**INSTITUTO POLITÉCNICO NACIONAL**

**CENTRO DE ESTUDIOS CIENTÍFICOS Y TECNOLÓGICOS No. 3**

**“ESTANISLAO RAMÍREZ RUÍZ”**

**SISTEMAS DE CONTROL ELÉCTRICO**

**INTEGRACIÓN DE SISTEMAS AUTOMATIZADOS**

**TESIS CURRICULAR**

**** BRAZO ROBÓTICO “D.I.R.I. CH-3001”

**QUE PRESENTA:**

* GUADALUPE MARTINEZ RICARDO
* MORENO JUAREZ IVAN
* ROMERO SOSA DANIEL
* SALGADO TAPIA PEDRO ISAAC

**DIRECTOR**

ING. JUAN IGNACIO LIMA VELASCO

**CODIRECTOR**

ING. LUIS CABRERA HERNÁNDEZ

ECATEPEC DE MORELOS EDO. DE MEX. JUNIO DE 2015

ÍNDICE

[Objetivo general. 1](#_Toc423347224)

[Objetivo específico. 1](#_Toc423347225)

[Justificación. 1](#_Toc423347226)

[Resumen. 1](#_Toc423347227)

[Introducción. 2](#_Toc423347228)

[Capítulo 1 Planeación del prototipo. 2](#_Toc423347229)

[1.1 Lluvia de ideas. 2](#_Toc423347230)

[1.2 Elección de prototipo. 3](#_Toc423347231)

[1.3 Diagrama de Gantt. 4](#_Toc423347232)

[1.4 Diagrama de flujo. 5](#_Toc423347233)

[Capítulo 2 Estructura del prototipo. 6](#_Toc423347234)

[2.1 Clasificación de los robots. 6](#_Toc423347235)

[2.2 ¿Qué es un brazo robótico? 6](#_Toc423347236)

[2.3 Tipos de Robots. 7](#_Toc423347237)

[2.4 Motores. 7](#_Toc423347238)

[Capítulo 3 Desarrollo del prototipo. 8](#_Toc423347239)

[3.1 Elaboración de trazos o esquemas. 8](#_Toc423347240)

[3.2 Desarrollo de prototipo cero. 9](#_Toc423347241)

[3.3 Armado de brazo robótico. 10](#_Toc423347242)

[3.4 Programación. 12](#_Toc423347243)

[3.5 Realización de circuitos en protoboard. 22](#_Toc423347244)

[3.6 Esquemas de circuitos electrónicos. 23](#_Toc423347245)

[3.7 Diseño de placas. 24](#_Toc423347246)

[3.8 Elaboración de placas fenólicas. 25](#_Toc423347247)

[3.9 Realización de maqueta. 26](#_Toc423347248)

[3.1.0 Pruebas finales. 28](#_Toc423347249)

[Conclusiones. 29](#_Toc423347250)

[Referencias 29](#_Toc423347251)

[Anexos 30](#_Toc423347252)

# Objetivo general.

Diseñar y construir un brazo robótico de 4 grados de libertad con un rango de alcance de 30 cm de tal manera que pueda ser programado mediante movimientos preestablecidos por potenciómetros.

# Objetivo específico.

* Elaborar planos estructurales y mecánicos.
* Diseñar y elaborar tarjetas electrónicas de control.
* Implementar servomotores para el movimiento del brazo.
* Utilizar el micro controlador PIC 18F4550 y 16F628A para el control electrónico.

# Justificación.

En la actualidad los brazos robóticos que se utilizan a nivel industrial, son muy costosos además que presentan dificultades para el manejo de estos ya que cuentan con un sistema de programación de bajo nivel; así que para cambiar su programación se necesita de una persona especializada en el tema, lo cual implica gastos y tiempo perdido.

La solución que se propone es realizar un brazo robótico que tenga una manera más sencilla de programar y realice movimientos simples pero eficaces.

# Resumen.

El prototipo que realizamos es un brazo robótico, el cual consta de 4 grados de libertad con un alcance de 30cm, este consta de una programación muy sencilla para que un usuario final lo pueda maniobrar con facilidad. Consta de una estructura de aluminio, así como de sus servomotores los cuales permiten el movimiento del brazo. Su programación será a base de movimientos que primero realice el usuario y después quedaran guardados en una memoria para que se repitan “n” veces. También se podrá usar en un modo libre para que el usuario interactúe con el brazo sin tener que guardar posiciones.

# Introducción.

En este trabajo demostramos en general él porque es que se llevó a cabo el proyecto del brazo robótico, donde demostramos todo el proceso que llevamos a cabo para obtener la idea especifica del proyecto así como la realización de los circuitos y del diseño del brazo también explicamos consideraciones teóricas para tener ideas más claras acerca de lo que consta la robótica, lo que es un robot, y todo acerca del brazo.

# Capítulo 1 Planeación del prototipo.

En este capítulo mencionamos algunas de las ideas inconclusas de lo que pudo haber sido el prototipo, también damos a conocer como logramos realizar el proyecto con las fechas planteadas al inicio del proyecto.

## Lluvia de ideas.

Primero comenzamos a pensar en distintos proyectos que podríamos llevar a cabo, como:

|  |  |  |
| --- | --- | --- |
| Idea | Características | Factibilidad |
| Drone | En este proyecto pudimos demostrar las habilidades que obtuvimos en la carrera uniendo todos los conocimientos | Al analizar el proyecto a futuro nos dimos cuenta de que su realización sería muy difícil y costosa. |
| Coche | Es un proyecto fácil desarrollar y además con pocos elementos de control que lo podemos aplicar de manera fácil | Este proyecto ya lo teníamos realizado y lo íbamos a mejorar pero nos dimos cuenta de que era muy simple para prototipo final. |
| Carro a control remoto como sirviente personal | Este proyecto lo pudimos utilizar montándole una cámara y ponerlo encima del mismo para inspeccionar lugares de difícil acceso | Esta idea tenía muchas desventajas pues teníamos que ver muchos factores del ambiente en donde se iba a desarrollar como las escaleras y lugares de difícil acceso. |
| Casa inteligente | Es un proyecto en el cual se puede aplicar la domótica al ámbito del hogar pues usa elementos de control un tanto avanzados para su uso. | El prototipo era muy factible y tenía gran posibilidad de que lo eligiéramos pero resultaba ser muy grande en cuanto a dimensiones. |
| Robot | Este proyecto abarca un parte de lo aprendido en la escuela, pero además abarca un tipo de conocimiento un tanto avanzado debido a su grado de complejidad. | Esta idea era muy avanzada ya que la programación, circuitos y estructura serían muy avanzados y costosos además que tardaríamos más tiempo porque teníamos que investigar muchos conceptos. |
| Brazo robótico | Es el proyecto perfecto al incluir en este la una amplia gama de conocimientos obtenidos de la carrera, además contiene una parte de robótica | Esta idea fue la más factible a pesar de que sería costosa por los motores y estructura del brazo. |

Pero nos dimos cuenta que la mayoría de las ideas pensadas no eran viables en lo económico, la programación era muy complicada y teníamos que considerar muchos aspectos en algunos proyectos o simplemente no eran posibles realizarlas pues no se tenía una idea clara de cómo hacerlo.

## Elección de prototipo.

Después de analizar todos los prototipos pensados, decidimos realizar el brazo robótico, esto debido a que cumplía mejor con la parte robótica que se nos estaba pidiendo al realizar el proyecto, además que tenía mayores ventajas, en lo económico, cierta facilidad de realizarlo, menor uso de tiempo en su elaboración.

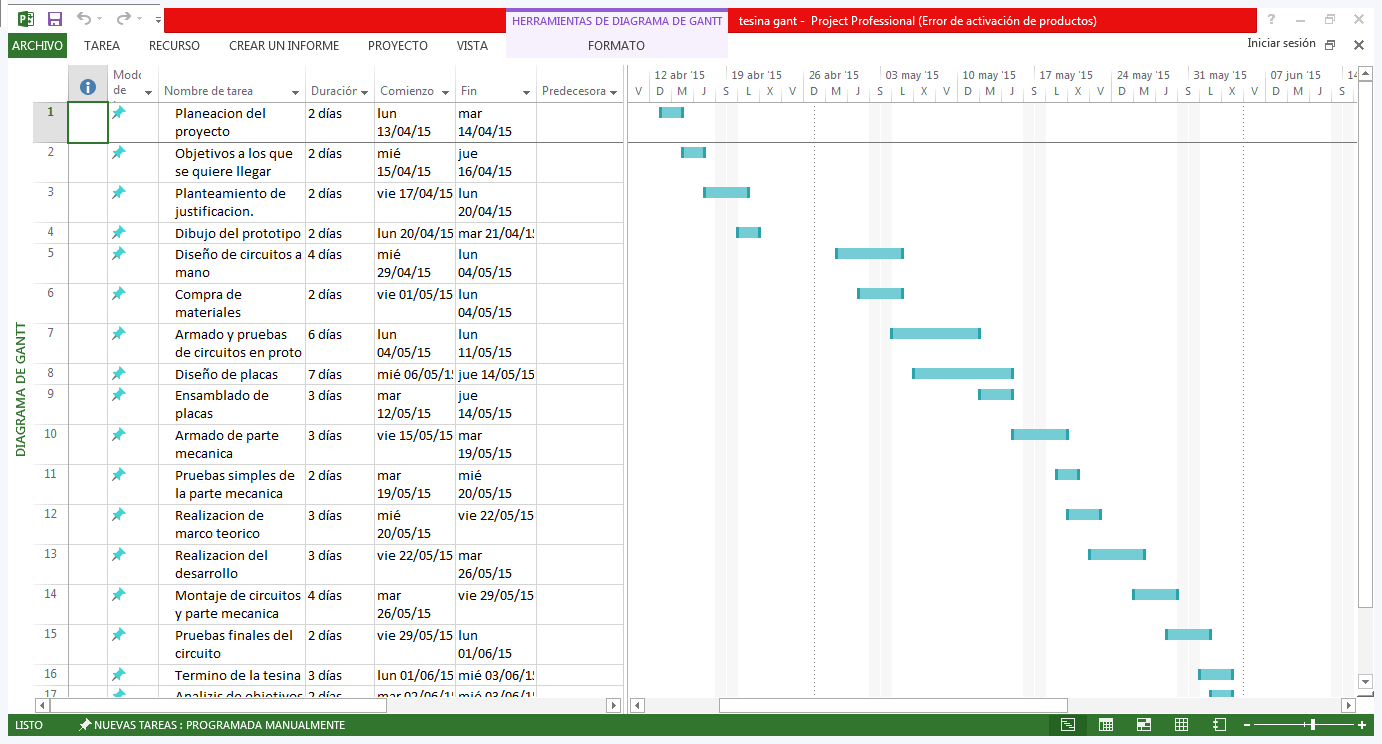
En esta decisión también tomamos en cuenta el factor del uso que se le iba a dar al mismo, es decir la factibilidad que tendría suponiendo que este llegara al mercado, si realmente tendría un uso útil y productivo para que pudiera competir en la solución de problemas, en donde su función tenga cabida.

## Diagrama de Gantt.

La elaborancion de el diagrama de Gantt es con el fin de llevar un orden de realizacion de cada paso del proyecto, darle uno fecha de inicio y final a cada actividad que se realizara en el proceso de la creacion del brazo.

Al realizar el diagrama de Gantt nos resulto un poco complicado fijarnos fechas para la realizacion de los objetivos a alcanzar pues no encontrabamos un orden para llevarlas a cabo o simplemente no sabiamos que relizar despues de acabarlo.

Planteandonos principalmente el inicio de todo y el final del proyecto para asi poder ir midiendo el tiempo y saber si hay que ir mas a prisa o no. En la figura 1.1 se muestra el orden que planteamos y seguimos para la realizacion del proyecto.



