Validación en Tiempo Real de un Controlador Predictivo de Glucosa en Pacientes IN-SILICO

Julieth Ximena Arias Guzmán y Santiago Andrés Vergara Hernández

Trabajo de Grado para Optar el título de Ingeniero Electrónico

Director
Rodolfo Villamizar Mejía.
PhD. Tecnologías de la Información

Codirector
David Alberto Padilla Tolosa
MSc. Ingeniería Electrónica

Universidad Industrial de Santander
Facultad de Ingenierías Físico-mecánicas
Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones
Bucaramanga
2021

Contenido

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Tabla de Apéndices

Pág.

Resumen

Título: Validación en Tiempo Real de un Controlador Predictivo de Glucosa en Pacientes IN-SILICO [*]

Autor: Julieth Ximena Arias Guzmán, Santiago Andrés Vergara Hernández[**]

Palabras Clave: Glucosa, T1DM, ESP32, Control Predictivo, Diabetes Mellitus tipo I

Descripción:

La diabetes Mellitus es una de las enfermedades más preocupantes a nivel mundial debido a sus efectos en la salud, índice de mortalidad y la tendencia creciente de su prevalencia. Algunas de las personas que padecen esta patología requieren de inyecciones diarias de insulina para regular la glucemia y evitar episodios de hiperglucemia.

Por tal motivo la automatización y el control de procesos han desarrollado un papel fundamental en el área de la investigación buscando recuperar algunas funciones que el cuerpo humano ha perdido. Por esto, el objetivo principal de este proyecto consiste en validar en tiempo real un controlador predictivo basado en la técnica MPC de modelo desacoplado, usando la tarjeta DS1104 del fabricante DSPACE, para programar sobre el modelo de pacientes T1DM presentados en el simulador metabólico T1DMS de UVA / PADOVA, que es aceptado por la FDA como validador de pruebas in silico, con el fin de obtener un perfil dinámico de la glucosa en pacientes con diabetes mellitus tipo I similar a un perfil de paciente sano, al dosificarle una tasa de insulina obtenida por la estrategia de control programada en el microcontrolador ESP32 que cumple con las exigencias computacionales exigidas por el algoritmo a implementar, además de realizar los cálculos de la ley de control de manera óptima.

---

[*] Trabajo de Grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones Director: RODOLFO VILLAMIZAR MEJÍA. PhD. Tecnologías de la información, Co-director: DAVID ALBERTO PADILLA TOLOSA. MSc Ingeniería Electrónica.

Abstract

Title: Real time validation of a predictive controller of glucose in IN-SILICO patients

Author: Julieth Ximena Arias Guzmán, Santiago Andrés Vergara Hernández[**]

Key Words: Glucose, T1DM, ESP32, Predictive Controller, Type I Diabetes Mellitus

Description:

Diabetes Mellitus is one of the most worrying diseases due to the grown in its prevalence worldwide, its effects on healthy and the mortality rate. People with this pathology need daily insulin injections to regulate the blood glucose and avoid episodes of hyperglycemia.

For this reason, automation and process control have played a fundamental role in the area of research, seeking to recover some functions that the human body has lost. Thus, the main objective of this project is to validate in real time a predictive controller based on the MPC technique of decoupled model, using the DS1104 card from the manufacturer DSPACE, to program on the model of T1DM patients models presented in the T1DMS metabolic simulator of UVA/PADOVA, which has been accepted by the FDA as an in silico test validator, in order to obtain a dynamic glucose profile in patients with type 1 Diabetes Mellitus similar to a healthy patient profile, by dosing an insulin rate obtained by the control strategy programmed in the ESP32 microcontroller that complies with the computational requirements demanded by the algorithm to be implemented, in addition to optimally performing the control law calculations.

[**] Faculty of Physics and Mechanical Engineering. School of Electrical, Electronic and Telecommunications Engineering. Director: RODOLFO VILLAMIZAR MEJÍA. PhD in Information Technology Co-director: DAVID ALBERTO PADILLA TOLOSA MSc in Engineer Electronic.

Introducción

La diabetes mellitus tipo I (T1DM) es una de las enfermedades crónicas más comunes a nivel mundial. Esta, se caracteriza por un aumento anormal en los niveles de glucosa en la sangre, ocasionado por el déficit en la secreción de insulina por parte del páncreas, debido a que las células beta de los islotes pancreáticos se encuentran dañadas. El principal tratamiento para la diabetes tipo I consiste en administrar insulina exógena al paciente, por medio de inyecciones periódicas o bombas programables. Por esta razón, diversas ramas del conocimiento, como la medicina y la ingeniería se han enfocado en el desarrollo y la aplicación de herramientas de hardware computacional para el tratamiento de esta clase de patologías tal como el "Type 1 Diabetes Mellitus Simulator" (T1DMS) desarrollado por las universidades de Padova y Virginia.

En el 2019 la organización mundial de la salud presentó la última versión del atlas de la diabetes, donde se estimó que 463 millones de personas tienen diabetes. A su vez, se espera que esta cifra aumente a 578 millones para el año 2030 y a 700 millones para 2045. Así mismo, el 11,3 % de las muertes a nivel mundial son causadas por esta patología (Cho & Williams, 2019).

Este trabajo de investigación grado se basa en la programación en hardware y validación en tiempo real de un esquema de control realimentado, que busca seguir un perfil saludable de glucosa en sangre del paciente, mediante una estrategia de control predictivo basado en el modelo (MPC). Esta estrategia de control utiliza un modelo dinámico bien descrito para realizar predicciones de la salida del sistema utilizando la información presente y pasada del proceso, con base en la optimización de una función de coste (Bordóns, & Rodríguez. 2005). Para la adquisición

de datos se utilizó el modelo UVA/ Padova del simulador diabético (T1DMS), utilizado para probar controladores de páncreas artificiales y recientemente nuevas formulaciones de insulina y sensores de glucosa. Este consiste en un modelo que describe la dinámica de la glucosa-insulina en una población de pacientes virtuales clasificados en niños, jóvenes y adultos.

De esta forma, se realiza la elección de una unidad de procesamiento digital de acuerdo con los requerimientos computacionales del algoritmo de control que se pretende programar, con el objetivo de implementarlo en un dispositivo de cómputo y validarlo mediante pruebas de desempeño funcional.

Objetivos

Objetivo General

Validar en tiempo real el funcionamiento de una estrategia de control predictivo programada en una unidad de procesamiento digital sobre los modelos de diez pacientes in-silico del simulador UVA/Padova.

Objetivos Específicos

Establecer una unidad de cómputo adecuada a partir de las exigencias computacionales de la estrategia de control.

Implementar a estrategia de control predictivo en una unidad de cómputo aplicando programación orientada a objetos.

Validar el desempeño funcional del controlador predictivo implementado en la unidad de procesamiento digital y compararlo con un controlador PID clásico.

1. Marco Conceptual.

1.1 Diabetes

Como expresa la Organización Mundial de la Salud (OMS, 2020), la diabetes es una enfermedad crónica caracterizada por episodios de hiperglucemia (concentración elevada de azúcar en la sangre) debido a que el páncreas no produce insulina suficiente o porque el organismo no utiliza la insulina que produce. La insulina es la hormona encargada de regular el azúcar en la sangre. Episodios prolongados de hiperglucemia, ocasionan una afectación permanente de varios órganos en el paciente, especialmente los nervios y los vasos sanguíneos dañándolos gravemente. Entre sus principales síntomas se incluyen la excreción excesiva de orina (poliuria), sed (polidipsia), hambre constante, pérdida de peso, trastornos visuales y cansancio, sintomatología que se presenta de forma súbita (OMS, 2020).

En 2019, se estimó que 351,7 millones de personas en edad activa (20 – 64 años) tenían diabetes diagnosticada o sin diagnosticar. Se espera que esta cifra aumente a 417,3 millones para el año 2030, y a 486.1 millones para 2045, a nivel mundial. El 11,3% de las muertes están causadas por esta patología, en las mujeres se asocia una mayor tasa de mortalidad que en varones (Cho & Williams, 2019).

1.1.1 Diabetes Mellitus Tipo I (T1DM)

Es una enfermedad crónica, también conocida como diabetes insulino independiente. Esta enfermedad produce cambios en el metabolismo de los carbohidratos y lípidos. Se caracteriza por

alterar las células beta pancreáticas, logrando una deficiencia absoluta en la producción de insulina que impide al organismo regular de forma adecuada la glucosa plasmática. En consecuencia, el paciente presenta episodios de hiperglucemia, condición que puede ocasionar la muerte de este si no es tratada con insulina a tiempo (Mejía et al., 2020).

Esta patología afecta a la población pediátrica y adolescente. Dentro de los síntomas están la sed excesiva, visión borrosa, micción frecuente, hambre constante, falta de energía y pérdida de peso.

1.1.2 Modelo Uva Padova

Es el modelo que usa el simulador de diabetes mellitus tipo I (T1DMS) desarrollado por el departamento de Ingeniería de las universidades de Virginia y Padova. Este modelo describe la dinámica de glucosa en un paciente con diabetes tipo I a partir en un conjunto de 17 ecuaciones diferenciales de primer orden que modelan la dinámica del tracto digestivo, el subsistema de glucosa, la utilización de la glucosa, la producción de endógena de glucosa efectuada por el hígado y el subsistema de insulina. Ha permitido lograr avances en la investigación referente a la diabetes mellitus tipo 1 (T1DM), permitiendo la posibilidad de efectuar diversas pruebas en pacientes in silico (Visentin et al., 2018).

El simulador fue programado en Matlab cuya versión comercial se lanzó en el 2013. La nueva versión del simulador T1D UVA/Padova (S2017) incorpora algunos parámetros que varían en el tiempo como la variabilidad intradía de la sensibilidad a la insulina y el fenómeno amanecer, además de la actualización en el modelo del glucagón, lo que permite la descripción de la dinámica

de la glucosa mediante la descripción de la sensibilidad a la insulina y la tasa de absorción intestinal de las comidas. Además, el modelo describe los últimos avances en los dispositivos de medición de la glucosa y sus rutas de administración (Visentin et al., 2018). Su versión comercial cuenta con los parámetros de 30 pacientes con diabetes tipo I, divididos en 3 grupos: niños, adolescentes y adultos.

1.2 Control predictivo basado en el modelo (MPC)

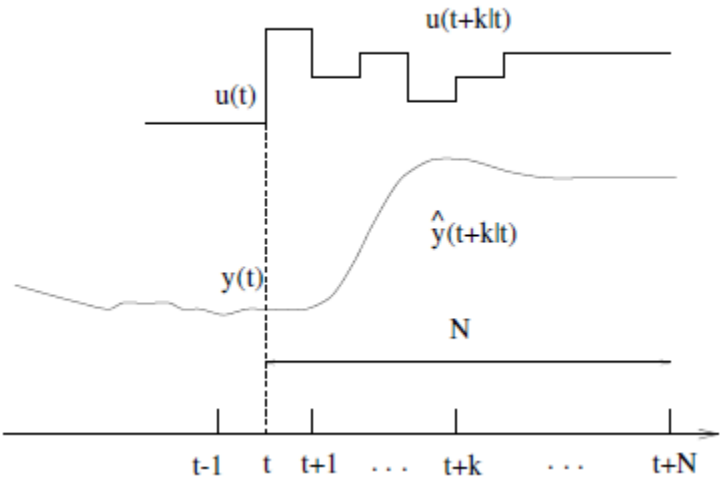
El control predictivo basado en el modelo o MPC ("Model Predictive Control"), es una estrategia de control que aplica entradas a un modelo bien descrito para hacer predicciones de la salida del proceso a lo largo de una ventana de tiempo (horizonte). Estas predicciones se usan para calcular las acciones de control a partir de la minimización de una función objetivo. Cada instante de muestreo, se envía al proceso la acción de control calculada únicamente para el instante siguiente, se eliminan los demás cálculos y el horizonte de predicción se desplaza hacia el futuro, a esto se le conoce como estrategia deslizante. En la actualidad se han desarrollado distintas aplicaciones de controladores predictivos en la industria de procesos como en el control de motores o robótica.

1.2.1 Estrategia controlador predictivo

La Figura 1 presenta una idea básica de la estrategia de control predictivo.

Figura 1

Estrategia de control predictivo



Nota: Adaptado de Bordóns, C & Rodríguez, D. (2005). Análisis y control de sistemas en espacio de estado, identificación de sistemas, control adaptativo, control predictivo. Recuperado de http://www.esi2.us.es/~danirr/apuntesIC4.pdf.

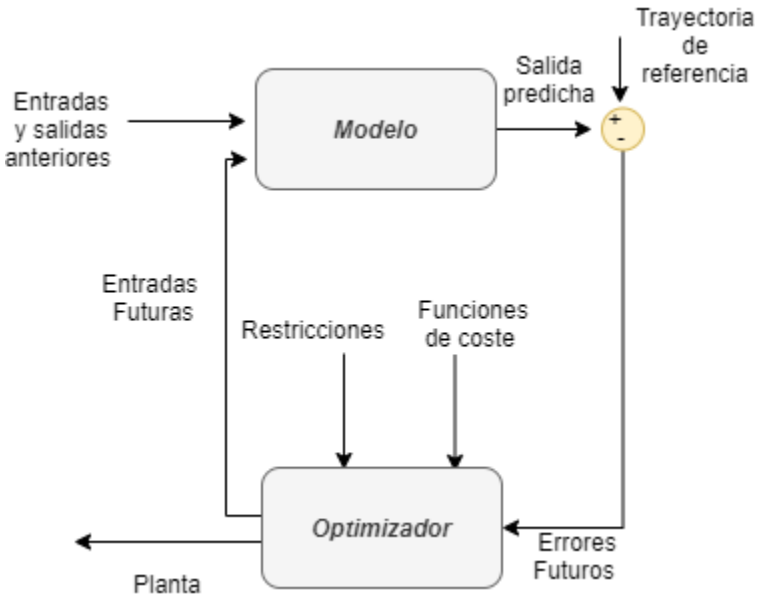
Como se observa en la Figura 1, para un instante de tiempo t, se realizan predicciones de la salida a lo largo de una ventana denominada horizonte de predicción (N) (Bordóns, C & Rodríguez, D, 2005). Se predicen salidas a partir de la información presente, pasada y las acciones de control futuras. Estas señales se determinan mediante la optimización de una función cuadrática del error entre la salida y la trayectoria de referencia, incluyendo el esfuerzo de control. Cuando el modelo es lineal y no se tienen en cuenta restricciones, se puede obtener una solución explícita, en otro caso se realiza un método iterativo de optimización. Si se desea, se puede utilizar una trayectoria de referencia, que consiste en una aproximación exponencial de la salida actual al *set point*. Las señales futuras son calculadas a partir del optimizador utilizando la función de coste

(donde aparece el futuro error de seguimiento), así como las restricciones del modelo, parámetros importantes para el controlador. La señal de control $u(t|t)$ se envía al proceso, mientras las señales de control $u(t+k|t)$ son desechadas. En el siguiente instante de muestreo se repite el proceso con las condiciones iniciales actualizadas, aplicando el concepto de horizonte deslizante.

