



Universidad Nacional de la Patagonia San Juan Bosco

FACULTAD DE INGENIERÍA

AUTOMATIZACIÓN DE TÚNEL DE VIENTO

*Proyecto Final de Carrera
- Ingeniería Electrónica -*

Autores:

Caamiña Quineros, Daniela Beatriz
Yapura, Cristian Alejandro

Diciembre 2021

Agradecimientos

Agradecemos a Gerardo, quien nos convocó para la realización de este proyecto y al Laboratorio de Mecánica de Fluidos que nos abrió las puertas para poder realizarlo.

Queremos dedicar este proyecto a cada una de las personas que nos acompañaron de alguna forma durante nuestro paso por la universidad. Gracias!

...la foto del recuerdo



Autores al realizar el proyecto final de carrera en contexto de "Pandemia" por *Covid-19*.

Índice

1. Introducción	8
2. Objetivo	9
3. Túnel Aerodinámico	10
3.1. Clasificación	10
3.2. Túnel UNPSJB	10
3.2.1. Historia	10
3.2.2. Motor y Variador de velocidad	11
3.2.3. Instrumentación	12
4. Fluido: Aire	13
4.1. Ecuación velocidad del aire	13
4.2. Características del flujo	14
4.2.1. Estimación del número de Reynolds	14
4.2.2. Flujo turbulento	15
5. Desarrollo	16
5.1. Microcontrolador, sensores y comunicación	16
5.2. Adquisición de datos por puerto serie	18
5.3. Filtros	19
5.3.1. Filtro de mediana	20
5.4. Error observado	21
5.5. Lazo de corriente	22
5.6. Estimación de la planta	23
5.6.1. Diagrama de trabajo	23
5.6.2. Método de estimación	24
5.7. Control	27
5.7.1. Implementación en μ C	30
5.7.2. Pruebas de control	31
6. Pruebas realizadas	34
6.1. Comparación de densidades	34
6.2. Estimación de la densidad ante cambios de Temperatura y Humedad	35
6.3. Contrastación de diferencia de presión	35
6.4. Contrastación de velocidades	38
7. Implementación	40
7.1. Bloque Aplicación	40
7.2. Bloque Hardware	43
7.3. Costos	44
8. Recomendaciones futuras	45
9. Conclusión	46
10.Bibliografia	47

A. Descargas	48
B. Manual de usuario de la aplicación	49
B.1. Requerimientos del sistema	49
B.2. Puesta en marcha	49
B.3. Gráfico en tiempo real	49
B.3.1. Visibilidad de las señales en tiempo real	49
B.3.2. Escala	50
B.3.3. Límite del eje vertical	50
B.4. AutoFucion	50
B.5. Ail	51
B.5.1. Control desactivado	51
B.5.2. Control activado	51
B.6. Encendido/ apagado del motor	52
B.6.1. Panel frontal	52
B.6.2. Aplicación	52
B.7. Guardado de datos	52
B.8. Lectura de datos obtenidos	53
B.9. Falla externa	54
C. Esquemático de la placa implementada	55
D. Código Arduino	59
E. Código Processing	66

Índice de figuras

2.1. Diagrama del objetivo	9
3.1. Clasificación túneles	10
3.2. Tunel vertical de uso recreativo	10
3.3. Tunel UNPSJB	11
3.4. Motor y variador de velocidad	12
3.5. Instrumentación calibrada	13
4.1. Diámetro hidráulico	14
4.2. Dimensiones de la cámara de ensayos en mm	15
4.3. Perforaciones dentro de la cámara de ensayos y "Panal de abeja"	16
4.4. Fluctuaciones alrededor de una velocidad media	16
5.1. Sensores utilizados	17
5.2. Pantalla Processing	19
5.3. Datos y diversos filtros	20
5.4. Datos y filtro mediana	21
5.5. Curva aceleración y desaceleración	21
5.6. Terminales de control	22
5.7. Placa electrónica del variador de velocidad	22
5.8. Placa adaptadora de señal	23
5.9. Diagrama de bloques del procedimiento de modelado de la planta	24
5.10. Estimación de planta	24
5.11. Mediciones de velocidad a partir de distintos escalones dados	25
5.12. Estimación de la planta	25
5.13. Corroboration de estimación de la planta. Ejemplo 1	26
5.14. Corroboration de estimación de la planta. Ejemplo 2	27
5.15. Lazo de control	27
5.16. Ejemplo de prueba realizada	28
5.17. Comparación de PI, escalon de subida	28
5.18. Comparación de PI, escalon de bajada	29
5.19. Comparación 3 sistemas PI	29
5.20. Datos Experimentales y Simulados para la <i>Prueba6</i>	30
5.21. Datos Experimentales y Simulados para la <i>Prueba7</i>	30
5.22. Compuertas utilizadas como perturbaciones	31
5.23. Fotos del reóstato (izquierda) y del banco de resistencias (derecha)	32
5.24. Comportamiento del sistema ante perturbaciones. Ejemplo 1	33
5.25. Comportamiento del sistema ante perturbaciones. Ejemplo 2	33
6.1. Esquema de placa con sensores utilizados	34
6.2. Contraste de valores de presión diferencial del MPX7002	36
6.3. Contraste de valores de velocidad estimada	39
6.4. Captura de pantalla del video utilizado para la realización del contraste de velocidad	39
7.1. Bloques implementados	40
7.2. Bloque <i>Aplicación</i>	40
7.3. Pantalla de la aplicación desarrollada.	42
7.4. Bloque <i>Hardware</i>	43
7.5. Placa construida	43
B.1. Puerto activado. Visualización de los valores del μ C.	49

B.2. Gráfico en tiempo real	50
B.3. Archivos dentro de la carpeta "autofun"	51
B.4. Diagrama para el formato de autofunción	51
B.5. Aplicación en modo Autofunción	52
B.6. Guardado de datos	53
B.7. Archivo .csv generado	54
C.1. Esquema del microcontrolador	55
C.2. Esquema de la placa adaptadora de la señal del sensor de presión diferencial.	56
C.3. Esquema de entradas y salidas del microcontrolador hacia el variador.	57
C.4. Esquema de la placa reguladora de tensión.	57
C.5. Esquema de sensores	58

Índice de tablas

5.1. Factor de escala	18
5.2. Valores de PID's	28
6.1. Comparación de densidades calculadas	34
6.2. Cálculo de la densidad ante cambios de temperatura y humedad	35
6.3. Cálculo de velocidad del aire ante cambios de temperatura y humedad	35
6.4. Contraste de los valores de presión diferencial.	37
6.5. Contraste de los valores de velocidad.	38
7.1. Datos adquiridos desde el μ C	41
7.2. Datos enviados al μ C	42
7.3. Gastos para realizar la implementación	44
B.1. Formato para la confección del archivo .csv	51

1. Introducción

En el Laboratorio de Fluidos de la Universidad, se utiliza el Túnel de Viento para realizar el contraste de anemómetros y experimentos para distintas materias. Gran parte de estas aplicaciones requieren que se conozca la velocidad del fluido (aire). Por lo tanto, la presión diferencial del tubo pitot, humedad, presión atmosférica y temperatura son variables requeridas para lograr estimarla con mayor precisión. Antes del desarrollo de éste proyecto, cada una de éstas variables se medían de forma manual, con sus respectivos instrumentos, para luego ingresar estos valores a una tabla de cálculo para obtener una estimación de la velocidad del aire.

El túnel en sus comienzos, para realizar distintas mediciones, utilizaba un control de velocidad a lazo abierto en el que se modificaba la resistencia rotórica del motor (de inducción de rotor bobinado), para cambiar su velocidad y por lo tanto la velocidad del aire, en pasos discretos. Actualmente, desde principios del año 2020 se utiliza un variador de velocidad de la marca **Long Shenq**, con él se obtiene un control continuo de la velocidad, aunque todavía el control es de lazo abierto.

Realizar este proceso de forma manual, se torna engorroso y poco práctico para la realización de varias mediciones por lo que se realiza este trabajo final de carrera para realizar la *Automatización del Túnel de Viento de la UNPSJB*.

2. Objetivo

El objetivo de este proyecto es generar un lazo de control que tenga como entrada la velocidad de referencia, y que contemple las perturbaciones externas del sistema en el cálculo de la velocidad de salida que se utilizará como lazo de realimentación (Figura 2.1).

Para cumplir con el objetivo, es necesario generar y adaptar una acción de control, que será la referencia de frecuencia del variador de velocidad.

Además se realizará una interfaz gráfica para un mejor manejo y control del sistema.

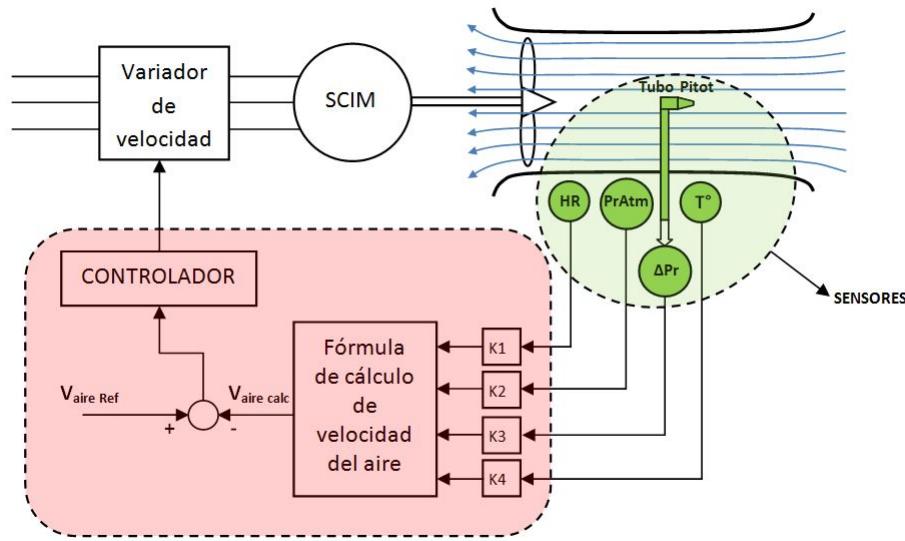


Figura 2.1: Diagrama del objetivo

3. Túnel Aerodinámico

Túnel de viento

Es una herramienta que puede tener dos fines hoy en día, ya sea para un uso recreativo o para propósito científico. Como uso científico se utiliza para observar los efectos del movimiento de aire alrededor de objetos sólidos, como también para la calibración de anemómetros, entre otros.

3.1. Clasificación

Los túneles de viento se pueden clasificar en túneles abiertos o cerrados y a su vez pueden ser verticales u horizontales (Figuras: 3.1 y 3.2)

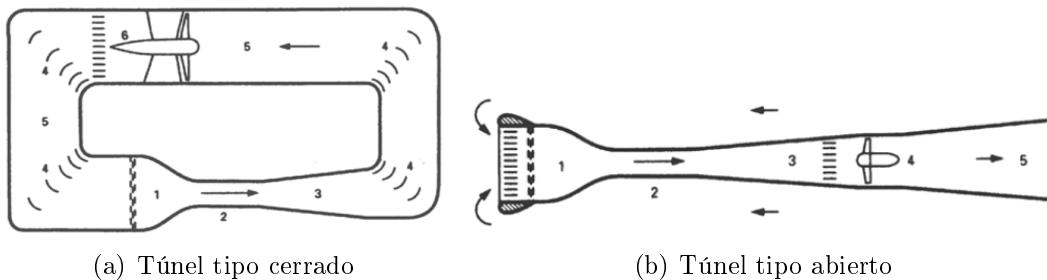


Figura 3.1: Clasificación túneles



Figura 3.2: Tunel vertical de uso recreativo

3.2. Túnel UNPSJB

3.2.1. Historia

[1] El túnel aerodinámico del Laboratorio de Mecánica de Fluidos (**LMF**) de la Facultad de Ingeniería de la Universidad Nacional de la Patagonia San Juan Bosco (**UNPSJB**) es un circuito abierto (tipo Eiffel) con cámara de ensayos cerrada. Puede

clasificarse como un túnel “pequeño de baja velocidad”, con una longitud total de 11 m, una velocidad máxima de 18 m/s y una cámara de ensayos con un área de 0,8 m².



Figura 3.3: Tunel UNPSJB

La entrada del túnel cuenta con canalizadores, comúnmente denominados "panal de abejas", que favorecen la formación de un flujo uniforme y homogéneo.

La cámara de ensayos es vidriada para poder observar con claridad el flujo y está incorporada en un módulo extraíble del túnel, lo cual permite fácil acceso para el armado de los distintos objetos a ensayar.

Los distintos ensayos que se realizan en el túnel son:

- Determinación de coeficientes de resistencia y sustentación de distintos cuerpos y perfiles aerodinámicos.
- Determinación de distribución de presiones a través de diferentes objetos como perfiles aerodinámicos, edificios, puentes, automóviles, etc.
- Visualización con humo del flujo a través de distintos obstáculos.
- Estudio del comportamiento dinámico de generadores eólicos.
- Calibración de anemómetros.

3.2.2. Motor y Variador de velocidad

Variador de velocidad

Es utilizado para controlar la velocidad de giro de un motor. Para regular las revoluciones, se debe tener en cuenta las características del motor, ya que este tiene una curva propia de funcionamiento. Un variador de velocidad es capaz de controlar la aceleración, frenado, torque, generar operaciones que mejoran la eficiencia energética y brindar seguridad.

El motor utilizado para accionar el ventilador del túnel es del tipo de inducción de rotor bobinado, de 30kW de potencia, Marca **AEG**[2]. Desde el año 2020, el laboratorio de Fluidos cuenta con un variador de velocidad de la marca **Long Shenq** modelo **LS650-4045** de 45kW trifásico [3].

Al utilizar este variador, la frecuencia puede ser controlada de forma manual por su panel frontal y/o por entradas tanto de corriente o tensión, las mismas se pueden configurar para el rango máximo de que podría ser de 0 a 300Hz en ambos sentidos, pero

por razones constructivas del motor se utiliza en el rango de 13Hz a 53Hz en un sentido. Al ser un motor antiguo, de la década de 1920, no posee rodamientos sino cojinetes, y estos necesitan funcionar en rangos de aceleración y velocidades acotados, para evitar trabajar sin la necesaria película lubricante que evita el contacto directo entre los cojinetes y el eje de la máquina.



(a) Motor y ventilador durante la construcción del túnel



(b) Variador de velocidad LS650

Figura 3.4: Motor y variador de velocidad

3.2.3. Instrumentación

Los instrumentos utilizados en las mediciones del túnel para la contrastación de anemómetros están calibrados y certificados por el **INTI** (Instituto Nacional Tecnología Industrial).

- **Micromanovacuómetro**

Instrumento utilizado para medir la diferencia de presión.

Marca: Alnor. Modelo: 560. Número de serie: 56057034.

- **Instrumento multifunción**

Utilizado para medir la temperatura, humedad y presión atmosférica. Este mismo elemento puede ser utilizado para medir la velocidad del aire.

Tipo de instrumento: Termómetro electrónico. Sonda de hilo caliente para TESTO 435. Modelo: Testo 435-2. Sonda número: 0635 1025.

Tipo de instrumento: Anemómetro electrónico. Modelo: Testo 435-2. Sonda número: 0635 1025.



(a) AXD 650 (b) Testo 435

Figura 3.5: Instrumentación calibrada

4. Fluido: Aire

4.1. Ecuación velocidad del aire

El cálculo de la velocidad del aire necesita conocer la presión diferencial del tubo pitot, la cual se mide con un sensor de presión diferencial, y la densidad del aire. Ésta última debe ser calculada (Ecuaciones 1, 2 y 3) en función de la presión, la temperatura y la humedad atmosférica [4]. Hasta el momento de la realización de éste proyecto, los cálculos descriptos se hacían en una planilla "*Excel*", con posterioridad a los ensayos en el túnel.

$$\rho = \frac{3,48353 \cdot 10^{-3} \text{ kg K J}^{-1} \cdot p \cdot (1 - 0,378 \cdot x_v)}{Z \cdot T} \quad (1)$$

$$x_v = \frac{(\alpha + \beta \cdot p + \gamma \cdot t^2) \cdot (1Pa \cdot e^{AT^2+BT+C+D/T}) \cdot h / 100}{p} \quad (2)$$

$$Z = 1 - \frac{p}{T} \cdot [a_0 + a_1 t + a_0 t^2 + (b_0 + b_1 t)x_v + (c_0 + c_1 t)x_v^2] + (d + x_v^2) \frac{p^2}{T^2} \quad (3)$$

dónde:

- **p** [Pa] presión atmosférica medida,
 - **t** [°C] temperatura medida,
 - **T** [K] temperatura absoluta ($T=t + 273,15\text{ K}$)
 - **h** [%] humedad relativa medida,
 - y constantes **A,B,C,D, α , β , γ , a_0 , a_1 , a_2 , b_0 , b_1 , c_0 , c_1 , D .**

Finalmente, la ecuación 1 se utiliza para el cálculo final de la velocidad del aire.

$$\Delta P = \frac{v^2 \rho}{2} \rightarrow v = \sqrt{\frac{2 \cdot \Delta P}{\rho}} \quad (4)$$

En vista al objetivo y por las ecuaciones 1 y 4 se estableció necesario la obtención de las variables presión diferencial del tubo pitot, causada por el aire, y temperatura, presión y humedad relativa atmosféricas.

4.2. Características del flujo

La inspección cuidadosa en una tubería revela una corriente de fluidos currentilínea a bajas velocidades y caótica mientras la velocidad aumenta por arriba de un valor crítico. Se dice que el régimen de flujo en el primer caso es laminar, y se caracteriza por líneas de corriente suaves y movimiento sumamente ordenado; mientras que en el segundo caso es turbulento, y se caracteriza por fluctuaciones de velocidad y movimiento desordenado. La mayoría de los flujos que se encuentran en la práctica son turbulentos.

4.2.1. Estimación del número de Reynolds

Número de Reynolds

El número de Reynolds (Re) es un número adimensional utilizado en mecánica de fluidos para caracterizar el movimiento de un fluido. Su valor indica si el flujo sigue un modelo laminar o turbulento.

El número de Reynolds fue calculado a partir de la ecuación 5, que relaciona las fuerzas inerciales y las fuerzas viscosas.

$$R_e = \frac{\rho D_h v}{\mu} \quad (5)$$

dónde:

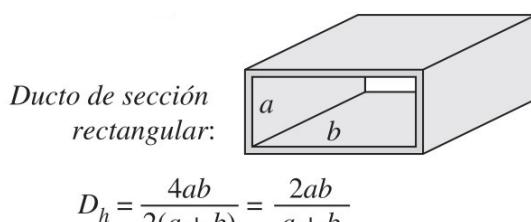
- ρ [kg/m^3] densidad del fluido
- v [m/s] velocidad del fluido
- D_h [m] diámetro hidráulico
- μ [$kg/m.s$] viscosidad dinámica del fluido

La *densidad del fluido* se calculó con la ecuación 1 vista en la sección 4.1 con mediciones de temperatura, humedad y presión atmosférica obtenidas con el instrumento *Testo 435*.

La *viscosidad dinámica* del fluido se obtuvo de la tabla "Propiedades del aire a 1 atm de presión" del libro *Mecánica de Fluidos* [5].

El *diámetro hidráulico* de una tubería de sección rectangular es dependiente del valor de sus lados (Figura 4.1 y 4.2) [6].

Para la *velocidad del fluido* se utilizó un valor de $4,7 m/s$, que corresponde a la velocidad mínima generada dentro de la cámara de ensayos del túnel. Ya que al ser proporcional al número de Reynolds, este será el mínimo valor estimado.



$$D_h = \frac{4ab}{2(a+b)} = \frac{2ab}{a+b}$$

Figura 4.1: Diámetro hidráulico

Si se reemplaza los valores se obtiene:

$$R_e \approx \frac{1,193 \text{ } kg/m^3 \text{ } 0,86 \text{ } m \text{ } 4,7 \text{ } m/s}{0,000018 \text{ } kg/m.s} \approx 270658 \quad (6)$$

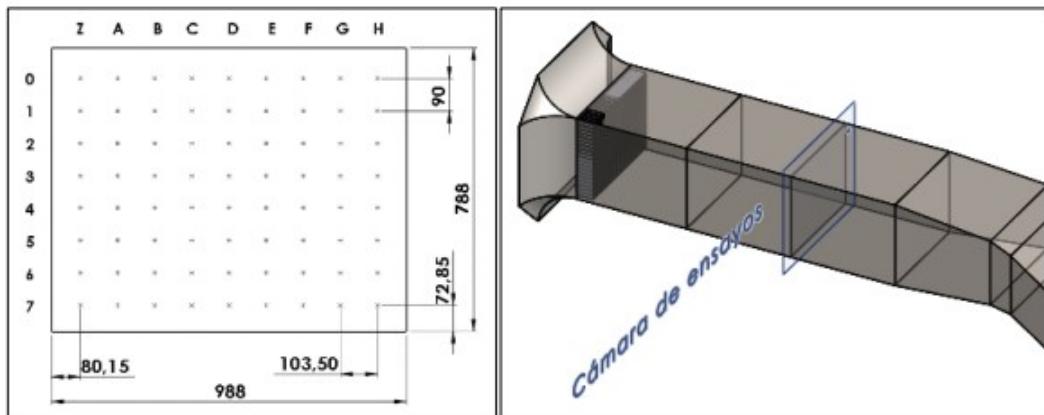


Figura 4.2: Dimensiones de la cámara de ensayos en mm

Por definición, un flujo es turbulento si el valor del número de Reynolds es mayor a 4000. Por lo que se observa claramente, que para los valores de velocidad utilizados en el túnel este valor siempre será mayor.

4.2.2. Flujo turbulento

Flujo turbulento

Este se caracteriza por fluctuaciones aleatorias y rápidas de regiones giratorias de fluido, llamadas remolinos a lo largo del flujo. En flujo laminar, las partículas fluyen en orden a lo largo de trayectorias, en cambio, en flujo turbulento, los molinos giratorios transportan masa, cantidad de movimiento y energía a otras regiones del flujo. [5]

Aun mientras el flujo promedio sea estacionario, el movimiento de remolinos en flujo turbulento provoca fluctuaciones importantes en los valores de velocidad, temperatura, presión e incluso densidad (en flujo compresible). Las posibles causas de la formación de estos remolinos se pueden deber a la construcción del túnel. Este, posee una rejilla de entrada en forma de panal de abeja, que al estar cerca de la cámara de ensayos, no logra realizar una correcta formación de flujo uniforme y homogéneo, otra causa es la existencia de orificios en la parte inferior de la cámara que son utilizados para la colocación de instrumentos (Figura 4.3).

La figura 4.4 muestra, en este caso, las fluctuaciones alrededor de una velocidad media dentro de un tiempo específico. Para determinar la velocidad media se analiza un intervalo de tiempo suficientemente grande, de modo que el valor promediado en el tiempo se estabilice en una constante.



Figura 4.3: Perforaciones dentro de la cámara de ensayos y "Panal de abeja"

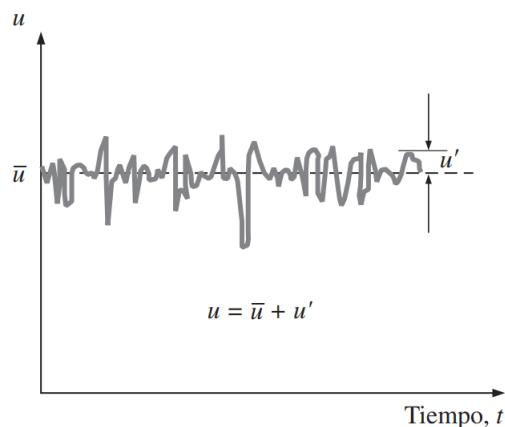


Figura 4.4: Fluctuaciones alrededor de una velocidad media

5. Desarrollo

5.1. Microcontrolador, sensores y comunicación

Microcontrolador

Es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto por unidad central de procesamiento (CPU), memorias (ROM y RAM), líneas de entrada/salida, módulos de comunicación, DAC, ADC, entre otros.

I²C

Es un protocolo de comunicación serial, define la trama de datos y las conexiones físicas para transferir bits entre 2 dispositivos digitales. El puerto incluye dos cables de comunicación, SDA (Datos seriales) y SCL (reloj serial). Además el protocolo permite conectar hasta 127 dispositivos esclavos mediante dos líneas, con velocidades de hasta 5 Mbps, 10 veces mayor que un puerto serial estándar.

