Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Robótica

Desarrollo y construcción de una tarjeta para la adquisición de datos y control de robots cooperativos.

T E S I S

que para obtener el grado de:

Ingeniero Eléctrico Electrónico

presenta:

Francisco Javier García Gutiérrez

director de tesis:

Dr. Marco Antonio Arteaga Pérez

Coyoacán Ciudad de México Mayo, 2016

Desarrollo y construcción de una tarjeta para la adquisición de datos y control de robots cooperativos.

por

Francisco Javier García Gutiérrez

Tesis presentada para obtener el grado de

Ingeniero Eléctrico Electrónico

en la

Facultad de Ingeniería

Universidad Nacional Autónoma de México

Coyoacán Ciudad de México. Mayo, 2016

A mis padres.

Agradecimientos

- A mi familia por la ayuda que siempre me han otorgado y por ser parte fundamental de lo que soy ahora.
- Al Doctor Marco Antonio Arteaga Pérez por su apoyo y haberme brindado la oportunidad de realizar esta tesis.
- Al proyecto DGAPA-PAPIIT IN116314 por el apoyo económico durante la realización de este trabajo.
- A la Universidad Nacional Autónoma de México mi alma mater por todos los conocimientos adquiridos.

Índice general

1. Introducción **2**

1.1. Estado del Arte 5

1.1.1. Robótica Experimental 5

1.1.2. Robótica Cooperativa 8

1.1.3. Adquisición de Datos 9

1.1.4. Tarjetas de Circuitos Impresos 11

1.2. Objetivos de la Tesis 11

1.3. Planteamiento del Problema de la Tesis 12

1.4. Contribución de la Tesis 12

1.5. Distribución de la Tesis 12

2. Problemática de diseñar una tarjeta para la adaptación de un robot industrial a un robot experimental **13**

2.1. Ingeniería Inversa 14

2.1.1. Terminología 14

2.1.2. Metodologías 16

2.2. Expectativas de la tarjeta 20

2.3. Problemáticas del diseño de la tarjeta 20

2.4. Metodologías 21

3. Especificaciones del Sistema **22**

3.1. Sistema original 22

3.1.1. Robots Industriales A255 y A465 23

3.1.2. Controlador C500 . . . 27
3.2. Modificaciones sobre el sistema original . . . 28
3.2.1. Inhibición de la etapa de control . . . 28
3.2.2. Botonera E-Stop Externa . . . 29
3.3. Elementos a interconectar por la tarjeta . . . 32
3.3.1. Sistema embebido CompactRIO . . . 32
3.3.2. Módulos NI 9263 y NI 9401 . . . 33
3.3.3. Fuente de alimentación Quint Power . . . 34
3.3.4. Botonera externa E-Stop . . . 34
3.4. Galería de Fotos . . . 35

4. Diseño de Hardware **40**
4.1. Tarjeta para la re-distribución de canales . . . 40
4.2. Etapa Digital: Robot A465 . . . 45
4.3. Etapa Digital: Robot A255 . . . 53
4.4. Etapa Analógica . . . 56
4.5. Botonera . . . 57
4.6. Etapa de potencia y protección . . . 61
4.7. Gabinete . . . 62
4.8. Galería de Fotos . . . 64

5. Aplicación **68**
5.1. Trayectorias deseadas . . . 69
5.2. Control *PID* . . . 71
5.2.1. Adquisición de Datos y Escritura de Señales Analógicas . . . 72
5.2.2. Interfaz de usuario . . . 76
5.2.3. Algoritmo de Control . . . 78

6. Resultados Experimentales **81**
6.1. Gráficas del Experimento . . . 82
6.1.1. Robot A255 . . . 82

6.1.2. Robot A465 87

7. Conclusiones **94**

7.1. Comentarios Finales 94

7.2. Trabajo a Futuro 95

A. Especificaciones de los Robots A255 y A465 **96**

A.1. Robot A255 98

A.2. Robot A465 100

B. Especificaciones del controlador C500 **102**

B.1. Conectores 104

B.1.1. DB-25 Internos 104

B.1.2. Conectores del Controlador C500 al Gabinete 108

B.1.3. Conectores del controlador C500 a los Robots 111

C. Especificaciones de elementos de la tarjeta y el gabinete **115**

C.1. CompactRIO NI-cRIO 9014 116

C.1.1. Módulo NI 9263 118

C.1.2. Módulo NI 9401 119

C.2. Quint Power PS-100 121

C.3. Características Físicas del Gabinete 122

C.4. Tarjeta 123

C.4.1. Integrado 26LS32 123

C.4.2. Integrado HCPL2630 124

C.4.3. Integrado 4N35 126

C.4.4. Integrado TIP31C 127

D. Software y Código **128**

D.1. Creación de proyecto FPGA CompactRIO 129

D.2. Proyecto en LabVIEV para la adquisición de datos y escritura de señales analógicas 135

D.3. Creación de proyecto en *Visual Studio* 146

D.4. Código para la Interfaz Gráfica 152

Índice de tablas

3.1. Especificaciones de los robots A465 y A255. 24
3.2. Características de Corriente y Voltaje de la etapa de potencia de los robots . . . 25

4.1. Tabla de verdad del circuito 26LS32 [26]. 46

5.1. Descripción de Sub-VIs. 72

1.1. Especificaciones de los robots A465 y A255 [15][16]. 97

2.1. Asignación de pines del conector DB-25[I] del controlador C500. 104
2.2. Asignación de pines del conector DB-25[II] del controlador C500. 105
2.3. Asignación de pines del conector DB-25[1] del controlador C500. 106
2.4. Asignación de pines del conector DB-25[2] del controlador C500. 107
2.5. Asignación de pines del conector DB-25[AB] del controlador C500. 108
2.6. Asignación de pines del conector DB-25[Z] del controlador C500. 109
2.7. Asignación de pines del conector DB-25[VCOM] del controlador C500. 110
2.8. Asignación de pines del conector de Potencia de los Motores del controlador C500 (Robot A255). 111
2.9. Asignación de pines del conector de Potencia de los Motores del controlador C500 (Robot A465). 112
2.10. Asignación de pines del conector redondo de Realimentación del controlador C500. 113
2.11. Asignación de pines del conector redondo de Realimentación del controlador C500. 114

3.1. Especificaciones del Sistema CompactRIO NI-cRIO-9014 [20]. 117
3.2. Especificaciones del Sistema CompactRIO NI-cRIO-9014 [20]. 118

3.3. Especificaciones del módulo NI-9263 [21]. 118
3.4. Especificaciones del módulo NI-9401 [22]. 119
3.5. Especificaciones del módulo NI-9401 [22]. 120
3.6. Especificaciones de la unidad Quint Power PS-100 [23]. 121
3.7. Especificaciones del integrado 26LS32 [26]. 123
3.8. Especificaciones del integrado HCPL 2630 [27]. 124
3.9. Especificaciones del integrado 4N35 [28]. 126

Índice de figuras

1.1. Suministro Mundial Anual de Robots Industriales 2005-2014 [3]. 4
1.2. Equipo existente en el laboratorio LAR-DEIS [4]. 6
1.3. Equipo actualmente utilizado en el laboratorio PRISMA [6]. 7
1.4. Tarjetas de adquisición de datos y tarjetas de desarrollo. 10

2.1. Relación entre términos [10]. 15
2.2. Proceso de ingeniería inversa usado en BOS/X [11]. 17
2.3. Proceso de Redoff para la ingeniería inversa [12]. 18
2.4. Proceso REV-ENGE para la ingeniería inversa [13]. 19

3.1. Disposición general de un controlador industrial. 23
3.2. Componentes de fábrica de los robot A255 y A465 [15][16]. 24
3.3. Diagrama de bloques del sistema de los actuadores. 25
3.4. Señal de codificador en cuadratura con índice. 26
3.5. Parte trasera del controlador C500 otorgado de fábrica. 27
3.6. Parte trasera del controlador C500 con conectores DB-25 [I], [II], [1] y [2]. 28
3.7. Disposición interna del controlador C500 con inhibición del controlador [19]. . . . 30
3.8. Circuito de paro de emergencia del controlador C500. 31
3.9. CompactRIO National Instruments NI-cRIO 9014 [20]. 32
3.10. Módulos utilizados para la adquisición de los datos [21][22]. 33
3.11. Fuente de alimentación Quint Power [23]. 34
3.12. Foto del Robot A465. 35
3.13. Foto del Robot A255. 36

3.14. Foto de los Robots A255 y A465. 36
3.15. Controlador C500 del robot A465. 37
3.16. Botonera externa para paro de emergencia. 37
3.17. Sistema CompactRIO NI-cRIO 9014. 38
3.18. Módulos NI-9263. 38
3.19. Módulos NI-9401. 39
3.20. Unidad Quint Power PS-100. 39

4.1. Esquema de Conexiones de la tarjeta para la re-distribución de canales del controlador C500 41
4.2. Tarjeta para la re-distribución de canales del controlador C500. 43
4.3. Re-distribución de canales del controlador C500. 44
4.4. Gráfica de la señal A de la primera articulación del robot A465. 45
4.5. Diagrama lógico del circuito 26LS32. 46
4.6. Gráfica de la señal A no balanceada de la primera articulación del robot A465. . 47
4.7. Circuito HCPL2630 con elementos externos [27]. 48
4.8. Gráfica de la señal A no balanceada y opto-acoplada de la primera articulación del robot A465. 49
4.9. Circuito para el acondicionamiento de señales A, B y Z del robot A465 parte I. . 50
4.10. Circuito para el acondicionamiento de señales A, B y Z del robot A465 parte II. 51
4.11. Circuito para el acondicionamiento de señales HSW del robot A465. 52
4.12. Gráficas de la señal A de la primera articulación del robot A255. 53
4.13. Circuito para el acondicionamiento de señales A, B y Z del robot A255 parte I. . 54
4.14. Circuito para el acondicionamiento de señales A, B y Z del robot A255 parte II. 55
4.15. Conexiones de la etapa analógica. 56
4.16. Integrado 4N35 y elementos externos [28]. 57
4.17. Micro-controlador MSP430g2553. 58
4.18. Circuito para controlar un relé [29]. 58
4.19. Circuito para el control de la botonera. 60
4.20. Header y jumpers utilizados en la tarjeta. 61
4.21. Circuito para protección de la tarjeta. 61

4.22. Circuito interno del gabinete. 62
4.23. Tarjeta para la adquisición de datos y control de los robots. 63
4.24. Tarjetas sin ensamblar. 64
4.25. Tarjeta ensamblada del controlador C500 para el ordenamiento de señales. 65
4.26. Interior del controlador C500 e inhibición de etapa de control 65
4.27. Tarjeta para la adquisición de señales y control ensamblada 66
4.28. Circuito interno del gabinete sin conectores externos 67

5.1. Posición de *HOME* de los robots. 68
5.2. Trayectoria deseada de la primera articulación del robot A255 y sus derivadas . . 70
5.3. Diagrama a bloques de un sistema con controlador PID. 71
5.4. Fragmento del programa para controlar la botonera. 73
5.5. Fragmento del programa para determinar la frecuencia de muestreo. 73
5.6. Diagrama de flujo del proceso de interpretación de datos y escritura de señales analógicas para una articulación. 74
5.7. Programa base para la interpretación de datos y escritura de señales analógicas para una articulación. 75
5.8. Interfaz de usuario gráfica. 77

6.1. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la primera articulación del robot A255. 82
6.2. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la segunda articulación del robot A255. 83
6.3. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la tercera articulación del robot A255. 84
6.4. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la cuarta articulación del robot A255. 85
6.5. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la quinta articulación del robot A255. 86
6.6. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la primera articulación del robot A465. 87

6.7. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la segunda articulación del robot A465. 88
6.8. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la tercera articulación del robot A465. 89
6.9. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la cuarta articulación del robot A465. 90
6.10. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la quinta articulación del robot A465. 91
6.11. Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la sexta articulación del robot A465. 92
6.12. Distribución final del sistema. 93

1.1. Articulaciones del robot A255 [16]. 98
1.2. Alcance en vista de planta del robot A255 [16]. 98
1.3. Alcance del robot A255 [16]. 99
1.4. Alcance del robot A255 con Gripper [16]. 99
1.5. Articulaciones del robot A465 [15]. 100
1.6. Alcance en vista de planta del robot A4655 [15]. 100
1.7. Alcance del robot A465 [15]. 101
1.8. Alcance del robot A465 con Gripper [15]. 101

2.1. Medidas del controlador C500 [18]. 103
2.2. Vista trasera del controlador C500. 103
2.3. Conector DB25 Macho-Hembra. 104
2.4. Conector DB25 Macho-Hembra 105
2.5. Conector DB25 Macho-Hembra 106
2.6. Conector DB25 Macho-Hembra. 107
2.7. Conector DB25 Macho-Hembra 108
2.8. Conector DB25 Macho-Hembra 109
2.9. Conector DB25 Macho-Hembra 110
2.10. Conector Redondo de 14 pines 111

2.11. Conector Redondo de 24 pines 112
2.12. Conector Redondo de 57 pines 113
2.13. Conector Redondo de 57 pines 114

3.1. Panel frontal del sistema CompactRIO [20]. 116
3.2. Dimensiones del sistema CompactRIO [20]. 116
3.3. Módulo NI 9263. [21] 119
3.4. Módulo NI 9401 [22]. 120
3.5. Diagrama a bloques de la unidad Quint Power PS-100 [23]. 121
3.6. Tapa lateral del gabinete. 122
3.7. Conexión a tierra del gabinete y tapa lateral desprendida. 122
3.8. Voltaje de salida vs Corriente de salida [26] 123
3.9. Diagrama de tiempos de la señal de entrada y salida [27]. 124
3.10. Gráficas del comportamiento del integrado HCPL-2630 respecto a R_L [27]. 125
3.11. Gráficas del comportamiento del integrado 4N35 respecto a R_L [28]. 126
3.12. Gráficas del comportamiento típico del integrado TIP31 [29]. 127

4.1. Menu principal 129
4.2. Sub-menu de nuevo elemento. 130
4.3. Ventana de selección de FPGA. 131
4.4. Ventana de creación de nuevo proyecto FPGA 131
4.5. Ventana de estado del FPGA. 132
4.6. Mensaje de error por ausencia de FPGA. 132
4.7. Mensaje de advertencia de FPGA usado. 133
4.8. Ventana final para la creación del proyecto. 133
4.9. Ventana de proyecto creado. 134
4.10. Diagrama de flujo del sub-vi *Limits* 135
4.11. Sub-VI *Limits* 135
4.12. Diagrama de flujo del sub-vi *Increase* 136
4.13. Sub-VI *Counter Increase* 137
4.14. Diagrama de flujo del sub-vi *Decrease* 138

4.15. Sub-VI *Counter Decrease* 139
4.16. Diagrama de flujo del sub-vi *Counter Z* 140
4.17. Sub-VI *Z Counter* 141
4.18. Programa para la interpretación de datos y escritura de señales analógicas parte I 142
4.19. Programa para la interpretación de datos y escritura de señales analógicas parte II143
4.20. Programa para la interpretación de datos y escritura de señales analógicas parte III 144
4.21. Programa para la interpretación de datos y escritura de señales analógicas parte IV145
4.22. Menu de nuevo proyecto. 146
4.23. Overview MFC Application Wizard. 147
4.24. Appication Type MFC Application Wizard. 147
4.25. User Interface Features MFC Application Wizard. 148
4.26. Advanced Features MFC Application Wizard. 148
4.27. Generated Classes MFC Application Wizard. 149
4.28. Incluir archivos al proyecto. 150
4.29. Proyecto con archivos correctamente agregados. 150
4.30. Ventana para generar API en *LabVIEW*. 151
4.31. Ventana final para la creación del proyecto. 151

Capítulo 1

Introducción

"Robots of the world! Many people have fallen.
By seizing the factory we have become the masters of everything.
The age of mankind is over. A new world has begun! The rule of Robots!"
— Karel Čapek, *Robots Universales Rossum* (1920)

La historia del ser humano siempre ha sido impulsada por los sueños y personas que hacen realidad dichos sueños, pero todo sueño tiene sus cimientos basados en la realidad, y la realidad para el año de 1920 en cuanto a tecnología se refiere, fue el comienzo del uso en masa de motores eléctricos y de motores impulsados con el uso de combustibles fósiles, los cuales dejaban atrás la era de las máquinas de vapor. Esta realidad hizo posible en la mente de Karel Čapek la creación de la obra teatral de ciencia ficción R.A.R. (*Robots Universales Rossum*) e introdujo al mundo la palabra *Robot*, haciendo referencia a seres electro-mecánicos muy parecidos a los seres humanos, un concepto que actualmente puede ser destinado a cyborgs o androides. La idea de la creación de estos seres ha prevalecido desde entonces, de tal forma que para el año de 1939, la empresa Westinghouse presentó en la feria mundial a Elektro, el nombre asignado al primer robot de aspecto humanoide exhibido públicamente, el cuál era capaz de realizar 26 movimientos diferentes, así como fumar y caminar, pero ejecutados de forma burda y con ayuda humana. Sin embargo pequeños logros como éste, dejaban volar la imaginación y el ingenio, haciendo pensar a las personas en la posibilidad de tecnologías más avanzadas cayendo en ocaciones en el ámbito de la ciencia ficción y muchas veces con un enfoque literario. Un ejemplo de ello es el escritor Issac Asimov, quién en sus obras realizó aportaciones además de culturales, científicas[1]. En su obra "Círculo Vicioso de Asimov" (1942), declara las tres leyes fundamentales de la robótica en

donde se dictan las condiciones de la relación robot-humano y expone generalmente a los robots como entidades programadas. Por su parte, George Devol, en 1948 solicitó la patente del primer robot manipulador con tal capacidad. Fue en el año de 1954 que se otorgó la patente del primer robot industrial programable [2].

Hoy en día la ciencia ficción y la realidad siguen estando separadas, pero ésta diferencia es la que sigue impulsando a ingenieros, científicos e investigadores a continuar con sus metas e ideales. Sin embargo, la robótica ha tomado varios caminos y los robots pueden ser clasificados por su morfología, por su aplicación o por su procedimiento de control. Un robot por su morfología puede ser móvil, como orugas, con patas, acuáticos, voladores o con ruedas. Puede ser fijo como los robots poliarticulados (robots manipuladores y algunos robots industriales) o puede ser híbrido el cual se constituye por un robot móvil y uno fijo.

Los robots industriales son manipuladores programables, controlados automáticamente y reprogramables. Son capaces de realizar varias operaciones distintas de forma simultánea o consecutiva sin necesidad de intervención humana.

Aunque muchos robots industriales han cambiado muy poco físicamente, los avances que éstos han presentado en aspectos mecánicos, electrónicos y de control, los han hecho más robustos, rápidos y fiables, acoplandose correctamente a la industria para la realización de trabajos repetitivos y arduos.

> *"La demanda mundial de robots industriales se ha disparado en 2014, llegando por primera vez en su historia a superar las 200,000 unidades"*
> — Arturo Baroncelli, *International Federation of Robotics* (2015)

Arturo Baroncelli[3], habla del gran crecimiento de la demanda de robots industriales, la cuál se debe en gran medida al fuerte impulso de la industria automotriz, seguida por la electrónica de consumo. De acuerdo a los resultados preliminares de las estadísticas mundiales de robots industriales, que elabora la Federación Internacional de Robótica, se estima que en 2014 se vendieron alrededor de 225,000 unidades, lo que supone un incremento del 27 % respecto al 2013 (Figura 1.1).

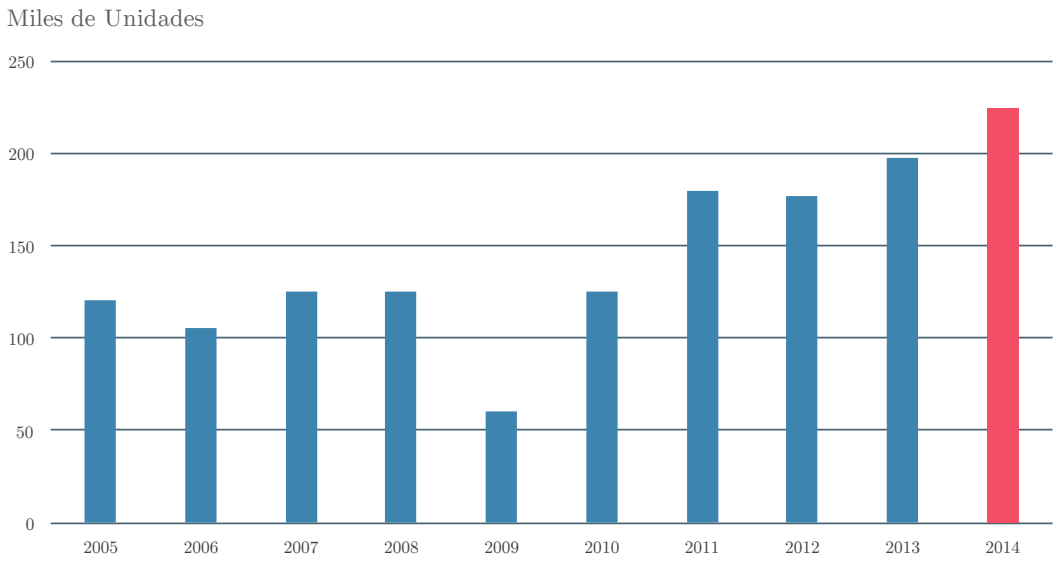


Figura 1.1: Suministro Mundial Anual de Robots Industriales 2005-2014 [3].

Por lo anterior, la robótica es una ciencia que, a pesar de los grandes avances que ha tenido, seguirá sufriendo modificaciones y creciendo debido a su gran demanda. Así mismo impulsa a realizar, algoritmos de control cada vez mas complejos tales como control adaptable, difuso, neuronal, métodos de pasividad o control por modos deslizantes.

Una de las ventajas de contar con laboratorios de robótica enfocados en analizar algoritmos de control es llevar la teoría a la práctica y probar su desempeño en condiciones reales. A pesar de las ventajas que esto conlleva, en México existen pocos laboratorios enfocados al control de robots industriales ya que su costo de creación es muy elevado. Por ello los robots con los que se cuentan son generalmente para propósitos educativos.

Hoy en día es necesario en las universidades contar con laboratorios que trabajen con robots experimentales. La disponibilidad de comprar un robot es únicamente limitada por el costo, ya que existen fábricas y distribuidoras en todo el mundo.

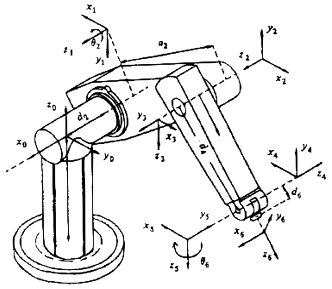
1.1 | Estado del Arte

En los últimos años, los mayores esfuerzos de la investigación en la robótica han sido enfocados en mejorar la actuación de robots móviles individuales utilizando sensores, actuadores y algoritmos de control avanzados. Como resultado, los robots individuales se han vuelto sofisticados, capaces de realizar tareas arduas o difíciles requeridas por aplicaciones del mundo real. No obstante muchas de estas tareas aún siguen siendo complicadas para éstos. Una alternativa para la solución de estos problemas es utilizar un conjunto de robots más simples para lograr tareas complejas mediante el funcionamiento cooperativo.

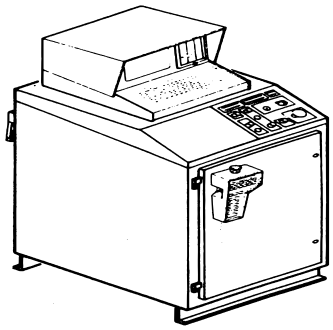
1.1.1 | Robótica Experimental

De las distintas referencias encontradas de laboratorios enfocados en la robótica industrial se encuentran dos universidades que cuentan con sistemas de robots experimentales basados en robots industriales en los que se llevan a cabo prácticas y proyectos en donde se ve involucrada la robótica cooperativa. Ambos laboratorios se encuentran en Italia. El primero es el Laboratorio de Automatización y Robótica del Departamento de Electrónica, Ciencias de la Computación y Sistemas de la universidad de Bologna (LAR-DEIS)[4], que dentro de su equipo cuenta con dos sistemas manipuladores, el primer sistema es el robot UNIMATE Puma 562 MARK III (Figura 1.2a) con tres grados de libertad utilizado para probar algoritmos de control utilizando el controlador VAL (Figura 1.2b) para programar al robot y el segundo es el sistema basado en un robot industrial SMART 3-S de COMAU (Figura 1.2c), un controlador C3G-9000 (Figura 1.2d), una pinza y una cámara de video. El sistema, que desde el año 2000 está en funcionamiento, fue adaptado originalmente para probar algoritmos de control con retroalimentación por visión utilizando una plataforma de Linux en tiempo real. Posee la capacidad de comprobar diferentes algoritmos de control como se expone en *A. Macchelli and C. Melchiorri (2002)* [5].

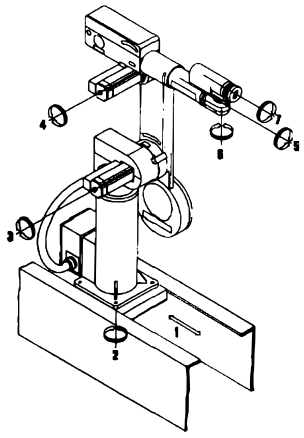
El segundo laboratorio pertenece a la División de Robots Industriales del Laboratorio PRISMA (Projects of Robotics for Industry and Services, Mechatronics and Automation) en el Departamento de Ingeniería Eléctrica y Tecnología de la Información de la Universidad de Nápoles



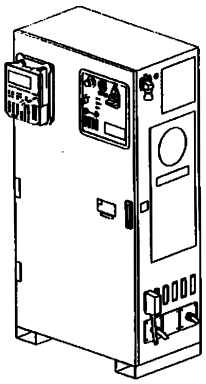
(a) Puma 562 MARK III



(b) Controlador VAL



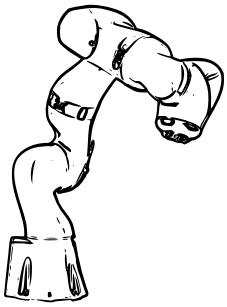
(c) Smart 3-S COMAU



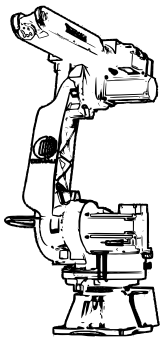
(d) Controlador C3G 9000

Figura 1.2: Equipo existente en el laboratorio LAR-DEIS [4].

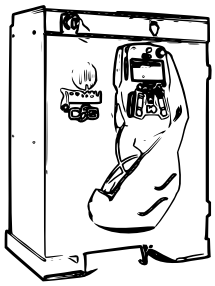
Federico II, coordinado por el profesor Bruno Siciliano. Cuentan con el robot KUKA LWR 4 (Figura 1.3a) de siete grados de libertad con el cual realizan experimentos de control. Anteriormente el laboratorio contaba con un sistema de robots cooperativos, dicho sistema se basaba en dos robots industriales SMART 3-S de COMAU (Figura 1.2c) de seis grados de libertad cada uno y dos controladores C3G-9000 (Figura 1.2d), pero el principal proyecto resulta de la actualización del equipo empleado y es el sistema de robots cooperativos que está constituido por un par de robots industriales COMAU Smart5 Six (Figura 1.3b). Cada robot manipula-



(a) LWR 4 KUKA



(b) Smart5 Six COMAU



(c) Controlador C5G Plus

Figura 1.3: Equipo actualmente utilizado en el laboratorio PRISMA [6].

dor cuenta con una estructura cinemática con seis articulaciones angulares. Los ejes de las tres articulaciones exteriores se juntan en una muñeca esférica. Los controladores ahora utilizados son los C5G plus (Figura 1.3c), el cual, al igual que el C3G-9000, acepta programación a nivel nativo.

Con estos ejemplos se observa que gracias a la versatilidad de los controladores de fábrica de COMAU y KUKA se ha podido lograr el manejo experimental de los robots en estas universidades, permitiendo así la elaboración de controladores más sofisticados. En el Laboratorio de Robótica del Departamento de Control y Robótica de la División de Ingeniería Eléctrica de la Universidad Nacional Autónoma de México se busca desarrollar de igual forma controladores más robustos y probarlos con el par de robots CRS con los que se cuenta y enfocar el desarrollo principalmente para los sistemas de robots cooperativos.

1.1.2 | Robótica Cooperativa

La robótica cooperativa o también llamada robótica colectiva busca diseñar sistemas compuestos de varios robots capaces de resolver problemas de manera conjunta. Los robots que forman parte de un sistema multi - robot son simples en términos de diseño y control, y menos costosos que los sistemas de un sólo robot especializado qué en muchos casos implica mejores sensores o sistemas mecánicos más complicados. Esos sistemas multi-robot están orientados a resolver problemas en los cuales la participación de un solo robot no es suficiente o resulta ser muy costosa, en términos de diseño y tiempo, por ejemplo el transporte de objetos voluminosos, el manejo de material peligroso, la exploración y cobertura de terreno.

A medida que aumenta el número de robots individuales, dependiendo de la complejidad de la tarea, la complejidad del control del sistema también aumenta. Los métodos utilizados para manejar tal complejidad incluyen el control centralizado y el control distribuido.

En el caso de control distribuido o descentralizado se equipa a cada robot con multiples sensores y un controlador para que reconozca su entorno y planifique su trayectoria con los datos recaudados por si mismo. El robot es programado para generar nuevas rutas sin esperar orden alguna, logrando así un sistema más dinámico y rápido en donde el centro de control, si es que existe, es limitado al flujo de información y a la asignación de tareas del sistema. En caso de que alguno de los robots deje de funcionar no inhabilita a los demás robots del sistema.

En el control centralizado, los robots que integran el sistema, son manipulados por una unidad central, los datos recabados por estos, son enviados al centro de control el cual se encarga de realizar la planificación además de atender los movimientos y conflictos. La desventaja de este método se ve reflejado en el momento en el que deje de operar la unidad central repercutiendo la inactividad de éste en todo el sistema.

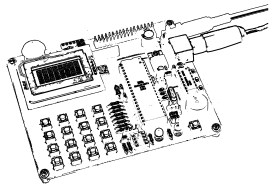
Todos los *organismos* tienen su propia inteligencia pero además tienen capacidad de comunicación, y por lo tanto, de cooperación y coordinación [7]. De esta manera se tiene que la inteligencia colectiva es la suma de la inteligencia de cada individuo. Cada robot debe tener la capacidad de transferir información a otro compañero y tener la suficiente inteligencia para pedir ayuda y coordinar los movimientos con los demás individuos. Esto se logra cuando un sistema es completamente acoplado. En los sistemas poco acoplados o manadas los robots fun-

cionan independientemente del resto, sin tener en cuenta los movimientos o decisiones de los demás. No hay una inteligencia central sino que todo el control es distribuido, cada elemento simplemente sigue una o varias reglas especificas. Este tipo de comportamiento trata de imitar sistemas biológicos como el de los insectos sociales.

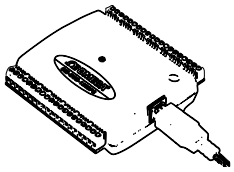
1.1.3 | Adquisición de Datos

Detrás de cada sistema digital que interactúa con el mundo real, de cada robot y del sistema que lo compone, se encuentra el proceso de adquisición de señales. El propósito de la adquisición de datos es el medir los variables físicas como voltaje, corriente, presión o sonido. Mientras que cada sistema de adquisición de datos es definido por los requerimientos de su aplicación. Todos los sistemas comparten un mismo objetivo final de adquisición, análisis y procesamiento de información. Los sistemas de adquisición de datos incorporan sensores, actuadores, acondicionadores de señal, dispositivos de adquisición de datos y aplicaciones de software [8].

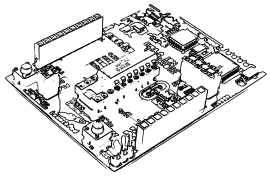
La adquisición de datos recolecta las señales de las fuentes medibles y digitaliza la señal para almacenarla, analizarla o aplicar algún algoritmo de control. Los diferentes sistemas de detección de señales requieren de cierto desarrollo para lograr la correcta adquisición de la señal. Una solución ideal es el recolectar y si es posible analizar la señal en proceso inmediatamente después de ser detectada y así la información recolectada podrá ser transmitida mediante un bus digital a la unidad central para ser procesada. El sensor a elegir depende del proceso y del parámetro a ser medido. Generalmente la salida del sensor puede ser voltaje o corriente, el cual, tiene que ser digitalizado. Con la señal finalmente digitalizada, el dato puede ser guardado, analizado o aplicado a un control, dependiendo del objetivo para el cual fue generado. En la robótica los principales sensores suelen ser de posición, fuerza o distancia, pero en el mercado y en la aplicación existen distintos tipos de detectores que cumplen con los requerimientos para solucionar un problema.



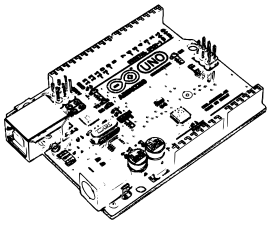
(a) Tarjeta de adquisición de datos con uC PIC



(b) Módulo de adquisición de datos OM-USB-1208FS 8 canales USB

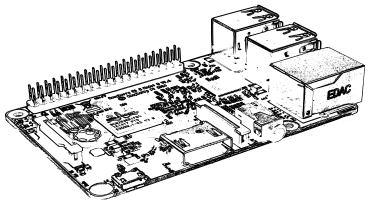


(c) Tarjeta de desarrollo Launchpad con uC MSP-430

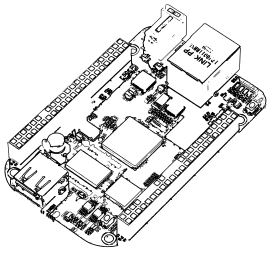


(d) Tarjeta de desarrollo Arduino UNO con uC ATMEL

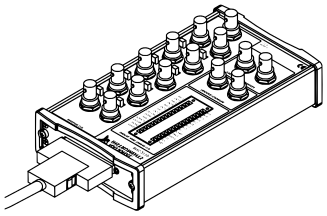
(e) Módulo de adquisición de datos NI USB-6000



(f) SBC Raspberry PI



(g) SBC Beaglebone Black



(h) Dispositivo de adquisición de datos NI BNC-2110

Figura 1.4: Tarjetas de adquisición de datos y tarjetas de desarrollo.

Hoy en día el hardware empleado para la adquisición de datos se ha desarrollado de tal forma que ha dejado de ser solo un dispositivo para entradas y salidas digitales y/o analógicas con protocolo de comunicación para ser conectado a un CPU. Los nuevos procesadores, microcontroladores (uC) y memorias han hecho permisible el desarrollo de computadoras completas dentro de una sola tarjeta de circuito impreso (SBC por sus siglas en inglés *Single Board Computer*) a un tamaño y costo que permite recolectar y digitalizar la información, almacenarla, analizarla y aplicar un algoritmo de control por sí sola. Es importante mencionar que cada una de estas tecnologías, por muy similares que parezcan, su elección depende del proyecto a realizar.

1.1.4 | Tarjetas de Circuitos Impresos

La elaboración de tarjetas de circuitos impresos o PCBs (Printed Circuit Boards) es la forma más común de conjuntar un circuito. Es un grupo de una o más capas en donde se encuentran distribuidos los elementos que conforman al sistema de parámetros concentrados conectados mediante pistas de cobre. El diseño del circuito impreso puede ser tan importante como el diseño del circuito, dependiendo de la complejidad del sistema, teniendo como problemas la interconexión de trazos, corrientes parásitas, caídas de voltaje, interferencia (fenómeno también conocido como crosstalk) influencias de la capacitancia generada por el acumulamiento de planos del circuito que hace las veces de un capacitor, efectos parasitos por el acumulamiento de humedad entre otros[9]. Para la etapa de adquisición de datos de sistemas digitales como el de robots cooperativos es importante considerar el diseño de la tarjeta para disminuir el error real entre la planta y la referencia establecida en el controlador del sistema.

1.2 | Objetivos de la Tesis

Esta tesis tiene como objetivo hacer permisible el desarrollo de algoritmos de control más sofisticados para sistemas de robots cooperativos mediante la elaboración de una tarjeta que facilite la comunicación de dos robots industriales A255 y A465 de la marca CRS acondicionados como robots experimentales con una unidad central. El control de estos robots se logrará mediante la adquisición de datos con ayuda del sistema embebido NI CompactRIO y la funcionalidad de la tarjeta se probará mediante la implementación de algoritmos de control.

1.3 | Planteamiento del Problema de la Tesis

La mayoría de los fabricantes y distribuidores de robots industriales ofrecen un control en sus productos de alto nivel, es decir, el controlador no llega a desarrollarse a profundidad y por ello, en muchos casos, el robot se ve incapaz de obedecer leyes de control en tiempo real u opciones de control más elaboradas debido a que el hardware y software de éstos se ve limitado a los ajustes realizados desde fábrica.

Debido a que el problema deriva del impedimento de los robots para obedecer leyes de control más sofisticados o en tiempo real a causa de la tarjeta proporcionada por el distribuidor se propone hacer re-ingeniería para el rediseño parcial o total de la tarjeta.

Con el objetivo de un mejor análisis en la problemática del desarrollo y la construcción de una tarjeta para la adquisición de datos y control de robots cooperativos, en el Capítulo 2 se aborda con mayor profundidad este tema.

1.4 | Contribución de la Tesis

Al finalizar este proyecto se debe experimentar con algoritmos y métodos de control más avanzados que los proporcionados por la fábrica dentro del Laboratorio de Robótica de la División de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México.

1.5 | Distribución de la Tesis

El presente trabajo se conforma en cuatro partes, la primera se compone por el Capítulo 1 y el Capítulo 2, donde se expone la problemática a tratar en la tesis, así como proponer una metodología para brindar una solución. En los Capítulos 3 y 4 se recaba información sobre el sistema y con base en ésta se brinda una solución. Para poder elaborar una comprobación del trabajo realizado, en los Capítulos 5 y 6 se muestra un experimento y se dan a conocer los resultados obtenidos. Finalmente se presentan las conclusiones del trabajo realizado.

Capítulo 2

Problemática de diseñar una tarjeta para la adaptación de un robot industrial a un robot experimental

El problema de adaptar un robot industrial a un robot experimental se debe a que las especificaciones internas del sistema original no están disponibles para el usuario y son parte del *"know how"* que las empresas utilizan para proteger su producto de la piratería o ataques a su equipo limitándonos por defecto del conocimiento. Esta situación deriva en tener que aplicar técnicas de ingeniería inversa y re-ingeniería para alcanzar un entendimiento del diseño del sistema debido a que la fuente de información del fabricante no es abierta y no está disponible para realizar el proceso de rediseño. Las herramientas de la ingeniería inversa extraen información acerca de la arquitectura y diseño del sistema o sub-sistema existente y así poderlo modificar parcial o completamente mediante ingeniería directa (*Forward Engineering*). Dicha modificación abarca desde abstracciones de alto nivel y diseños lógicos hasta la implementación física de otro sistema complementario. En esencia, una ingeniería inversa con éxito da lugar a la obtención de una o más especificaciones de diseño y construcción del sistema a tratar. Por esta razón es importante revisar algunos conceptos y metodologías de ingeniería inversa para aplicarlas al problema de adaptación.

2.1 | Ingeniería Inversa

"Reverse engineering in and of it self does not involve changing the subject system. It is a process of examination, not change or replication."
— Chifofsky, *Reverse Engineering and Design Recovery: A Taxonomy* (1990)

El objetivo de la ingeniería es desarrollar la solución a un problema a partir de la nada. En cambio, la ingeniería inversa propone solucionar ese mismo problema, pero a partir de alguna tecnología existente, desentrañando los secretos que existen dentro de un sistema.

La historia de la ingeniería inversa es tan larga como la de la misma ingeniería. Cada vez que una población tenía una tecnología más avanzada, sus rivales buscaban la forma de obtenerla y/o mejorarla. Pero el verdadero surgimiento de ésta se dio con la revolución industrial, lo que provocó el nacimiento y desarrollo de los derechos reservados, las leyes de patentes y demás esquemas de protección a la propiedad intelectual.

En la actualidad existen países en los cuales se prohibe la mayoría de las prácticas de la ingeniería inversa, ya que se considera que ésta atenta contra la propiedad intelectual por su naturaleza. Sin embargo, hay países, como Japón, donde es una práctica recurrente y que ayudó a crear toda una industria de la tecnología llevando al país a ser parte de la lista de principales potencias económicas a nivel mundial.

Cabe mencionar que este trabajo no busca reproducir o copiar ningún sistema ni dañar la propiedad intelectual de las empresas de las marcas de los productos a utilizar. Sólo se remite estrictamente al beneficio académico.

2.1.1 | Terminología

Con el objetivo de indagar en el tema de la metodología que se utilizó en esta tesis para el proceso, a continuación se muestra un breve glosario de términos, ya que es común encontrar en los medios el uso indiscriminado de los conceptos ingeniería inversa y re-ingeniería por igual a pesar de tener significados distintos.

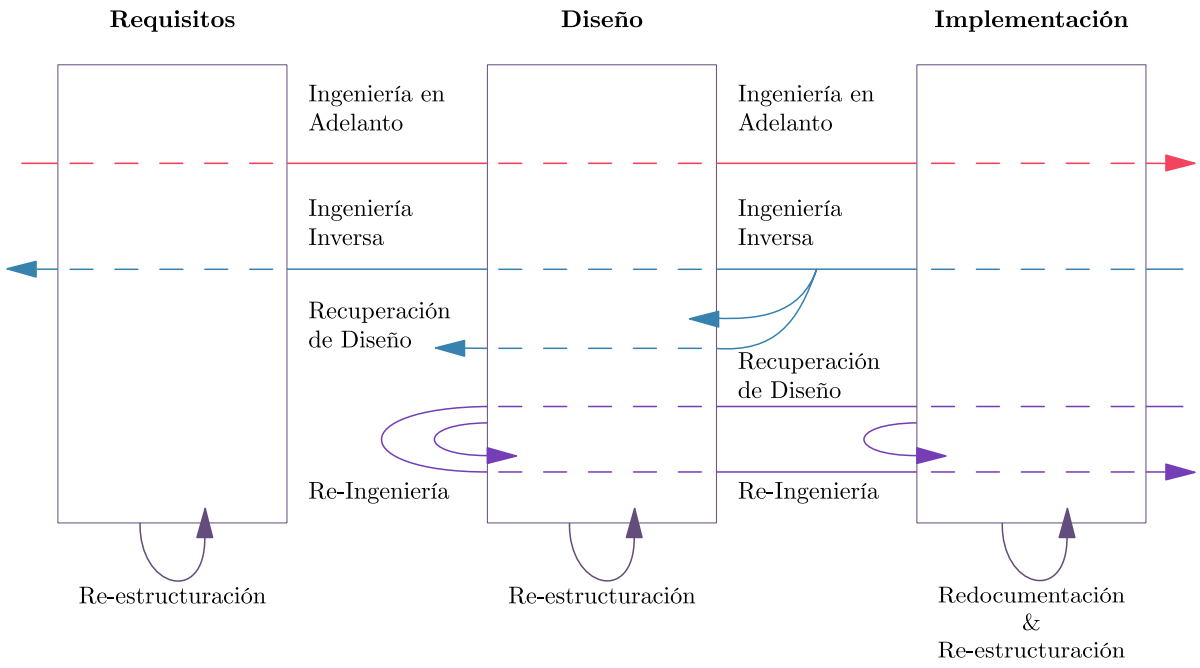


Figura 2.1: Relación entre términos [10].

- **Requisitos (*Requirements*):** Especificación del problema a resolver, incluyendo objetivos, restricciones y reglas.
- **Diseño (*Design*):** Especificaciones de la solución.
- **Implementación (*Implementation*):** Codificación, prueba y entrega del proyecto final.
- **Ingeniería hacia adelante (*Forward engineering*):** O también traducida como *Ingeniería en Adelanto* es el proceso tradicional que lleva desde abstracciones de alto nivel y diseños lógicos a la implementación física de un sistema.
- **Ingeniería inversa (*Reverse engineering*):** Proceso de analizar un sistema con dos metas en mente: analizar los componentes del sistema y sus interrelaciones; y crear representaciones del sistema en otra forma o en un nivel más alto de abstracción. La implementación física del sistema es el punto de partida.
- **Re-ingeniería (*Reengineering*):** Examinar un sistema para reconstruirlo en una nueva forma con su subsecuente puesta en práctica.

- **Re-estructuración (*Restructuring*):** Transformación de una forma de representación a otra en el mismo nivel de abstracción. La nueva representación está destinada a preservar el comportamiento externo de la original.
- **Recuperación del diseño (*Design recovery*):** Subconjunto de la ingeniería inversa en el cual el dominio del conocimiento, la información externa y la deducción se agregan a la observación del sistema a analizar. El objetivo de la recuperación del diseño es identificar abstracciones significativas de alto nivel mas allá de las obtenidas directamente examinando el sistema.

Con base en estos conceptos, la ingeniería inversa y los procesos relacionados son transformaciones entre o dentro de los niveles de abstracción representados en la Figura 2.1 como tres faces de ciclos de vida [10].

2.1.2 | Metodologías

El concepto metodología hace referencia al conjunto de pautas y acciones orientadas a describir un problema. Por lo general, la metodología es un apartado de la investigación científica y la ingeniería, en donde se parte de una hipótesis como posible explicación de un problema e intenta hallar una ley que lo explique o una solución a dicho problema.

En la literatura existen pocos antecedentes para el procedimiento correcto de la implementación de la ingeniería inversa y la mayoría se enfoca al software o a distintos procesos en diversas disciplinas. Sin embargo, se puede llegar a un método general con el análisis de las referencias encontradas mediante su comparación y extrayendo elementos aplicables en el campo de la ingeniería de hardware.

A continuación se muestran tres de las referencias más significativas para el método de ingeniería inversa utilizado en esta tesis como resultado de su análisis.

Método de BOS/X

Un ejemplo de la ingeniería inversa se encuentra en un proceso financiero "Experiences Reverse Engineering Manually" por *D. Swafford, D. Elman, P. Aiken y J. Merhout (2000)* [11], donde se propone y analiza la reestructuración del sistema operacional de negocios. En el artículo se encuentra una lista de acciones generalizadas para llevar acabo la ingeniería inversa la cual se realizará una y otra vez hasta haber finalizado y haber llegado al objetivo (Figura 2.2).

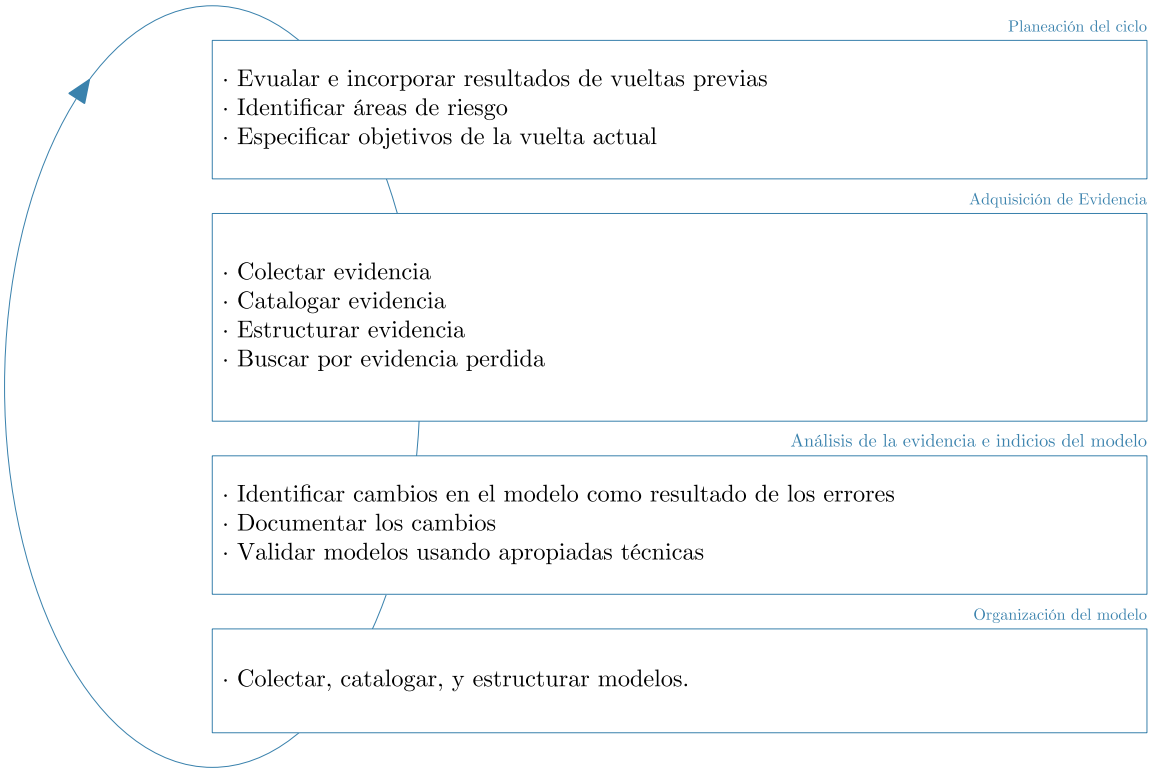


Figura 2.2: Proceso de ingeniería inversa usado en BOS/X [11].

Método de Rekoff

En el ámbito del hardware para M. G. Rekoff (1984) [12], ingeniería inversa es el acto de crear un conjunto de especificaciones para sistemas hardware por medio del análisis de un espécimen. En esta definición, un sistema de hardware puede ser un sistema mecánico, eléctrico o electrónico. Su método mostrado en la Figura 2.3, se basa en la premisa en que el hardware se puede caracterizar como una estructura jerárquica, partiendo de eso se pueden efectuar 5 pasos de forma cíclica para poder definir y entender al sistema.

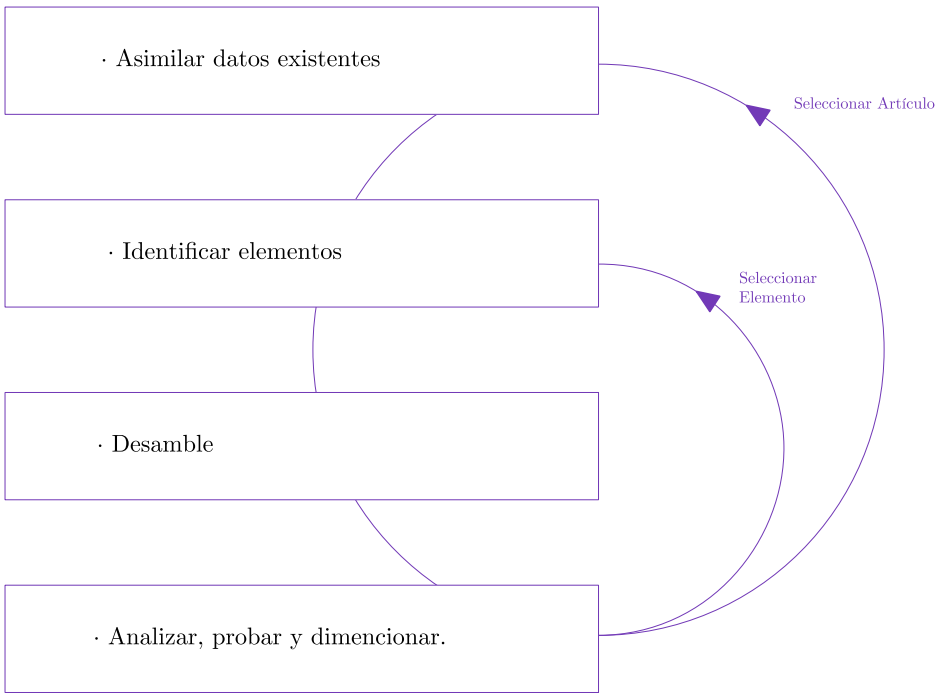


Figura 2.3: Proceso de Redoff para la ingeniería inversa [12].

Método de Kim y Bekey

Por otra parte para *Gerard Jounghyun Kim y George A. Bekey* [13] la ingeniería inversa se refiere a una actividad de inferir el proceso y recolectar información utilizada en la creación de un diseño dado y reutilizar el conocimiento inferido para su rediseño. Ellos proponen una arquitectura nombrada REV-ENGE (Figura 2.4) la cual opera bajo el supuesto de un modelo de diseño de tres etapas: análisis, alteración y evaluación.

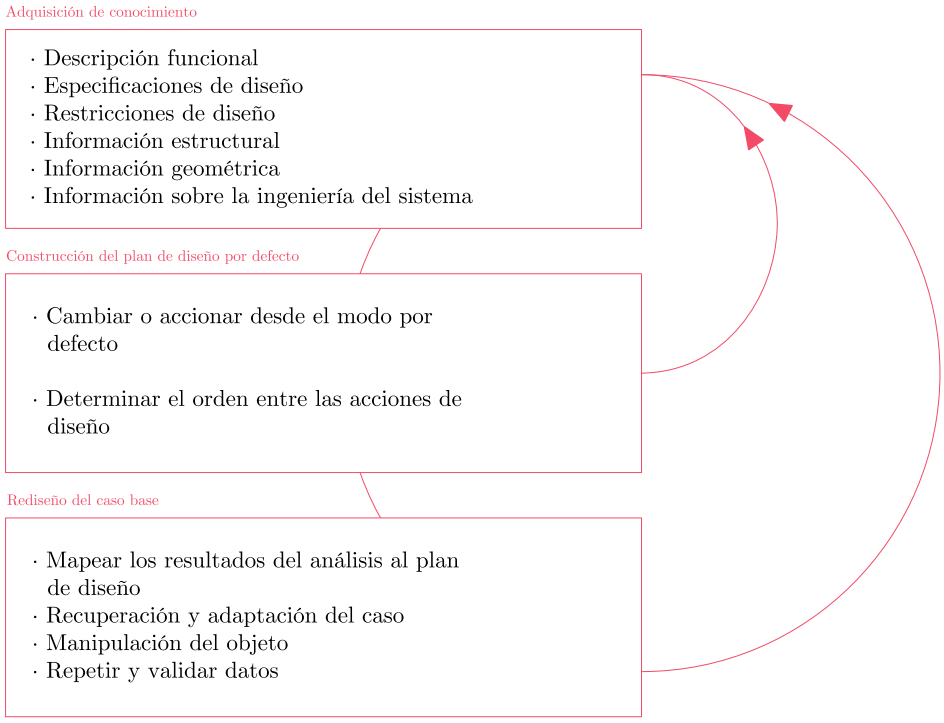


Figura 2.4: Proceso REV-ENGE para la ingeniería inversa [13].

Al realizar el análisis entre los métodos anteriores, se puede concluir para el beneficio de esta tesis que la correcta metodología para el proceso de la ingeniería inversa debe comenzar por el entendimiento del sistema, identificar los elementos y crear sub-elementos de forma ordenada, documentar y volver a revisar para estar completamente seguros del funcionamiento de éste.

2.2 | Expectativas de la tarjeta

Para explicar la problemática de diseñar una tarjeta para la adaptación de un robot industrial a un robot experimental es necesario analizar primero los objetivos y metas.

El sistema debe ser de fácil operación y comprensión para el usuario, técnicos y especialistas en control y robótica, por ello el sistema debe de mantener el estándar industrial, con la capacidad y las características de un robot industrial. Debe permitir programar de manera sencilla algoritmos de control desde una PC utilizando programación estructurada basada en un lenguaje abierto y popular (ej. *C++*). Debe ser un sistema versátil que permita realizar experimentos con varias configuraciones del robot haciendo uso de sensores y herramientas. Adicionalmente, la tarjeta debe de brindar seguridad y permitir la fácil comprensión de éste para detectar alguna falla. Debe ser un sistema económico. Esto se puede lograr aprovechando la mayoría del hardware original; haciendo cambios mínimos pero significativos en el robot y su controlador para evitar el diseño o rediseño de los sub-sistemas involucrados.

2.3 | Problemáticas del diseño de la tarjeta

Teniendo claras las expectativas de la tarjeta, se procede por identificar las fallas en el sistema original o los obstáculos que impiden llegar a las metas establecidas.

- **Arquitectura Cerrada**: debido a que la fábrica no puede revelar los "secretos" de elaboración provoca que gran parte de los circuitos que se encuentran dentro del equipo C500 de CRS sean difíciles descifrar, lo que impide conocer su función parcial o completamente.

- **Información Desactualizada**: Durante la estancia de los robot A255 y A465 en el laboratorio, el equipo que conforma dichos sistemas, ha sufrido cambios de los cuales algunos no han sido documentados, lo que puede provocar confusión al momento de detectar fallas llevando al usuario a la incapacidad de resolver el problema.

- **Interfaz de Usuario Limitada**: La interfaz que proporciona el fabricante es limitada al hardware incluido, ya que no puede modificarse ni ampliar sus posibilidades para algorit-

mos de control más sofisticados.

- **Características Físicas y Dinámicas no establecidas**: El fabricante no aporta un modelo dinámico de los robots lo cual genera conflictos a la hora de diseñar el control.

2.4 | Metodologías

Para poder cubrir las expectativas del proyecto y solucionar las complicaciones que esto conlleva, se propone seguir la siguiente metodología considerando la mayoría de los inconvenientes que se puedan presentar.

- **Ingeniería Inversa**: Se aplicará ingeniería inversa mediante la separación del sistema por partes, determinando la función de cada elemento, se crearán sub conjuntos de forma ordenada y las conclusiones sobre la revisión serán documentadas. Estos pasos se realizarán repetidamente hasta tener un conocimiento general del sistema.
- **Documentación de Información**: Se documentará la información recabada por la ingeniería inversa, así como los procesos que se realicen durante el desarrollo del proyecto. Esto con el objetivo de brindar un mejor entendimiento del sistema para poder detectar problemas o de modificar al sistema. Se generarán archivos tipo plantillas para facilitar la investigación y el desarrollo de nuevos algoritmos de control.
- **Protección en Hardware**: Se detectará fallos sobre el hardware existente y se propondrá soluciones así como las medidas de seguridad para preservar la estabilidad del sistema.
- **Interfaz de Usuario Gráfica**: Se generará una interfaz gráfica programada con código abierto capaz de ampliar sus capacidades y ésta deberá poder establecer comunicación con los robots mediante la tarjeta.
- **Comprobación de la Tarjeta**: Se realizará un experimento para comprobar la eficacia del trabajo realizado. El control se debe aplicar de forma sencilla y entendible.

Con los pasos descritos anteriormente debe ser posible el desarrollo de una tarjeta para la adquisición de datos y el control de los robots A465 y A255 de manera cooperativa.

Capítulo 3

Especificaciones del Sistema

En este capítulo se describen los elementos que integran al sistema así como los elementos que forman parte de la adquisición de datos y el control de los dos robots industriales de la marca *CRS Robotics* de los que dispone el Laboratorio de Robótica del Departamento de Control y Robótica de la División de Ingeniería Eléctrica de la Universidad Nacional Autónoma de México. Esta descripción es resultado de la ingeniería inversa aplicada al sistema y fue fundamental para la planificación de la tarjeta para la adquisición de datos y control de los robots.

3.1 | Sistema original

El sistema utilizado generalmente para comunicar un robot industrial y el controlador se encuentra mostrado en la Figura 3.1, donde el robot industrial es accionado por una etapa de potencia con protecciones electro-mecánicas, adicionalmente como se vio en la Sección 1.1.3, el robot industrial posee sensores cuya información es adaptada y procesada por la unidad central o CPU para poder implementar un controlador. Es también en la unidad central donde se encuentra la interfaz de usuario con el cual se establece comunicación con el robot.

