Proyecto final: Control del péndulo de Furuta con lógica difusa

Erick Santiago Dumas Puma
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
erick.dumas@ucuenca.edu.ec

Jorge Geovanny Zhangallimbay Coraizaca
Facultad de ingeniería
Universidad de Cuenca
Cuenca, Ecuador
jorge.zhangallimbay@ucuenca.edu.ec

I. Introducción

El péndulo de Furuta es un ejemplo clásico de un sistema no lineal, que presenta desafíos significativos para el control debido a su dinámica compleja. Una opción efectiva para controlar sistemas no lineales es la aplicación de la lógica difusa, que permite manejar la incertidumbre y la imprecisión de estos sistemas de manera flexible y adaptable.

En este documento se presenta el diseño e implementación de un controlador para un péndulo de Furuta utilizando lógica difusa, específicamente mediante una máquina de Mamdani. Se implementó un algoritmo en Arduino para equilibrar el péndulo controlando tanto la posición del brazo como la del péndulo. El sistema genera una señal PWM con un sentido de giro basado en el error de posición y su variación respecto al error anterior. Estas variables se fuzzifican utilizando funciones de membresía que aplican el método de mínimos de Mamdani. La salida se defuzzifica mediante el método ponderado, empleando valores de singletons obtenidos a través de pruebas exhaustivas para optimizar el control del equilibrio. La lógica difusa proporciona un control robusto al ponderar las variables de entrada en todos los estados posibles, asegurando que cada función de membresía impacte en la salida y evitando valores fijos que podrían desestabilizar el sistema.

II. Descripción metodológica

A. Hardware usado

Para la implementación del sistema de control del péndulo se utilizaron los siguientes componentes:

- **Péndulo de Furuta:** consiste en un brazo que gira en el plano horizontal, montado sobre el eje de un motor, y un péndulo que rota en el plano vertical, ubicado en el extremo del brazo horizontal. La figura 1 muestra una representación gráfica del dispositivo, donde se puede observar que incluye un motor con encoder y un potenciómetro de precisión.
  - Potenciómetro de precisión (WDD35D4-5k): Este potenciómetro se alimenta con 5 V, y se mide el valor en su pin intermedio para determinar la posición del péndulo en función del voltaje. Es necesaria una calibración mediante mediciones directas para establecer el valor del voltaje correspondiente al equilibrio del péndulo.
  - Motor con encoder: Este motor controla el brazo del dispositivo. Recibe una señal de control que ajusta la posición del péndulo. El encoder mide la posición del brazo para el control de su posición.

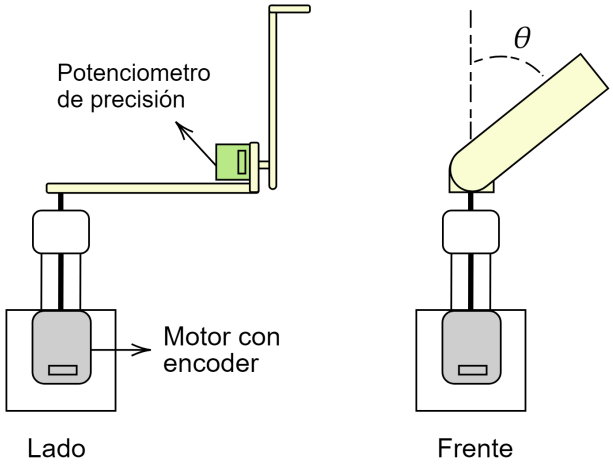

Fig. 1: Representación gráfica del péndulo de Furuta.

- **Puente H: Módulo TB6612FNG** Este módulo proporciona la potencia al motor según las entradas a sus pines, como se muestra en la Figura 2.

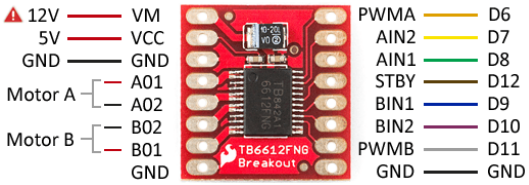

Fig. 2: Módulo TB6612FNG puente H utilizado para la etapa de potencia del motor. Tomado de [1]

En la figura anterior, el estado de los pines BIN1 y BIN2 determina el sentido de giro del motor B, mientras que PWMB controla la velocidad del motor mediante una

señal PWM.