Para la elección del microcontrolador se analizaron varios dispositivos, como por

ejemplo *Arduino UNO*, *Arduino MEGA*, *Arduino NANO*, *EDUCIAA*, entre otros. Se llegó a la conclusión que **Arduino NANO** era un microcontrolador de bajo costo, tamaño reducido, baja complejidad en la programación, con bibliotecas de diversos sensores y que cumplía con las prestaciones necesarias para lograr el objetivo (señal de PWM, comunicación I²C, entradas analógicas y digitales).

Como se determinó en la sección 4.1, es necesario medir la temperatura, la humedad relativa, la presión atmosférica y la presión diferencial del tubo pitot. Por lo tanto se realizaron ensayos con diversos sensores en una protoboard comunicados con el microcontrolador por medio del protocolo de comunicación I²C. Analizando dichas mediciones, se eligieron aquellos que generaron la menor variación de temperatura, humedad y presión atmosférica (THP) respecto a los instrumentos del laboratorio (Figura 3.5).

Sensores elegidos

- **BME280:** Sensor de presión atmosférica, temperatura y humedad relativa.

Resolución de temperatura: 0.01 °C

Resolución de humedad: 0,008 % HR

Resolución de presión atmosférica: ±1hPa

- **SHT21:** Sensor de temperatura y humedad relativa.

Resolución de temperatura: 0.01 °C

Resolución de humedad: 0,04 % HR

- **MPXV7002:** Sensor de presión diferencial.

Sensibilidad: 1 V/kPa

Rango de presión: -2 kPa a 2 kPa

- **ADS1115:** Convertidor analógico digital utilizado en conjunto con MPXV7002.

Resolución: 15 bits

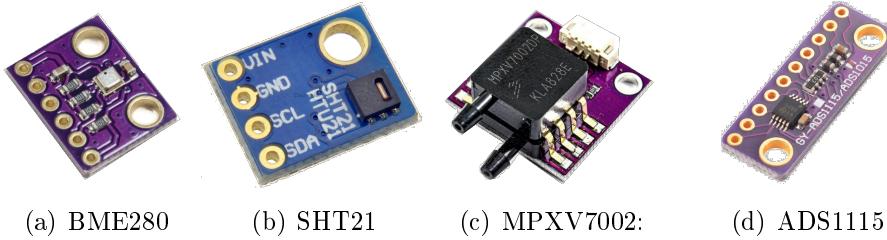


Figura 5.1: Sensores utilizados

Se realizaron ensayos para probar y corroborar el funcionamiento del **MPXV7002**, un sensor de diferencia de presión de alto costo en el país. Este sensor es capaz de medir de -2kPa a 2kPa en un rango de 4V (0.5 a 4.5V), al tener un único sentido de flujo, solo se necesitó medir valores positivos de presión.

El sensor MPXV7002 posee una salida de voltaje analógico, por lo que se utilizó el conversor analógico/ digital ADS 1115. El uso del ADC interno del microcontrolador se descartó debido a su resolución de 10 bits, mucho menor que la del ADS1115, que es de 15 bits. Además, el rango de valores a convertir en el ADC del Arduino, está definido por un voltaje de referencia, de 5V, y en el ADS1115 ese valor puede ser seleccionado por un amplificador de ganancia programable (PGA). Por defecto este valor de referencia es ±6,144 V, quiere decir que el valor de 32.677 (valor máximo con 15-bit) corresponde a 6,144 V.

Resolución en Arduino NANO:

$$\text{Factor de escala} = \frac{5 \text{ V}}{1023} = 0,0048875 \text{ V} = 4,88 \text{ mV}$$

Resolución en ADS115:

$$\text{Factor de escala} = \frac{6,144 \text{ V}}{32677} = 0,0001875 \text{ V} = 0,1875 \text{ mV}$$

En la tabla 5.1 se observa los valores posibles de factor de escala para distintos valores del PGA. Este valor se eligió para aprovechar la mayor resolución posible al tener en cuenta el sensor utilizado.

PGA	Referencia (V)	Factor de Escala (mV)
2/3	6,144	0,1875
1	4,096	0,125
4	1,024	0,0312
8	0,512	0,0156
16	0,256	0,0078

Tabla 5.1: Factor de escala

En el caso de este proyecto, se utilizó un valor de PGA igual a 1, cuya tensión de referencia es 4,096V.

$$\text{Factor de escala} = \frac{4,096 \text{ V}}{32677} = 0,000125 \text{ V} = 0,125 \text{ mV}$$

5.2. Adquisición de datos por puerto serie

Processing

Es un lenguaje de programación basado en Java, aunque hace uso de una sintaxis simplificada y de un modelo de programación de gráficos. [7]

Para realizar la adquisición de datos por puerto serie, se utilizó en un primer momento Matlab, dónde, al realizar la comunicación con el microcontrolador, se guardaban los valores, pero estos no podían ser observados en tiempo real, sino que luego de las pruebas, se utilizaba otro *script* para obtener conclusiones. En consecuencia, después de cierta cantidad de pruebas realizadas, se tornó indispensable observar los valores adquiridos, como la velocidad estimada en tiempo real.

Como solución para el inconveniente anteriormente nombrado, se realizó una primera versión (Figura 5.2) de una interfaz gráfica generada con Processing. Este código fue capaz de adquirir datos desde el microcontrolador en tiempo real y mostrarlos tanto gráfica como numéricamente.

Para realizar la estimación de la planta (Sección 5.6), fue necesario enviar escalones de señal al variador de velocidad a través del microcontrolador. Esto fue posible al ingresar valores numéricos por teclado que eran enviados al μC para generar la respectiva señal. Luego ésta es ingresada al variador por un lazo de corriente. Ver (Sección 5.5).

Al presionar una tecla específica configurada por los comandos, el programa guardaba los datos adquiridos durante la prueba en un archivo .csv para luego ser observadas con Matlab.

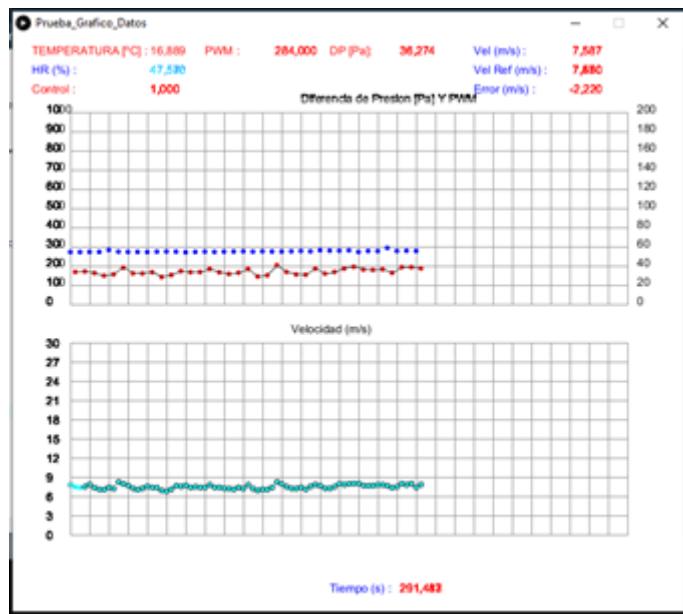


Figura 5.2: Pantalla Processing

5.3. Filtros

Filtro digital

Es una operación matemática que toma una secuencia de números (señal de entrada) y la modifica para producir otra (señal de salida) con el objetivo de resaltar o atenuar ciertas características.

Una vez que se procedió a tomar diversos valores de velocidad estimada, se notó que era necesaria la implementación de un filtro a causa de las fluctuaciones alrededor de una velocidad media. Para esto se utilizaron varias bibliotecas preestablecidas de **Arduino** con las cuales se realizó la implementación de diversos filtros en el μ C, con diferentes parámetros para una posterior comparación (Figura 5.3).

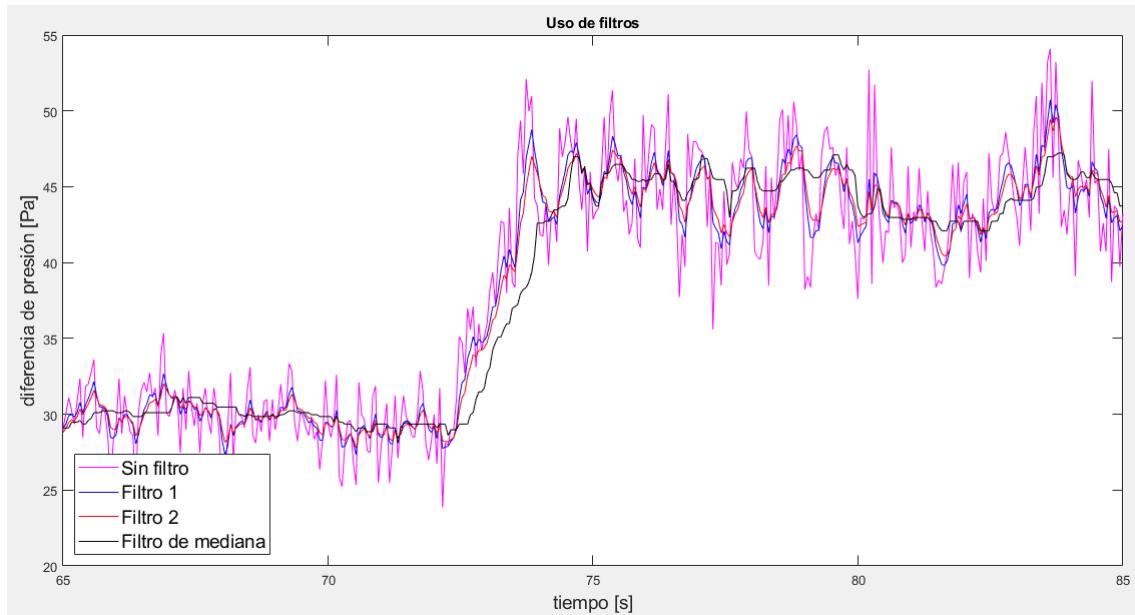


Figura 5.3: Datos y diversos filtros

5.3.1. Filtro de mediana

Mediana

Es una técnica de filtrado digital no lineal que suele utilizarse para eliminar el ruido de una imagen o señal. La mediana es el número que está justo en el medio de un conjunto de datos ordenados de menor a mayor o de mayor a menor. La idea principal del filtro de mediana es recorrer la señal entrada, y sustituir cada dato por la mediana de una ventana de “N” datos.

Al observar y realizar la comparación de los filtros, se decidió utilizar el filtro de mediana (Figura 5.4), con una ventana igual a 40 muestras, que producía menor ruido en la estimación del aire. Como consecuencia del uso de este filtro, la señal se vio desfasada aproximadamente 1 segundo.

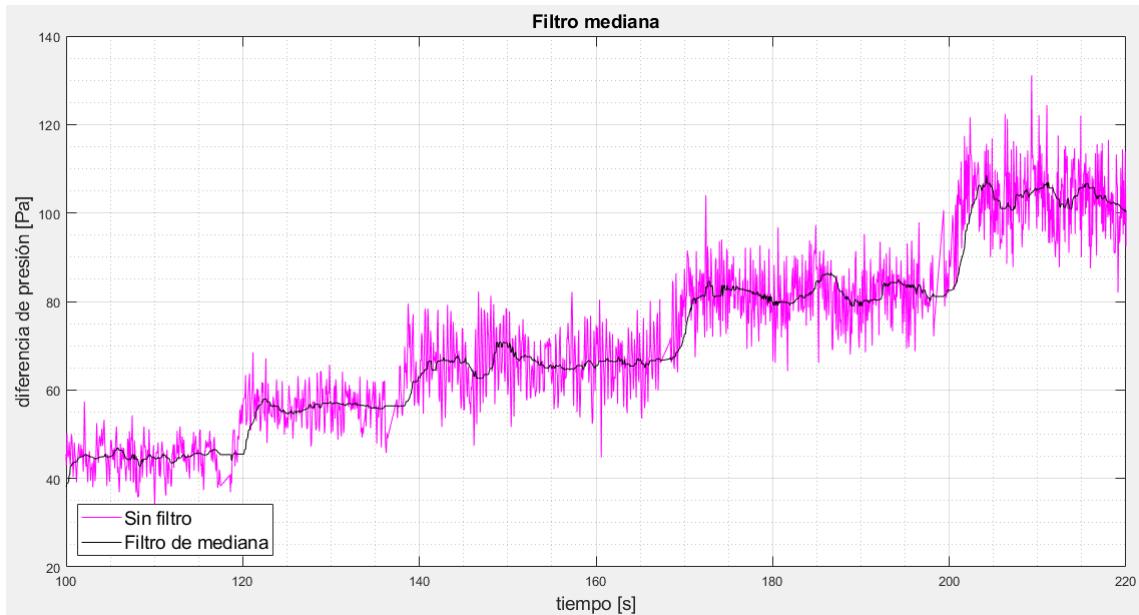


Figura 5.4: Datos y filtro mediana

5.4. Error observado

Al utilizar el potenciómetro frontal del variador de velocidad u otro modo de funcionamiento distinto al panel digital frontal, se originaba un error en la aceleración o desaceleración. Al estudiar el manual del variador se determinó que dicho error se debía a una configuración interna de éste, y a las características propias del motor.

Si el variador se utiliza con el panel digital frontal, la curva de aceleración y desaceleración sigue una “s” (Figura 5(a)) y no produce un cambio brusco en la velocidad del motor, en cambio, para el uso del potenciómetro u otro modo de funcionamiento la curva es lineal (Figura 5(b)). Para revertir esto, se modificaron los parámetros del tiempo de aceleración y desaceleración del variador a 20 segundos.

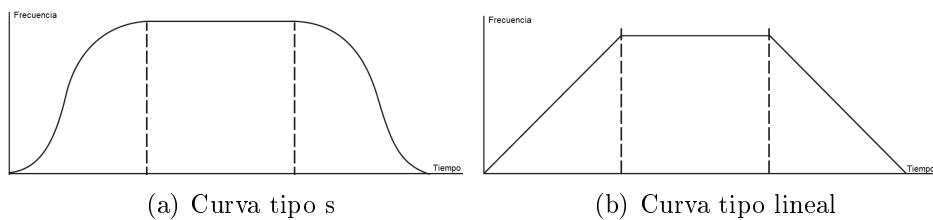


Figura 5.5: Curva aceleración y desaceleración

5.5. Lazo de corriente

Se realizó la modificación de los parámetros nombrados en la sección 5.4 y se corroboró que el error no aparecía al cambiar el modo de funcionamiento del variador.

Para realizar la comunicación del microcontrolador con el variador de velocidad se decidió utilizar el modo de funcionamiento de entrada analógica de dos hilos, ingresando por el puerto (Ai1) de la bornera del variador (Figura 5.6). Este modo analógico puede ser configurado de 0-10V o 0-20mA a través del “jumper J3” (Figura 5.7).

Se optó por la utilización de un lazo de corriente, por ser más estable e inmune a los ruidos eléctricos e interferencias electromagnéticas. Normalmente, se utilizan lazos de corriente de 4-20mA para poder observar si hubiera fallas en el circuito, por lo que fue necesario adaptar la señal generada por el μ C para seguir el estándar.

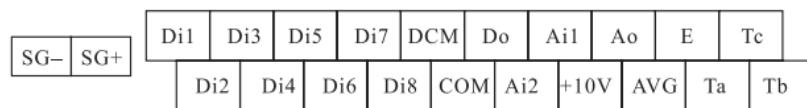


Figura 5.6: Terminales de control

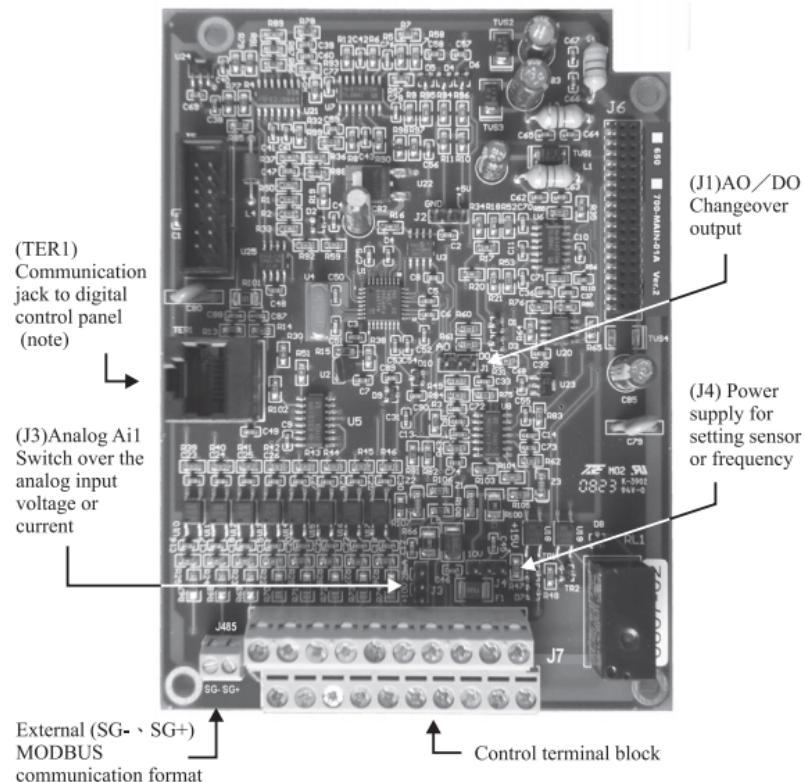


Figura 5.7: Placa electrónica del variador de velocidad

La señal analógica para realizar el control del variador de velocidad fue generada por una señal PWM estipulada a través de la biblioteca *TimerOne* del μ C. Con esta biblioteca se configuró la frecuencia de la señal PWM, la cual fue de 25 kHz.

Para la señal PWM generada, se utilizó el Pin 9 del μ C, y se ajustó el ciclo de trabajo entre 0 y 1023 ya que era la resolución máxima que disponía la biblioteca. El Pin 9 de salida tiene un rango de tensión de 0 a 5V, por lo que fue necesario realizar la transformación

de esta señal a una señal de corriente a través una “placa adaptadora de señal” (Figura 5.8). Esta placa, tiene como entrada el valor de tensión anteriormente nombrado y genera una señal de salida de 0 a 20mA. Por medio de un parámetro del variador, se modificó la señal para adaptarla al estándar de 4 a 20mA.

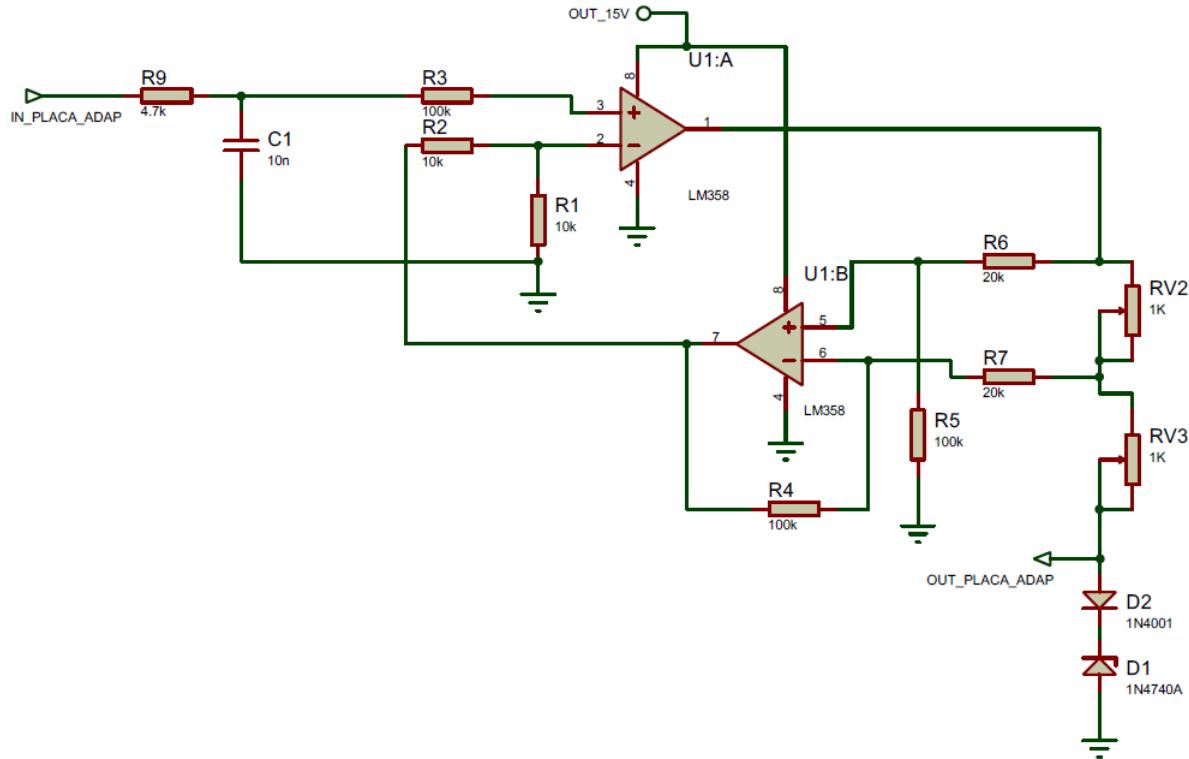


Figura 5.8: Placa adaptadora de señal

5.6. Estimación de la planta

5.6.1. Diagrama de trabajo

Para realizar la estimación de la planta se muestra un diagrama de bloques del procedimiento que se siguió de forma resumida.

Generación de datos: De acuerdo a diversas pruebas realizadas, se eligen las variables de mayor interés para analizarlas y así poder enviar la información de forma eficaz.

Captura de datos: A través del puerto serie y con Processing se realiza el almacenamiento de los datos de respuesta del sistema ante el estímulo de las señales de excitación. Posteriormente, el análisis de los datos y la generación de las gráficas correspondientes es realizado por medio de rutinas de código implementadas en Matlab.

Identificación del modelo matemático: Se utilizaron varios métodos de identificación experimental, todos ellos analizados con la respuesta al escalón del sistema.

Validación del modelo matemático: Una vez obtenida la mejor estimación, se efectuó una validación adicional a partir de la comparación de datos experimentales con los teóricos generados en simulaciones.

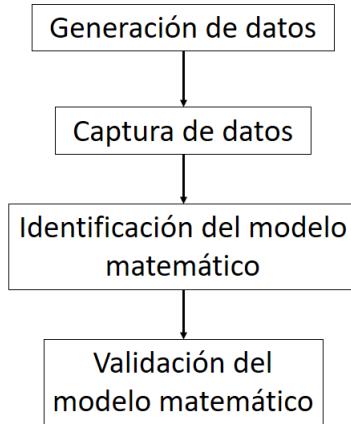
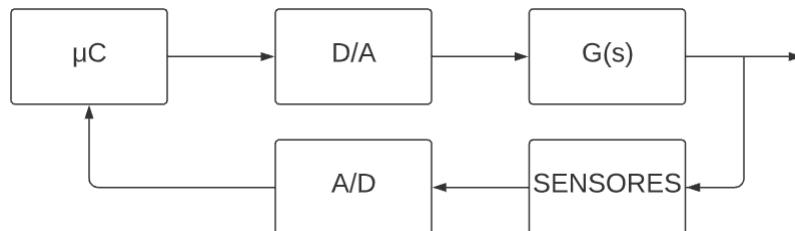


Figura 5.9: Diagrama de bloques del procedimiento de modelado de la planta

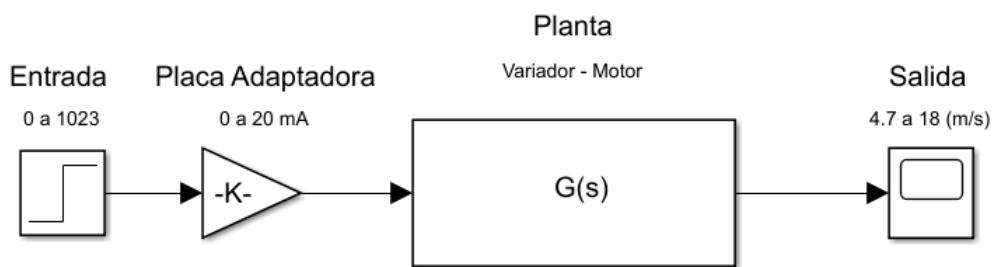
5.6.2. Método de estimación

Una vez que se determinó el valor de la ventana del filtro, velocidad de conmutación de PWM, tiempos de aceleración y desaceleración, etc, se utilizó el modo de ingreso de señal por lazo de corriente al variador de velocidad.

La Figura 5.10 muestra un diagrama resumido de los pasos a realizar para tomar datos de la planta. $G(s)$ es el conjunto del túnel de viento, variador de velocidad y motor.



