Universidad Austral de Chile

Facultad de Ciencias de la Ingeniería

Escuela de Ingeniería Civil Electrónica

DESARROLLO DE LA ELECTRONICA DE COMUNICACIÓN DE DATOS Y VIDEO, PARA UNA CAMARA SUBMARINA SOSTENIDA POR CABLE

Tesis para optar al título de:
Ingeniero Electrónico.

Profesor Patrocinante:
Sr. Franklin Castro Rojas.
Ingeniero Electrónico,
Licenciado en Ciencias de la Ingeniería,
Diplomado en Ciencias de la Ingeniería.

ALDER JOSEF ALEXANDER CISTERNA SANCHEZ

VALDIVIA - CHILE

2009

PROFESOR PATROCINANTE

Sr. FRANKLIN CASTRO ROJAS ----------

PROFESORES INFORMANTES

Sr. DANIEL LÜHR SIERRA ----------

Sr. MARCEL ZAPATA SILVA ----------

FECHA EXAMEN DE TITULACIÓN: *22 de junio de 2009*

Agradecimientos:

Doy agradecimientos a mis padres y hermanos por el apoyo brindado en el desarrollo de mi tesis.

Por otra parte le doy las gracias a don Marcel Zapata, profesor informante, quien me guió a cada momento.

También es importante destacar a dos compañeros de carrera con los cuales compartí la universidad y que estuvieron siempre disponibles frente a alguna consulta en particular, estos compañeros son Mauricio Zapata y Hugo Muñoz.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE GRAFICOS

RESUMEN

El presente trabajo expone el desarrollo que se realiza para construir el prototipo de la electrónica de datos y video, de una cámara submarina. Equipo necesario en empresas del rubro salmonero y empresas de investigación para dejar registro de actividades y acontecimientos ocurridos en el fondo marino. Para ello se tomó como base los requerimientos de una empresa en particular.

El desarrollo esta dividido en etapas como: la factibilidad del proyecto, la elección de componentes, el diseño e implementación de las etapas, la creación de las placas de circuito impreso, el desarrollo del programa que se cargará en el microcontrolador y el software de prueba que se instala en el PC para la captura y control de los datos.

El trabajo se enfoca principalmente a la creación del sistema electrónico y el software del microcontrolador. En cuanto al programa que se instala en el computador (envío y captura datos) sólo se desarrolla una versión de prueba que demuestra la funcionalidad del sistema.

SUMMARY

The present work exposes the development accomplished to construct prototype submarine camera electronics for data and video. Necessary equipment in companies of the Salmon industry and investigation companies to leave record of activities and events happened in the sea bed. For this it was taken the requirements of a company as a base especially.

The development is divided in stages as: the feasibility of the project, the election of components, the design and implementation of the stages, the creation of the plates of printed circuit, the development of the program that will be loaded in the microcontroller and the software of test that is installed in the PC for the acquisitions and control of the information.

The work focuses principally on the creation of the electronic system and the software of the microcontroller. As for the program that one installs in the computer (sending and it captures information) only there develops a version of test that demonstrates the functionality of the system.

INTRODUCCION

Hoy en día existe en el mercado una gran variedad de dispositivos con tecnología desarrollada en el extranjero. Nuestro país debe incorporarse al desarrollo de su propia tecnología ya sea en un principio tomando como base las ideas implementadas en otros lugares que en definitiva entregan la experiencia necesaria para avanzar en el camino del desarrollo y la investigación.

El presente trabajo describe los pasos necesarios para construir el sistema electrónico de una cámara de televisión submarina, se toma como base las necesidades de un usuario final para enfocar el proyecto.

Este trabajo cuenta con etapas que pueden ser utilizadas en otros proyectos de desarrollo.

Para llevar a cabo el proyecto se estudió las necesidades a cubrir para luego buscar información actualizada en la red y seleccionar la alternativa más viable.

OBJETIVOS GENERALES

El objetivo principal, de este trabajo, es el desarrollo de la electrónica de una cámara submarina. Este equipo en su fase final será una cámara de televisión submarina que aparte de entregar imágenes del fondo marino, podrá dar datos como orientación y la profundidad a la que se encuentra.

Empresas como la industria salmonera, acuícola y centros de investigación están interesados en equipos como éste para explorar y tener registros del fondo marino.

OBJETIVOS ESPECÍFICOS

1.- Implementar un sistema de transmisión de imagen de calidad, para ser monitoreada en la consola central de control.

2.- Diseñar e implementar el sistema de iluminación de la cámara submarina, en niveles de iluminación (0-7), de tal forma que pueda ser controlada desde la superficie.

3.- Lograr reducir el sistema electrónico para que se adapte a determinados espacios.

4.- Implementar una forma de comunicar los sensores (brújula electrónica, sensor presión) con la unidad central de control, de la unidad remota.

5.- Implementar un protocolo de comunicación entre la unidad consola y la unidad remota, para la obtención de datos y control de la iluminación.

CAPITULO I
REQUERIMIENTOS Y FACTIBILIDAD DEL PROYECTO

1.1 Requerimientos del proyecto

Según la información que se tiene, se requiere un sistema que deberá de estar compuesto por una unidad consola (ubicada en la superficie) y una unidad remota (fondo marino). Estas dos unidades deben de estar unidas por un cable que proporcione energía y comunicación.

La unidad remota, en su fase final, deberá estar compuesta por 4 cajas cilíndricas estancas, de las cuales dos de ellas contendrán los focos de iluminación, otra contendrá en su interior una brújula electrónica, un sensor de presión y toda la electrónica de control y comunicación. Por último tenemos la cuarta caja que contendrá la unidad captadora de video.

Referente a los focos de iluminación se requiere que sean 2 cajas negras alimentadas por 12V y controladas por 3 líneas lógicas de control.

En cuanto a la caja que contiene la cámara de video, esta dependerá de la unidad captadora de video y el usuario final decidirá cual colocar, sólo se requiere que se disponga de un total de 2.8 Watt máximo, en 12V, para energizarla.

Referente a la unidad que se encontrará en la superficie (unidad consola) deberá ser alimentada por 24VDC y entregar el video a través de un conector RCA (norma base). Por otra parte la información deberá ser entregada a través de un puerto de comunicación para un PC o similar.

Se requiere que las unidades estén en comunicación entre sí a través de Cable Cat. 5E flexible y que los sistemas deban de funcionar a una distancia máxima de 150 metros, utilizando la norma RS-485 para comunicación las dos unidades.

Para medir la profundidad se especifica utilizar un sensor de presión que entrega una señal de salida con protocolo 4 a 20ma.

Se requiere una brújula electrónica de tamaño reducido que entregue orientación respecto al norte e inclinación.

A continuación se presenta un esquema de los requerimientos de este proyecto:

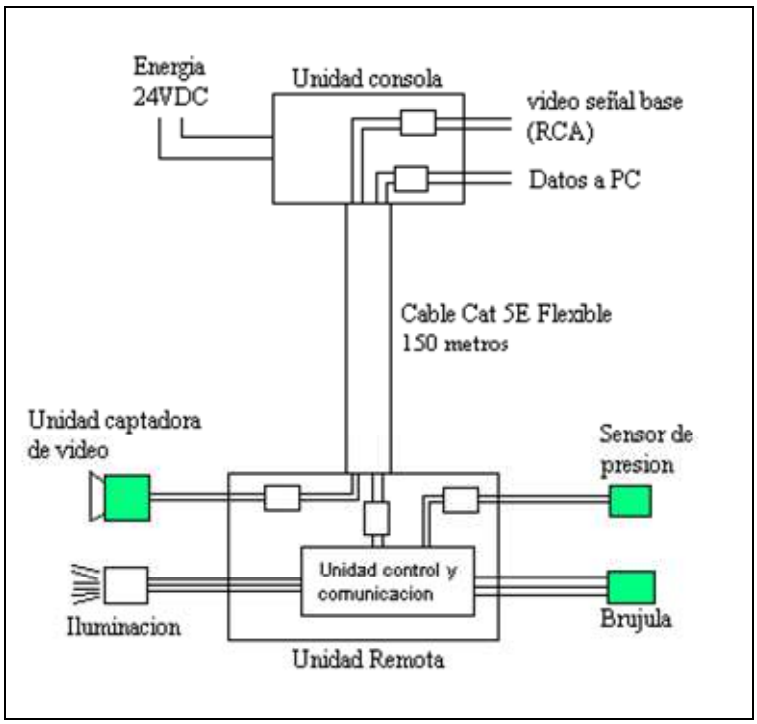


Figura Nº 1.1: Esquema requerimientos del proyecto

Los elementos en verde, del esquema, podrán ser reemplazarse por equivalente previa reprogramación del microcontrolador, si se requiere.

1.2 Pruebas de factibilidad del proyecto

Se realizan dos pruebas cruciales, una determinarán la respuesta del cable Cat.5E frente a la trasmisión de datos y la otra calcula aproximadamente la corriente máx. que se dispondrá a los 150 metros de profundidad.

1.2.1 Pruebas de transmisión RS-485 en cable Cat 5E

Se realiza un circuito básico para emisión y recepción de señal para norma RS-485, para lo cual se toma cable UTP Cat 5E rígido de 121 metros. Se transmitió a 9.600 y 57.600 baudios el resultado de las pruebas fue exitoso, por lo tanto la comunicación no debería presentar mayores problemas.

1.2.2 Prueba de resistencia eléctrica del cable

Para tener una referencia de las perdidas por caída de tensión, se tomaron 121 metros de cable Cat 5E rígido y se hizo circular corriente. Posteriormente se calculó la resistencia del cable mediante la corriente y la caída de tensión, obteniéndose los valores que se describen a continuación,

En la primera prueba se midió sólo un par, para lo cual se coloco una carga en su extremo y se midió la corriente que circulaba en el circuito, obteniéndose los siguientes valores.

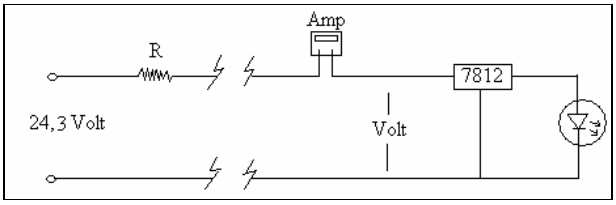


Figura Nº 1.2: Conexión realizada para pruebas

Corriente circulante= 89ma
Voltaje llegada= 22,25 volt
Caída tensión par = 2.05volt
Por lo tanto la resistencia del conductor es de

R= V/I= 2,05/89ma= 2,05/0,089=23 ohms por par a 121 metros.

Posteriormente se realizó la prueba con dos pares, donde se obtuvieron las siguientes medidas.

Corriente circulante= 96ma
Voltaje llegada= 23,25 volt
Caída tensión 2 pares = 1.05volt
Por lo tanto la resistencia del conductor es de
R= V/I= 1,05/96ma= 1,05/0,096=10,9 ohms por 2 pares a 121 metros.

Nota: La resistencia de cada par puede variar con respecto de su cercano debido a que no tienen todo el mismo trenzado y por ende varían en el largo.

Si tomamos como referencia estas medidas y calculamos la caída de tensión a 150 metros tenemos los siguientes resultados (en dos pares):

121 metros = 10.9 ohms
150 metros = X

X= (150*10,9)/ 121 = 13,51
Tomando lo anterior podemos calcular la caída de tensión que se produce en el cable al tener un consumo estimativo de 700ma.

V= R* I = 13,51 * 0,7 = 9,4

Por lo tanto, a 150 metros es necesario utilizar dos pares con lo cual contamos con aproximadamente 14,6 V para un consumo de 700ma. Si el consumo es mayor a 700ma habría que analizar las cargas en la unidad remota. De todas maneras, los cables flexibles trenzados tienen un poco menos resistencia que los rígidos y esto es punto a favor, por otro lado si se llegara a requerir mas potencia se puede cambiar el cable UTP Cat 5 por el UTP Cat 6E Flexible.

CAPITULO II
ELECCIÓN COMPONENTES PRINCIPALES Y DISEÑO PRELIMINAR

2.1 Microcontrolador central para la unidad remota.

Este sistema necesita un microcontrolador capaz de leer información entregada por una brújula electrónica, un sensor de presión, controlar la iluminación en el fondo marino y tener continua interacción con la unidad que se encontrará en la superficie. Para ello necesitamos un microcontrolador que sea capaz de realizar toda esta labor sin problemas. Por un lado tenemos los microcontrolador PIC que ha sido bastante ocupados por su amplia información en la red y por el otro lado tenemos los microcontroladores AVR (de Atmel) del cual no se encuentra mucha información pero que tiene algunas ventajas significativas con respecto a los PIC.

Sus ventajas:

En principio son 4 veces más rápidos que los PIC, ya que no dividen la velocidad del cristal por 4, aprovechando toda la velocidad de éste.

Utilizan un conjunto de instrucciones más eficientes, que utilizan menos memoria de programa por lo que generan programas mas cortos para realizar la misma labor que los PIC.

Hoy en día una gran cantidad de empresas por ejemplo; telefonía celular como Nokia y Motorota, Compaq etc. que trabajan con este tipo de microcontroladores por su alto desempeño.

En este trabajo ocuparé las ventajas de este microcontrolador bajo la plataforma de desarrollo Arduino la cual fue diseñada para trabajar con algunos modelos en particular de AVR (ATmega 8, ATmega168, ATmega328, ATmega1280) y que gracias a su entorno de programación no es necesario conocer el código nativo de los microcontroladores.

La ***tarjeta de desarrollo Arduino*** es una plataforma de desarrollo programable para realizar prototipos electrónicos que sigue la filosofía de código abierto (open source), por lo tanto el hardware y todos los detalles del diseño y su entorno se han

hecho públicos, como el esquema y circuito impreso, el entorno de programación IDE, el programa cargador (Bootloader) y los comandos de programación, los cuales están en lenguaje de programación C++.

La **tarjeta de desarrollo arduino Diecimila**, consta de las siguientes características:

- Microcontrolador *AVR ATmega168* (CPU de 8 bits, con 16Mhz de reloj).
- 16K de memoria de programa.
- 1K de RAM.
- 512Bytes de EEPROM.
- 6 conversores analógico/digital de 10bits.
- 1 Puerto serie UART, SPI (comunicación serial).
- 6 canales PWD (Pulse Width Modulation).

Todas estas características son propiedades del Microcontrolador ATmega168, además, la placa de desarrollo Arduino Diecimila tiene muchas más ventajas:

-Tiene un chip (FTDI) que se encarga de convertir el puerto serie del Atmel en un puerto serie sobre USB, con lo que tenemos conectividad USB hacia/desde un ordenador, por ejemplo, para programarlo o para enviar y recibir datos para monitorear la ejecución del programa.

- La placa nos hace accesibles los pines del micro en conectores, para poder conectar nuestros circuitos a él.

- Dispone de un circuito de alimentación para cuando no lo alimentemos por USB

- Posee un conector ICSP que de momento no utilizaremos a no ser que desprogramemos el Bootloader del micro o que queramos ganar los 2K de memoria de programa que utiliza el Bootloader.

Nota: El “Bootloader” es una pequeña pieza de código insertada en los chips (microcontrolador) que vienen programados para las placas Arduino. Es el que posibilita la descarga de código a la placa, sin necesidad de utilizar Hardware externo.

A continuación se presenta la tarjeta de desarrollo Arduino Diecimila, con la cual se puede realizar un sin número de proyectos.

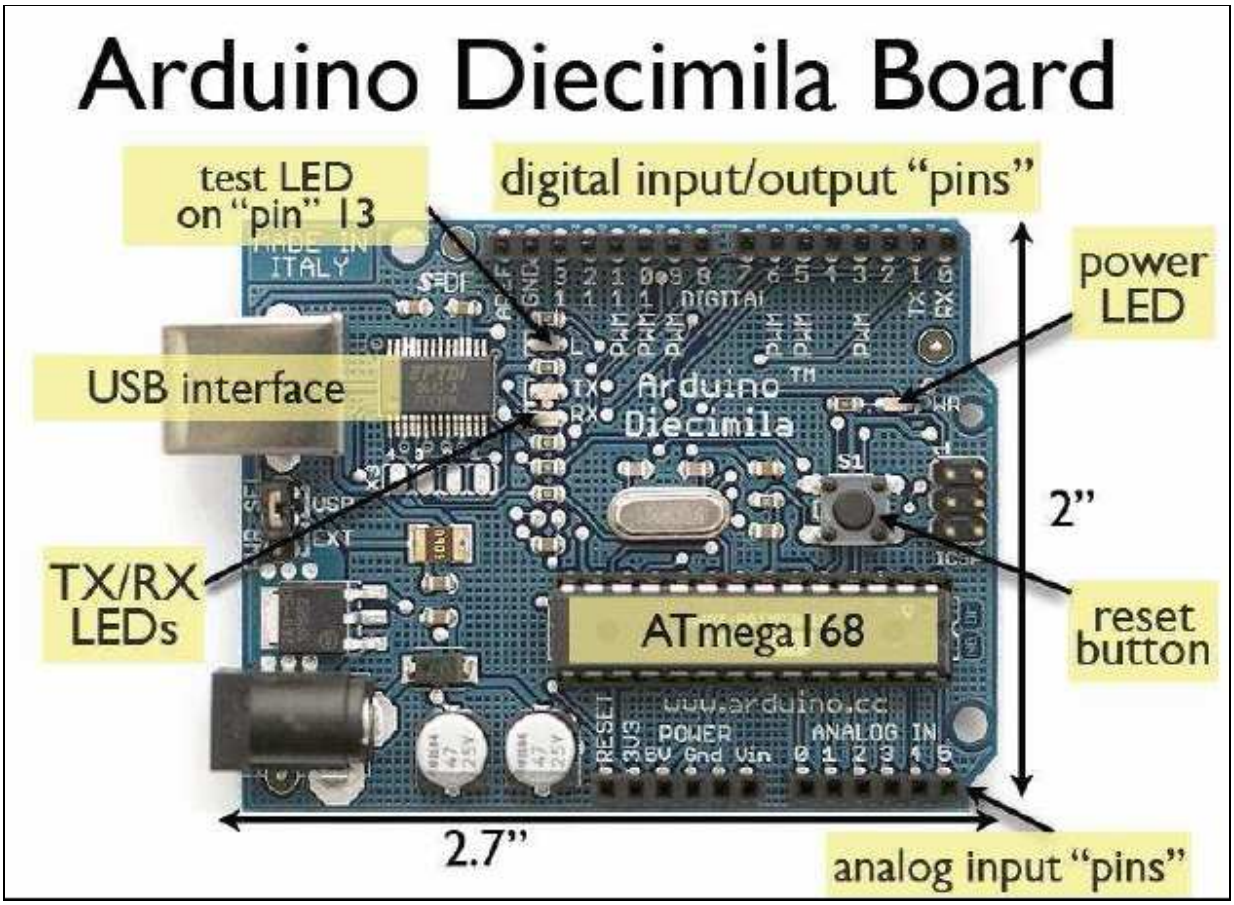


Figura Nº 2.1: Tarjeta de desarrollo Arduino Diecimila

Fuente imagen: http://electrolabo.com/arduino/Tutorial%20Arduino%2001%20-%20presentaci%f3n.pdf

A continuación se presenta el esquemático de la tarjeta de desarrollo Arduino.

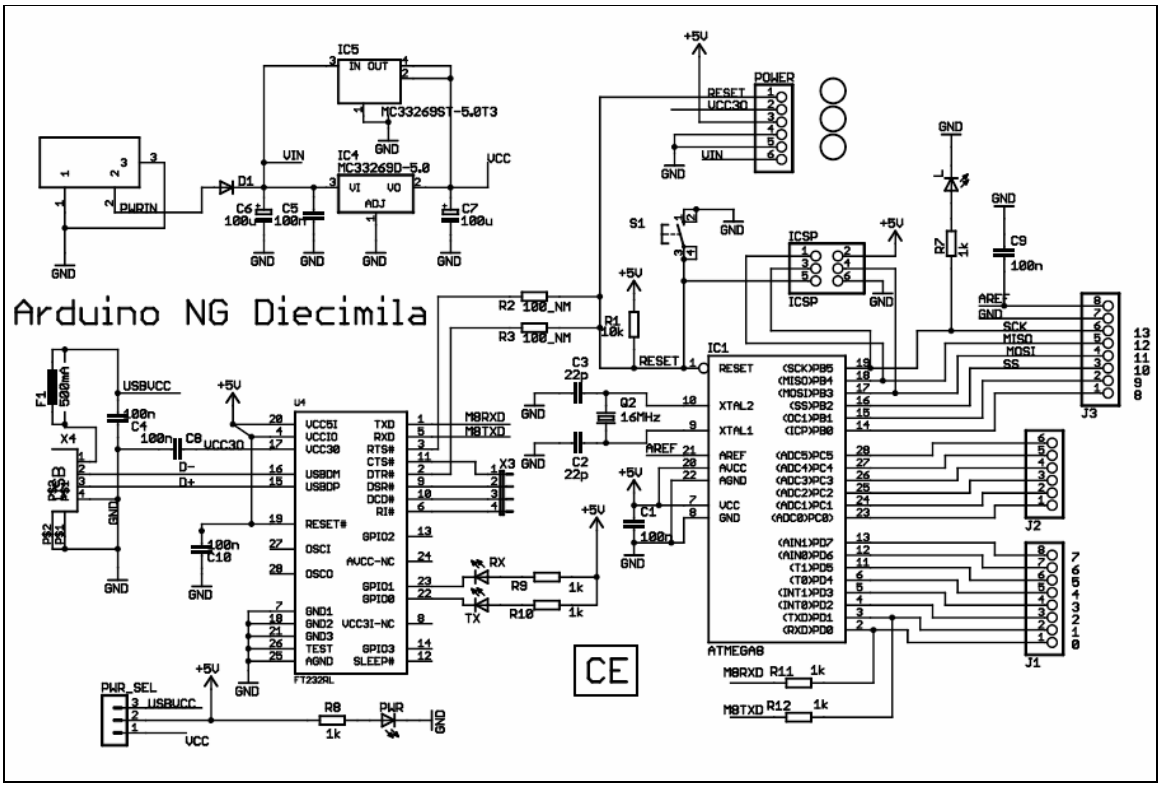


Figura Nº 2.2: Esquemático tarjeta de desarrollo Arduino Diecimila

Fuente imagen: http://www.arduino.cc/en/Main/ArduinoBoardDiecimila

En la placa final, a construir, sólo se colocaran los componentes necesarios para que el microcontrolador ATmega168 funcione, más todas la electrónica para poder comunicarse con sus dispositivos externos.

Una particularidad especial de la plataforma de desarrollo Arduino es la comunicación serial (con buffer) que viene incorporada para interactuar con cualquier otro dispositivo. Dicha comunicación se realiza a través de las salidas 0(Rxd) y 1(Txd) equivalente a los pines 2 (Rxd) y 3 (Txd) del microcontrolador, también dispone de 6 conversores análogo/digital que transforma un voltaje (0-5v) en valores entre 0 y 1111111111 que es igual a 0-1023 y viceversa. Es importante destacar que la

plataforma Arduino ha especificado de forma diferente los pines del microcontrolador, por lo tanto a la hora de programar en Arduino, es importante tener claro ésto (si luego vamos a construir un circuito donde sólo se ocupe lo necesario para hacer funcionar el sistema).

Pines Microcontrolador Puerto entrada/salida	Pines plataforma Arduino Diecimila	Pines Microcontrolador Puerto entrada/salida	Pines Plataforma Arduino Diecimila
Pin 2 = PD0(Rxd)	Pin 0 (Rxd – digital)	Pin 23= PC0 (analógico)	Pin 0 (analógico)
Pin 3 = PD1(Txd)	Pin 1 (Txd – digital)	Pin 24= PC1 (analógico)	Pin 1 (analógico)
Pin 4 = PD2	Pin 2 (digital)	Pin 25= PC2 (analógico)	Pin 2 (analógico)
Pin 5= PD3	Pin 3 (digital-PWM)	Pin 26= PC3 (analógico)	Pin 3 (analógico)
Pin 6= PD4	Pin 4 (digital)	Pin 27= PC4 (analógico)	Pin 4 (analógico)
Pin 11= PD5	Pin 5 (digital-PWM)	Pin 28= PC5 (analógico)	Pin 5 (analógico)
Pin 12= PD6	Pin 6 (digital-PWM)		
Pin 13= PD7	Pin 7 (digital)		
Pin 14= PB0	Pin 8 (digital)		
Pin 15= PB1	Pin 9 (digital-PWM)		
Pin 16= PB2	Pin 10 (digital-PWM)		
Pin 17= PB3	Pin 11 (digital-PWM)		
Pin 18= PB4	Pin 12 (digital)		
Pin 19= PB5	Pin 13 (digital)		

Tabla Nº 1.1: Equivalencias entre pines microcontrolador y Arduino Diecimila

Para realizar la programación del microcontrolador, existe el software de Arduino el cual esta basado en el uso de Processing (un lenguaje de programación) que está hecho en Java y C++. Este software permite editar, compilar y cargar el programa en el microcontrolador, el único inconveniente es que carece de depuración paso a paso. Muchas veces a este software también se le llama IDE (*Integrated Development Environment, Entorno de desarrollo integrado)* o entorno de programación.

Una vez realizado un programa en el entorno de desarrollo Arduino, hay que compilarlo con la opción **Verify/compile** que esta dentro del programa y luego para pasarlo al microcontrolador se descarga con la opción **Upload to I/O board**

A continuación se presenta el entorno de programación Arduino versión 0015, disponible en Internet.

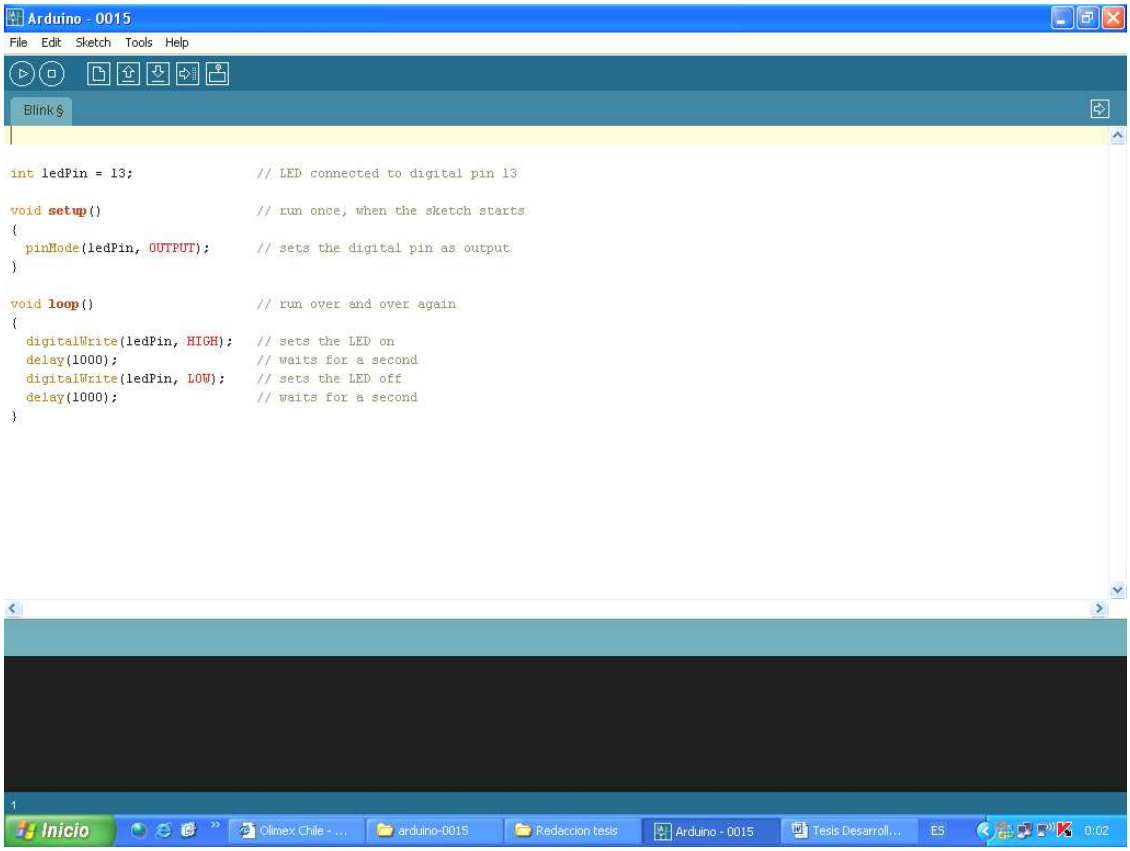


Figura Nº 2.3: Entorno de desarrollo de Arduino (IDE)

2.2 Chip para Comunicación RS-485

Para realizar la comunicación con la superficie se trabajara con la norma RS-485, para realizar esta comunicación tenemos integrados especialmente diseñado para ello, como ser el **SP485CS** o el **MAX485**.

