TECNOLÓGICO DE MONTERREY

CAMPUS MONTERREY

Automatización Industrial (Gpo 605)

REPORTE FINAL

Roberto García Martínez	| A01721760
Jonatan David De la Rosa Patlán	| A01735993
Samuel Alejandro Meléndez Guerrero	| A01198289
Javier Alejandro Santillán Cuéllar	| A00833467

3 de diciembre de 2023

Profesores

Valery Moreno

Ricardo Roberts

REPORTE FINAL

I. Introducción

El objetivo de este proyecto es el diseño, ensamble y prueba de un vehículo autónomo de cuatro ruedas. Utilizando un sistema de control PID, el vehículo responde a señales del usuario y del entorno, siguiendo una trayectoria preprogramada en las direcciones cardinales. La libertad de elección abarca la selección de sistemas sensoriales y comunicación, destacando la estética sin cables sueltos. Además, la transferencia de datos, como telemetría, añade conectividad. Alimentado por baterías y garantizando el retorno al punto de inicio, cada equipo se propone una creación de un vehículo autónomo que fusiona innovación y tecnología intentando minimizar el error lo máximo posible. Finalmente, este vehículo deberá completar un recorrido, el cual está conformado por un cuadrado con conos en cada vértice.

II. Materiales y métodos

Debido a que el reto consiste más en la implementación de un carrito autónomo, optamos por un mayor enfoque en la electrónica y el diseño de control, es por ello que decidimos utilizar una parte mecánica prefabricada.

A. Materiales y fabricación

El kit por el que se obtuvo para la fabricación de este carro es un *"Chasis de carro Kit Arduino"* como el que se puede ver en la **figura 1,** los elementos que incluye este kit son:

- Chasis de carro en acrílico transparente (x2)
- Motor de engranajes 1:48 (x4)
- Ruedas encoders de velocidad (x4)
- Llantas (x4)
- Caja para 4 baterías AAA (Las baterías no están incluidas)
- Kit Tornillos, tuercas y separadores metálicos

Figura 1. Chasis de acrílico

En la **Tabla 1.** se presenta el BOM utilizado para este proyecto.

Material	Número	Costo	Función	Link de compra
Chasis de carro Kit Arduino	1	$194	Sera nuestra parte mecánica encargada de soportar toda la electrónica	Link
Bateria Recargable 18650	4	$128	Será encargado de la alimentación para el carro	Link
Arduino Nano V3.0 + Cable Usb	1	$118	El microcontrolador a usar encargado de todo el control del carro	Link
NodeM	1	$159	Utilizad	Link

cu (ESP8266)			a para transmitir datos por puerto serial a la nube IOT	
Encoder Óptico	2	$58	Sensor para medir RPMs del carro así como distancia	Link
MPU 6050	1	$95	Sensor de 6DOF que nos ayudará a medir los cambios de ángulo en la dirección	Link
Modulo Puente H L298	1	$70	Driver para controlar Motores	Link

Tabla 1. Lista de materiales (BOM)

B. Diagrama de conexión

En las **Figuras 2 -5,** se muestra el diagrama de conexiones que se utilizó para la implementación de una PCB **Figura 6.** En este último podemos observar las conexiones de los actuadores y sensores al arduino Nano Atmega328P, así como la comunicación en serie con el NodeMcu.

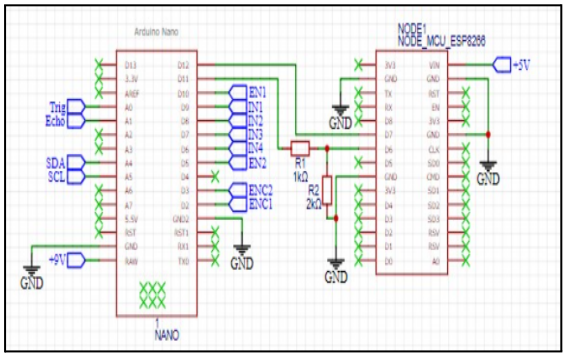

Figura 2. Diagrama de conexiones Arduino-Node.