(a) Diagrama en bloques



(b) Diagrama de conexión

Figura 5.10: Estimación de planta

Para realizar la estimación del modelo matemático de la planta, se obtuvieron y guardaron en tablas los datos de las variables, para distintos escalones de entrada, mediante el programa realizado en Processing. En la figura 5.11 se observa los valores de velocidad y los escalones que se realizaron durante una prueba generada para obtener los parámetros necesarios para la estimación de la planta.

Seguidamente, se procedió a generar un nuevo código de Matlab donde se cargaron los datos obtenidos de la prueba (Figura 5.11) y con ellos se observaron las características

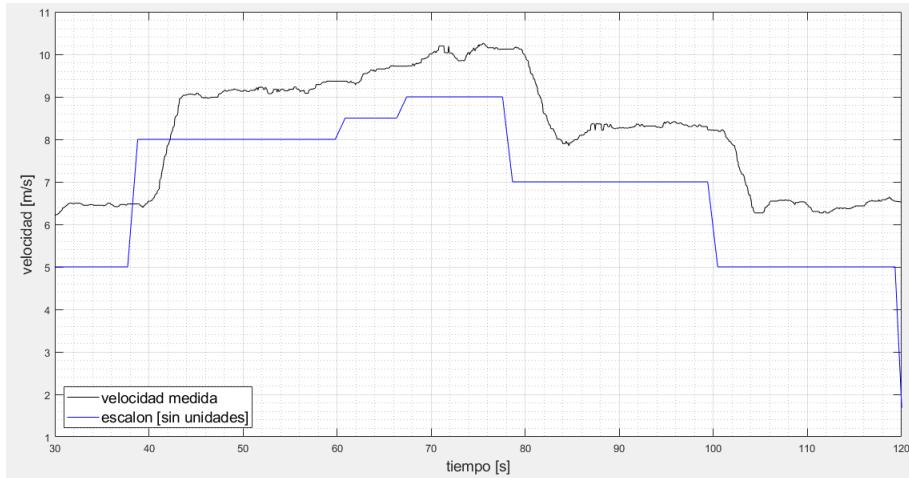
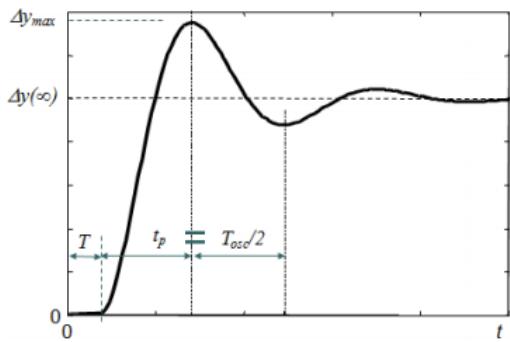
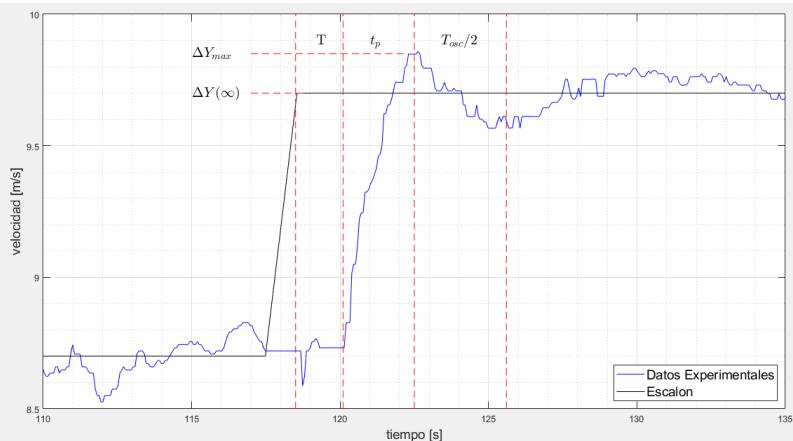


Figura 5.11: Mediciones de velocidad a partir de distintos escalones dados

necesarias de la figura (a) 5.12 para realizar la estimación de la planta a través de la comparación con la función de transferencia de un sistema de "*2º orden con retardo*" [8].



(a) Parámetros respuesta 2º orden



(b) Estimación de parámetros

Figura 5.12: Estimación de la planta

El sistema de "*2º orden con retardo*" posee la función de transferencia observada en la ecuación 7.

$$G(s) = \frac{k\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} * e^{-T.s} \quad (7)$$

La ecuación de sobre oscilación (Ecuación 8), es utilizada para calcular el factor de amortiguamiento (ξ)

$$\delta = \frac{\Delta y_{max} - \Delta y(\infty)}{\Delta y(\infty)} = e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}} \quad (8)$$

La ecuación 9 es utilizada para obtener el valor de la frecuencia natural (ω_n) a partir del tiempo de pico.

$$t_p = \frac{T_{osc}}{2} = \frac{\pi}{\omega_n \sqrt{1 - \xi^2}} \quad (9)$$

Con ayuda de un código generado en Matlab y ajustes manuales se llegó a la siguiente función de transferencia de segundo orden:

$$\frac{Y(s)}{U(s)} = \frac{0,02483}{s^2 + 1,846s + 1,535} * e^{-1,2.s} \quad (10)$$

Esta función de transferencia se realizó para una entrada entre 0 y 1023, los cuales son el límite mínimo y máximo para determinar el ancho de pulso de la señal PWM como se ve en la sección 5.5.

Para corroborar la elección del modelo matemático de la planta se realizó en *Simulink* la respuesta del sistema a los mismos escalones experimentales. En la figura 5.13 y 5.14, se dispuso la respuesta experimental y la simulada para realizar la comparación.

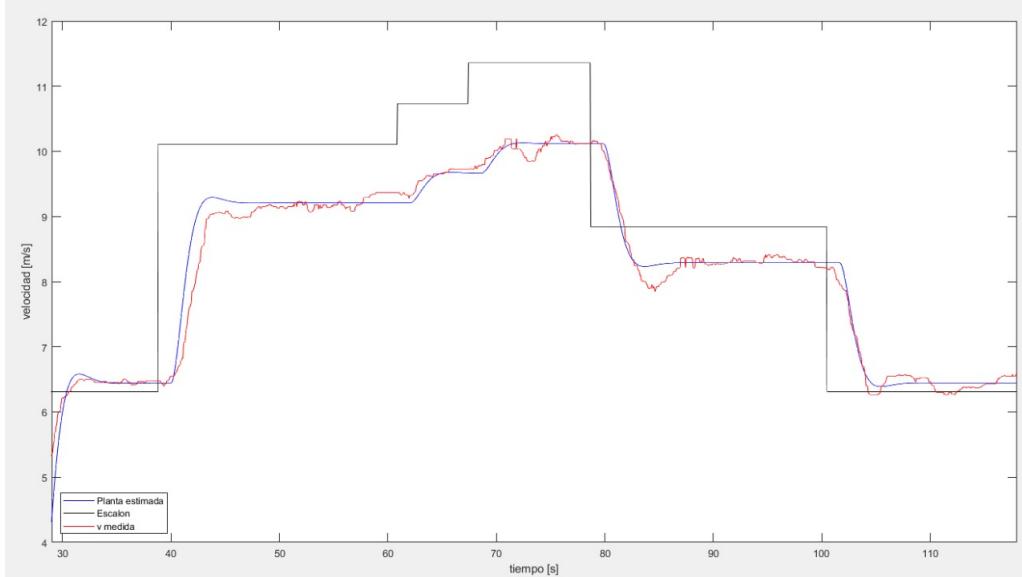


Figura 5.13: Corroboration de estimación de la planta. Ejemplo 1

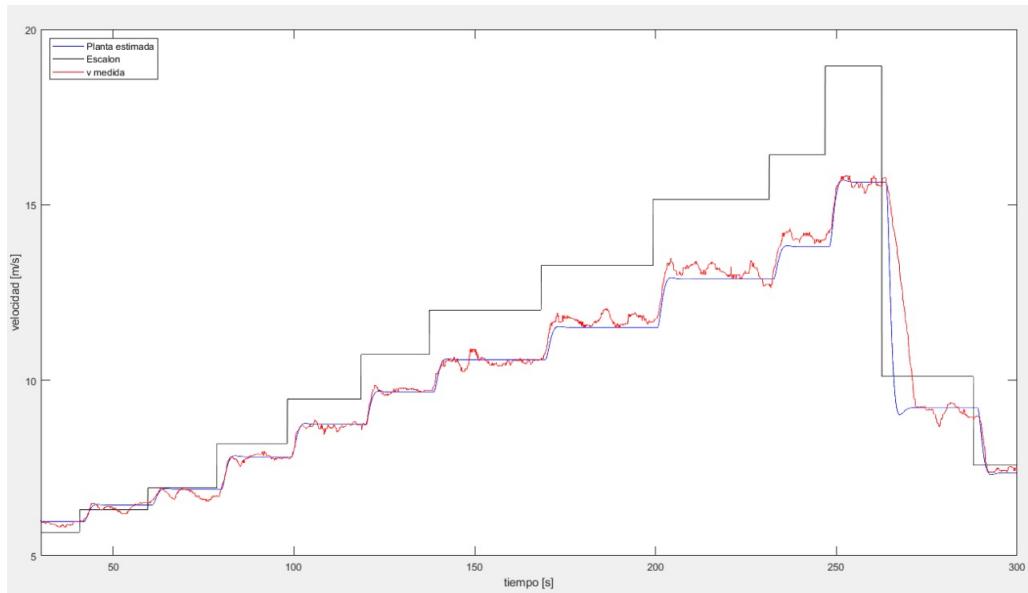


Figura 5.14: Corroboration de estimación de la planta. Ejemplo 2

5.7. Control

Para el cálculo en primera instancia del controlador PID, se utilizó la herramienta de Matlab "PID Tuner", y a partir de los primeros valores se realizaron pequeñas modificaciones, todas con el formato PI (proporcional - integrador) hasta obtener varios tipos de respuesta. Con los distintos valores de PI (Tabla 5.2) se realizaron pruebas en el Túnel del viento con escalones en la velocidad de referencia de 6 - 7,5 - 8,5 - 7,5 - 7 - 6 m/s, un total de 5 ensayos para los mismos estímulos de entrada. En la figura 5.15 se observa el lazo de control que se utilizó.

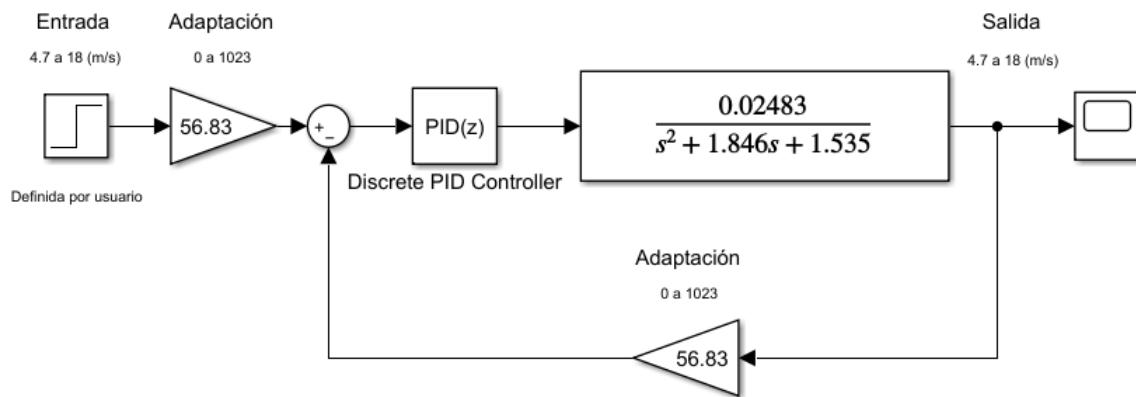


Figura 5.15: Lazo de control

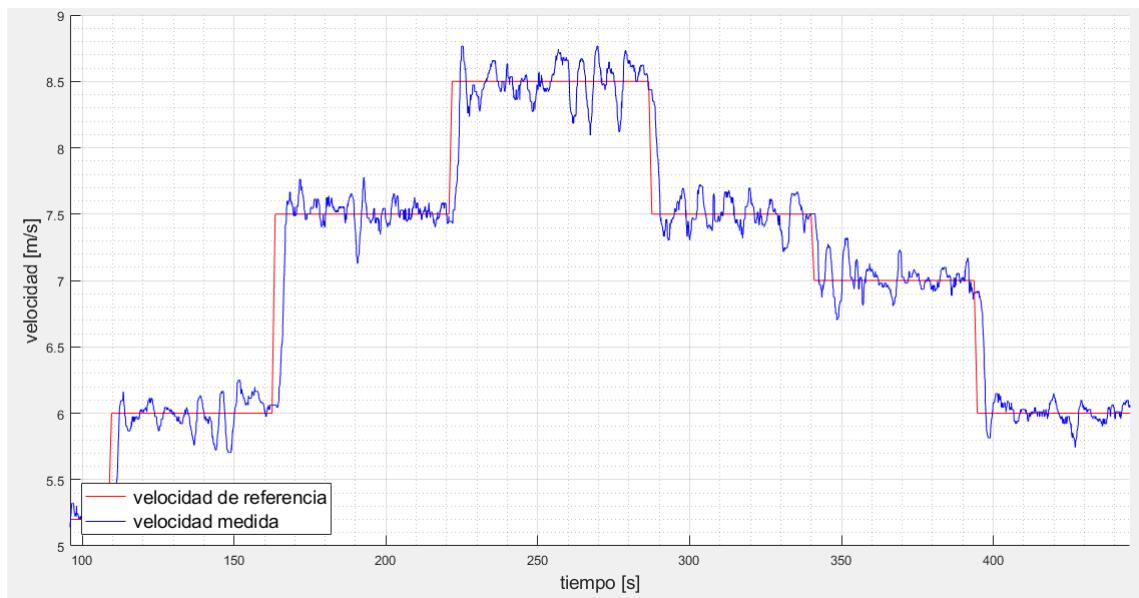


Figura 5.16: Ejemplo de prueba realizada

Los valores de cada configuración PI se muestran en la tabla siguiente

	PI anterior	Prueba3	Prueba4	Prueba5	Prueba6	Prueba7
P=	0.225	0.0699	0.244	0.5451	0.6846	0.3286
I=	0.326	0.2035	0.2756	0.3599	0.4183	0.3107

Tabla 5.2: Valores de PID's

Al visualizar las respuestas obtenidas por las diferentes configuraciones se puede realizar la comparación para un escalón en donde la velocidad sube (figura 5.17) y otro donde la velocidad baja (figura 5.18).

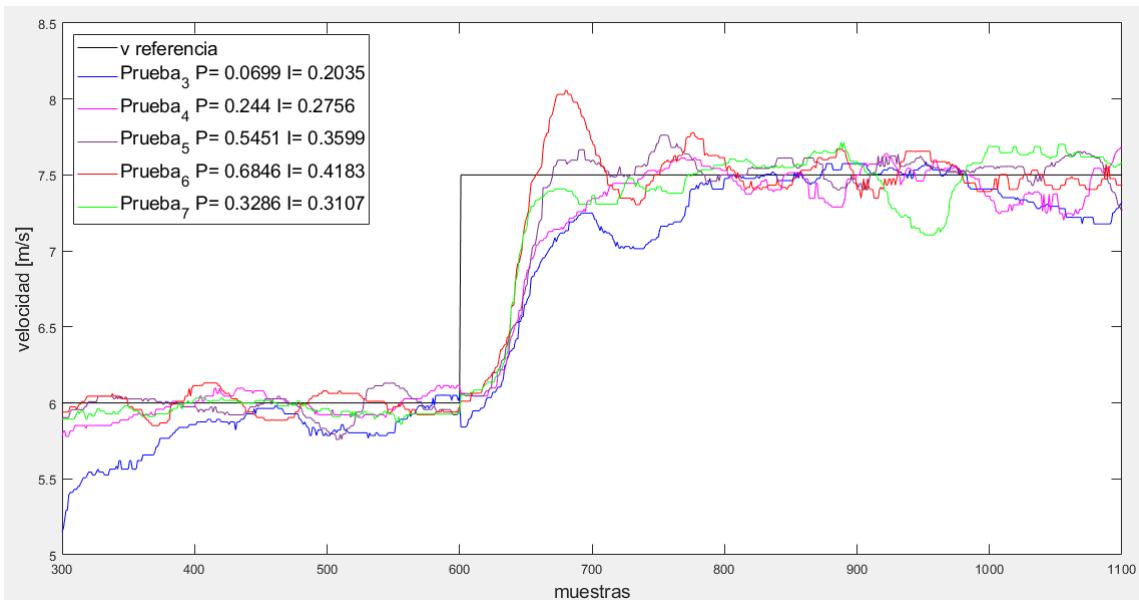


Figura 5.17: Comparación de PI, escalon de subida

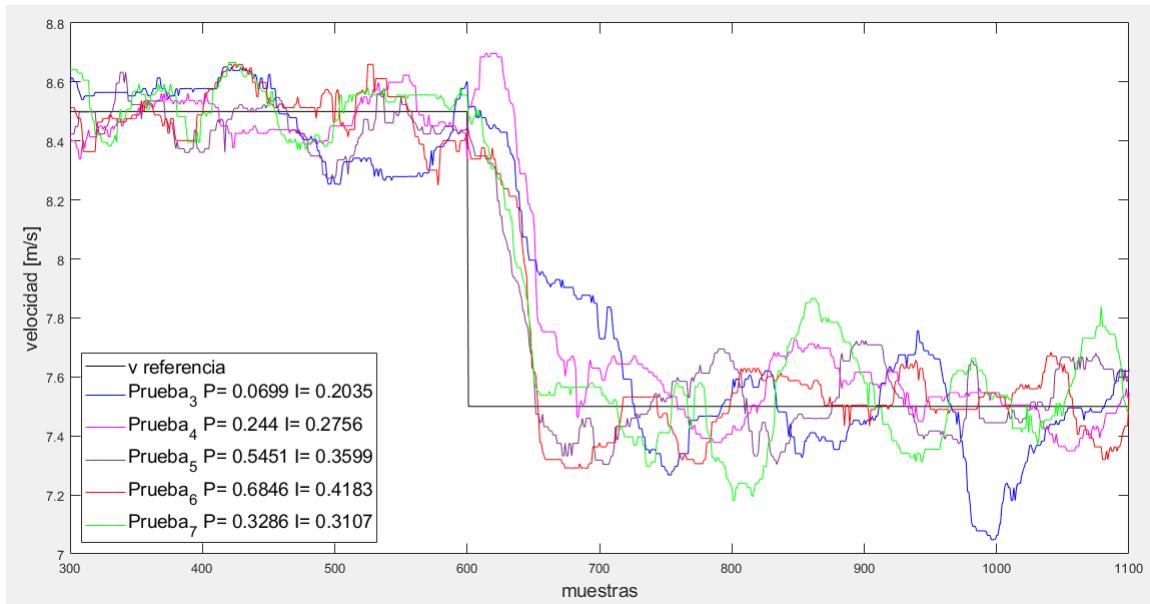


Figura 5.18: Comparación de PI, escalon de bajada

En la figura 5.19 se pueden observar el sistema que genera mayor sobrepico, un sistema con una respuesta mas lenta, y un sistema con respuesta intermedia.

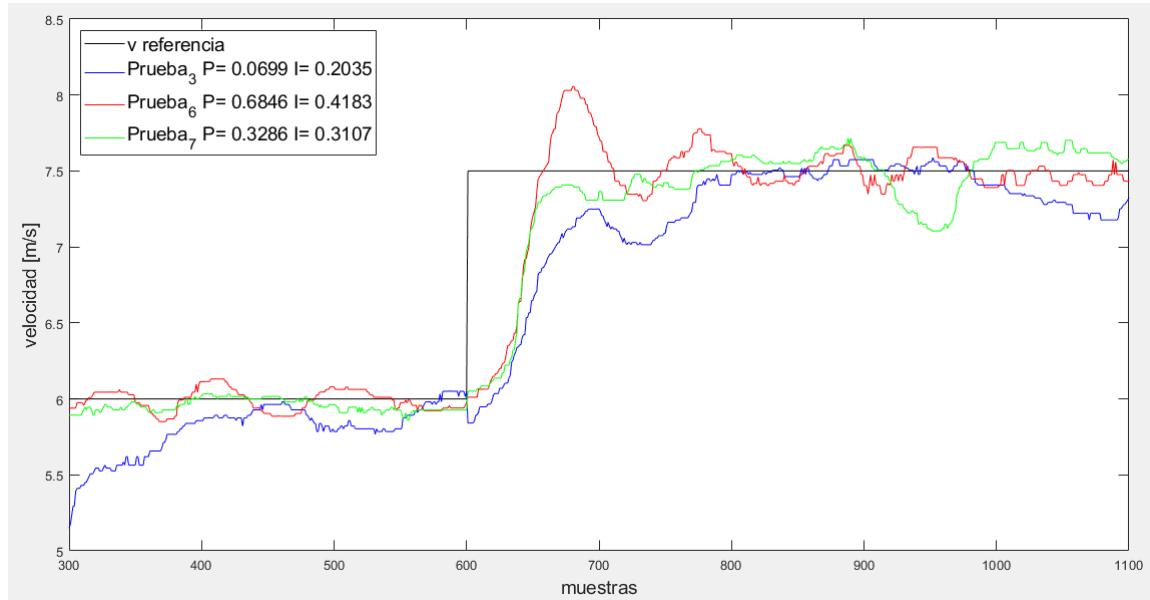


Figura 5.19: Comparación 3 sistemas PI

Al analizar los ensayos y obtener respuestas del sistema parecidas (Figura 5.20 y 5.21) a las generadas en simulación, se determinó que la estimación de la planta se realizó de manera correcta, esto añadió mayor validez al sistema de simulación empleado en todas las pruebas.