RS-485 es una norma de comunicación serie asíncrona que utiliza dos líneas (A y B) de manera que la tensión diferencial (sin tierra absoluta) entre ambas marca el nivel lógico que se está enviando. La transferencia es semi-dúplex (al trabajar con 2 hilos solamente) ya que sólo es posible que un equipo envíe información gobernando las líneas de datos (A y B) y otro u otros equipos reciban. Está pensada para una

comunicación multipunto. El medio físico es un par de hilos trenzados entre sí para reducir el posible ruido electromagnético inducido. El estándar RS-485 permite la interconexión de hasta 32 dispositivos sobre un único par de hilos, con velocidades de hasta 10Mbits por segundo y una distancia máxima de 1200 metros. Ambas magnitudes, velocidad y distancia están ligadas entre sí, de manera que si se aumenta una, se reduce la otra.

Al tratarse de un estándar bastante abierto permite muchas y muy diferentes configuraciones y utilidades. Esta norma sólo especifica características eléctricas de una unidad, pero no especifica o recomienda ningún protocolo de datos como tampoco algún conector especifico.

Los circuitos integrados que manejan esta norma además pueden soportar "colisiones", es decir que más de un circuito transmisor esté emitiendo.

Al transmitir en modo diferencial, si el terminal A está a una tensión superior a B (con un valor diferencial superior a 0,2V) se estará recibiendo un "1" y en caso contrario (tensión de B superior en más de 0,2V a A) se interpreta un "0"

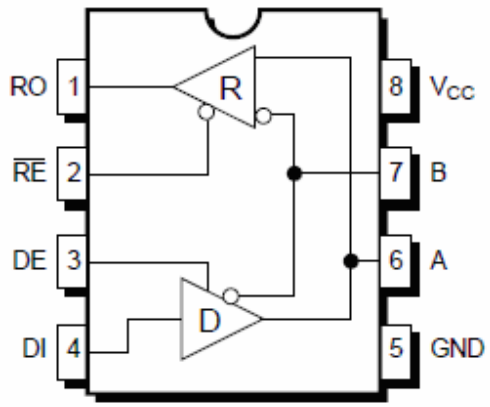


Figura N° 2.4: CI. Max 485

Pin	Nombre	Descripción
1	RO	Receiver Output
2	RE	Receiver Output Enable Active LOW.
3	DE	Driver Output Enable Active HIGH.
4	DI	Driver Input
5	GND	Ground Connection.
6	A	Driver Output/Receiver Input Non-inverting.
7	B	Driver Output/Receiver Input Inverting
8	Vcc	Positive Supply 4.75V<Vcc< 5.25V

Tabla N°1.2: Descripción pines CI. MAX485

Transmit Function Truth Table

INPUTS				OUTPUTS	
$\overline{RE}$	DE	DI	LINE CONDITION	B	A
X	1	1	No Fault	0	1
X	1	0	No Fault	1	0
X	0	X	X	Z	Z
X	1	X	Fault	Z	Z

Receive Function Truth Table

INPUTS			OUTPUTS
$\overline{RE}$	DE	A - B	R
0	0	+0.2V	1
0	0	-0.2V	0
0	0	Inputs Open	1
1	0	X	Z

Tabla Nº 1.3: Parámetros CI. MAX485

Al trabajar con un integrado MAX485 en modo half- duplex la transmisión debe de ser compartida, en un momento uno envía y luego este escucha.

Para realizar una transmisión a través de estos integrados necesitamos controlar el terminal DE (habilita envío) y RE (habilita recepción), como presentan una lógica opuesta se puede unir para ser controlados por una línea solamente. Al transmitir información hay que colocar el pin 2 y el 3 en nivel alto, de esta forma se habilita el integrado para enviar información por el pin 4 y se bloquea para recibir información por el pin 1, en caso contrario, para recibir información hay que colocar el pin 2 y el 3 en nivel bajo, en este caso se bloquea el envío de información por el pin 4 y se activa la recepción de información por el pin1.

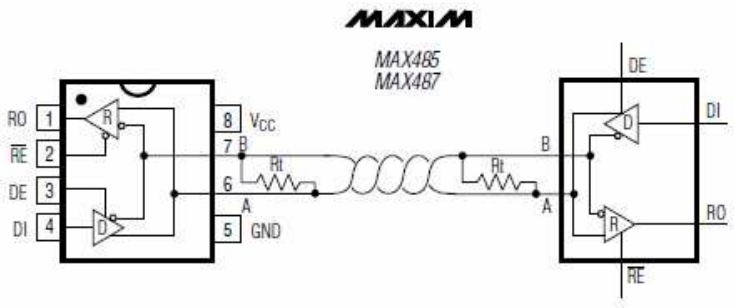


Figura Nº 2.5: Esquema de conexión entre MAX485

Muchos enlaces con RS-485 trabajan con una resistencia de 120 ohms (Rt) a través de las líneas A y B en cada extremo de la línea, para proveer a los dispositivos de una adaptación de impedancia al medio de transmisión.

2.3 Medidor de profundidad

Para medir la profundidad a la que opera un dispositivo se utiliza la presión del agua, en su relación más simple 1m de profundidad es igual a 0.098bar de presión, a los 100m la presión es de 9.8bar. En las profundidades del mundo real, la relación de presión no es constante, depende de varias situaciones como temperatura del agua, la salinidad, su densidad etc. Pero para la mayoría de las aplicaciones se puede simplemente trabajar con la relación: 100m es igual a 9.8bar.

Como nuestro sistema debe de trabajar a una distancia máxima de 150 metros de profundidad podemos trabajar con un sensor de presión de 0 a 15 Bar, pero todo sensor pierde precisión a través del tiempo, por lo cual se opta por sobredimensionar este dispositivo para tener una vida útil más larga sin tener que calibrarlo. Por lo tanto en este trabajo se ha optado trabajar con un sensor de presión de 0 a 25 bar, el cual trabajará como máximo a un 58% de su capacidad.

En el mercado existen varios fabricantes de sensores de presión, los hay de todos los tamaños hasta sensores electrónicos que tienen rangos de presión limitado y son sensibles a ambientes agresivos como es la sal del mar, para más información visitar sitio Web. http://www.gmelectronica.com.ar/catalogo/pag45.html

En Santiago de Chile tenemos proveedores como Veto http://www.veto.cl/ y Vignola http://www.vignola.cl/ empresas que se dedican a comercializar productos de automatización.

Transmisor de presión a utilizar.

- 0 - 25 BAR.
- 1% precisión, G1/4ª.
- Salida 4-20mA.
- Alimentación: 10 - 30 VDC .

Figura Nº 2.6: Sensor presión

En términos generales, el sensor de presión (de 0 a 25 bar.) responde como una fuente de corriente que entrega 4(mA) para una presión de 0 bar y 20(mA) para una presión de 25 bar. Para hacer funcionar el sensor de presión es necesario incorporarle una fuente de voltaje en serie al dispositivo, de esta forma el sensor se comportará como una resistencia variable dependiendo de la presión.

Por un tema de seguridad en los sistemas de 4-20ma se reservan el “0” (mA) para identificar un problema en el lazo de corriente.

Estos sistemas son muy utilizados en procesos industriales, porque las distancias no afectan a la corriente del lazo y, por ende, la información enviada, el sistema se adaptara a la resistencia que le proporcionaría el cable de transmisión.

2.4 Brújula electrónica

Son detectores de inducción magnética para leer electrónicamente el campo magnético de la tierra (es llamado también compás digital) como los tradicionales también se ven afectados por el magnetismo cercano. El campo magnético de la tierra es aproximadamente 0.6 gauss en un ambiente al aire libre, y tiene una dirección del polo Sur magnético al polo Norte magnético. En el ecuador, la dirección del campo magnético es completamente horizontal, pero en lugares más cercanos al los hemisferios Norte o Sur el campo magnético apuntará parcialmente hacia abajo (éste inconveniente es corregido electrónicamente por estas brújulas).

Las brújulas electrónicas definen sus medidas en el sistema de coordenadas esféricas donde los ejes se definen como Giro (**Roll**) que corresponde a la rotación en el plano X, cabeceo o elevación (**Pitch**) que corresponde a la rotación en el plano Y, por último alabeo (**Yaw o Heading**) que corresponde a la rotación en el plano Z.

Este sistema de ejes es muy utilizado en sistemas aeroespaciales como también en la navegación. La definición de cada uno de los ejes viene dada a continuación tomando como ejemplo un avión:

Roll = Rotación respecto al eje horizontal punta-cola del avión.

Pitch = Inclinación del morro del avión, o rotación respecto al eje horizontal ala-ala.

Yaw o Heading = Rotación respecto de un eje vertical.

Figura Nº 2.7: Representación definiciones brújula electrónica

Todas las brújulas electrónicas requieren ser calibradas para que entreguen una correcta medición, esto se debe a que la composición de materiales ferrosos de la Tierra varía dependiendo del lugar geográfico donde se encuentre. Para realizar dicha calibración es necesario tener una brújula tradicional a mano y apuntar ambas hacia el norte terrestre, generalmente las brújulas electrónicas, en su proceso de calibración mediante software, solicitan que uno rote la brújula 90 º hasta dar una vuelta completa, con esto la brújula realiza un cálculo y un ajuste a las condiciones del lugar.

En cuanto a brújulas electrónicas, en el mercado actual se encuentran una gran variedad que varían sus valores de acuerdo a la precisión, tamaño y cantidad de prestaciones. Existen algunas que sólo indican el Norte y otras que además indican lo anterior más roll, pich, temperatura, aceleración vectorial e incluso algunas pueden interactuar con algún sensor de presión.

A continuación se presentan algunas brújulas para su apreciación.

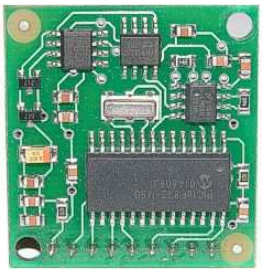


Figura Nº 2.8: Brújula digital CMPS03
Link: http://www.superrobotica.com/S320160.htm

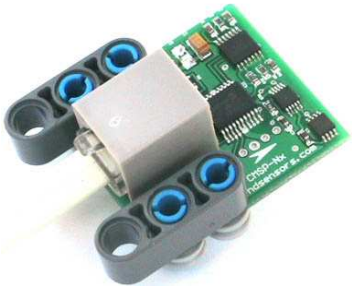


Figura Nº 2.9: Brújula digital Mindsensors
Link:
http://www.mindsensors.com/index.php?module=pagemaster&PAGE_user_op=view_page&PAGE_id=56

Figura Nº 2.10: Brújula digital Honeywell con el chip HMC6352
Link:
http://www.olimex.cl/product_info.php?products_id=283

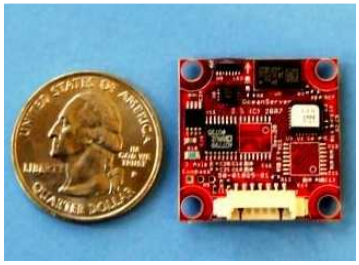


Figura Nº 2.11: Brújula digital Ocean-Server.
Link:
http://www.ocean-server.com/compass.html

Dependiendo el tipo de brújula a utilizar depende la programación para obtener los datos de ésta. En este proyecto se trabajará con la brújula de Ocean-Server, características técnicas en Capitulo “V” (Pág. 47)

2.5 Unidad capturadora de video

Como se menciona al principio, la unidad captadora de video puede variar dependiendo de la calidad de imagen que se quiera obtener.

En este proyecto se contempla colocar una unidad que no consuma más de 2.8Watt (240ma en 12V), pero para temas de probar la funcionalidad del sistema, se utilizara cámara video de las más simples existente en el mercado

En cuanto a cámaras de video uno puede encontrar un sin número de variedades que se podrían clasificar de acuerdo a su resolución, tamaño o consumo.

Hoy en día se fabrican cámaras de video con tecnologías CCD y CMOS. Las cámaras con sensores CMOS tienden a ser más económicos que los CCD, pero los sensores CCD son usados en cámaras que tienden a tener una mejor calidad y una gran sensibilidad a la luz.

Cuando se selecciona una cámara, es importante saber que tipo de sensor tiene, pero no debe ser el único factor a tomar en cuenta para su selección, ya que también es importante su resolución, el barrido de líneas, tamaño de la cámara, el tipo de trabajo que se realizará, etc.

Cámara color

- **Marca:** Tcom
- **Origen:** RPC
- **Modelo:** 81S303D
- **Elemento de imagen:** Lente pre-enfocado.
- **Sensor** CMOS de 8.5 mm
- **Resolución:** 380 líneas TV
- **Iluminación mínima:** 1.5Lux
- **Ángulo:** 6.0 mm/52º
- **Alimentación:** 9-12 Volt
- **Consumo:** 80 m A
- **Dimensiones:** 7x5 cm.

Figura Nº 2.12: Cámara de video para pruebas.

2.6 Transmisión del video en forma diferencial

Para transmitir el video en forma diferencial disponemos de dos tecnologías, una con circuitos integrados y la otra por baluns.

Transmisión de video por circuitos integrados.

Por un lado tenemos circuitos integrados especialmente diseñados para transmitir video por par trenzado, uno de los fabricantes de dichos circuitos es la empresa Maxim.

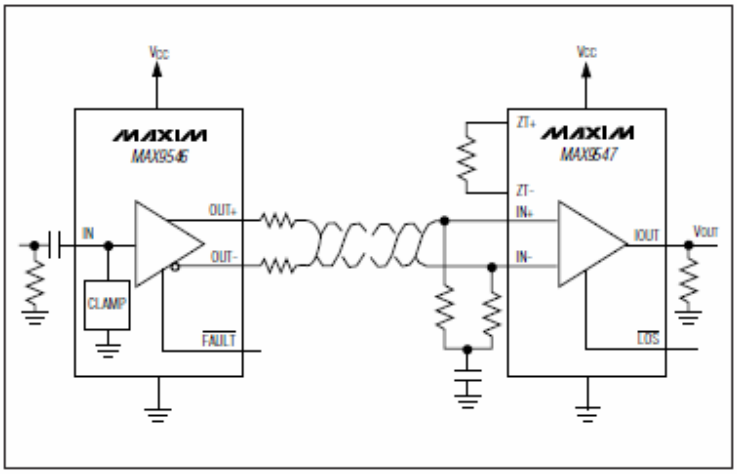


Figura Nº 2.13: Esquema conexión Chips MAX9546 y MAX 9547

Los Baluns

El termino *"balún"* proviene del inglés y significa *"Balanced - Unbalanced"*. Es generalmente un tipo especial de transformador que se conecta a una salida desbalanceada como la de una cámara y los otros dos extremos se conectan a un par trenzado, estos dos conductores (par trenzado) se dicen que están balanceados respecto de tierra. Siempre es necesario un segundo balún para volver a convertir a la entrada desbalanceada del monitor. Los balunes pasivos no necesitan fuente de energía externa y son bilaterales. Pueden ser transmisores o receptores.

El principio de funcionamiento de los pares trenzados o líneas balanceadas es el siguiente: toda interferencia que llegue a ambos conductores a la vez se cancelará

debido a que el sistema admite solo señales en modo diferencial (distinta polaridad en cada conductor del par) ya que están balanceados respecto de masa. Lo mismo sucede cuando se emiten señales. El campo de un conductor será igual pero opuesto al del otro conductor y se producirá un efecto de cancelación impidiendo la emisión y por lo tanto eliminando las pérdidas.

El balún se debe conectar al par trenzado siguiendo la polaridad marcada. Es indispensable hacer esto ya que en caso contrario la imagen resultará completamente distorsionada.

Se conecta uno en cada extremo con lo cual se habrá pasado de salida desbalanceada *(cámara)* a balanceada *(par)* y nuevamente de balanceada *(par)* a desbalanceada *(monitor)*. Al utilizar un solo par trenzado es posible transmitir video a una distancia máxima de alrededor de 600 metros para imagen color y 800 metros para imagen blanco y negro.

Tipo	VUTP-800TL	VUTP-800RJ
Cable UTP	Cat. 5. Impedancia 100 Ω a 1 MHz	
Cable BNC	RG 59. Impedancia 75 Ω a 1MHZ Máximo 6m de cable	
Distancia máxima	Color: 600 m. B/N: 800 m.	
Tensión de entrada	Máximo 1 Vp.p.	
Atenuación	2 dB a 8 MHz	
Ancho de banda	0 Hz ~ 8 MHz	
Sistema de conexionado UTP	Conexionado rápido sin herramientas	Conector RG59 hembra
Dimensiones	17,0 (A) x 17,0 (H) x 57,5 (L)	
Condiciones de trabajo	Temperatura: 0°C ~ 55°C Humedad relativa: 5% ~ 95%	

VUTP-800TL

VUTP-800RJ

Figura Nº 2.14: El balun y sus Características técnicas

Fuente: http://www.circontrol.com/appl/botiga/img/DS_550227.pdf
http://www.rnds.com.ar/articulos/030/RNDS_134W.pdf

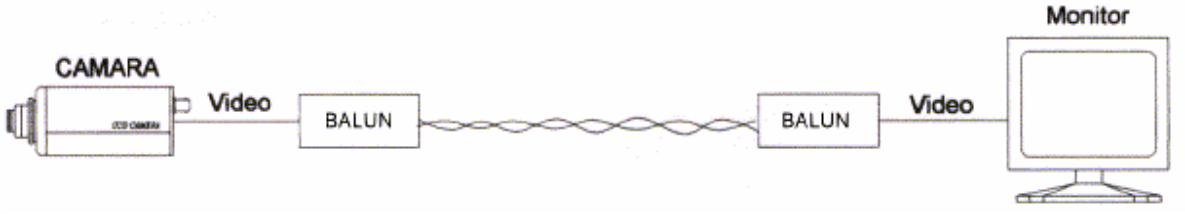


Figura Nº 2.15: Conexionado de los baluns

El principal parámetro de calidad de un balún será la medida de cuan cerca esté de un balance perfecto.

Hay una medida de cuan bueno es el balance. Se llama “Relación de Rechazo de Modo Común” (en inglés es CMRR – Common Mode Reject Ratio) y para ello se utiliza un generador de RF de 10 KHz a 8 MHz y un osciloscopio de 2 canales.

En el lado balanceado se juntan ambos conductores y se aplica una señal fuerte Vcm (ej. 2 Volts) entre los conductores unidos y masa. Esta será una señal de “modo común”. Si ahora medimos la señal sobre el lado desbalanceado, cargado con los 75 Ohms de impedancia característica, tendremos un voltaje Vdm. Si el balance fuera perfecto este voltaje debería ser cero.

Definimos el rechazo de modo común del balún como:

CMRR= 20 x log (Vmd/Vmc)

De las dos tecnologías mencionadas se opto por la segunda opción “los balun” si esta forma de conversión no cumple con los esperado se utilizaran los circuitos integrados MAXIM razón por la cual se desarrolla esta etapa separada y se le hará llegar energía para su fácil reemplazo.

2.7 Esquema preliminar sección Consola y Remota

De acuerdo a los datos obtenidos hasta este momento, se realiza un esquema simple que nos dará una idea de como van ha estar formada las distintas secciones.

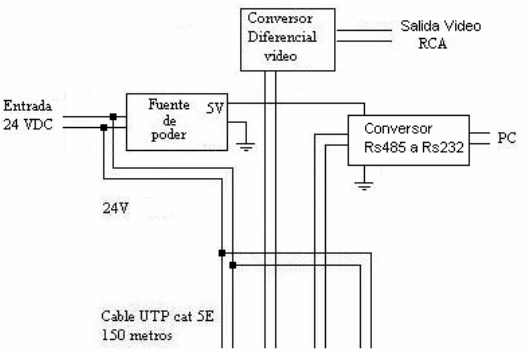


Figura Nº 2.16: Esquema sección consola

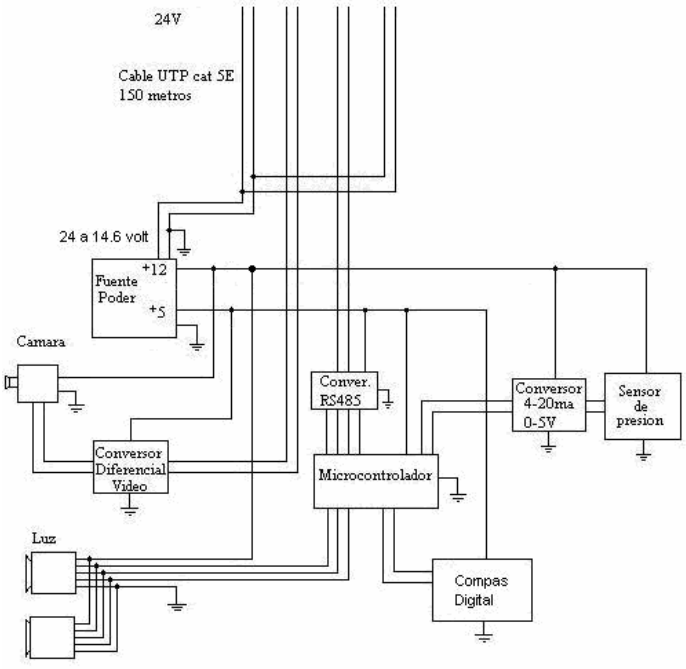


Figura Nº 2.17: Esquema sección remota

CAPITULO III
DESARROLLO DE LA UNIDAD REMOTA

La unidad remota esta compuesta por 3 módulos electrónicos independientes a construir que son:

- Unidad central de control y comunicación.
- Conversor de video.
- Sistema de Iluminación.

3.1 Unidad central de control y comunicación

3.1.1 Diseño de unidad central de control y comunicación

En lo que respecta a la unidad central de control y comunicación estará constituída por las partes que se mencionan a continuación:

3.1.1.1 La unidad central de control.

Para implementar la unidad central de control, como se menciono anteriormente, estará formada por el microcontrolador ATmega168, bajo el entorno de desarrollo Arduino Diecimila. A continuación se presenta los pines del microcontrolador en cuestión:

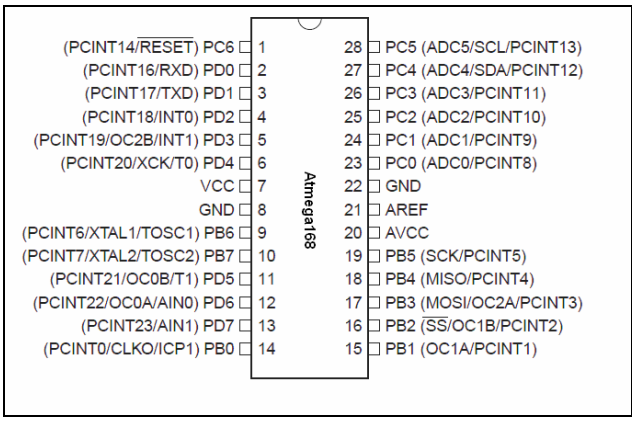


Figura Nº 3.1: Microcontrolador ATmega168

De la tarjeta de desarrollo Arduino Diecimila sólo se utilizará el microcontrolador ATmega168, los condensadores de 22pf, el cristal de 16Mhz, las resistencia de 10Komhs (para mantener en estado alto el Reset), un condensador de 100nf para eliminar ruido, el led indicador de actividad (pin 19—"salida13") y el led indicador de energía con sus respectivas resistencias de 1KΩ.

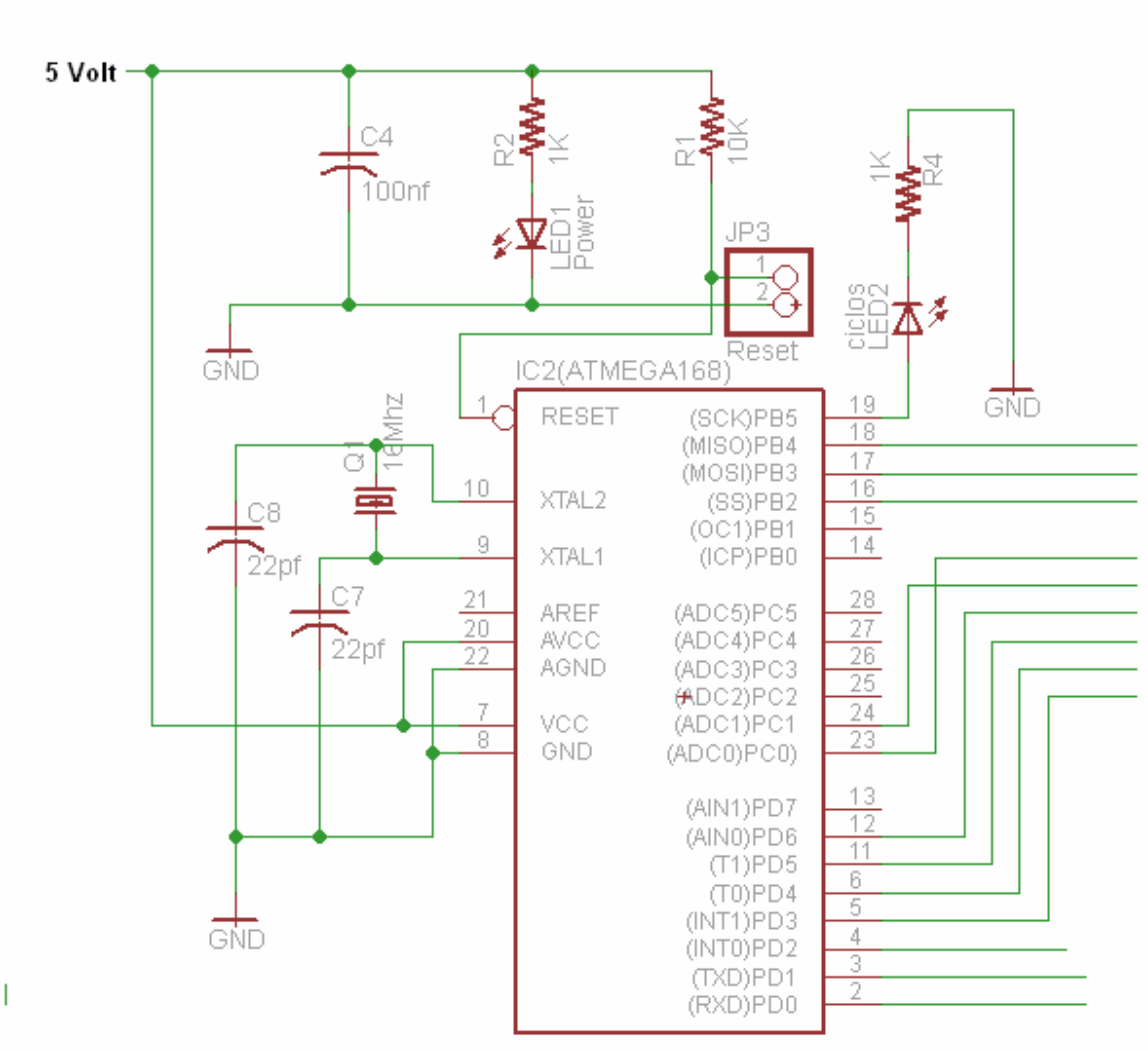


Figura Nº 3.2: Microcontrolador central

Como se mencionó anteriormente, existe una equivalencia entre los pines del microcontrolador y el entorno de programación Arduino. A continuación se detalla el uso de cada pin a utilizar.