En esta tesis se propone llegar a un sistema conformado por dos robots industriales en el cual compartan un solo módulo de control que se comunique con la interfaz de usuario y con la etapa de potencia que cada subsistema posee para cada robot.

3.1.1 | Robots Industriales A255 y A465

Los sistemas de los robots industriales A255 y A465 proporcionados de fábrica se constituyen por *teach pendant* (herramienta utilizada para programar al robot paso por paso [14]), controlador C500 (unidad electrónica computarizada con memoria que proporciona las señales de control necesarias para la operación del brazo robot) y el respectivo robot industrial de cada uno de los sistemas (Figura 3.2a,3.2b).

Uno de los principales experimentos a realizar con los robots de forma cooperativa es la manipulación de objetos los cuales deben pesar de máximo $2[kg]$ para evitar un daño en los actuadores como se analiza en la Tabla 3.1. También se listan algunas de las propiedades y especificaciones que poseen los robots A465 y A255 [15][16]. Otras especificaciones como repetitividad, torque o distancias de alcance de los robots pueden verse en el Apéndice A al igual que las especificaciones del controlador C500 (Apéndice B).

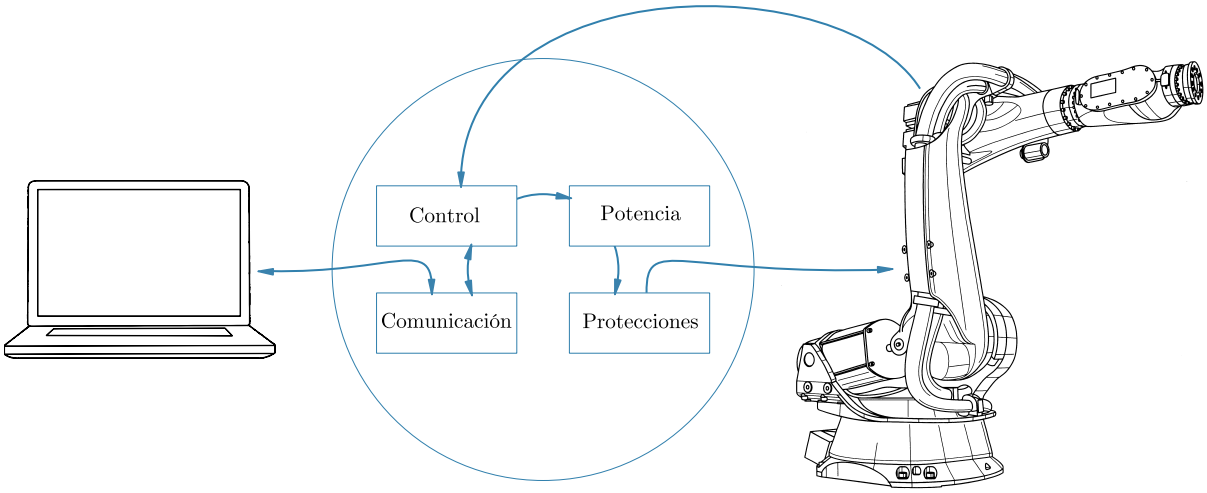
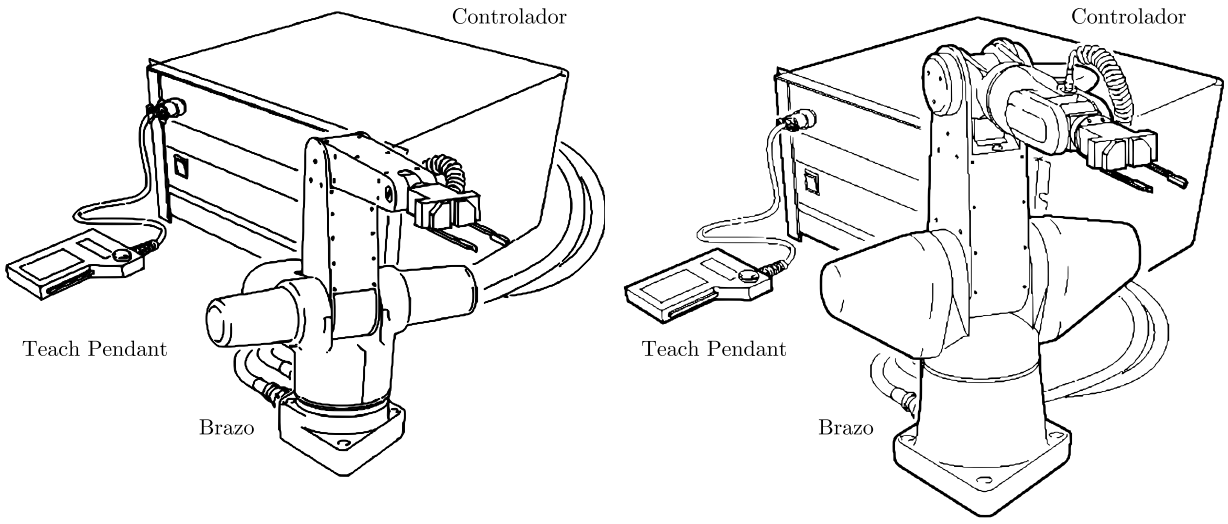


Figura 3.1: Disposición general de un controlador industrial.



(a) A255 (b) A465

Figura 3.2: Componentes de fábrica de los robot A255 y A465 [15][16].

Especificación	Robot A465	Robot A255
Peso	32 Kg	17 Kg
Carga (Máxima)	3 Kg	2 Kg
Carga (Nominal)	2 Kg	1 Kg
Alcance (X-Y)	711.2 mm	560 mm
Impulsión	Servo motores de DC	Servo motores de DC
Transmisión	Impulsores Armónicos	Impulsores Armónicos y engranes cónicos/rectos
Rango de acción de las Articulaciones		
Cintura	+175° to -175°	+175° to -175°
Hombro	+ 90° to -90°	+110° to 0°
Codo	+110° to -110°	0° to -125°
Muñeca	+180° to -180°	-
Muñeca (Pitch)	+105° to -105°	+110° to -110°
Herramienta	+180° to -180°	+180° to -180°

Tabla 3.1: Especificaciones de los robots A465 y A255.

Actuadores

Los actuadores utilizados en los robots (motores de corriente directa) reaccionan ante el voltaje y la corriente de alimentación, con un torque proporcional a la corriente y a su vez, la velocidad proporcional al voltaje. El suministro de alimentación de los actuadores se encuentra dentro del controlador C500 mediante amplificadores, donde el rango de voltaje de entrada para las articulaciones de ambos robots es de $\pm 10[V]$. En la Tabla 3.2 se muestran los valores de voltaje de entrada y salida de dichos amplificadores así como la relación entre voltajes y la corriente de salida.

El sistema proporcionado de fábrica de ambos robots se caracteriza por el esquema de la Figura 3.3 en donde los únicos elementos que son externos al controlador C500 son los motores y codificadores que forman parte de la estructura del robot.

	Robot A465				Robot A255			
Eje	$V_e[V]$	$Vs[V]$	$I_s[A]$	V_s/V_e	$V_e[V]$	$Vs[V]$	$I_s[A]$	V_s/V_e
Cintura	± 10	± 70	12	7	± 10	± 25	2	2.5
Hombro	± 10	± 70	12	7	± 10	± 25	2	2.5
Codo	± 10	± 70	12	7	± 10	± 25	2	2.5
Muñeca	± 10	± 30	3	3	-	-	-	-
Pitch	± 10	± 30	3	3	± 10	± 25	2	2.5
Herramienta	± 10	± 30	3	3	± 10	± 25	2	2.5

Tabla 3.2: Características de Corriente y Voltaje de la etapa de potencia de los robots

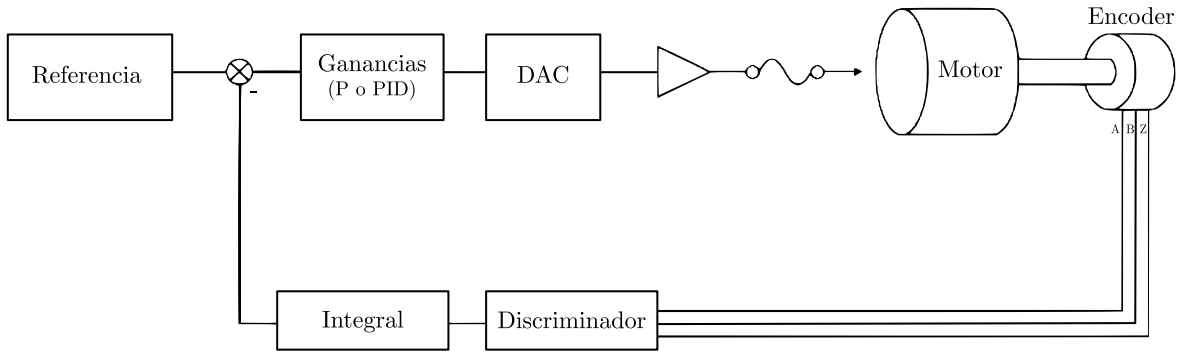


Figura 3.3: Diagrama de bloques del sistema de los actuadores.

Codificadores

Los actuadores son responsables de percibir la dinámica del proceso y para tener control sobre éste es necesario contar con sensores que le permitan al robot conocer el estado en el que se encuentra. En el caso de los robots A465 y A255 se cuanta con codificadores que son sensores de posición angular que suministran (en un código binario) la posición angular del eje sobre el que están montados. Los codificadores de ambos robots son del tipo incremental, los cuales constan de un fototransistor que detecta las variaciones de iluminación debido a la obstrucción de luz que generan trazos negros en discos dentro del sensor. Este tipo de codificadores otorgan una específica cantidad de pulsos por cada rotación. La salida puede ser una sola vía de pulsos o dos (Canal "A" o canal "B") en los que se puede detectar un offset entre las señales y así se puede determinar el sentido en el que se está rotando, a este desfasamiento se le llama cuadratura. Una tercera señal "Z" (O también nombrada índice) se utiliza para verificar las señales "A" y "B" con un pulso por revolución.

Los codificadores que se encuentran actualmente en los robots son de la marca japonesa SUMTAK modelo LDA-051-1000 [17]. Se alimentan con un voltaje de 5[v] y consumen 140m[A], las señales que entregan son "A", "B", "Z", "$\bar{A}$", "$\bar{B}$" y "$\bar{Z}$". El robot A255 se limita únicamente a las señales no negadas debido a que el cable *umbilical* (Cable para la transmisión de datos de los codificadores del robot al controlador) no posee las terminales requeridas para transmitir esta información. Una vez entendida la naturaleza de las señales del lazo de control industrial se prosigue con el desarrollo de la tarjeta.

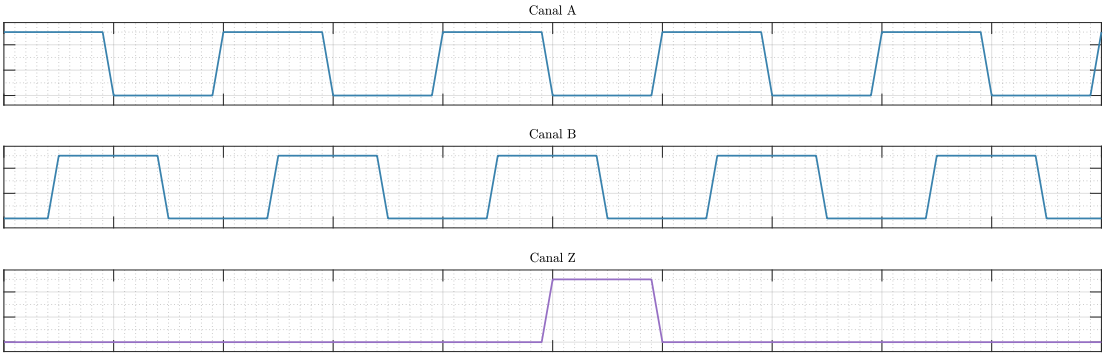


Figura 3.4: Señal de codificador en cuadratura con índice.

3.1.2 | Controlador C500

Las partes más importantes del sistema de los robots A465 y A255 se encuentran dentro del controlador C500 ya que es ahí donde se realizan todos los procesos de acoplamiento y análisis de señales, potencia de los motores, control e interfaz de usuario con el sistema.

El hardware electrónico del controlador C500 se divide en tres compartimentos o secciones principales. El compartimiento superior contiene la electrónica del control del robot. El compartimiento inferior contiene la fuente de poder, el circuito de paro de emergencia y la etapa de acoplamiento de señales de los encoders. En el espacio sobrante, al costado de los compartimentos superior e inferior, se encuentra ubicada la etapa de potencia de los motores [18].

En la Figura 3.5 se encuentra la vista trasera del controlador C500 al momento de ser adquirido por el distribuidor, donde se distinguen las terminales *Robot Power* y *Robot Feedback*. La terminal *Robot Power* en el controlador C500 es la salida de la etapa de potencia de los motores y la terminal *Robot Feedback* es la entrada de señales de los encoders. En el Apéndice B.1.3 se encuentran las especificaciones sobre dichas terminales.

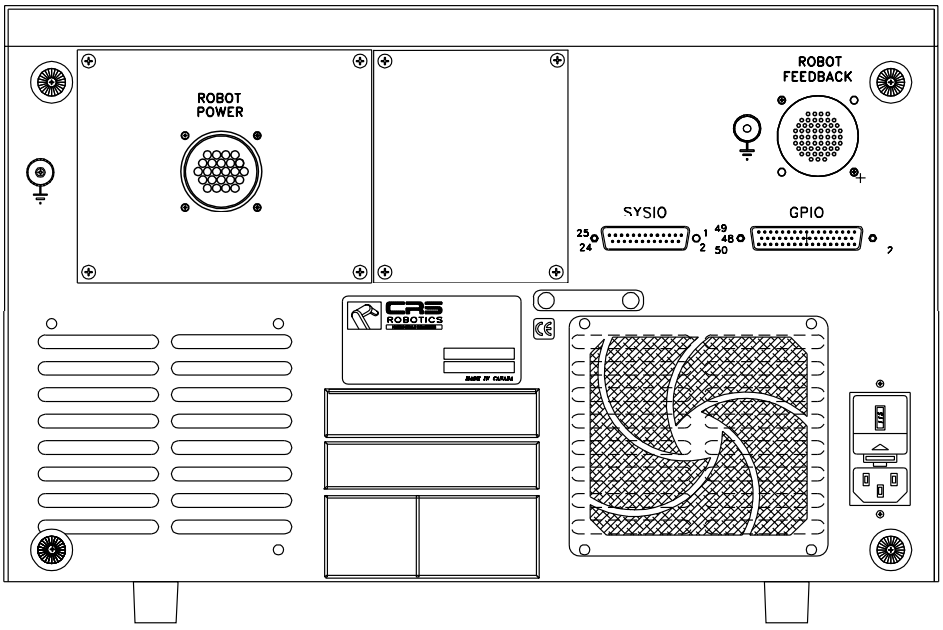


Figura 3.5: Parte trasera del controlador C500 otorgado de fábrica.

3.2 | Modificaciones sobre el sistema original

Durante su estancia en el Laboratorio de Robótica, los sistemas han sufrido modificaciones con el objetivo de acoplarse mejor a los experimentos que se realizan dentro de éste. A continuación se mencionan algunas de estas modificaciones hechas antes de la elaboración de esta tesis.

3.2.1 | Inhibición de la etapa de control

Para poder llegar al objetivo de desarrollar una tarjeta de adquisición de datos para adaptar los robots industriales a robots experimentales, es necesaria la inhibición de la etapa de control proporcionado desde la fábrica del controlador C500 para poder implementar un controlador propio. Esto se logró mediante la intervención del cableado del controlador C500 que fue documentada en *"Análisis del controlador C500 e inhabilitación de la etapa de control" (2002)* [19]. En ese trabajo se ubicaron e identificaron las señales de interés (señales generadas por codificadores y salidas de la etapa de potencia de los actuadores) para su redistribución e interrupción

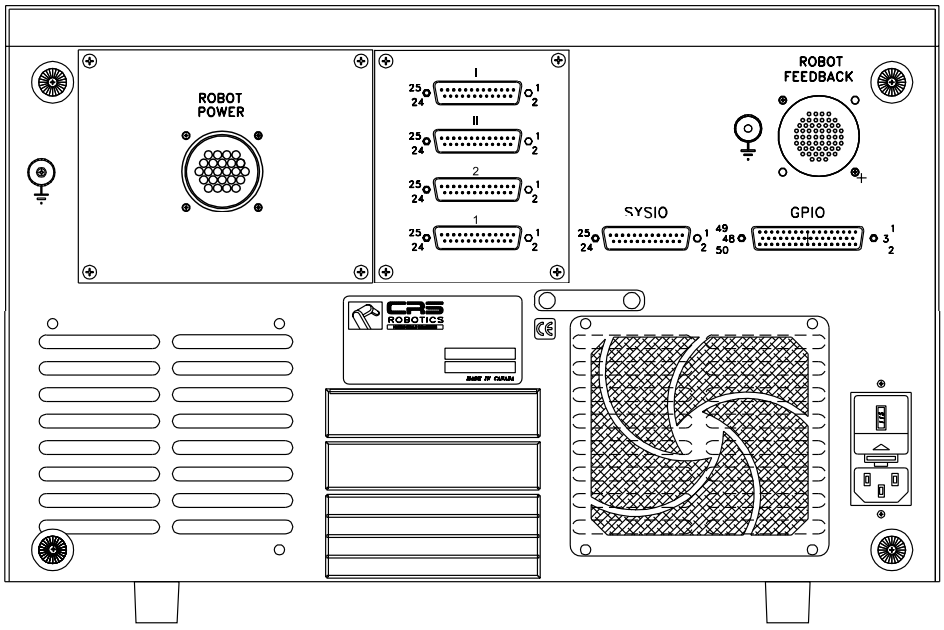


Figura 3.6: Parte trasera del controlador C500 con conectores DB-25 [I], [II], [1] y [2].

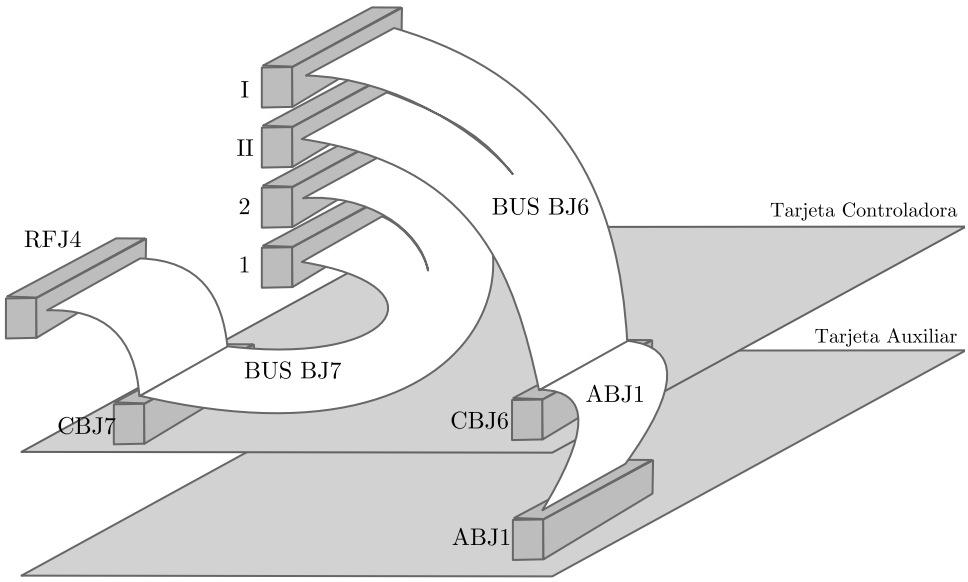
con la etapa nativa del controlador C500, el resultado interno de esta intervención se muestra en la Figura 3.7 y 4.2c. En el Apéndice B se encuentran la asignaciones de pines para los conectores I, II, 1, 2, de la Figura 3.7, en las Tablas 2.1, 2.2, 2.3 y 2.4 respectivamente.

3.2.2 | Botonera E-Stop Externa

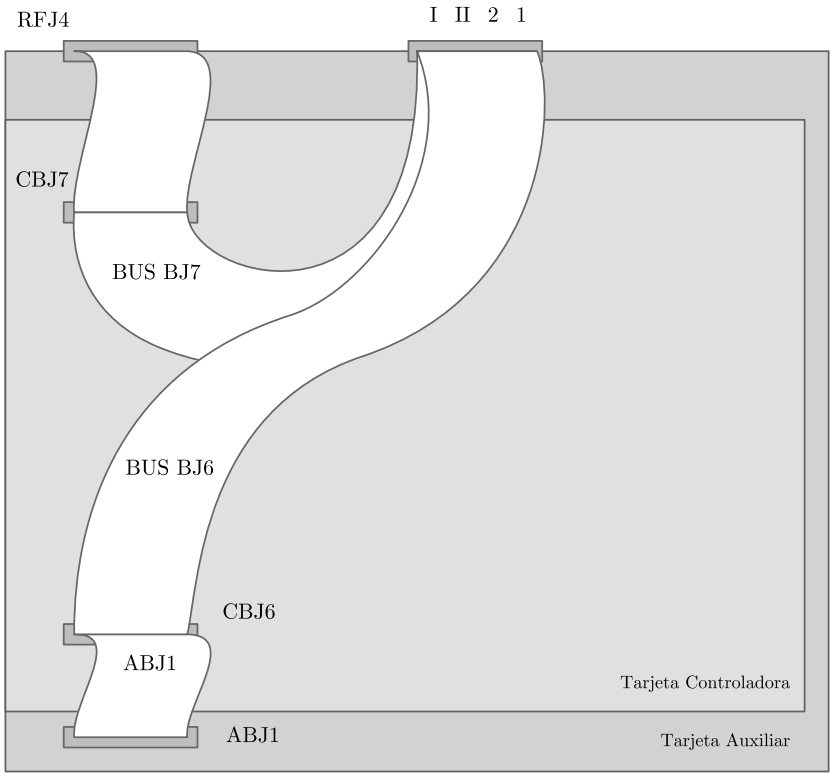
El botón E-Stop es un botón de emergencia de color rojo en un recuadro amarillo ubicado en la zona frontal del controlador C500 en la parte inferior derecha. El botón es parte de un circuito en serie dentro del controlador C500 (Figura 3.8). Al ser presionado el botón E-stop abre el circuito impidiendo la alimentación a los servo-motores de los robots A255 y A465. Éstos están equipados con servos de agarre. Al quitar la alimentación estos quedarán fuertemente inmovilizados y para volver a accionarse es necesario liberar el botón de su posición activa y presionar el botón que active al robot. Es importante mencionar que el programa seguirá corriendo a pesar de que los robots hayan sido parados.

Otro botón de emergencia con la misma función se encuentra ubicado en el panel de trabajo del *teach pendant* y al ser accionado, además de parar al robot, activa un indicador sonoro. En la Figura 3.8 se representan encerrados con lineas punto-guión elementos externos al controlador, por lo que se sobreentiende que al no ser conectado el teach pendant los robots permanecerán sin movimiento, en caso de no contarlo se sugiere conectar un puente en forma de rosca en el conector del mismo llamado *Over-Ride Plug*.

Con el objetivo de poder realizar paros de emergencia mediante un sólo botón para ambos robots, se utilizaron los puertos 9, 10, 22 y 23 del conector SYSIO para tener acceso a los circuitos de paro y encendido, y se colocó un conector para una botonera externa en la parte trasera del controlador C500 de cada robot. Más información sobre el conector SYSIO y demás partes del controlador C500 está disponible en el Apéndice B.



(a) Vista isométrico del interior



(b) Vista superior del interior

Figura 3.7: Disposición interna del controlador C500 con inhibición del controlador [19].

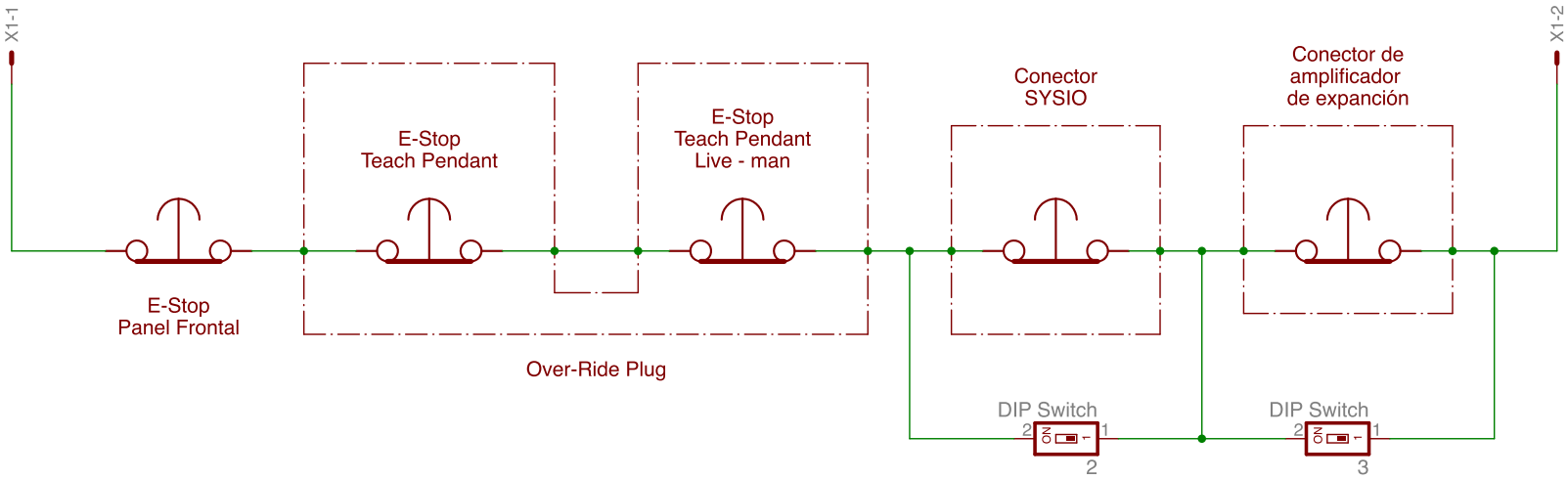


Figura 3.8: Circuito de paro de emergencia del controlador C500.

3.3 | Elementos a interconectar por la tarjeta

3.3.1 | Sistema embebido CompactRIO

Una vez deshabilitada la etapa de control original es necesario implementar un sistema el cual remplace la interpretación de datos y además realice el control para que funcionen los robots. En el laboratorio se cuenta con unidades CompactRIO de la marca National Instruments NI-cRIO 9014 por lo que se uso en este proyecto para su aprovechamiento.

Como se vio en la Sección 1.1.3 el hardware y software empleados para la adquisición de datos es muy diversa y su selección depende del sistema en el que se incluirá. El sistema embebido CompactRIO de National Instruments resulta una buena opción por su FPGA incluido. La comunicación Ethernet RJ-45, la seguridad que ofrece, ya que cumple con diversos estándares manejados en la industria y principalmente por su disposición en el laboratorio.

El número máximo de ranuras posibles a utilizar es de ocho donde se pueden ocupar distintos módulos o cartucho de hardware compatible con el sistema CompactRIO.

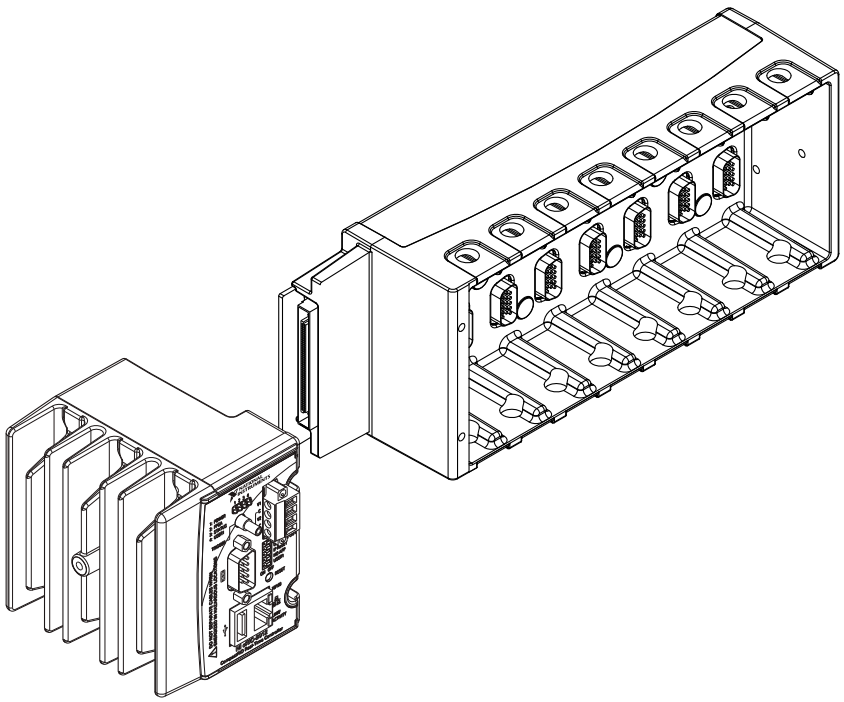
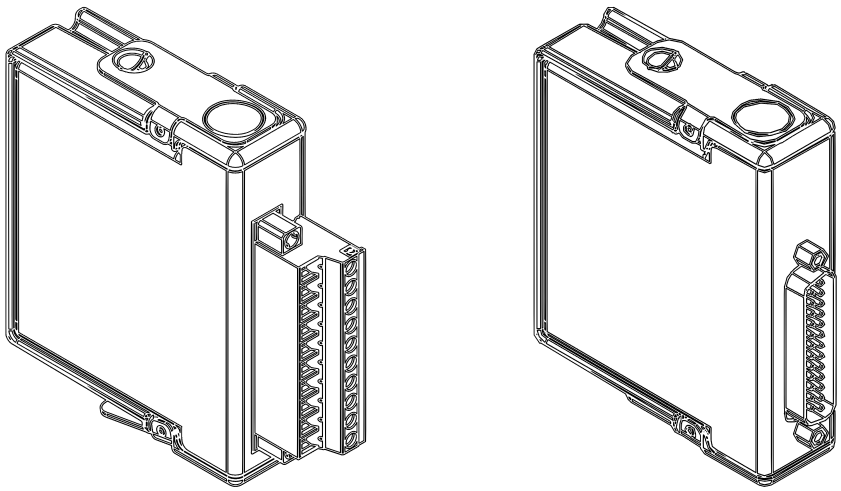


Figura 3.9: CompactRIO National Instruments NI-cRIO 9014 [20].

3.3.2 | Módulos NI 9263 y NI 9401

El módulo NI 9263 hace posible la generación de hasta cuatro señales analógicas mediante tres etapas que se componen por un convertidor digital analógico aislado de 16 bits, un amplificador y una etapa de protección contra sobre voltaje o corto circuito. Los cuatro canales pueden generar señales de $\pm 10[V]$ nominales y pueden entregar máximo $\pm 1[mA]$ a la carga, el slew rate es de $4[V/\mu s]$ que expresa el tiempo que tarda el módulo en generar un aumento a la salida de $4[V]$. Con el objetivo de generar once señales analógicas requeridas por las once articulaciones (cinco articulaciones del robot A255 y seis del robot A465) se escogieron tres módulos, teniendo así una señal sobrante.

El módulo NI 9401 puede ser utilizado para generar o leer señales TTL y puede ser configurado para que sus puertos sean entradas o salidas. Utilizando los ocho canales como entradas, el módulo tiene una frecuencia de switching de la señal de entrada máxima de $9M[Hz]$ (lo cual es un valor por debajo de la tendencia actual de las tarjetas de desarrollo y SBCs de $16M[Hz]$ aproximadamente dependiendo de la tarjeta), cada puerto permite voltajes desde $2[V]$ hasta $5.25[V]$. Para cubrir todos los puertos requeridos por el sistema de robots se escogieron cinco módulos NI 9401 con un total de cuarenta canales sobrando un puerto digital.



(a) Módulo NI 9263 (b) Módulo NI 9401

Figura 3.10: Módulos utilizados para la adquisición de los datos [21][22].

3.3.3 | Fuente de alimentación Quint Power

La potencia máxima requerida por la compactRIO con los ocho módulos es de 20[W] y puede ser alimentado con un voltaje de 19[V] o hasta 30[V] variando así la corriente desde 1.05 hasta 0.66. El fabricante recomienda una fuente de 24[V] a 2[A]. Se utilizó una fuente Quint Power (Figura 3.11) de la marca Phoenix Contact de 24[V] que otorgan hasta 5[A] para alimentar al sistema CompactRIO; además, se utilizará para la alimentación de la tarjeta de adquisición de datos. La fuente de alimentación Quint Power requiere de un voltaje de alimentación de 85[V] a 265[V] de corriente alterna para un correcto funcionamiento.

3.3.4 | Botonera externa E-Stop

Una parte adicional de la tarjeta de adquisición de datos es el manejo de los puertos de los conectores de paro de emergencia externa para que con un sólo botón puedan ser detenidos ambos robots además de permitir el paro mediante software y mediante sensores de impacto.

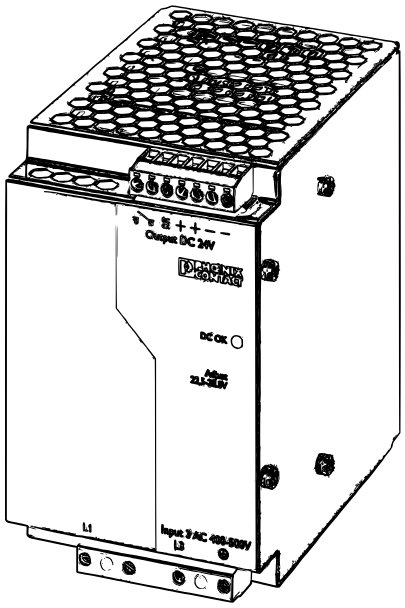


Figura 3.11: Fuente de alimentación Quint Power [23].

3.4 | Galería de Fotos

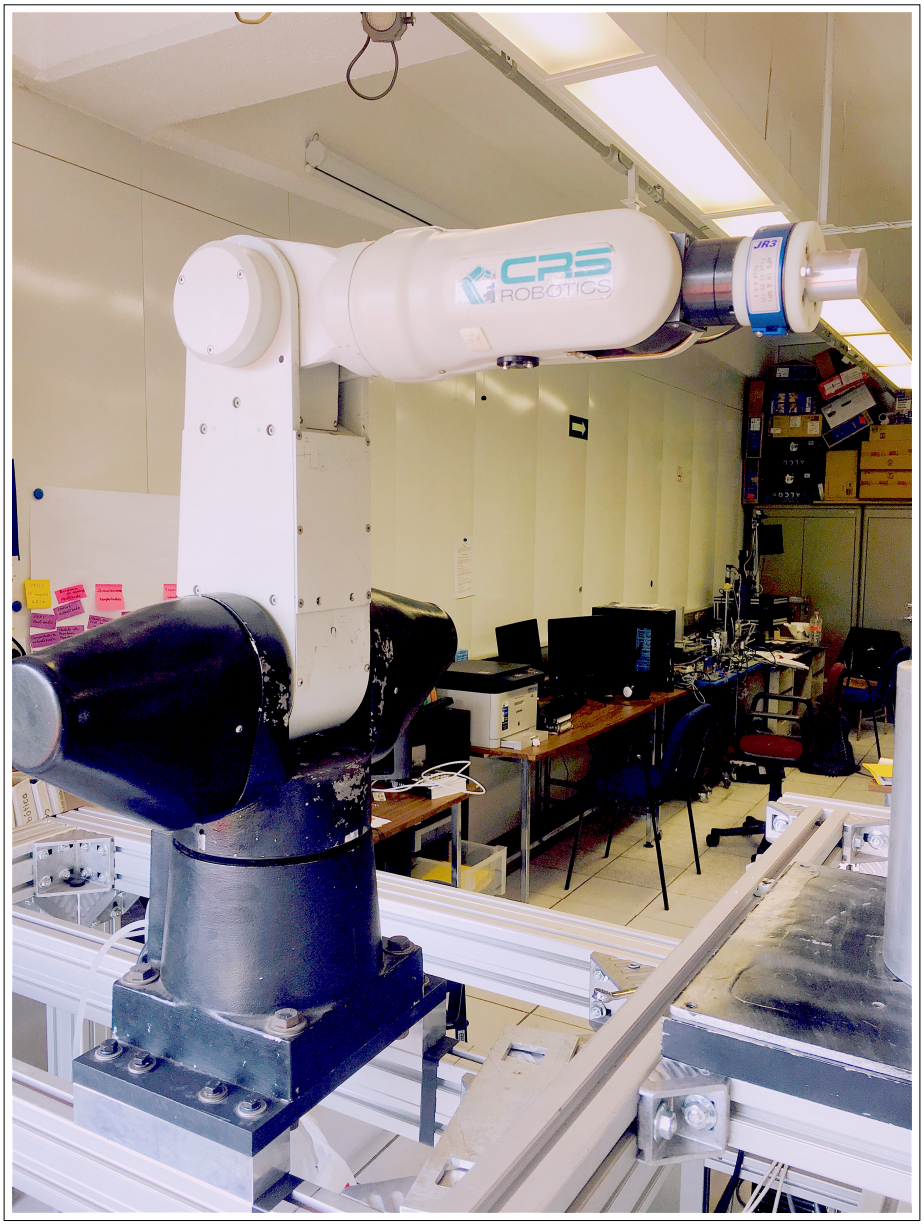


Figura 3.12: Foto del Robot A465.

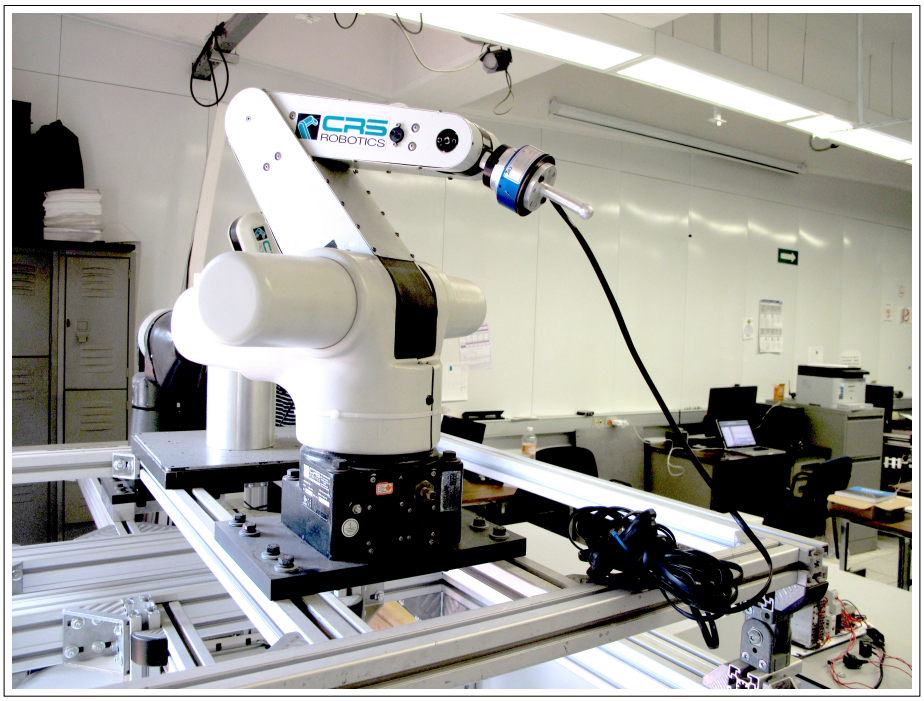


Figura 3.13: Foto del Robot A255.

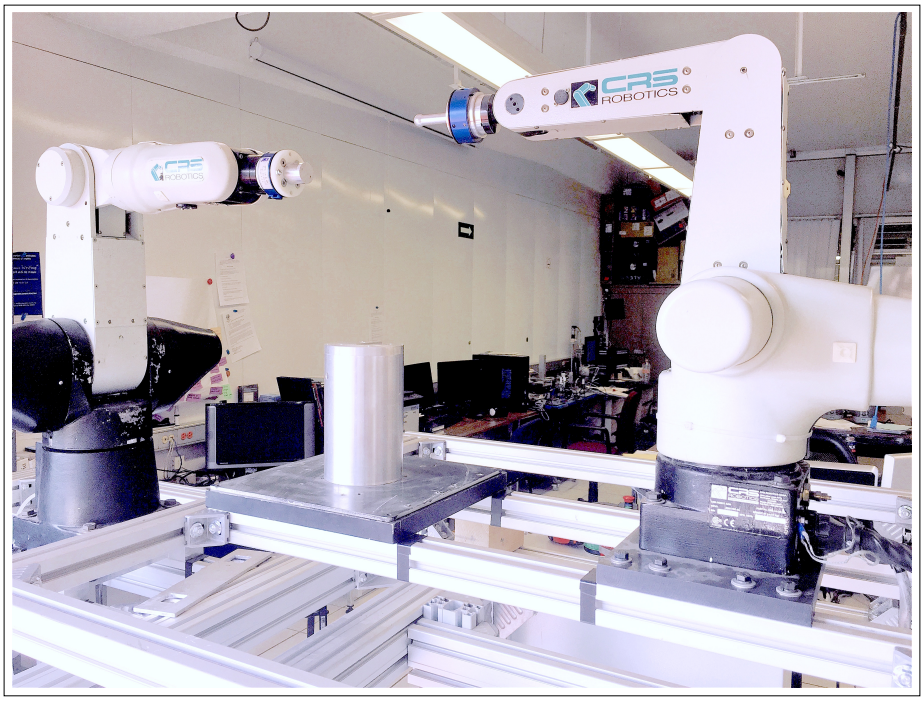


Figura 3.14: Foto de los Robots A255 y A465.

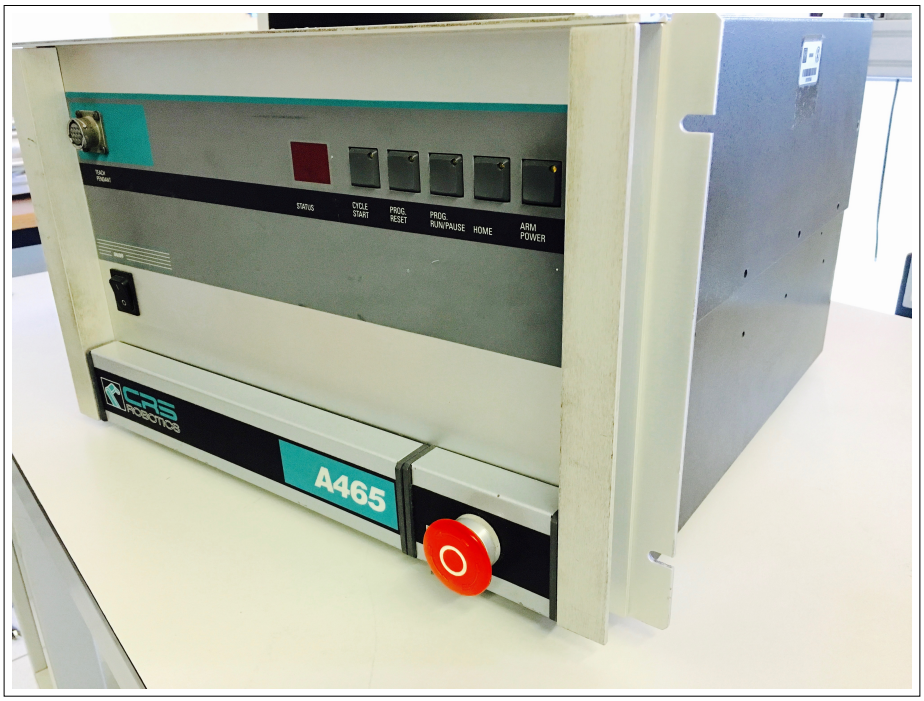


Figura 3.15: Controlador C500 del robot A465.

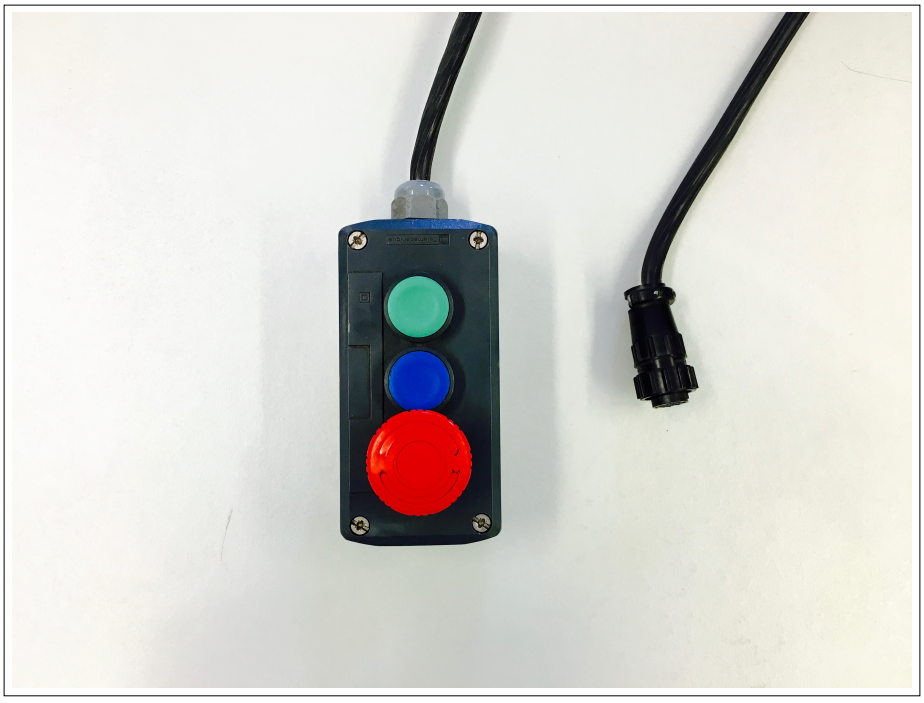


Figura 3.16: Botonera externa para paro de emergencia.

Figura 3.17: Sistema CompactRIO NI-cRIO 9014.

Figura 3.18: Módulos NI-9263.

Figura 3.19: Módulos NI-9401.

Figura 3.20: Unidad Quint Power PS-100.

Capítulo 4

Diseño de Hardware

En el capítulo anterior se habló de los elementos que tienen como objetivo interconectar la tarjeta, que de forma abreviada, debe de recibir información de los codificadores y transmitirla a los módulos NI 9401 así como recibir señales analógicas de los módulos NI 9263 y transmitirla a la etapa de potencia del controlador C500, así como hacer permisible la detención y activación de los robots mediante una botonera, software y por haber sufrido algún impacto.

A continuación se detalla el desarrollo de la construcción de la tarjeta de adquisición de datos y control así como la construcción del gabinete de trabajo.

4.1 | Tarjeta para la re-distribución de canales

Debido a que las señales de interés de los conectores DB25 [I], [II], [1] y [2] (provenientes de la inhibición del control original visto en la Sección 3.2.1) son unicamente las provenientes de los enconders "A", "B", "Z", "$\bar{A}$", "$\bar{B}$" y "$\bar{Z}$", las señales digitales de "Home" y los canales de entrada para comandos de voltaje de las articulaciones. Se elaboró una tarjeta de re-distribución ubicada dentro del controlador C500 para implementar sólo tres cables, en vez de cuatro, distribuyendo así los conectores DB25 en tres: "AB", "Z" y "VCOM".

Dicha tarjeta de re-distribución se propuso para mejorar el orden de las señales y así facilitar el diseño de la tarjeta de adquisición de datos y control, además de permitir el intercambio de información entre ésta y el controlador (Figura 4.1).

En el Apéndice B se encuentra más información referente a los conectores AB, Z y VCOM como las Tablas C.2.5, C.2.6 y C.2.7.

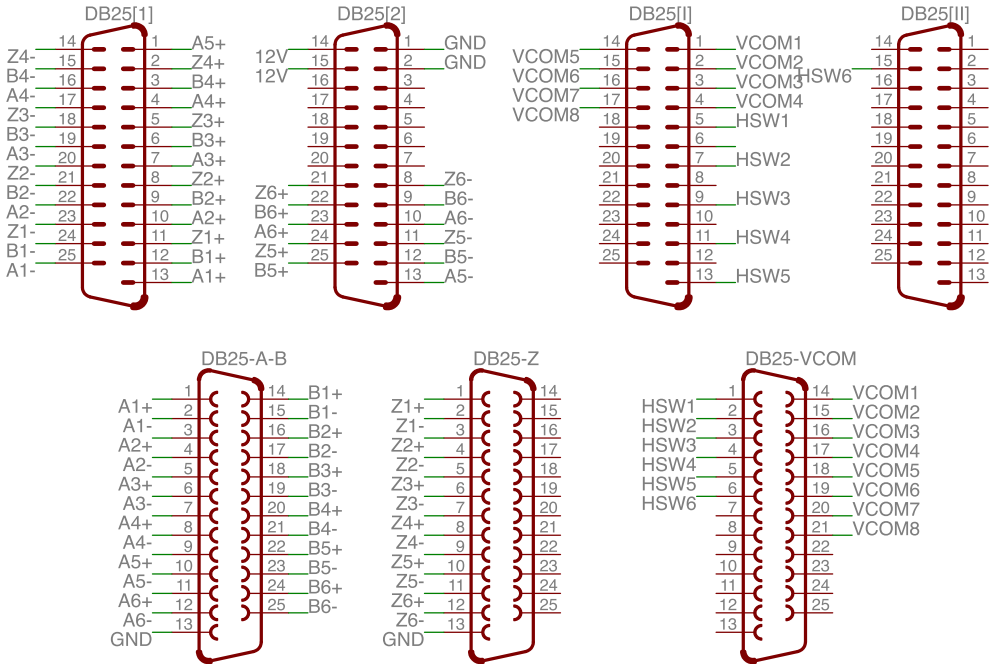


Figura 4.1: Esquema de Conexiones de la tarjeta para la re-distribución de canales del controlador C500

La tarjeta con el circuito impreso resultante es la que se muestra en la Figura 4.2. Para su diseño se recurrió al software *EAGLE PCB Design* versión 7.3.0. La tarjeta es seccionada en dos etapas, la primera consta de los conectores DB25: [I], [II] y [VCOM] cuyas señales son analógicas a diferencia de la etapa dos que se compone por los conectores DB25: [1], [2], [AB] y [Z] que manejan las señales digitales. En la Figura 4.2c además de mostrar éstas secciones, se muestra también el flujo de datos, de esta forma los conectores ubicados en la parte superior de la tarjeta serán los conectores externos del controlador C500. El grosor adecuado de cada una de las pistas fue calculado con el uso de las ecuaciones (4.1) y (4.2)[24].

$$A = \frac{I}{k(T^b)}^{1/c} \tag{4.1}$$

$$W = A/L \tag{4.2}$$

Donde A es el área transversal de la pista, I es la corriente máxima qué atraviesa el área, T es el incremento de temperatura máxima, W es el ancho de la pista y L es la altura [9][24]. Las constantes k, b y c están dadas por la curva de acuerdo con la norma IPC-2221, información que fue facilitada por el fabricante *OSH Park* en su página en internet al igual que la altura [25].

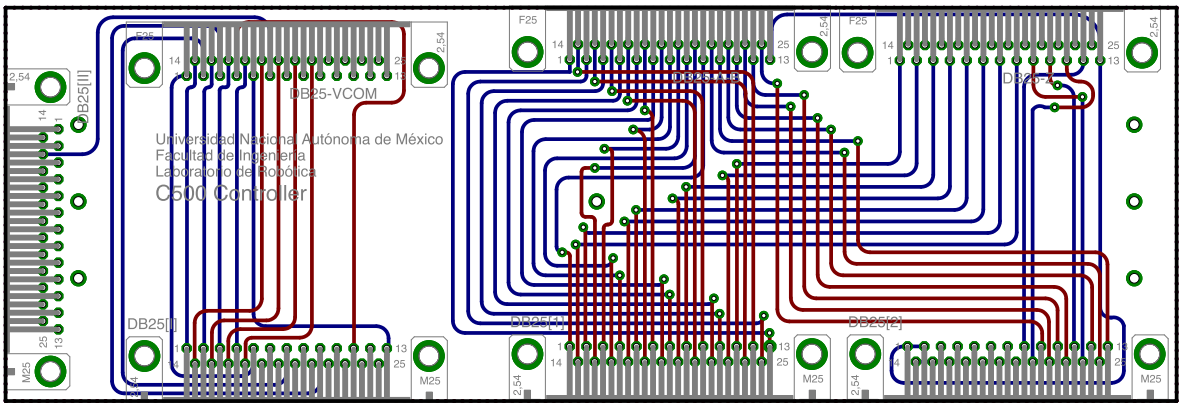
$$\begin{aligned} L = &\quad 1.6m[m] \\ k = &\quad 0.048 \\ b = &\quad 0.44 \\ c = &\quad 0.725 \end{aligned} \tag{4.3}$$

La corriente máxima que entregan los codificadores es de $1[mA]$ al igual que la corriente máxima que entrega el módulo NI 9263, descrito en la Sección 3.3.2. Sustituyendo datos el resultado es:

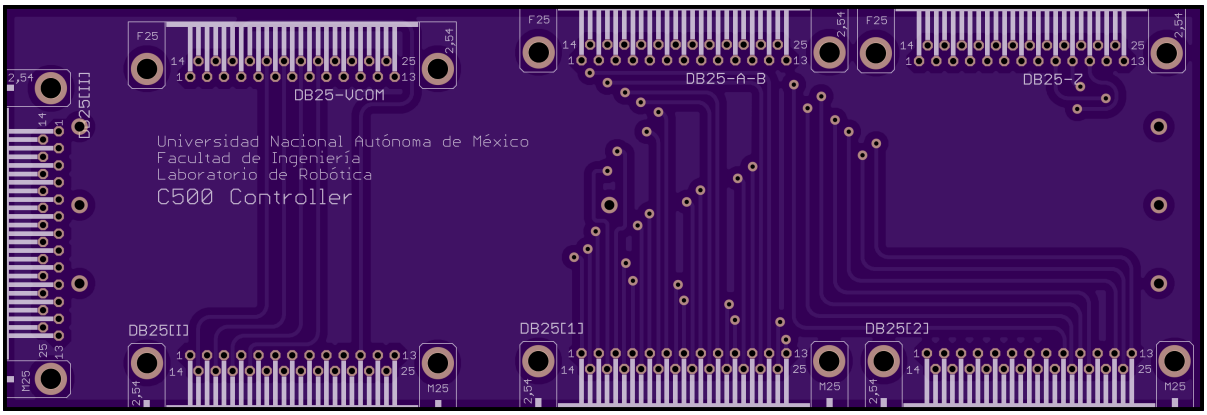
$$\begin{aligned} A = &\quad \frac{1m[A]}{(0.048)(10[^{\circ}C]^{0.44})}^{1/0.725} \\ = &\quad 0.0012[mil^2] \\ W = &\quad \frac{0.0012[mil^2]}{1.6m[m]} = \quad \frac{0.0012[mil^2]}{63[mil]} \\ = &\quad 1882_{[10^{-5}]}[mil] \end{aligned} \tag{4.4}$$

El grosor de pista resultante es sumamente pequeño debido a la corriente que se transmitirá. Se propuso un ancho de pista de $24[mil]$ para una corriente máxima de $460[mA]$ y un incremento máximo de temperatura de una milésima parte de grado centígrado, cabe mencionar que la resistencia promedio generada por las pistas es del orden de los $\mu[\Omega]$ por lo que se despreció su efecto en la caída de voltage de las señales.

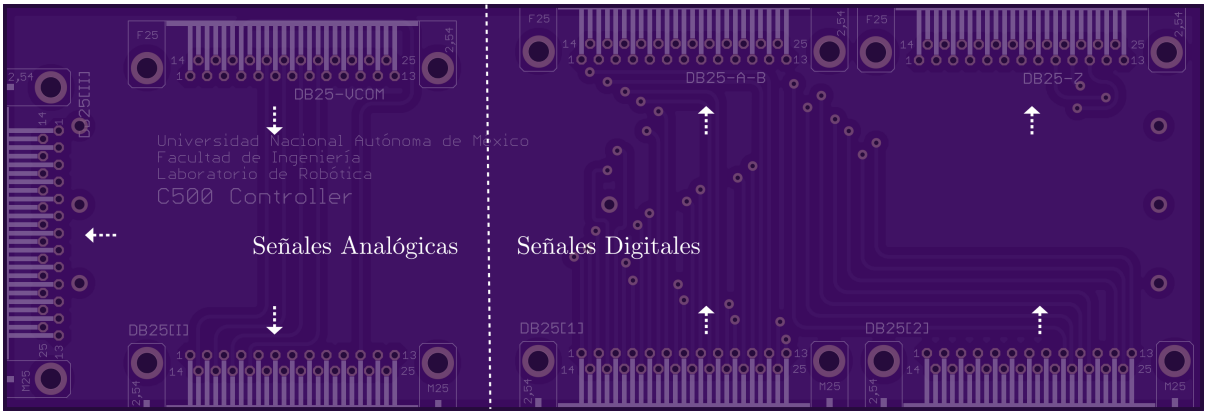
Se colocó un plano de tierra con un espesor de $35[mm]$ que va conectado al chasis del controlador C500 para evitar interferencia entre la señal de control de los servomotores y la lectura de los encoders. El uso del plano de tierra se recomienda en sistemas en donde las señales analógicas y digitales coexisten. En la Figura 4.3a se ilustra la tarjeta de re-distribución instalada al interior del controlador C500.



