

| | |
|----------------|----------------|
| Título: | Brazo Robótico |
|----------------|----------------|

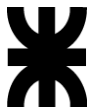
| | | | | |
|---------------------------|----------|-------|----------|---|
| Ciclo Lectivo 2017 | Curso N° | R2001 | Grupo N° | 7 |
|---------------------------|----------|-------|----------|---|

| Integrantes | | Calificación individual | Fecha |
|-----------------------------|----------|-------------------------|-------|
| Apellido Nombres | Legajo | | |
| Barreras Fauvety, Guillermo | 159675-5 | | |
| Mendez, Cristian | 159559-3 | | |
| Lopez, Luciano Alberto | 159357-2 | | |
| | | | |

| | |
|----------------------|-------------------|
| Calificación grupal: | Fecha: 02/12/2017 |
|----------------------|-------------------|

| | |
|----------------------|-------------------|
| Profesor: | Marcelo Giura |
| Auxiliar/es Docente: | Gabriel Soccodato |

| | |
|-------------------------------|--|
| Observaciones primera entrega | |
| Observaciones segunda entrega | |



INDICE

| | |
|---|----|
| INDICE | 2 |
| Desarrollo de la idea fuerza..... | 3 |
| Introducción..... | 4 |
| Objetivos principales..... | 4 |
| Descripción detallada (diagrama en bloques)..... | 5 |
| Descripción del hardware utilizado..... | 9 |
| El hardware utilizado:..... | 9 |
| Link a hojas de datos:..... | 9 |
| Esquemático del hardware externo..... | 10 |
| Funcionamiento de los servos:..... | 11 |
| Máquina de estados de la aplicación..... | 12 |
| Trama de la comunicación serie | 17 |
| Problemas encontrados a lo largo del desarrollo del TP0..... | 18 |
| Beneficios encontrados a lo largo del desarrollo del TP0..... | 21 |
| Conclusiones..... | 21 |
| Bibliografía, links, etc. | 22 |



Desarrollo de la idea fuerza

El proyecto consiste en un brazo robótico, el cual podrá ser controlado por un usuario. La libertad de movimiento se verá comprendida por los siguientes tres: rotación sobre su eje, altura y agarre de la pinza.

El brazo se encontrará sobre una base de madera, la cual actuará de contrapeso para el mismo. Su cuerpo estará principalmente conformado por cartón ya que es un material resistente, liviano y barato. La necesidad de que no sea pesado radica en que para moverlo poseerá tres jeringas (una por cada una de las articulaciones), conectadas por mangueras a otras tres jeringas. Éste último trío será el cual, al regularlo, gracias a las propiedades de la presión hidráulica, moverá los émbolos de las jeringas ubicadas estratégicamente en las articulaciones y cuanto menor peso tengan las piezas, menor será la fuerza que se le deberá aplicar.

El trío nombrado de jeringas será regulado por motores paso a paso, los cuales estarán conectados al KIT Info-Tronic a través de interfaces para poder utilizar los puertos de salida digital en vez de los relays.

En cuanto al software, se realizará un programa que permita utilizar los pulsadores del KIT para controlar los movimientos del brazo. Una vez conseguido esto se comenzará a crear una interfaz gráfica para que el brazo pueda ser controlado por computadora por el usuario.

De haber tiempo, se trabajará la idea de poder implementar el agregado de un joystick para que el modo de controlar el robot sea más didáctico.



Introducción

El proyecto surge a partir de la necesidad de manipular objetos indirectamente, pero a la vez con precisión humana, que se puede tener en ciertos ámbitos. Por ejemplo, dentro de un laboratorio en el que se trabaja con sustancias altamente nocivas para la salud.

Para cumplir con esta necesidad, se decidió construir un brazo robótico que pudiese ser controlado por el usuario tanto por un teclado matricial como a través de una interfaz gráfica desde una computadora.

