

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Objetivo:

- Realización de un auto robótico el cual se maneje inalámbricamente a través de la PC. Para ello se utiliza dos transceptores RF de la familia APC.
- El auto también debe tener una cámara, así se puede controlar sus movimientos y ver lo que hace. Se utiliza una cámara IP, conectada a través de WIFI con un router, la PC se conecta a dicho router y mediante el soft que trae la cámara, se puede ver lo que el auto está viendo. La cámara esta en un soporte el cual tiene 2 servomotores, lo que permite que la cámara tenga un movimiento vertical y horizontal.
- El auto cuenta con un brazo, el cual le permite tomar objetos livianos y llevarlos con él. El movimiento del brazo se realiza a través de 3 servomotores.

Introducción

Este proyecto tiene como fin la fabricación de un móvil el cual se maneja inalámbricamente, con ello uno se familiariza con la comunicación a través de módulos RF, con un entorno grafico el cual sirve para controlar al móvil desde la PC. También ya que cuenta con una cámara IP, uno también se familiariza con el armado de una pequeña red, para poder tener conectado la PC con la cámara. Tanto los motores DC, como los servomotes y los módulos RF son conectados a un microcontrolador Atmel SAM3X8E, el cual se encuentra integrado en el Arduino DUE, más adelante se explica porque la elección de este microcontrolador.

El manejo del móvil tiene 2 opciones, a través de un programa realizado en Visual C#, se lo puede controlar mediante el teclado de la PC, o mediante los botones que tiene el programa.

Elección del microcontrolador

Se elige la plataforma de Arduino por ser un hardware libre y gracias a ello, hay muchos ejemplos desarrollados con ella y su entorno de programación es bastante sencillo, sumado que se encuentra ampliamente disponible en el mercado local. Primeramente probé utilizando un Arduino UNO, el cual tiene un microcontrolador de 8bits, con este podía controlar la mayoría de los servos, pero los pines me resultaron escasos para la parte de los controles de los motores DC, entonces debía multiplexar varios pines y utilizar PWM para controlar la velocidad de estos motores.

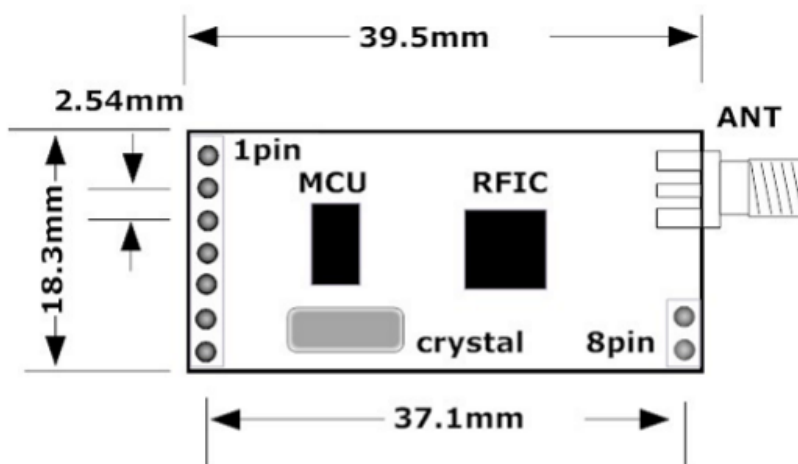
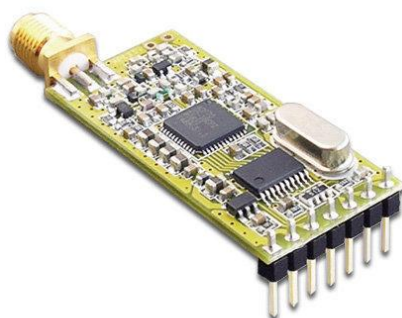
Entonces opte por utilizar Arduino DUE, el cual tiene un microcontrolador de 32bits, este micro me da la opción de elegir si quiero un PWM de 8 o 12 bits, además tengo más pines con lo que me evito utilizar multiplexores. Ya que el entorno de programación para todos los Arduino es idéntico, la programación de este micro de 32bits no fue complicado.



Configuración de parámetros de los módulos APC

En el proyecto se utiliza 2 módulos APC230, los cuales tiene un rango de frecuencia entre 418Mhz y 455Mhz, un alcance máximo de 1500mts (2400bps), Interfaz de comunicación UART TTL.

Este módulo es un transceptor half-duplex, totalmente transparente permitiendo configurar todos sus parámetros a criterio del proyecto.



PIN OUT

PIN	NOMBRE	FUNCION	DESCRIPCION
1	GND	POWER	TIERRA (0v)
2	VCC	POWER	ALIMENTACION DC 3.5 – 5.5 V
3	EN	ENTRADA	HABILITADO = 1 LÓGICO SLEEP = 0 LÓGICO
4	RXD	ENTRADA	ENTRADA UART TTL
5	TXD	SALIDA	SALIDA UART TTL
6	AUX	ENTRADA	NO SE UTILIZA
7	SET	ENTRADA	MODO SETEO = 0 LÓGICO MODO NORMAL = 1 LÓGICO

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Estos APC tienen unos parámetros que se deben fijar, para poder usarlos posteriormente:

- **Canal de Frecuencia:** Dentro de la banda de frecuencia del módulo, se puede elegir en que frecuencia exacta quiere que el módulo trabaje.
- **Baude Rate Serie:** Con este parámetro le vamos a indicar al módulo con que tasa de transferencia nos vamos a comunicar con él. Esta tasa de transferencia es igual a la que utiliza la comunicación SERIAL del Arduino.
- **Paridad serie:** Acá seleccionaremos el chequeo de paridad que queremos realizar, podemos elegir entre paridad par, paridad impar o sin paridad.
- **Potencia de salida:** Nos indica la máxima potencia que el módulo puede generar.
- **RF Rate:** Con este parámetro indicamos a qué velocidad queremos que se transmitan los datos y a qué velocidad se prepara para recibir.

Tabla de parámetros del APC230

Parámetro	Opciones	De fábrica
Velocidad serie	2400, 4800, 9600, 19200, 38400, 57200	9600
Paridad	Par, Impar, Sin paridad	Sin paridad
Frecuencia	418 – 455 Mhz	434 Mhz
Velocidad RF	2400, 4800, 9600, 19200	9600
Potencia RF	1 a 9	9 (100mW)

La configuración de estos parámetros se los puede hacer a través de un software y un conector USB. Este es un conversor para adaptar los niveles de tensión del USB de la PC a valores TTL UART para el APC.


Se instala el programa RF-Magic, y con este se fijan los parámetros antes mencionados. En ambos módulos se deben fijar los mismos parámetros así puedan comunicarse entre ellos.

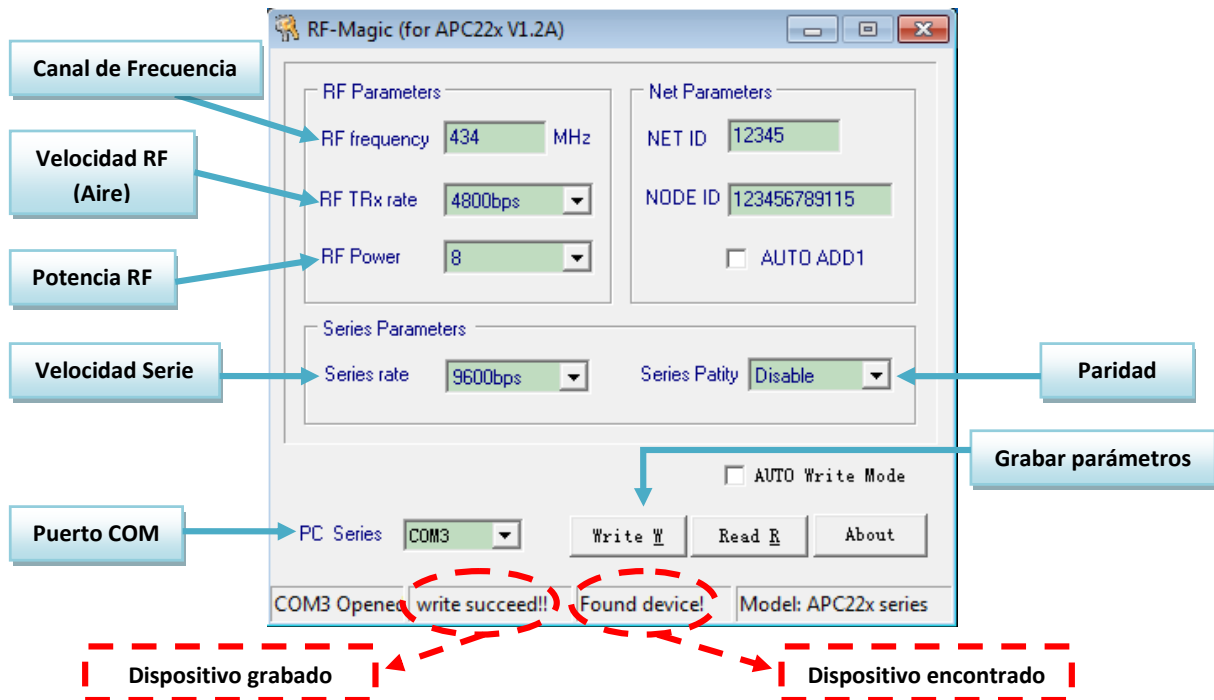
NOTA IMPORTANTE: Primeramente tuve problemas con este soft, porque no lograba que el soft detecte al APC conectado, lo que tuve que realizar fue ejecutarlo al soft como administrador (en Windows 7) y luego conectarlo al APC, a partir de ahí el soft reconocía al APC.

Como se puede ver en la tabla anterior, los módulos APC vienen configurados de fabrica con ciertos parámetros, con esos los módulos pueden comunicarse sin problemas.

Se mantienen la mayoría de los parámetros de fábrica en este proyecto, pero se cambia la velocidad de transmisión en el aire, para alcanzar una distancia próxima a la máxima. Por eso se fija un valor de 4800bps.

Una vez fijados todos esos parámetros, se debe presionar sobre el botón “Write W” así se guardan las configuraciones en los APC.

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	



Control de los motores DC

El móvil cuenta con 4 motores DC, cada uno con caja de reducción de velocidad, esto le permite tener velocidad y tracción al robot en las 4 ruedas. La etapa de potencia de control de estos motores se la realiza mediante 2 integrados L293D, de la familia ST, estos son mejores en comparación con los mismos integrados de la familia de Texas, ya que cuentan con Diodos internos para evitar corrientes parasitas provenientes de los motores.

El integrado L293D incluye cuatro circuitos para manejar cargas de potencia media, en especial pequeños motores y cargas inductivas, con la capacidad de controlar corriente hasta 600 mA en cada circuito y una tensión entre 4,5 V a 36 V.

El integrado permite formar, entonces, dos puentes H completos, con los que se puede realizar el manejo de dos motores. En este caso el manejo será bidireccional, con frenado rápido y con posibilidad de implementar fácilmente el control de velocidad.