(a) Diseño en EAGLE

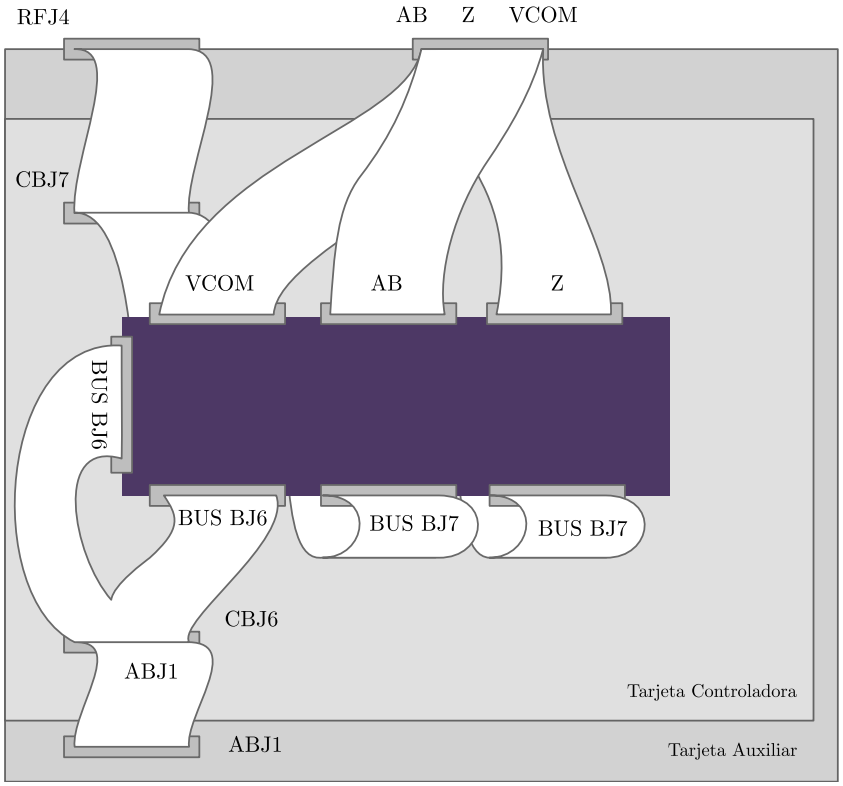


(b) Vista frontal

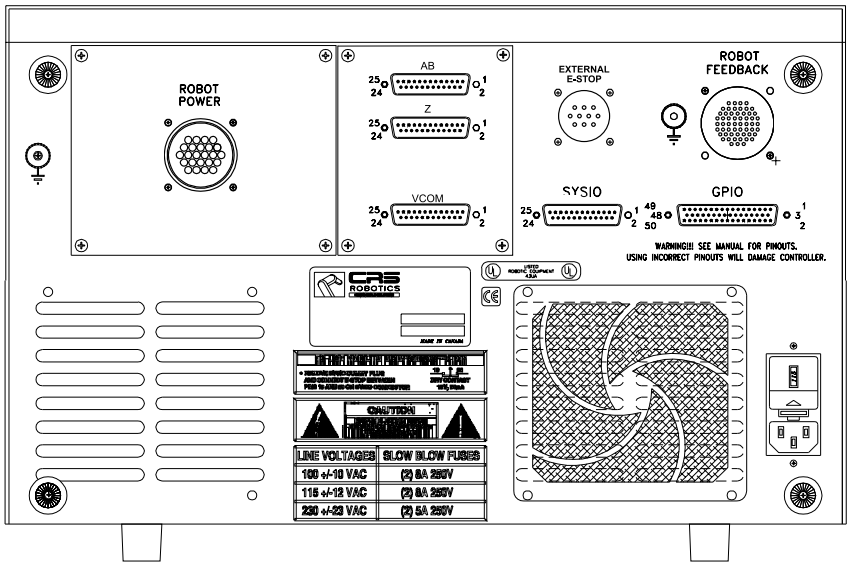


(c) Tipo de datos y flujo de información

Figura 4.2: Tarjeta para la re-distribución de canales del controlador C500.



(a) Vista superior del interior



(b) Vista trasera exterior del controlador C500 con conectores DB-25 AB, Z y VCOM

Figura 4.3: Re-distribución de canales del controlador C500.

4.2 | Etapa Digital: Robot A465

Como hemos visto, las señales provenientes de los codificadores (A, B, Z y sus complementos) le permiten saber al robot el estado en el que se encuentra mediante el análisis de éstas. Pero al transmitir las señales al controlador, sufren una caída de voltaje además de generarse ruido provocado por elementos externos, lo que conllevaría al robot a situarse en un estado erróneo. Para disminuir tal error, uno de los objetivos de esta tesis es tratar las señales en la tarjeta de adquisición de datos y control antes de ser adquiridas por el sistema CompactRIO.

En la gráfica de la Figura 4.4 se muestra un ejemplo del comportamiento de las señales antes de ser tratadas, donde la amplitud de la señal es menor al que se otorga directamente del codificador y además presenta ruido.

Por lo anterior se propuso utilizar el circuito integrado 26LS32 (Figura 4.5) el cual contiene cuatro receptores diferenciales y es utilizado para la transmisión de datos digitales tanto balanceados (como es el caso), como no balanceados. Este integrado reduce el error en la señal y no necesariamente debe de estar referida a la misma tierra que el circuito de donde provienen las señales gracias al comparador diferencial.

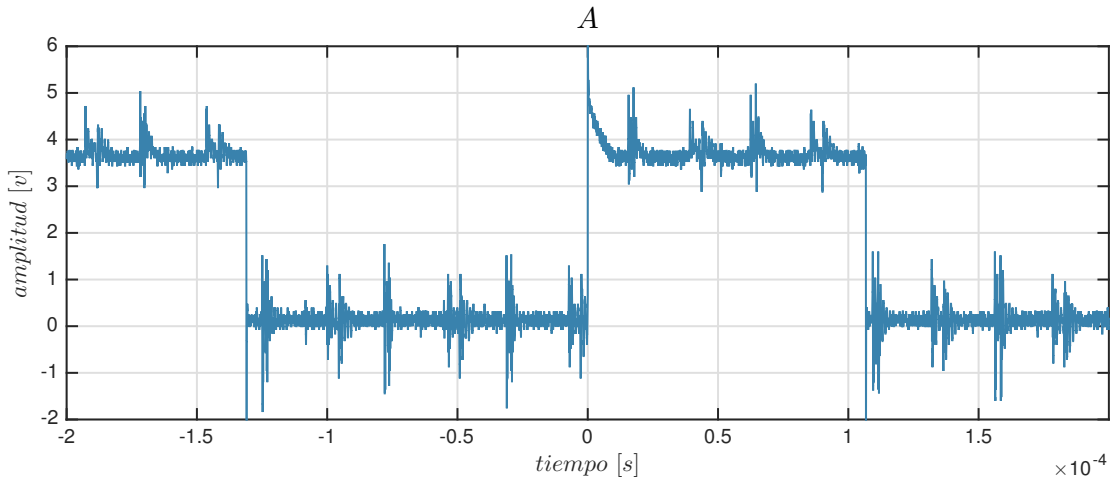


Figura 4.4: Gráfica de la señal A de la primera articulación del robot A465.

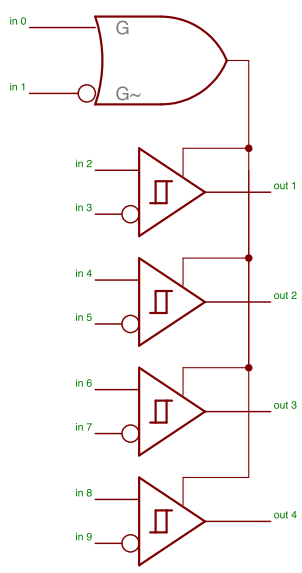


Figura 4.5: Diagrama lógico del circuito 26LS32.

Conociendo la tabla de verdad, Tabla 4.1, y la naturaleza de las señales provenientes de los codificadores se concluye que sólo existirán dos casos. El primero es con $V_{ID} \geq V_{IT+}$ donde $A = 5[V]$ y $B = 0[V]$ obteniendo un uno lógico a la salida. El segundo caso es el opuesto, cuando $V_{ID} \leq V_{IT-}$ y $Y = 0[V]$.

Para evitar ambigüedad, el puerto G se conectó a V^+ y $\bar{G}$ a V^- reduciendo la tabla a los cuatro casos limitados por líneas moradas horizontales en la Tabla 4.1.

Con el uso del circuito 26LS32 el número de señales se redujo ya que se comparan las señales A, B y Z con sus complementos dando como resultado una señal no balanceada con menor ruido como se aprecia en la gráfica de la Figura 4.6.

Diferencia	Activación		Salida
$V_{ID} = A - B,\ V_T = \pm 0.2$	G	$\bar{G}$	Y
$V_{IT-} \leq V_{ID} \leq V_{IT+}$	H	X	?
$V_{IT-} \leq V_{ID} \leq V_{IT+}$	X	L	?
$V_{ID} \geq V_{IT+}$	H	X	H
$V_{ID} \geq V_{IT+}$	X	L	H
$V_{ID} \leq V_{IT-}$	H	X	L
$V_{ID} \leq V_{IT-}$	X	L	L
X	L	H	Z
Abierto	H	X	H
Abierto	X	L	H

Tabla 4.1: Tabla de verdad del circuito 26LS32 [26].

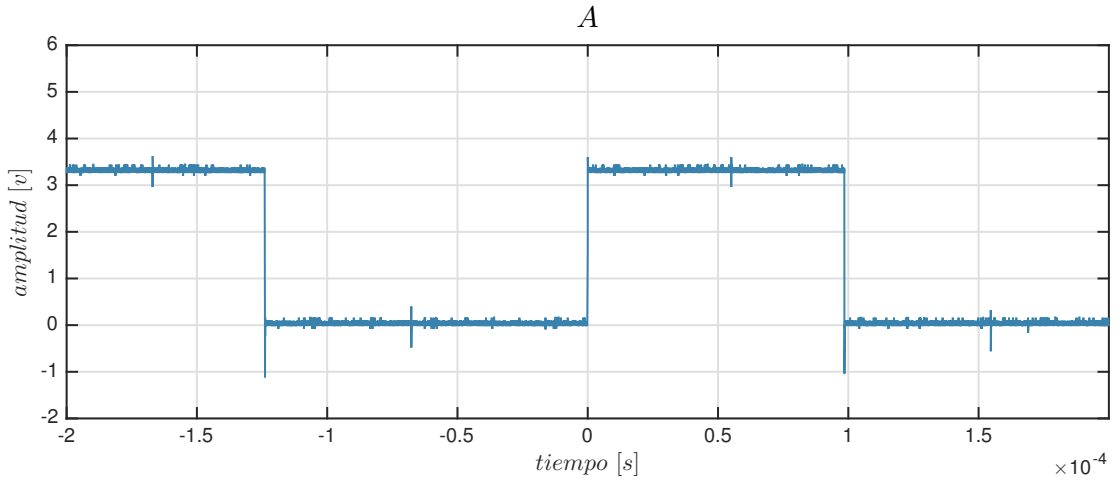


Figura 4.6: Gráfica de la señal A no balanceada de la primera articulación del robot A465.

Para aislar las señales de los módulos de la compactRIO se propuso utilizar integrados HCPL2630 para opto-acoplar al circuito. Cada empaquetado DIP (Dual In-line Package) se compone por dos canales con salida de compuerta lógica NOT a una velocidad máxima de $10[MB/s]$. En la Figura 4.7 se muestran encerrados en un recuadro con linea punto-guión, los canales A y B de un integrado HCPL2630. Fuera de él se encuentran los elementos requeridos para su correcto funcionamiento. Las resistencias $R1$ y $R2$ son necesarias para evitar que el diodo emisor de luz se queme y sus valores son obtenidos con el uso de la ley de voltajes de Kirchhoff y la ley de OHM:

Ecuación dada por la Ley de voltajes de Kirchhoff para el circuito:

$$V_{input} = V_D + V_R \tag{4.5}$$

Ecuación dada por la Ley de corrientes de Kirchhoff para el circuito:

$$I_D = I_R \tag{4.6}$$

Sustituyendo y despejando valores en (4.5):

$$5[V] = 1.8[V] + V_R \therefore V_R = 3.2[V] \tag{4.7}$$

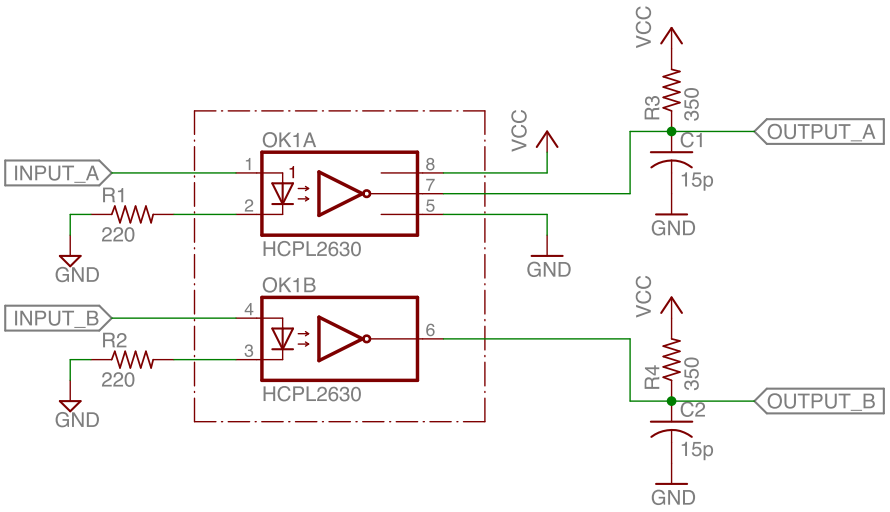


Figura 4.7: Circuito HCPL2630 con elementos externos [27].

Ley de Ohm para la resistencia $R1$ o $R2$

$$V_R = R(I_R) \tag{4.8}$$

Sustituyendo (4.6) en (4.8)

$$V_R = R(I_D) \tag{4.9}$$

Sustituyendo valores y despejando en (4.9)

$$3.2[V] = R(15[mA]) \therefore R = 213.3[\Omega] \tag{4.10}$$

La resistencia adecuada resultó ser de 213.3[Ω] pero se utilizó una resistencia de valor comercial de 220[Ω]. A la salida del opto-acoplador se tiene una compuerta lógica NOT de colector abierto por lo que se requiere una resistencia en configuración push-pull para poder alzar el voltaje a la salida ($R3$ y $R4$ en el diagrama de la Figura 4.7).

Con el análisis de las gráficas de la hoja de datos del integrado (ver apartado en el Apéndice C.4.2) se concluyó que con resistencias menores a $1[K\Omega]$ el circuito consume mayor cantidad de energía pero el tiempo de retraso de propagación es menor que cuando se usan resistencias mayores, esto produce una deformación y una caída en amplitud de la señal. Se busca tener la

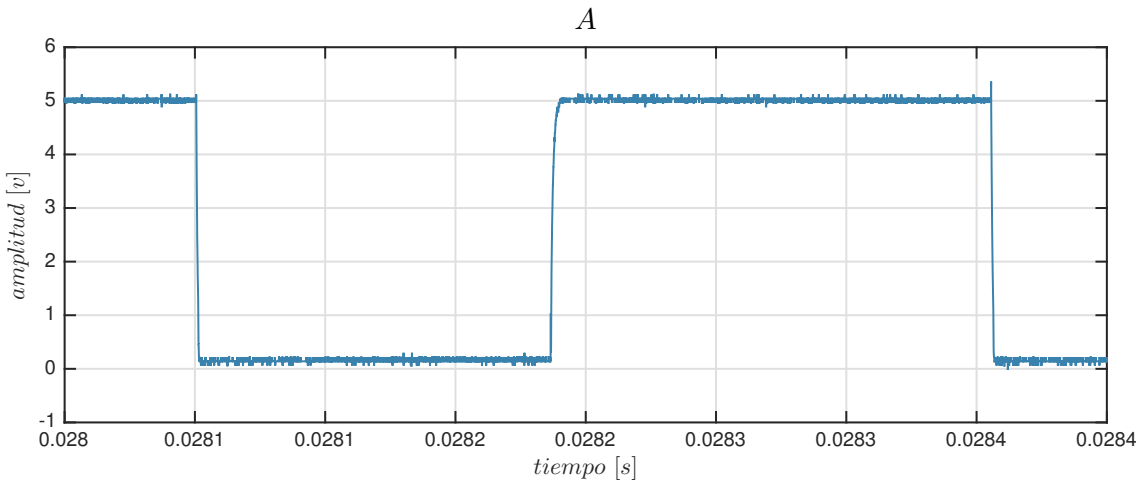


Figura 4.8: Gráfica de la señal A no balanceada y opto-acoplada de la primera articulación del robot A465.

menor alteración posible en la señal por lo que se procedió a encontrar la resistencia cuya señal resultante fuera la mejor. Los capacitores $C1$ y $C2$ a pesar de no ser requeridos, el fabricante recomienda su uso y el valor que sugiere es de $15[pF]$.

En el ejemplo del canal A de la primera articulación del robot A465, la señal resultante es la mostrada en la Figura 4.8, donde se aprecia un ligero comportamiento similar a un sistema de primer orden pero con la amplitud adecuada para ser transmitida a los módulos de lectura del sistema compactRIO. El esquema del circuito para los canales A, B y Z de las seis articulaciones del robot A465 se ilustra en la Figura 4.9.

Adicional a las señales de los encoders, el robot A465 provee la señal HSW que indica con un uno lógico cuando el robot está en estado de *home*, para opto-acoplar estas señales se utiliza el circuito de la Figura 4.11.

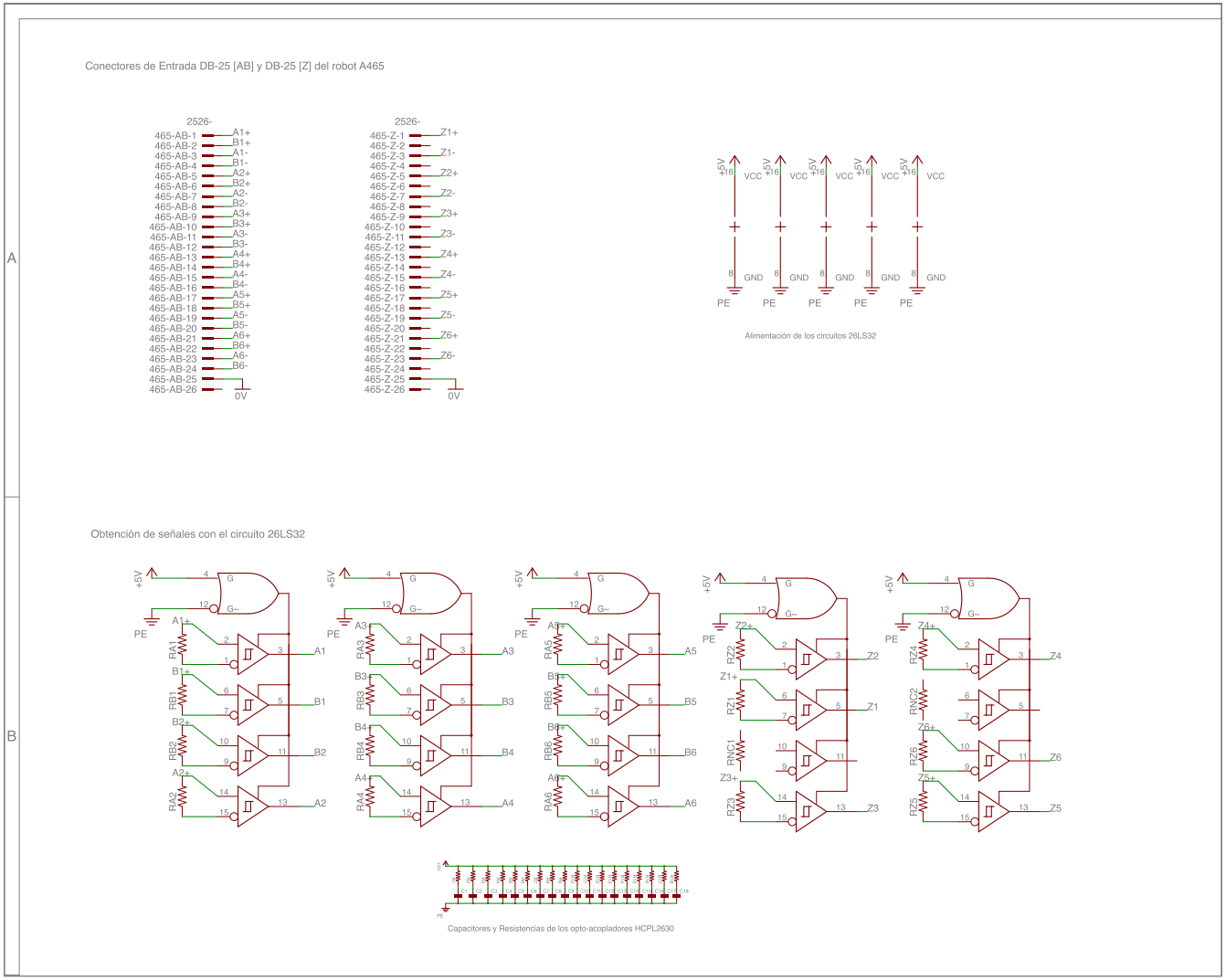


Figura 4.9: Circuito para el acondicionamiento de señales A, B y Z del robot A465 parte I.

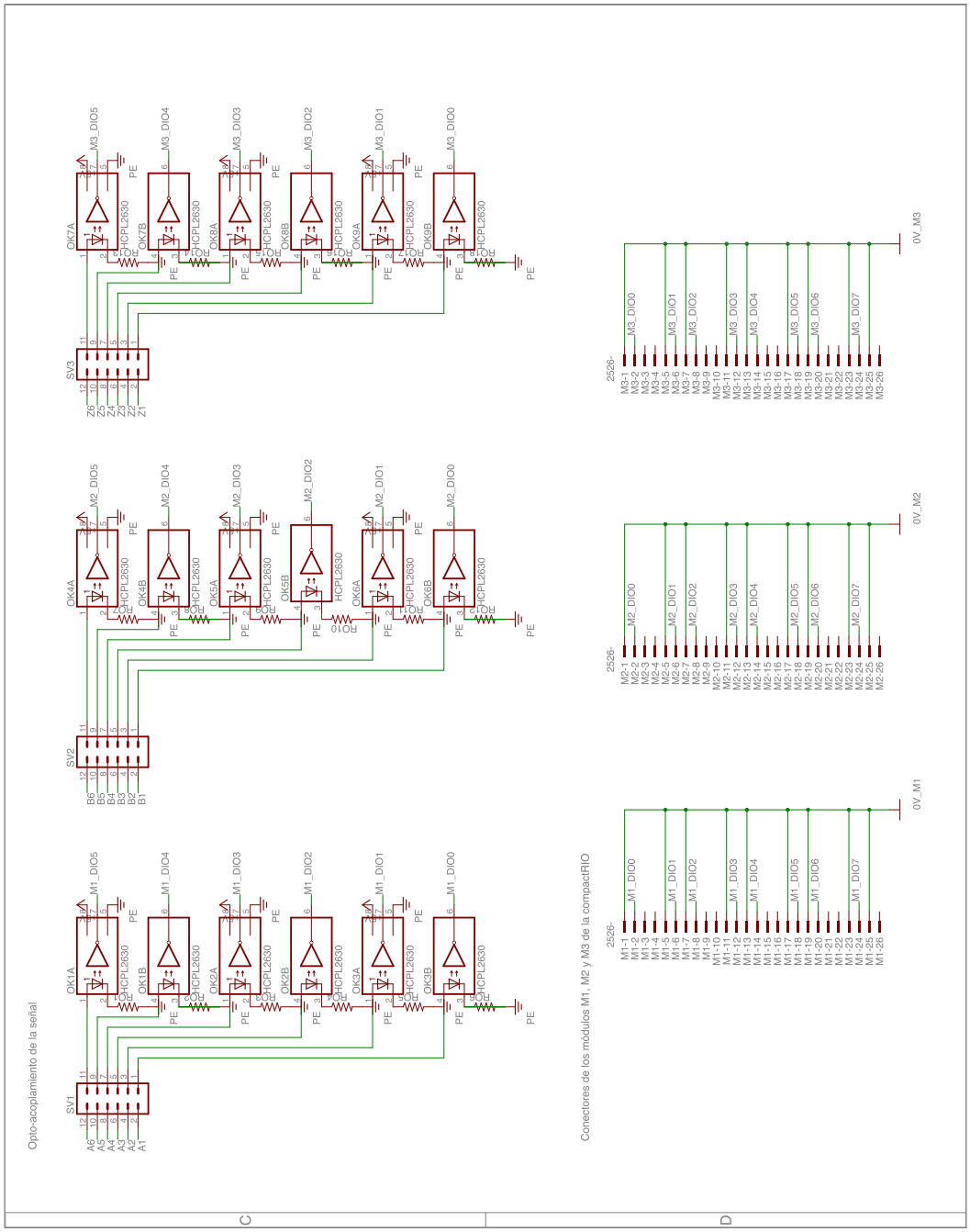


Figura 4.10: Circuito para el acondicionamiento de señales A, B y Z del robot A465 parte II.

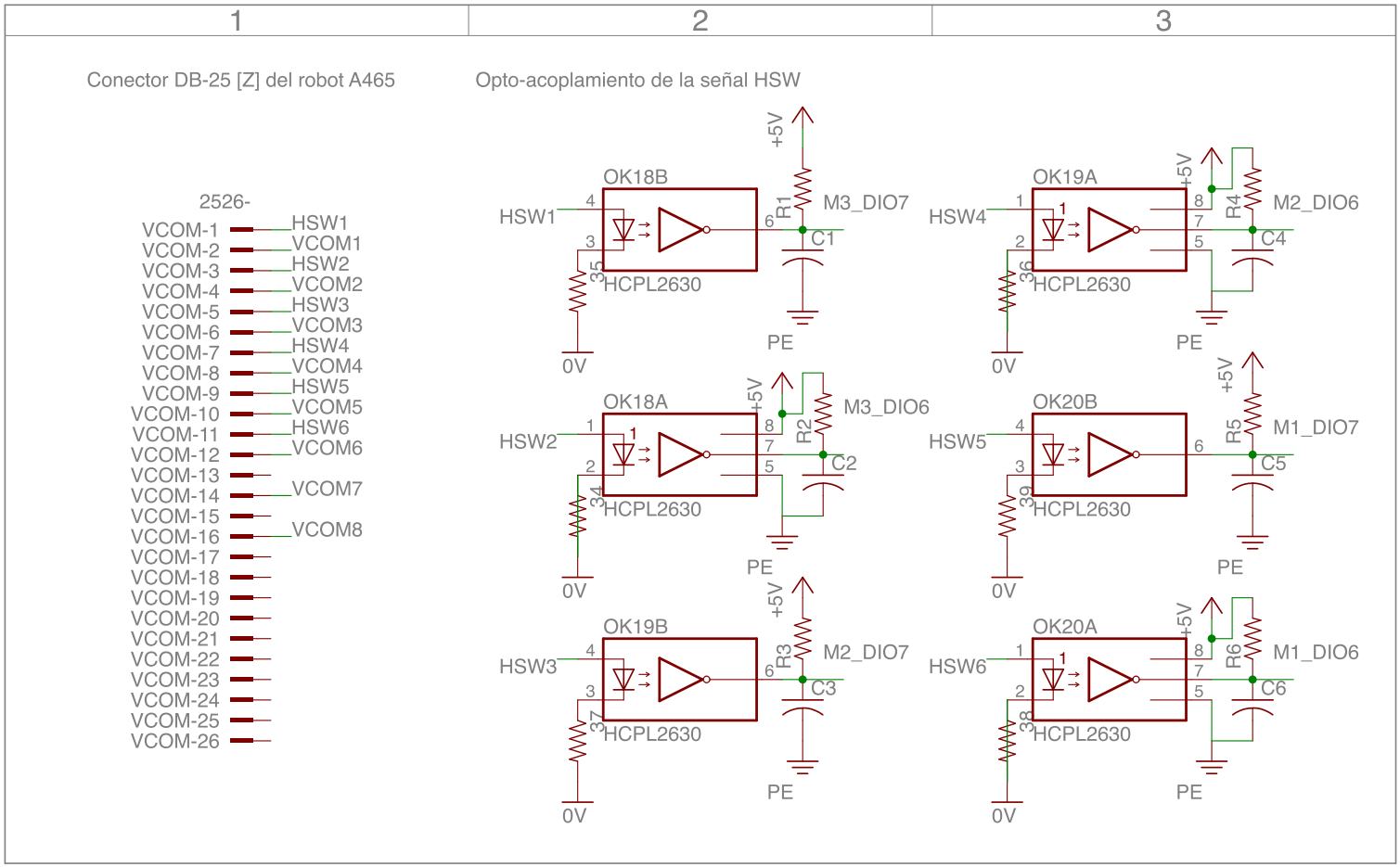
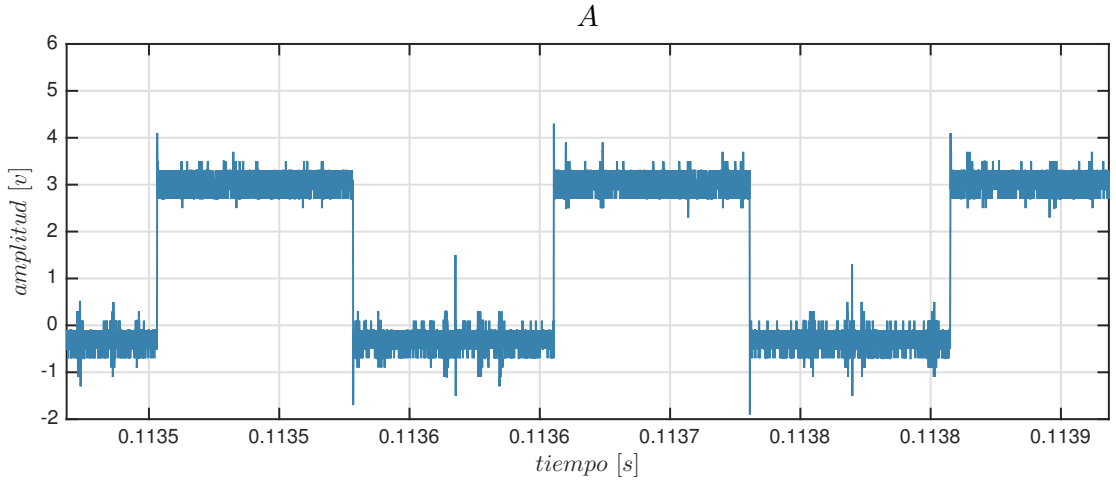


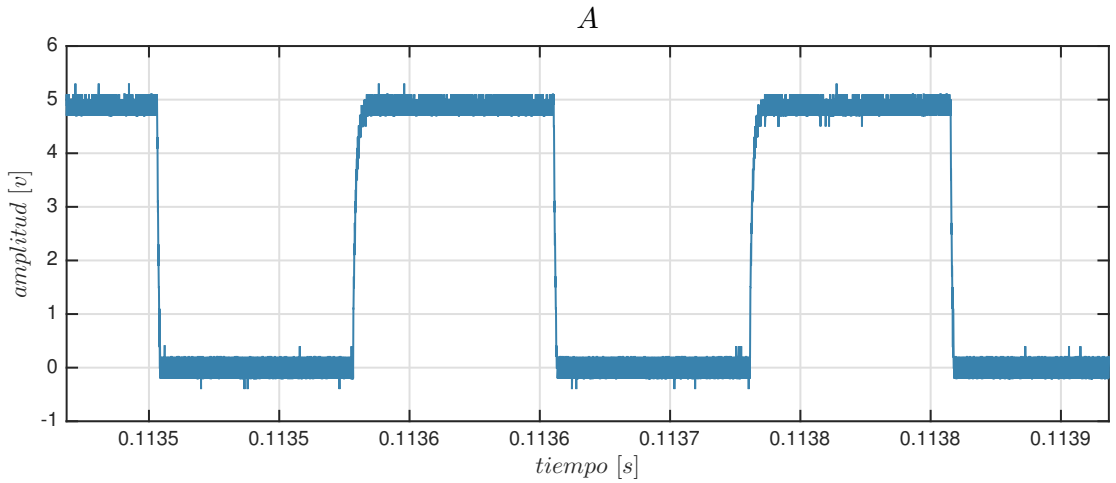
Figura 4.11: Circuito para el acondicionamiento de señales *HSW* del robot A465.

4.3 | Etapa Digital: Robot A255

Como se mencionó en la Sección 3.1.1, el robot A255 no dispone del complemento de las señales A, B y Z así como de la señal HSW en ninguna de sus articulaciones. Al carecer de una señal balanceada, se omitió el uso del circuito 26LS32 para el acondicionamiento de señales y únicamente se opto-acoplaron los canales con los integrados HCPL2630 utilizando la misma configuración que se usó en el robot A465. El circuito para esta etapa se encuentra en el diagrama de la Figura 4.13.



(a) No balanceada



(b) No balanceada y opto-acoplada

Figura 4.12: Gráficas de la señal A de la primera articulación del robot A255.

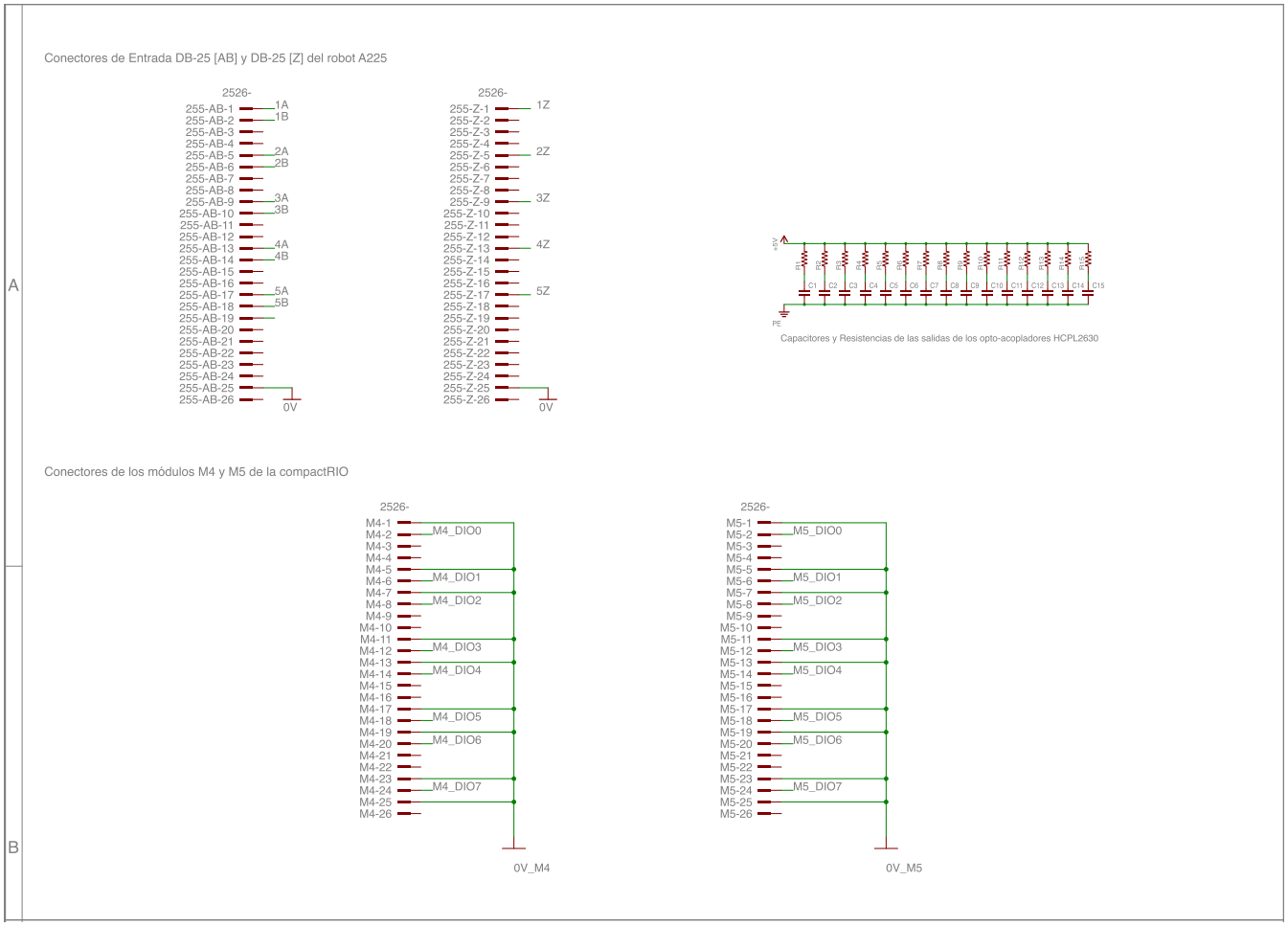


Figura 4.13: Circuito para el acondicionamiento de señales A, B y Z del robot A255 parte I.

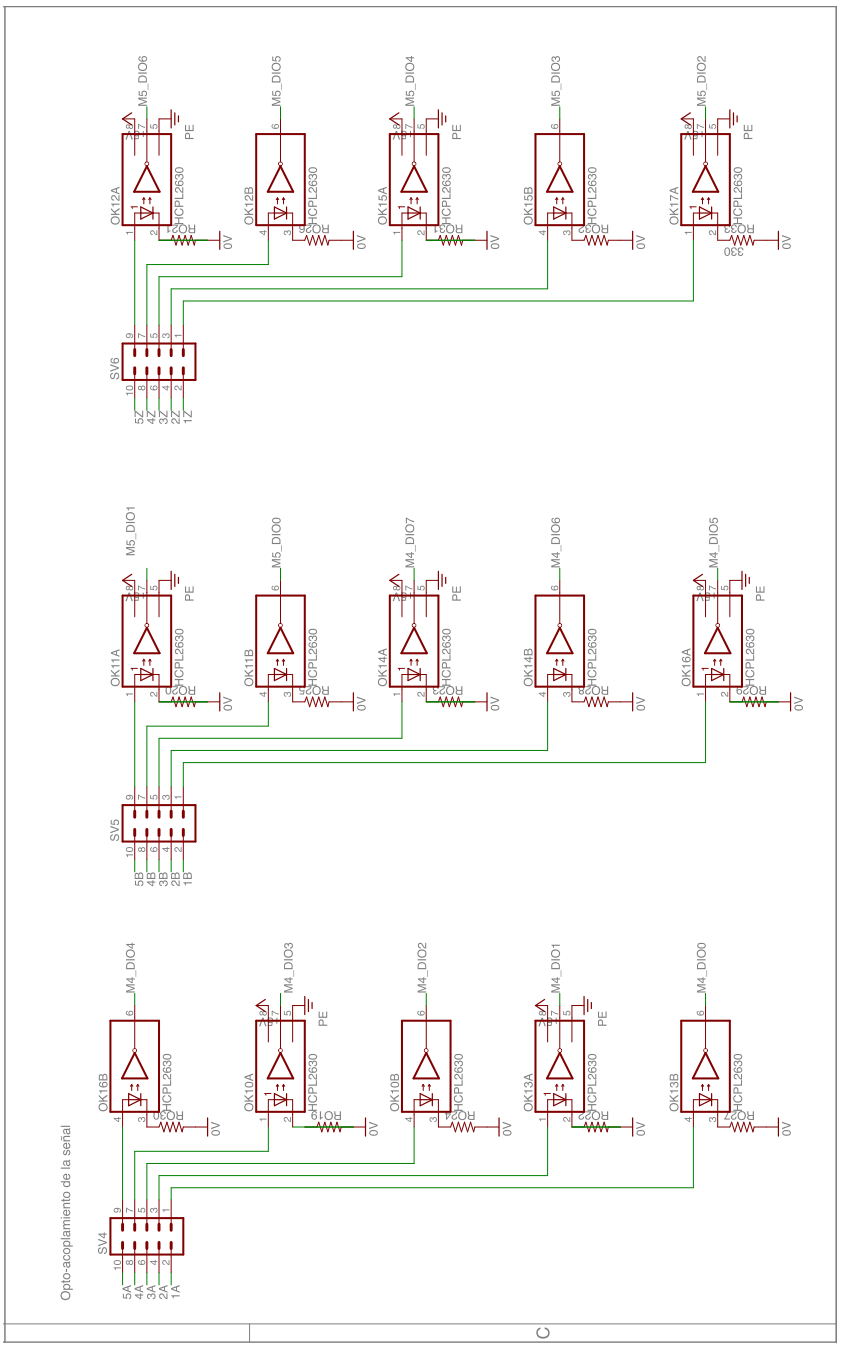


Figura 4.14: Circuito para el acondicionamiento de señales A, B y Z del robot A255 parte II.

4.4 | Etapa Analógica

Los módulos utilizados por la compactRIO se conforman por tres etapas: un convertidor digital a analógico aislado, un amplificador y una etapa de protección contra sobre voltaje y corto circuito. Aprovechando estas propiedades y con la experiencia de no haber reportado ningún tipo de problema con anterioridad al comunicar la señal de control a la etapa que le da potencia a los motores de los robots ubicada en el controlador C500, se decidió conectar directamente las señales provenientes del módulo NI 9263 al conector DB-25 [VCOM] del controlador C500 (ver Figura 4.15).

Cabe aclarar que los tres módulos NI 9263 ($M6$, $M7$ y $M8$) comparten la misma tierra junto con el controlador C500 pero por ser módulos aislados no hay continuidad en tierra con los demás módulos, esto quiere decir que el sistema se maneja al mismo potencial pero no se comparte la continuidad a tierra por protección.

La señal analógica sobrante, en la Figura 4.15 bajo la etiqueta de BTN_CTRL se destinó al control de la botonera mediante software.

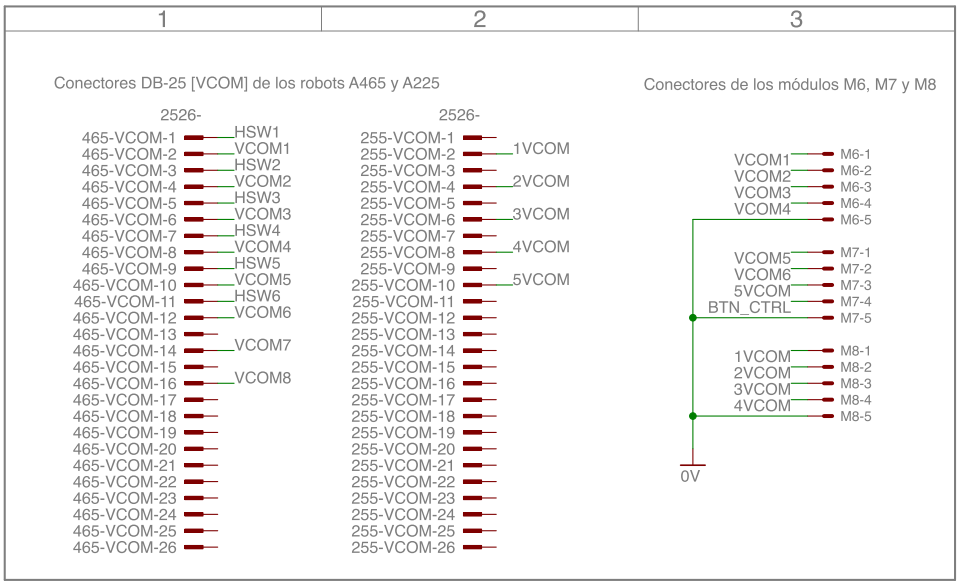


Figura 4.15: Conexiones de la etapa analógica.

4.5 | Botonera

Para poder controlar la botonera mediante una señal analógica se propuso utilizar un micro-controlador para interpretar la señal y establecer casos de acuerdo al nivel del voltaje. A comparación de utilizar un convertidor analógico a digital únicamente, el uso de un micro-controlador extiende las posibilidades de experimentación para el accionamiento de actuadores o indicadores adicionales que se deseen implementar a futuro y evita sobrecargar el programa que se ejecutará por el sistema compactRIO.

La señal proveniente del módulo NI 9263 otorga un nivel de voltaje de $-10[V]$ a $10[V]$, este rango puede provocar errores y/o dañar al micro-controlador, para su protección se utilizó un integrado 4N35 para poder acoplar la señal. En la Figura 4.16 se muestran los elementos necesarios para el correcto funcionamiento de éste.

La resistencia $R1$, al igual que en el circuito HCPL2630, fue calculada mediante el uso de la ecuación (4.5) resultado de la ley de voltajes de Kirchhoff. Al sustituir valores, el voltaje de la resistencia es de $8.2[V]$:

$$10[V] = 1.8[V] + V_R \therefore V_R = 8.2[V] \tag{4.11}$$

Con el uso de la ley de Ohm (ecuación (4.8)) y sustituyendo valores, la resistencia adecuada es de $546.66[\Omega]$ para el cual el valor comercial más cercano de $560[\Omega]$.

$$8.2[V] = R(15[mA]) \therefore R = 546.66[\Omega] \tag{4.12}$$

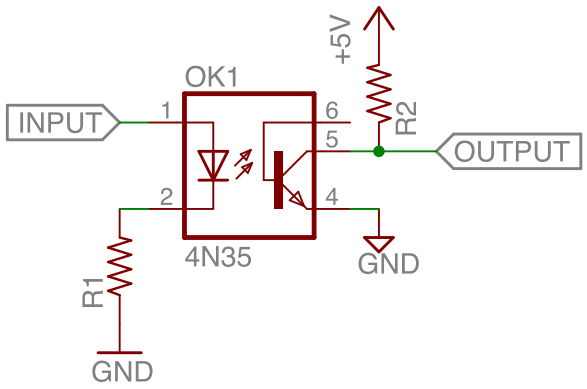


Figura 4.16: Integrado 4N35 y elementos externos [28].

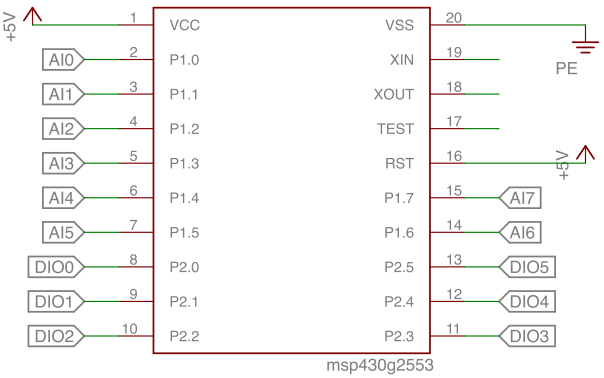


Figura 4.17: Micro-controlador MSP430g2553.

Para la resistencia $R2$ se recurrió al análisis de especificaciones incluidas en la hoja de datos del integrado (incluidas en el Apéndice C.4.3) y los mejores resultados se encuentran con el uso de resistencias menores a mil Ohms. Con esto la señal que llega al micro-controlador es de $0[V]$ a $5[V]$ con una resolución 15 bit de 32768 niveles (la mitad de estados del módulo NI 9263 cuya resolución es de 16 bit).

El micro-controlador seleccionado fue el MSP430g2553 por su accesibilidad, calidad y bajo costo. Éste es un integrado de bajo consumo con frecuencia interna mayor a $16[MHz]$ y convertidor analógico digital de 10 bit que representa 1024 niveles de voltaje.

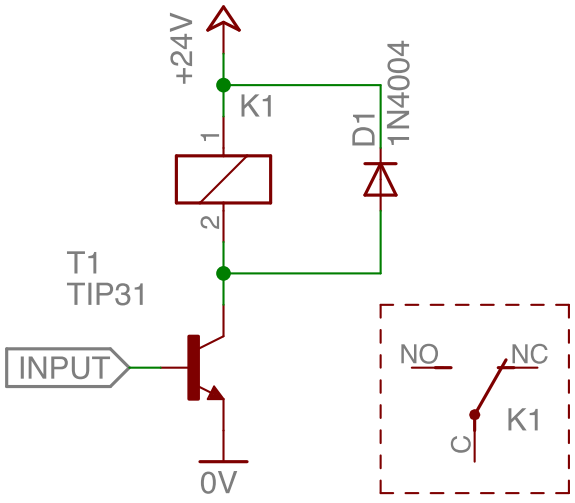


Figura 4.18: Circuito para controlar un relé [29].

En la Figura 4.17 se encuentra una de las varias configuraciones que puede tener el micro-controlador, en este caso se configuró con una sola entrada analógica (pin 2 AI0), cuatro salidas digitales para el control de paro y encendido de los robots (pines 11, 12, 13 y 14) y ocho pines de propósito general a los cuales se puede acceder mediante los conectores tipo molex AUX1 y AUX2. Dichos conectores se muestran en el esquema del circuito utilizado para esta etapa en la Figura 4.19.

Haciendo uso del conector para botonera externa del controlador C500 de cada robot, se intervino al circuito de paro y activación de los robots con relés conectados al micro-controlador, por una botonera física y sensores de impacto. El circuito propuesto para esto es el mostrado en la Figura 4.18 y 4.19 donde se muestra la configuración de corte y saturación del transistor TIP31C. El inductor del relé se conectó entre la alimentación de 24[V] y el colector con un diodo de protección en paralelo para disipar las corrientes que éste pueda almacenar. Más información referente a éste integrado se muestra en la Sección C.4.4.

Al no ser accionado alguno de los relés, el paro y encendido de los robots corre a cargo de la botonera física y de los sensores de impacto. Estos últimos pueden o no ser parte del de circuito gracias al switch SW1, el cuál cierra el circuito en estado de ON a ausencia del sensor de fuerza. Como medida de protección del sensor se sugiere cambiar a estado de OFF al switch antes de ser colocado y mantenerlo así hasta después de haber sido desconectado. El interruptor SW2 permite seleccionar el tipo de botonera, en estado de OFF permite ampliar la botonera física a 4 botones para parar o activar a cada robot. En estado de ON el switch permite detener a los robots mediante un sólo botón.

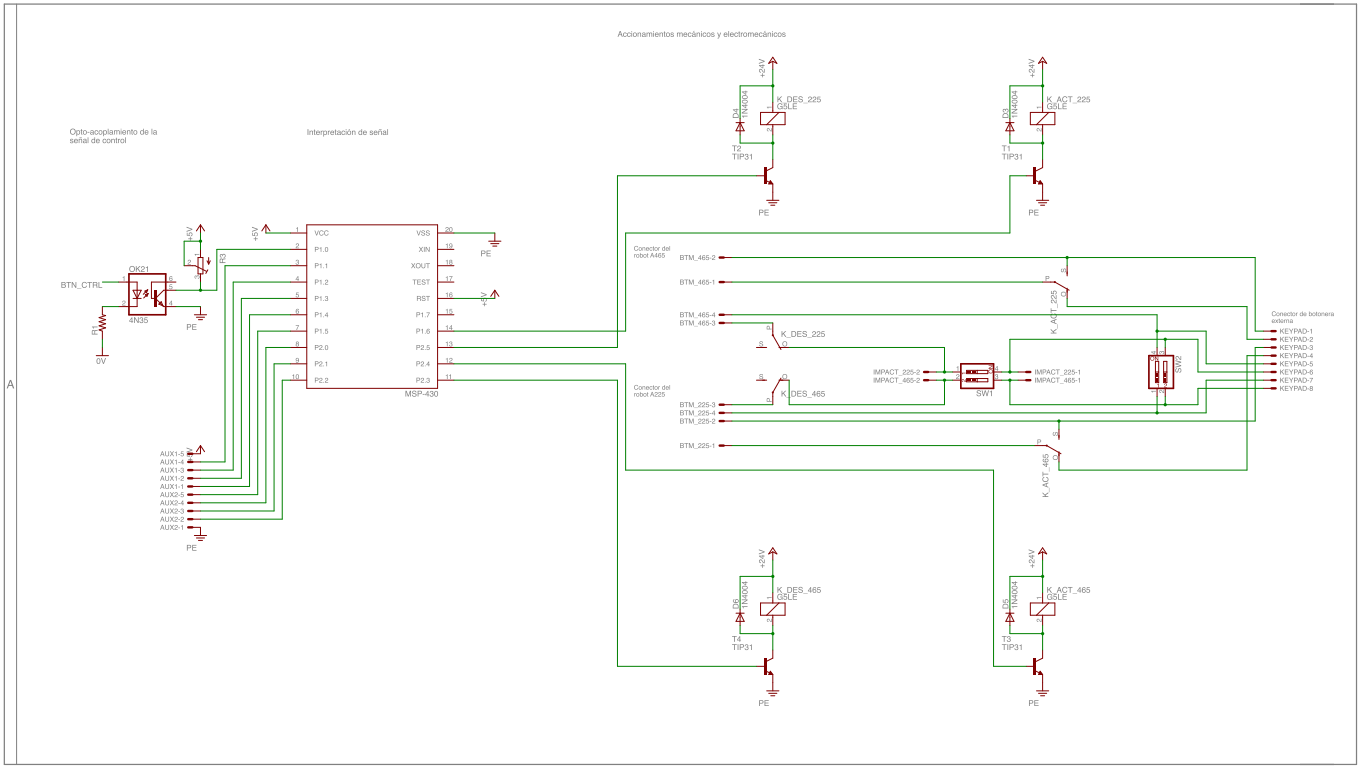


Figura 4.19: Circuito para el control de la botonera.

4.6 | Etapa de potencia y protección

Como se vio en la Sección 3.3.3, la alimentación de la tarjeta y del sistema compactRIO proviene de una unidad Quint Power que provee un voltaje de 24$[V]$, para reducir éste voltaje para alimentar a los integrados de la tarjeta, se utilizó el circuito integrado R785.0-0.5 de la marca *Recom*, que realiza la misma función que un integrado LM7805 pero es capaz de entregar mayor corriente. Los capacitores $C1$ y $C2$ en el esquema de la Figura 4.21 no son fundamentales para el funcionamiento del sistema, pero se utilizan para suavizar la fluctuación en los niveles de voltaje.

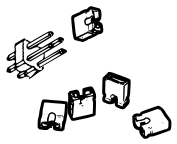


Figura 4.20: Header y jumpers utilizados en la tarjeta.

Para la protección de la tarjeta, además de incluir conectores tipo Molex para los elementos externos a la tarjeta, se incluye un diodo 1N4004 a la entrada que permite la alimentación en un sólo sentido, este diodo genera una caída de voltaje de 1$[V]$ y soporta 50$[V]$ en inversa. De forma opcional se añadió un circuito conformado por un relé que controla dos switches SPDT (un polo, dos tiros) que al ser accionado y con ayuda del diodo 1N4004 anterior al relé, permite la alimentación correcta al circuito. Para la selección de éste modo, se requiere conectar dos jumpers (Figura 4.20) entre los pines 1-3 y 2-4 de cada header de 2x2 (JP1 y JP2 en la Figura 4.21).

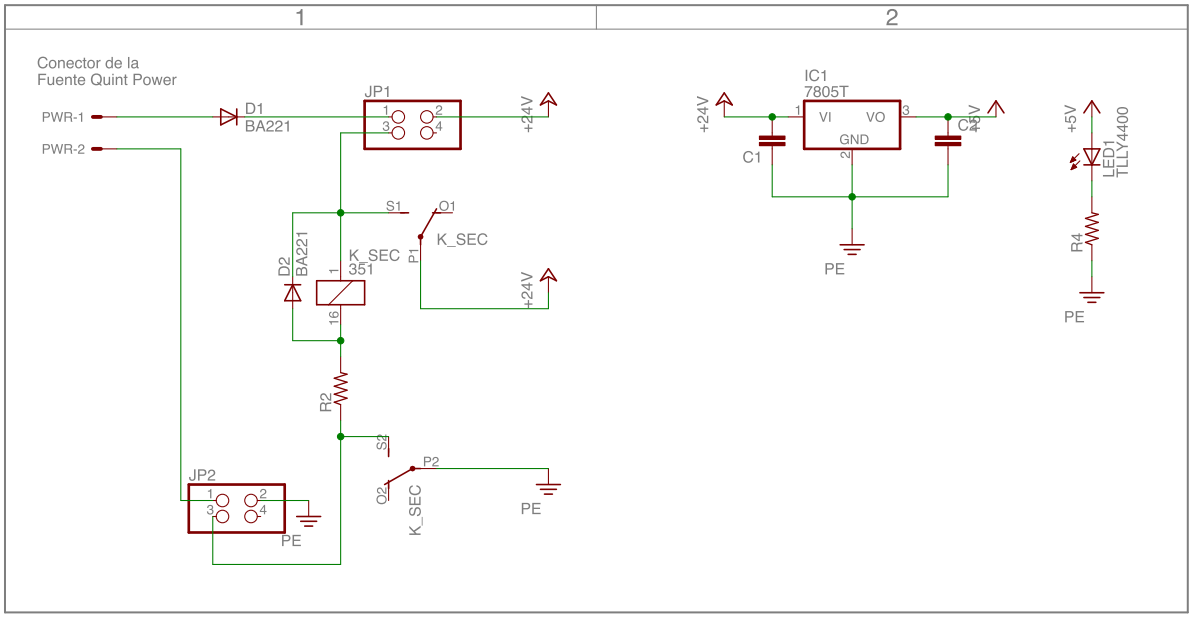


Figura 4.21: Circuito para protección de la tarjeta.

4.7 | Gabinete

Con el objetivo de concentrar los elementos vistos en este capítulo dentro de una misma unidad se integraró un gabinete tal y como se expone en el esquema de Figura 4.22. Dichos elementos son: la fuente de poder Quint Power, el sistema embebido compactRIO, la tarjeta de adquisición de datos y control y dos relés. Éstos últimos junto con los interruptores $S1$ y $S2$ conforman un circuito para suministrar energía a la fuente de poder Quint Power y a una salida para alimentar algún elemento externo.

El relé de la marca Finder tipo 95.65 tiene una configuración SPDT que al ser accionado mediante el interruptor $S1$ permite la alimentación a la fuente Quint Power, de igual forma habilita al puerto de alimentación de salida si el interruptor $S2$ del relé Moeller modelo ETR4-69-A se encuentra accionado.

El acabado del gabinete y especificaciones técnicas de éste se encuentran en la SecciónC.3.

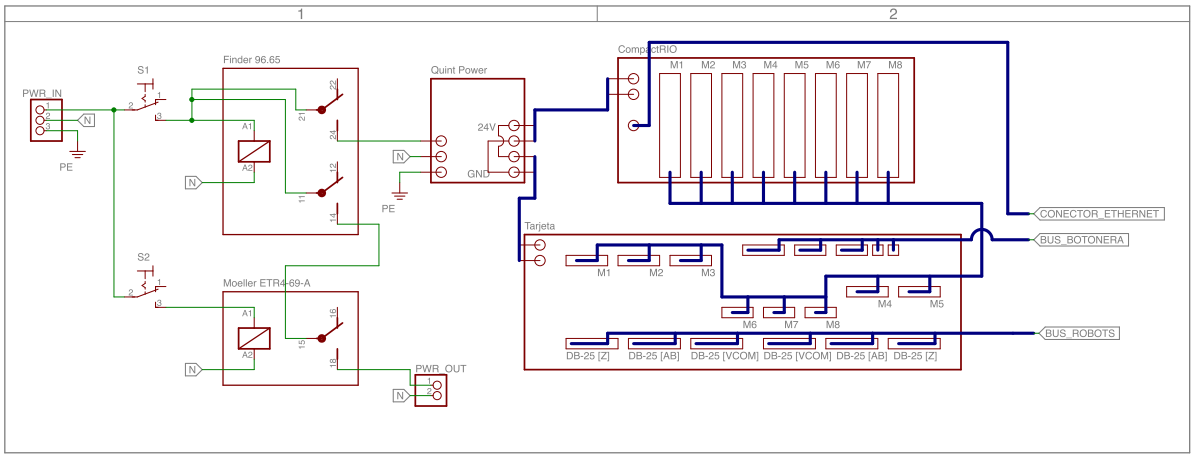
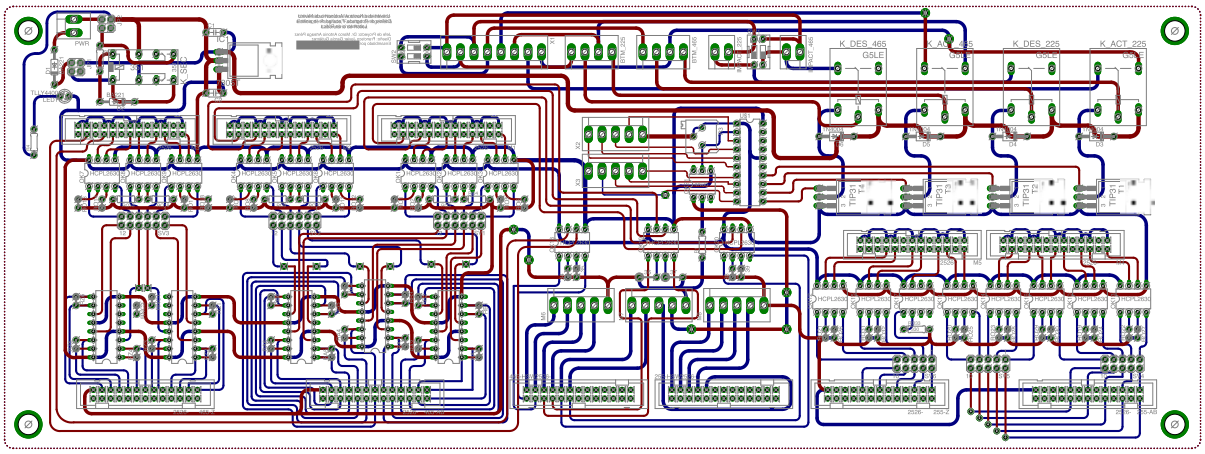
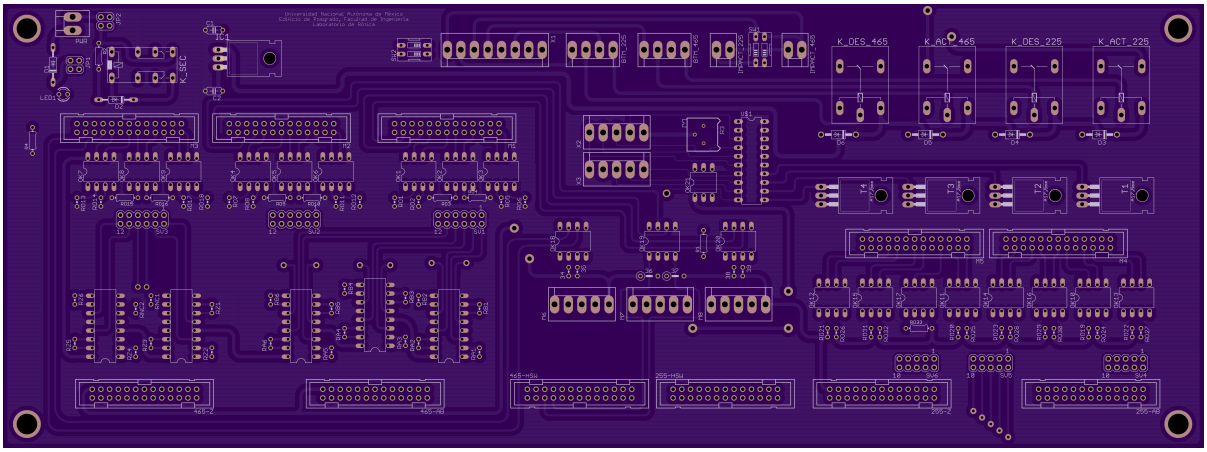


Figura 4.22: Circuito interno del gabinete.

