UNIVERSIDAD DE COSTA RICA

Escuela de Ingeniería Eléctrica

IE-0623 Microprocesadores

Proyecto Final

RADAR 623

Realizado por

Stuart Leal Quesada B53777

San José, Costa Rica 2019

Resumen

Aquí va el resumen del trabajo escrito.

Índice general

Índice de figuras

Índice de tablas

1 Resumen

La introducción va aquí.

2 Desarrollo

2.1. Explicación general

2.2. Subrutina ATD_ISR

2.2.1. Cálculos para interrupción ATD_ISR

En lo que respecta a esta interrupción, debemos realizar los cálculos para encontrar el valor de Presescalador que vamos a utilizar. La fórmula que relaciona la frecuencia con el valor del preescalador es la siguiente:

$$frs = \frac{BUS\_CLK}{2 \cdot (PRS + 1)} \tag{2.1}$$

Para este caso, queremos la frecuencia más baja posible para este periférico, por tanto sería utilizar el valor más alto de PRS posible. Lo cuál corresponde a:

$$PRS = 31 = \$1F \tag{2.2}$$

2.2.2. Configuración de ATD_ISR

Para el primer registro de control $ATD0CTL2$, queremos habilitar el módulo de conversiones con el bit $ADPU = 1$, queremos habilitar la opción de $AFFC$, para que se borra la bandera de interrupción cuando se leen los registros de datos y también, habilitar las interrupciones con el bit $ASCIE = 1$. Esto significa que:

$$ATD0CTL2 = \$C2$$

Para el siguiente registro de control $ATDOCTL3$, queremos 6 conversiones, y además $FIFO = 0$. Entonces tenemos que:

$$ATD0CTL3 = \$30$$

En $ATD0CTL4$, vamos a querer configurar 4 periodos de reloj para el muestreo, y además $SRE8 = 1$ para tener conversiones a 8 bits. Además, vamos a querer el valor del preescalador en 31. Esto significa que:

$$ATD0CTL4 = \$BF$$

Finalmente, tenemos que activar la justificación a la derecha, y las conversiones hacerlas sin signo ($DJM = 1$ y $DSGN = 0$). Además queremos configurar $SCAN = MULT = 0$ para que se muestree sólo la entrada definida, 6 veces y se guarden los valores de $ADR0$ hasta $ADR5$. Y finalmente, queremos seleccionar la entrada 7 con los bits CC, CB y CA. Entonces:

$$ATD0CTL5 = \$87$$

2.2.3. Vector de interrupción ATD_ISR

Finalmente, el vector de interrupción para ATD se encuentra en la dirección $\$3E52$.

2.2.4. Diagrama y explicación de la subrutina ATD_ISR

Descripción

Explicación

Diagrama

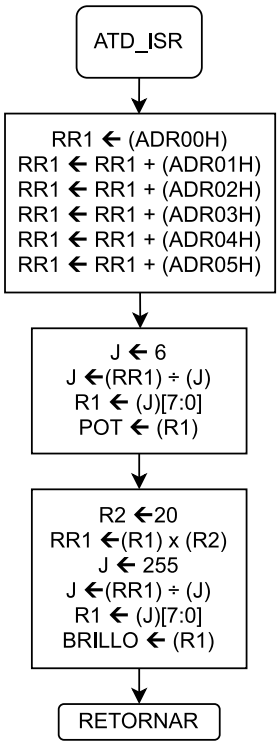

Figura 2.1: Diagrama de flujos para ATD_ISR.

2.3. Subrutina TCNT_ISR

2.3.1. Cálculos para interrupción TCNT_ISR

Para esta interrupción, lo que debemos de calcular es el tiempo que toma entre una interrupción y otra. Esto se puede calcular con la siguiente fórmula 2.3.

NOTA: Se usa un Preescalador de 8, puesto que la subrutina $OC4\_ISR$ utiliza este Preescalador la el módulo de TIMERS.

$$T_{TOI} = \frac{2^{16} \cdot 8}{24MHz} = \frac{1024}{46875} seg \tag{2.3}$$

2.3.2. Configuración de TCNT_ISR

Para esta interrupción, la configuración del módulo de reloj se detallará para la subrutina de $OC4\_ISR$.

La configuración propia de la interrupción por rebase, se hace escribiendo un 1 en el bit 7 del registro $TSCR2$.

Para deshabilitar la interrupción, se escribe un 0 en el mismo bit 7 de ese registro.

2.3.3. Vector de interrupción TCNT_ISR

El vector de interrupción para esta subrutina se encuentra en la dirección $\$3E5E$.

2.3.4. Diagrama y explicación de la subrutina TCNT_ISR

Descripción

Explicación

Diagrama

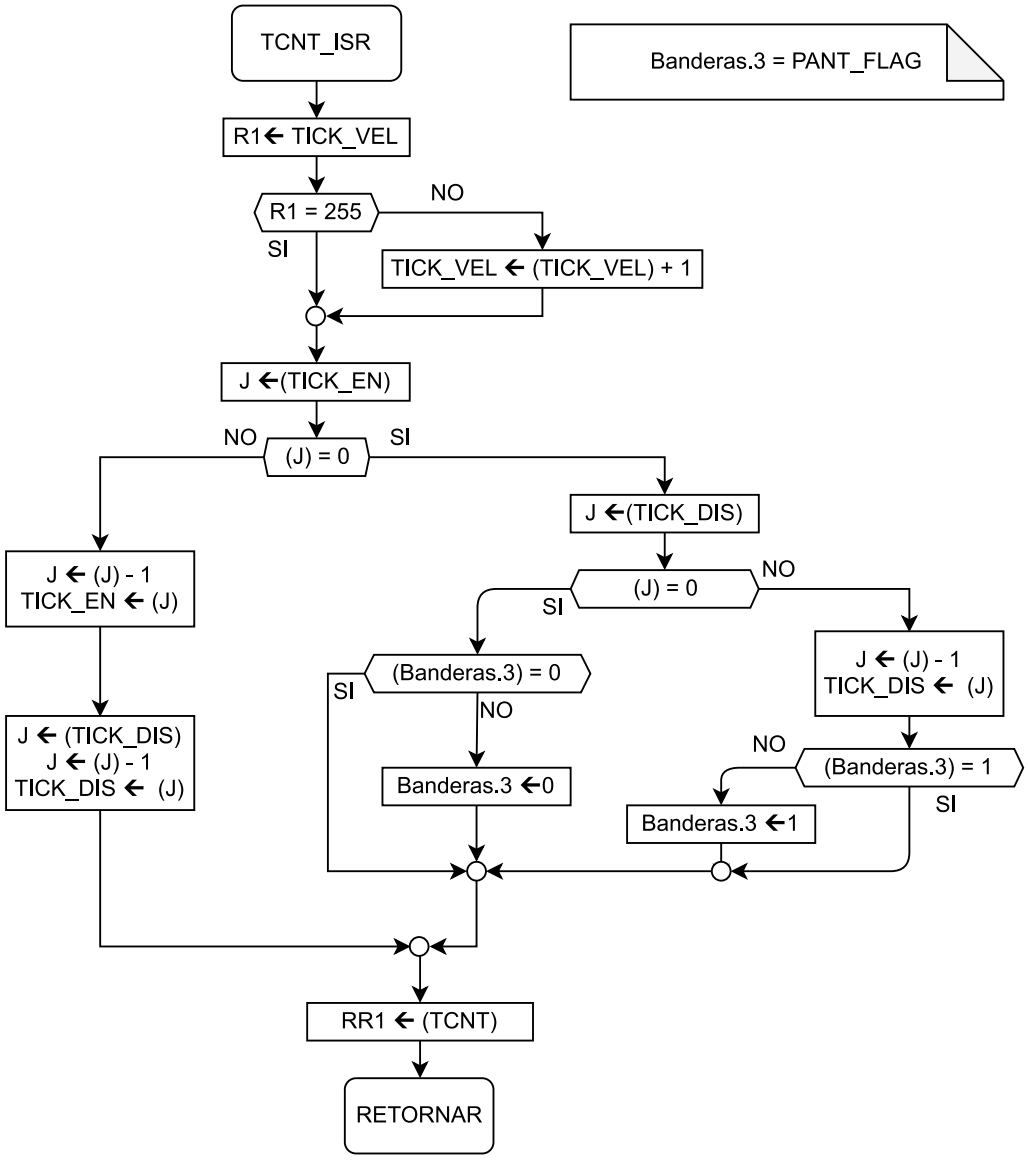

Figura 2.2: Diagrama de flujos para TCNT_ISR.

2.4. Subrutina CALCULAR

2.4.1. Cálculos para interrupción CALCULAR

Para realizar el cálculo de la velocidad, usamos la siguiente fórmula:

$$VELOC = \frac{40m}{n_{ticks}} \cdot \frac{tick}{seg} \cdot \frac{seg}{hora} \cdot \frac{km}{metro} \tag{2.4}$$

Desarrollando lo anterior, tenemos que:

$$VELOC = \frac{40}{n_{ticks}} \cdot \frac{46875}{1024} \cdot \frac{3600}{1} \cdot \frac{1}{1000}$$

$$= \frac{25}{n_{ticks}} \cdot \frac{16875}{64}$$

$$= \frac{421875}{n_{ticks} \cdot 64}$$

Entonces, la estrategia para realizar el cálculo de velocidad, será:

- Realizar la multiplicación de $n_{ticks} \cdot 64$ y guardarlo en X.
- Cargar en D $\#200$, luego en Y $\#16785$. Multiplicación queda en $Y : D$.
- Dividir $Y : D$ entre X, y guardar el resultado en VELOC.