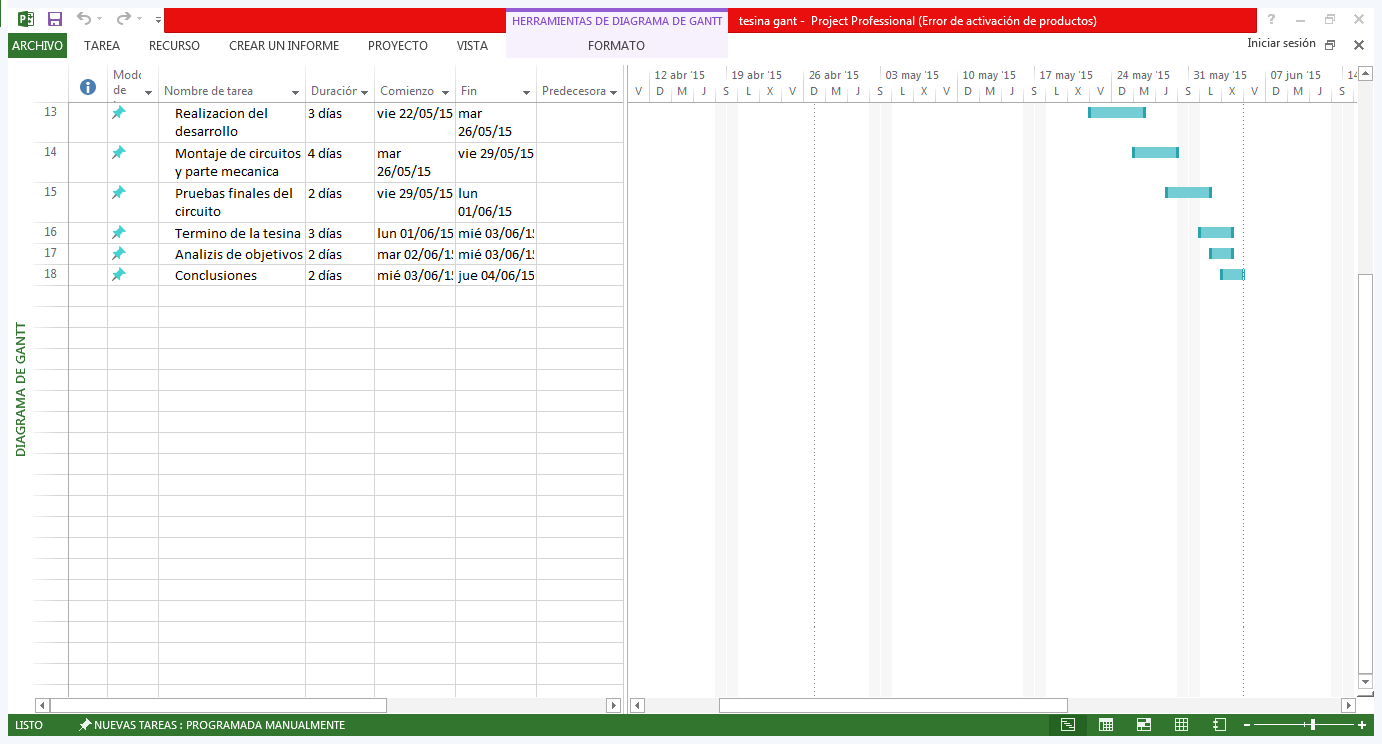
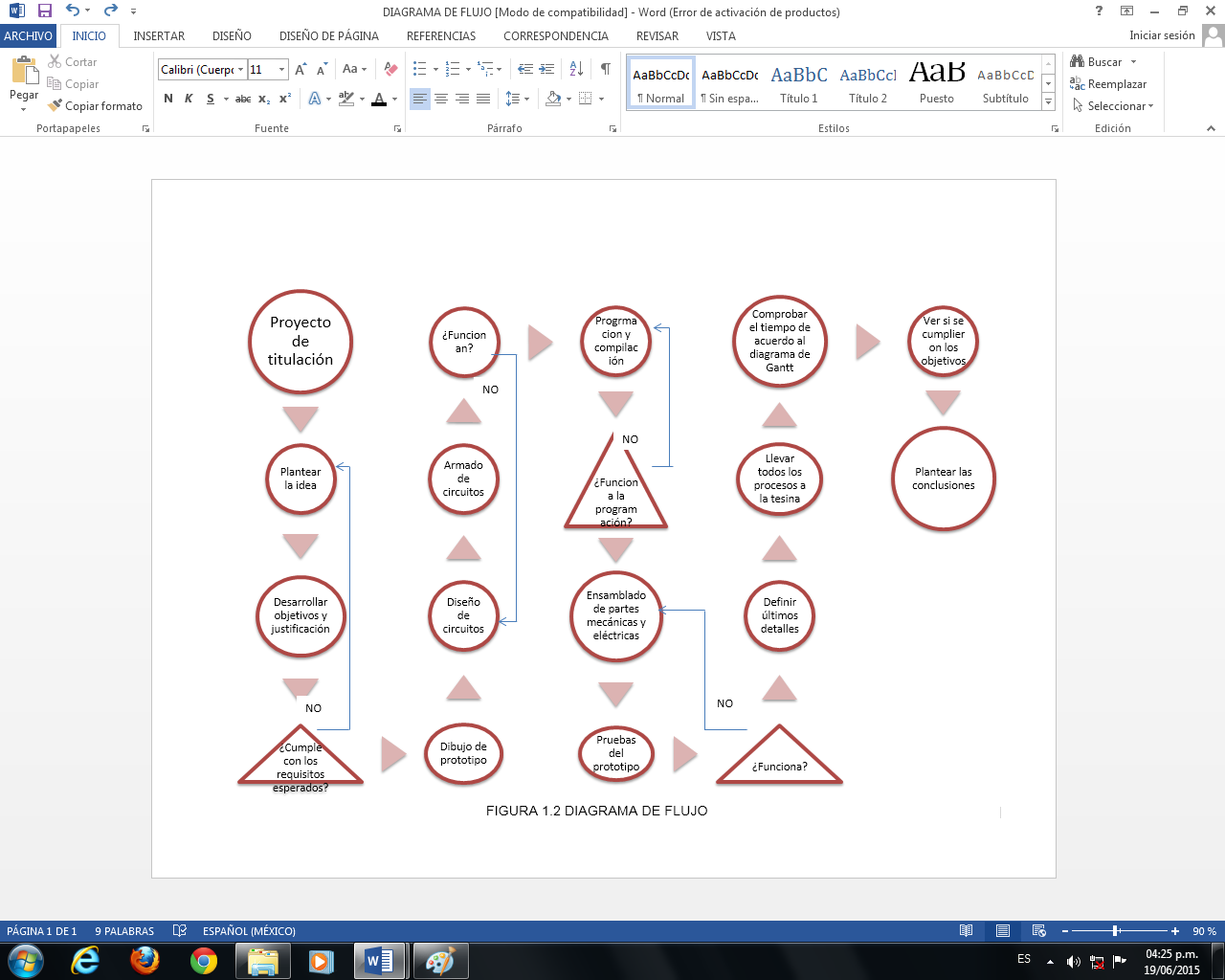


Figura 1.1 Diagrama de Gantt.

## Diagrama de flujo.

En la figura 1.2 mostramos el proceso que llevamos a cabo para elaboración del brazo, donde se toman en cuenta los pasos más importantes del proceso, también hacemos cuestiones acerca del mismo proceso para ver si se continua o se corrige algo.



# Capítulo 2 Estructura del prototipo.

La robótica es una rama que se dedica a la construcción, diseño, operación y aplicación de los robots. Un robot es una maquina automática programable capaz de realizar determinadas operaciones de manera autónoma y sustituir a los seres humanos en algunas tareas, en especial las pesadas, repetitivas o peligrosas; puede estar dotada de sensores, que le permiten adaptarse a nuevas situaciones.

## 2.1 Clasificación de los robots.

1ra Generación.

Manipuladores. Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.

2da Generación.

Robots de aprendizaje. Repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano.

3ra Generación.

Robots de control con sensores. El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

4ta Generación.

Robots inteligentes. Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso. Esto permite una toma inteligente de decisiones y el control del proceso en tiempo real.

## 2.2 ¿Qué es un brazo robótico?

Un brazo robótico es un tipo de brazo mecánico, normalmente programable, con funciones parecidas a las de un brazo humano; este puede ser la suma total del mecanismo o puede ser parte de un robot más complejo. Las partes de estos manipuladores o brazos son interconectadas a través de articulaciones que permiten, un movimiento rotacional (tales como los de un robot articulado) o un movimiento traslacional o desplazamiento lineal.

## 2.3 Tipos de Robots.

Robot cartesiano: Usado para trabajos de “pick and place” (tomar y colocar), aplicación de impermeabilizantes, operaciones de ensamblado, manipulación de máquinas herramientas y soldadura por arco. Es un robot cuyo brazo tiene tres articulaciones prismáticas, cuyos ejes son coincidentes con los ejes cartesianos.

Robot articulado: Usado para operaciones de ensamblaje, fundición a presión, máquinas de desbarbado, soldadura a gas, soldadura por arco, y pintado en spray. Es un robot cuyo brazo tiene como mínimo tres articulaciones rotatorias.

Robot paralelo: Uno de los usos es la plataforma móvil que manipula las cabinas de los simuladores de vuelo. Es un robot cuyos brazos tienen articulaciones prismáticas o rotatorias concurrentes.

Robot Antropomórfico: Similar a la mano robótica de Luke Skywalker que se le coloca al final de The Empire Strikes Back. Se le da forma para que pueda sustituir a una mano humana, p.e. con dedos independientes incluido el pulgar.

En nuestro caso utilizamos el robot tipo articulado, ya que es el mejor para la función que le vamos a dar, además que, para construir su estructura no necesitamos de amplios conocimientos mecánicos ya que son de lo más sencillo.

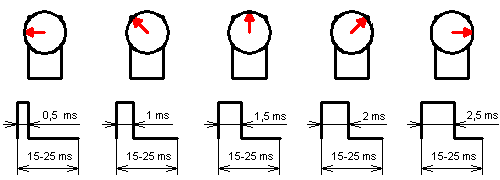
## 2.4 Motores.

El brazo se compone de 5 servomotores, los cuales le permiten realizar sus correspondientes movimientos, a esto se le conoce como los grados de libertad.

Un servomotor es un motor que tiene la capacidad de colocarse en cualquier posición dentro de su rango de operación, a este también se le puede controlar la velocidad que debe manejar.

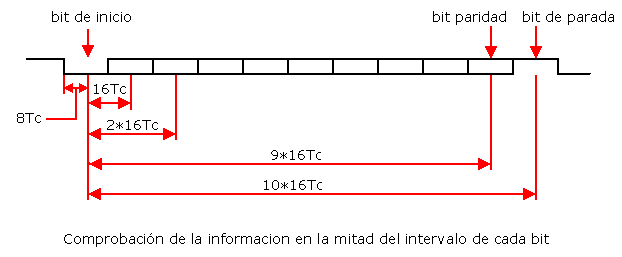
Este está conformado por un motor, un sistema reductor de velocidad y un circuito de control. Con anterioridad los servomotores no permitían que el motor girara 360 grados, solo aproximadamente 180; sin embargo, hoy en día existen servomotores en los que puede ser controlada su posición y velocidad en los 360 grados.

Los servomotores hacen uso de la modulación por ancho de pulsos (PWM) para controlar la dirección o posición de los [motores de corriente continua](http://es.wikipedia.org/wiki/Motor_de_corriente_continua). La electrónica dentro del servomotor responderá al ancho de la señal modulada. Si los circuitos dentro del servomotor reciben una señal de entre 0,5 a 1,4 milisegundos, éste se moverá en sentido horario; entre 1,6 a 2 milisegundos moverá el servomotor en sentido anti horario; 1,5 milisegundos representa un estado neutro para los servomotores estándares.

Figura 2.1 Control de servos.

La comunicación serial

La comunicación serial consiste en el envío de un bit de información de manera secuencial, es decir, un bit a la vez y a un ritmo acordado entre el emisor y el receptor véase figura 2.2.

Figura 2.2 Comunicación serial

# Capítulo 3 Desarrollo del prototipo.

## 3.1 Elaboración de trazos o esquemas.

Ya teniendo la idea clara de lo que se iba a desarrollar, es decir la elaboración de un brazo de unos 30 cm de alcance aproximadamente, nos dimos a la tarea de diseñarlo, trazando las distintas partes de este, a dimensiones reales de lo que se buscaba, empezando con el gripper y la muñeca como se muestra en la figura 3.1 donde previamente, hicimos una investigación acerca de las distintas partes que componen el brazo y con base en ello deducimos el tipo y tamaño de motor a utilizar.

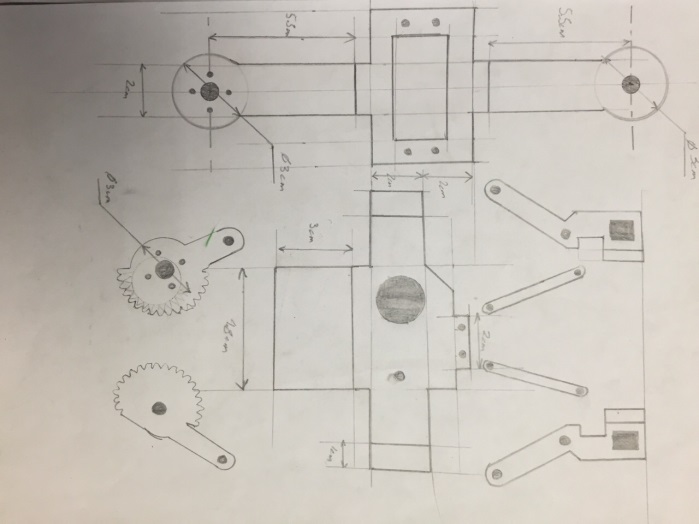


Figura 3.1 Muñeca y gripper.

Luego diseñamos las bases del brazo que lo podríamos relacionar con el hombro y codo de un brazo humano, esto se puede apreciar en la figura 3.2.

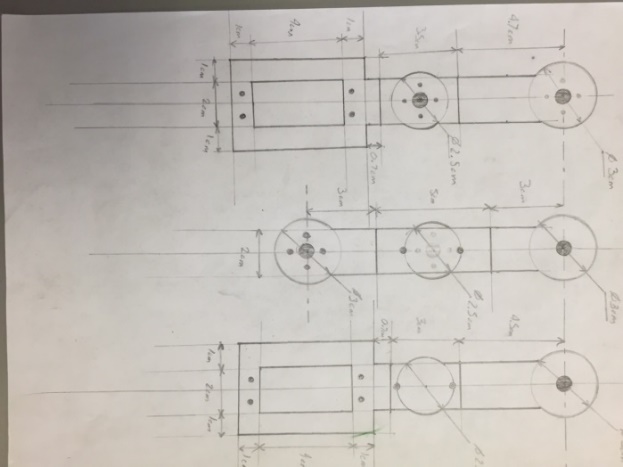


Figura 3.2 Partes laterales.