Pines Microcontrolador Puertos entrada/salida	Pines plataforma Arduino Diecimila	Usos
Pin 2 = PD0(Rxd)	Pin 0 (Rxd)	Recepción Rs485
Pin 3 = PD1(Txd)	Pin 1 (Txd)	Transmisión Rs485
Pin 4 = PD2	Pin 2 (digital)	Habilitar Tx o Rx en Rs485
Pin 5= PD3	Pin 3 (digital-PWM)	Comunicación, Tx a Compás (Rx de compás)
Pin 6= PD4	Pin 4 (digital)	Comunicación, Rx a Compás (Tx de compás)
Pin 11= PD5	Pin 5 (digital-PWM)	Pin digital Libre
Pin 12= PD6	Pin 6 (digital-PWM)	Pin digital Libre
Pin 23= PC0 (analógico)	Pin 0 (analógico)	Entrada analógica sensor presión
Pin 24= PC1 (analógico)	Pin 1 (analógico)	Pin analógico libre
Pin 16= PB2	Pin 10 (digital-PWM)	Salida control luces “a”
Pin 17= PB3	Pin 11 (digital-PWM)	Salida control luces “b”
Pin 18= PB4	Pin 12 (digital)	Salida control luces “c”
Pin 19= PB5	Pin 13 (digital)	Salida led indicador procesos

Tabla Nº 3.1: Pines microcontrolador a utilizar

Como se puede apreciar se han dejado disponible 2 salidas digitales-PWM y una salida analógica, para futuras aplicaciones.

3.1.1.2 Conversor TTL a RS485.

Para tener comunicación con la superficie se utilizan los pines 2 y 3 del microcontrolador, los cuales entregan y reciben información en niveles binarios (0-5V). Para convertir estos niveles a diferenciales se utilizan los circuitos integrados que trabajan con ésta norma, como son el MAX485, SP485CS o similar. Es importante destacar que estos CI necesitan una resistencia de 120Ω a la salida de acuerdo a las especificaciones técnicas del fabricante. Para habilitar estos CI. como emisor o receptor se ha destinado el pin 4 del microcontrolador, por lo tanto, la habilitación será controlada por software.

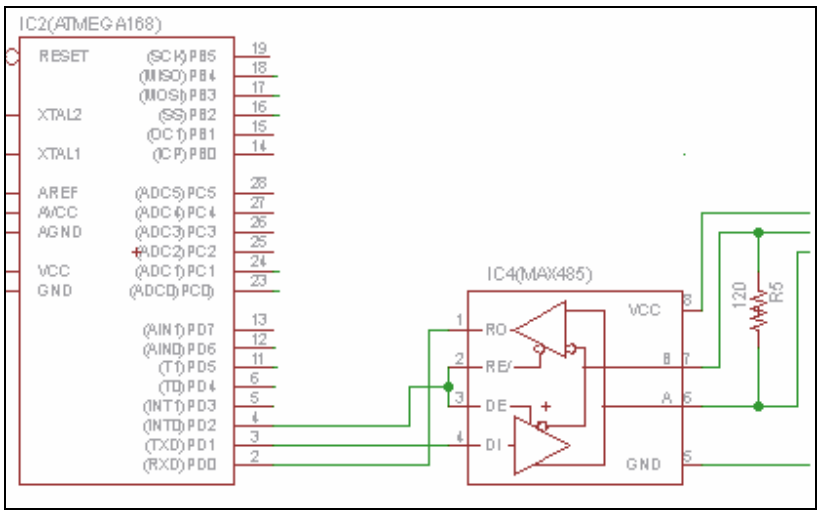


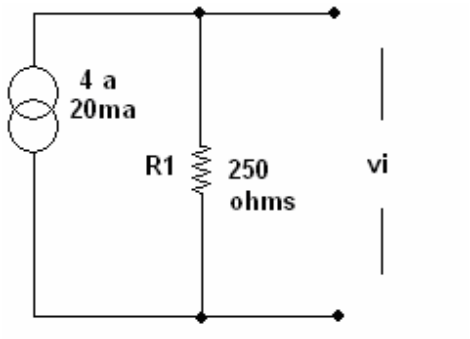
Figura Nº 3.3: Esquema comunicación, norma RS-485

Como se aprecia el pin 4 del microcontrolador es el que controla la habilitación del integrado Max485 para el envío o recepción de información.

3.1.1.3 Diseño del conversor de 4-20ma a 1-5v

Para realizar el sistema de medición de profundidad se utilizará un sensor industrial de 4-20ma el cual tendrá un rango de 0 a 25 Bar. De acuerdo a la información encontrada en Internet, existen diversas formas de convertir 4-20ma a 0-5V, de todos los métodos encontrado se seleccionó el mas simple por un tema de espacio y voltajes disponibles en nuestra fuente de poder.

El esquema más simple encontrado en la red, convierte los 4-20ma a 1-5V y es el que se muestra a continuación.



En este conversor la caída de tensión producida en R1 es igual a 1 V al circular 4ma y de 5V al circular una corriente de 20ma.
Fuente información:
http://aquaticus.info/pressure_sensor

Figura Nº 3.4: Conversor de corriente a voltaje

Por un tema de protección se le incorporó un diodo Zener de 5.1 volt -1Watt más un fusible de 150ma, de esta forma, ante eventuales desperfectos del sensor de presión nuestro microcontrolador no se ve afectado por voltajes superiores a 5,1V.

En los casos donde el sensor de presión no tiene pines para energizarlo, y el receptor no le entrega energía, hay que colocar una fuente en serie para que éste sea capaz de producir el lazo de corriente de 4-20ma.

Este tipo de sensores generalmente trabajan en un rango de voltaje de 10 hasta 24 volt, por lo que una fuente de 12 Voltios cumple con lo requerido.

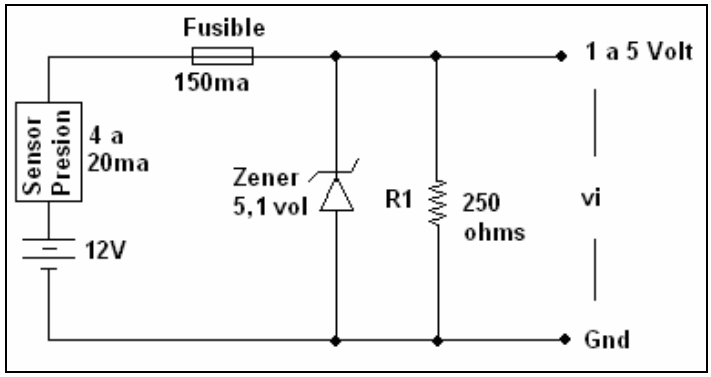
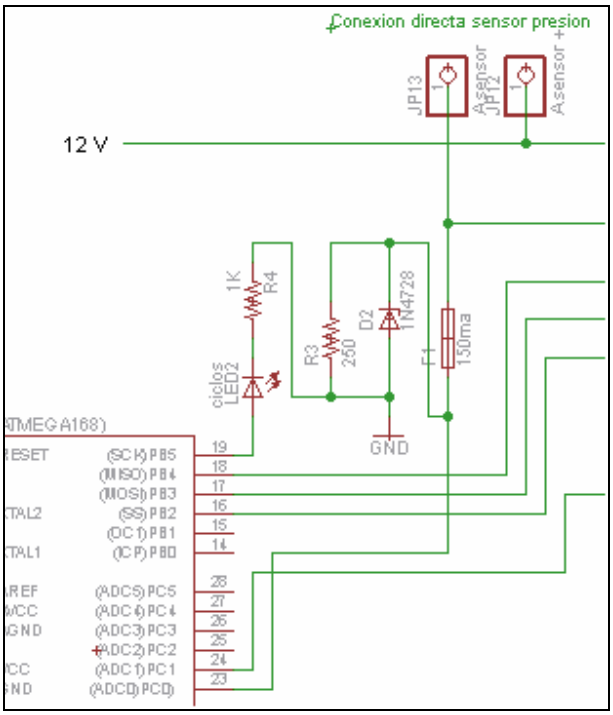


Figura Nº 3.5: Conversor de corriente a voltaje con protección

El conversor análogo digital de la plataforma Arduino, trabaja con una resolución de 10 Bit, como se está midiendo un voltaje que irá entre 1 y 5 volt, se debe realizar un ajuste por software para obtener la conversión a metros de profundidad.



Para realizar la medición se ha destinado el pin 23 del microcontrolador ATmega168, correspondiente a la salida 0 analógica de arduino.

Figura Nº 3.6: Conversor unido a microcontrolador ATmega168

3.1.1.4 Diseño de la fuente de poder para la unidad remota

En este proyecto tenemos un voltaje de 24 voltios en la unidad consola, pero producto de la caída de tensión que se produce en el cable disponemos de un voltaje, en la unidad remota, que va desde los 24 voltios hasta los 14.6 voltios (al tener una carga de no más de 700ma), por lo tanto debemos tomar estos parámetros en cuenta a la hora de diseñar nuestra fuente de poder.

En primera instancia se pensó en una fuente (switch), pero en las primeras pruebas se apreció que este tipo de fuente produce bastante ruido y tratar de estabilizar dicho ruido implicaría incorporar más electrónica quitando espacio, por lo que se opta por los reguladores convencionales 7805 y 7812. El tema del ruido es fundamental en este proyecto ya que cualquier interferencia de este tipo afectaría a los datos enviados y a la calidad de imagen transmitida a la superficie.

Como se esta trabajando con corriente continua no hacen falta condensadores de gran capacidad. En temas de consumo, al tener la configuración que se presenta

más abajo, cada 7812 queda trabajando al 30 % de su capacidad con lo cual también se distribuye la disipación de temperatura. Un regulador 7812 exclusivo para la iluminación y el otro para alimentar el compás, microcontrolador, sensor de presión y cámara de video. Se han incorporaros condensadores cerámicos de 100uf para absorber el posible ruido por las variaciones de corriente absorbidas por los componentes. Para el regulador de voltaje que alimenta a los focos se encontró innecesario incorporar condensadores cerámicos, ya que esta etapa no produce grandes perturbaciones y tampoco le afectan las mismas.

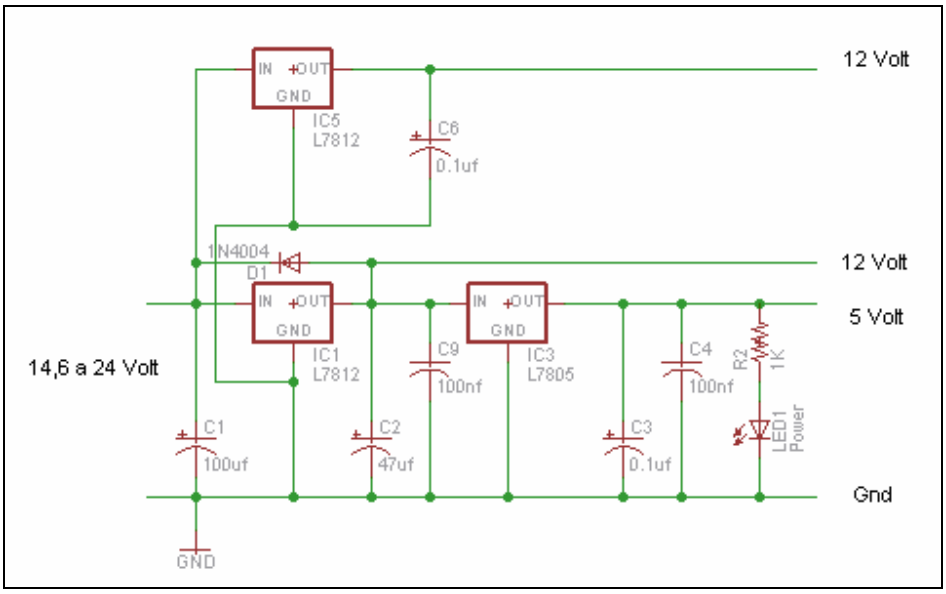


Figura Nº 3.7: Esquema fuente de poder

En circuito anterior se incorporo un diodo común en uno de los reguladores, esto se realizó tomando como base el datasheet del regulador L7412CV, donde indica que debe de colocarse un condensador no mayor a 0.1uf a la salida del regulador y es requiere colocar uno de mayor capacidad hay que incorporar un diodo común de protección para descargar el condensador ante un posible cortocircuito de la entrada. A la hora de seleccionar los condensadores electrolíticos es importante tener en cuenta el voltaje máximo a los cuales van ha estar sometidos, en este caso se seleccionaron condensadores mayores a 25 V a la entrada y de 16 V en la face intermedia (pasado los reguladores 12V), y de 10V pasado el regulador de 5V.

3.1.2 Esquema Unidad de control y comunicación.

A continuación se presenta el esquema de la unidad remota unida con todas las partes mencionadas anteriormente. El paso siguiente es pasar esta electrónica a una placa de circuito impreso.

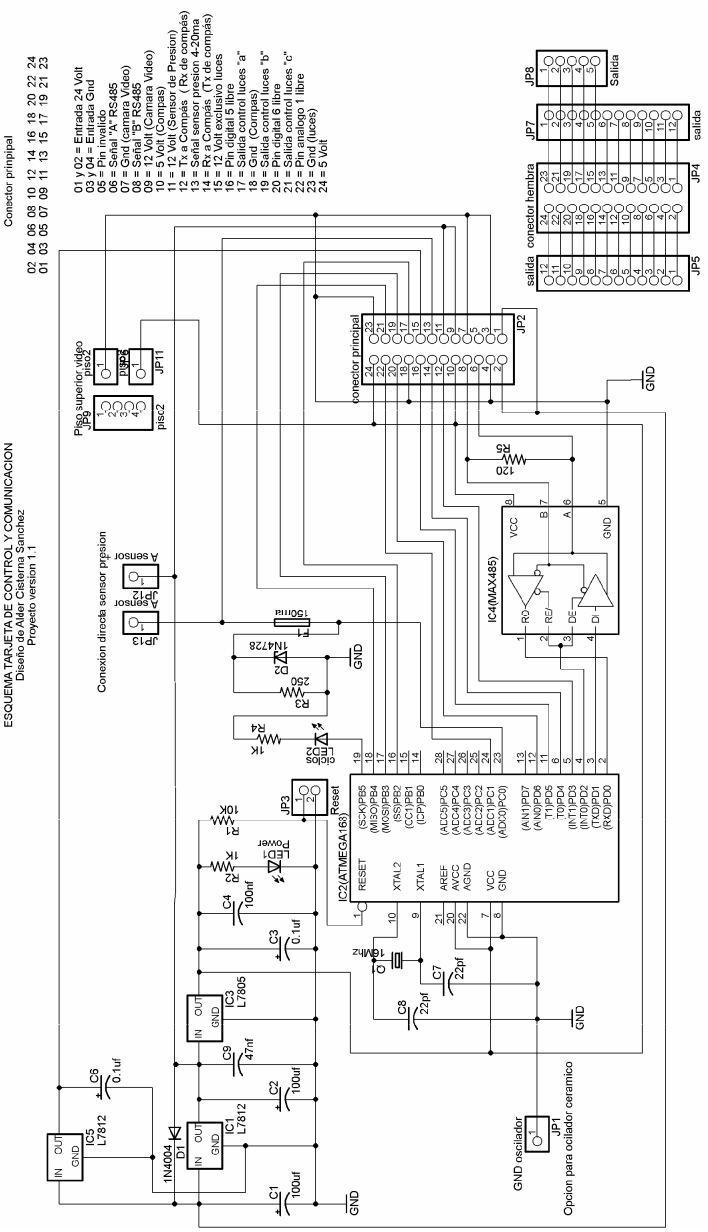


Figura Nº 3.8: Esquema unidad central de control y comunicación

Cabe mencionar que para llegar a realizar el diseño mostrado anteriormente, se debió armar algunas partes en protoboard y conectarlas a la **Tarjeta de Desarrollo Arduino Diecimila,** una vez que se tuvo claro que el sistema funcionaba se pasó a la etapa de construcción. En el primer prototipo construido se encontró que el regulador 7812 generaba mucha temperatura y no había suficiente espacio para un disipador, por lo cual se opto por separar las cargar en dos reguladores 7812 dividiendo también la temperatura generada, con esto se soluciona los posibles problemas de falla del regulador por exceso de temperatura.

3.1.3 Distribución de componentes y fabricación placa circuito impreso.

En el esquema mostrado a continuación se da a conocer la distribución realizada, para ello se utilizo el programa Eagle versión 5.3.0 en su versión gratuita que está limitada a diseñar placas de máximo de 100x80mm, este programa se puede descargar sin costo en el sitio http://www.cadsoft.de/

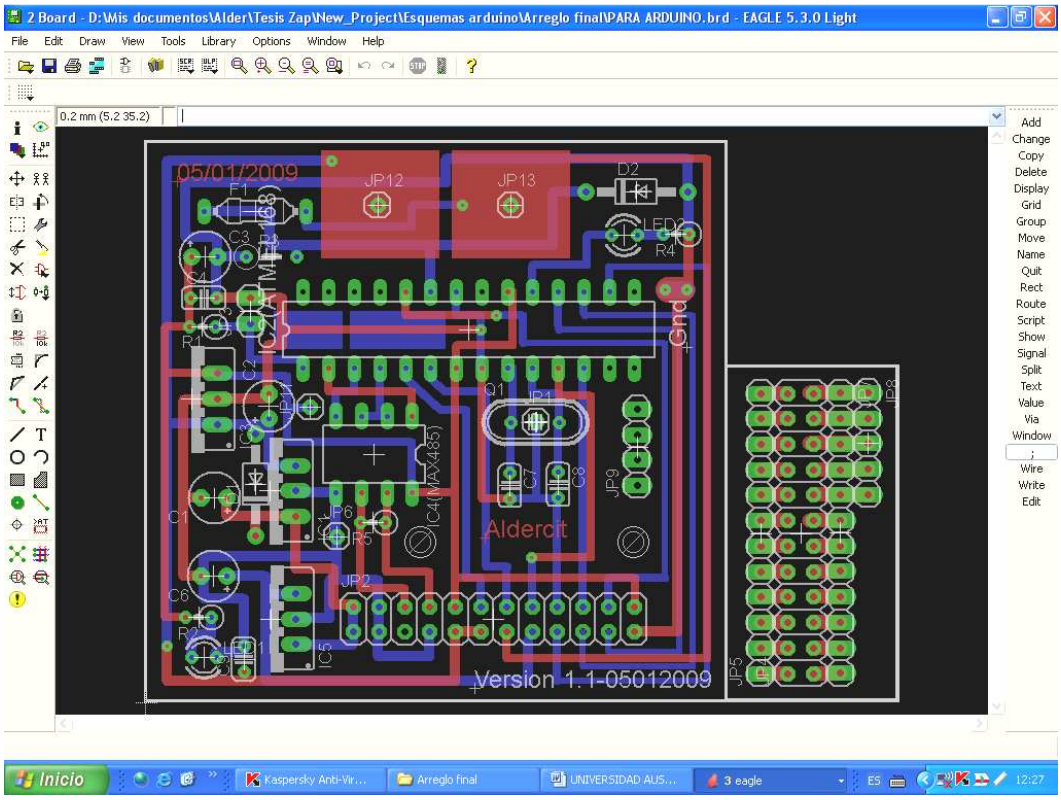


Figura Nº 3.9: Diseño de placa-unidad central de control y comunicación

Como se aprecia, la distribución hubo que realizarla en placa doble faz porque se dispuso de un espacio de no más de 58mm de ancho por 55 de largo donde se debía colocar los componentes electrónicos. Las línea azules son las pista que van por la cara inferior, las líneas rojas son las pistas cara superiores donde van montado los componentes, en lado inferior derecho se incorporó la placa que después será separada de la placa principal y que en definitiva sirve para montar el conector general de 24 pines hembra.

Una vez listos el diseños, el paso siguiente es traspasar ésto a la placa de cobre, para ello se exportó del programa Eagle (como imagen blanco y negro), se procesó en el programa Paint de Windows, se pegó como imagen en Word y luego se imprimió en una impresora láser en papel couché brillante autoadhesivo (que sea papel autoadhesivo no es importante pero fue el que mejor respondió frente a este proceso, es importante no retirar el autoadhesivo), posteriormente se fijó bien a la placa con cinta adhesiva y se aplicó calor por medio de una plancha, de esta forma se traspasa el diseño. Al ir retirando el papel couché se tuvo cuidado que la tinta vaya quedado traspasada a la placa de cobre, en los lugares donde no pasó completamente se repasó con cuidado la plancha, una vez retirado completamente el papel se repitió el procedimiento en la otra cara, pero para mantener la alineación se perforaron dos orificios. Una vez listo ésto, se remojó en agua la placa para retirar los restos de papel que quedaron (los lugares donde no pasó completamente la tinta se repaso con pintura esmalte). El próximo paso es poner la placa a la solución ácida para dejar solamente las pistas requeridas.

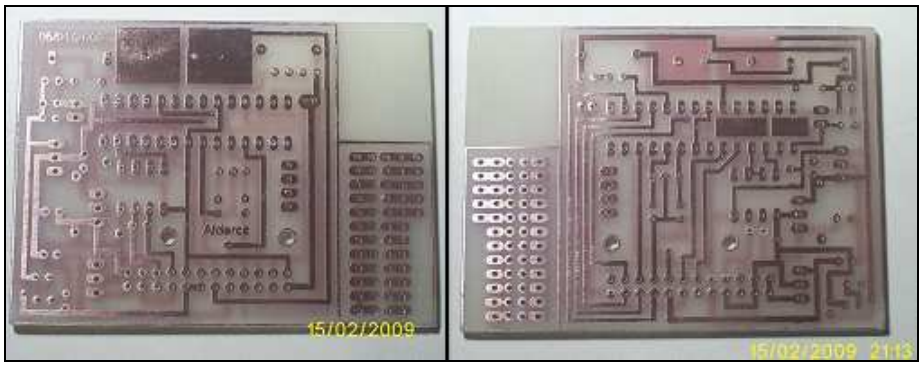


Figura Nº 3.10: Placa circuito impreso-imagen por ambos lados

Una vez realizada cada placa en cuestión se lija con una esponja de cocina fina y posteriormente se disuelve un poco de pasta de soldar (Pez de Castilla o colofonia) en alcohol y se le aplica, con esto la placa queda protegida contra la sulfatación.

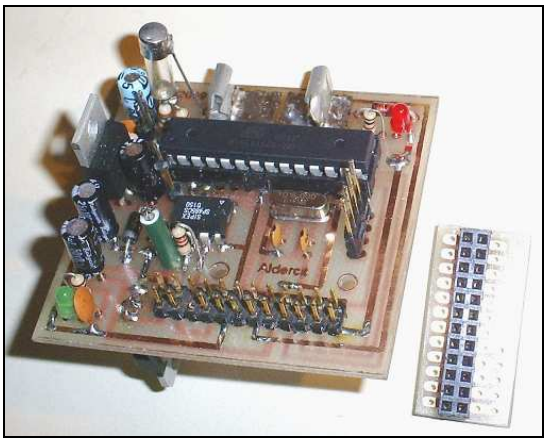


Figura Nº 3.11: Unidad de control y comunicación terminada

Como se aprecia, los reguladores L7812CV se colocaron por la parte inferior donde tenemos más espacio para colocar un disipador de temperatura.

Para la producción final, se mandó a fabricar esta placa a la empresa desarrolladora de placas electrónicas “Electrometal”, que esta ubicada en la ciudad de Santiago- Chile. (Sitio Web: www.electrometal.cl) lugar donde se cotizó a un precio bastante razonable y la calidad fue buena.

En la fabricación de esta placa los orificios fueron metalizaron y se escribió la ubicación de los componentes.

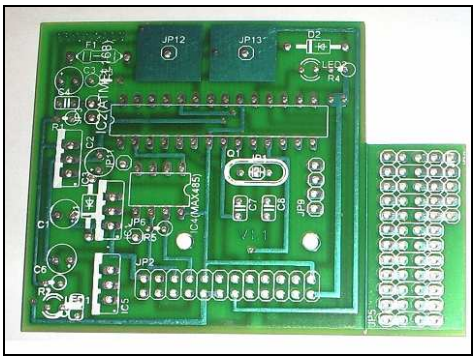


Figura Nº 3.12:
Placa fabricada en Electrometal

3.2 Desarrollo del modulo de conversión de video a señal diferencial

Como se menciona al principio, para transmitir el video se realiza mediante balun, para ello se ocupará la electrónica interna de estos dispositivos para ser montada en segundo nivel sobre la placa principal.

En el mercado se encuentran distintos tipos de balun, unos sólo incorporaban una bobina y otros, aparte de la bobina, también incorporan resistencia y algunos condensadores, también los hay con salida/entrada RJ45 llevando la alimentación por el par trenzado para alimentar la cámara en el otro extremo.

Para este caso se seleccionó uno sencillo que sólo incorpora una bobina, que es el que se muestra en la imagen a continuación.

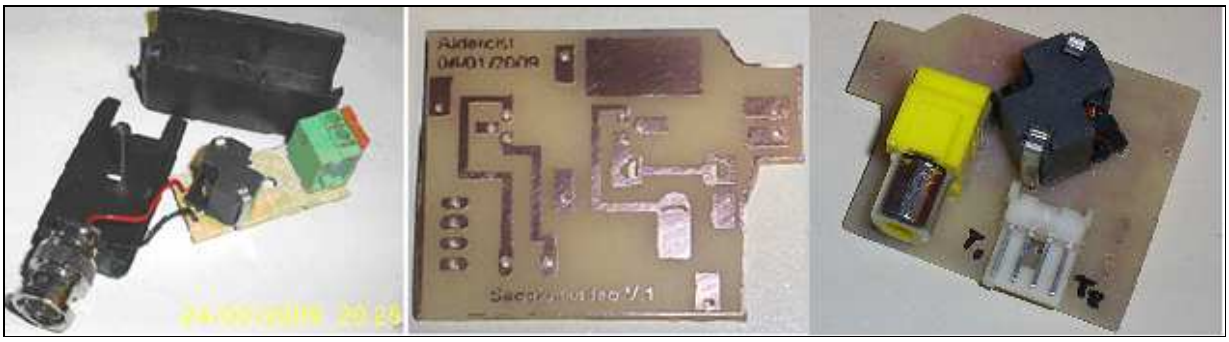


Figura Nº 3.13: Conversor de video a señal diferencial

En este caso no se muestra el esquemático porque es demasiado sencillo. Este modulo realiza la misma función del balun original, toma la señal en norma base RCA y la transforma a señal diferencial que se conecta al par trenzado.

En la imagen mostrada a la derecha, de los 4 pines que tiene el conector de salida, sólo se ocupan dos.

3.3 Diseño y desarrollo de los modulo de iluminación

3.3.1.1 Diseño del control de niveles (0-7)

Para esta etapa se necesitan dos fuentes luminosas compuesta por led, alimentadas por un voltaje máximo de 12Vdc y su control tiene que ser mediante 3 líneas lógicas.

En cuanto a las posibilidades para alcanzar este requisito, existirían 3 métodos que se podrían aplicar:

1.-El primer método consistiría en controlar la luminosidad de los led por pulsos PWM. En este caso los circuitos a construir serían pocos, pero el inconveniente que se presenta sería el ruido que posiblemente genere ésto en la transmisión de la imagen de video y en la imagen misma por efecto electroboscópico.

2.-El segundo método posible seria: controlar la luminosidad del led por medio del voltaje aplicado, el inconveniente que ocurre en este caso es que la luz, en los led, se deteriora a medida que se disminuye el voltaje aplicado y como consecuencia se pierde el brillo, por otro lado se tiene que la curva de luminosidad versus voltaje no es lineal, por lo tanto esta técnica quedaría descartada porque abría un deterioro de la calidad de luz.