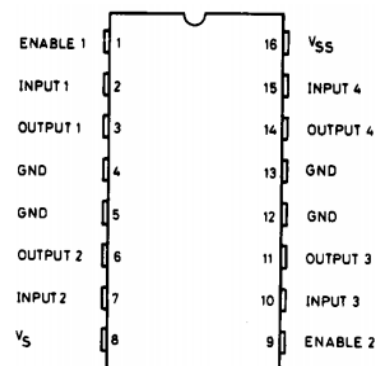
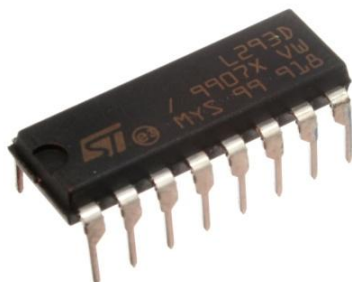




Diagrama detallado del circuito interno

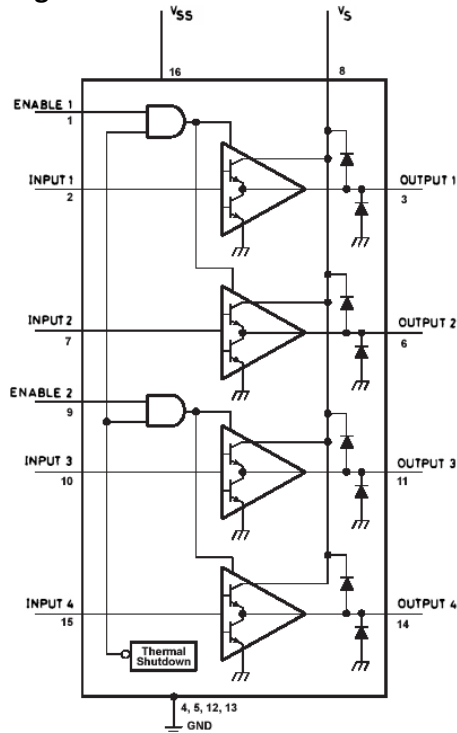
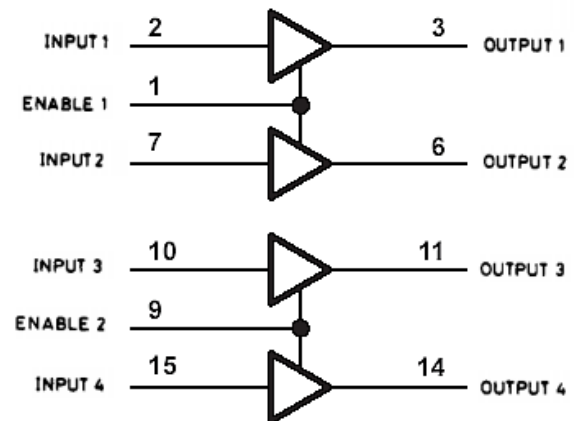


Diagrama simplificado



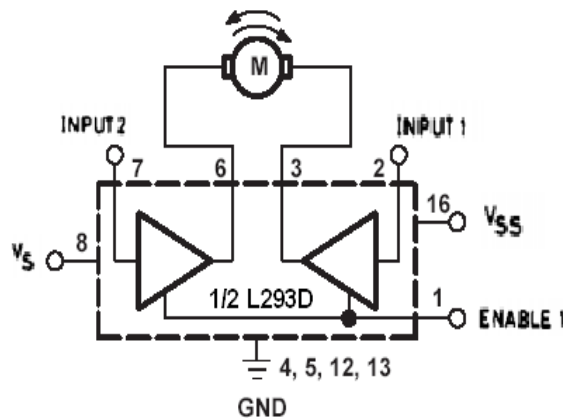
Las entradas son compatibles con niveles de lógica TTL. Para lograr esto, incluso cuando se manejen motores de voltajes no compatibles con los niveles TTL, el chip tiene patas de alimentación separadas para la lógica (V_{SS} , que debe ser de 5V) y para la alimentación de la carga (V_S , que puede ser entre 4,5V y 36V).

Los circuitos de salida se pueden habilitar en pares por medio de una señal TTL. Los circuitos de manejo de potencia 1 y 2 se habilitan con la señal “**ENABLE 1**” y los circuitos 3 y 4 con la señal “**ENABLE 2**”.

Las entradas de habilitación permiten controlar con facilidad el circuito, lo que facilita la regulación de velocidad de los motores por medio de PWM.

Las salidas actúan cuando su correspondiente señal de habilitación (ENABLE) está en alto. En estas condiciones, las salidas (OUTPUT) están activas y su nivel varía en relación con las entradas (INPUT). Cuando la señal de habilitación del par de circuitos de manejo está en bajo, las salidas están desconectadas y en un estado de alta impedancia.

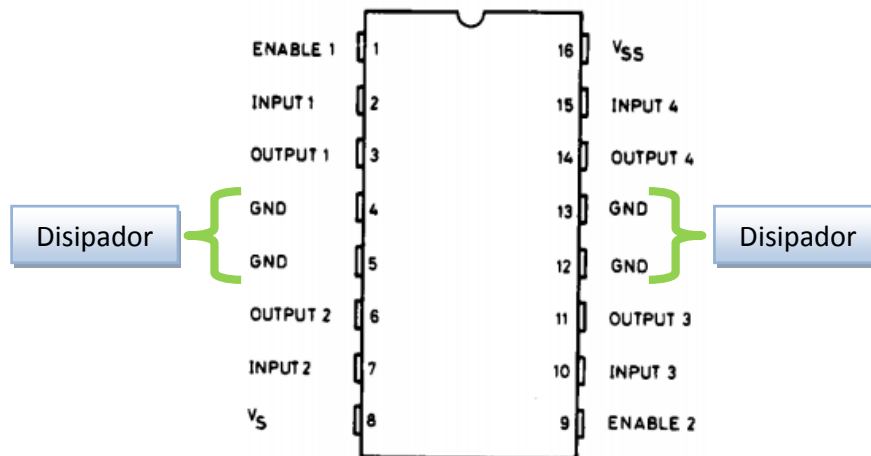
Por medio de un control apropiado de las señales de entrada (INPUT y ENABLE) y conectando el motor a salidas de potencia, cada par de circuito de manejo de potencia conforma un **PUENTE H** completo, permitiendo de esta manera un control bidireccional del motor y además una detección rápida del motor.



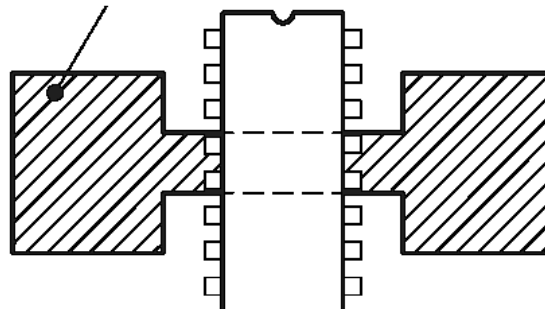
ENABLE1	INPUT1	INPUT2	FUNCION
H	L	H	Giro a la derecha
H	H	L	Giro a la izquierda
H	L	L	Detención rápida
H	H	H	Detención rápida
L	X	X	Detención rápida

L = Bajo H = Alto X = No afecta

Las pines centrales del integrado están pensadas para proveer el contacto térmico con un disipador que permitirá lograr la potencia, también dichos pines son de conexión a GND. Por eso el fabricante ofrece dos opciones de disipador, uno externo y otro realizado en el propio PCB del circuito. En este caso se utilizó el disipador en el PCB, ya que no se encontró disipador para este integrado.



Área de cobre de 35 micrómetros de espesor



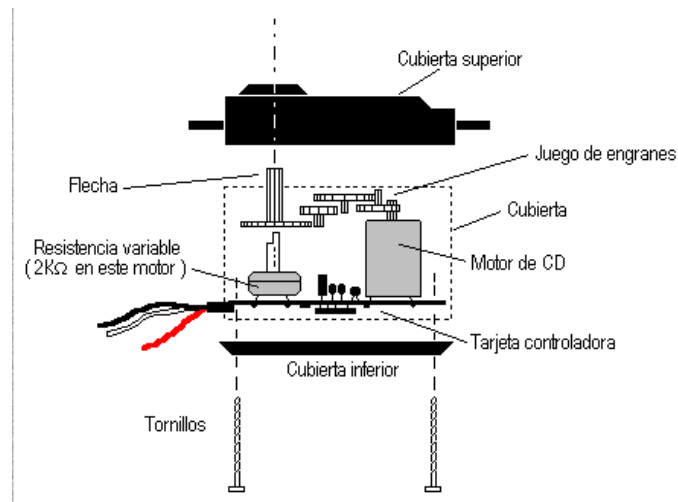
BASE DEL DISIPADOR EN EL CIRCUITO IMPRESO

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Control de los servomotores

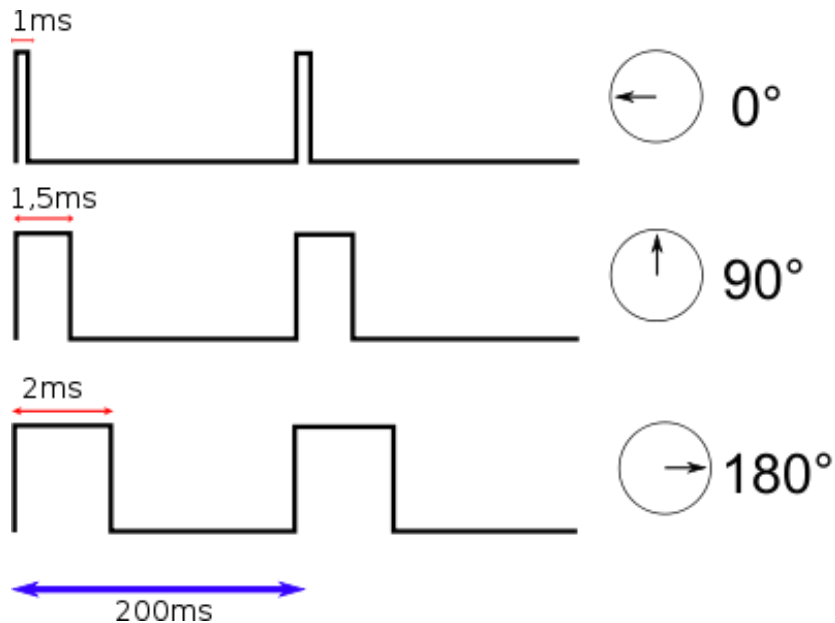
La característica de los servomotores es que mediante PWM estos pueden girar un cierto ángulo, la mayoría de 0° a 180°. Esto se logra porque internamente tiene una caja reductora, permitiendo que tenga un gran torque, y un circuito de control (realimentación) con un potenciómetro.

