INSTITUTO TECNOLOGICO DE TUXTLA GUTIERREZ

PROYECTO DE RESIDENCIA PROFESIONAL

“AUTOMATIZACIÓN DE UN CRIADERO DE MOJARRA TILAPIA**”**

Nombre	Carrera	Numero de Control
ROJAS PECHA ROXUEL	Ingeniería en electrónica	05270165

ASESOR INTERNO

Dr. HECTOR RICARDO HERNANDEZ DE LEON

ASESOR EXTERNO

Dr. MADAIN PEREZ PATRICIO

REVISOR

M. en C. ANGEL SEIN PEREZ RODRIGUEZ

TUXTLA GUTIÉRREZ, CHIAPAS, JUNIO DEL 2010

INTRODUCCION

En estos tiempo donde la humanidad se ha ido aumentando drásticamente y la demanda de alimentos es excesiva se tiene que tomar otras medidas para la producción de alimentos así pues es cómo surge la necesidad de cultivar mojaras, con ciertas características para lograr criar mojaras de un tamaño considerable.

En la crianza de mojarras hay que tomar en cuenta que es necesario monitorear ciertas variables del agua diariamente como lo es la oxigenación, la acidez y la temperatura con esto se asegura la salud de las mojarras; pero al hacer este monitoreo manualmente no se logra una eficiencia optima y por tanto el tamaño y calidad de las mojarras es menor al esperado, es por eso que nace la necesidad de construir un criadero automatizado con el fin de estar monitorizando cada determinado tiempo durante el día y la noche las condiciones del agua, y así el sistema regule estas variables a su nivel óptimo de forma automática, adema con este sistema automático se podrán guardar los registros de los eventos para que posteriormente se puedan analizar cuáles fueron las causas por las que estas variables salieran de su nivel, así como poder determinar que tanto oxigeno consumen las mojaras en sus diferentes etapas de crecimiento y así poder lograr mojarras de buen tamaño para brindar a la población alimentos de calidad.

CAPITULO 1:
PLANTEAMIENTO DEL PROBLEMA.

1.1 Justificación

El crecimiento de los peces está determinado fundamentalmente por la cantidad de alimento ingerido (energía y nutrientes) y por la temperatura del agua.

Los peces, como animales poiquilotermos son incapaces de regular su temperatura corporal, por lo que su metabolismo únicamente funciona de forma óptima dentro de un rango de temperaturas adecuadas, dentro del cual la ingestión y el crecimiento son máximos, pero disminuyen cuando la temperatura está por encima o por debajo del rango óptimo.

También se toma en cuenta que si los peces sufren algún tipo de estrés debido a la falta de oxígeno en el agua, esto dejan de moverse quedándose estáticos y por tanto dejan de alimentarse, teniendo así repercusiones en su crecimiento y calidad.

Cuando se aumenta la acidez del agua el Ion Ferroso (Fe^{2+}) se vuelve soluble afectando las células de los arcos branquiales, incidiendo directamente en los procesos de la respiración, ocasionando altas mortalidades por anoxia (asfixia por falta de O_2).

En aguas ácidas (por debajo de 6.0), el crecimiento se reduce, pérdida del apetito (inapetencia), hay problemas de aletargamiento, disminuye la fecundidad, la piel se decolora por excesiva producción de mucus, la muerte se produce por falla respiratoria; por el contrario en aguas totalmente alcalinas (por encima de 11.0) se inicia una alta mortalidad.

Al surgir estas necesidades se toma la decisión de crear un sistema de monitoreo constante de las variables de oxigenación, acidez y temperatura del agua con el fin de tenerlas a su nivel óptimo y poder guardar los datos de cada evento de estas variables para poder estudiarlos posteriormente y determinar cuál fue el motivo que ocasiono la alteración de los valores de estas tres variables, así como automatizar el sistema de oxigenación del agua por medio de aereación, esperando así mejorar la crianza de la mojarra tilapia.

1.2 Objetivos

1.2.1 Objetivos Generales

Mejorar la crianza de la mojarra tilapia optimizando las condiciones del agua; manteniendo la oxigenación y la temperatura a los niveles óptimos para mantener a los peces sanos, monitoreando estas variables con ayuda de sensores, una tarjeta de adquisición de datos y un software implementado en una computadora.

1.2.2 Específicos

1. Estudio e implementación de un sistema de adquisición de datos.

2. Determinación de los parámetros de temperatura oxigenación y acidez adecuados para un criadero de mojarra

3. Generación de un programa que automatice el criadero, alimente a los peces en tiempos definidos y monitorice las condiciones de las variables de temperatura oxigenación y acidez.

4. Oxigenar el agua contenida del criadero utilizando aereadores de forma automática.

5. Facilitar la crianza de las mojarras para impulsar el cultivo de tilapia en la región.

1.3 Caracterización del área en que se participo

El desarrollo del presente trabajo de residencia profesional fue desarrollado en el Área de Posgrado del Instituto Tecnológico de Tuxtla Gutiérrez (Figura 1.1).

El Instituto Tecnológico de Tuxtla Gutiérrez es una institución pública dependiente de la Secretaría de Educación Pública. Imparte 9 licenciaturas y 2 programas de posgrado.

Figura 1.1. Instituto Tecnológico de Tuxtla Gutiérrez

1.4 Misión, visión y valores de la institución

Misión: Formar de manera integral profesionistas de excelencia en el campo de la ciencia y la tecnología con actitud emprendedora, respeto al medio ambiente y apego a los valores éticos.

Visión: Ser una Institución de excelencia en la educación superior tecnológica del Sureste, comprometida con el desarrollo socioeconómico sustentable de la región.

Valores: El ser humano, el espíritu de servicio, el liderazgo, el trabajo en equipo, la calidad y el alto desempeño.

1.5 Ubicación geográfica del área

Macrolocalización
El Instituto Tecnológico de Tuxtla Gutiérrez se encuentra localizado en el país Estados Unidos Mexicanos, específicamente en la Ciudad llamada Tuxtla Gutiérrez capital política del estado de Chiapas al sureste del país antes mencionado. La Figura 1.2 muestra la ubicación de Tuxtla Gutiérrez.

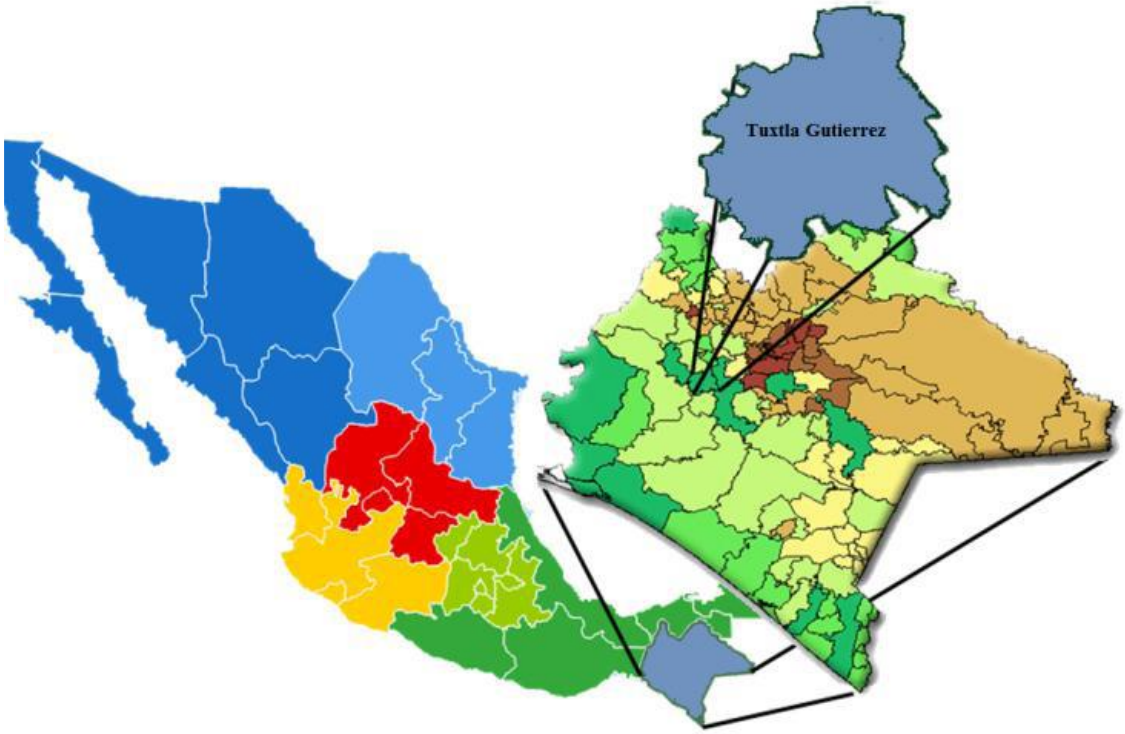


Figura 1.2. Tuxtla Gutiérrez, Chiapas

Macrolocalización
Las instalaciones del Instituto Tecnológico de Tuxtla Gutiérrez se ubican dentro de la ciudad de Tuxtla Gutiérrez, Chiapas. La dirección de ésta institución educativa es Carretera Panamericana Km. 1080. Tuxtla Gutiérrez, Chiapas, México. C. P. 29000, Apartado Postal 599 (Ver Figura 1.3).

Figura 1.3. Instituto Tecnológico De Tuxtla Gutiérrez

CAPITULO 2:
FUNDAMENTOS TEORICOS

2.1 Reseña Histórica de la Mojarra Tilapia.

La tilapia es un pez teleósteo, del orden Perciforme perteneciente a la familia Cichlidae. Originario de África, habita la mayor parte de las regiones tropicales del mundo donde las condiciones son favorables para su reproducción y crecimiento.

Es un pez de buen sabor y rápido crecimiento, se puede cultivar en estanques y en jaulas, soporta altas densidades, resiste condiciones ambientales adversas, tolera bajas concentraciones de oxigeno y es capaz de utilizar la productividad primaria de los estanques, y puede ser manipulado genéticamente.
Actualmente se cultivan con éxito unas diez especies. Como grupo las tilapias representan uno de los peces más ampliamente producidos en el mundo. Las especies más cultivadas son *Oreochromis aureus*, *O. niloticus* y *O. mossambicus* así como varios híbridos de esta especie. La menos deseable es la *O. mossambicus* a pesar de que fue la primera especie en distribuirse fuera de África; tanto la *O. aureus* como la *O. mossambicus* y se reproducen en mayor número. La tilapia roja es un híbrido proveniente de líneas mejoradas partiendo de las cuatro especies parentales del híbrido son: *Oreochromis aureus, Oreochromis niloticus, Oreochromis mossambicus* y *Oreochromis urolepis hornorum.* Por estar emparentadas entre si, sus comportamientos reproductivos y alimenticios son similares. El desarrollo de este híbrido permitió obtener muchas ventajas sobre otras especies, como alto porcentaje de masa muscular, filete grande, ausencia de espinas intramusculares, crecimiento rápido, adaptabilidad al ambiente, resistencia a enfermedades, excelente textura de carne y una coloración de muy buena aceptación en el mercado.

2.1.1 Factores para la selección de la especie a cultivar.

Dentro de las principales características que se deben tener en cuenta para la elección de una especie a cultivar tenemos:

- Curva de crecimiento rápido.
- Hábitos alimenticios adaptados a dietas suplementarias que aumentan los rendimientos (facilidad de administrar alimentos balanceados).

- Tolerancia a altas densidades de siembra, debido a los altos costos de adecuación de terrenos e insumos.
- Tolerancia a condiciones extremas: resistencia a concentraciones bajas de oxigeno, niveles altos de amonio, valores bajos de pH.
- Fácil manejo: resistencia al manipuleo en siembra, trasferencias, cosechas, manejo de reproductores.
- Capacidad de alcanzar tamaños de venta antes de la madurez sexual: la cosecha se hace a los 8 meses y la madurez sexual se alcanza dependiendo de la pureza de la línea (luego de los 3 meses).
- Facilidad de reproducción levante de reproductores y disponibilidad de alevines.
- Buen fenotipo y de fácil aceptación en el mercado.
- Buenos parámetros de producción (conversión alimenticia, ganancia de peso, supervivencia, etc).