3.-En tercer lugar tenemos un método un poco complejo, en su fabricación, pero nos asegura que no producirá ruido en el sistema y se obtendría la máxima eficiencia en los led, este consiste en encender determinados led por niveles, al tener 8 niveles, el primero nivel seria completamente apagado y el último seria todo encendido.

En cuanto a los circuitos que generen este tipo de señal se pensó en primer lugar en un decodificador binario a decimal, pero este no cumplía con todos lo requisitos. Por otro lado también se tuvo en consideración CI LM3914 que se ocupan generalmente para los voltímetros.

De todas las formas posibles, para controlar los led, se opta por generar una tabla de verdad que nos represente los niveles que se deben ir energizando de acuerdo a la combinación lógica de entrada.

C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	1	1	0	0	0	0	0
0	1	1	0	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	0	0	0
1	0	1	0	1	1	1	1	1	0	0
1	1	0	0	1	1	1	1	1	1	0
1	1	1	0	1	1	1	1	1	1	1

Tabla Nº 3.2: Tabla de verdad

A continuación se realiza el cálculo para obtener la combinación de compuertas lógicas de acuerdo a la algebra de Morgan.

$Y1 = \bar{C}\bar{B}A + \bar{C}B\bar{A} + \bar{C}BA + C\bar{B}\bar{A} + C\bar{B}A + CB\bar{A} + CBA$

$Y1 = A + \bar{A}(\bar{C}B + C)$

$Y2 = \bar{C}B\bar{A} + \bar{C}BA + C\bar{B}\bar{A} + C\bar{B}A + CB\bar{A} + CBA$

$Y2 = (\bar{C}B + C)$

$Y3 = \bar{C}BA + C\bar{B}\bar{A} + C\bar{B}A + CB\bar{A} + CBA$

$Y3 = \bar{C}BA + C$

$Y4 = C\bar{B}\bar{A} + C\bar{B}A + CB\bar{A} + CBA$

$Y4 = C$

$Y5 = C\bar{B}A + CB\bar{A} + CBA$

$Y5 = C(A + B\bar{A})$

$Y6 = CB\bar{A} + CBA$

$Y6 = CB$

$Y7 = CBA$

Luego para reducir el número de compuestas lógicas se mezclan algunas expresiones y para reforzar la salida Y4 se aprovecha las compuertas NOT sobrantes obteniéndose la combinación que se presenta a continuación.

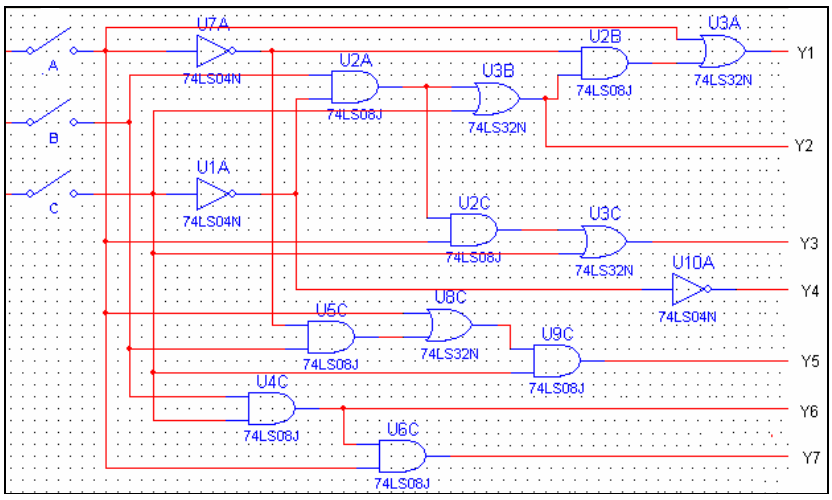


Figura Nº 3.14: Combinación de compuestas lógicas

3.3.1.2 Diseño de la etapa de potencia control led.

En este proyecto se requiere como mínimo 390.000 mcd por foco, por lo tanto con unos pocos leds de hoy en día se puede obtener ese nivel de luminosidad. Por un tema de concentración de luz se utilizan led de 20º de apertura, ya que la luz en el agua se expande.

A continuación se presentan las especificaciones técnicas de los led a utilizar:

White led Clear -Led Blanco Claro-20 grados 40000mcd

Características Típicas eléctricas y ópticas Ta=25ºC

DC forward voltage : VF (IF =20mA)	Typ 3.2V-3.4V, 3.8VMax
DC reverse current : IR (VR =5V)	100u
Intensity luminous : Iv (IF =20mA)	BA40 : 30000-40000mcd 35000-43000mcd (IF=25mA) 43000-53000mcd (IF=30mA)

Tabla Nº 3.3: Características técnicas led a utilizar

Fuente de Información:http://www.besthongkong.com/product_info.php?products_id=132

En cuanto a los led éstos serán controlados por transistores, en el siguiente dibujo se aprecia la combinación de led que se irán energizando, desde el nivel 0 hasta el nivel 7. Cada nivel enciende un arreglo de 2 led en serie (pudiéndose también haber realizado un arreglo de 3 led, para no desperdiciar potencia en calor), al estar encendidos todos los niveles se obtendrá un total de luminosidad que estará comprendido entre los 490.000 mcd y 560.000 mcd por foco con lo cual se cumple con lo requerido. Se realizó un arreglo de encender dos leds, por niveles, más que nada por un tema de simetría de la luz

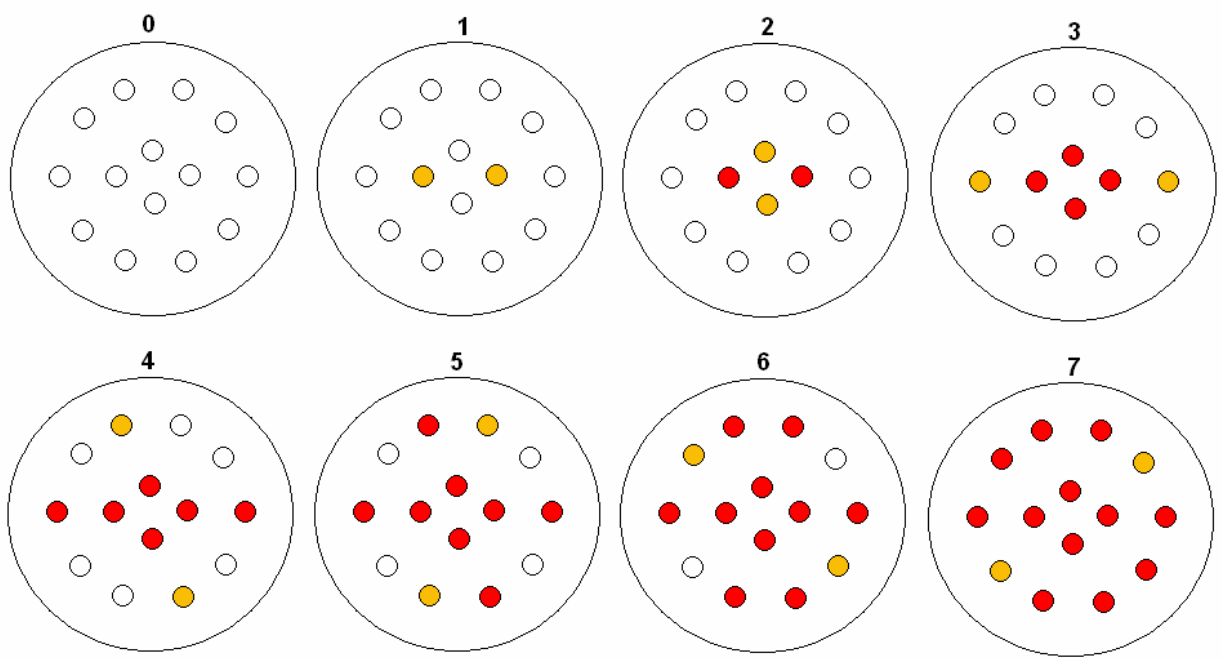


Figura Nº 3.15: Niveles de iluminación (0-7)

De acuerdo a los datos que se tienen se procede a calcular la resistencia que se coloca en serie con los diodos led para su protección.

Para controlar la corriente fuerte de los led se utiliza el transistor BC547 (ECG 123AP) y de acuerdo a las especificaciones técnicas, de este transistor, al trabajar en su punto de saturación se produce una caída de tensión, entre colector y emisor, de V_{CE}(sat)= 0.1 volt aproximadamente.

Cálculo de la resistencia serie para los led.

Como: $Vt = Rx * It + (2 \times Vf) + V_{CE}(sat)$

Vt =12 Volt.

IF=Id =It=20ma o 25ma (Corriente típica led)

Vf = 3.2 V (voltage típico led)

$V_{CE}(sat)$ = 0.1 volt

Para una corriente de 20ma tendríamos lo siguiente.

$Rx = (Vt - (2 * Vf) - V_{CE}(sat)) / If$

Rx= (12- (2*3.2)-0.1) / (20ma) =

RX = 5.5/ 20ma = 275 Ω

Para una corriente de 25ma tendríamos lo siguiente

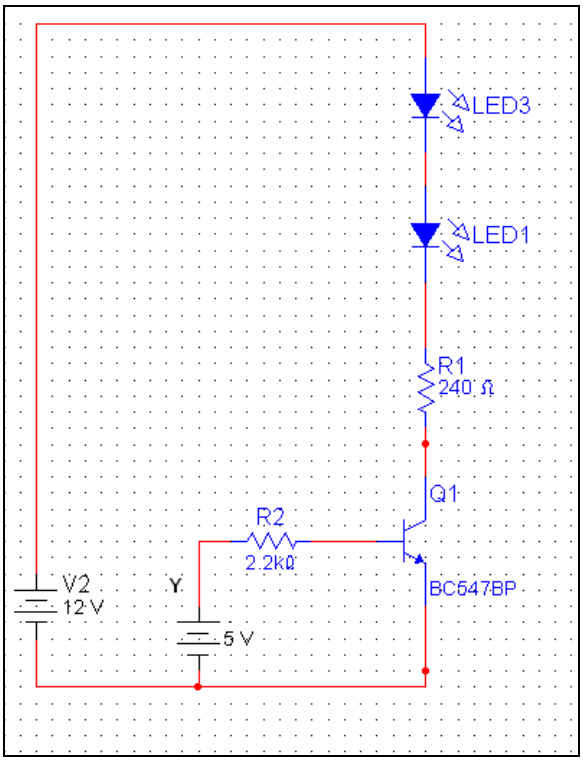


Figura Nº 3.16: Activación de un nivel

Rx = (Vt - (2 * Vf) - V_{CE} (sat)) / If =

Rx = (12- (2*3.2)-0.1) / (25ma)

Rx = 5.5/ 25ma = 220Ω

Como precaución se deja un valor en 240Ω ½ Watt ya que es el promedio entre los dos valores calculados y corresponde a un valor Standard de resistencia.

Para la activación del transistor se debe calcular la resistencia que limita la corriente a la base de este, para ello se baso en los datos técnicos del transistor BC547. Para colocar el transistor en saturación, se observa uno de los gráficos de la hoja técnica, donde se especifica que para una IC= 23ma aproximadamente V_{BE} (sat)=0,85 Volt y como la corriente a circular por la base esta dada por Ic/Ib=10, para este caso, despejando tenemos Ib= 23ma/10= 2.3ma

Por lo tanto para calcular la resistencia de base se tiene que $Vb= Rb*Ib-V_{BE}(sat)$

Donde $Rb= (Vb - V_{BE}(sat))/Ib = (5 – 0,85) / 2.3ma = 1804\ \Omega$

Como resistencia de base se puede utilizar una de 1.8KΩ, 2KΩ y por seguridad una de 2.2KΩ, que es la que se aplico en este caso y que es la que generalmente se utiliza para colocar este transistor en saturación.

Una vez clara la electrónica a utilizar, se armó el circuito en un protoboard para ver la respuesta del diseño y posteriormente se procede a diseñar la placa circuito impreso utilizando el programa Eagle 5.3.

Para comprender mejor este modulo a construir se ha dividido en dos etapas:

- Etapa de control.
- Etapa de distribución de los leds.

3.3.2.1 Desarrollo de la etapa de control.

A continuación se presenta el esquema completo del sistema controlador de los leds.

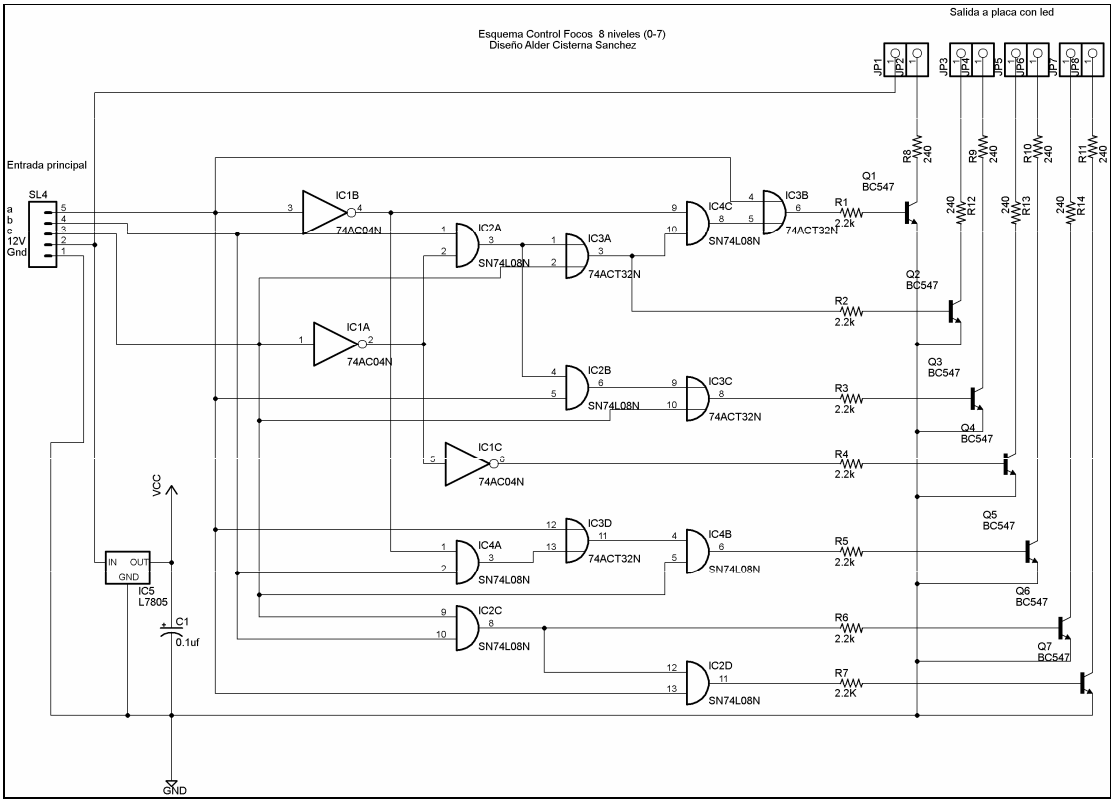


Figura Nº 3.17: Esquemático etapa de control

Posteriormente se realiza una distribución de los componentes para que entre en un círculo de 58mm de diámetro, para lo cual obligadamente tuvo que realizarse en una placa de doble faz.

A continuación se presenta la distribución de componentes:

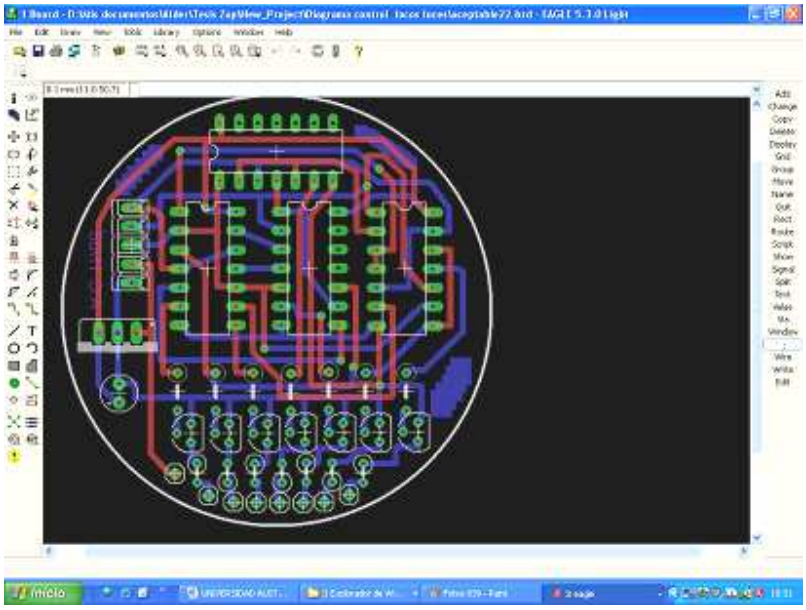


Figura Nº 3.18: Diseño de placa control led

3.3.2.2 Desarrollo de la etapa de distribución de los led.

Como se observa esta etapa sólo contempla a los leds.

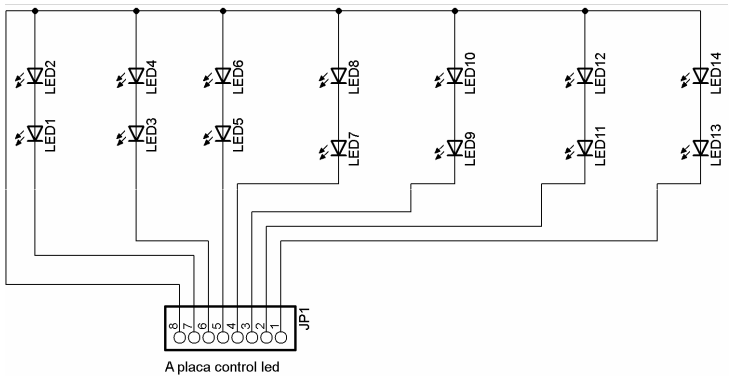


Figura Nº 3.19: Esquemático etapa distribución led

A continuación se muestra la distribución de los leds y las conexiones. En la etapa de producción final se amplio los caminos (en el programa Paint) para aprovechar más eficientemente el ácido, también se traspasó el diseño a dos placas para poder soldar sin problemas y se estañaron las pistas para protección de corrosión, ya que estas placas pudieran estar expuestas a algún agente salino.

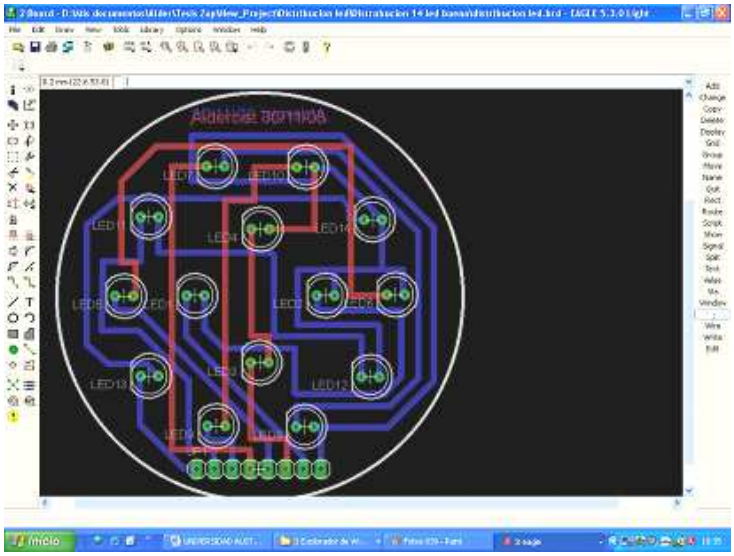


Figura Nº 3.20: Diseño de la placa distribución led

A continuación se muestra una imagen donde se identifican todas las partes de esta etapa.

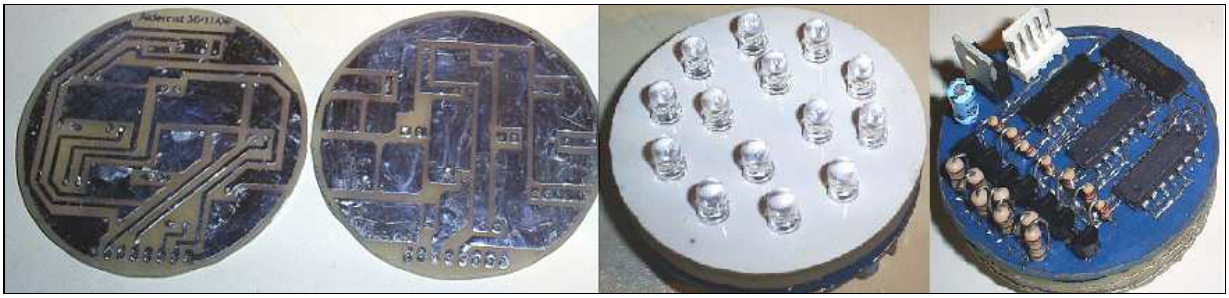


Figura Nº 3.21: Foco led

Nota: Cada foco quedó con un consumo de 180ma al alimentarlos con 12v, por lo tanto la potencia consumida es de 2,16 Watt por foco.

CAPITULO IV
DESARROLLO DE LA UNIDAD CONSOLA

4.1 Implementación de la unidad consola.

Esta etapa contendrá dos partes que se dan conocer a continuación.

4.1.1 Conversor de RS-232 a RS-485

Para esta etapa se estudiaron 2 tipos de circuitos, uno utilizando la señal RTS (Request to Send, petición de envío) que envía el puerto serial cuando va a transmitir datos. Al colocarse en el casos de trabajar con un Notebook, conectado a la sección consola, lo más probable es que se utilice un conversor USB a RS-232 y en este caso estos conversores generalmente no activan la señal RTS de salida por lo tanto nunca se va a enviar la señal para que el circuito integrado MAX485 trasmita.

La segunda alternativa es un circuito que funciona activando la transmisión del MAX485 solamente cuando se envía información (activación por la señal Txd,) siendo una buena alternativa para este proyecto, para comprobar la funcionalidad de estos circuito se construyó una placa que reunía las 2 posibilidades encontradas después de algunas pruebas se elige la segunda opción (activación por la señal Txd,) que es la que se presentará a continuación.

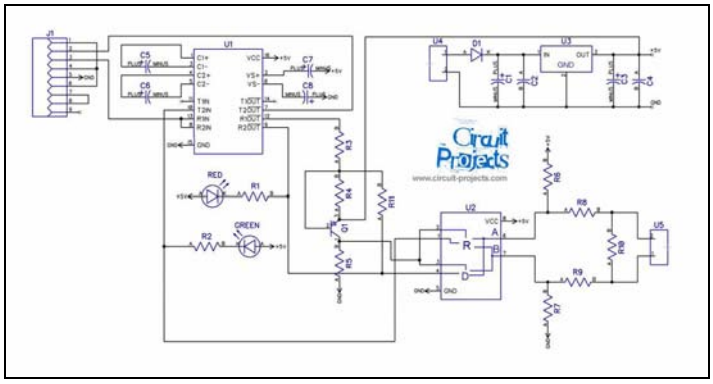


Figura Nº 4.1: Alternativa conversor RS-232 a RS-485

Para más información visitar página Web:
http://www.circuit-projects.com/converter-circuits/rs232-rs485-converter-with-automatic-rx-tx-control.html

El circuito anterior fue levemente modificado en su fuente de poder, el diodo común de entrada tendrá la finalidad de proteger el sistema ante errores al momento de conectar la energía (inversión de polaridad).

4.1.2 Conversor de video

La unidad consola también incorpora otro balun para convertir la señal de un medio balanceado a un medio no balanceado. Para ello se retiró el transformador de uno de estos dispositivos y se incorporó a la placa a construir, que contendrá el conversor RS-485 a RS-232. Se recuerda que el balun no es necesario energizarlo, su fuente de energía es la propia cámara de video de la unidad remota.

4.2 Desarrollo de la placa de circuito impreso

Esquema circuito conversor de RS232-RS485/video

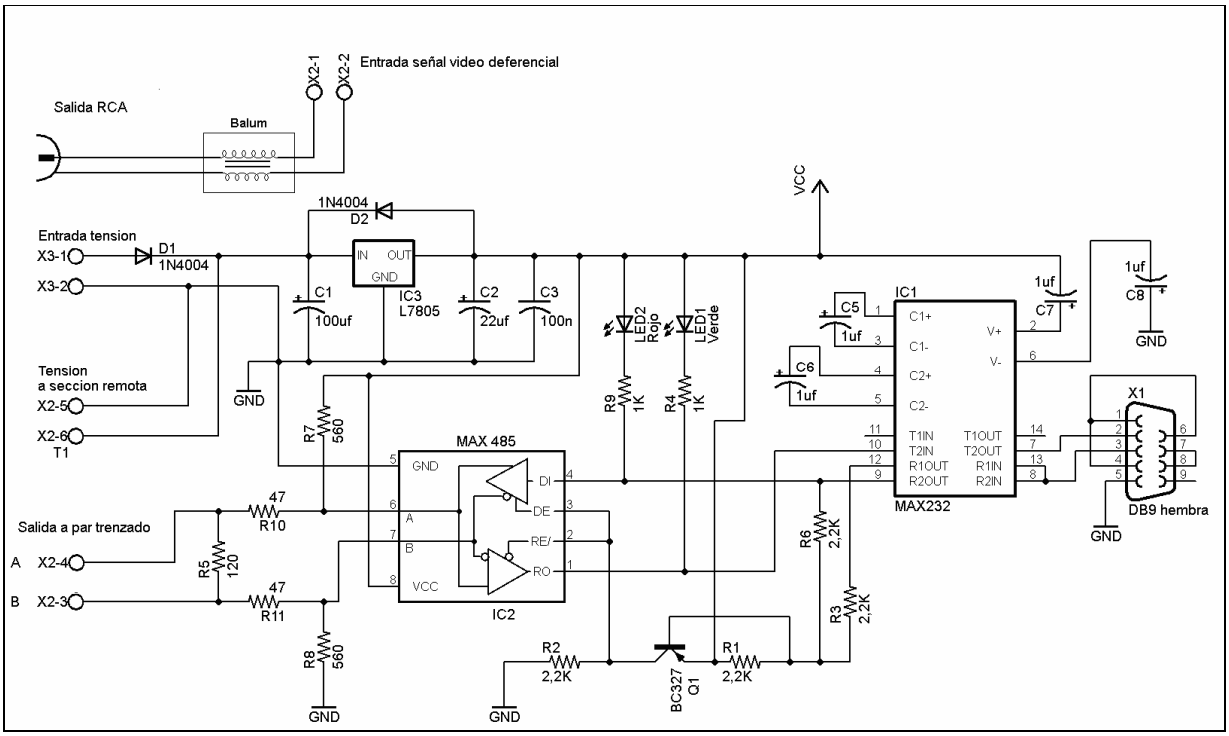


Figura Nº 4.2: Esquemático conversor RS232-RS485/video

El funcionamiento de este circuito es el siguiente, cada vez que el computador está enviando información, mediante el pin 3 del puerto serial (+-15 Volt), el CI. MAX232 toma estos datos y los convierte en niveles TTL (0-5V). Posteriormente mediante el transistor y su configuración, éste le envía un nivel alto a los pines 2 y 3 del C.I. MAX485 (inicio de palabra nivel bajo en pin 12 y 9 de MAX232) y de esta forma deja habilitado este circuito para que envíe información por el par trenzado, en el caso contrario en que se esté recibiendo información, el transistor queda en estado de reposo y coloca un 0 lógico en los pines 2 y 3 del integrado MAX485 y éste queda habilitado para recibir información. La razón por la que se puentean algunos pines de salida del DB9 esta explicada en los anexos de este trabajo. Dicho de otra forma, en este circuito, el que tiene el control de la transmisión es el computador, ya que al transmitir información automáticamente se habilita el integrado MAX485 sólo para enviar información (las resistencias de 560omhs y 47omhs a la salida del MAX485 son necesarias para que el circuito pueda enviar una palabra completa). Es posible que en algún momento exista alguna colisión producto de que se este enviando un dato para la unidad remota y desde la unidad remota también se esté enviando información, en este caso las dos tramas se destruyen perdiéndose en la colisión.