## 3.2 Desarrollo de prototipo cero.

Con los trazos del brazo ya realizados, procedimos a realizar un prototipo cero, para comprobar que el modelo que estábamos diseñando fuera el correcto, es decir, cumpliera con los movimientos que requeríamos, fuera ágil, practico en su estructura, cumpliera con los grados de libertad requeridos y que fuera posible construirlo en otro material.

El material que utilizamos para desarrollar el prototipo cero fue cartón en su estructura y tornillos para las uniones, de esta manera le otorgábamos al brazo una libertad de movimiento, para encontrar las fallas que pudieran presentarse en la estructura final, el resultado final del prototipo cero se muestra en la figura 3.3.

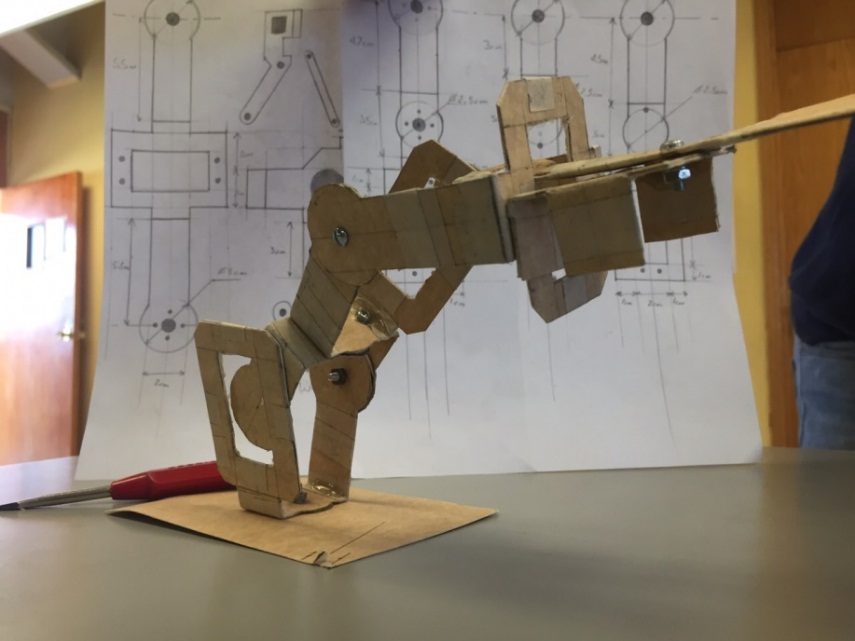


Figura 3.3 Prototipo cero.

## 3.3 Armado de brazo robótico.

Ya teniendo el diseño final del brazo robótico y después de encontrar fallas y corregirlas en el prototipo cero lo que procedió fue mandarlo a construir en un material más resistente pero que a la vez fuese liguero para que los movimientos que realizara fueran rápidos y no tuviera mucha fricción al girar, como en el prototipo cero por esto se decidió que se las piezas fueran de aluminio, en la figura 3.4 se muestra la estructura final.



Figura 3.4 Partes.

Después procedimos a pintar las piezas de color negro como se muestra en la figura 3.5 pues consideramos que este color le daría una buena presentación y elegancia a la estructura del brazo



Figura 3.5 Proceso de pintura.

Después procedimos a ensamblar todas las piezas del brazo y ajustar los servomotores en las partes correspondientes, quedando como se muestra en la figura 3.6 la estructura final del brazo.



Figura 3.6 Estructura.

## 3.4 Programación.

Desarrollamos tres distintos modos de programación en el software “PIC C compiler”, estos modos van enfocados al funcionamiento y uso del brazo.

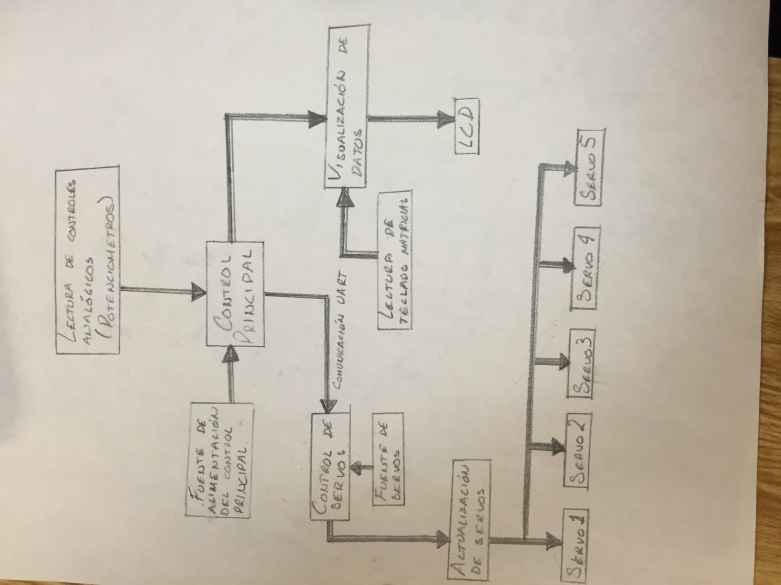


Figura 3.7 Modos.

Lo primero que tuvimos que tener en cuenta es solucionar el problema a través de tres modos de uso:

* Modo A “Modo libre”:
  + Este modo consiste en leer el valor de potenciómetros a través de canales analógico-digital, actualizar las salidas que van conectadas a los servomotores para que estos se muevan a la par de los potenciómetros logrando la coordinación entre ambos.
* Modo B “Modo programación”:
  + Este modo es similar a el “Modo libre” la diferencia es que en este modo el usuario tiene que ir guardando las posiciones que desee guardar en una rutina que va hasta 500 posiciones otorgándole una pausa a cada uno de los movimientos.
* Modo C “Modo ejecución”
  + Este es el último modo en el cual el programa realiza los movimientos antes guardados en el modo programación, esto se realiza en base al método de ejecutar que se guardaron en la memoria “EEPROM” del pic, el programa se muestra en la figura 3.8

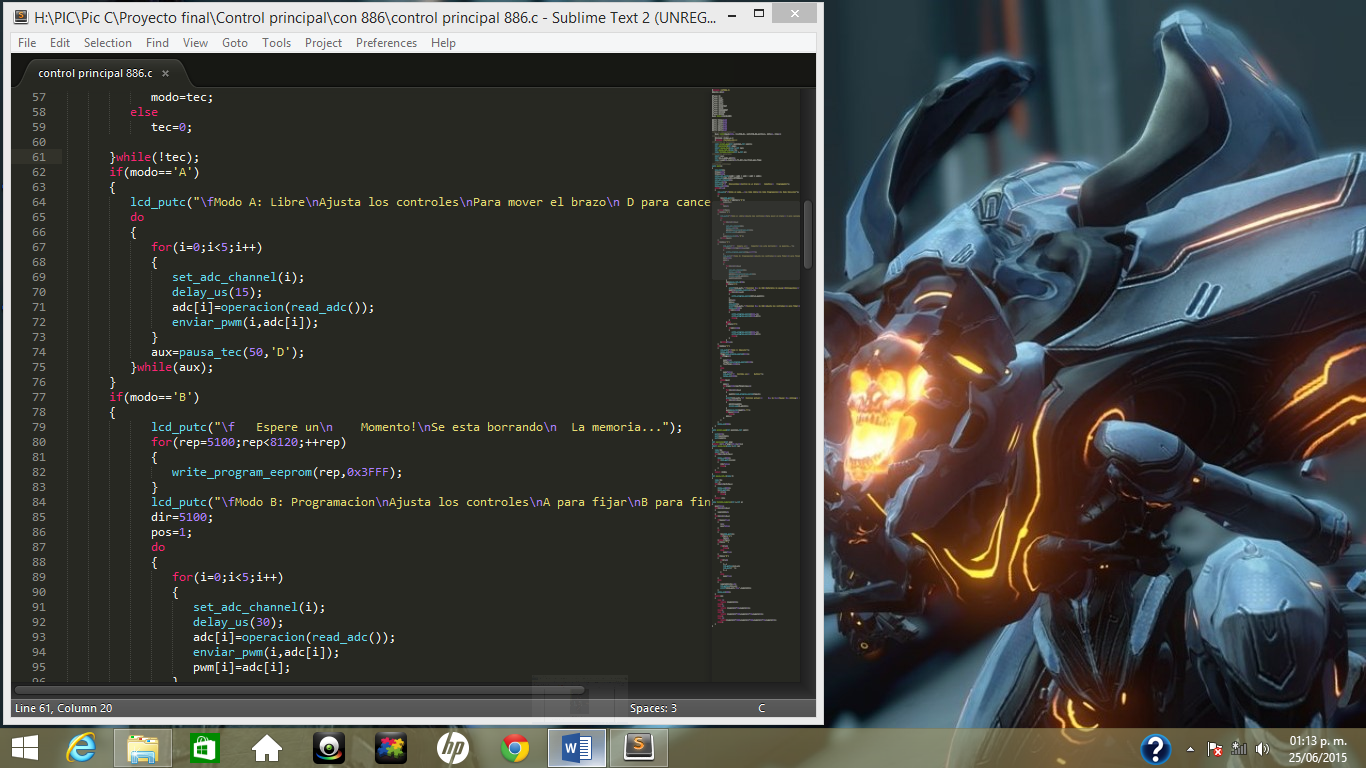


Figura 3.8 Programa en C.

También se desarrolló el programa de control de los servomotores para tener un mejor manejo en los movimientos del brazo.

La forma en que se realizó el programa de los servomotores fue a través de dos interrupciones por medio de dos “timer”:

El primer timer controla el ancho de pulso que se actualiza aproximadamente cada 24.5 us, y para lo que nos sirve es para que cada overflow (Desbordamiento del timer) verifique una variable llamada “control pwm”, la compara con el valor de pwm del servomotor en específico y en caso de que estas sean iguales manda el valor a cero. Véase en la figura 3.9

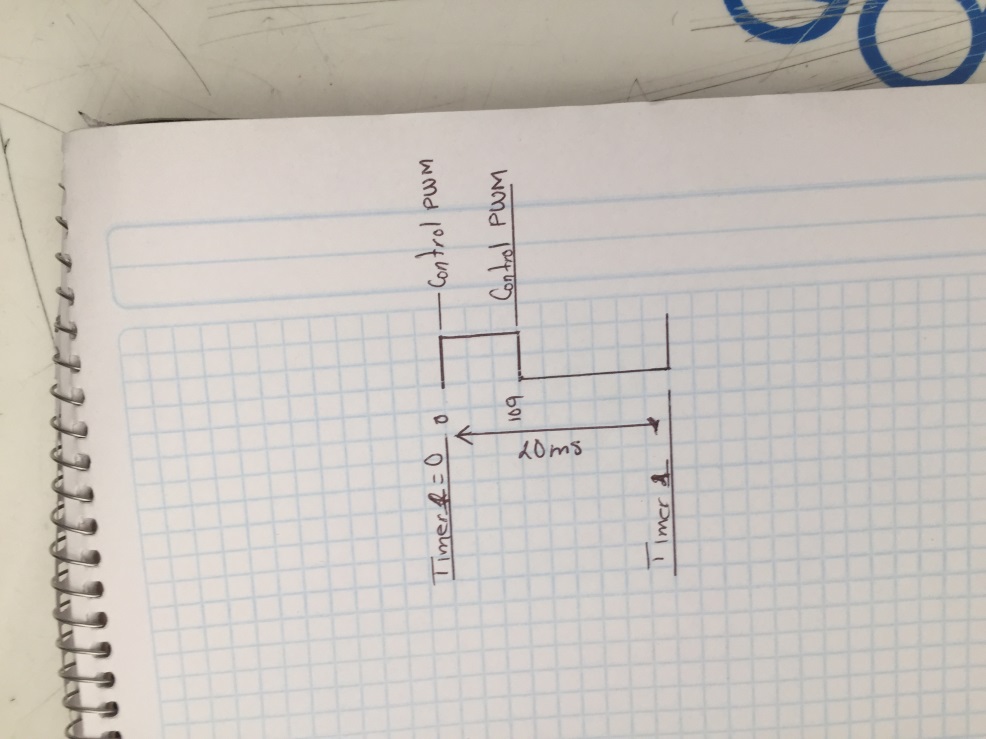


Figura 3.9 Explicación de PWM.

Placa de control de servos

Especificaciones generales: Este circuito servirá para el control de 5 servomotores que con controlarán la totalidad de los movimientos de nuestro brazo robótico.