El motor del servo tiene algunos circuitos de control y un potenciómetro conectado al eje central del motor. Este potenciómetro permite a la circuitería de control, supervisar el ángulo actual del servomotor. Si el eje está en el ángulo correcto, entonces el motor está apagado. Si el circuito chequea que el ángulo no es correcto, el motor volverá a la dirección correcta, hasta llegar al ángulo que es correcto. El eje del servo es capaz de llegar alrededor de los 180 grados.

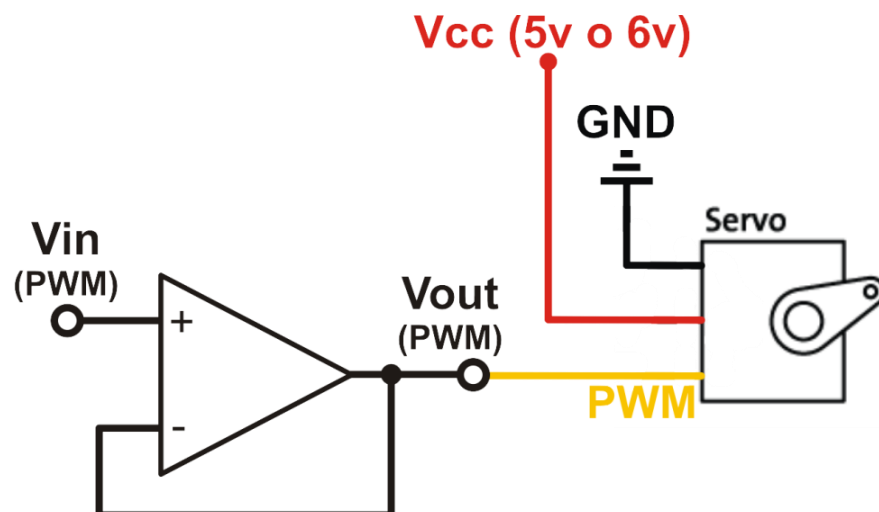


El sistema de control de un servo se limita a indicar en qué posición se debe situar. Esto se lleva a cabo mediante una serie de pulsos tal que la duración del pulso indica el ángulo de giro del motor. Cada servo tiene sus márgenes de operación, que se corresponden con el ancho del pulso máximo y mínimo que el servo entiende. Los valores más generales se corresponden con pulsos de entre 1 ms y 2 ms de anchura, que dejarían al motor en ambos extremos (0° y 180°). El valor 1.5 ms indicaría la posición central o neutra (90°). Si se sobrepasan los límites de movimiento del servo, éste comenzará a emitir un zumbido, indicando que se debe cambiar la longitud del pulso. El factor limitante es el tope del potenciómetro y los límites mecánicos constructivos.

Es importante destacar que para que un servo se mantenga en la misma posición durante un cierto tiempo, es necesario enviarle continuamente el pulso correspondiente. De este modo, si existe alguna fuerza que le obligue a abandonar esta posición, intentará resistirse. Si se deja de enviar pulsos (o el intervalo entre pulsos es mayor que el máximo) entonces el servo perderá fuerza y dejará de intentar mantener su posición, de modo que cualquier fuerza externa podría desplazarlo.



En el proyecto se utilizan 5 servos, 2 TowerPro que se alimentan con 5V, y 3 Hitec que se alimentan con 6V. Debido a que era un gran consumo cuando funcionaban todos juntos, lo que hacía que los servos se muevan con error, se conecto un grupo de servos a una fuente de 5V y otro grupo de servos a una fuente de 6V. Además tenían un consumo de corriente excesivo funcionando todos juntos, entonces se utiliza dos LM324, y se hace 5 seguidor-emisivo, uno para cada servo.

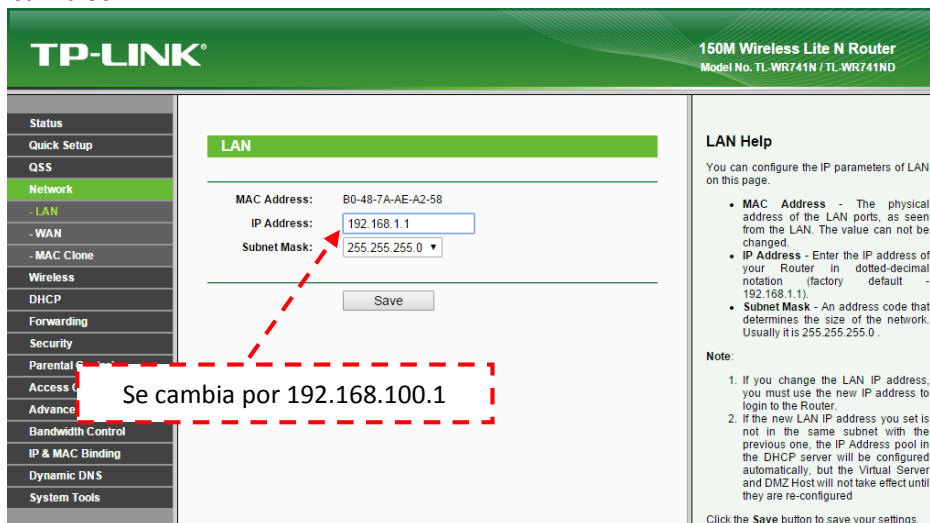


Utilizo el LM324 para realizar el seguidor ya que cuenta con 4 operacionales por integrado, y no es necesario utilizar fuente con tensión negativa para alimentar $-V_{cc}$, simplemente con conectarlo a GND funciona bien.

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez		Tema: Auto Robótico
Alumno: Belmonte Carlos		Fecha: Febrero/2015

Configuración de Cámara IP y router

Tanto el router como la cámara IP son de la marca TP-LINK, ya que ambos cuentan con WIFI, se puede armar una red pequeña inalámbrica, los cuales cumplen con los fines del proyecto. Lo primero que se realiza es la configuración del router. Este viene por defecto con la siguiente dirección IP 192.168.1.1, pero en este caso se la cambia por 192.168.100.1, en la opción "Network → LAN". Para entrar a estas opciones, se abre el explorador de internet y se pone la dirección que trae el router por defecto. Una vez cambiado, se presiona el botón "SAVE" para guardar los cambios.



TP-LINK® 150M Wireless Lite N Router
Model No. TL-WR741N / TL-WR741ND

LAN

MAC Address: B0-48-7A-AE-A2-58
IP Address: 192.168.1.1
Subnet Mask: 255.255.255.0

Save

LAN Help

You can configure the IP parameters of LAN on this page.

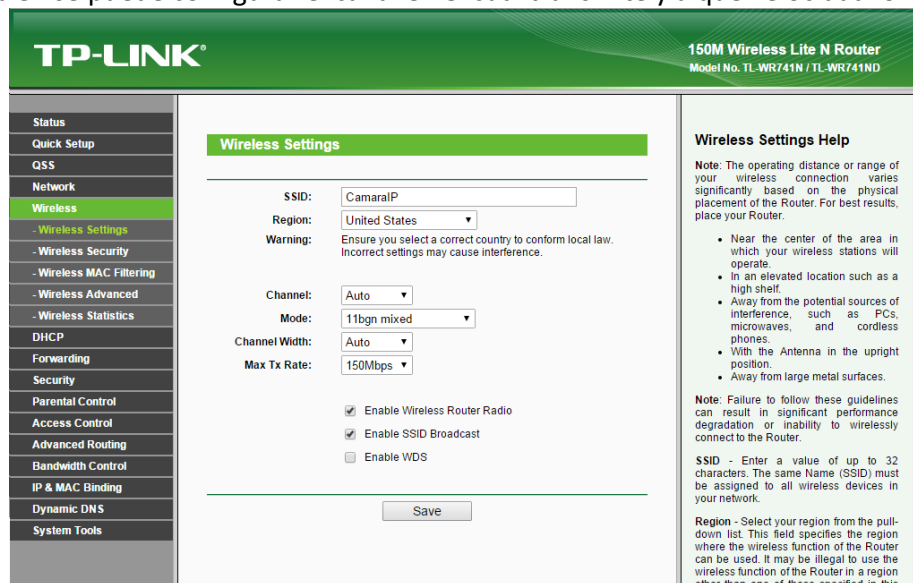
- MAC Address** - The physical address of the LAN ports, as seen from the LAN. The value can not be changed.
- IP Address** - Enter the IP address of your Router in dotted-decimal notation (factory default - 192.168.1.1).
- Subnet Mask** - An address code that determines the size of the network. Usually it is 255.255.255.0.

Note:

- If you change the LAN IP address, you must use the new IP address to login to the Router.
- If the new LAN IP address you set is not in the same subnet with the previous one, the IP Address pool in the DHCP server will be configured automatically, but the Virtual Server and DMZ Host will not take effect until they are re-configured.

Click the Save button to save your settings.

En la opción "Wireless → Wireless Settings" se cambia el nombre al router, se fija la región donde esta, y también se puede configurar el canal en el cual transmite y a qué velocidad lo hace.



TP-LINK® 150M Wireless Lite N Router
Model No. TL-WR741N / TL-WR741ND

Wireless Settings

SSID: CamaraIP
Region: United States
Warning: Ensure you select a correct country to conform local law. Incorrect settings may cause interference.
Channel: Auto
Mode: 11bgn mixed
Channel Width: Auto
Max Tx Rate: 150Mbps

☒ Enable Wireless Router Radio
☒ Enable SSID Broadcast
☐ Enable WDS

Save

Wireless Settings Help

Note: The operating distance or range of your wireless connection varies significantly based on the physical placement of the Router. For best results, place your Router.

- Near the center of the area in which your wireless stations will operate.
- In an elevated location such as a high shelf.
- Away from the potential sources of interference, such as PCs, microwaves, and cordless phones.
- With the Antenna in the upright position.
- Away from large metal surfaces.

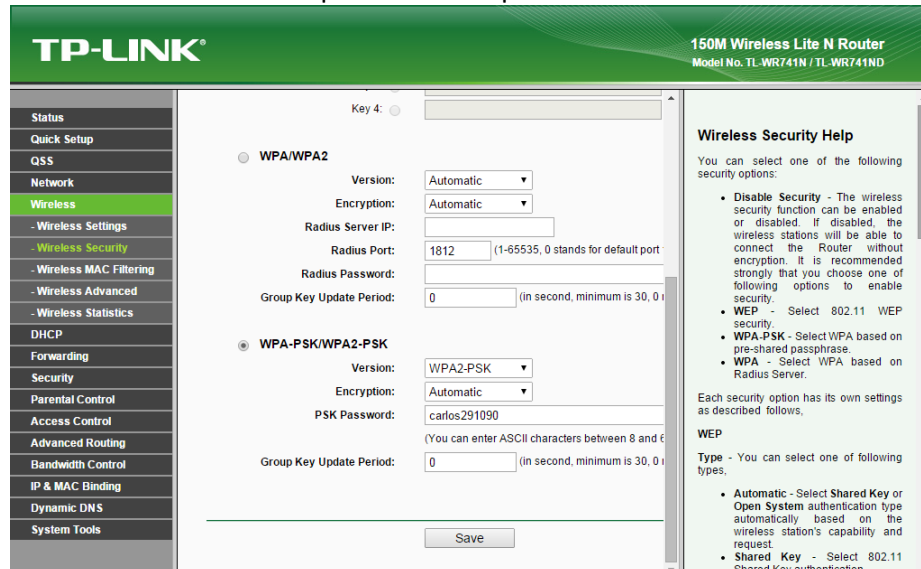
Note: Failure to follow these guidelines can result in significant performance degradation or inability to wirelessly connect to the Router.