NOTA: Hay que verificar que el resultado no sea más grande que 255 (es posible si n es muy pequeño, como por ejemplo $n = 1$ significa que $resultado = 6591$, lo cuál no es una velocidad con sentido. En caso de ser mayor a 255, guardar el máximo valor posible en VELOC (255).

2.4.2. Configuración de CALCULAR

2.4.3. Vector de interrupción CALCULAR

2.4.4. Diagrama y explicación de la subrutina CALCULAR

Descripción

Explicación

Diagrama

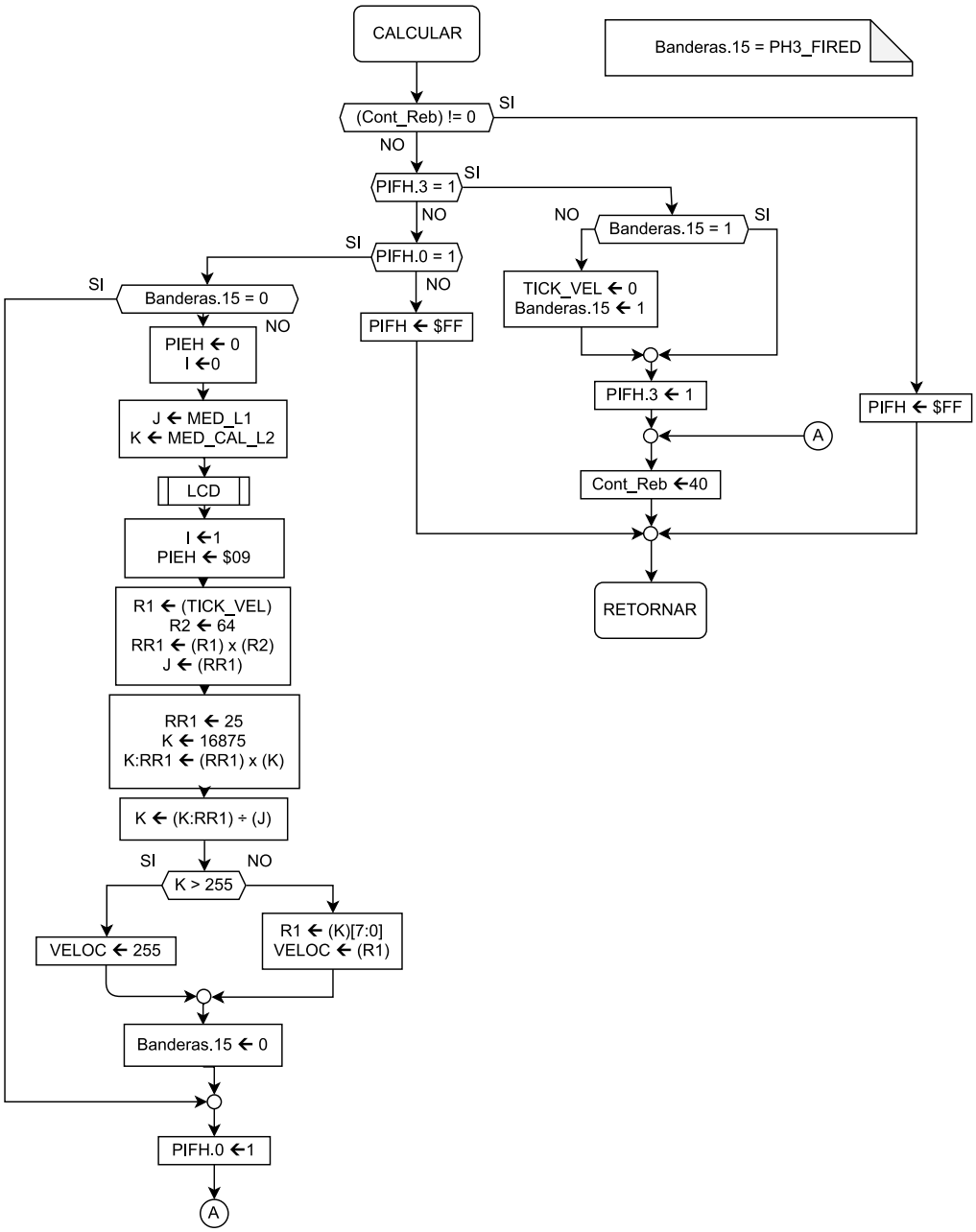

Figura 2.3: Diagrama de flujos para CALCULAR.

2.5. Subrutina RTI_ISR

2.5.1. Cálculos para interrupción RTI_ISR

Para esta subrutina, queremos que se ejecute cada 1ms. Entonces, tenemos la siguiente fórmula:

$$T_{RTI} = \frac{(N+1) \cdot 2^{M+9}}{OSC\_CLK} \tag{2.5}$$

Haciendo un poco de retrospección, OSC_CLK tiene un valor de $8MHz$, por lo que en el numerador necesitamos un número muy cercano a este valor.

Recordando que $2^{13} = 8192 \approx 8 \cdot 10^3$, entonces tendríamos que:

$$T_{RTI} = \frac{(0+1) \cdot 2^{4+9}}{8 \cdot 10^6} = 1{,}024ms$$

Entonces, con $N = 1$ y $M = 4$ logramos nuestro objetivo.

2.5.2. Configuración de RTI_ISR

La configuración para este periferico es bastante sencilla en realidad. Básicamente, lo primero es configurar el tiempo de T_{RTI} con los valores calculados anteriormente. Esto sería:

$$RTICTL = \$40$$

Lo segundo, habilitar el puerto de RTI. Esto último se hace con la siguiente configuración:

$$CRGINT = \$80$$

2.5.3. Vector de interrupción RTI_ISR

El vector de interrupción se encuentra en la dirección $\$3E70$ para el Debug12.

2.5.4. Diagrama y explicación de la subrutina RTI_ISR

Descripción

Explicación

Diagrama

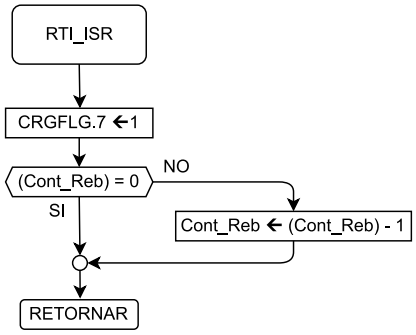

Figura 2.4: Diagrama de flujos para RTI_ISR.

2.6. Subrutina OC4_ISR

2.6.1. Cálculos para interrupción OC4_ISR

Tenemos que usar un Preescalador de 8 para la interrpución por overflow de $TCNT\_ISR$. Entonces, haciendo los cálculos para la cantidad de TICKS que tenemos que contar, para que $OC4\_ISR$ se ejecute con una frecuencia de $50kHz$ serían los siguiente:

$$TCS = \frac{20\mu s \cdot 24MHz}{PRS} \tag{2.6}$$

Esto significa entonces que, usando $PRS = 8$, encontramos que:

$$TCS = 60$$

Cada interrupción entonces, debemos cargar $TC4$ con $TCNT + 60$.

NOTA: Para realizar cuentas de $1ms$ por ejemplo, debemos contar hasta 50.

Cuentas de $100ms$ (Para llamar a $CONV\_BIN\_BCD$ y BCD_7SEG) se realizan contando hasta 5000.

Cuentas de $200ms$ (Para llamar a $PATRON\_LEDS$ y $ATD0\_CTL5$) se realizan contando hasta 10000.

2.6.2. Configuración de OC4_ISR

Para el primer registro de configuración $TSCR1$, queremos habilitar el módulo de Timers (bit TEN) y además, queremos habilitar la bandera de $TFFCA$.

Esto quiere decir que, la bandera de $C4F$ se va a borrar cuando se escriba un dato en $TC4$, y además, la bandera de TOF se va a borrar cuando se lea el registro de $TCNT$.

Entonces, para este primer registro, tenemos que:

$$TSCR1 = \$90$$

Luego tenemos el segundo registro de control, $TSCR2$, en donda vamos a configurar el preescalador con el valor de 8. Esto significa guardar un 3 en los bits de PRS. Por lo tanto, tenemos que:

$$TSCR2 = \$03$$

NOTA: Para habilitar o deshabilitar las interrupciones por rebase, la configuración se hace por este mismo registro, en el bit 7 del registro. Sin embargo no están habilitadas por defecto. Sólo se habilitan cuando se está en el modo medición.