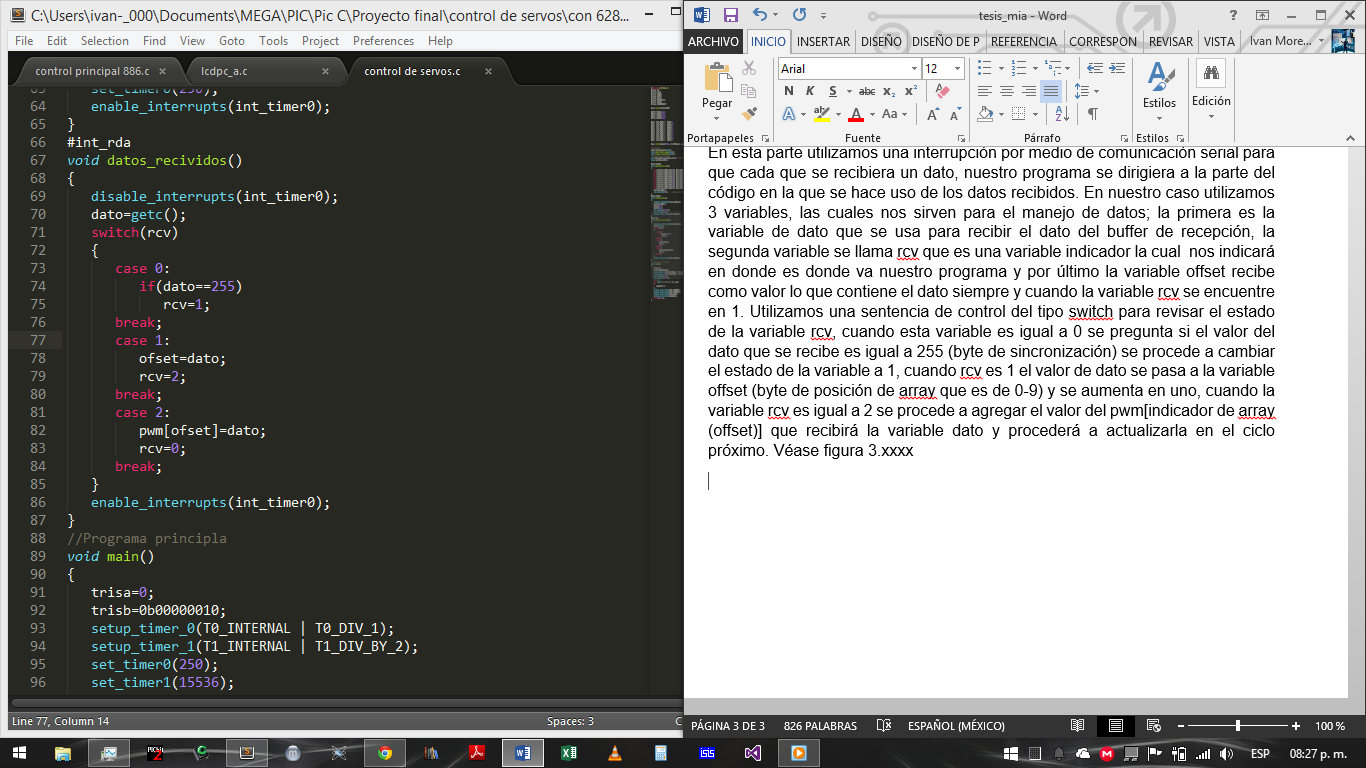
El funcionamiento: La placa actualizará la salida del pwm para los servomotores que es el que realmente determinará la posición de los servos, la actualización de datos se hará por medio de comunicación serial la que nos dará el tiempo de subida que tendrán las salidas de nuestros servomotores.

Figura 3.1.1 Comunicación.

El código de la placa controladora de servos

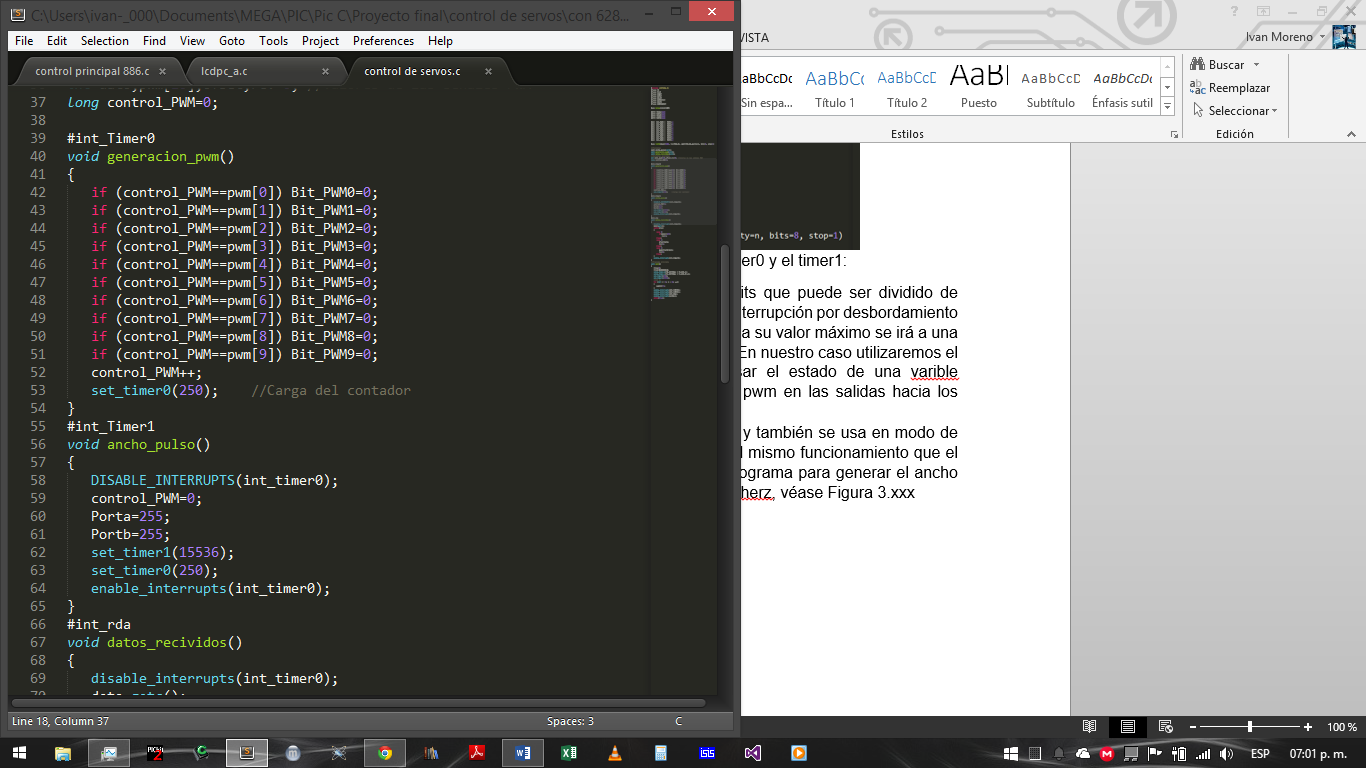
La primera parte de nuestro código se refiere al uso de los fusibles de configuración para nuestro microcontrolador que en es te cado es un PIC16F628A, seguido encontramos los valores de los puertos; tanto el A como el B con sus respectivos “TRIS” también encontramos la declaración de los “bits” de pwm para el control de nuestro ancho de pulso y por ultimo encontramos el uso de la comunicación.

El control del pwm se usaron 2 timers; el timer0 y el timer1:

* El timer0: El tamier0 es un timer de 8 bits que puede ser dividido de distintos múltiplos y se usará en modo de interrupción por desbordamiento del mismo timer, es decir, cada que llegue a su valor máximo se irá a una secuencia que ya la tenemos predefinida. En nuestro caso utilizaremos el timer0 como desbordamiento para revisar el estado de una variable llamada control pwm con el valor de los pwm en las salidas hacia los servomotores, véase Figura 3.xxx
* El timer1: el timer1 es un timer de 16 bits y también se usa en modo de desbordamiento del mismo timer y tiene el mismo funcionamiento que el timer0. El timer1 lo usamos en nuestro programa para generar el ancho de pulso y así lograr una frecuencia de 50 Hertz, véase Figura 3.1.2

Figura 3.1.2 Timer 1.

Cuando usamos el timer0 y el timer1 lo hacemos de tal forma de que se produzca un desbordamiento cada 24.5us en cuanto al timer0 y por eso se usa al final la instrucción de *set\_timer0(250),* esto se debe a que si hacemos las operaciones tenemos que el tiempo de cada conteo estará determinado por la siguiente formula : Tiempo=1/(Oscilador/4) y esto nos devuelve un resultado en segundos, debido a que tenemos un oscilador de 20mhz nos devuelve un tiempo de 2x10-7 segundos que son 200ps, ahora cada conteo en el timer0 será de un tiempo de 200ps y por lo tanto se producirá una interrupción de cada 1us y si timamos en cuenta los tiempos de chequeo de las variables nos dará en total 24.5us.



Uso de la comunicación serial para cargar las variables de pwm

En esta parte utilizamos una interrupción por medio de comunicación serial para que cada que se recibiera un dato, nuestro programa se dirigiera a la parte del código en la que se hace uso de los datos recibidos. En nuestro caso utilizamos 3 variables, las cuales nos sirven para el manejo de datos; la primera es la variable de dato que se usa para recibir el dato del buffer de recepción, la segunda variable se llama rcv que es una variable indicador la cual nos indicará en donde es donde va nuestro programa y por último la variable offset recibe como valor lo que contiene el dato siempre y cuando la variable rcv se encuentre en 1. Utilizamos una sentencia de control del tipo switch para revisar el estado de la variable rcv, cuando esta variable es igual a 0 se pregunta si el valor del dato que se recibe es igual a 255 (byte de sincronización) se procede a cambiar el estado de la variable a 1, cuando rcv es 1 el valor de dato se pasa a la variable offset (byte de posición de array que es de 0-9) y se aumenta en uno, cuando la variable rcv es igual a 2 se procede a agregar el valor del pwm[indicador de array (offset)] que recibirá la variable dato y procederá a actualizarla en el ciclo próximo. Véase figura 3.1.3.

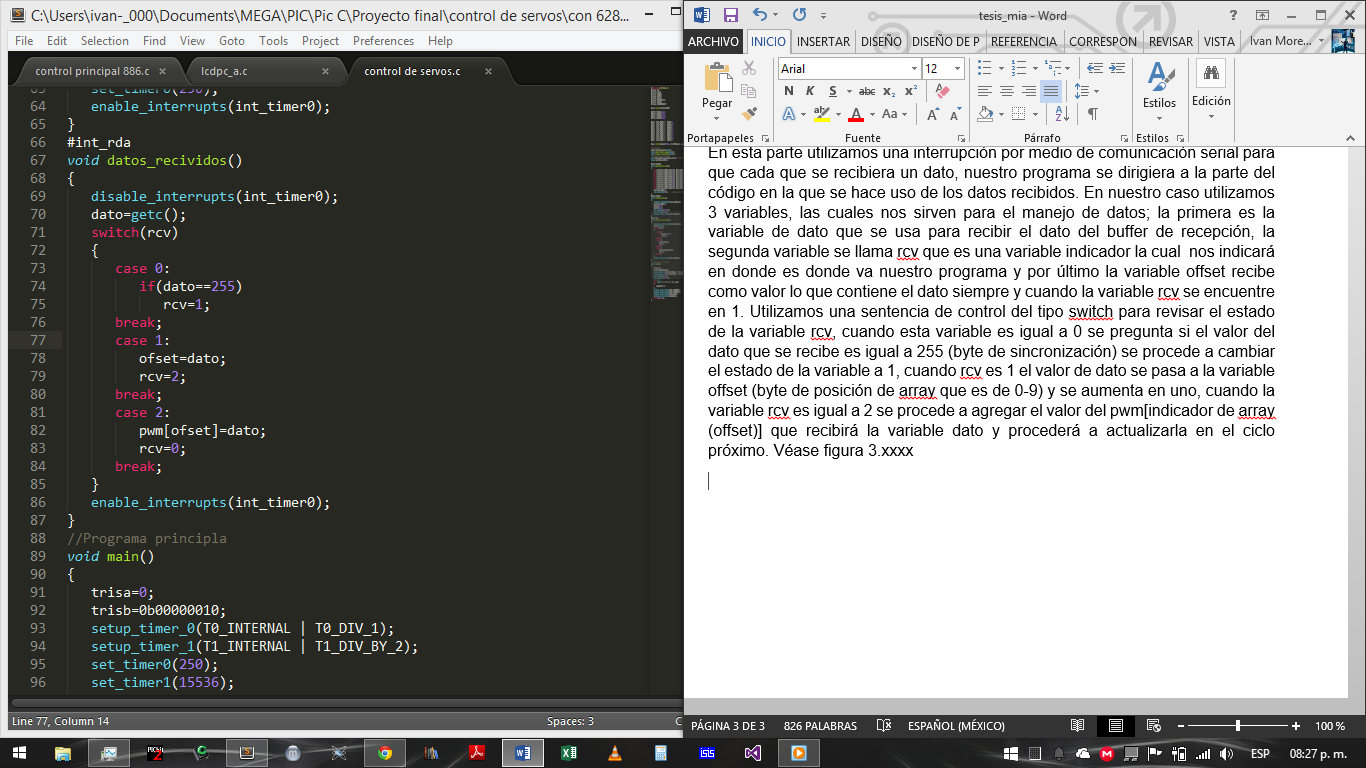


Figura 3.1.3 Comunicación 2



Figura 3.1.4 Modos de uso.

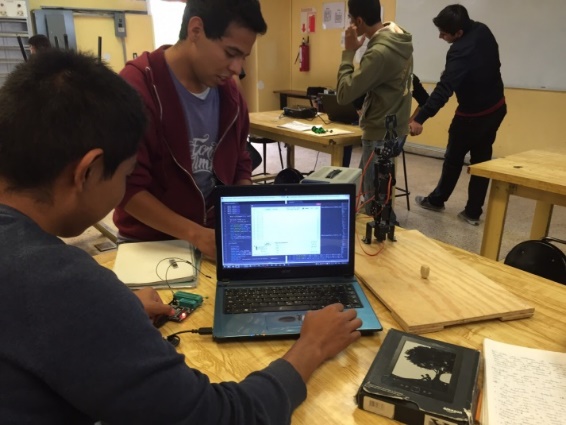


Figura 3.1.5 Control de servomotores.

**Placa del control principal**

**Especificaciones generales:**

Esta placa servirá para realizar la interacción con el usuario y además gestionar las posiciones de los servomotores en la memoria.