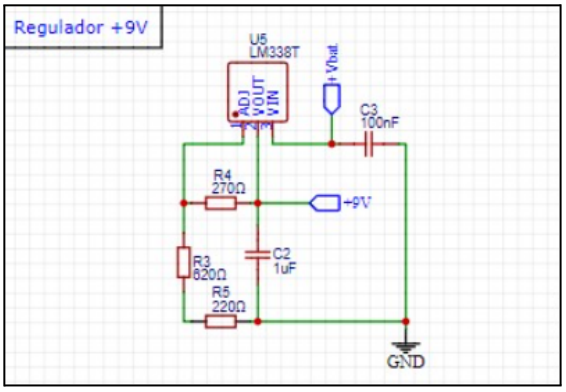

Figura 3. Diagrama de conexiones Regulador +9V.

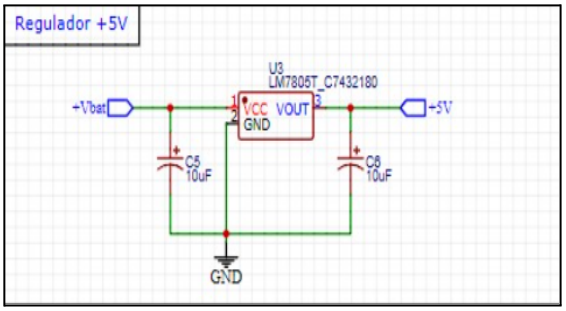

Figura 4. Diagrama de conexiones Regulador +5V.

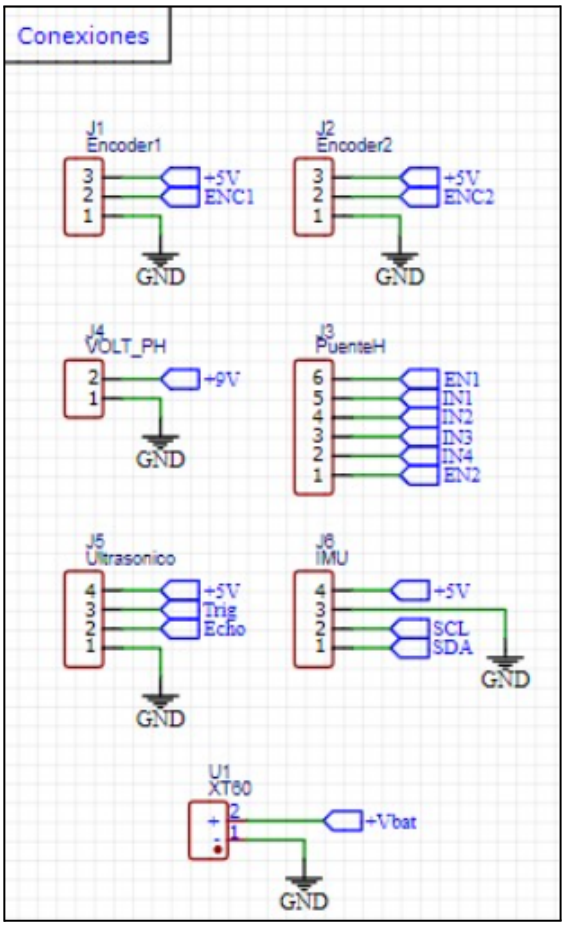

Figura 5. Diagrama de conexiones de sensores.