Siguiendo con la configuración, para Output Compare, tenemos que habilitar la opción de output compare para el canal 4, en el registro de $TIOS$. Entonces, tenemos que:

$$TIOS = \$10$$

Y además, tenemos que habilitar la interrupción para cuando la bandera de $C4F$ se levanta. Esto se hace en el registro de configuración llamado TIE, en donde tenemos entonces que:

$$TIE = \$10$$

2.6.3. Vector de interrupción OC4_ISR

El vector de interrupción para $OC4\_ISR$ se encuentra en la dirección $\$3E66$.

2.6.4. Diagrama y explicación de la subrutina OC4_ISR

Descripción

Explicación

fasdf

Diagrama

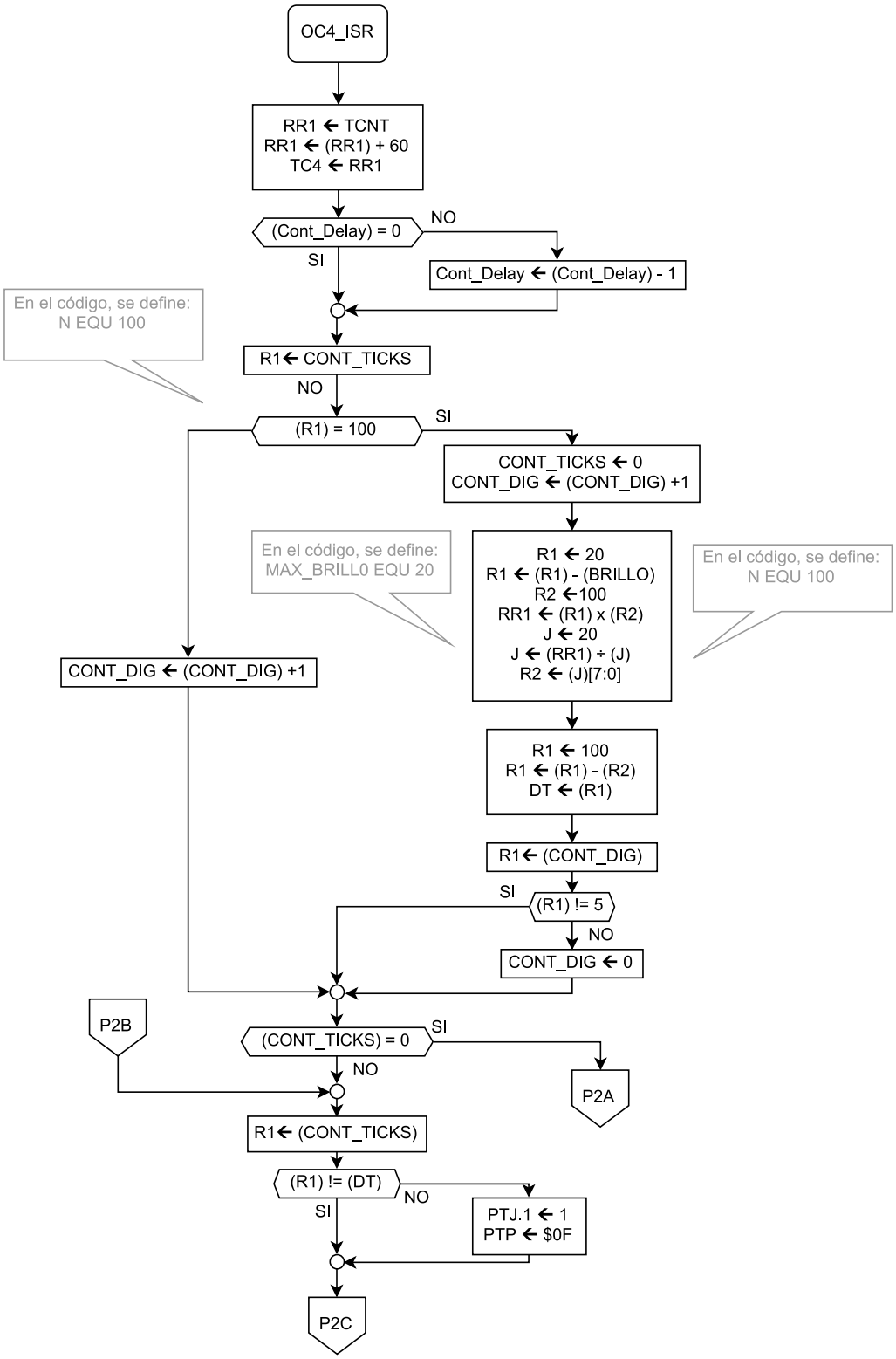

Figura 2.5: Diagrama de flujos para OC4_ISR. Primera parte

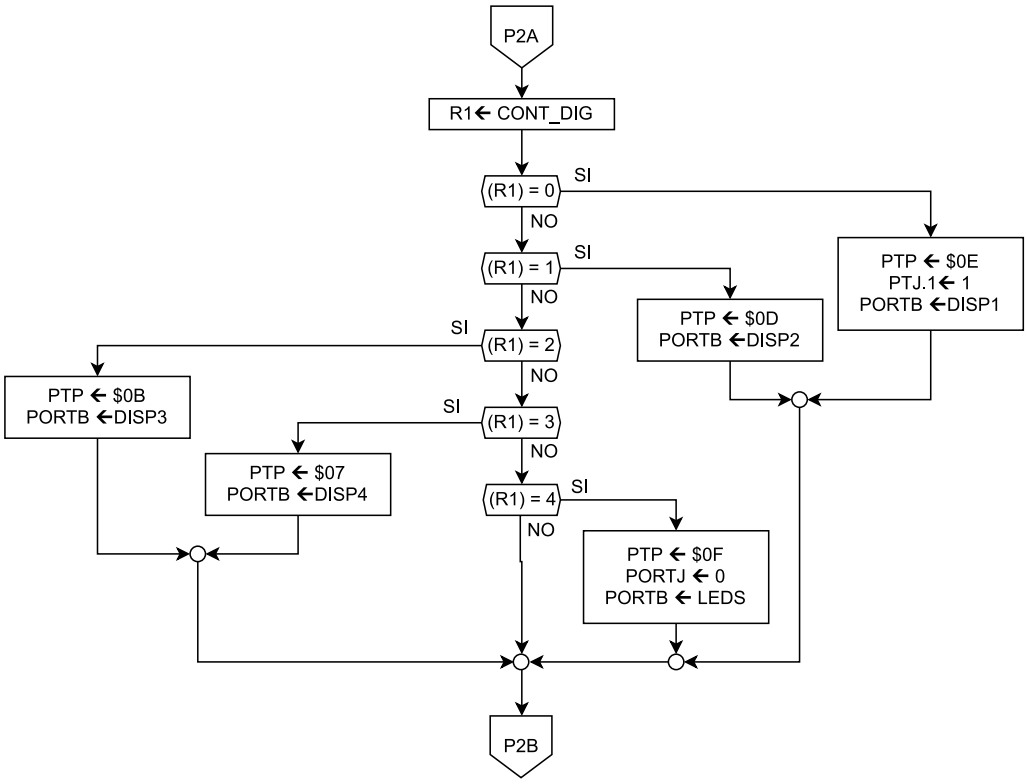

Figura 2.6: Diagrama de flujos para OC4_ISR. Segunda parte