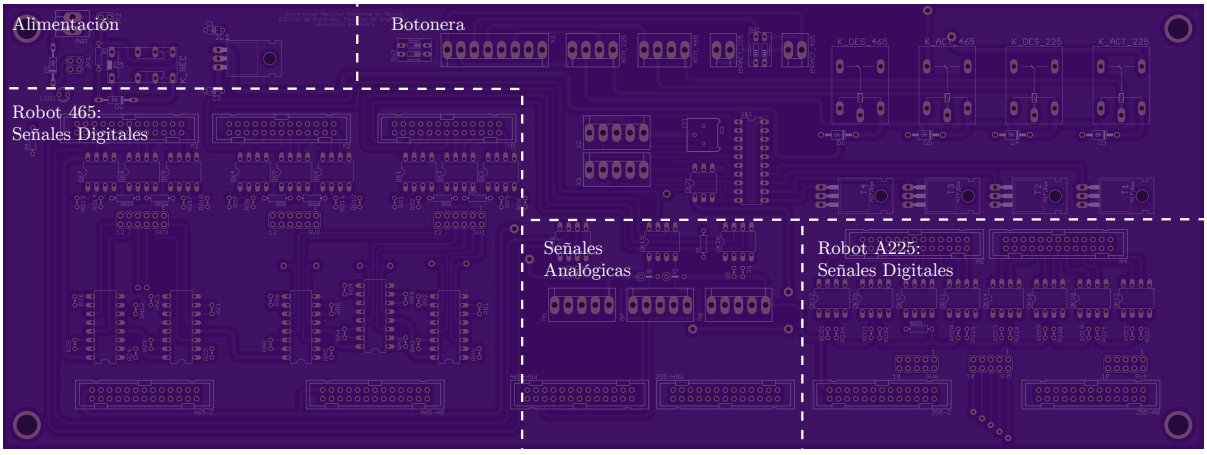
En la Figura 4.23 se muestra el diseño en *EAGLE*, la vista frontal del circuito impreso y la tarjeta dividido en las secciones vistas en esta sección. En la Sección C.4 se encuentra mayor información referente a la tarjeta.



(a) Diseño en EAGLE.



(b) Vista frontal.



(c) Tarjeta por secciones

Figura 4.23: Tarjeta para la adquisición de datos y control de los robots.

4.8 | Galería de Fotos

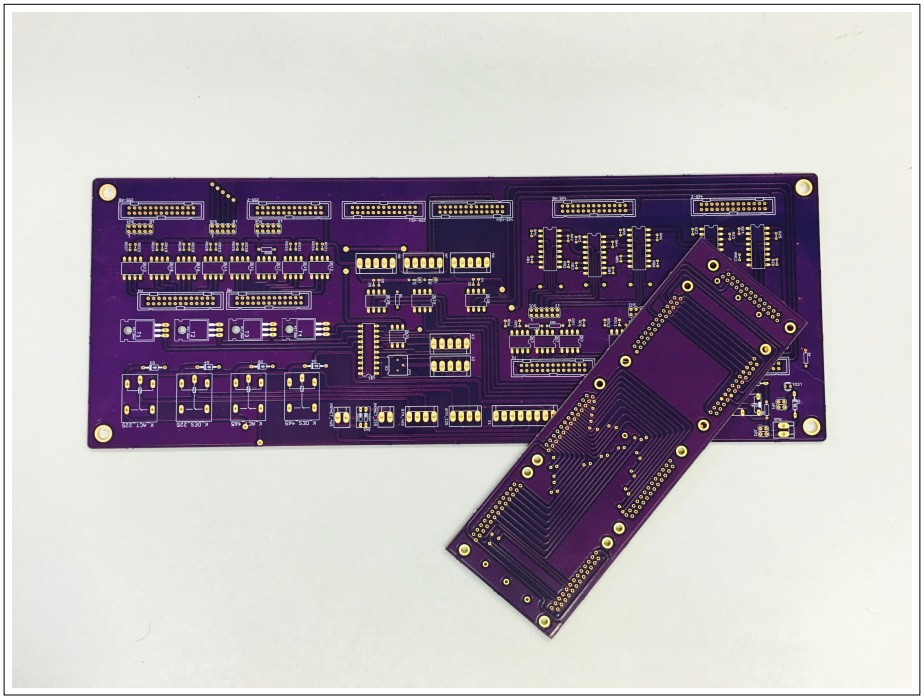


Figura 4.24: Tarjetas sin ensamblar.

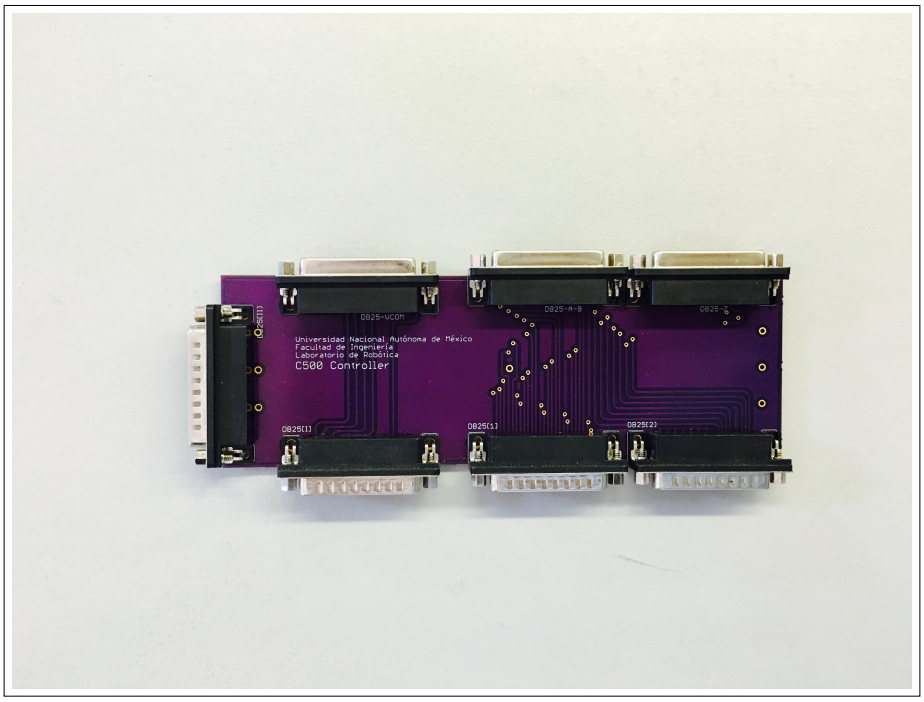


Figura 4.25: Tarjeta ensamblada del controlador C500 para el ordenamiento de señales.

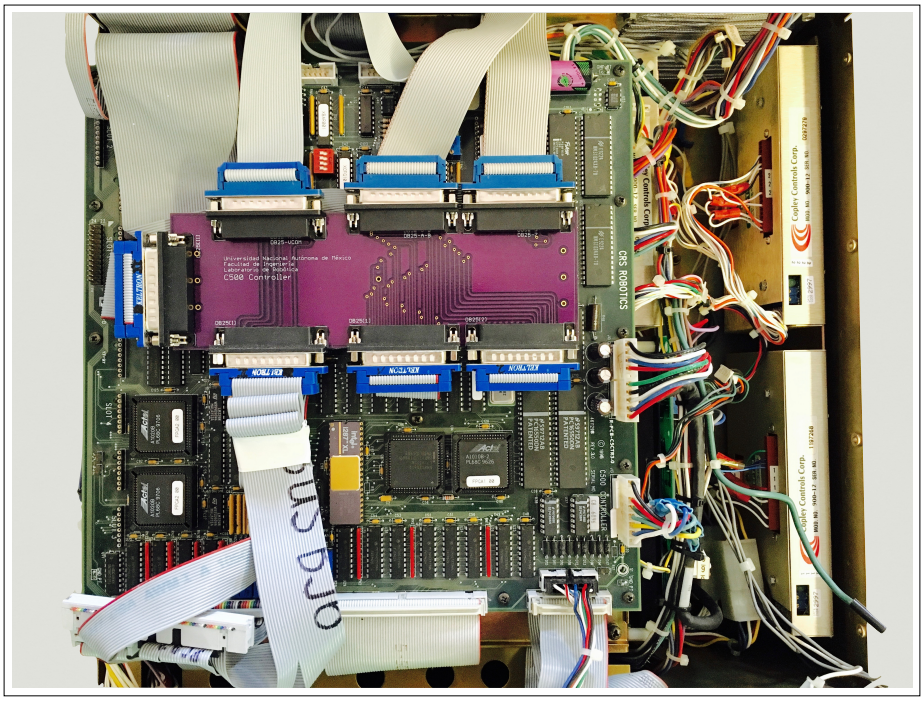


Figura 4.26: Interior del controlador C500 e inhibición de etapa de control

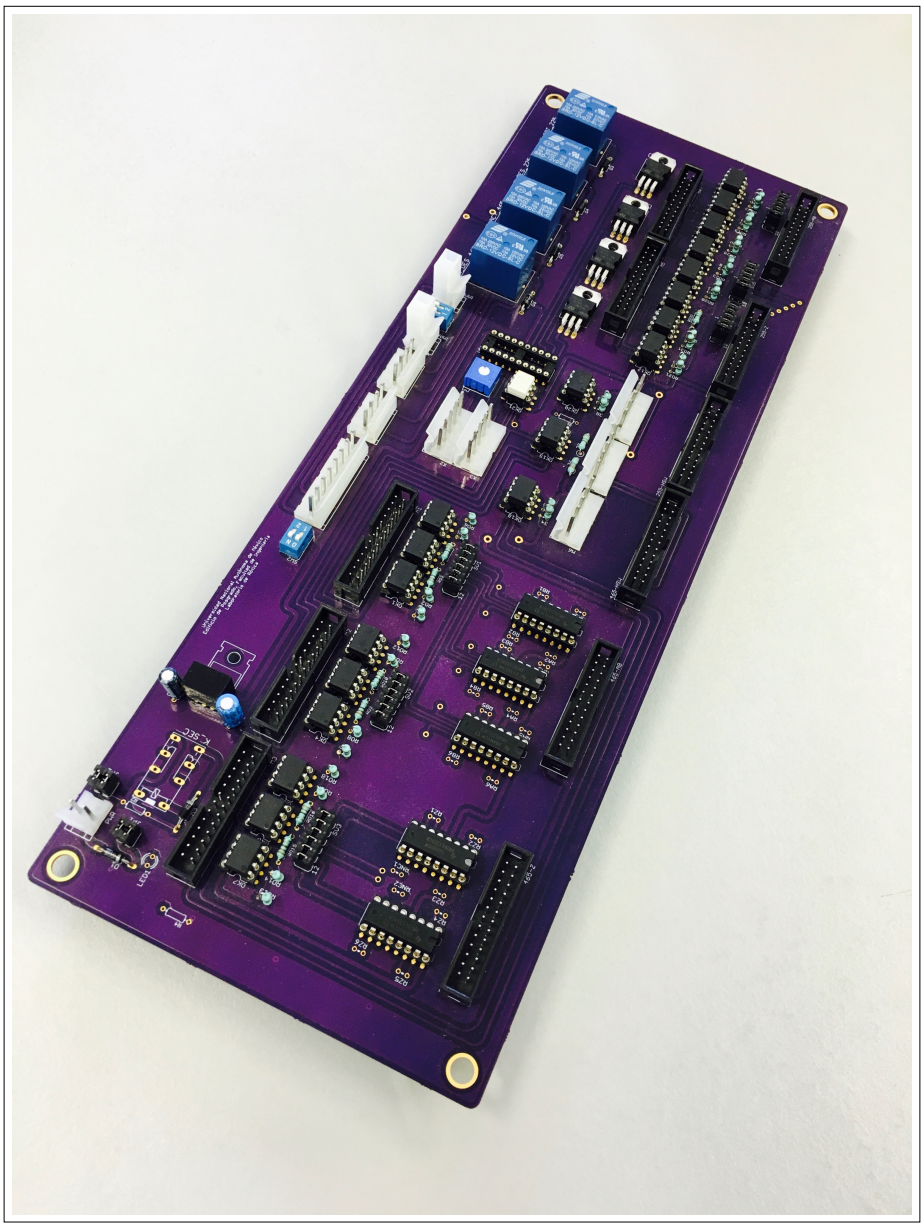


Figura 4.27: Tarjeta para la adquisición de señales y control ensamblada

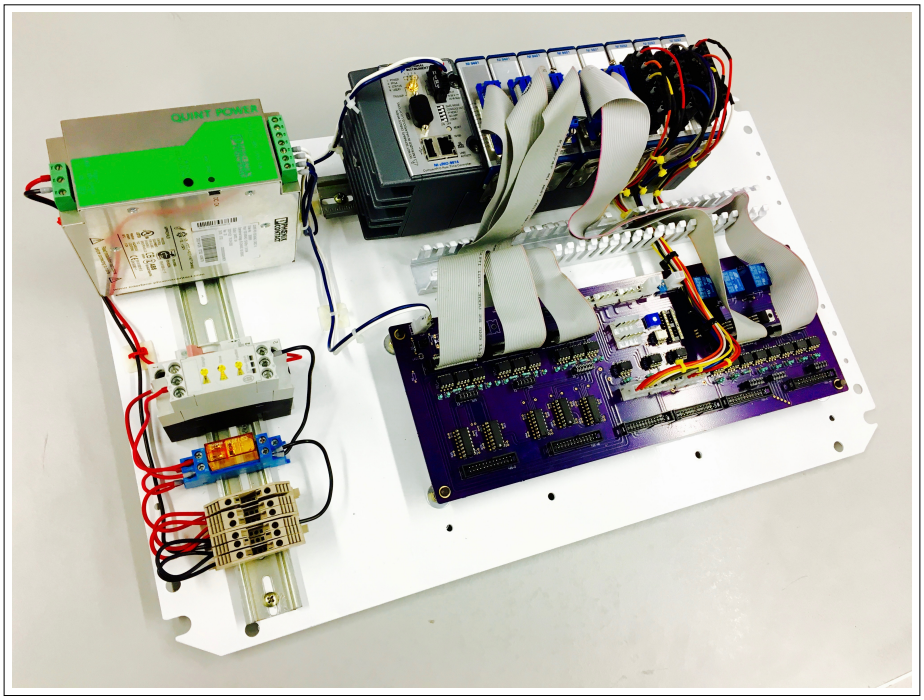


Figura 4.28: Circuito interno del gabinete sin conectores externos

Capítulo 5

Aplicación

Para probar el funcionamiento de la tarjeta de adquisición de datos y control que se realizó en esta tesis se propuso realizar el control de posición de ambos robots mediante trayectoria suave que lleven a cada robot a su posición de *HOME*.

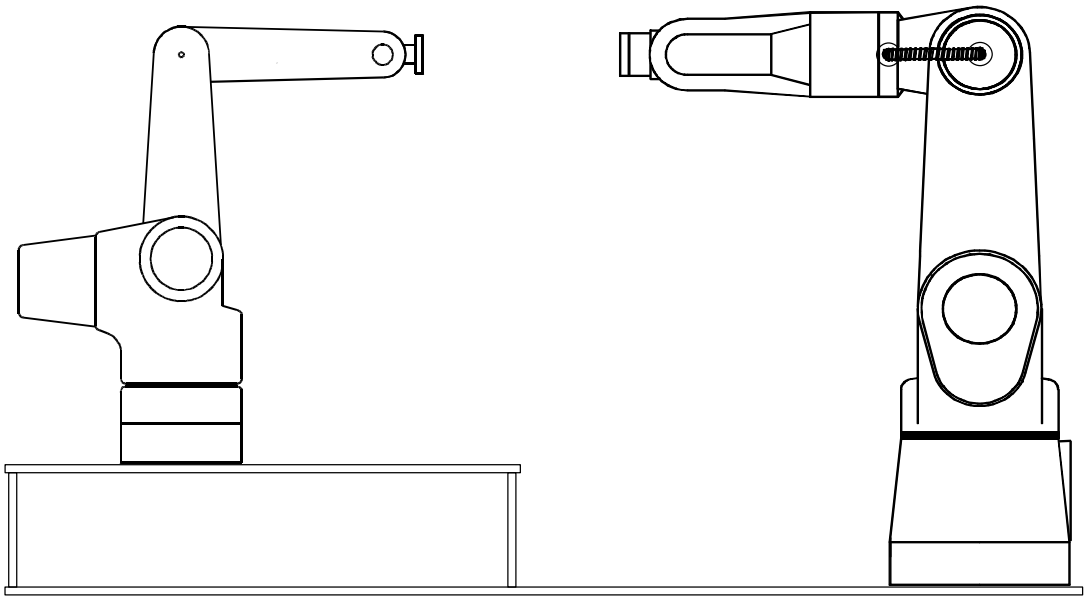


Figura 5.1: Posición de *HOME* de los robots.

5.1 | Trayectorias deseadas

En la robótica la posición de *HOME* es el estado inicial en el que se encuentra el robot antes de realizar alguna tarea. Para los robots A465 y A255, cuyo destino es trabajar conjuntamente, la posición de *HOME* es la mostrada en la Figura 5.1, donde se encuentran frente a frente para la fácil manipulación de objetos entre ellos.

Para el diseño de la trayectoria se tomó en cuenta que el robot parte de un estado en reposo q_0 y se desplazará a un estado deseado en el tiempo t_f.

$$\begin{aligned} q(t_0) &= q_0 & \qquad q(t_f) &= q_f \\ \dot{q}(t_0) &= v_0 & \qquad \dot{q}(t_f) &= v_f \\ \ddot{q}(t_0) &= \alpha_0 & \qquad \ddot{q}(t_f) &= \alpha_f \end{aligned} \tag{5.1}$$

Debido a que la posición de *HOME* se utiliza para el inicio de procesos, la posición inicial es aquella posición en la que se encuentre antes de iniciar el control, cualquiera que ésta sea.

Diseñar la trayectoria mediante un polinomio es una de las formas más comunes y como resultado se obtiene una trayectoria suave y continua. Este polinomio se construye partiendo de la necesidad de cumplir la condiciones (5.1) asignando una constante por cada condición.

$$\begin{aligned} q(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \\ \dot{q}(t) &= a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \\ \ddot{q}(t) &= 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \end{aligned} \tag{5.2}$$

Al cumplir la ecuación con las condiciones iniciales y finales se construye un sistema de seis ecuaciones de donde surge la ecuación (5.3)[30].

$$q(t) = q_i + 10(q_f - q_i)t^3/t_f^3 - 15(q_f - q_i)t^4/t_f^4 + 6(q_f - q_i)t^5/t_f^5 \tag{5.3}$$

El resultado es una trayectoria suave y al igual que su primera y segunda derivada, es continua en el tiempo. En la Figura 5.2 se muestra el ejemplo de la trayectoria deseada para la articulación uno del robot A255. Cabe señalar que el comportamiento es idéntico para todas las articulaciones debido al uso del mismo polinomio.

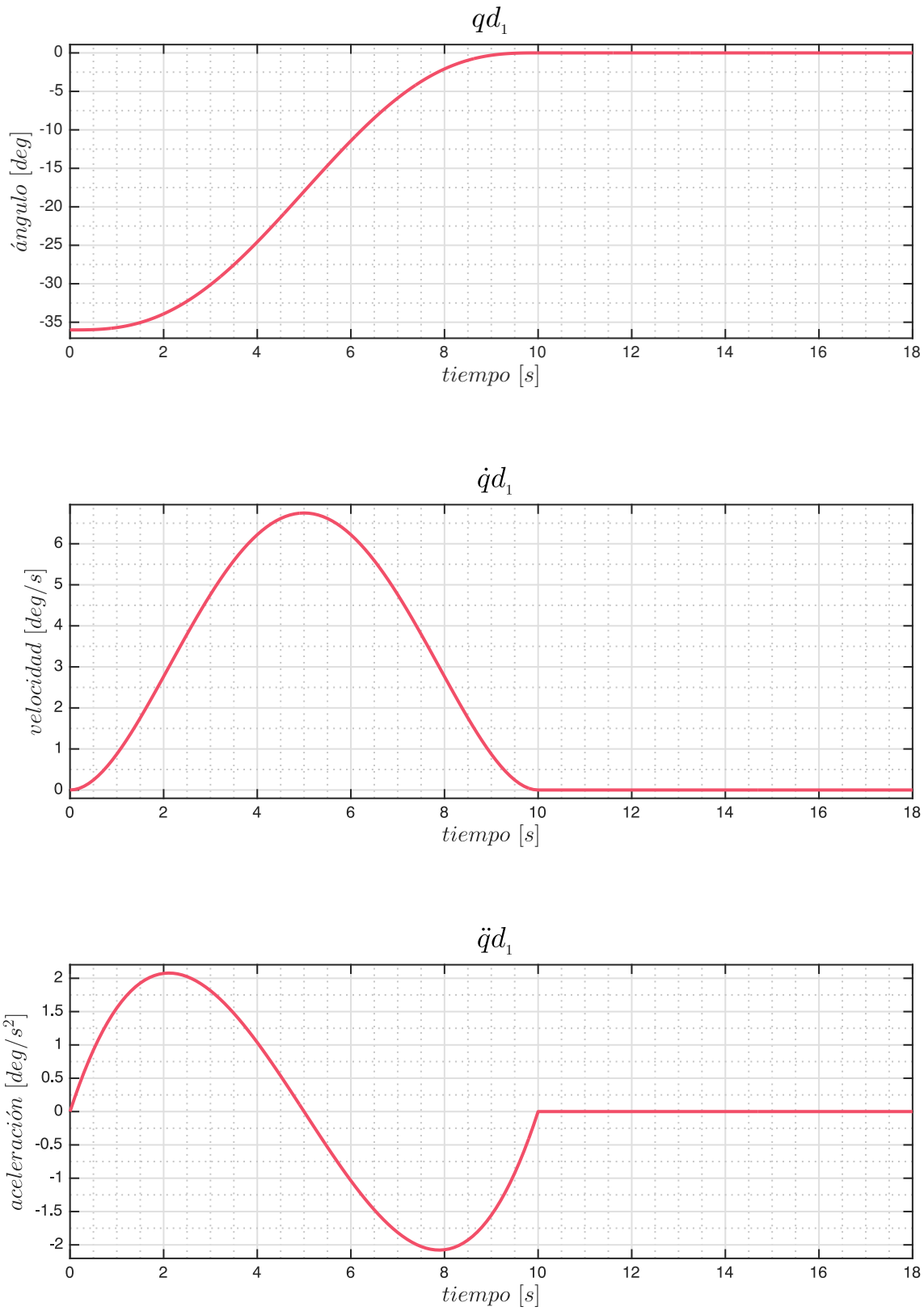


Figura 5.2: Trayectoria deseada de la primera articulación del robot A255 y sus derivadas

5.2 | Control *PID*

En la industria es altamente recurrido el uso de controladores *PID* (Proporcional, Integral y Derivativo) a pesar de los distintos métodos de control que existen en la actualidad. Esto debido al gran desempeño que éstos tienen y facilidad para ser implementados en diversos procesos industriales. Es por ello que se decidió implementar un controlador *PID* en esta tesis para demostrar la eficacia de la tarjeta de adquisición de datos y control sobre el sistema.

La ley de control que define al controlador *PID* es la siguiente:

$$\tau \;=\; -K_p\widetilde{q} \;-\; K_d\frac{d}{dt}\widetilde{q} \;-\; K_i\int_0^t \widetilde{q}dt \tag{5.4}$$

Donde:

- $\widetilde{q}$ es el error entre la variable deseada y la actual.
- K_p ganancia proporcional
- K_i ganancia integral
- K_d ganancia derivativa

La siguiente figura muestra el diagrama a bloques de un sistema con controlador *PID*.

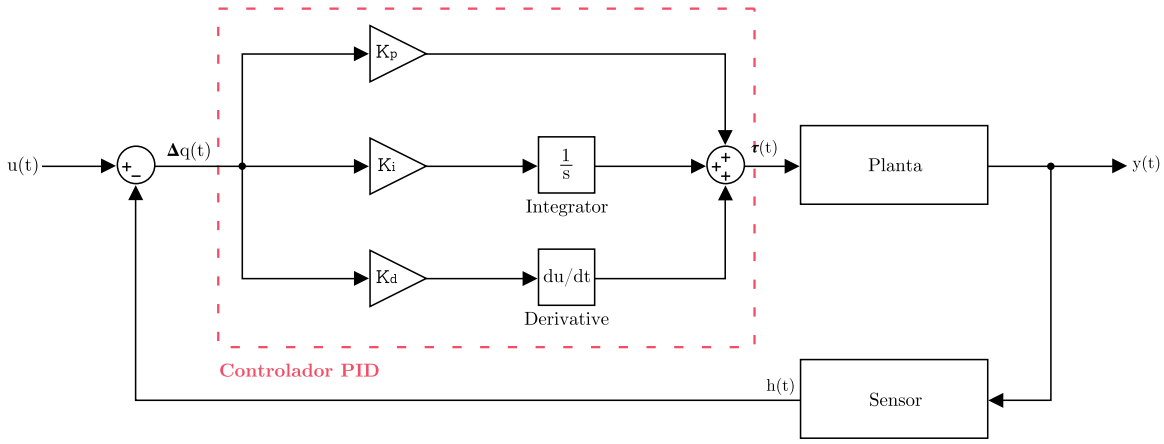


Figura 5.3: Diagrama a bloques de un sistema con controlador PID.

Debido a que la planta es no lineal, los métodos de sintonización de Ziegler y Nichols (en lazo cerrado y abierto), Cohen y Coon o Tyreus y Luyben no aplican para el sistema, por lo que la sintonización del controlador *PID* se realizó mediante el método heurístico.

5.2.1 | Adquisición de Datos y Escritura de Señales Analógicas

Antes de realizar el control y la interfaz de usuario fue necesario interpretar las señales del los codificadores mediante el sistema embebido FPGA CompactRIO, además de generar con ésta la señal de voltaje que controla el movimiento de los motores de los robots. Para para poder programarlo fue requerido el software de desarrollo de sistemas *NI LabVIEW* 2009 el cual utiliza una sintaxis de programación gráfica. En la Sección D.1 se describe el proceso que se debe realizar para crear un proyecto vacío en dicho software.

Una vez creado el proyecto, se generaron archivos con la extensión *vi*, este tipo de extensión se asigna al archivo generado por *LabVIEW* y significa *Virtual Instrument*. Un atributo que poseen dichos archivos es la capacidad de ser invocados dentro de otro *vi* convirtiendo a éste en una función que puede ser usado multiples veces dentro de un mismo programa.

Debido a que el proceso para las once articulaciones es similar, se identificaron tres acciones que se repetían en cada articulación, por lo que se introdujeron en distintos *sub-vi* para simplificar el programa principal:

	Archivo	Descripción
	`counter_increase.vi`	Identifica el sentido de giro del motor y realiza la cuenta en sentido horario.
	`counter_decrease.vi`	Identifica el sentido de giro del motor y realiza la cuenta en sentido antihorario.
	`count_z.vi`	Realiza la cuenta de pulsos en Z y su incremento o decremento depende de un valor de referencia.
	`limits.vi`	Restringe el valor ingresado y realiza la conversión de mV a V.

Tabla 5.1: Descripción de Sub-VIs.

Considerando los *Sub-VIs* de la Tabla 5.1, el proceso para la interpretación de las señales en cuadratura A y B, y la generación de la señal analógica para el control de un motor se ha reducido al diagrama de

flujo mostrado en la Figura 5.6. El programa base realizado a partir de dicho diagrama se presenta en la Figura 5.7 donde todos los parámetros son variables para poder ser ajustados desde la interfaz de usuario.

Para el programa donde se consideran las articulaciones de ambos robots se replico once veces el programa base en un nuevo archivo, compartiendo un sólo ciclo *Do-While*. Además se añadió el fragmento de la Figura 5.4 para realizar el control de la botonera mediante software, en donde se convierte el voltaje de una representación de mV a V.

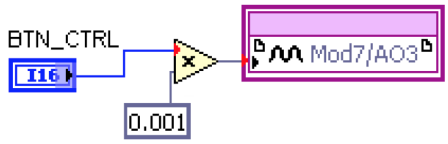


Figura 5.4: Fragmento del programa para controlar la botonera.

De igual forma, se agregó un bloque *Tick Count* que permite conocer el tiempo acumulado desde que comenzó a correr el programa. Al restar este valor con el valor anterior se determina el tiempo que dura la muestra. Esto nos permite conocer la frecuencia de muestreo.

Al ejecutar el programa, el tiempo promedio de cada muestra fue de $5[\mu s]$, esto representa una frecuencia de muestreo de $200,000[mustras/s]$. Con tal resultado se decidió considerar al sistema en tiempo continuo.

El diagrama de flujo de los *Sub-VIs* y su código se encuentra en el Apéndice D.2 al igual que el programa completo para las articulaciones de ambos robots.

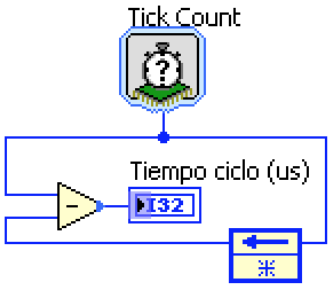


Figura 5.5: Fragmento del programa para determinar la frecuencia de muestreo.

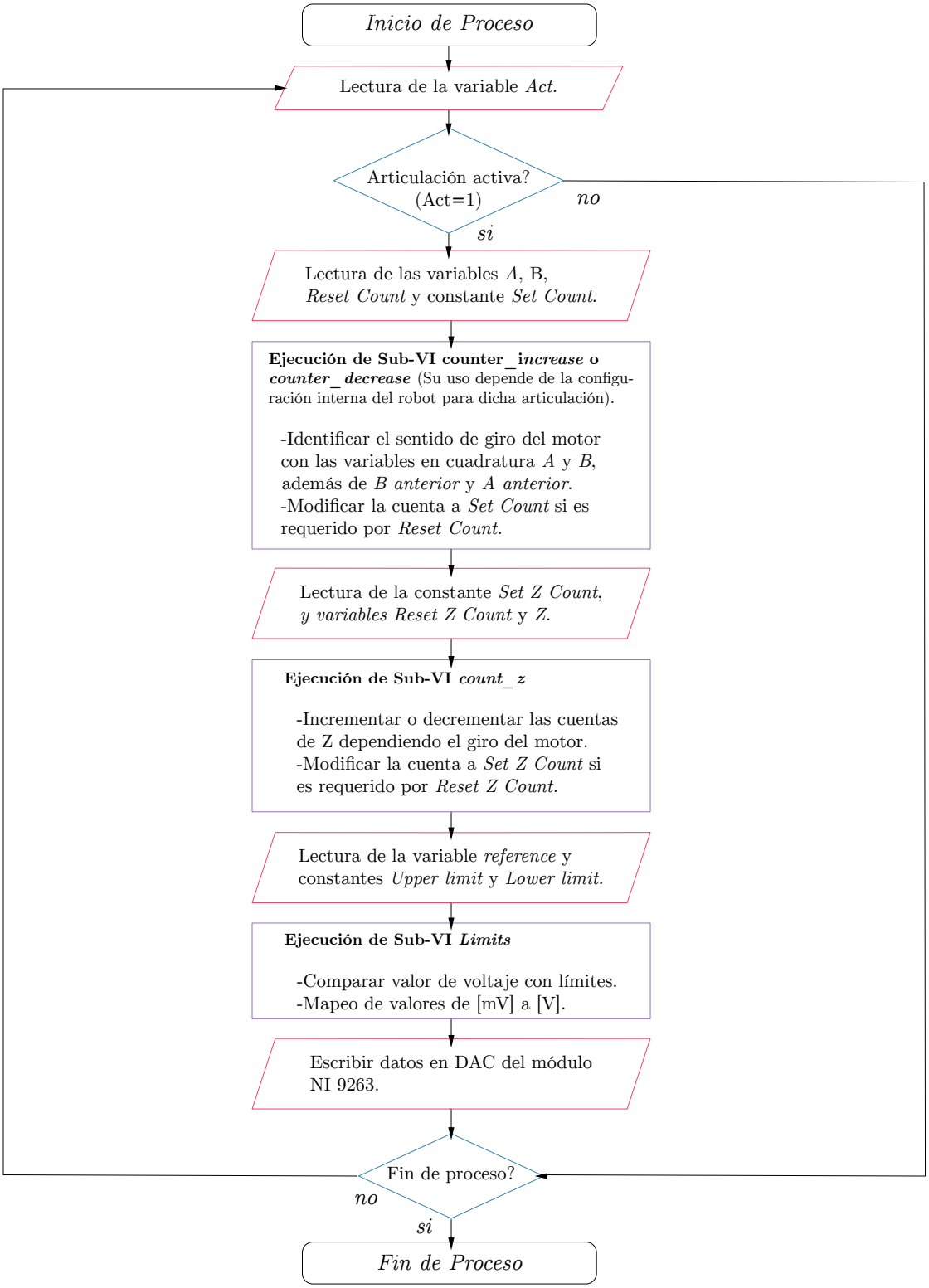


Figura 5.6: Diagrama de flujo del proceso de interpretación de datos y escritura de señales analógicas para una articulación.

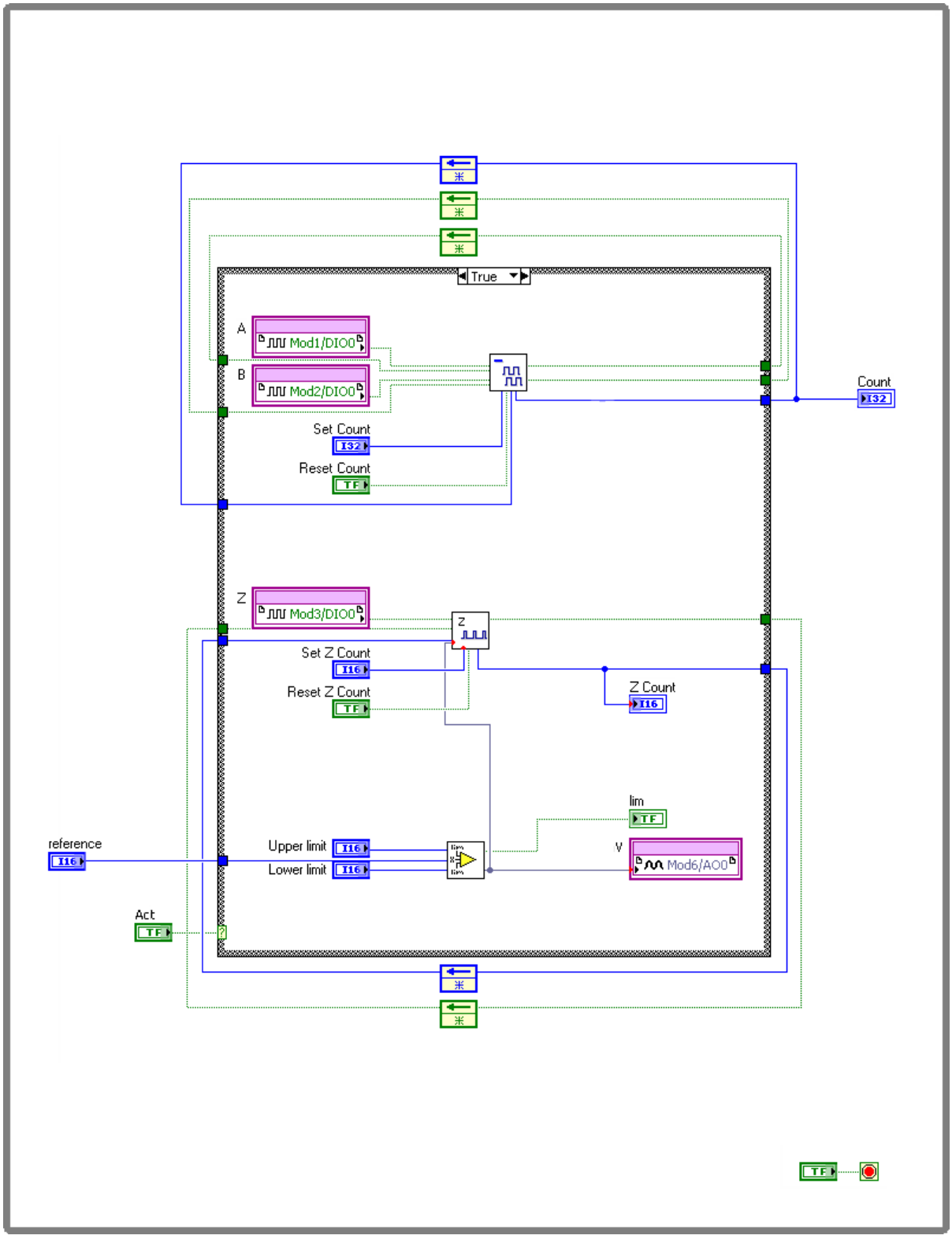


Figura 5.7: Programa base para la interpretación de datos y escritura de señales analógicas para una articulación.

5.2.2 | Interfaz de usuario

La interfaz de usuario se realizó en el ambiente de programación integrado *Visual Studio* 2008 para el sistema operativo *Windows XP*. En la Sección D.3 se describe el procedimiento para la creación del proyecto, así como los requerimientos que éste necesita para establecer comunicación con el FPGA *CompactRIO*.

Una vez que el proyecto en *Visual Studio* fue creado bajo el nombre de *Robots* y los archivos para entablar comunicación con el FPGA fueron agregados, se definieron las funciones siguientes:

- `Comunication()`: Aquí se establece la comunicación con el FPGA y se verifica si existe algun error para completar el enlace.
- `EneableJoints_A255()`: Tiene como parámetros cinco valores booleanos, uno para cada articulación del robot A255, donde un uno lógico activa a la articulación del parámetro correspondiente. Esta función designa el valor ingresado a la variable *Act* del programa en *LabVIEW* como el que se muestra en la Figura 5.7. De esta forma se habilita la lectura de datos y el envío de la señal analógica. En esta función también se establecen los límites de voltaje.
- `EneableJoints_A465()`: Esta función activa las articulaciones del robot A465. Tiene como parámetros seis valores booleanos y establece los límites de voltaje para cada articulación.
- `GetEncoders_A255()`: Con el objetivo de conocer la posición del robot A255. Esta función lee el valor del indicador numérico *Cuentas* del programa en *LabVIEW* y realiza su conversión a radianes. Los coeficientes para realizar dicha conversión fueron obtenidos con la relación de reducción de engranes encontrada en la hoja de datos del robot.
- `GetEncoders_A465()`: De forma análoga, esta función determina el valor en radianes mediante la lectura de cuentas del robot A465.
- `SetVoltage_A255()`: Convierte el valor de voltaje en todas las articulaciones del robot A255 a su representación en mV y es enviado al programa en el FPGA para realizar su conversión de valores digitales a analógicos.
- `SetVoltage_A465()`: Convierte el valor de voltaje en todas las articulaciones del robot A465 a su representación en mV y es enviado al programa en el FPGA para realizar su conversión de valores digitales a analógicos.
- `OnInitDialog()`: Esta función es generada a la hora de crear el proyecto y es un lugar correcto para realizar el inicio de la comunicación con el FPGA y activar las articulaciones a utilizar. Aquí se puede gestionar cualquier otro tipo de inicializaciones adicionales.

Para permitir al usuario manipular los robot mediante la activación de botones, se construyó una interfaz de usuario gráfica (GUI). Esto se logró relacionando funciones a elementos gráficos booleanos. Las acciones básicas a ejecutar por las funciones mencionadas se describen a continuación.

- `OnBnClickedBtn_Get_q()`: Despliega en pantalla la posición actual de los robots en grados.
- `OnBnClickedBtn_Set_v()`: Envía los valores de voltaje asignados por el usuario al FPGA para su conversión digital a analógico.
- `OnBnClickedBtn_ESTOP()`: Botón de paro de emergencia mediante software.
- `OnBnClickedBtn_EACT()`: Botón de accionamiento mecánico de los robots mediante software.
- `OnBnClickedCancel()`: Función para detener al programa.
- `OnBnClickedBtn_Home()`: Ejecuta el algoritmo de control visto en esta tesis para llevar al robot a su posición de *HOME*.

El código con las funciones descritas en esta sección se encuentran en la Sección D.4. El acabado de la interfaz de usuario se muestra en la Figura 5.8.

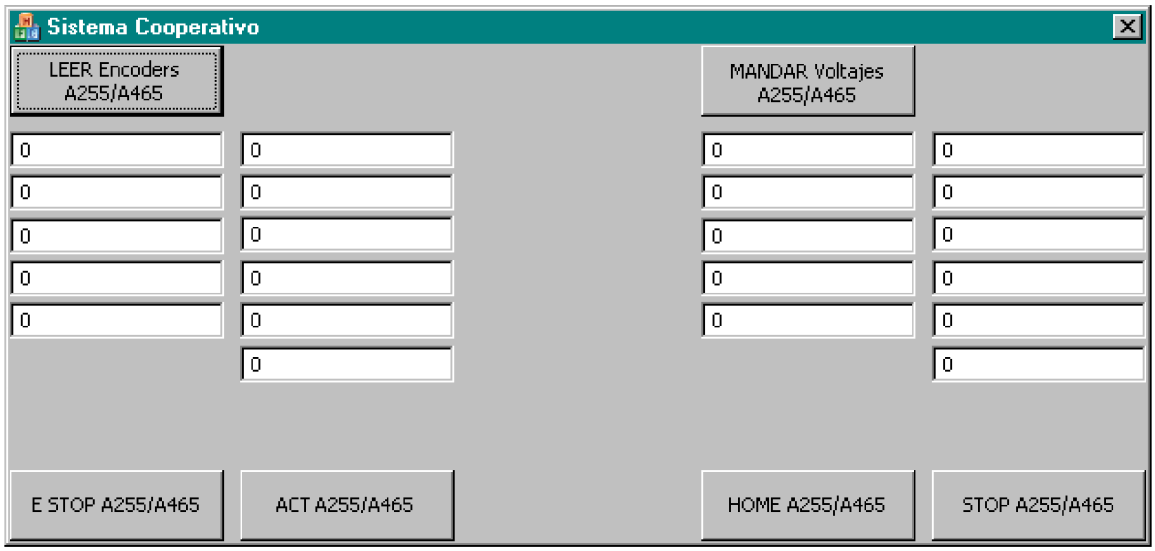


Figura 5.8: Interfaz de usuario gráfica.

5.2.3 | Algoritmo de Control

A continuación se describe el código del controlador *PID* utilizado para llevar la a posición de *HOME* a los robots A465 y A255. Esta función entra en acción al ser accionado el interruptor de "*HOME*" en la interfaz de usuario.

```
// Función para llevar a home a los robots
void CALLBACK CRobotsDlg::ControlHome( UINT uTimeID, UINT uMsg, DWORD dwUser,
                                       DWORD dw1, DWORD dw2 )
{

        GetEncoders_A255();     /// Actualiza el valor de
        GetEncoders_A465();     /// q_A255 y q_A465.


        // Punto inicial y final de la trayectoria
        const double  qf_A255[5] = {0.0, pi/2.0, -pi/2.0, pi/2.0, 0.0};
        const double  qf_A465[6] = {0.0, pi/2.0, 0.00000, 0.0000, 0.00, 0.0};

        static double qi_A255[5], qi_A465[6];    /// q inicial

        static double qa_A255[5], qa_A465[6];    /// q aterior


        // Velocidad articular
        double qp_A255[5]  = {0.0};
        double qp_A465[6]  = {0.0};


        // Trayectoria deseada
        double qd_A255[5]  = {0.0};
        double qd_A465[6]  = {0.0};

        double qpd_A255[5] = {0.0};
        double qpd_A465[6] = {0.0};


        // Constantes del polinomio deseado
        double a0_A255[5], a3_A255[5], a4_A255[5], a5_A255[5];
        double a0_A465[6], a3_A465[6], a4_A465[6], a5_A465[6];


        // Tiempo final
        const double tf = 10.0;


        // Errores
        double Deltaq_A255[5], Deltaqp_A255[5];
        double Deltaq_A465[6], Deltaqp_A465[6];

        static double sigma_A255[5];
        static double sigma_A465[6];


        // Par de salida
        double tau_A255[6] = {0.0};
        double tau_A465[6] = {0.0};


        // Ganancias del control PID
        const double kp_A255[5] = {80.0, 110.0, 112.0, 20.0, 38.0};
        const double kv_A255[5] = {8.00, 8.090, 8.500, 2.50, 1.00};
        const double ki_A255[5] = {4.00, 3.236, 5.200, 1.00, 0.10};
        const double kp_A465[6] = {20.0, 132.0, 132.0, 62.0, 102.0, 70.0};
        const double kv_A465[6] = {4.20, 10.20, 10.20, 5.00, 5.000, 4.50};
        const double ki_A465[6] = {1.00, 3.000, 3.000, 2.00, 3.500, 1.50};

        CRobotsDlg* pMainWnd = (CRobotsDlg *)AfxGetApp()->m_pMainWnd;
```

```
// Condiciones iniciales
if(HomeInicio){ ///Bandera global booleana
        for(int i=0;i<5;i++){
                qi_A255[i]    = q_A255[i];
                qa_A255[i]    = q_A255[i];
                sigma_A255[i] = 0.0;
        }

        for(int i=0;i<6;i++){
                qi_A465[i]    = q_A465[i];
                qa_A465[i]    = q_A465[i];
                sigma_A465[i] = 0.0;
        }
}


// Tiempo
t = (timeGetTime() - ti)/1000.0;
/// ti = timeGetTime() en OnBnClickedBtn_Home()


// Derivada numérica
for(int i=0;i<5;i++)
        qp_A255[i] = (q_A255[i] - qa_A255[i])/T;

for(int i=0;i<6;i++)
        qp_A465[i] = (q_A465[i] - qa_A465[i])/T;


// Trayectoria deseada
if(HomeInicio){
        for(int i=0;i<5;i++){
                a0_A255[i] = qi_A255[i];
                a3_A255[i] =  10.0*(qf_A255[i] - qi_A255[i])/(tf*tf*tf);
                a4_A255[i] = -15.0*(qf_A255[i] - qi_A255[i])/(tf*tf*tf*tf);
                a5_A255[i] =   6.0*(qf_A255[i] - qi_A255[i])/(tf*tf*tf*tf*tf);
        }

        for(int i=0;i<6;i++){
                a0_A465[i] = qi_A465[i];
                a3_A465[i] =  10.0*(qf_A465[i] - qi_A465[i])/(tf*tf*tf);
                a4_A465[i] = -15.0*(qf_A465[i] - qi_A465[i])/(tf*tf*tf*tf);
                a5_A465[i] =   6.0*(qf_A465[i] - qi_A465[i])/(tf*tf*tf*tf*tf);
        }

        HomeInicio = false;
}

if(t<=tf){
        for(int i=0;i<5;i++){
                qd_A255[i]  = a0_A255[i] + a3_A255[i]*(t*t*t)
                + a4_A255[i]*(t*t*t*t) + a5_A255[i]*(t*t*t*t*t);
                qpd_A255[i] = 3.0*a3_A255[i]*(t*t) + 4.0*a4_A255[i]*(t*t*t)
                + 5.0*a5_A255[i]*(t*t*t*t);
        }
        for(int i=0;i<6;i++){
                qd_A465[i]  = a0_A465[i] + a3_A465[i]*(t*t*t)
                + a4_A465[i]*(t*t*t*t) + a5_A465[i]*(t*t*t*t*t);
                qpd_A465[i] = 3.0*a3_A465[i]*(t*t) + 4.0*a4_A465[i]*(t*t*t)
                + 5.0*a5_A465[i]*(t*t*t*t);
        }
}
else{
        for(int i=0;i<5;i++){
                qd_A255[i]  = qf_A255[i];
                qpd_A255[i] = 0.0;
        }
        for(int i=0;i<6;i++){
                qd_A465[i]  = qf_A465[i];
                qpd_A465[i] = 0.0;
        }
        if(t<tf+2*T){
                pMainWnd->m_Edit_Status_Cop.SetWindowTextA("␣En␣Home␣");
        }
}
```

```
        // Errores
        for(int i=0;i<5;i++){
                Deltaq_A255[i]  =  q_A255[i] -  qd_A255[i];
                Deltaqp_A255[i] = qp_A255[i] - qpd_A255[i];
        }
        for(int i=0;i<6;i++){
                Deltaq_A465[i]  =  q_A465[i] -  qd_A465[i];
                Deltaqp_A465[i] = qp_A465[i] - qpd_A465[i];
        }


        // Ley de control
        for(int i=0;i<5;i++)
                tau_A255[i] = -kp_A255[i]*Deltaq_A255[i]
                              -kv_A255[i]*Deltaqp_A255[i]
                              -ki_A255[i]*sigma_A255[i];
        for(int i=0;i<6;i++)
                tau_A465[i] = -kp_A465[i]*Deltaq_A465[i]
                              -kv_A465[i]*Deltaqp_A465[i]
                              -ki_A465[i]*sigma_A465[i];


        // Se envían los voltajes a los motores
        SetVoltage_A255( 1.0*tau_A255[0], 1.0*tau_A255[1], 1.0*tau_A255[2],
                         1.0*tau_A255[3], 1.0*tau_A255[4]);
        SetVoltage_A465( 1.0*tau_A465[0], 1.0*tau_A465[1], 1.0*tau_A465[2],
                         1.0*tau_A465[3], 1.0*tau_A465[4], 1.0*tau_A465[5]);


        // Integración numérica
        for(int i=0;i<5;i++)
                sigma_A255[i] += T*Deltaq_A255[i];
        for(int i=0;i<6;i++)
                sigma_A465[i] += T*Deltaq_A465[i];


        // Actualización
        for(int i=0;i<5;i++)
                qa_A255[i] = q_A255[i];
        for(int i=0;i<6;i++)
                qa_A465[i] = q_A465[i];
}
```

Capítulo 6

Resultados Experimentales

En la Sección 6.1 se muestran las gráficas obtenidas durante la ejecución del controlador para cada una de las articulaciones del los robots A255 y A465. El tiempo en el que la trayectoria deseada es variable fue de $10[s]$, a partir de ese momento la trayectoria permanece constante.

Como se puede apreciar en las gráficas de posición contra posición deseada, los robots llegaron a un estado similar al deseado, pero permanecen con un error constante al finalizar el experimento. Este error se mantiene dentro del rango de tolerancia de $\pm 0.5°$ en todas las articulaciones.

El controlador *PID* fue ajustado mediante un método heurístico. En virtud de lo anterios, las ganancias aplicadas al controlador con el que se generaron las gráficas son las siguientes:

$$\begin{aligned} K_{p_{A255}} &= [\quad 80.0\ 110.0\ 112.0\ 20.0\ 38.0\quad] \\ K_{i_{A255}} &= [\quad 8.00\ 8.090\ 8.500\ 2.50\ 1.00\quad] \\ K_{d_{A255}} &= [\quad 4.00\ 3.236\ 5.200\ 1.00\ 0.10\quad] \end{aligned} \tag{6.1}$$

$$\begin{aligned} K_{p_{A465}} &= [\quad 20.0\ 132.0\ 132.0\ 62.0\ 102.0\ 70.0\quad] \\ K_{i_{A465}} &= [\quad 4.20\ 10.20\ 10.20\ 5.00\ 5.000\ 4.50\quad] \\ K_{d_{A465}} &= [\quad 1.00\ 3.000\ 3.000\ 2.00\ 3.500\ 1.50\quad] \end{aligned} \tag{6.2}$$

6.1 | Gráficas del Experimento

En este apartado se muestran los resultados gráficos obtenidos en el experimento. Por cada articulación se exhiben dos gráficas, la primera corresponde a la comparación entre la trayectoria real y la trayectoria deseada, mientras que la segunda corresponde al comportamiento del error.

6.1.1 | Robot A255

Primera Articulación

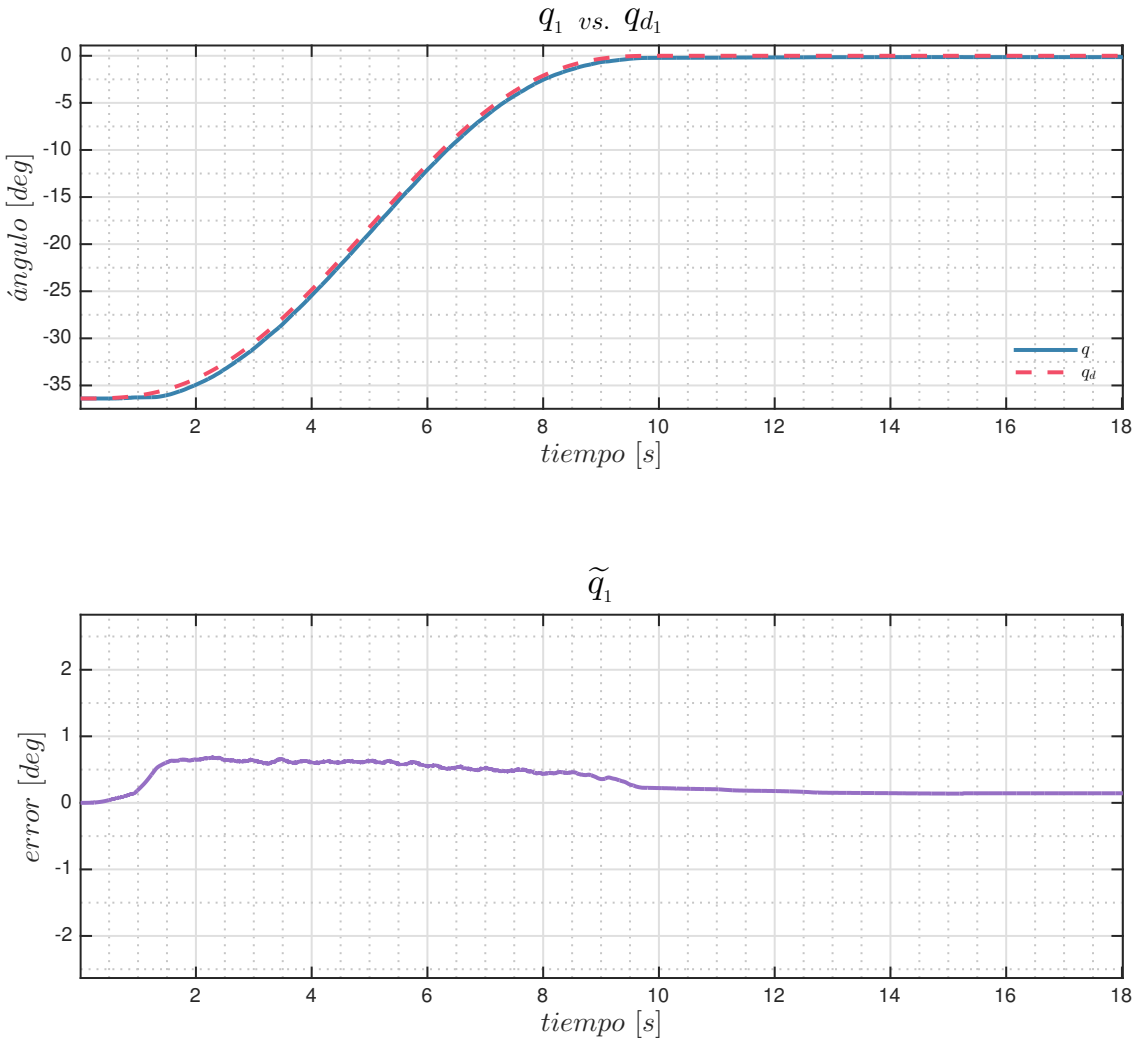


Figura 6.1: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la primera articulación del robot A255.