- **Arduino UNO:** Encargado de la adquisición de la señal de salida del potenciómetro de precisión y de la posición del encoder para el procesamiento a través del algoritmo de control. Además, genera la señal PWM al puente H y controla el sentido de giro mediante las salidas lógicas en los pines digitales que ingresan al puente H.

  Para la adquisición de la señal del potenciómetro, se realiza un proceso de muestreo. Esto permite controlar el tiempo entre muestras y asegurar que el valor de la derivada del error sea más consistente. Para lograrlo, se utiliza la biblioteca `MsTimer2`, la cual permite realizar muestreos cada $5, \text{ms}$.
  Para la adquisición de los datos del encoder, se utiliza otro temporizador del Arduino. Esto habilita las interrupciones en los pines conectados al encoder.
  La figura 3 ilustra un diagrama el diagrama de flujo para el cálculo de la señal de control del motor.

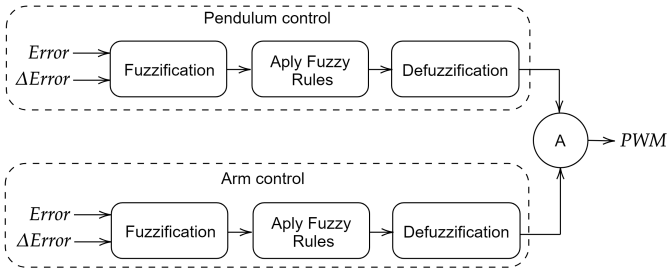

Fig. 3: Flujo de cálculo de la señal PWM.

B. Explicación de parámetros

El modelo adoptado para este proyecto es un sistema de una entrada y dos salidas:

- **Señal de control del motor:** Señal PWM enviada desde el microcontrolador al puente H que controla la velocidad del motor. El sentido de giro está controlado por la combinación de los pines digitales, como se muestra en la Tabla I.

BIN1	BIN2	Estado
LOW	LOW	OFF
LOW	HIGH	HORARIO
HIGH	LOW	ANTIHORARIO
HIGH	HIGH	OFF

TABLE I: Estado del motor en función de la combinación de los pines digitales

- **Señal del potenciómetro de precisión:** Señal de voltaje analógico que determina la posición del péndulo y es utilizada para ejecutar el algoritmo de control.

- **Señales de los encoders del motor:** Señales de reloj que indican el movimiento del motor en sentido horario y antihorario para determinar la posición y estabilizar el brazo.

C. Diseño del controlador difuso

El algoritmo de control se implementa a las señales de ingreso correspondientes al brazo y el péndulo:

- **Diseño del controlador del péndulo:** En el controlador difuso se manejan dos variables: el error de la posición del péndulo respecto a su punto de equilibrio y la derivada del error:

  $Error = \text{Posición} - \text{Set point} \Delta Error = \text{Error actual} - \text{Error anteri}$

  Para la fuzzificación de las variables se usan funciones de membresía. Para el error se han establecido los estados mostrados en la figura 4. En los que se destacan 3 estados que transforman el valor del error de 10 bits (resolución ADC del Arduino UNO) que está en el rango de -512 a +512 con respecto al punto de equilibrio.