SSID - Enter a value of up to 32 characters. The same Name (SSID) must be assigned to all wireless devices in your network.

Region - Select your region from the pull-down list. This field specifies the region where the wireless function of the Router can be used. It may be illegal to use the wireless function of the Router in a region other than one of those specified in this

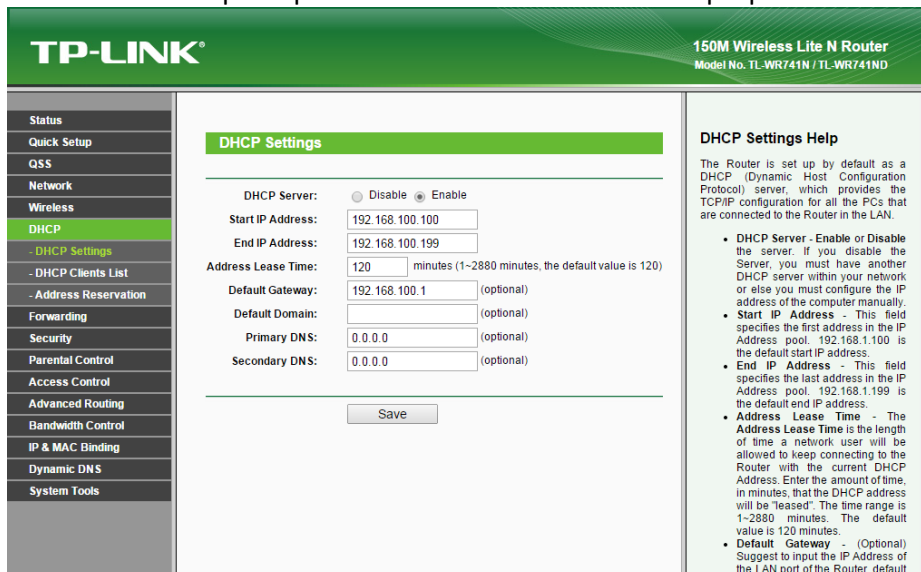
	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

En la opción “Wireless → Wireless Security” se pone una contraseña a la red WIFI, así solo los usuarios que conozcan la contraseña podrán ver lo que ve el robot.



The screenshot shows the 'Wireless Security' configuration page of a TP-LINK 150M Wireless Lite N Router. The left sidebar contains a menu with options like Status, Quick Setup, QSS, Network, Wireless, and Security. The 'Wireless Security' section is selected. The main area shows two security options: WPA/WPA2 (selected) and WPA-PSK/WPA2-PSK. Under WPA/WPA2, fields for Version (Automatic), Encryption (Automatic), Radius Server IP, Radius Port (1812), Radius Password, and Group Key Update Period are visible. Under WPA-PSK/WPA2-PSK, fields for Version (WPA2-PSK), Encryption (Automatic), PSK Password (carlos291090), and Group Key Update Period are visible. A 'Save' button is at the bottom. A 'Wireless Security Help' sidebar on the right provides additional information about security options.

DHCP (Dynamic Host Configuration Protocol, protocolo de configuración dinámica de host) es un protocolo de red que permite a los clientes de una red obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme éstas van quedando libres, sabiendo en todo momento quién ha estado en posesión de esa IP, cuánto tiempo la ha tenido y a quién se la ha asignado después. Esta opción se habilita y se configura un rango de IPs que van desde 192.168.100.100 a 192.168.100.199, lo que da un rango 100 IPs disponibles para conectarse con el router. Se toma el valor de 100 como comienzo para que estas IP no interfieran con las propias del router.

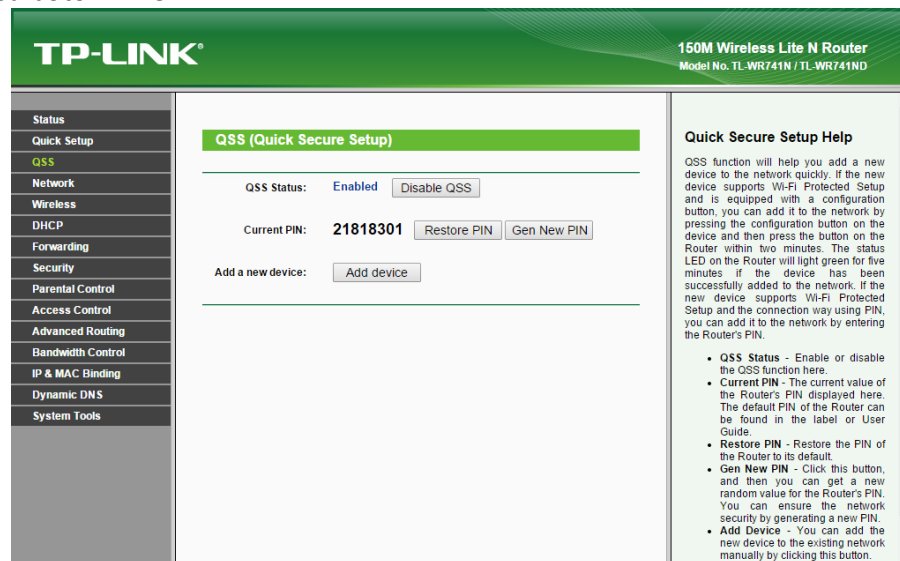


The screenshot shows the 'DHCP Settings' configuration page of a TP-LINK 150M Wireless Lite N Router. The left sidebar contains a menu with options like Status, Quick Setup, QSS, Network, Wireless, and DHCP. The 'DHCP' section is selected. The main area shows the 'DHCP Settings' section with fields for DHCP Server (Disable/Enable), Start IP Address (192.168.100.100), End IP Address (192.168.100.199), Address Lease Time (120 minutes), Default Gateway (192.168.100.1), Default Domain, Primary DNS (0.0.0.0), and Secondary DNS (0.0.0.0). A 'Save' button is at the bottom. A 'DHCP Settings Help' sidebar on the right provides additional information about DHCP settings.

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

QSS (Configuración de seguridad rápida) es propio de routers TP-LINK y cumple con WPS, este nos permite conectar cualquier aparato a nuestra red doméstica con tan sólo apretar un botón en el router y con introducir un número PIN.

Esta opción tiene que estar habilitada, así al presionar el botón QSS del router pueda conectar inalámbricamente la cámara IP, sin necesidad de ingresar la clave de la red. La cámara también cuenta con su botón WPS.



TP-LINK® 150M Wireless Lite N Router
Model No. TL-WR741N / TL-WR741ND

QSS (Quick Secure Setup)

QSS Status: **Enabled**

Current PIN: **21818301**

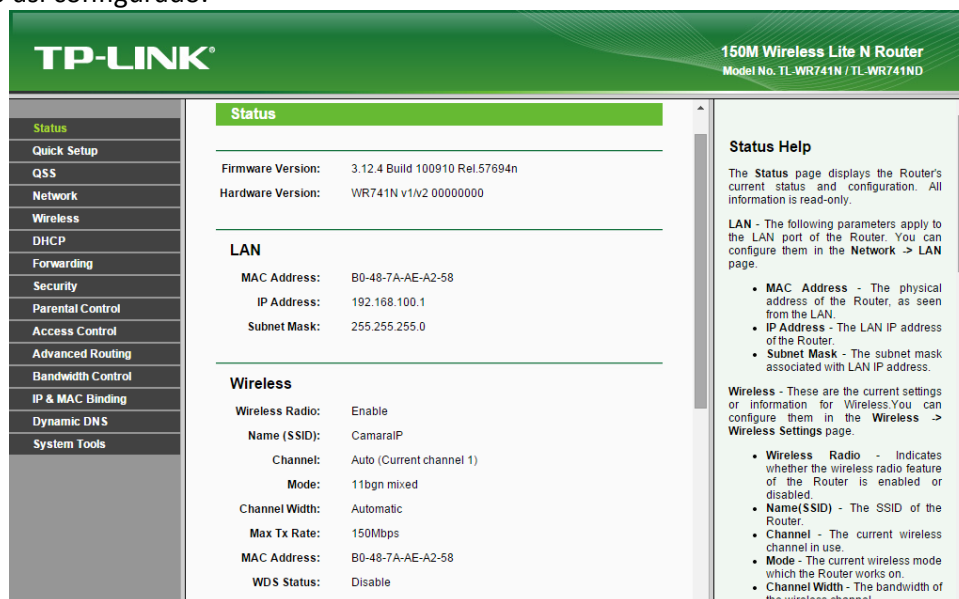
Add a new device:

Quick Secure Setup Help

QSS function will help you add a new device to the network quickly. If the new device supports Wi-Fi Protected Setup and is equipped with a configuration button, you can add it to the network by pressing the configuration button on the device and then press the button on the Router within two minutes. The status LED on the Router will light green for five minutes if the device has been successfully added to the network. If the new device supports Wi-Fi Protected Setup and the connection way using PIN, you can add it to the network by entering the Router's PIN.

- **QSS Status** - Enable or disable the QSS function here.
- **Current PIN** - The current value of the Router's PIN displayed here. The default PIN of the Router can be found in the label or User Guide.
- **Restore PIN** - Restore the PIN of the Router to its default.
- **Gen New PIN** - Click this button, and then you can get a new random value for the Router's PIN. You can ensure the network security by generating a new PIN.
- **Add Device** - You can add the new device to the existing network manually by clicking this button.

Con estos parámetros mínimos ya se puede utilizar el router para hacer una pequeña red inalámbrica y poder conectar la cámara a esta. Se guardan todos los datos y se reinicia el router, quedando así configurado:



TP-LINK® 150M Wireless Lite N Router
Model No. TL-WR741N / TL-WR741ND

Status

Firmware Version: 3.12.4 Build 100910 Rel.57694n
Hardware Version: WR741N v1/V2 00000000

LAN

MAC Address: B0-48-7A-AE-A2-58
IP Address: 192.168.100.1
Subnet Mask: 255.255.255.0

Wireless

Wireless Radio: Enable
Name (SSID): CamaraIP
Channel: Auto (Current channel 1)
Mode: 11bgn mixed
Channel Width: Automatic
Max Tx Rate: 150Mbps
MAC Address: B0-48-7A-AE-A2-58
WDS Status: Disable

Status Help


The Status page displays the Router's current status and configuration. All information is read-only.

LAN - The following parameters apply to the LAN port of the Router. You can configure them in the **Network** -> **LAN** page.