Segunda Articulación

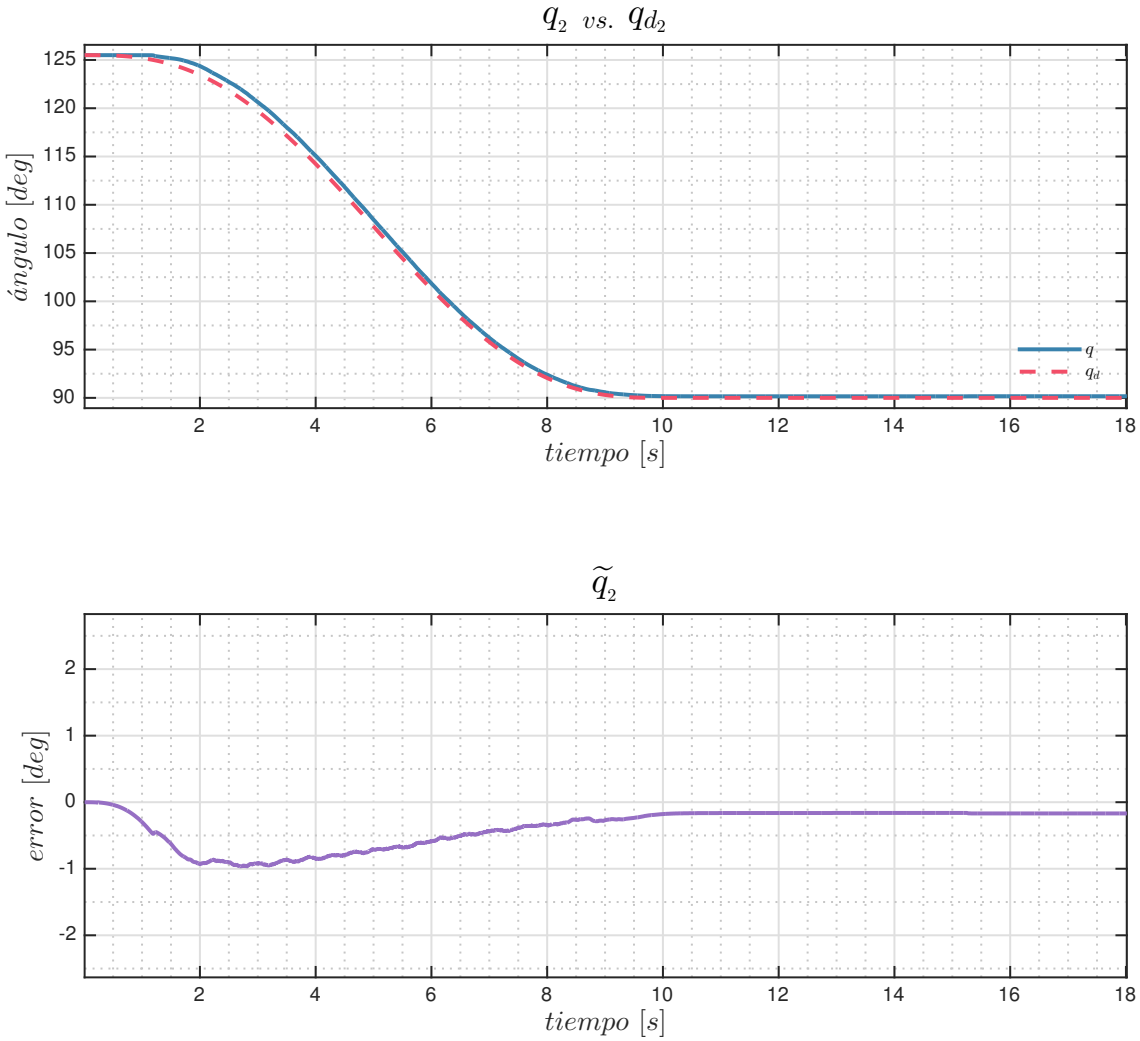


Figura 6.2: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la segunda articulación del robot A255.

Tercera Articulación

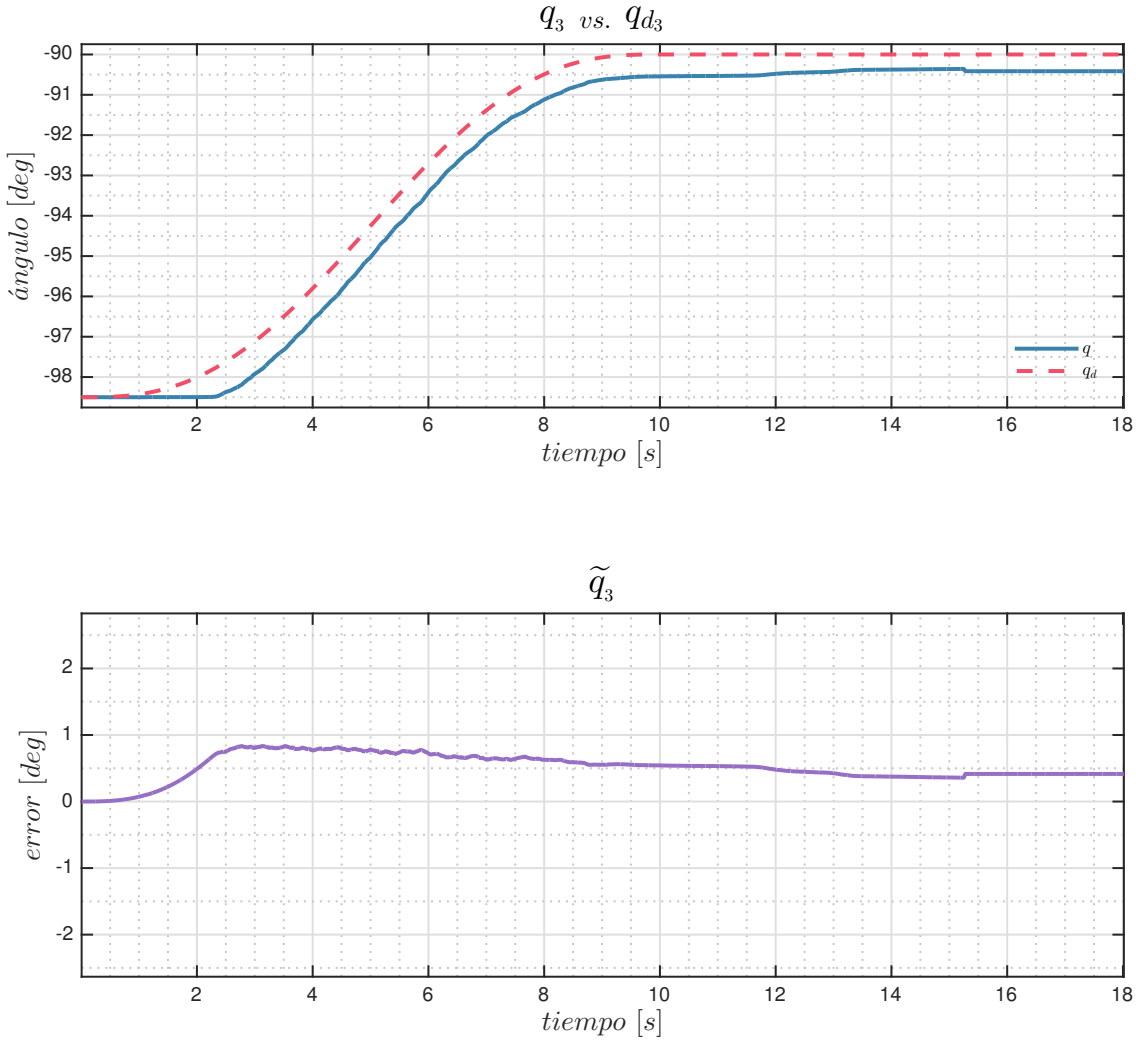


Figura 6.3: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la tercera articulación del robot A255.

Cuarta Articulación

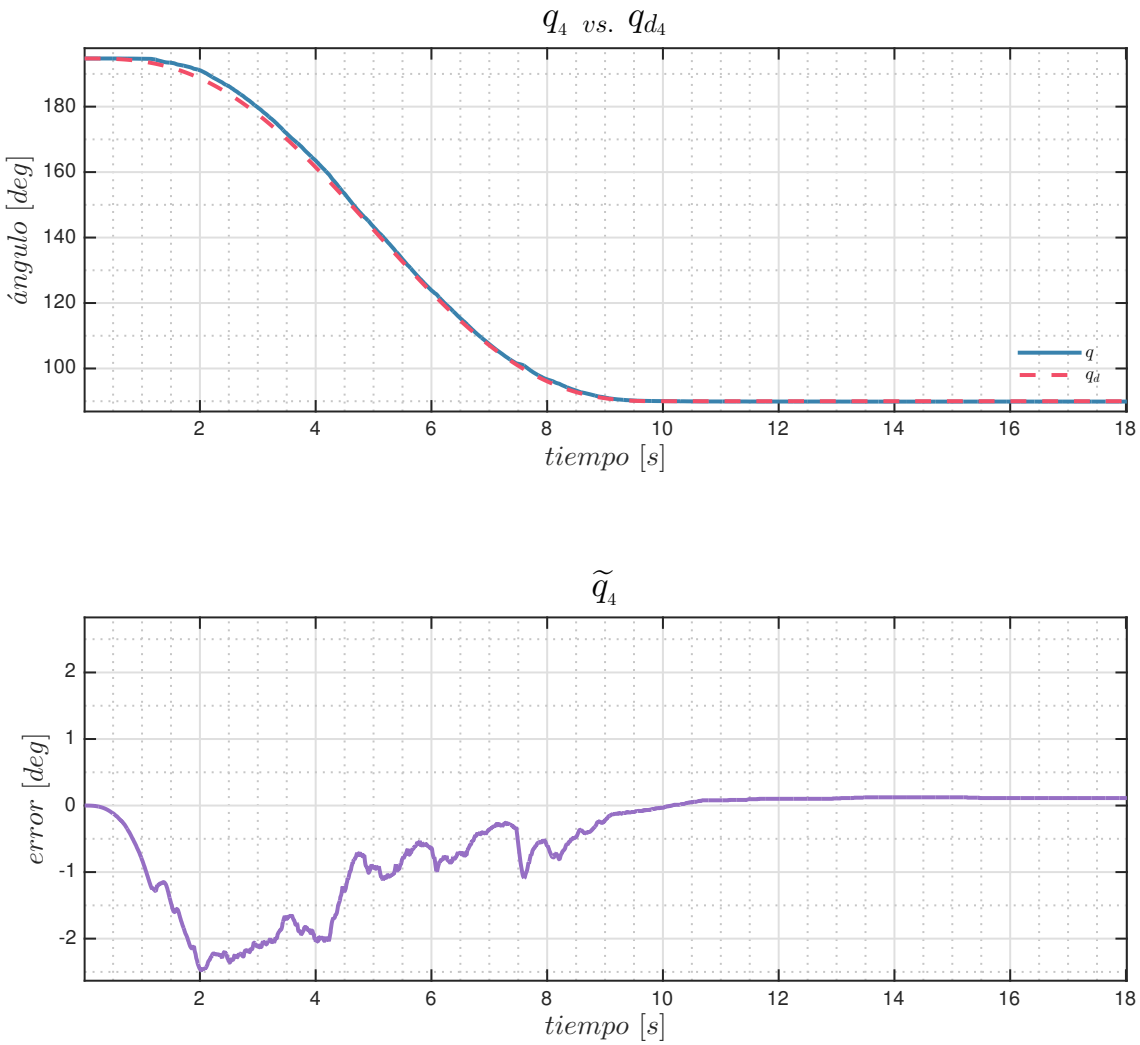


Figura 6.4: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la cuarta articulación del robot A255.

Quinta Articulación

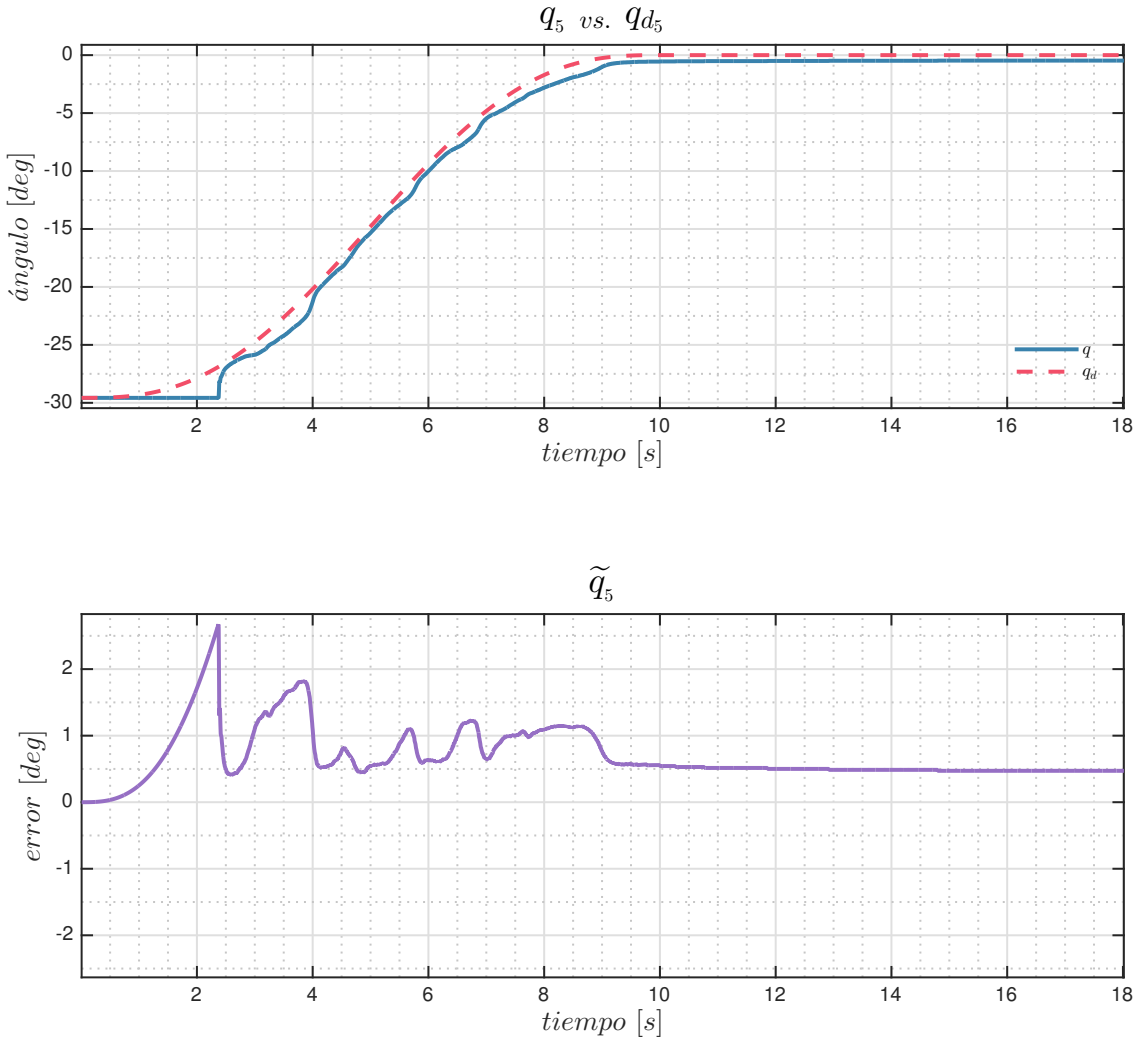


Figura 6.5: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la quinta articulación del robot A255.

6.1.2 | Robot A465

Primera Articulación

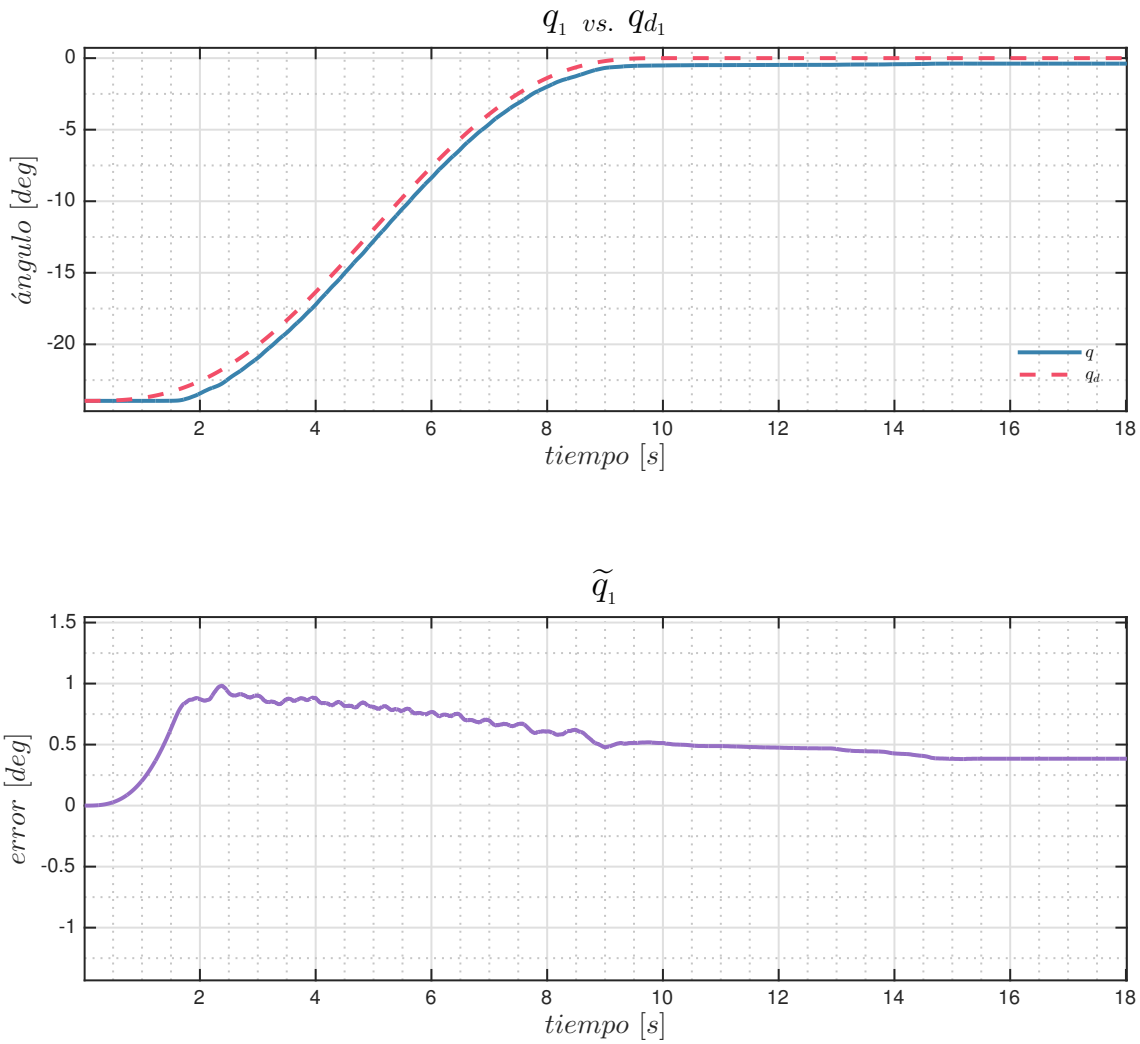


Figura 6.6: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la primera articulación del robot A465.

Segunda Articulación

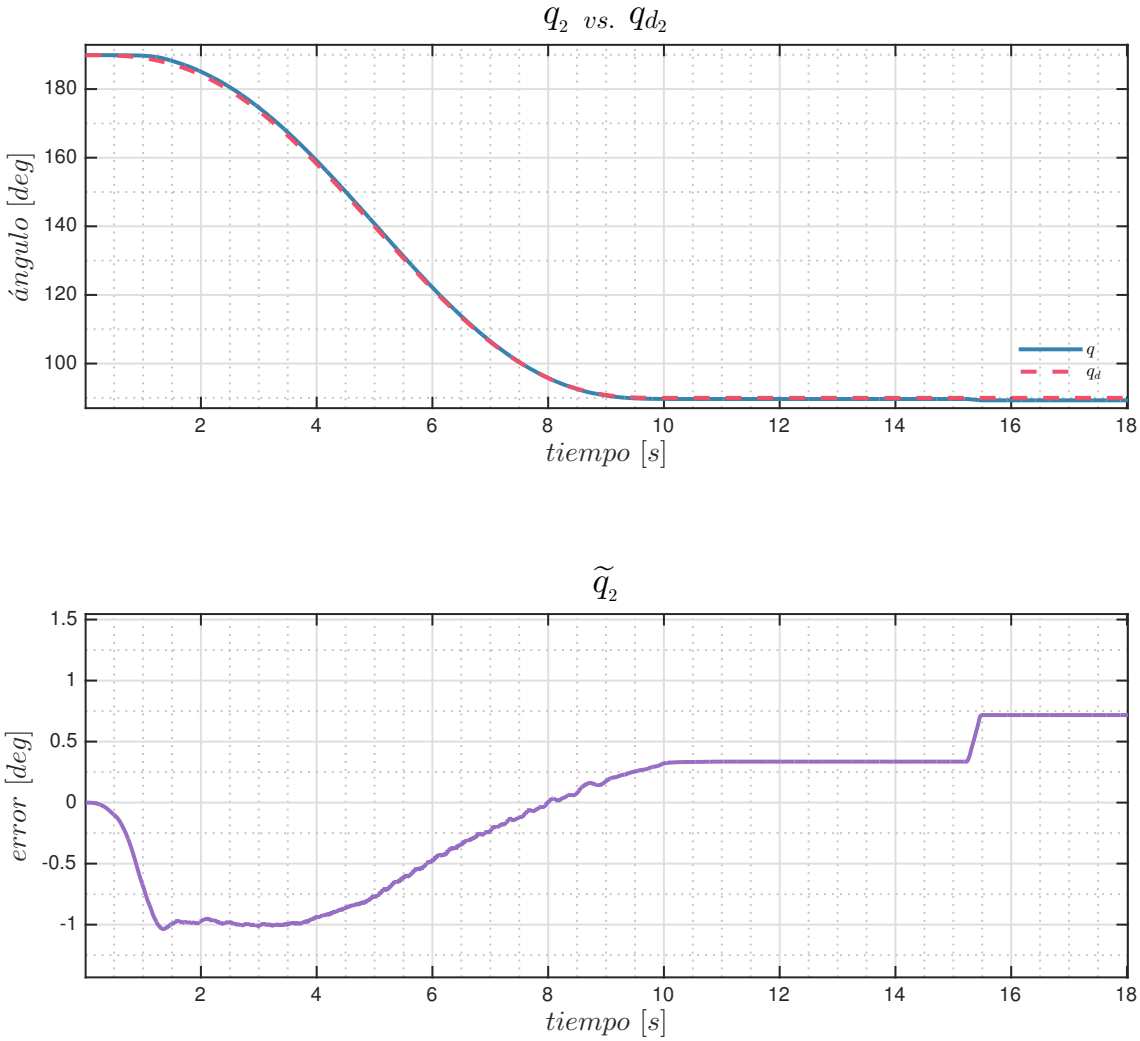


Figura 6.7: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la segunda articulación del robot A465.

Tercera Articulación

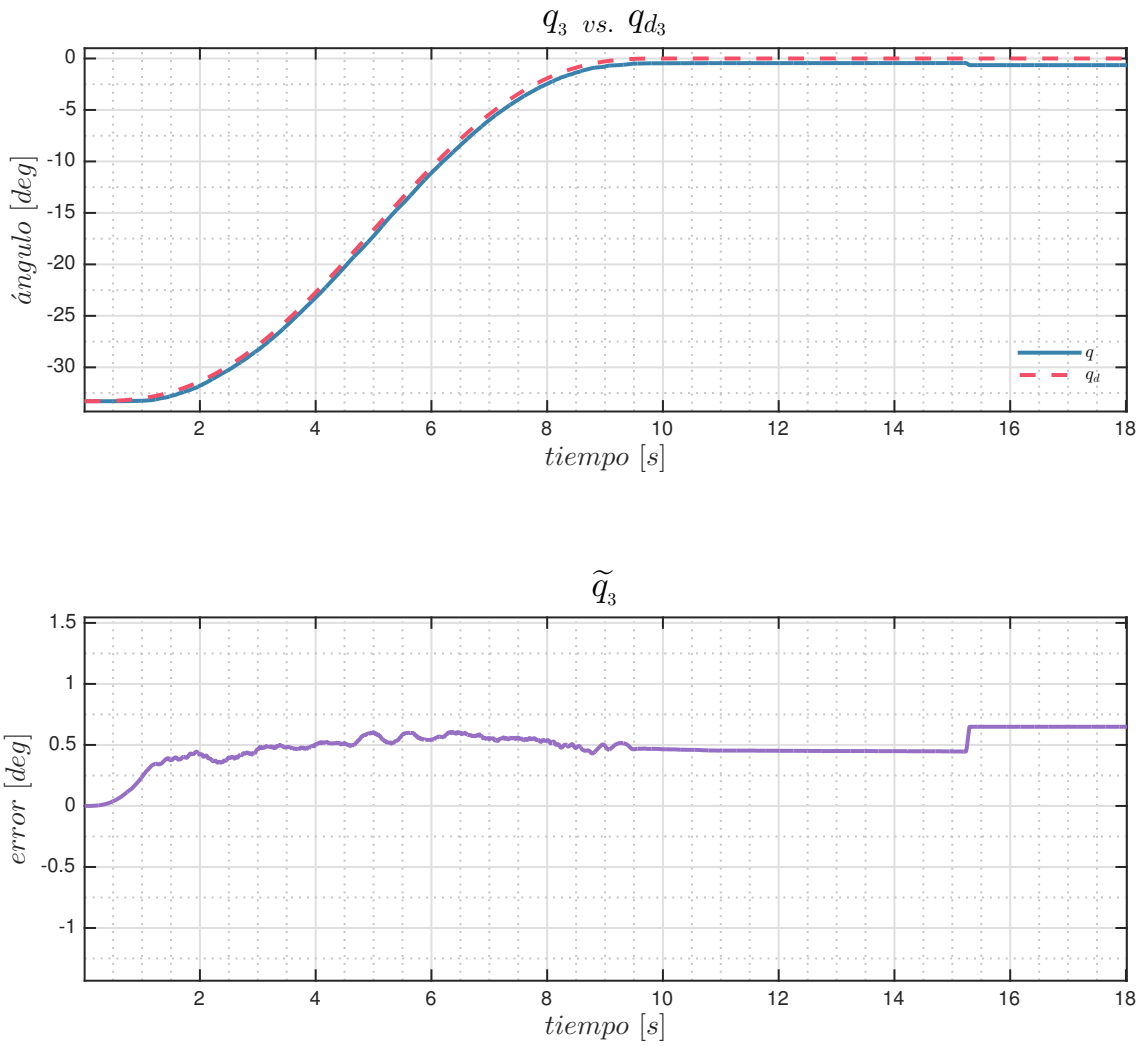


Figura 6.8: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la tercera articulación del robot A465.

Cuarta Articulación

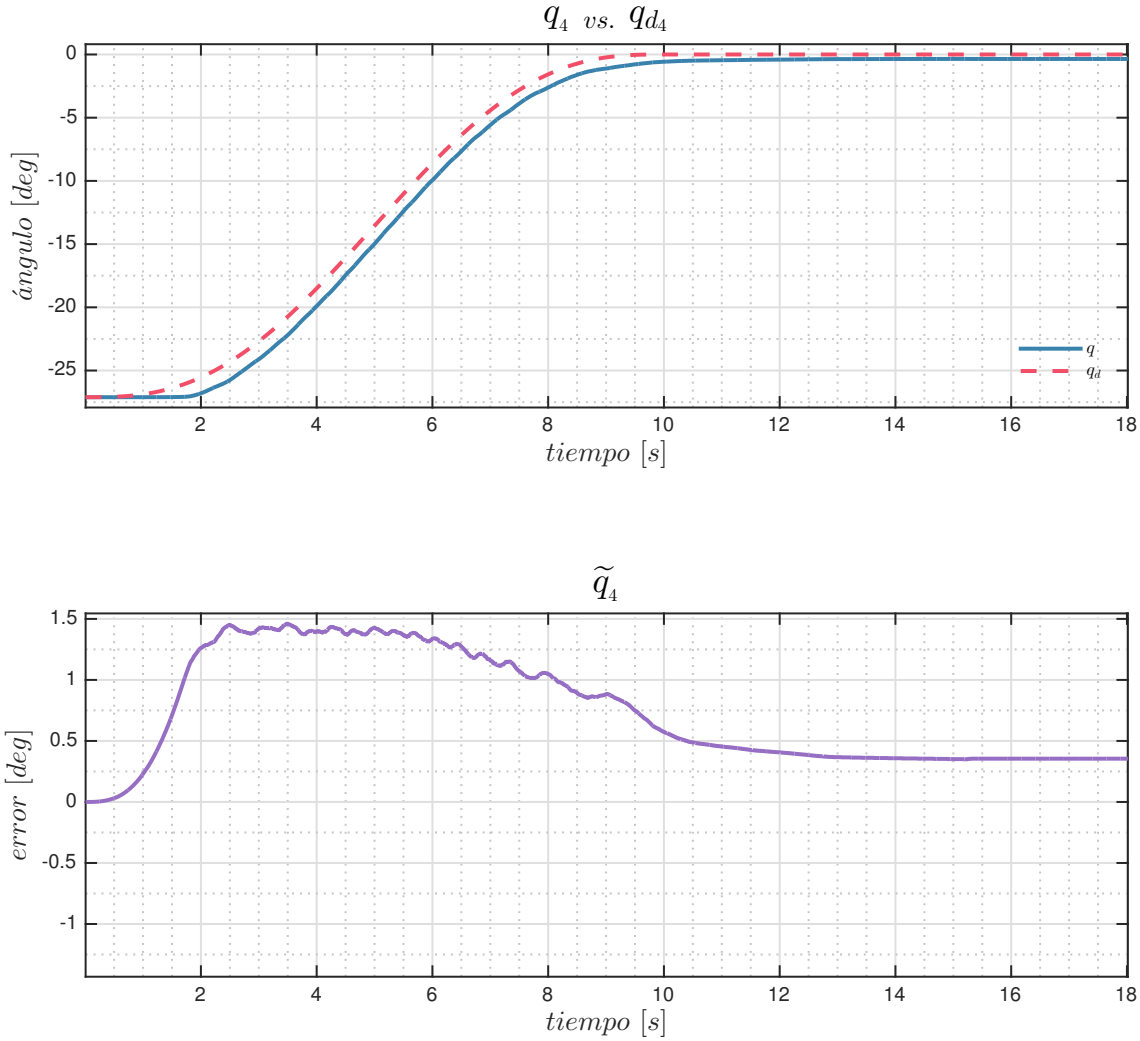


Figura 6.9: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la cuarta articulación del robot A465.

Quinta Articulación

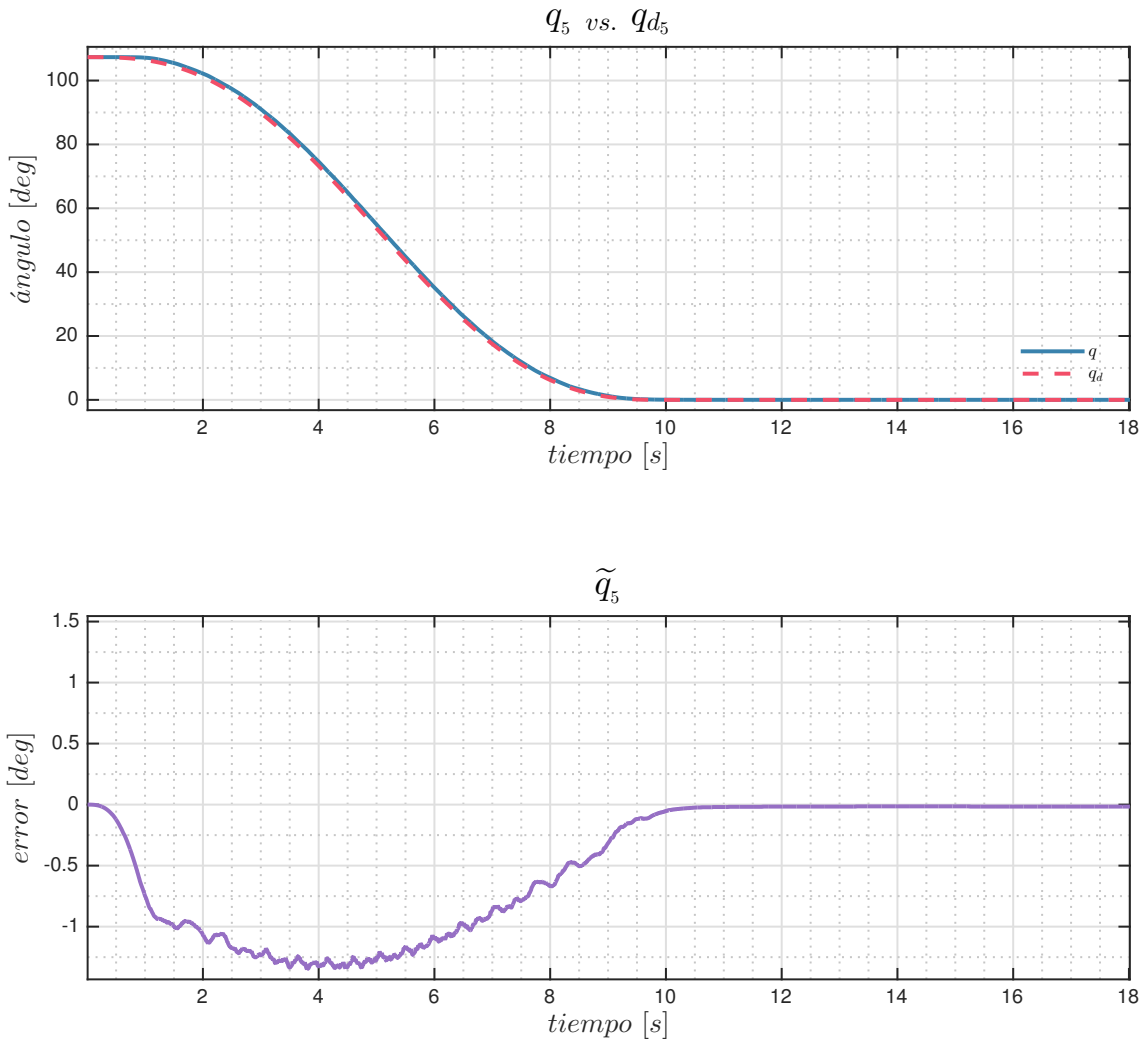


Figura 6.10: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la quinta articulación del robot A465.

Sexta Articulación

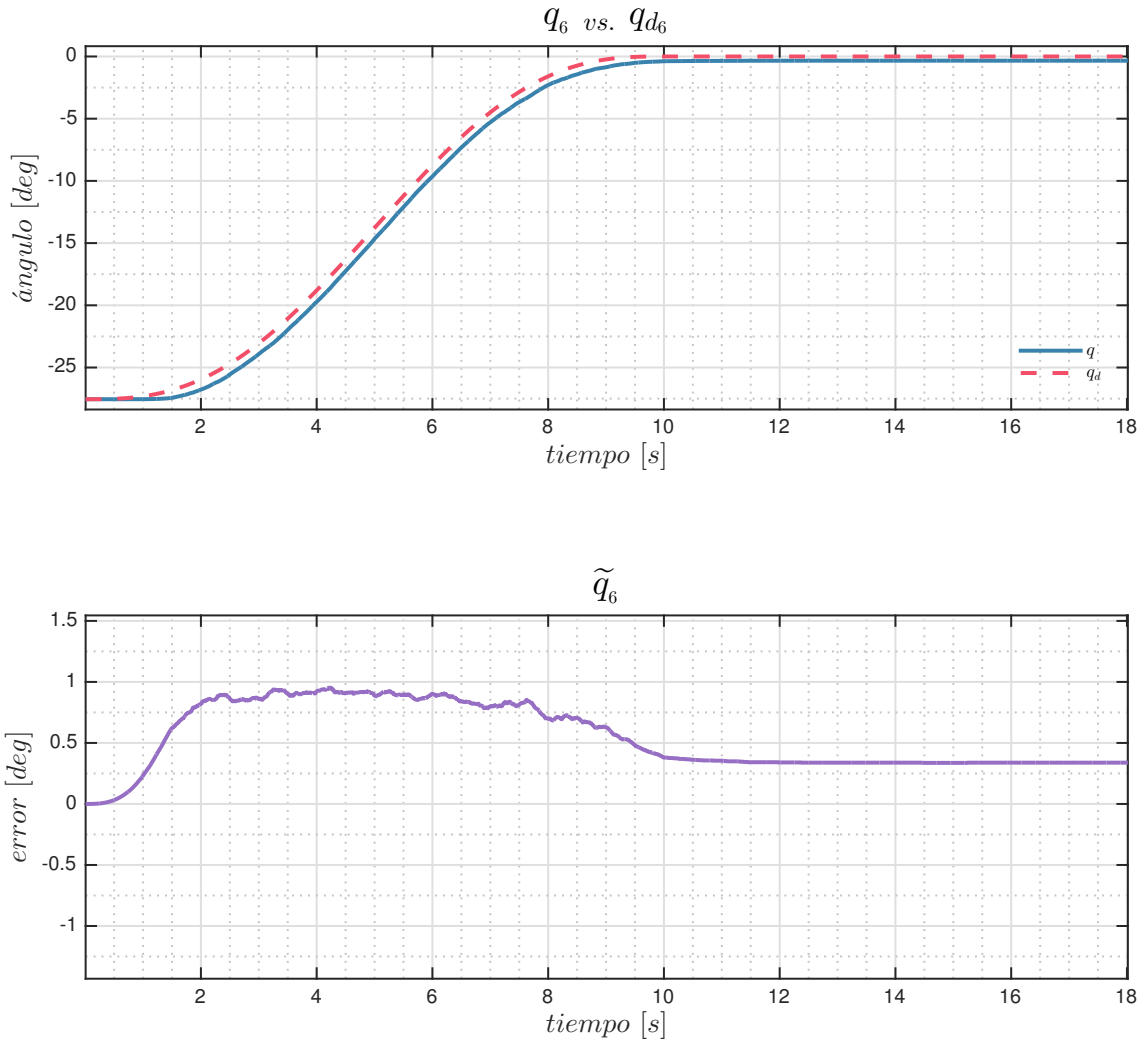


Figura 6.11: Gráfica de la posición q contra la posición deseada q_d y su error $\widetilde{q}$ de la sexta articulación del robot A465.

La distribución final del sistema se muestra en el diagrama de la Figura 6.12, dónde se exponen de manera gráfica los dispositivos principales que interactúan entre si para poder ejecutar el experimento.

Como se vio en la sección 4.7 la tarjeta de adquisición de datos y control de robots cooperativos se encuentra dentro del gabinete junto con los elementos necesarios para que ésta trabaje de forma adecuada.

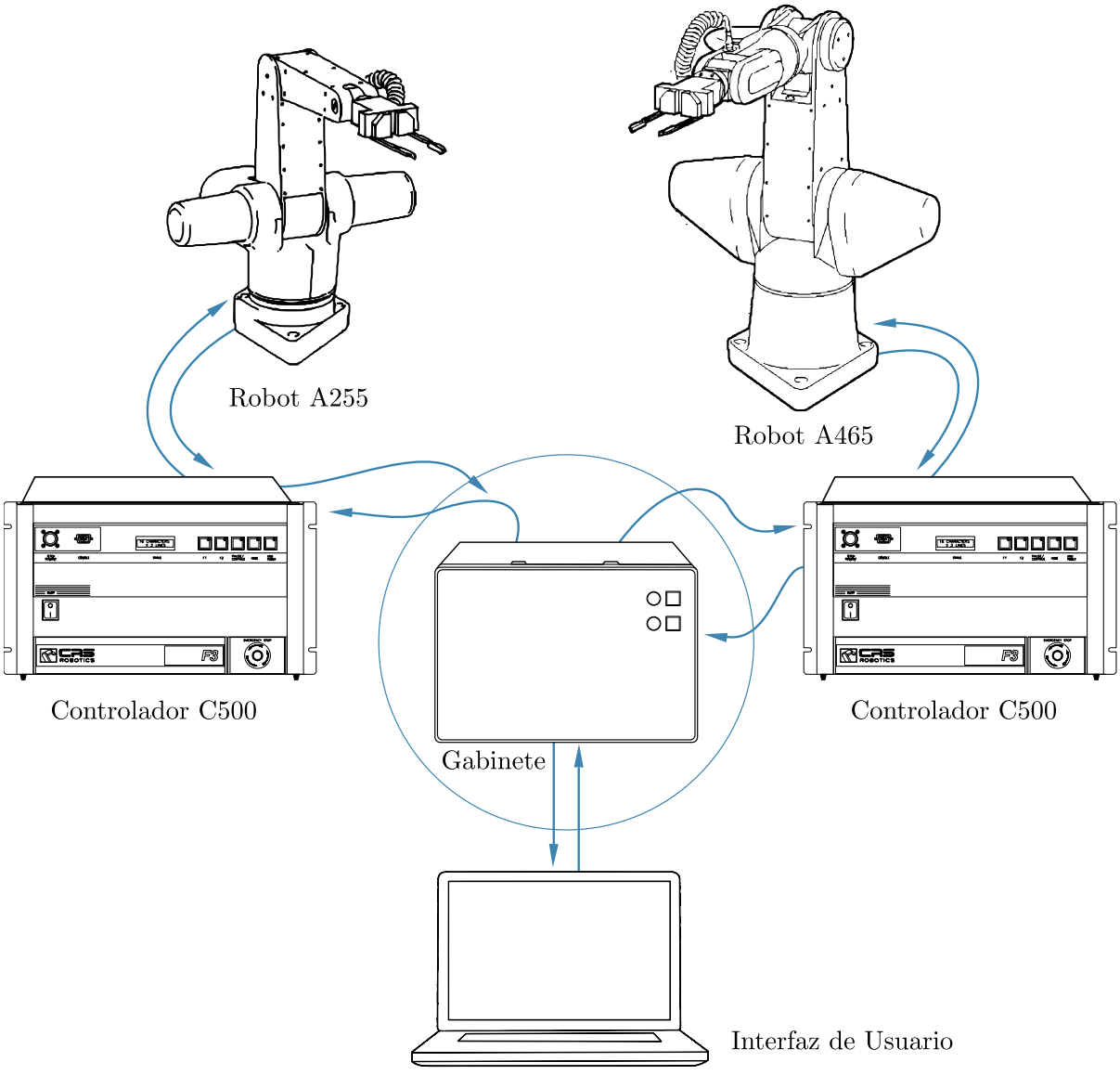


Figura 6.12: Distribución final del sistema.

Capítulo 7

Conclusiones

7.1 | Comentarios Finales

En el presente trabajo se diseñó y construyó una tarjeta para la adquisición de datos y control de los robots cooperativos A255 y A465. El diseño se realizó a partir de los conocimientos recabados mediante ingeniería inversa y fueron documentados para el apoyo de trabajos futuros en el Laboratorio de Robótica de la División de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México.

Cabe mencionar que la tarjeta es parte fundamental del sistema al que brinda seguridad, además de que distribuye y acopla las señales de interés para el control de los robots.

El costo de producción se conformó únicamente por el costo de fabricación y ensamblaje de la tarjeta debido a que se reutilizaron elementos existentes en el laboratorio.

Para comprobar el desempeño de la tarjeta, se diseñó un controlador PID para el seguimiento de la trayectoria que lleva a los robots a su posición de HOME. El experimento se realizó en el entorno de programación Visual Studio que permitió el desarrollo de una interfaz de usuario gráfica capaz de interactuar con el robot. El programa elaborado cumple con los siguientes puntos:

- Leer la posición actual de ambos robots en coordenadas articulares.
- Mandar voltajes para cada articulación de los robots.
- Realizar paro de emergencia mediante software y hardware.
- Accionar mecánicamente a los robots mediante software.

- Realizar control para llevar a los robots a su posición de HOME.
- Detener al programa.

Las funciones mencionadas fueron consideradas como básicas para la implementación y experimentación de nuevos y diversos algoritmos de control, dichos algoritmos se pueden ampliar en el programa base debido a la versatilidad que ofrece Visual Studio.

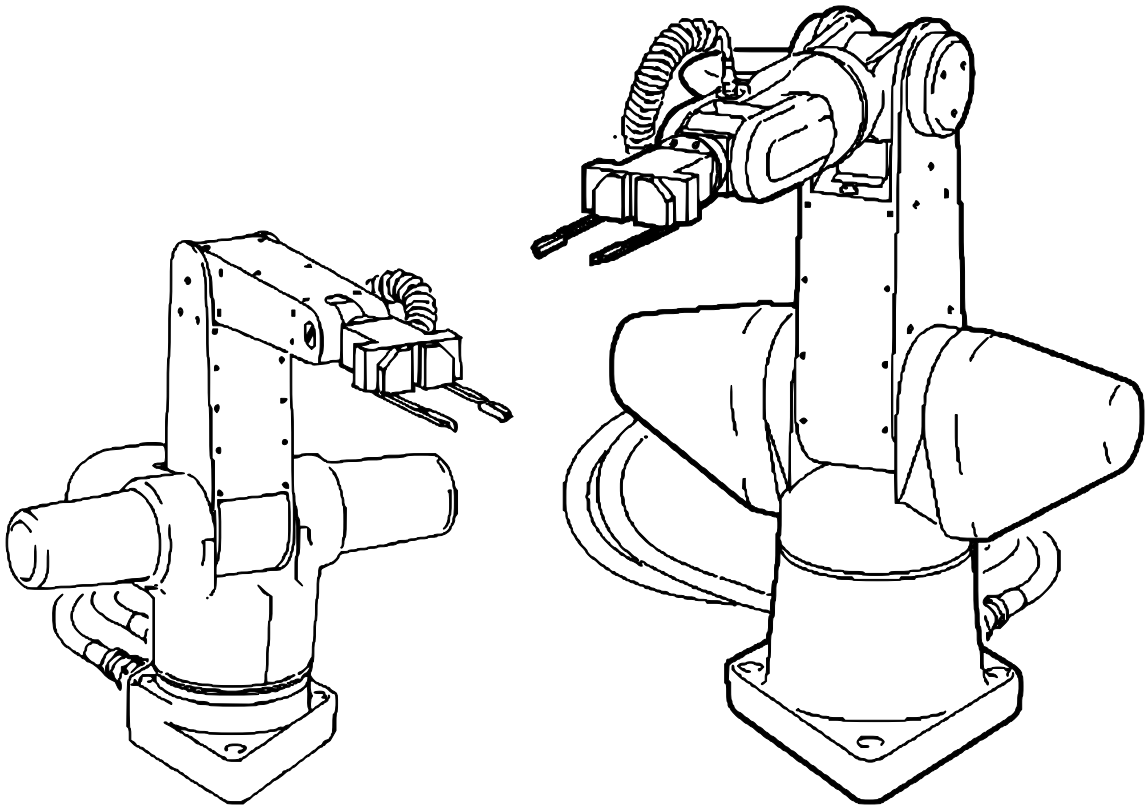
El error obtenido en el experimento permanece constante con un valor dentro de los $\pm 0.5°$ cuando el tiempo de ejecución sobrepasa el tiempo final establecido de $10[s]$ y en donde la trayectoria es igual a la posición final deseada. Este error no es identificable a simple vista y permanece en un margen de tolerancia aceptable, con lo que se demuestra la eficacia de la tarjeta.

7.2 | Trabajo a Futuro

Para simplificar el cableado utilizado en el sistema se puede conectar directamente las señales provenientes de los codificadores al gabinete y utilizar al controlador C500 para la etapa de potencia y protecciones mecánicas únicamente, así como también se puede implementar un algoritmo que detecte posiciones límite para la prevención de colisiones y facilite calibrar al robot de manera automática.

Apéndice A

Especificaciones de los Robots A255 y A465



Especificaciones Técnicas		
Especificación	Robot A465	Robot A255
Peso	32 Kg	17 Kg
Carga (Máxima)	3 Kg	2 Kg
Carga (Nominal)	2 Kg	1 Kg
Alcance (X-Y)	711.2 mm	560 mm
Impulsión	Servo motores de DC	Servo motores de DC
Transmisión	Impulsores Armónicos	Impulsores Armónicos y engranes cónicos/rectos
Repetibilidad	$\pm 0.002[inch]$	$\pm 0.002[inch]$
Rango de acción de las Articulaciones		
Cintura	+175° to -175°	+175° to -175°
Hombro	+ 90° to -90°	+110° to 0°
Codo	+110° to -110°	0° to -125°
Muñeca	+180° to -180°	-
Muñeca (Pitch)	+105° to -105°	+110° to -110°
Herramienta	+180° to -180°	+180° to -180°
Torque Máximo $[N-m]$		
Cintura	39.50	6.4
Hombro	66.08	6.4
Codo	39.50	6.4
Muñeca	6.89	-
Muñeca (Pitch)	6.82	1.4
Herramienta	2.50	0.71
Máxima Velocidad $[rad/s]$		
Cintura	3.14	3.67
Hombro	3.14	3.67
Codo	3.14	3.67
Muñeca	2.99	-
Muñeca (Pitch)	3.02	11.8
Herramienta	2.99	23.6
Aceleraciones por defecto $[rad/s^2]$		
Cintura	12.6	8.7
Hombro	12.6	8.7
Codo	12.6	8.7
Muñeca	24.9	-
Muñeca (Pitch)	25.1	39.3
Herramienta	24.9	78.5

Tabla 1.1: Especificaciones de los robots A465 y A255 [15][16].

A.1 | Robot A255

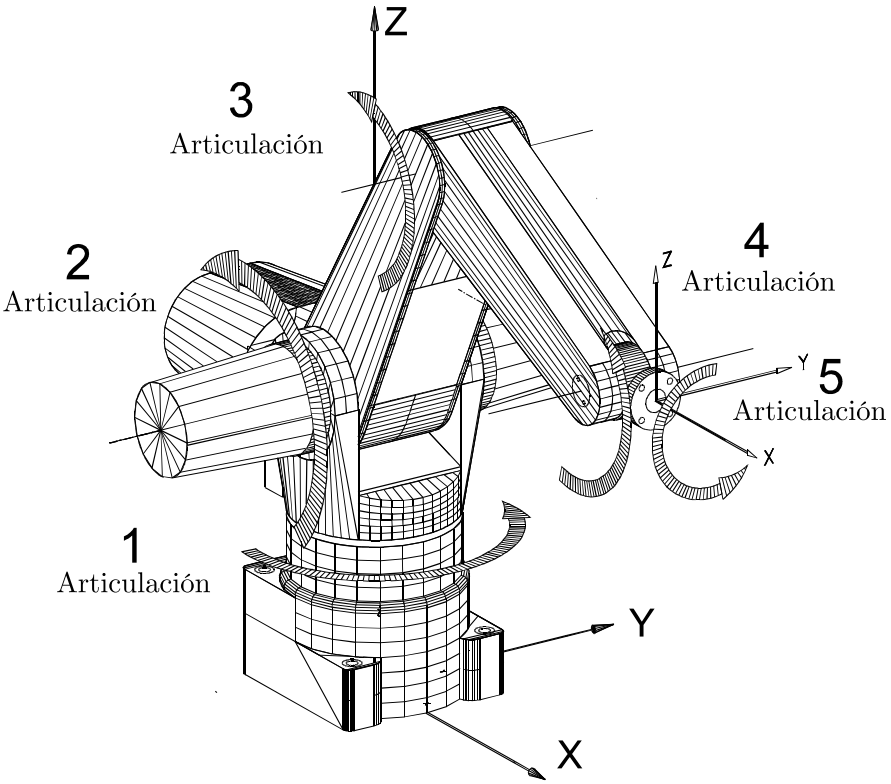


Figura 1.1: Articulaciones del robot A255 [16].

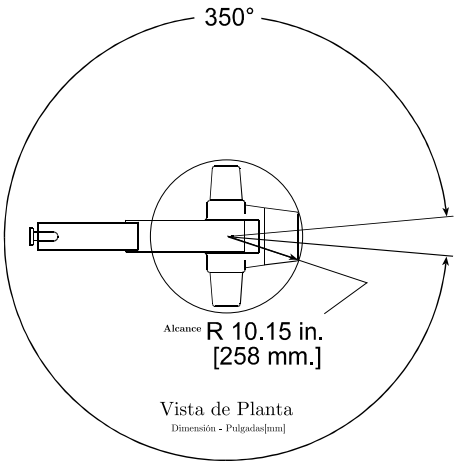


Figura 1.2: Alcance en vista de planta del robot A255 [16].

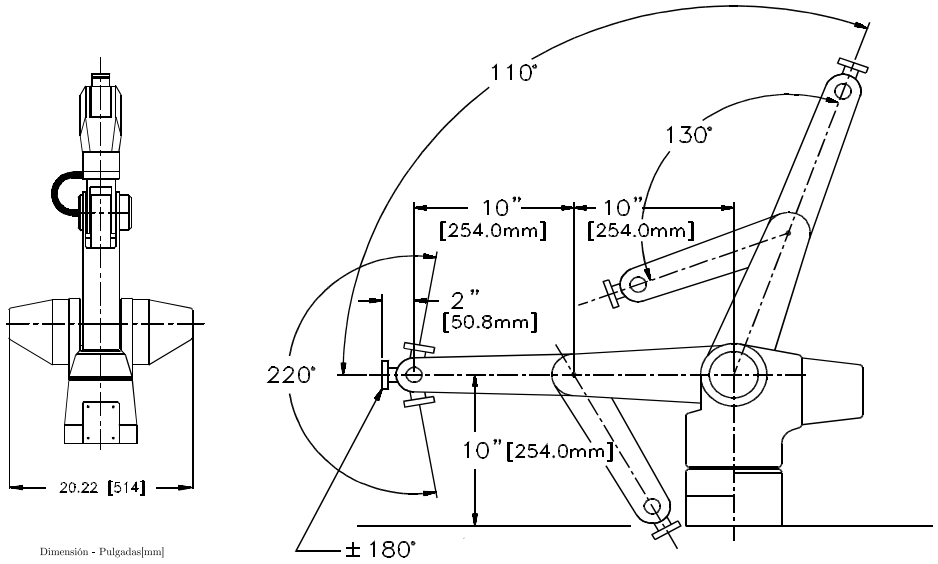


Figura 1.3: Alcance del robot A255 [16].

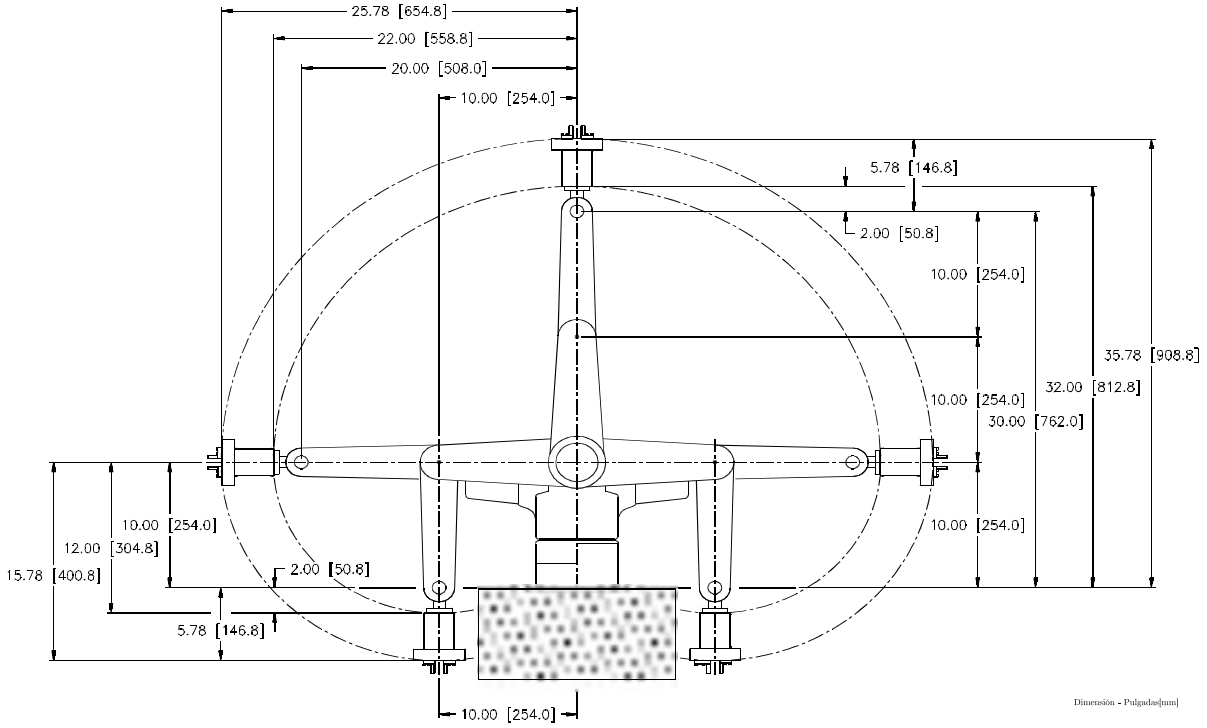


Figura 1.4: Alcance del robot A255 con Gripper [16].

A.2 | Robot A465

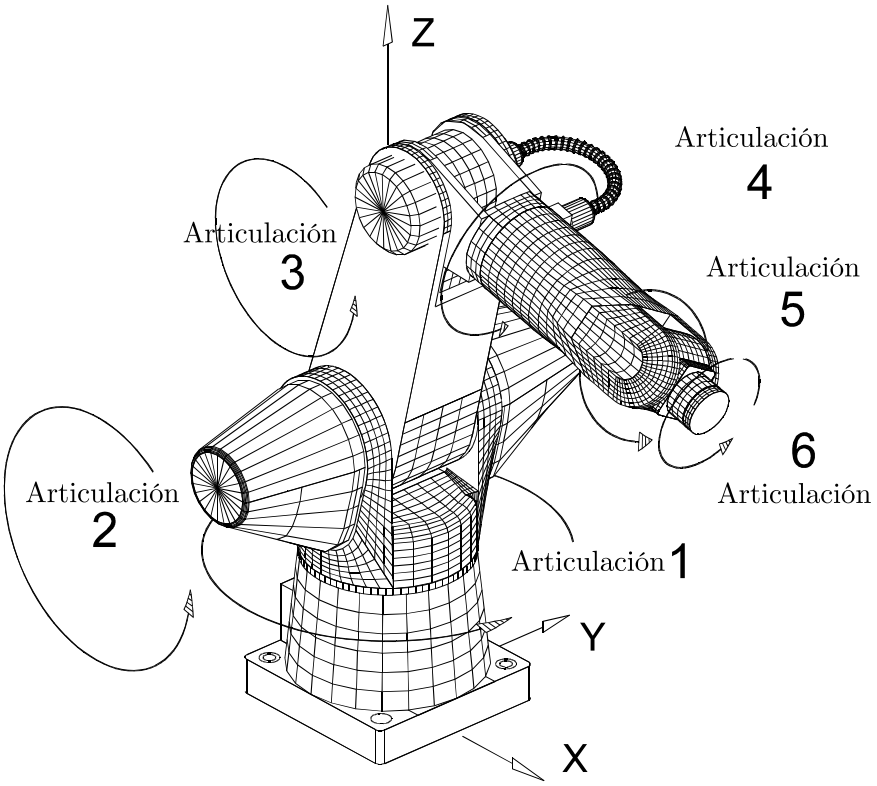


Figura 1.5: Articulaciones del robot A465 [15].

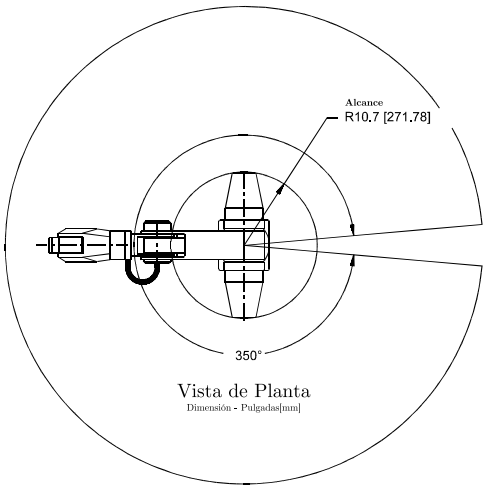


Figura 1.6: Alcance en vista de planta del robot A4655 [15].