Objetivos principales

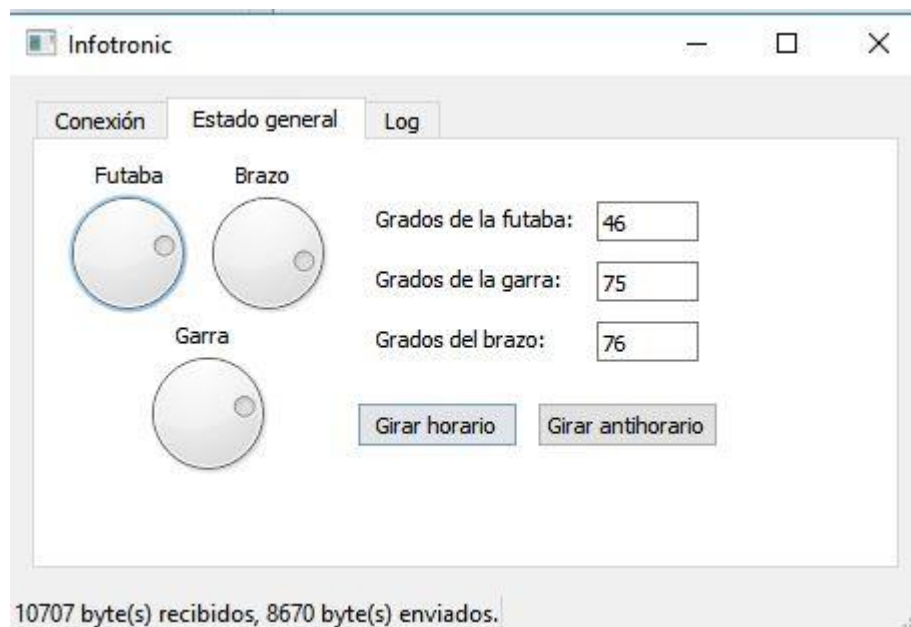
- + Aprender a utilizar los periféricos del kit
- + Manejar el brazo a través de la PC
- + Lograr una buena conexión entre los módulos de manera tal de lograr la mayor eficiencia
- + Aprender a realizar una interfaz gráfica sencilla y de fácil uso para el usuario
- + Aprender a realizar una maqueta precisa según las especificaciones necesarias para el brazo.
- + Lograr la conexión electro-mecánica para el brazo.

Descripción detallada (diagrama en bloques)



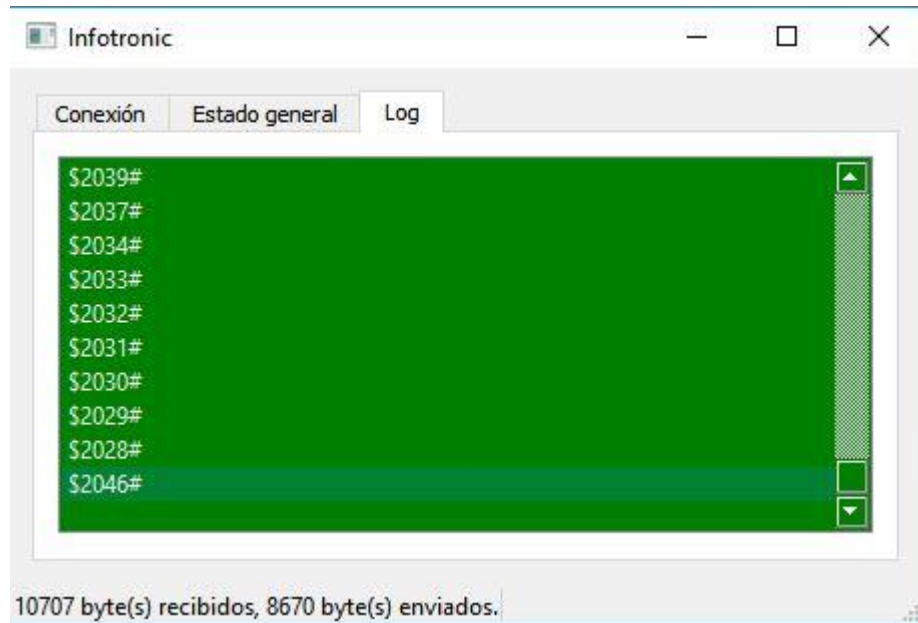
PC: El método principal mediante el cual el usuario podrá controlar el brazo será una interfaz gráfica creada con el Qt Creator, la cual se encontrará en la PC. Esta se comunicará con el kit a través de su puerto serie. En este se indicará la posición en grados y tendrá controles tanto para mover las articulaciones del brazo, como su rotación.

A continuación, se es detallada la aplicación grafica utilizada en el proyecto. Al abrir la aplicación, elegir el puerto serie el cual pertenece a el proyecto, se observa la ventana de **Estado General**, esta es la encargada de contener todos los comandos para que el brazo pueda ser controlada desde la PC y sus movimientos se vean reflejados en el brazo.