El modelo debe ser capaz de tomar la dinámica del proceso para predecir las acciones futuras, además debe ser fácil de usar y comprender. La estructura utilizada para cumplir esta estrategia se muestra en la Figura 2, donde se observa el modelo de predicción, la función objetivo y la obtención de la ley de control.

Figura 2

Esquema aplicado por el algoritmo de control predictivo para calcular las acciones de control



1.2.2 Modelo de predicción

El modelo de predicción es importante para el algoritmo de control predictivo, ya que de este depende la fiabilidad de las predicciones. Por lo tanto, a mayor precisión del modelo más efectivo será la ley de control calculada. El modelo se usa para calcular salidas predichas denominadas $\hat{y}[k+i|k]$, donde en la parte derecha de la llave se indica el instante de tiempo actual en que se encuentra el sistema y en la izquierda el instante de tiempo futuro estimado durante la predicción, que se indica con el índice *i*. El modelo se puede separar en dos partes, modelo del proceso y modelo de las perturbaciones.

- Modelo del proceso: Describe la dinámica del proceso y, aparece con la formulación de MPC utilizando la respuesta impulsional, función de transferencia, espacio de estados, y respuesta al escalón.
- Modelo de las Perturbaciones: Es importante la elección de un modelo utilizado para representar las perturbaciones. Es decir, la diferencia entre la salida medida y la calculada por el modelo.

1.2.3 Horizonte de predicción

Es una ventana de tiempo que contiene todas las predicciones utilizadas en cada iteración para el cálculo de la acción de control. Un horizonte de predicción debe ser capaz de albergar la respuesta transitoria y en estado estable del sistema.

1.2.4 Horizonte de control

Es una ventana dentro del horizonte de predicción, dentro de la que se permiten variaciones de la señal de control, fuera de esta ventana el valor de la acción de control permanecerá constante en lo que quede del horizonte de predicción. El valor del horizonte de control debe ser menor o igual al valor del horizonte de predicción.

1.2.5 Función objetivo

Se plantean diferentes funciones objetivo para la obtener la ley de control. Sin embargo, una de las más utilizadas penaliza errores de seguimiento durante las predicciones y el esfuerzo de control.

La expresión general de esta función viene dada por la ecuación (**1**) tomada del libro "Predictive Control with Constraints" de (Maciejowski, 2000), donde δ es la matriz que pondera el error cuadrático entre la salida predicha y la referencia y λ es la matriz que pondera la variación del esfuerzo de control.

$$J(N1, N2, Nu) = \sum_{j=N1}^{N2} \delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{Nu} \lambda(j)\,[\Delta u(t+j-1)]^2 \quad (1)$$

El primer componente de la función objetivo agrupa el comportamiento del futuro, la salida, trayectoria de referencia y la señal de control. El segundo componente es denominado esfuerzo de control, que penaliza cambios abruptos en la acción de control. Dentro de la función de coste se considera:

Parámetros: N_1 y N_2 se denominan los horizontes mínimo y máximo de coste (o de predicción) y Nu el horizonte de control, que no coincide con el horizonte máximo. N_1 y N_2 representan los limites en los que se desea que la salida siga a la referencia.

1.2.6 Obtención de la ley de control

La ley de control se basó en el algoritmo de control predictivo QDMC, donde la principal característica consiste en usar el modelo de predicción para generar sus propias condiciones iniciales en cada iteración. A continuación, se explicará cómo se desarrolló el algoritmo de control basados en (Padilla Toloza, 2021):

- Modelo de predicción: Se planteó el modelo en el dominio en espacio de estados discreto, definiendo la matriz de estado G_m, la matriz de entradas H_m, la matriz de salida C_m, que representa la medición del estado de glucosa subcutánea y la matriz $\mathrm{D_m}$. Para calcular las predicciones, la salida del modelo se divide en respuesta libre que depende de las condiciones iniciales y la respuesta forzada que depende únicamente de la entrada.

$$\hat{y}[k+i|k] = \hat{y}_{libre}[k+i|k] + \theta \Delta \hat{u}[k|k] \quad (2)$$

- Matriz de respuesta forzada: se calcula para cada instante a lo largo del horizonte de predicción, aplicando al modelo una entrada escalón de amplitud unitaria y definiendo condición inicial cero para los estados, cada valor se denomina $S[k|k]$, donde se agrupan dependiendo del horizonte de control.

$$x[k|k] = G_m x[k-1] + H_m \tag{3}$$

$$S[k|k] = C_m x[k|k] \tag{4}$$

- Matriz Hessiana: Es el componente cuadrático que queda luego de ajustar la función objetivo a la forma de un problema de optimización de programación cuadrática.
- Matrices de restricciones: Para la programación cuadrática se define las restricciones del problema de optimización en forma de desigualdades lineales de la forma:

$$A\Delta u \leq b \tag{5}$$

- Matrices de respuesta libre: Se calcula para cada instante a lo largo del horizonte de predicción, usando las condiciones iniciales y manteniendo la condición de las entradas, incluyendo la acción de control calculada en la iteración anterior y la medida actual de las entradas de la perturbación $u_d(k|k)$. Cada valor se denomina $y_{libre}(k|k)$ que varía en cada iteración, debido a que puede depender de entradas externas. Esto hace necesario definir una matriz H_{md} que modele su efecto en el sistema y la condición inicial actual del modelo de predicción.

$$x[k|k] = G_m x[k-1|k] + H_m u[k-1|k] + H_{md} u_d[k|k] \tag{6}$$

$$y_{libre}[k|k] = C_m x[k|k] \quad (7)$$

- Cálculo del coeficiente lineal de la función objetivo: Se denomina f y está definida por la ecuación (8), que al trabajar con el modelo desacoplado incluye la variable de correlación *d(k|k)* para eliminar el error de estado estable generado por la incertidumbre del modelo de predicción. Donde *d(k|k)* es la diferencia entre la salida de la planta y la salida calculada por el modelo de predicción.

$$f = -2\left(\theta^T Q(r[k+i|k] - \hat{y}_{libre}[k+i|k])\right) \quad (8)$$

$$d[k|k] = y_p[k-1|k] - y_m[k-1|k] \quad (9)$$

Donde $y_p[k-1|k]$ es la medición de la salida en la planta y $y_m[k-1|k]$ el cálculo de la salida al final de la iteración anterior usando el modelo de predicción.

$$f = -2\left(\theta^T Q(r[k+i|k] - d[k|k] - \hat{y}_{libre}[k+i|k])\right) \quad (10)$$

- Cálculo de la acción de control: Se encuentra Δu usando la función quadprog que recibe como parámetros de entada las matrices H, f, A_c y b_c, devuelve un vector con todos los Δu calculados de dimensión $H_u x 1$. Del vector encontrado se toma el

primer valor, que corresponde al instante actual y se le suma el valor de la acción de control de la interacción anterior.

$$u[k|k] = u[k-1|k] + \Delta u[k|k] \quad (11)$$

- Condiciones iniciales: Se calculan después de encontrar $u[k|k]$, obteniendo el valor de $y_m\ (k|k)$ al usar el modelo de predicción y las entradas actuales.

$$y_m[k|k+1] = C_m x[k|k] + H_m u(k|k) \quad (12)$$

2. Controlador MPC

En el trabajo de investigación de maestría titulado "CONTROL PREDICTIVO ROBUSTO DE GLUCOSA EN PACIENTES CON DIABETES MELLITUS TIPO I: VALIDACION INSILICO" (Padilla Toloza, 2021), se realizó una estrategia para la dosificación automática de insulina. Esta estrategia está basada en un control realimentado para generar perfiles de glucosa saludables en pacientes con diabetes mellitus tipo I. El algoritmo de control predictivo programado se basa una estrategia de control predictivo con modelo desacoplado.

2.1 MPC con modelo desacoplado.

De acuerdo con (Padilla Toloza, 2021) se abordó el enfoque QDMC, donde la característica principal del algoritmo consiste en usar el modelo de predicción para generar sus propias

condiciones iniciales en cada iteración. Este controlador aplica realimentación de salidas, donde las entradas del controlador son: la salida del proceso $y_p[k]$ (Glucosa subcutánea G_{sc} en mg/sL), la referencia $r[k]$ también en mg/dL y la perturbación de carbohidratos $CHO[k]$ en mg; la salida del controlador será la acción de control $u[k]$(dosificación de insulina en $pmol/min$). A continuación, se explica como funciona el algoritmo de control.

El sistema recibe los parámetros de configuración (Periodo de muestreo T_s, el horizonte de predicción H_p, Horizonte de control H_u, Matrices de ponderación Q y R), adicionalmente los parámetros del paciente. Luego halla el modelo de predicción, condiciones iniciales, la respuesta forzada, la matriz hessiana y parte de las matrices de restricción.

3. Implementación estrategia de control

Para la implementación de la estrategia de control se definieron los requisitos mínimos utilizados por la estrategia de control diseñado como son la memoria y el tiempo de ejecución. A partir de los resultados se escogió un microcontrolador adecuado.

3.1 Requerimientos de procesamiento del controlador MPC desacoplado.

Para el controlador MPC desacoplado implementado en Matlab se realizaron pruebas para verificar la memoria y el tiempo de procesamiento usado por el algoritmo de control para un tiempo de 4 horas. Para esto, se utilizó la herramienta Matlab *profiler* que permite visualizar el tiempo total (total time), además de la memoria propia (*self memory*) utilizada por cada función en ejecución. Los valores obtenidos se presentan en la Tabla 1.

Tabla 1

Requerimientos por parte del controlador MPC desacoplado.

Función	**Memoria**	**Tiempo**
MPC_QDMC2stepImpl	1078.25 KB	14439.38 s
MPC_QDMC2resetImpl	317 KB	1119.319 s
Quadprog	58.5 KB	0.816 s

A partir de las exigencias computacionales de la estrategia de control predictivo diseñado en (Padilla Toloza, 2021), en la Tabla 2 se presentan algunos microcontroladores en el mercado, comparando la frecuencia de su procesador, memoria RAM, memoria flash y costo.

Tabla 2

Tipos de microcontroladores

Microprocesador	**Procesador**		**Memoria RAM**	**Memoria FLASH**	**Costo ($)**
	Frecuencia	Bits			
Arduino Uno	16MHz	32	2kB	32kB	25000
Arduino Mega	16MHz	32	8kB	256kB	50000
Tiva	80MHz	32	32kB	256kB	80000
Esp32	240MHz	32	520kB	4MB	35000
Raspberry P3	1.2 GHz	64	1Gb	4GB	200000
Rasberry P4	1.5Ghz	64	4Gb	16GB	350000

Para elaborar el sistema embebido se seleccionó la tarjeta ESP32 entre las opciones presentadas en la Tabla 2. Esto se debe a que cuenta con la memoria flash necesaria para la implementación del controlador, es decir 1.45375MB. Además, su procesador es 15 veces más rápido que el de una placa de Arduino. Gracias a esto, permite ejecutar operaciones matemáticas con números grandes de forma instantánea y finalmente, es de bajo costo. Estas características la hacen ideal para el tipo de control planteado en el presente trabajo.

El entorno de programación se llama ESP-IDF y está basado en el sistema Linux, donde el propio fabricante ofrece las herramientas para poder adaptarlo y utilizarlo en otros sistemas operativos como en este caso en el IDE ARDUINO. Una ventaja de este entorno es que se encuentran guías de programación oficial en la que se puede encontrar librerías, definición de funciones, ejemplos de uso e innumerables recursos que resultan imprescindibles para aprovechar al máximo la capacidad del dispositivo.

En la Tabla 3, se resaltan algunas características generales de la ESP32, que realiza las principales tareas de gestión y proceso de datos.

Tabla 3

Características ESP32

CPU	Xtensa single-/dual-core 32-bit
Voltaje	De operación: 2,3V - 3,6V
Pines I/O	Digitales: 16 (proveen salida PWM)
Interface GPIO	34 pines de propósito general I/O

Corriente DC	De salida: 0,5 A
ROM	448 KB
Memoria RAM	520
Memoria flash	16MB
Velocidad del reloj	160 MHz - 240 MHz
ADC	12- bits
DAC	8-bits

3.2 Implementación del controlador desacoplado en Arduino.

El microcontrolador se programó en el IDE de Arduino, donde se utilizó programación orientada a objetos, y se define una clase MPCQDMC2 para la creación de los objetos presentado en el apéndice A. Se establecieron los parámetros iniciales del controlador definidos en (Padilla Toloza, 2021) , mostrados en la Tabla 4. Dentro de esta clase se definieron los métodos que se ejecutarán y permitirán realizar el controlador MPC desacoplado.