2.1.2 DISTRIBUCION GEOGRAFICA.

Las tilapias son originarias de África y se encuentran habitando la mayor parte de las regiones tropicales del mundo donde las condiciones son favorables para su reproducción y crecimiento.

La tilapia se encuentra en las aguas libres, tanto dulces como salobres; su cultivo está extendido en casi todos los Estados de la Republica Mexicana, sobre todo en zonas cálidas y semicálidas, aunque también se desarrolla en las regiones norteñas por su gran resistencia. Son peces robustos, con pocas exigencias respiratorias, soportan bien el calor y son fáciles de transportar, su cultivo se registra en los siguientes Estados: Baja California, Sinaloa, Coahuila, Nuevo León, Tamaulipas, Durango, Aguascalientes, Jalisco, Hidalgo, Morelos, Puebla, Guanajuato, Michoacán, Colima, Veracruz, Tabasco, Campeche, Yucatán, Quintana Roo, Oaxaca. En base a la información anterior se estima que casi el 70% de las entidades federativas cuentan con tilapia en sus cultivos.

2.1.3 HABITAT

Las tilapias o mojarras africanas como se les conoce comúnmente en México, son especies aptas para el cultivo en zonas tropicales y subtropicales del país.

Se les encuentra habitando en aguas lénticas (lentas), principalmente someras o turbias (estancadas o inactivas) como lagos, lagunas, litorales, bordos, estanques, charcos así como también en loticas (aguas corrientes) a orillas de ríos entre piedras y plantas acuáticas e inclusive en aguas marinas.

El hábitat que prefieren es de fondo lodoso, toleran altas salinidades, son peces eurihalinos, o sea que pueden vivir en aguas dulces, salobres y marinas, el rango de tolerancia es de 0°/00 a 40°/00(partes por mil) y en algunos casos, se ha presentado por arriba de esta salinidad.

Son especies euritermas, siendo el rango de tolerancia de 12°C a 42°C. La temperatura ideal para su cultivo fluctúa entre 29°C, aunque se reproduce aún a los 18°C., además soportan concentraciones de oxígeno bastante bajas, su requerimiento mínimo es de 1 mg/lt.
Se reproducen a temprana edad, alrededor de las 8 ó 10 semanas, teniendo una talla entre 7 a 16 cm., por lo que dificulta el control de la población en los estanques donde se cultiva.
Los machos establecen posesiones territoriales durante la temporada de reproducción; este territorio se observa claramente definido y defendido de los depredadores que atacan a sus crías, puede ser fijo o cambiar a medida que se mueven las crías en busca de alimento.
Para asegurar una buena producción y sanidad, es necesario que los parámetros físico-químicos (°C., O2, pH, etc). de la calidad del agua, se mantengan entre los límites de tolerancia de la especie a tratar.

2.1.4 HABITOS ALIMENTICIOS

Son Ciclidos considerados como omnívoros que hasta su etapa de cría de 5 cm. presenta preferencias fitoplanctofagas, puesto que su alimentación se basa en el consumo de zooplancton, insectos y vegetales acuáticos, y de alimentos artificiales como harinas y granos.
Los juveniles se alimentan preferentemente de fitoplancton y zooplancton, inclusive aceptan alimentos preparados que se utilizan en la crianza de pollos.

Los adultos comen plancton, algas filamentosas, algunas plantas superiores y detritus vegetal.

2.1.5 CICLO DE VIDA

El ciclo de vida de la tilapia comprende solo 4 etapas básicas:

2.1.6 Desarrollo embrionario

Cuando se lleva a cabo la fecundación, a medida que avanza la división celular las células comienzan a envolver el vítelo hasta rodearlo completamente, dejando en el extremo una abertura que más tarde se cierra. Posteriormente, una vez formada la mayor parte del organismo, el embrión comienza a girar dentro del espacio peri-vitelino, ese movimiento giratorio y los demás movimientos se hacen más enérgicos antes de la eclosión. Los metabolitos del embrión contienen algunas enzimas que actúan sobre la membrana del huevo y la disuelven desde adentro, permitiendo al embrión romperla y salir fácilmente figura 2.1.

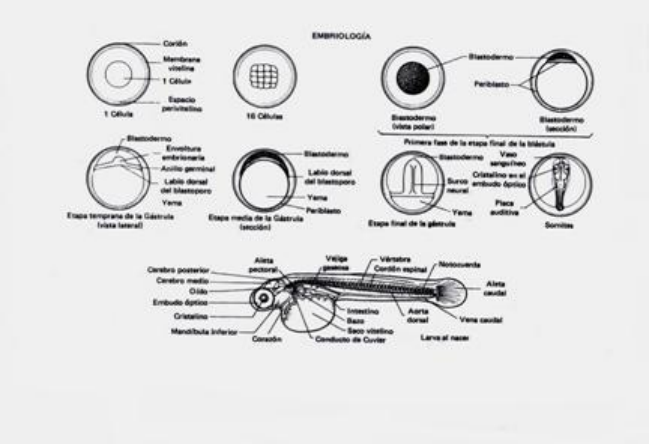


Figura 2.1a Proceso del desarrollo embrionario

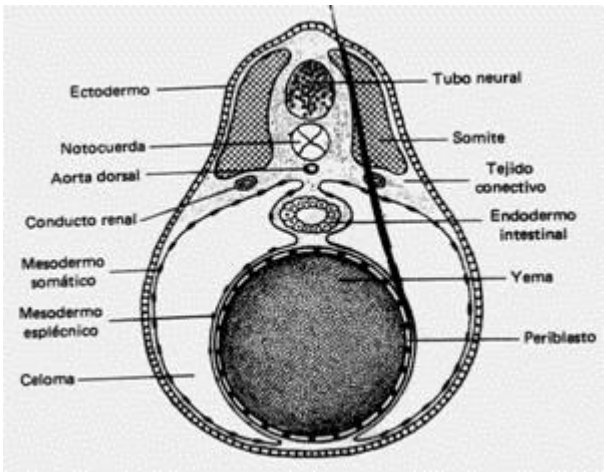


Figura 2.1b Corte transversal de un huevo embrionado

2.1.7 Alevín

Es la etapa el desarrollo subsecuente al embrión y a la eclosión, dura alrededor de 3 a 5 días; en esta fase, el alevín, se caracteriza porque presenta un tamaño de 0.5 a 1 cm y posee un saco vitelino en el vientre que es de donde se alimenta los primeros días de nacido. Posteriormente a esta talla se le considera cría Figura 2.2.

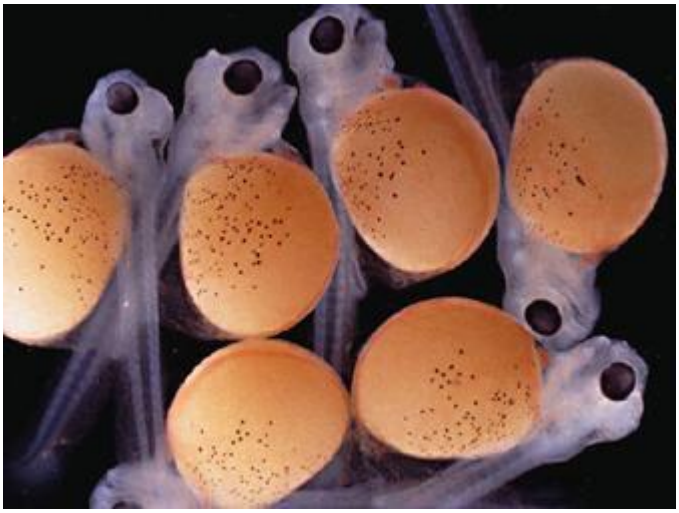


Figura 2.2 Alevines recién eclosionados se observa el saco vitelino

2.1.8 Cría

Se les llama cría cuando los peces han absorbido el saco vitelino y comienzan a aceptar alimento balanceado, y han alcanzado una talla de 1 a 5 cm. De longitud figura 2.3.

Figura 2.3 Cría de tilapia de 45 días de nacida

2.1.9 Juvenil

Son peces con una talla que varía entre 5 y 10 cm, la cuál alcanzan a los 2 meses de edad y aceptan alimento balanceado para crecimiento figura 2.4.

Figura 2.4 Juvenil de tilapia de 3 meses

2.1.10 Adulto

Es la última etapa del desarrollo, los individuos presentan tallas entre 10 y 18 cm y pesos de 70 a 100 gr, características que obtienen alrededor de los 3.5 meses de edad.

Tallas y pesos estimados para
cada etapa de vida de la tilapia

Tabla 2.1

Estadio	Talla (cm)	Peso (gr)	Tiempo (días)
Huevo	0.2-0.3	0.01	3-8
Alevín	0.7-1.0	0.10-0.12	10-15
Cría	1-5	0.5-4.7	15-30
Juvenil	5-10	10-50	45-60
Adulto	10-18	70-100	70-90

2.1.11 principales características de la tilapia:

- Rango de pesos adultos: 1 000 a 3 000 gramos.
- Edad de madurez sexual: Machos (4 a 6 meses), hembras (3 a 5 meses).
- Número de desoves: Rango 25 a 31°C.
- Número de huevos/hembra/desove: bajo buenas condiciones mayor de 100 huevos hasta un promedio de 1,500 dependiendo de la hembra.
- Vida útil de los reproductores: 2 a 3 años.
- Tipo de incubación: bucal.
- Tipo de incubación: 3 a 6 días.
- Proporción de siembra de reproductores: 1.5 a 2 machos por cada 3 hembras.
- Tiempo de cultivo: bajo buenas condiciones de 7 a 8 meses, cuando se alcanza un peso comercial de 300 gramos (depende de la temperatura del agua, variación de temperatura día vs noche, densidad de siembra y técnica de manejo).

2.1.12 Oxígeno.

Es el requerimiento más importante, al igual que la temperatura, para los cultivos de las especies hidrobiológicas.

Su grado de saturación es inversamente proporcional a la altitud y directamente proporcional a la temperatura y el pH.

El rango óptimo está por encima de las 4 ppm medido en la estructura de salida del estanque.

Tabla 2.2

Oxigeno (ppm)	**Efectos**
0.0 – 0.3	Los peces pequeños sobreviven en cortos períodos.
0.3 –2.0	Letal en exposiciones prolongadas.
3.0 – 4.0	Los peces sobreviven pero crecen lentamente.
> 4.5	Rango deseable para el crecimiento del pez.

La concentración de Oxigeno Disuelto varía de acuerdo con la profundidad, del estancamiento del agua y de la estratificación térmica. En aguas totalmente estratificadas, se carece de oxigeno en sus capas mas bajas (hipolimnio), en donde el oxigeno es consumido pero no producido, mientras que en las capas superficiales se mantienen niveles aceptables de oxigeno, producidos por la fotosíntesis.

La Tolerancia a bajos niveles de Oxígeno es muy variable según la especie. Por ejemplo: las Tilapias pueden sobrevivir extrayendo el OD de la interfase agua-aire que en algunos casos puede estar por debajo de 1 mg/l, mediante el sistema de “boqueo”.

El nivel mínimo óptimo siempre debe estar por encima de 3 mg/l, ya que este determinará la capacidad de carga en biomasa en los estanques.
el grado de saturación de oxígeno es inversamente proporcional a la altitud sobre el nivel del mar y directamente proporcional a la temperatura y pH.

2.1.12.1Factores que disminuyen el nivel de oxígeno disuelto.

- Descomposición de la materia orgánica.
- Alimento no consumido.
- Heces.
- Animales muertos.
- Aumento de la tasa metabólica por el incremento en la temperatura (variación de la temperatura del día con respecto a la noche).

- Respiración del plancton (organismos microscópicos vegetales y animales que conforman la productividad primaria).
- Desgacificación: salida del oxígeno del agua hacia la atmósfera.
- Nubosidad: en días opacos o nublados las algas no producen el suficiente oxígeno.
- Aumento de sólidos en suspensión: residuos de sedimentos en el agua, heces, etc.
- Densidad de siembra.

