodeRED para Internet de las cosas. Las palmas, Gran Canaria, España. Obtenido de https://accedacris.ulpgc.es/bitstream/10553/95257/1/TFG-CarlosEspaciosLopez.pdf

Tapia, V. A. (2020). Diseño de un sistema SCADA con control remoto, usando un controlador lógico programable (PLC), un sistema CCTV, un servidor VNC y el software Team Viewer, aplicado a la seguridad residencial. Arequipa, Perú. Obtenido de http://hdl.handle.net/20.500.12773/11509

Terán, A. F., & Narváez, D. P. (junio de 2018). SISTEMA DE CONTROL EN RED INALÁMBRICA (WNCS) IMPLEMENTADO AL CONTROL DE VOLTAJE DE UN GENERADOR DC DIDÁCTICO. Quito, Pichincha, Ecuador. Obtenido de http://dspace.ups.edu.ec/handle/123456789/15651

Vega, E. G., Acuña, F. E., Acuña, F. E., & Velázquez, M. Á. (2019). Análisis de prototipo de vehículo autónomo con base en sistema de visión y bajo concepto del internet de las cosas en plataforma Intel Edison. *Pistas Educativas, 41*, 271-289. Obtenido de http://www.itc.mx/ojs/index.php/pistas/article/view/2154

Yuan, H., Xia, Y., Yang, H., & Yuan, Y. (2018). Resilient control for wireless networked control systems under DoS attack via a hierarchical game. *Wiley*, 1-20. doi:https://doi.org/10.1002/rnc.4272

Zambrano, E. I. (Febrero de 2020). Analisis comparativo de técnicas de optimización para la sintonización de controladores PID adaptativos. Quito, Pichincha, Ecuador. Obtenido de http://bibdigital.epn.edu.ec/handle/15000/20917

ANEXOS

Anexo A: Programación en Arduino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
void ICACHE_RAM_ATTR handleInterrupt();

//const char* ssid = "ine4c";
//const char* password = "ine4c4000";
//const char* mqtt_server = "192.168.0.102";

const char* ssid = "LUMBIBOTS";
const char* password =    "1722958939";
//const char* mqtt_server = "192.168.0.101";   /// example 192.168.0.19
const char* mqtt_server = "proxy22.rt3.io:36905";

WiFiClient espClient3;
PubSubClient client(espClient3);

volatile int contador = 0, vel=0;
const int PWMM = 2;
const int ENCO = 1;
const int buzzer = 0;
const int PWML = 5;
float integral = 0.718;
float x1,px22,px33,px44,px55,px1,px2,px3,px4,px5;
float x2 =1.0;
int cont ;
String pestimado;
float inuevo ;
float pestimado1;
float fkp, fki, fkd;
float resultado, referencia,resultado1;

 String referen;
 String sLQG;
 char rpm1 [10];
 char error [10];
 char contador1 [10];
 int ref;
```

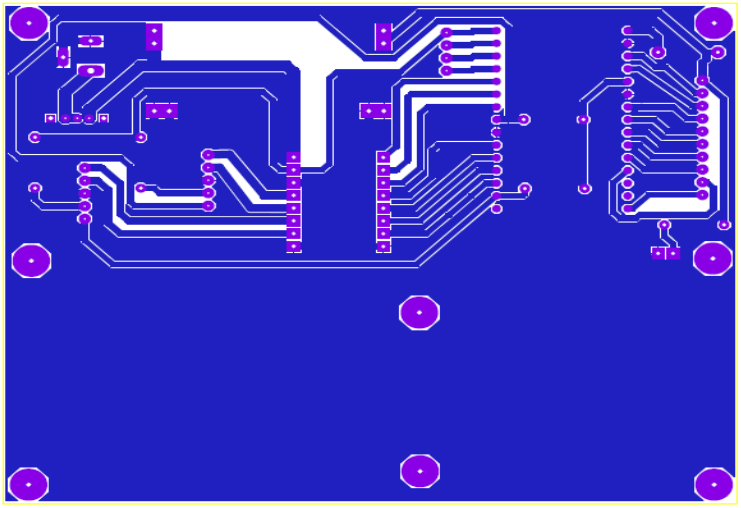
```
//////////////ENCODER/////////////////
float  dpwm;
int  dp,dp1;