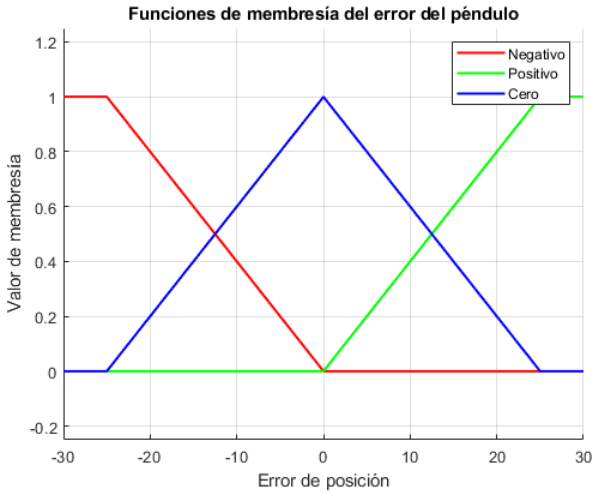

Fig. 4: Funciones de membresía para la fuzzificación del error en el péndulo

Es importante resaltar que lo equivalente a enviar una señal de voltaje negativa (para cambiar el sentido de giro) es invertir el estado de los pines digitales como se describe en la tabla I. Por esta razón se optó por considerar únicamente el valor absoluto del error, esto quiere decir que solo se considera las funciones de membresía de cero y positivo para el cálculo de la salida, el sentido de giro se determinará con revisar si el error es menor o mayor a cero. Esta lógica permite simplificar la implementación a nivel de programación con la eliminación de cálculos innecesarios, por lo que se usa en todas las funciones de membresía.
De igual manera se muestran las funciones de membresía para la diferencia del error en la figura 5 en donde se considera la misma lógica de programación descrita para el error del péndulo.

Fig. 5: Funciones de membresía para la fuzzificación de la diferencia del error en el péndulo

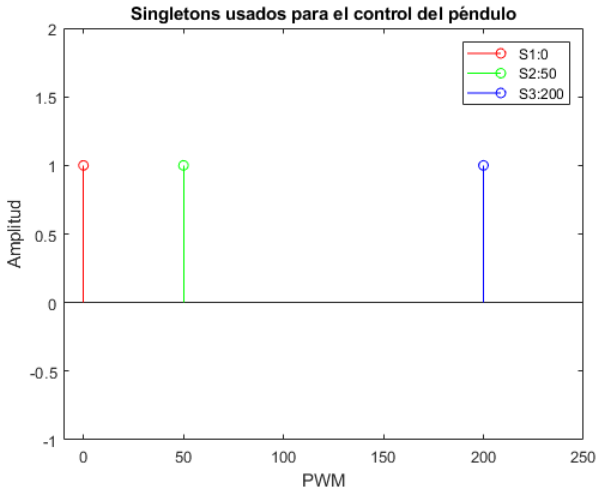

Fig. 6: Singletons usados para el método ponderado en el péndulo

Luego de fuzzificar las variables para el error y su diferencia se aplican las reglas descritas en la siguiente tabla:

		ΔE	
		Pequeño (Cero)	**Grande (Positivo)**
E	**Pequeño (Cero)**	Cero	Pequeño
	Grande (Positivo)	Grande	Grande

TABLE II: Reglas de control difuso para el control del péndulo

Para la aplicación de las reglas se ha usado el método de Mamdami usando los valores mínimos como operación AND determinando así el valor correspondiente a la regla según la tabla II. En cuanto a la defuzzificación se usó el método ponderado dado por:

$$z_{pendulo} = \frac{\sum_{i=1}^{N} w_i c_i}{\sum_{i=1}^{N} w_i} \quad (1)$$

donde w_i son los valores inferidos (consecuentes) de las reglas con el método de Mandami y c_i son los singletons definidos para el control del péndulo dados por la figura 6. En caso de que se active la misma regla en dos ocasiones, tomamos el máximo de estos valores consecuentes que es equivalente a una operación XOR.
El valor $z_{pendulo}$ obtenido de la ecuación 1 es un componente de la señal de control PWM que se enviará al motor del péndulo.

- **Diseño del controlador del brazo** El control del brazo sigue la misma lógica que el controlador del péndulo con el cambio de los valores característicos de sus funciones de membresía y sus singletons que se han ajustado en función del análisis visual del comportamiento del péndulo al comprobar su funcionamiento.
  Las funciones de membresía usadas para la fuzzificación del error de la posición respecto a un valor de referencia establecido en código, esta posición se determina con la ayuda de los encoders del motor y a través de interrupciones que cuentan la cantidad de pulsos mientras el brazo se mueve. Las funciones de membresía para el error se muestran en la figura 7.

Fig. 7: Funciones de membresía para la fuzzificación del error en el brazo

De igual manera se ha considerado la lógica explicada en el control del péndulo para considerar únicamente el valor absoluto del error y controlar el giro con los valores digitales descritos en la tabla I. Las funciones de membresía aplicadas a la diferencia del error del brazo se muestran en la figura 8
Las reglas aplicadas a los valore fuzzificados varían con respecto a las del péndulo como se describe en la tabla III con los singletons de la figura 9. De igual manera se usó el método ponderado para la defuzzificación de la señal resultante (z_{brazo}).