Tabla 4

Valores óptimos para el controlador MPC_QDMC $\boldsymbol{H_p}$, $\boldsymbol{H_u}$, *Q y R*

Parámetros	Valores óptimos
Horizonte de predicción	280
Horizonte de control	1
Ponderación de seguimiento	1
Ponderación de esfuerzo de control	20000

El proceso iterativo que realiza el controlador es el siguiente:

- Se leen las entradas del proceso son la perturbación CHO[k], la salida del proceso $y_p[k]$ y la referencia $r[k]$.
- Se usan las condiciones iniciales $y_m[k-1|k]$ y $u[k-1|k]$, almacenadas en memoria y entrada de perturbación *CHO[k]* para calcular la respuesta libre y_{libre}. Siguiendo el esquema de control QDMC, ya que no se conoce como estará la perturbación del valor actual medido de esta, se mantendrá constante durante todo el horizonte de predicción H_P. Esto mismo aplica para el valor de la referencia $r[k]$.
- Se calcula el término de corrección $d[k|k]$ a partir de la ecuación (9).
- Se calcula el coeficiente lineal f de la función objetivo aplicando la ecuación término de corrección $d[k|k]$ a partir de la ecuación (10).
- Se calcula la matriz b_c correspondiente a las restricciones en $u[k]$ y la salida $y_p[k|k]$, que varían entre iteraciones
- Se calcula Δu usando una librería que ejecuta la función *quadprog* adaptada al entorno de Arduino y se calcula $u[k|k]$ aplicando la ecuación (11).
- Se calculan las nuevas condiciones iniciales, $y_m[k|k+1]$ usando el modelo de predicción mediante la ecuación (12) y $u[k|k+1]$ será la acción de control $u[k|k]$ que es enviada al proceso al final de la interacción.

4. Validación en Tiempo Real

La validación experimental del controlador se realiza utilizando una configuración Hardware *in the loop* (HIL). En la Figura 3 se muestra el proceso realizado para validar el controlador en tiempo real, donde se realiza la conexión entre la tarjeta DSPACE 1104, la cual tiene el modelo del sistema en tiempo continuo y los parámetros de los pacientes, con el microcontrolador ESP32, que recibe la señal de salida (glucosa subcutánea), el valor de referencia y la perturbación de carbohidratos, este ejecuta la ley de control predictivo y encuentra la acción de control que representa la dosificación de insulina, la cual se envía nuevamente al modelo. La conexión física se presenta en la Figura 4.

Figura 3

Esquema de simulación HIL en tiempo real

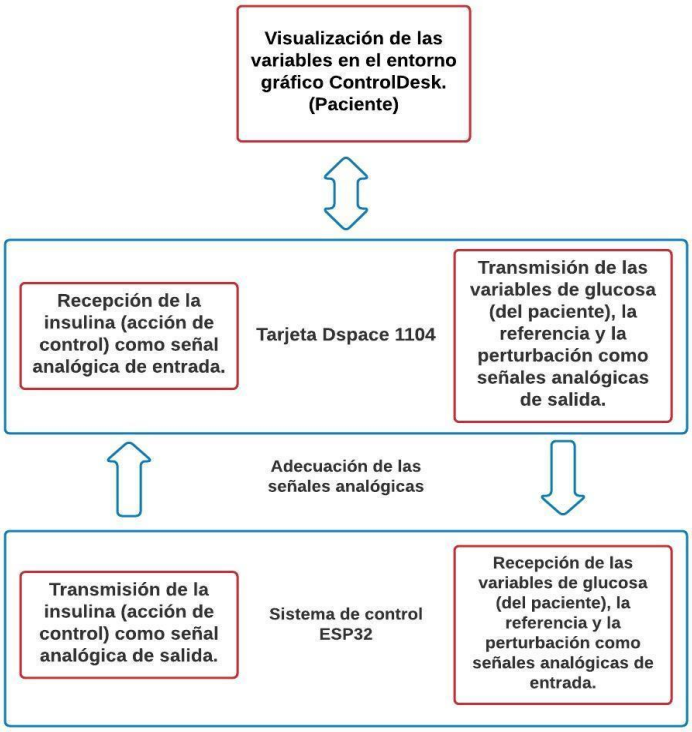
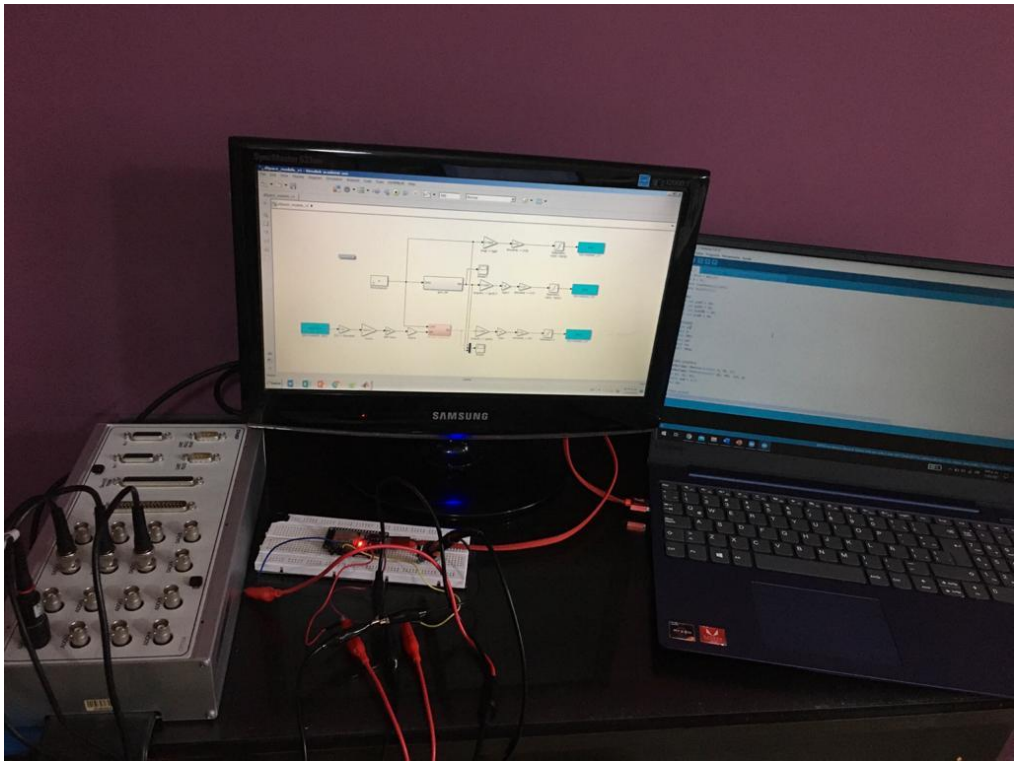


Figura 4

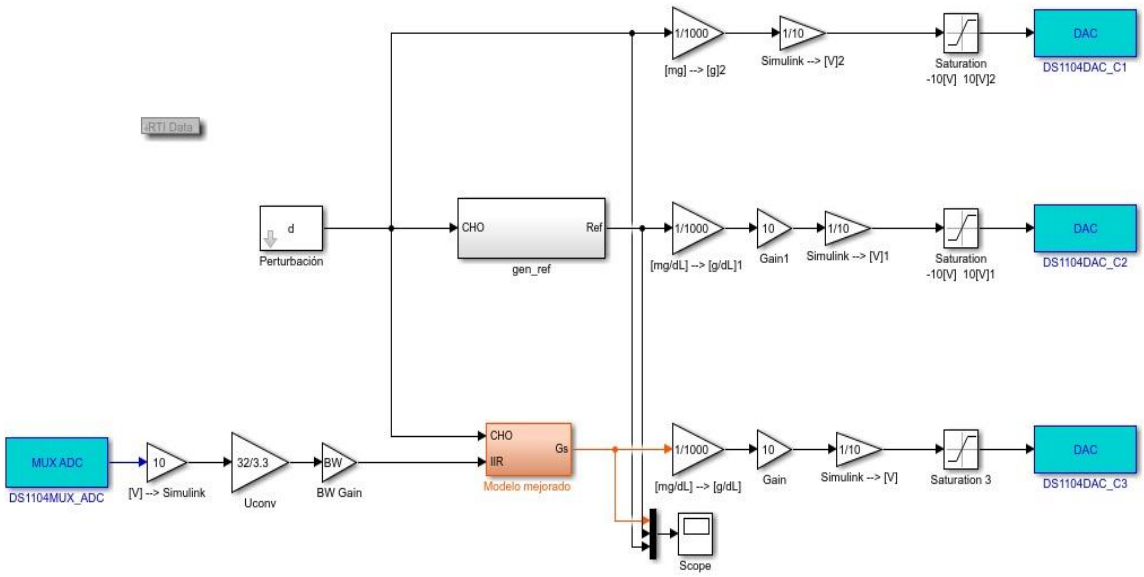
Conexión física para la implementación HIL



Para la implementación se utiliza la tarjeta DS1104, sus especificaciones se presentan en el apéndice B. En la Figura 5 se presenta el diagrama de bloques utilizado en Simulink, que consiste en un bloque de lectura para la acción de control y tres bloques de escritura analógicos de la tarjeta DS1104 para las variables de referencia, perturbación y glucosa sensada. Es necesario tener en cuenta que por protección interna de la tarjeta al entrar la señal esta se divide en 10 V y al enviar la señal se multiplica por 10V.

Figura 5

Modelo Implementado en la tarjeta DS1104



Para la generación del bloque de la referencia de la Figura 5 , se buscó que el controlador pueda seguir la curva generada y que esta represente una condición saludable para la paciente in silico.

La función de transferencia se obtiene del modelo que describe la relación entre la ingesta de alimentos y la glucosa plasmática de un paciente como respuesta al bolo óptimo de insulina calculado a partir de CR (*Carb Ratio*) ecuación 13.

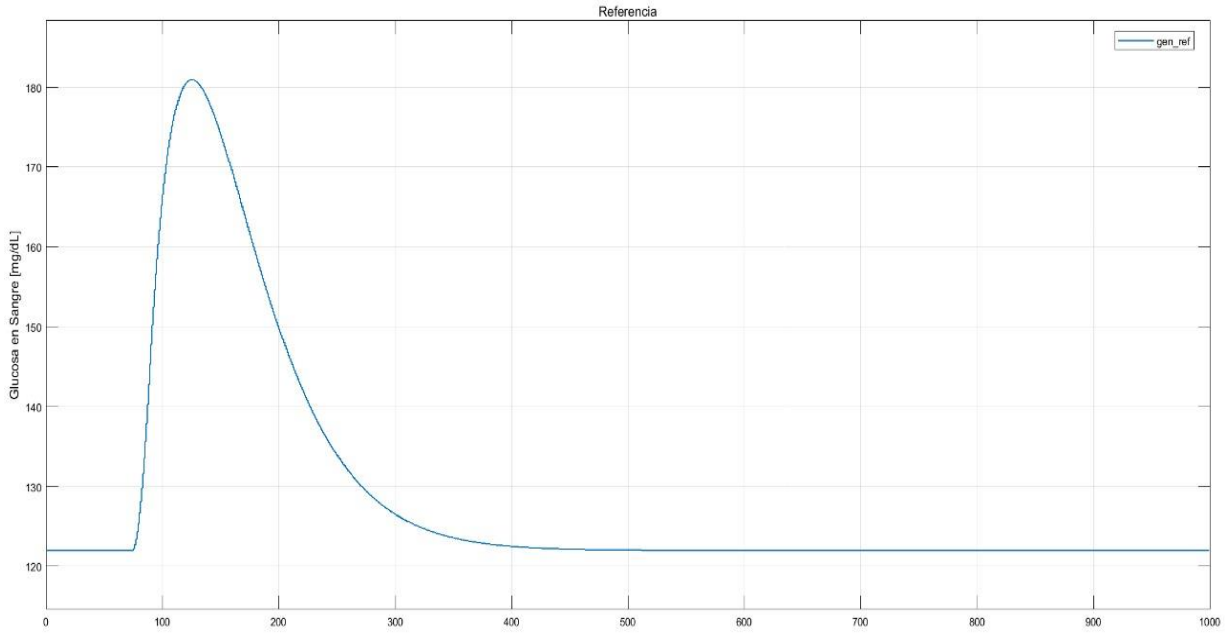
$$CR = \frac{Ingesta\ de\ CHO}{Bolus\ optimo} \tag{14}$$

A partir del simulador T1DM se obtiene la curva de glucosa del paciente 1, ante una entrada de carbohidratos de 60g (*CHO*) y el bolo óptimo encontrado de tal manera que, iniciando de su valor basal, la concentración de glucosa se encuentre entre el 85% y 110% de dicho valor, al cabo

de 3 horas. Teniendo en cuenta que su valor mínimo no se encuentre por debajo del valor de hipoglucemia y que su valor máximo se encuentre cerca de 200(mg/dL). Posteriormente usando la toolbox "*SystemIdentification*" de Matlab se halló la función de transferencia aproximada para generar la curva de referencia de glucosa. En la Figura 6 se muestra el resultado de la referencia para el paciente Adult_001.