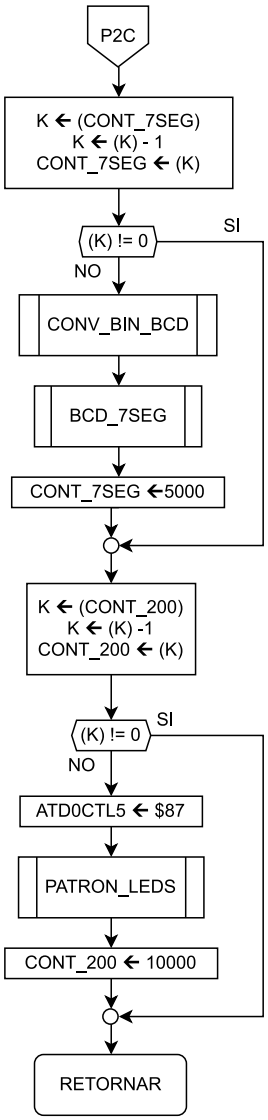

Figura 2.7: Diagrama de flujos para OC4_ISR. Tercera parte

2.7. Subrutina MUX_TECLADO

2.7.1. Diagrama y explicación de la subrutina MUX_TECLADO

Descripción

fasdf

Explicación

Diagrama

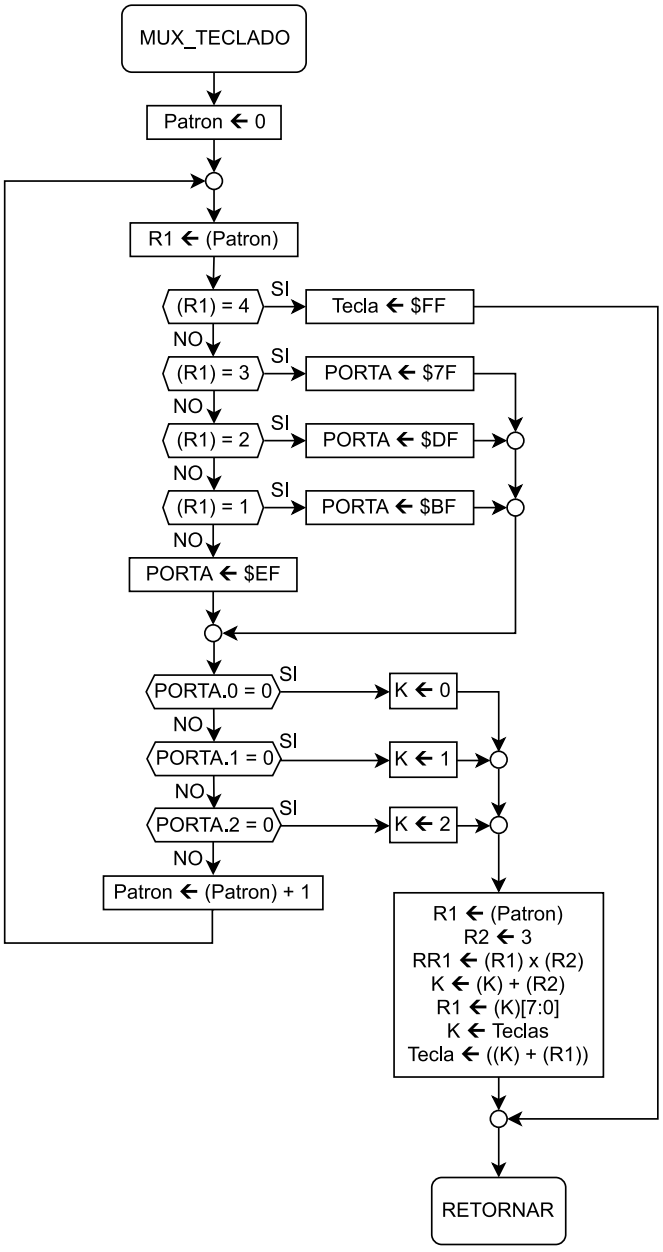

Figura 2.8: Diagrama de flujos para MUX_TECLADO.

2.8. Subrutina TAREA_TECLADO

2.8.1. Diagrama y explicación de la subrutina TAREA_TECLADO

Descripción

fasdf

Explicación

Diagrama

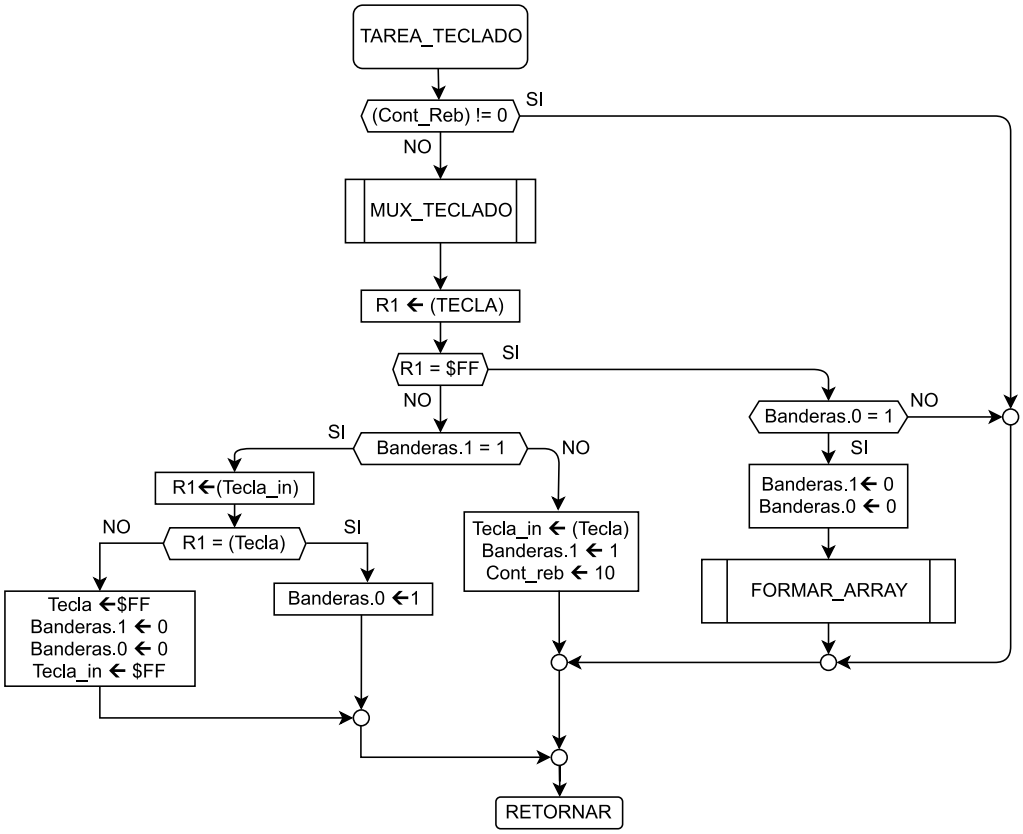

Figura 2.9: Diagrama de flujos para TAREA_TECLADO.

2.9. Subrutina FORMAR_ARRAY

2.9.1. Diagrama y explicación de la subrutina FORMAR_ARRAY

Descripción

fasdf

Explicación

Diagrama

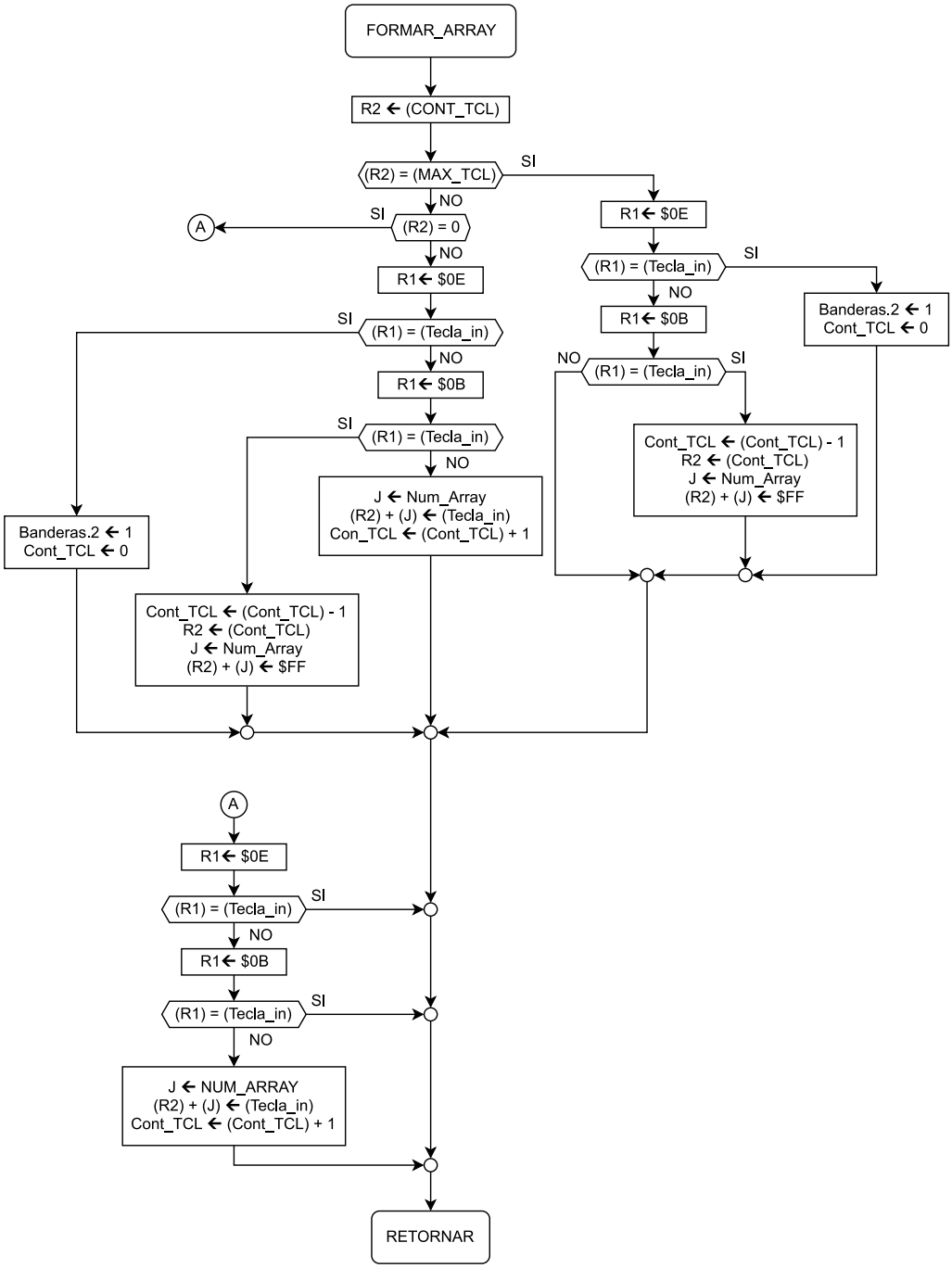

Figura 2.10: Diagrama de flujos para FORMAR_ARRAY.