- **Estimación del valor de salida:** Una vez obtenidos los valores $z_{\text{péndulo}}$ y z_{brazo} de los algoritmos de control

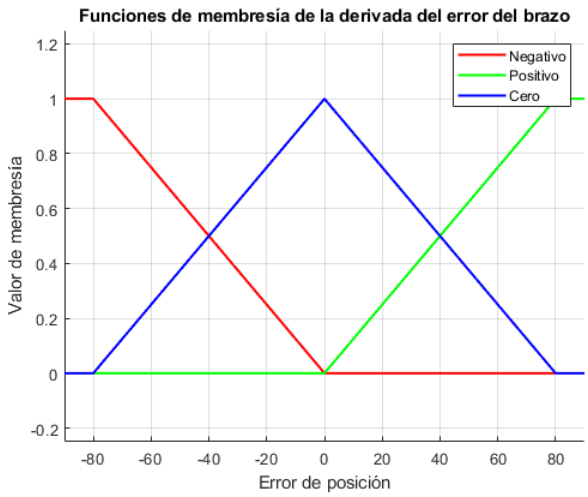

Fig. 8: Funciones de membresía para la fuzzificación de la diferencia del error en el brazo

		ΔE	
		Pequeño (Cero)	**Grande (Positivo)**
E	**Pequeño (Cero)**	Cero	Pequeño
	Grande (Positivo)	Pequeño	Grande

TABLE III: Reglas de control difuso para el control del brazo

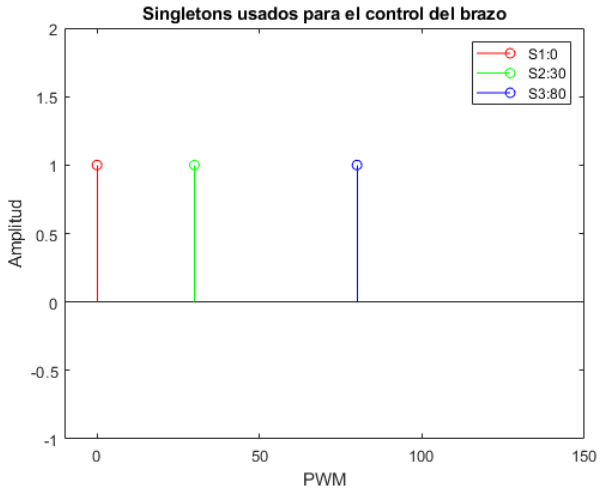

Fig. 9: Singletons usados para el método ponderado en el brazo

difusos implementados, se debe construir la señal que se enviará al driver del motor. Como se detalla en [2], consideramos que la salida generada por $z_{\text{péndulo}}$ desestabilizará la posición del brazo respecto a la referencia debido al movimiento necesario para equilibrar el péndulo. Por esta razón, se compensa la salida restando z_{brazo} de $z_{\text{péndulo}}$ para contrarrestar el movimiento generado al equilibrar el péndulo.

$$PWM = z_{\text{péndulo}} - z_{\text{brazo}}$$

III. Análisis de resultados

El algoritmo implementado ha funcionado de manera adecuada, logrando equilibrar el péndulo en su posición vertical durante periodos considerables de tiempo. En la figura 11, se muestra una imagen de la implementación física, donde se observa la efectividad del control difuso para este tipo de sistemas.

Una vez implementado el algoritmo de control y demostrado su efectividad, se han tomado medidas de las variables analógicas del sistema con la tarjeta de adquisición de datos NI myDAQ con una frecuencia de muestreo de 10 ms con el fin de analizar el valor generado por el algoritmo de control difuso respecto a una variable de error a su entrada. En la figura 10 se muestran las señales de valor absoluto del error en la posición péndulo y la respuesta generada por el control difuso en tiempo real en PWM.

La razón por la que se observa una señal PWM en lugar de una señal analógica en la salida se debe a la forma en que Arduino envía una señal análógica en sus pines. En realidad, Arduino envía una señal PWM con un dutty cycle específico en función del valor análogico que se requiere, el motor se moverá según el valor promedio de esta señal por lo que el dutty cycle del PWM es directamente proporcional al valor de voltaje analógico que se establece en el código.

Refiriéndonos a la Figura 10, se observa que existe una señal PWM cuando el error excede un umbral. Esto se debe a que hemos detectado que el valor de voltaje en el potenciómetro no es fijo, sino que existe un rango de valores en los que el péndulo se mantiene en equilibrio.