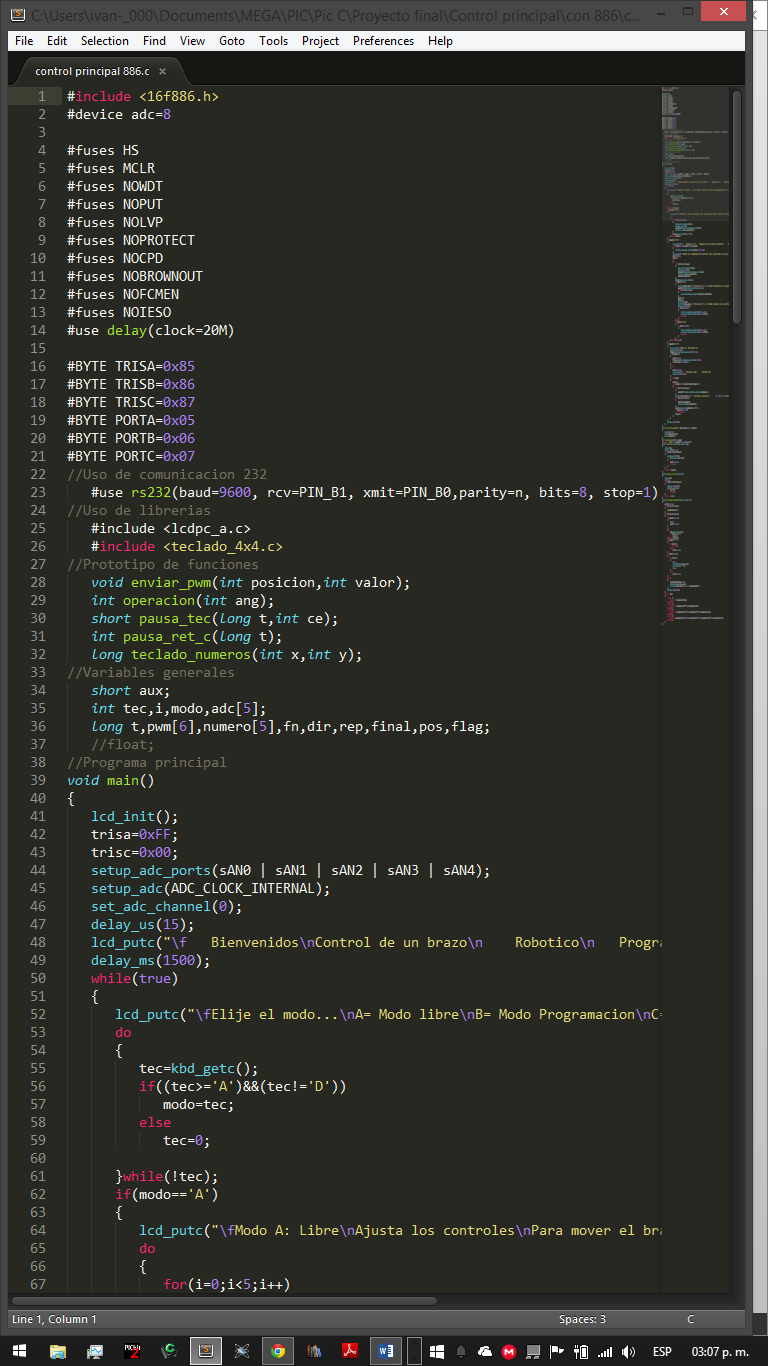
**El funcionamiento:**

Primero empezamos mostrando un mensaje de bienvenida para el usuario, después dependiendo dela opción que el usuario introduzca se ejecutaran unas líneas de código:

* Para el modo A “Libre”: Se ejecutara el modo libre y su funcionamiento consiste en seleccionar un canal del convertidor análogo-digital, leerlo y por último enviar el dato recibido después de hacer la conversión a los niveles del servomotor.
* Para el modo B “Programación”: Primero se borrara la memoria de programa para poder introducir una nueva serie de rutinas y después se estabilizarán los controles para que funcionen en paralelo a los servomotores y estarán en actualizando hasta que se presione la tecla “A” que dejara de actualizar a los servomotores y el control pedirá una pausa que estará dada en milisegundos una vez que esta se introduzca se procederá a guardarse los datos en la memoria interna del PIC y al final cuando se presione la tecla “B” se finalizará este modo y grabará la última posición en la memoria que se guardó.
* Para el modo C “Ejecución”: Este modo depende del modo B pues si no se ha grabado ninguna posición anterior el control pedirá que se grabe una rutina, de lo contrario se procederá a ejecutarse las posiciones de la memoria y se cargaran los datos en la tarjeta controladora de servos para después esperarse un tiempo definido en el modo B, esto lo realizará durante ‘n’ veces.

**El código de la placa principal**

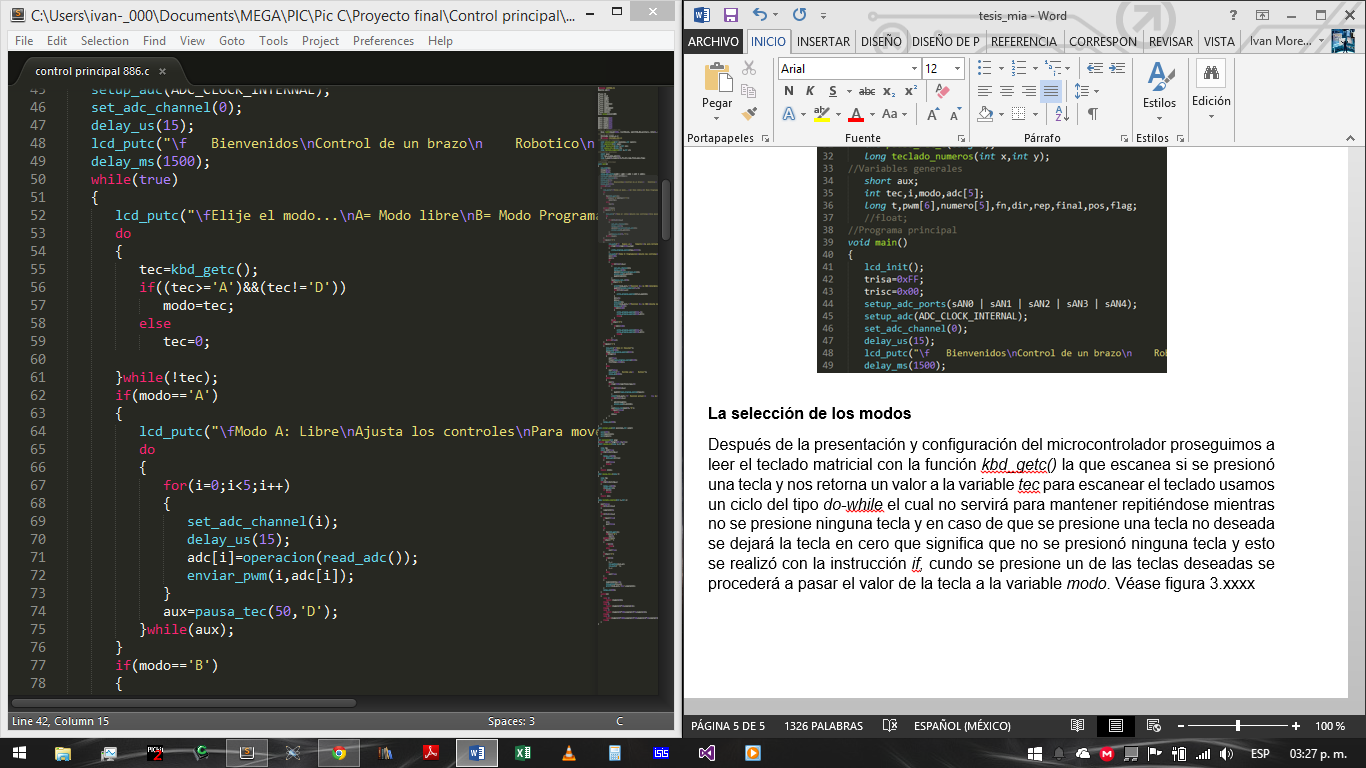
La primera parte del código consiste en la declaración de las directivas de pre procesamiento, es decir, declarar los fusibles, la dirección de los puertos y sus respectivos tris, uso de la comunicación serial con sus respectivos parámetros, el uso de las librerías, la declaración de las funciones que usamos, la declaración de las variables generales y por ultimo ya dentro de la parte central (main) se puede ver la función lcd\_init() que es parte de la librería para el control de una pantalla lcd, después estoy iniciando los valores para los registros tris seguido de la configuración del convertidor analógico-digital y mostramos un mensaje de bienvenida. Véase figura 3.1.6

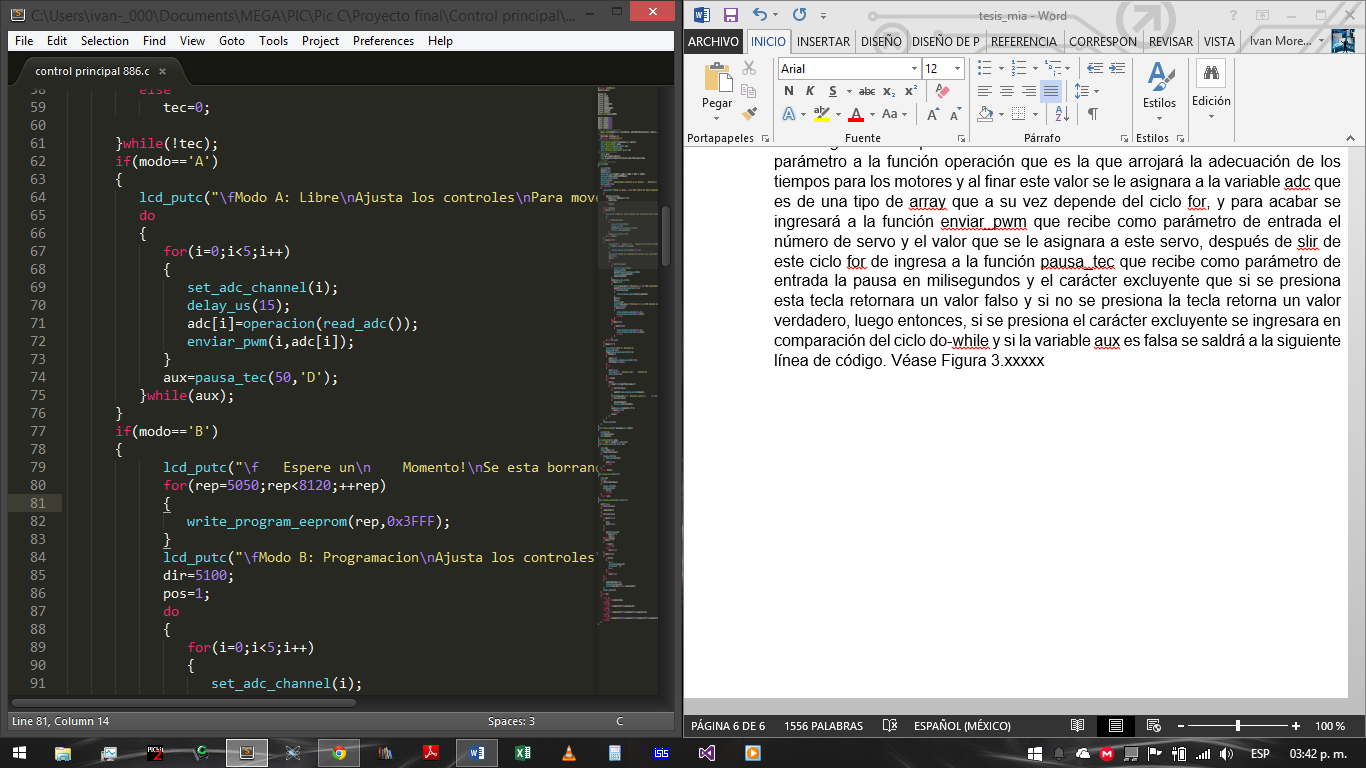
Figura 3.1.6 Control placa principal

**La selección de los modos**

Después de la presentación y configuración del microcontrolador proseguimos a leer el teclado matricial con la función *kbd\_getc()* la que escanea si se presionó una tecla y nos retorna un valor a la variable *tec* para escanear el teclado usamos un ciclo del tipo *do-while* el cual no servirá para mantener repitiéndose mientras no se presione ninguna tecla y en caso de que se presione una tecla no deseada se dejará la tecla en cero que significa que no se presionó ninguna tecla y esto se realizó con la instrucción *if,* cundo se presione un de las teclas deseadas se procederá a pasar el valor de la tecla a la variable *modo*. Véase figura 3.1.7

**Figura 3.1.7 Selección de modos**

**El modo A “Libre”**

Al entrar en este modo se mostrará un mensaje en la pantalla *lcd*, seguido de un ciclo *do-while* que dependerá de la variable *aux*, después de entrar en un ciclo *do-while* se entrara en un ciclo *for* el cual se repetir 5 veces (del 0 al 4), ya en el ciclo *for* se seleccionara un canal analógico-digital y se demorara 15 microsegundos después lo leído en el canal seleccionado entrara como parámetro a la función operación que es la que arrojará la adecuación de los tiempos para los motores y al finar este valor se le asignara a la variable *adc* que es de una tipo de *array* que a su vez depende del ciclo *for*, y para acabar se ingresará a la función *enviar\_pwm* que recibe como parámetro de entrada el número de servo y el valor que se le asignara a este servo, después de salir de este ciclo *for* de ingresa a la función *pausa\_tec* que recibe como parámetro de entrada la pausa en milisegundos y el carácter excluyente que si se presiona esta tecla retornara un valor falso y si no se presiona la tecla retorna un valor verdadero, luego entonces, si se presiona el carácter excluyente se ingresara en comparación del ciclo *do-while* y si la variable *aux* es falsa se saldrá a la siguiente línea de código. Véase Figura 3.1.8