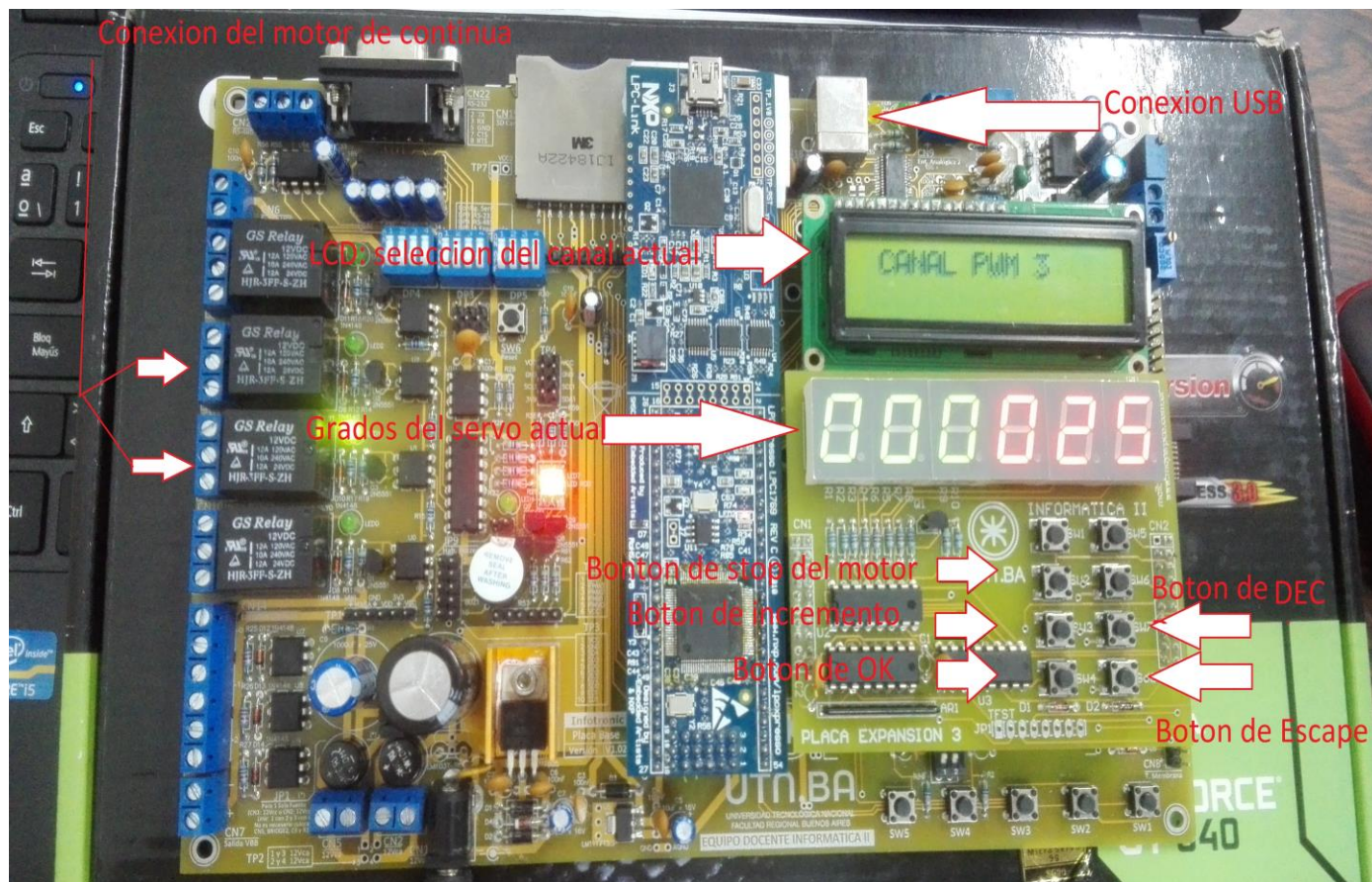
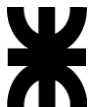


Por último, la aplicación grafica posee una consola, de **LOG** que permite ver que se está mandando.



KIT Info-Tronic: El kit contendrá el microcontrolador que procesará toda la información recibida de la PC y la decodificará para actuar en consecuencia a través de los periféricos, tales como: servos, LCD y displays. Se podrá también, a través del kit, elegir qué articulación mover, y con el teclado matricial realizar los movimientos, mientras en los displays se muestra los grados de giro.

A continuación, es detallado con una imagen del kit, todo el hardware utilizado en él.



- **Botón Incremento:** Este botón es utilizado para incrementar el valor en grados del servo o cambiar de canal, dependiendo del modo en el que este.
- **Botón DEC:** Este botón es utilizado para decrementar el valor en grados del servo o cambiar de canal, dependiendo del modo en el que este.
- **Botón OK:** Este botón es utilizado para entrar a modificar el valor de los grados.
- **Botón ESCAPE:** Este botón es utilizado para salir del modo en el que el valor del servo es modificado.
- **Botón STOP:** Cuando el motor es utilizado. Este botón es utilizado para pararlo.



Brazo: El brazo contendrá 3 servomotores que controlarán los movimientos de cada una de sus articulaciones y un motor de continua que se encargará de su movimiento rotacional. Todos estos estarán conectados al kit, el cual se encargará de suministrarles la alimentación necesaria para lograr movimientos precisos en los momentos indicados.