Además, se ha establecido una medida de seguridad en el código con un valor umbral para el cálculo de la señal de control. Con un error muy grande, el ciclo de trabajo de la PWM sería elevado, lo que generaría un giro muy fuerte del motor, desestabilizando el sistema y provocando daños en los componentes. En la Figura 10, se observa que cuando el error (curva de color naranja) aumenta, se genera una señal PWM en cierto sentido de giro que reduce el error, manteniendo el péndulo relativamente en equilibrio hasta que nuevamente se eleva y el algoritmo de control actúa para reducirlo.

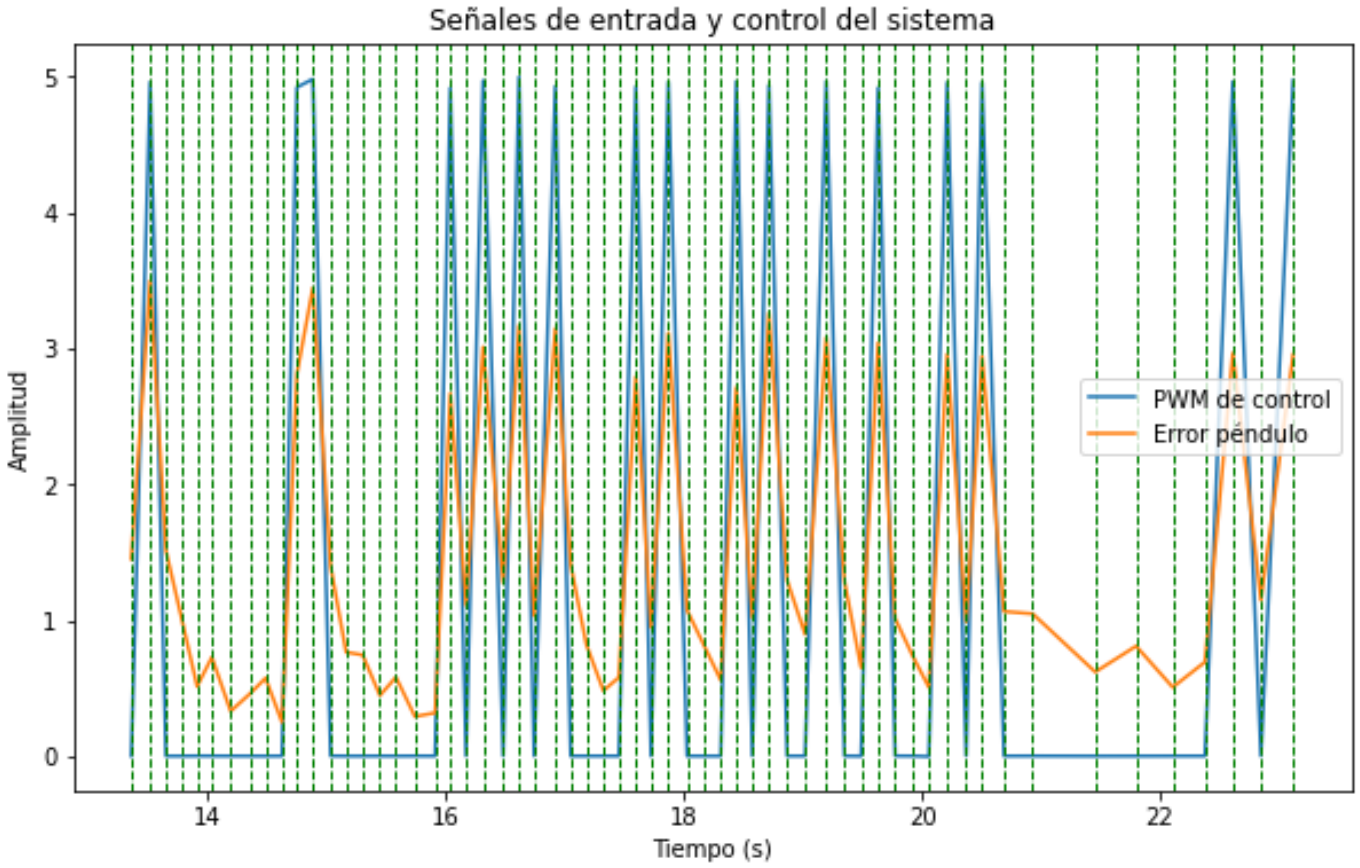

Fig. 10: Variables de entrada y salida del sistema en tiempo real obtenidos del NI myDAQ