Figura 6

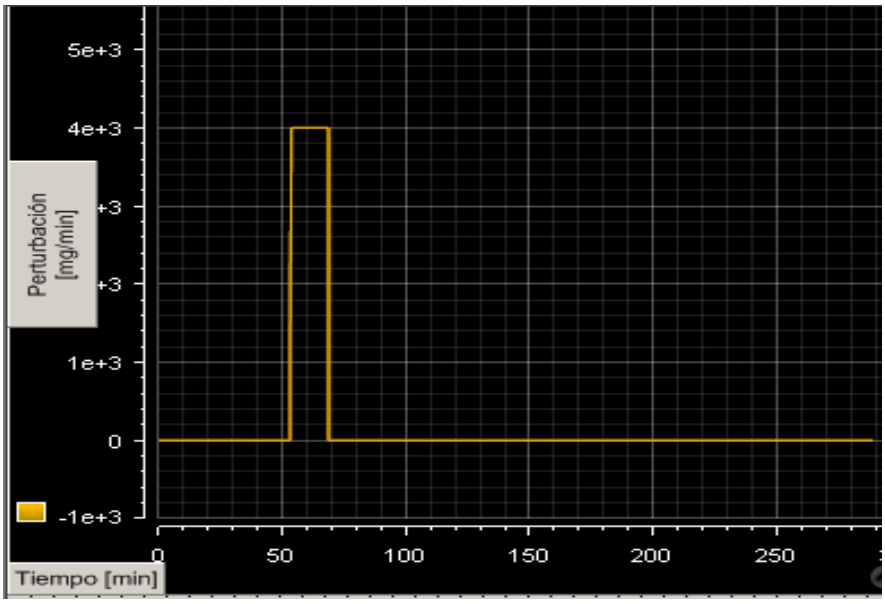
Referencia del Adult_001



En cuanto a la ingesta de alimentos se le suministra 4000 *mg/min* de carbohidratos durante 15 minutos a las 7a.m. En la Figura 7 se presenta su comportamiento. Se utiliza esta misma ingesta para la muestra de validación de diez adultos.

Figura 7

Ingesta de alimentos de 60[g] de carbohidratos a las 7 a.m.



4.1 Resultados de la validación HIL

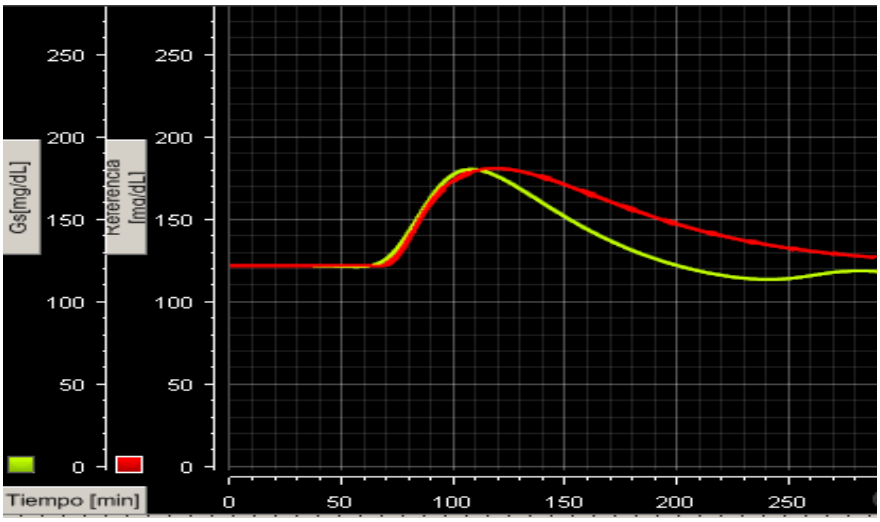
Para los siguientes resultados se utilizó un periodo de muestreo de 1 minuto, para representar aproximadamente 15 muestras de una ingesta de alimentos, que dura 15 minutos. Además, para generar la ley de control predictivo se necesita que el microprocesador sea capaz de calcular la acción en un tiempo maximo al 10% del periodo de muestro (Padilla Toloza, 2021). De esta forma, se encontró un tiempo de procesamiento maximo por iteración de 40 [ms].

Asi mismo, se estableció un tiempo de simulación de 360 minutos, correspondiente a 6 horas para cada paciente, ya que con este tiempo el comportamiendo de la glucosa en la sangre de una persona se regula después recibir una ingesta de alimentos.

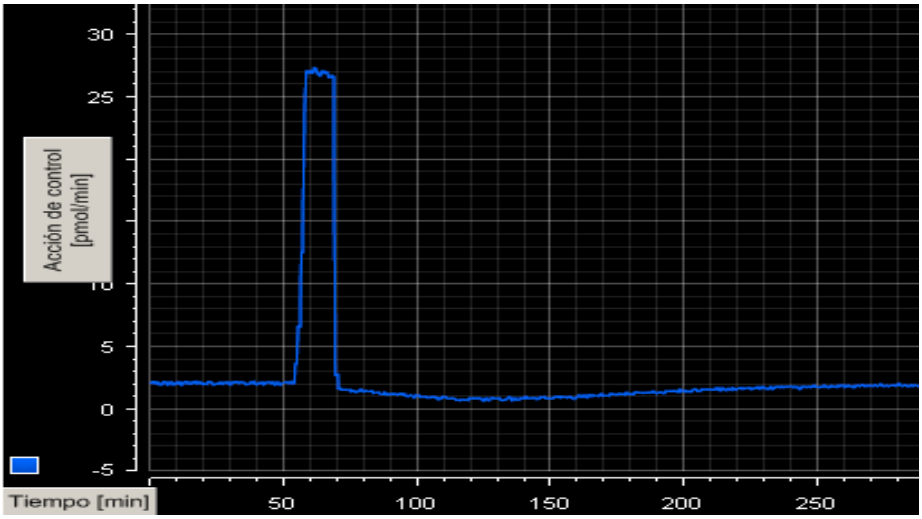
4.1.1 Resultados del controlador MPCQDMC2 para diferentes pacientes

Figura 8

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 1



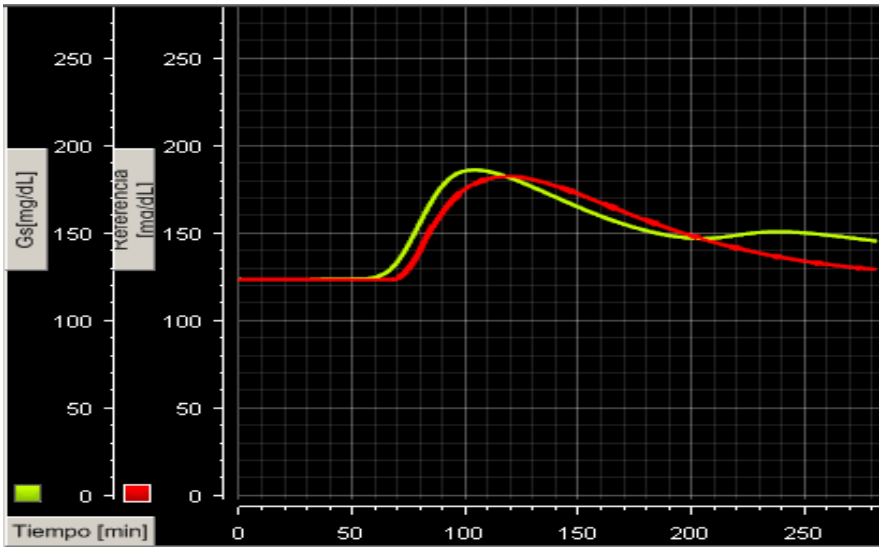
(a) Perfil de glucosa.



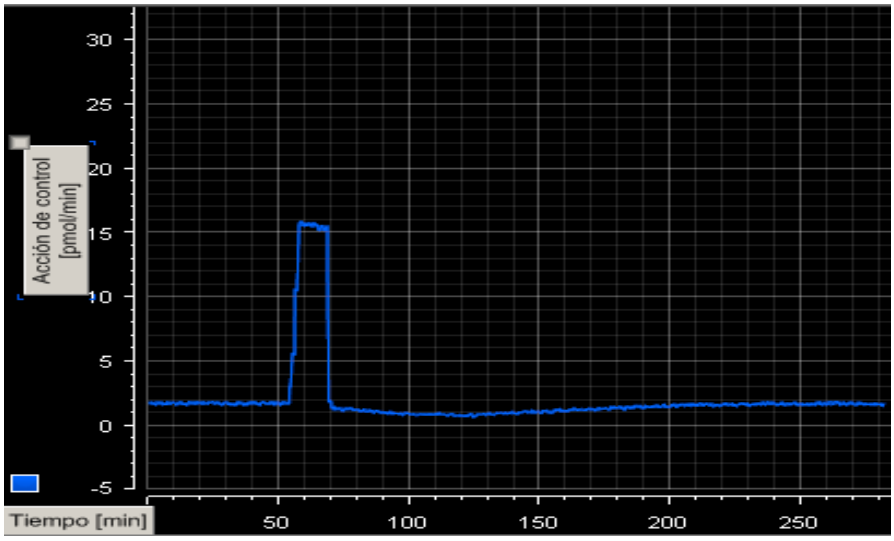
(b) Dosificación de insulina.

Figura 9

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 2



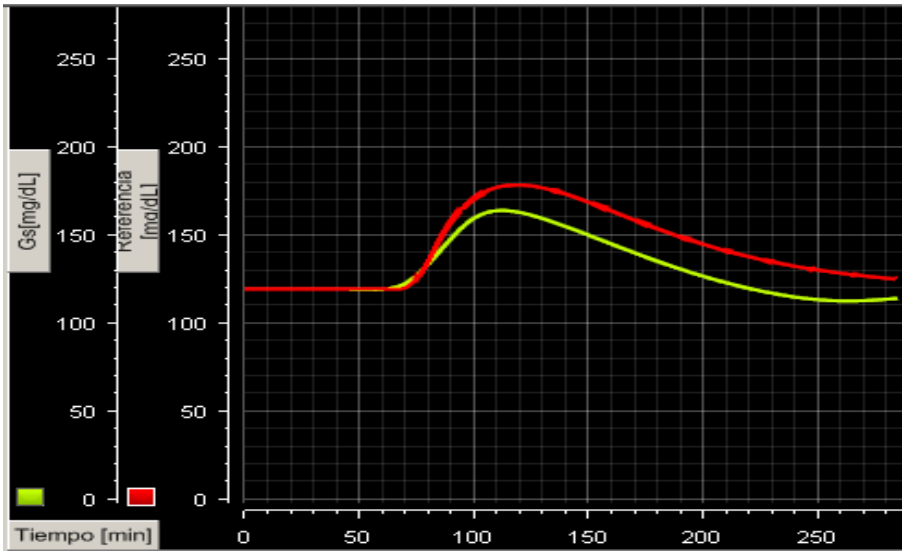
(a) Perfil de glucosa.



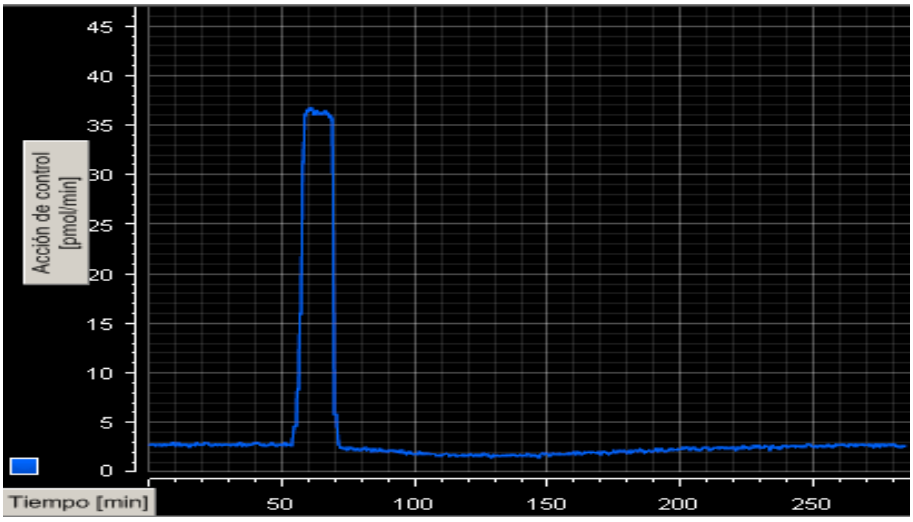
(b) Dosificación de insulina

Figura 10

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 3



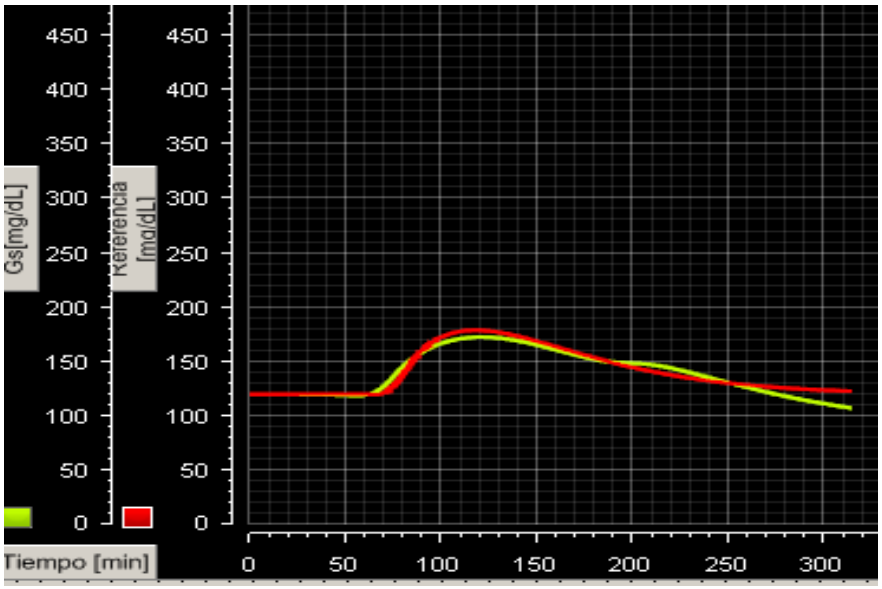
(a) Perfil de glucosa.