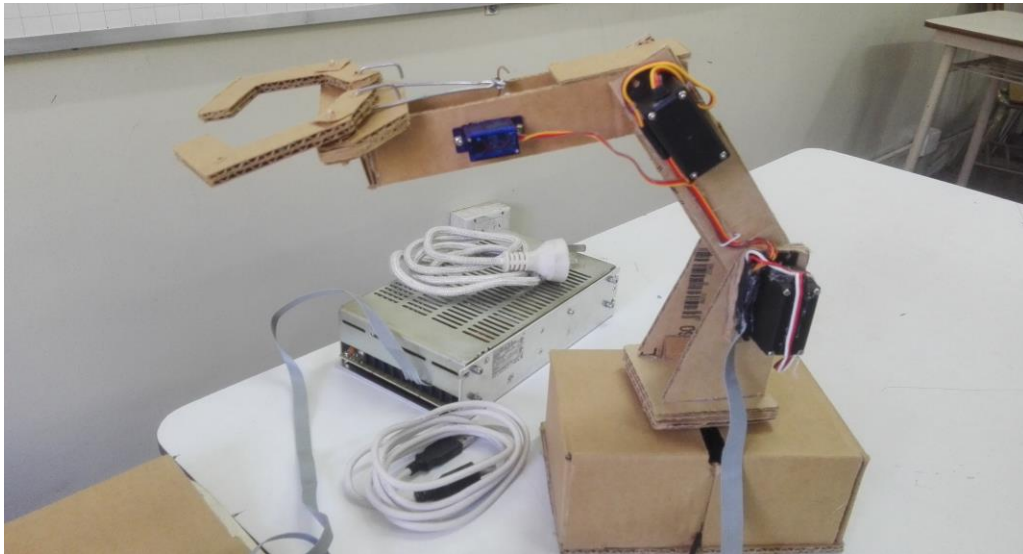


Imagen del prototipo final al momento de la presentación.



Imagen del prototipo completo final al momento de la presentación.



Descripción del hardware utilizado

El hardware utilizado:

1. Motor de continua: Controla tanto el giro horario y antihorario, el control se realiza a través de dos relays, que le brinda energía tanto en un sentido como otro, logrando así el giro.
2. Servo sg90: Se encarga del movimiento de la mano. El servomotor funciona como un PWM, de 20ms y con un tiempo en alto que varía de 2ms a 18ms, logrando que el servo gire 180 o 0 grados.
3. Servo Futaba s3003: Se encarga del movimiento del brazo. El funcionamiento de este servo motor es idéntico al anterior.
4. Servo sg92r: Se encarga del movimiento del antebrazo. El funcionamiento de este servo motor es idéntico al anterior.

Nota: El funcionamiento de los servos se verá con una mayor explicación mas adelante.

Link a hojas de datos:

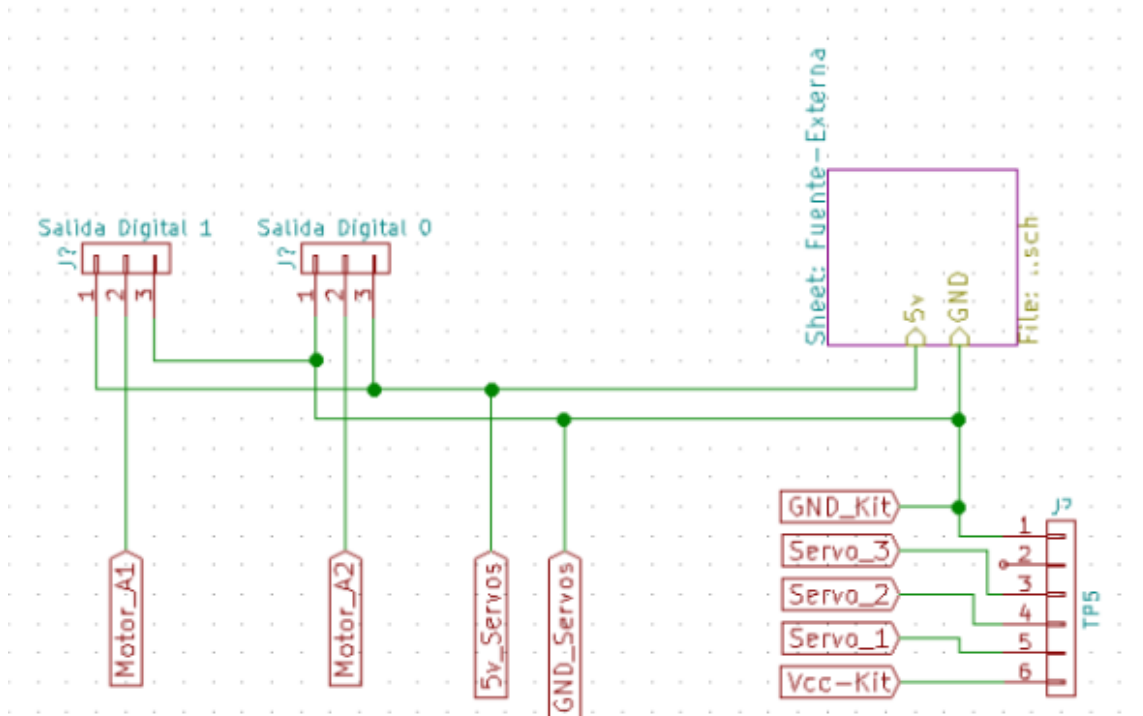
<http://akizukidenshi.com/download/ds/towerpro/SG90.pdf>