La tilapia es capaz de sobrevivir a niveles bajos de oxígeno disuelto (1.0 mg/l), no obstante, el efecto de estrés al cual se somete es la principal causa de infecciones patológicas. Los niveles mínimos de oxígeno disuelto para mantener un crecimiento normal y baja mortandad se debe mantener un nivel superior a los 3.0 mg/l, valores menores a éste reducen el crecimiento e incrementan la mortandad.

2.1.12.2 Consecuencias de las bajas prolongadas de oxígeno.

- Disminuye la tasa de crecimiento del animal.
- Aumenta la conversión alimenticia (relación alimento consumido / aumento de peso).
- Se produce inapetencia y letargia.
- Causa enfermedad a nivel de branquias.
- Produce inmunosupresión y susceptibilidad a enfermedades.
- Disminuye la capacidad reproductiva.

2.1.12.3 Tipos de Aireación.

- Natural: Caídas de agua, escaleras, chorros, cascadas, sistemas de abanico.
- Mecánica: Motobombas, difusores, aireadores de paletas, aireadores inyección O_2, generadores de oxígeno líquido.

2.1.12.4 Ventajas de una buena aireación.

- Permite incrementar las densidades de siembra hasta en un 30% y manejar densidades más altas por unidad de área, como en el caso de las jaulas.
- Buenos rendimientos (crecimiento, conversión alimenticia, incremento de peso y menor mortandad).
- Control de los excesos en los niveles de amonio, fósforo y nitritos.
- Compensa los consumos de oxígeno demandados en la degradación de la materia orgánica, manteniendo niveles más constantes dentro del cuerpo de agua.
- Controla el crecimiento excesivo de algas, ya que evita altas concentraciones de nutrientes.
- Elimina los gases tóxicos.

2.1.12.5DQO (Demanda Química de Oxigeno) y DBO (Demanda Biológica de Oxigeno)

A mayor disponibilidad de nutrientes varían también dos parámetros que casi nunca se toman en cuenta en piscicultura y que son: la demanda química de oxigeno (DQO) y la demanda biológica de oxigeno (DBO), las cuales demuestran la cantidad de oxigeno consumido por los procesos de degradación de la materia orgánica. Por ejemplo en las piscinas de peces con alimentación la DBO varia entre 4 a 6 mg/L por hora y el incremento puede ser mayor dependiendo de la comida “extra” suministrada y no consumida por los peces.

La caída del plancton es una condición que se presenta en aguas eutróficas donde las cantidades masivas de algas mueren repentinamente. Usualmente la muerte del fitoplancton ocurre durante el tiempo claro y cálido. El plancton muerto se descompone rápidamente aumentando el SDBO debido a la degradación y a la reducción de la fotosíntesis.

Entre el 80 y el 85% de los nutrientes de los alimentos (especialmente peletizados), son liberados en el agua como materia fecal o compuestos metabolizados, los cuales incluyen fosfatos, amonio, CO_2 que a su vez

promueven la formación de fitoplancton. La materia orgánica por la fotosíntesis del fitoplancton puede algunas veces exceder la materia orgánica producida por los desechos fecales, por lo tanto algunas veces el metabolismo del fitoplancton es muchas veces mayor que el metabolismo del pez.

El metabolismo del zooplancton, de las bacterias y de otros microorganismos que provienen del fitoplancton pueden en ocasiones ser tan altos como el metabolismo de los peces.

Los desechos del alimento aumentan directamente con el consumo del mismo, aumentando las densidades del fitoplancton, disminuyendo la profundidad de la fotosíntesis, aumentando la DBO y la DQO.

Estos cambios producen un deterioro crítico en la calidad del agua, manifestándose en el síndrome de OD en horas de la mañana.

2.1.13Temperatura.

Normalmente todos los organismos acuáticos de aguas frías, templadas y cálidas susceptibles de cultivo, tienen un rango óptimo de temperatura, y comienzan a tener problemas con las temperaturas subóptimas (por debajo o por encima del rango óptimo) llegando a ser letales, ya que afecta directamente la tasa metabólica del pez. Por ejemplo: si la temperatura aumenta la tasa metabólica también aumenta, por consiguiente aumenta el consumo de oxígeno.

Los peces son animales poiquilotermos (su temperatura corporal depende de la temperatura del medio) y altamente termófilos (dependientes y sensibles a los cambios de la temperatura).

Por lo que en muchas especies variaciones bruscas de solo 2 ^{o}C ocasionan tensión y muerte de los mismos.

- El rango óptimo de temperatura para el cultivo de tilapias fluctúa entre 28 y 32°C, con variaciones de hasta 5°C.

- Los cambios de temperatura afectan directamente la tasa metabólica, mientras mayor sea la temperatura, mayor tasa metabólica y, por ende, mayor consumo de oxígeno.
- Variaciones grandes de temperatura entre el día y la noche deben subsanarse con el suministro de alimentos con porcentajes altos de proteína (30%, 32%, etc).

Según la Temperatura del agua los peces se clasifican en 3 grandes grupos:

Tabla 2.3

PECES	ALTURA	TEMPERATURA
Aguas Frías	2.000 a 3.000	8 a 18 ºC
Aguas Templadas	1.200 a 2.000	18 a 22 ºC
Aguas Cálidas	0 a 1.200	22 a 30 ºC

Uno de los problemas más importantes, es que a temperaturas subóptimas los peces dejan de alimentarse, el sistema inmune se debilita, y los peces se tornan altamente susceptibles a enfermedades, mortalidad por manipulación, se inhibe la reproducción, etc.

Normalmente las grandes variaciones en la temperatura son subsanadas con una excelente alimentación.

En estanques profundos sin recambio eficiente de agua, se presenta estratificación termal del agua, por la diferencia de las densidades, el agua caliente es menos densa que la fría, y entre ellas se forma una línea limítrofe llamada TERMOCLINA, la cual impide el paso de oxígeno desde la superficie (epilimnio) hacia aguas más profundas (hipolimnio) y la salida de gases tóxicos desde aguas profundas hacia la atmósfera.

2.1.14 pH.

Es la concentración de iones de hidrógeno en el agua.

La gran mayoría de los organismos acuáticos sobreviven sin problemas en aguas neutrales (pH = 7.0) o ligeramente alcalinas, en peces el rango normal se encuentra entre 6.5 y 9.0, ya que esto permite la secreción normal de mucus en la piel, combinado con una dureza normalmente alta.

La Basicidad o Acidez del agua se ve influenciada directamente por la concentración de CO_2, la densidad del fitoplancton, la alcalinidad total y la dureza.

A una alcalinidad total de 20 ppm y una dureza de 150 ppm, los valores diarios de pH durante un día claro pueden fluctuar entre 7 +/- 0.5 al amanecer y pH de 9,0 +/- 0,5 en la tarde. En aguas con baja alcalinidad, el pH puede fluctuar entre 5,7 al amanecer y 9,7 en la tarde, siendo estos extremos potencialmente estresantes para los peces.

En aguas con alta alcalinidad total y baja dureza los valores de pH en las tardes pueden exceder niveles de pH de 11, máximo valor tolerado por los peces.

Las aguas con baja alcalinidad total (< 15 ppm) son consideradas no aptas para la acuicultura debido a que pueden presentar acidez que interfiere en los resultados esperados de producción, el CO_2 y el ácido carbónico presentes limitan la producción de fitoplancton y se producen niveles extremos de pH que causan condiciones de estrés ácida en las mañanas y condiciones de estrés alcalinas en las tardes.

Cuando se aumenta la acidez del agua el Ion Ferroso (Fe^{2+}) se vuelve soluble afectando las células de los arcos branquiales, incidiendo directamente en los procesos de la respiración, ocasionando altas mortalidades por anoxia (asfixia por falta de O_2).

En aguas ácidas (por debajo de 6.0), el crecimiento se reduce, pérdida del apetito (inapetencia), hay problemas de aletargamiento, disminuye la fecundidad, la piel se decolora por excesiva producción de mucus, la muerte se produce por falla respiratoria; por el contrario en aguas totalmente alcalinas (por encima de 11.0) se inicia una alta mortalidad.

- El rango óptimo está entre 6.5 a 9.0
- Valores por encima o por debajo, causan cambios de comportamiento en los peces como letargia, inapetencia, disminuyen y retrasan la reproducción y disminuyen el crecimiento.
- Valores de pH cercanos a 5 producen mortandad en un período de 3 a 5 horas, por fallas respiratorias, además causan pérdidas de pigmentación e incremento en la secreción de mucus.
- Cuando se presentan niveles de pH ácidos el ion Fe^{++} se vuelve soluble afectando los arcos branquiales y disminuyendo los procesos de respiración, causando la muerte por anoxia (asfixia por falta de oxígeno).

2.1.15 Cultivo en Estanques.

El cultivo de tilapia en estanques se puede llevar a cabo en diferentes grados de intensidad dependiendo de las características del estanque, según estas se pueden desarrollar diferentes tipos de cultivo.

2.1.15.1Estanquería.

Un medio de cultivo muy común es la estanquería rústica, aunque existen otros sistemas como jaulas flotantes, estanques de concreto o de plástico.

2.1.15.2 El estanque de concreto:

Normalmente es de un tamaño inferior a un ¼ de hectárea, tiene la ventaja de un fácil manejo y saneamiento cada fin de ciclo, además de un bajo costo de mantenimiento si está bien construido. Al tener paredes relativamente lisas, es muy fácil desprender cualquier incrustación de organismos y microalgas adheridas a las mismas.

La desventaja es que estos estanques son muy caros de construir, además de que permiten el intercambio de temperatura con el medio y por tanto el agua pierde calor en estos sistemas.

Generalmente se utilizan en sistemas intensivos o hiperintensivos, lo que permite amortizar su costo y operación pues requieren instalación de sistemas de aireación.
En estos sistemas se llega a cultivar tilapia u otras especies a una densidad de más de 100 Kg de biomasa por m^3 de agua, lo cual es una gran cantidad si compramos con los 3 a 4 kg/m^3 que se cultivan en un estanque rústico.

Figura 2.4 Estanque de concreto para engorda intensiva

2.1.15.3 Los estanques de plástico o fibra de vidrio.

Se utilizan comúnmente para reproducción o cría de juveniles, normalmente se utilizan en laboratorios o institutos de investigación, su tamaño es aún menor a los de concreto, y por sus características, deben tener capacidad de un vaciado y llenado rápido, equipados con suministro de aire u oxígeno, así como agua dulce y/o salada.

Figura 2.5 Estanques de fibra de fibrio

La superficie lisa y sintética favorece su limpieza y desinfección lo que evita cualquier problema sanitario si se maneja adecuadamente.
Dado que estos estanques se utilizan para guardería, precrianza u hormonado, su capacidad se mide en número de organismos por m^3, así, vemos que un tanque de 2 m^3 puede albergar hasta 10,000 crías de tilapia durante su proceso de hormonado contando con un sistema de aireación eficiente.

2.1.15.4 El estanque rústico.

Es más barato de construir por m^2, si tiene un suministro de agua adecuado, puede no necesitar aireación, sus bordos ayudan a mantener una temperatura más estable del agua, sin embargo, requieren un programa de mantenimiento permanente.

Figuera 2.6 Engorda en estanque rústico

A continuación se describen algunas estructuras básicas de una estanquería rústica.

Los estanques rústicos deben ser construidos en suelos impermeables, para evitar filtraciones y deterioro de sus estructuras.
Los estanques deben rehabilitarse antes de cada ciclo de cultivo.
Especialmente cuando se trata de estanquería rústica, para prevenir erosión, deslaves o incluso rompimientos.
Se debe dar mantenimiento a bordos, fondo de los estanques canal de llamada y drenes.
Hay que reforzar coronas, taludes y fondos.