unsigned int rpm = 0;          // Revoluciones por minuto calculadas.
long lastMsg = 0, lastvel=0;
char msg[50];
int value = 0,btn_las_state=0, estado_bt=0, encendido=0;
int pot=0, pot_last=0, pot1=0;
int PWM;

void setup() {

  pinMode(PWMM, OUTPUT);
  pinMode(ENCO, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(PWML, OUTPUT);
  analogWrite(PWMM, 0);

  attachInterrupt(ENCO, interrupcion0, RISING); // Configuración de la interrupción 0, donde esta conectado.
  rpm = 0;
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void setup_wifi() {
  delay(10);

  Serial.println();
  Serial.print("Conectando a:");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  String mensaje= topic;

  Serial.print("Mensaje Recibido del topico: ");
  Serial.println(mensaje);

if (mensaje == "entrada" )  {
    referen ="" ;
    for (int i = 0; i < length; i++) {
        referen+=((char)payload[i]);
    }
 }

else if (mensaje == "salida_estimada") {
  pestimado ="" ;
  for (int i = 0; i < length; i++) {
        pestimado+=((char)payload[i]);
   }
 }
 else if (mensaje == "salida_LQG")    {
  sLQG ="" ;
  for (int i = 0; i < length; i++) {
        sLQG+=((char)payload[i]);
        PWM = sLQG.toInt();
  }
 }

   pestimado1 = pestimado.toInt();
   referencia = referen.toFloat();

   if ( x1 < 0  ) { PWM --;  }
   else if ( x1 > 0  ) {PWM ++ ;}

   analogWrite(PWMM,PWM);
```

```
   Serial.print("pwm");   Serial.println(PWM);
}
void loop() {
  if (!client.connected()) { reconnect(); }
  client.loop();

  digitalWrite(buzzer, LOW);
 // analogWrite(PWMM,PWM);
 ////////////////////ENCODER

long now = millis();
if (now - lastMsg > 500) {
  lastMsg = now;

 // ***************** publicador **********************
    snprintf (rpm1, 75, "%ld", vel);
    dtostrf(x1,0, 0, error);
    delay (50);
    client.publish("DIF",error);
    //if ( x1 <= -15  && x1 >= -15){
    client.publish("ENCODER",rpm1);
    //}

    Serial.println(vel);
 }

  if (now-lastvel>999){
    lastvel = now;
    vel= round (contador*1.6666666666667);
    //if ( x1 >8 && x1 < 15 ){ vel = referencia;}
    //else if ( x1 < -15 &&  x1 > -5 ){ vel =referencia;}
    contador=0;
  }
 // ******************ERROR****************//
    x1 =  pestimado1 - vel ;
    x2 = pestimado1/vel ;
    Serial.print("pwm");   Serial.println(PWM);
}
void reconnect() {
```

```
   while (!client.connected()) {
     Serial.print("Attempting MQTT connection...");
     // Attempt to connect
     if (client.connect("ESP8266Client")) {
       Serial.println("connected");
       client.subscribe("salida_estimada");
       client.subscribe("entrada");
        client.subscribe("salida_LQG");
     } else {
       digitalWrite(buzzer, HIGH);
       Serial.print("failed, rc=");
       Serial.print(client.state());
       Serial.println(" try again in 5 seconds");
       // Wait 5 seconds before retrying
       delay(5000);
       digitalWrite(PWMM, LOW);
     }
   }
 }
 ICACHE_RAM_ATTR void interrupcion0(){
   contador++;
}
```

Pistas de la placa PCB



Apreciación de los caminos o pistas según las normas de elaboración de placas PCB para evitar corto circuitos entre los dispositivos usados, Juan Escobar & Patricio Sulca