<http://www.towerpro.com.tw/product/sg92r-7>

<http://www.es.co.th/schemetic/pdf/et-servo-s3003.pdf>



Esquemático del hardware externo



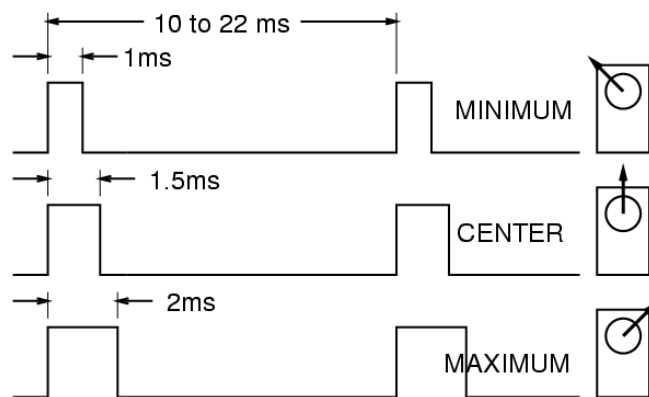
Se utilizó muy poco hardware externo, para controlar el motor se optó por cablear un puente H realizado con relays, y para enviar la señal a los servos se utilizó el conector TP5 ubicado en el KIT Info-Tronic. Por último, una fuente de múltiples salidas da alimentación tanto a los servos como al kit, haciendo provecho de sus múltiples tensiones de salida.

En cuanto al hardware provisto por el kit, se utilizó el puerto USB (que es reconocido como un puerto COM virtual desde el lado de la PC y desde el lado del micro se pueden mandar datos mediante la UART), el LCD para informar el canal que se estaba modificando, cuando se utiliza el robot mediante el teclado, displays de 7 segmentos para mostrar el valor actual de los servos en grados y un teclado matricial para operar el equipo. Tanto los displays de 7 segmentos como el teclado matricial pertenecen a la expansión 2 del kit.



Funcionamiento de los servos:

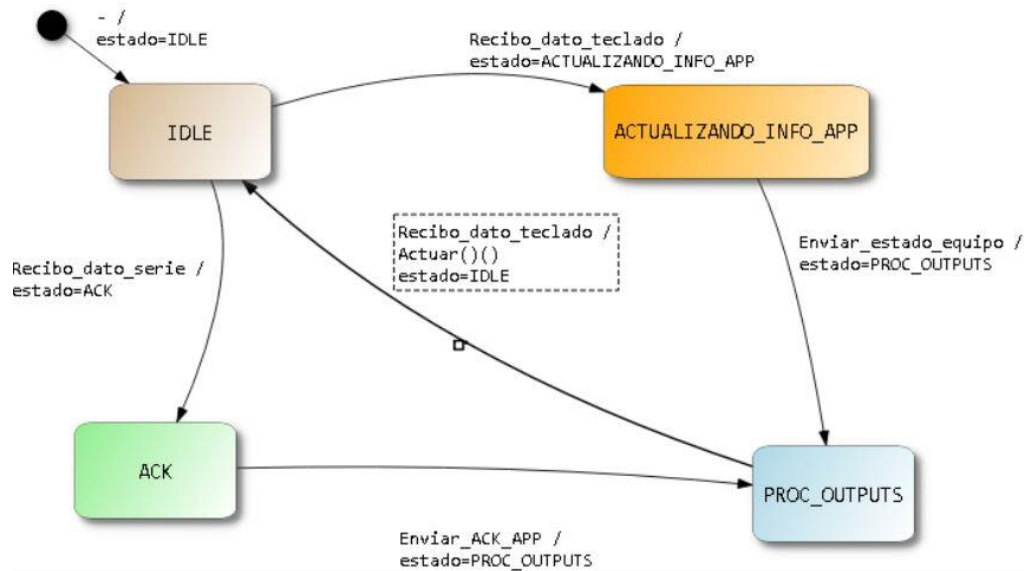
El funcionamiento de los servos, se logra creando una señal con un periodo de 20ms equivalente a 50Hz. Para variar la posición del cursor lo que se tiene que hacer es variar el tiempo en alto de la señal. Como se ve en la figura, el tiempo oscila entre 1ms a 2ms. El recorrido de los servos es de 180° recorrido más que suficiente para cubrir las necesidades del proyecto.