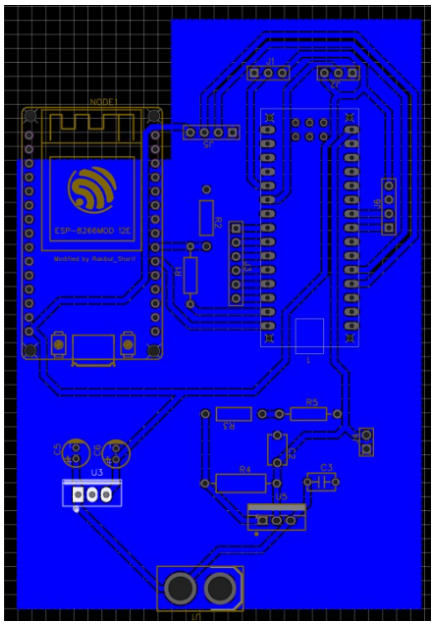

Figura 6. Esquema de PCB

También podemos observar en la **Figura 3 y 4,** dos reguladores modelo LM338T los cuales regulan a 9V, que será la alimentación tanto para el puente H como para el arduino. Además, otro regulador a 5V para la alimentación de los sensores.

C. Tabla de entradas y salidas

Los pines son los siguientes mostrados en la Tabla 2 . Son en en base al arduino Nano Atmega328P

Actuadores/Sensores	**E/S**	**Pin**	**Variable**
Encoder 1	E. Digital	D3	EncoderR
Encoder 2	E. Digital	D2	EncoderL
ENA	S. PWM	D10	Objeto Motor
ENB	S.PWM	D5	Objeto Motor
EN1	S. Digital	D8	Objeto Motor
EN2	S. Digital	D9	Objeto Motor
EN3	S. Digital	D7	Objeto Motor
EN4	S. Digital	D6	Objeto Motor
MPU6050	I2C	A4/A5	Objeto Giros

Tabla 2. Entradas y salidas de Arduino NANO

D. Arquitectura de control

La arquitectura de control utilizada fue la AVR RISC donde se utilizó el microcontrolador ATMega328p de un Arduino Nano, este fue seleccionado ya que esta arquitectura se caracteriza por su eficiencia en rendimiento, bajo consumo de energía y una facilidad de programación siendo el ATMega328P un microcontrolador de 8 bits basado en esta arquitectura AVR de Atmel.

<u>Características del Microcontrolador:</u>

- Velocidad de reloj de 16Mhz
- Memoria Flash 32 Kb
- Memoria Ram 2 Kb
- 23 pines E/S
- Comunicación UART, I2C y SPI
- ADC de 10 bits
- Oscilador interno de 8 bits

E. Sistema de control

El sistema de control por el que se optó fue un control PI en el IMU para poder decretar un ángulo y por medio de cambios en la velocidad en uno de los motores, controlar los cambios de ángulo que podría tener a la hora de avanzar. El motivo por el cual se realizó el control aquí fue para poder asegurarnos de que el carro fuera recto durante el tramo de 10 metros que se tiene planeado que recorrer.

Para obtener nuestra ecuación de transferencia fue necesario primero sacar los datos en Lazo abierto (**Figura 7).** Para el cual pusimos los dos lados del carro a una misma velocidad de 127 PWM, luego metimos un escalón a uno de los lados poniéndolo a su máxima velocidad de 255 PWM para crear un cambio en el ángulo. Después de un tiempo, se establece de nuevo a la misma velocidad para lograr una estabilidad en el ángulo. Sin embargo, como se muestra en la **Figura 7**, cuando se volvió a poner los dos lados al mismo PWM, no se asegura que vayan a la misma velocidad, por lo que hubo un declive en el ángulo a pesar de recibir el mismo PWM.

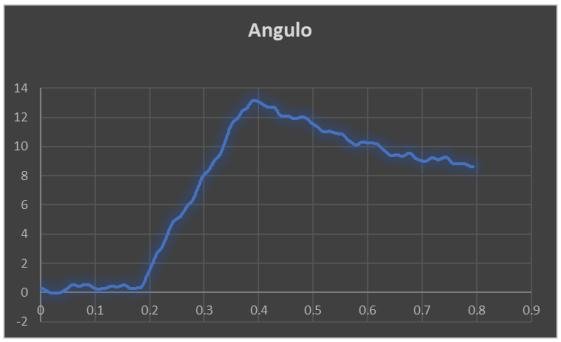

Figura 7. Gráfica de ángulo vs tiempo.