2.10. Subrutina MODO_MEDICION

2.10.1. Diagrama y explicación de la subrutina MODO_MEDICION

Descripción

fasdf

Explicación

Diagrama

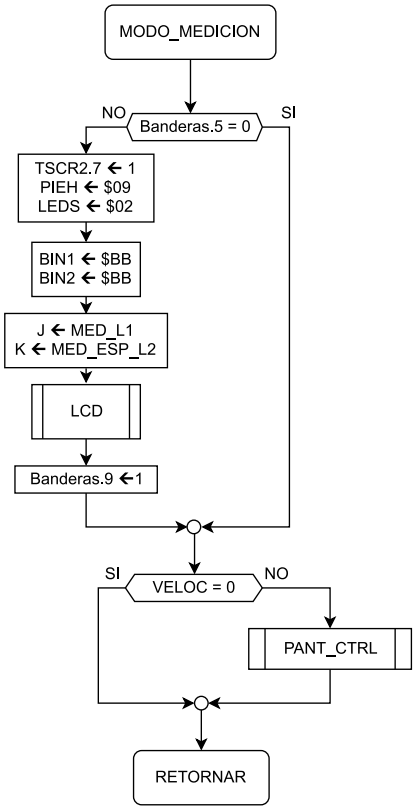

Figura 2.11: Diagrama de flujos para MODO_MEDICION.

2.11. Subrutina PANT_CTRL

2.11.1. Diagrama y explicación de la subrutina PANT_CTRL

Descripción

fasdf

Explicación

Diagrama

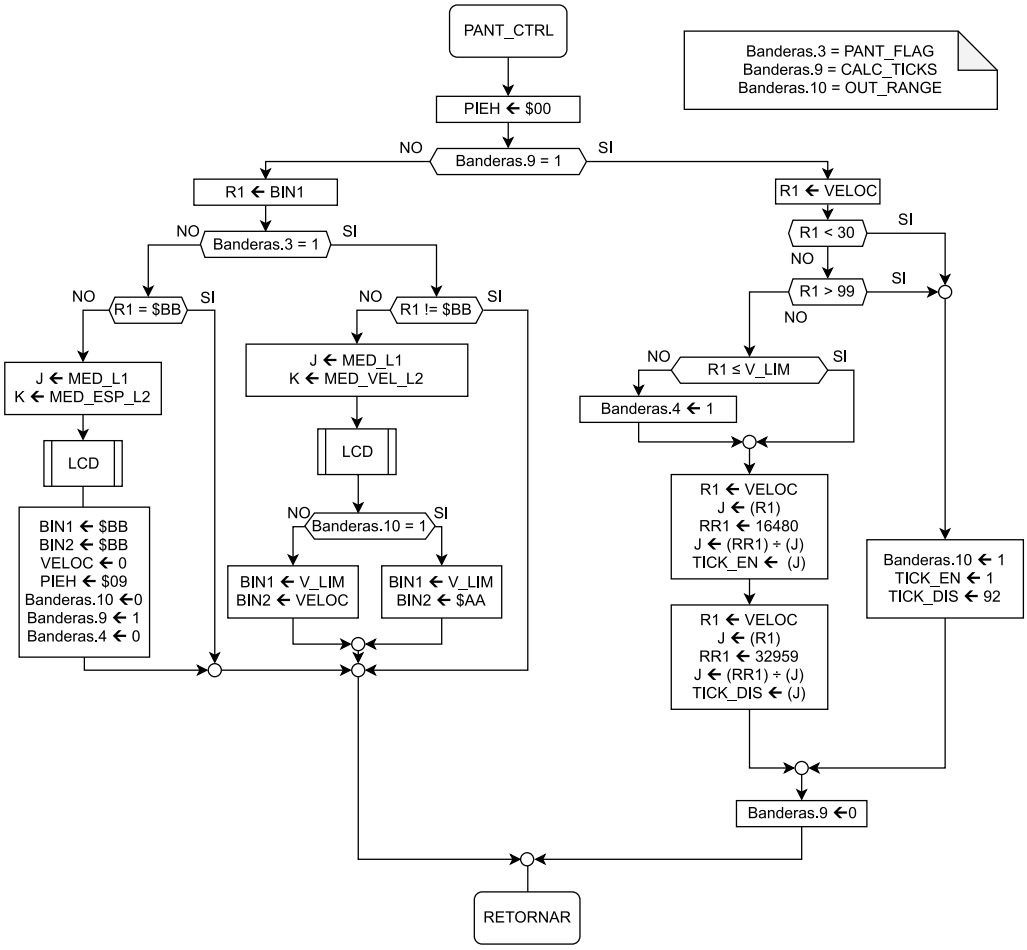

Figura 2.12: Diagrama de flujos para PANT_CTRL.

2.12. Subrutina MODO_CONFIG

2.12.1. Diagrama y explicación de la subrutina MODO_CONFIG

Descripción

fasdf

Explicación

Diagrama

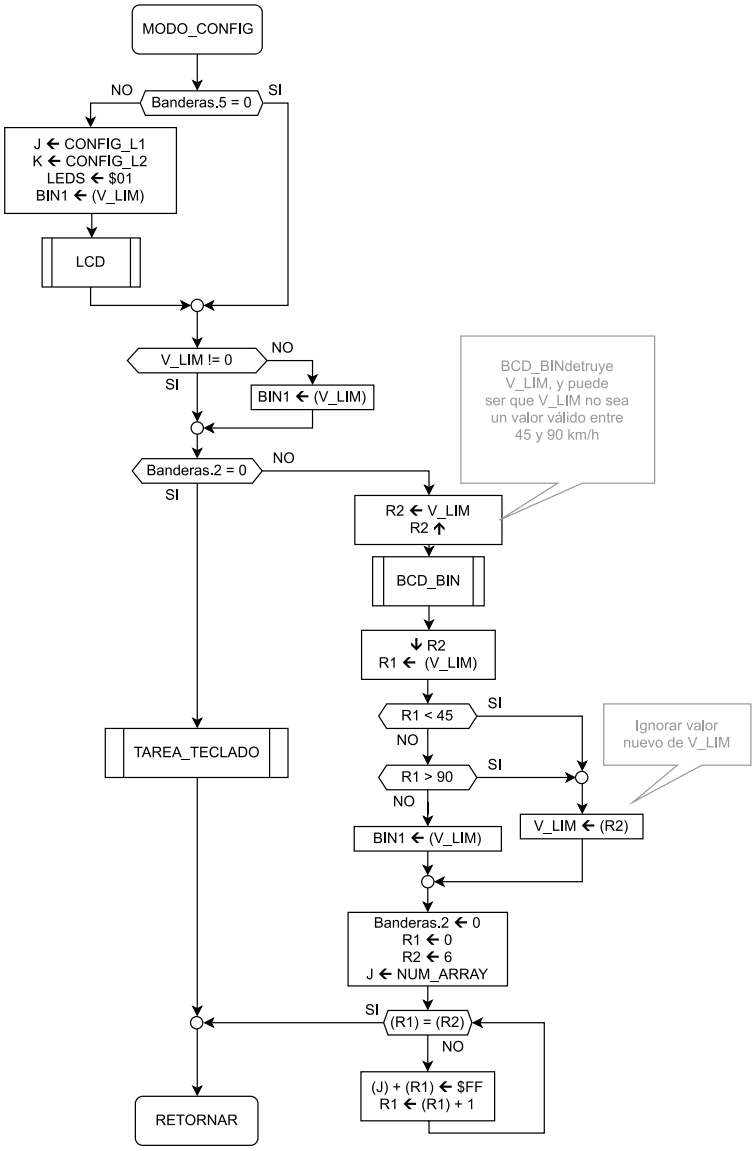

Figura 2.13: Diagrama de flujos para MODO_CONFIG.

2.13. Subrutina MODO_LIBRE

2.13.1. Diagrama y explicación de la subrutina MODO_LIBRE

Descripción

fasdf

Explicación

Diagrama

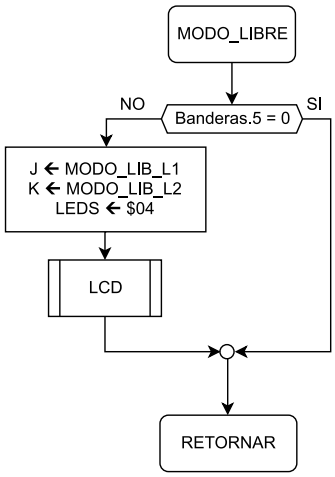

Figura 2.14: Diagrama de flujos para MODO_LIBRE.

