



Tecnológico de Monterrey

Avance 2 de situación problema

Carlos Eduardo Lopez Cuevas A01640751

Ricardo Sebastián González Rivas A01646105

Rodrigo Armando Reveles Picie A01641220

12 de septiembre de 2025

Modelación de sistemas mínimos y arquitecturas computacionales

Grupo 103

Equipo HEXA

Primera etapa

La industria automotriz ha ido evolucionando durante los años hasta convertirse en una herramienta del día a día con métodos de seguridad que previenen la pérdida de muchas vidas y lesiones. Sin embargo, todavía existen pequeños detalles en los vehículos que pueden ser un riesgo a la vida de cualquier ser vivo tanto para el conductor como para el peatón.

Sistema de frenado ABS

El sistema antibloqueo de frenos es una forma de seguridad cuya función principal es evitar que las ruedas se bloqueen cuando frenas de golpe debido a que esta técnica de control permite seguir maniobrando mientras frenas, reduciendo hasta un 30% de los accidentes por derrape de acuerdo con Hernández del Arco. (Hernández del Arco, L, 2024). Aunque este sistema fue enfocado principalmente para aviones entre 1929, no fue hasta 1970 que lo implementaron para vehículos.

Sin embargo, a pesar de los beneficios que aporta el sistema antibloqueo de frenos, también presenta diversas fallas que pueden comprometer su funcionamiento. Volkswagen México indica que uno de los problemas más comunes son los sensores de las ruedas. Si estos están sucios o dañados, el sistema no recibe la información necesaria para funcionar correctamente. Asimismo, la falta o contaminación del líquido de frenos afectará el rendimiento de este proceso y lo mostrará como activo para el conductor pese a que no esté activado. Otro problema frecuente se encuentra en la bomba hidráulica, cuyo desgaste reduce la eficiencia del sistema ABS. Del mismo modo, el módulo de control al actuar como el cerebro del ABS si esta falla, no se podrá procesar la información y generará errores en el sistema. Finalmente, aspectos como fallas eléctricas o el desgaste de discos y pastillas de freno también pueden provocar que el ABS se desactive y encienda la alerta en el tablero del vehículo.

Variables

Para recopilar algunas mencionadas previamente son:

- **Sensores de rueda:** nivel de suciedad, daño físico, señal eléctrica generada.
- **Líquido de frenos:** nivel, viscosidad, contaminación en el líquido.
- **Bomba hidráulica:** presión generada, desgaste, temperatura.
- **Módulo de control:** estado de software, tiempo de respuesta, integridad de datos.
- **Componentes mecánicos:** grosor de discos y pastillas, desgaste total.

Una manera en la que las computadoras puedan medir estas variables para recopilar datos e informar al vehículo, es principalmente mediante la unidad de control (ECU) debido a que conforme con Romacarabs la unidad de control y la bomba hidráulica regulan la presión de frenado, evitando que las ruedas se bloqueen para que exista un mejor control durante el frenado.

Otra manera de análisis computacional es mediante la detección de anomalías que cheque los patrones en la presión de frenos para observar si no se responde como se le ordena y en los sensores para observar si se mandan señales incoherentes para detectarlo como fallo eléctrico o suciedad.

Por último, también se puede analizar el sistema ABS mediante simulaciones computacionales que repliquen este sistema en diferentes condiciones como lluvia, frío, sobrecalentamiento, frenado rápido, entre otros para ver cómo reaccionaría el modelo sin tener que arriesgar vidas humanas.