Posteriormente se procede a distribuir los componentes mediante el programa Eagle quedando como se muestra a continuación.

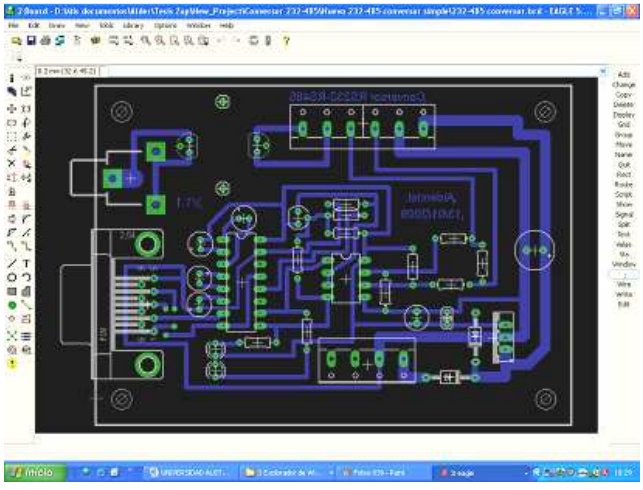


Figura Nº 4.3: Diseño distribución componentes conversor

Mediante el procedimiento anteriormente mencionado se procedió a realizar la placa de circuito impreso, como se observa en la siguiente imagen, en el lado izquierdo se ha retirado el papel y se ha remojado en agua para retirar lo sobrante, al lado derecho ya esta lista la placa con su protección de pez de castilla disuelta en alcohol.

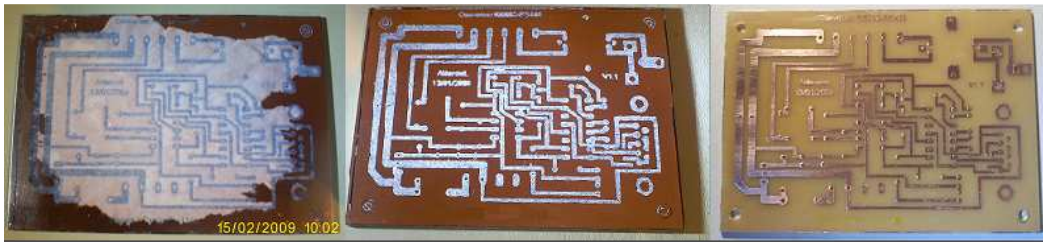


Figura Nº 4.4: Placa circuito impreso-conversor

Figura Nº 4.5: Conversor RS232-RS485/video terminado y funcionando.

Esta unidad al estar transmitiendo (sin tener conectada la unidad remota) y estar alimentada por 24V tiene un consumo 0.020ma, por lo tanto consume una potencia de 0,48 Watt.

Nota: Al realizar un leve análisis es posible construir un conversor USB a RS-485, tomando como base el esquema del Arduino y el antes mencionado. Por una parte la tarjeta Arduino Decimila trabaja con un Chip FT232RL el cual convierte de USB a TTL.

Si en el conversor RS-232 a RS-485 se retira el MAX232 y todos sus componentes de polarización queda un circuito TTL a RS-485, con estas dos partes USB a TTL y TTL a RS-485 se puede formar este circuito. Hay que ver si las salidas Txd del Chip FT232RL soportan la corriente que absorbe el transistor para activación del MAX485 (transmitir-recibir), por otro lodo los driver de reconocimiento del Chip FT232RL por parte del PC están en el paquete de Arduino Diecimila.

A continuación se aprecian dos imágenes con el prototipo terminado en su parte física (etapas probadas independientemente). El próximo paso es la programación del microcontrolador.

Figura Nº 4.5: Prototipo - parte física terminada

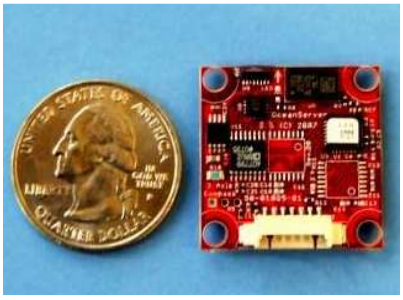
Como este diseño corresponde al prototipo, la protección que contiene la electrónica de la unidad remota es sólo para ver las dimensiones físicas.

Bajo las pruebas finales se corroboró que el consumo total de la unidad remota, con sus unidades conectadas (sensor de presión, brújula, 2 focos, en este caso la cámara de video no se conectó) y tomando el peor caso en que el voltaje de llegada sea 14.6 Volt la corriente de consumo es de 420ma. Si a ésto le sumamos ahora la corriente que consumiría la cámara de video, para la cual fue diseñado el sistema (240ma) tendremos un consumo total de 660ma aprox. que vendría a corresponder a 9,6 Watt, por lo tanto se ha cumplido con no sobrepasar los 700ma.

CAPITULO V
COMUNICACIÓN CON LA BRÚJULA ELECTRÓNICA.

5.1 Preliminar

De todas las brújulas electrónicas mostradas al comienzo de este trabajo se utilizará la del fabricante Ocean-Server modelo:



OS5000-T (Signals: TTL Levels: 3 -Axis Compass)

Figura Nº 5.1: Brújula Electrónica

5.2 Características técnicas de brújula.

• Precision compass accuracy, 0.5 degrees nominal, 0.1 resolution
• Roll & Pitch full rotation (<1 degree (0-60 degrees))
• Pitch Angles +/-90 degrees, Roll Angles +/- 180 degrees
• Tilt-compensated (electronically gimbaled)
• Tiny size, 1”x1”x0.3”, less than 2 grams weight
• Low Power Consumption, <30ma @3.3V
• Hard and soft-iron compensation routines
• Optional support for a high resolution Depth or Altitude sensor (24bit AD)
• Serial Interface:
    - TTL
    -Baud rate programmable 4,800 to 115,000 baud
• Rugged design
    -10,000 G shock survival
    - 20C to 70C operating temperature, -40C to 85C non-op
• ASCII sentence output, in several formats, NMEA checksum
• High Data Update Rate to 40HZ
• Support for True or Magnetic North Output
• Precision components
    - 3 Axis magnetic sensors from Honeywell
    - 3 Axis Accelerometers from ST Microelectronics
    - 24 bit differential Analog to Digital converters from Analog Devices
    - 50 MIPS processor supporting IEEE floating point math

A continuación se presentan los pines de conexión de esta brújula, enumerados de izquierda a derecha.

Pin	Color	Signal	Description	Additional OS5000-T (TTL version compass notes)
1	White	P-in	Pressure Input from Transducer (OS5500-USG only)	same
2	Black	GND	Ground	Same
3	Red	Vin- Unreg	DC Power Input, DC voltage in the range of 3.3V - 5V	same
4	Orange	NC	RESERVED - Do Not Connect	Same
5	Black	GND	Ground – Use with RS232	Same
6	Green	Tx	RS-232 Transmit (DB9-F pin: 2)	Voltage Level TTL, 3V drive level
7	Blue	Rx	RS-232 Receive (DB9-F Pin: 3)	Voltage Level TTL, 3V or 5V tolerant on input

Tabla Nº 5.1: Pines brújula electrónica OS5000-T

Como se trabaja con la brújula que entrega la información en niveles TTL, sólo se ocupa los pines que se han resaltado en azul.

Para configurar y colocar a punto la brújula se utilizo el programa proporcionado por el fabricante, el cual se bajo en el sitio www.ocean-server.com. Una vez instalado el programa se construyó un conversor RS-232 a TTL para poder conectar la brújula al puerto serial del PC.

La brújula electrónica envía la información utilizando un protocolo de comunicación **NMEA 0183** (National Marine Electronic Asociation) que es un protocolo estándar entre todos los equipos electrónicos marítimos.

Según el fabricante, para poder comunicarse con la brújula por primera vez, es preciso configurar el puerto serial del PC a una velocidad de 19200,8,1,N.

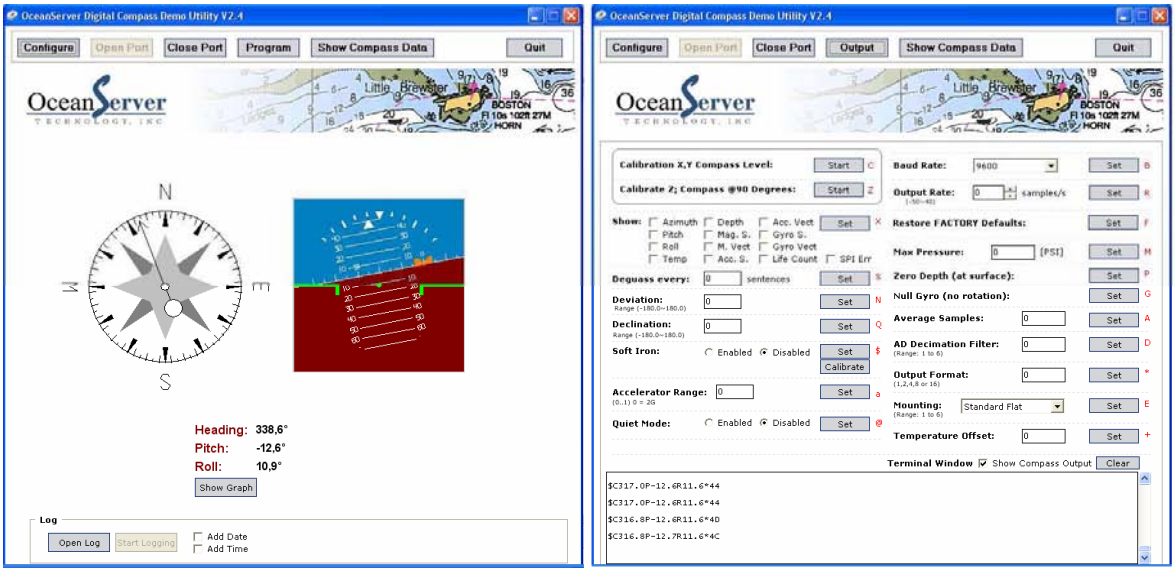


Figura Nº 5.2: Software de configuración brújula electrónica

Una vez comunicado con la brújula se configuró para que entregara los datos Heading, Pitch y Roll, se calibró con respecto al norte magnético, se configuró para que entregara 4 muestras por segundo, como también para que se comunique a 9.600 baudios. Cabe mencionar que la brújula puede entrega los datos en distintos tipos de formatos, en nuestro caso se seleccionó la opción Formato “$C” datos en String (código ASCII).

5.3 Datos enviados por la brújula (Formato “$C”)

El Formato $C entrega la información en un String y de acuerdo a la configuración realizada, la trama contendrá los datos Heading, Pitch y Roll.

Ejemplo

$Chhh.hPpp.pRrr.r*cc

$C357.8P-3.3R-1.1*67

Este formato de sentencia proporciona una etiqueta de texto delante de cada tipo de datos. Entonces el programa de usuario debe de ubicar estas letras para determinar

donde comienza y termina la información respectiva. En efecto cada elemento de datos tiene un nombre único.

Chhh.h = Heading-Yaw en grados, “C” precede el heading en grados.
Ppp.p = Pitch orientación en grados, “P” precede el Pitch en grados.
Rrr.rr = Roll orientación en grados, “R” precede el ángulo del Roll en grados.
cc = código chequea Checksum, ““ precede el Checksum en ASCII, el código Checksum es cálculo (operación Xor en hexadecimal o binario) de los caracteres que se encuentran entre el $ y el *.

Ejemplos:
String enviados por la brújula (ASCII): $C357.8P-3.3R-1.1*67
Código Hex: 24 43 33 35 36 2E 38 50 2D 33 2E 33 52 2D 31 2E 31 2A 36 37 0D 0A
Heading: 357.8 grados
Pitch angle: -3.3 grados
Roll angle: -1.1 grados
Checksum: 67

Donde:
0D= CR (retorno de Carro)
0A= LF (el salto de línea)

Largo máximo String-Código ASCII: $C339.6P-85.1R-175.8*5A
Código Hex: 24 43 33 33 39 2E 36 50 2D 38 35 2E 31 52 2D 31 37 35 2E 38 2A 37 33 0D 0A

Según la información del fabricante, el checksum se entrega después que llega el signo asterisco, también es importante ver que el vector de memoria que se necesita para almacenar la información antes descrita, debe de ser como mínimo de 25 bytes, aunque por precaución, en el programa, se tonará un vector de memoria de 32 bytes (si se han seleccionados 3 variables de lectura Heading, Roll y Pitch.).

CAPITULO VI
DESARROLLO DEL PROGRAMA PARA LA UNIDAD CENTRAL DE CONTROL DE LA UNIDAD REMOTA

6.1 El entorno de desarrollo de Arduino.

Para programar la placa Arduino Diecimila es necesario descargar, en forma gratuita, de la página Web de Arduino http://www.arduino.cc/en/Main/Software el entorno de desarrollo (IDE). Al estar utilizando la versión USB es necesario instalar los drivers del chip FTDI para que el PC la reconozca. Estos drivers vienen incluidos en el paquete de Arduino mencionado en el apartado anterior.

Una vez instalado el entorno de desarrollo Arduino (IDE) es necesario definir el tipo de tarjeta de desarrollo que se esta utilizando, en este caso Arduino Diecimila. Posteriormente se debe elegir el puerto serial de comunicación, para ello hay que abrir en el menú *"Tools"* las opción *"Serial Port"* y seleccionar el puerto Com. 3, Com. 4, Com. 5 dependiendo al puerto USB al cual se haya conectado.

El software de Arduino (IDE) maneja una determinada cantidad de comandos los que se dan a conocer en la página oficial Arduino

http://arduino.cc/en/Reference/HomePage (versión Ingles)
http://arduino.cc/es/Secundaria/Referencia (versión español)

Nota: Al utilizar el navegador Microsoft Internet Explorer, es necesario cambiar la codificación a "Europeo Occidental" para poder visualizar correctamente la literatura en la página en español.

Es importante destacar que el entorno de desarrollo (IDE) soporta códigos de programación utilizados en C++ y que no están definidos en la página oficial de Arduino.

A continuación se dan a conocer algunos puntos importantes en la programación en el entorno de desarrollo Arduino.

Estructura

Un programa Arduino se ejecuta en dos partes:

- void setup()
- void loop()

Setup () es la parte de la preparación del programa, y **loop ()** es la parte de la ejecución del programa. En la sección setup, siempre al comienzo del programa, se asignarían los modos a los pines (pinMode), se inicializaría la comunicación serie, etc. La sección loop es el código que va a ser ejecutado - lectura de entradas, disparar las salidas, etc. El tiempo del bucle, varía según el número de instrucciones que contenga.

Estructuras de control

Las estructuras de control sirven para controlar el camino a seguir del programa, de acuerdo al algoritmo programado y a los acontecimientos de lectura de entradas.

- if
- if...else
- for
- switch case
- while

Variables

Las variables son expresiones que se pueden usar en los programas para almacenar o guardar valores. Éstas pueden ser de varios tipos, los cuales son descritos a continuación.

Char: Tipo de datos para definir caracteres (ASCII), símbolos tipográficos tales como A, d, y $. Cada tipo char ocupa un byte (8 bits) de memoria y debe ser delimitado por comillas sencillas.

int (entero corto) : El tipo entero es un tipo de datos que almacena números, y los almacena en campos de 2 byte (16 bits), tomando valores negativos y positivos lo que nos da un rango de -32.768 a 32.767.

long (entero largo) : El tipo de dato Long o tipo de entero largo almacena un número de rango extendido, y los almacena en campos de 4 byte (32 bits), tomando valores negativos y positivos lo que nos da un rango de -2.147.483.648 a 2.147.483.647 .

Boolean: Tipo de datos para valores Booleanos de verdadero (true) o falso (false). Es común usar valores de tipo boolean con la estructuras de control y para determinar el flujo o secuenciación de un programa.

byte : Tipo de datos para bytes, 8 bits de información que almacenan valores numéricos desde 127 a -128. Estos tipos de datos son adecuados para enviar información a y desde el puerto serie, y para representar letras en un formato más simple (caracteres ASCII) que el tipo de datos char.

Array (Vector): Un vector se define como una estructura, formada por un conjunto o agrupación de datos del mismo tipo cuyo acceso se realiza por un índice o puntero que representa su posición en el vector. El primer elemento en el vector es [0], el segundo elemento es [1], y así sucesivamente. En todos los vectores el índice del primer elemento siempre es 0 por lo tanto, cuando se declara int Vec[5] se está declarando un vector que tiene 5 elementos de tipo entero, desde Vec[0] hasta Vec[4].

6.2 Desarrollo del protocolo de comunicación.

Para comunicar las dos unidades es preciso definir un protocolo de comunicación, para ello se ha nombrado a la unidad remota F1 (esclavo) y a la unidad consola F0 (maestro “PC”) se le han asignados nombres ya que la norma RS-485 soporta hasta 32 puntos de conexión donde cualquiera pudiera estar enviando información destinada a un receptor especifico.

Para solicitar datos a la unidad remota se ha creado el siguiente comando en código hexadecimal (el signo $ indica Hex).

$F1$01$00$0A$FA (*Solicitud datos)*

Donde F1 quiere decir que la información que se enviará será para la unidad remota, “01” corresponde a la solicitud de datos, “00” con “0A” es un relleno, y por ultimo “FA” corresponde al código de chequeo checksum (operación Xor entre los octetos anteriores enviados)

Un caso similar son los comandos de control de iluminación, donde se solicitara cambiar a un nuevo nivel de iluminación.

$F1$02$00$0A$F9 *(cambiar nivel de iluminación a 00)*
$F1$02$01$0A$F8 *(cambiar nivel de iluminación a 01)*
$F1$02$02$0A$FB *(cambiar nivel de iluminación a 02)*
$F1$02$03$0A$FA *(cambiar nivel de iluminación a 03)*
$F1$02$04$0A$FD *(cambiar nivel de iluminación a 04)*
$F1$02$05$0A$FC *(cambiar nivel de iluminación a 05)*
$F1$02$06$0A$FF *(cambiar nivel de iluminación a 06)*
$F1$02$07$0A$FE *(cambiar nivel de iluminación a 07)*

Donde F1 quiere decir que la información que se enviará será para la unidad remota, 02 corresponde a la solicitud de cambio nivel de luces, el octeto siguiente identifica el nuevo nivel de luces que se requiere (00-07), 0A corresponde a un código de relleno, y por último viene el código de chequeo checksum. El nuevo nivel de

iluminación será corroborado cada vez que se soliciten datos para ver si el sistema capturó el último comando de solicitud de cambio nivel de luces. También es importante destacar que el programa en la unidad remota necesita un vector de memoria de memoria mínimo de 5 Byte para almacenar el comando enviado, por precaución se tomará uno de 8 bytes.

Las respuestas que deberá enviar la unidad esclava al maestro se definen a continuación.

La información recibida representa lo siguiente.

$F0$F1$01$00$00$00$00$00$00$00$00$00 *(comando desconocido)*
$F0$F1$00$00$00$00$00$00$00$00$00$01 *(unidad remota conectada)*
$F0$F1$00$01$2D$FF$F4$FF$E7$01$7F$01$41 *(datos solicitados)*

Cuando se envía un dato a la unidad consola (PC) ésta representara lo siguiente:

Primer octeto = Identifica que el dato enviado es para la unidad maestra "F0"

Segundo octeto = Identifica que unidad esta respondiendo al maestro, en este caso responde la unidad consola F1, en caso de haber más de un esclavo.

Octeto tercero = Identifica si el dato enviado a la unidad esclava fue leído correctamente o no, 00 todo OK, 01 comando desconocido, en caso de detectar el código 01, la unidad maestra no tomará en cuenta los datos que vienen a continuación y volverá a enviar el último comando.

Octeto cuarto y quinto= Corresponde a la lectura Heading orientación respecto al Norte del compás electrónico, en nuestro caso este dato siempre será positivo, y su rango es de 0 a 360 grados, en nuestro ejemplo $01$2D corresponde a 301 grados.

Octeto sexto y séptimo = Corresponde a la lectura Pitch inclinación eje horizontal ala-ala (cabeceo) del compás electrónico, este dato puede ser positivo o negativo, su rango es de +/-90 grados, en ejemplo FFF4 correspondería a un dato negativo = -12, tomando la lógica que FFF4 =1111111111110100 al aplicarle Not queda 1011 y sumarle 1 lógico queda 1100 = 12 (complemento a 2)

Octeto octavo y noveno= Corresponde a la lectura Pitch inclinación eje horizontal punta cola del compás electrónico, su rango es de +/-180 grados, este dato puede ser positivo o negativo, en ejemplo FF$E7 correspondería a un dato negativo = -25, tomando la lógica que FFE7 =1111111111100111 al aplicarle Not 11000 y sumarle 1 lógico queda 11000= 25 (complemento a 2)

Octeto décimo y décimo primero = Corresponde a la lectura del sensor de presión, este dato esta comprendido entre 204 y 1023(CC y 3FF), y siempre será positivo. La conversión a profundidad se realizará en el maestro-PC, en nuestro ejemplo el dato recibido es $01$7F= 383. Si el valor de llegada estuviera entre 195 y 204 el dato recibido se tomará como 204, para la representación a metros de profundidad se calculará con la siguiente fórmula que se explicará más abajo.
Profundidad= (Presión - 204)*255/819

Octeto décimo segundo = Corresponde al nivel en que se encuentran las luces (00-07), en nuestro ejemplo el dato recibido es 01=01.

Octeto décimo Tercero = Corresponderá al código checksum que corrobora que el dato recibido es válido. Para ello se realiza una operación Xor entre cada uno de los octetos anteriores, ejemplo
$F0$F1$00$01$2D$FF$F4$FF$E7$01$7F$01 = $ 41 en este caso al recibir la trama y calcular el checksum y si es distinto de 41 se dice que el dato esta corrompido y se desecha.

6.3 Programando en el entorno de desarrollo Arduino (IDE)

Como se mencionó anteriormente es importante tener presente las asignaciones que se le han dado a los pines de la tarjeta de desarrollo Arduino, los cuales se describen a continuación y serán los que se tomaran en cuenta en la programación.

Pines plataforma Arduino Diecimila	Usos
Pin 0 (Rxd)	Recepción Rs485
Pin 1 (Txd)	Transmisión Rs485
Pin 2 (digital)	Habilitar Tx o Rx en Rs485
Pin 3 (digital-PWM)	Comunicación, Tx a Compás (Rx de compás)
Pin 4 (digital)	Comunicación, Rx a Compás (Tx de compás)
Pin 5 (digital-PWM)	**Pin digital Libre (uso Futuro)**
Pin 6 (digital-PWM)	**Pin digital Libre (uso Futuro)**
Pin 0 (analógico)	Entrada analógica sensor presión
Pin 1 (analógico)	**Pin analógico libre (uso Futuro)**
Pin 10 (digital-PWM)	Salida control luces “a”
Pin 11 (digital-PWM)	Salida control luces “b”
Pin 12 (digital)	Salida control luces “c”
Pin 13 (digital)	Salida led indicador procesos

Tabla Nº 6.1: Pines tarjeta desarrollo Arduino a utilizar

La velocidad de comunicación con la unidad remota, se establecerá en 19.200, con 8 bit de datos, sin paridad y 1 bit de stop.

Para realizar el programa, en el entorno de desarrollo, se ha separado la programación en módulos los cuales contienen los comandos específicos para cada función, se realiza de esta forma para poder ubicarse mejor dentro del programa.
En este caso se han creado 4 módulos.

Módulo InicioPrograma: Contiene la rutina principal del programa, su función es cargar librerías, definir parámetros, definir los pines como entrada o salida e inicializar los puertos de comunicación.

Módulo Compás: Contiene las subrutinas que reciben los datos del compás electrónico identifica el comienzo de la trama de datos, captura la trama, chequeando el Checksum y procesando la información para identificar el Heading, Roll y Pitch.

Módulo ComunicaPC: Contiene las subrutinas que reciben los datos desde la unidad consola (PC - maestro), identificando el inicio de la trama de datos, captura la trama, chequea el Checksum y si éste esta correcto continúa con el programa o, de lo contrario, borra lo almacenado y le indica a la unidad remota “comando desconocido”.

Módulo procesacomandos: Este modulo analiza la información recibida desde la unidad consola para determinar si la trama corresponde a una solicitud de datos o una solicitud de cambio nivel de iluminación de los focos. En caso de ser una solicitud de datos el sistema toma los valores almacenados en la memoria volátil del microcontrolador Heading, Pitch, Roll, Presión, Luz y calcula el Checksum y envía esta información por el puerto serial hacia la unidad consola PC(maestro).

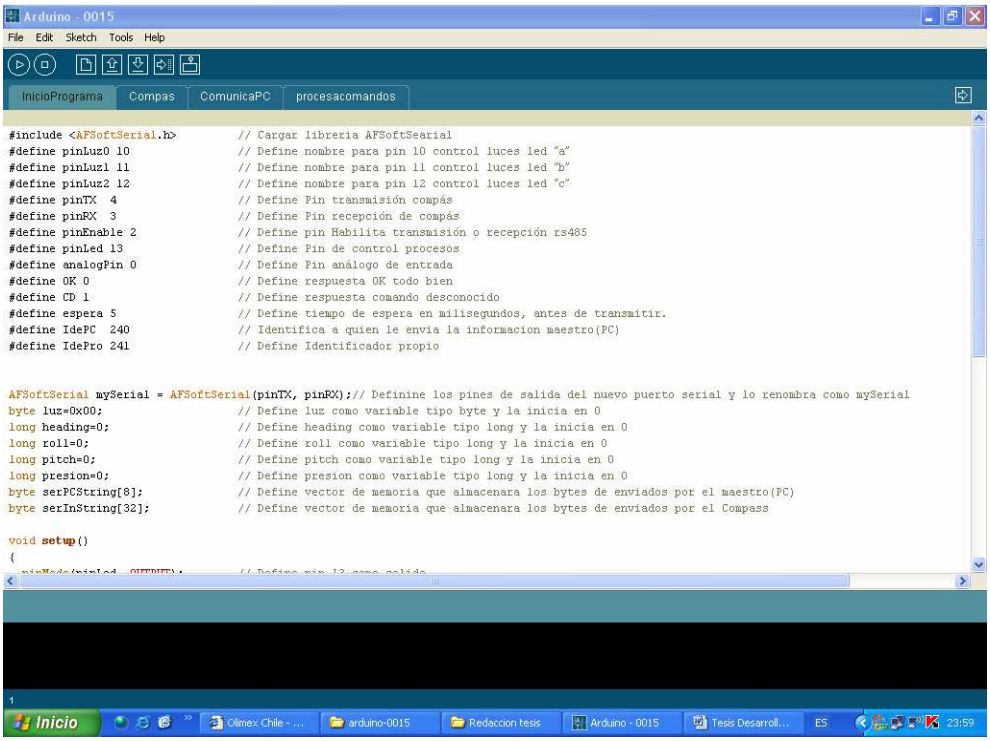


Figura Nº 6.1: Programa en Entorno de desarrollo Arduino

En el próximo diagrama se muestra la rutina principal del programa (modulo InicioPrograma) junto con las subrutinas que se van llamando a medida que se ejecuta el programa en el microcontrolador.