La necesidad de generar una señal con DUTY variable puede ser resuelta mediante dos formas. La primera por software, la ventaja de esto es que nos abstraemos del hardware, pero la contra es que el programa se hace más denso y complicado de seguir. La segunda es utilizar el módulo PWM del microcontrolador la ventaja es que las funciones de control se acortarían mucho, pero la desventaja es que hay que entender cómo hacer funcionar hardware del microcontrolador. Finalmente se optó por utilizar la opción de Hardware.



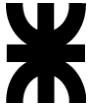
Máquina de estados de la aplicación



La aplicación comienza en el estado **IDLE**, y saldrá de este estado en cuanto reciba un mensaje del puerto serie (el cual se analiza carácter por carácter para asegurar la correcta llegada del mensaje y la estructura) o se utilice el teclado.

En el caso de que la orden sea enviada por el puerto serie, luego de llegada la acción, se pasa al estado de **ACK**, donde se envía por puerto serie un mensaje para hacerle saber a la PC que la información llegó correctamente, y se actualiza en los displays los grados en los que el servo debe moverse. Por último, se pasará al estado **PROC_OUTPUTS**, donde se actualizarán las salidas.

Si la orden llega por el teclado matricial, la aplicación inmediatamente informará de su estado a la interfaz gráfica, para ello se pasa al estado **ACTUALIZANDO_APP_IFNO**, y en último lugar se pasará a estado **PROC_OUTPUTS**, donde se actualizará el valor de las salidas.



La siguiente imagen ayudara a comprender mejor que es lo que ocurre en nuestra aplicación.

```
void Aplicacion(void)
{
    uint8_t static estado = idle;
    switch(estado){
        case idle:
            if(Recive())                // Si recibo algo por la UART...
                estado = ack;
            if(Ent_teclado())           //Si quiero modificar el valor de un servo...
                estado = actualizando_info_app;
            break;
        case ack:
            EnviarString("$OK#");       //Como me mandaron un dato por el puertos serie le respondo con un ACK
            Display(_servo.grados);    // Si la app no recibe esto, me deberia volver a mandar los datos..
            estado = proc_outputs;     //Muestro en el display de 7s el valor en grados del servo actual que estoy tocando
            break;
        case actualizando_info_app:
            Enviar(_servo);             // Aca le tengo que pasar los datos del canal y los grados a la app
            estado = proc_outputs;     //Le digo a la app de la pc que esta pasando con el brazo
            break;
        case proc_outputs:
            Actuar();                  //Modifico el valor de las salidas
            estado = idle;
            break;
        default:
            estado = idle;
    }
}
```

Tanto actuar como set grados merecen ser anexadas al informe debido a que son funciones fuera de lo comun de lo que se uso en la catedra. La funcion **Actuar()**, se encarga de accionar las salidas, si el canal seleccionado es distinto a el canal que tiene anexado el motor de continua entonces **Actuar()**, llama a **Setgrados()**, esta se encarga de hacer la conversion de grados a valor de cuenta del registro. Para lograr esto se obtuvieron los valores de tiempo de alto que requeria el servo para que lograra una posicion de 0 grados, lo mismo para lograr que la posicion del servo fuera 180°, luego mediante una regla de tres simple podemos obtener todos los valores intermedios. Por ultimo ese valor se pasa a buffers intermedios quien se encargan de actualizar los registros correspondientes.



```
void Actuar(void)
{
    if(_servo.canal != 1)
        Setgrados(_servo.canal,_servo.grados);
    else
    {
        if(_servo.grados == 0)
        {
            GIRO_HORARIO
        }

        else if (_servo.grados == 1)
        {
            GIRO_ANTIHORARIO
        }
        else
        {
            APAGAR_MOTORES
        }
    }
}

void Setgrados(uint8_t canal, uint8_t grados)
{
    volatile uint32_t grados_canal = 0;

    if(grados<=180) // Si grados es menor a 180 entramos y modificamos el valor correspondiente a cada canal de PWM
    {
        //grados_canal=(grados*BufferPeriodo)/100;

        grados_canal=conversion(grados);
        switch(canal)
        {
            case AZUL:
                buffer_azul = grados_canal;
                break;
            case ROJO:
                buffer_rojo = grados_canal;
                break;
            case VERDE:
                buffer_verde = grados_canal;
                break;
            default:
                break;
        }
    }
}

uint32_t conversion(uint8_t grados)
//Esta funcion convierte un valor de grados al debido valor correspondiente del REGISTRO de PWM
{
    return (975*grados+POSICION_MIN);
}
```