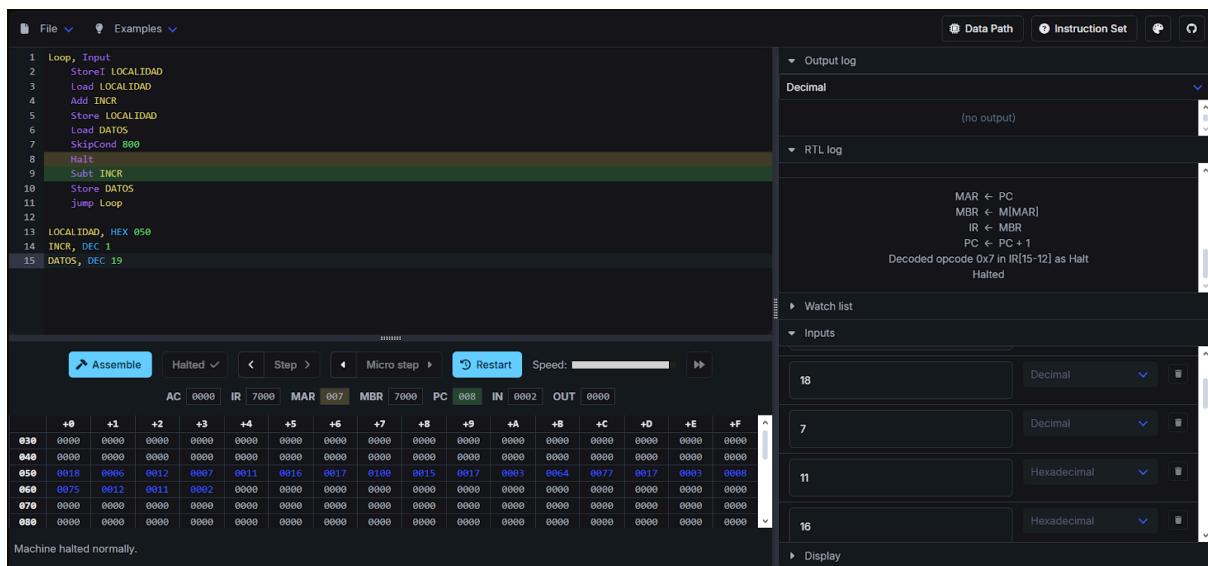
Los problemas que vemos en la industria automotriz, como fallas en frenos, dirección o sistemas electrónicos, también aparecen en la industria aeroespacial, solo que allá son todavía más críticos. Como vimos previamente el ABS fue creado principalmente para aviones, pero

durante el transcurso del tiempo este se fue implementando más al coche mientras que en los aviones se optimizo más el sistema ya que estos manejan velocidades extremas por lo le cambiaron el nombre a Antiskid.

El sistema ABS es un caso ciberfísico que combina sensores, actuadores y software con el mundo real para controlar el frenado de un vehículo. En cambio, la máquina de Von Neumann, es un modelo teórico de cómo funciona una computadora tradicional.

La diferencia principal es que el modelo de Von Neumann solo procesa datos de manera secuencial y abstracta, mientras que un sistema ciberfísico como el visto conecta directamente el mundo digital con el físico para actuar de forma inmediata para evitar accidentes.

Segunda etapa



The screenshot shows a computer simulation interface for a von Neumann architecture. At the top, there's a menu bar with 'File' and 'Examples'. Below it is a code editor window displaying assembly code:

```

1 Loop, Input
2 Store LOCALIDAD
3 Load LOCALIDAD
4 Add INCR
5 Store LOCALIDAD
6 Load DATOS
7 SkipCond 800
8 Halt
9 Sub INCR
10 Store DATOS
11 jump Loop
12
13 LOCALIDAD, HEX 050
14 INCR, DEC 1
15 DATOS, DEC 19

```

Below the code editor is a control panel with buttons for 'Assemble', 'Halted', 'Step', 'Micro step', 'Restart', and 'Speed'. To the right of the control panel is a status message: 'Machine halted normally.'

Underneath the control panel is a register dump table:

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0018	0006	0012	0007	0011	0016	0017	0100	0015	0017	0003	0064	0077	0017	0003	0008
060	0075	0012	0011	0002	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

To the right of the register dump is a memory dump table:

	AC	IR	MAR	MBR	PC	IN	OUT
030	0000	7000	007	7000	0008	0002	0000
040	0000	0000	0000	0000	0000	0000	0000
050	0018	0006	0012	0007	0011	0016	0017
060	0075	0012	0011	0002	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000

On the far right, there are several panels for 'Output log', 'RTL log', 'Watch list', and 'Inputs'. The 'Output log' panel shows '(no output)'. The 'RTL log' panel shows the machine state: MAR ← PC, MBR ← MIMAR, IR ← MBR, PC ← PC + 1, Decoded opcode 0x7 in IR[15-12] as Halt, Halted. The 'Inputs' panel shows values for inputs 18, 7, 11, and 16.

The screenshot shows a microcontroller simulation interface. At the top, there are buttons for 'Assemble' (highlighted in blue), 'Halted', 'Step', 'Micro step', 'Restart', and a speed slider. Below this is a header row with registers: AC (0000), IR (7000), MAR (007), MBR (7000), PC (008), IN (0002), OUT (0000). The main area displays assembly code in columns labeled +0 to +F. The code consists of 20 lines, each starting with an address (e.g., 030, 040, ..., 080) followed by various binary values. A message at the bottom states 'Machine halted normally.'

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0018	0006	0012	0007	0011	0016	0017	0100	0015	0017	0003	0064	0077	0017	0003	0008
060	0075	0012	0011	0002	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Diferentes datos

This screenshot shows the same simulation interface as the first one, but with different data values in the assembly code. The code now starts at address 000 and includes more lines than the previous version. The data values are mostly binary strings like '0000' or '0001'. A message at the bottom again states 'Machine halted normally.'