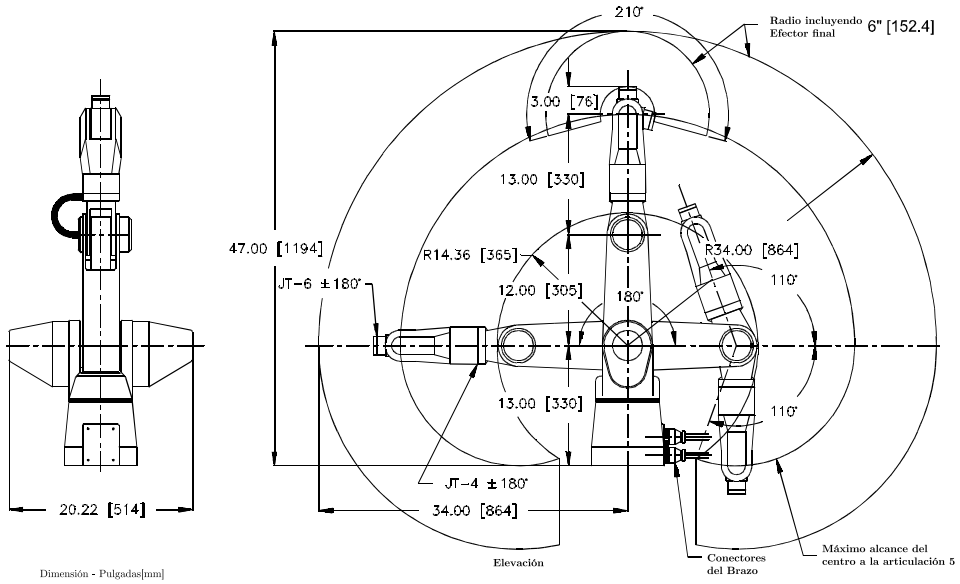


Figura 1.7: Alcance del robot A465 [15].

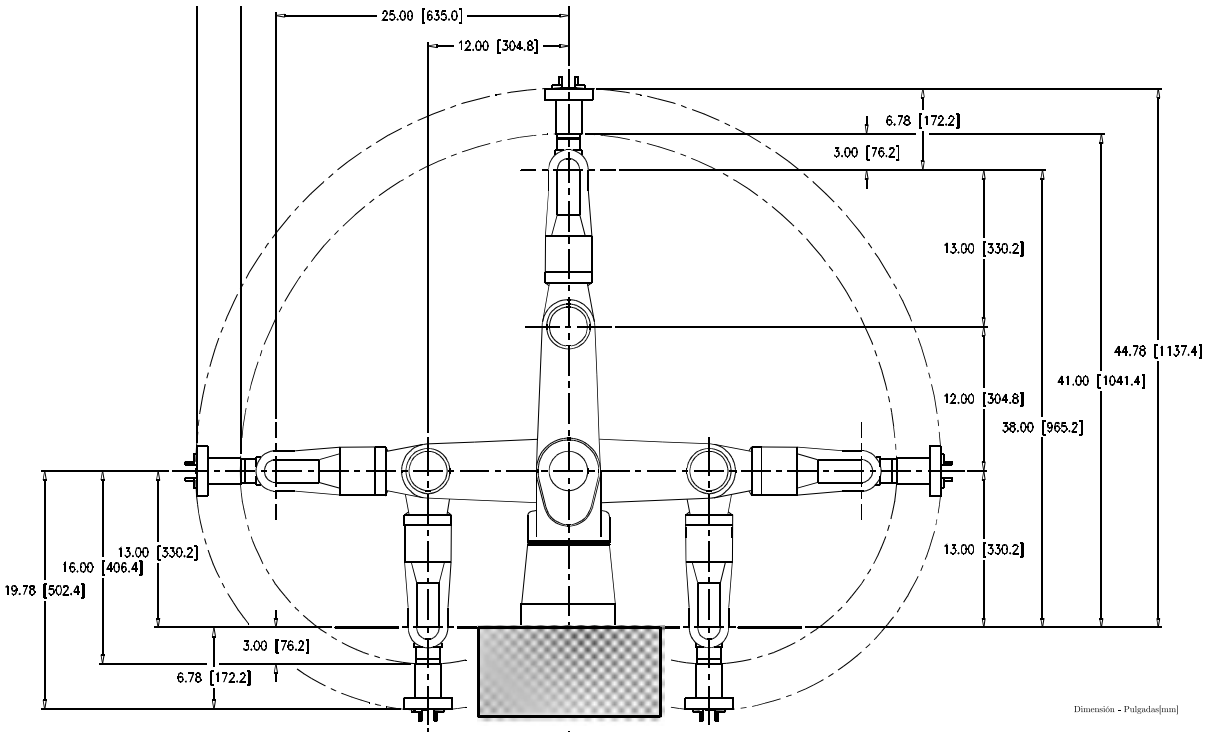
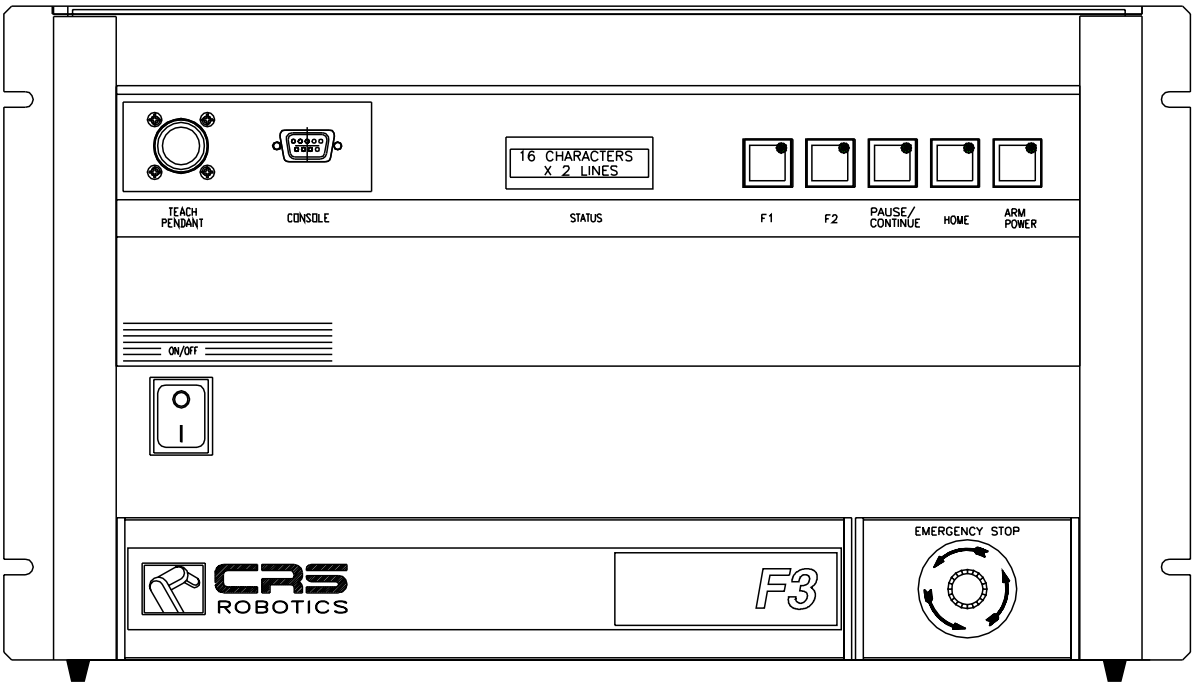


Figura 1.8: Alcance del robot A465 con Gripper [15].

Apéndice B

Especificaciones del controlador C500



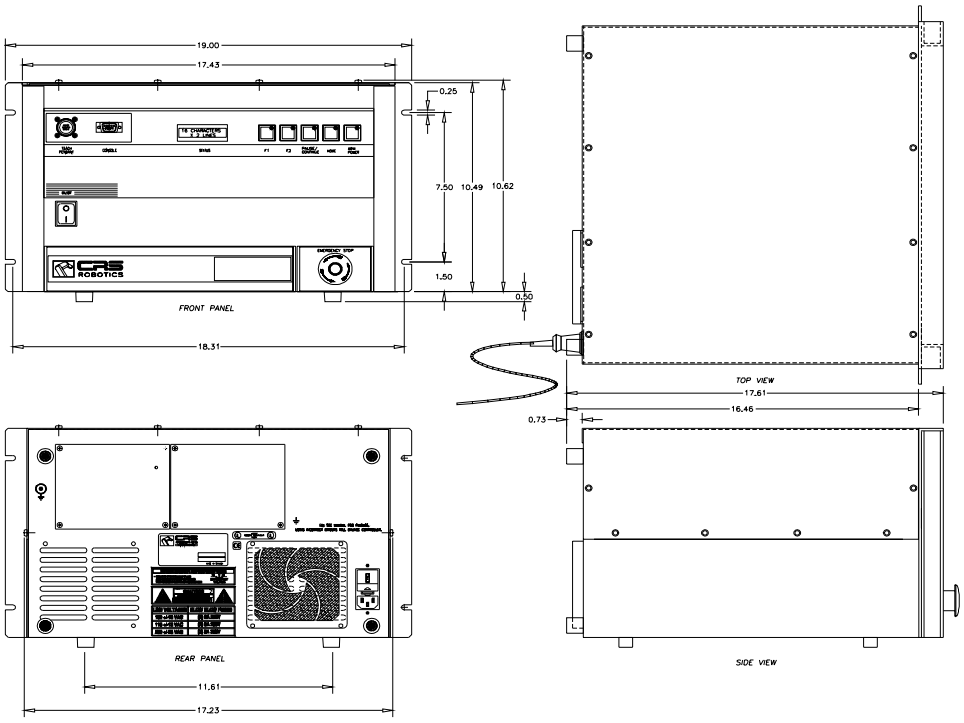


Figura 2.1: Medidas del controlador C500 [18].

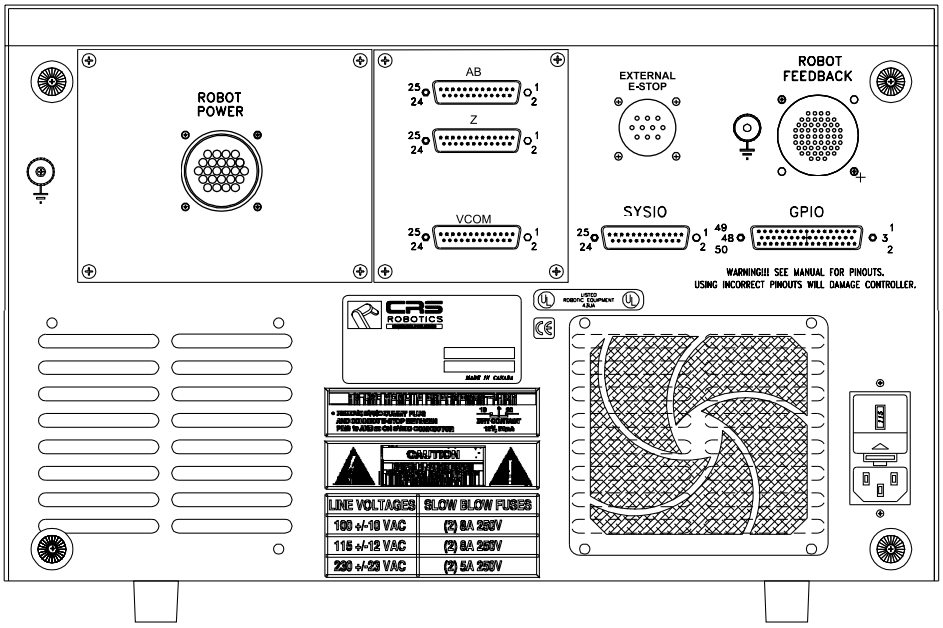


Figura 2.2: Vista trasera del controlador C500.

B.1 | Conectores

B.1.1 | DB-25 Internos

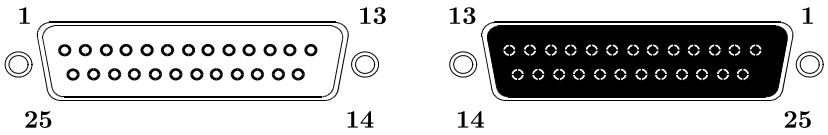


Figura 2.3: Conector DB25 Macho-Hembra.

Conector DB25 [I]		
Pin	Nombre	Descripción
1	Vcom1	Comando de Voltaje para el Eje 1
2	Vcom2	Comando de Voltaje para el Eje 2
3	Vcom3	Comando de Voltaje para el Eje 3
4	Vcom4	Comando de Voltaje para el Eje 4
5	HSW1	Switch de Homing para el Eje 1
6	NSW1	Switch de Viaje Negado para el Eje 1
7	HSW2	Switch de Homing para el Eje 2
8	NSW2	Switch de Viaje Negado para el Eje 2
9	HSW3	Switch de Homing para el Eje 3
10	NSW3	Switch de Viaje Negado para el Eje 3
11	HSW4	Switch de Homing para el Eje 4
12	NSW4	Switch de Viaje Negado para el Eje 4
13	HSW5	Switch de Homing para el Eje 5
14	Vcom5	Comando de Voltaje para el Eje 5
15	Vcom6	Comando de Voltaje para el Eje 6
16	Vcom7	Comando de Voltaje para el Eje 7
17	Vcom8	Comando de Voltaje para el Eje 8
18	PSW1	Switch de Viaje Positivo para el Eje 1
19	TSW1	Switch Térmico para el Eje 1
20	PSW2	Switch de Viaje Positivo para el Eje 2
21	TSW2	Switch Térmico para el Eje 2
22	PSW3	Switch de Viaje Positivo para el Eje 3
23	TSW3	Switch Térmico para el Eje 3
24	PSW4	Switch de Viaje Positivo para el Eje 4
25	TSW4	Switch Térmico para el Eje 4

Tabla 2.1: Asignación de pines del conector DB-25[I] del controlador C500.

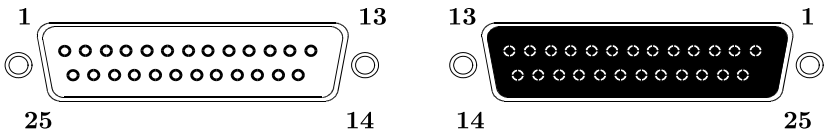


Figura 2.4: Conector DB25 Macho-Hembra

Conector DB25 [II]		
Pin	Nombre	Descripción
1	PSW5	Switch de Viaje Positivo para el Eje 5
2	TSW5	Switch Térmico para el Eje 5
3	PSW6	Switch de Viaje Positivo para el Eje 6
4	TSW6	Switch Térmico para el Eje 6
5	TPEStop-	E-Stop del Teach Pendant
6	LiveMan-	E-Stop del Teach Pendant
7	Brake-	Switch brake relay return
8	ArmOn-	Arm Power relay return
9	Analogln2	
10	AEEStopt-	
11	N/C	
12	N/C	
13	N/C	
14	NSW5	Switch de Viaje Negado para el Eje 5
15	HSW6	Switch de Homing para el Eje 6
16	NSW6	Switch de Viaje Negado para el Eje 6
17	TPEStop+	E-Stop del Teach Pendant
18	LiveMan+	Teach pendant LiveMan SP
19	Break+	Break relay source
20	ArmOn+	Arm Power relay source
21	Analogln1	
22	AEEStop+	Auxiliary E-Stop Switch pair.
23	N/C	
24	N/C	
25	N/C	

Tabla 2.2: Asignación de pines del conector DB-25[II] del controlador C500.

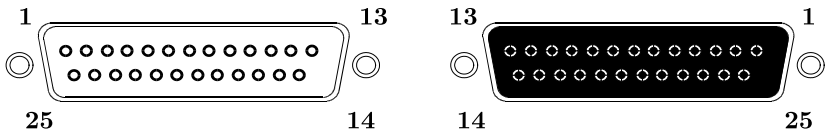


Figura 2.5: Conector DB25 Macho-Hembra

Conector DB25 [1]		
Pin	Nombre	Descripción
1	A5+	Eje 5 Canal A
2	Z4+	Eje 4 Canal Z
3	B4+	Eje 4 Canal B
4	A4+	Eje 4 Canal A
5	Z3+	Eje 3 Canal Z
6	B3+	Eje 3 Canal B
7	A3+	Eje 3 Canal A
8	Z2+	Eje 2 Canal Z
9	B2+	Eje 2 Canal B
10	A2+	Eje 2 Canal A
11	Z1+	Eje 1 Canal Z
12	B1+	Eje 1 Canal B
13	A1+	Eje 1 Canal A
14	Z4-	Eje 4 Canal Z (negado)
15	B4-	Eje 4 Canal B (negado)
16	A4-	Eje 4 Canal A (negado)
17	Z3-	Eje 3 Canal Z (negado)
18	B3-	Eje 3 Canal B (negado)
19	A3-	Eje 3 Canal A (negado)
20	Z2-	Eje 2 Canal Z (negado)
21	B2-	Eje 2 Canal B (negado)
22	A2-	Eje 2 Canal A (negado)
23	Z1-	Eje 1 Canal Z (negado)
24	B1-	Eje 1 Canal B (negado)
25	A1-	Eje 1 Canal A (negado)

Tabla 2.3: Asignación de pines del conector DB-25[1] del controlador C500.

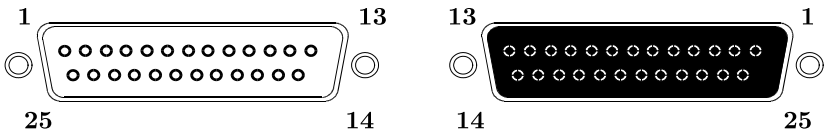


Figura 2.6: Conector DB25 Macho-Hembra.

Conector DB25 [2]		
Pin	Nombre	Descripción
1	GND	
2	GND	
3	GND	
4	GND	
5	SGTOR Salida	Servo gripper torque (sin uso)
6		
7	Shield	No conectar
8	Z6-	Eje 6 Canal Z Entrada (negado)
9	B6-	Eje 6 Canal B Entrada (negado)
10	A6-	Eje 6 Canal A Entrada (negado)
11	Z5-	Eje 5 Canal Z Entrada (negado)
12	B5-	Eje 5 Canal B Entrada (negado)
13	A5-	Eje 5 Canal A Entrada (negado)
14	+12 V	+12 Volts supply to the servo gripper
15	+12 V	+12 Volts supply to the servo gripper
16		
17	AirGrip-	Solenoid return
18	SGPos	Servo gripper position
19		
20	N/C	No Connect
21	Z6+	Eje 6 Canal Z Entrada
22	B6+	Eje 6 Canal B Entrada
23	A6+	Eje 6 Canal A Entrada
24	Z5+	Eje 5 Canal Z Entrada
25	B5+	Eje 5 Canal B Entrada

Tabla 2.4: Asignación de pines del conector DB-25[2] del controlador C500.

B.1.2 | Conectores del Controlador C500 al Gabinete

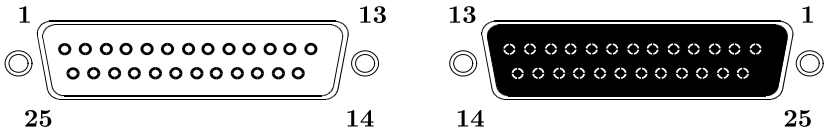


Figura 2.7: Conector DB25 Macho-Hembra

Conector DB25 [AB]		
Pin	Nombre	Descripción
1	A1+	Eje 1 Canal A
2	A1-	Eje 1 Canal A (negado)
3	A2+	Eje 2 Canal A
4	A2-	Eje 2 Canal A (negado)
5	A3+	Eje 3 Canal A
6	A3-	Eje 3 Canal A (negado)
7	A4+	Eje 4 Canal A
8	A4-	Eje 4 Canal A (negado)
9	A5+	Eje 5 Canal A
10	A5-	Eje 5 Canal A (negado)
11	A6+	Eje 6 Canal A
12	A6-	Eje 6 Canal A (negado)
13	B1+	Eje 1 Canal B
14	B1-	Eje 1 Canal B (negado)
15	B2+	Eje 2 Canal B
16	B2-	Eje 2 Canal B (negado)
17	B3+	Eje 3 Canal B
18	B3-	Eje 3 Canal B (negado)
19	B4+	Eje 4 Canal B
20	B4-	Eje 4 Canal B (negado)
21	B5+	Eje 5 Canal B
22	B5-	Eje 5 Canal B (negado)
23	B6+	Eje 6 Canal B
24	B6-	Eje 6 Canal B (negado)
25	GND	0[v]

Tabla 2.5: Asignación de pines del conector DB-25[AB] del controlador C500.

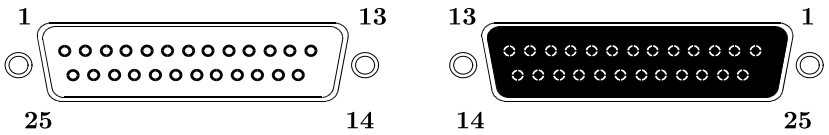


Figura 2.8: Conector DB25 Macho-Hembra

Conector DB25 [Z]		
Pin	Nombre	Descripción
1	Z1+	Eje 1 Canal Z
2	Z1-	Eje 1 Canal Z (negado)
3	Z2+	Eje 2 Canal Z
4	Z2-	Eje 2 Canal Z (negado)
5	Z3+	Eje 3 Canal Z
6	Z3-	Eje 3 Canal Z (negado)
7	Z4+	Eje 4 Canal Z
8	Z4-	Eje 4 Canal Z (negado)
9	Z5+	Eje 5 Canal Z
10	Z5-	Eje 5 Canal Z (negado)
11	Z6+	Eje 6 Canal Z
12	Z6-	Eje 6 Canal Z (negado)
13	N/C	Sin Conección
14	N/C	Sin Conección
15	N/C	Sin Conección
16	N/C	Sin Conección
17	N/C	Sin Conección
18	N/C	Sin Conección
19	N/C	Sin Conección
20	N/C	Sin Conección
21	N/C	Sin Conección
22	N/C	Sin Conección
23	N/C	Sin Conección
24	N/C	Sin Conección
25	GND	0[v]

Tabla 2.6: Asignación de pines del conector DB-25[Z] del controlador C500.

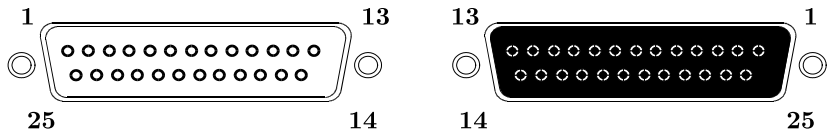


Figura 2.9: Conector DB25 Macho-Hembra

Conector DB25 [VCOM]		
Pin	Nombre	Descripción
1	HSW1	Switch de Homing para el Eje 1
2	HSW2	Switch de Homing para el Eje 2
3	HSW3	Switch de Homing para el Eje 3
4	HSW4	Switch de Homing para el Eje 4
5	HSW5	Switch de Homing para el Eje 5
6	HSW6	Switch de Homing para el Eje 6
7	N/C	Sin Conección
8	N/C	Sin Conección
9	N/C	Sin Conección
10	N/C	Sin Conección
11	N/C	Sin Conección
12	N/C	Sin Conección
13	VCOM1	Comando de Voltaje para el Eje 1
14	VCOM2	Comando de Voltaje para el Eje 2
15	VCOM3	Comando de Voltaje para el Eje 3
16	VCOM4	Comando de Voltaje para el Eje 4
17	VCOM5	Comando de Voltaje para el Eje 5
18	VCOM6	Comando de Voltaje para el Eje 6
19	N/C	Sin Conección
20	N/C	Sin Conección
21	N/C	Sin Conección
22	N/C	Sin Conección
23	N/C	Sin Conección
24	N/C	Sin Conección
25	N/C	Sin Conección

Tabla 2.7: Asignación de pines del conector DB-25[VCOM] del controlador C500.

B.1.3 | Conectores del controlador C500 a los Robots

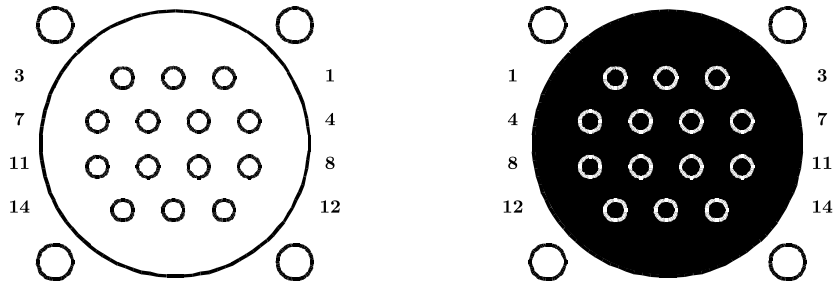


Figura 2.10: Conector Redondo de 14 pines

Conector Redondo de Potencia de los Motores (Robot A255)		
Pin	Nombre	Descripción
1	Motor 1+	Motorpower±25V 2Amax
2	Motor 1-	Motor power return
3	Motor 2+	Motorpower±25V 2Amax
4	Motor 2-	Motor power return
5	Motor 3+	Motorpower±25V 2Amax
6	Motor 3-	Motor power return
7	N/C	
8	Motor 4+	Motorpower±25V 2Amax
9	Motor 4-	Motor power return
10	Motor 5+	Motorpower±25V 2Amax
11	Motor 5-	Motor power return
12	Motor 6+	Motorpower±25V 2Amax
13	Motor 6-	Motor power return
14	N/C	
15	N/C	
16	N/C	
17	N/C	
18	N/C	
19	N/C	
20	N/C	
21	N/C	
22	N/C	
23	N/C	
24	N/C	

Tabla 2.8: Asignación de pines del conector de Potencia de los Motores del controlador C500 (Robot A255).

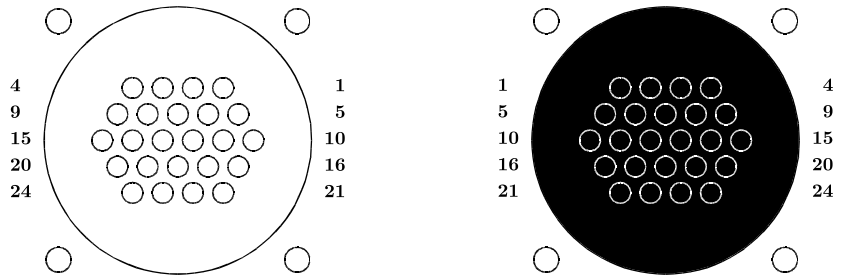


Figura 2.11: Conector Redondo de 24 pines

Conector Redondo de Potencia de los Motores (Robot A465)		
Pin	Nombre	Descripción
1	Motor 1+	Motorpower±25V 2Amax
2	Motor 1-	Motor power return
3	Motor 2+	Motorpower±25V 2Amax
4	Motor 2-	Motor power return
5	Motor 3+	Motorpower±25V 2Amax
6	Motor 3-	Motor power return
7	N/C	
8	Motor 4+	Motorpower±25V 2Amax
9	Motor 4-	Motor power return
10	Motor 5+	Motorpower±25V 2Amax
11	Motor 5-	Motor power return
12	Motor 6+	Motorpower±25V 2Amax
13	Motor 6-	Motor power return
14	N/C	
15	N/C	
16	N/C	
17	N/C	
18	HomeSw 1	Home switch
19	HomeSw 2	Home switch
20	HomeSw 3	Home switch
21	N/C	
22	HomeSw 4	Home switch
23	HomeSw 5	Home switch
24	HomeSw 6	Home switch

Tabla 2.9: Asignación de pines del conector de Potencia de los Motores del controlador C500 (Robot A465).

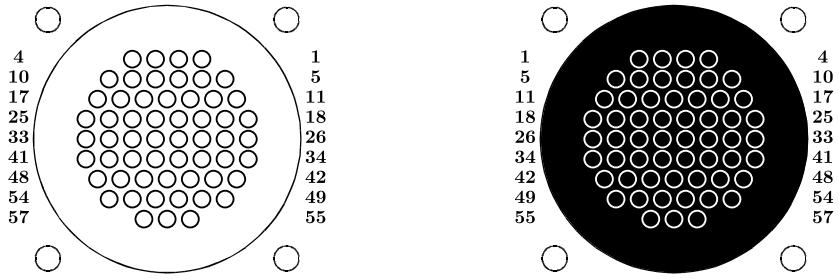


Figura 2.12: Conector Redondo de 57 pines

Conector Redondo de Feedback		
Pin	Nombre	Descripción
1	1A	RS422+, 200 Khz max pulse rate
2	1B	RS422+, 200 Khz max pulse rate
3	2A	RS422+, 200 Khz max pulse rate
4	2B	RS422+, 200 Khz max pulse rate
5	1A*	RS422-, 200 Khz max pulse rate
6	1B*	RS422-, 200 Khz max pulse rate
7	1Z	RS422+, 200 Khz max pulse rate
8	2Z	RS422+, 200 Khz max pulse rate
9	2A*	RS422-, 200 Khz max pulse rate
10	2B*	RS422-, 200 Khz max pulse rate
11,18	BRAKE	35 V 100 mA
12,13	SGM	±15 V 300 mA
14	1Z*	RS422-, 200 Khz max pulse rate
15,16	Vcc	Encoder Supply +5 VDC 80 mA
17	Vcc	Encoder Supply +5 VDC 80 mA
19	6A	RS422+, 200 Khz max pulse rate
20	3A	RS422+, 200 Khz max pulse rate
21	3B	RS422+, 200 Khz max pulse rate
22	4A	RS422+, 200 Khz max pulse rate
23	4B	RS422+, 200 Khz max pulse rate
24,25	GND	Encoder digital return
26	6B	RS422+, 200 Khz max pulse rate
27	6Z	RS422+, 200 Khz max pulse rate
28	3Z	RS422+, 200 Khz max pulse rate
29	3A*	RS422-, 200 Khz max pulse rate
30	3B*	RS422-, 200 Khz max pulse rate

Tabla 2.10: Asignación de pines del conector redondo de Realimentación del controlador C500.

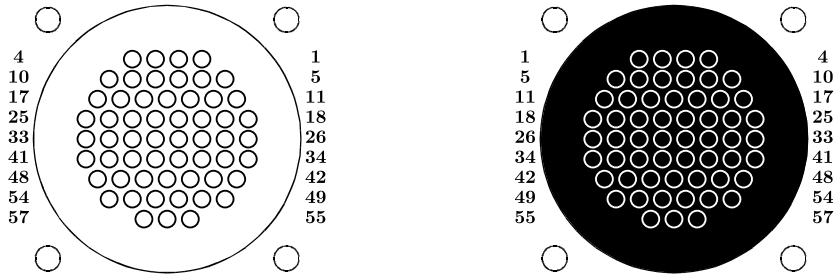


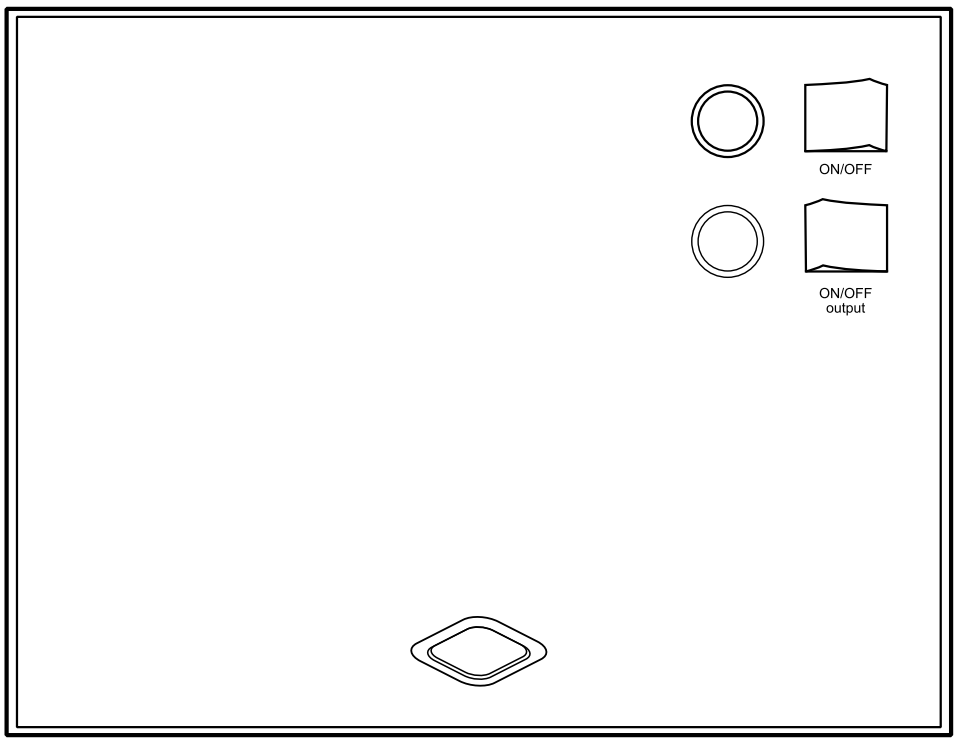
Figura 2.13: Conector Redondo de 57 pines

Conector Redondo de Realimentación		
Pin	Nombre	Descripción
31	4Z	RS422+, 200 Khz max pulse rate
32,33	BRAKERET*	return for brake supply
34	6A*	RS422-, 200 Khz max pulse rate
35	6B*	RS422-, 200 Khz max pulse rate
36	5A*	RS422-, 200 Khz max pulse rate
37	5B*	RS422-, 200 Khz max pulse rate
38	3Z*	RS422-, 200 Khz max pulse rate
39	4A*	RS422-, 200 Khz max pulse rate
40	GND	Encoder Digital return
41	2Z*	RS422-, 200 Khz max pulse rate
42	6Z*	RS422-, 200 Khz max pulse rate
43	5A	RS422+, 200 Khz max pulse rate
44	5Z*	RS422-, 200 Khz max pulse rate
45	4Z*	RS422-, 200 Khz max pulse rate
46	4B	RS422-, 200 Khz max pulse rate
47	SGRIPTRQ	Sin uso
48	SGRIPPOS	0-4.7 V
49	5B	RS422+, 200 Khz max pulse rate
50	5Z	RS422+, 200 Khz max pulse rate
51,52	Vcc	+5 VDC 80 mA
53	S/A G+	Air solenoid 12 V 200 mA
54	S/A G-	Air solenoid return.
55,57	GND	Encoder Digital return
56	N/C	

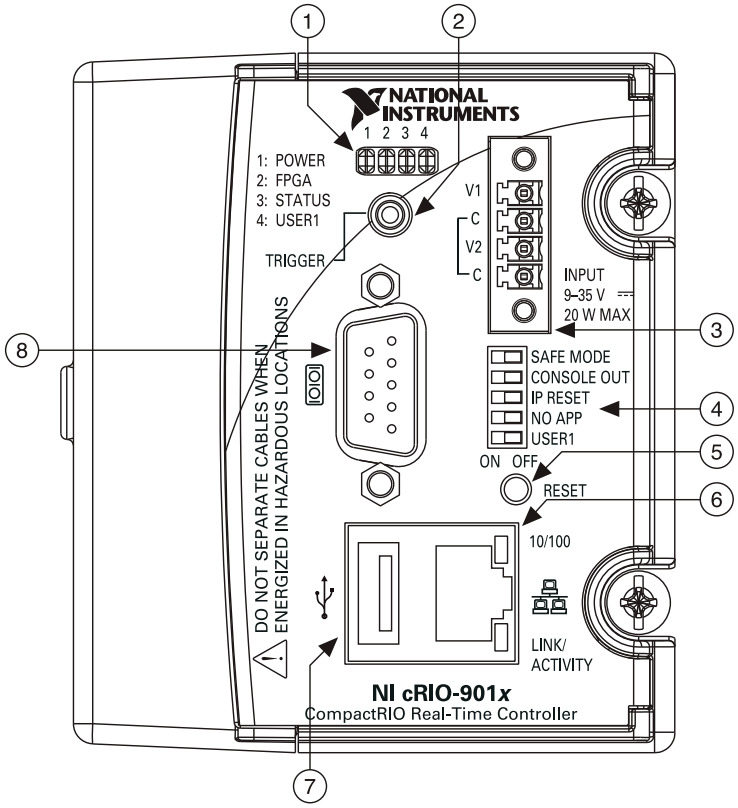
Tabla 2.11: Asignación de pines del conector redondo de Realimentación del controlador C500.

Apéndice C

Especificaciones de elementos de la tarjeta y el gabinete



C.1 | CompactRIO NI-cRIO 9014



1. LEDs
2. Conector SMB
3. Potencia de Entrada
4. DIP Switches
5. Botón de RESET
6. Puerto Ethernet RJ-45
7. Puerto USB
8. Puerto Serial RS-232

Figura 3.1: Panel frontal del sistema CompactRIO [20].

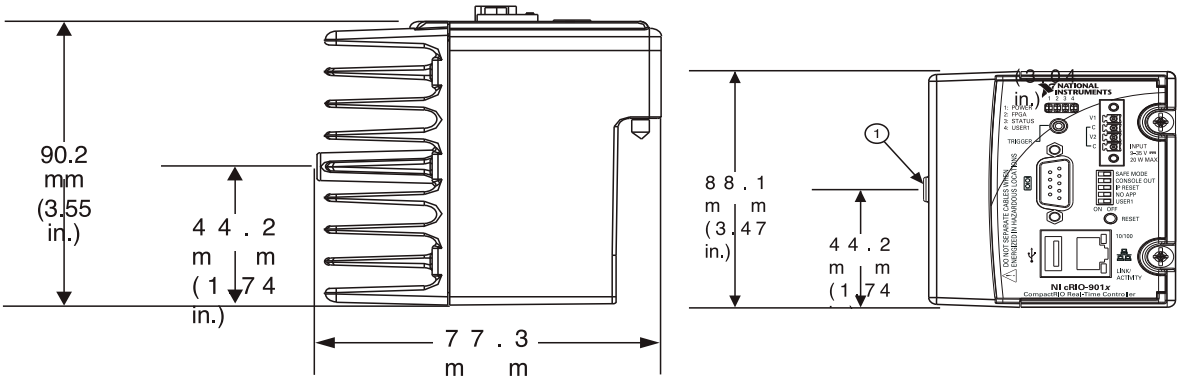


Figura 3.2: Dimensiones del sistema CompactRIO [20].

Las siguientes especificaciones son típicas en el rango de temperatura de funcionamiento de -20 a 55 °C, a menos que se indique lo contrario.

Especificaciones Técnicas	
Característica	Cantidad
Red	
Interfaz de red	10BaseT-100BaseTX Ethernet
Compatibilidad	IEEE 802.3
Tasas de Comunicación	10 Mbps, 100 Mbps
Distancia máxima de cableado	100 m / segmento
Puerto Serial RS-232	
Velocidad de transmisión máxima	115.200 bps
Bits de datos	5, 6, 7, 8
Bits de parada	1, 2
Paridad	Impar, Par, Marcos
Control de flujo Espacio	RTS/CTS,XON/XOFF
Conector SMB	
Características de salida	
Nivel lógico alto	3.3V
Nivel lógico bajo	0V
Tipo de salida	CMOS
Corriente de salida	±50mA
Características de entrada	
Voltaje de entrada mínimo	-500 mV
Voltaje de entrada de bajo nivel máximo	0,94 V
Voltaje de entrada de alto nivel mínimo	2,43 V
Voltaje de entrada máxima	5,5 V
Capacitanci de entrada	2.5 pF
Puerto USB	
Velocidad máxima	12 Mb/s
Máxima Corriente	500 mA
FPGA Reconfigurable	
Número de celdas lógicas	46.08
Disponible RAM embebida	720 kbits

Tabla 3.1: Especificaciones del Sistema CompactRIO NI-cRIO-9014 [20].

Especificaciones Técnicas	
Característica	Cantidad
Memoria	
No volátil	2 GB
Memoria del sistema	128 MB
Reloj Interno en Tiempo Real	
Exactitud	200 ppm
Requerimientos de Energía	
Fuente de alimentación recomendada	48 W, 24 V CC
Consumo de energía	20W
Fuente de alimentación	19 al 35 V

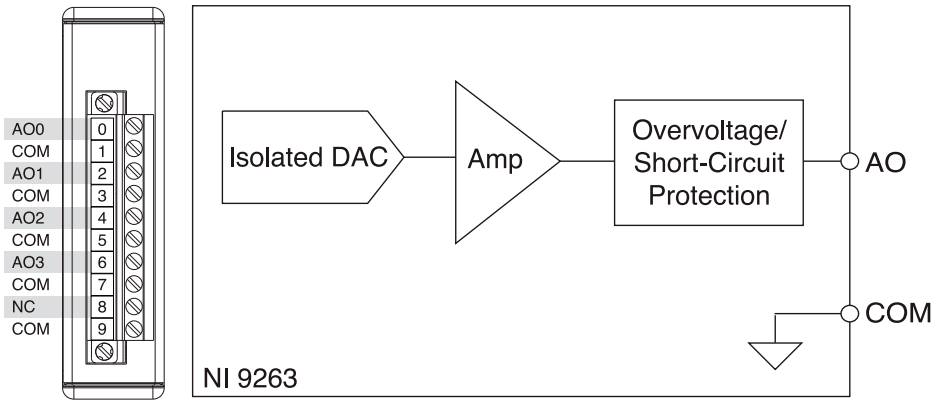
Tabla 3.2: Especificaciones del Sistema CompactRIO NI-cRIO-9014 [20].

C.1.1 | Módulo NI 9263

Las siguientes especificaciones son típicas en el rango de temperatura de funcionamiento de -40 a 70 °C, a menos que se indique lo contrario.

Especificaciones Técnicas	
Característica	Cantidad
Características de salida	
Número de canales	4 salidas analógicas
Resolución del DAC	16bits
Tipo de DAC	String
Estado de salida de encendido	Canales off
Tensión de inicio	0 V
Voltaje de la Energía-abajo	0 V
Impedancia de salida	2Ω
Velocidad de respuesta	4V/μs
Diafonía	76 dB
Rango de voltaje de salida	
Nominal	±10 V
Mínimo	±10.4 V
Típico	±0.7 V
Máxima	±11 V

Tabla 3.3: Especificaciones del módulo NI-9263 [21].



(a) Conector (b) Diagrama interno

Figura 3.3: Módulo NI 9263. [21]

C.1.2 | Módulo NI 9401

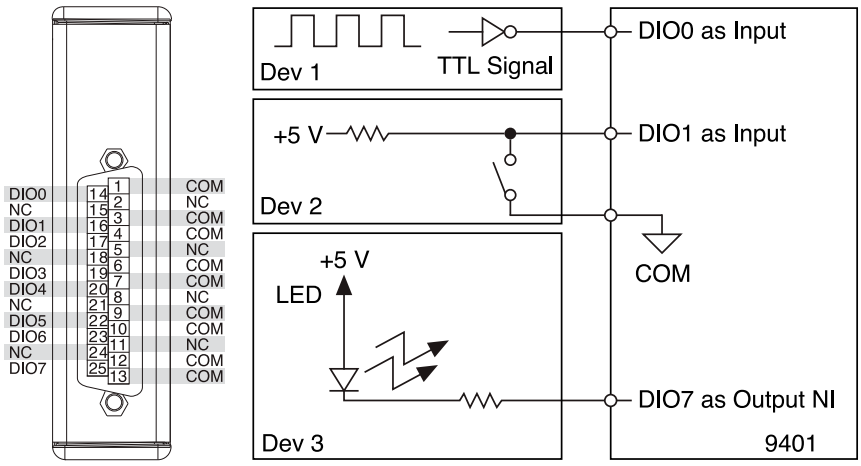
Las siguientes especificaciones son típicas en el rango de temperatura de funcionamiento de -40 a 70 °C, a menos que se indique lo contrario.

Especificaciones Técnicas	
Característica	Cantidad
Características de entrada y salida	
Número de canales	8 DIO
Dirección predeterminada	Entrada
Tipo de salida	TTL
Niveles lógicos digitales	
Entrada	
Tensión	5.25 V max
Alto, V_{IH}	2 V min
Bajo, V_{IL}	0,8 V max
Salida	
Alto, V_{OH}	5,25 V max
Sourcing 100 mu	4.7 V min
Sourcing 2 mA	4.3 V min
Low, V_{OL}	
Sinking 100 mu	0,1 V max
Sinking 2 mA	0,4 V max

Tabla 3.4: Especificaciones del módulo NI-9401 [22].

Especificaciones Técnicas	
Característica	Cantidad
Características de entrada y salida	
Retraso de propagación	100ns max
Corriente de entrada	$\pm 250\mu$A
Frecuencia de conmutación de la señal de entrada máxima por número de canales de entrada	
8 canales de entrada	9 MHz
4 canales de entrada	16 MHz
2 canales de entrada	30 MHz
Frecuencia de conmutación de la señal de salida máxima por número de canales de entrada	
8 canales de entrada	5 MHz
4 canales de entrada	10 MHz
2 canales de entrada	20 MHz

Tabla 3.5: Especificaciones del módulo NI-9401 [22].



(a) Conector (b) Uso sugerido de los puertos

Figura 3.4: Módulo NI 9401 [22].

C.2 | Quint Power PS-100

Las siguientes especificaciones son típicas en el rango de temperatura de funcionamiento de -25 a 60 °C, a menos que se indique lo contrario.

Especificaciones Técnicas	
Característica	Cantidad
Características de entrada	
Rango de voltaje nominal	100V AC - 240 V AC
Rango de voltaje AC	85 V - 264 V
Rango de voltaje DC	90 V - 250 V
Rango de frecuencia	45 Hz - 65 Hz
Consumo nominal	240W
Corriente transitoria	$<$ 15A
Características de salida	
Voltaje nominal	24V DC $\pm$1 %
Rango de voltajes ajustados	22.5V DC - 28.25V DC
Corriente nominal	10 A
Boost de potencia	15A
Potencia de salida	240W
Tiempo de respuesta	$<$1s

Tabla 3.6: Especificaciones de la unidad Quint Power PS-100 [23].

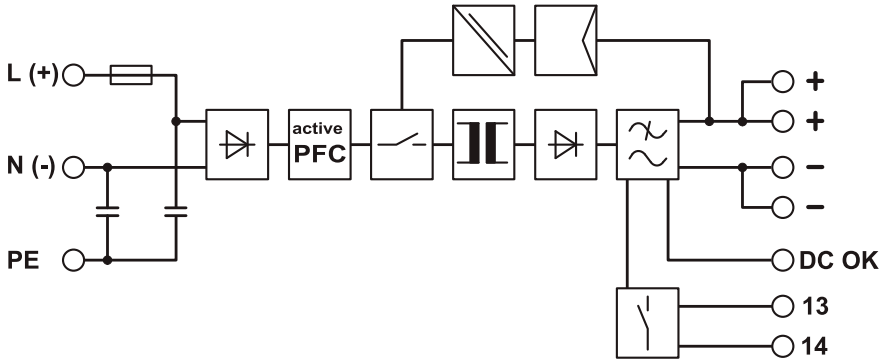


Figura 3.5: Diagrama a bloques de la unidad Quint Power PS-100 [23].

C.3 | Características Físicas del Gabinete

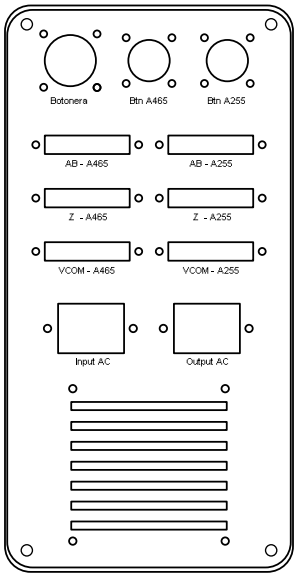


Figura 3.6: Tapa lateral del gabinete.

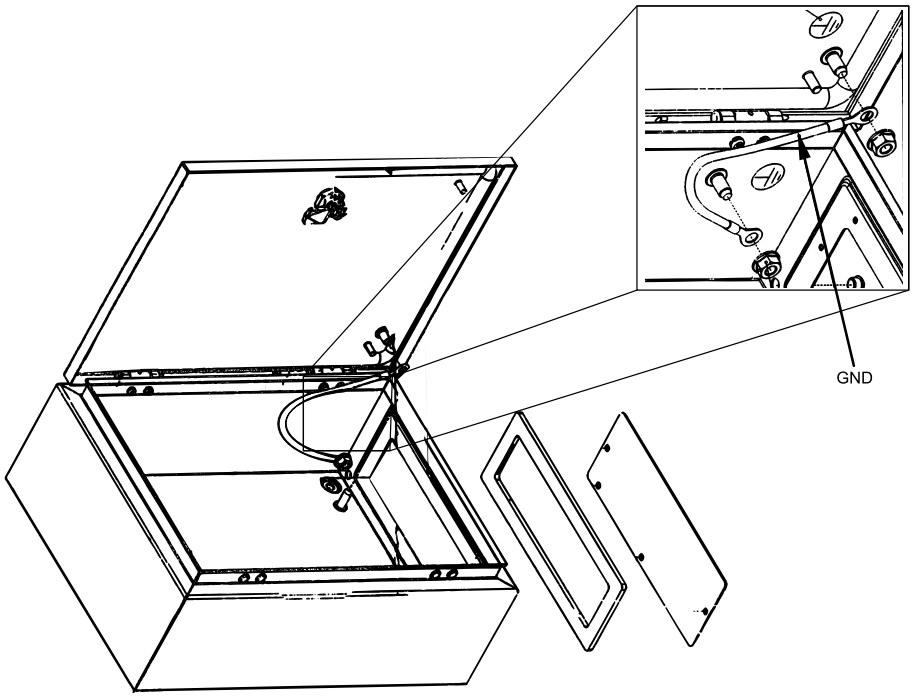


Figura 3.7: Conexión a tierra del gabinete y tapa lateral desprendida.

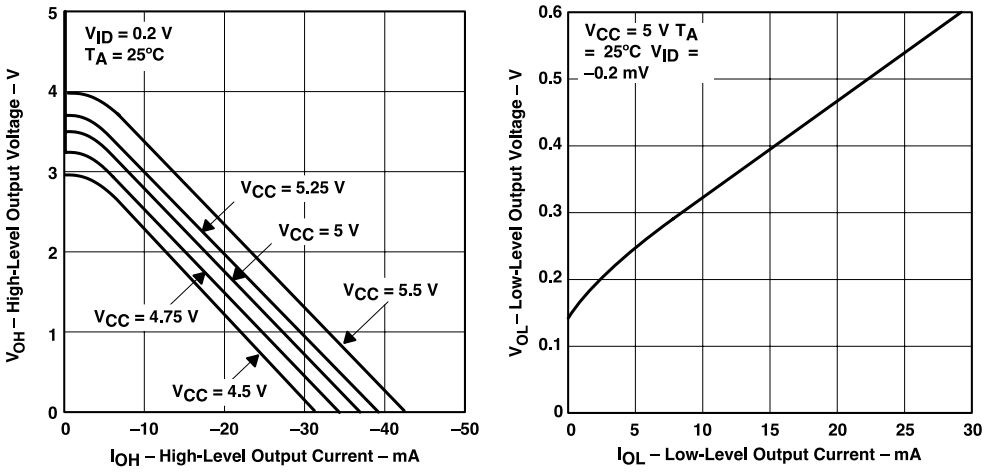
C.4 | Tarjeta

C.4.1 | Integrado 26LS32

Los dispositivos AM26LS32A y AM26LS33A son receptores de línea diferencial cuádruples para la transmisión de datos digitales balanceadas y desbalanceadas. La función de eneable es común a los cuatro receptores y ofrece una selección de-activa alta o activa baja a la entrada. La conexión de las salidas 3-permiso de estado directamente a un sistema de bus-organizado. A prueba de fallos de diseño se asegura de que, si las entradas están abiertas, las salidas siempre son altas.

Especificaciones Técnicas	
Característica	Cantidad
Rango de voltaje de alimentación	4.75V - 5.25V
Voltaje de alimentación nominal V_{cc}	5V
Voltaje de entrada V_I	±25V
Voltaje de entra nivel alto V_{IH}	2V min
Voltaje de entra nivel bajo V_{IL}	0.8V max
Voltaje de entrada diferencial V_{ID}	±25V
Rango de temperatura de operación	0°C - 70°C

Tabla 3.7: Especificaciones del integrado 26LS32 [26].



(a) Nivel alto (b) Nivel bajo

Figura 3.8: Voltaje de salida vs Corriente de salida [26] .

C.4.2 | Integrado HCPL2630

El integrado HCPL-2630 esta conformado por compuertas ópticamente acopladas que combinan un diodo de luz GaAsP emit- ting y un foto-detector de alta ganancia integrada. La salida del IC es un colector abierto Schottky. Estos integrados son adecuados para interfaces de alta velocidad.

Especificaciones Técnicas	
Característica	Cantidad
Corriente de salida alto nivel	5.5μA
Umbral de entrada de corriente	2.5mA
Voltaje de salida bajo nivel	0.36V max
Corriente de alimentación bajo nivel	10mA
Corriente de alimentación alto nivel	13mA
Voltaje de entrada mínimo	1.5V
Voltaje de ruptura	5V
Tiempo de retraso de propagación alto nivel t_{PLH}	48ns
Tiempo de retraso de propagación bajo nivel t_{PHL}	50ns
Distorción de ancho de pulso	3.5ns
Capacitancia de entrada	60pF
Tiempo de cambio de la salida 10-90 %	24ns
Tiempo de cambio de la salida 90-10 %	10ns
Aislamiento entrada-salida	1μA max.
Resistencia entrada-salida	1TΩ
Resistencia entrada-entrada	100GΩ

Tabla 3.8: Especificaciones del integrado HCPL 2630 [27].

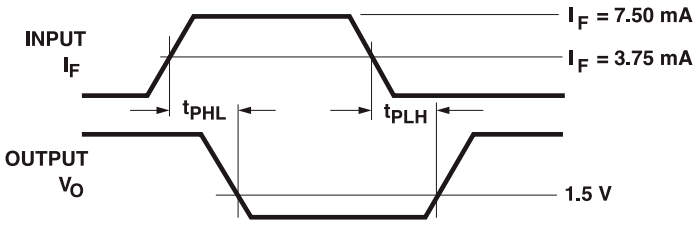
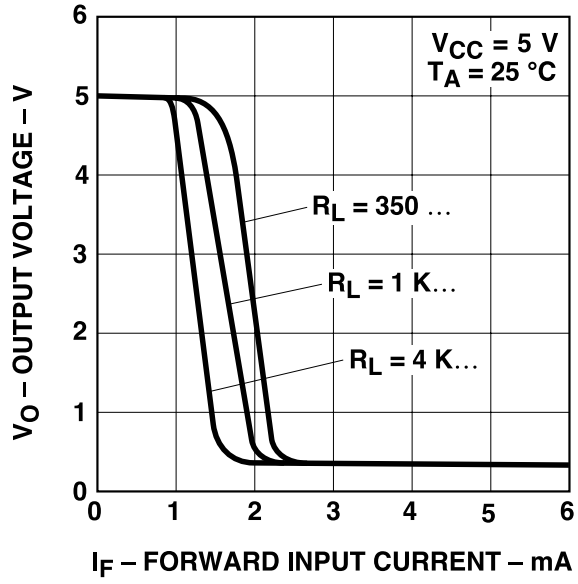
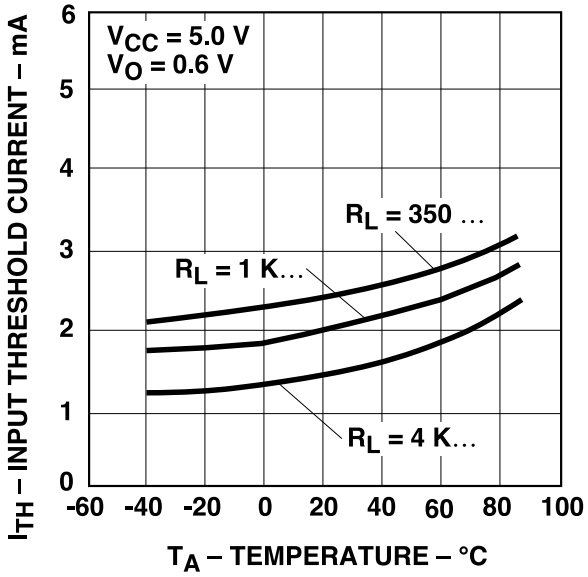


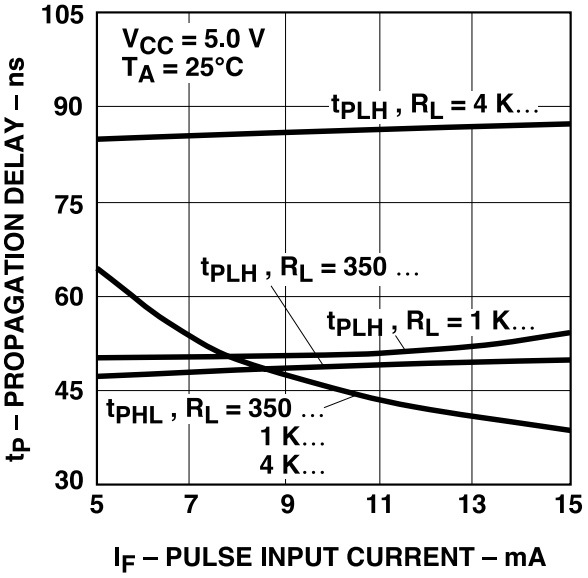
Figura 3.9: Diagrama de tiempos de la señal de entrada y salida [27].



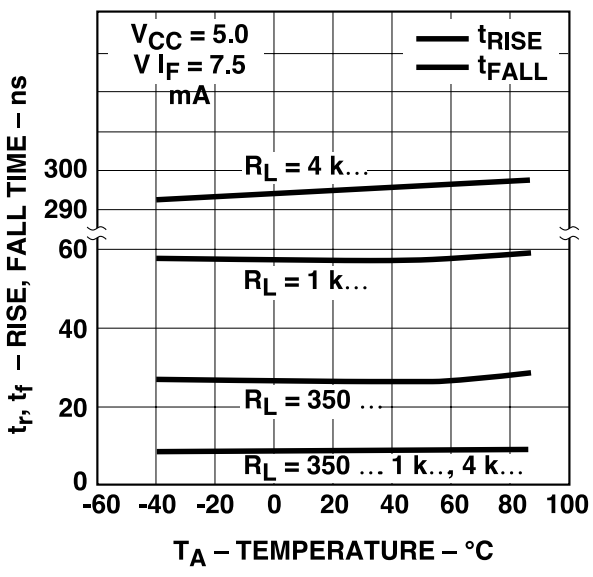
(a) Voltaje de salida vs Corriente de entrada



(b) Umbral de corriente de entrada vs Temperatura



(c) Retraso de propagación vs Pulso de corriente de entrada



(d) Tiempo de subida, bajada vs Temperatura

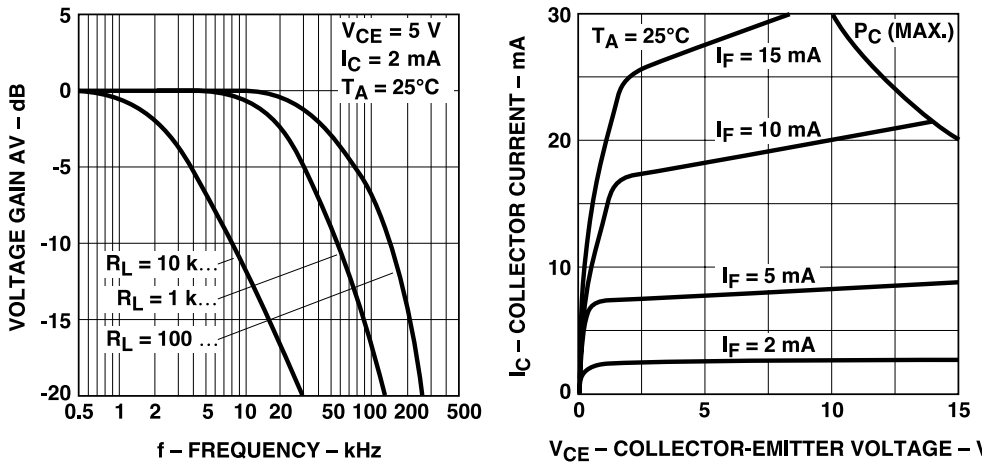
Figura 3.10: Gráficas del comportamiento del integrado HCPL-2630 respecto a R_L [27].