DriverPWM1(), se encarga de tomar los valores de los buffers intermedios de cada canal y actualizar los valores de PWM. El registro **PWM1MRx** se encarga de almacenar el valor de tiempo en alto de la señal mientras que el registro **PWM1LER** cuando mediante una mascara de bits inserto un '1' dependiendo la posicion del canal este actualiza la salida. Esta funcion esta colocada en el SysTick que es llamado cada cierto tiempo 10ms.

```
9  #include "Aplicacion.h"
10
11 extern volatile uint32_t buffer_azul;
12 extern volatile uint32_t buffer_rojo;
13 extern volatile uint32_t buffer_verde;
14 extern volatile uint32_t BufferPeriodo;
15
16 void DriverPWM1(void)
17 {
18     PWM1MR2=buffer_azul;
19     PWM1MR3=buffer_rojo;
20     PWM1MR4=buffer_verde;
21
22     PWM1LER |= 0x01<<0;    //Actualizo el valor de duty
23     PWM1LER |= 0x01<<1;
24     PWM1LER |= 0x01<<2;
25     PWM1LER |= 0x01<<3;
26     PWM1LER |= 0x01<<4;
27 }
28
```



Todo esto no sería posible sin antes inicializar el módulo de PWM del LPC1769, a continuación observamos como es la rutina de inicialización de dicho módulo

```
void Inicializar_PWM1( void )
{
    /*Selecciono los pines a usar con el pwm*/
    PCLKSEL0 &=~(0x02<<12);
    PCLKSEL0 |= (0x01<<12);
    SetPINSEL ( RGBR , FUNC_PWM);
    SetPINSEL ( RGBG , FUNC_PWM);
    SetPINSEL ( RGBB , FUNC_PWM);
    PWM1TCR |= 0x01<<0;    // Counter Enable
    PWM1TCR |= (0x01<<1);  // Clear Reset
    PWM1TCR |= 0x01<<3;    // PWM Enable
    PWM1PR = 0x0;
    PWM1MCR = 0x02;
    PWM1MR0 = 2000000;      //Periodo de 20ms
    PWM1MR1 = 400000;      //Duty de
    PWM1MR2 = 0/*250000*/;  //Duty de LED AZUL /*LA PLACA DICE PWM2*/
    PWM1MR3 = 0/*125000*/;  //Duty de LED ROJO /*LA PLACA DICE PWM1*/
    PWM1MR4 = 0;           //Duty de LED VERDE /*LA PLACA DICE PWM0*/
    PWM1LER |= 0x01<<0;    //Actualizo el valor de duty
    PWM1LER |= 0x01<<1;
    PWM1LER |= 0x01<<2;
    PWM1LER |= 0x01<<3;
    PWM1LER |= 0x01<<4;
    PWM1PCR &= ~(0x01<<2); //Salida 2 Single edge
    PWM1PCR &= ~(0x01<<3); //Salida 3 Single edge
    PWM1PCR &= ~(0x01<<4); //Salida 4 Single edge
    PWM1PCR |= 0x01<<9;    //Habilito Salida 1
    PWM1PCR |= 0x01<<10;   //Habilito Salida 2
    PWM1PCR |= 0x01<<11;   //Habilito Salida 3
    PWM1PCR |= 0x01<<12;   //Habilito Salida 4
    PWM1TCR &= ~(0x01<<1);
    PWM1TC = 0;
}
```



Trama de la comunicación serie

Para el proyecto se adoptó una trama de comunicación para que tanto la aplicación grafica como el programa realizado en el LPC1769 se pudieran comunicar.



El valor de grados originalmente puede obtener el valor de 0° -180° pero una vez finalizado el prototipo se pusieron limitaciones por software para evitar daños mecánicos.

A continuación, se puede observar la función **Verificarlimites()**, encargada de que los limites establecidos no sean superados. Estos limites fueron obtenidos de manera empírica.

```
uint8_t Verificarlimites(struct sServo serv, uint8_t grados, uint8_t incodec)
// Esta funcion es util para poner limites fisicos al robot mediante el soft, para que no ocurran accidentes
{
    switch(serv.canal)
    {
        case 2:
            if(incodec)
            {
                if((grados >= 0 && grados < 60))
                    return 1;
                else
                    return 0;
            }
            else
            {
                if((grados >= 0 && grados <= 61))
                    return 1;
                else
                    return 0;
            }
        break;
    }
}
```

```
case 3:
    if(incodec)
    {
        if((grados >= 25 && grados < 90))
            return 1;
        else
            return 0;
    }
    else
    {
        if((grados >= 26 && grados <= 91))
            return 1;
        else
            return 0;
    }
    break;
}
```



```
case 4:
    if(incdec)
    {
        if((grados >= 0 && grados < 90))
            return 1;
        else
            return 0;
    }
    else
    {
        if((grados >= 0 && grados <= 91))
            return 1;
        else
            return 0;
    }
    default:
        break;
}

return 0;
}
```

Problemas encontrados a lo largo del desarrollo del TPO



En un principio se pensó en controlar tanto las articulaciones como el movimiento rotacional mediante la utilización de motores anexados a jeringas a través de mangueras. Al tener dificultades mecánicas para lograr una buena conversión del movimiento circular de los motores al movimiento lineal requerido para los émbolos de las jeringas, se decidió directamente poner servomotores en las articulaciones.