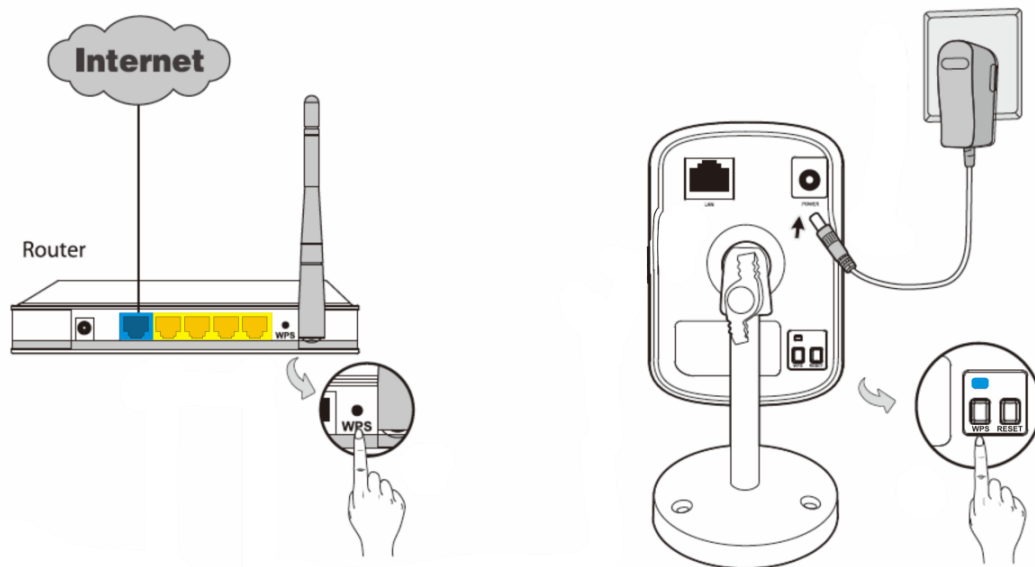
- **MAC Address** - The physical address of the Router, as seen from the LAN.
- **IP Address** - The LAN IP address of the Router.
- **Subnet Mask** - The subnet mask associated with LAN IP address.

Wireless - These are the current settings or information for Wireless. You can configure them in the **Wireless** -> **Wireless Settings** page.

- **Wireless Radio** - Indicates whether the wireless radio feature of the Router is enabled or disabled.
- **Name (SSID)** - The SSID of the Router.
- **Channel** - The current wireless channel in use.
- **Mode** - The current wireless mode which the Router works on.
- **Channel Width** - The bandwidth of the wireless channel.

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Una vez configurado el router, queda configurar la cámara IP para conectarla inalámbricamente. La conexión de la cámara y el router se hace más sencilla a través de WPS, ya que ambas cuentan con esta configuración, lo que se debe hacer es presionar el botón QSS del router y luego apretar el botón WPS de la cámara, la cámara trae un led frontal, tiene 2 colores, rojo y azul. En rojo indica que la cámara no está conectada, cuando está en azul, la cámara ya se conectó al router. La cámara tomará un IP dinámico que debe verificarse en las configuraciones del router, este es dinámico debido a las configuraciones DHCP de la cámara y del router, debido a que la red es doméstica y solo para el manejo de la cámara IP, con esta configuración básica de IP sirve para ver inalámbricamente la cámara. En caso de necesitar una red más segura (que no es el caso del proyecto), se debe configurar la cámara primeramente mediante el cable de Ethernet y ponerle una IP fija, a su vez el router debe tener deshabilitada la opción DHCP.



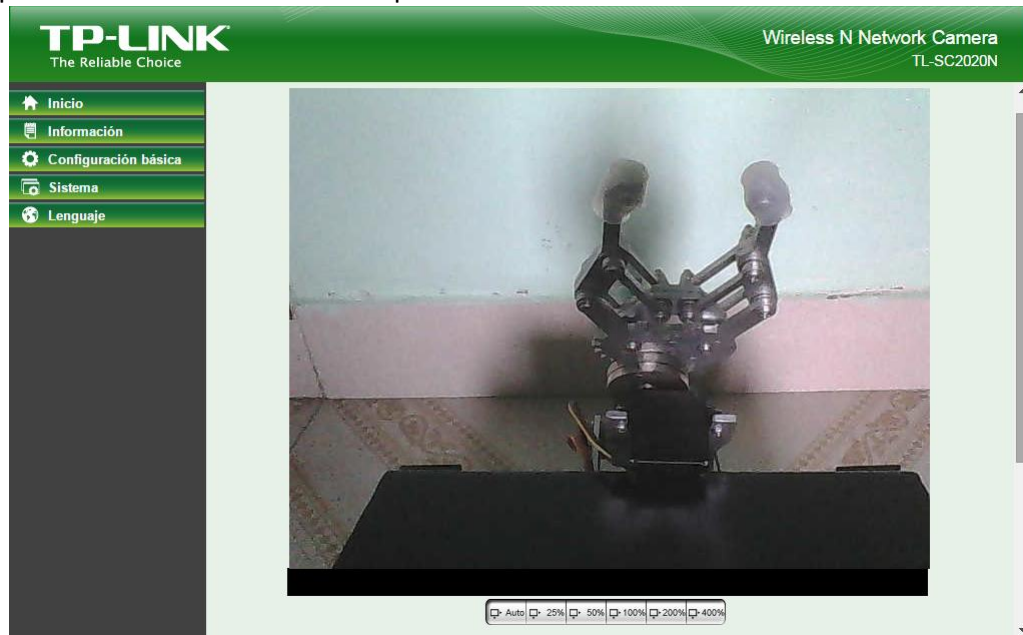
En el router se puede ver lo siguiente:

TP-LINK®
150M Wireless Lite N Router
Model No. TL-WR741N / TL-WR741ND

<ul style="list-style-type: none"> Status Quick Setup QSS Network Wireless <li style="background-color: #008000; color: white;">DHCP <li style="background-color: #cccccc;">- DHCP Settings <li style="background-color: #cccccc;">- DHCP Clients List <li style="background-color: #cccccc;">- Address Reservation Forwarding Security Parental Control Access Control Advanced Routing Bandwidth Control IP & MAC Binding Dynamic DNS System Tools 	<div style="background-color: #008000; color: white; padding: 2px; text-align: center; font-weight: bold;">DHCP Clients List</div> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th>ID</th> <th>Client Name</th> <th>MAC Address</th> <th>Assigned IP</th> <th>Lease Time</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>BelmonteNetbook</td> <td>E0-CA-94-2D-F9-92</td> <td>192.168.100.101</td> <td>01:40:49</td> </tr> <tr> <td>2</td> <td>TL-SC2020N</td> <td>F8-D1-11-7D-16-CE</td> <td>192.168.100.100</td> <td>01:46:14</td> </tr> </tbody> </table> <div style="text-align: center; margin-top: 10px;"> <input type="button" value="Refresh"/> </div>	ID	Client Name	MAC Address	Assigned IP	Lease Time	1	BelmonteNetbook	E0-CA-94-2D-F9-92	192.168.100.101	01:40:49	2	TL-SC2020N	F8-D1-11-7D-16-CE	192.168.100.100	01:46:14	DHCP Clients List Help <small>This page shows Client Name, MAC Address, Assigned IP and Lease Time of each DHCP Client connected to the Router.</small> <ul style="list-style-type: none"> Client Name - The name of the DHCP client. MAC Address - The MAC address of the DHCP client. Assigned IP - The IP address that the Router has allocated to the DHCP client. Lease Time - The time of the DHCP client leased. <small>You cannot change any of the values on this page. To update this page and to show the current connected devices, click on the Refresh button.</small>
ID	Client Name	MAC Address	Assigned IP	Lease Time													
1	BelmonteNetbook	E0-CA-94-2D-F9-92	192.168.100.101	01:40:49													
2	TL-SC2020N	F8-D1-11-7D-16-CE	192.168.100.100	01:46:14													

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Se asignó una dirección IP (dinámica) a la cámara, con esta dirección podemos entrar a las opciones de esta y a su vez visualizar las imágenes/video que transmite la cámara. Para entrar a estas opciones se debe escribir en el explorador de internet la dirección IP de la cámara.



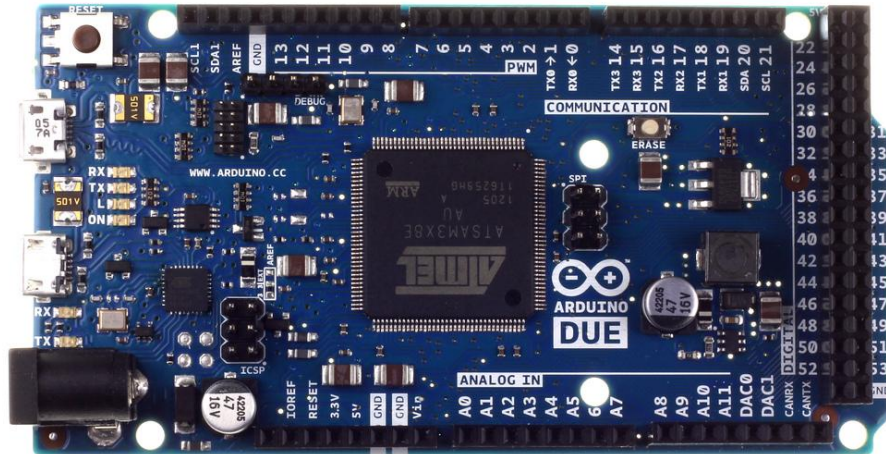
El router es un TL-WR741ND y la cámara IP es una TL-SC2020N



	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Distribución de pines del microcontrolador y funciones

Debido a las amplias ventajas de Arduino DUE (pines, velocidad, programación) se elije esta plataforma, se podría usar Arduino UNO o Mega, pero cierto pines se tendrían que multiplexar aumentando la cantidad de integrados y haciendo el programa más extenso.