2.1.15.5 Fondos

El fondo del estanque debe ser bien compactado, con una pendiente que permita un vaciado rápido y total, sin dejar áreas con agua (charcos).
Desde cualquier punto del fondo del estanque, la pendiente deberá estar dirigida hacia los monjes de drenaje.
El fondo debe ser tratado después de cada ciclo con una lechada de cal viva, esto es un mantenimiento sanitario preventivo, en el cual después de aplicar la lechada de cal, el suelo se voltea con arado, y se seca completamente antes de ser vuelto a llenar.

Las zonas que lleguen a presentar charcos durante su drenado, deberán ser niveladas.
Se debe recoger cualquier basura o material extraño del fondo, eliminar vegetación y raíces.

Se debe prevenir la erosión diseñando un adecuado sistema de llenado, colocando incluso un lecho de roca o una losa en la entrada del agua.
Normalmente el fondo del estanque al final de cada ciclo se llena de baches provocados por los nidos de las tilapias en cultivo, esto no le afecta siempre y cuando las pendientes para el drenado sean las adecuadas.

2.1.15.6 Bordos.

Los bordos del estanque se componen de corona, base y taludes.
La corona es la superficie del bordo, que debe ser un piso regular, bien aplanado y compactado y con una anchura mínima suficiente para el transito de un vehículo ligero para las labores de cosecha y mantenimiento. Debe estar totalmente libre de obstáculos, totalmente limpia y debe darse mantenimiento para prevenir la erosión.
Los taludes deben tener una pendiente suave, para prevenir la erosión, además de dar fortaleza a la base y por tanto a la estructura del bordo.
Se deben mantener libres de vegetación, la falta de mantenimiento favorecen la proliferación de depredadores como insectos, aves y reptiles.

2.2 MATLAB

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

2.2.1 Historia de MATLAB

Fue creado por The MathWorks en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices LINPACK y EISPACK sin tener que usar Fortran.

En 2004, se estimaba que MATLAB era empleado por más de un millón de personas en ámbitos académicos y empresariales.

2.3 Adquisición de datos

La adquisición de datos o adquisición de señales, consiste en la toma de muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador u otras electrónicas (sistema digital). Consiste, en tomar un conjunto de señales físicas, convertirlas en tensiones eléctricas y digitalizarlas de manera que se puedan procesar en una computadora o PAC. Se requiere una etapa de acondicionamiento, que adecua la señal a niveles compatibles con el elemento que hace la transformación a señal digital. El elemento que hace dicha transformación es el módulo de digitalización o tarjeta de Adquisición de Datos (DAQ).

2.3.1 Proceso de adquisición de datos

2.3.1.1 Definiciones

Dato: Representación simbólica (numérica, alfabética...), atributo o característica de un valor. No tiene sentido en sí mismo, pero convenientemente tratado (procesado) se puede utilizar en la relación de cálculos o toma de decisiones.

Adquisición: Recogida de un conjunto de variables físicas, conversión en voltaje y digitalización de manera que se puedan procesar en un ordenador.

Sistema: Conjunto organizado de dispositivos que interactúan entre sí ofreciendo prestaciones más completas y de más alto nivel. Una vez que las señales eléctricas se transformaron en digitales, se envían a través del bus de datos a la memoria del PC. Una vez los datos están en memoria pueden procesarse con una aplicación adecuada, archivarlas en el disco duro, visualizarlas en la pantalla, etc...

Bit de resolución: Número de bits que el convertidor analógico a digital (ADC) utiliza para representar una señal.

Rango: Valores máximo y mínimo entre los que el sensor, instrumento o dispositivo funcionan bajo unas especificaciones.

Teorema de Nyquist: Al muestrear una señal, la frecuencia de muestreo debe ser mayor que dos veces el ancho de banda de la señal de entrada, para poder reconstruir la señal original de forma exacta a partir de sus muestras. En caso contrario, aparecerá el fenómeno del aliasing que se produce al infra-muestrear. Si la señal sufre aliasing, es imposible recuperar el original. Velocidad de muestreo recomendada:

–2*frecuencia mayor (medida de frecuencia)

–10*frecuencia mayor (detalle de la forma de onda)

Los componentes de los sistemas de adquisición de datos, poseen sensores adecuados que convierten cualquier parámetro de medición de una señal eléctrica, que se adquiriere por el hardware de adquisición de datos. Los datos adquiridos se visualizan, analizan, y almacenan en un ordenador, ya sea utilizando el proveedor de software suministrado u otro software. Los controles y visualizaciones se pueden desarrollar utilizando varios lenguajes de programación de propósito general como VisualBASIC, C++, Fortran, Java, Lisp, Pascal. Los lenguajes especializados de programación utilizados para la adquisición de datos incluyen EPICS, utilizada en la construcción de grandes sistemas de adquisición de datos, LabVIEW, que ofrece un entorno gráfico de programación optimizado para la adquisición de datos, y MATLAB. Estos entornos de adquisición proporcionan un lenguaje de programación además de bibliotecas y herramientas para la adquisición de datos y posterior análisis.

De la misma manera que se toma una señal eléctrica y se transforma en una digital para enviarla al ordenador, se puede también tomar una señal digital o binaria y convertirla en una eléctrica. En este caso el elemento que hace la transformación es una tarjeta o módulo de Adquisición de Datos de salida, o tarjeta de control. La señal dentro de la memoria del PC la genera un programa adecuado a las aplicaciones que quiere el usuario y, luego de procesarla, es recibida por mecanismos que ejecutan movimientos mecánicos, a través de servomecanismos, que también son del tipo transductores.

Un sistema típico de adquisición utiliza sensores, transductores, amplificadores, convertidores analógico - digital (A/D) y digital - analógico (D/A), para procesar información acerca de un sistema físico de forma digitalizada.

2.3.2 ¿Cómo se adquieren los datos a la computadora?

La adquisición de datos se inicia con el fenómeno físico o la propiedad física de un objeto (objeto de la investigación) que se desea medir. Esta propiedad física o fenómeno podría ser el cambio de temperatura o la temperatura de una habitación, la intensidad o intensidad del cambio de una fuente de luz, la presión dentro de una cámara, la fuerza aplicada a un objeto, o muchas otras cosas. Un eficaz sistema de adquisición de datos pueden medir todos estos diferentes propiedades o fenómenos.

Un sensor es un dispositivo que convierte una propiedad física o fenómeno en una señal eléctrica correspondiente medible, tal como tensión, corriente, el cambio en los valores de resistencia o condensador, etc. La capacidad de un sistema de adquisición de datos para medir los distintos fenómenos depende de los transductores para convertir las señales de los fenómenos físicos mensurables en la adquisición de datos por hardware. Transductores son sinónimo de sensores en sistemas de DAQ. Hay transductores específicos para diferentes aplicaciones, como la medición de la temperatura, la presión, o flujo de fluidos. DAQ también despliega diversas técnicas de acondicionamiento de Señales para modificar adecuadamente diferentes señales eléctricas en tensión, que luego pueden ser digitalizados usando CED.

Las señales pueden ser digitales (también llamada señales de la lógica) o analógicas en función del transductor utilizado.

El acondicionamiento de señales suele ser necesario si la señal desde el transductor no es adecuado para la DAQ hardware que se utiliza. La señal puede ser amplificada o desamplificada, o puede requerir de filtrado, o un cierre patronal, en el amplificador se incluye para realizar demodulación. Varios otros ejemplos de acondicionamiento de señales podría ser el puente de conclusión, la prestación actual de tensión o excitación al sensor, el aislamiento, linealización, etc. Este pretratamiento del señal normalmente lo realiza un pequeño módulo acoplado al transductor.

DAQ hardware son por lo general las interfaces entre la señal y un PC. Podría ser en forma de módulos que pueden ser conectados a la computadora de los puertos (paralelo, serie, USB, etc...) o ranuras de las tarjetas conectadas a (PCI, ISA) en la placa madre. Por lo general, el espacio en la parte posterior de una tarjeta PCI es demasiado pequeño para todas las conexiones necesarias, de modo que una ruptura de caja externa es obligatorio. El cable entre este recuadro y el PC es cara debido a los numerosos cables y el blindaje necesario y porque es exótico. Las tarjetas DAQ a menudo contienen múltiples componentes (multiplexores, ADC, DAC, TTL-IO, temporizadores de alta velocidad, memoria RAM). Estos son

accesibles a través de un bus por un micro controlador, que puede ejecutar pequeños programas. El controlador es más flexible que una unidad lógica dura cableada, pero más barato que una CPU de modo que es correcto para bloquear con simples bucles de preguntas.

Driver software normalmente viene con el hardware DAQ o de otros proveedores, y permite que el sistema operativo pueda reconocer el hardware DAQ y dar así a los programas acceso a las señales de lectura por el hardware DAQ. Un buen conductor ofrece un alto y bajo nivel de acceso.

Ejemplos de Sistemas de Adquisición y control: · DAQ para recoger datos(datalogger) medioambientales (energías renovables e ingeniería verde). · DAQ para audio y vibraciones (mantenimiento, test). · DAQ + control de movimiento(corte con laser). · DAQ + control de movimiento+ visión artificial (robots modernos).

2.3.3 Tiempo de conversión

Es el tiempo que tarda en realizar una medida el convertidor en concreto, y dependerá de la tecnología de medida empleada. Evidentemente nos da una cota máxima de la frecuencia de la señal a medir.

Este tiempo se mide como el transcurrido desde que el convertidor recibe una señal de inicio de "conversión" (normalmente llamada SOC, Start of Conversión) hasta que en la salida aparece un dato válido. Para que tengamos constancia de un dato válido tenemos dos caminos:

- Esperar el tiempo de conversión máximo que aparece en la hoja de características.
- Esperar a que el convertidor nos envíe una señal de fin de conversión.

Si no respetamos el tiempo de conversión, en la salida tendremos un valor, que dependiendo de la constitución del convertidor será:

- Un valor aleatorio, como consecuencia de la conversión en curso
- El resultado de la última conversión

2.3.4 La etapa de acondicionamiento de la señal

Con más detalle, en una etapa de acondicionamiento podemos encontrar estas etapas, aunque no todas están siempre presentes:

- Amplificación
- Excitación

- Filtrado
- Multiplexado
- Aislamiento
- Linealización

Amplificación Es el tipo más común de acondicionamiento. Para conseguir la mayor precisión posible la señal de entrada deber ser amplificada de modo que su máximo nivel coincida con la máxima tensión que el convertidor pueda leer.

Aislamiento - Otra aplicación habitual en el acondicionamiento de la señal es el aislamiento eléctrico entre el transductor y el ordenador, para proteger al mismo de transitorios de alta tensión que puedan dañarlo. Un motivo adicional para usar aislamiento es el garantizar que las lecturas del convertidor no son afectadas por diferencias en el potencial de masa o por tensiones en modo común.

Cuando el sistema de adquisición y la señal a medir están ambas referidas a masa pueden aparecer problemas si hay una diferencia de potencial entre ambas masas, apareciendo un "bucle de masa", que puede devolver resultados erróneos.

Multiplexado - El multiplexado es la conmutación de las entradas del convertidor, de modo que con un sólo convertidor podemos medir los datos de diferentes canales de entrada. Puesto que el mismo convertidor está midiendo diferentes canales, su frecuencia máxima de conversión será la original dividida por el número de canales muestreados. Se aconseja que los multiplexores se utilizen antes del conversor y después del condicionamiento del señal, ya que de esta manera no molestará a los aislantes que podamos tener.

Filtrado - El fin del filtro es eliminar las señales no deseadas de la señal que estamos observando. Por ejemplo, en las señales cuasi-continuas, (como la temperatura) se usa un filtro de ruido de unos 4 Hz, que eliminará interferencias, incluidos los 50/60 Hz de la red eléctrica.

Las señales alternas, tales como la vibración, necesitan un tipo distinto de filtro, conocido como filtro antialiasing, que es un filtro pasabajo pero con un corte muy brusco, que elimina totalmente las señales de mayor frecuencia que la máxima a medir, ya que se si no se eliminasen aparecerían superpuestas a la señal medida, con el consiguiente error.