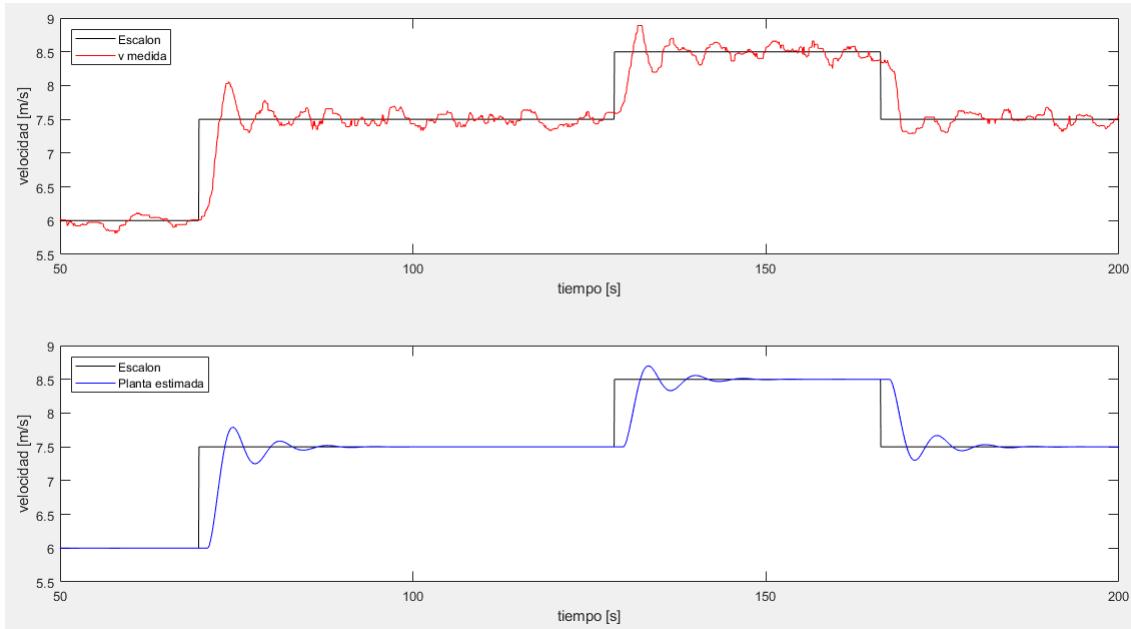


Figura 5.20: Datos Experimentales y Simulados para la *Prueba 6*

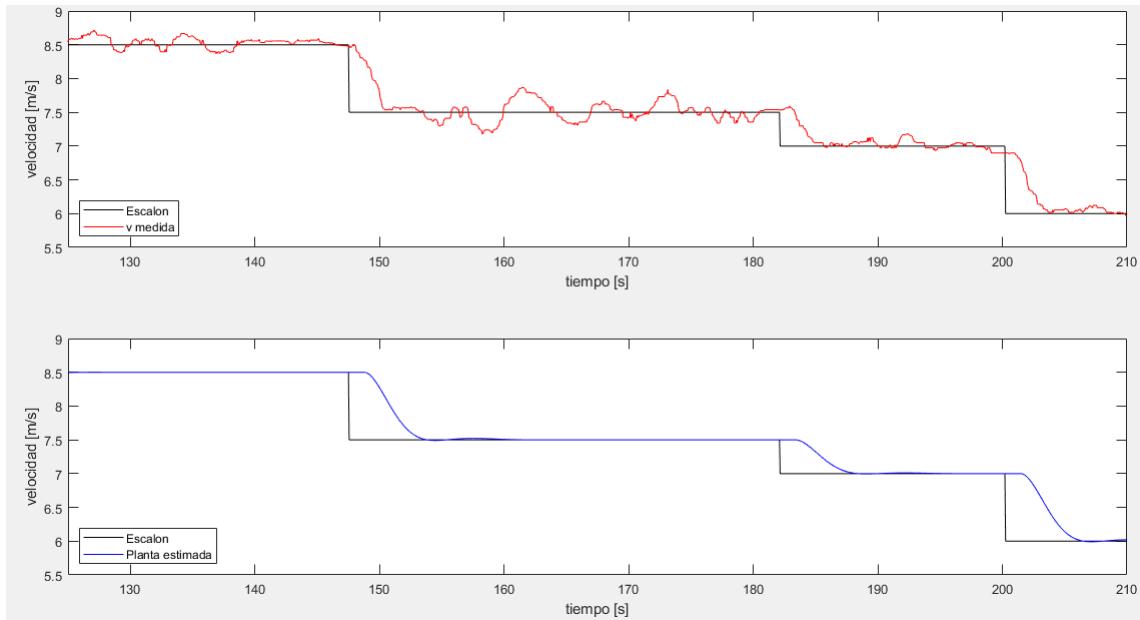


Figura 5.21: Datos Experimentales y Simulados para la *Prueba 7*

5.7.1. Implementación en μ C

Para poder implementar en nuestro μ C el PID, se partió de la estructura interna o fórmula del controlador general discreto en el dominio Z (Ecuación 11) y se trabajó algebraicamente hasta llegar a una forma en donde aplicar la *Transformada Inversa Z* sea más simple. Se utiliza esta herramienta matemática para llevar la ecuación del $PID(z)$ al dominio de tiempo discreto.

$$PID(z) = \frac{U(z)}{E(z)} = P + I.T_s \cdot \frac{1}{z-1} + D \cdot \frac{1}{T_s} \cdot \frac{z-1}{z} \quad (11)$$

$$\frac{U(z)}{E(z)} = P \cdot \frac{z^{-1}}{z-1} + I.T_s \cdot \frac{1}{z-1} \cdot \frac{z^{-1}}{z-1} + D \cdot \frac{1}{T_s} \cdot \frac{z-1}{z} \cdot \frac{z^{-1}}{z-1} \quad (12)$$

$$\frac{U(z)}{E(z)} = P + I.T_s \cdot \frac{z^{-1}}{1-z^{-1}} + D \cdot \frac{1}{T_s} \cdot (1-z^{-1}) \quad (13)$$

$$\frac{U(z)}{E(z)} = \frac{1}{1-z^{-1}} \left[P \cdot (1-z^{-1}) + I.T_s \cdot z^{-1} + D \cdot \frac{1}{T_s} \cdot (1-z^{-1})^2 \right] \quad (14)$$

$$U(z) \cdot (1-z^{-1}) = E(z) \left[P + \frac{D}{T_s} + \left(I.T_s - P - 2 \cdot \frac{D}{T_s} \right) \cdot z^{-1} + \frac{D}{T_s} \cdot z^{-2} \right] \quad (15)$$

Y si aplicamos la transformada inversa Z a la ecuación 15:

$$u(k) = u(k-1) + \left(P + \frac{D}{T_s} \right) \cdot e(k) + \left(I.T_s - P - 2 \cdot \frac{D}{T_s} \right) \cdot e(k-1) + \frac{D}{T_s} \cdot e(k-2) \quad (16)$$

De esta forma obtenemos la ecuación del PID en función de la muestra actual y anteriores, que nos sirve para implementar de manera directa en el µC.

5.7.2. Pruebas de control

Para observar la respuesta del control se generaron perturbaciones con las compuertas laterales del túnel (Figura 5.22), antes utilizadas para realizar variaciones de velocidad [2] cuando el laboratorio no poseía un control continuo de la velocidad del motor, y el mismo se realizaba de a saltos mediante el uso de un reóstato de 8 posiciones que variaba la resistencia rotórica (Figura 5.23).



Figura 5.22: Compuertas utilizadas como perturbaciones



Figura 5.23: Fotos del reóstato (izquierda) y del banco de resistencias (derecha)

Para realizar estas pruebas de funcionamiento primero se encendió el túnel, se comenzó a guardar valores y se controló la velocidad del aire en 6 m/s , luego de unos segundos de funcionamiento se procedió a abrir las compuertas con el panel que se ve del lado derecho de la figura 5.22. Esta entrada de aire producía variaciones en la velocidad que el control implementado corregía según el PID utilizado. Se realizó pruebas donde se abrieron y cerraron las compuertas para observar estos cambios y el correcto control de velocidad (Figura 5.24 y 5.25).

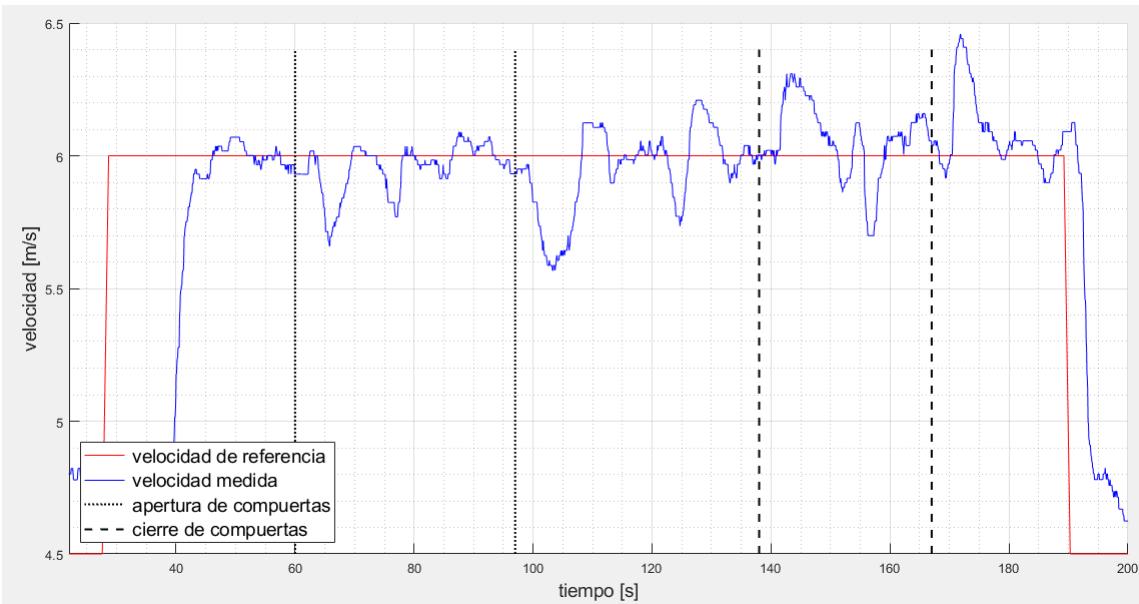


Figura 5.24: Comportamiento del sistema ante perturbaciones. Ejemplo 1.

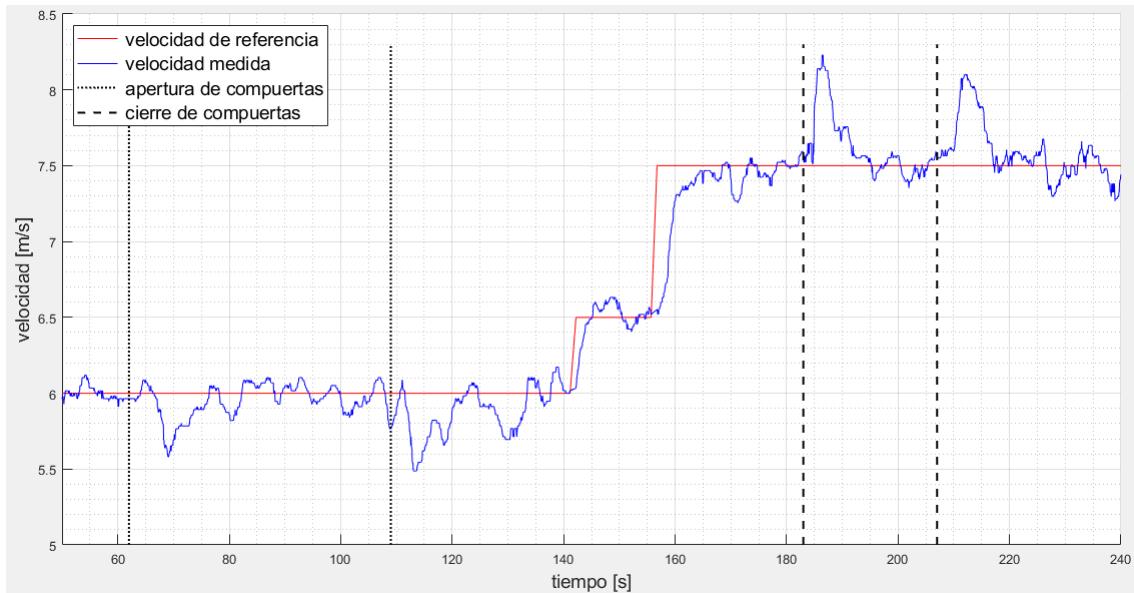


Figura 5.25: Comportamiento del sistema ante perturbaciones. Ejemplo 2.

6. Pruebas realizadas

6.1. Comparación de densidades

Una vez que se tuvo seguridad con los sensores elegidos se procedió a realizar una placa con estos elementos para que no se desconecten y produzcan errores como solía suceder mientras estaban en la protoboard (Figura 6.1).

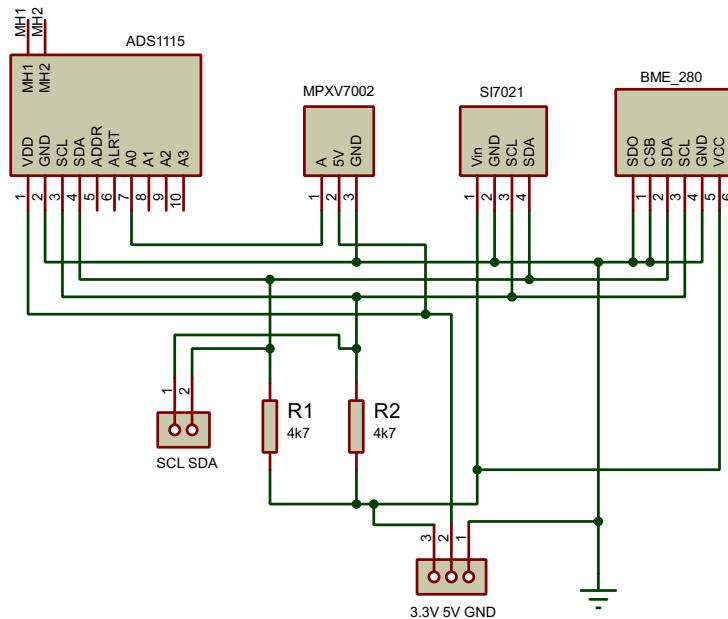


Figura 6.1: Esquema de placa con sensores utilizados

A raíz de los resultados de varias mediciones, realizadas durante distintos días, y al realizar el contraste con el instrumento **TESTO 435** se decidió retirar de la caja a los sensores de temperatura y humedad para que luego el cálculo de la velocidad del aire no esté afectado por posibles cambios de estos, ya que esta caja utilizada generaba un ambiente distinto al real dentro del laboratorio. Al colocar el sensor del lado externo, se realizaron tomas de valores en distintos momentos. Con ambos conjuntos de datos se procedió a obtener los valores de la densidad calculados con la fórmula 1. Estos datos están expresados en la tabla 6.1.

Fecha	Obs.	T [°C]		H [%]		P [Pa]		Dens. calculada [kg/m³]		$e_r [\%]$
		I	S	I	S	I	S	I	S	
29-abr	interior	19,4	19,3	43,5	38,5	100400	100450	1,1916	1,1931	0,128
14-may	interior	16,1	18,4	54,6	43,5	101590	101570	1,2195	1,2099	0,781
14-may	interior	16,5	18,9	53,7	42,5	101559	101559	1,2173	1,2077	0,794
17-jun	exterior	15,2	15,5	54,3	48,6	103100	103090	1,2418	1,2404	0,109
17-jun	exterior	13,8	15,1	59,2	52,4	102970	102910	1,2463	1,2397	0,527
07-jul	interior	17,6	18,8	43,3	34,8	100260	100261	1,1978	1,1933	0,375

Tabla 6.1: Comparación de densidades calculadas

Referencias de la tabla:

- **I:** Instrumentos- Datos obtenidos con los instrumentos del Laboratorio de fluidos.
- **S:** Sensores- Datos obtenidos a partir de la medición con los sensores utilizados en el proyecto.
- **interior-** Mientras se realizaron las mediciones el sensor SH21 se encontraba dentro de la caja donde estaba el Arduino y la placa reguladora.
- **exterior-** Las mediciones se realizaron con el sensor SH21 en el lado exterior sin que fuera afectado por el calentamiento de la placa reguladora.

Al tomar como valor verdadero los valores de THP medidos con los instrumentos calibrados, se puede observar que el error relativo del cálculo de densidad es menor al 1%.

6.2. Estimación de la densidad ante cambios de Temperatura y Humedad

Notamos conveniente realizar la comparación de las densidades calculadas ante las variaciones que se tenían de humedad y temperatura respecto a los datos tomados por los elementos calibrados. En la tabla 6.2 se observa en las celdas internas los valores calculados de densidad para humedad de 38 % y 40 % y temperatura 19°C y 21°C, estos datos fueron elegidos al observar las máximas variaciones de los datos tomados con el sensor *SH21* y el instrumento *Testo 435*, al mantener la presión atmosférica constante. (Los valores con fondo gris corresponden a la diferencia de densidades).

		Temperatura		
		19°C	21°C	
Humedad	38 %	1,1945758	1,1859383	-0,0086376
	40 %	1,1943783	1,1857162	-0,0086621
		-0,0001976	-0,0002221	

Tabla 6.2: Cálculo de la densidad ante cambios de temperatura y humedad

Si se calcula la velocidad del aire con la formula 4 en conjunto con los datos de la tabla 6.2 y se utiliza una diferencia de presión constante de 32Pa se puede observar que la diferencias de velocidades son inferiores a 0,03m/s (tabla 6.3) ante un cambio de dos grados de temperatura por lo que no se ve necesario realizar una corrección de valores.

		Temperatura		
		19°C	21°C	
Humedad	38 %	7,3195288	7,3461357	0,0266068
	40 %	7,3201342	7,3468236	0,0266894
		0,0006054	0,0006879	

Tabla 6.3: Cálculo de velocidad del aire ante cambios de temperatura y humedad

6.3. Contrastación de diferencia de presión

El sensor *MPX7002* tiene como salida una tensión proporcional a la diferencia de presión medida, por lo que fue necesario utilizar un *ADS1115* para transformar estos valores, a través de una constante, en datos de presión digitalizados. (Sección 5.1).

Para realizar un contraste del valor de presión diferencial producido por la velocidad del aire dentro del túnel, en el tubo Pitot, se procedió a realizar mediciones con el instrumento calibrado *AXD 650*. Al poseer fluctuaciones anteriormente mencionadas, debidas al flujo turbulento, y dado que el AXD650 no posee salida de datos (solo por display), se filmaron el instrumento y los datos obtenidos por nuestro sistema, para poder, con posterioridad, ver la correlación entre ambos. Los dos videos se unieron y se realizaron diversas pausas para tomar al mismo tiempo ambos valores leídos. Como resultado se obtuvo la tabla 6.4, luego se generó un gráfico de puntos con las muestras tomadas y se observó que el error es mayor para diferencias de presión mayores a 100 Pa (Figura 6.2).

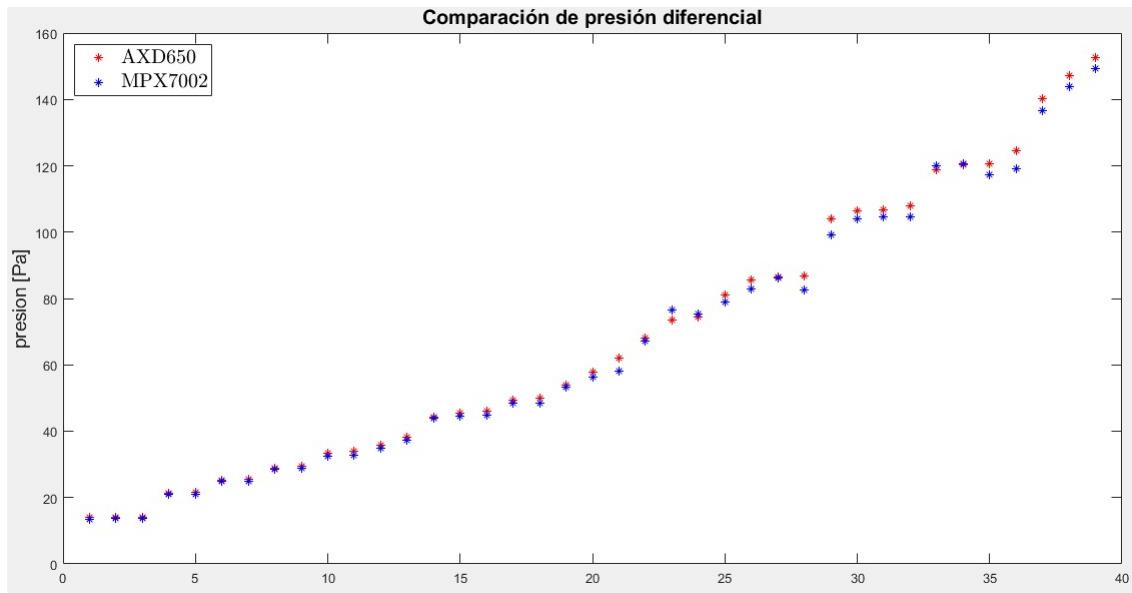


Figura 6.2: Contraste de valores de presión diferencial del MPX7002

Diferencia de presión [Pa]		
AXD650	MPX7002	E_r
14	13,51	3,50 %
14	13,76	1,71 %
14	13,76	1,71 %
21,3	20,88	1,97 %
21,5	20,88	2,88 %
25,3	24,8	1,98 %
25,4	24,8	2,36 %
28,9	28,51	1,35 %
29,3	28,8	1,71 %
33,4	32,5	2,69 %
34,1	32,88	3,58 %
35,9	34,8	3,06 %
38,1	37,38	1,89 %
44,2	43,88	0,72 %
45,5	44,5	2,20 %
45,9	44,88	2,22 %
49,4	48,51	1,80 %
49,9	48,51	2,79 %
54	53,26	1,37 %
57,7	56,2	2,60 %
62,1	58,13	6,39 %
68	67,13	1,28 %
73,4	76,6	-4,36 %
74,3	75,26	-1,29 %
81,1	79,01	2,58 %
85,5	82,76	3,20 %
86,4	86,1	0,35 %
86,8	82,63	4,80 %
104	99,26	4,56 %
106,5	104,13	2,23 %
106,8	104,5	2,15 %
107,8	104,63	2,94 %
118,8	120,13	-1,12 %
120,4	120,6	-0,17 %
120,5	117,38	2,59 %
124,4	119	4,34 %
140,1	136,6	2,50 %
147,2	143,8	2,31 %
152,7	149,38	2,17 %

Tabla 6.4: Contraste de los valores de presión diferencial.

6.4. Contrastación de velocidades

Como prueba final se decidió realizar la contrastación de la velocidad estimada contra la velocidad de otro dispositivo. Para esto, se procedió a colocar en el interior del túnel un anemómetro digital **Avm-01 - Prova** perteneciente al laboratorio. A medida que se realizaban pruebas, se realizó tanto la filmación del anemómetro como la de la pantalla para luego unir ambos videos y poder tomar valores al mismo tiempo (Figura 6.4).

Estos valores fueron volcados a la tabla 6.5 y luego graficados para observar posibles errores(Figura 6.3).

velocidad [m/s]		
estimada	anemómetro	E_r
4,6	4,7	2,13 %
4,7	4,7	0,00 %
4,8	4,7	-2,13 %
5	5	0,00 %
5,9	6	1,67 %
6	6,1	1,64 %
6,1	6,2	1,61 %
7,8	7,8	0,00 %
7,9	8	1,25 %
8,9	9,2	3,26 %
9,1	9,3	2,15 %
11,9	12,4	4,03 %
12	12,3	2,44 %
12	12,4	3,23 %
12,2	12,4	1,61 %
12,3	12,5	1,60 %
13,7	14,5	5,52 %
13,8	14,4	4,17 %
13,8	14,4	4,17 %
13,9	14,3	2,80 %
14	14,4	2,78 %
14,2	14,5	2,07 %
14,3	14,9	4,03 %
14,7	15,6	5,77 %
14,8	15,7	5,73 %
14,9	15,5	3,87 %
14,9	15,8	5,70 %
15	15,7	4,46 %
15,1	16,2	6,79 %
15,3	16,3	6,13 %

Tabla 6.5: Contraste de los valores de velocidad.

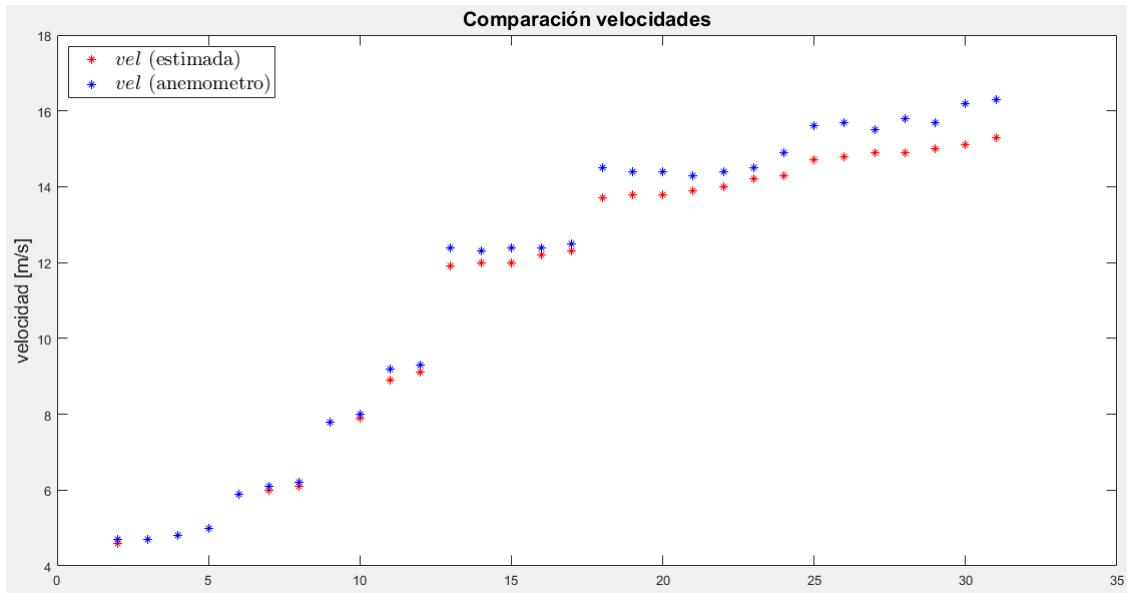


Figura 6.3: Contraste de valores de velocidad estimada

Al examinar la tabla y el gráfico se observa errores mayores a medida que las velocidades se incrementan.