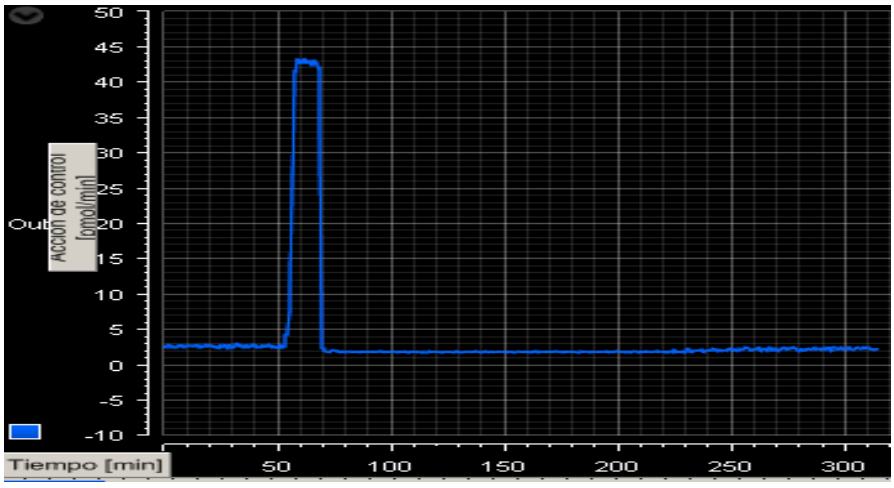
(b) Dosificación de insulina.

Figura 11

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 4



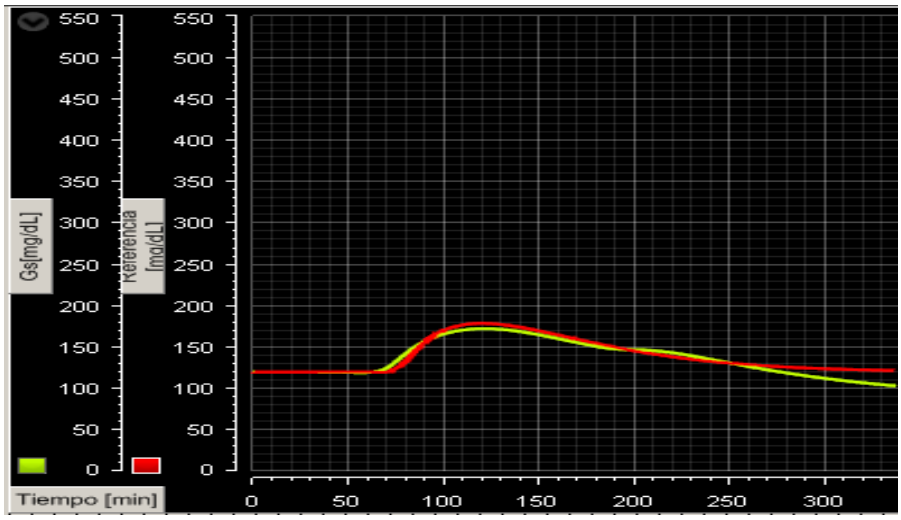
(a) Perfil de glucosa.



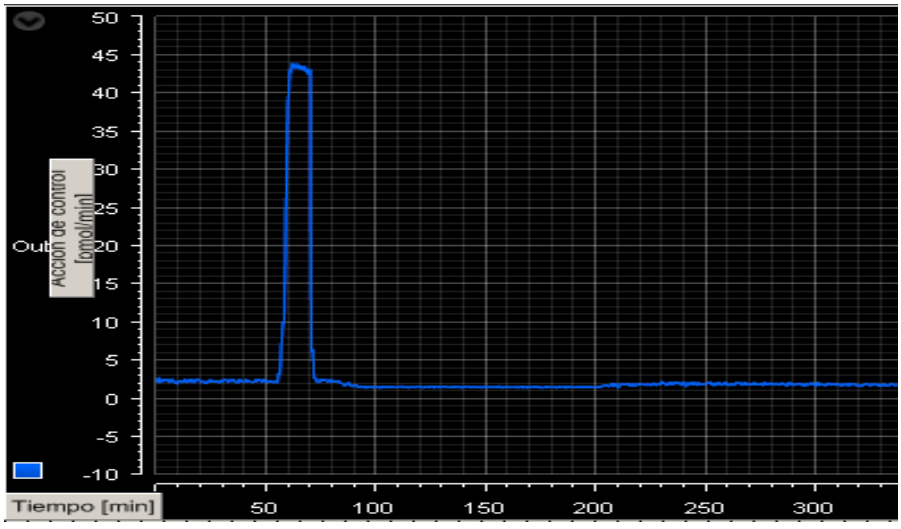
(b) Dosificación de insulina.

Figura 12

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 5



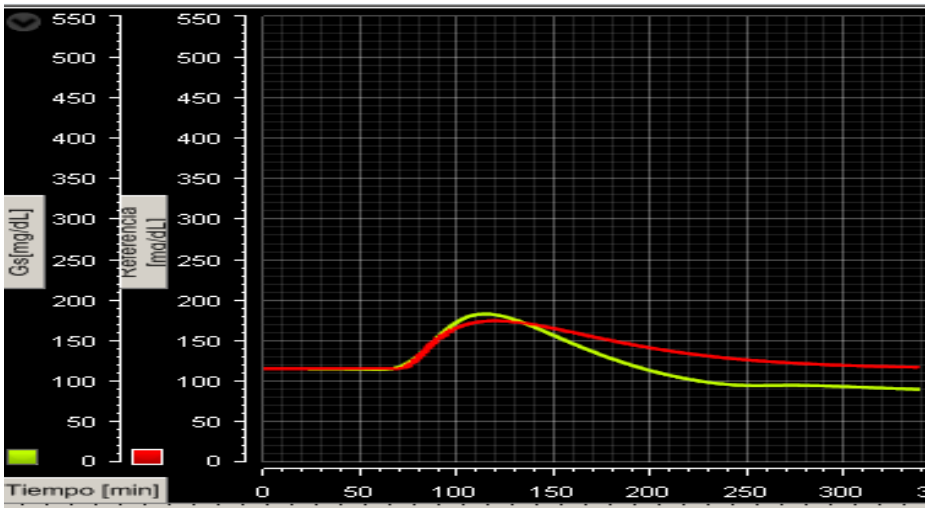
(a) Perfil de glucosa.



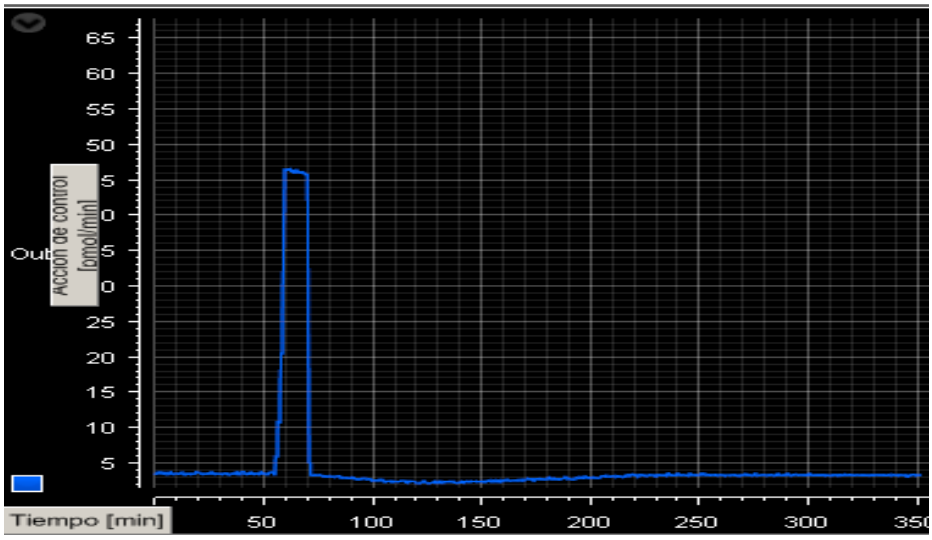
(b) Dosificación de insulina.

Figura 13

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 6



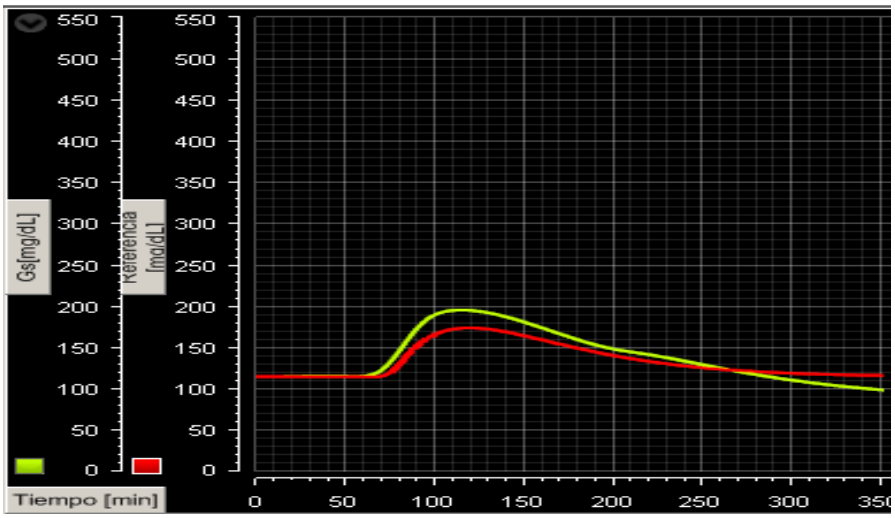
(a) Perfil de glucosa.



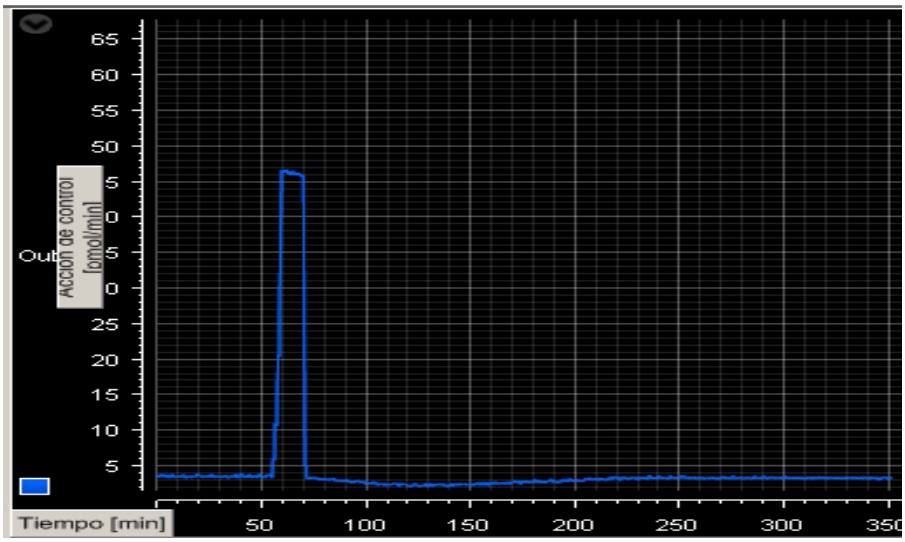
(b) Dosificación de insulina.

Figura 14

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 7



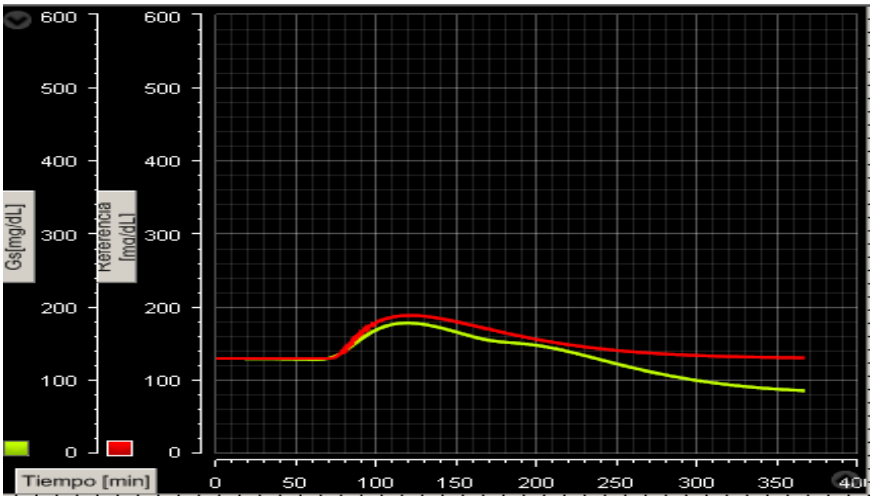
(a) Perfil de glucosa.



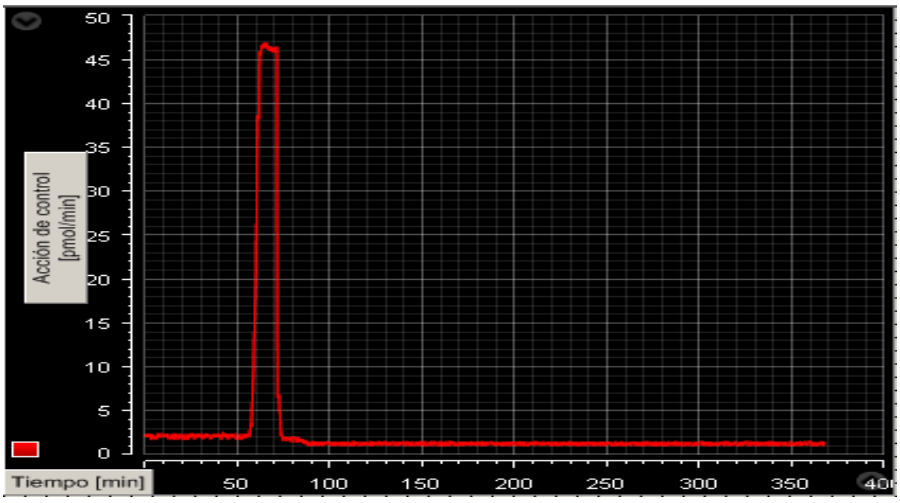
(b) Dosificación de insulina.