Con estos datos recopilados pudimos sacar nuestra ecuación de transferencia utilizando el método analítico que se muestra en la **Figura 8.** Teniendo que :

ΔY=13.15
ΔU=128
K=ΔY/ΔU=0.1027
0.284*ΔY=3.73
(3.7)→(0.23)
(3.73)→(x=0.232)

0.632*ΔY=8.31
(8.29)→(0.306)
(8.31)→(x=0.306)

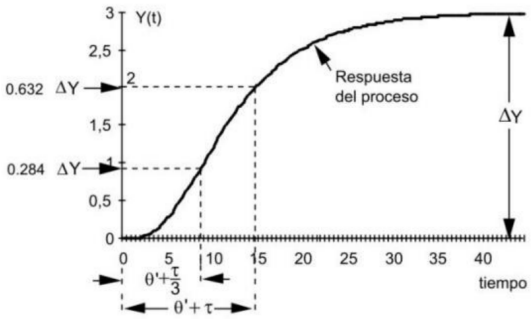

Figura 8. Método analítico de respuesta a la frecuencia.

Con ello podemos hacer nuestro sistema de ecuaciones y conseguir nuestro Tiempo muerto (theta) y Tao.

$$\theta + \frac{t}{3} = 0.232$$

$$\theta + \frac{t}{3} = 0.3068$$

$$\theta = 0.1946$$

$$t = 0.1122$$

El tipo de sistema de control a utilizar es un regulatorio, este sistema como se muestra en la figura 9. Está diseñado para compensar perturbaciones que afectan la variable de proceso por lo cual es ideal para nuestro propósito, Utilizaremos las constantes de tunning a, b, c y d del sistema de control regulatorio ITAE que se muestran en la **figura 9.** Y con las fórmulas de la **figura 11**.

	Control Regulatorio			Servocontrol		
	ISE	IAE	ITAE	ISE	IAE	ITAE
a	1.495	1.435	1.357	NA	1.086	0.965
b	-0.945	-0.921	-0.947	NA	-0.869	-0.855
c	1.101	0.878	0.842	NA	0.740	0.796
d	-0.771	-0.749	-0.738	NA	-0.130	-0.147
e	0.560	0.482	0.381	NA	0.348	0.308
f	1.006	1.137	0.995	NA	0.914	0.9292

Figura 9. Índices de desempeño para control regulatorio.

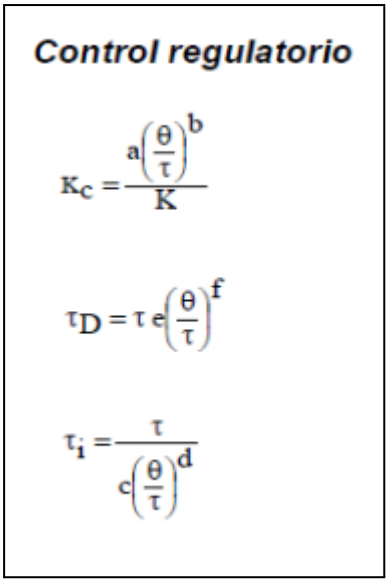

Figura 10. Fórmulas de control regulatorio.