Excitación - La etapa de acondicionamiento de señal a veces genera excitación para algunos transductores, como por ejemplos las galgas "extesométricas", "termistores" o "RTD", que necesitan de la misma, bien por

su constitución interna, (como el termistor, que es una resistencia variable con la temperatura) o bien por la configuración en que se conectan (como el caso de las galgas, que se suelen montar en un puente de Wheatstone).

Linealización - Muchos transductores, como los termopares, presentan una respuesta no lineal ante cambios lineales en los parámetros que están siendo medidos. Aunque la linealización puede realizarse mediante métodos numéricos en el sistema de adquisición de datos, suele ser una buena idea el hacer esta corrección mediante circuitería externa.

2.4 Qué es y para qué sirve el SQL

Las aplicaciones en red son cada día más numerosas y versátiles. En muchos casos, el esquema básico de operación es una serie de scripts que rigen el comportamiento de una base de datos.

Debido a la diversidad de lenguajes y de bases de datos existentes, la manera de comunicar entre unos y otras sería realmente complicada a gestionar de no ser por la existencia de estándares que nos permiten el realizar las operaciones básicas de una forma universal.

Es de eso de lo que trata el Structured Query Language que no es mas que un lenguaje estándar de comunicación con bases de datos. Hablamos por tanto de un lenguaje normalizado que nos permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL...).

El hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos. En efecto, determinadas bases de datos implementan funciones específicas que no tienen necesariamente que funcionar en otras.

Aparte de esta universalidad, el SQL posee otras dos características muy apreciadas. Por una parte, presenta una potencia y versatilidad notables que contrasta, por otra, con su accesibilidad de aprendizaje.

El manual de SQL de desarrolloweb pretende dar a conocer las operaciones básicas que se pueden realizar con SQL y que tienen una aplicación directa

con la creación de aplicaciones en red sin profundizar más de lo estrictamente necesario. Buscamos con ello ofrecer al webmaster un manual de referencia práctico y aplicado.

2.4.1 Diferentes tipos campos empleados en las bases de datos

Como sabemos una base de datos esta compuesta de tablas donde almacenamos registros catalogados en función de distintos campos (características).

Un aspecto previo a considerar es la naturaleza de los valores que introducimos en esos campos. Dado que una base de datos trabaja con todo tipo de informaciones, es importante especificarle qué tipo de valor le estamos introduciendo de manera a, por un lado, facilitar la búsqueda posteriormente y por otro, optimizar los recursos de memoria.

Cada base de datos introduce tipos de valores de campo que no necesariamente están presentes en otras. Sin embargo, existe un conjunto de tipos que están representados en la totalidad de estas bases. Estos tipos comunes son los siguientes:

Tabla 2.4

Alfanuméricos	Contienen cifras y letras. Presentan una longitud limitada (255 caracteres)
Numéricos	Existen de varios tipos, principalmente, enteros (sin decimales) y reales (con decimales).
Booleanos	Poseen dos formas: Verdadero y falso (Sí o No)
Fechas	Almacenan fechas facilitando posteriormente su explotación. Almacenar fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra...
Memos	Son campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados (veremos más adelante lo que esto quiere decir).

Autoincrementables	Son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta más que evidente: Servir de identificador ya que resultan exclusivos de un registro.

2.4.2 Sintaxis y ejemplos para introducir registros en una tabla

Los registros pueden ser introducidos a partir de sentencias que emplean la instrucción Insert.

La sintaxis utilizada es la siguiente:

Insert Into nombre_tabla (nombre_campo1, nombre_campo2,...) Values (valor_campo1, valor_campo2...)

Un ejemplo sencillo a partir de nuestra tabla modelo es la introducción de un nuevo cliente lo cual se haría con una instrucción de este tipo:

Insert Into clientes (nombre, apellidos, direccion, poblacion, codigopostal, email, pedidos) Values ('Perico', 'Palotes', 'Percebe n°13', 'Lepe', '123456', 'perico@desarrolloweb.com', 33)

Como puede verse, los campos no numéricos o booleanos van delimitados por apostrofes: '. También resulta interesante ver que el código postal lo hemos guardado como un campo no numérico. Esto es debido a que en determinados paises (Inglaterra,como no) los codigos postales contienen también letras.

> **Nota:** Si deseamos practicar con una base de datos que está vacía primero debemos crear las tablas que vamos a llenar. Las tablas también se crean con sentencias SQL.
>
> Aunque, de todos modos, puede que sea más cómodo utilizar un programa con interfaz gráfica, como Access, que nos puede servir para crear las tablas en bases de datos del propio Access o por ODBC a otras bases de datos como SQL Server o MySQL, por poner

dos ejemplos.

Otra posibilidad en una base de datos como MySQL, sería crear las tablas utilizando un software como PhpMyAdmin.

Por supuesto, no es imprescindible rellenar todos los campos del registro. Eso sí, puede ser que determinados campos sean necesarios. Estos campos necesarios pueden ser definidos cuando construimos nuestra tabla mediante la base de datos.

Nota: Si no insertamos uno de los campos en la base de datos se inicializará con el valor por defecto que hayamos definido a la hora de crear la tabla. Si no hay valor por defecto, probablemente se inicialice como NULL (vacío), en caso de que este campo permita valores nulos. Si ese campo no permite valores nulos (eso se define también al crear la tabla) lo más seguro es que la ejecución de la sentenca SQL nos de un error.

Resulta muy interesante, ya veremos más adelante el por qué, el introducir durante la creación de nuestra tabla un campo autoincrementable que nos permita asignar un único número a cada uno de los registros. De este modo, nuestra tabla clientes presentaría para cada registro un número exclusivo del cliente el cual nos será muy util cuando consultemos varias tablas simultáneamente.

CAPITULO 3: DESARROLLO DEL PROYECTO

3.1 CONSTRUCCION DE UNA INTERFACE USB USANDO UN PIC Y ADQUISICIÓN DE DATOS

Para la construcción de una interface por usb se presenta la comunicación USB de tipo Bulk Transfer utilizando matlab y un microcontrolador de microchip. Con este método se puede enviar y recibir datos masivos de información hasta una velocidad de 12Mbps. La comunicación se realiza a través de un subvi(picusb) creado con uso de la librería *mpusbapi.dll* que nos proporciona microchip. El puente que establece la comunicación entre el microcontrolador y la PC se realiza mediante las librerías USB del sofware CCS C Compiler y el driver *mchpusb.inf* para Microsoft Windows XP y Windows 7.

Implementamos el siguiente circuito básico para hacer funcionar nuestra aplicación USB.

Una observación en cuanto al monateje del circuito. Hay que tener mucho cuidado con el valor del capacitor que estara conectado en la entrada Vusb, por lo general se usa un capacitor de 47uF electrolitico o 470nf ceramico. La funcion que tienen estos capacitores es regular la señal que se trasmite entre los cables d+ y d-. Regula un volts de aproximadamente 3.3V sin ruidos, estos ruidos puede provocar envios de datos incorrectos. Sin este capacitor el dispositivo no funciona correctamente y no sera reconocido por el ordenador o host.

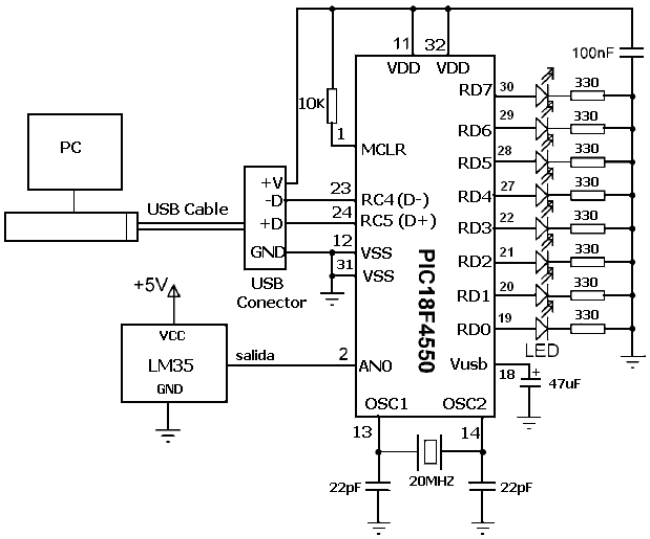


FIGURA 3.1

TABLA 3.1

Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

En la siguiente figura se muestra el diagrama de como funciona los reguladores de voltaje, los hay interno y externos. Nosotros utilizaremos el regulador interno, para ello conectamos el capacitor en la entrada Vusb

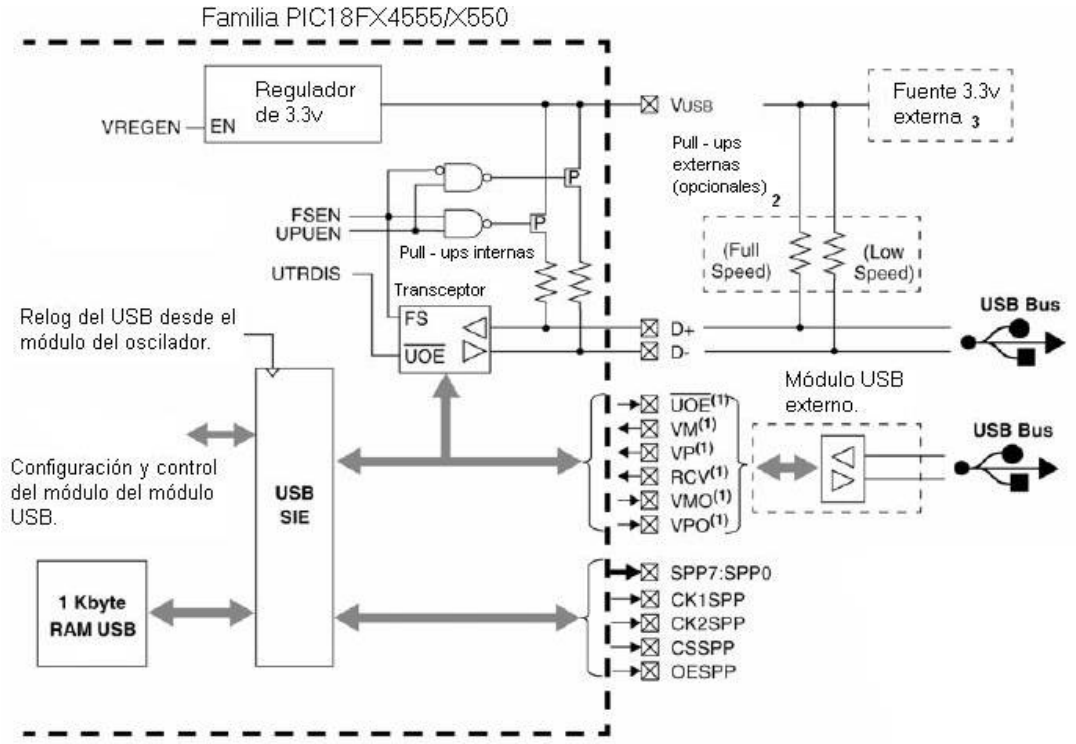


FIGURA 3.2

El regulador interno usb lo habilitamos con el siguiente codigo en CCS:

```
#fuse VREGEN
```

3.1.2Configuración de Oscilador

Otro aspecto muy importante es la configuración de oscilador en los PIC's de la familia 18Fxx5x que son los que soportan el USB 2.0(datasheet 18F2455-2550-4455-4550).