Figura 15

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 8



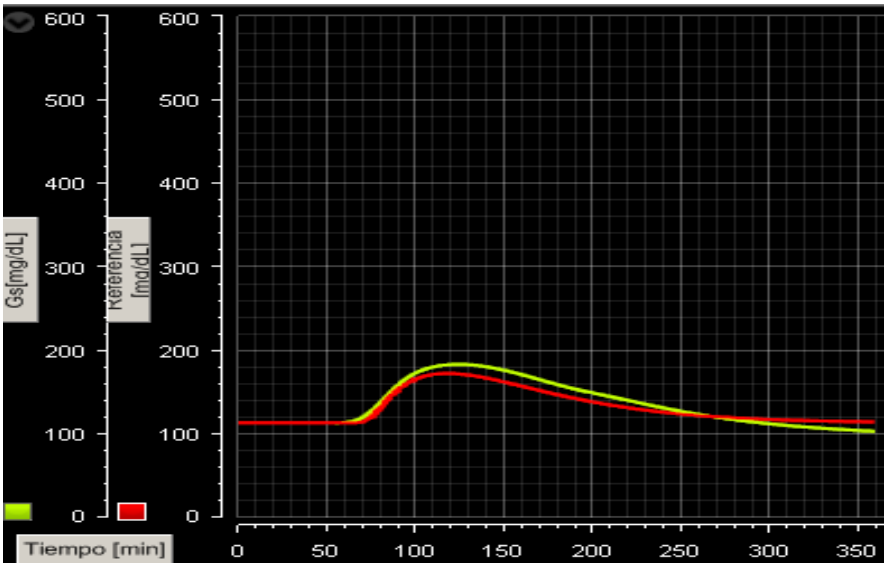
(a) Perfil de glucosa.



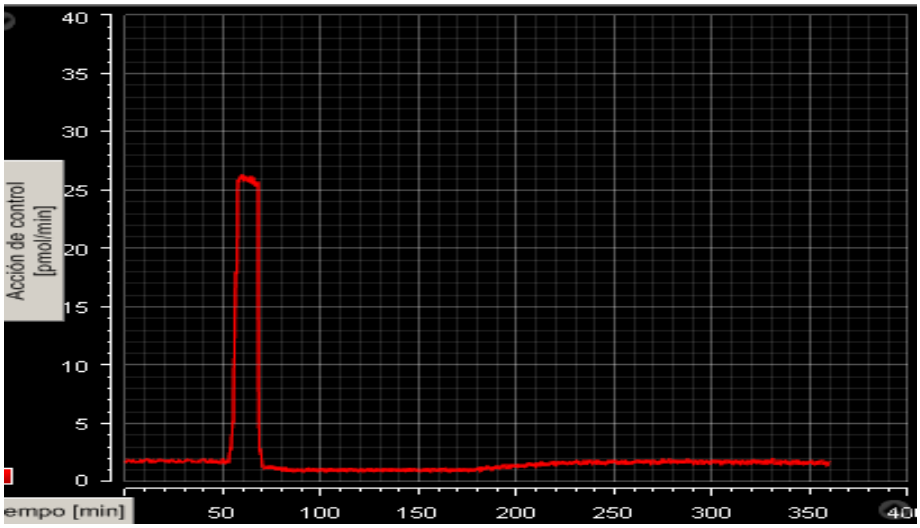
(b) Dosificación de insulina.

Figura 16

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 9



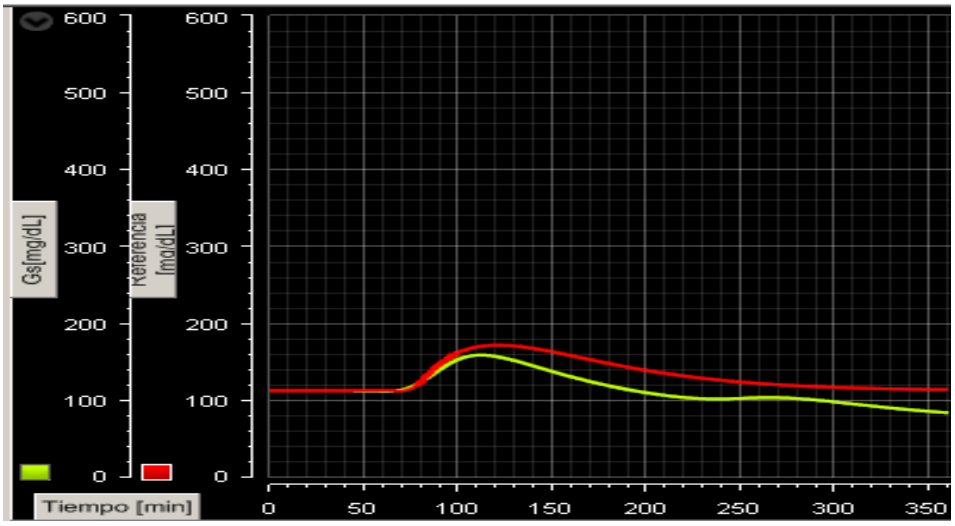
(a) Perfil de glucosa.



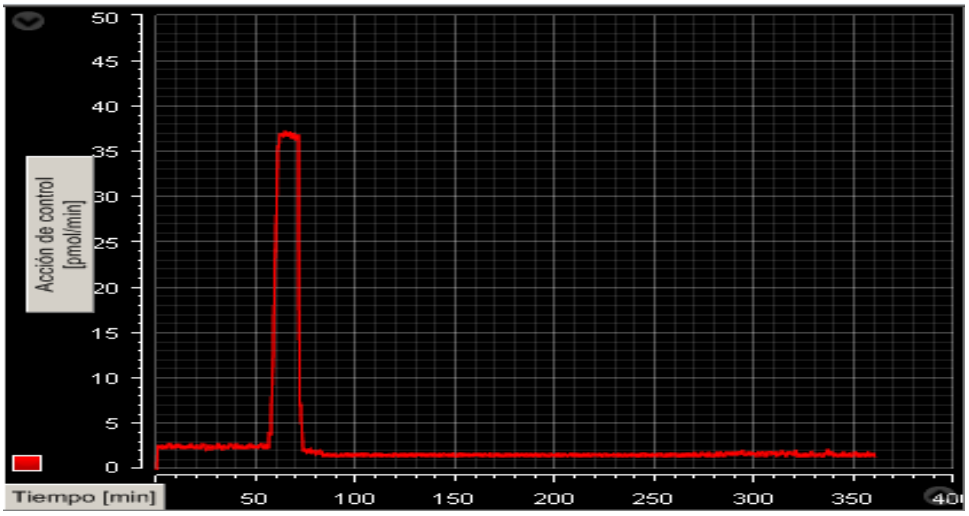
(b) Dosificación de insulina.

Figura 17

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] de carbohidratos suministrados a las 6a.m para adulto 10



(a) Perfil de glucosa.



(b) Dosificación de insulina.

En las figuras 8 a 17 se observan los resultados del controlador MPCQDCM2 en validación in silico con su referencia y su respectiva acción de control. De esta forma se determina que los

valores máximos y mínimos de glucosa no están por debajo del nivel de hipoglucemia (70 mg/dL) ni superan el nivel de hiperglucemia (200 mg/dL). Así mismo se aprecia que el adulto 10 presenta un nivel de glucosa en la sangre cercano al límite inferior mientras que el adulto 7 se aproxima al nivel superior. Por consiguiente, la aplicación realizada de insulina ayudó a regular el nivel basal de glucosa en los pacientes.

Adicionalmente en la Tabla 5, se presentan los valores para el criterio de rendimiento integral donde se observa que el paciente 3 presenta menor error ITAE, IAE e ISE mientras que el paciente 9 tiene valor más alto en los errores ya mencionados. Estos errores representan el comportamiento dinámico del sistema.

Tabla 5

Error de seguimiento de referencia del controlador MPCQDMC2 durante la validación HIL

Paciente	**ITAE** $\left[\frac{mg}{dL} \cdot min^2\right]$	**IAE** $\left[\frac{\boldsymbol{mg}}{\boldsymbol{dL}} \cdot \boldsymbol{min}\right]$	**ISE** $\left[\left(\frac{\boldsymbol{mg}}{\boldsymbol{dL}}\right)^2 \cdot \boldsymbol{min}\right]$
Adulto 3	$\mathbf{6.2058x\ 10^5}$	$\mathbf{3.2796x\ 10^3}$	$\mathbf{4.5705x\ 10^4}$
Adulto 4	$\mathbf{1.8323x\ 10^6}$	$\mathbf{7.0342x\ 10^3}$	$\mathbf{2.2563x\ 10^5}$
Adulto 5	$\mathbf{1.6299x\ 10^6}$	$\mathbf{6.5626x\ 10^3}$	$\mathbf{1.7672x\ 10^5}$
Adulto 6	$\mathbf{1.5880x\ 10^6}$	$\mathbf{6.4826x\ 10^3}$	$\mathbf{1.7316x\ 10^5}$
Adulto 7	$\mathbf{1.6007x\ 10^6}$	$\mathbf{6.4337x\ 10^3}$	$\mathbf{1.8202x\ 10^5}$
Adulto 8	$\mathbf{1.4270x\ 10^6}$	$\mathbf{5.388x\ 10^3}$	$\mathbf{1.4923x\ 10^5}$
Adulto 9	$\mathbf{1.8972x\ 10^6}$	$\mathbf{7.2058x\ 10^3}$	$\mathbf{2.4645x\ 10^5}$
Adulto 10	$\mathbf{1.4877x\ 10^6}$	$\mathbf{6.5618x\ 10^3}$	$\mathbf{1.6347x\ 10^5}$

4.1.2 Comparación entre controlador MPCQDMC2 diseñado con el implentado en el microcontrolador

Con el fin de analizar el desempeño del controlador MPCQDMC2 implementado en el microcontrolador se realiza la comparación de los resultados obtenidos con el controlador diseñado en (Padilla Toloza, 2021).

Figura 18

Comparación entre el controlador MPCQDMC2 diseñado e implementado para una ingesta de alimentos de 60 [gr] para el adulto 1

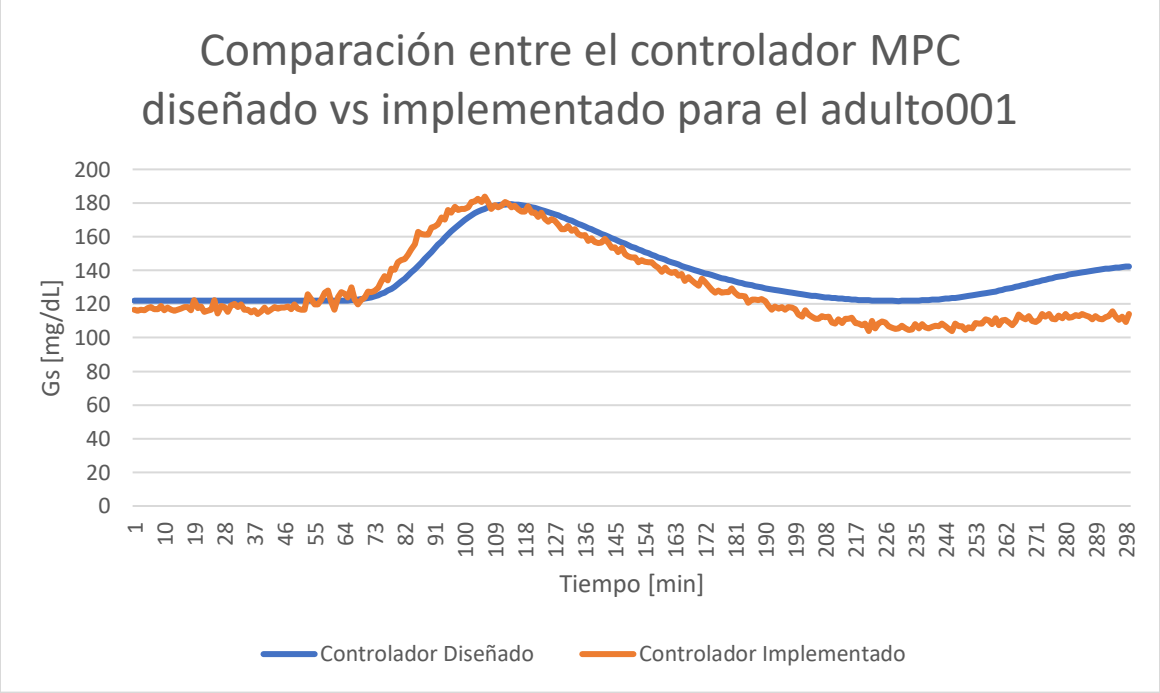
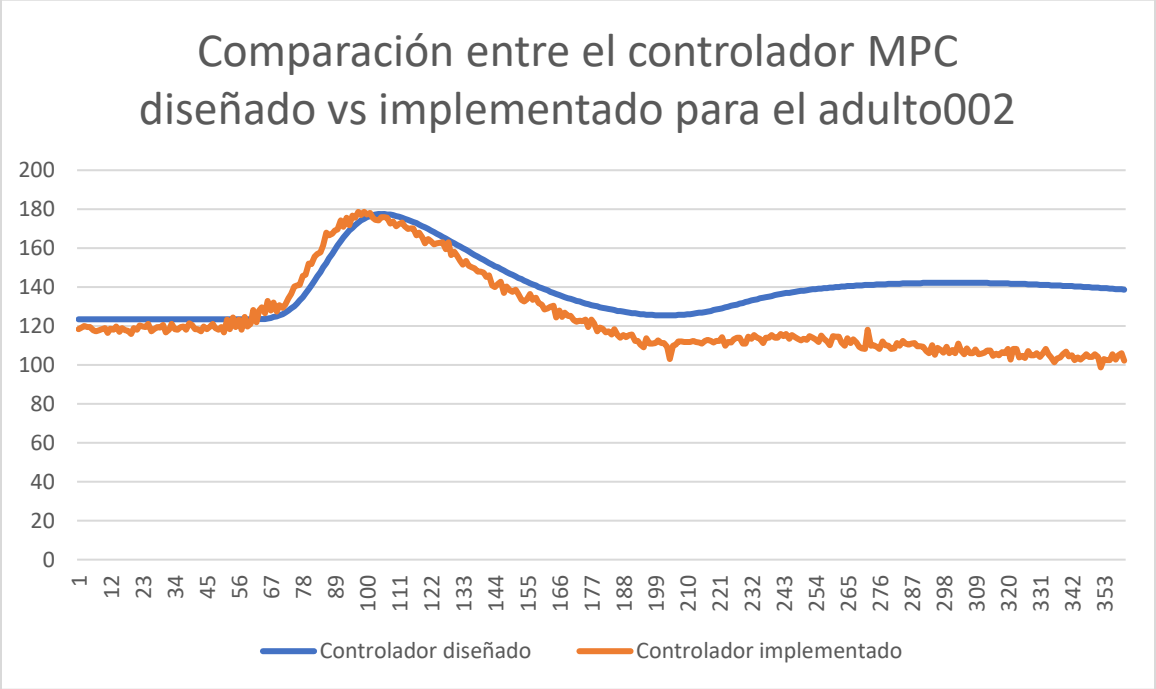


Figura 19

Comparación entre el controlador MPCQDMC2 diseñado e implementado para una ingesta de alimentos de 60 [gr] para el adulto 2



En las Figura 18 y 19 se observa que la curva de salida del perfil de glucosa es similar para una ingesta de alimentos de 60 [gr] para 5 horas debido a que presenta un error RMSE de 8.2 y 9.3 para cada paciente respectivamente, lo que permite determinar que la señal de salida del controlador implementado realiza una aproximación adecuada al controlador diseñado, además el valor máximo de glucosa en todas pruebas esta entre 180 mg/dL y 190 mg/dL.