Sacaremos nuestras constantes K_c y T_i siendo estas:

$K_p = K_c = 10.4812$
$K_i = 1/T_i = 5.3326$

F. Programación

El desarrollo de la programación se realizó en lenguaje C++ por medio de Visual Studio Code en PlataformIO, esto nos permite crear un ambiente de desarrollo para programar arduino teniendo que importar arduino como #INCLUDE <Arduino.h>. Del mismo modo se realizó una clase para los motores que se puede encontrar en el repositorio de Github en la carpeta lib así como una clase para el Giroscopio donde se involucró un filtro kalman y un filtro condicional para eliminar el error que suele dar el giroscopio en el ángulo yaw, estas clases se tuvieron que importar al código principal "/src/mainArduino.cpp" así como una librería aparte de arduino PID_V1 que nos facilita la discretización del PID para meterlo a código, su documentación se puede encontrar aquí (br3ttb/Arduino-PID-Library (github.com))

También, por parte del NodeMcu se desarrolló la comunicación con el arduino para que al mandar por el serial del arduino el NodeMcu pueda recibir los datos y mandarlos al ThingSpeak, lamentablemente los datos solo se pueden mandar cada 15 segundos por tema de que estamos usando la versión gratuita. Esto nos obliga a esperar mucho para recibir datos significativos.

Código principal:

```
#include <Arduino.h>
#include <Giroscopio.h>
#include <motores.h>
#include <PID_v1.h>

float angle;
float angleSet;
int pwm = 127;
int count = 0;

const int trayecto = 11; // distancia en
metros a recorrer

const int encoderPin = 3;
volatile long pulseCount = 0;
volatile long frecuenciaEncoder = 0;
const int pulsePerRevolution = 56;
float distancia = 0;

unsigned long interval = 15000;
unsigned long previousMillis = 0;

motores motor(10, 8, 9, 5, 7, 6);
Giroscopio giros(1);

double Setpoint = 0;
double Input, Output;

PID myPID(&Input, &Output, &Setpoint,
10.4812, 5.3326, 0.00, DIRECT);

void countPulse()
{
    pulseCount++;
}

void setup()
{
    Serial.begin(115200);
    motor.setup();
    motor.setSpeedLeft(120);
    motor.setSpeedRight(120);
    giros.setup();
    myPID.SetMode(AUTOMATIC);
    myPID.SetOutputLimits(100, 200);


attachInterrupt(digitalPinToInterrupt(encode
rPin), countPulse, RISING);
    delay(50);
}

void loop()
{
    //   giros.getAngle("Yaw");

    if (pulseCount >= pulsePerRevolution)
    {
        distancia += 20.42;
        pulseCount = 0;
    }
    if (distancia <= trayecto * 100)
    // if (true)
    {
        motor.avanzar();
        angle = giros.getAngle("Yaw");
        Input = angle;
        myPID.Compute();
        motor.setSpeedRight(Output);
        // Serial.print(">angle: ");
        // Serial.println(angle);
        // Serial.print(">output: ");
        // Serial.println(Output);
        unsigned long currentMillis =
millis();
        if (currentMillis - previousMillis
>= interval)
        {
            Serial.print(angle);
            Serial.print(',');
            Serial.print(pwm);
            Serial.print(',');
            Serial.print(distancia);
            Serial.print(',');
            Serial.println(1);

            previousMillis = currentMillis;
        }
    }
    else
    {
        pulseCount = 0;
        motor.stop();
        motor.setSpeedRight(255);
        motor.setSpeedLeft(255);
        delay(1000);
        angle = giros.getAngle("Yaw");
        angleSet = angle + 55;
        while (angle <= angleSet)
        {
            motor.girarIzq();
            angle = giros.getAngle("Yaw");
        }
        motor.stop();
        delay(1000);
        distancia = 0;
        motor.setSpeedLeft(120);
        motor.setSpeedRight(120);
        angle = giros.getAngle("Yaw");
        Setpoint = angle;
        Input = angle;
    }
}
```