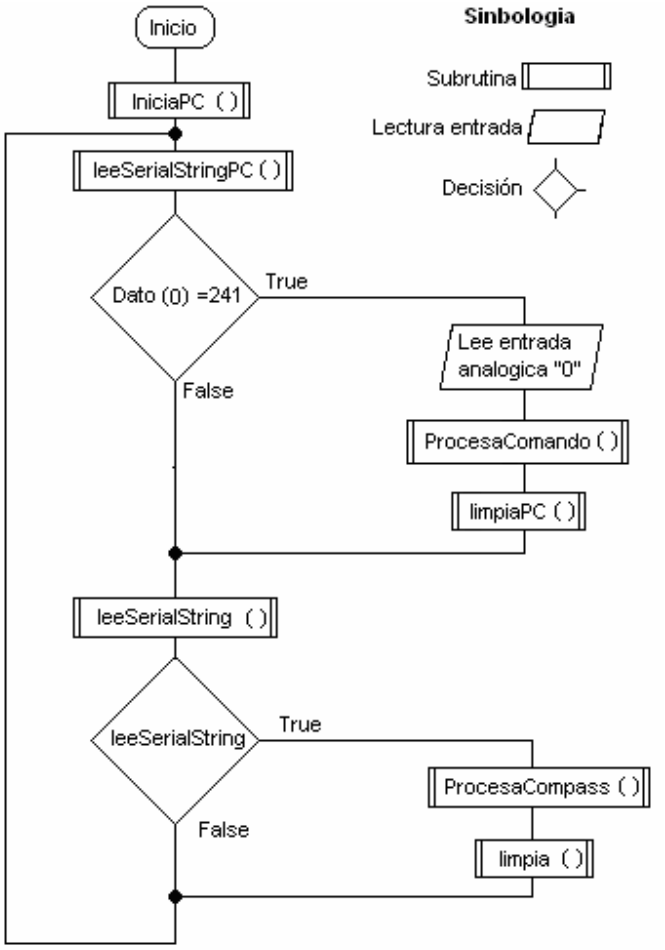


Figura Nº 6.2: Diagrama flujo modulo InicioPrograma

El programa desarrollado, por su extensión, se encuentra explicado con sus respectivos módulos en el anexo 2 de este trabajo (página 79).

6.4 Software Terminal V1.9b

Para poder ver la respuesta que tiene el sistema, se utilizara el programa **Terminal V1.9b** el cual es más completo y amigable que el Hyperterminal de Windows. Se puede bajar en forma gratuita en el siguiente enlace: http://www.surveyor.com/srvdownload/termv19b.zip.

Este programa permite configurar fácilmente el puerto de comunicación del PC, fijar su velocidad, los BITs de datos, la paridad etc. En la parte intermedia está la recepción de datos que llegan al puerto del PC, los que pueden ser vistos simultáneamente en decimal, hexadecimal, binario o String (código ASCII), la parte inferior está destinada para el envío de información, con la opción para enviar datos cada ciertos milisegundos.

6.5 Pruebas de comunicación

Una vez realizado una parte del programa se cargaba en el microcontrolador y se le realizaban varias pruebas para comprobar su funcionamiento.

Utilizando el programa Terminal V1,9b se vio la respuesta entregada por el microcontrolador frente a los comandos enviados por el puerto serial, a medida que se comprobando el buen funcionamiento de un proceso se pasaba a otro y así sucesivamente hasta alcanzar el grado de complejidad del programa.

En el transcurso de este proceso se encontraron variados problemas, el más significativo fue la captura de datos del compás electrónico, donde del 100% de datos recibidos el 2 % llegaba con error (considerando que el compás quedó entregando 4 muestras/segundo). Ésto se debe principalmente a que el puerto de comunicación que captura los datos del compás esta basado en una librería y que convierte dos salidas digitales en un puerto serial, por lo cual es comprensible el error. Para solucionar este problema hubo que implementar la corroboración del Checksum enviado por el compás electrónico y con esto se solucionaron los contínuos reseteo que presentaba el microcontrolador ATmega168.

A continuación se muestra una imagen del programa Terminal V1.9b donde se aprecian los datos recibidos en código ASCII (programa de prueba creado para ver los

datos, en String, que eran recibidos) y más abajo se aprecia la misma información pero ésta viene representada por el protocolo de comunicación que se había implementado, en lo que respecta a los números negativos se explica en la pág. 65.

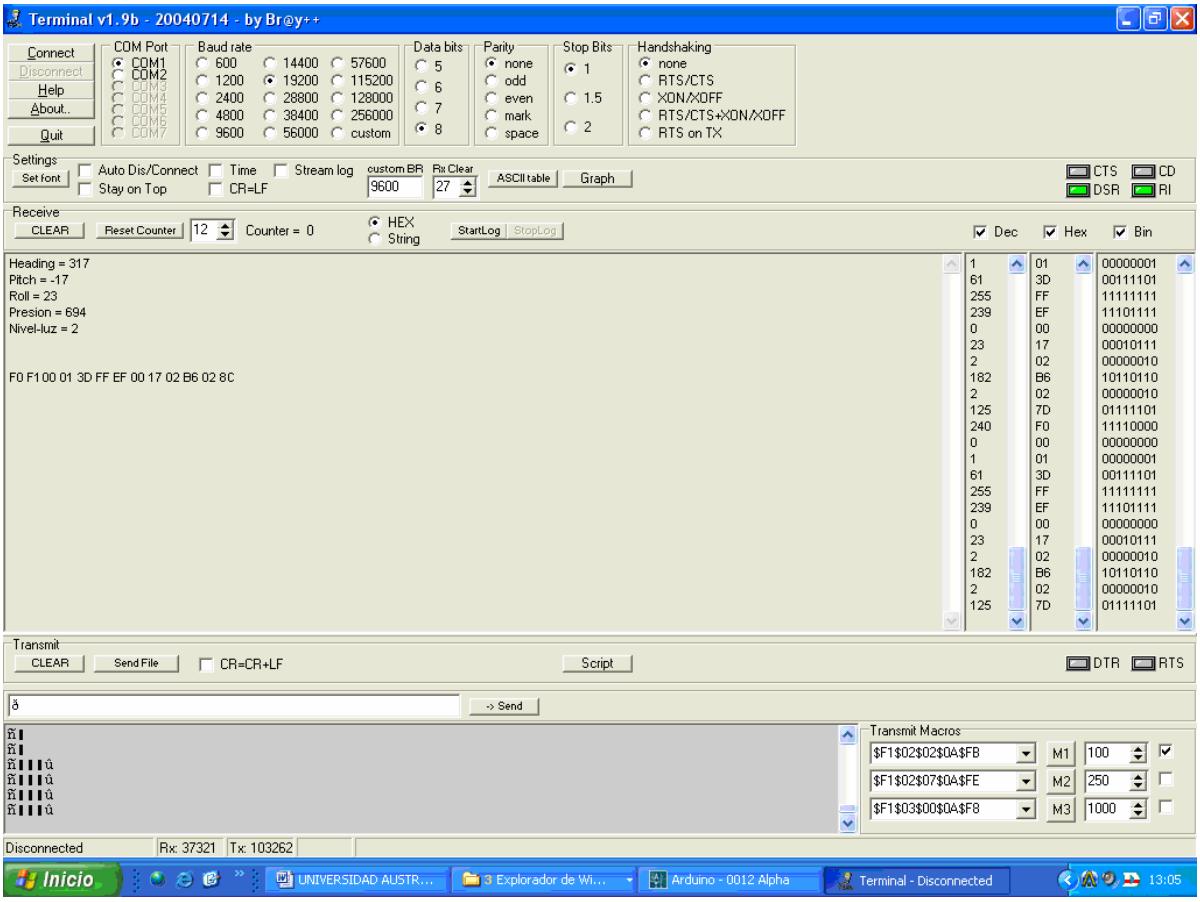


Figura Nº 6.3: Programa Terminal V1.9b

Una vez comprobado el correcto funcionamiento del sistema, de acuerdo al protocolo creado, se pasó a la nueva etapa que corresponde a la creación de un programa de prueba para el maestro-PC.

CAPITULO VII
DESARROLLO DEL PROGRAMA PARA EL MAESTRO (PC).

7.1 Preliminar

Para poder visualizar la información entregada por el sistema es preciso desarrollar un programa de prueba con ambiente amigable que nos interprete los datos enviados por la unidad remota y que a la vez sea capaz de enviar los comandos para solicitar datos y controlar el nivel de iluminación.

Como ya esta definido el protocolo de comunicación ahora hay que encontrar la fórmula que convierta a metros de profundidad los datos recibidos desde la unidad remota e interpretar lo numeración negativa.

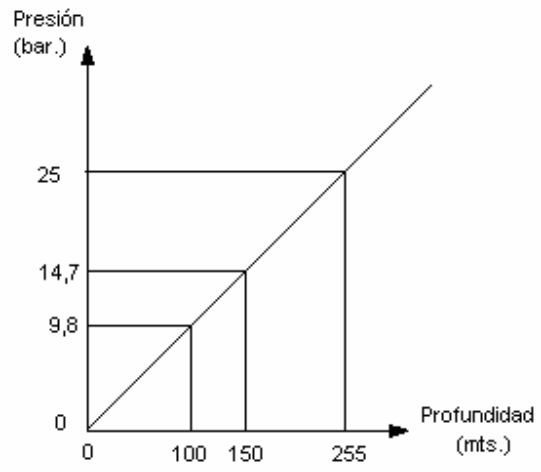
7.1.1 Ecuación para convertir a metros de profundidad

Como el conversor análogo a digital del microcontrolador tiene una resolución de 10 bit, por lo tanto, este conversor puede convertir un voltaje de 0-5V en valores en binario de 0000000000 a 1111111111 equivalente a 0 a 1023 en decimal y como se esta utilizando un conversor de corriente a voltaje que toma la señal de 4-20ma y la convierte en una señal de 1-5V, en el microcontrolador se almacenara en sus registro valores que irán de 0011001100=204 hasta 1111111111=1023 y que representaran el voltaje capturado.

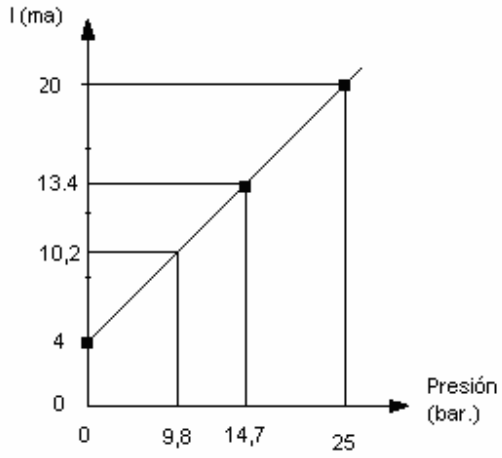
En lo que respecta al sensor de presión, éste entrega una señal que está comprendida entre 4ma para 0 bar y 20ma para 25bar, también se sabe que a 100 metros de profundidad la presión de 9.8 bar aproximadamente, por lo tanto, este sistema entregará una medición máxima de 14,7bar para los 150 metros. Si se realiza una ecuación sencilla encontraremos que el sensor a utilizar puede ser usado para entregar lectura de profundidad de hasta 255 metros (dato importante). A continuación se muestra el cálculo para esta afirmación.

100 metros = 9.8 bar
X metros = 25 bar
Despejando tenemos que: X = (25 * 100)/ 9.8 = 255,1 metros

A continuación se presentan las gráficas que dan a conocer la relación profundidad versus presión, como también la gráfica del sensor de presión en su conversión a ma.



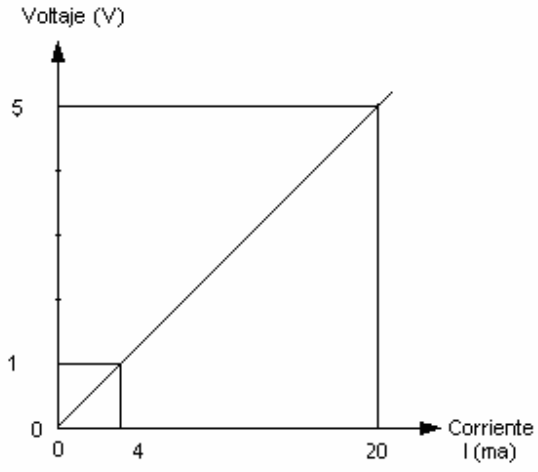
Grafica N° 7.1
Profundidad versus presión



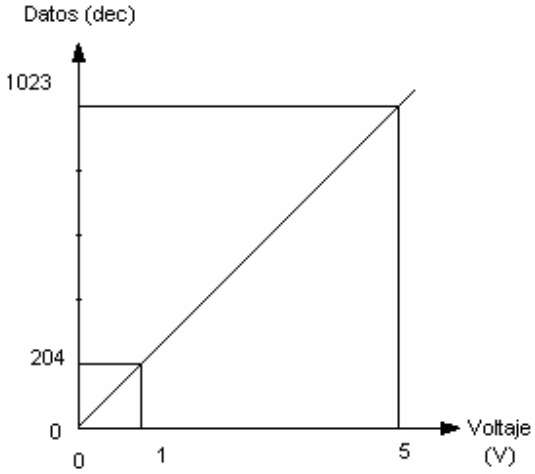
Grafica N° 7.2
Presion versus corriente (ma)

Ecuación sensor de presión (grafica 7.2): Corriente (ma) = 0,64 * (presión (bar)) + 4

Con estos datos construiremos nuestras graficas que nos representaran las señales con las cuales estamos trabajando y que en definitiva nos darán la formula final de conversión a metros de profundidad.

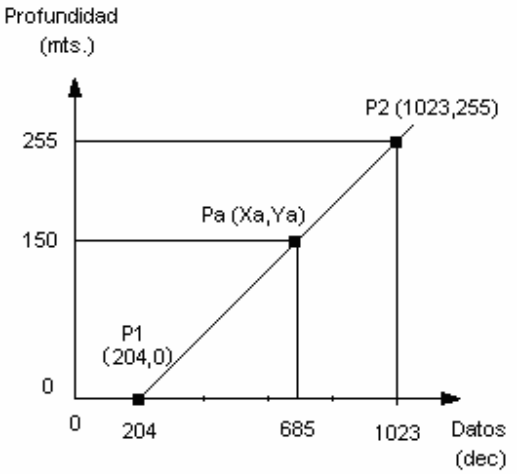


Grafica N° 7.3
Corriente versus voltaje (4-20ma a 1-5V)



Grafica N° 7.4
Voltaje versus datos decimal
(conversor analogo/digital microcontrolador)

De las graficas mostradas la que nos interesa en este memento es la grafica numero 5 que representa la información que se entregara desde el microcontrolador al PC y que es la información que se debe convertir a metros de profundidad. Para ello necesitamos primero encontrar la pendiente de la recta, donde utilizamos la siguientes fórmulas.



Grafica N° 7.5
Dato decimal versus profundidad

Calculo pendiente de la recta:

$m= (Y_2-Y_1)/(X_2-X_1)$

$P1= (X_1, Y_1) = (204,0)$

$P2= (X_2, Y_2) = (1023,255)$

$m= (255-0)/(1023-204) =255/819$

Posteriormente se busca la ecuación que representa a la recta. Para ello nos basamos en la ecuación $m= (Y_2-Y_1)/(X_2-X_1)$ donde Y2=Ya " representara nuestra incógnita profundidad", X2=Xa " representa la información que entrega el microcontrolador al PC", despejando la incógnita "Ya" obtenemos:

$m (X_a - X_1) = (Y_a - Y_1)$

$Y_a = m (X_a – X_1) + Y_1$, Reemplazando en el punto P1 (204,0) obtenemos.

$Y_a = (X_a – 204) *255/819$

Por lo tanto la expresión será:

Metros de profundidad = (presión - 204) * 255 / 819

Nota: La variable "presión" corresponde al dato capturado por el conversor análogo/digital del microcontrolador.

7.1.2 Números negativos (binaria negativa).

Como el compás electrónico entrega en algún momento datos que son negativos, información que posteriormente es enviaran a la unidad consola, es aquí donde hay que tener claro como se realiza este procedimiento para poder representar esta información después en la interfaz grafica del PC.

Como las computadoras utilizan la representación binaria en “complemento a 2” para representar los números negativos, por lo tanto por un tema lógico de programación, al enviar números negativos desde el microcontrolador, éste automáticamente los envió siguiendo la siguiente lógica.

Ejemplo: Al enviar el numero -25 en dos bytes (16 bit) el microcontrolador envía el siguiente dato 1111111111100111 = FFE7

La representación de números positivos en “complemento a 2” sigue la misma regla del sistema signo magnitud que en definitiva no afecta al dato enviado, pero la representación de los números negativos en complemento a 2 se obtiene de la siguiente forma.

Primero se representa el número en forma binaria positiva, luego se obtiene el complemente a 1 del número binario mediante el cambio de los unos por ceros y viceversa, posteriormente al “complemento 1” se le suma uno y el resultado es la representación en el complemento 2.

Ejemplo

Obtener el “complemento a 2” del número negativo de 8 bits 00000101_2 (-5_{10}).

El equivalente en complemento a 1 es 11111010. El complemento a 2 del número es 11111011. Comprobando los pesos en decimal se puede demostrar la obtención del negativo del número inicial utilizando el método del complemento a 2:

$11111011_2 = (-128 + 64 + 32 + 16 + 8 + 0 + 2 + 1)_{10} = -5_{10}$

En la representación en complemento 2 el primer bit del lado más significativo se interpreta como el signo, siendo cero para números positivos y 1 para números negativos.

Para convertir de binario a un número decimal, primero se ve el peso del dato recibido, si el bit más significativo es 1 corresponde a un dato negativo y se realiza el proceso inverso anterior, se toma el valor en binario, se le aplica el complemento a 1 (operación Not) y se le suma 1 lógico (complemento a 2), con ésto obtenemos el valor numérico el cual vendría siendo negativo porque el bit más significativo era 1.

Una vez sabido ésto, nos dedicamos a la creación del programa que nos representará en un ambiente amigable la información enviada por la unidad remota.

7.2 Desarrollo del programa en Visual Basic 6

En la programación en Visual Basic 6 el envío de información no fue mayor problema (se encontró en Internet algunas rutinas, en foros de discusión, que se adaptaron y se hicieron funcionar sin mayores inconvenientes) el problema principal se presentó a la hora de capturar la información presente en el puerto serial del computador, primero se intentó trabajar netamente como viene preconfigurado el puerto serial para recibir datos (código ASCII) pero los comandos que se encontraron en Internet para tomar estos datos y pasarlos a hexadecimal no eran bien claros, por lo que se optó por cambiar la forma de captar la información, recibiéndola en Bytes, de esta forma se pudo comprender el proceder de Visual y se logró obtener los datos en un lenguaje comprensible y que fue fácilmente almacenado y convertirlo a otro formato para posteriormente representarlos en la pantalla grafica. El cambio realizado en Visual para captar los datos en byte tiene relación con el comando que se incorpora en el procedimiento Sub Form_Load().

```
MSComm1.InputMode = comInputModeBinary   'Captura de datos en Byte en puerto serial
MSComm1.InputLen = 1                     'Cantidad de Byte que se recuperan del puerto por leída
MSComm1.RThreshold = 13                  'Cantidad de Byte necesita en Puerto para un evento MSComm1_OnComm()
MSComm1.InBufferSize = 13                'Buffer disponible para almacenar bytes en el puerto
```

De esta forma al tener 13 bytes, en buffer, del puerto serial del PC se produce un evento OnComm() que realiza la lectura de información, si el primer byte leído

corresponde a “240- F0” se almacena los 12 byte restante disponible en puerto, de lo contrario se espera que se produzca otro evento nuevamente de 13 byte en los buffer.

A continuación se presenta una imagen de la interfaz de control de prueba que se realizó para representar los datos enviados por la unidad remota.

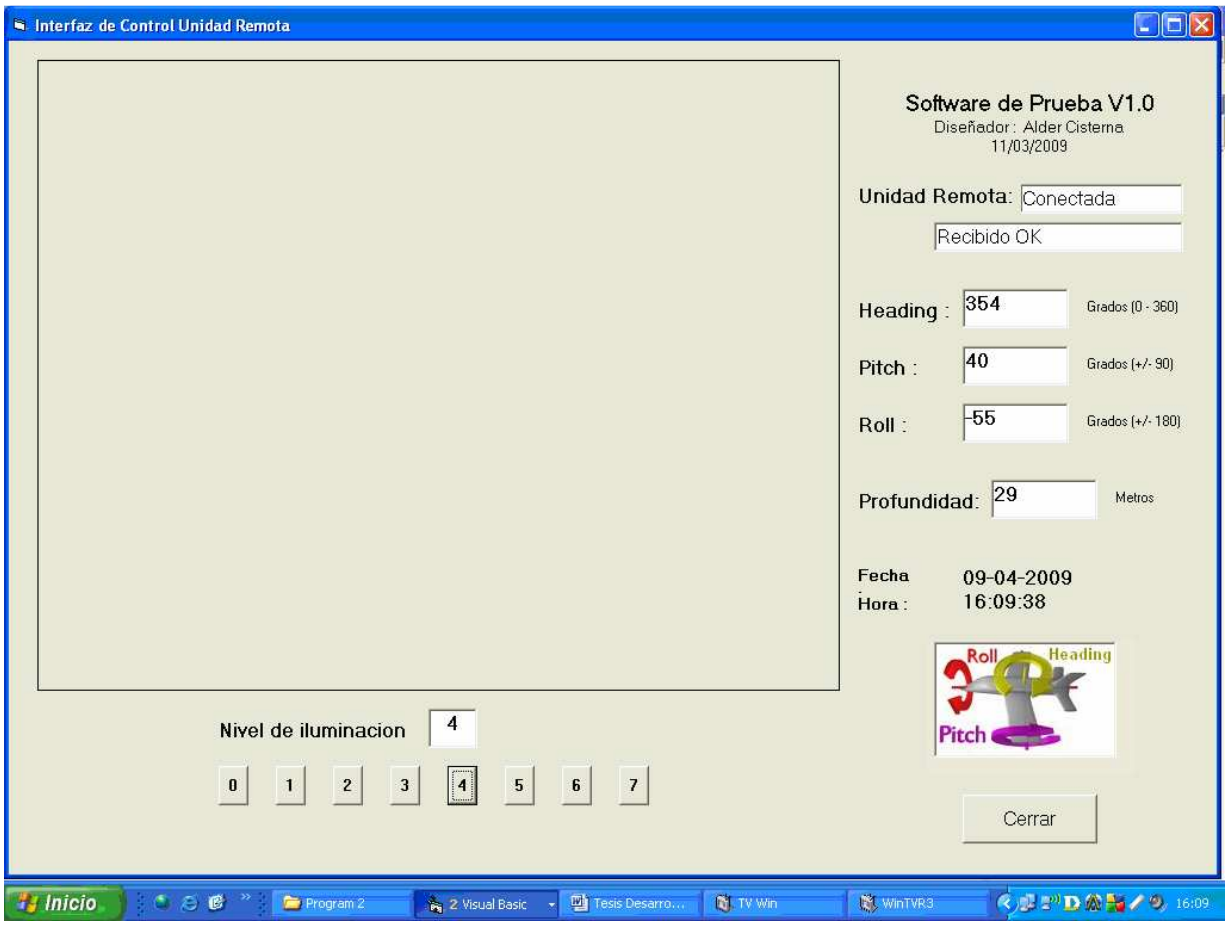


Figura Nº 7.1: Interfaz grafica del programa de prueba

Como referencia en el costado derecho inferior se ha incorporado una imagen que representa cada uno de las lecturas tomadas desde el compás electrónico, a modo de ejemplo se ha representado con un avión.

El programa desarrollado en Visual Basic se encuentra detallado en el anexo 3 de este trabajo (pagina 84).

7.3 Visualización de la imagen de video en el computador (PC)

Para poder visualizar la imagen de video que nos envía la unidad remota, se utilizará una tarjeta de televisión con entrada RCA (para ranura PCI), la tarjeta elegida es de la marca PCtronix S801T(F)TV, tarjeta de momento disponible.

Con el programa creado en Visual Basic y la tarjeta con entrada RCA se comprobará que la unidad está respondiendo a los comandos enviados desde la unidad consola como también se podrá visualizar la información recibida y comprobar que la imagen de video no sufre perturbaciones.

Lo referente a grabar los acontecimientos junto con la imagen de video es tema que no se abordará en este proyecto.

A continuación se presenta la tarjeta de televisión que se utiliza para recibir la imagen de video y presentarla en la pantalla del computador, está claro que existe alternativas mucho mejores pero con ésta opción se lograra el objetivo planteado.

Figura Nº 7.2: Tarjeta con entrada RCA para capturar video

CAPITULO VIII
PRUEBAS FINALES Y COSTO DEL PROYECTO

8.1 Consumo de energía del prototipo.

La corriente que consume la unidad renota (al 100 % de su funcionamiento) es de aproximadamente 660ma, si sumamos a esto el consumo de la unidad remota nuestra corriente es de 680ma. aprox. al multiplicar lo anterior por 24 V tenemos que la potencia aproximada consumida total es de 16.32 Watt. Este cálculo fue realizado tomando en cuenta que la corriente del sistema tiende a mantenerse gracias a los reguladores de voltaje, (unidad remota al 100% de su funcionamiento y constante).

Al variar el largo del cable lo que varía son las caídas de tensión que se producen en el cable y en los reguladores de voltaje de entrada de la unidad remota, al subir una, la otra baja, siempre y cuando no se sobrepase el voltaje mínimo de entrada de los reguladores de voltaje.

8.2 Pruebas finales de comunicación.

En el prototipo, para probar la funcionalidad del sistema, no es muy práctico tener un compresor para inyectar presión al sensor de presión, por tal motivo, se diseñó un circuito que realice la misma función. El circuito creado controla la corriente del lazo entre 4ma y 17ma que es suficiente para representar los 150mts de profundidad, consta de un potenciómetro de 2kΩ, más dos resistencias en serie y un pequeño potenciómetro de ajuste de 100Ω, la idea es dejar pasar 4ma como mínimo para “0” metros de profundidad de 13,4ma para los 150 metros con lo cual se comprobara el funcionamiento de esta etapa.

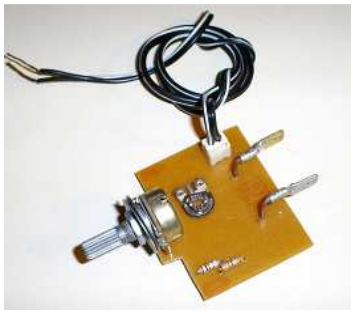


Figura Nº 8.1: Circuito de prueba

Con este circuito se logra comprobar las gráficas mostradas anteriormente mediante el programa desarrollado en Visual Basic.

Por otra parte se pudo obtener correctamente los datos que entrega el compás electrónico, controlar el nivel de iluminación de la unidad remota y visualizar sin ninguna perturbación la imagen de video que es capturada en la unidad remota (aunque la imagen de video que se presenta a continuación es solo referencial)

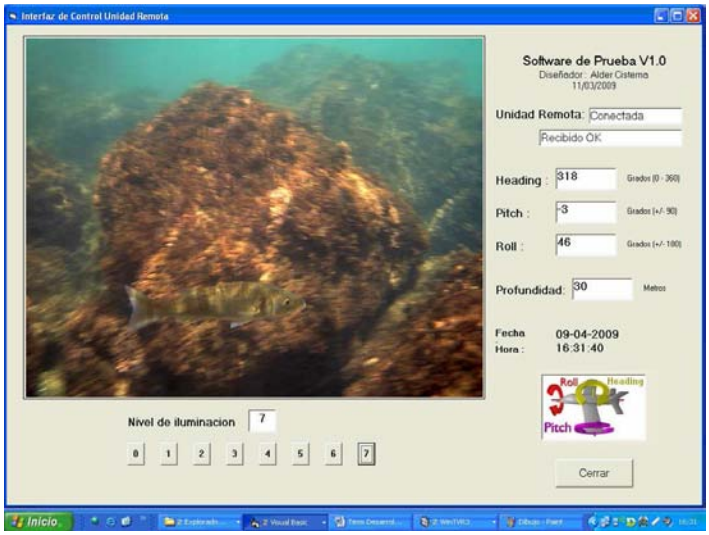


Figura Nº 8.2: Programa terminado y funcionando

8.3 Costos materiales del proyecto

Ahora que se ha dado por terminado la parte práctica se puede dar a calcular el costo económico para la construcción del prototipo.