Por otro lado, se presenta un error de sintonización entre el microcontrolador y el periférico de la DSPACE 1104 debido a los tiempos de conexión, donde la tarjeta DSPACE realiza la simulación del modelo y las respectivas señales de salida al momento de ejecutar el programa en

el computador, mientras que para el microcontrolador ESP32 se necesita de una acción manual para iniciar el proceso lo que genera un retardo en la ingesta de alimentos, y por lo tanto la acción de control se ve afectada inyectando la dosis de insulina al paciente en tiempo diferente al deseado, sin embargo, esto no genera una modificación en la respuesta del perfil de glucosa, solo en los tiempos de simulación.

Tabla 6

Error de seguimiento de referencia del controlador MPCQDMC2 diseñado

Paciente	**ITAE** $\left[\frac{mg}{dL} \cdot min^2\right]$	**IAE** $\left[\frac{mg}{dL} \cdot min\right]$	**ISE** $\left[\left(\frac{mg}{dL}\right)^2 \cdot min\right]$
Adulto 1	$1.0474x\ 10^6$	$4.6924x\ 10^3$	$8.3941x\ 10^4$
Adulto 2	$1.01632x\ 10^6$	$5.5581x\ 10^3$	$1.00299x\ 10^5$

Tabla 7

Error de seguimiento de referencia del controlador MPCQDMC2 durante la validación HIL

Paciente	**ITAE** $\left[\frac{mg}{dL} \cdot min^2\right]$	**IAE** $\left[\frac{mg}{dL} \cdot min\right]$	**ISE** $\left[\left(\frac{mg}{dL}\right)^2 \cdot min\right]$
Adulto 1	$1.0834x\ 10^7$	$5.0025x\ 10^3$	$1.00278x\ 10^5$
Adulto 2	$1.2775x\ 10^6$	$6.2058x\ 10^3$	$1.5406x\ 10^5$

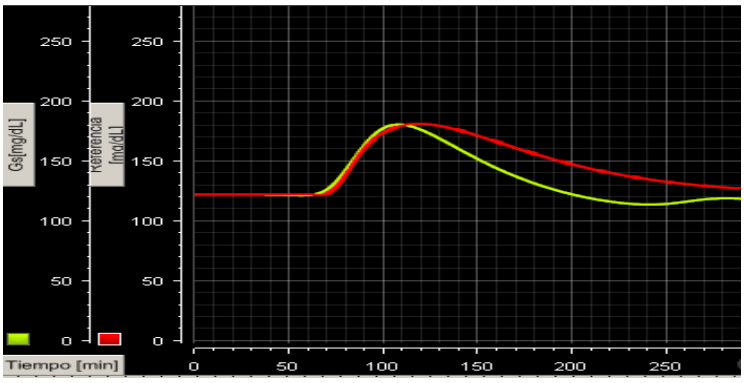
En la Tabla 7, se presentan los errores de criterio de rendimiento integral para el controlador implementado en la ESP32, mientras que en la Tabla 6, se muestran estos mismo criterios para el controlador diseñado en (Padilla Toloza, 2021). Se observa que en el caso del microcontrolador

los valores presentan una variación pequeña con respecto al diseñado. Esto puede deberse a la perdida de información ocasionada por la diferencia entre la resolución de los DAC y ADC de la ESP32 con la DS1104.

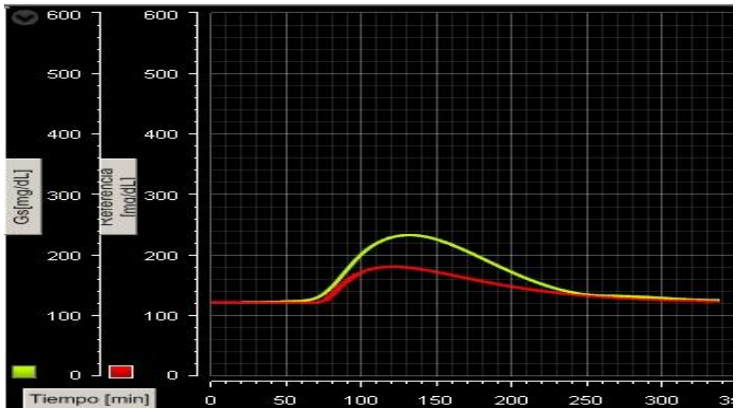
4.1.3 Comparación s del controlador MPCQDMC2 con un controlador PID.

Figura 20

Resultado del controlador MPCQDMC2 y el controlador PID para el paciente 1



(a) Perfil de glucosa.



(b) Dosificación de insulina.

En la Figura 20 se observa la comparación entre el perfil de glucosa del controlador PID y el MPCQDMC para el paciente 1, donde se observa que el controlador PID no tiene la capacidad de realizar un seguimiento adecuado a la referencia después de realizarse la ingesta de alimento lo que causa mayores niveles de glucosa en la sangre.

Tabla 8

Errores de seguimiento entre MPCQDMC2 y PID para el adulto 1

Controlador	**ITAE** $\left[\frac{mg}{dL} \cdot min^2\right]$	**IAE** $\left[\frac{mg}{dL} \cdot min\right]$	**ISE** $\left[\left(\frac{mg}{dL}\right)^2 \cdot min\right]$
MPCQDMC2	$1.0834x\ 10^7$	$5.0025x\ 10^3$	$1.00278x\ 10^5$
PID	$1.4112x\ 10^6$	$1.0149x\ 10^4$	$5.8959x\ 10^5$

En la Tabla 8 se realiza una comparación entre los criterios de comportamiento dinámico de los controladores donde se aprecia que el PID presenta menor error ITAE, mientras que el otro controlador presenta menor valor ISE e IAE, debido a esto es posible determinar el MPCQDMC2 permite una mejor corrección para errores grandes y pequeños de la salida con respecto al valor basal.

4.1.4 Resultados del controlador MPCQDMC2 para dos pacientes utilizando dos ingesta de alimentos.

Figura 21

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] y 50[g] de carbohidratos suministrados a las 8a.m y 12 p.m. respectivamente para paciente 1

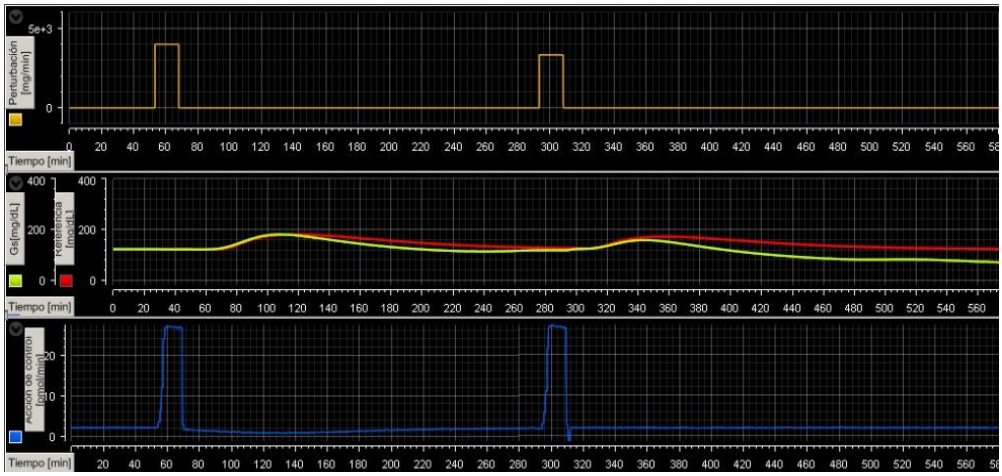
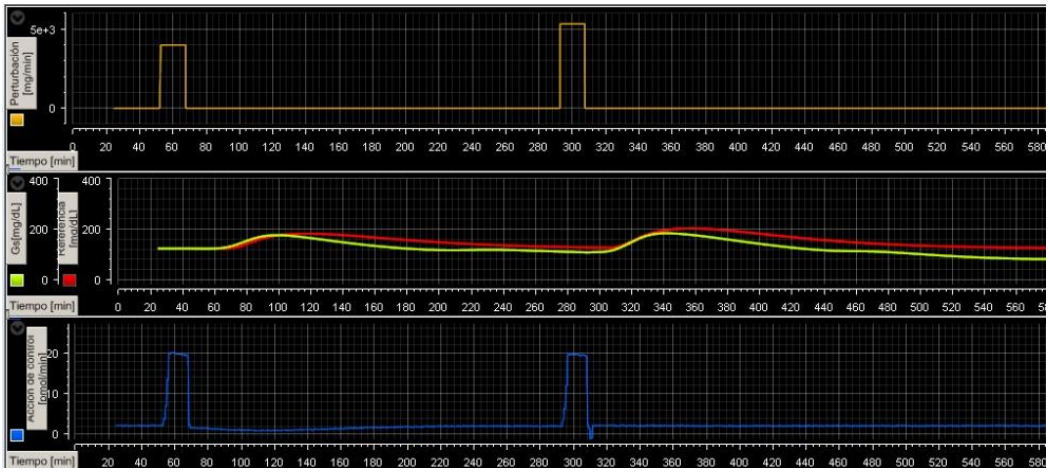


Figura 22

Gráfica generada por el controlador MPC con modelo desacoplado con una ingesta de alimento de 60 [g] y 80[g] de carbohidratos suministrados a las 8a.m y 12 p.m. respectivamente para paciente 2



Para los resultados observados en las Figura 21 y 21 , se utilizó un tiempo de simulación de 720 minutos que corresponden a 12 horas. Esto con el fin de simular dos ingestas de alimentos, de 4000 mg/min y 5333.33 mg/min , durante 15 minutos, a las 8.am y 12 p.m., respectivamente. De esta forma se aprecia un resultado óptimo de respuesta para el perfil de glucosa donde esta permanece dentro de los límites de una persona sana.

5. Conclusiones

Se pudo validar experimentalmente un controlador predictivo en un esquema de *Hardware in the Loop* (HIL), donde el microcontrolador seleccionado para programar el controlador cumplió con las exigencias computacionales exigidas del algoritmo, debido a que la memoria utilizada por el controlador corresponde al 36.34% de la memoria total disponible en la ESP32. Además, en la implementación se encontró un tiempo de procesamiento máximo de 40ms lo que equivale al 0.06% del periodo de muestreo. Esto significa que el microprocesador tiene la capacidad de realizar los cálculos de la ley de control de manera óptima, ya que es necesario un tiempo de procesamiento menor al 10% del periodo de muestreo.

Durante el desarrollo de este proyecto se consiguió programar la estrategia de control predictivo en la unidad de procesamiento digital seleccionada, además se logró validar en tiempo real usando la herramienta *Hardware in the loop* obteniendo resultados similares a los hallados en (Padilla Toloza, 2021). Sin embargo, existen ciertas diferencias en los perfiles de glucosa sensada en los pacientes entre el controlador validado con el diseñado. Esto puede ser debido a que el rango de tensión de salida de los DAC de la ESP32 es 0v a 3.3v, mientras que la entrada analógica en la DSPACE 1104 permite valores entre 0v a 10v, lo que genera una pérdida de 6.7 v en la transmisión de la información además del ruido presente en los DAC y ADC en el periférico.

Al realizar la validación entre el controlador PID y el MPCQDMC2 se encontraron diferencias significativas en la respuesta de perfil de glucosa, debido a que el valor máximo alcanzado por el PID supera los 200 mg/dL correspondientes al nivel superior de glucosa para una persona sana

ocasionando episodios más largos de hiperglucemia. Adicionalmente, el controlador PID presenta un mayor error de seguimiento, dado que este tiene en cuenta el instante actual para la acción de control mientras que el MPCQDMC2 trabaja con una predicción.

6. Recomendaciones

- Implementar el controlador MPC desacoplado en un microcontrolador que tenga los mismos DAC y ADC que el periférico para que no hay pérdida de resolución.

- Mediante el desarrollo de un filtro reducir el ruido de la comunicación en HIL, debido a que el ruido en la insulina inyectada causa que los valores basales de insulina tengan error, afectando el establecimiento de la glucosa en su valor basal.

- Validar en tiempo real algoritmo de control MPC_RPI diseñado en el trabajo de investigación (Padilla Toloza, 2021).