Código NodeMcu:

```
#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>
#include "secrets.h"
#include "ThingSpeak.h"

char ssid[] = "OnePlus 9 5G";   // your
network SSID (name)
```

```
char pass[] = "q3et8thi";       // your
network password
int keyIndex = 0;                // your
network key Index number (needed only for
WEP)
WiFiClient client;

unsigned long myChannelNumber = 2356326;
const char *myWriteAPIKey =
"72AMB84GOEX63M8P";

// Initialize our values
float number1, number2, number3, number4;

// Define SoftwareSerial pins on NodeMCU
const byte rxPin = 12; // Change to the
appropriate pin
const byte txPin = 13; // Change to the
appropriate pin

SoftwareSerial mySerial(rxPin, txPin,
false);

void setup() {
  Serial.begin(115200);  // Initialize
serial
  while (!Serial) {
    ; // wait for serial port to connect.
Needed for Leonardo native USB port only
  }

  mySerial.begin(115200); // Initialize
SoftwareSerial

  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);  // Initialize
ThingSpeak
}

void loop() {

  // Connect or reconnect to WiFi
  if (WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to
SSID: ");
    Serial.println(SECRET_SSID);
    while (WiFi.status() != WL_CONNECTED) {
      WiFi.begin(ssid, pass);  // Connect to
WPA/WPA2 network. Change this line if using
open or WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }

  // Read data from Arduino Nano via
SoftwareSerial
  if (mySerial.available() > 0) {
    // Read data until a newline character
is found
    String data =
mySerial.readStringUntil('\n');

    // Convert the received data to the
appropriate variables
    char charArray[data.length() + 1];
    data.toCharArray(charArray,
sizeof(charArray));

    // Use strtok to tokenize the string
    char *token = strtok(charArray, ",");
    if (token != NULL) {
      number1 = atof(token);
      token = strtok(NULL, ",");
    }
    if (token != NULL) {
      number2 = atof(token);
      token = strtok(NULL, ",");
    }
    if (token != NULL) {
      number3 = atof(token);
      token = strtok(NULL, ",");
    }
    if (token != NULL) {
      number4 = atof(token);
      token = strtok(NULL, ",");
    }
  }

  // set the fields with the values
  ThingSpeak.setField(1, number1);
  ThingSpeak.setField(2, number2);
  ThingSpeak.setField(3, number3);
  ThingSpeak.setField(4, number4);

  // write to the ThingSpeak channel
  int x =
ThingSpeak.writeFields(myChannelNumber,
myWriteAPIKey);
  if (x == 200) {
    Serial.println("Channel update
successful.");
  } else {
    Serial.println("Problem updating
channel. HTTP error code " + String(x));
  }

  delay(20000); // Wait 20 seconds to update
the channel again
}
```

El repositorio del código lo pueden encontrar aqui (Jonatandlr/JohnDeereAutomatization (github.com))

III. Resultados

a. Carrito

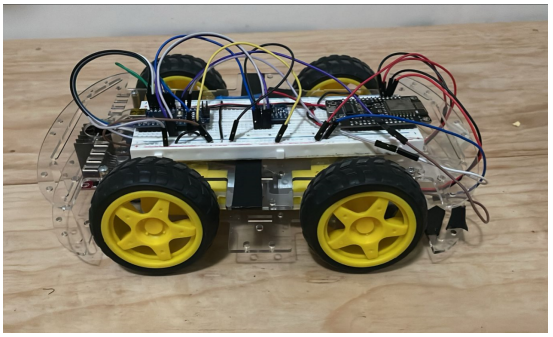

Figura 11. Estructura del carrito completo.

En la **Figura 11,** podemos observar la totalidad del carrito armado y construido de manera final.

b. Sistema de control

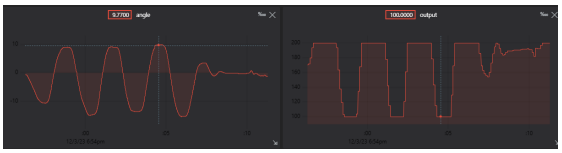

Figura 12. Ángulo positivo y respuesta PWM mínimo.

