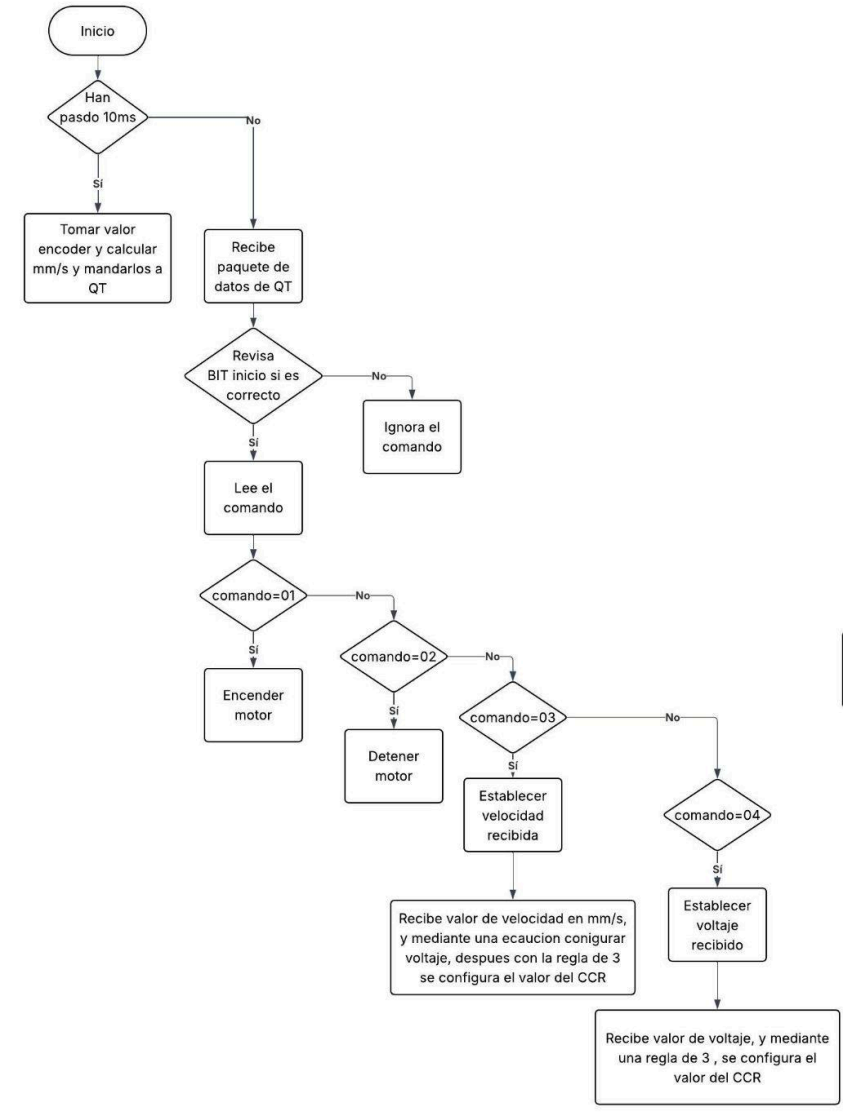
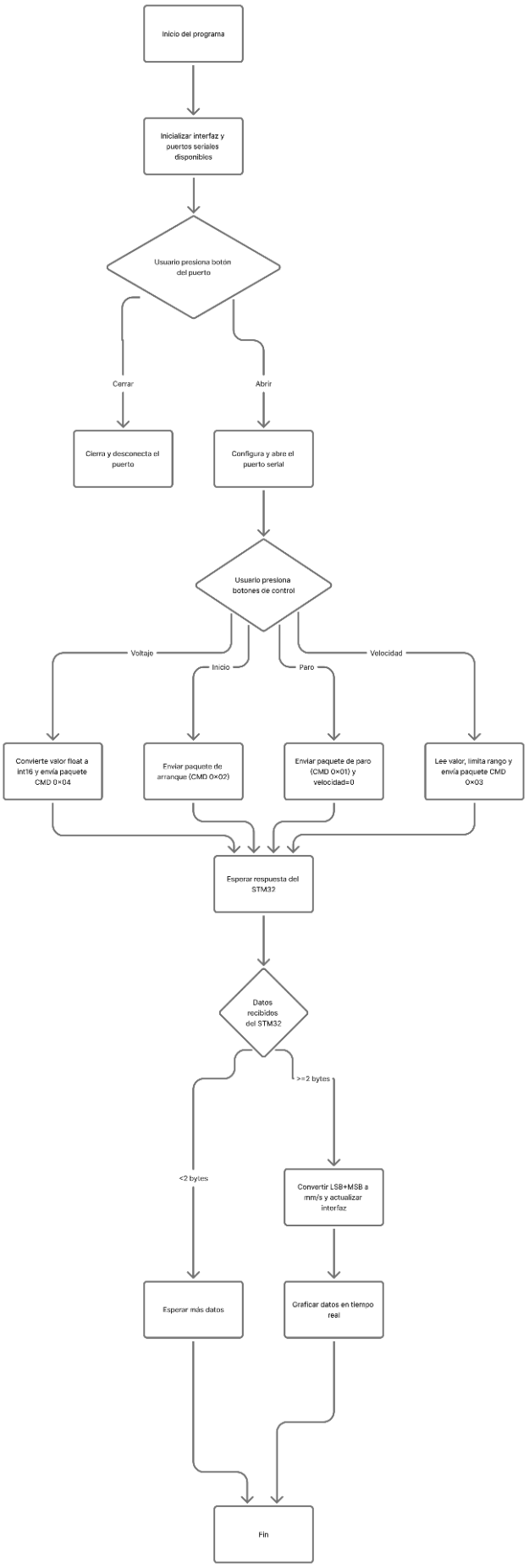
Informe de laboratorio #5
Juan Manrique Jose Jimenez
Universidad Sergio Arboleda