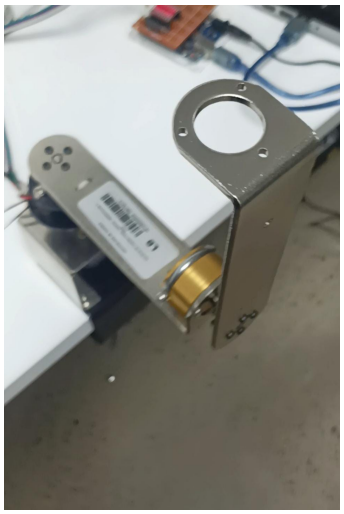

Fig. 11: Péndulo equilibrado con el algoritmo de control implementado

IV. Conclusiones

En este proyecto se ha implementado un algoritmo en Arduino para equilibrar un péndulo de Furuta utilizando lógica difusa para el control de la posición del brazo y del péndulo. El código completo esta en https://github.com/ErickDum/FuzzyControl_FurutaPendulum.

La señal PWM se genera con un sentido de giro determinado en función del error de la posición (tanto del péndulo como del brazo) y la diferencia con respecto al error anterior. Estas variables son fuzzificadas mediante funciones de membresía que cumplen ciertas reglas utilizando el método de mínimos de Mamdani. Posteriormente, se determina la salida defuzzificada mediante el método ponderado, estableciendo valores de singletons obtenidos a través de numerosas pruebas en el péndulo y analizando los valores que ofrecen el mejor control para el equilibrio.

El control difuso es una técnica de control muy robusta debido a la ponderación de la variable de entrada en cada estado posible, garantizando que el efecto de cada función de membresía en la salida del sistema se manifieste en cierto grado. Esto permite generar una señal de salida que corresponda mejor al valor de ingreso, sin establecer valores fijos a la salida que podrían desestabilizar el equilibrio del péndulo.

References

[1] "TB6612FNG Driver para motor DC &x2013; Novatronic — novatronicec.com," https://novatronicec.com/index.php/product/tb6612fng-driver-para-motor-dc/, [Accessed 17-07-2024].

[2] H. Guo, J. Wu, Z. Yin, and W. Zhang, "Comparative study of furuta pendulum based on lqr and pid control," *Journal of Physics: Conference Series*, vol. 2562, no. 1, p. 012075, aug 2023. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2562/1/012075

[3] I. Minchala Avila and R. Marbán Romero, "MR5036-NLAC Final Project 1: Nonlinear Control of the Furuta Pendulum," 2015, student Member, IEEE, and MAT09 Student, ITESM.

[4] H. Guo, J. Wu, Z. Yin, and W. Zhang, "Comparative Study of Furuta Pendulum Based on LQR and PID Control," *Journal of Physics: Conference Series*, vol. 2562, p. 012075, 2023, corresponding author's e-mail: junwu@csust.edu.cn.

[5] D. Driankov, H. Hellendoorn, and M. Reinfrank, *An Introduction to Fuzzy Control*. Springer Berlin Heidelberg, 1993. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-11131-4

V. Biografía de los autores

Jorge Geovanny Zhangallimbay Coraizaca

Nació el 14 de octubre del 2002 en Azogues, Ecuador. Cursó sus estudios secundarios en la Unidad Educativa Luis Rogerio González en la especialización de electrónica. Aficionado al área de las tecnologías y la electrónica, optó por la carrera de Ingeniería en Telecomunicaciones en la Universidad de Cuenca.

Erick Santiago Dumas Puma

Tras culminar sus estudios en la unidad educativa Hermano Miguel de la Salle, denotaría su gran interés por el arte tecnológico y virtual. Delimitaría su interés total por el área de Telecomunicaciones y programación, puesto que este es un medio de globalización en la era tecnológica actual. A día de hoy cursa sus estudios en la Universidad de Cuenca en la provincia del Azuay; su meta es ser un Ingeniero en telecomunicaciones para cumplir su sueño de desenvolverse en un entorno físico y tecnológico.