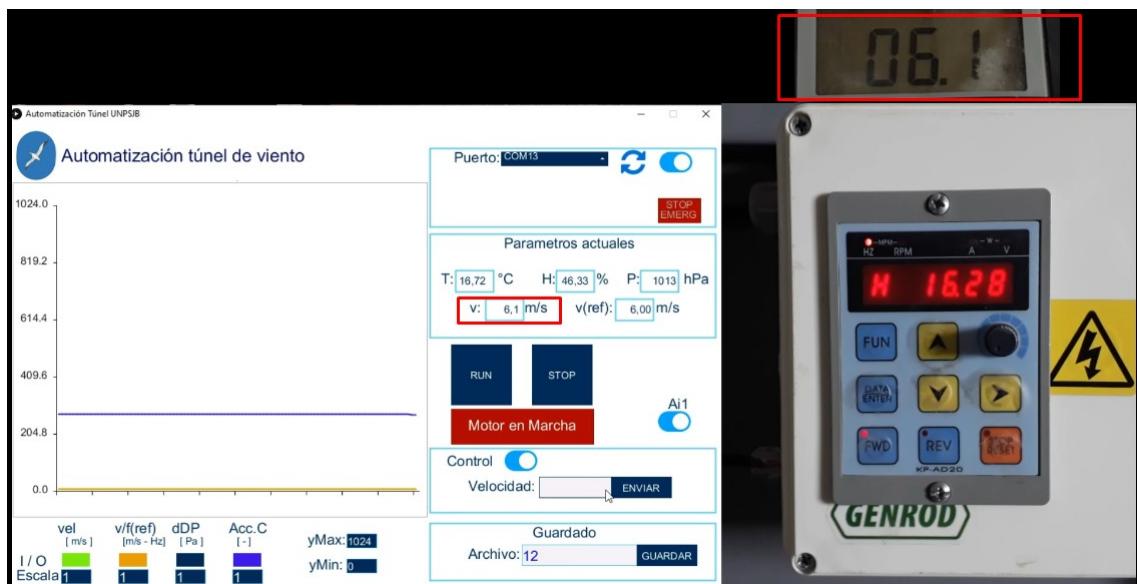


Figura 6.4: Captura de pantalla del video utilizado para la realización del contraste de velocidad

7. Implementación

Luego de realizar las pruebas y observar los cambios necesarios para un buen uso en el laboratorio, se procedió a la implementación final (Figura 7.1). La misma se dividió en dos bloques, uno perteneciente a la Aplicación (Sección 7.1), programación de la parte gráfica y adquisición de datos, y otro bloque de Hardware (Sección 7.2), parte física del proyecto.

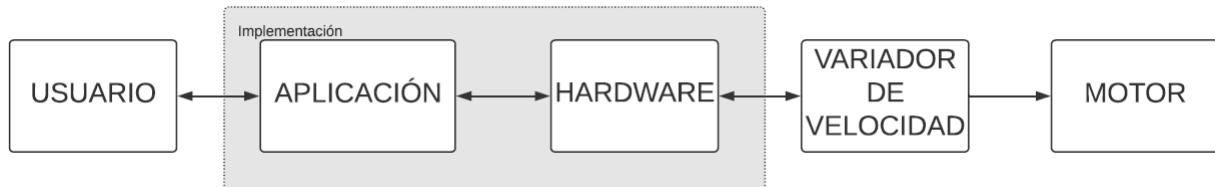


Figura 7.1: Bloques implementados

7.1. Bloque Aplicación

GUI

La interfaz gráfica de usuario utiliza un conjunto de objetos gráficos para representar información y realizar acciones. El principal uso es un entorno de comunicación entre el usuario y algún elemento a controlar.

Para la interfaz gráfica se creó un ejecutable para Windows (Figura 7.3), donde se utilizó el programa de código abierto *Processing* en conjunto con la biblioteca *ControlP5* [9]. La aplicación cuenta con comandos para configurar el funcionamiento del variador, configurar velocidades, realizar adquisición de datos y observar parámetros en tiempo real. En el diagrama de flujo de la Figura 7.2 se observa que “aplicación” se divide en dos partes:

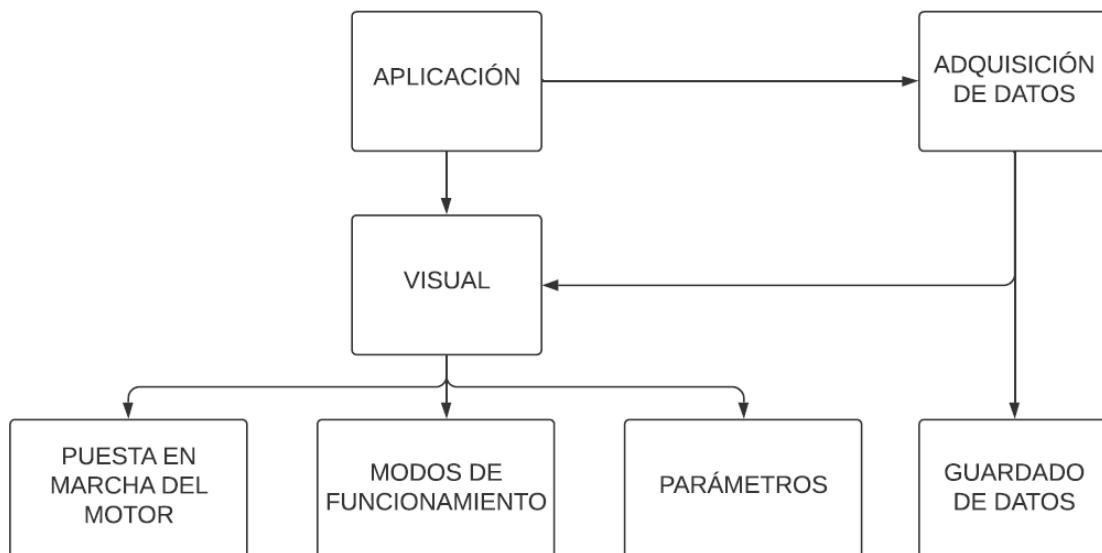


Figura 7.2: Bloque *Aplicación*

1. Bloque “Adquisición de datos”:

Obtiene datos provenientes del microcontrolador (Tabla 7.1) que se guardan en un archivo “.csv”.

Los datos que se reciben desde el microcontrolador durante el proceso de adquisición de datos consta de una cadena de caracteres en formato “string” dónde los distintos parámetros son separados por comas.

D_0 ,	D_1 ,	D_2 ,	D_3 ,	D_4 ,	D_5 ,	D_6 ,	D_7 ,	D_8 ,	D_9 ,	D_{10} ,	D_{11} ,	D_{12} ,	D_{13} ,
---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	------------	------------	------------	------------

Tabla 7.1: Datos adquiridos desde el μC

Referencia de cada dato:

- D_0 : velocidad del aire estimada.
- D_1 : valor de frecuencia o velocidad preestablecida.
- D_2 : diferencia de presión medida a través del MPX7002 y digitalizada por el ADS1115.
- D_3 : señal de acción de control.
- D_4 : tiempo otorgado por el microcontrolador.
- D_5 : temperatura ambiente.
- D_6 : humedad relativa del ambiente.
- D_7 : presión atmosférica.
- D_8 : densidad estimada por el microcontrolador.
- D_9 : señal de que el lazo de control está cerrado.
- D_{10} : error entre la velocidad estimada y la de referencia.
- D_{11} : estado encendido/apagado del motor.
- D_{12} : indicación de algún error del variador de velocidad.
- D_{13} : indicación de comienzo y fin del control automático.

2. Bloque Visual:

- Sub-bloque **Parámetros**: con los datos adquiridos, muestra los valores de THP, velocidad, v/f de referencia y realiza gráficos en tiempo real donde el usuario decide la visibilidad de distintos parámetros y su correspondiente escala.
- Sub-bloque **Puesta en marcha del motor**: En el sub-bloque se encuentran botones de RUN y STOP que pondrán en marcha y pararán el motor según la configuración de F7 del variador de velocidad. En este bloque también se encuentran el botón de parada de emergencia y el botón de “Reset” ante alguna falla.
- Sub-bloque **Modos de funcionamiento**: posee las siguientes formas de ejecución del μC :
 - **Modo Lectura de datos**: al realizar la conexión del puerto serial, el μC comienza a adquirir datos y muestra valores de THP, velocidad y frecuencia de referencia por defecto. El modo solo recibe datos del microcontrolador pero esta configuración espera comandos de *autofunción* o *Modo Ai1* para enviar al μC una nueva orden.
 - **Modo Autofunción**: El modo cierra el lazo de control, lee un archivo .csv donde se encuentran los escalones de velocidad y el tiempo de duración de estos. Envía al μC los datos del archivo elegido, y este manda el comando para que se implementen los escalones preestablecidos.
 - **Modo Ai1**: Este modo envía y recibe información del μC al variador, tiene dos funcionamientos

Automatización de Túnel de Viento

- Control desactivado: la planta se encuentra a lazo abierto y puede ser ingresada una frecuencia estimada.
- Control activado: este método cierra el lazo de control y durante su funcionamiento se retroalimenta velocidad por lo que ante perturbaciones el sistema hace las correcciones necesarias para obtener la velocidad de referencia configurada.

El formato de los datos enviados de la aplicación al μ C es un vector, separados por comas, en formato “string” como se observan en la tabla 7.2, y la referencia de cada elemento son las siguientes:

D_0	,	D_1	,	D_2	,	D_3	,	D_4	,	D_5	,	D_6	,	D_7	,
-------	---	-------	---	-------	---	-------	---	-------	---	-------	---	-------	---	-------	---

Tabla 7.2: Datos enviados al μ C

- D_0 : señal para que se encienda o se apague el motor desde la aplicación.
- D_1 : valor de velocidad o frecuencia de referencia.
- D_2 : señal para que el microcontrolador ingrese al modo “Ai1- Control activado”
- D_3 : señal para que el microcontrolador ingrese al modo “Ai1”
- D_4 : señal para que el microcontrolador se reinicie luego de alguna falla.
- D_5 : señal de falla externa que detiene el motor.
- D_6 : largo del vector ingresado por el archivo .csv.
- D_7 : vector de velocidades de referencias y tiempos de los escalones obtenidos del .csv con longitud enviado por el valor D_6 .



Figura 7.3: Pantalla de la aplicación desarrollada.

7.2. Bloque Hardware

Para llevar a cabo la implementación final del “hardware”, se procedió a realizar mediante el programa “Proteus” el diseño de una única placa (Figura 7.4) con los bloques utilizados en la sección 5. Esta nueva placa fue realizada por la máquina de prototipado **LPKF Protomat S63** perteneciente al Departamento de Electrónica de la Universidad. Una vez que se obtuvo la plaqueta se soldaron los componentes.

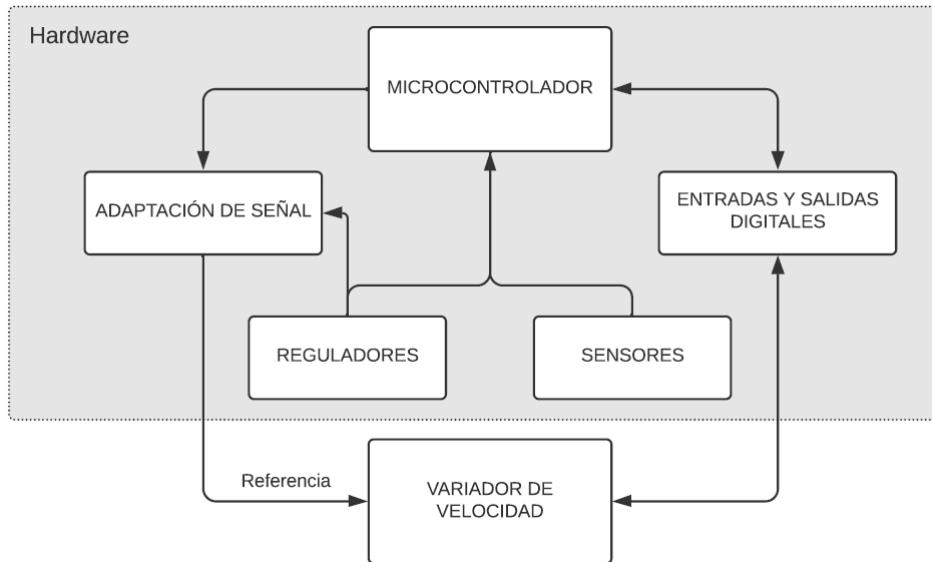


Figura 7.4: Bloque *Hardware*

En la figura 7.5 se observa la placa terminada. Este nuevo prototipo elimina ruidos eléctricos ocasionados al utilizar cables para interconectar bloques y sensores.

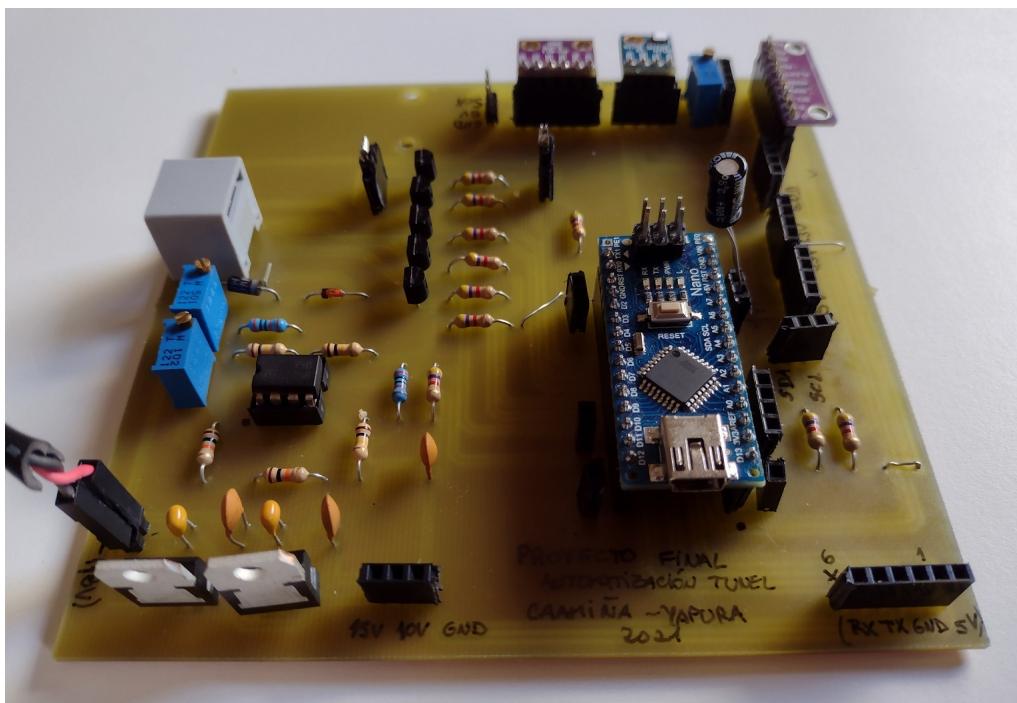


Figura 7.5: Placa construida

7.3. Costos

El proyecto se comenzó a mediados del año 2019, por lo que los sensores de temperatura, humedad, presión atmosférica y diferencia de presión fueron comprados en esa fecha en el exterior y los demás componentes, en la actualidad, en la ciudad adaptándonos al stock encontrado. En la tabla 7.3 se observan los valores totales para realizar la implementación final del *hardware* con precios actualizados al mes de agosto de 2021. *Precios obtenidos de Mercado Libre.

<i>Unidad(es)</i>	<i>Descripción</i>	<i>Precio (\$)</i>	<i>Sub-Total (\$)</i>
1	BME280*	3850	3850
1	MPXV7002dp*	8489	8490
1	SI7021*	3769	3770
10	Resistencia 4.7k	5	50
5	Resistencia 10k	5	25
5	Resistencia 100k	5	25
4	Resistencia 20k	7.5	30
2	Resistencia 22k	5	10
2	Preset 1k	180	360
1	Preset 500	180	180
1	Preset 10k	180	180
1	Diodo Zener 1N4740 (10V)	25	25
1	Diodo Zener 1N4001	15	15
2	Capacitor 0.33uF	35	70
2	Capacitor 0.1uF	15	30
1	Capacitor 220uF-16V	20	20
3	Capacitor 10nF	15	45
5	Transistor BC548	30	150
1	CI 74LS126	200	200
1	CI LM358	70	70
1	Borneras x2	180	180
1	Borneras x3	200	200
1	Fuente conmutada	1500	1500
1	Regulador 7810	180	180
1	Regulador 7815	180	180
1	ADC ADS1115	800	800
1	Pines machos	245	245
1	Pines hembra	245	245
1	Arduino NANO	900	900
1	Cable UTP x5m	300	300
1	Ficha RJ45	50	50
	Caja contenedora		
Total=			\$ 22375

Tabla 7.3: Gastos para realizar la implementación

8. Recomendaciones futuras

- Utilización de un cable UTP mallado para mejorar interferencias que puedan producirse entre el variador y el μ C.
- Realización de una aplicación con *bluetooth* para Android.
- Automatización de las compuertas laterales utilizadas para generar perturbaciones al sistema.
- Realización de una nueva GUI según necesidades del laboratorio.
- Realización de distintas señales de control para diversos rangos de velocidades.
- Realización de una nueva parte del programa del microcontrolador para que puedan ser cargadas otros tipos de perfiles de velocidad (rampas, funciones cuadráticas, etc)
- Adquisición de un nuevo motor con tecnología actual y lograr bajar el tiempo de los parámetros de aceleración y desaceleración (F35 y F36).

9. Conclusión

Siempre que se habla de control de procesos industriales, los PLC son los dispositivos más adecuados para el desarrollo de sistemas de este tipo. Para nosotros fue un desafío utilizar un microcontrolador (μ C) para llevar a cabo el control de velocidad del túnel de nuestra Universidad.

Al utilizar un μ C no se tiene la robustez que un dispositivo industrial posee, pero sin embargo, logramos una comunicación estable y un control eficaz sobre el variador de velocidad. Además, al disponer de esta opción, los costos de diseño e implementación son más económicos contra el empleo de dispositivos industriales.

Si bien es cierto que teníamos conocimientos básicos en el uso de herramientas de programación, el proyecto de automatización del túnel fue un aprendizaje que sumó a nuestros conocimientos nuevos lenguajes de programación, técnicas de trabajo y adicionalmente, saberes sobre el funcionamiento de los dispositivos que realizan las acciones sobre el proceso: motores, variadores de velocidad, señales analógicas y digitales, etc.

La realización de una aplicación que permite la adquisición de datos, la visualización de los mismos en tiempo real, el guardado de éstos en tablas, y la realización de gráficos de los datos guardados, hacen al sistema flexible tanto para la realización de los ensayos en el túnel de viento, como para el análisis posterior de los datos medidos. Además, se posibilitó el manejo remoto del variador de velocidad, y un control de lazo cerrado, en reemplazo de la utilización del panel frontal del variador, y del control de lazo abierto.

Como resultado final, se obtuvo un control de velocidad, en donde las mediciones obtenidas fueron cercanas a los valores estimados por el Laboratorio Mecánica de Fluidos, lo que ayudaría a minimizar la cantidad de procesos para la determinación de velocidad durante la contrastación de anemómetros. Además, nuestro controlador permitiría realizar nuevas experiencias de laboratorio de asignaturas afines a la temática, ya que se podrían generar cambios de velocidad controlados, perfiles de viento determinados, simular ráfagas, entre otras aplicaciones.

10. Bibliografía

Referencias

- [1] *Página del Laboratorio de Mecánica de Fluidos - UNPSJB.* URL: <http://www.ing.unp.edu.ar/mecanica/Paginas/Tunel.htm>.
- [2] Daniel Barila y Daniel Zucas. “Desarrollo y construcción de un túnel de viento de 30kW”. En: *Proyecto de investigación, Consejo de investigación CIUNPAT, Universidad Nacional de la Patagonia San Juan Bosco* (1993).
- [3] Long Shenq. *Manual de instalación y programación : Variador de velocidad LS650 para motor de C.A.*
- [4] INTI. “Cálculo de la densidad del aire húmedo”. En: *Procedimientos del INTI* (2015).
- [5] Yunus A Cengel. *Mecánica de fluidos: Fundamentos y aplicaciones.* (2006).
- [6] Federico Licci y Lucas Rivera. “Estudio fluidodinámico de túnel de viento en condiciones reales de uso y validación experimental”. En: *Proyecto final de Ingeniería Mecánica, Departamento de Mecánica, Universidad Nacional de la Patagonia San Juan Bosco* (2020).
- [7] Processing. URL: <https://processing.org>.
- [8] Jorge Pomares y Ángel Martínez Bueno. “Sistemas de Control Automático. Identificación experimental de sistemas”. En: *Sistemas de Control Automático* (2011).
- [9] Andreas Schlegel. *Biblioteca ControlP5.* URL: <http://www.sojamo.de/libraries/controlP5/>.

Anexos

A. Descargas

El siguiente proyecto con sus respectivos archivos se encuentra para realizar la descarga en el repositorio GITHUB. poner la tarjeta como imagen https://opengraph.githubassets.com/1/UNPSJB-YC/Automatizacion_Tunel_UNPSJB

https://github.com/UNPSJB-YC/Automatizacion_Tunel_UNPSJB

B. Manual de usuario de la aplicación

B.1. Requerimientos del sistema

- Windows
- Driver de comunicación serial para CH340

B.2. Puesta en marcha

1. Alimente variador de velocidad y el motor.
2. Enchufe a 220V el cable de alimentación que sale de la caja de control.
3. Conecte el cable USB.
4. Ejecute el programa *Gui_Tunel.exe* que se encuentra dentro de la carpeta *application.windows64* dentro de *Aplicación*
5. Seleccione el puerto dónde se encuentra conectado el microcontrolador.
6. Active el puerto.
7. Al inicializar el puerto, el microcontrolador comienza a entregar valores a la aplicación por lo que se observa THP, velocidad estimada y frecuencia de referencia.



Figura B.1: Puerto activado. Visualización de los valores del μ C.

B.3. Gráfico en tiempo real

B.3.1. Visibilidad de las señales en tiempo real

En la parte izquierda inferior del programa, se puede activar o desactivar la visibilidad de las variables a observar, solo basta realizar un click en los rectángulos, y estos al estar activados se colocarán del color correspondiente a la señal.

B.3.2. Escala

En la parte inferior se encontrará “Escala” donde se establece la escala correspondiente de cada señal.

1. Realice click en el interior del rectángulo
2. Borre el número que posee y coloque el valor nuevo a ingresar.
3. Presione “enter”.
4. La escala de la señal elegida será modificada.

Ejemplo de escala:

Si se coloca en una señal “escala: 10”, la variable observada en tiempo real estará aumentada 10 veces.

B.3.3. Límite del eje vertical

El eje vertical del gráfico en tiempo real puede ser modificado según necesidad.

1. Realice click en el interior del rectángulo.
2. Borre el número que posee y coloque el valor nuevo a ingresar.
3. Presione “enter”.
4. El eje “vertical” se verá modificado.



Figura B.2: Gráfico en tiempo real

B.4. AutoFucion

El modo de auto-función carga un archivo “.csv” que se encuentra dentro de la carpeta “autofun” (Figura B.3). El archivo “.csv” posee los datos pertenecientes a los escalones de velocidad deseados y el tiempo de ejecución de cada uno (Figura B.4).

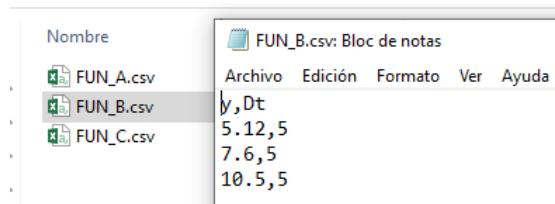


Figura B.3: Archivos dentro de la carpeta "autofun"

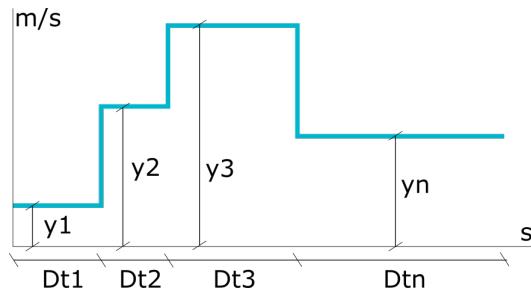


Figura B.4: Diagrama para el formato de autofunción

El archivo debe ser generado por el usuario en un bloc de notas, guardado como formato ".csv" tener el formato observado en la Tabla B.1, donde se ve que posee un encabezado el cual debe incluirse y las columnas son separadas por *comas*. La primer columna, valores de velocidad en *m/s*, deben ser números comprendidos entre 4.7 y 18, los números decimales deben ser separados de la parte entera por medio de un punto. La segunda columna establece la cantidad de segundos de cada escalón preestablecido.

y,Dt
y1,Dt1
y2,Dt2
y3,Dt3
...yn,Dtn

Tabla B.1: Formato para la confección del archivo .csv

Una vez que se eligió el archivo deseado del menú desplegable, se debe presionar el botón *.Abrir* aparecerá una señal visual debajo del mismo dando a conocer la implementación en el variador de dicha función (Figura B.5), el motor se pondrá en marcha y se apagará al finalizar la *autofunción*.