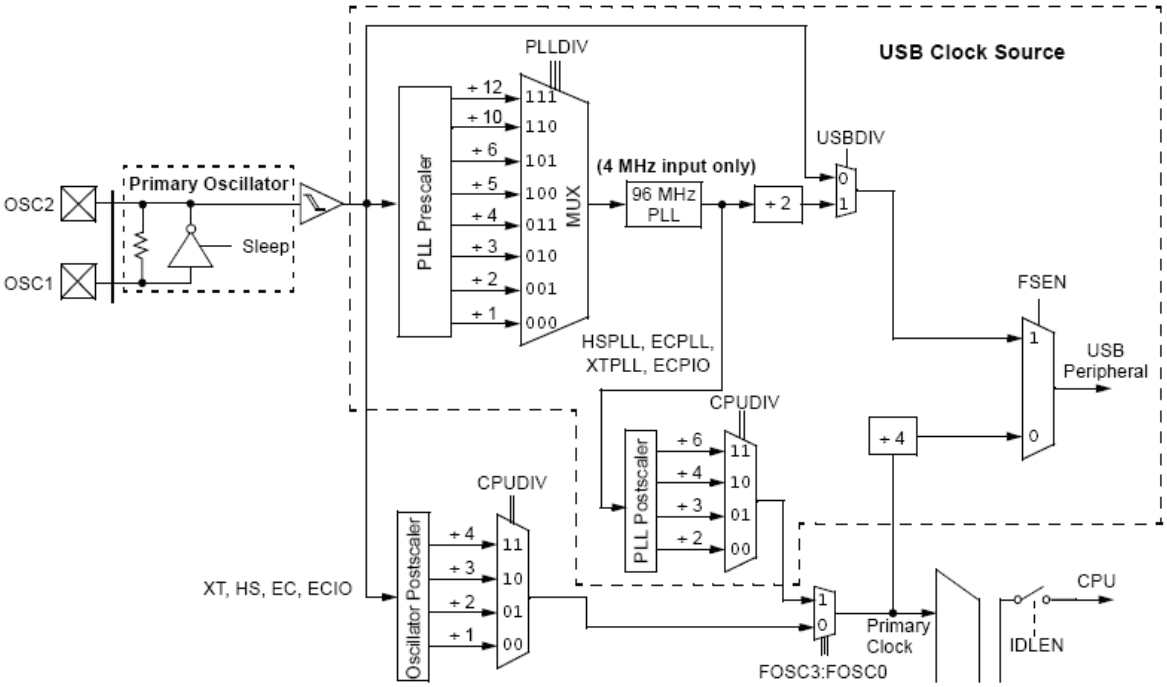


FIGURA 3.3

La configuracion del oscilador nos indica que se debe obtener una entrada de 4Mhz independientemente de cualquier crystal que se este utilizando.

El módulo USB Clock Source tiene a su entrada un PLL Prescaler, o sea un divisor de frecuencia. En cada una de sus salidas vamos a tener FOSC dividida por 1, 2, 3, 4, 5, 6, 10 ó 12. Y mediante PLLDIV que no es mas que un Multiplexor vamos a seleccionar la que deseamos usar.

Así si nuestro cristal es de 20 Mhz y en PLLDIV colocamos un 100 estaremos dividiendo por 5 el valor de FOSC con lo que tendremos 4 Mhz a la salida del MUX. Si por el contrario el cristal es de 4 Mhz y en PLLDIV colocamos un 000 entonces dividiremos por 1 FOSC con lo que tendremos también 4 Mhz a la salida del MUX.

Esta salida del MUX es lo que utilizamos para inyectársela al PLL de 96 Mhz. Si le metemos 4 Mhz él genera 96 Mhz. Es esta capacidad de pasar de 4 Mhz a 96 Mhz la que nos da la posibilidad de usar un montón de cristales distintos.

Pero 96 Mhz es el doble de lo que nos hace falta para el USB que son 48 Mhz. Asi que inmediatamente después tenemos que tener, y tenemos, un divisor por 2 que es el segundo camino por el que llegamos a USBDIV y en este caso le pondremos un 1 para usar la señal proveniente del PLL
Observemos que además de inyectar la señal oscilante en USBDIV también se conecta la señal del PLL a 96 Mhz en un Postscaler, otro divisor, en este caso por 2, 3, 4 ó 6 y cuyas señales van al CPUDIV. O sea que podemos generar una señal de reloj para nuestro PIC, no para el USB sino para la velocidad de ejecución de nuestro programa tomándola del PLL y que puede ser de 16 Mhz, 24 Mhz, 32 Mhz ó 48 Mhz.

Pero además la señal original llegaba en paralelo al Oscilator Postcaler, otro divisor más, que de forma directa, sin pasar por el módulo PLL nos divide la frecuencia original del cristal por 1, 2, 3 ó 4 y que también va a parar al CPUDIV pero desde otro origen. Con este módulo podemos obtener otra gama de frecuencias distinta para hacer correr el programa.

Cual de ambos CPUDIV vamos a utilizar lo seleccionamos con el switch FOSC3:FOSC0 que es de donde sacaremos la definitiva frecuencia de ejecución de programas.

Por último también tenemos disponible una entrada proveniente del Primary Clock y que dividida por 4 llega también a FSEN y podemos utilizarla en lugar de la que le llega desde el canal directo/PLL.

Como puedes ver es toda una maravilla cómo esta montado este tema de los osciladores, sobre todo por lo que respecta a las inmensas capacidades que tiene para hacer correr nuestro PIC a decenas de velocidades distintas siendo capaz, al mismo tiempo de tener disponibles los 48 Mhz imprescindibles para el USB 2.0.

Por lo tanto nuestro circuito o configuración quedaria de la siguiente manera:

Configuracion para obtener 48 Mhz para el USB y para la CPU a partir de un Xtal de 20 Mhz

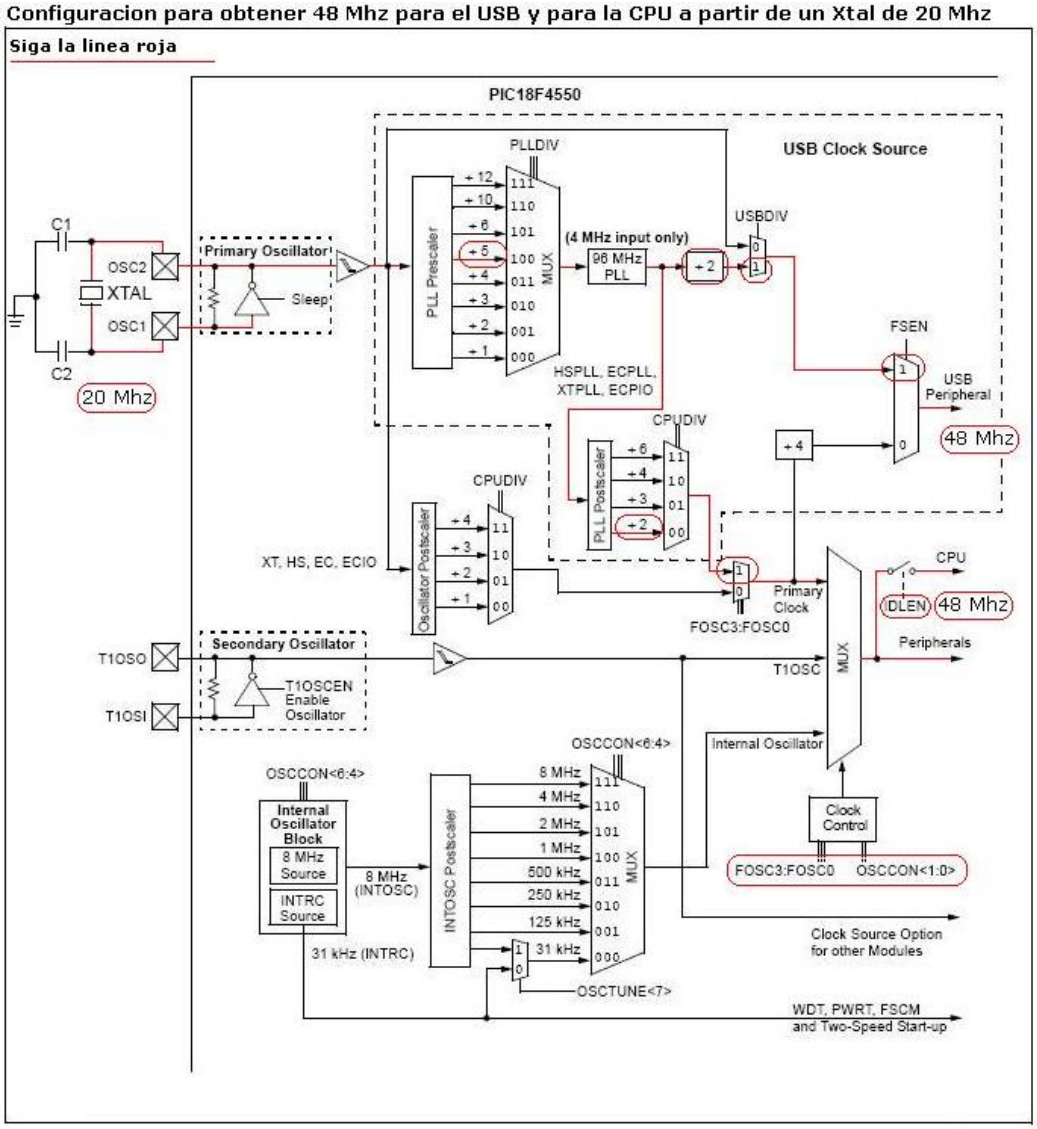


FIGURA 3.4

Realizado en CCS C :

```
#fuses
HSPLL,MCLR,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,
CPUDIV1,VREGE
```

3.1.3 Firmware PicUSB realizado en CCS C Compiler.

El firmware es un pequeño código de bajo nivel que se realiza para poder controlar entradas y salidas dependiendo de las necesidades del usuario. Además este pequeño código tendrá el objetivo de poder entenderse con la computadora o PC, realizando así la función de avisarte en el momento que ha sido conectado y detectado por la PC. En ese momento el firmware estará en espera de cualquier dato que se esté enviando por la PC atreves de cualquier software que se esté utilizando, A continuación se muestra en un pequeño diagrama a bloques de lo que hace el firmware y enseguida se muestra la programación en ccs c compiler.