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	E020	1023	B020	2023	1020	3021	2020	1022	8800	900E	4021	2022	9000	1023	2026
010	1024	2025	1026	4025	2026	1027	3021	2027	1026	8800	901C	9012	1027	4021	6000	7000
020	0064	0001	0000	0014	0014	0014	0000	0001	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001
060	0001	0001	0001	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
090	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0A0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

El código que presentamos a continuación es bastante sencillo. Su función principal es registrar 20 datos en ubicaciones específicas, aunque pueden ser los datos que se necesiten solo se debe registrar en la variable datos siguiendo la estructura n-1, en este caso como queremos 20 sería 19. La primera de estas localidades se definió como 050, con simplemente para saber donde se registran los datos. Para lograrlo, cada localidad se va incrementando de uno en uno, pasando de 050 a 051, luego 052, y así sucesivamente hasta completar las 20 entradas. Todo esto se maneja mediante un loop, que se detiene cuando los datos dejan de cumplir con la condición de ser mayores a 0. En ese momento, se cumple la condición para cerrar el loop, evitando que sea un programa infinito.

En cada iteración, se realiza un incremento y se almacenan los datos correspondientes. Por eso, aunque hablamos de 20 datos, el loop realmente recorre los valores de 0 a 19, no de 1 a 20.

Tercera etapa

```

1 /Leer entrada de 20 numeros y sumarlos
2 Loop, Input           /Ler la entrada
3 Store LOCALIDAD       /Guardar el valor en la localidad LOCALIDAD
4 Load SUMA              /Cargar la suma
5 AddI LOCALIDAD         /Añadir el valor de la entrada a SUMA
6 Store SUMA             /Guardar la suma
7 Load LOCALIDAD         /Cargar la localidad
8 Add INCR               /Añadir 1 para la siguiente localidad
9 Store LOCALIDAD         /Guardar localidad en LOCALIDAD
10 Load DATOS            /Cargar numeros por guardar en AC
11 Load DATOS            /Cargar numeros por guardar en AC
12
13 SkipCond 800          /Si AC > 0, omitir la siguiente linea
14 Jump Continuar        /Continuar con la division
15
16 Subt INCR             /Restar a datos 1
17 Store DATOS           /Guardar la cantidad de numeros que queda por guardar
18 Jump Loop              /Reiniciar el bucle
19
20 /Continuamos guardando valores en estas nuevas variables para dividir claro que si
21 Continuar, Load SUMA   /Cargar la suma
22 Store DIVIDENDO        /Guardaria como el dividendo
23 Load NUM                /Cargar la cantidad de numeros
24 Store DIVISOR           /Guardar como el divisor
25
26 /Bucle para dividir
27 Dividir, Load DIVIDENDO /Cargar dividendo que es la suma de todo lo anterior
28 Subt DIVISOR           /Restar por el divisor que es 20
29 Store DIVIDENDO        /Lo guardamos
30 Load COCIENTE          /Aumentar el cociente
31 Add INCR               /Vamos sumando el cociente 1 hasta que el dividendo sea menor a 0
32 Store COCIENTE          /Guardamos
33 Load DIVIDENDO          /Si el dividendo es menor a 0 o es 0 se acaba todo
34 SkipCond 800
35 jump end
36 Jump Dividir           /Sino repetimos hasta tener menor a 0 :)
37

```

The screenshot shows the assembly code for the division process. The code starts at address 26, loading the dividend from memory into the AC register. It then enters a loop starting at address 27, where it subtracts the divisor (20) from the dividend. If the result is non-negative, it increments the quotient (COCIENTE) and loops back. Once the dividend becomes negative, it exits the loop and prints the quotient. The debugger also shows the state of various registers and memory locations.

```

26 /Bucle para dividir
27 Dividir, Load DIVIDENDO /cargar dividendo que es la suma de todo lo anterior
28 Subt DIVISOR           /Restar por el divisor que es 20
29 Store DIVIDENDO        /Lo guardamos
30 Load COCIENTE          /Aumentar el cociente
31 Add INCR               /Vamos sumando el cociente 1 hasta que el dividendo sea menor a 0
32 Store COCIENTE          /Guardamos
33 Load DIVIDENDO          /Si el dividendo es menor a 0 o es 0 se acaba todo
34 SkipCond 800
35 jump end
36 Jump Dividir           /Sino repetimos hasta tener menor a 0 :)
37
38 end, Load COCIENTE     /Cargamos e imprimimos
39 Output
40 halt
41 LOCALIDAD, HEX 050      /localidad donde se empiezan a guardar las entradas
42 INCR, DEC 1              /Constante para restar o sumar por 1
43 DATOS, DEC 19            /Contador, cantidad de numeros por guardar
44 SUMA, DEC 0               /Suma de los 20 numeros
45 NUM, DEC 20              /Contador, cantidad de numeros que ya se guardaro
46 DIVISOR, DEC 0            /Numero por el que se divide la suma
47 DIVIDENDO, DEC 0          /Numero que se està dividiendo
48 COCIENTE, DEC 0           /Resultado de la division

```

Inputs registrados= 8, 9, 10, 11, 7, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

The output log window displays the decimal results of the division process. It shows the quotient (COCIENTE) and the remainder (DIVIDENDO) for each iteration, along with the final result of the division.