C.4.3 | Integrado 4N35

El 4N35 es un acoplador óptico para aplicaciones de propósito general. Contiene un diodo emisor de luz acoplada ópticamente a un fototransistor. El producto está envasado en un paquete DIP de 6 pines y disponible en opción espaciamiento widelead y la opción de SMD curva plomo. El tiempo de respuesta (t_r), es típicamente 3 microsegundos y CTR mínimo 100 % a la corriente de entrada de 10 mA. Para un tiempo de respuesta típico de 3μs $V_{CE} = 10$V, $I_C = 2$ mA, $R_L = 100\Omega$.

Especificaciones Técnicas	
Característica	Cantidad
Corriente en inversa	10μA
Voltaje de entrada	1.5V min
Voltaje de ruptura	6V
Resistencia entrada-salida	100GΩ

Tabla 3.9: Especificaciones del integrado 4N35 [28].

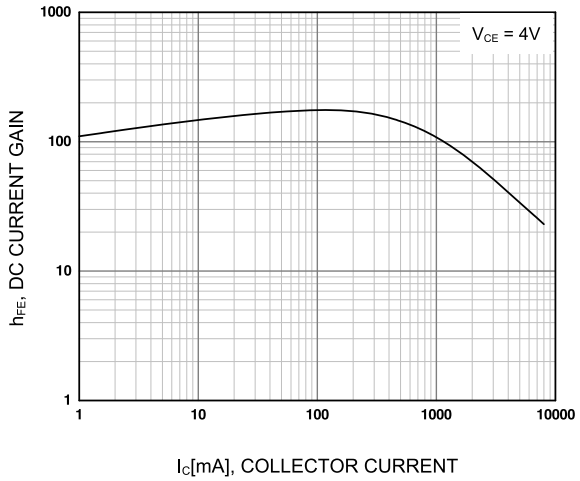


(a) Nivel alto

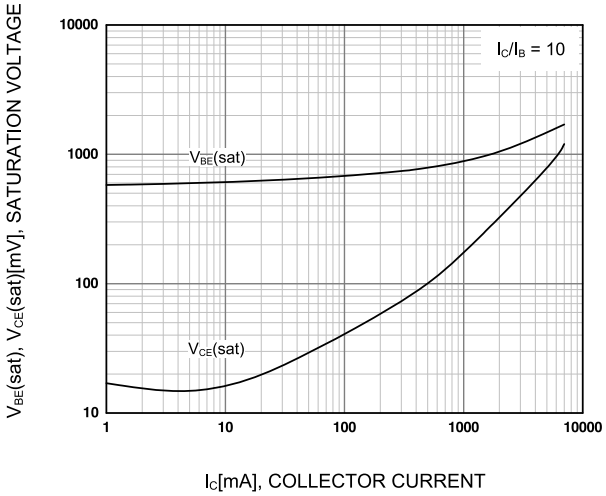
(b) Nivel bajo

Figura 3.11: Gráficas del comportamiento del integrado 4N35 respecto a R_L [28].

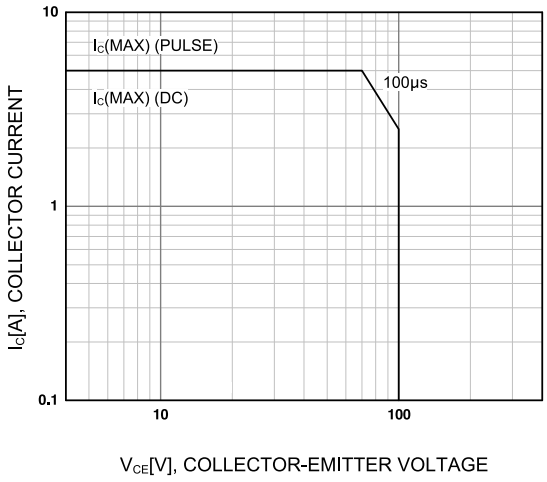
C.4.4 | Integrado TIP31C



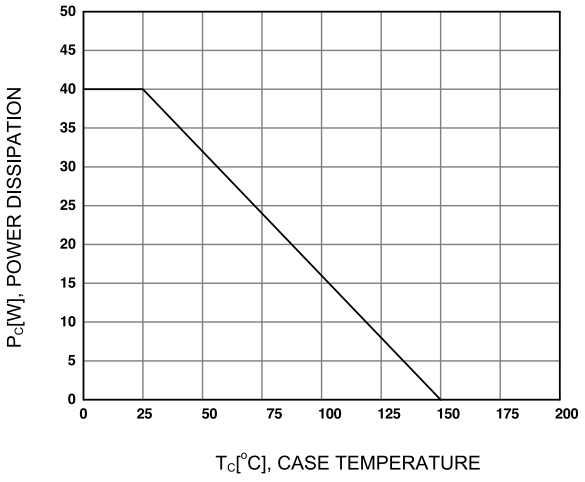
(a) Ganancia de corriente en DC



(b) Voltaje de saturación V_{BE} y V_{CE}



(c) Área de operación

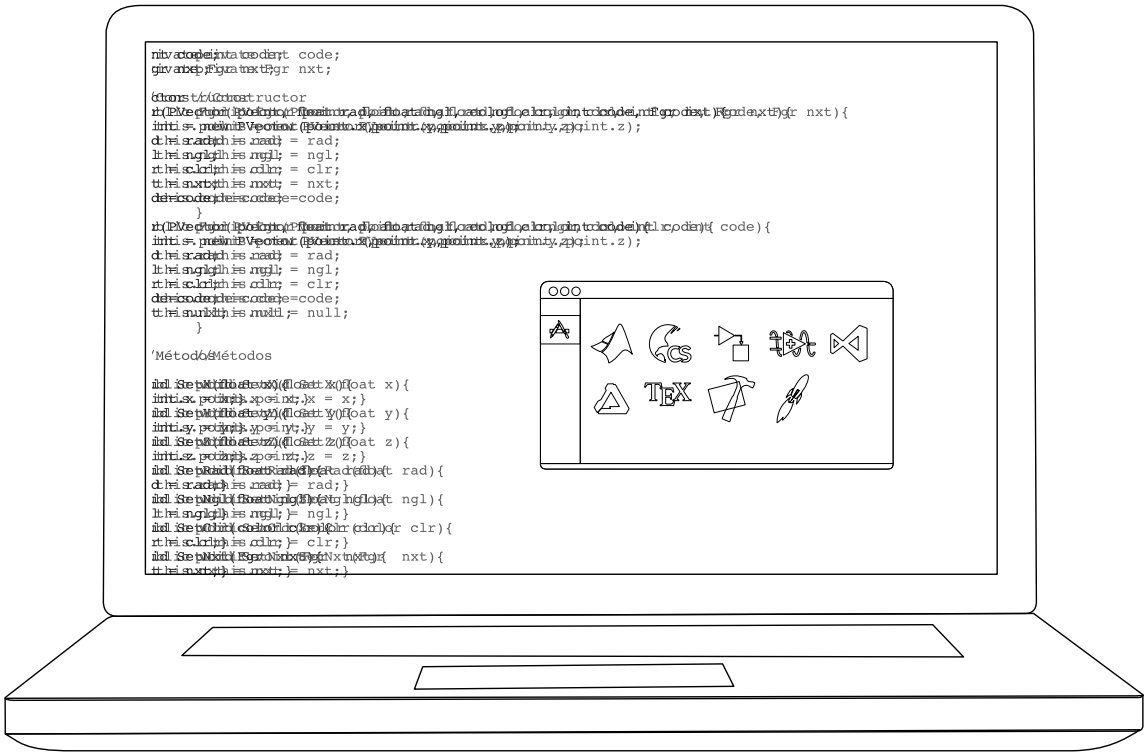


(d) Reducción de potencia

Figura 3.12: Gráficas del comportamiento típico del integrado TIP31 [29].

Apéndice D

Software y Código



D.1 | Creación de proyecto FPGA CompactRIO

En esta sección se redacta a detalle los pasos a seguir para la creación de un proyecto FPGA para el sistema *CompactRIO* con el objetivo de facilitar la elaboración de proyectos futuros. Es importante aclarar que el procedimiento es valido para la versión 2009 del software *LabVIEW* y en versiones posteriores a ésta los pasos pueden variar parcial o completamente. Si usted utiliza una versión posterior y durante el proceso existe algún inconveniente, verificar la página ***www.ni.com*** para una mejor asesoría.

- **Paso 1:** Iniciar el programa *LabVIEW* 2009 y esperar a que se despliegue en pantalla el menu principal (Figura 4.1). Elegir la opción `/New/More`.

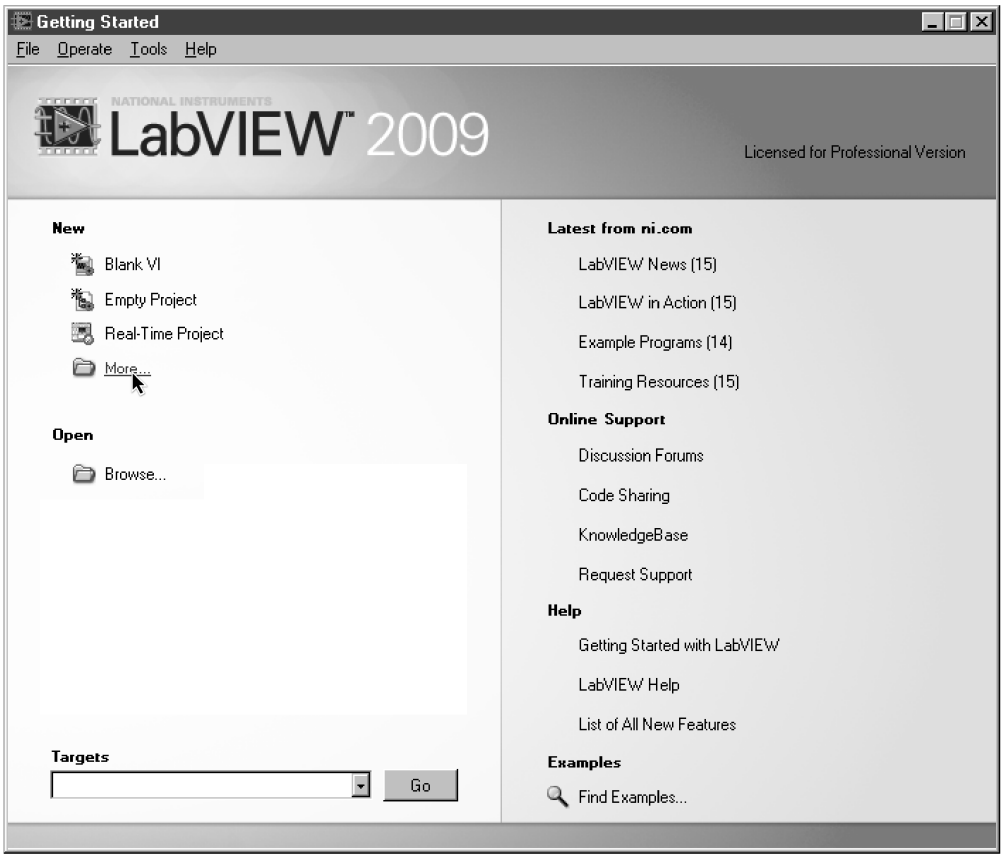


Figura 4.1: Menu principal

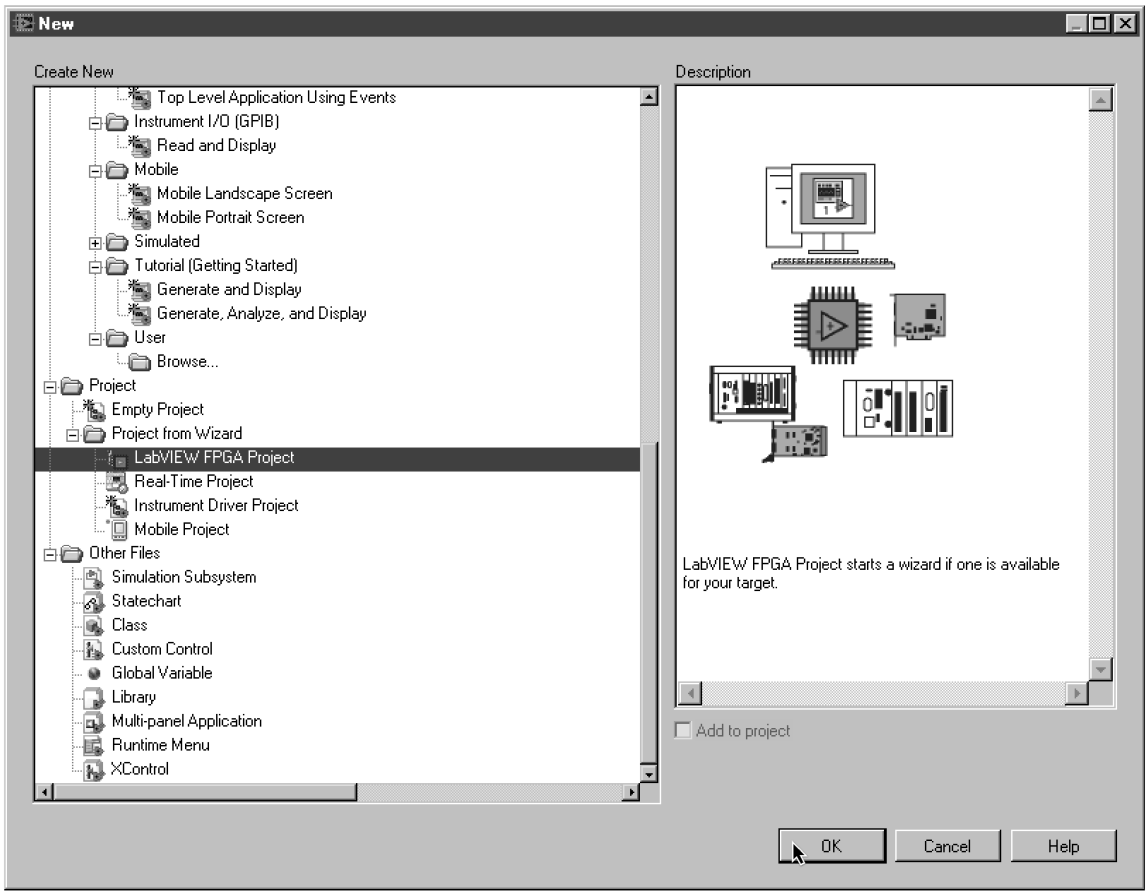


Figura 4.2: Sub-menu de nuevo elemento.

- **Paso 2:** Al desplegarse el sub-menu para la creación de nuevos proyectos (Figura 4.2) seleccionar la opción `/Project/Project from Wizard/LabVIEW FPGA Project` y dar click en el botón **OK**.

- **Paso 3:** Se continua por elegir el tipo de proyecto mediante la selección del tipo del sistema con el que se cuenta. Seleccionar la opción **CompactRIO Reconfigurable Embedded System** para sistemas *CompactRIO* modelo cRIO-9014 y dar click en el botón **Next** (Figura 4.3).

- **Paso 4:** Para identificar el sistema existente seleccionar **Discover existing system** y especificar la dirección *IP* (Figura 4.4). Ésta se encuentra especificada en la aplicación *NI Max* o se puede verificar en elementos del sistema directamente desde la computadora.

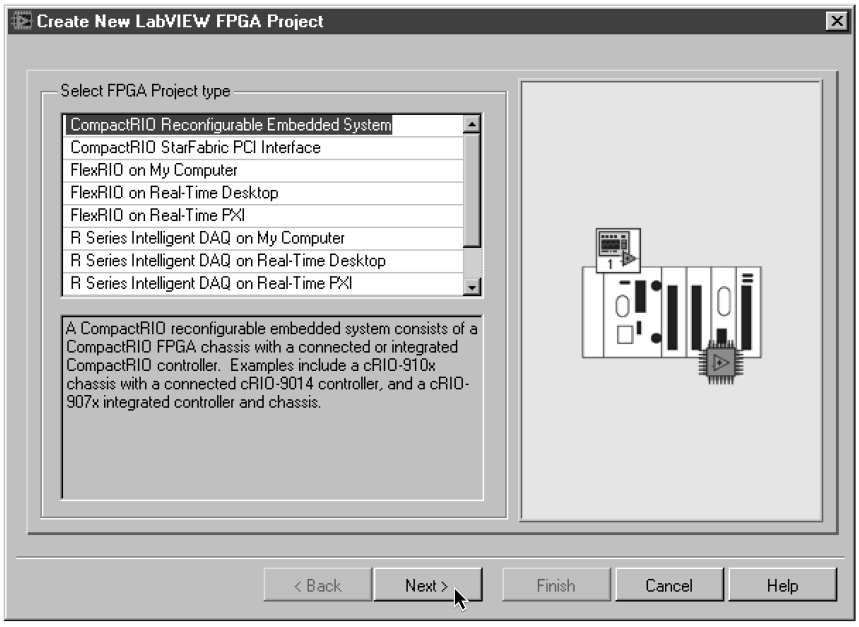


Figura 4.3: Ventana de selección de FPGA.

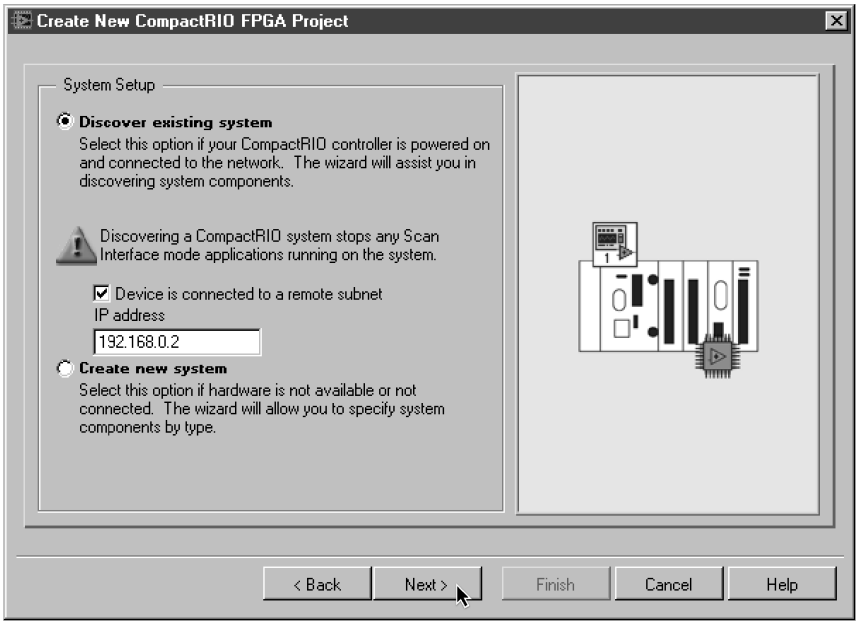


Figura 4.4: Ventana de creación de nuevo proyecto FPGA

nota: Es necesario en este punto que el sistema se encuentre en estado de encendido y conectado a la red, de lo contrario no será reconocido por el programa y aparecerán los siguientes mensajes de error.

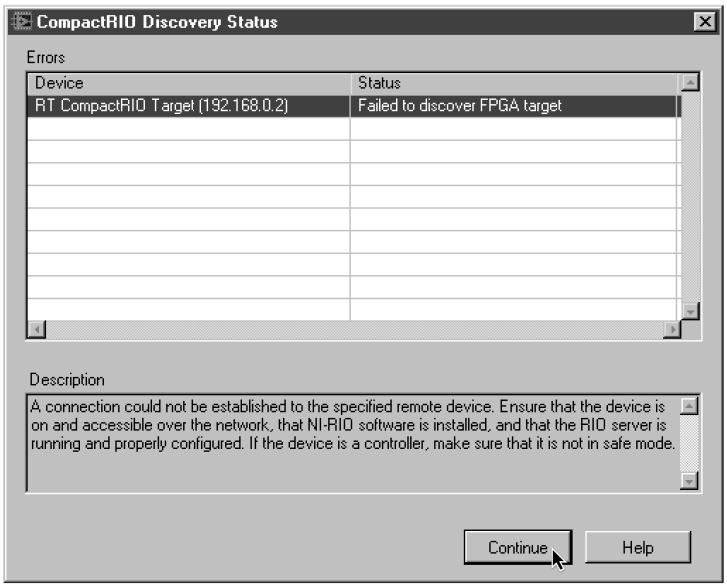


Figura 4.5: Ventana de estado del FPGA.

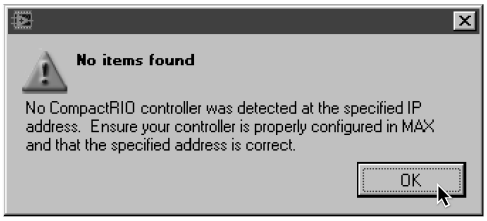


Figura 4.6: Mensaje de error por ausencia de FPGA.

Para resolver este problema basta con liberar las ventanas de las Figuras 4.5 y 4.6, presionando **Continue** and **OK** respectivamente y conectar el FPGA a la alimentación y/o a la red.

- **Paso 5:** Si el sistema ya había sido utilizado se mostrará en pantalla el mensaje de la Figura 4.7 y se deberá seleccionar la opción **Discover**.

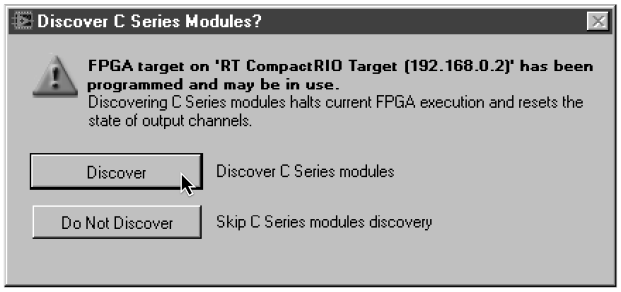


Figura 4.7: Mensaje de advertencia de FPGA usado.

Si el proceso fue exitoso aparecerá una última ventana (Figura 4.31) para la confirmación del proyecto donde se puede elegir **Finish** para finalizar la creación del proyecto, **Cancel** para salir del modo de creación y **Help** para obtener ayuda adicional.

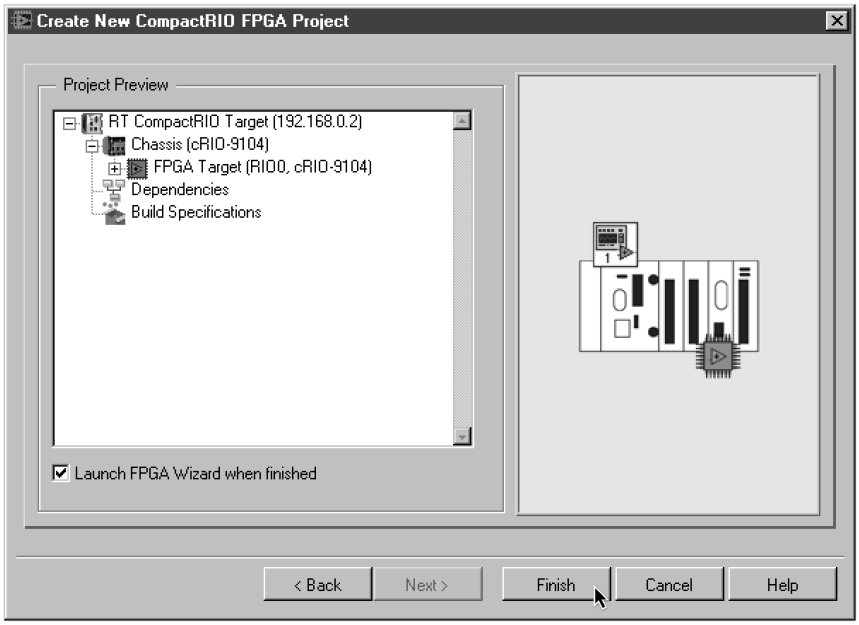


Figura 4.8: Ventana final para la creación del proyecto.

La opción **Launch FPGA Wizard when finished** en la Figura 4.31, ejecuta un ayudante para la generación de código pre-establecido y puede ser des-habilitado antes o después de finalizar el proyecto.

Al finalizar el procedimiento se muestra el proyecto creado bajo el nombre *Untitled* (Figura 4.9), se recomienda antes de continua guardar y renombrar al proyecto. Para añadir código al proyecto se debe presionar click derecho sobre la sección `/Project/RT CompactRIO Target/Chasis/FPGA Target(RIO0, cRIO-9104)` y seleccionar *añadir nuevo VI.*

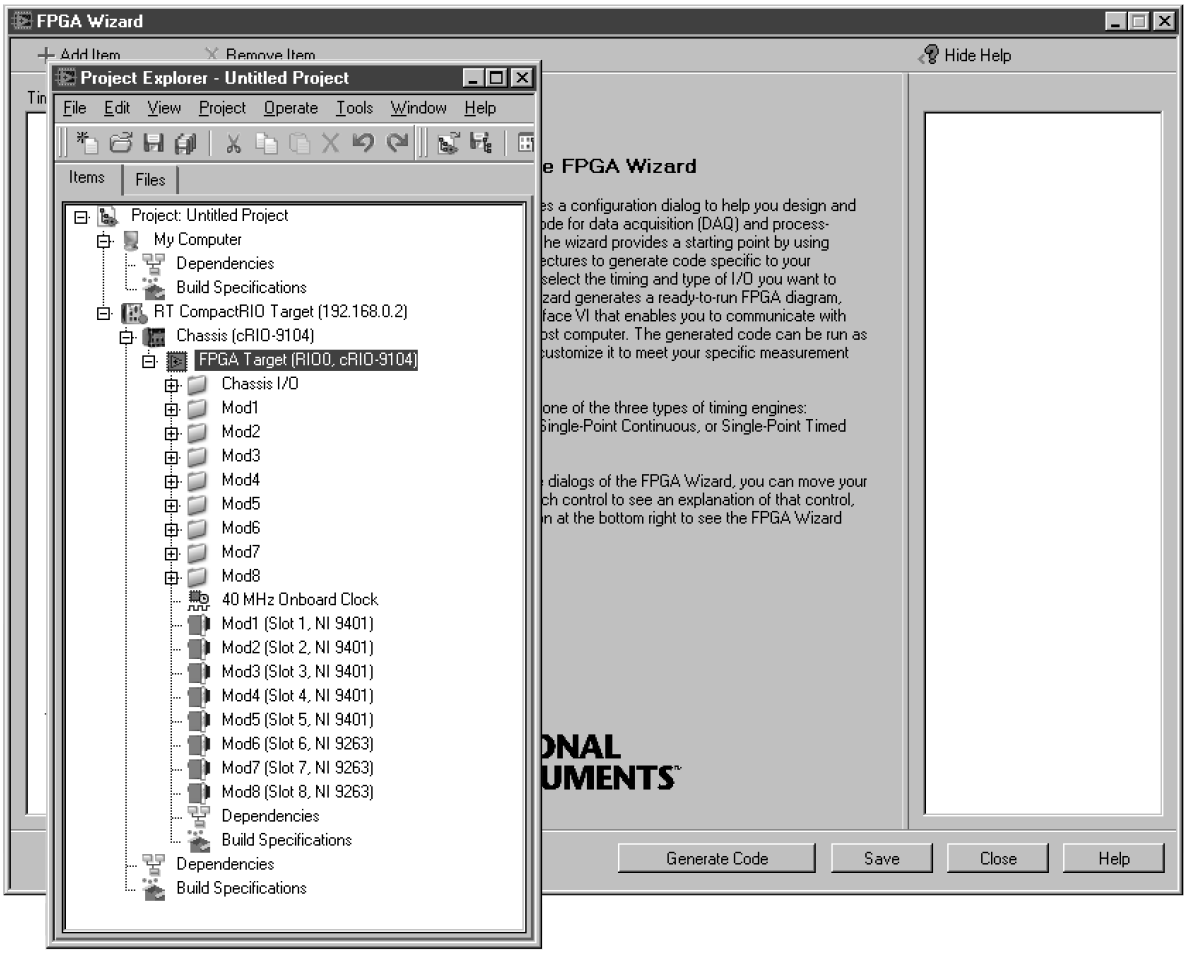


Figura 4.9: Ventana de proyecto creado.

D.2 | Proyecto en LabVIEV para la adquisición de datos y escritura de señales analógicas

A continuación se muestran los sub programas en lenguaje gráfico que se realizaron para la adquisición y escritura de señales para el FPGA acompañadas de su diagrama de flujo.

Figura 4.10: Diagrama de flujo del sub-vi *Limits*

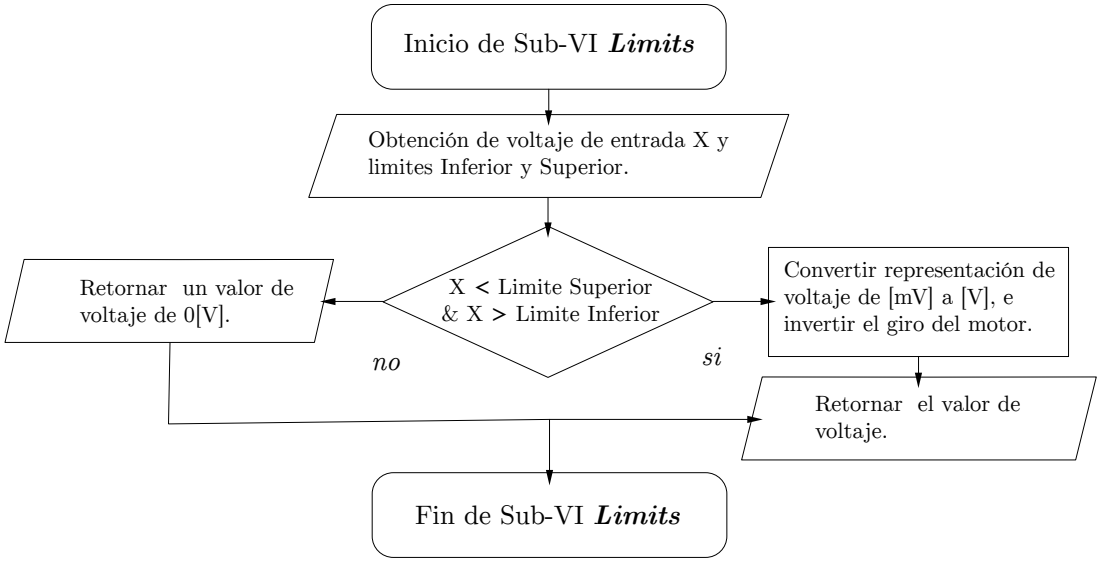
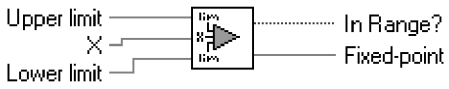
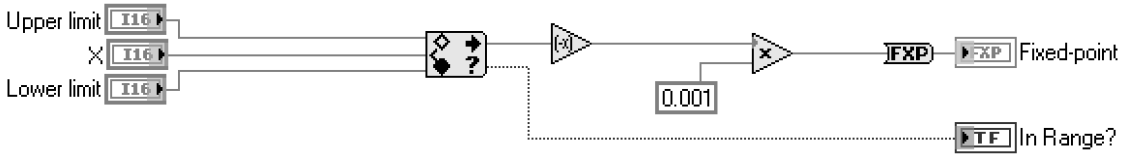


Figura 4.11: Sub-VI *Limits*



(a) Icono y conector del panel



(b) Diagrama a bloques del programa

Figura 4.12: Diagrama de flujo del sub-vi *Increase*

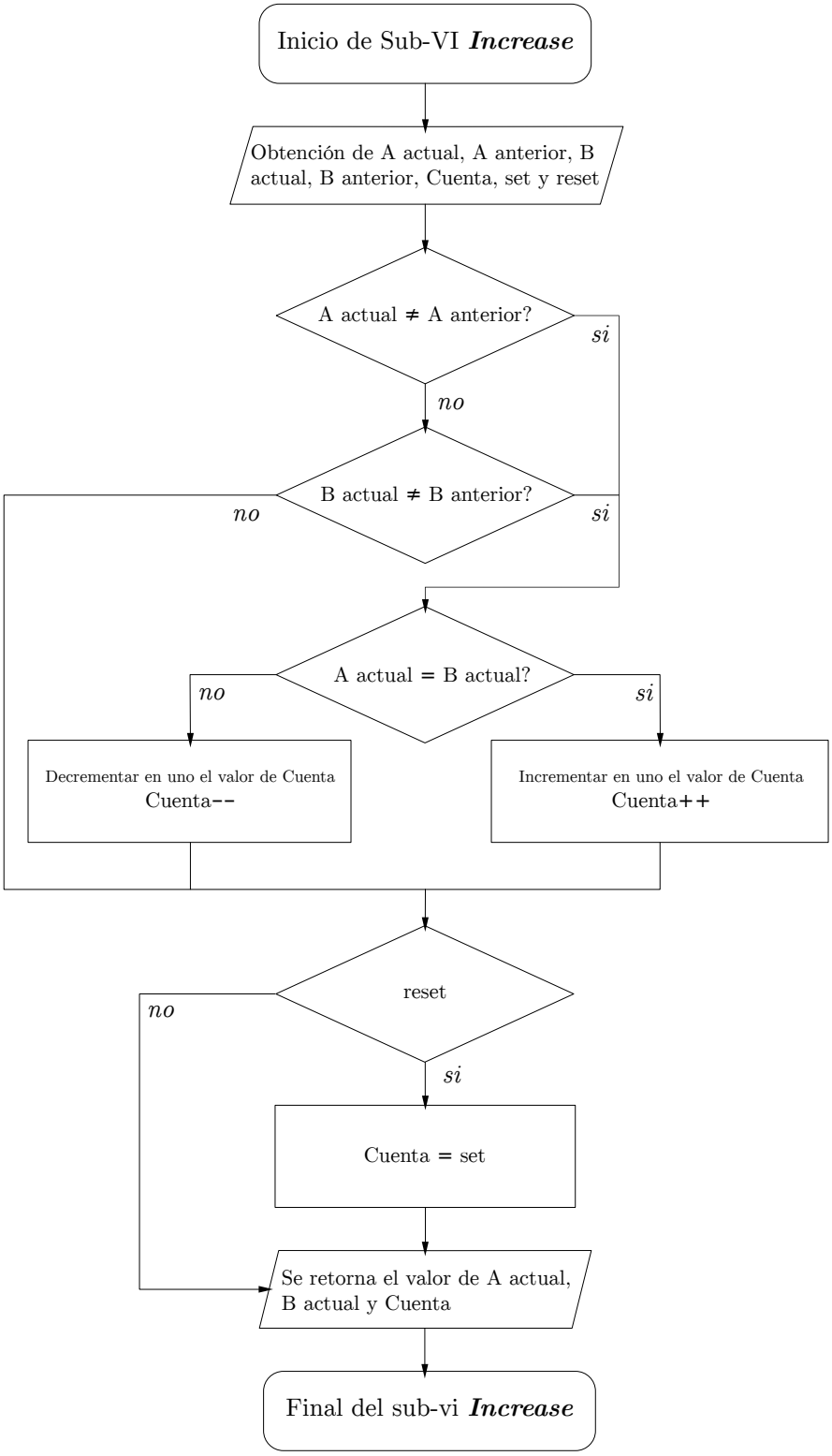
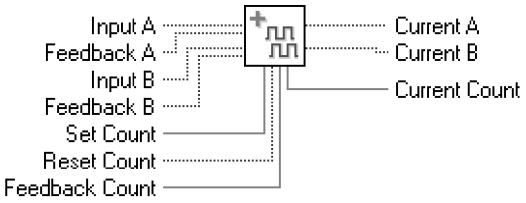
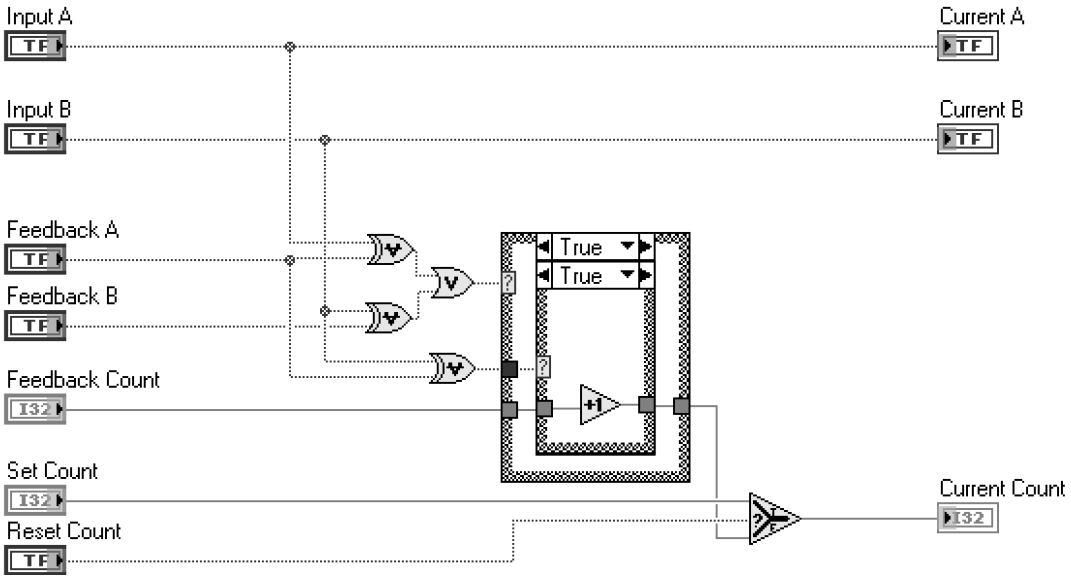


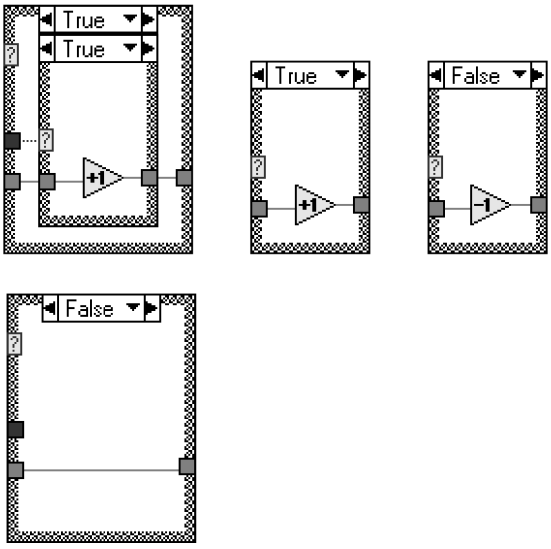
Figura 4.13: Sub-VI *Counter Increase*



(a) Icono y conector del panel



(b) Diagrama a bloques del programa



(c) Casos ocultos

Figura 4.14: Diagrama de flujo del sub-vi *Decrease*

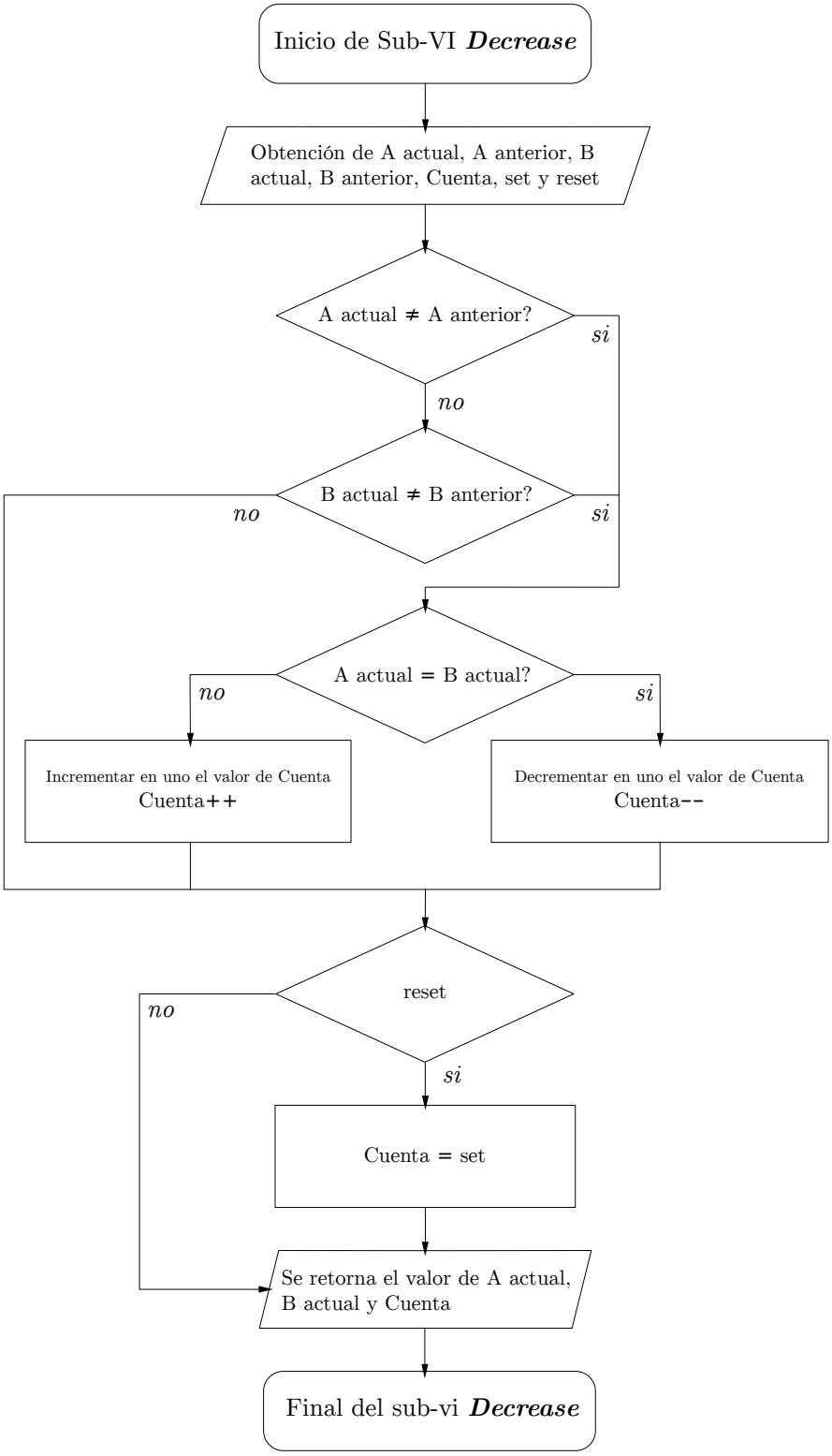
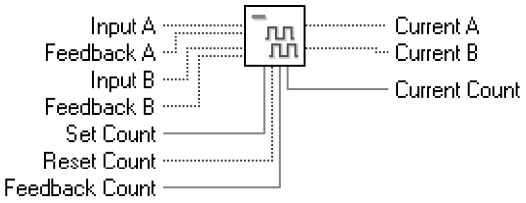
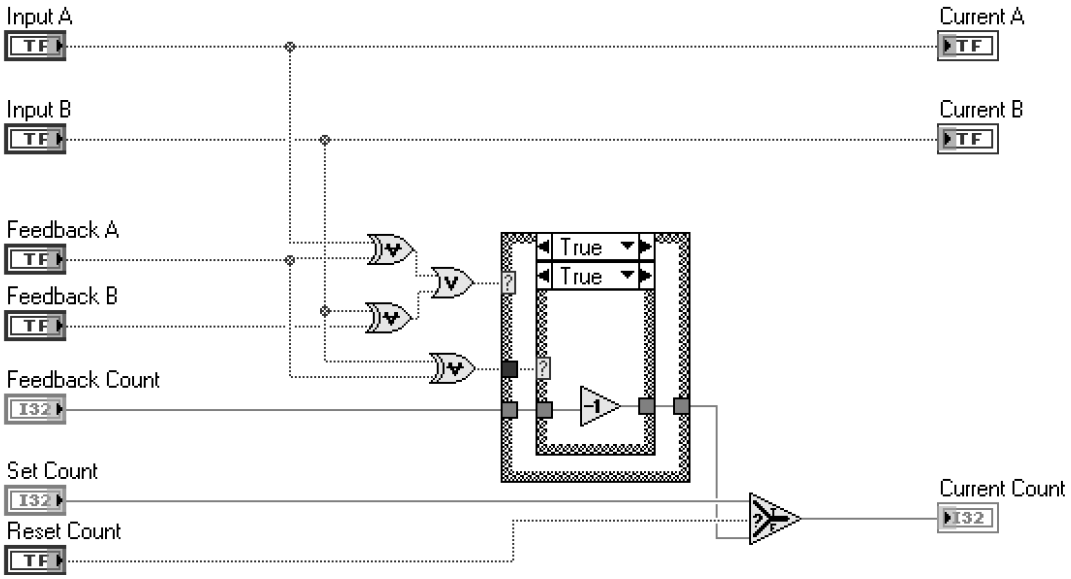


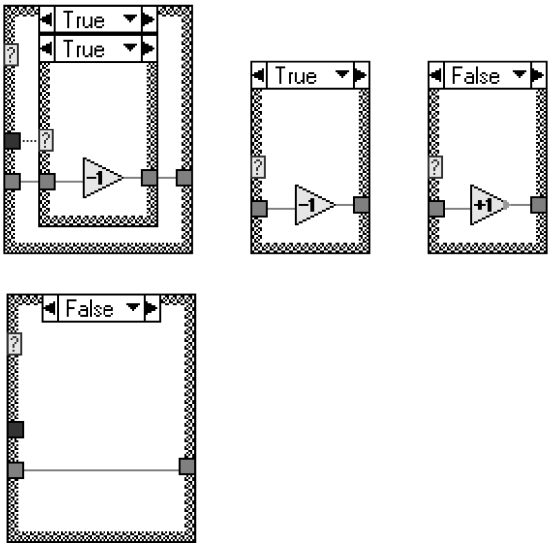
Figura 4.15: Sub-VI *Counter Decrease*



(a) Icono y conector del panel



(b) Diagrama a bloques del programa



(c) Casos ocultos

Figura 4.16: Diagrama de flujo del sub-vi *Counter Z*

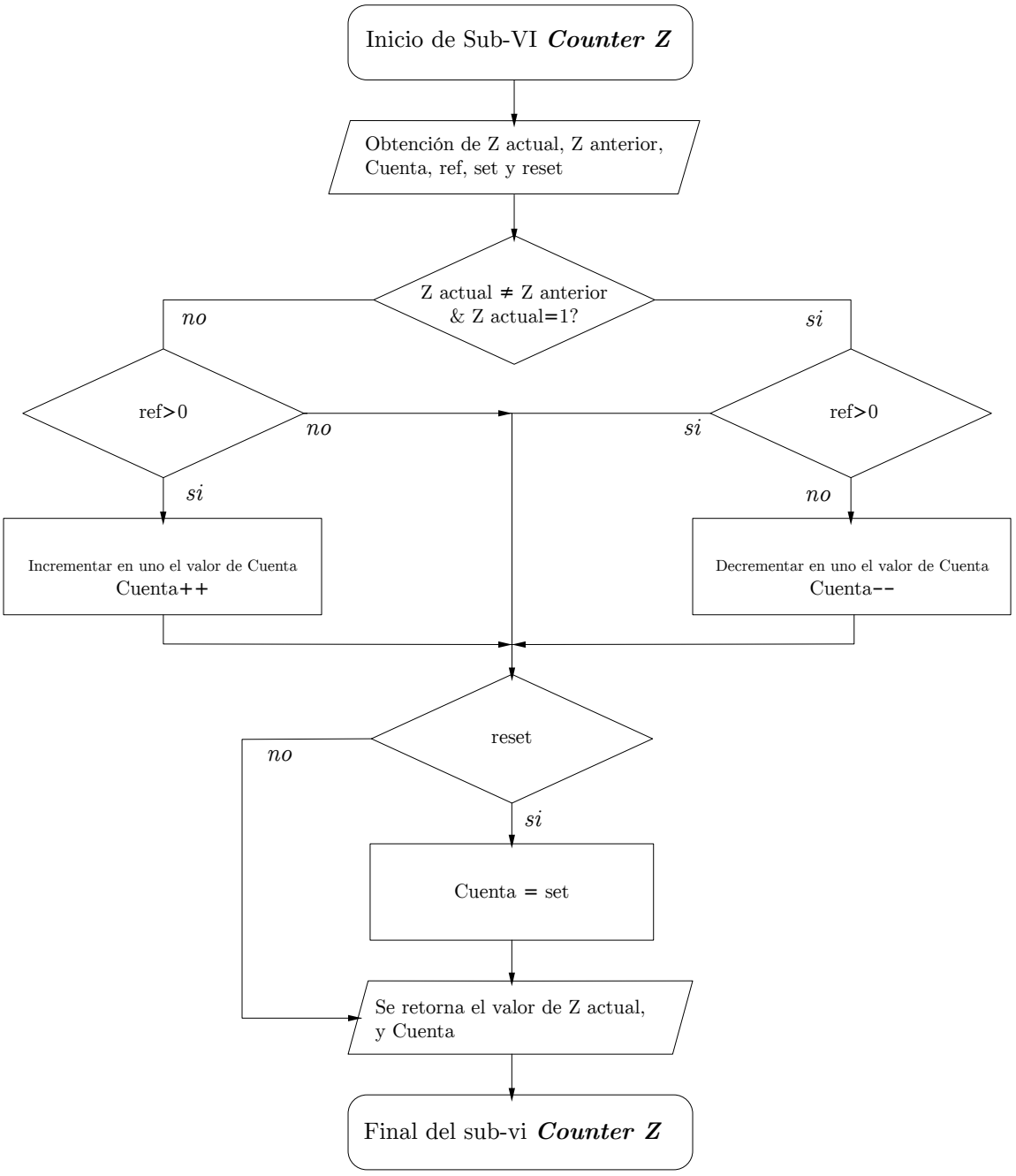
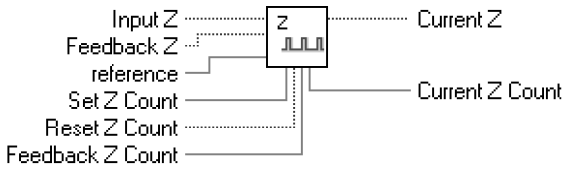
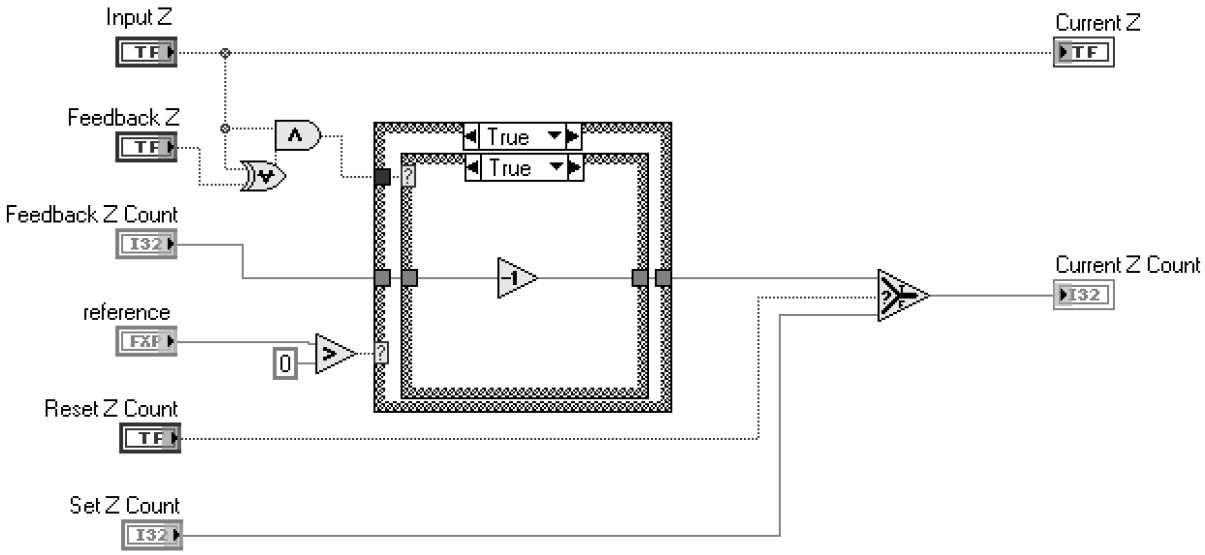


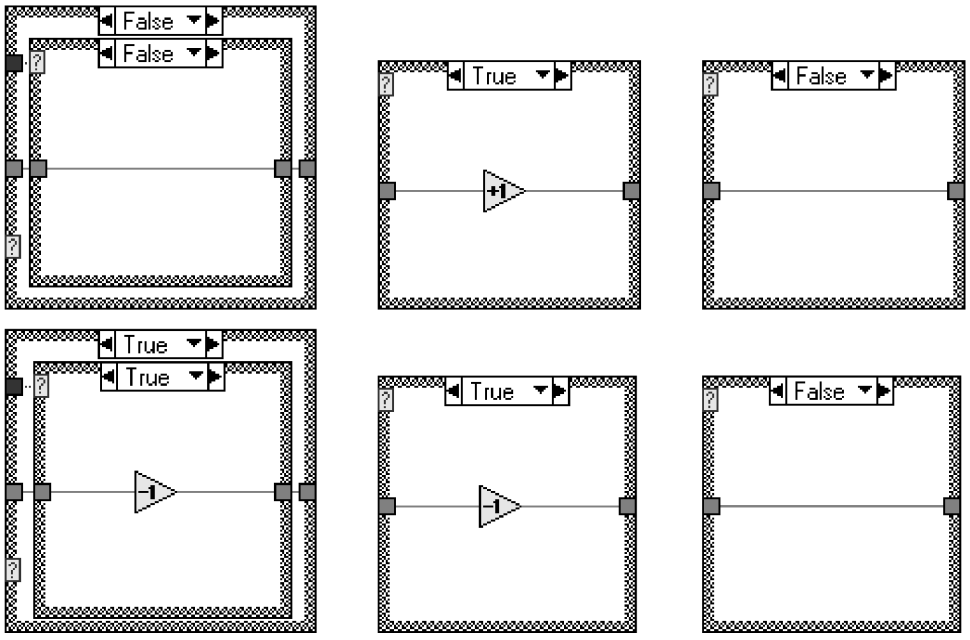
Figura 4.17: Sub-VI *Z Counter*



(a) Icono y conector del panel



(b) Diagrama a bloques del programa



(c) Casos ocultos

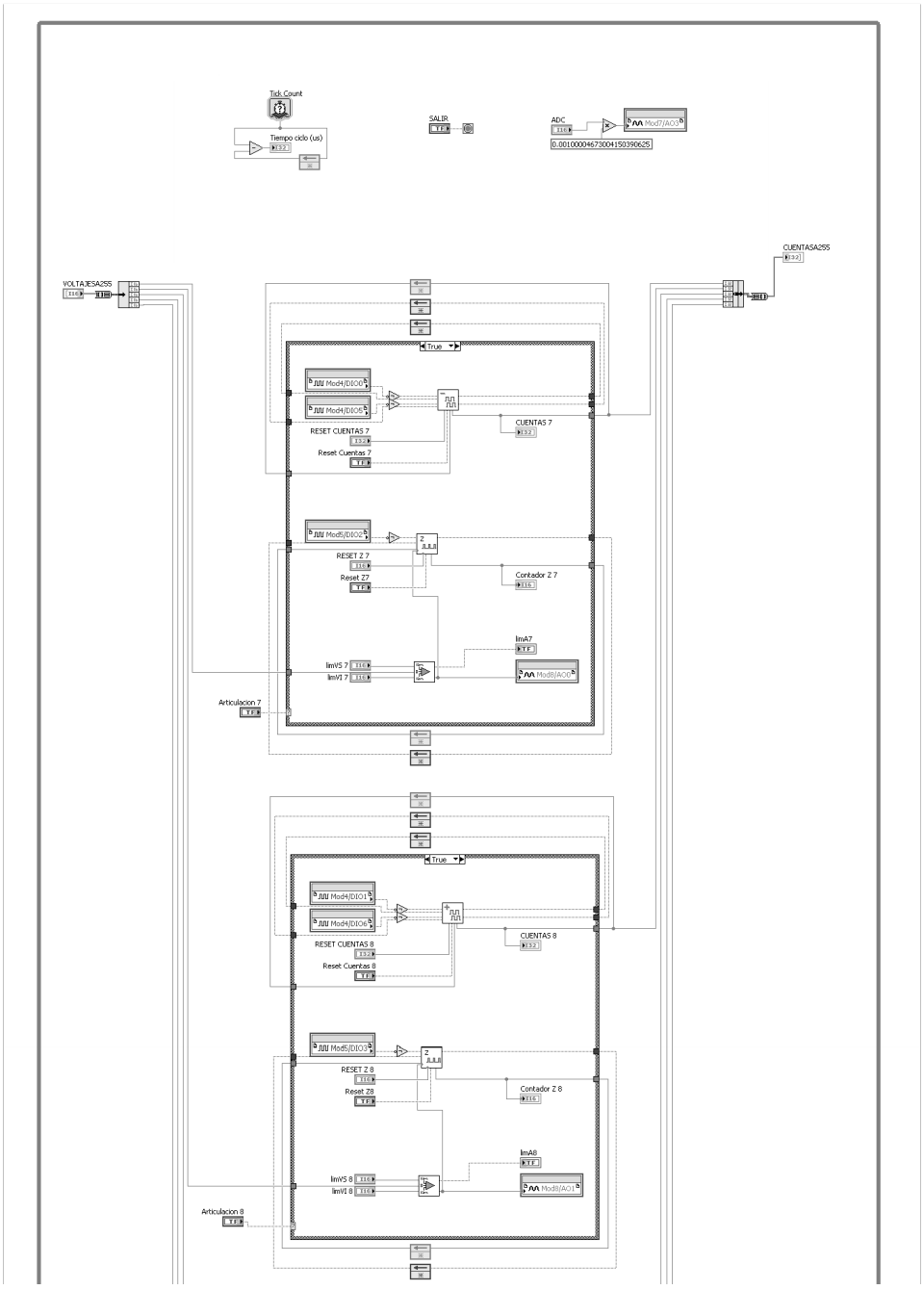


Figura 4.18: Programa para la interpretación de datos y escritura de señales analógicas parte I