1. **Elementos utilizados**
   - Microcontrolador STM32
   - Cable USB
   - Motor DC con ENCODER de cuadratura.
   - Etapa de potencia, puente H, transistores etc. Según diseño.
   - Software excel para gráficas y ecuaciones
   - Programa QT
   -
2. **Diagrama de flujo**




Inicio del programa
Inicializar interfaz y puertos seriales disponibles
Usuario presiona botón del puerto
Cerrar
Abrir
Cierra y desconecta el puerto
Configura y abre el puerto serial
Usuario presiona botones de control
Voltaje
Inicio
Paro
Velocidad
Convierte valor float a int16 y envía paquete CMD 0×04
Enviar paquete de arranque (CMD 0×02)
Enviar paquete de paro (CMD 0×01) y velocidad=0
Lee valor, limita rango y envía paquete CMD 0×03
Esperar respuesta del STM32
Datos recibidos del STM32
<2 bytes
>=2 bytes
Esperar más datos
Convertir LSB+MSB a mm/s y actualizar interfaz
Graficar datos en tiempo real
Fin

3. Procedimiento

Para empezar, le dimos una vuelta completa al motor con el montaje de la rueda con las canicas, esto para obtener el número de pulsos de encoder que toma una vuelta, este procesos lo hicimos una 10 veces para obtener un promedio de 1435 pulsos.

Luego de eso se configuró un TIM en interrupción, esto para poder garantizar que entre a esa parte del código cada 10ms exactamente, y asi poder hacer los cálculos de velocidad, en esta parte del codigo solo obtenemos cuantos pulsos han pasado y calculamos un delta de pulsos.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
encodermotor1=TIM2->CNT;
delta=encodermotor1-encodpas;
encodpas=encodermotor1;
}
```

Ya teniendo el dato lo pasamos por las siguientes ecuaciones para extraer todos los datos que necesitamos que en este caso es velocidad angular en radianes y pasarlo a milímetros por segundo.