B.5. A1

B.5.1. Control desactivado

Al utilizar este modo de funcionamiento, se puede ingresar el valor deseado de frecuencia de salida del variador. El lazo de control estará abierto.

B.5.2. Control activado

El lazo de control estará activado, ante perturbaciones el sistema tiende a establecerse a la velocidad estipulada. Para determinar la velocidad de referencia se debe presionar sobre la caja de “velocidad”, ingresar un valor y presionar sobre el botón “enviar”.



Figura B.5: Aplicación en modo Autofunción

B.6. Encendido/ apagado del motor

El encendido/ apagado del motor se puede realizar de dos formas:

B.6.1. Panel frontal

Si el parámetro F7 del variado de velocidad se encuentra en “0”, la marcha y parada del motor será realizada por los botones físicos en el panel frontal del variador.

B.6.2. Aplicación

Si el parámetro F7 del variador se encuentra en “1”, la marcha y parada del motor será realizada por los botones propios de la aplicación.

B.7. Guardado de datos

Una vez que se establece la comunicación, el programa realiza la captura de datos, al colocar un nombre y presionar el botón guardar, se generará un archivo “.csv” con los datos que el microcontrolador capturó cada aproximadamente 55ms hasta ese momento. Estos archivos se guardarán en la carpeta "data1" dentro de la carpeta general de la aplicación.

Alerta: Una vez que se presiona el botón guardar o el puerto se cierra los datos son borrados y se comenzará una nueva tabla.



Figura B.6: Guardado de datos

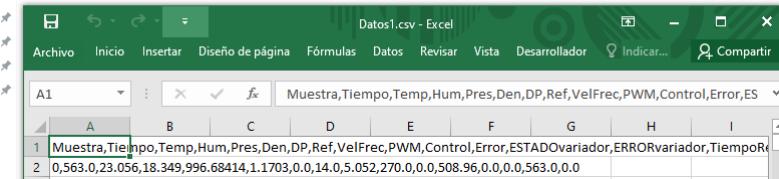
B.8. Lectura de datos obtenidos

El encabezado de la tabla (Figura B.7) muestra las siguientes leyendas:

- *Muestra*: número de muestra tomada
- *Tiempo*: tiempo desde el inicio del puerto serie [ms].
- *Temp*: temperatura ambiente [° C].
- *Hum*: humedad relativa del ambiente [%].
- *Pres*: presión atmosférica [hPa].
- *Den*: densidad estimada por el microcontrolador [kg/m^3].
- *DP*: diferencia de presión medida a través del MPX7002 en conjunto con ADS1115 [Pa]
- *Ref*: Valor de frecuencia o velocidad preestablecida [Hz ó m/s].
- *VelFrec*: Velocidad del aire estimada [m/s].
- *PWM*: señal de acción de control.
- *Control*: señal de que el lazo de control está cerrado.
- *Error*: error entre la velocidad estimada y la de referencia.
- *ESTADOvariador*: estado encendido/apagado del motor.
- *ERRORvariador*: indicación de algún error del variador de velocidad.
- *TiempoRel*: indica tiempo desde el ultimo guardado de datos.
- *ControlAutomatico*: indicación de comienzo y fin del control automático.

Para hacer lectura o interpretación de los datos se recomienda utilizar el script de Matlab *captura_datos.m* ubicado dentro de la carpeta donde fue guardado el archivo.

Nombre	Fecha de modificación	Tipo	Tamaño
Datos1.csv	19/08/2021 11:02	Archivo de valores...	112 KB



A	B	C	D	E	F	G	H	I
1	Muestra,Tiempo,Temp,Hum,Pres,Den,DP,Ref,VelFrec,PWM,Control,Error,ESTADOVariador,ERRORvvariador,TiempoR							
2	0,563,0,23,056,18,349,996,68414,1,1703,0,0,14,0,5,052,270,0,0,0,508,96,0,0,0,0,563,0,0							

Figura B.7: Archivo .csv generado

B.9. Falla externa

Al presionar el botón de “Falla externa” el μ C enviará una señal al variador y el sistema se detendrá. Luego, se podrá restablecer al presionar el botón de *reset*.

C. Esquemático de la placa implementada

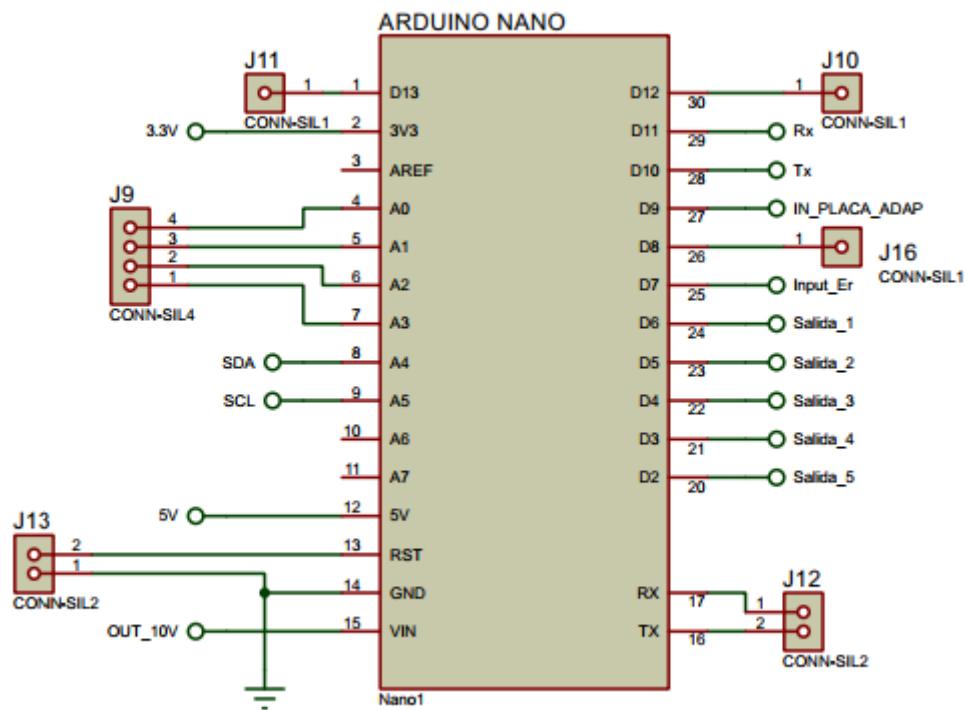


Figura C.1: Esquema del microcontrolador

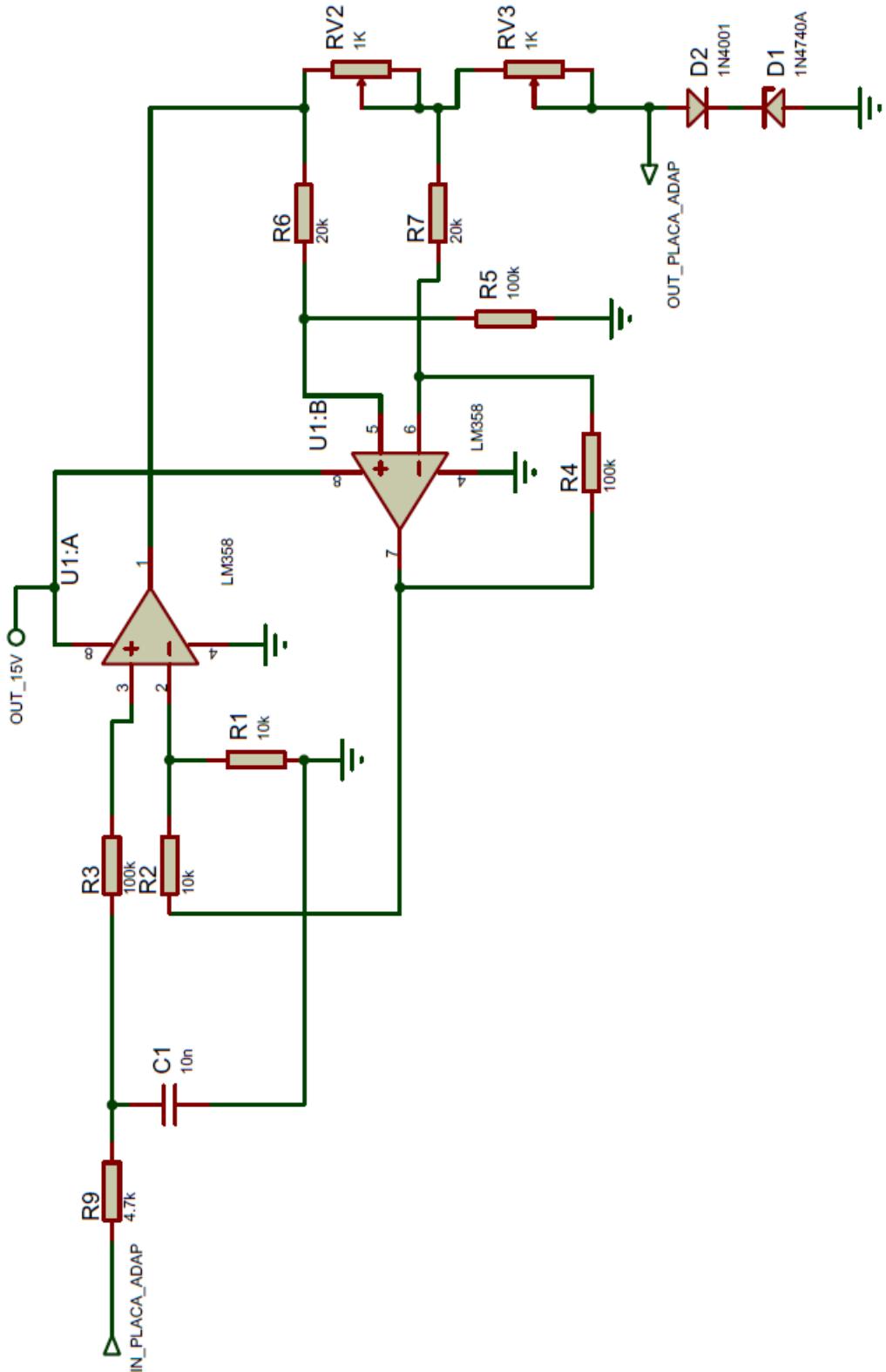


Figura C.2: Esquema de la placa adaptadora de la señal del sensor de presión diferencial.

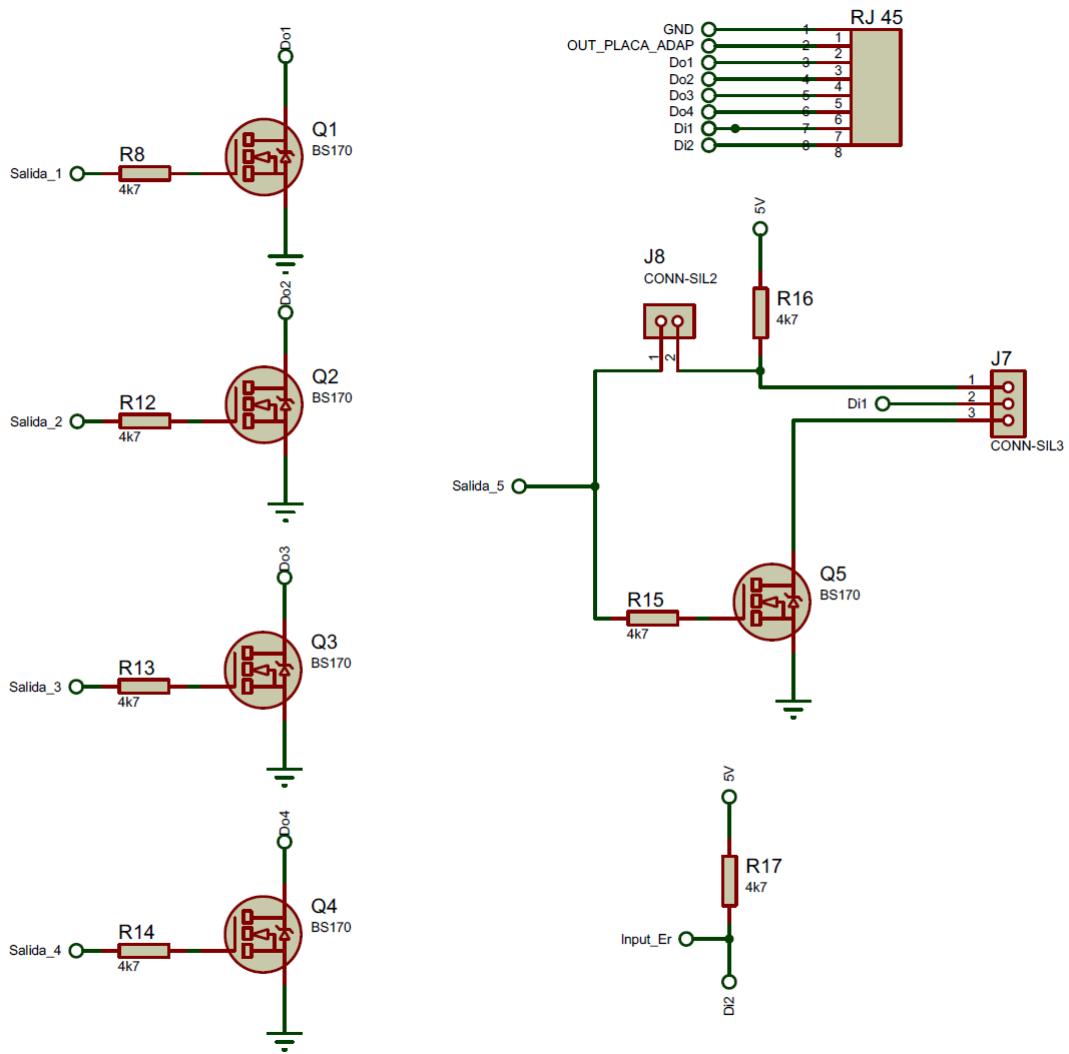


Figura C.3: Esquema de entradas y salidas del microcontrolador hacia el variador.

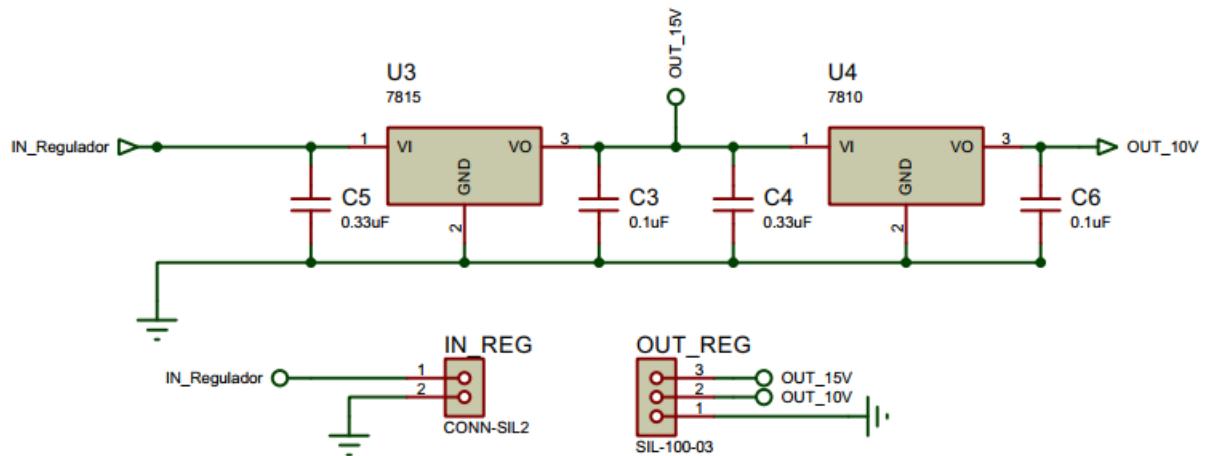


Figura C.4: Esquema de la placa reguladora de tensión.

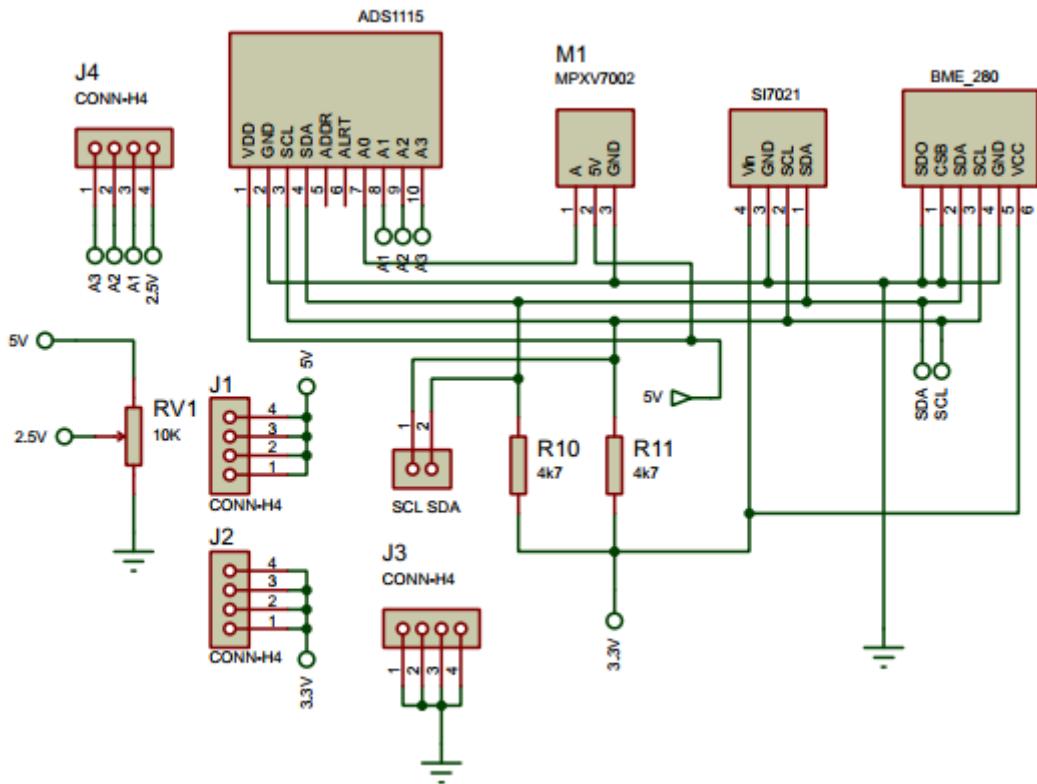


Figura C.5: Esquema de sensores

D. Código Arduino

```

1 //Bibliotecas
#include <Arduino.h>
3 #include <Wire.h>
#include <SoftwareSerial.h>
5 #include "Filter.h" //Filtro
#include "MegunoLink.h"
7 #include "MedianFilterLib.h" //Filtro de Mediana
#include "MeanFilterLib.h" //Filtro de Media
9 #include <TimerOne.h> //Libreria PWM
#include "Adafruit_Si7021.h"
11 #include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h> //sensor THP
13 #include <Adafruit_ADS1015.h> //Convertidor analogico digital
#include <Custom_PID_Tunel.h> //Libreria modificada por Cristian Yapura
15
Adafruit_Si7021 sensor = Adafruit_Si7021(); //DHT21
17 Adafruit_BME280 bme; // //BME280
Adafruit_ADS1115 ads; //ADS115
19
//Variables utilizadas en los filtros
21 MeanFilter<float> meanFilter(20);
MedianFilter<float> medianFilter(40);
23
//Inicializacion de variables
25 #define Fallaext 5
float t1 = 0, h1 = 0, p2 = 0; //variables utilizadas en THP
27 float presionSF2 = 0, presionSF21, ADCfilterM = 0, ADCFilterM1; //Variab
float velocidadA = 0, velocidadB = 0; //Variables velocidad
29 float adc0 = 0, offsetp2 = 0, offsetdf = 0, offsetdf1 = 2.592;
float entrada1[30], entrada2[30];
31 float Inref1, In = 0;
float VelRef = 0, VelRef1 = 0; //PID
33 int entrada[7], Inref, len, inc = 0, incl = 1, conserie = 0, contador = 0;
long tiemporead = 0, vtiempoant, tiempoautomatico;
35 long time1, time2, time3, tiemporetardo;
double output = 0; //PID
37 boolean paro, BOT = 0, BOT2 = 0, step1 = 0, Estado = 0, Errorvar = 0;
boolean pulsador, Estado1 = 0, Errorvar1 = 0, Estadoant = 0, Errorvarant = 0;
39 boolean EnableA1 = 0, RUNSTOP = 0, Control = 0, FallaExterna = 0;
boolean Resetfalla = 0, Encendido;
41 bool ControlAutomatico = 0, cambio = 0, cambiol = 0, terminoautoma = 0;
String data; //Guardo los datos del buffer para utilizar en funcion princ.
43
//Constantes para calculo de densidad/
45 float a0 = 0.00000158123, a1 = -0.000000029331, a2 = 0.00000000011043;
float b0 = 0.000005707, b1 = -0.00000002051;
47 float c0 = 0.00019898, c1 = -0.000002376;
float d = 0.0000000000183, e = -0.000000000765;
49 float A = 0.000012378847, B = -0.019121316;
float C = 33.93711047, D = -6343.1645, alfa = 1.00062;
51 float beta = 0.0000000314, gamma = 0.00000056;
float T1; //temperatura en absoluto
53 float fpt, psv, xv, Z, den;

55 //PID
PID pid(0.6846, 0.4183, 0);
57 float Error1 = 0;
long tiempo = 0;
59 int pw = 0;
float Cte = 56.88;
61
void setup() {
63   Serial.begin(115200);
   if (!sensor.begin()) { //SHT21
     Serial.println("Problema_sensor_-_Si7021");
     while (true);
67   }
   if (!bme.begin(0x76)) { //BME280

```

```

69     Serial.println("Problema_sensor_BME280");
70     while (1);
71 }
72 ads.setGain(GAIN_ONE); // +/- 4.096V 1bit=2mV 0.125mV
73 ads.begin(); //ADS1115
74 /////Inicilizo Entradas/Salidas
75 pinMode(2, INPUT_PULLUP);
76 pinMode(3, OUTPUT);
77 pinMode(4, OUTPUT);
78 pinMode(Fallaext, OUTPUT);
79 pinMode(6, OUTPUT);
80 pinMode(7, INPUT_PULLUP);
81 pinMode(13, OUTPUT);
82 /////Inicilizo pwm//
83 Timer1.initialize(90);
84 }
85
86 void loop() {
87 THP();
88 difPresion();
89 vel_tiempo();
90 if (terminoautoma == 1 & (millis() -tiempoautomatico)>=3500) {
91 cambioAutomatico();
92 }
93 PIDS();
94 entradas();
95 salidas();
96 pulsador = digitalRead(8);
97 imprimir_datos();
98 }
99

100 void serialEvent() {
101     if (Serial.available() > 0) { // Lectura del puerto mientras sigue abierto
102         data = Serial.readStringUntil('\n'); //lectura hasta \n
103         int n, n1 = 0, n2 = 0; // Variables para algoritmo de lectura
104
105         for (int i = 0; i <= data.length(); i++) { // Lectura total
106             if (data.substring(i, i + 1) == ",") { // Lectura hasta ","
107                 if (n1 == 0) {
108                     entrada[n1] = data.substring(0, i).toInt();
109                     n1 = n1 + 1;
110                     n = i + 1;
111                 }
112                 else {
113                     entrada[n1] = data.substring(n, i).toInt();
114                     n1 = n1 + 1;
115                     n = i + 1; // Posicion de la letra final leida + 1
116                 }
117             }
118         }
119         n = 0;
120         ////////////////////STRING A FLOAT/////////////////
121         for (int i = 0; i <= data.length(); i++) { // Lectura
122             if (data.substring(i, i + 1) == ",") { // Lectura hasta ","
123                 if (n2 == 0) {
124                     entrada2[n2] = data.substring(0, i).toFloat();
125                     n2 = n2 + 1;
126                     n = i + 1;
127                 }
128                 else {
129                     entrada2[n2] = data.substring(n, i).toFloat();
130                     n2 = n2 + 1;
131                     n = i + 1; // Posicion de la letra final leida + 1
132                 }
133             }
134         }
135         if (entrada[6] != 0) {
136             int j = 0;
137             for (int i = 7; i <= data.length(); i++) {