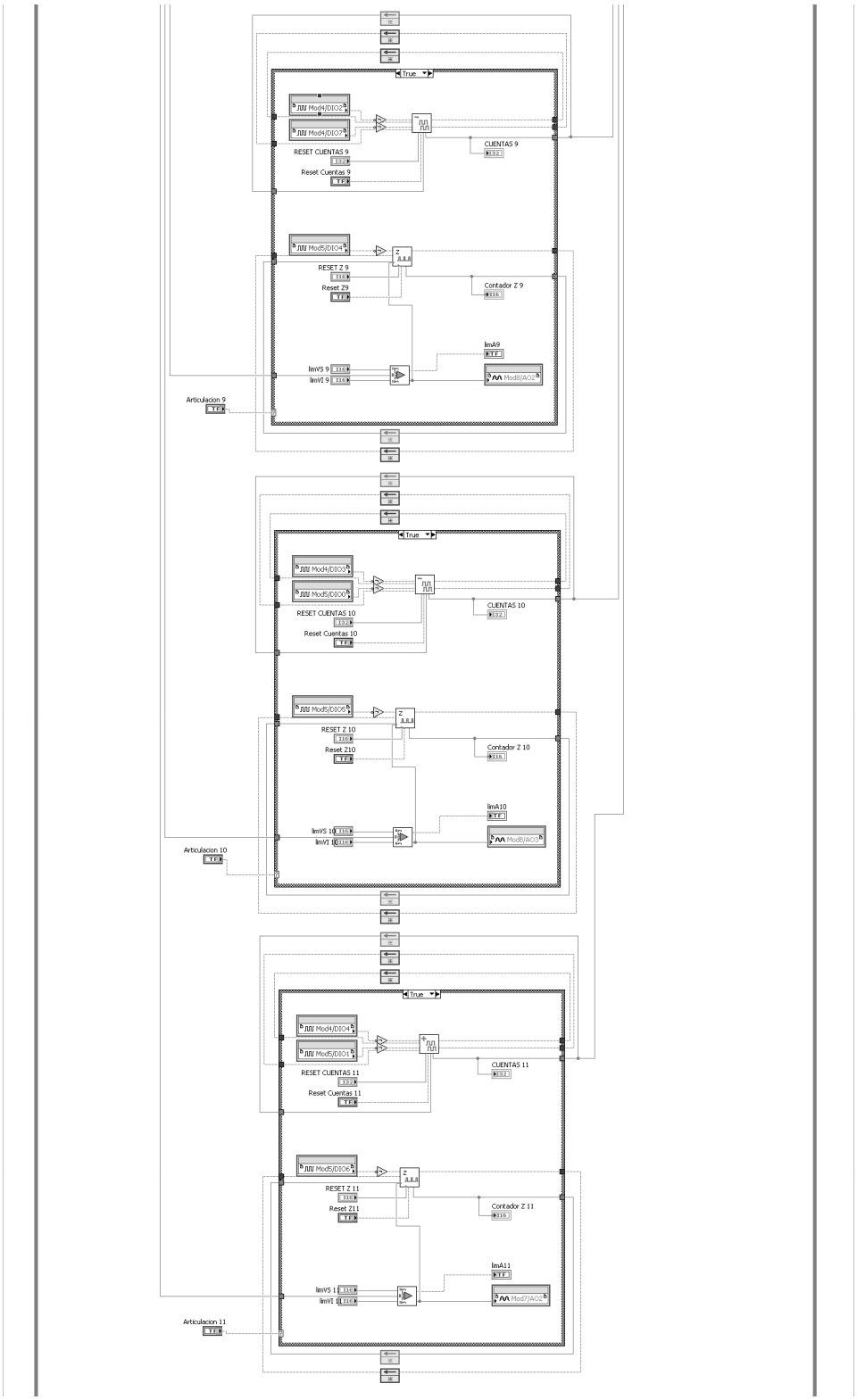


Figura 4.19: Programa para la interpretación de datos y escritura de señales analógicas parte II

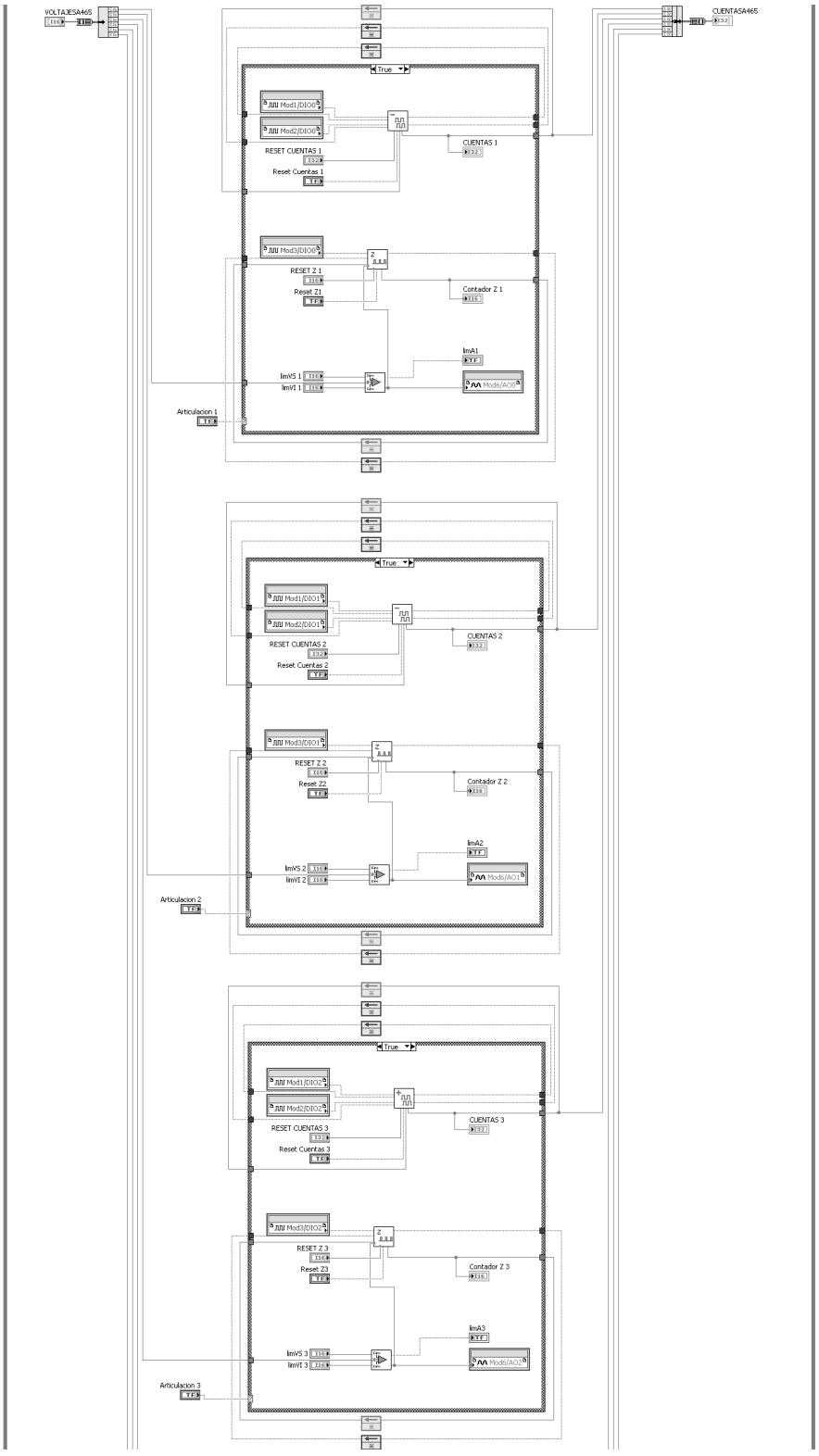


Figura 4.20: Programa para la interpretación de datos y escritura de señales analógicas parte III

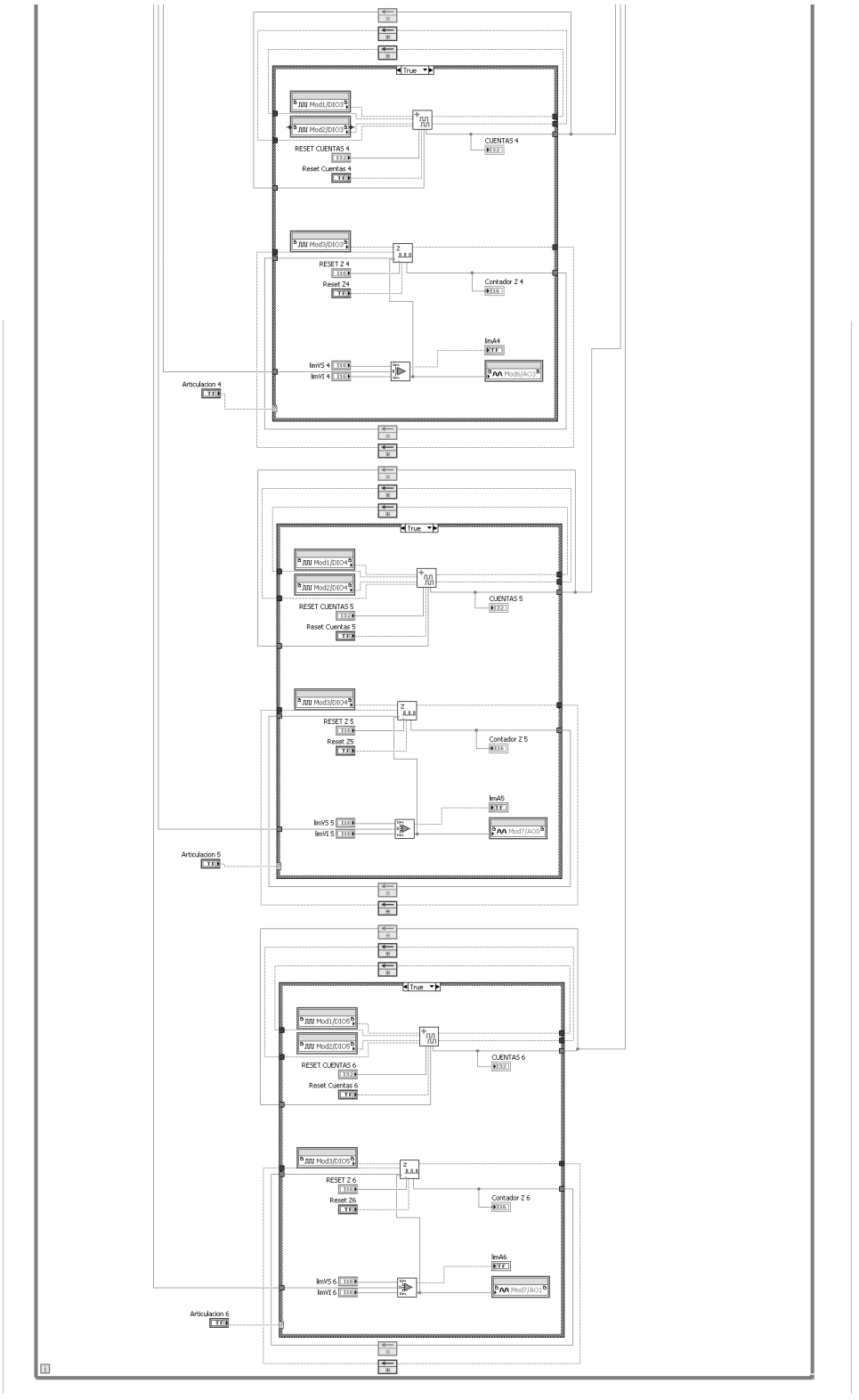


Figura 4.21: Programa para la interpretación de datos y escritura de señales analógicas parte IV

D.3 | Creación de proyecto en *Visual Studio*

Para poder realizar un proyecto en Visual Studio que se pueda comunicar con el FPGA *CompactRIO* se siguieron los siguientes pasos.

- **Paso 1**: Se selecciona el menu de nuevo proyecto y se se elige dentro de las plantillas instaladas la opción MFC Application conocido también como *Microsoft Foundation Classes*, este tipo de de proyecto conjunta gran parte de las librerías de *Windows API* en una clases en lenguaje *C++*.

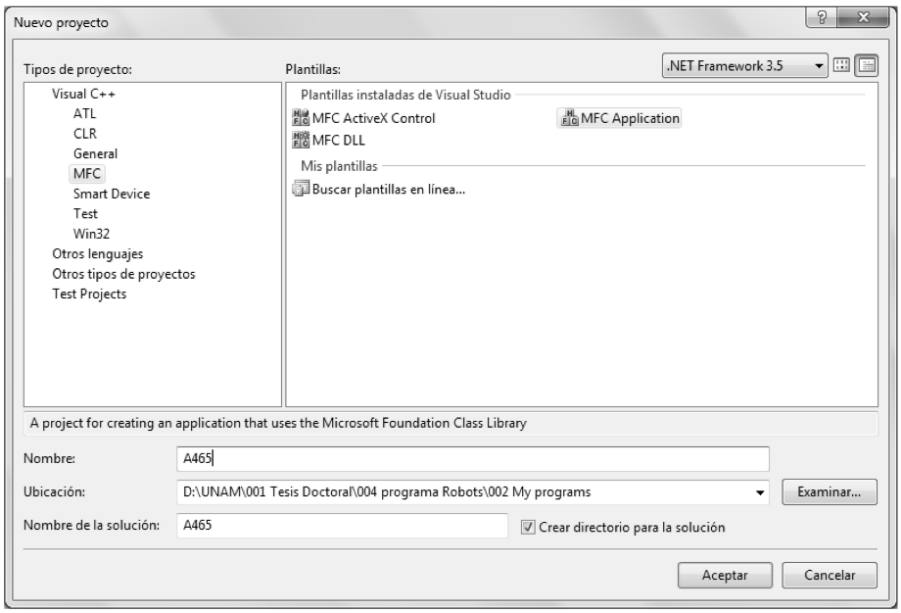


Figura 4.22: Menu de nuevo proyecto.

Se desplegara en pantalla un ayudante para realizar el proyecto como se muestra en la captura de pantalla de la Figura 4.23. Seleccionar la opción **Next**.

- **Paso 2**: El ayudante mostrará cuatro pasos más para personalizar el proyecto, estos son: tipo de aplicación, base de datos, características de interfaz de usuario, características avanzadas y clases generadas. Se recomienda seleccionar las opciones igual que en las Figuras 4.23, 4.24, 4.25, 4.26 y 4.27.

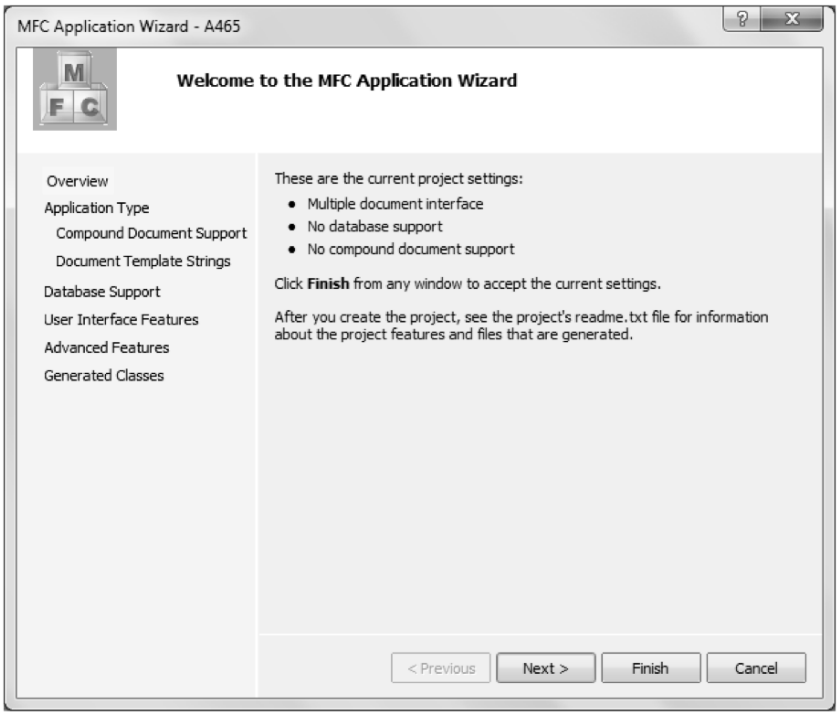


Figura 4.23: Overview MFC Application Wizard.

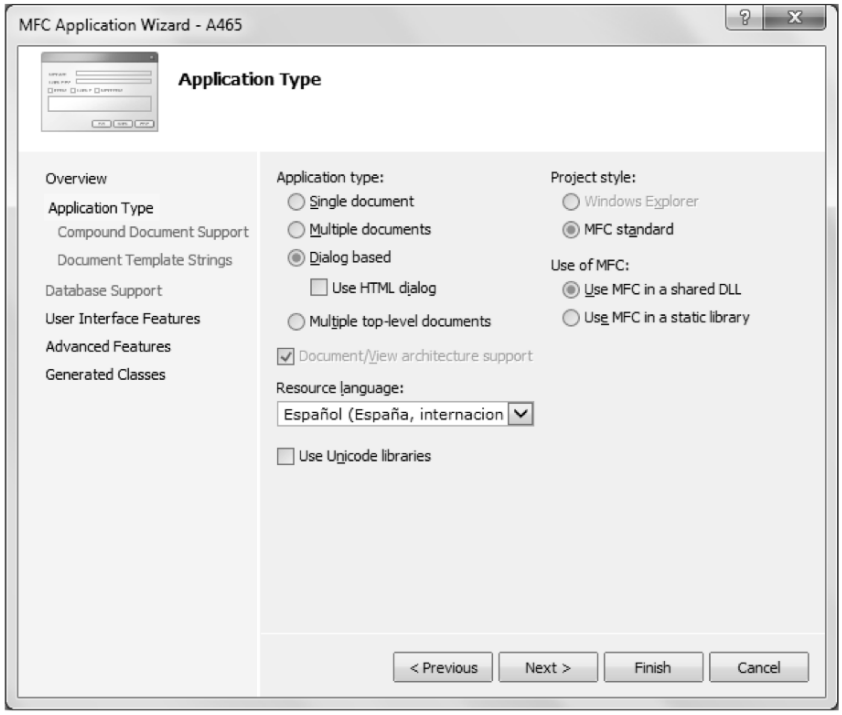


Figura 4.24: Appication Type MFC Application Wizard.

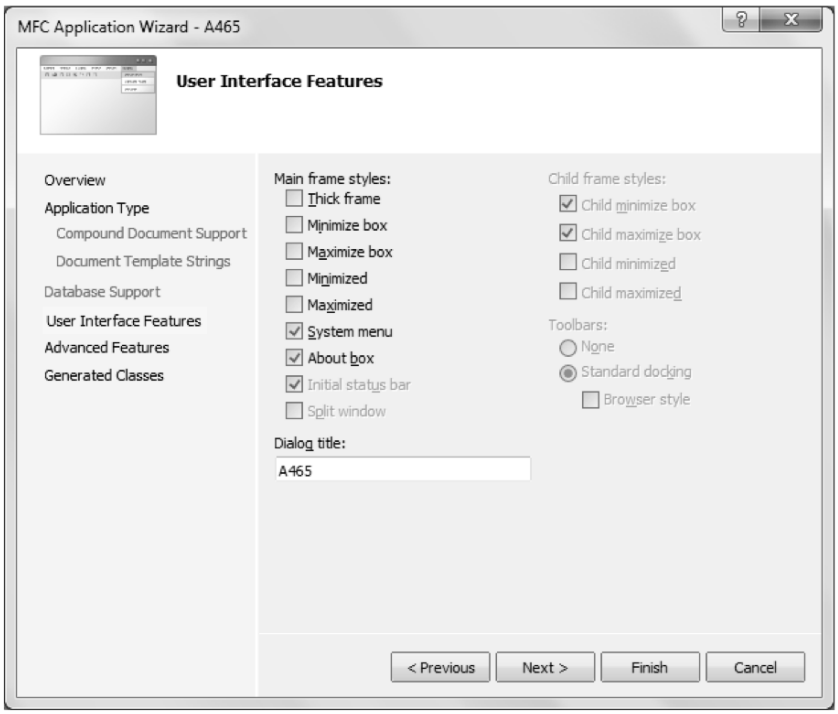


Figura 4.25: User Interface Features MFC Application Wizard.

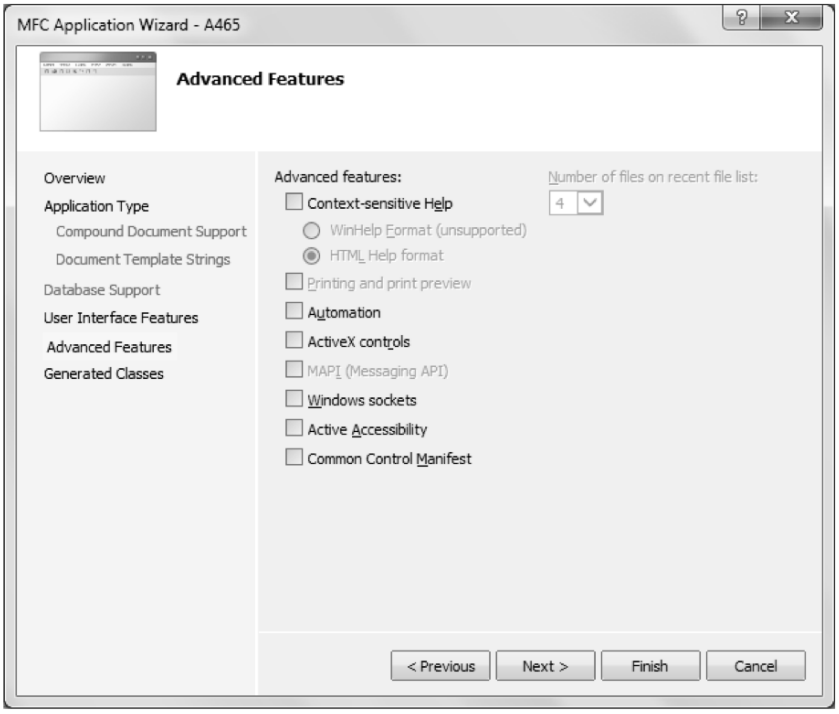


Figura 4.26: Advanced Features MFC Application Wizard.

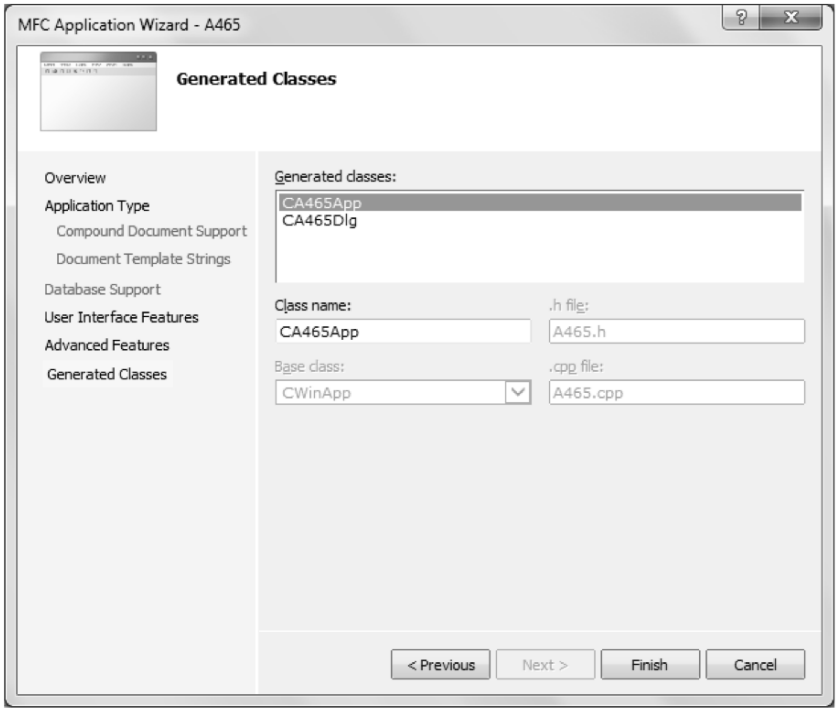


Figura 4.27: Generated Classes MFC Application Wizard.

- **Paso 3**: Una vez generado el proyecto se tienen que ingresar los siguientes archivos al programa, tal y como se muestra en las Figuras 4.28 y 4.29.

`APICRIO.h`: Archivo creado para reducir el tamaño del nombre en las funciones del FPGA.

`NiFpga.c`: Archivo generado desde el proyecto en LabVIEW.

`NiFpga.h`: Archivo generado desde el proyecto en LabVIEW.

`NiFpga_final.h`: Archivo generado desde el proyecto en LabVIEW.

`WINMM.lib`: Librerías en *C* el uso de *timers*, algunas veces no es necesario ingresarla al proyecto ya que esta se encuentra dentro del sistema operativo.

`FINAL.lvproj_FPGA Target_modulos.vi.lvbitx`: Se sugiere añadir este archivo al proyecto.

nota: Para generar los archivos de *C* desde el programa *LabVIEW* basta con abrir el proyecto, presionar con click derecho el *VI* principal y seleccionar **Generate API files**, se desplegará el menú de la Figura 4.30. Seleccionar carpeta y generar librerías.

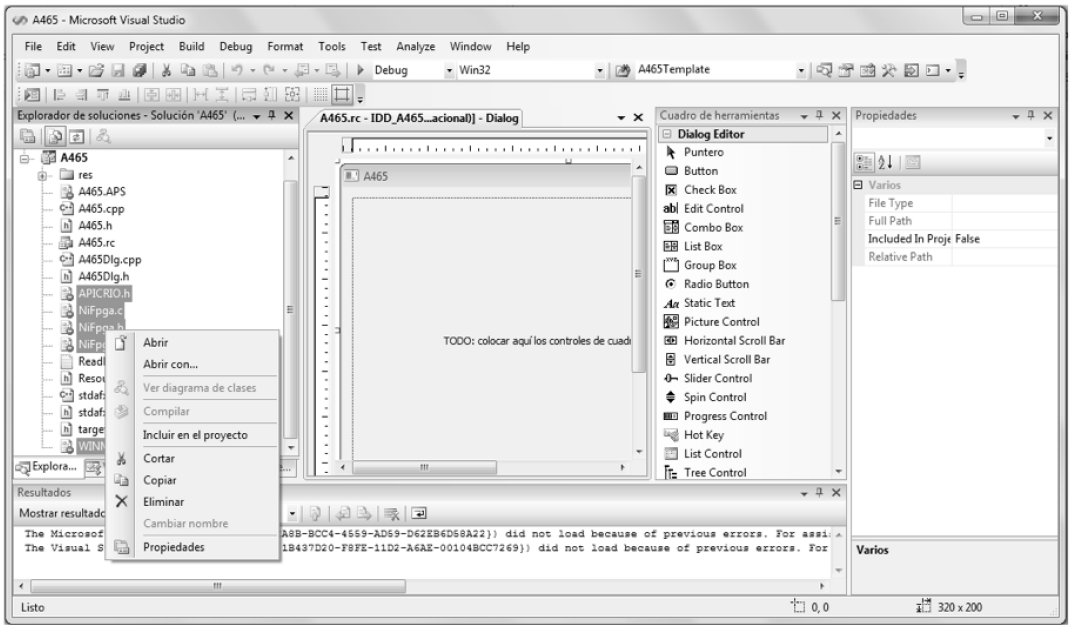


Figura 4.28: Incluir archivos al proyecto.

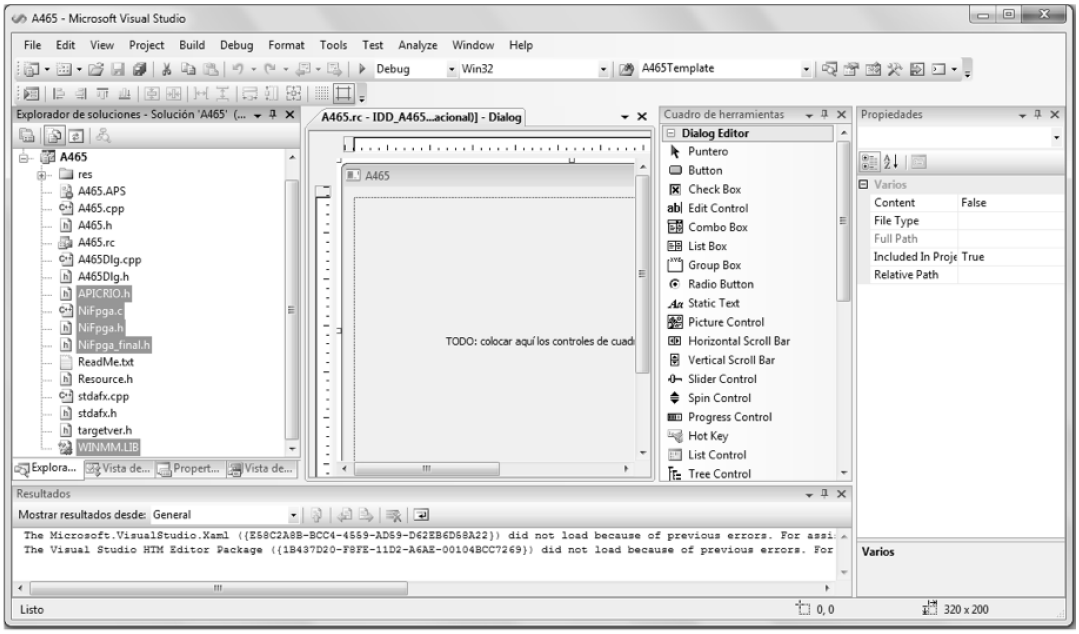


Figura 4.29: Proyecto con archivos correctamente agregados.

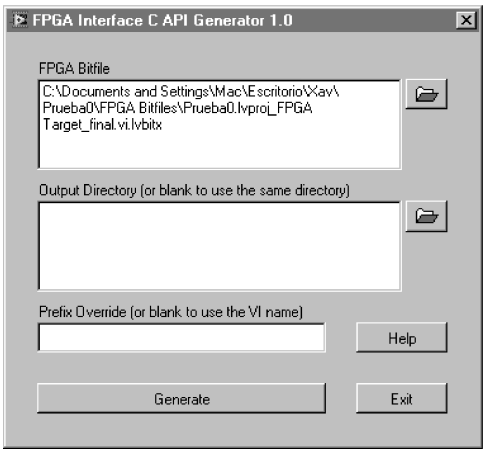


Figura 4.30: Ventana para generar API en *LabVIEW*.

- **Paso 4**: En algunas ocaciones se encuentra inhabilitado el uso de Headers pre-compilados por default, en caso de que ésta opción se encuentre habilitada no compilara el proyecto. Para inhabilitarla se deberá ir a `/preferencias del proyecto/C-C++/Precompiled Headers` y seleccionar la opción **Not Using Precompiled Headers**.

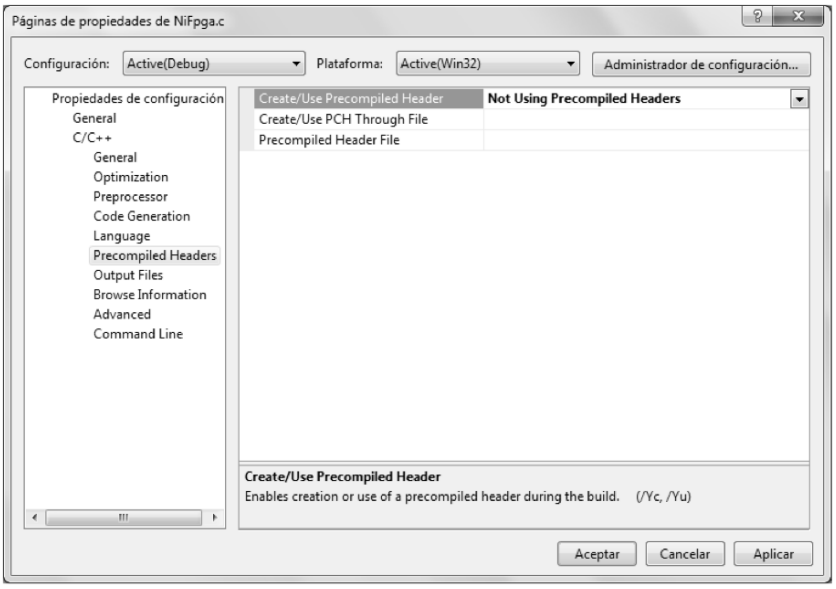


Figura 4.31: Ventana final para la creación del proyecto.

D.4 | Código para la Interfaz Gráfica

En este apartado se encuentra parte del código que se escribió para generar la interfaz gráfica. Debido a su extensión, únicamente se agrego la parte más significativa de éste.

```
// RobotsDlg.cpp

#include "stdafx.h"
#include "Robots.h"
#include "RobotsDlg.h"
#include "math.h"

// LibrerÌas del FPGA
#include "NiFpga_final.h"
#include "APICRIO.h"

// Librerías Win32
#include "Mmsystem.h"
#pragma comment(lib, "winmm.lib")

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// Timers
UINT_PTR TimerHome;


// Constantes
    const double pi = 3.14159265358979323846264338327 95;

    //Factor de conversion volts/torque
    const double vpn_A465[6] = {00.044755, 00.044755, 00.044755,
                                000.50439, 00.509434, 01.742776};
    const double vpn_A255[5] = {0.5075884, 0.5075884, 0.5075884,
                                02.284148, 04.568296};

    //Factor de conversion volts/salida
    const double DACR=1000;

    // Grados por cuenta
    const double gpcA255[5] = {0.0012500, 0.00125, 0.00125,
                               -0.0056250, 0.0143};
    const double gpcA465[6] = {0000.0009, 00.0009, 00.0009,
                               9.0/5050.0, 00.0018, 0.0018};

    // Radianes por cuenta
    const double rpcA255[5] = {gpcA255[0]*pi/180.0, gpcA255[1]*pi/180.0,
                               gpcA255[2]*pi/180.0, gpcA255[3]*pi/180.0,
                               gpcA255[4]*pi/180.0};
    const double rpcA465[6] = {gpcA465[0]*pi/180.0, gpcA465[1]*pi/180.0,
                               gpcA465[2]*pi/180.0, gpcA465[3]*pi/180.0,
                               gpcA465[4]*pi/180.0, gpcA465[5]*pi/180.0};

    // Tiempo de muestreo
    double T = 0.005;
    long  ti = 0;


// Variables
    //Tiempo de muestreo
    double t = 0.0;

    // Vector de posicion
    double q_A465[6] = {0.0};
    double q_A255[5] = {0.0};

    // Variables de control de bucles
    bool HomeInicio = true;

    // Variables para comunicación
    bool BeginComunication = false;
```

```
// CRobotsDlg message handlers
BOOL CRobotsDlg::OnInitDialog(){
        CDialog::OnInitDialog();

        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);

        CMenu* pSysMenu = GetSystemMenu(FALSE);

        if (pSysMenu != NULL){
                CString strAboutMenu;
                strAboutMenu.LoadString(IDS_ABOUTBOX);

                if (!strAboutMenu.IsEmpty()){
                        pSysMenu->AppendMenu(MF_SEPARATOR);
                        pSysMenu->AppendMenu(MF_STRING,
                        IDM_ABOUTBOX, strAboutMenu);
                }
        }


        // TODO: Añadir inicialización extra aquí:

        // Iniciar comunicación con el FPGA

        BeginComunication = Comunication();
        if(!BeginComunication)
            MessageBox("Error␣al␣comunicarse␣con␣el␣FPGA");

        // Activar las articulaciones del robot A465

        Activar_ArtA465 = EneableJoints_A465(1,1,1,1,1,1);
        if(!Activar_ArtA465)
            MessageBox("Error␣al␣activar␣las␣articulaciones␣del␣robot␣A465");

        // Activar las articulaciones del robot A255

        Activar_ArtA255 = EneableJoints_A255(1,1,1,1,1,1);
        if(!Activar_ArtA255)
            MessageBox("Error␣al␣activar␣las␣articulaciones␣del␣robot␣A255");



        }
        return TRUE;  // return TRUE  unless you set the focus to a control
}


// Funcion para iniciar el FPGA
bool CRobotsDlg::Comunication(void){

        status = NiFpga_Initialize();

        if (NiFpga_IsNotError(status)){

                //Inici sesión, desgar bitstream y se comunica con el FPGA
                NiFpga_MergeStatus(&status, NiFpga_Open(NiFpga_final_Bitfile,
                                            NiFpga_final_Signature,
                                            "rio://192.168.0.2/RIO0",
                                            ///Se puede verificar en Max+devices
                                            0,
                                            &session));
                NiFpga_MergeStatus(&status, NiFpga_Run(session, 0));

                // Se verifica si algo ha salido mal
                if (NiFpga_IsError(status))
                        return false;
                else
                        return true;
        }
        else
                return false;
}
```

```
// Funcion para activar las articulaciones y
// fijar los límites de voltaje del Robot A465
bool CRobotsDlg::EneableJoints_A465(bool A465_art_1, bool A465_art_2,
                                    bool A465_art_3, bool A465_art_4,
                                    bool A465_art_5, bool A465_art_6){
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA465_art1,
                                                     A465_art_1));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA465_art2,
                                                     A465_art_2));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA465_art3,
                                                     A465_art_3));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA465_art4,
                                                     A465_art_4));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA465_art5,
                                                     A465_art_5));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA465_art6,
                                                     A465_art_6));

        NiFpga_MergeStatus(&status, NiFpga_Run(session, 0));

        // Establecer lÌmites de voltaje superior
        NiFpga_WriteI16(session,LVSA465_art1,static_cast<int16_t>(2*DACR));
        NiFpga_WriteI16(session,LVSA465_art2,static_cast<int16_t>(4*DACR));
        NiFpga_WriteI16(session,LVSA465_art3,static_cast<int16_t>(4*DACR));
        NiFpga_WriteI16(session,LVSA465_art4,static_cast<int16_t>(5*DACR));
        NiFpga_WriteI16(session,LVSA465_art5,static_cast<int16_t>(5*DACR));
        NiFpga_WriteI16(session,LVSA465_art6,static_cast<int16_t>(5*DACR));

        // Establecer limites de voltaje inferior
        NiFpga_WriteI16(session,LVIA465_art1,static_cast<int16_t>(-2*DACR));
        NiFpga_WriteI16(session,LVIA465_art2,static_cast<int16_t>(-4*DACR));
        NiFpga_WriteI16(session,LVIA465_art3,static_cast<int16_t>(-4*DACR));
        NiFpga_WriteI16(session,LVIA465_art4,static_cast<int16_t>(-5*DACR));
        NiFpga_WriteI16(session,LVIA465_art5,static_cast<int16_t>(-5*DACR));
        NiFpga_WriteI16(session,LVIA465_art6,static_cast<int16_t>(-5*DACR));

        if (NiFpga_IsError(status))
                return false;
        else
                return true;}

// Funcion  para activar las articulaciones y
// establecer los limites de voltajes del robot A255
bool CRobotsDlg::EneableJoints_A255(bool A255_Art_1, bool A255_Art_2,
                                    bool A255_Art_3, bool A255_Art_4,
                                    bool A255_Art_5){
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA255_art1,
                                                     A255_Art_1));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA255_art2,
                                                     A255_Art_2));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA255_art3,
                                                     A255_Art_3));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA255_art4,
                                                     A255_Art_4));
        NiFpga_MergeStatus(&status, NiFpga_WriteBool(session, actA255_art5,
                                                     A255_Art_5));
        NiFpga_MergeStatus(&status, NiFpga_Run(session, 0));

        // Establecer lÌmites de voltaje superior
        NiFpga_WriteI16(session,LVSA255_art1,static_cast<int16_t>(2*DACR));
        NiFpga_WriteI16(session,LVSA255_art2,static_cast<int16_t>(4*DACR));
        NiFpga_WriteI16(session,LVSA255_art3,static_cast<int16_t>(4*DACR));
        NiFpga_WriteI16(session,LVSA255_art4,static_cast<int16_t>(5*DACR));
        NiFpga_WriteI16(session,LVSA255_art5,static_cast<int16_t>(5*DACR));

        // Establecer limites de voltaje inferior
        NiFpga_WriteI16(session,LVIA255_art1,static_cast<int16_t>(-2*DACR));
        NiFpga_WriteI16(session,LVIA255_art2,static_cast<int16_t>(-4*DACR));
        NiFpga_WriteI16(session,LVIA255_art3,static_cast<int16_t>(-4*DACR));
        NiFpga_WriteI16(session,LVIA255_art4,static_cast<int16_t>(-5*DACR));
        NiFpga_WriteI16(session,LVIA255_art5,static_cast<int16_t>(-5*DACR));

        if (NiFpga_IsError(status))
                return false;
        else
                return true;}
```

```
// Funcion para leer los codificadores del robot A465
bool CRobotsDlg::GetEncoders_A465(void){

        int32_t lqA465[6]={0};
        NiFpga_MergeStatus( &status,NiFpga_ReadArrayI32(session,
                            NiFpga_final_IndicatorArrayI32_CUENTASA465,&lqA465[0],
                            NiFpga_final_IndicatorArrayI32Size_CUENTASA465));

        if (NiFpga_IsError(status))
                return false;

        for(int i=0; i<6; i++){
                q_A465[i] = 1.0*lqA465[i]*rpcA465[i];
        }
        return true;
}


// Funcion para leer los codificadores del robot A255
bool CRobotsDlg::GetEncoders_A255(void){

        int32_t lqA255[5]={0};
        NiFpga_MergeStatus( &status, NiFpga_ReadArrayI32(session,
                            NiFpga_final_IndicatorArrayI32_CUENTASA255,&lqA255[0],
                            NiFpga_final_IndicatorArrayI32Size_CUENTASA255));

        if (NiFpga_IsError(status))
                return false;

        for(int i=0; i<5; i++){
                q_A255[i] = 1.0*lqA255[i]*rpcA255[i];
        }
        return true;
}


// Funcion para mandar voltajes al robot A465
void CRobotsDlg::SetVoltage_A465( double A465_V1, double A465_V2,
                                  double A465_V3, double A465_V4,
                                  double A465_V5, double A465_V6){
        int16_t v_A465[6] = {0};

        v_A465[0] = static_cast<int16_t>(A465_V1*DACR);
        v_A465[1] = static_cast<int16_t>(A465_V2*DACR);
        v_A465[2] = static_cast<int16_t>(A465_V3*DACR);
        v_A465[3] = static_cast<int16_t>(A465_V4*DACR);
        v_A465[4] = static_cast<int16_t>(A465_V5*DACR);
        v_A465[5] = static_cast<int16_t>(A465_V6*DACR);

        NiFpga_WriteArrayI16(session, NiFpga_final_ControlArrayI16_VOLTAJESA465,
        v_A465, NiFpga_final_ControlArrayI16Size_VOLTAJESA465);
        return;
}


// Funcion para mandar voltajes al robot A255
void CRobotsDlg::SetVoltage_A255( double A255_V1, double A255_V2,
                                  double A255_V3, double A255_V4,
                                  double A255_V5){

        int16_t v_A255[5] = {0};

        v_A255[0] = static_cast<int16_t>(A255_V1*DACR);
        v_A255[1] = static_cast<int16_t>(A255_V2*DACR);
        v_A255[2] = static_cast<int16_t>(A255_V3*DACR);
        v_A255[3] = static_cast<int16_t>(A255_V4*DACR);
        v_A255[4] = static_cast<int16_t>(A255_V5*DACR);

        NiFpga_WriteArrayI16(session, NiFpga_final_ControlArrayI16_VOLTAJESA255,
        v_A255, NiFpga_final_ControlArrayI16Size_VOLTAJESA255);
        return;
}
```

```
// Boton para leer las posiciones articulares
void CRobotsDlg::OnBnClickedBtn_Get_q(){
        bool leer_A465 = true;
        bool leer_A255 = true;

        leer_A465 = GetEncoders_A465();
        leer_A255 = GetEncoders_A255();

        if(!leer_A465){
                MessageBox("Error␣al␣leer␣los␣codificadores␣del␣robot␣A465");
        }
        else{
                CString A465textq1;
                CString A465textq2;
                CString A465textq3;
                CString A465textq4;
                CString A465textq5;
                CString A465textq6;

                A465textq1.Format("%3.4lf",q_A465[0]*180/pi);
                A465textq2.Format("%3.4lf",q_A465[1]*180/pi);
                A465textq3.Format("%3.4lf",q_A465[2]*180/pi);
                A465textq4.Format("%3.4lf",q_A465[3]*180/pi);
                A465textq5.Format("%3.4lf",q_A465[4]*180/pi);
                A465textq6.Format("%3.4lf",q_A465[5]*180/pi);

                m_Edit_A465_q1.SetWindowTextA(A465textq1);
                m_Edit_A465_q2.SetWindowTextA(A465textq2);
                m_Edit_A465_q3.SetWindowTextA(A465textq3);
                m_Edit_A465_q4.SetWindowTextA(A465textq4);
                m_Edit_A465_q5.SetWindowTextA(A465textq5);
                m_Edit_A465_q6.SetWindowTextA(A465textq6);
        }
        if(!leer_A255){
                MessageBox("Error␣al␣leer␣los␣codificadores␣del␣robot␣A255");
        }
        else{
                CString A255textq1;
                CString A255textq2;
                CString A255textq3;
                CString A255textq4;
                CString A255textq5;

                A255textq1.Format("%3.4lf",q_A255[0]*180/pi);
                A255textq2.Format("%3.4lf",q_A255[1]*180/pi);
                A255textq3.Format("%3.4lf",q_A255[2]*180/pi);
                A255textq4.Format("%3.4lf",q_A255[3]*180/pi);
                A255textq5.Format("%3.4lf",q_A255[4]*180/pi);

                m_Edit_A255_q1.SetWindowTextA(A255textq1);
                m_Edit_A255_q2.SetWindowTextA(A255textq2);
                m_Edit_A255_q3.SetWindowTextA(A255textq3);
                m_Edit_A255_q4.SetWindowTextA(A255textq4);
                m_Edit_A255_q5.SetWindowTextA(A255textq5);
        }
}


// Boton para mandar voltajes
void CRobotsDlg::OnBnClickedBtn_Set_v(){
        // Eliminar Timers
        timeEndPeriod(1);

        timeKillEvent(TimerHome);

        UpdateData(TRUE);
        SetVoltage_A465(m_A465_v1, m_A465_v2,
                        m_A465_v3, m_A465_v4,
                        m_A465_v5, m_A465_v6);
        SetVoltage_A255(m_A255_v1, m_A255_v2,
                        m_A255_v3, m_A255_v4,
                        m_A255_v5);
        UpdateData(FALSE);
        return;
}
```

```
// Función para mandar voltaje de control de botonera E-STOP
void CRobotsDlg::Send_BTN_CTRL(double v_BTN_CTRL){

        int16_t v_BTN_CTRL = static_cast<int16_t>(activa_BTN_CTRL*DACR);
        NiFpga_WriteI16(session, NiFpga_final_ControlI16_BTN_CTRL , v_BTN_CTRL;
        return;
}


// Botón para mandar paro de emergencia
void CRobotsDlg::OnBnClickedBtn_ESTOP(){

        // Eliminar Timers A465
        timeEndPeriod(1);

        timeKillEvent(TimerHome);

        UpdateData(TRUE);
        Send_BTN_CTRL(-5.0);
        Sleep(300);
        Send_BTN_CTRL(0.0);
        UpdateData(FALSE);
        return;
}


// Botón para activar a los robots mecanicamente
void CRobotsDlg::OnBnClickedBtn_EACT(){

        // Eliminar Timers
        timeEndPeriod(1);

        timeKillEvent(TimerHome);

        UpdateData(TRUE);
        Send_BTN_CTRL(5.0);
        Sleep(300);
        Send_BTN_CTRL(0.0);
        UpdateData(FALSE);
        return;
}


// Botón para detener al programa
void CRobotsDlg::OnBnClickedCancel(){
        // TODO: Add your control notification handler code here
        timeEndPeriod(1);

        // Detener timers
        timeKillEvent(TimerHome);

        // Mandar cero volts a las articulaciones
        SetVoltage_A465(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
        SetVoltage_A255(0.0, 0.0, 0.0, 0.0, 0.0);

        // Desactivar las articulaciones
        EneableJoints_A465(0, 0, 0, 0, 0, 0);
        EneableJoints_A255(0, 0, 0, 0, 0);

        OnCancel();
}


// Boton para mandar a Home los robots A255 y A465
void CRobotsDlg::OnBnClickedBtn_Home(){

        // Eliminar timers
        timeEndPeriod(1);
        timeKillEvent(TimerHome);

        m_Edit_Status_Cop.SetWindowTextA("␣Llevando␣a␣la␣posiciÛn␣de␣Home");

        Sleep(5);

        timeBeginPeriod(1);
        ti = timeGetTime();
        TimerHome = timeSetEvent(static_cast<UINT>(1000*T), 0,
                                 ControlHome, 0, TIME_PERIODIC);
        HomeInicio = true;
}
```

```
// Función para llevar a home a los robots
void CALLBACK CRobotsDlg::ControlHome( UINT uTimeID, UINT uMsg, DWORD dwUser,
                                       DWORD dw1, DWORD dw2 )
{

        GetEncoders_A255();     /// Actualiza el valor de
        GetEncoders_A465();     /// q_A255 y q_A465.


        // Punto inicial y final de la trayectoria
        const double  qf_A255[5] = {0.0, pi/2.0, -pi/2.0, pi/2.0, 0.0};
        const double  qf_A465[6] = {0.0, pi/2.0, 0.00000, 0.0000, 0.00, 0.0};

        static double qi_A255[5], qi_A465[6];    /// q inicial

        static double qa_A255[5], qa_A465[6];    /// q aterior


        // Velocidad articular
        double qp_A255[5]  = {0.0};
        double qp_A465[6]  = {0.0};


        // Trayectoria deseada
        double qd_A255[5]  = {0.0};
        double qd_A465[6]  = {0.0};

        double qpd_A255[5] = {0.0};
        double qpd_A465[6] = {0.0};


        // Constantes del polinomio deseado
        double a0_A255[5], a3_A255[5], a4_A255[5], a5_A255[5];
        double a0_A465[6], a3_A465[6], a4_A465[6], a5_A465[6];


        // Tiempo final
        const double tf = 10.0;


        // Errores
        double Deltaq_A255[5], Deltaqp_A255[5];
        double Deltaq_A465[6], Deltaqp_A465[6];

        static double sigma_A255[5];
        static double sigma_A465[6];


        // Par de salida
        double tau_A255[6] = {0.0};
        double tau_A465[6] = {0.0};


        // Ganancias del control PID
        const double kp_A255[5] = {80.0, 110.0, 112.0, 20.0, 38.0};
        const double kv_A255[5] = {8.00, 8.090, 8.500, 2.50, 1.00};
        const double ki_A255[5] = {4.00, 3.236, 5.200, 1.00, 0.10};
        const double kp_A465[6] = {20.0, 132.0, 132.0, 62.0, 102.0, 70.0};
        const double kv_A465[6] = {4.20, 10.20, 10.20, 5.00, 5.000, 4.50};
        const double ki_A465[6] = {1.00, 3.000, 3.000, 2.00, 3.500, 1.50};

        CRobotsDlg* pMainWnd = (CRobotsDlg *)AfxGetApp()->m_pMainWnd;
```

```
// Condiciones iniciales
if(HomeInicio){ ///Bandera global booleana
        for(int i=0;i<5;i++){
                qi_A255[i]    = q_A255[i];
                qa_A255[i]    = q_A255[i];
                sigma_A255[i] = 0.0;
        }

        for(int i=0;i<6;i++){
                qi_A465[i]    = q_A465[i];
                qa_A465[i]    = q_A465[i];
                sigma_A465[i] = 0.0;
        }
}


// Tiempo
t = (timeGetTime() - ti)/1000.0;
/// ti = timeGetTime() en OnBnClickedBtn_Home()


// Derivada numérica
for(int i=0;i<5;i++)
        qp_A255[i] = (q_A255[i] - qa_A255[i])/T;

for(int i=0;i<6;i++)
        qp_A465[i] = (q_A465[i] - qa_A465[i])/T;


// Trayectoria deseada
if(HomeInicio){
        for(int i=0;i<5;i++){
                a0_A255[i] = qi_A255[i];
                a3_A255[i] =  10.0*(qf_A255[i] - qi_A255[i])/(tf*tf*tf);
                a4_A255[i] = -15.0*(qf_A255[i] - qi_A255[i])/(tf*tf*tf*tf);
                a5_A255[i] =   6.0*(qf_A255[i] - qi_A255[i])/(tf*tf*tf*tf*tf);
        }

        for(int i=0;i<6;i++){
                a0_A465[i] = qi_A465[i];
                a3_A465[i] =  10.0*(qf_A465[i] - qi_A465[i])/(tf*tf*tf);
                a4_A465[i] = -15.0*(qf_A465[i] - qi_A465[i])/(tf*tf*tf*tf);
                a5_A465[i] =   6.0*(qf_A465[i] - qi_A465[i])/(tf*tf*tf*tf*tf);
        }

        HomeInicio = false;
}

if(t<=tf){
        for(int i=0;i<5;i++){
                qd_A255[i]  = a0_A255[i] + a3_A255[i]*(t*t*t)
                + a4_A255[i]*(t*t*t*t) + a5_A255[i]*(t*t*t*t*t);
                qpd_A255[i] = 3.0*a3_A255[i]*(t*t) + 4.0*a4_A255[i]*(t*t*t)
                + 5.0*a5_A255[i]*(t*t*t*t);
        }
        for(int i=0;i<6;i++){
                qd_A465[i]  = a0_A465[i] + a3_A465[i]*(t*t*t)
                + a4_A465[i]*(t*t*t*t) + a5_A465[i]*(t*t*t*t*t);
                qpd_A465[i] = 3.0*a3_A465[i]*(t*t) + 4.0*a4_A465[i]*(t*t*t)
                + 5.0*a5_A465[i]*(t*t*t*t);
        }
}
else{
        for(int i=0;i<5;i++){
                qd_A255[i]  = qf_A255[i];
                qpd_A255[i] = 0.0;
        }
        for(int i=0;i<6;i++){
                qd_A465[i]  = qf_A465[i];
                qpd_A465[i] = 0.0;
        }
        if(t<tf+2*T){
                pMainWnd->m_Edit_Status_Cop.SetWindowTextA("␣En␣Home␣");
        }
}
```

```
        // Errores
        for(int i=0;i<5;i++){
                Deltaq_A255[i]  =  q_A255[i] -  qd_A255[i];
                Deltaqp_A255[i] = qp_A255[i] - qpd_A255[i];
        }
        for(int i=0;i<6;i++){
                Deltaq_A465[i]  =  q_A465[i] -  qd_A465[i];
                Deltaqp_A465[i] = qp_A465[i] - qpd_A465[i];
        }


        // Ley de control
        for(int i=0;i<5;i++)
                tau_A255[i] = -kp_A255[i]*Deltaq_A255[i]
                              -kv_A255[i]*Deltaqp_A255[i]
                              -ki_A255[i]*sigma_A255[i];
        for(int i=0;i<6;i++)
                tau_A465[i] = -kp_A465[i]*Deltaq_A465[i]
                              -kv_A465[i]*Deltaqp_A465[i]
                              -ki_A465[i]*sigma_A465[i];


        // Se envían los voltajes a los motores
        SetVoltage_A255( 1.0*tau_A255[0], 1.0*tau_A255[1], 1.0*tau_A255[2],
                         1.0*tau_A255[3], 1.0*tau_A255[4]);
        SetVoltage_A465( 1.0*tau_A465[0], 1.0*tau_A465[1], 1.0*tau_A465[2],
                         1.0*tau_A465[3], 1.0*tau_A465[4], 1.0*tau_A465[5]);


        // Integración numérica
        for(int i=0;i<5;i++)
                sigma_A255[i] += T*Deltaq_A255[i];
        for(int i=0;i<6;i++)
                sigma_A465[i] += T*Deltaq_A465[i];


        // Actualización
        for(int i=0;i<5;i++)
                qa_A255[i] = q_A255[i];
        for(int i=0;i<6;i++)
                qa_A465[i] = q_A465[i];
}
```

Bibliografía

[1] Biography.com Editors, "Isaac Asimov Biography," March 2016.

[2] J. Pearce, "George C. Devol, Inventor of Robot Arm, Dies at 99," August 2011.

[3] Manutención y Almacenaje.com, "Las ventas de robots industriales se disparan en 2014," March 2015.

[4] LAR-DEIS, "Equipments."

[5] A. Macchelli and C. Melchiorri, "A real-time control system for industrial robots and control applications based on real-time linux," *15th IFAC World Congress*, July 2002.

[6] PRISMA Lab, "Industrial robotics lab.."

[7] S. S. Bermejo, *Desarrollo de robots basados en el comportamiento*. Barcelona, España: Edicions UPS, primera ed., Diciembre 2004.

[8] R. A. Siddiqui, R. I. Grosvenor, and P. W. Prickett, "dspic-based advanced data acquisition system for monitoring, control and security applications," *International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp. 293 – 298, January 2015.

[9] H. Zumbahlen, *Linear Circuit Design Handbook*. Massachusetts, United States: Analog Devices, Inc., first ed., February 2008.

[10] E. J. Chikofsky and J. H. Cross II, "Reverse engineering and design recovery: A taxonomy," *IEEE Software*, vol. 7, pp. 13–17, January 1990.

[11] D. Swafford, D. Elman, P. Aiken, and J. Merhout, "Experiences reverse engineering manually," *Proceedings of the Fourth European Software Maintenance and Reengineering*, pp. 189 – 197, February 2000.

[12] M. G. Rekoff Jr., "On reverse engineering," *IEEE Transactions on System, Man and Cybernetics*, vol. SMC-15, pp. 244–252, March 1985.

[13] G. K. Jounghyun and A. G. Bekey, "Constructing design plans of dfa redesign," *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 312–318, August 1993.

[14] PCMag, "Encyclopedia."

[15] CRS Robotics Corporation, *A465 Robot Arm User Guide*. CRS Robotics Corporation, Burlington, Ontario, Canada, 2000.

[16] CRS Robotics Corporation, *A255 Robot Arm User Guide*. CRS Robotics Corporation, Burlington, Ontario, Canada, 2000.

[17] Sumtak Corporation of America, *Size 20 Series LDA Incremental Modular Encoder*. Sumtak Corporation of America, Shibuya, Tokyo, Japan, December 2001.

[18] CRS Robotics Corporation, *C500 Controller User Guide*. CRS Robotics Corporation, Burlington, Ontario, Canada, January 2000.

[19] A. M. Castillo Sánchez, "Adaptación de dos robots industriales para su utilización en el desarrollo de nuevas técnicas y algoritmos de control," April 2002.

[20] National Instruments Corporation, *Operating Instructions and Specifications CompactRIO NI cRIO-9012/9014*. National Instruments Corporation, Austin, Texas, United States, Junio 2010.

[21] National Instruments Corporation, *Operating Instructions and Specifications NI 9263*. National Instruments Corporation, Austin, Texas, United States, August 2009.

[22] National Instruments Corporation, *Operating Instructions and Specifications NI 9401*. National Instruments Corporation, Austin, Texas, United States, November 2012.

[23] Phoenix Corporation, "Power Supply Unit Quint-PS-100."

[24] The CircuitCalculator Blog, "Path Calculator."

[25] OSH Park, "Pricing and Specifications."

[26] Texas Instruments Incorporated, *Quadruple Differential Line Receivers*. Texas Instruments Incorporated, Dallas, Texas, United States, November 2002.

[27] Avago Technologies, *High CMR, High Speed TTL Compatible Optocouplers*. Avago Technologies, San José, California, United States, July 2014.

[28] Avago Technologies, *4N35 Phototransistor Optocoupler General Purpose Type*. Avago Technologies, San José, California, United States, October 2007.

[29] Fairchild Semiconductor International, *TIP31 Series(TIP31/31A/31B/31C)*. Fairchild Semiconductor International, San José, California, United States, February 2000.

[30] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. John Wiley and Sons, Inc., second ed., July 2006.