Los pines que se utilizan son los siguientes:

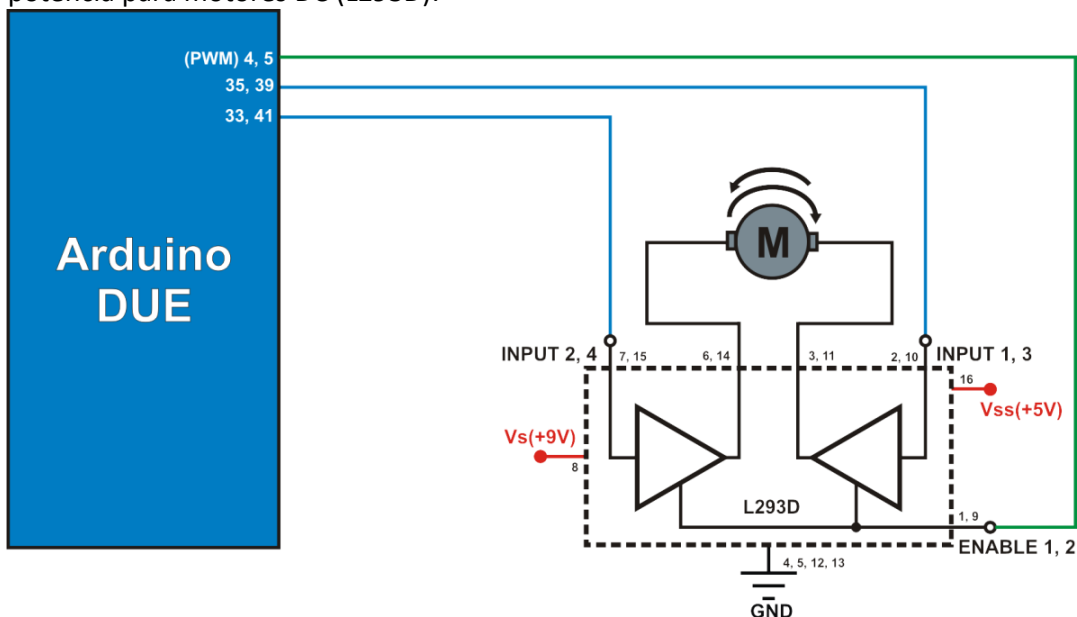
Pines Arduino	Función
0 (Rx)	Es el receptor de la comunicación serial que se hace con el APC, este va conectado al Tx del APC.
1 (Tx)	Es el transmisor de la comunicación serial que se hace con el APC, este va conectado al Rx del APC.
2 (PWM)	Genera la señal PWM que utiliza el ENABLE 1 del primer L293D.
3 (PWM)	Genera la señal PWM que utiliza el ENABLE 2 del primer L293D.
4 (PWM)	Genera la señal PWM que utiliza el ENABLE 1 del segundo L293D.
5 (PWM)	Genera la señal PWM que utiliza el ENABLE 2 del segundo L293D.
8	Pin digital de salida para controlar el servo de la pinza. Genera PWM.
9	Pin digital de salida para controlar el servo del brazo, movimiento vertical. Genera PWM.
10	Pin digital de salida para controlar el servo del brazo, movimiento rotacional. Genera PWM.
11	Pin digital de salida para controlar el servo de la cámara, movimiento horizontal. Genera PWM.
12	Pin digital de salida para controlar el servo de la cámara, movimiento vertical. Genera PWM.
25	Pin digital de salida para controlar las luces del robot.
33	Pin digital de salida para controlar INPUT4 del segundo L293D.
35	Pin digital de salida para controlar INPUT3 del segundo L293D.
39	Pin digital de salida para controlar INPUT1 del segundo L293D.
41	Pin digital de salida para controlar INPUT2 del segundo L293D.

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

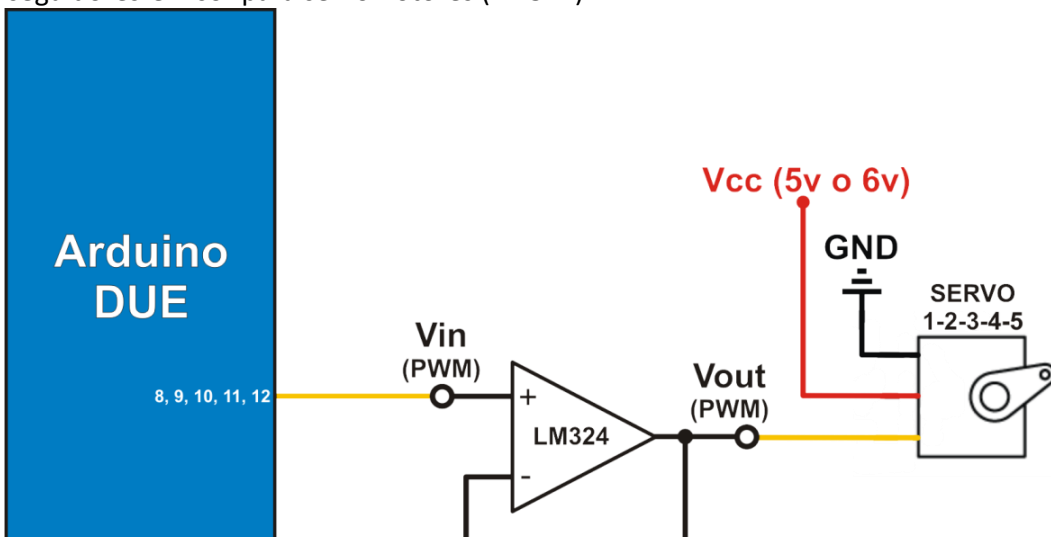
Pines Arduino	Función
45	Pin digital de salida para controlar INPUT2 del primer L293D.
47	Pin digital de salida para controlar INPUT1 del primer L293D.
51	Pin digital de salida para controlar INPUT3 del primer L293D.
53	Pin digital de salida para controlar INPUT4 del primer L293D.

Muchos pares de pines cumplen funciones similares, por eso se muestra parte del diagrama para un grupo de pines, ya que los otros se repiten. El esquemático detallado de todo el proyecto se adjunta al final.

Etapas potencia para motores DC (L293D):

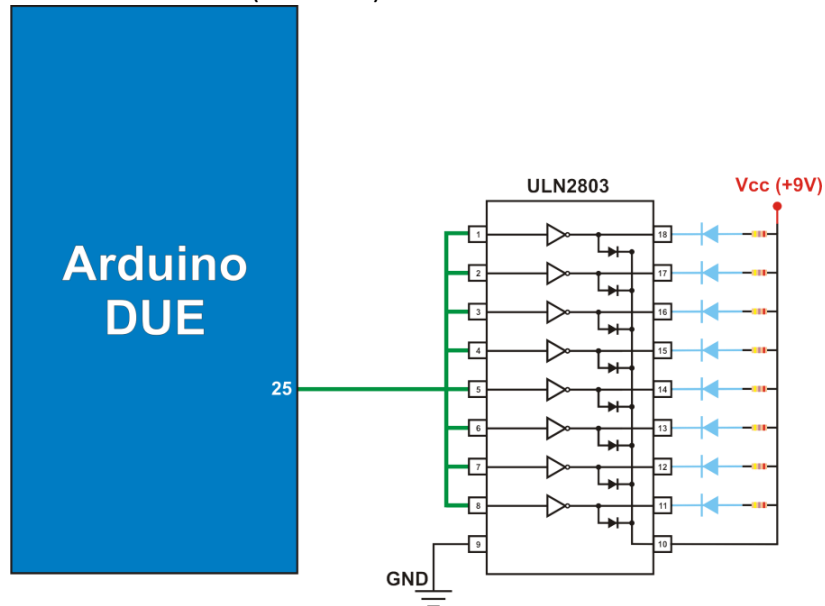


Etapas seguidores-emisor para servomotores (LM324):

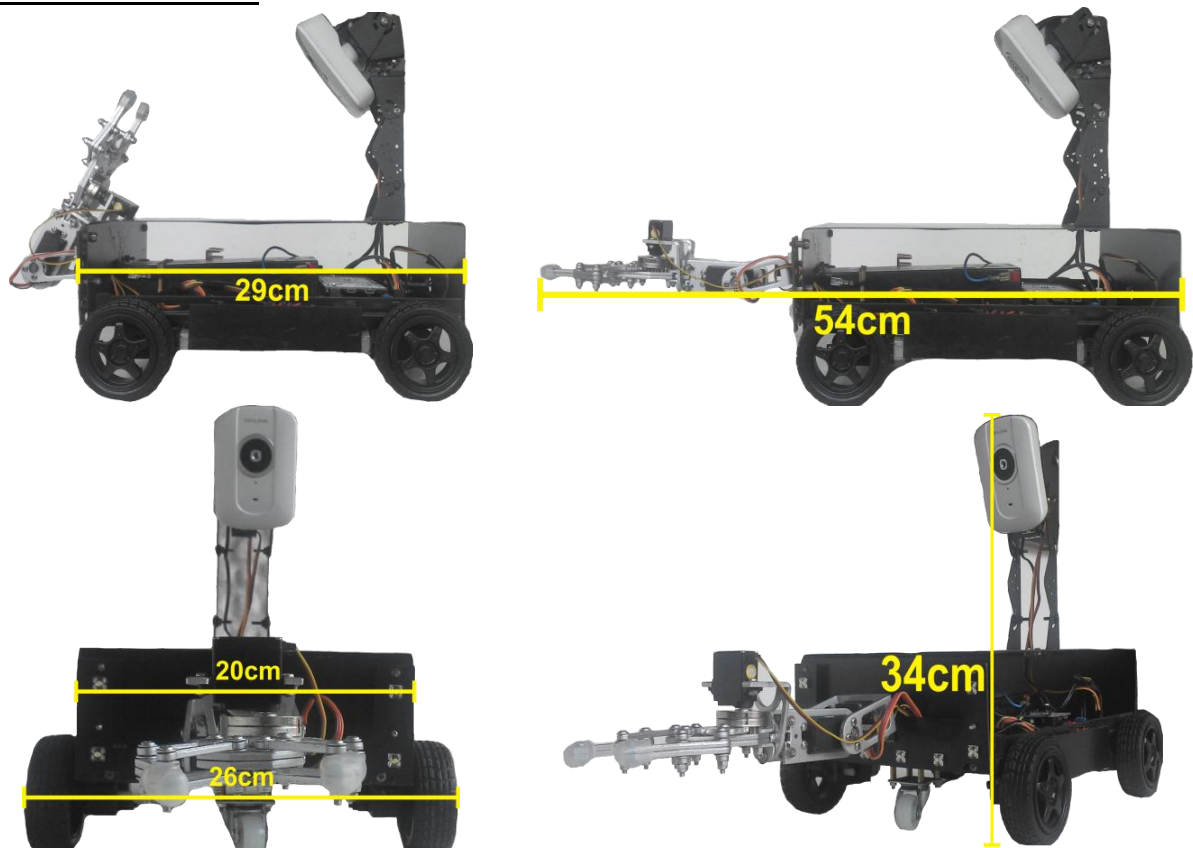


	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Etapa del control de luces del robot (ULN2803):



Dimensiones del robot



	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

Programación del proyecto

La etapa de programación del proyecto se divide en dos partes, una parte dedicada a la programación del micro y la otra parte de la programación del software de manejo del proyecto. Se explicara solo las partes importantes de la programación, ya que algunos procesos son repetitivos, la programación también es así y se asemeja bastante, pero al final se adjunta todo la programación del proyecto.

Etapa del micro:

La programación de Arduino DUE es idéntica a la programación de otros de su misma familia (UNO, Mega, etc.) el lenguaje que utiliza es propio de Arduino, pero se asemeja bastante al C/C++ ya que está basado en estos.