**Figura 3.1.8 Modo A**

**El modo B “Programación”**

Cundo se seleccione este modo se procederá a mostrar un mensaje una la pantalla *lcd* y después se borrará la memoria del programa a partir de un ciclo *for* donde su inicio es la posición de la memoria 5100 y su final de 8120 a partir de la instrucción *write\_program\_eeprom* que recibe 2 parámetros de entrada los cuales con la posición en memoria y el valor asignado esa posición que en nuestro caso el valor de borrado es en hexadecimal 0x3FFF, después de borrar la memoria de proceder a desplegar otro mensaje en la pantalla y se cargan 2 variables las cuales son: la variable *dir* que es la que se va a encargar la dirección en la memoria del *pic* y la variable pos la cual se encargará de establecer las posiciones que el usuario podrá grabar. Después se entrará a un ciclo *do-while* el cual será infinito, una vez en el ciclo *do-while* se entra a la estructura del modo A, después del ciclo *for* se entra en la función *pausa\_ret\_c* que tiene como parámetro de entrada la pausa en milisegundos y como parámetro de salida l tecla que se presionó, una vez que se presionó la tecla se verifica si una de ellas es “A” o “B”, se realizara una acción que dependerá de la tecla que se presionó, en caso de que se presione:

* La tecla “A”: Se desplegará un mensaje que indica las posiciones que se tienen usadas y disponibles así como la pausa que introducirá el usuario una vez que se ha introducido la pausa se le dará el valor al *array* *pwm[5]*, luego se procede a escribir en la memoria del microcontrolador a través de un ciclo *for* de 0 a 5 que se hará por medio de la instrucción *write\_program\_eeprom* que recibe como parámetro de entrada la posición *dir* mas el valor de la variable i y el valor del *array* pwm, una vez escrita la memoria del programa de sumar 6 a la variable *dir* y se incrementara en 1 la variable pos después se inicializa la pantalla *lcd*, ya que puede haber una error debido a la multiplexación de la pantalla con el teclado y se desplegará un mensaje en la pantalla que nos dice las posiciones que el usuario lleva, seguido de esto se verificará que no se exceda la última posición y si excede se grabará la última posición grabada así como el byte de verificación de grabado y por último se saldrá del ciclo *do-while* y en caso de que no se exceda esta memoria se seguirá corriendo el programa. Véase figura 3.1.9
* La tecla “B”: Si se presiona la tecla “B” se escribirá en la memoria la última posición grabada y el byte de testeo de grabado y por último se saldrá del ciclo *do-while*. Véase figura 3.1.9

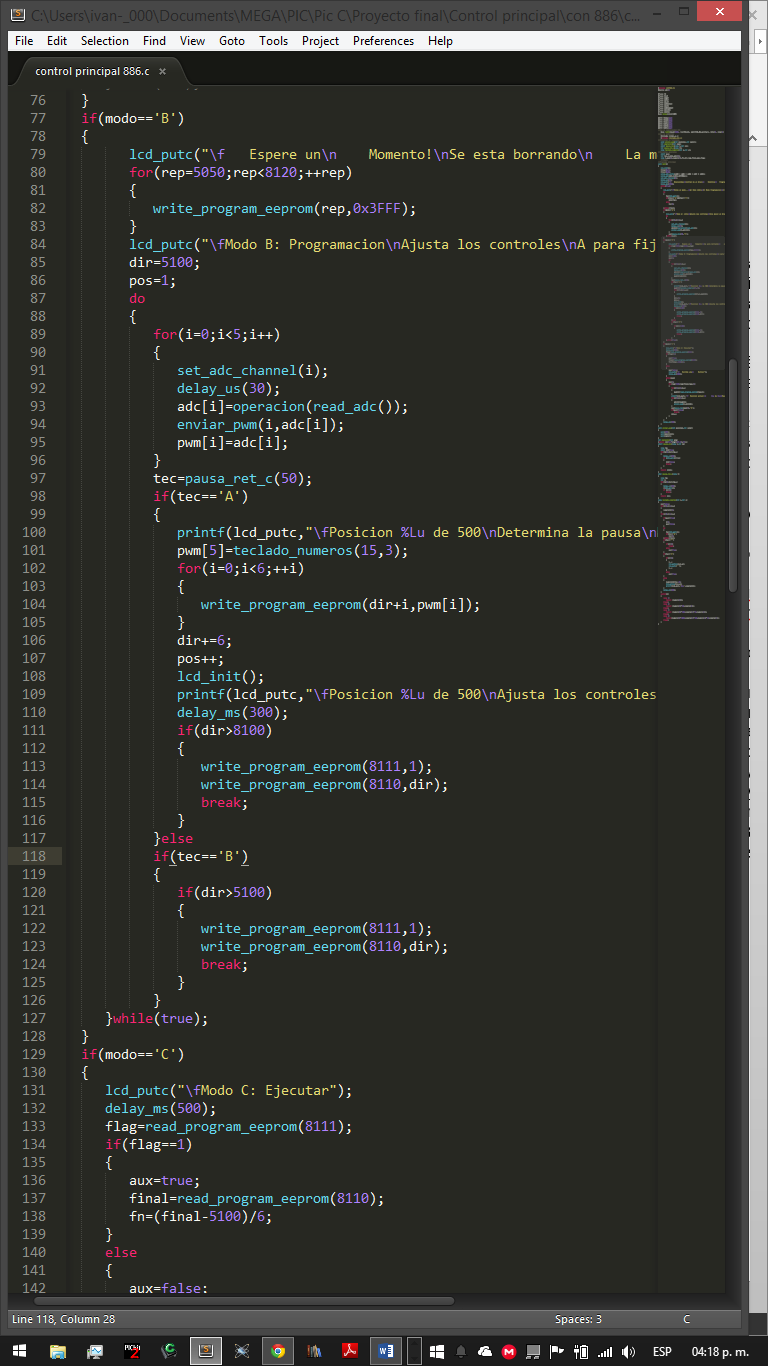


Figura 3.1.9 Modo B

**El modo C “Ejecución”**

Una vez presionada la tecla C se desplegará un mensaje indicativo del modo luego se cargara una variable llamada *flag* de la dirección de la memoria 8111 que es el byte indicativo de grabado, si el byte es 1 se procede a cargar la variable *aux* con un valor verdadero, la variable final con la posición final de grabado y la variable *fn* con la resta de final menos 5100 sobre 6 que nos arroja el total de posiciones grabadas por el usuario, en caso de que el byte no sea 1 se despliega un mensaje que dice que grabe una rutina, si se grabó un dato se accederá a un ciclo *while* con parámetro de entrada la variable *aux* que previamente se cargó con una valor verdadero/ falso, después se carga una variable llamada pos con un 1 pues es la primera posición después se accede a un ciclo *for* que inicia en la primera posición donde se grabó el programa (5100) y tiene como final la posición final de grabado, cada incremento corresponde a 6, es decir, que cada ciclo se aumenta en 6 la variable *rep*, ya dentro del ciclo encontramos otro ciclo *for* que va de 0 a 5 y en cada ciclo lee lo que hay en la memoria gracias al *array* y la posición está dada por la variable *rep* mas la variable i, se despliega un mensaje en la pantalla que indica la posición y la pausa entre posición después se entra en otro ciclo *for* que envía os datos por el puerto serie para que actualice el valor de los servomotores, después hace un retardo con la función *pausa\_tec* que recibe como parámetros de entrada la pausa en milisegundos y el carácter excluyente y arroja como parámetro del tipo booleano que se carga en la variable *aux* y después revisa si la variable *aux* es falsa se sale del ciclo *for* y también se sale del ciclo *while* ingresando así al menú principal. Véase figura 3.2.0

Figura 3.2.0 Modo C

## 3.5 Realización de circuitos en protoboard.

Los circuitos que anteriormente se habían diseñado en el software Proteus ahora se llevaron a la parte física, es decir, se montaron o se armaron en una protoboard para hacer pruebas de su funcionamiento.

Adecuando los componentes al tamaño de protoboard y colocándolos de manera que hubiese espacio para todos para poder manipularlos y que no se estorbaran unos a otros.

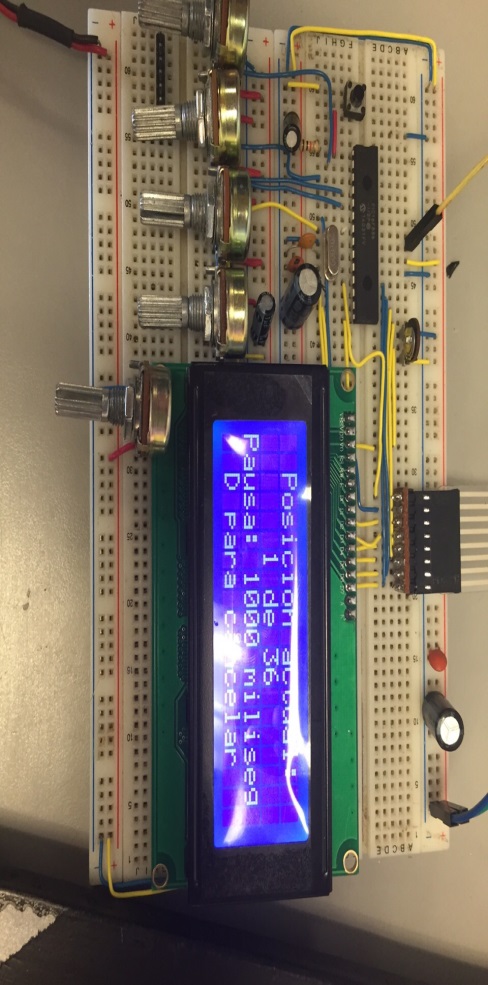


Figura 3.2.1 Circuito en protoboard.

Después de observar cómo funcionaban se fueron adecuando o modificando para que por ultimo funcionara como habíamos pensado, dándonos resultados exitosos para proseguir con lo siguiente.

## 3.6 Esquemas de circuitos electrónicos.

Elaboramos todos los circuitos electrónicos de la parte del control del brazo robótico en el software “Proteus”, basándonos en los requerimientos que solicitábamos del brazo.

Comenzamos con el diseño de un circuito para controlar servomotores, este tenía que cumplir con el control de por lo menos cuatro servomotores, pero que no todos hicieran el mismo movimiento si no que cada uno se pudiera controlar por separado quedando el circuito siguiente: (figura 3.2.2).

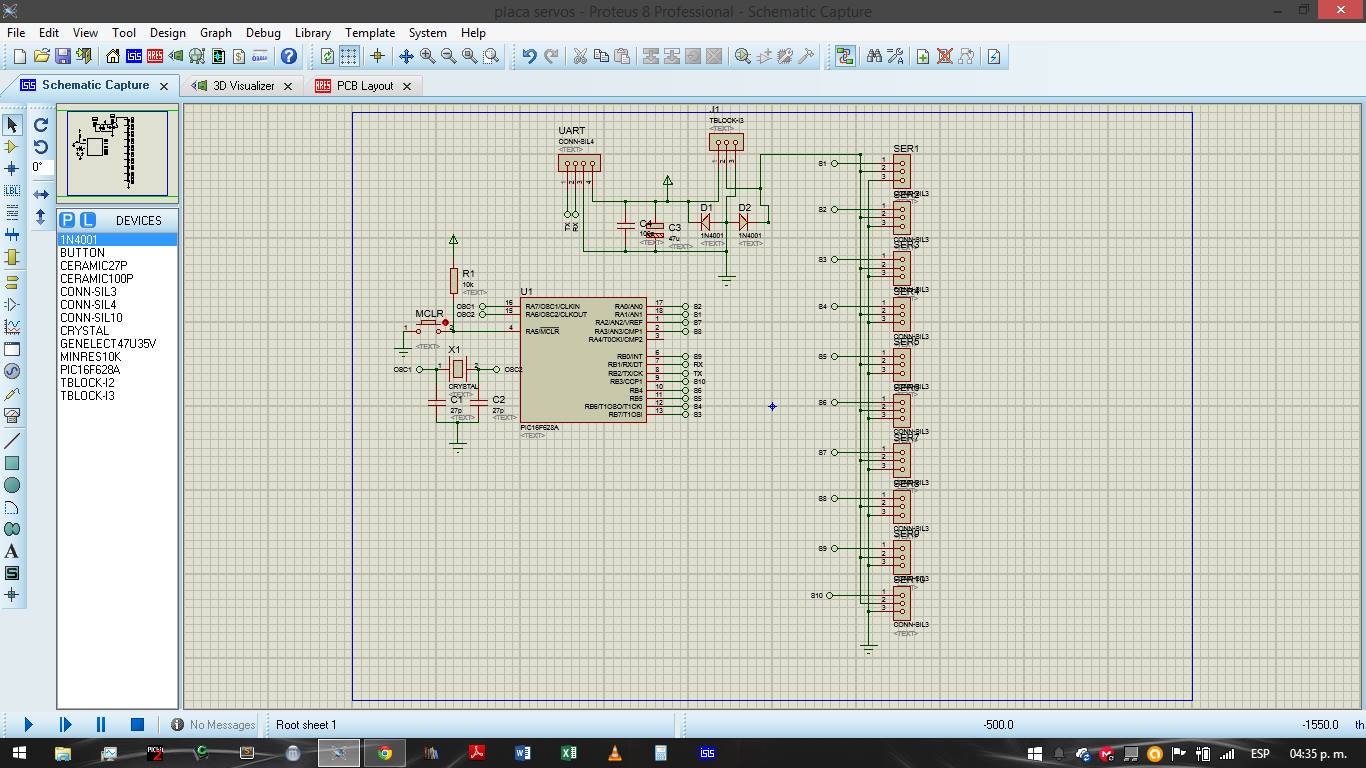


Figura 3.2.2 Control de servos.