En esta gráfica (**Figura 12**) podemos apreciar los datos de una de las iteraciones realizadas por los alumnos en donde el ángulo se muestra como 9.7700 grados siendo 0 grados el valor óptimo. Esto, en otras palabras, quiere decir que el carro está rotado de manera horaria fuera de la línea recta ideal. Es por esto que podemos observar el valor del PWM como 100, siendo este el mínimo valor posible intentado minimizar este ángulo y llevarlo a 0.

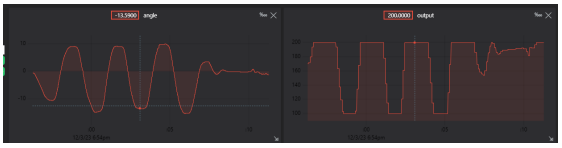

Figura 13. Ángulo negativo y respuesta PWM máximo.

Algo similar ocurre con la **Figura 13.** Sin embargo, este es el caso contrario. El ángulo se muestra como -13.5900 grados significando que el carro está rotado de manera antihoraria. En este caso el PWM se encuentra en 200 siendo el máximo intentado corregir este ángulo.

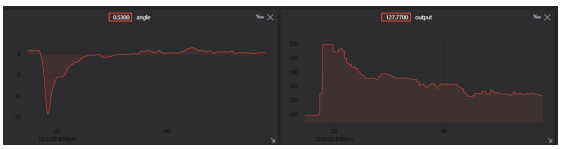

Figura 14. Sistema de control ángulo-PWM.

En la **Figura 14,** podemos observar que cuando se estabiliza el ángulo (gráfica izquierda) se estabiliza el PWM y se mantiene como un output de manera constante y controlada lo cual ayuda a que el carro se mueva de manera rectilínea.

c. ThingSpeak

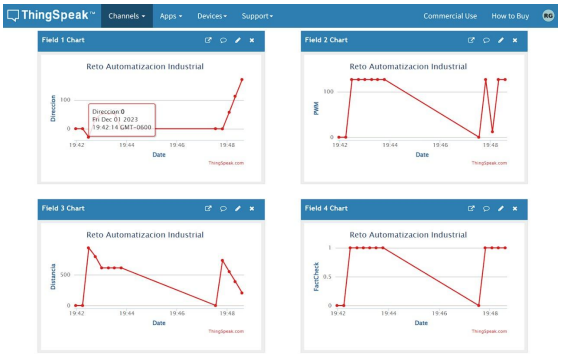

Figura 15. Comunicación robot-ThingSpeak

En este apartado se puede observar el regreso de datos que nos proveé la plataforma ThingSpeak. Las gráficas que se muestran en esta interfaz son las siguientes:

- Dirección vs Tiempo
- PWM vs Tiempo
- Distancia vs Tiempo
- FactCheck vs Tiempo

Lamentablemente, la plataforma solo nos permite mandar datos cada 15 segundos, por lo que, no consideramos que sea una buena opción para mostrar el potencial inmediato de nuestro carro. Sin embargo, la comunicación funciona y es una gran adición práctica a la totalidad del proyecto.

Anexos

- Drive del carrito

IV. Conclusiones

Finalmente, el proyecto de nuestro Vehículo Autónomo ha culminado con un notable éxito, demostrando la capacidad del equipo para enfrentar desafíos y superar expectativas. A pesar de un margen de error de un metro, que se considera excepcionalmente bajo, el vehículo ha navegado de manera autónoma por la trayectoria predefinida con precisión admirable, incluso corrigiendo desvíos de 180°. Este resultado habla no solo de la eficacia del sistema de control PID implementado, sino también de la calidad en la elección de los sistemas sensoriales y de comunicación. La combinación de la autonomía fluida, la presentación estética sin cables visibles y la capacidad de transferir datos a sistemas externos destaca la excelencia lograda en este proyecto multidisciplinario. Los aprendizajes obtenidos refuerzan con creces las aptitudes del equipo para futuras innovaciones en la exploración robótica.