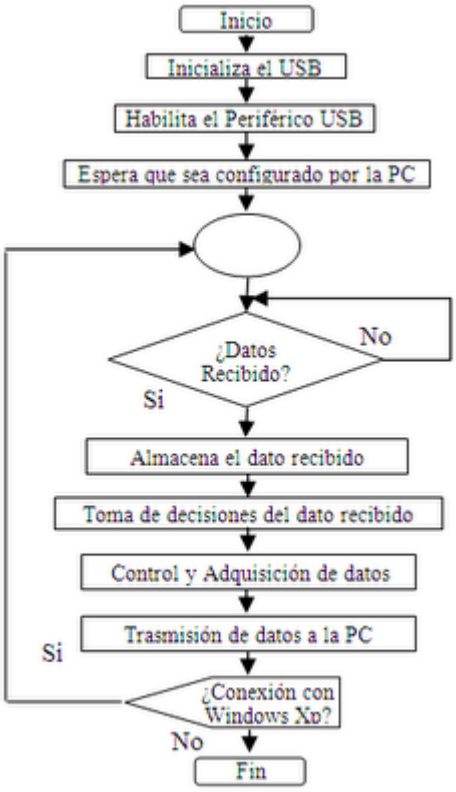


FIGURA 3.5

```
#include <18F4550.h>
#device ADC=10
#fuses
HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,
VREGEN,MCLR,NOPBADEN
#use delay(clock=48000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

//ESTAS DOS LINEAS HACEN QUE EL PIC PUEDA AUTO PROGRAMARCE


#build (reset=0x800:0x800,interrupt=0x808:0x808)
#org 0x000, 0x07FF{}


#define USB_HID_DEVICE     FALSE            //deshabilitamos el uso de las
directivas HID

#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK    //turn on
EP1(EndPoint1) for IN bulk/interrupt transfers

#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK    //turn on
EP1(EndPoint1) for OUT bulk/interrupt transfers

#define USB_EP1_TX_SIZE    64               //size to allocate for the tx
endpoint 1 buffer

#define USB_EP1_RX_SIZE    64               //size to allocate for the rx
endpoint 1 buffer

#include <pic18_usb.h>           //Microchip PIC18Fxx5x Hardware layer for
CCS's PIC USB driver
```

```
#include <usb_desc_scope.h>        //descriptors del Pic USB
#include <usb.c>                   //handles usb setup tokens and get descriptor
reports

#define  LEDV    PIN_D1
#define  LEDR    PIN_D0
#define  LED_ON  output_high
#define  LED_OFF output_low
#BYTE TRISA  = 0x0F92                    // Registro de control de E/S del
puerto A
#BYTE TRISD  = 0x0F93                    // Registro de control de E/S del
puerto D
#BYTE PORTA  = 0x0F80                    // Registro del puerto A
#BYTE PORTD  = 0x0F81                    // Registro del puerto D
#BYTE ADCON0  = 0x0FC2                   // Registro de control del ADC
#BYTE ADCON1  = 0x0FC1                   // Registro de control del ADC
#BYTE CMCON   = 0x0FB4                   // Registro del modulo
comparador

int8 dato[64];
unsigned int16 entrada;
void main(void) {

  LED_ON(LEDV);      //encendemos led en RD1 para indicar presencia de
energia
  LED_OFF(LEDR);

  usb_init();                      // inicializamos el USB
  usb_task();                      // habilita periferico usb e interrupciones
  usb_wait_for_enumeration();      // esperamos hasta que el PicUSB
sea configurado por el host


  LED_OFF(LEDV);
  LED_ON(LEDR);                    // encendemos led en RD0 al
establecer contacto con la PC
```

```
  TRISA  = 0x0FF;                    // Se declara el puerto A como entradas
(instrucción opcional)
  TRISD  = 0x00;                     // Se declara el puerto D como salidas
(instrucción opcional)
  ADCON1 = 0x0F;                     // Se configura al ADC para entradas
digitales (apagar el ADC)
  CMCON  = 0x07;                     // Se configuran los comparadores para
entradas digitales (apagar los comparadores)


//***************  CONFIGURACIÓN DEL ADC   **************
  setup_comparator(NC_NC_NC_NC);
    setup_port_a( ALL_ANALOG );      //habilitamos el puerto a para entrada
analogica
  setup_adc(ADC_CLOCK_INTERNAL);   //Utilizamos el reloj interno

  while (TRUE)
  {
    if(usb_enumerated())             // si el Pic está configurado via USB
    {
      if (usb_kbhit(1))              // si el endpoint de salida contiene datos
del host
      {
        usb_get_packet(1, dato, 64);       // tomamos el paquete de tamaño
8bytes del EP1 y almacenamos en dato

        if (dato[0]==1){
        Delay_ms(10);
        set_adc_channel(0); // Tomamos datos del  (Pin4 RA0/AN0)
          delay_us(4);       // Hacemos un retardo de 4 ms
          entrada=read_adc();    // Leemos el dato

        dato[2]= make8(entrada,0);
        dato[3] = make8(entrada,1);
        usb_put_packet(1, dato, 64, USB_DTS_TOGGLE); //y enviamos el
mismo paquete de tamaño 64bytes del EP1 al PC
        }
```

```
          if (dato[0]==2){
          Delay_ms(10);
          set_adc_channel(1); // Tomamos datos del (Pin4 RA1/AN1)
            delay_us(4);       // Hacemos un retardo de 4 ms
            entrada=read_adc();    // Leemos el dato

          dato[2]= make8(entrada,0);
          dato[3] = make8(entrada,1);
          usb_put_packet(1, dato, 64, USB_DTS_TOGGLE); //y enviamos el
mismo paquete de tamaño 64bytes del EP1 al PC
          }

          if (dato[0]==3){
          Delay_ms(10);
          set_adc_channel(2); // Tomamos datos del  (Pin4 RA2/AN2)
            delay_us(4);       // Hacemos un retardo de 4 ms
            entrada=read_adc();    // Leemos el dato

          dato[2]= make8(entrada,0);
          dato[3] = make8(entrada,1);
          usb_put_packet(1, dato, 64, USB_DTS_TOGGLE); //y enviamos el
mismo paquete de tamaño 64bytes del EP1 al PC
          }

       }
      }
   }
}
```

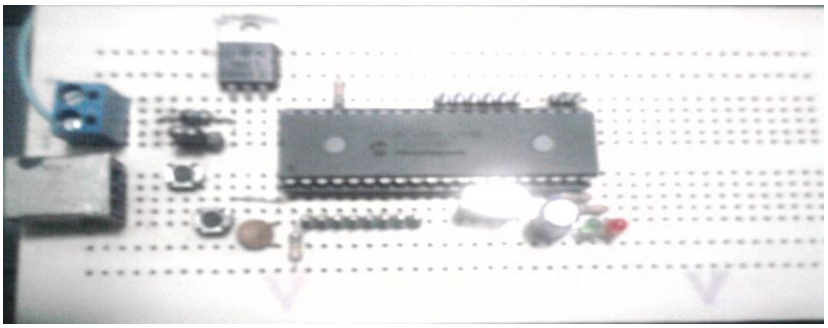


Figura 3.5a INTERFACE USB

3.2 CONFIGURAR E INSTALAR LA INTERFACE USB A LA COMPUTADORA

3.2.1 Configurando el Software

1. La DLL que proporciona Microchip se puede descargar desde su sitio web. (www.microchip.com) Asegúrese de obtener la versión más actual. En la dirección web que se menciona en la bibliografía se descarga el driver (link de acceso directo), en caso de haber caducado busque en la sección Aplication and markets ->USB -> MCHPFSUSB FrameworK ->Software/Tools-> << Microchip Application Libraries v2010-02-09 >> ó simplemente teclee “usb software/tools” en la ventanita de búsqueda y déle un clic en el botón “site search”, generalmente el acceso a la página queda en los primeros resultados de la búsqueda, el cual, al darle click lleva directamente al driver. En el mismo paquete incluye ejemplos que incluyen el programa fuente para la compresión de su uso.

2. Ejecute el driver descargado en el paso anterior e instale en la dirección que trae ya predeterminada. Este ejecutable trae muchos ejemplos de aplicación, entre ellos trae el driver que queda ubicado en: "C:\MICROCHIP SOLUTIONS\USB TOOLS\MCHPUSB CUSTOM DRIVER\MCHPUSB DRIVER\ RELEASE\ "

3. Instale el hardware a la PC de manera similar al que se instala un dispositivo USB que adquiere en el mercado: conecte al dispositivo a la PC, en cuanto le solicite los driver, sólo proporcione la dirección donde fue descomprimido el driver (la misma dirección del paso anterior). Si todo es correcto debemos de observar en el administrador de dispositivos un nuevo hardware que se agregó tal como se muestra en la figura 3.6. **NOTA:** Si Ud. Olvida ó no sustituye correctamente en el descriptor las 3 líneas que se comentan en el código del programa, el compilador CCS compilará correctamente pero al conectarse el PIC en la PC, éste no reconocerá el driver.

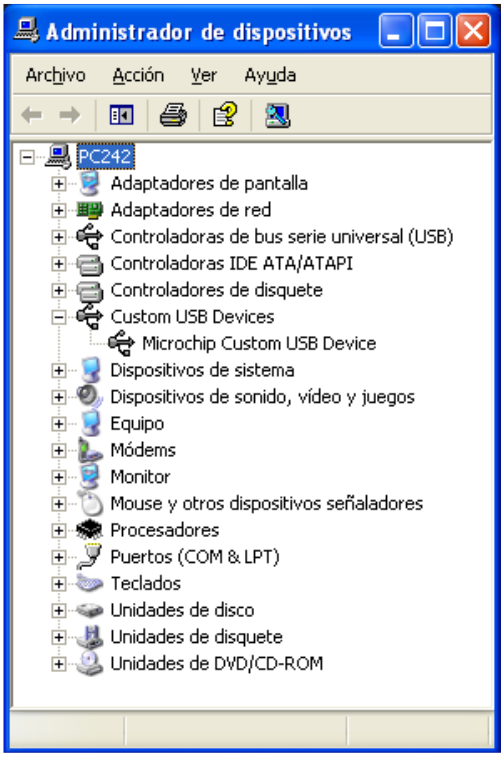


Fig. 3.6 Instalación del PIC en la PC

4. En propiedades del dispositivo instalado se puede observar el número PID&VID que se configuró en el PIC tal como se muestra en la figura 3.7.

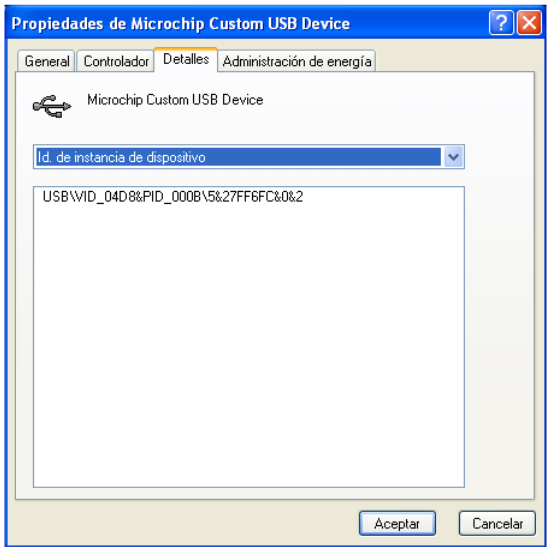


Fig.3. 7 En propiedades del PIC instalado en la PC se observa el número de PID&VID

3.3 ENLAZAR LA INTERFACE A MATLAB PARA LA TRANSFERENCIA DE DATOS

Para poder iniciar el enlace con el PIC es necesario que se haya concluido satisfactoriamente la instalación del paquete de drivers de Microchip. En el mismo paquete que se descarga de la red obtenemos instrucciones y ejemplos en C++ que muestran como manipular el driver y los parámetros requeridos para aplicarlo. Sin embargo, aún no es suficiente la información, porque la dificultad está en identificar el tipo de variable que acepta cada software de desarrollo y que también sea aceptada por la DLL. Por lo que de nuevo se sugiere consultar las notas de aplicaciones de Microchip y ejemplos publicados en la web (con cualquier software de desarrollo). Para hacer la conexión con MATLAB simplemente se “copiaron” las instrucciones al lenguaje de matlab y se acondicionaron las variables para que la DLL pueda reconocerlo.

NOTA: mas adelante se encuentra el código en matlab de de cómo el programa envía y recibe datos del PIC, los archivos _mpusbapi.c y mpusbapi.dll son necesarios para la ejecución en MATLAB y deben de estar en la misma carpeta que el archivo

Seguidamente se explica el desarrollo de la comunicación. Primero es necesario conocer qué función de MATLAB es más fácil de implementar:

Existen varios métodos, el que se utilizó fue: a. Manipular directamente la dll con la instrucción “Loadlibrary”

Ya que se manipula la librería de una manera directa y sin intermediarios.

Para iniciar con la implementación del código, primero inicie el programa de MATLAB

Siguiendo los pasos descritos en la sección de antecedentes de éste documento, las instrucciones que se requieren implementar tienen la siguiente secuencia y formato en MATLAB son:

a. Primero copie los archivos mpusbapi.c y mpusbapi.dll en la misma carpeta de trabajo (se obtienen de la descarga del driver en la página de microchip ("Microchip MCHPFSUSB v2.4 Installer.zip")), al instalarse queda ubicado en X:\Microchip Solutions\USB Tools\ MCHPUSB Custom Driver\ Mpusbapi\Dll\Borland_C, en caso de descargar una version de driver más reciente, reemplace éstos archivos por los más nuevos.

b. Se abre el editor de MATLAB y comenzamos cargando la librería en memoria.

Formato:

loadlibrary mpusbapi _mpusbapi.h alias
librería

c. Luego se identifica el número de dispositivos conectados con el PID&VID y ubicar el que corresponde al hardware de su desarrollo.

Formato:

[conectado] = calllib ('libreria',
'MPUSBGetDeviceCount', vid_pid_norm)

De donde:
vid_pid_norm = libpointer('int8Ptr',
[uint8('vid_04d8&pid_000b') 0]);

d. Seguidamente abra la pipe para leer (si no desea leer puede omitir éste paso).