También se diseñó la otra parte del control que es un teclado matricial y una pantalla LCD donde se muestran los modos de control del brazo así como los potenciómetros con los cuales se maneja en la forma manual. (Figura 3.2.3).

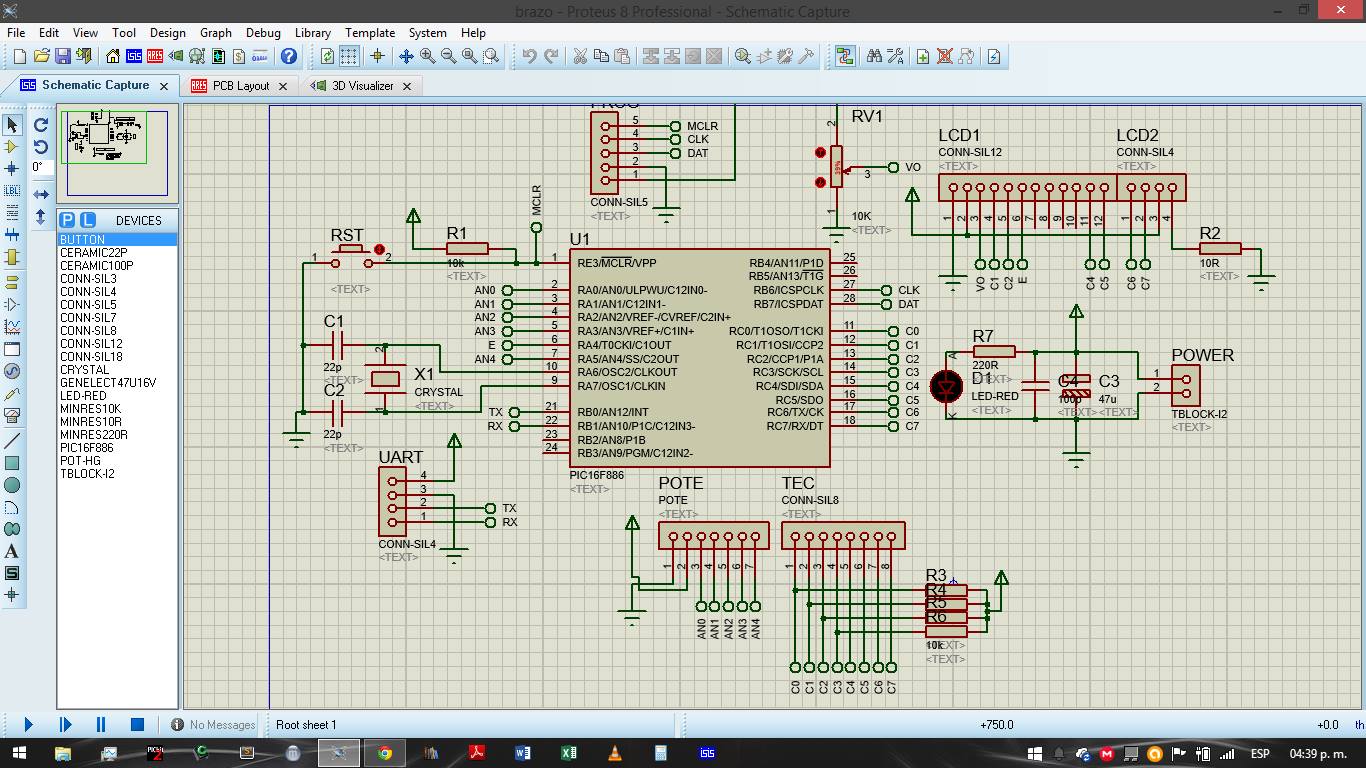


Figura 3.2.3 Sistema de control.

## 3.7 Diseño de placas.

Ya teniendo los resultados esperados en las pruebas procedimos a elaborar el diseño de las placas fenólicas, basándonos en los circuitos que ya teníamos.

De igual manera realizada con el software Proteus, donde nosotros mismos realizamos sus diseños trazándolos pasó por pasos así como también diseñando componentes que el software no contiene. Su elaboración fue buscando el ahorro de espacio en la placa, reduciendo su tamaño lo que más se pudiera para que la placa fuera estéticamente más visible.

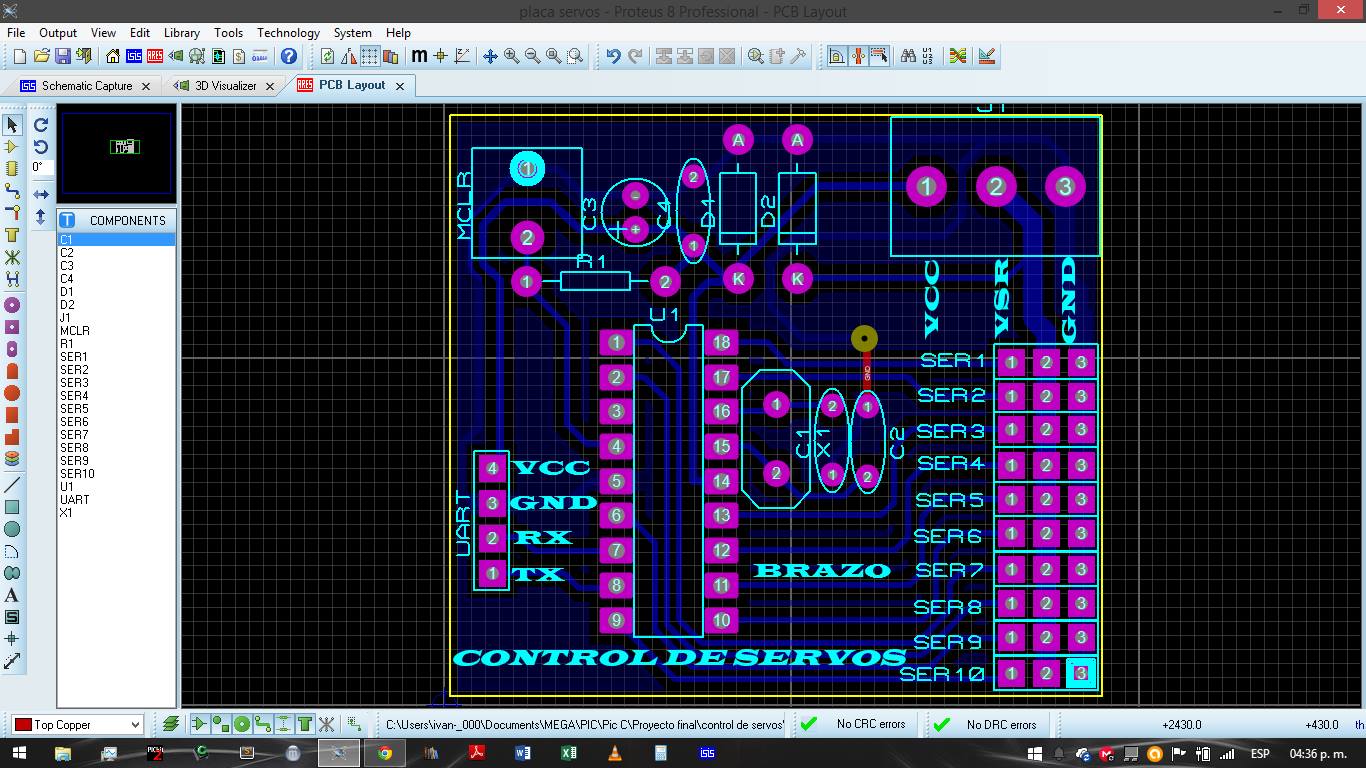


Figura 3.2.4 Control de servos.

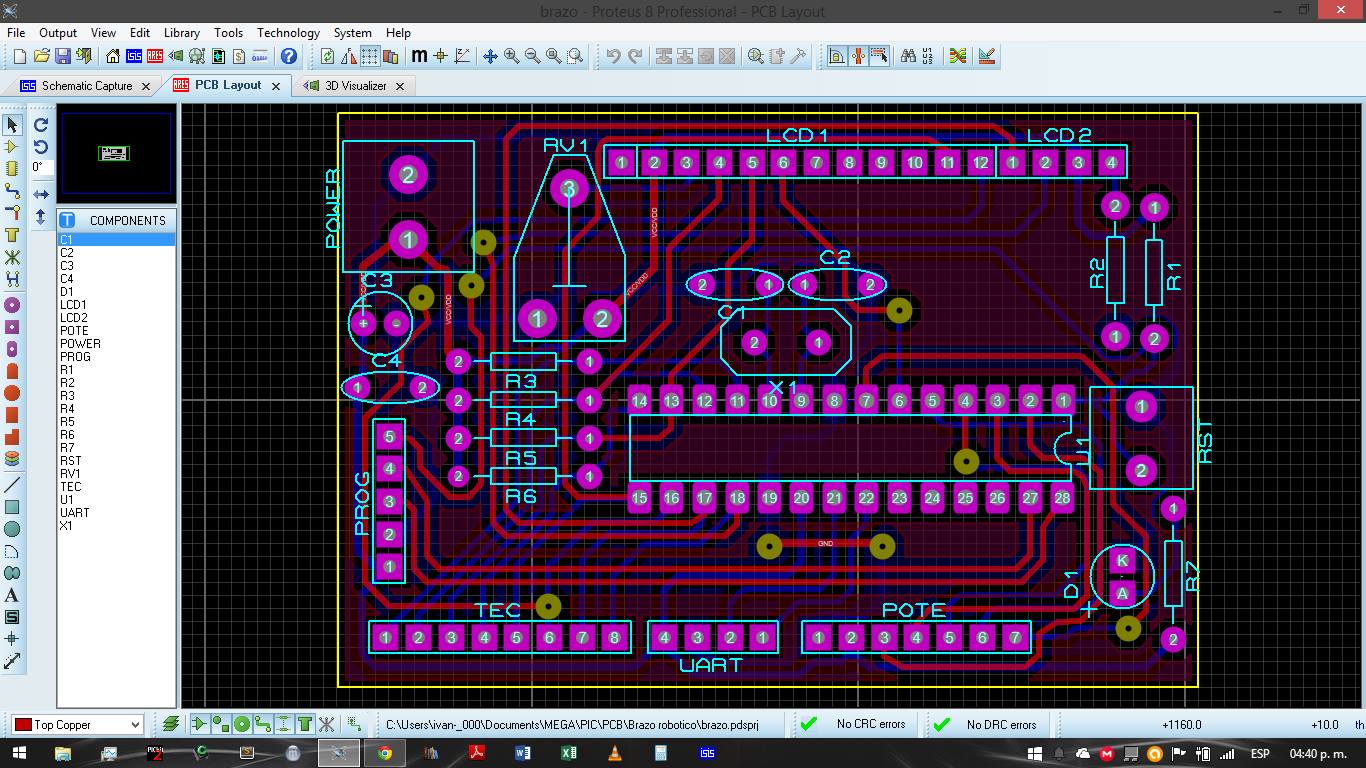


Figura 3.2.5 Control.

## 3.8 Elaboración de placas fenólicas.

Después de tener los diseños se procedió a imprimirlos y desarrollar todo el procedimiento que conlleva esto.

* Planchado.

El diseño de la placa el cual se encuentra impreso en una hoja se recorta a las medidas de la placa, así mismo la placa se ajusta al tamaño del diseño, después se coloca la hoja en sima de la placa y se comienza a planchar, cuidando que la plancha no desprenda vapor ya que mojaría la hoja, se plancha en un tiempo alrededor de 10 o 15 min.

Se mete la placa a un recipiente con agua para que la hoja se moje y se pueda desprender, quedando ya impresa la tinta en la placa.



Figura 3.2.6 Planchado.

* Quemado de placa.

Después de que la placa fenólica allá quedado con el diseño ya grabado se procede a retirar el exceso de cobre, es decir quitar la parte que no está cubierta de tinta.

Esto utilizando cloruro férrico disuelto en agua. En un recipiente se coloca el cloruro férrico y se introduce la placa, se comienza a mover para que el proceso se aún más rápido, esto tarda alrededor de 20 minutos.

Ya quedando desprendida la parte sobrante del cobre se enjuaga con agua para quitar el exceso de cloruro férrico.



Figura 3.2.7 Quemado.

* Perforación de placa.

En este paso le quitamos la tinta a la placa de modo que se vean las pistas de cobre; siguiendo de perforar la placa en cada punto en el cual se necesite hacerlo para poder colocar los componentes.

* Colocación de componentes

Se colocan los componentes en las partes que van según el diseño de la placa, para posteriormente soldarlos a placa.

## 3.9 Realización de maqueta.

Se comenzó a realizar la estructura en la cual se coloca el brazo, tanto la base y la caja donde se encuentra la parte del sistema de control. Primero comenzamos la base de toda la estructura, como se muestra en las siguientes imágenes:





Figura 3.2.8 Armado. Figura 3.2.9 Pintado.