La estructura del programa es relativamente fácil y se compone de 2 partes, la parte de **setup()** encargada de recoger la información, se declaran variables, se definen los pines a utilizar y como son sus estados (Entrada o Salida). Es la primera función a ejecutar en el programa, se ejecuta sólo una vez. Debe ser incluido en un programa aunque no haya declaración que ejecutar. La otra parte es **loop()** es la que contiene el programa que se ejecutará cíclicamente (lectura de entradas, activación de salidas, etc.). Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

Otra parte importante de este programa son las **funciones**, estas son un bloque de código que tiene un nombre y un conjunto de instrucciones que son ejecutadas cuando se llama a la función. Se las utilizan para realizar tareas repetitivas y reducir el tamaño del programa. Las funciones se declaran asociadas a un tipo de valor "**type**". Este valor será el que devolverá la función, por ejemplo "int" se utilizará cuando la función devuelve un dato numérico de tipo entero. Si la función no devuelve ningún valor entonces se colocará delante la palabra "void", que significa "función vacía". Después de declarar el tipo de dato que devuelve la función se debe escribir el nombre de la función y entre paréntesis se escribirán, si es necesario, los parámetros que se deben pasar a la función para que se ejecute. Más adelante se muestra esto en el ejemplo del proyecto, ya que utiliza muchas funciones VOID.

Lo primero que se declara en el programa antes de utilizar la función setup(), son las variables globales y las librerías que se utilizan.

En las partes de variables globales entran los pines, para cambiar sus números de reconocimiento por nombres específicos, como los pines en Arduino utilizan números enteros se definen estas variables como **INT**. Entonces en esta variable primero se define "int" luego el nombre que se desea poner a esos pines y luego a que pin pertenece (número).

Para incluir alguna librería específica en el proyecto se la llama con **#include <>**, adentro se pone el nombre de la librería a utilizar, en este proyecto se utilizo únicamente la librería Servo.h.

A continuación se muestra parte del programa y se explica qué función cumple:

```
#include <Servo.h>    /*Cualquier pin DIGITAL puede generar una señal PWM para controlar servos*/
int ENB_1 = 2;
int ENA_1 = 3;
int ENA_2 = 4;
int ENB_2 = 5;    /*Habilitación de los ENABLE de los L293D, trabaja con PWM*/
```



```
int IN1_1 = 47;
int IN2_1 = 45;
int IN3_1 = 51;
int IN4_1 = 53;
int IN1_2 = 39;
int IN2_2 = 41;
int IN3_2 = 35;
int IN4_2 = 33;

/*INPUT de los L293D, funcionan junto con los ENABLE, para obtener el
sentido de giro*/

int LUZ = 25;          /*Para controlar las luces*/
int val=0;             /*Sirve para almacenar el valor que entrega el modulo APC*/

Servo camara1;
Servo camara2;
Servo brazo1;
Servo brazo2;
Servo pinza;
int cont1=40;
int cont2=90;
int cont3=0;
int cont4=25;
int cont5=50;

/*De esta manera se define que nombre utilizan cada Servo, se hace así por
la librería utilizada*/

/*Variables contadoras para cada servo, sirve para saber en qué posición
están*/

void setup(){
  Serial.begin(9600); /*Se habilita la comunicación serial a 9600baudios*/

  pinMode(ENA_1, OUTPUT);
  pinMode(ENB_1, OUTPUT);
  pinMode(ENA_2, OUTPUT);
  pinMode(ENB_2, OUTPUT);
  pinMode(IN1_1, OUTPUT);
  pinMode(IN2_1, OUTPUT);
  pinMode(IN3_1, OUTPUT);
  pinMode(IN4_1, OUTPUT);
  pinMode(IN1_2, OUTPUT);
  pinMode(IN2_2, OUTPUT);
  pinMode(IN3_2, OUTPUT);
  pinMode(IN4_2, OUTPUT);
  pinMode(LUZ, OUTPUT);

  /*Define a los pines que se utilizan como SALIDA o ENTRADA*/

  camara1.attach(12);
  camara2.attach(11);
  brazo1.attach(10);
  brazo2.attach(9);
  pinza.attach(8);

  /*De esta manera se define que pines utilizan cada Servo,
  anteriormente nombrados*/
```




```
camara1.write(cont1);
camara2.write(cont2);
brazo1.write(cont3);
brazo2.write(cont4);
pinza.write(cont5);
digitalWrite(LUZ, 0);} /*Los servos empiezan en una posición*/

/*Las luces cuando se encienda el robot están apagadas*/

void loop(){
  val = Serial.read(); /*Se lee que valor hay en el puerto Rx, y se lo almacena en val*/
  if(-1 != val){ /*Si VAL no es igual a -1 se ejecuta las otras funciones*/
    estado_luz(); /*Esta función se encarga del Encendido-Apagado de la luz*/
    adelante();
    atras();
    izquierda();
    derecha(); /*Estas funciones se encargan del movimiento de las ruedas del robot*/
    camara_vertical();
    camara_horizontal();
    brazo_rotacional();
    brazo_vertical();
    brazo_pinza();} /* Estas funciones se encargan del movimiento de los servos del robot*/

void estado_luz(){
  const char ON='G';
  const char OFF='B';
  if(val == ON){
    digitalWrite(LUZ, 1);}
  if(val == OFF){
    digitalWrite(LUZ, 0);} } /*ON es igual a G (valor en ASCII), OFF es igual a B (valor en ASCII)
luego se compara VAL con ambos, si es uno u otro se enciendo o
apaga las luces*/

void adelante(){
  const char AD='8'; /*AD es igual a 8 (valor en ASCII), luego se compara este con val*/
  if(val == AD){
    analogWriteResolution(12); /*Se ajusta la resolución del PWM a 12bits*/
    analogWrite(ENA_1, 4000);
    analogWrite(ENB_1, 4000);
    analogWrite(ENA_2, 4000);
    analogWrite(ENB_2, 4000); /*Con esto se puede ajustar la velocidad del robot, está casi
al máximo*/
    digitalWrite(IN1_1, 1);
    digitalWrite(IN2_1, 0);
    digitalWrite(IN3_1, 1);
    digitalWrite(IN4_1, 0);
    digitalWrite(IN1_2, 0);
    digitalWrite(IN2_2, 1);
    digitalWrite(IN3_2, 0);
    digitalWrite(IN4_2, 1); /*Con esto se controla el sentido de giro de las ruedas*/
    delay(50); /*Se hace un pequeño delay para frenar las ruedas*/
```

	Universidad Tecnológica Nacional - Facultad Regional Tucumán Técnicas Digitales III (FINAL)	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

```

analogWrite(ENA_1, 0);
analogWrite(ENB_1, 0);
analogWrite(ENA_2, 0);
analogWrite(ENB_2, 0);}} } /*Se pone los ENABLE a 0 para frenar rápidamente las ruedas*/

void izquierda(){ /*Es igual a la función anterior, pero se frena las ruedas izquierdas*/
  const char IZ='4';
  if(val == IZ){
    analogWriteResolution(12);
    analogWrite(ENA_1, 4000);
    analogWrite(ENB_1, 4000);
    analogWrite(ENA_2, 4000);
    analogWrite(ENB_2, 4000);
    digitalWrite(IN1_1, 1);
    digitalWrite(IN2_1, 0);
    digitalWrite(IN3_1, 1);
    digitalWrite(IN4_1, 0);

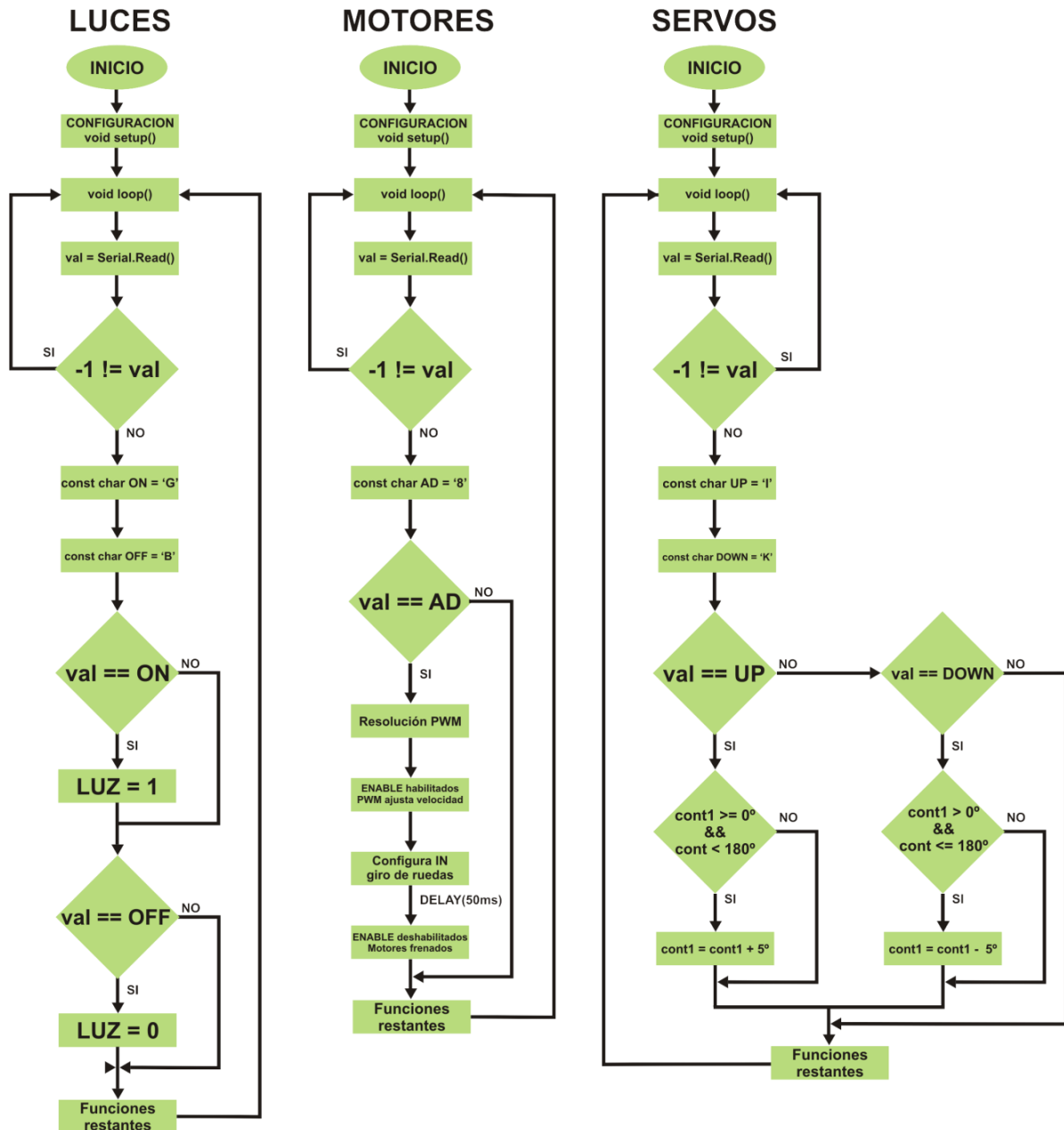
    digitalWrite(IN1_2, 1);
    digitalWrite(IN2_2, 1);
    digitalWrite(IN3_2, 1);
    digitalWrite(IN4_2, 1); } /*Se ponen todos los IN en estado alto (1) para frenar las ruedas*/
    delay(300);
    analogWrite(ENA_1, 0);
    analogWrite(ENB_1, 0);
    analogWrite(ENA_2, 0);
    analogWrite(ENB_2, 0);}}

void camara_vertical(){ /*Función para servo cámara, idéntica para los otros servos*/
  const char UP='I';
  if(val == UP){ /*UP es igual a I (valor en ASCII), luego se compara este con val*/
    if (cont1 >= 0 && cont1 < 180){ /*Si el valor de cont1 es mayor o igual que 0º, y además si es menor que 180º, aumenta el ángulo cont1 en 5º. Con esto el ángulo no aumenta más que 180º.*/
      cont1 = cont1 +5;
      camara1.write(cont1);}}

  const char DOWN='K';
  if(val == DOWN){
    if (cont1 > 0 && cont1 <= 180){ /*Si el valor de cont1 es mayor que 0º, y además si es menor o igual que 180º, disminuye el ángulo cont1 en 5º. Con esto el ángulo no disminuye más que 0º.*/
      cont1 = cont1 -5;
      camara1.write(cont1);}}}