Costo materiales unidad control y comunicación	$ 14.180 C/IVA
Costo materiales unidad conversora de video (unidad remota)	$ 5.150 C/IVA
Costo materiales para focos de iluminación (x Foco)	$ 8.530 C/IVA
Costo materiales conversor Rs232 a RS485 + conversor de video	$ 13.960 C/IVA
Costo tarjeta con entrada RCA para ranura PCI	$ 23.000 C /IVA
Sensor de presión 0 a 25 Bar	***$ 180.000 C/IVA***
Brújula electrónica Ocean Server OS5000-T 3 Axis	***$ 177.000 C/IVA***

Por lo tanto el prototipo tendrá un costo aproximado de 430.350 pesos sin incluir el **costo del desarrollo del proyecto**.

CONCLUSIONES.

El diseño y construcción del prototipo antes detallado fue posible gracias a la experiencia adquirida a través del tiempo, tanto en mis años de estudio de universidad como en el desempeño laboral; llegando a comprender, de ésta forma, el funcionamiento de componentes electrónicos como variables existentes en el entorno.

El presente trabajo represento un desafío que me permitió comprender mejor los sistemas de comunicación que se utilizarán normalmente en el medio industrial.

Con respecto a este proyecto debo indicar que se alcanzaron los objetivos propuestos, transmitir información e imágenes de calidad, gracias al diseño planteado y a la correcta elección de componentes.

En lo que respecta al conversor de RS-232 a RS-485 debo indicar que la opción elegida es una buena alternativa para llevar a cabo esta labor, ya que este circuito no necesita trabajar con las señales de control enviadas por el puerto serial del computador. Es importante indicar que en el programa cargado en el microcontrolador hubo que habilitar el chip MAX485 unos 5 milisegundos antes de iniciar la transmisión y deshabilitar unos 5 milisegundos después de enviar el último dato, para asegurar la integridad de la trama.

En cuanto a los focos de iluminación, éstos respondieron sin problemas a los cambios de nivel solicitado, tomando en cuenta que para controlar los 8 niveles sólo se utilizaron tres líneas de control (+Gnd) hacia los focos. Por otro lado la luz que se emitía no produjo distorsión en la imagen captada por la cámara de video.

Como se pudo apreciar se redujo bastante la placa que contiene el sistema electrónico, gracias al programa Eagle 5.3 que permitió diseñar las pistas por ambos lados de la placa circuito impreso.

Con respecto al conversor 4-20ma a 1-5 V se puede decir que la idea mostrada es la más simple que se puede construir, lo más importante que se requiere es que la resistencia de 250Ω sea de baja tolerancia (alta precisión) con esto obtenemos lectura de voltaje proporcionales y acordes con la corriente de entrada.

Con respecto a la brújula electrónica debo indicar que ésta parte fue la más complicada ya que había que implementar otro puerto serial mediante una librería,

capturar la información enviada, almacenar los datos en memoria para luego retransmitirlos. En principio esta etapa reseteaba continuamente el microcontrolador por lo que hubo que comparar el Checksum enviado por la brújula electrónica con el calculado al decepcionar los datos, con esto se soluciona el problema antes descrito.

En cuanto al protocolo de comunicación, se presenta un modelo que permite comunicarse con más de un esclavo, donde la trama que responde este contiene la dirección del maestro y del propio esclavo que esta enviando la información. También es importante destacar que este proyecto fue pensado para que el maestro pueda manejar múltiples esclavos, por lo tanto, cualquier otra función que se quisiera realizar en la unidad remota bien puede ser controlada por otros microcontroladores dividimos las tareas a realizar como también se dispone de mayor cantidad de salidas/entradas y la velocidad de trabajo aumenta.

Luego de haber realizado mi tesis me siento con mas confianza en mi mismo para realizar cualquier otro desafío, también puedo indicar que este trabajo puede ser el inicio para un proyecto de mayor envergadura, donde al sistema anterior se le pueden incluir algunos motores para el desplazamiento en el fondo marino, como también algún GPS (conectado al PC) para registrar el lugar donde se grabaron los acontecimientos y de esta manera tener un sistema más completo. Para realizar funciones, como la antes descrita, Arduino ya esta preparado para ello, incorporando librerías (programas precreados) que permiten con un par de líneas de comando controlar procesos más complejos.

REFERENCIA BIBLIOGRAFÍA.

Sitios Web:

Arduino y su entorno de desarrollo (fecha acceso 16-01-2009):
http://arduino.cc
http://arduino.cc/es/Secundaria/Referencia
http://www.arduino.cc/en/Main/Software
http://electrolabo.com/arduino/Tutorial%20Arduino%2001%20-%20presentaci%f3n.pdf
http://www.scribd.com/doc/3964585/hortanet

Norma rs232 (fecha acceso 10-11-2008):
http://perso.wanadoo.es/pictob/comserie.htm#la_norma_rs232

Conversor de 4-20ma a 1-5V (fecha acceso 10-10-2008):
http://aquaticus.info/pressure_sensor

Fabricante compás electrónico Ocean-Server (fecha acceso 10-10-2008):
http://www.ocean-server.com/compass.html

Información de Baluns (fecha acceso 10-01-2008):
http://www.rnds.com.ar/articulos/030/RNDS_134W.pdf

Fabricante de led alto brillo (fecha acceso 10-10-2008):
http://www.besthongkong.com/product_info.php?products_id=132

Información transistor BC547 (fecha acceso 10-10-2008):
http://www.datasheetcatalog.net/es/datasheets_pdf/B/C/5/4/BC547.shtml

Interpretación de números negativos en binario (fecha acceso 10-10-008):
http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/010301.htm

Circuito conversor Rs232 a Rs485 (fecha acceso 10-11-008):
http://www.circuit-projects.com/converter-circuits/rs232-rs485-converter-with-automatic-rx-tx-control.html

Libro

Fco. Javier Ceballos, Curso de programación de Visual Basic 6, ed., Alfaomega Grupo Editor, Col Del Valle, México, 2000.

Anexos 1: "Alternativa conversor de 4-20ma a 0-5V"

Este conversor es una buena alternativa para obtener voltajes de salida de 0-5V, el inconveniente es que necesita un amplificador operacional.

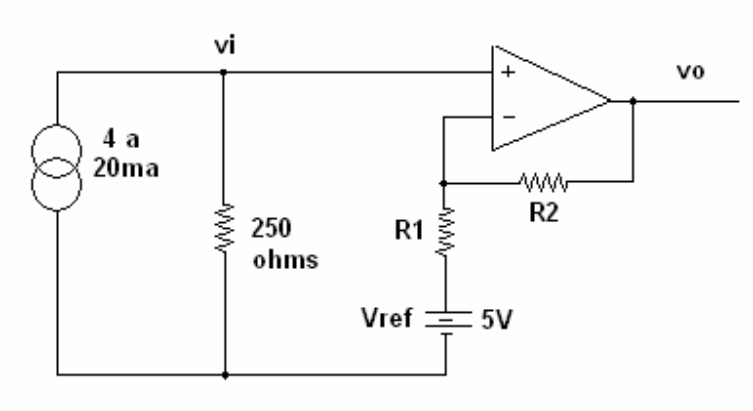


Tabla Nº A1.1: Conversor 4-20ma a 0-5 volt

Ecuación representativa del esquema:

Vo = (R2/R1+1)*VI – Vref *R2/R1 (para Vi=1 implica que Vo=0)
0 = R2/R1 * 1 + 1 – 5 * R2/R1
0 = 1 – 4 R2/R1

R2/R1= ¼ Calculando R1 y R2, para una resistencia R2 de 2,5KΩ implica R1=10KΩ

Por lo tanto la ecuación anterior queda
Vo = (¼+ 1) * Vi – 5* ¼
Vo= 1,25 Vi – 1,25

(Comprobación: para Vi=1 implica que Vo=0; para Vi=5 implica que Vo=5)

Anexo 2: “Programa Cargado en el Microcontrolador ATmega168”

Módulo InicioPrograma

El modulo inicioPrograma comienza cargando una librería para habilitar otro puerto de comunicación y luego define algunos nombres para usarse en todo el programa, de tal forma si hay que modificar algún pin de salida o valor de algún identificador, sólo se realiza aquí y cambia en toda la programación. Las librerías y variables se cargar antes del **void setup ()** para que sean globales.

A continuación se detalla la programación del modulo InicioPrograma.

```
#include <AFSoftSerial.h>          // Cargar librería AFSoftSearial
#define pinLuz0 10                 // Define nombre para pin 10 control luces led “a”
#define pinLuz1 11                 // Define nombre para pin 11 control luces led “b”
#define pinLuz2 12                 // Define nombre para pin 12 control luces led “c”
#define pinTX  4                   // Define Pin transmisión compás
#define pinRX  3                   // Define Pin recepción de compás
#define pinEnable 2                // Define pin Habilita transmisión o recepción rs485
#define pinLed 13                  // Define Pin de control procesos
#define analogPin 0                // Define Pin análogo de entrada
#define OK 0                       // Define respuesta OK, todo bien
#define CD 1                       // Define respuesta comando desconocido
#define espera 5                   // Define tiempo de espera en milisegundos, antes de transmitir.
#define IdePC  240                 // Identifica a quien le envía la información maestro (PC) “F0”
#define IdePro 241                 // Define Identificador propio “F1”


AFSoftSerial mySerial = AFSoftSerial(pinTX, pinRX);// Define los pines de salida del nuevo puerto serial, lo renombra como mySerial
byte luz=0x00;                     // Define luz como variable tipo byte y la inicia en 0
long heading=0;                    // Define heading como variable tipo long y la inicia en 0
long roll=0;                       // Define roll como variable tipo long y la inicia en 0
long pitch=0;                      // Define pitch como variable tipo long y la inicia en 0
long presion=0;                    // Define presión como variable tipo long y la inicia en 0
byte serPCString[8];               // Define vector de memoria que almacenara los bytes de enviados por el maestro(PC)
byte serInString[32];              // Define vector de memoria que almacenara los bytes enviados por el Compás

void setup()
{
  pinMode(pinLed, OUTPUT);         // Define pin 13 como salida
  pinMode(pinLuz0, OUTPUT);        // Define pin 10 como salida
  pinMode(pinLuz1, OUTPUT);        // Define pin 11 como salida
  pinMode(pinLuz2, OUTPUT);        // Define pin 12 como salida
```

```
  pinMode(pinEnable,OUTPUT);          // Define pin 2 como salida
  Serial.begin(19200);                // Inicializa puerto serial 1, para comunicación con PC
  mySerial.begin(9600);               // Inicializa puerto serial 2, para comunicación con compás
  IniciaPC();                         // Inicia comunicación con PC, indicando conexión al encender unidad remota
}

void loop()                           // Inicia Bucle
{
  leeSerialStringPC(serPCString);     // Verifica datos de entrada (comunicación con maestro), si los hay llena el arreglo de
                                      // entrada verificando el checksum
  if (serPCString[0]==IdePro){        // Verifica  primer dato guardador en vector de memoria, si es 241 entra a la sentencia
    presion =analogRead(analogPin);   // Lee la entrada analógica, sensor de presión
    ProcesaComando(serPCString);      // Procesa la información de la trama recibida
    limpiaPC(serPCString,8);          // borra la información guardada en memoria
  }
  if(leeSerialString(serInString)){   // Verifica datos de entrada lectura compás, si hay los almacena chequeando el checksum
    ProcesaCompass(serInString);      // Realiza un arreglo de la información compás, identificando heading, roll, pich
    limpia(serInString,32);           // borra la información guardada en memoria
  }
}                                     // Fin - retorna a inicio Bucle
```

Una vez que el programa llega a **void loop ()** empieza a ejecutarse el bucle que se repetirá indefinidamente. Al pasar a esta etapa el programa llama a la subrutina leeSerialStringPC(serPCString) para ver si hay datos para leer del puerto serial, si es así verifica el comienzo de la trama detectando el primer octeto F1=240 y comienza a almacenar los datos en un arreglo en la memoria volátil, una vez terminado éste procedimiento verifica que el código de chequeo checksum es correcto- Si todo está bien, continúa el programa, de lo contrario borra toda su memoria y le indica a la unidad remota que el comando que se le envió es desconocido. Posteriormente si el primer dato almacenado en el primer registro de la memoria es IdePro=F1, lee el puerto analógico y llama a la subrutina ProcesaComando(serPCString) para que verifique si la trama es una petición de datos o se solicita cambiar el nivel de iluminación.

Módulo ComunicaPC

El módulo ComunicaPC posee las subrutinas que tienen que ver con la comunicación con el PC-maestro, donde se le indica al computador que la unidad

remota esta conectada “subrutina iniciaPC()”, también se encuentra la subrutina leeSerialStringPC (serPCString) que almacena la información recibida del PC-maestro.

```
void IniciaPC(void)                    // Inicio Subrutina IniciaPC
{
  digitalWrite(pinEnable,HIGH);        // Habilita entrada chip 485 para transmitir
  delay(espera);                       // Espera 5 ms para enviar el primer dato
  Serial.print(IdePC,BYTE);            // Envía dirección destino trama
  Serial.print(IdePro,BYTE);           // Envía la dirección propia de la unidad
  for (int i=0; i<10; i++){
    Serial.print(0,BYTE);              // Rellena con ceros para completar la trama
  }
  Serial.print(01,BYTE);               // Envía el Cheksum, en este caso es 1
  delay(espera);                       // Espera 5 ms
  digitalWrite(pinEnable,LOW);         // Deshabilita la transmisión en Chip 485(habilita recepción)
}                                      // Fin Subrutina IniciaPC

byte leeSerialStringPC(byte *strArray) {// Inicia Subrutina leeSerialStringPC
  byte val = 0;                        // Establece e inicializa variable
  byte checksum = 0;                   // Establece e inicializa variable
  byte bytesread = 0;                  // Establece e inicializa variable

  if(Serial.available() > 0) {         // Ve si hay datos para leer en entrada puerto serial 1(principal)
    if((val = Serial.read()) == IdePro) {  // Lee un dato del puerto serial 1 y compara si la trama es para esta unidad
      strArray[0] = val;               // Almacena dato en vector de memoria (0)
      bytesread++;                     // Aumenta en 1 el valor de la variable bytesread
      checksum ^= strArray[0];         // Realiza operación Xor para ir calculando el Checksum
      while (bytesread < 5) {          // Bucle, se realizara mientras variable bytesread sea menor que 5
        if( Serial.available() > 0) {  // Ve si hay datos para leer en entrada puerto serial 1
          val = Serial.read();         // Lee un dato del puerto serial 1
          strArray[bytesread] = val;   // Almacena dato en vector de memoria
          if(bytesread < 4){           // Compara el valor de bytesread
          checksum ^= strArray[bytesread];    // Realiza operación Xor para seguir calculando el Checksum
          }
          bytesread++;                 // Aumenta en 1 la variable bytesread
        }
      }
      if ((checksum==strArray[4])&&(strArray[3]==10)) // Sentencia se realiza si checksum calculado es igual al checksum recibido,
      {                                // como también debe de cumplirse que código relleno (03) sea 0A
      }
      else                             // Salto realizado en función if anterior
      {
        ReturnPC(CD);                  // Subrutina responderá al maestro(PC) que dato recibido era desconocido
```

```
      for (int i=0; i<6; i++){          // Borra los datos guardados, asta el momento, ya que la trama estaba corrupta
        strArray[i] = 0;                // Borra datos
      }
    }
    Serial.flush();                     // limpia el buffer de entrada de cualquier otro dato existente en puerto 1 que no se leyó
  }
 }
}                                       // Fin Subrutina leeSerialStringPC


void ReturnPC(byte valor)               // Inicia Subrutina ReturnPC
{
 digitalWrite(pinEnable,HIGH);          // Habilita entrada chip 485 para transmitir
 delay(espera);                         // Espera 5 ms para enviar el primer dato
 Serial.print(IdePC,BYTE);              // Envía dirección destino trama
 Serial.print(IdePro,BYTE);             // Envía la dirección propia de la unidad
 Serial.print(valor,BYTE);              // Envía dato "valor" a unidad maestra
 for (int i=0; i<10; i++){
  Serial.print(0,BYTE);                 // Rellena con ceros para completar la trama
 }                                      // En este caso Cheksum enviado es 0
 delay(espera);                         // Espera 5 ms
 digitalWrite(pinEnable,LOW);           // Deshabilita la transmisión en Chip 485(habilita recepción)
}                                       // Fin Subrutina ReturnPC

void limpiaPC(byte *strArray,int j){    // Inicio Subrutina limpiaPC
 for (int i=0; i< j; i++){
  strArray[i]=0;                        // Borra los datos almacenados en memoria
 }
}                                       // Fin Subrutina limpiaPC
```

Módulo Procesacomandos

Una vez que el sistema ha almacenado los datos en su memoria volátil, llama a la subrutina ProcesaComando(serPCString) la que verifica si la petición es sólo de datos o se solicita modificar el nivel de iluminación.

```
void ProcesaComando(byte *strArray) // Inicio Subrutina ProcesaComando
{
 int brillo=0;                          // Establece e inicializa variable
 byte highByte, highestByte,checksum; // Establece variables
```

```
switch(strArray[1]){                    // Analiza segundo dato de la trama recibida, si es una solicitud de datos o un cambio nivel luces
case 1:                                 // solicitud envío de datos
 {
   checksum=0;                          // Inicializa la variable Checksum en cero
   digitalWrite(pinEnable,HIGH);        // Habilita entrada chip 485 para transmitir
   delay(espera);                       // Espera 5 ms antes de enviar el primer dato
   Serial.print(IdePC,BYTE);            // Envía dirección destino trama
   checksum ^=IdePC;                    // Realiza operación Xor para ir calculando el Checksum a enviar
   Serial.print(IdePro,BYTE);           // Envía la dirección propia de la unidad
   checksum ^=IdePro;                   // Realiza operación Xor para ir calculando el Checksum a enviar
   Serial.print(OK,BYTE);               // Envía código indicando recibo ok de trama de solicitud de datos
   checksum ^=OK;                       // Realiza operación Xor para ir calculando el Checksum a enviar

   highByte = ((heading >> 8) & 0xFF); // Convierte el valor de la variable heading a bit y toma 1 byte a partir del 8 bit para
                                        // almacenarlo en highByte
   highestByte = (heading & 0xFF);      // Convierte el valor de la variable heading a bit y toma 1 byte para almacenarlo
                                        // en highestByte, desborde se pierde
   Serial.print(highByte,BYTE);         // Envía la variable highByte por puerto serial 1
   checksum ^=highByte;                 // Realiza operación Xor para ir calculando el Checksum a enviar
   Serial.print(highestByte,BYTE);      // Envía la variable highestByte por puerto serial 1
   checksum ^=highestByte;              // Realiza operación Xor para ir calculando el Checksum a enviar

   highByte = ((pitch >> 8) & 0xFF);    // Convierte el valor de la variable pitch a bit y toma 1 byte a partir del 8 bit para
                                        // almacenarlo en highByte
   highestByte = (pitch & 0xFF);        // Convierte el valor de la variable pitch a bit y toma 1 byte para almacenarlo en
                                        // highestByte, desborde se pierde
   Serial.print(highByte,BYTE);         // Envía la variable highByte por puerto 1
   checksum ^=highByte;                 // Realiza operación Xor para ir calculando el Checksum a enviar
   Serial.print(highestByte,BYTE);      // Envía la variable highestByte por puerto 1
   checksum ^=highestByte;              // Realiza operación Xor para ir calculando el Checksum a enviar

   highByte = ((roll >> 8) & 0xFF);     // Convierte el valor de la variable roll a bit y toma 1 byte a partir del 8 bit para
                                        // almacenarlo en highByte
   highestByte = (roll & 0xFF);         // Convierte el valor de la variable roll a bit y toma 1 byte para almacenarlo en
                                        // highestByte, desborde se pierde
   Serial.print(highByte,BYTE);         // Envía la variable highByte por puerto 1
   checksum ^=highByte;                 // Realiza operación Xor para ir calculando el Checksum a enviar
   Serial.print(highestByte,BYTE);      // Envía la variable highestByte por puerto 1
   checksum ^=highestByte;              // Realiza operación Xor para ir calculando el Checksum a enviar

   highByte = ((presion >> 8) & 0xFF); // Convierte el valor de la variable presión a bit y toma 1 byte a partir del 8 bit para
                                        // almacenarlo en highByte
   highestByte = (presion & 0xFF);      // Convierte el valor de la variable presión a bit y toma 1 byte para almacenarlo en
                                        // highestByte, desborde se pierde
   Serial.print(highByte,BYTE);         // Envía la variable highByte por puerto serial 1
```

```
   checksum ^=highByte;              // Realiza operación Xor para ir calculando el Checksum a enviar
   Serial.print(highestByte,BYTE);   // Envía la variable highestByte por puerto serial 1
   checksum ^=highestByte;           // Realiza operación Xor para ir calculando el Checksum a enviar

   Serial.print(luz,BYTE);           // Envía la variable highByte por puerto serial 1
   checksum ^=luz;                   // Realiza operación Xor para ir calculando el Checksum a enviar

   Serial.print(checksum, BYTE);     // Envía la variable checksum por puerto 1

   delay(espera);                    // Espera 5 ms
   digitalWrite(pinEnable,LOW);      // Deshabilita la transmisión en Chip 485(habilita recepción)

   break;                            // Sale de la sentencia case 1
  }
case 2:                              // Solicitud cambio nivel de luces
 {
   if ( (strArray[2] <0)||(strArray[2]>7)) // Sentencia compara el tercer dato de la trama verificando si esta fuera de rango
    ReturnPC(CD);                    // Se llama a subrutina ReturnPC
   else                              // en caso contrario se realiza función siguiente
   {
    switch(strArray[2]){             // Analiza el tercer dato de la trama recibida para ver que nivel de luces se solicita
    case 0:                          // Solicitud nivel luces 0
     {
       luz=0;                        // Almacena en variable luz el valor 0
       break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
     }
    case 1:                          // Solicitud nivel luces 1
     {
       luz=1;                        // Almacena en variable luz el valor 1
       break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
     }
    case 2:                          // Solicitud nivel luces 2
     {
       luz=2;                        // Almacena en variable luz el valor 2
       break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
     }
    case 3:                          // Solicitud nivel luces 3
     {
       luz=3;                        // Almacena en variable luz el valor 3
       break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
     }
    case 4:                          // Solicitud nivel luces 4
     {
       luz=4;                        // Almacena en variable luz el valor 4
       break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
     }
```

```
      case 5:                          // Solicitud nivel luces 5
       {
         luz=5;                        // Almacena en variable luz el valor 5
         break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
       }
      case 6:                          // Solicitud nivel luces 6
       {
         luz=6;                        // Almacena en variable luz el valor 6
         break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
       }
      case 7:                          // Solicitud nivel luces 7
       {
         luz=7;                        // Almacena en variable luz el valor 7
         break;                        // Indica a la estructura que pare de buscar en el resto de los casos, y finalice la función
       }
      default:                         // Si el valor no pertenecía a ningún case, hacer la parte de default
       {
         break;                        // Indica a la estructura que pare de buscar y finalice la función
       }
      }
      digitalWrite(pinLuz0,(luz & 1));  // Coloca el nivel correspondiente en el pin 10 control luces led “a”
      digitalWrite(pinLuz1,(luz & 2));  // Coloca el nivel correspondiente en el pin 11 control luces led “b”
      digitalWrite(pinLuz2,(luz & 4));  // Coloca el nivel correspondiente en el pin 12 control luces led “c”
    }
    break;                             // Indica a la estructura que pare de buscar y finalice la función
  }

 default:                              // Si el valor no pertenecía a ningún case, hacer la parte de default
  {
    ReturnPC(CD);                      // Se llama a subrutina ReturnPC
                                       // Indica a la estructura que pare de buscar y finalice la función
  }
 }
}                                      // Fin de Subrutina ProcesaComando
```

Módulo Compás

A continuación se presenta la programación que tiene relación con el compás electrónico. En este caso se presentará la recolección de los datos, la subrutina que ordena la información y la subrutina que limpia el registro.

```
byte leeSerialString(byte *strArray)          // Inicia Subrutina leeSerialString
{
  byte bytesread = 2;                         // Establece e inicializa variable
  int i=0;                                    // Establece e inicializa variable
  byte val=0;                                 // Establece e inicializa variable
  byte checksum=0;                            // Establece e inicializa variable
  char check[2];                              // Establece vector de memoria tipo char
  byte chec=0;                                // Establece e inicializa variable

  strArray[0]=0;                              // Primer byte de memoria se establece en 0
  strArray[1]=0;                              // Segundo byte de memoria se establece en 0
  if(mySerial.available()>0) {                // Verifica si hay datos para leer en entrada puerto serial 2
    if((val = mySerial.read()) ==0x24) {      // Lee un dato del puerto serial 1 y lo compara con signo $
      while(bytesread<32){                    // Bucle, se realizara mientras variable sea menor que 32
          val = mySerial.read();              // Lee puerto serial  y almacena valor en variable val
        strArray[bytesread] = val;            // Almacena dato leido en vector de memoria
        checksum ^=strArray[bytesread-2];     // Calcula el checksum de los datos capturados

        if(strArray[bytesread-2] =='*' ) {    // Compara dato, si es "*" entra a la sentencia
          check[0]=strArray[bytesread-2];     // Cambia de tipo de variable el dato guardado en vector de memoria
          check[1]=strArray[bytesread];       // Cambia de tipo de variable el dato guardado en vector de memoria
          strArray[bytesread-1]='\0';         // Coloca un termino a los datos guardados en vector de memoria
          chec= strtod (check,NULL,16);       // Transforma el String Check a dato hexadecimal

          if (checksum==chec)                 // Verifica si el Chec recibido es correcto con el checksum calculado
          {
            for (i=(bytesread); i<32; i++){   // Borra cualquier dato después de la información guardada
              strArray[i] = 0;
            }
            return(true);                     // Responde Verdadero a sentencia inicial consultora
            break;                            // Sale de la subrutina
          }
          else{
            for (int i=0; i<32; i++){         // Si checksum fuera distinto borra toda información guardada
              strArray[i] = 0;
            }
          }
        }
            bytesread++;                      // Aumenta en 1 la variable
        }
    }
  }
  return(false);                              // Responde falso a sentencia inicial consultora
}                                             // Fin subrutina leeSerialString
```

```
void ProcesaCompass(byte *strArray)          // Inicio Subrutina ProcesaCompass
{
  int HPos=-1;                               // Establece e inicializa variable
  int RPos=-1;                               // Establece e inicializa variable
  int PPos=-1;                               // Establece e inicializa variable
  int AsteriskPos=-1;                        // Establece e inicializa variable
  int i=0;                                   // Establece e inicializa variable
  char strHeading[7];                        // Establece vector de memoria tipo char
  char strPitch[7];                          // Establece vector de memoria tipo char
  char strRoll[7];                           // Establece vector de memoria tipo char

  while ((strArray[i]!='*')||(i<=31)){       // Bucle, se realizara mientras las variable sean verdaderas

    if(strArray[i]=='C')HPos=i;              // Guarda en variable  HPos la posición del string(ASCII) "C"
    if(strArray[i]=='P')PPos=i;              // Guarda en variable  PPos la posición del string(ASCII) "P"
    if(strArray[i]=='R')RPos=i;              // Guarda en variable  RPos la posición del string(ASCII) "R"
    i++;                                     // Aumenta en 1 la variable
  }
  AsteriskPos=i;                             // Almacena en variable AsteriskPos la posición, en memoria, del "*"

  for(i=HPos+1;i<PPos;i++){                  // Se realiza sentencia para buscar el valor, en string, de Heading
    strHeading[i-HPos-1]=strArray[i];        // Almacena en nuevo vector de memoria solo el string que esta entre C y P
      }
  for(i=PPos+1;i<RPos;i++){                  // Se realiza sentencia para buscar el valor, en string, de Pitch
    strPitch[i-PPos-1]=strArray[i];          // Almacena en nuevo vector de memoria solo el string que esta entre P y R
  }
  for(i=RPos+1;i<AsteriskPos;i++){           // Se realiza sentencia para buscar el valor, en string, de Roll
    strRoll[i-RPos-1]=strArray[i];           // Almacena en nuevo vector de memoria solo el string que esta entre R y *
      }
   heading=strtod(strHeading,NULL);          // Convierte los Bytes de la variable String a valor numérico decimal
   roll=strtod(strRoll,NULL);                // Convierte los Bytes de la variable String a valor numérico decimal
   pitch=strtod(strPitch,NULL);              // Convierte los Bytes de la variable String a valor numérico decimal
}                                            // Fin subrutina ProcesaCompass

void limpia(byte *strArray,int j)            // Inicio Subrutina limpia
{
  for (int i=0; i< j; i++){
    strArray[i]=0;                           // Borra los datos almacenados en memoria
  }
}                                            // Fin subrutina limpia
```