2.14. Subrutina BCD_BIN

2.14.1. Diagrama y explicación de la subrutina BCD_BIN

Descripción

fasdf

Explicación

Diagrama

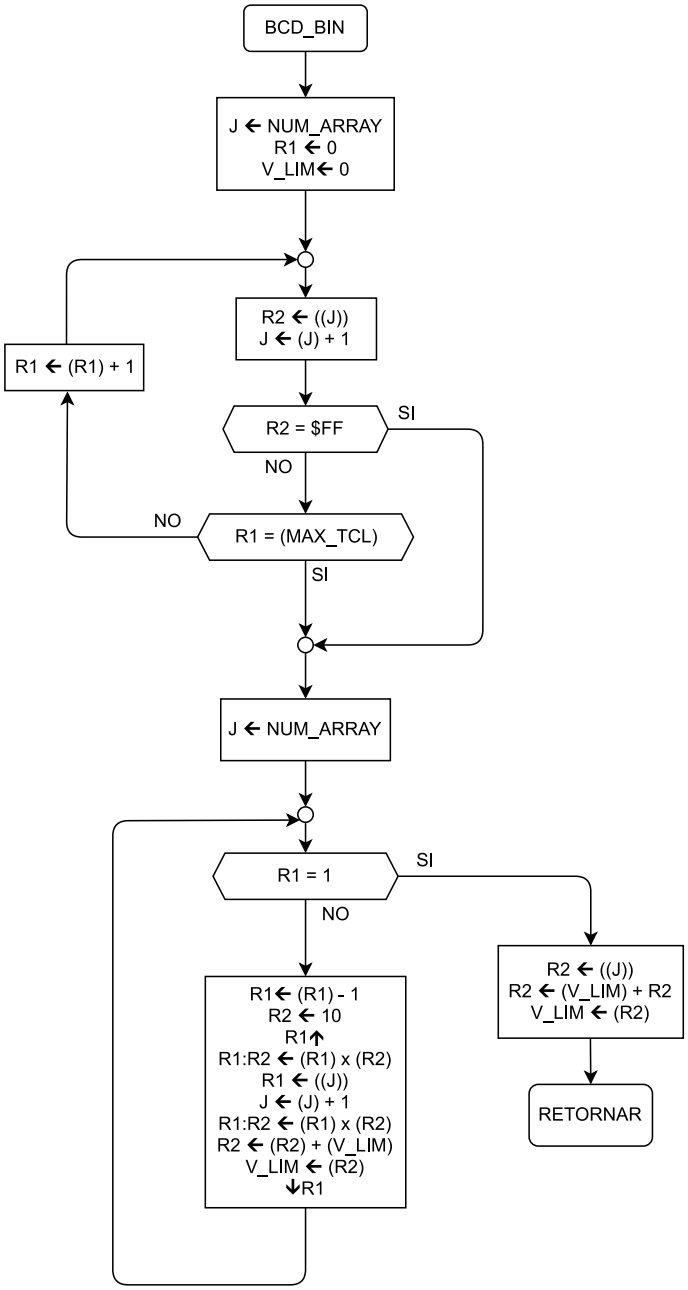

Figura 2.15: Diagrama de flujos para BCD_BIN.

2.15. Subrutina CONV_BIN_LCD

2.15.1. Diagrama y explicación de la subrutina CONV_BIN_LCD

Descripción

fasdf

Explicación

Diagrama

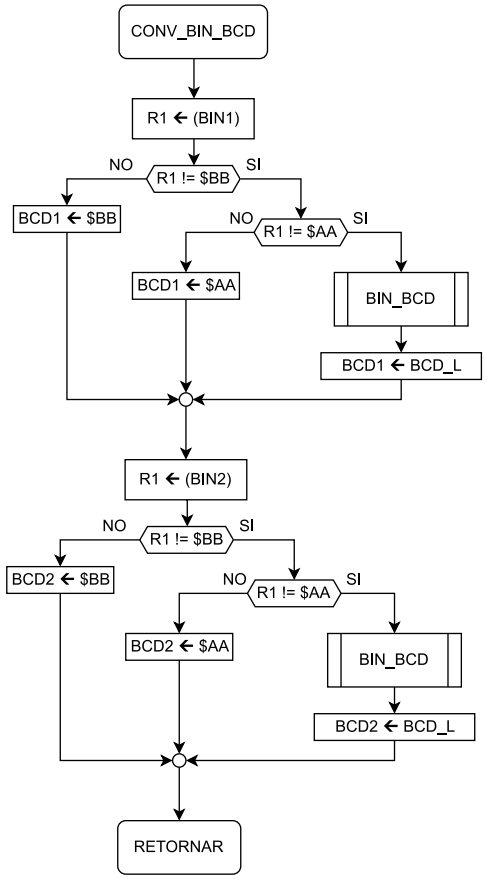

Figura 2.16: Diagrama de flujos para CONV_BIN_LCD.

2.16. Subrutina BIN_BCD

2.16.1. Diagrama y explicación de la subrutina BIN_BCD

Descripción

fasdf

Explicación

Diagrama

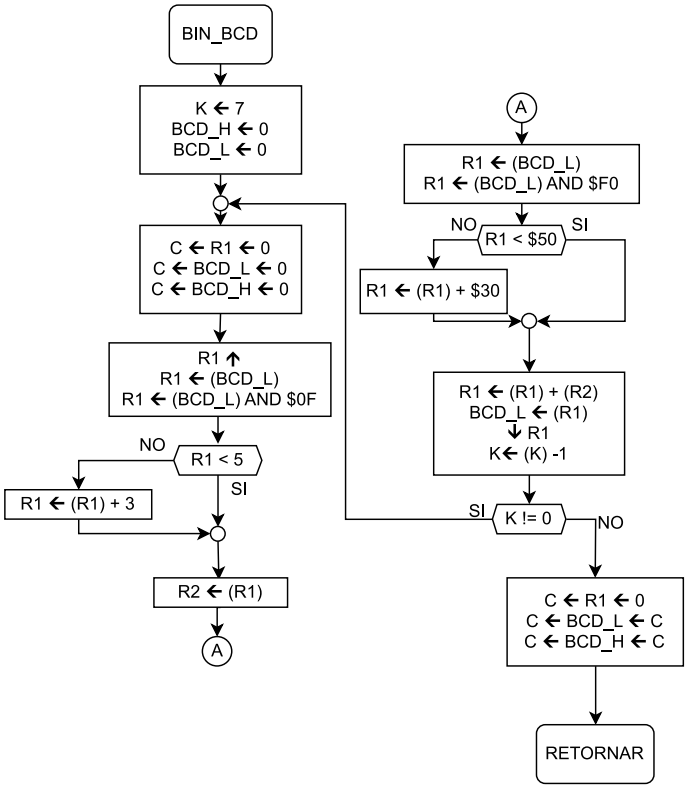

Figura 2.17: Diagrama de flujos para BIN_BCD.

2.17. Subrutina BCD_7SEG

2.17.1. Diagrama y explicación de la subrutina BCD_7SEG

Descripción

fasdf

Explicación

Diagrama

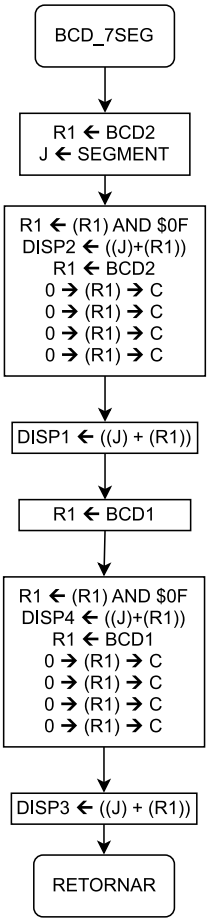

Figura 2.18: Diagrama de flujos para BCD_7SEG.

2.18. Subrutina DELAY

2.18.1. Diagrama y explicación de la subrutina DELAY

Descripción

fasdf

Explicación

Diagrama

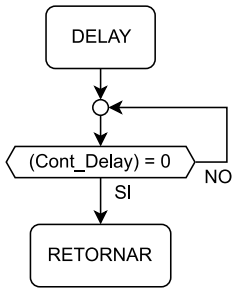

Figura 2.19: Diagrama de flujos para DELAY.

2.19. Subrutina SEND

2.19.1. Diagrama y explicación de la subrutina SEND

Descripción

fasdf

Explicación

Diagrama

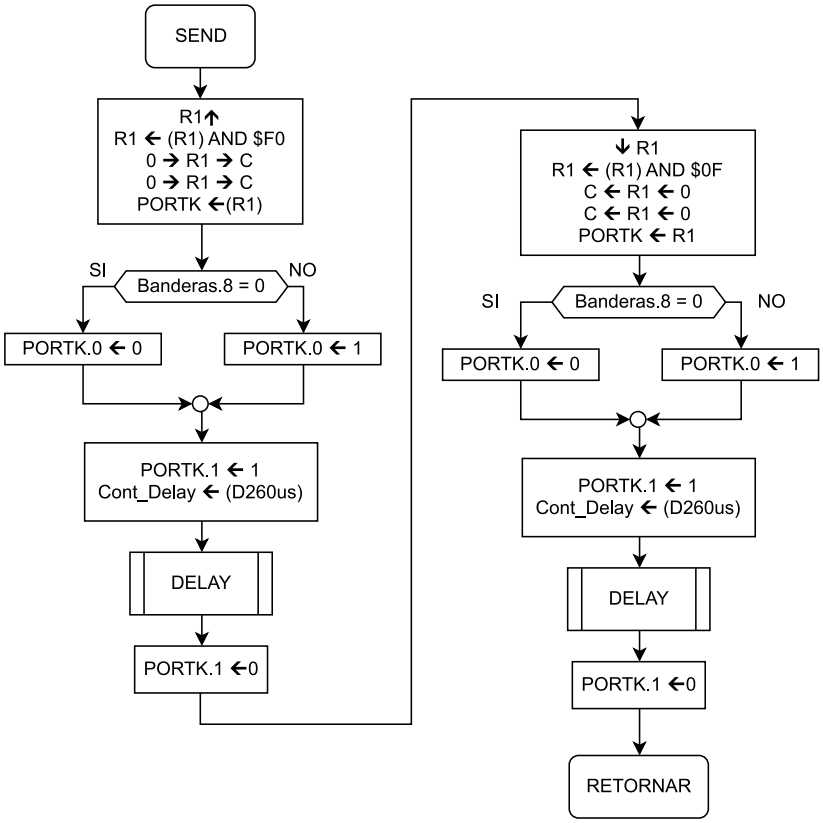

Figura 2.20: Diagrama de flujos para SEND.