```

```

139         entradad1[j] = entradad2[i];
140         j++;
141     }
142     tiempoautomatico = millis();
143     cambio = 1;
144     terminoautoma = 1;
145     len = entrada[6];
146     ControlAutomatico = 1;
147 } else {
148     terminoautoma = 0;
149     cambio = 0;
150     ControlAutomatico = 0;
151 }
153 }
155 /*          LECTURA THP, ESTIMACION DE VELOCIDAD */
157 void THP() {
158     t1 = sensor.readTemperature(); //SHT21
159     T1 = t1 + 273.15; //T abs
160     h1 = sensor.readHumidity(); //SHT21
161     p2 = bme.readPressure(); //BME280
162 }
163 void difPresion() {
164     adc0 = ads.readADC_SingleEnded(0) * 4.096 / 32768; //ADS1115
165     presionSF2 = (adc0 - offsetdf1 - offsetp2) * 1000; //Presion
166     ADCFilterM = medianFilter.AddValue(presionSF2);
167 }
168 void calculo_offset() {
169     if (contador < 35) { //contador para descartar primeros valores
170         offsetdf = meanFilter.AddValue(adc0 - offsetdf1);
171         contador = contador + 1;
172     }
173     else {
174         if (contador == 35) {
175             offsetp2 = offsetdf;
176             contador = contador + 1;
177         }
178     }
179 }
180 }
181 }

183 void vel_tiempo() {
182     ///////////Calculo densidad///////////
183     fpt = alfa + beta * p2 + gamma * pow(17, 2);
184     psv = 1 * exp(A * (pow(T1, 2)) + B * T1 + C + D / T1);
185     xv = (h1 / 100) * fpt * psv / p2;
186     Z = 1 - (p2 / T1) * (a0+a1*t1+a2*pow(t1, 2)) + (b0 + b1 * t1) *
187     xv + (c0+c1*t1)*pow(xv, 2)) +(d+e*pow(xv, 2))*pow((p2/T1), 2);
188     den = 0.00348374 * p2 * (1 - 0.378 * xv) / (Z * T1); //densidad
189     ///////////Calculo velocidad/////////
190     if (ADCFilterM < 0) {
191         ADCFilterM1 = 0;
192     } else ADCFilterM1 = ADCFilterM;
193     if (presionSF2 < 0) {
194         presionSF21 = 0;
195     } else presionSF21 = presionSF2;
196     velocidadA = sqrt((2 * (abs(presionSF21))) / den); //sin filtro
197     velocidadB = sqrt((2 * (abs(ADCFilterM1))) / den); //con filtro
198     tiempo = millis();
199 }
200 }

203 /*          AUTOFUNCION */
204 void cambio_automatico() {

205     if (cambiol == 0) {
206         entradad[3] = 1; //Encendido = 1; Habilito Aii

```

```

209     if ((millis() - tiempoautomatico) >= 4000) {
210         entrada[0] = 1;
211     } //RUNSTOP = 1;      Doy Marcha
212     if ((millis() - tiempoautomatico) >= 9000) {
213         entrada[2] = 1;
214         cambio1 = 1;
215         inc = 0;
216         incl = 1;
217     }
218 }
219 else {
220
221     if (incl <= len - 1) {
222         if (cambio == 1) {
223             Inref1 = entradad1[inc];
224             vtiempoant = millis();
225             cambio = 0;
226         }
227         if ((millis() - vtiempoant) >= entradad1[incl] * 1000) {
228             inc = inc + 2;
229             incl = incl + 2;
230             cambio = 1;
231             tiemporetardo = millis();
232         }
233     } else {
234         entrada[0] = 0; //RUNSTOP = 0;      Apago Motor
235         entrada[2] = 0;
236         Control = 0;
237         if ((millis() - tiemporetardo) > 3000) {
238             entradad1[3] = 0; //Encendido = 0; Deshabilito Ail
239             terminoautoma = 0;
240             cambio1 = 0;
241             cambio = 0;
242             ControlAutomatico = 0;
243         }
244     }
245 }
246 }
247 /*          ENTRADAS Y SALIDAS                  */
248 void entradas() {
249     ///////////////////////////////////////////////////////////////////
250     if ((millis() - tiemporead) > 300) {
251         Estado1 = !digitalRead(2);
252         Errorvar1 = !digitalRead(7);
253         tiemporead = millis();
254         if (Estado1 == 1 && Estadoant == 1) {
255             Estado = 1;
256         }
257         if (Estado1 == 0 && Estadoant == 0) {
258             Estado = 0;
259         }
260         if (Errorvar1 == 1 && Errorvarant == 1) {
261             Errorvar = 1;
262         }
263         if (Errorvar1 == 0 && Errorvarant == 0) {
264             Errorvar = 0;
265         }
266         Estadoant = Estado1;
267         Errorvarant = Errorvar1;
268     }
269     if (Errorvar == 1) {
270         entrada[0] = 0; //RUNSTOP = 0;      Apago Motor
271         entrada[3] = 0; //Encendido = 0; Deshabilito Ail
272         entrada[2] = 0;
273         terminoautoma = 0;
274         cambio = 0;
275         ControlAutomatico = 0;
276     }
277 ///////////////////////////////////////////////////////////////////

```

```

279 RUNSTOP = entrada[0];      //Variable Marcha- Parada
280 Inref = entrada[1];        //velref o freq aprox.
281 Control = entrada[2];     //Control activado/desactivado
282 Encendido = entrada[3];   //ON - OFF . Habilitacion Ail
283 Resetfalla = entrada[4];  //Reset falla
284 FallaExterna = entrada[Fallaext]; //Falla externa
285 }

287 void salidas() {
288     digitalWrite(3,RUNSTOP); //senal de on off
289     digitalWrite(4,Encendido); //Habilito control por Ail
290     digitalWrite(Fallaext,FallaExterna); //senal que ocurrio una falla
291     digitalWrite(6,Resetfalla); //senal para reset de fallas
292 }
293

295 /*          ESCALONES          */
296

297 void escalon() {
298     if (tiempo >= 20000 & tiempo <= 40000 & pw <= 250) {
299         pw = pw + 2;
300     }
301     if (tiempo >= 40000 & tiempo <= 50000) { //13+10hz
302         pw = 250;
303     }
304     if (tiempo >= 50000) { //13+20hz
305         pw = 500;
306     }
307     if (tiempo >= 60000) { //13+24hz
308         pw = 600;
309     }
310     if (tiempo >= 70000) { //13+22hz
311         pw = 550;
312     }
313     if (tiempo >= 80000) { //13+18hz
314         pw = 450;
315     }
316     if (tiempo >= 90000) { //13+14hz
317         pw = 350;
318     }
319     if (tiempo >= 100000) { //13hz
320         pw = 0;
321     }
322     Timer1.pwm(9, pw);
323 }

325 void escalon2() {
326     pw = analogRead(A0);
327     Timer1.pwm(9, pw);
328 }
329

330 void escalon22() {
331     pw = analogRead(A0);
332     pw = map(pw, 0, 1023, 50, 170);
333     Timer1.pwm(9, pw);
334 }
335

336 void escalon3() {
337     if (tiempo >= 20000 & tiempo <= 33000 ) {
338         //pw=220;    //// supongo 17,33 Hz
339         pw = 235;   //// supongo 17,33 Hz
340     }
341     if (tiempo >= 33000 & tiempo <= 45000) { //
342         //pw=270;    ////Supongo 19.14 HZ
343         pw = 392;   ////Supongo 25 HZ
344     }
345     if (tiempo >= 45000 & tiempo <= 55000) { //
346         //pw=220;    ////Supongo 17.33 Hz
347         pw = 235;   ////Supongo 17.33 Hz
348     }
349 }
```

```

349     if (tiempo >= 55000) { //  
350         pw = 0;  
351     }  
352     Timer1.pwm(9, pw);  
353 }  
  
355 void escalon4() {  
356     if (tiempo >= 20000 & tiempo < 35000 ) {  
357         //pw=325;      //// supongo 21.05Hz  
358         pw = 193;      //// supongo 21.05Hz  
359     }  
360     if (tiempo >= 35000 & tiempo < 55000) { //  
361         //pw=595;      ////Supongo 32.73 HZ  
362         pw = 314;      //// supongo 21.05Hz  
363     }  
364     if (tiempo >= 55000 & tiempo < 70000) { //  
365         //pw=325;      ////Supongo 21.05 Hz  
366         pw = 434;      //// supongo 21.05Hz  
367     }  
368     if (tiempo >= 70000 & tiempo < 85000) { //  
369         //pw=325;      ////Supongo 21.05 Hz  
370         pw = 531;      //// supongo 21.05Hz  
371     }  
372     if (tiempo >= 85000 & tiempo < 105000) { //  
373         //pw=325;      ////Supongo 21.05 Hz  
374         pw = 434;      //// supongo 21.05Hz  
375     }  
376     if (tiempo >= 105000 & tiempo < 120000) { //  
377         //pw=325;      ////Supongo 21.05 Hz  
378         pw = 314;      //// supongo 21.05Hz  
379     }  
380     if (tiempo >= 120000 & tiempo < 140000) { //  
381         //pw=325;      ////Supongo 21.05 Hz  
382         pw = 193;      //// supongo 21.05Hz  
383     }  
384     if (tiempo >= 140000) { //  
385         pw = 0;  
386     }  
387     Timer1.pwm(9, pw);  
388 }  
389 void escalonserial() {  
390     In = entrada[2];  
391     //Timer1.pwm(9, pw);  
392 }  
  
395 void escalonautoma() {  
396 }  
397  
398 /*  
399      PID  
400 */  
401 void PIDS() {  
402     if (terminoautoma == 0) {  
403         In = Inref;  
404     } else In = Inref1;  
405     if (Control == 1) {  
406         if (terminoautoma == 1) {  
407             VelRef = In;  
408         } else {  
409             In = map(In, 0, 32767, 9059, 32767);  
410             VelRef = In / 1927.47;  
411         }  
412         if (VelRef > 17.5) {  
413             VelRef = 17;  
414         }  
415         if (VelRef < 4.7) {  
416             VelRef = 4.7;  
417         }  
418     }  
419     else if (Control == 0) {  
420

```

```

419     VelRef1 = map(In, 0, 32767, 9175, 32767); ///9175
420     VelRef1 = VelRef1 / 655.3;
421     In = map(In, 0, 32767, 7300, 32767); ///9175
422     VelRef = In / 655.34;
423     if (VelRef >= 50) {
424         VelRef = 50;
425     }
426     if (VelRef <= 11.14) {
427         VelRef = 11.14;
428     }
429     step1 = 0;
430 } else VelRef = 0;
431 Error1 = Cte * (VelRef - velocidadB);
432 if (Control == 1) {
433     if (step1 == 0) {
434         pid.Initialize(output);
435         step1 = 1;
436     }
437     output = pid.Update(Error1, 55); //Calcula PID c/Error
438 }
439 else {
440     output = VelRef * 19.8; //Out determinada por la frec_aprox
441     step1 = 0;
442 }
443 pw = output;
444 Timer1.pwm(9, pw);
445 }

447 /*           IMPRIMIR DATOS */
448 void imprimir_datos() {
449     Serial.print(velocidadB, 3); Serial.print(";");
450     if (Control == 1) {
451         Serial.print(VelRef); Serial.print(";");
452     }
453     else {
454         Serial.print(VelRef1); Serial.print(";");
455     }
456     Serial.print(ADCFilterM, 3); Serial.print(";");
457     Serial.print(pw); Serial.print(";");
458     Serial.print(tiempo); Serial.print(";");
459     Serial.print(tl, 3); Serial.print(";");
460     Serial.print(h1, 3); Serial.print(";");
461     Serial.print(p2, 3); Serial.print(";");
462     Serial.print(den, 4); Serial.print(";");
463     Serial.print(Control); Serial.print(";");
464     Serial.print(Error1); Serial.print(";");
465     Serial.print(Estado); Serial.print(";");
466     Serial.print(Errorvar); Serial.print(";");
467     Serial.print(ControlAutomatico); Serial.println(";");
468     digitalWrite(13, Control);
469 }

```

E. Código Processing

```

1  /*
2   * =====
3   * GUI realizada para proyecto final
4   * "Automatizacion del tunel de viento UNPSJB"
5   * de la carrera de ingenieria electronica de la
6   * Universidad Nacional de la Patagonia San Juan Bosco (UNPSJB.)
7   *
8   * Autores:
9   * Caamina Daniela
10  * Yapura Cristian
11  * Repositorio: https://github.com/UNPSJB-YC/Automatizacion\_Tunel\_UNPSJB
12  */
13
14 // BIBLIOTECAS
15 import controlP5.*;
16 import grafica.*;
17 import processing.serial.*;
18 import java.util.Random;
19 import java.util.*;
20 import java.awt.Frame; //gui
21 import java.awt.BorderLayout; //gui
22
23 CColor ccS = new CColor((#AA0000), (#AA0000), (#AAAAAA),
24                         (#AAAAAA), (#000000));
25 CColor cc = new CColor((#5DD2EA), color(248, 240, 248),
26                         (#002D5A), color(248, 240, 248),
27                         color(0, 0, 255));
28 CColor ccONOFF = new CColor((#5DD2EA), (#009900),
29                             (#000000), (#0000ff), (#000000));
30
31 ControlP5 cp5;
32
33 // Settings for the plotter are saved in this file
34 JSONObject plotterConfigJSON;
35
36 //grafica en tiempo real
37 Graph LineGraph = new Graph(70, 115, 500, 400, color(20, 20, 200));
38 float[][] lineGraphValues = new float[6][100];
39 float[] lineGraphSampleNumbers = new float[100];
40 color[] graphColors = new color[6]; //grafica en tiempo real
41 String topSketchPath = "";
42
43 //serial
44 Serial Arduino;
45 String puerta;
46 List l;
47 int baudrate = 115200;
48
49 //variables
50 String textV, textF; // usado para ingresar numero de vel o freq
51 String text2; // nombre del texto del archivo .csv
52 Float textVV, textFF; // usado para ingresar numero de vel o freq
53 int textVVV, textFFF; // usado para ingresar numero de vel o freq
54 int unicavez; // contadores
55 boolean unicavez2 = false, unicavez3 = false; //contadores
56 int variablecontrol1;
57 long tiempo1, tiempo2; //contadores
58 float tiempo3; //contadores
59 float[] algo;
60 Table table;
61 String val, val1;
62 byte[] inBuffer = new byte[500]; // holds serial message
63 String[] nums;
64 int tomonuestra = 0;
65 int colfun; //color para autofun
66 String myString = "";
67 float tiempo = 0, t = 0, h = 0, p = 0, d = 0, dp = 0, VelRef = 0,
68 v1 = 0, pwm = 0, controlsINO = 0, error = 0, Ev = 0,
69 ErrorVariador = 0, ControlAutomatico = 0;

```

```

    boolean ONOFF = false;
70 boolean Corre = false;
    boolean Parada = true;
72 boolean ERROR = true;
    boolean ControlONOFFnueva;
74 String fitem; //autofun
Table table1;
76 int colorONOFF;
    boolean Run = false;
78 boolean Stop = true;
    boolean DD4 = false;
80 boolean DD5 = false;
    boolean DD6 = false;
82 boolean reinicioreset = false;
    boolean HabilitacionV;
84 int contgraf = 0;
    int a = 0;
86 //inicializacion de elementos de la biblioteca controlP5.
88 Textlabel TL1, TL2, TL3, TL4, TL5, TL6, TL7, TL77, TL8, TL9,
    TL99, TL10, TL11, TL12, TL13, TL14, TL15, TL16, TL17,
90     Tx3, Tx4, TL144, TL155, TL166, TL177, TL18,
    TL19, TL20, TL21, TL22, TL23;
92 Textfield TF1, TF11, TF2, TF3, TF4, TF5, TF6, TF7, TF8;
    Button TB2, TB22, TB3, TB4, TB5, TB10;
94 controlP5.Button TB11;
    Toggle TB6, TB7, TB8, TB9;
96 ScrollableList dd1; //autofun
    Button ICO; //autofun
98 ScrollableList SL, autoLista;
    Icon ser, serono, TODOonoff, TB1, TB12, TB13;
100 //envio de datos a Arduino
    String Dato0 = "0", Dato1 = "0", Dato2 = "0", Dato3 = "0", Dato4 = "0",
102     Dato5 = "0", Dato6 = "0", Dato7 = "", DatosWrite, DatosWrite2;

104 void setup() {
106     surface.setTitle("Automatizacion_Tunel_UNPSJB");
108     size(1000, 650); // Configura resolucion interfaz
    ControlONOFFnueva = false;
110     unicavez = 0;
    variablecontrol1 = 0;
112     println(unicavez);
    cp5 = new ControlP5(this);
114     PFont font = createFont("arial", 20);
    PFont font2 = createFont("arial", 14);
116     PFont font3 = createFont("arial", 14);

118 //Serial
    l = Arrays.asList(Serial.list());
120     SL = cp5.addScrollableList("dropdown").setPosition(690, 40)
        .setSize(150, 120).setBarHeight(20).setFont(font3)
122         .setItemHeight(20).addItems(l)
        .setOpen(false).setLabel("Elija puerto...");
124     cp5.get(ScrollableList.class, "dropdown").removeItem("COM1");
    ser = cp5.addIcon("actualizar", 10).setPosition(850, 30)
        .setSize(50, 50).setRoundedCorners(20)
        .setFont(createFont("fontawesome-webfont.ttf", 40))
128         .setFontIcon(#00f021).setColorBackground(color(255, 100))
        .hideBackground();
130     serono = cp5.addIcon("seronoser", 10).setPosition(900, 30)
        .setSize(70, 50).setRoundedCorners(20)
        .setFont(createFont("fontawesome-webfont.ttf", 40))
132         .setFontIcons(#00f205, #00f204).setSwitch(true)
    .setColorBackground(color(255, 100))
        .hideBackground();
134     //fin serial

136 //Elementos texto, botones, etc

```

```

140 cp5.addButton("buttonA").setPosition(10, 10)
141     .setImage(loadImage("unpsjb.png"));
142 cp5.addTextlabel("label1")
143     .setText("Automatizacion_tunel_de_viento").setPosition(70, 30)
144     .setColorValue(#03045e).setFont(createFont("Arial", 25));
145 TL1 = cp5.addTextlabel("label12").setText("Puerto:")
146     .setPosition(620, 35).setColorValue(#002D5A)
147     .setFont(createFont("Arial", 20));
148 TL2 = cp5.addTextlabel("label12").setText("Parametros_actuales")
149     .setPosition(690, 155).setColorValue(#002D5A)
150     .setFont(createFont("Arial", 20));
151 TL3 = cp5.addTextlabel("label3").setText("T: _____C")
152     .setPosition(602, 205).setColorValue(#002D5A)
153     .setFont(createFont("Arial", 20));
154 TL4 = cp5.addTextlabel("label4").setText("H: _____ %")
155     .setPosition(743, 205).setColorValue(#002D5A)
156     .setFont(createFont("Arial", 20));
157 TL5 = cp5.addTextlabel("label5").setText("P: _____ hPa")
158     .setPosition(862, 205).setColorValue(#002D5A)
159     .setFont(createFont("Arial", 20));
160 TL6 = cp5.addTextlabel("label6").setText("v: _____ m/s")
161     .setPosition(642, 245).setColorValue(#002D5A)
162     .setFont(createFont("Arial", 20));
163 TL7 = cp5.addTextlabel("label7")
164     .setText("v (ref): _____ m/s").setPosition(790, 245)
165     .setColorValue(#002D5A).setFont(createFont("Arial", 20));
166 TL77 = cp5.addTextlabel("label77")
167     .setText("f (aprox): _____ Hz").setPosition(770, 245)
168     .setColorValue(#002D5A).setFont(createFont("Arial", 20));
169 TL8 = cp5.addTextlabel("label8").setText("Control")
170     .setPosition(610, 460).setColorValue(#002D5A)
171     .setFont(createFont("Arial", 20));
172 TB1 = cp5.addIcon("ControlONOFF", 10).setPosition(705, 465)
173     .setSize(25, 15).setRoundedCorners(20)
174     .setFont(createFont("fontawesome-webfont.ttf", 40))
175     .setFontIcons(#00f205, #00f204).setSwitch(true)
176     .setColorBackground(color(255, 100)).hideBackground();
177 TL9 = cp5.addTextlabel("label10").setText("Velocidad:")
178     .setPosition(640, 495).setColorValue(#002D5A)
179     .setFont(createFont("Arial", 20));
180 TF1 = cp5.addTextfield("Veloc").setPosition(744, 495)
181     .setSize(100, 30).setAutoClear(false).setColor(cc)
182     .setFont(font).setLabel("");
183 TB2 = cp5.addButton("EnviarVel").setPosition(744 + 100, 495)
184     .setSize(85, 30).setLabel("Enviar").setFont(font2);
185 TL99 = cp5.addTextlabel("label110").setText("Frecuencia:")
186     .setPosition(635, 495).setColorValue(#002D5A)
187     .setFont(createFont("Arial", 20));
188 TF11 = cp5.addTextfield("Frec").setPosition(744, 495)
189     .setSize(100, 30).setAutoClear(false)
190     .setColor(cc).setFont(font).setLabel("");
191 TB22 = cp5.addButton("EnviarFrec").setPosition(744 + 100, 495)
192     .setSize(85, 30).setLabel("Enviar").setFont(font2);
193 TL10 = cp5.addTextlabel("label9").setText("Guardado")
194     .setPosition(730, 560).setColorValue(#002D5A)
195     .setFont(createFont("Arial", 20));
196 TL11 = cp5.addTextlabel("label11").setText("Archivo:")
197     .setPosition(640, 590)
198     .setColorValue(#002D5A).setFont(createFont("Arial", 20));
199 TF2 = cp5.addTextfield("Nombre_archivo").setPosition(720, 590)
200     .setSize(160, 30).setAutoClear(false).setColor(cc)
201     .setFont(font).setLabel("");
202 TB3 = cp5.addButton("Guardar").setPosition(720 + 160, 590)
203     .setSize(85, 30).setFont(font2);
204 TB4 = cp5.addButton("Run").setPosition(620, 310).setSize(85, 85)
205     .setLabel("RUN").setFont(font2).setColorActive(#009900);
206 TB5 = cp5.addButton("Stop").setPosition(733, 310)
207     .setSize(85, 85).setLabel("STOP").setFont(font2)
208     .setColorActive(#990000);
209 TL20 = cp5.addTextlabel("estado_de_motor").setText("")

```

```

210     .setPosition(640, 410).setColorValue(#002D5A)
211     .setFont(createFont("Arial", 20));
212 TB10 = cp5.addButton("reset").setPosition(910, 75)
213     .setSize(60, 25).setLabel("RESET")
214     .setFont(createFont("Arial", 14)).setColorActive(#990000);
215 TB11 = cp5.addButton("fallaext").setPosition(910, 105)
216     .setSize(60, 35).setLabelVisible(false)
217     .setFont(createFont("Arial", 14)).setColor(ccS);
218 TL21 = cp5.addTextlabel("emerg").setText("STOP \n EMERG")
219     .setPosition(905, 105).setColorValue(#AAAAAA)
220     .setFont(createFont("Arial", 14));
221 TL18 = cp5.addTextlabel("valiTL18")
222     .setText("ingrese un numero valido").setPosition(725, 525)
223     .setColorValue(#770000).setFont(createFont("Arial", 15));
224 TB12 = cp5.addIcon("Habilitacion", 10).setPosition(920, 410)
225     .setSize(25, 15).setRoundedCorners(20)
226     .setFont(createFont("fontawesome-webfont.ttf", 40))
227     .setFontIcons(#00f205, #00f204).hideBackground()
228     .setSwitch(true).setColorBackground(color(255, 100));
229 TL19 = cp5.addTextlabel("Habili").setText("A1")
230     .setPosition(920, 380).setColorValue(#002D5A)
231     .setFont(createFont("Arial", 20));
232 TB13 = cp5.addIcon("autoFun", 10).setPosition(920, 350)
233     .setSize(25, 15).setRoundedCorners(20)
234     .setFont(createFont("fontawesome-webfont.ttf", 40))
235     .setFontIcons(#00f205, #00f204).setSwitch(true)
236     .setColorBackground(color(255, 100)).hideBackground();
237 TL22 = cp5.addTextlabel("autoFuncion").setText("AutoFuncion")
238     .setPosition(870, 320)
239     .setColorValue(#002D5A).setFont(createFont("Arial", 20));

240 List funciones = Arrays.asList("FUN_A.csv", "FUN_B.csv",
241     "FUN_C.csv");
242 ddl1 = cp5.addScrolledList("Settings").setPosition(750, 460)
243     .setSize(150, 300).setItemHeight(30).setBarHeight(30)
244     .setOpen(false).setFont(font3)
245     .setLabel("Elija archivo").addItems(funciones);
246 ICO = cp5.addButton("ICONO").setPosition(902, 460)
247     .setSize(85, 30).setLabel("Abrir").setFont(font2);
248 TL23 = cp5.addTextlabel("autoFunciontex")
249     .setText("Elija un archivo donde\n"
250     "esten los datos de\nlos escalones")
251     .setPosition(595, 475).setColorValue(#002D5A)
252     .setFont(createFont("Arial", 14));

254 table = new Table(); //realizo una tabla
255 table.addColumn("Muestra");
256 table.addColumn("Tiempo");
257 table.addColumn("Temp");
258 table.addColumn("Hum");
259 table.addColumn("Pres");
260 table.addColumn("Den");
261 table.addColumn("DP");
262 table.addColumn("Ref");
263 table.addColumn("VelFrec");
264 table.addColumn("PWM");
265 table.addColumn("Control");
266 table.addColumn("Error");
267 table.addColumn("ESTADOvariador"); //motor funcionando
268 table.addColumn("ERRORvariador");
269 table.addColumn("TiempoRel");
270 table.addColumn("ControlAutomatico");

272 grafica();
273 }
274

276 void draw() {
277     background(248, 240, 248); //Configura color fondo
278     graf_draw();