Anexo 3: “Programa desarrollado en Visual Basic 6”

A continuación se presenta el código de programación desarrollado en Visual Basic 6 para capturar y enviar datos que controlarán la unidad remota.

```
Private nivel As Integer                        'Variable nivel  global
Private n As Integer                            'Variable n global

Private Sub MSComm1_OnComm()                    'Este evento se produce al completarse Buffer de entrada con 13 byte
   Dim I As Long, Heading As Long, Pitch As Long    'Se definen variables
   Dim Roll As Long, Presion As Long
   Dim Profundidad As Long, Luz As Integer
   Dim Matriz(25) As Byte
   Dim Matrizstring(25) As String
   Dim Val As Byte, Checksum As Byte
   Dim b As Integer

   If MSComm1.CommEvent = 2 Then                'Este evento es igual a 2 al haber 13 bytes en buffer puerto serial
        Val = (MSComm1.Input(0))
      If Val = 240 Then                          'Chequea si el primer Byte es 240
        b = 1
        Matriz(b) = Val                         'Almacena el primer Byte en memoria
        b = 2
          While b < 14                          'Almacena el resto de la información
            Val = (MSComm1.Input(0))
            Matriz(b) = Val
            b = b + 1
          Wend
        End If
   End If                                       'Termina proceso almacenaje

   For I = 1 To 12                              'Calcula el checksum de la trama, en Byte
      Checksum = (Checksum Xor Matriz(I))
   Next

   For I = 1 To 12                              'Pasa a hexadecimal los datos y luego le incorpora
      If Matriz(I) >= 0 And Matriz(I) <= 15 Then    'un "0" al comienzo, si es necesario para pasar
        Matrizstring(I) = "0" & Hex(Matriz(I))      'datos a un arreglo String (1 byte)
      Else                                          '(arreglo por la forma de conversión de visual basic)
        Matrizstring(I) = Hex(Matriz(I))
      End If
   Next
   If (Matriz(1) = 240) And (Matriz(2) = 241) And Matriz(3) = 1 And Matriz(13) = Checksum Then
      Label24.Caption = "Recibido comando desconocido"  'Si se recibe un comando desconocido en unidad remota,
```

```
End If                                                  ' informa en pantalla
 If (Matriz(1) = 240) And (Matriz(2) = 241) And Matriz(3) = 0 And Matriz(13) = Checksum Then
    n = 1
    Label24.Caption = "Recibido OK "                    'Si los datos recibidos están correctos informa recibo Ok
  If Matriz(6) > 128 Then                               'en pantalla
    Matriz(6) = (Not Matriz(6))                         'verifica si estos datos son números negativos
    Matriz(7) = (Not Matriz(7))                         'si lo son, los modifica bajo el complemento a 2
    Pitch = (-1 * (CDec(Matriz(7) + 1)))
  Else
    Pitch = CDec(Matriz(7))
  End If

  If Matriz(8) > 128 Then
    Matriz(8) = (Not Matriz(8))                         'verifica si estos datos son números negativos
    Matriz(9) = (Not Matriz(9))                         'si lo son los modifica bajo el complemento a 2
    Roll = (-1 * (CDec(Matriz(9) + 1)))
  Else
    Roll = CDec(Matriz(9))
  End If

    Heading = CLng("&H" & ("00" & Matrizstring(4) & Matrizstring(5)))   'Pasa dos Byte String a Valor numérico
    Presion = CLng("&H" & ("00" & Matrizstring(10) & Matrizstring(11))) 'Pasa dos Byte String a Valor numérico
    Luz = Matriz(12)                                    'pasa nivel luz a valor numérico entero

  If Presion < 204 And Presion > 195 Then               'Ajuste para no presentar datos de profundidad negativos
    Presion = 204                                       'en caso de algún desajuste del sensor de presión
  End If

    Profundidad = (Presion - 204) * 255 / 819           'Formula para pasar información a metros de profundidad

    Label14.Caption = Heading                           'Informa en pantalla dato Heading
    Label15.Caption = Pitch                             'Informa en pantalla dato Pitch
    Label16.Caption = Roll                              'Informa en pantalla dato Roll
    Label17.Caption = Profundidad                       'Informa en pantalla dato Profundidad

 If nivel <> Luz Then                                   'Compara nivel luz en unidad remota con valor almacenado en PC
  For I = 0 To 600000                                   'Retardo de tiempo mayor a 5ms
  Next
    Select Case nivel                                   'Reenvía, si es necesario, nuevamente comando cambio nivel de luces
      Case 0
        MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H0) & Chr(&HA) & Chr(&HF9)
      Case 1
        MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H1) & Chr(&HA) & Chr(&HF8)
      Case 2
        MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H2) & Chr(&HA) & Chr(&HFB)
      Case 3
```

```
            MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H3) & Chr(&HA) & Chr(&HFA)
        Case 4
            MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H4) & Chr(&HA) & Chr(&HFD)
        Case 5
            MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H5) & Chr(&HA) & Chr(&HFC)
        Case 6
            MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H6) & Chr(&HA) & Chr(&HFF)
        Case 7
            MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H7) & Chr(&HA) & Chr(&HFE)
        Case Else
      End Select
    End If
    Else
  Label22.Caption = "Desconectada"                    'Informa en pantalla que la unidad remota esta desconectada
  End If
End Sub

Private Sub Timer1_Timer()
  MSComm1.Output = Chr(&HF1) & Chr(&H1) & Chr(&H0) & Chr(&HA) & Chr(&HFA)    'Comando solicitud datos
  Label18.Caption = Now                               'Indica la hora en pantalla

  If n = 1 Then
    Label22.Caption = "Conectada"                     'Indica en pantalla que la unidad remota esta conectada
  Else
    Label22.Caption = "Desconectada"                  'Indica en pantalla que la unidad remota esta desconectada
    Label24.Caption = " "                             'indica que no hay recibo de comandos en unidad remota
  End If
    n = 0
End Sub

Private Sub cerrar_Click()
  If MSComm1.PortOpen = True Then
  MSComm1.PortOpen = False                            'Cerrar el puerto serial
  End If
  Unload Me                                           'Cierra el programa
End Sub

Private Sub Form_Load()
  Form1.Caption = "Interfaz de Control Unidad Remota"
  MSComm1.Settings = "19200,n,8,1"                    'Configura- puerto serial- velocidad, paridad, bit de datos, bit stop
  MSComm1.CommPort = 1                                'Configura puerto serial 1 a utilizar
  MSComm1.InputMode = comInputModeBinary              'Configura el puerto serial para entregar los datos en byte
  MSComm1.InputLen = 1                                'Cantidad de Byte que se recuperan del puerto por leída
  MSComm1.RThreshold = 13                             'Cantidad de Byte necesita para un evento MSComm1_OnComm()
  MSComm1.InBufferSize = 13                           'Cantidad de Byte que almacena el buffer del puerto serial
  MSComm1.PortOpen = True                             'Abre el puerto
```

```
    nivel = 0                                          'Inicia variable global “nivel” en valor  "0"
    Label20.Caption = nivel                            'Nuestra en pantalla variable nivel
End Sub

Private Sub luz0_Click(Index As Integer)
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H0) & Chr(&HA) & Chr(&HF9)     'Envía comando nivel luces 0
    nivel = 0                                          'Almacena "0" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub

Private Sub luz1_Click()
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H1) & Chr(&HA) & Chr(&HF8)     'Envía comando nivel luces 1
    nivel = 1                                          'Almacena "1" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub
Private Sub luz2_Click()
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H2) & Chr(&HA) & Chr(&HFB)     'Envía comando nivel luces 2
    nivel = 2                                          'Almacena "2" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub
Private Sub luz3_Click()
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H3) & Chr(&HA) & Chr(&HFA)     'Envía  comando nivel luces 3
    nivel = 3                                          'Almacena "3" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub
Private Sub luz4_Click()
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H4) & Chr(&HA) & Chr(&HFD)     'Envía comando nivel luces 4
    nivel = 4                                          'Almacena "4" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub
Private Sub luz5_Click()
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H5) & Chr(&HA) & Chr(&HFC)     'Envía comando nivel luces 5
    nivel = 5                                          'Almacena "5" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub
Private Sub luz6_Click()
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H6) & Chr(&HA) & Chr(&HFF)     'Envía comando nivel luces 6
    nivel = 6                                          'Almacena "6" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub
Private Sub luz7_Click()
    MSComm1.Output = Chr(&HF1) & Chr(&H2) & Chr(&H7) & Chr(&HA) & Chr(&HFE)     'Envía comando nivel luces 7
    nivel = 7                                          'Almacena "7" en variable nivel
    Label20.Caption = nivel                            'Nuestra en pantalla nuevo nivel
End Sub
```

Anexos 4: Norma RS-232

La morma RS-232 esta definida para conectores de DB9 y DB25 pero en este caso particular sólo se enfocará al conector (DB9)

Para ilustrar mejor el significado de cada terminal, consideremos a modo de ejemplo que el DTE (Equipo Terminal de Datos) que podría ser un PC y el DCE (Equipo de Comunicación de Datos) un ratón.

Definición general de señales

TXD (Transmit Data, transmisión de datos, salida,): Señales de datos que se transmiten del DTE al DCE. En principio, los datos no se pueden transmitir si alguno de los terminales RTS, CTS, DSR ó DTR está desactivado.

RXD (Receive Data, recepción de datos, entrada,): Señales de datos transmitidos desde el DCE al DTE.

DTR (Data Terminal Ready, terminal de datos preparado, salida,): Señal del DTE que indica que está conectado, generalmente en "0" indica que el DTE está listo para transmitir o recibir.

DSR (Data Set Ready, dispositivo preparado, entrada,): Señal del DCE que indica que el dispositivo está en modo de transmisión de datos.

RTS (Request To Send, petición de envío, salida,): Señal del DTE al DCE, notifica al DCE que el DTE dispone de datos para enviar. Se emplea en líneas semiduplex para controlar la dirección de transmisión. Una transición de 1 a 0 avisa al DCE que tome las medidas necesarias para prepararse para la transmisión.

CTS (Clear To Send, preparado para transmitir,): Señal del DCE al DTE indicando que puede transmitirle datos.

CD (Carrier Detect, detección de portadora, entrada,): Señal del DCE que ha detectado la señal portadora enviado por un modem remoto o que la línea telefónica está abierta.

RI (Ring Indicator, timbre o indicador de llamada entrante, entrada,): Señal del DCE indicando que está recibiendo una llamada por un canal conmutado.

SG (GND) (System Ground ó Signal Ground, masa de señal,): Masa común para todas las líneas.

FG (GND) (Shield ó Protective Ground, tierra de protección,): El conductor esta eléctricamente conectado al equipo.

Una secuencia normal, a través de la RS-232, es la siguiente:

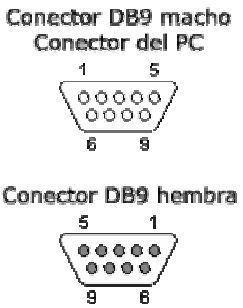
1.-Ambos dispositivos son alimentados, indicando encendido (si ha sido establecido en el equipo). El DTE activa el terminal DTR y el DCE activa el terminal DSR. Una interfase RS232 bien diseñada no comunicará hasta que estos dos terminales estén activos. El DTE esperará la activación del terminal DSR y el DTE la activación del terminal DTR. Aunque DTR y DSR algunas veces pueden ser utilizados para el control del flujo, estos terminales sólo indican que los dispositivos están conectados.

2.-El DTE pregunta al DCE si este está listo. El DTE activa la línea RTS. El DCE si está listo, responde activando la línea CTS. Puestos de acuerdo ambos equipos, se puede entrar a comunicar.

3.-Los datos son transferidos en ambos sentidos. El DTE envía información al DCE a través del terminal TXD. El DCE envía información al DTE a través del terminal RXD.

Conector Serie DB9

Pat.	Nombre	RS232	Dir	Descripción
1	CD	CF	←	Carrier Detect, detección de portadora
2	RXD	BB	←	Receive Data, recepción de datos
3	TXD	BA	→	Transmit Data, transmisión de datos
4	DTR	CD	→	Data Terminal Ready, terminal de datos preparado
5	GND	AB	—	System Ground ó Signal Ground, tierra de señal
6	DSR	CC	←	Data Set Ready, dispositivo preparado
7	RTS	CA	→	Request to Send, petición de envío
8	CTS	CB	←	Clear to Send, preparado para transmitir
9	RI	CE	←	Ring Indicator, indicador de llamada entrante

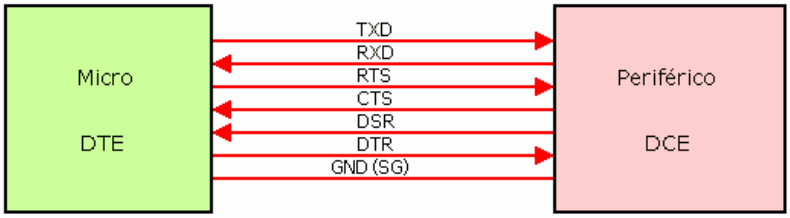


DTE (PC) ← DCE (Dispositivo), entrada en el DTE (PC).
DTE (PC) → DCE (Dispositivo), salida en el DTE (PC).

Interfaz TTL / RS-232

Para una comunicación full duplex desde un microprocesador o microcontrolador deben conectarse un mínimo número de señales, concretamente TXD y RXD así como la masa (GND, SG o Signal Ground). Sin embargo una interfaz típica RS-232 requiere al menos 7 señales.

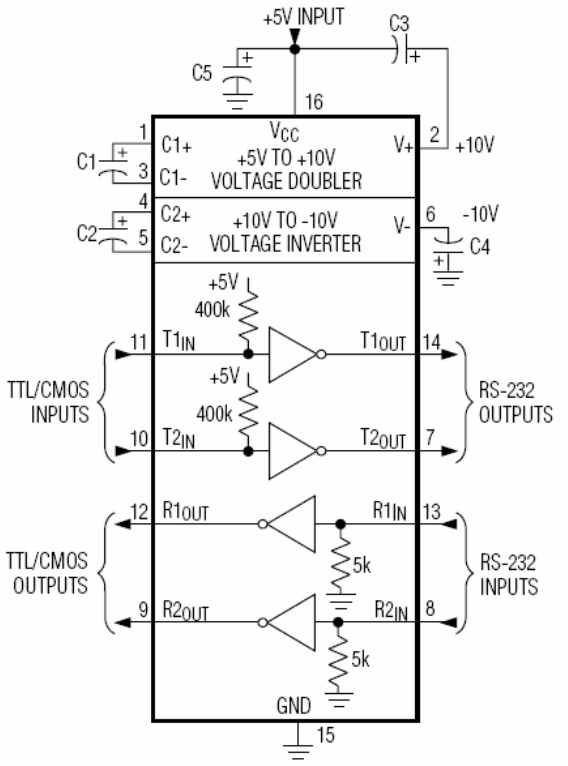
Requerimientos de señal para mínima conexion full duplex



Requerimientos de señal típica para comunicacion full duplex

Para convertir TTL a RS-232 se pueden usar circuitos típicos de transistores y diodos discretos pero hoy en el mercado existe un integrado especifico para esta función, ese es el CI MAX232 que es un conversor de nivel TTL/RS-232, para su funcionamiento solo requiere de 4 condensadores.

Funcionamiento: El circuito integrado lleva internamente 2 conversores de nivel de TTL a RS-232 y otros 2 de RS-232 a TTL con lo que en total podremos manejar 4 señales del puerto serie del PC, por lo general las más usadas son; TXD, RXD, RTS, CTS, estas dos últimas son las usadas para el protocolo handshaking pero no es imprescindible su uso. Para que el MAX232 funcione correctamente debemos poner unos condensadores externos, todo esto lo podemos ver en la siguiente figura.



RS-232 en el PC

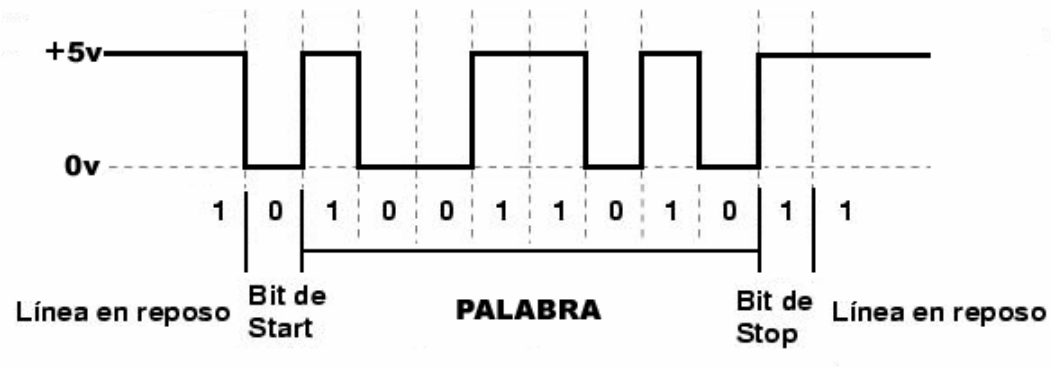
El puerto serie de un ordenador trabaja en modo síncrono o asincrónico (USART), recibe y envía información fuera del ordenador mediante un determinado software de comunicación o un driver del puerto serie. La información se envía al puerto carácter a carácter. Cuando se ha recibido un carácter, el puerto serie envía una señal por medio de una interrupción indicando que el carácter está listo. Cuando el ordenador ve la señal, los servicios del puerto serie leen el carácter.

Existen dos tipos de interfaces RS-232 puesto que la norma fue diseñada para dos tipos de equipos, el DTE (Equipo Terminal de Datos) y el DCE (Equipo de Comunicación de Datos). Existen entonces dos tipos de interfaz RS-232, la DTE (conector macho) y la DCE (conector hembra):

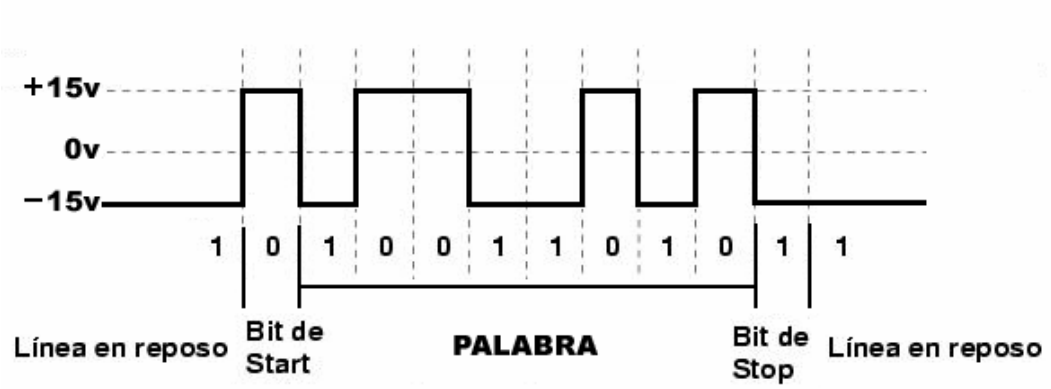
Por tanto, en un PC, se utilizan conectores DB9 macho de 9 patillas por los que se conectan los dispositivos al puerto serie. Los conectores hembra que se enchufan tienen una colocación de patillas diferente, de manera que se conectan la patilla 1 del macho con la patilla 1 del hembra, la patilla 2 con el 2, etc... La norma RS-232 no admite comunicaciones a más de 15 metros y 20 Kbps, se puede utilizar mayor distancia y velocidad pero no es el estándar. RS-232 está definida tanto para la comunicación síncrona como asíncrona, pero cuando se utiliza esta última sólo se utiliza un conjunto de los terminales.

Normalmente, las comunicaciones serie en el PC tienen los siguientes parámetros: 9.600 baudios, 1 bit de Start, 8 bits de Datos, 1 bit de Stop y sin paridad.

En la figura siguiente se puede ver un ejemplo de la transmisión en TTL del dato binario **01011001**. La línea en reposo está a nivel lógico alto (+5 voltios).



En la figura siguiente se puede ver un ejemplo de la transmisión en RS-232 del dato binario **01011001**. La línea en reposo está a nivel lógico alto (-15 voltios).



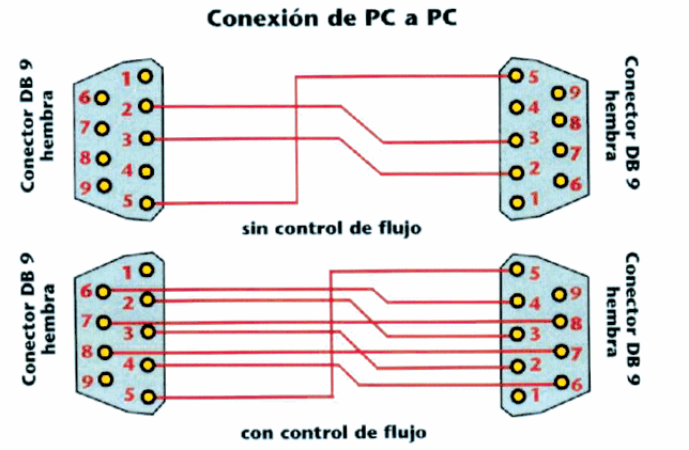
Tipos de conexiones con DB9

Dos PC´s no se puede conectar de manera directa entre sí, pues son dos DTE, pero no obstante se pueden comunicar invirtiendo algunas líneas.

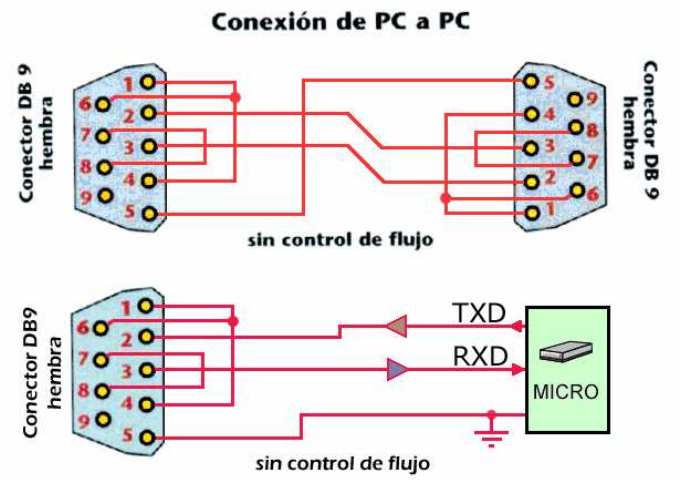
Para conectar dos DTE hay que tener en cuenta que ambos transmiten por la línea 2 y reciben por la línea 3, por ello, basta cruzar RXD (2) y TXD (3). También debe conectarse la línea de tierra de señal. Esta conexión es válida cuando el software que controla la comunicación no utiliza los terminales de control.

Si es necesario utilizar los terminales "en línea" (DSR y DTR) se debe considerar que ambos DTE activarán el terminal DTR (4) y esperarán por la activación del terminal DSR (6). Como ninguno activará el terminal DSR, estarán esperando siempre. Este problema se puede solucionar mediante el intercambio de las señales de control, basta cruzar los terminales DSR (6) y DTR (4)

Con respecto a los terminales RTS (7) y CTS (8) sucede algo similar a DSR y RTS, por ello se pueden cruzar los terminales 7 y 8.

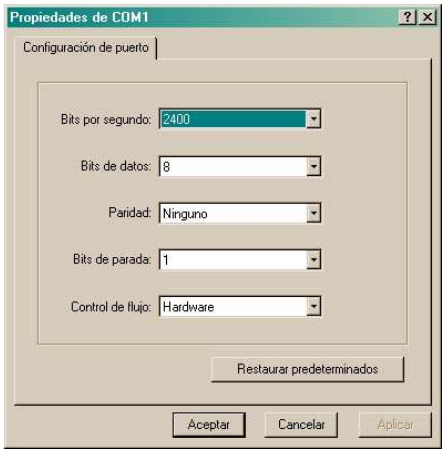


Otra forma de conexión, en este caso sin control de flujo, se haría considerando que como cada DTE espera la activación del terminal DSR al mismo tiempo que activa el DTR, se unan en cada DTE, para que cada DTE se de a sí mismo la posibilidad de transmisión. Lo mismo se haría con RTS y CTS. También se conectará el terminal CD a DTR. Algunos programas no trabajan si este terminal no está activo. De manera que como CD es entrada en ambos DTE, se debe mantener activo conectándolo a DTR.



Las conexiones que presenta la figura garantizan que cualquier programa de comunicación acepte la transmisión del microcontrolador, si bien se realizará sin control de flujo. La salida DTR (patilla 4, Terminal de Datos Preparado) entrega señal a la entrada DCD (patilla 1, Detección de Portadora) y a la entrada DSR (patilla 6, Dispositivo Preparado). Por otro lado la salida RTS (patilla 7, Petición de Envío), entrega señal a la entrada CTS (patilla 8, Preparado para el Envío).

Configuración de los puertos



- **Bit por segundo:**
  - Define la velocidad máxima, en bits por segundo (bps), a la que se transmiten los datos a través del puerto.

- **Bits de datos:**
  - Cambia el número de bits de datos a utilizar para cada carácter transmitido y recibido. El equipo o dispositivo con el que comunica debe tener la misma configuración que aquí. La mayor parte de los caracteres se transmiten con siete u ocho bits de datos.

- **Paridad:**
  - Cambia el tipo de comprobación de errores a utilizar para el puerto seleccionado. El equipo o dispositivo con el que se comunica debe tener la misma configuración que aquí. Se debe elegir una de las siguientes:
    - **Ninguna:** significa que no se agregará ningún bit de paridad a los bits de datos enviados desde este puerto. Esto deshabilitará la comprobación de errores.
    - **Par:** significa que el bit de paridad se establece a 1 si se necesita para que el número de unos (1) de los bits de datos sea par. Esto habilitará la comprobación de errores.
    - **Impar:** significa que se agrega un bit de paridad si se necesita para que el número de unos (1) de los bits de datos sea impar. Esto habilitará la comprobación de errores.
    - **Marca:** significa que se agrega un bit de paridad, pero siempre está establecido a 0.
    - **Espacio:** significa que se agrega un bit de paridad, pero siempre está establecido a 1.
- **Bit de parada:**
  - Cambia el tiempo entre cada carácter que se transmite (cuando el tiempo se mide en bits por segundo).
- **Control de flujo:**
  - Cambia la forma en que se controla el flujo de datos.
    - **Ninguno**
    - **off**, llamado en ocasiones protocolo de enlace software, es el método de software estándar para controlar el flujo de datos entre dos módems.
    - **Control de flujo Hardware**, llamado en ocasiones protocolo de enlace hardware, es el método estándar de controlar el flujo de datos entre un equipo y un dispositivo serie.