2.20. Subrutina LCD

2.20.1. Diagrama y explicación de la subrutina LCD

Descripción

fasdf

Explicación

Diagrama

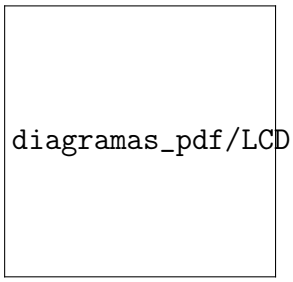

Figura 2.21: Diagrama de flujos para LCD.

2.21. Subrutina Cargar_LCD

2.21.1. Diagrama y explicación de la subrutina Cargar_LCD

Descripción

fasdf

Explicación

Diagrama

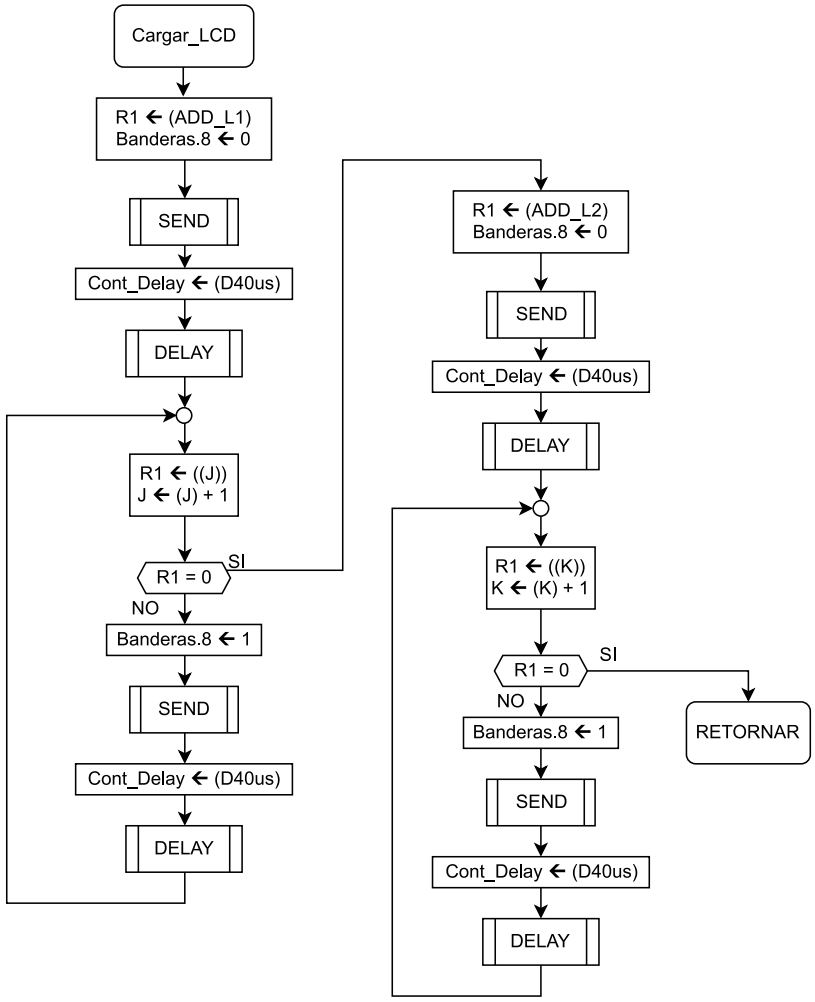

Figura 2.22: Diagrama de flujos para Cargar_LCD.

2.22. Subrutina PATRON_LEDS

2.22.1. Diagrama y explicación de la subrutina PATRON_LEDS

Descripción

fasdf

Explicación

Diagrama

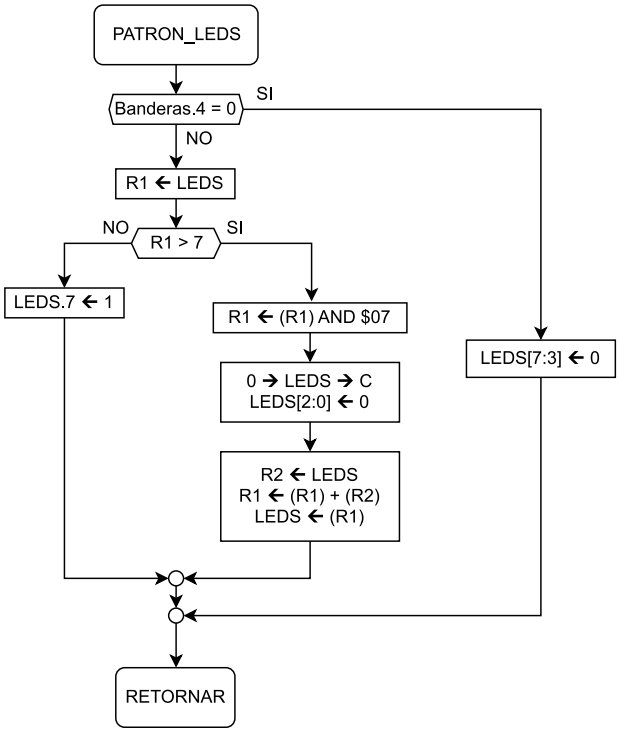

Figura 2.23: Diagrama de flujos para PATRON_LEDS.

2.23. Rutina MAIN

2.23.1. Configuración de MAIN

2.23.2. Diagrama y explicación de la rutina MAIN

Descripción

fasdf

Explicación

Diagrama

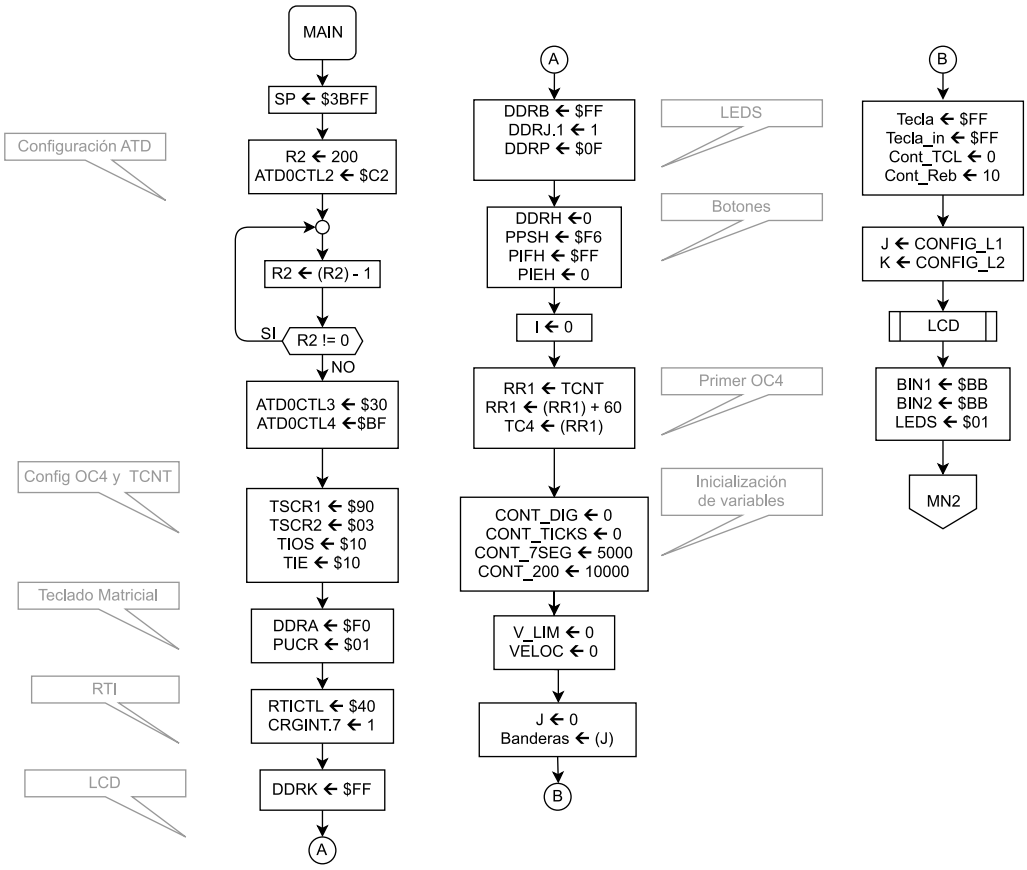

Figura 2.24: Diagrama de flujos para MAIN. Primera parte.

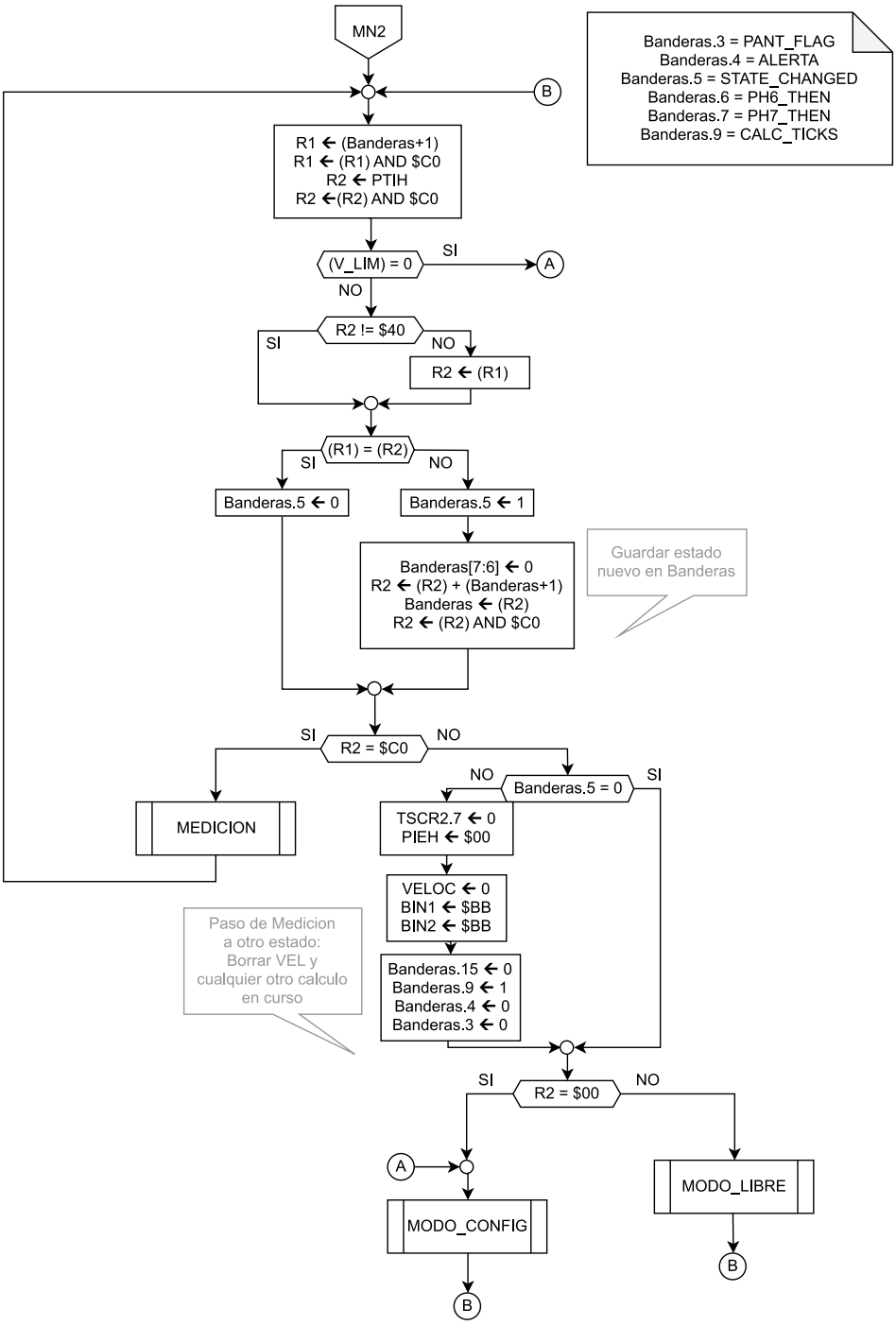

Figura 2.25: Diagrama de flujos para MAIN. Segunda parte.

2.24. VIEJO

2.24.1. Configuración para PH

Para hablitar todos los pines del puerto H como entradas, utilizamos el registro $DDRH$, escribiendo cero en todos los bits:

$$DDRH = \$00$$

Para habilitar las interrupciones para $PH3$ y $PH0$, utilizamos el registro $PIEH$:

$$PIEH = \$09 \tag{2.7}$$

Para definir la activación con flanco decreciente, se pone en cero los bits 3 y 0 del registro $PPSH$:

$$PPSH = \$F6$$

Finalmente, para borrar todas las banderas de interrupción, en caso de que haya alguna en cola:

$$PIFH = \$FF$$

2.24.2. Cálculos para PANT_CTRL

Lo primero, para lograr que $TICK\_DIS$ llegue a cero en dos segundos, entonces tenemos que cargar:

$$TICK\_DIS = 2seg \cdot \frac{46875}{1024} = 91{,}55 \approx 92$$

Ahora, en general, dado una velocidad $VELOC$, la cantidad de ticks que debemos cargar en contar para que el carro haya avanzado una distancia en metros DISTANCIA, se calcula de la siguiente manera:

$$TICKS = \frac{DISTANCIA}{VELOC} \cdot \frac{1km}{1000m} \cdot \frac{3600s}{1h} \cdot \frac{46875}{1024} \tag{2.8}$$

Ahora, particularmente, para calcular $TICK\_EN$, la $DISTANCIA = 60m$. Entonces tendríamos lo siguiente:

$$TICK\_EN = \frac{100m \cdot 1km \cdot 3600 \cdot 46875}{VELOC \cdot 1000m \cdot 1h \cdot 1024}$$

$$= \frac{16479{,}4921875}{VELOC}$$

$$\approx \frac{16480}{VELOC}$$

Por otro lado, para calcular $TICK\_DIS$, simplemente es el doble de $TICK\_EN$. Es decir:

$$TICK\_DIS = \frac{32958{,}984375}{VELOC}$$

$$\approx \frac{32959}{VELOC}$$

3 Conclusiones y recomendaciones

3.1. Conclusiones

3.2. Recomendaciones

Bibliografía

[1] Osorio M. (2011). *Los robots basados en una arquitectura deliberativa y la toma de decisiones*. México: Revista saberes compartidos.

[2] Urdiales C. & Bandera A. & Sandoval S. (2014). *Historia y tendencias actuales de la robótica*. España: Editorial Universidad de Málaga.

[3] Batle, J.A & Barjau A. (2008). *Holonomy in mobile robots*. España: Universidad de Catalunya, Barcelona.

[4] Oliveira, H. P., Sousa, A. J., Moreira, A. P., & Costa, P. J. (2009). Modelado de robots omnidireccionales de 3 y 4 ruedas. Contemporary Robotics: Challenges and Solutions.

[5] Carton Geek. (2016). Robot Omnidireccional. [online] Cartongeek.blogspot.com. Available at: http://cartongeek.blogspot.com/2016/02/robot-omnidireccional.html [Accessed 20 May 2017].

[6] Muñoz V. & Gil-Gómez G. & García A. *Modelado cinemático y dinámico de un robot móvil omnidireccional*. Málaga: Universidad de Málaga.

[7] Barrero L. & Villegas A. & Gómez D. *Robot transportador y distribuidor de objetos según su peso*. Redes ingeniería. Volumen 5.

[8] Martínez S. & Sisto R. *Proyecto de Grado: Control y Comportamiento de Robots Omnidireccionales*. Instituto de Computación. Facultad de Ingeniería - Universidad de la República Montevideo. Available at: https://www.fing.edu.uy/inco/grupos/mina/pGrado/easyrobots/doc/SOA.pdf[Accessed 05 June 2017]

[9] García D.(2012) *Modelado y Simulacion de un Robot Autónomo Omnidireccional*. Universidad Auntónoma de Querétaro.

[10] Barrero L. & Villegas A. & Gómez D. (2014) *Robot Transportador Omnidireccional* . AMDM 2014.

[11] Ramos E. & Morales R. & Silva R. (2010) *Modelado, simulación y construcción de un robot móvil de ruedas tipo diferencial*. México: CIDETEC.

[12] Rojas R. (2005) *Omnidirectional Control*. Alemania: Freie Universitat Berlin.

[13] Suárez A. & Sánchez A. *Plataforma Móvil omnidireccional de cuatro llantas suecas (Mecanum) en configuración AB*. México: Universdad Autónoma de México.

[14] V. F. Muñoz Martínez, *Modelado cinemático y dinámico de un robot móvil omnidireccional*. Instituto Andaluz de Automática Avanzada y Robótica.

[15] Autonomy Lab of Simon Fraser University, *Moving or Sensing, Time and Energy*. Vancouver, Canada.

[16] Cornwell J. *Mecánica Vectorial para Ingenieros DINÁMICA* Novena Edición.

[17] Williams, R., Carter, B. , Gallina, P., & Rosati, G. (2002). *Dynamic Model With Slip for Wheeled Omnidirectional Robots*. IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 18, NO. 3, JUNE 2002.