Referencias Bibliográfica

Bordóns, C & Rodríguez, D. (2005). Análisis y control de sistemas en espacio de estado, identificación de sistemas, control adaptativo, control predictivo. *Depto. de Ingeniería de Sistemas y Automática.* http://www.esi2.us.es/~danirr/apuntesIC4.pdf.

Cho, N. H., & Williams, R. (2019). Atlas De La Diabetes De La FID. In *International Diabetes Federation* (Novena edi). http://www.idf.org/sites/default/files/Atlas-poster-2014_ES.pdf

Maciejowski, J. M. (2000). *Predictive Control with Constraints*. Pearson Education. https://dlscrib.com/queue/maciejowski-predictive-control-with-constraints_58e14d80dc0d601b1b8970ec_pdf?queue_id=598edbd8dc0d600f15300d18

Mejía, C., Sandí, N., & Salazar, N. (2020). Diabetes mellitus tipo I: retos para alcanzar un óptimo control glicémico. *Revista Ciencia & Salud: Integrando Conocimientos.* Volumen (4). http://revistacienciaysalud.ac.cr/ojs/index.php/cienciaysalud/article/view/183/246

OMS. (2020). *Diabetes*. Organización Mundial De La Salud. https://www.who.int/es/news-room/fact-sheets/detail/diabetes

Padilla Toloza, D. A. (2021). *Control Predictivo Robusto de Glucosa en Pacientes con Diabetes Mellitus Tipo I: Validación in Silico*.

Visentin, R., Campos, E., Schiavon, M., Lv, D., Vettoretti, M., Breton, M., Kovatchev, B., Dallaman, C., & Cobelli, C. (2018). The UVA/Padova type I diabetes simulator goes from single meal to single day. *Journal of Diabetes Science and Technology*. https://journals.sagepub.com/doi/10.1177/1932296818757747

Apéndices

APENDICE A. Controlador MPCQDMC2 programado en el IDE de Arduino en programación orientada a objetos.

```
#ifndef MPCQDMC2_h
#define MPCQDMC2_h

#include "Arduino.h"
#include "QuadProg.h"

class MPCQDMC2 //CLASE
{
  public:
    MPCQDMC2();
    const int Ts = 1;
    const int Hp = 280;
    const int Hu = 1;
    const int Q = 1;
    const int R = 20000;
    double xm[13][1];
    double ym[1][1];
    double u;
    double theta[280][1];
    double H[1][1];
    double P;
    int deltaumax;
    int deltaumin;
    int umax;
    int umin;
    int ymax;
    int ymin;
    double Ac[4][1];
    double b1[2][1];
    double BWu;
    const double Gm[13][13];
    const double Hm[13][3];
    const int Cm[1][13];
    const int Dm[1][3] = {{0, 0, 0}};
    double stepImpl(double y, double r, double CHO);
    void resetImpl();
    //-------------------------------------------
          private:
      double xf[13][281]={};
      double yf[280][1];
};
#endif
```

```
#include "Arduino.h"
#include "MPCQDMC2.h"
#include <MatrixMath.h>
#include <iostream>
#include <sstream>
#include <string>
#include "QuadProg.h"

MPCQDMC2::MPCQDMC2()
{
}

void MPCQDMC2::resetImpl()
{
  double x0[13][1] = {{216.6795},
    {86.8199},
    {6.4165},
    {3.0324},
    {0},
    {0},
    {0},
    {105.7348},
    {105.7348},
    {0},
    {85.09527},
    {84.4699},
    {122.0211}
  };
  ym[0][0] = x0[12][0];
  for (int i = 0; i < 13; i++)
  {
    xm[i][0] = x0[i][0];
  }
  BWu = 79.7963;
  u = 1.5336;
  //x
  double x[13][1] = {};
  double S[Hp][1] = {};
  double xb[1][1];


  for (int m = 0; m < Hp; m++)
  {
    double xa[13][1] = {};
    for (int i = 0; i < 13; i++)
```

```
    {
      for (int k = 0; k < 13; k++)
      { xa[i][0] = xa[i][0] + Gm[i][k] * x[k][0];

      }

    }

    for (int f = 0; f < 13; f++)
    {
      x[f][0] = xa[f][0] + Hm[f][0];
    }
    //S
    for (int i = 0; i < 1; i++)
    { for (int j = 0; j < 1; j++)
      { xb[i][j] = 0;
        for (int k = 0; k < 13; k++)
        { xb[i][j] = xb[i][j] + Cm[0][k] * x[k][0];
        }
      }
    }
    S[m][0] = xb[m][0];
  }

  // Theta
  double theta0[Hp][Hu] = {};
  for (int j = 0; j < Hp; j++)
  {
    for (int i = 0; i < j; i++)
    {
      theta0[j][i] = S[j - i][0];
    }
  }

  for (int i = 0; i < Hp; i++)
  {
    for (int j = 0; j < Hu; j++)
    {
      theta[i][j] = theta0[i][j];
    }
  }

  double trastheta[Hu][Hp];
```

```
for (int i = 0; i < Hp; i++)
{
  for (int j = 0; j < Hu; j++)
  {
    trastheta[j][i] = theta[i][j];
  }
}
// Ho
double H01[Hu][Hu];
double H0[Hu][Hu];
for (int i = 0; i < Hu; i++)
{ for (int j = 0; j < Hu; j++)
  { H01[i][j] = 0;
    for (int k = 0; k < Hp; k++)
    { H01[i][j] = H01[i][j] + (trastheta[i][k] *
    theta[k][j] * Q);
    }
  }
}

for (int i = 0; i < Hu; i++)
{
  for (int j = 0; j < Hu; j++)
  {
    H0[i][j] = 2 * (H01[i][j] + R);
  }
}

double trasH0[Hu][Hu];
for (int i = 0; i < Hu; i++)
{
  for (int j = 0; j < Hu; j++)
  {
    trasH0[j][i] = H0[i][j];
  }
}

//obj.H
for (int f = 0; f < Hu; f++)
{ for (int c = 0; c < Hu; c++)
  {
    H[f][c] = (H0[f][c] + trasH0[f][c]) / 2;
  }
}
```

```
deltaumax = 4000;
deltaumin = -4000;
umax = 3000;
umin = 0;
ymax = 180;
ymin = 70;

// I1
int I1 = 1;


//A1

int A1[2 * Hu][Hu] = {{1},
  { -1}
};

//A2
int A2[2 * Hu][Hu] = {{1},
  { -1}
};


int fi;
for (int i = 0; i < 2 * Hu; i++)
{
  fi++;
  for (int j = 0; j < Hu; j++)
  {
    Ac[fi - 1][j] = A1[i][j];
  }
}

//Ac


for (int i = 0; i < 2 * Hu; i++)
{
  fi++;
  for (int j = 0; j < Hu; j++)
  {
    Ac[fi - 1][j] = A2[i][j];
  }
}
```

```
  //b1
  for (int i = 0; i < 2 * Hu; i++)
  {
    b1[i][0] = 1 * 400;
  }

}

////////////////////////////////////////////////////////////

double MPCQDMC2::stepImpl(double y, double r, double CHO)
{
   //xf
    double xfaux[13][1]={};
    double op1[13][1];
    double op2[13][1];
    double xfa[3][1];
    for (int i = 0; i < 13; i++)
    {
      xf[i][0] = xm[i][0];
    }

    for (int col=0;col<Hp;col++)
    {
      for (int fil=0;fil<13;fil++)
      {
        xfaux[fil][0]= xf[fil][col];
      }
      Matrix.Multiply((double*)Gm, (double*)xfaux, 13, 13,
      1, (double*)op1);
      xfa[0][0]=u;
      xfa[1][0]=CHO;
      xfa[2][0]=1;
      Matrix.Multiply((double*)Hm, (double*)xfa, 13, 3, 1,
      (double*)op2);
      for(int w=0;w<13;w++)
      {
        xf[w][col+1]= op1[w][0]+op2[w][0];
      }
    }

  //yf

  double yfa[1][Hp];
```

```
  for (int i = 0; i < 1; i++)
{ for (int j = 0; j < Hp; j++)
  { yfa[i][j] = 0;
    for (int k = 0; k < 13; k++)
    { yfa[i][j] = yfa[i][j] + Cm[0][k] * xf[k][j + 1];
    }
  }
}
//  double yf[Hp][1];

Matrix.Transpose((double*) yfa, 1, Hp, (double*) yf);

//d

double d;
d = y - ym[0][0];

// b2

double b2[2 * Hu][1];
double b2a[2 * Hu][1];
double b2b[2 * Hu][1];

for (int i = 0; i < Hu; i++)
{
  b2a[i][0] = umax;
  b2a[i + Hu][0] = -umin;
  b2b[i][0] = u;
  b2b[i + Hu][0] = -u;
}

Matrix.Subtract((double*) b2a, (double*) b2b, 2 * Hu, 1,
(double*) b2);
//bc
double bc[4 * Hu][1];
for (int i = 0; i < 2 * Hu; i++)
{
  bc[i][0] = b1[i][0];
  bc[i + 2 * Hu][0] = b2[i][0];
}

//f
double trastheta[Hu][Hp];
Matrix.Transpose((double*) theta, Hu, Hp, (double*)
trastheta);
```

```
Matrix.Scale((double*) trastheta, Hu, Hp, (double)
-2 * Q);
double fa[Hp][1];
for (int i = 0; i < Hp; i++)
{
  fa[i][0] = r - yf[i][0] - d;
}
double f[1][1];
Matrix.Multiply((double*)trastheta, (double*)fa, Hu, Hp,
1, (double*)f);

/****************************** QUADPROG ********

quadprogpp::Matrix<double> Gax, CEax, CIax;
quadprogpp::Vector<double> g0ax, ce0ax, ci0ax, xax;

n1 = 1;
  G.resize(n1, n1);
  for (int i = 0; i < n1; i++)
  {
    for (int j = 0; j < n1; j++ )
    {
      G[i][j] = H[i][j];
    }
  }

  //prueba[0][0] = Gax[0][0];

  //f
  g0.resize(n1);
  g0[0] = f[0][0];

  //prueba[0][0] = g0ax[0];

  //  //CE = 0
  n2 = 0;
  CE.resize(n1, n2);
  {
    std::istringstream is("0 ");

    for (int i = 0; i < n1; i++)
      for (int j = 0; j < n2; j++)
        is >> CE[i][j] >> ch;
  }
```

```
    ce0.resize(n2);
  {
    std::istringstream is("0 ");

    for (int j = 0; j < n2; j++)
      is >> ce0[j] >> ch;
  }

  //prueba[0][0] = ce0ax[0];

  //Ac
  n3 = 4;
  CI.resize(n1, n3);
  for (int i = 0; i < n1; i++)
  {
    for (int j = 0; j < n3; j++ )
    {
      CI[i][j] = Ac[j][0];
    }
  }

  //bc
  ci0.resize(n3);
  for (int i = 0; i < n3; i++ )
  {
    ci0[i] = bc[i][0];
  }

 //x ->Respuestas de quadprog
  x.resize(n1);

  std::cout << "f1: " << solve_quadprog(G, g0, CE, ce0,
  CI, ci0,x) << std::endl;
  std::cout << "x: " << x << std::endl;

  /*FOR DOUBLE CHECKING COST since in the solve_quadprog
  routine the matrix G is modified */
  for (int i = 0; i < n1; i++)
  {
    for (int j = 0; j < n1; j++ )
    {
      G[i][j] = H[i][j];
    }
  }
```

```
        for (int i = 0; i < n1; i++)
       for (int j = 0; j < n1; j++)
         sum += x[i] * G[i][j] * x[j];
    sum *= 0.5;
    for (int i = 0; i < n1; i++)
      sum += g0[i] * x[i];
    prueba[0][0] = x[0];
  /****************************** QUADPROG ********/
  double deltau = x[0];
  double u0;
  // isempty

  if (deltau == 0)
  {
    deltau = 0;
  }
  else
  {
    deltau = deltau;
  }
  u0 = deltau + u;
  // xm
  double xm1[13][1];
  Matrix.Multiply((double*)Gm, (double*)xm, 13, 13, 1,
  (double*)xm1);
  double xm3[3][1];
  xm3[0][0] = u0;
  xm3[0][1] = CHO;
  xm3[0][2] = 1;
  double xm2[13][1];
  Matrix.Multiply((double*)Hm, (double*)xm3, 13, 3, 1,
  (double*)xm2);
  Matrix.Add((double*) xm1, (double*) xm2, 13, 1,
  (double*) xm);
  //ym
     for ( int k = 0; k < 13; k++)
    { ym[0][0]=0;
      ym[0][0] = ym[0][0] + Cm[0][k] * xm[k][0];
    }
     u = u0;
   double u1;
  return u1;

}
```

APENDICE B. Tarjeta DSPACE (DS1104)

La DS1104 es una tarjeta de control que cuenta con entradas analógicas y digitales, es un procesador en tiempo real diseñado para conectarse directamente a un computador mediante un puerto PCI. Para el uso de la tarjeta se utiliza la herramienta Real Time Interface (RTI), que consiste en un enlace entre el hardware de Dspace y el software de desarrollo Matlab/Simulink. El toolbox de Dspace para este software se llama rti1104 ofrece distintos bloques para la tarjeta de control y están disponibles en la liberaría de Simulink. Las principales especificaciones se presentan en la tabla 1.

Tabla 1. Características DS1104

Procesador	MPC8240
	Procesador punto flotante de 64 bits
	Reloj de la CPU: 250 MHz
Convertidor A/D	4 canales paralelos A/D (4x12bis)
	4 canales multiplexados A/D (4x16bis)
	Rango de voltaje de entrada 10V
	Error de offset ± 5mV
Convertidor D/A	8 canales
	Resolución 16 bits
	Rango de voltaje de salida ±10V
	Error de offset ±1mV

I/O digitales	20 canales de propósito general (GPIO, General – Purpuse Input/Output), con niveles de entrada /salida ttl