Iteration	Quotient (COCIENTE)	Dividend (DIVIDENDO)
1	0	8
2	0	9
3	0	10
4	0	11
5	0	7
6	0	6
7	0	7
8	0	8
9	0	9
10	0	10
11	0	11
12	0	12
13	0	13
14	0	14
15	0	15
16	0	16
17	0	17
18	0	18
19	0	19
20	0	20

$$(8 + 9 + 10 + 11 + 7 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 19 + 20) / 20 = 12$$

El código que se muestra a continuación es una mejora de la parte 2 del avance de la evidencia. Esta versión no solo registra 20 entradas, sino que también las suma y calcula el

promedio dividiendo la suma entre la cantidad de datos. Como en MARIE no existe una instrucción de división, implementamos la operación mediante restas sucesivas: se va restando la cantidad de datos al total hasta que el dividendo llegue a cero o tenga un residuo.

Para esto, se agregó la variable SUMA, donde se registran y acumulan todos los inputs. Una vez completada la entrada de datos, se carga SUMA en DIVIDENDO y la cantidad de datos en DIVISOR, lo que permite realizar el loop de división de manera clara y entendible. Durante este loop, se resta el divisor al dividendo y cada vez que se realiza la operación se incrementa COCIENTE en 1. Este proceso se repite hasta que el dividendo sea menor o igual a cero, momento en el cual el loop termina.

De esta manera, el código logra registrar los inputs, calcular su suma y obtener el promedio usando únicamente sumas y restas, demostrando cómo se pueden implementar operaciones más complejas en MARIE con lógica básica de loops y acumuladores.

Referencias:

Autoingenieria. (2025). *Los 10 problemas más comunes de tu vehículo | Auto Ingeniería.* [Autoingenieria.com;](https://www.autoingenieria.com/los-10-problemas-mas-comunes-de-tu-vehiculo.html)

Rently. (2021). *Problemas en la industria automotriz. ¿Cuál es el estado en el 2021?* [. https://www.rentlysoft.com/blog-noticias/problemasindustriaautomotriz](https://www.rentlysoft.com/blog-noticias/problemasindustriaautomotriz)

Infor. (2023). *Cinco principales retos de la industria automotriz actual.* Infor. <https://www.infor.com/latam/blog/top-five-challenges-of-the-current-automotive-industry>

Mazzae, E., Barickman, F., Scott Baldwin, G., & Forkenbrock, G. (n.d.). *Driver Crash Avoidance Behavior with ABS in an Intersection Incursion Scenario on Dry Versus Wet Pavement.* <https://www.nhtsa.gov/sites/nhtsa.gov/files/sae1999-01-1288.pdf>

Hernández, L. (2024, May 17). *Frenos ABS: Cómo funciona el sistema antibloqueo de ruedas* - Autos. Autos.

<https://us.as.com/autos/curiosidades/frenos-abs-como-funciona-el-sistema-antibloqueo-de-ruedas/>

oneair. (2024, August 13). *¿Cómo frena un avión? Sistemas de frenos, cálculos y mucho más.* One Air. <https://www.oneair.es/como-frena-un-avion/>

Volkswagen México. (2025, February 18). *Testigo ABS: causas, consecuencias y soluciones.* Vw.com.mx; Volkswagen de México.

<https://www.vw.com.mx/es/experiencia/tips/que-significa-luz-testigo-abs.html>

Joaquín Patiño. (2025, March 27). *Fallas en frenos, cambios de marcha y problemas en los limpiaparabrisas: los defectos más comunes de los automóviles en México.* El País México.

<https://elpais.com/mexico/2025-03-27/fallas-en-frenos-cambios-de-marcha-y-problemas-en-los-limpiaparabrisas-los-defectos-mas-comunes-de-los-automoviles-en-mexico.html>

Romacarabs. (2025, August 12). *¿Qué es el ABS y cómo funciona? Sistema antibloqueo.* Romacar ABS.

<https://www.romacarabs.com/noticias-automocion/que-es-y-como-funciona-el-abs/>