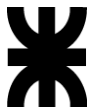


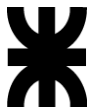
Imagen del prototipo original.







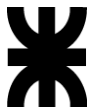
En algunas ocasiones se tuvieron problemas con el kit, por ejemplo: en una oportunidad el microcontrolador dejó de funcionar ya que se había quemado y luego el display LCD tuvo el mismo fin. Al reponer ambos componentes todo volvió a funcionar, aunque nunca se supo exactamente cuál fue la falla, ni se consiguió que alguien lograra detectarla. Gracias Mariano Vedovato por prestar el Styck al grupo.



Tras terminar la aplicación de la PC, testeándola, esta no recibía de forma adecuada los datos que le eran enviados desde el kit a través del puerto serie. Para solucionar este problema, se tuvieron que inicializar las variables de los índices de los buffers de transmisión y recepción de la UART.



-  Al poner a prueba la función de controlar el brazo haciendo uso de la interfaz gráfica en la PC, se notó que el tiempo de respuesta era demasiado alto. Por esta razón se decidió modificar la aplicación del microcontrolador, poniéndole flags para evitar así que ciertas instrucciones que causaban la sobredemora se ejecutaran.
-  Luego de un largo tiempo, de que se estaba probando el kit conectado al brazo robot, el módulo PWM no actualizaba sus salidas de inmediato al establecer un valor de DUTY, después de debuggear varias horas el código, no se encontró nada. Hasta que se le pregunto a Gabriel Soccodatto y comentó que lo valores del registro **PWM1R0** (establece el periodo de la señal) estaban mal, al arreglar este bug, no hubo mucho tiempo de por medio hasta que apareciera otra falla, pero esta vez el causante fue la no inicialización del registro **PWM1TC** que básicamente es el registro que cuenta el tiempo, esto causaba que hasta que el registro desborde había un tiempo largo de por medio, lo que causaba a simple vista que el servo no funcionara.
-  La fuente de alimentación fue un problema muy grande, en un principio se planeo utilizar alguna salida del KIT Info-Tronic que tuviera 5v, pero esta no soportaba tanta corriente debido a que el fusible reseteable se abría. Luego se colocó una fuente utilizada para cargar el celular, que no tuvo buenos resultados y por último se optó por utilizar una vieja fuente switching AT de PC con una salida de 5v 15A, según su hoja de datos.
-  El día anterior a la presentación se encontró que el brazo robot tenía un servo roto, ese imprevisto agarró al equipo de sorpresa y se tuvo que concurrir al laboratorio dos horas antes que sea arreglado.



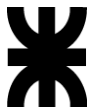
Beneficios encontrados a lo largo del desarrollo del TPO

- Se le podría agregar una memoria para que almacene los valores de los rangos de movimiento de los servos y así poder setear los límites dinámicamente.
- Agregar la opción de programar secuencias de movimientos para que el robot pueda realizar ciertas tareas de modo automático.
- Dotarle al equipo la posibilidad de que sean cargados scripts de movimientos mediante una tarjeta SD.

Este proyecto unió como participantes, enseñó a aprender a programar en equipo como es en la vida real, como así también a moldear el código a lo que el otro realiza, cosa que se cree muy importante a la hora de abordar un proyecto en la vida real.

Conclusiones

Se tuvieron muchos percances al realizar el proyecto, se llegó a callejones sin salida que se debieron sortear buscando alternativas en las cuales no se había pensado previamente para así llegar a la meta establecida, pero trayendo como consecuencia el desvirtuó de la idea fuerza. A lo largo de dicho camino se fue pasando por varios prototipos hasta dar con el que se expone. También, un par de servos se rompieron al ser la primera vez que se trabajaba con ellos, pero de esos errores se aprendió para mejorar así el funcionamiento y evitar futuras pérdidas. Finalmente, cuando se consiguió que el proyecto anduviera íntegramente, se vio para atrás y notó la cantidad de experiencias atravesadas y nuevos conocimientos adquiridos que se exponen en este informe.



Bibliografía, links, etc.

https://exploreembedded.com/wiki/LPC1768:_PWM

Hoja de datos LPC1769

Códigos Catedra



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Cátedra de Informática II