```
rads=(((float)delta*2.0*M_PI)/(ppv*0.010));
mms=rads*radio*1000.0;
```

Luego de obtener los datos en el ST pasamos al QT que funciona como interfaz gráfica, en este pusimos la selección del puerto COM, y una función que recibe los mms que mandaba ST y los grafica en tiempo real pero para facilitar el análisis de los datos también los pasamos a un archivo .log para meterlos a excel y poder graficar mejor.

En QT también pusimos dos cajas para que el usuario pueda colocar el voltaje al que quiere que el motor ande y para la velocidad a la que quiere andar pero por el momento nos centramos en la de voltaje, en este caso en QT hicimos un protocolo de comunicaciones que constaba de lo siguiente:

- Byte de inicio: 0xaa, para indicar y verificar que empezamos a mandar un mensaje.
- Byte de comando: podía ser 0x01 (indica que el motor arranca), 0x02(detiene el motor), 0x03 (se está mandando una velocidad), 0x04 (se está mandando un voltaje).
- Byte de tamaño:indica el tamaño de la data que se está mandando.

- Bytes de información: en estos bytes va la información cuando se manda un voltaje o una velocidad, en los otros dos casos no manda nada.
- Byte de check sum: es una xor para confirmar que el mensaje llegó completo.

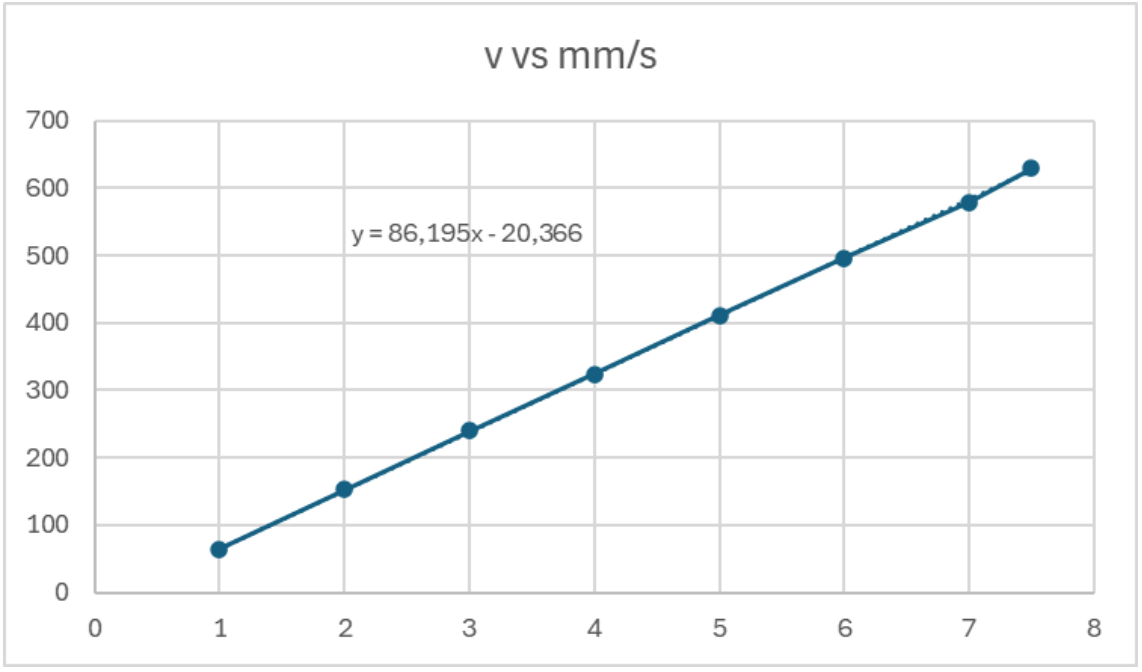
Luego de tener este protocolo de comunicaciones en ST hicimos una ecuacion para que cuando el usuario mande un voltaje, este se configure en el motor, ajustando el CCR.