```

```

280 fill(255); //cuadros con fondo
281 stroke(#5DD2EA);
282 rect(590, 35, 400, 110); //serial
283 if (serono.isOn() == true) {
284     fill(colorONOFF); //cuadros con fondo
285     noStroke();
286     rect(620, 400, 200, 50); //Prendido o apagado motor
287     fill(255); //cuadros con fondo
288     stroke(#5DD2EA);
289     rect(590, 155, 400, 145); //Parametros actuales
290     rect(626, 206, 53, 30); //T
291     rect(767, 206, 53, 30); //H
292     rect(885, 206, 53, 30); //P
293     rect(668, 246, 53, 30); //v
294     rect(850, 246, 53, 30); //vref
295     rect(590, 455, 400, 90); //control
296     rect(590, 560, 400, 80); //guardado
297     TL2.setVisible(true); TL10.setVisible(true);
298     TL3.setVisible(true); TL4.setVisible(true);
299     TL5.setVisible(true); TL6.setVisible(true);
300     TL11.setVisible(true); TF2.setVisible(true);
301     TB3.setVisible(true); TL12.setVisible(true);
302     TL13.setVisible(true); TL12.setVisible(true);
303     TL14.setVisible(true); TL15.setVisible(true);
304     TL16.setVisible(true); TL17.setVisible(true);
305     TL144.setVisible(true); TL155.setVisible(true);
306     TL166.setVisible(true); TL177.setVisible(true);
307     TF3.setVisible(true); TF4.setVisible(true);
308     TF5.setVisible(true); TF6.setVisible(true);
309     TF7.setVisible(true); TF8.setVisible(true);
310     TB6.setVisible(true); TB7.setVisible(true);
311     TB8.setVisible(true); TB9.setVisible(true);
312     Tx3.setVisible(true); Tx4.setVisible(true);
313     TB12.setVisible(true); TL19.setVisible(true);
314     TB11.setVisible(true); TL21.setVisible(true);
315     textSize(15);
316     fill(#002D5A);
317     text(nf(t, 0, 2), 669, 226);
318     text(nf(h, 0, 2), 813, 226);
319     text(nf(p, 0, -1), 935, 226);
320     text(nf(v1, 0, 1), 715, 266);
321     text(nf(VelRef, 0, 2), 900, 266);

322     if (TB12.isOn() == true) { //ail
323         TB4.setVisible(true);
324         TB5.setVisible(true);
325         if (TB1.isOn() == true) { //control onoff
326             TL9.setVisible(true); TF1.setVisible(true);
327             TB2.setVisible(true); TL7.setVisible(true);
328             TL99.setVisible(false); TF11.setVisible(false);
329             TB22.setVisible(false); TL77.setVisible(false);
330         } else {
331             TL9.setVisible(false); TF1.setVisible(false);
332             TB2.setVisible(false); TL7.setVisible(false);
333             TL99.setVisible(true); TF11.setVisible(true);
334             TB22.setVisible(true); TL77.setVisible(true);
335         }
336         TB13.setVisible(false);
337         TL22.setVisible(false); //boton de autofuncion
338     } else {
339         if (TB1.isOn() == true) {
340             TL9.setVisible(false); TF1.setVisible(false);
341             TB2.setVisible(false); TL7.setVisible(true);
342             TL99.setVisible(false); TF11.setVisible(false);
343             TB22.setVisible(false); TL77.setVisible(false);
344         } else {
345             TL9.setVisible(false); TF1.setVisible(false);
346             TB2.setVisible(false); TL7.setVisible(false);
347             TL99.setVisible(false); TF11.setVisible(false);
348             TB22.setVisible(false); TL77.setVisible(true);
349         }
350     }
351 }

```

```

350 }
351 TB13.setVisible(true);
352 TL22.setVisible(true);
353 }
354 if (TB13.isOn() == true) { //boton de autofucion
355 TB12.setVisible(false); //apago ail
356 TL19.setVisible(false); //apago ail
357 DLL.setVisible(true); //autofucion
358 ICO.setVisible(true); //autofucion
359 TL23.setVisible(true); //autofucion
360 TL77.setVisible(false); //no muestro frecuencia
361 TL7.setVisible(false); //muestro vref
362 TB5.setVisible(true);
363 } else {
364 TB12.setVisible(true); //ail
365 TL19.setVisible(true); //ail
366 DLL.setVisible(false); //autofucion
367 ICO.setVisible(false); //autofucion
368 TL23.setVisible(false); //autofucion
369 }
370 if (TB13.isOn() == false && TB12.isOn() == false) {
371 TB4.setVisible(false);
372 TB5.setVisible(false);
373 }
374 if (unicavez2 == false) {
375 unicavez2 = true;
376 }
377 if (int(Ev) == 1) {
378 colorONOFF = #AA0000;
379 TL20.setVisible(true);
380 TL20.setText("Motor_en_Marcha").setColorValue(#FFFFFF);
381 } else {
382 colorONOFF = #00AA00;
383 TL20.setVisible(true);
384 TL20.setText("Motor_Apagado");
385 }
386 if (int(ErrorVariador) == 1) { //triangulo reset
387 TB10.setVisible(true);
388 fill(255, 0, 0);
389 noStroke();
390 triangle(850, 90, 900, 90, 875, 135);
391 if (unicavez3 == false) {
392 TB1.setOff();
393 TB12.setOff();
394 TB13.setOff();
395 unicavez3 = true;
396 }
397 } else {
398 TB10.setVisible(false);
399 fill(255);
400 noStroke();
401 triangle(850, 90, 900, 90, 875, 135);
402 unicavez3 = false;
403 }
404 if (int(ControlAutomatico) == 1 && TB1.isOn() == false) { //autofucion
405 fill(colfun); //de que envie el archico
406 noStroke();
407 rect(930, 500, 30, 30);
408 colfun = #008800;
409 TL77.setVisible(false); //no muestro frecuencia
410 TL7.setVisible(true); //muestro vref
411 }
412 if (int(ControlAutomatico) == 0 && TB1.isOn() == false) {
413 colfun = #228888;
414 TL77.setVisible(true); //no muestro frecuencia
415 TL7.setVisible(false); //muestro vref
416 }
417 if (DD4 == true) { //reset
418 if ((millis() - tiempo2)>3000) {
419

```

```

        DD4 = false;
420    Dato4 = "0";
        DatosW();
422    reinicioreset = false;
    }
424 }
} else { //si esta apagado serial arduino
426    TL2.setVisible(false);    TL3.setVisible(false);
    TL4.setVisible(false);    TL5.setVisible(false);
428    TL6.setVisible(false);    TL7.setVisible(false);
    TL9.setVisible(false);    TF1.setVisible(false);
430    TB2.setVisible(false);    TL77.setVisible(false);
    TL99.setVisible(false);    TF11.setVisible(false);
432    TB22.setVisible(false);   TL8.setVisible(false);
    TL10.setVisible(false);   TL11.setVisible(false);
434    TF2.setVisible(false);   TB3.setVisible(false);
    TB4.setVisible(false);   TB5.setVisible(false);
436    TB1.setVisible(false);   TL12.setVisible(false);
    TL20.setVisible(false);   TL13.setVisible(false);
438    TL14.setVisible(false);   TL15.setVisible(false);
    TL16.setVisible(false);   TL17.setVisible(false);
440    TL144.setVisible(false);  TL155.setVisible(false);
    TL166.setVisible(false);  TL177.setVisible(false);
442    TF3.setVisible(false);   TF4.setVisible(false);
    TF5.setVisible(false);   TF6.setVisible(false);
444    TF7.setVisible(false);   TF8.setVisible(false);
    TB6.setVisible(false);   TB7.setVisible(false);
446    TB8.setVisible(false);   TB9.setVisible(false);
    Tx3.setVisible(false);   Tx4.setVisible(false);
448    TB10.setVisible(false);  TL18.setVisible(false);
    TB12.setVisible(false);  TL19.setVisible(false);
450    TB11.setVisible(false);  TL21.setVisible(false);
    TB13.setVisible(false);  TL22.setVisible(false);
452    TL23.setVisible(false);  dll1.setVisible(false);
    ICO.setVisible(false);
}
454 }
456 void DatosW() {
458    DatosWrite = (Dato0 + "," + Dato1 + "," + Dato2 + "," + Dato3 + ","
                   + Dato4 + "," + Dato5 + "," + Dato6 + "," + '\n');
460    Arduino.write(DatosWrite);
}
462 void verificacion() {
464    if (unicavez == 0) {
        variablecontrol1 = int(controlsINO);
466    tiempo3 = tiempo;
        if (variablecontrol1 == 1) {
468        cp5.get(Textfield.class, "Veloc").setText(str(VelRef));
        EnviarVel();
        TB12.setOn();
        TB1.setOn();
472    } else {
        TB12.setOff();
        TB1.setOff();
    }
476    unicavez = 1;
}
478 }

480 void serialEvent(Serial Arduino) {
482    if ((millis() - tiempol)>1000) {
        try {
484        val = Arduino.readStringUntil('\n'); //separador de nueva linea
        if (val!= null) { //verifica q no esta vacio
486            if (tomomuestra == 1) {
                val1 = val;
                val = trim(val); //

```

```

490     float algo[] = float(split(val, ',')); //separo; y paso flotante.
491     tiempo = algo[4];
492     t = algo[5];
493     h = algo[6];
494     p = algo[7] / 100;
495     d = algo[8];
496     dp = algo[2];
497     VelRef = algo[1];
498     v1 = algo[0];
499     pwm = algo[3];
500     controlSINO = algo[9];
501     error = algo[10];
502     Ev = algo[11]; //estado del motor
503     ErrorVariador = algo[12]; //indicacion de error en variador
504     ControlAutomatico = algo[13];
505
506     TableRow newRow = table.addRow(); //crea nueva columna
507     newRow.setInt("Muestra", table.lastRowIndex());
508     newRow.setFloat("Tiempo", tiempo); //0
509     newRow.setFloat("Temp", t); //1
510     newRow.setFloat("Hum", h); //2
511     newRow.setFloat("Pres", p); //3
512     newRow.setFloat("Den", d); //4
513     newRow.setFloat("DP", dp); //5
514     newRow.setFloat("Ref", VelRef); //6
515     newRow.setFloat("VelFrec", v1); //7
516     newRow.setFloat("PWM", pwm); //8
517     newRow.setFloat("Control", controlSINO); //9
518     newRow.setFloat("Error", error); //10
519     newRow.setFloat("ESTADOvariador", Ev); //11 //motor funcionando
520     newRow.setFloat("ERRORvariador", ErrorVariador); //12
521     newRow.setFloat("TiempoRel", tiempo - tiempo3);
522     newRow.setFloat("ControlAutomatico", ControlAutomatico);
523     tomomuestra = 0;
524 }
525         tomomuestra = tomomuestra + 1;
526     }
527     catch(RuntimeException e) { //catches errors
528     }
529 }
530 }

532 /**
533 * =====
534 * Realiza el grafico
535 * =====
536 */
537 class Graph {
538     boolean Dot=true; // Draw dots at each data point if true
539     boolean RightAxis; // Draw the next graph using the right axis if T
540     boolean ErrorFlag=false; // If the time array isn't in ascending order
541     boolean ShowMouseLines=true;
542     int xDiv=5, yDiv=5; // Number of sub divisions
543     int xPos, yPos; // location of the top left corner of the graph
544     int Width, Height; // Width and height of the graph
545     color GraphColor;
546     color BackgroundColor=color(255);
547     color StrokeColor=color(180);
548
549     String Title="Title"; // Titulos
550     String xLabel="x_Label";
551     String yLabel="y_Label";
552
553     float yMax=1024, yMin=0; // Default axis dimensions
554     float xMax=10, xMin=0;
555     float yMaxRight=1024, yMinRight=0;
556     PFont Font; // Selected font used for text
557
558     Graph(int x, int y, int w, int h, color k) {

```

```

560     xPos = x;
561     yPos = y;
562     Width = w;
563     Height = h;
564     GraphColor = k;
565 }
566
567 void DrawAxis() {
568     fill(BackgroundColor);
569     color(0);
570     stroke(StrokeColor);
571     strokeWeight(1);
572     int t=40;
573     rect(xPos-t*1.6, yPos-t*0.8, Width+t*2, Height+t*1.7);
574     textAlign(CENTER);
575     textSize(18);
576     float c=textWidth>Title;
577     fill(BackgroundColor);
578     color(0);
579     stroke(0);
580     strokeWeight(1);
581     rect(xPos+Width/2-c/2, yPos-35, c, 0);
582
583     fill(0);
584     textSize(10);
585     noFill();
586     stroke(0);
587     smooth();
588     strokeWeight(1);
589     //Edges
590     line(xPos-3, yPos+Height, xPos-3, yPos);
591     line(xPos-3, yPos+Height, xPos+Width+5, yPos+Height);
592
593     stroke(200);
594     if (yMin<0) {
595         line(xPos-7, // zero line
596             yPos+Height-(abs(yMin)/(yMax-yMin))*Height, //
597             xPos+Width,
598             yPos+Height-(abs(yMin)/(yMax-yMin))*Height
599         );
600     }
601
602     if (RightAxis) {
603         stroke(0);
604         line(xPos+Width+3, yPos+Height, xPos+Width+3, yPos);
605     }
606
607     stroke(0);
608     for (int x=0; x<=xDiv; x++) {
609         /* =====
610            x-axis
611            ===== */
612
613         line(float(x)/xDiv*Width+xPos-3, yPos+Height, // x-axis Sub devisions
614             float(x)/xDiv*Width+xPos-3, yPos+Height+5);
615         textSize(15); // x-axis Labels
616         String xAxis=str(xMin+float(x)/xDiv*(xMax-xMin));
617         String[] xAxisMS=split(xAxis, '.');
618     }
619
620     /* =====
621        left y-axis
622        ===== */
623
624     for (int y=0; y<=yDiv; y++) {
625         line(xPos-3, float(y)/yDiv*Height+yPos, // ...
626             xPos-7, float(y)/yDiv*Height+yPos); // y-axis lines
627         textAlign(RIGHT);
628         fill(20);

```

```

630     String yAxis=str(yMin+float(y)/yDiv*(yMax-yMin));
631     String[] yAxisMS=split(yAxis, '.');                                // Split string
632     text(yAxisMS[0]+". "+yAxisMS[1].charAt(0), // ...
633           xPos-15, float(yDiv-y)/yDiv*Height+yPos+3);                // y-axis Labels
634     stroke(0);
635   }
636 }

638 /* =====
639 Straight line graph
640 ===== */
642 void LineGraph(float[] x, float[] y) {
644   for (int i=0; i<(x.length-1); i++) {
645     strokeWeight(2);
646     stroke(GraphColor);
647     noFill();
648     smooth();
649     line(xPos+(x[i]-x[0])/((x[x.length-1]-x[0])*Width,
650       yPos+Height-(y[i]/(yMax-yMin)*Height)+(yMin)/(yMax-yMin)*Height,
651       xPos+(x[i+1]-x[0])/((x[x.length-1]-x[0])*Width,
652       yPos+Height-(y[i+1]/(yMax-yMin)*Height)+(yMin)/(yMax-yMin)*Height);
653     }
654   }
655 }
656 /* =====
657 Adquiere los valores que seran representados en tiempo real
658 ===== */
660 int i = 0; // loop variable
662 void graf_draw() {
664   /* Read serial and update values */
666   if (serono.isOn() ==true) {
667     String myString = "";
668     myString=str(v1)+";"+str(VelRef)+" ; "+str(dp)+" ; "+str(pwm)+" ; ";
669     String[] nums = split(myString, ',');
670

672   for (i=0; i<nums.length; i++) { //nums.length
673     // update line graph
674     try {
675       if (i<lineGraphValues.length) {
676         for (int k=0; k<lineGraphValues[i].length-1; k++) {
677           lineGraphValues[i][k] = lineGraphValues[i][k+1];
678         }
679         lineGraphValues[i][lineGraphValues[i].length-1] =
680           float(nums[i])*float(getPlotterConfigString("lgMultiplier"+(i+1)));
681       }
682     } catch (Exception e) {
683     }
684   }
685 }

686   // draw the bar chart
687   background(255);

689   // draw the line graphs
690   LineGraph.DrawLine();
691   for (int i=0; i<lineGraphValues.length; i++) {
692     LineGraph.GraphColor = graphColors[i];
693     if (int(getPlotterConfigString("lgVisible"+(i+1))) == 1)
694       LineGraph.LineGraph(lineGraphSampleNumbers, lineGraphValues[i]);

```

```

    }
700 }
702 /* =====
704 Eventos de la grafica
===== */
706 void setChartSettings() {
708 LineGraph.xLabel="Samples";
709 LineGraph.yLabel="Value";
710 LineGraph.Title="";
711 LineGraph.xDiv=10;
712 LineGraph.xMax=0;
713 LineGraph.xMin=-100;
714 LineGraph.yMax=int(getPlotterConfigString("lgMaxY"));
715 LineGraph.yMin=int(getPlotterConfigString("lgMinY"));
716 }

718 // handle gui actions
719 void controlEvent(ControlEvent theEvent) {
720 if (theEvent.isAssignableFrom(Textfield.class) ||
721     theEvent.isAssignableFrom(Toggle.class) ||
722     theEvent.isAssignableFrom(Button.class)) {
723     String parameter = theEvent.getName();
724     String value = "";
725     if (theEvent.isAssignableFrom(Textfield.class))
726         value = theEvent.getStringValue();
727     else if (theEvent.isAssignableFrom(Toggle.class) ||
728              theEvent.isAssignableFrom(Button.class))
729         value = theEvent.getValue()+"";
730     plotterConfigJSON.setString(parameter, value);
731     saveJSONObject(plotterConfigJSON, topSketchPath+"/plotter_config.json");
732 }
733 setChartSettings();
734 }
735

736 // get gui settings from settings file
737 String getPlotterConfigString(String id) {
738     String r = "";
739     try {
740         r = plotterConfigJSON.getString(id);
741     } catch (Exception e) {
742         r = "";
743     }
744     return r;
745 }
746

747 /* =====
748 Elementos de la parte grafica de tiempo real
749 ===== */
750
751
752
753
754 void grafica() {
755     // set line graph colors
756     graphColors[0] = color(131, 255, 20);
757     graphColors[1] = color(232, 158, 12);
758     graphColors[2] = color(255, 0, 0);
759     graphColors[3] = color(62, 12, 232);
760
761     //settings save file
762     topSketchPath = sketchPath();
763     plotterConfigJSON = loadJSONObject(topSketchPath + "/plotter_config.json");
764
765     //init charts
766     setChartSettings();
767 }

```

```

770     for (int i = 0; i < lineGraphValues.length; i++) {
771         for (int k = 0; k < lineGraphValues[0].length; k++) {
772             lineGraphValues[i][k] = 0;
773             if (i == 0)
774                 lineGraphSampleNumbers[k] = k;
775         }
776     }

778     //build the gui
779     int x = 24;
780     int y = 105;
781     TF3=cp5.addTextfield("lgMaxY").setPosition(475, 575).setLabel("")
782         .setText(getPlotterConfigString("lgMaxY")).setWidth(40)
783         .setAutoClear(false).setFont(createFont("Arial", 15));
784     TF4=cp5.addTextfield("lgMinY").setPosition(475, 610).setLabel("")
785         .setText(getPlotterConfigString("lgMinY")).setWidth(40)
786         .setAutoClear(false).setFont(createFont("Arial", 15));
787     Tx3=cp5.addTextlabel("yMax3").setText("yMax:").setPosition(415, 570)
788         .setColor(#002D5A).setFont(createFont("Arial", 20));
789     Tx4=cp5.addTextlabel("yMin4").setText("yMin:").setPosition(417, 605)
790         .setColor(#002D5A).setFont(createFont("Arial", 20));

792     TL14=cp5.addTextlabel("variable1").setText("vel")
793         .setPosition(x = 65, y = 554)
794         .setColor(#002D5A).setFont(createFont("Arial", 20));
795     TL15=cp5.addTextlabel("variable2").setText("v/f(ref)")
796         .setPosition(x= x+80, y)
797         .setColor(#002D5A).setFont(createFont("Arial", 20));
798     TL16=cp5.addTextlabel("variable3").setText("dDP")
799         .setPosition(x = x+80, y)
800         .setColor(#002D5A).setFont(createFont("Arial", 20));
801     TL17=cp5.addTextlabel("variable4").setText("Acc.C")
802         .setPosition(x = x+80, y)
803         .setColor(#002D5A).setFont(createFont("Arial", 20)); //accion de control
804     TL144=cp5.addTextlabel("variable11").setText("[m/s]")
805         .setPosition(x = 76, y = 576)
806         .setColor(#002D5A).setFont(createFont("Arial", 14));
807     TL155=cp5.addTextlabel("variable22").setText("[m/s-Hz]")
808         .setPosition(x = x+80, y)
809         .setColor(#002D5A).setFont(createFont("Arial", 14));
810     TL166=cp5.addTextlabel("variable33").setText("[Pa]")
811         .setPosition(x = x+80, y)
812         .setColor(#002D5A).setFont(createFont("Arial", 14));
813     TL177=cp5.addTextlabel("variable44").setText("[--]")
814         .setPosition(x = x+80, y)
815         .setColor(#002D5A).setFont(createFont("Arial", 14));

817     TL12=cp5.addTextlabel("label").setText("I_O")
818         .setPosition(x = 12, y = 600)
819         .setColor(#002D5A).setFont(createFont("Arial", 20));
820     TL13=cp5.addTextlabel("multipliers").setText("Escala")
821         .setPosition(x = 7, y = 620)
822         .setColor(#002D5A).setFont(createFont("Arial", 20));
823     TF5=cp5.addTextfield("lgMultiplier1").setPosition(x = 75, y = 625)
824         .setLabel("").setText(getPlotterConfigString("lgMultiplier1"))
825         .setWidth(40).setAutoClear(false).setFont(createFont("Arial", 20));
826     TF6=cp5.addTextfield("lgMultiplier2").setPosition(x = x + 80, y)
827         .setLabel("").setText(getPlotterConfigString("lgMultiplier2"))
828         .setWidth(40).setAutoClear(false)
829         .setFont(createFont("Arial", 20));
830     TF7=cp5.addTextfield("lgMultiplier3").setPosition(x = x + 80, y)
831         .setLabel("").setText(getPlotterConfigString("lgMultiplier3"))
832         .setWidth(40).setAutoClear(false).setFont(createFont("Arial", 20));
833     TF8=cp5.addTextfield("lgMultiplier4").setPosition(x = x + 80, y)
834         .setLabel("").setText(getPlotterConfigString("lgMultiplier4"))
835         .setWidth(40).setAutoClear(false)
836         .setFont(createFont("Arial", 20));
837

```

```
840   TB6=cp5.addToggle("lgVisible1").setPosition(x = 75, y = 602).setLabel("")  
841     .setValue(int(getPlotterConfigString("lgVisible1")))  
842     .setColorActive(graphColors[0]);  
842   TB7=cp5.addToggle("lgVisible2").setPosition(x = x + 80, y).setLabel("")  
843     .setValue(int(getPlotterConfigString("lgVisible2")))  
844     .setColorActive(graphColors[1]);  
844   TB8=cp5.addToggle("lgVisible3").setPosition(x = x + 80, y).setLabel("")  
845     .setValue(int(getPlotterConfigString("lgVisible3")))  
846     .setColorActive(graphColors[2]);  
848   TB9=cp5.addToggle("lgVisible4").setPosition(x = x + 80, y).setLabel("")  
849     .setValue(int(getPlotterConfigString("lgVisible4")))  
850     .setColorActive(graphColors[3]);  
850 }
```