Se prosiguió con la parte de la caja de control del sistema:



Figura 3.3.0 Construcción caseta de control.

## 3.1.0 Pruebas finales.

En este paso colocamos lo que es la base, la caja de control y el brazo en un solo sistema o maqueta. Dándole un orden o mejor dicho una organización para que todo que de forma correcta, es decir, que se pueda maniobrar fácilmente sí que una cosa estorbe a la otra.



Figura 3.3.1 Estructura final.

De esta manera concluimos con la realización de toda la parte mecánica y electrónica del prototipo, solo basta por hacer pruebas para verificar su correcto funcionamiento. Ya teniendo el sistema final simplemente se hacen pruebas del funcionamiento para confirmar que ya no exista ningún error en el proceso de ejecución del brazo.

Si el error hubiese existido simplemente se modifica la parte en la que está mal ya sin cambiar todo e prototipo, esto lo hace más fácil y eficiente.

Como en este caso no hubo errores y todo salió de acuerdo a lo planteado se dio por terminado el prototipo.

# Conclusiones.

Al término de todo el desarrollo del prototipo se logró cumplir con los objetivos planteados de manera exitosa, gracias a que se realizaron las cosas con un cierto orden y además aplicamos nuestros conocimientos en la materia siguiendo un plan estratégico ya planteado. Nos dimos cuenta que nos causó un gran impacto de manera personal ya que nos ayudó para poder desarrollar o exponer los conocimientos que fuimos adquiriendo a lo largo de toda nuestra preparación académica, así como para demostrar que somos capes de plantearnos una objetivo y cumplir con él. Claro que también tuvimos dificultades o tropiezos al realizarlos, como por ejemplo en la programación y sobre todo en desarrollar la parte teórica, aunque si volviéramos a tener la oportunidad de desarrollar de nuevo el prototipo solo cambiaríamos el orden de hacer las cosas llevar un plan más estratégico con tiempo más medidos para poder terminarlo más rápido, pero pues como decíamos logramos el objetivo y estamos complacidos con ello.

# Referencias

Wikipedia (2015 Mayo 21), Brazo robótico, Recuperado de <https://es.wikipedia.org/wiki/Brazo_rob%C3%B3tico>

Manuel (2010 Junio 30), Brazos robóticos en la industria, Recuperado de <http://electronica-jaimes.blogspot.mx/2010/06/descripcion-de-un-brazo-robotico.html>

Víctor R. Gonzáles (2002 Marzo), Servomotores, Recuperado de <http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/sistema/motores_servo.htm>

Cejaruso (2005), Mecanismos, Recuperado de <http://concurso.cnice.mec.es/cnice2006/material107/maquinas/maq_mecanismos.htm>

## Anexos

**Control principal:** en esta parte se da el control de todo el proceso de los modos que tiene el brazo, así como la integración de un teclado y una pantalla LCD.

#include <16f886.h>

#device adc=8

#fuses HS

#fuses MCLR

#fuses NOWDT

#fuses NOPUT

#fuses NOLVP

#fuses NOPROTECT

#fuses NOCPD

#fuses NOBROWNOUT

#fuses NOFCMEN

#fuses NOIESO

#use delay(clock=20M)

#BYTE TRISA=0x85

#BYTE TRISB=0x86

#BYTE TRISC=0x87

#BYTE PORTA=0x05

#BYTE PORTB=0x06

#BYTE PORTC=0x07

//Uso de comunicacion 232

#use rs232(baud=9600, rcv=PIN\_B1, xmit=PIN\_B0,parity=n, bits=8, stop=1)

//Uso de librerias

#include <lcdpc\_a.c>

#include <teclado\_4x4.c>

//Prototipo de funciones

void enviar\_pwm(int posicion,int valor);

int operacion(int ang);

short pausa\_tec(long t,int ce);

int pausa\_ret\_c(long t);

long teclado\_numeros(int x,int y);

//Variables generales

short aux;

int tec,i,modo,adc[5];

long t,pwm[6],numero[5],fn,dir,rep,final,pos,flag;

//float;

//Programa principal

void main()

{

lcd\_init();

trisa=0xFF;

trisc=0x00;

setup\_adc\_ports(sAN0 | sAN1 | sAN2 | sAN3 | sAN4);

setup\_adc(ADC\_CLOCK\_INTERNAL);

set\_adc\_channel(0);

delay\_us(15);

lcd\_putc("\f Bienvenidos\nControl de un brazo\n Robotico\n Programable");

delay\_ms(1500);

while(true)

{

lcd\_putc("\fElije el modo...\nA= Modo libre\nB= Modo Programacion\nC= Modo Ejecucion");

do

{

tec=kbd\_getc();

if((tec>='A')&&(tec!='D'))

modo=tec;

else

tec=0;

}while(!tec);

if(modo=='A')

{

lcd\_putc("\fModo A: Libre\nAjusta los controles\nPara mover el brazo\n D para cancelar");

do

{

for(i=0;i<5;i++)

{

set\_adc\_channel(i);

delay\_us(15);

adc[i]=operacion(read\_adc());

enviar\_pwm(i,adc[i]);

}

aux=pausa\_tec(50,'D');

}while(aux);

}

if(modo=='B')

{

lcd\_putc("\f Espere un\n Momento!\nSe esta borrando\n La memoria...");

for(rep=5100;rep<8120;++rep)

{

write\_program\_eeprom(rep,0x3FFF);

}

lcd\_putc("\fModo B: Programacion\nAjusta los controles\nA para fijar\nB para finalizar");

dir=5100;

pos=1;

do

{

for(i=0;i<5;i++)

{

set\_adc\_channel(i);

delay\_us(30);

adc[i]=operacion(read\_adc());

enviar\_pwm(i,adc[i]);

pwm[i]=adc[i];

}

tec=pausa\_ret\_c(50);

if(tec=='A')

{

printf(lcd\_putc,"\fPosicion %Lu de 500\nDetermina la pausa\nMilisegundos=\n \* aceptar # borrar",pos);

pwm[5]=teclado\_numeros(15,3);

for(i=0;i<6;++i)

{

write\_program\_eeprom(dir+i,pwm[i]);

}

dir+=6;

pos++;

lcd\_init();

printf(lcd\_putc,"\fPosicion %Lu de 500\nAjusta los controles\nA para fijar\nB para finalizar",pos);

delay\_ms(300);

if(dir>8100)

{

write\_program\_eeprom(8111,1);

write\_program\_eeprom(8110,dir);

break;

}

}else

if(tec=='B')

{

if(dir>5100)

{

write\_program\_eeprom(8111,1);

write\_program\_eeprom(8110,dir);

break;

}

}

}while(true);

}

if(modo=='C')

{

lcd\_putc("\fModo C: Ejecutar");

delay\_ms(500);

flag=read\_program\_eeprom(8111);

if(flag==1)

{

aux=true;

final=read\_program\_eeprom(8110);

fn=(final-5100)/6;

}

else

{

aux=false;

lcd\_putc("\f Escriba una\n Rutina!");

delay\_ms(1000);

}

while(aux)

{

pos=1;

for(rep=5100;rep<final;rep+=6)

{

for(i=0;i<6;i++)

{

pwm[i]=read\_program\_eeprom(rep+i);

}

printf(lcd\_putc,"\f Posicion actual:\n %Lu de %Lu\nPausa: %Lu miliseg\n D para cancelar",pos,fn,pwm[5]);

for(i=0;i<5;i++)

{

adc[i]=pwm[i];

enviar\_pwm(i,adc[i]);

}

aux=pausa\_tec(pwm[5],'D');

if(aux==false)

break;

pos++;

}

}

}

Delay\_ms(500);

}

}

void enviar\_pwm(int posicion,int valor)

{

putc(255);

putc(posicion);

putc(valor);

}

int operacion(int ang)

{return (((76.0\*ang)/255.0)+33);}

short pausa\_tec(long t,int ce)

{

long tp;

short ind=true;

for(tp=0;tp<t;tp++)

{

delay\_us(898);

if (kbd\_getc()==ce)

{

ind=false;

break;

}

}

return (ind);

}

int pausa\_ret\_c(long t)

{

long tp;

int z;

for(tp=0;tp<t;tp++)

{

delay\_us(898);

z=kbd\_getc();

if (z!=0)

break;

}

return (z);

}

long teclado\_numeros(int x,int y)

{

aux=false;

for(i=0;i<4;i++)

{

numero[i]=0;

}

for(i=0;i<4;i++)

{

if(aux==true)

{

i=0;

aux=false;

}

do

{

tec=kbd\_getc();

if(tec>='A')

tec=0;

}while(!tec);

if(tec=='\*')

{

if(i!=0)

break;

else

aux=true;

}

if(tec=='#')

{

if(i!=0)

{

i--;

lcd\_gotoxy(x+i,y);

lcd\_putc(' ');

i--;

}

else

aux=true;

}

else

{

numero[i]=tec-48;

lcd\_gotoxy(i+x,y);

printf(lcd\_putc,"%Lu",numero[i]);

}

delay\_ms(400);

}

switch(i)

{

case 1:

return (numero[0]);

break;

case 2:

return (numero[0]\*10+numero[1]);

break;

case 3:

return (numero[0]\*100+numero[1]\*10+numero[2]);

break;

case 4:

return(numero[0]\*1000+numero[1]\*100+numero[2]\*10+numero[3]);

break;

}

}

**Control de servos:** es donde se conjuga el programa para el uso de varios servomotores a un mismo tiempo mediante comunicación serial.

#include <16f628a.h>

#fuses HS

#fuses MCLR

#fuses NOWDT

#fuses NOPUT

#fuses NOLVP

#fuses NOPROTECT

#fuses NOCPD

#fuses NOBROWNOUT

#use delay(clock=20M)

#BYTE TRISA=0x85

#BYTE PORTA=0x05

#BYTE TRISB=0x86

#BYTE PORTB=0x06

#bit Bit\_PWM0 = PORTA.1

#bit Bit\_PWM1 = PORTA.0

#bit Bit\_PWM2 = PORTB.7

#bit Bit\_PWM3 = PORTB.6

#bit Bit\_PWM4 = PORTB.5

#bit Bit\_PWM5 = PORTB.4

#bit Bit\_PWM6 = PORTA.2

#bit Bit\_PWM7 = PORTA.3

#bit Bit\_PWM8 = PORTB.0

#bit Bit\_PWM9 = PORTB.3

#use rs232(baud=9600, rcv=PIN\_B1, xmit=PIN\_B2,parity=n, bits=8, stop=1)

//Prototipos de funciones

void ancho\_pulso(void);

void generacion\_pwm(void);

void datos\_recividos(void);

//Variables generales

short b\_pwm[10];

int i,dato,pwm[10],ofset,rcv=0,srv=10;

long control\_PWM=0;

//interrupciones

#int\_Timer0

void generacion\_pwm()

{

/\*for(i=0;i<10;i++)

{

if(control\_PWM==pwm[i])

srv=i;

}

switch(srv)

{

case 0:

Bit\_PWM0=0;

break;

case 1:

Bit\_PWM1=0;

break;

case 2:

Bit\_PWM2=0;

break;

case 3:

Bit\_PWM3=0;

break;

case 4:

Bit\_PWM4=0;

break;

case 5:

Bit\_PWM5=0;

break;

case 6:

Bit\_PWM6=0;

break;

case 7:

Bit\_PWM7=0;

break;

case 8:

Bit\_PWM8=0;

break;

case 9:

Bit\_PWM9=0;

break;

}

srv=10;\*/

if (control\_PWM==pwm[0]) Bit\_PWM0=0;

if (control\_PWM==pwm[1]) Bit\_PWM1=0;

if (control\_PWM==pwm[2]) Bit\_PWM2=0;

if (control\_PWM==pwm[3]) Bit\_PWM3=0;

if (control\_PWM==pwm[4]) Bit\_PWM4=0;

if (control\_PWM==pwm[5]) Bit\_PWM5=0;

if (control\_PWM==pwm[6]) Bit\_PWM6=0;

if (control\_PWM==pwm[7]) Bit\_PWM7=0;

if (control\_PWM==pwm[8]) Bit\_PWM8=0;

if (control\_PWM==pwm[9]) Bit\_PWM9=0;

control\_PWM++;

set\_timer0(251);

}

#int\_Timer1

void ancho\_pulso()

{

control\_PWM=0;

Porta=255;

portb=255;

set\_timer1(15536);

set\_timer0(251);

enable\_interrupts(int\_timer0);

}

#int\_rda

void datos\_recividos()

{

disable\_interrupts(int\_timer0);

dato=getc();

switch(rcv)

{

case 0:

if(dato==255)

rcv=1;

break;

case 1:

ofset=dato;

rcv=2;

break;

case 2:

pwm[ofset]=dato;

rcv=0;

break;

}

enable\_interrupts(int\_timer0);

}

//Programa principla

void main()

{

trisa=0;

trisb=0b00000010;

setup\_timer\_0(T0\_INTERNAL | T0\_DIV\_1);

setup\_timer\_1(T1\_INTERNAL | T1\_DIV\_BY\_2);

set\_timer0(251);

set\_timer1(15536);

b\_pwm[0]=Bi;

pwm[0]=1;

pwm[1]=2;

pwm[2]=3;

pwm[3]=4;

pwm[4]=5;

pwm[5]=6;

pwm[6]=7;

pwm[7]=8;

pwm[8]=9;

pwm[9]=10;

enable\_interrupts(INT\_TIMER0);

enable\_interrupts(INT\_TIMER1);

enable\_interrupts(INT\_RDA);

enable\_interrupts(GLOBAL);

while(true);

}