```
int16_t raw_volt = (int16_t)(payload[0] | (payload[1] << 8));
volta = raw_volt / 1000.0f;
  float valor_loc = ((volta * 3200.0) / 7.5);
  int valor_ent = (int)valor_loc;
  TIM3->CCR1=valor_ent;
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_13,1);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_14,0);
```

Esto lo hicimos directamente en la parte donde procesamos los comandos que se reciben para solo hacerlo una vez y liberar más rápido al procesador, en este caso la función recibe el comando lo rearma y después lo divide entre 1000, ya que antes de pasarlo lo multiplicamos por el mismo número ya que el valor de voltaje podría ser un float o llevar puntos y era más fácil mandar un entero.

Luego lo pasamos a una regla de 3 donde sabemos que 7,5v son 3200 de valor de CCR, y ya que es necesario que el CCR sea un valor exacto lo casteamos y ya después lo asignamos al motor.

Ya teniendo esto simplemente desde QT mandamos los valores de voltaje y obtenemos la velocidad, para poder graficarla, pasamos los datos a excel y tomamos el promedio de datos que ya son mas estables, para poder hacer la grafica de voltaje vs mm/s y sacamos la linea de tendencia y su respectiva ecuación.

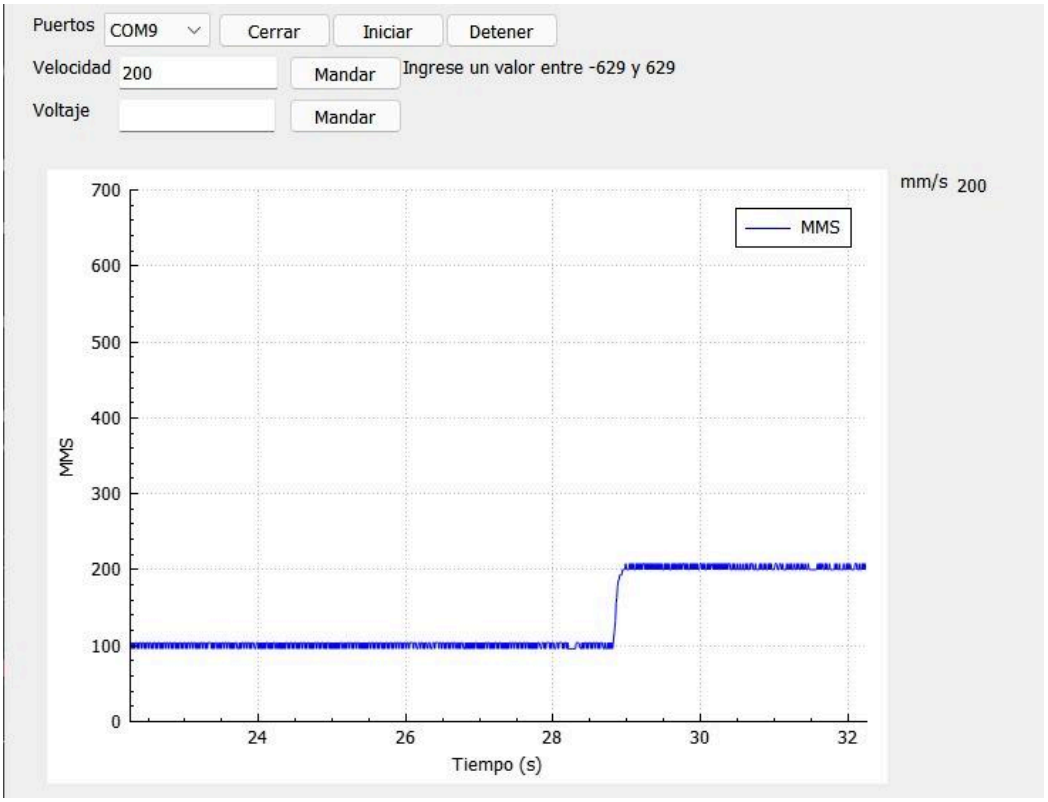


Los datos ya estaban bastante linealizados entonces coincide bastante con la linea de tendencia, de la ecuación despejamos x, y pasamos esa ecuación a ST, ya que desde QT vamos a recibir la velocidad, la transformamos en voltaje con esa ecuación y después la pasamos a CCR con la ecuación anterior.

```
        vel = (int16_t)(payload[0] | (payload[1] << 8));
        if(vel >=0){
          calculo = ((vel+20.366)/86.195);
          float valor_loc = ((calculo * 3200.0) / 7.5);
int valor_ent = (int)valor_loc;
TIM3->CCR1=valor_ent;
HAL_GPIO_WritePin(GPIOB,GPIO_PIN_13,1);
HAL_GPIO_WritePin(GPIOB,GPIO_PIN_14,0);
          }else{
            float yneg=vel*-1;
            calculo = ((yneg+20.366)/86.195);
  float valor_loc = ((calculo * 3200.0) / 7.5);
  int valor_ent = (int)valor_loc;
  TIM3->CCR1=valor_ent;
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_13,0);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_14,1);
          }
```

Aquí lo que se hizo fue recibir los datos y transformarlos a lo que necesitábamos y revisar que si la velocidad es negativa solo pasarla a positiva y hacer que el motor gire en la dirección contraria.

A continuación una imagen de como quedo la interfaz de QT.



4. Conclusiones

-Se comprobó una relación lineal entre el voltaje aplicado y la velocidad del motor, lo que permitió obtener una ecuación útil para su control.

-La STM32 ejecutó correctamente el control de velocidad mediante las funciones creadas, logrando un movimiento estable.

-El protocolo de comunicación entre Qt y la STM32 funcionó de forma confiable, permitiendo enviar y recibir datos sin errores.

-La interfaz en Qt facilitó la visualización, el registro y el control del sistema en tiempo real.