```

Con esto se explica la programación del microcontrolador, el diagrama de flujo siguiente muestra la lógica de la programación:

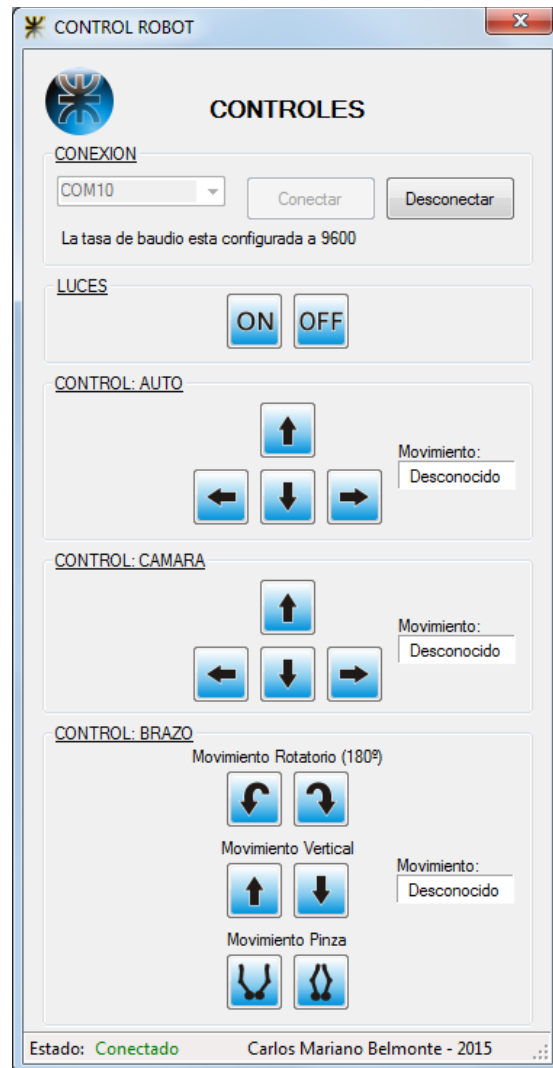


Etapas del programa de control del robot:

Ahora se explica la programación del software de control. Fue realizado en c#, una programación orientada a objetos. Lo primero que se realiza es abrir un nuevo **form** y darle las dimensiones que uno quiere, es decir que tamaño tendrá la ventana de nuestro programa. Luego se colocan los botones necesarios para nuestros programa y se cambia los nombre con los cual se identifican, también se puede modificar sus propiedades. Después se agregan **ComboBox** (lista desplegable), el cual listara todos los puertos COM que estén funcionando en la PC, y ahí podemos elegir el que utiliza el APC. También se agregan **Label** (textos) para títulos, información y/o estados. Por último

	<p align="center">Universidad Tecnológica Nacional - Facultad Regional Tucumán</p> <p align="center">Técnicas Digitales III (FINAL)</p>	
Profesora: Ing. A.M. Juárez Fernández Ing. Oscar Galvez	Tema: Auto Robótico	
Alumno: Belmonte Carlos	Fecha: Febrero/2015	

se agrega un **statusBar** (barra de estados) que indica si el APC está conectado o no con el programa.



Luego de hacer la parte visual del programa, se realiza la etapa de programación, dándole todas las funciones a los botones, listas y cuadro de textos. También se puede hacer un entorno visual mínimo e ir programando sus funciones y luego hacer el agregado de mas detalles. Como en el caso de la programación del microcontrolador, la programación de este soft de control tiene parte repetitivas, por eso se explicaran algunas ya que el resto es similar. Todo el programa se adjunta al final así se lo puede ver detalladamente. Cuando se empieza la programación de un nuevo form, este tiene una estructura básica que siempre se repite, primeramente aparecen los **using**, en el caso de este programa funcionan como librerías que permiten el correcto funcionamiento del form. El **namespace** se utiliza para declarar un ámbito que contiene un conjunto de objetos relacionados y dentro de este es donde todo el programa se realiza.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        private SerialPort Puerto;
        private string[] Puertos;
        public Form1()
        {
            InitializeComponent();
            listarPuertos();
        }

        #region STATUS
        private void defineStatus(bool status)
        {
            if (status == true)
            {
                statusLabel.Text = "Conectado";
                statusLabel.ForeColor = Color.Green;
            }
            else
            {
                statusLabel.Text = "Desconectado";
                statusLabel.ForeColor = Color.Red;
            }
        }
        #endregion

        #region LISTAR COM EN COMBOBOX
        private void listarPuertos()
        {
            Puertos = SerialPort.GetPortNames();
            if (Puertos.Length > 0)
            {
                comboBoxPuertos.Items.Clear();
                foreach (string s in SerialPort.GetPortNames())
                {
                    comboBoxPuertos.Items.Add(s);
                }
            }
        }
    }
}
```

//Librerías que aparecen por

//Librería para poder utilizar el puerto COM

//En esta aparte se escribe el programa, sus llaves se cierran al final

//Se define un nombre para el Puerto serie

//Crea una cadena con todos los puertos

//Función para listar los puertos en ComboBox

//Función Status (bool) que tiene 2 estados posibles, VERDADERO o FALSO. Sirve para cambiar el mensaje mostrado en la barra de estado.

//Funcion que lista los puertos, primero los nombra a todos y los guarda en Puertos, luego compara la cantidad de estos, si es mayor a 0 borra el comboBox. Luego en un array (foreach) carga todos los puertos.



```
defineStatus(false);
habilitar(true, true, false, false, false, false, false);
}
}
#endregion

#region HABILITACIONES
private void habilitar(bool COM, bool conectar, bool desconectar, bool automovil,
bool camara, bool brazo, bool Luz)
{
    comboBoxPuertos.Enabled = COM;
    botonConectar.Enabled = conectar;
    botonDesconectar.Enabled = desconectar;
    groupBoxAuto.Enabled = automovil;
    groupBoxCamara.Enabled = camara;
    groupBoxBrazo.Enabled = brazo;
    groupBoxLuces.Enabled = Luz;
}
#endregion

#region Boton CONECTAR
private void botonConectar_Click(object sender, EventArgs e)
{
    try
    {
        if (comboBoxPuertos.SelectedIndex != -1)
        {
            Puerto = new SerialPort();
            Puerto.PortName = comboBoxPuertos.SelectedItem.ToString();
            Puerto.BaudRate = 9600;
        }
        else
        {
            throw new Exception("Seleccione un puerto para conectar");
        }
        if (!Puerto.IsOpen)
        {
            Puerto.Open();
            defineStatus(true);
            habilitar(false, false, true, true, true, true, true);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, "NO ES POSIBLE LA CONEXION. \n" + ex.Message, "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

//Si el status es falso, todos los botones de movimiento están deshabilitados, solo el botón CONECTAR y el listado de los COM esta activo para su selección

//Función que sirve para habilitar de manera grupal todos los botones de comando del robot, si el programa se conecta a un puerto COM existente.

//Función CONECTAR: Try es una opción que si es incorrecta manda un mensaje de error.

//Si el comboBox tiene seleccionado un COM, este se lo asigna como nuevo, con 9600baudios.

//Si no se selecciona ningún puerto y se presiona el botón Conectar, aparece un mensaje de error.

//Si el puerto COM está abierto, deshabilita el comboBox y el botón CONECTAR.

//Excepción: si no se cumple Try, manda un mensaje de error



#endregion

#region BOTONES AUTO

```
private void btUpAuto_Click(object sender, EventArgs e)
{
```

```
    byte[] mBuffer = new byte[1];
    mBuffer[0] = 0x38; //Letra 8 en ascii
    Puerto.Write(mBuffer, 0, mBuffer.Length);
    labelMovAuto.Text = "Adelante...";
}
```

//Esta función se activa cuando se hace click sobre el botón.

new byte[1] indica que se manda una sola trama al presionar el botón (una sola letra). El valor de la letra se lo envía en ASCII

```
private void btDownAuto_Click(object sender, EventArgs e)...
```

```
private void btIzqAuto_Click(object sender, EventArgs e)...
```

```
private void btDerAuto_Click(object sender, EventArgs e)...
```

#endregion

//Los botones restantes se explican igual que el anterior.

#region TECLAS

```
protected override bool ProcessCmdKey(ref Message m, Keys keyData)
```

```
{
```

```
    bool blnProcess = false;
```

```
    #region TECLAS PARA AUTO
```

```
    if (keyData == Keys.Up)
```

```
    {
```

```
        blnProcess = true;
```

```
        byte[] mBuffer = new byte[1];
```

```
        mBuffer[0] = 0x38; //Letra 8 en ascii
```

```
        Puerto.Write(mBuffer, 0, mBuffer.Length);
```

```
        labelMovAuto.Text = "Adelante...";
```

```
    }
```

```
    if (keyData == Keys.Down)...
```

```
    if (keyData == Keys.Left)...
```

```
    if (keyData == Keys.Right)...
```

```
    #endregion
```

//Función para detectar cuando se pulsa una tecla. KeyData compara si su valor es igual a la tecla presionada (Keys). También se agrega una variable local blnProcess para hacer retornar al programa y volver a preguntar si se presiono una tecla o no. El proceso de envío esta información por el puerto COM es el mismo que los botones.

```
    if (blnProcess == true)
```

```
    {
```

```
        return true;
```

```
    }
```

```
    else
```

```
    {
```

```
        return base.ProcessCmdKey(ref m, keyData);
```

```
    }
```

```
}
```

```
#endregion
```

```
}
```

// If para hacer retornar al programa y detectar cuando se presiona una tecla.

El diagrama de flujo siguiente muestral la explicación de la programación:

