Control de movimientos de Equipo Rotativo de Ultrasonido.

Proyecto de investigación

Por

Daniel kaleb Ruiz Casales

En cumplimiento a los requerimientos para la obtención de la Especialidad de Tecnólogo en Mecatrónica

Revisor académico: Dr. Jorge Alberto Soto Cajiga.

Santiago de Querétaro, Qro., México, Julio 2020

Introducción.

En México, el transporte de líquidos, gases e incluso sólidos por medio de ductos, es un medio de transporte muy utilizado en el campo de energías. Este tipo de transporte es apropiado para la distribución de hidrocarburos, gases, residuos y tratado de aguas. Haciendo enfoque en los hidrocarburos y específicamente en la distribución de combustibles, las redes de distribución son diseñadas y llevadas a campo bajo normas de calidad y estándares establecidos, debido a que se han desarrollado bajo la estructura descrita deben de cumplir con especificaciones técnicas que garanticen un óptimo estado para evitar fallas que provoquen accidentes.

La inspección de tuberías es esencial para el mantenimiento de ductos, ya que estos ayudan a predecir el estado actual de la tubería o sistema en general. Con el paso del tiempo se han desarrollado técnicas para verificar el estado de estas, a día de hoy las pruebas más empleadas son los testeos no destructivos ya que estos garantizan iguales o mejores resultados que las pruebas destructivas, sin la necesidad de averiar, suspender o extraer el ducto del sistema. El avance de la tecnología fue quién permitió pasar de pruebas destructivas a las no destructivas, gracias al desarrollo de equipos de medición basados en fenómenos físicos. Sin embargo, aún se requiere de la intervención de personal capacitado que opere el equipo de medición. Una de las desventajas de usar personal es que existen ambientes peligrosos para el usuario o demasiado demandantes en cuestión de inspección. A día de hoy los robots abarcan más y más terreno en nuestras vidas, tal es el caso de los robots de inspección que han sido desarrollados para llegar a lugares que antes era difícil o peligroso para un usuario de inspección.

En el centro de Ingeniería y Desarrollo Industrial (CIDESI) se han desarrollado robots de inspección para tuberías conocidos como PIG [1]. Uno de los robots que se desarrollaron fue el Equipo Rotativo de Ultrasonido (ERUS). Como en todo proyecto de tecnología existen actualizaciones y mejoras que permitan mayor eficiencia en el desempeño del robot. El siguiente documento presenta una propuesta de distribuir el control total del robot que se aplica por medio de un dispositivo FPGA. Se busca tener un sistema distribuido en la asignación de tareas liberando recursos al FPGA e incorporar un microcontrolador que realice esas tareas que proporcionen al sistema una mayor robustez y eficiencia, ya que al día de hoy los sistemas se diseñan bajo el concepto de operar mediante una red de comunicación donde el dispositivo con mejores recursos y procesamiento se encargue controlar de manera general sin la necesidad de realizar todas las tareas de manera específica, para eso se encuentran dispositivos de menores capacidades encargados de resolver una sola tarea evitando congestionar de información al controlador principal que facilita la toma de decisiones programadas en el controlador principal, así como el diagnostico de fallas o aislar averías del sistema evitando que lo colisionen por completo.

Planteamiento del Problema.

El controlador de ERUS, es un FPGA capaz de mantener el movimiento del robot y a su vez el procesamiento de todas las señales provenientes de transductores de ultrasonido.

Sin embargo, esto genera que el sistema no sea robusto, ya que asumir todo el control de manera específica a un solo dispositivo nos genera la dependencia total al rendimiento de este. Considerando que si por algún motivo falla alguno de los mecanismos o se detecta una anomalía en el sistema podría perjudicar un proceso al otro cuando estos podrían llevarse de manera independiente, es decir, fallas en el procesamiento de señales podrían perjudicar al control de movimientos del robot o viceversa. Aunque son procesos que se relacionan no necesariamente su dependencia es total.

Objetivo general.

Delegar el control de movimientos del Core3S500E a un microcontrolador TM4C123GH6PM de la marca Texas Instruments. Para llevar a cabo la realización del objetivo principal es necesario cumplir con una serie de pasos u objetivos específicos para concluir de manera satisfactoria.

Objetivos específicos:

- Acondicionar el robot para su funcionamiento y desarrollo de pruebas durante el proyecto.
- Crear un nuevo algoritmo para el microcontrolador designado en el control de movimientos (TM4C123).
- Crear una interfaz para el usuario que manipule a ERUS.
- Implementar un control proporcional al cabezal rotativo respecto la velocidad del robot.
- Modificar el PCB o Core que controla al robot de inspección.

Justificación.

Sabemos que los FPGA debido a su arquitectura son muy superiores procesando información al compararlos con un microcontrolador, esto por su capacidad de procesar instrucciones en paralelo. Puede ser una de las ventajas por las que se considere que anexar un microcontrolador no es una opción viable. Sin embargo, esto lo convierte en un sistema menos robusto.

Si un solo dispositivo es el encargado de realizar el control absoluto de todo el sistema, por más poderoso que este sea una falla o anomalía en alguna de las partes del sistema podría afectar de manera general al dispositivo (FPGA) o hacer colisionar el sistema, cuando tal vez el problema no sea muy relevante y el robot pueda seguir trabajando o por lo menos ser operado para ser atendido. Hoy en día muchos sistemas trabajan delegando tareas específicas a un dispositivo que será comunicado en una red que comparte información y son controlados por uno superior a ellos en prestaciones, sin embargo, este no realiza todas las tareas del sistema, toma decisiones de acuerdo a la información que le proporcionan los dispositivos a los que se les asigno una de las tareas que debe realizar el sistema. Tener un sistema que delega actividades también proporciona la ventaja de modificar, actualizar o usar esa parte del sistema para ser implementado fuera de él es más fácil, debido a la distribución y arquitectura.

Otro aspecto a considerar es que los FPGA son tan poderos porque estos eran originalmente pensados para el desarrollo de proyectos nuevos donde se tenía de cierta manera incertidumbre del gasto total de recursos para el procesamiento de información, y aunque se tenían estimaciones por ser nuevos proyectos es difícil acertar completamente. Una vez que el proyecto es desarrollado se toma la descripción del hardware (VHDL o VERILOG) y lo ideal es generar un circuito o embebido con un porcentaje poco sobrado de recursos que consumirá el sistema ya desarrollado en el FPGA. Para ser implementado finalmente. Esto genera un costo adicional e injustificable cuando el sistema no será masificado. Resumiendo, que un FPGA aunque hoy en día es usado como embebido en versiones finales no es su finalidad principal.

Antecedentes.

El uso del medidor de inspección de tuberías “PIG” permite que el operador de la tubería realice las operaciones de mantenimiento requeridas sin detener el flujo de producto en la tubería. Los PIG modernos utilizan instrumentos sofisticados para recopilar datos de tuberías y a menudo requieren el uso de baterías a bordo para alimentar los dispositivos electrónicos. Además, debido a la ubicación variada de las tuberías y los materiales / grosores de las tuberías, la electrónica a bordo no solo necesita almacenar los datos durante la duración del servicio, sino que también utiliza tecnología de localización para correlacionar los datos adquiridos [1].

Al insertar el PIG en un lanzador (o estación de lanzamiento) y luego aplicar flujo bajo presión a la parte posterior del dispositivo se moverá a la tubería. La fuerza aplicada por un PIG cuando atraviesa una tubería se puede calcular multiplicando

El área de la sección transversal de la parte posterior del cerdo por la presión aplicada a la parte posterior del cerdo. Una vez que se ha lanzado y se mueve a través de la tubería, la presión diferencial se puede calcular restando la presión frente al PIG por la presión que actúa sobre la parte transversal del dispositivo.

Figure 1. Ejemplos de PIGs.

Ventajas de los PIG.

Las tuberías son generalmente aceptadas como el método más eficiente para transportar líquidos y gases a través de varias distancias Representan un importante compromiso financiero, ambiental y operativo por parte de todos los interesados y en orden para proteger estas valiosas inversiones, el mantenimiento continuo debe llevarse a cabo regularmente para asegurar la tubería continúa entregando un rendimiento óptimo [[2]1].

En el caso de las tuberías nuevas, una vez que se complete la construcción, deben ser probadas hidrostáticamente para demostrar que serán (Horizon Industrial) (Remotely Operated Vehicle, s.f.) xcapaz de cumplir con la MAOP designada (Presión de funcionamiento máxima permitida).

Microcontrolador.

Los principiantes en electrónica creen que un microcontrolador es igual a un microprocesador. Esto no es cierto. Difieren uno del otro en muchos sentidos. La primera y la más importante diferencia es su funcionalidad. Para utilizar al microprocesador en una aplicación real, se debe de conectar con componentes tales como memoria o componentes buses de transmisión de datos. Aunque el microprocesador se considera una máquina de computación poderosa, no está preparado para la comunicación con los dispositivos periféricos que se le conectan. Para que el microprocesador se comunique con algún periférico, se deben utilizar los circuitos especiales. Así era en el principio y esta práctica sigue vigente en la actualidad [5].

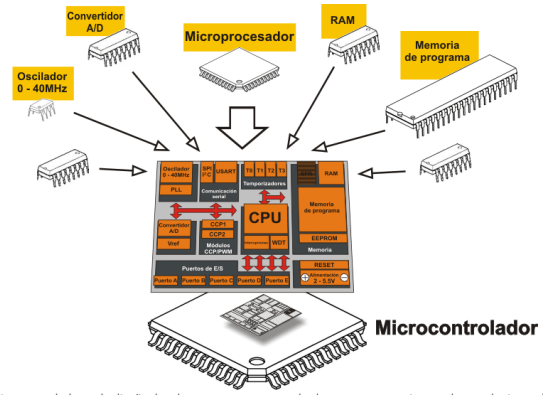


Figure 2. Infraestructura de un microcontrolador [5].

LabVIEW

LabVIEW es desarrollado y producido por National Instruments como un entorno utilizado para el diseño de sistemas gráficos.

LabVIEW ofrece un enfoque de programación gráfica de LabVIEW que le ayuda a visualizar cada aspecto de su aplicación, incluyendo configuración del hardware, datos de medidas y depuración. Esta visualización simplifica la integración del hardware de medidas de cualquier proveedor, representa una lógica compleja en el diagrama, desarrolla algoritmos de análisis de datos y diseña interfaces de usuario de ingeniería personalizadas.

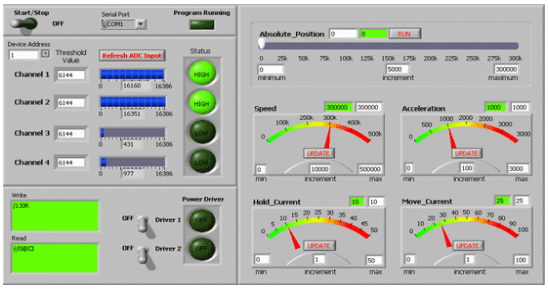


Figure 3. Panel de control en Labview.

Transductores.

Los transductores son aquellas partes de una cadena de medición que transforman una magnitud física en una señal eléctrica. Los transductores son especialmente importantes para que los medidores Transductores para profesionales para la inspección y control puedan detectar magnitudes físicas. Normalmente, estas magnitudes, como por ejemplo temperatura, presión, humedad del aire, presión sonora, caudal, o luz, se convierten en una señal normalizada (p.e. 4 ... 20 mA). Las ventajas de la transformación son por un lado la flexibilidad, ya que muchos medidores soportan la transformación de señales normalizadas. Por otro lado, las magnitudes medidas pueden ser leídas a grandes distancias sin prácticamente pérdida alguna. Cuando se usan transductores, la unidad de evaluación debe recibir sólo el rango de medición, pues a partir de ahí, se calculan desde la señal eléctrica las magnitudes eléctricas. Algunos transductores ofrecen adicionalmente una separación galvánica entre la señal de entrada y de salida.

Encoder.

Los codificadores rotatorios (conocidos genéricamente como encoders) son mecanismos utilizados para entregar la posición, velocidad y aceleración del rotor de un motor. Sus principales aplicaciones incluyen aplicaciones en robótica, lentes fotográficas, aplicaciones industriales que requieren medición angular, militares, etc.

Un codificador rotatorio es un dispositivo electromecánico que convierte la posición angular de un eje, directamente a un código digital.

Los tipos más comunes de encoders se clasifican en: absolutos y relativos (conocidos también como incrementales). Los encoders absolutos pueden venir codificados en binario o gray. Dentro de los encoders incrementales, se encuentran los encoders en cuadratura, ampliamente utilizados en motores de alta velocidad y en aplicaciones en las que interesa conocer la dirección del movimiento del eje.

Encoder en Cuadratura

Corresponde a un tipo de encoder incremental que utiliza dos sensores ópticos posicionados con un desplazamiento de ¼ de ranura el uno del otro, generando dos señales de pulsos digitales desfasada en 90º o en cuadratura. A estas señales de salida, se les llama comúnmente A y B. Mediante ellas es posible suministrar los datos de posición, velocidad y dirección de rotación del eje. Si se incluye la señal de referencia, se le denomina I (índice).

Figure 4. Lectura de canales A y B de un encoder.

Metodología.

	En-ab 2020				may-20				jun-20				jul-20					ago-20		
Actividad					1 al 8	11 al 15	18 al 22	23 al 28	03 al 07	10 al 14	17 al 21	24 al 28	01 al 05	08 al 12	15 al 19	22 al 26	29 al 31	0 1 al 02	05 al 09	12 al 16
Revision estado del arte.		X			X	X														
Bases del ERUS.						X														
Revisión de ERUS.	X	X	X			X	X													
Acondicionamiento mecanico.								X												
Prueba de drivers.		X						X												
Prueba de motores.		X						X												
Prueba de planetarios.								X												
Prueba de sensores.								X												
Prueba de encoder.								X												
Prueba de tarjeta de programación.								X												
Desarrollo de interfaz gráfica (Labview).		X					X	X	X	X	X	X	X	X						
Programación de microcontrolador.			X						X	X	X	X	X	X	X	X				
Diseño electrónico para pruebas.									X	X										
Diseño y programación de interfaz.	X									X		X								
Ajustes mecánicos.									X	X										
Análisis de información y señales.										X	X	X	X	X	X					

Tabla 1. Cronograma de actividades planeado.

En la tabla se puede observar la planeación de actividades de los cuales se consideraron de manera semanal, sin embargo, los avances de la semana se consideran en un horario de lunes a viernes.

Los cuadros marcados en color azul fueron avances de proyecto previstos ya que, la planeación del trabajo estaba dada durante el cuatrimestre mayo-agosto. El cual no es el periodo exclusivo para el desarrollo de proyectos según el plan de estudios de la especialidad, sin embargo, se designó avance para evitar contratiempos como los que se presentaron durante la contingencia.

A excepción de las pruebas de laboratorio se cumplió con el desarrollo del proyecto en tiempo y forma para llevar a cabo el funcionamiento del robot.

- Investigación teórica: durante esa semana se estudiarán las bases por las que el ERUS funciona de manera teórica para entender el principio de su funcionamiento.
- Bases del ERUS: una vez que se conoce sobre la teoría del proyecto se identificará el sistema de funcionamiento, en base a la documentación existente del proyecto, tales como planos mecánicos, eléctricos, diagramas, esquemáticos y toda información se contenga del proyecto.
    - Revisión del ERUS: es necesario checar el estado actual en el que se encuentra el proyecto, verificar que contenga todas sus piezas mecánicas y dispositivos que lo componen. Poner en funcionamiento todo el robot o parcial respecto a su situación física del robot para comprobar funcionamiento, de no ser así modificar y reparar.
- Acondicionamiento mecánico: en caso de que el robot requiera de refacciones o ajustes mecánicos para operarse en pruebas.
- Prueba de drivers: verificar que estén en condiciones adecuadas de lo contrario informar y remplazar para continuar con las pruebas del prototipo.
- Pruebas de los motores: verificar que estén en condiciones adecuadas de lo contrario informar y remplazar para continuar con las pruebas del prototipo.
- Pruebas de planetarios: verificar que estén en condiciones adecuadas de lo contrario informar y remplazar para continuar con las pruebas del prototipo.
- Pruebas de encoders: verificar que estén en condiciones adecuadas de lo contrario informar y remplazar para continuar con las pruebas del prototipo.
- Pruebas de sensores: verificar que estén en condiciones adecuadas de lo contrario informar y remplazar para continuar con las pruebas del prototipo.
- Prueba de tarjeta de desarrollo: conectar la tiva a un ordenador para programar ejemplos y verificar el correcto funcionamiento de sus periticos.
- Licencias de software: instalar en la PC de trabajo el software requerido para la programación del microcontrolador, la interfaz, la programación de los drivers, el diseño de PCBs entre otros programas.
- Programación de microcontrolador: desarrollar el código mediante el compilador Code Composer para el Tiva serie C.

- Diseño electrónico de pruebas: Realizar cualquier circuito necesario para el desarrollo de del proyecto ya sea con tablillas de pruebas (Portoboard), circuitería impresa o ajustes de los ya existentes.
- Diseño y programación de interfaz: por medio del software Labview se realizará una interfaz para leer de manera más eficiente y cómoda las lecturas del encoder así como el control proporcional de velocidad.
- Ajustes mecánicos: El ERUS podría requerir ajustes una vez que se actualice el sistema en caso de requerir ajustes se utilizará esa semana.
- Análisis de información: cuando se encuentre lista la interfaz, la programación y no se requieran ajustes mecánicos. Se tomarán datos resultantes de las pruebas para graficar, analizar la respuesta del robot.

Recursos materiales y humanos.

A continuación, se muestra una lista de los elementos necesarios para realizar las pruebas del proyecto.

Tabla 2. Lista de recursos para elaborar pruebas.

Recurso	Modelo	Cantidad
Driver para motor	ESCON 36/3	1
Planetario	Maxon Gear 166949	1
Motor sin escobillas	Maxon EC-max 272768	1
Encoder incremental.	AVAGO EAT-601B	1
Computador Portatil.	----------	1
Microcontrolador.	Tiva LaunchPad serie C	1
Compilador.	Code Composer	
Software para interfaz.	Labview	1
Software de diseño electrónico.	Altium Designer	

Para llevar a cabo la programación de ERUS se realizó una primera inspección respecto al estatus del robot. A continuación, se muestra el prototipo sobre el que se estuvo desarrollando el proyecto.

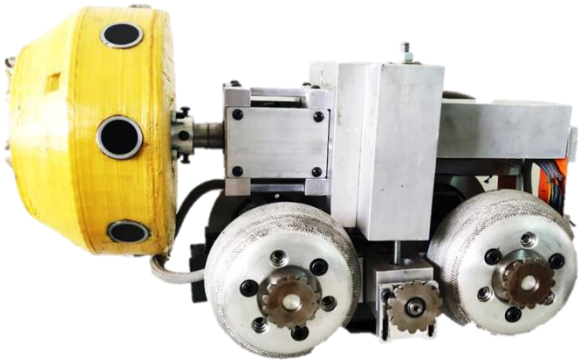


Figure 5. Vista lateral del robot ERUS.

La operación del robot prácticamente es dividida en dos partes principales: el movimiento del robot y la rotación del cabezal rotativo que contiene los transductores ultrasónicos. Estas dos partes del robot deben trabajar conjuntamente cuando el robot se encuentre operando, es decir, mientras el robot se desplaza por el interior de un ducto, el eje que contiene los sensores ultrasónicos deberá girar de manera proporcional a la velocidad de desplazamiento que el ERUS mantiene. Esto indica que mientras el robot no se encuentre en movimiento el revolver no deberá girar y cuando se encuentre transportándose a su máxima velocidad este deberá de girar a su máximo número de rpms. La velocidad máxima que el robot deberá de alcanzar son 0.05 m/s.

El robot deberá ser operado mediante una interfaz gráfica que facilite al operador la manipulación del ERUS y la visualización de la información que muestre el robot. tales como; velocidad del robot, velocidad del revólver, distancia recorrida del robot, posición del cabezal rotativo, dirección de la tracción en el robot, son algunos de los datos que arrojará el robot en la simulación y pruebas de laboratorio por el momento. Cabe recordar que esto es una propuesta de actualización y mejora para un desarrollar un nuevo prototipo más eficiente.

El diseño de la programación se basará en el siguiente diagrama secuencial para desarrollar el algoritmo capaz de desarrollar la propuesta mencionada anteriormente.

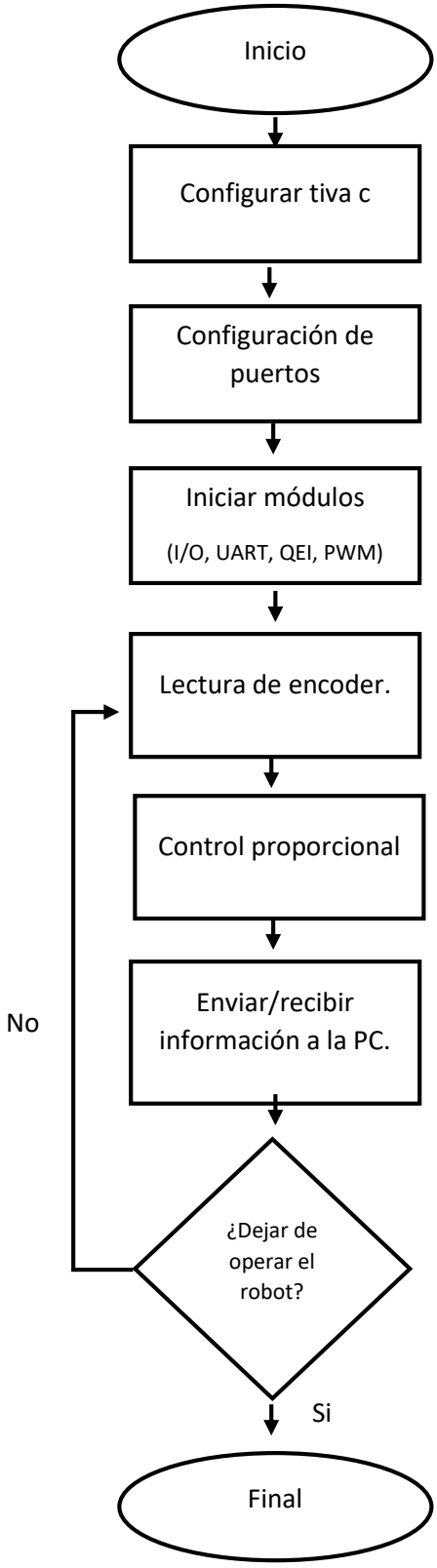


Figure 6. Diagrama de flujo para desarrollar algoritmo de programación.

El control del cabezal rotativo se basó en un control proporcional lineal. Se utilizó un encoder incremental magnético modelo AEAT-601B acoplado a una rueda loca del robot en la parte trasera de este. El encoder cuenta con una resolución de 256 pulsos por revolución. Con la información que ´proporciona su hoja de datos se obtuvieron las mediciones adecuadas para obtener la velocidad del robot, la distancia recorrida, la posición de la rueda loca. A continuación, se muestra el encoder adaptado a la rueda loca con un diámetro de 48.2 mm.

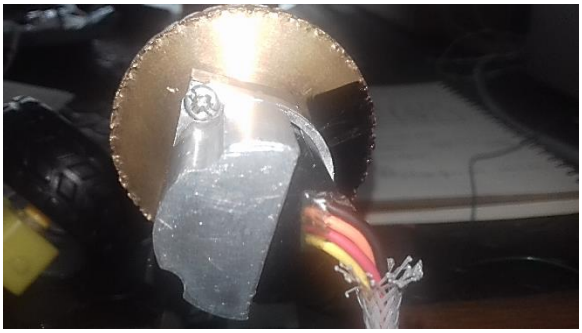


Figure 7. Encoder de cuadratura del robot ERUS.

De acuerdo con la información de la hoja de datos el encoder proporciona dos canales A y B con un Índex, esta información se representa mediante pulsos eléctricos. El microcontrolador tm4c123gh6pm cuenta con módulos QEI (Quadrature Encoder Interface) los cuales pueden ser configurados para leer la posición y la velocidad que proporciona el objeto al que está acoplado el encoder. A continuación, se presentan los métodos y técnicas con las que se obtuvieron las ecuaciones para obtener la variable a medir respecto a los pulsos que genera el encoder.

Velocidad de rueda loca:

El encoder cuenta con una resolución de 256 ppr. El modulo del microcontrolador se configura como lector de velocidad con las siguientes líneas de código.

```
//////////////////////////////////////////////////////////////////

//////////////       Habilitar módulo QEI1    ////////////////////////
//////////////////////////////////////////////////////////////////

QEIDisable(QEI1_BASE);
QEIIntDisable(QEI1_BASE,    QEI_INTERROR    |    QEI_INTDIR    |    QEI_INTTIMER    |
              QEI_INTINDEX);
QEIConfigure(QEI1_BASE,     QEI_CONFIG_CAPTURE_A     |     QEI_CONFIG_NO_RESET     |
             QEI_CONFIG_QUADRATURE | QEI_CONFIG_NO_SWAP , 512); //
QEIEnable(QEI1_BASE);
QEIPositionSet(QEI1_BASE,0);
QEIVelocityEnable(QEI1_BASE);
QEIVelocityConfigure(QEI1_BASE, QEI_VELDIV_1, SysCtlClockGet()/10);
```

Aquí solo se muestra la configuración y actividad de los módulos, pero cabe mencionar que deben configurarse parámetros principales del microcontrolador como los periféricos, la velocidad de reloj, los puertos configurados entre otros ajustes, para más información sobre el código completo consulte los anexos al final de este documento.

Para el modulo QEI1 se utilizó solo la lectura de un canal. El modulo leerá el número de pulsos por cambio de flanco es decir para leer 256 pulsos el controlador percibirá los cambios de estado que transcurren para cada pulso. En cada puso tenemos dos cambios de flanco. Esto se hace porque físicamente el encoder ofrece 256 pulso, pero electrónicamente para validar un pulso tiene que haber dos transiciones ya sea Low-High ó High-Low. De manera que obtendremos el doble de pulsos para una revolución del encoder, teniendo en total 512 pulsos por revolución.

Al configurar el modulo en modo velocidad es necesario tener un tiempo de muestreo, este sirve para obtener el número de pulsos que proporciona el objeto en constante movimiento. A diferencia del modo posición que mantiene la cuenta de los pulsos que se van generando conforme gira el encoder. Esta configuración también ser utilizo para obtener varíales de posicionamiento.

El tiempo de muestreo se definió a 0.01 segundos (10 ms). Para definir el tiempo en el microcontrolador es necesario convertir el tiempo a ciclos de reloj, por medio de la siguiente formula se obtuvo el número de ciclos que equivalen a la base de tiempo que requerimos:

$$numero\ de\ ciclos = \frac{\text{Tiempo deseado}}{\text{T de reloj}} \Rightarrow \frac{0.1}{0.0000002} = 500\ 000$$

La velocidad de reloj se configuro a 50 MHz. Obteniendo su periodo tenemos como resultado que “T de reloj” es igual a: 0.00000002 s ó 20 ns. Sabemos que el inverso multiplicativo de la frecuencia es el periodo de una señal.

$$T = \frac{1}{\text{Frecuencia.}} \Rightarrow \frac{1}{50000000\ \text{Hz}}$$

De manera que cada 500 000 ciclos de reloj el modulo del encoder se refrescara con el número de pulsos que capturo y de esta manera podemos calcular las revoluciones por minuto sin necesidad esperar el primer minuto para realizar el conteo. La siguiente ecuación determina el cálculo de velocidad en rpms con la información obtenida hasta el momento.

$$rpms = \frac{(\#\ pulsos) * (\ 6000)}{512}$$

Donde:

6000 equivale a la cantidad de veces que se repite el suceso en un minuto, es decir cómo se muestrea cada 0.01 segundos se divide 60 segundos entres 0.01 segundos.

512 es otra constante que ya habíamos calculado donde se usan 256 pulso multiplicado por el número de flancos que se requieren para validar el pulso en el módulo que es 2.

La siguiente instrucción obtiene la velocidad en pulsos que proporciona el módulo QEI1.

```
pulsos   =   QEIVelocityGet(QEI1_BASE);
```

Velocidad del robot.

Obtener la velocidad del robot depende de la lectura de la velocidad de la rueda loca. Se convirtió velocidad rotativa a velocidad lineal por medio de la siguiente formula.

$$velocidad = \left(\frac{(\text{rpms})*2\pi}{60}\right) * r \;\Rightarrow\; \left(\frac{(\text{rpms})*2\pi}{60}\right) * 24.1\ mm$$

Donde r equivale al radio de la rueda acoplado al encoder.

Posición de la rueda loca.

Para obtener la posición de la rueda loca se configuro el QEI en modo posición, en la configuración del código solo se agregó una línea de código.

```
QEIPositionSet(QEI1_BASE,0);
```

En esta configuración el modulo acumula el número de pulsos respecto a la posición de la rueda. Cada 512 pulsos el encoder habrá dado una vuelta y volverá a contar desde 0. Esto quiere decir que cada pulso el encoder se recorre 0.703125 grados. Cuando el encoder se encuentre a 180 grados de su posición inicial el modulo registrará 256 pulsos, 128 si el desplazamiento fueron 90 grados respecto al origen, estos valores son en sentido horario, ya que el modulo es capaz de detectar el sentido de giro si se gira en modo anti-horario 90 grados a partir de su origen el modulo registrara 384 pulsos y estos no serán modificados a menos que la posición cambie.

Las siguientes instrucciones obtienen la dirección y la posición que obtuvo el módulo QEI1 del encoder.

```
    position  =  QEIPositionGet(QEI1_BASE);
    direction =  QEIDirectionGet(QEI1_BASE);
```

La ecuación propuesta para determinar las variables se optó por desarrollarlas en la programación gráfica en vez de incluirlas al microcontrolador. De esta manera se disminuye la carga de procesar información. La información a procesar no requiere de una demanda estricta de manera que si se incluyera en el microcontrolador los cambios serian mínimos y poco notorios, pero se hace uso de técnicas de programación donde se asigna el procesamiento de datos al controlador con mayor capacidad en este caso el computador.

Interfaz de gráfica.

La interfaz gráfica se divide por defecto en dos partes: la primera hace referencia a la funcionalidad visual y el control sobre el sistema. Mientras que la segunda parte funge como la programación gráfica para mantener el procesamiento y comunicación de los datos con el microcontrolador. A continuación, se muestra la parte visual y de control de nuestra interfaz que podrá operar el usuario

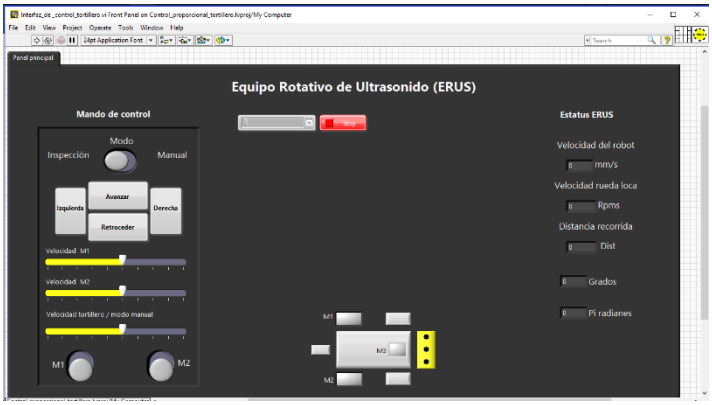


Figure 8. Interfaz gráfica desarrollada en LabView.

El mando de control contiene una botonera para manipular el robot en dos modos, inspección y manual. El modo inspección permite mover al robot con los botones “Avanzar, Retroceder, Izquierda, Derecha” cuando el robot es operado en esta modalidad el revolver gira de manera proporcional a la velocidad del robot y la velocidad del robot está condicionada a un máximo de 0.05 m/s. cada motor se puede regular su velocidad de manera independiente. La tercera barra regula la velocidad del revolver solo cuando el switch se encuentre en modo manual. Al igual que los switches M1 y M2 solo son funcionales cuando se encuentre en modo manual.

Del lado derecho podemos observar los indicadores que proporcionan la información del robot tales como la velocidad, distancia recorrida entre otras. A continuación, se mostrará la programación la programación de bloques donde se integraron las ecuaciones para obtener las variables a medir.

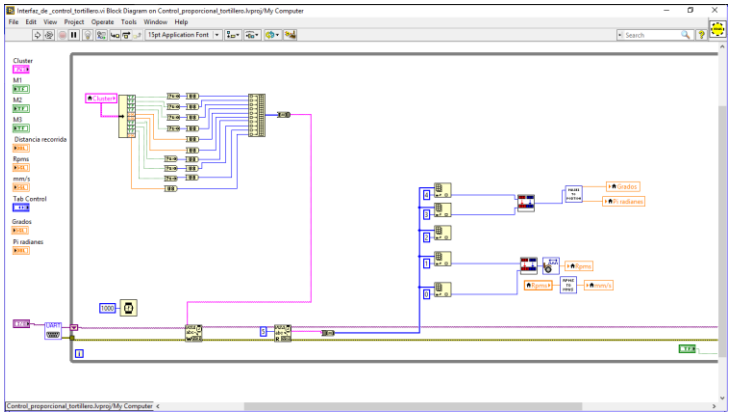


Figure 9. Programación de interfaz en Labiew.

Se mantiene un bucle while condicionado por el estado de un pulsador de paro que se visualiza en la interfaz. Antes de iniciar el bucle se configura la comunicación UART por la cual se hará la transición y recepción de datos, el bloque que con el nombre UART que se encuentra en la figura anterior es un SubVI con los parámetros cargados de nuestro puerto de comunicación. La siguiente figura muestra la programación que contiene el bloque UART.

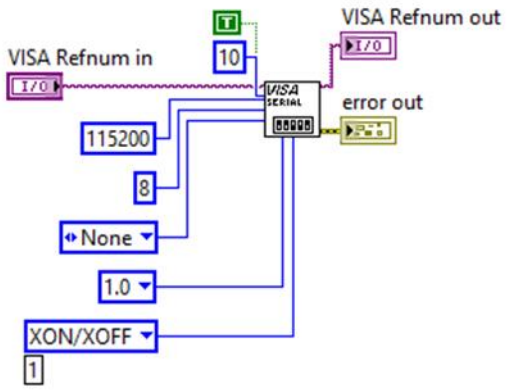


Figure 10. Configuración de protocolo de comunicación UART.

El puerto UART se configuro a una velocidad de 115 200 bauds, la longitud del dato es de 8 bits, sin dato de paridad, 1 bit de parada y el flujo de control es tipo XON/XOFF. El numero 10 que se puede observar indica una letra A en hexadecimal de esta manera genera un salto de linea al recibir cada dato de manera automatica. El tamaño maximo para nuestro buffer son 16 bytes, esto se encuentra condicionado por el microcontrolador que contiene una FIFO 16x8 ya que el buffer de labview puede ser mucho mayor.

La información que se manda desde la interfaz gráfica fue introducida en un Clusster de esta manera se tiene un mejor manejo de la información por medio del bloque unbundle todos los botones generan un valor booleano y los scroll valores enteros. Al tener datos booleanos y enteros es necesario homogenizar en un solo tipo de dato de esta manera el buffer no tendrá problemas con la comunicación. Observe como convierten todos a datos enteros de 8 bits con los siguientes bloques de de programación para después ser introducidos a un array que ocuparía 10 bytes de los 16 disponibles que tenemos en la configuración del buffer de nuestro microcontrolador, como último paso de conversión nuestro array de datos entero pasa a ser una cadena de caracteres con el bloque de función que se conecta al writte buffer.

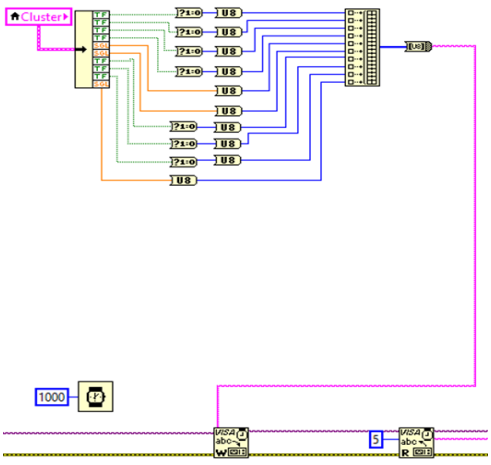


Figure 11. Agrupamiento de variables en clúster.

La recepción de los datos por ahora son la información que proporciona el encoder como los pulsos que entra en modo velocidad, modo posición y la dirección. La posición es un dato que aria de 0 a 512 pulsos dependiendo la posición del encoder, los pulsos de velocidad varían según la rapidez a la que gire el encoder y el dato que arroja la dirección es 1 y -1 según el sentido de giro.

Para enviar la información del microcontrolador se tuvieron que usar 5 bytes 2 bytes para la información de la posición ya que con 8 bits solo tenemos hasta 255

valores disponibles y para 512 valores disponibles utilizamos 16 bits, 8 bits como valor alto y 8 como valor bajo. Se aplicó el mismo método para la variable de velocidad y con la variable de dirección solo se utilizó 1 byte dejando 9 bytes disponibles para transmitir del microcontrolador a la PC. El microcontrolador tiene 2 memorias FIFO de 16x8 una de transmisión y la otra para la recepción de datos.

Las siguientes líneas de código distribuyen el valor de 16 bits a 2 bytes.

```
position_high = position >> 8;
 position_low = position & 0xFF;
 pulsos_high = pulsos >> 8;
 pulsos_low = pulsos & 0XFF;
```

para obtener los bytes de manera individual en labview se utilizao el bloque read buffer, esta entrega el tamaño del buffer como una cadena de caracteres. Es necesario obtener la información como un array de 5 bytes el array se descompone con el bloque de función Index array con este podemos usarlo para obtener el dato de cada byte de manera individual y trabajar con ellos.

Los datos de la variable posición y velocidad se adjuntaron con sus respectivos bytes. La siguiente figura muestra cómo se juntaron los bytes para formar el dato tipo Word sin afectar el resultado por el orden de bits.

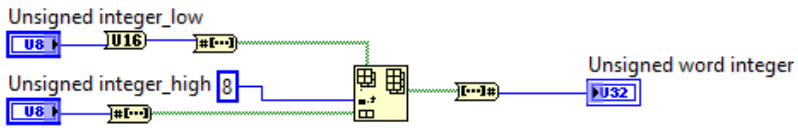


Figure 12. Concatenación de variable de 16 bits.

Linealización y control del revolver.

Para generar un control lineal se recurrió a utilizar la ecuación de la pendiente:

$$y = \text{xm} + \text{b}$$

$$(y2 - y1) = (x2 - x1)m + b$$

Los parámetros de la ecuación se obtuvieron con los siguientes datos:

- y: es el número de cuentas que será cargado al contador del módulo PWM, es decir este será el encargado de generar el ancho de pulso. Las cuantas van desde 0 hasta 1000, donde 0 es un 0% de PWM y 1000 un 100% de PWM.
- x: es el número de pulsos que entrega el módulo QEI en modo velocidad
- m es la pendiente para hacer nuestro control lineal.
- b: es el offset en el que se encuentra la función.

Usamos la segunda ecuación porque conocemos todos los valores de la ecuación a excepción de la constante de la pendiente.

Como y es el número de cuenta y tenemos un número máximo de cuentas y un valor mínimo podemos deducir que 999 es nuestro limite mayor, valor que sería asignado a y2 y por ende y1 tomaría el valor de 0. Pero el driver que utilizamos solo opera en un rango de 10% a 90% de PWM con una frecuencia definida, si el valor del PWM se encuentra fuera de estos valores los drivers se mantendrán en un estado default. Así que nuestro y2 tomaría el valor de 900 y y1 el de 90.

Para mi variable independiente es decir x mi mayor valor no está definido como los valores de y sin embargo sabemos que x1 siempre será cero porque mi valor mínimo no será el no tener ningún pulso y eso quiere decir que el robot no avanza y x2 estará cambiando constantemente. Se simularon en pruebas de laboratorio el número de pulsos que entra el encoder con Una velocidad de 0.05m/s dando como resultado 22 pulsos en promedio. El valor de offset se consideró en 0.A continuación se muestra la fórmula de la pendiente para obtener m.

$$m = \frac{(y2 - y1)}{(x2 - x1)} + b$$

$$m = \frac{(900 - 90)}{(22 - 0)} + 0 \qquad \Rightarrow \quad \frac{810}{22}$$

Ahora que conocemos el valor de nuestra pendiente podemos ingresar la ecuación de la pendiente sustituyendo valores ene l compilador para cargarla en el microcontrolador. La siguiente línea de código representa la ecuación en el compilador CodeComposer.

```
ui16Adjust = (pulsos*810) / 22;          //m = (y2-y1)/(x2-x1)
```

por último, se condiciona que el PWM siempre se mantenga dentro del rango de trabajo del driver 10 a 90% de señal PWM. Se utilizaron dos condiciones if para mantener el rango.

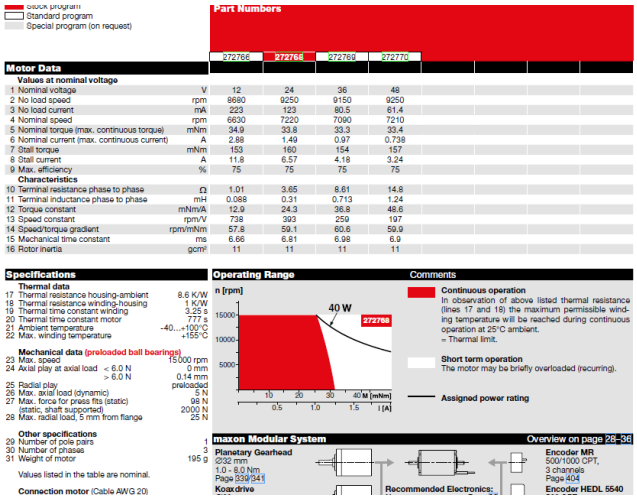
```
    if(pulsos>20){ui16Adjust=900;}
    if(pulsos==0){ui16Adjust=90;}
```

Para obtener más detalles de la programación consulte los anexos.

Configuración de los drivers.

Los motores que se encargan de operar el revólver y mover el ERUS son motores Maxon EC-max 272768 motores sin escobilla de manera que requieren de un driver que sea capaz de realizar las conmutaciones electrónicas en las bobinas que contiene el motor así como la lectura de los sensores de efecto Hall para disponer de su óptimo funcionamiento.

Se utilizaron driver ESCON 36/3 estos drivers ofrecen una amplia configuración de parámetros para motores sin escobillas. Mediante el software Escon Studio se configuraron con los parámetros que recomiendan la hoja de datos de los motores. Al ser tres motores del mismo modelo se usaron los mismos drivers con la misma configuración. La siguiente tabla muestra los parámetros adecuados para su funcionamiento esta fue recuperada de la hoja de datos del fabricante. Para más información consulte la documentación.



Stock program
Standard program
Special program (on request)

Part Numbers

	272766	272768	272769	272770
Motor Data				
Values at nominal voltage				
1 Nominal voltage V	12	24	36	48
2 No load speed rpm	8690	9250	9150	9250
3 No load current mA	223	123	80.5	61.4
4 Nominal speed rpm	6630	7220	7090	7210
5 Nominal torque (max. continuous torque) mNm	34.9	33.8	33.3	33.4
6 Nominal current (max. continuous current) A	2.88	1.49	0.97	0.738
7 Stall torque mNm	153	160	154	157
8 Stall current A	11.8	6.57	4.18	3.24
9 Max. efficiency %	75	75	75	75
Characteristics				
10 Terminal resistance phase to phase Ω	1.01	3.65	8.61	14.8
11 Terminal inductance phase to phase mH	0.069	0.31	0.713	1.24
12 Torque constant mNm/A	12.9	24.3	36.8	48.6
13 Speed constant rpm/V	738	393	259	197
14 Speed/torque gradient rpm/mNm	57.8	59.1	60.6	59.9
15 Mechanical time constant ms	6.66	6.81	6.98	6.9
16 Rotor inertia gcm²	11	11	11	11

Specifications

Thermal data
17 Thermal resistance housing-ambient 8.6 K/W
18 Thermal resistance winding-housing 1 K/W
19 Thermal time constant winding 3.25 s
20 Thermal time constant motor 777 s
21 Ambient temperature -40...+100°C
22 Max. winding temperature +155°C

Mechanical data (preloaded ball bearings)
23 Max. speed 15000 rpm
24 Axial play at axial load < 6.0 N 0 mm
> 6.0 N 0.14 mm
25 Radial play preloaded
26 Max. axial load (dynamic) 5 N
27 Max. force for press fits (static) 98 N
(static, shaft supported) 2000 N
28 Max. radial load, 5 mm from flange 25 N

Other specifications
29 Number of pole pairs 1
30 Number of phases 3
31 Weight of motor 195 g

Values listed in the table are nominal.

Connection motor (Cable AWG 20)

Operating Range — Comments

Continuous operation
In observation of above listed thermal resistance (lines 17 and 18) the maximum permissible winding temperature will be reached during continuous operation at 25°C ambient.
= Thermal limit.

Short term operation
The motor may be briefly overloaded (recurring).

Assigned power rating

maxon Modular System — Overview on page 28-36

Planetary Gearhead Ø32 mm 1.0 - 8.0 Nm Page 339/341
Koaxdrive
Recommended Electronics:
Encoder MR 500/1000 CPT, 3 channels Page 404
Encoder HEDL 5540

Figure 13. Especificaciones de motor EC-max 272768

También se recurrió a la hoja de datos del driver para encontrar el cableado eléctrico del motor, la alimentación del driver y sus pines de control. Cabe mencionar que se consultó todo el manual antes de conectarse para su correcto funcionamiento con parámetros como: alimentación, frecuencia de trabajo, consumos de corriente, corriente de suministro, potencia del driver, conexión eléctrica, rangos de trabajo en las señales de control, entre otros parámetros que especifica el manual a considerar antes de poner en funcionamiento. Para mayor información consulte la hoja de datos. Las siguientes figuras muestran un CAD básico del driver y su conexión con un motor sin escobillas.

4 Wiring

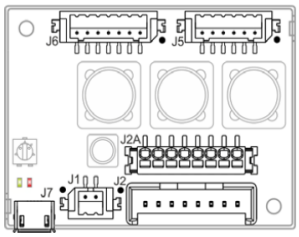


Figure 4-22 Interfaces – Designations and Location

Remark
The subsequent diagrams feature these signs:

- PCB mounting hole

Figure 14. Diagrama de conexiones driver.

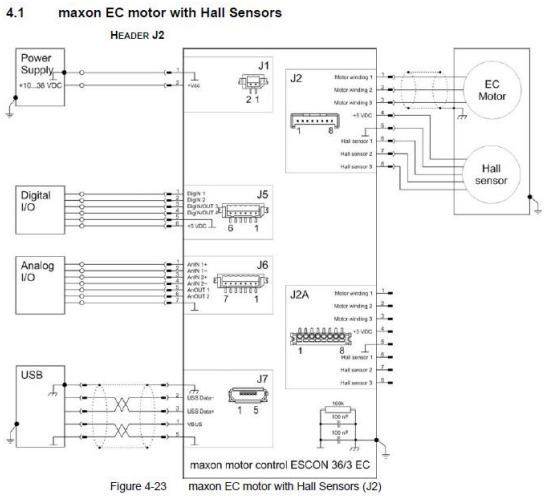


Figure 15. Diagrama de conexiones.

Se utilizaron el puerto J7 para monitorear el estado del drive por medio del software, así como su desempeño. El puerto J2 se conectó el motor. En el puerto J5 se conectaron las señales de control que provienen del tiva C así como las alimentaciones correspondientes. Y el puerto J1 se conectó a 24 VDC. La siguiente imagen muestra en los parámetros configurados en el motor.

Resultados.

Como interfaz final para el usuario, se mantuvo parecida a la proporcionada durante el desarrollo del proyecto, pero, se observó que en ocasiones no se tenía en claro el estado de los botones. Como solución final se agregó un clúster espejo de indicadores para proporcionar al usuario una mejor visualización de lo que está realizando en el momento de la operación del robot. La siguiente imagen muestra el resultado final de la interfaz.

Figure 16. Interfaz final para usuario.

El Core del microcontrolador se muestra a continuación el cual cuenta con los elementos mínimos pero necesario para el funcionamiento correcto, este PCB proporciona una gran optimización de espacio valiosa para la electrónica total del robot. La siguiente figura muestra la comparación de la tarjeta usada durante el desarrollo de las pruebas y el Core final que se destinará al proyecto.

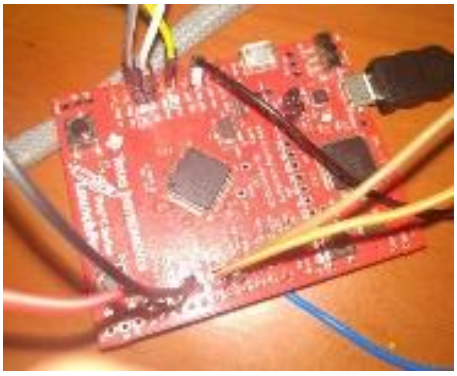


Figure 17. Tarjeta Tiva Launch Pad.

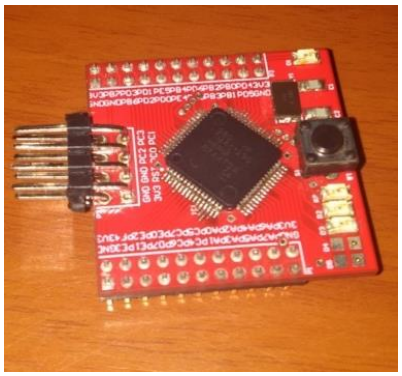


Figure 18. Core desarrollado para control de movimientos.

Conclusiones.

ERUS ahora puede ser operado mediante un controlador individual. La interfaz de operación se desarrolló en un principio con las instrucciones básicas para ejecutar la operación indicada, sin embargo nos fuimos percatando de que se requieran de condicionar acciones para la seguridad del robot y del usuario dando mejora en lo estético y funcional para la interfaz.

La primer interfaz no contaba con una estructura de “Maquina de estados” y provocaba un tráfico más lento de la información, dando como resultados obvios de operación que no podían ser aceptados para una operación adecuada y cómoda para el usuario.

Fue una buena elección usar el Tiva c debido a que el microcontrolador fuera de un procesamiento de información de 32 bits y eso ayudo a tener una velocidad aceptable en la comunicación UART así como el hecho de controlar el robot sin necesidad de usar microcontroladores secundarios comunicados entres si. Como suele hacerse con micrcontroladores de 8 u 16 bits.

Bibliografía

[1] Schlumberger, «How It Works Pipeline Pigging,» 2020 . [En línea]. Available: https://www.products.slb.com/resource-library/article/valve-academy/how-it-works-pipeline-pigging. [Último acceso: Julio 2020].

[2] National Instruments, «¿Qué es LabVIEW?,» 2020. [En línea]. Available: https://www.ni.com/es-mx/shop/labview.html. [Último acceso: 2020].

[3] M. Birchall, «Internal Ultrasonic Pipe & Tube Inspection - IRIS,» de *IV Conferencia Panamericana de END*, 2007.

[4] G. C. Briseño, «Desarrollo de un sistema electrónico para el control de desplazamiento de un robot de inspección para tuberías,» 2018.

[5] Horizon Industrial, «Horizon Industrial,» [En línea]. Available: https://www.horizonindustrial.com.au/images/pdf/What%20is%20a%20Pipeline%20Pig%20and%20how%20are%20they%20used.pdf. [Último acceso: 2020].

[6] MIKROE, «Mikro Electronika Books,» [En línea]. Available: https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/introduccion-al-mundo-de-los-microcontroladores. [Último acceso: 2020].

[7] Texas Instrments, «TM4C123G_LaunchPad_Workshop_Workbook».

[8] ESCON, «ESCON-36-3-EC-Hardware-Reference-En,» 2015.

[9] Maxon, «maxon EC-max,» 2017.

[10] Avago Technologies, «Incremental Magnetic Encoder-AEAT-601B,» 2009.

[11] Texa Instruments, «tm4c123gh6pm Data Sheet,» 2014.