Formato:

[my_out_pipe] = calllib('libreria',
'MPUSBOpen',uint8 (0), vid_pid_norm,
out_pipe, uint8(0), uint8 (0));

De donde:

vid_pid_norm = libpointer('int8Ptr',
[uint8('vid_04d8&pid_000b') 0]);
out_pipe = libpointer ('int8Ptr',
[uint8('\MCHP_EP1') 0]);

e. Siguiendo con la secuencia, abra la pipe para escribir (si no desea escribir puede omitir éste paso).

Formato:

[my_in_pipe] = calllib('libreria',
'MPUSBOpen',uint8 (0), vid_pid_norm,
in_pipe, uint8 (1), uint8 (0));

De donde:

vid_pid_norm = libpointer('int8Ptr',
[uint8('vid_04d8&pid_000b') 0]);
in_pipe = libpointer ('int8Ptr',
[uint8('\MCHP_EP1') 0]);

f. Lea los datos de la pipe (solamente si la pipe está abierta)

Formato:

**[aa,bb,data_in,dd] = calllib('libreria',
'MPUSBRead',my_in_pipe, data_in,
uint8(64), uint8(64), uint8(10));**

De donde:
data_in = eye(1,64,'uint8');

g. Escriba los datos de la pipe (solamente si la pipe está abierta)

Formato:

**calllib('libreria', 'MPUSBWrite',
my_out_pipe, data_out, uint8(64),
uint8(64), uint8(10));**

De donde:

data_out = eye(1,64,'uint8');

h. Cierre la(s) pipe(s) abierta(s), cada vez que finalice el programa, ya que si quedan abiertas windows genera errores y se pierde la comunicación.

Formato:

**calllib('libreria', 'MPUSBClose',
my_in_pipe);
calllib('libreria', 'MPUSBClose',
my_out_pipe);**

IMPORTANTE:

Al terminar el programa descargue la librería de memoria, ya que no se puedecargar más de una vez.

Formato:

unloadlibrary librería;

Una vez enlazado con el PIC, los datos pueden fluir las veces que sea necesario de un sentido a otro y manipularlos como se desee, ya que se tiene el completo control del software del PIC (por parte del Compilador C) y en la PC (por parte de MATLAB).

En caso de perderse la comunicación con el PIC (en casos donde el programa en MATLAB genere errores por cuestiones ajenas a la comunicación) desinstale del dispositivo “Microchip Custom USB Device” desde el administrador de dispositivos, desconecte el PIC del puerto USB, descargue la librería de memoria desde el MATLAB con “unloadlibrary librería” en la línea de comandos de MATLAB y resetee el PIC. Conecte de nuevo el PIC al puerto USB de su computadora y con eso es suficiente para restaurar las comunicaciones entre MATLAB y el PIC.

3.4 GENERAR UNA BASE DE DATOS Y CREAR TABLAS EN SQL

Abrimos sql y conectamos al servidor figura 3.8

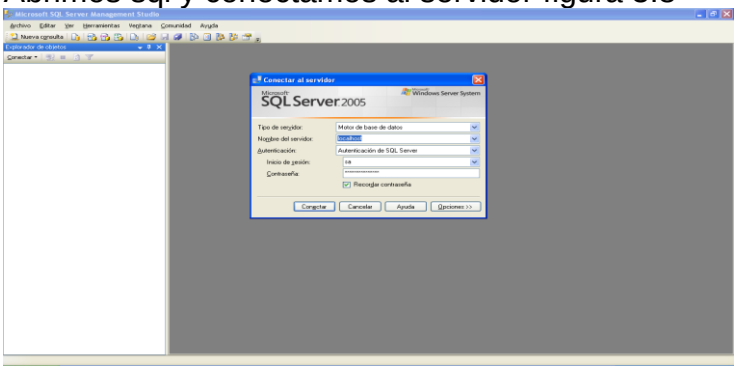


Figura 3.8

Una vez que sql se conecta al servidor desplegamos el menú de localhost y le damos clik derecho a base de datos y seleccionamos nueva base de datos como se muestra en la figura 3.9

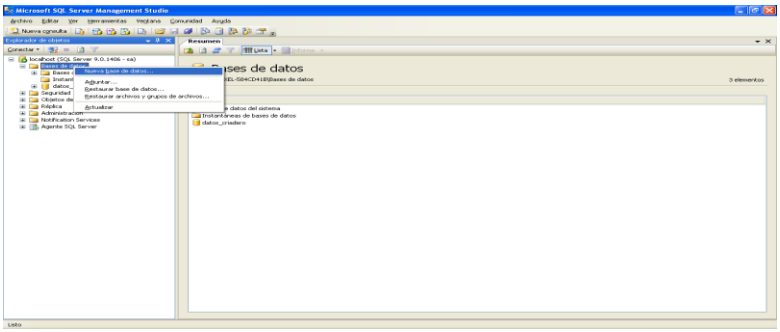


figura 3.9

Posteriormente le damos un nombre a la base de datos, en este caso se le puso (datos_criadero) y al propietario se le pone sa y se le da aceptar de esa manera se crea una nueva base de datos figura 3.10.

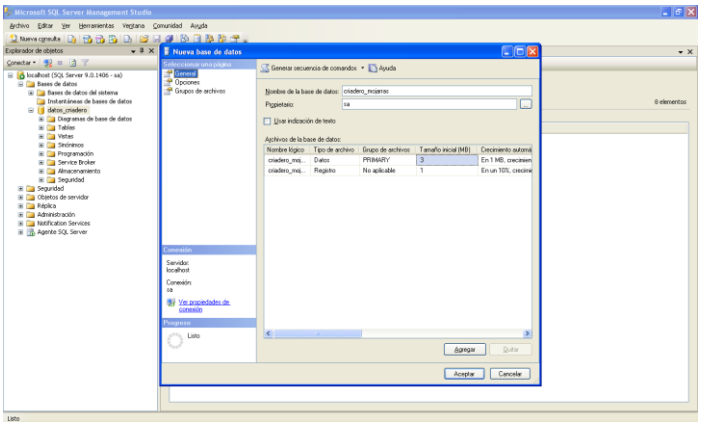


Figura 3.10

Una vez generado la base de datos se procede a la creación de tablas, para eso desplegamos el menú de la nueva base de datos creada (datos_criadero) y damos click derecho sobre la etiqueta tabla y seleccionamos crear nueva tabla como se indica en la figura 3.11

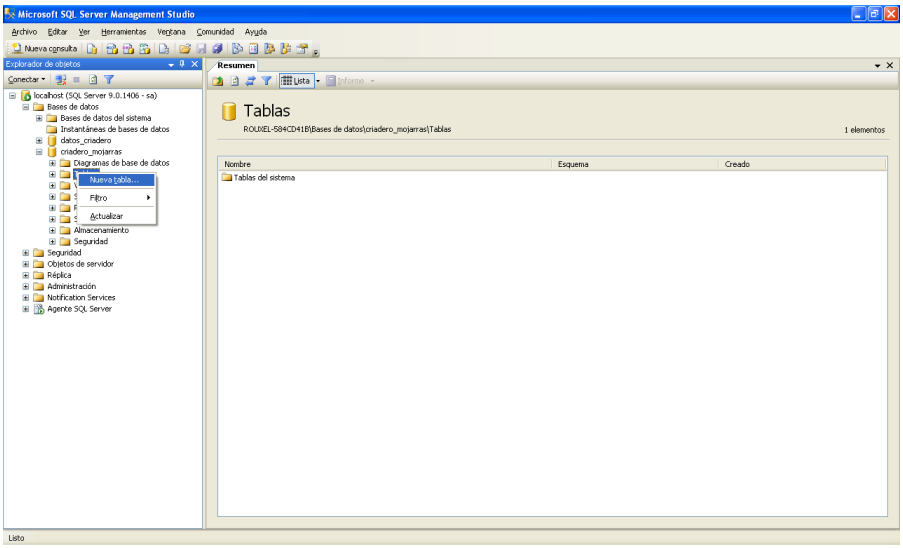


Figura 3.11

Después se abrirá una nueva ventana en donde introduciremos los nombres de cada columna, la primer columna se deberá llenar con el nombre de id y en tipo de datos será entero (int) y se deberá señalar como clave principal.

Posteriormente dependiendo el numero de columnas que utilizaremos se les nombrara y en tipo de datos se les ira asignando de acuerdo al dato que estas columnas vayan a recibir.

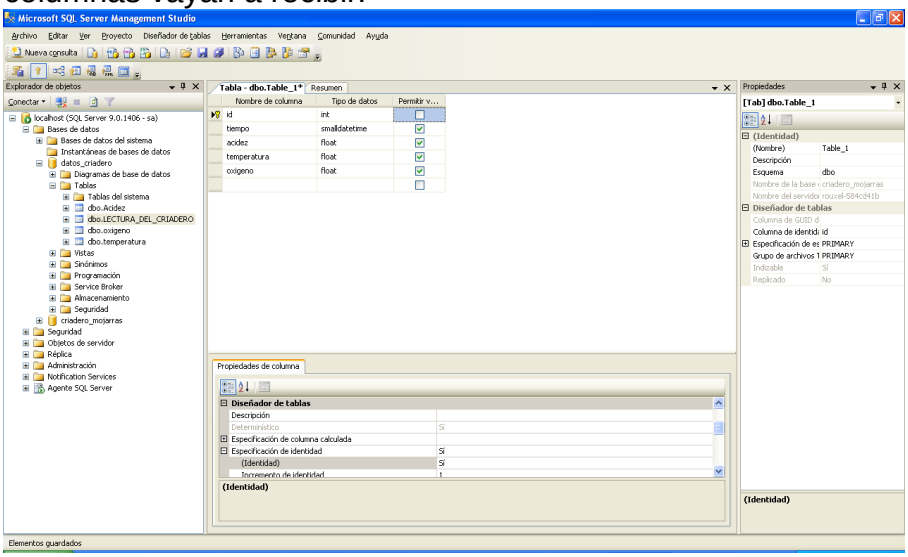


Figura 3.12

Posterior a esto se guarda la tabla y se le asigna un nombre en este caso (LECTURA_DEL_CRIADERO) y se le da aceptar, de esta forma queda nuestras tablas listas para recibir datos

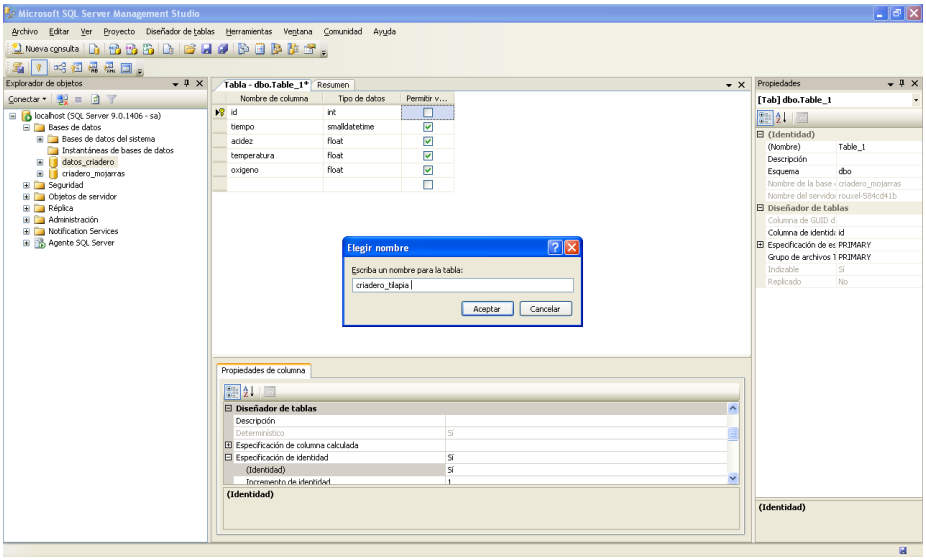


Figura 3.13

3.5 ENLAZAR MATLAB CON BASE DE DATOS DE SQL Y GUARDAR DATOS EN TABLAS

Para poder enlazar a SQL con MATLAB primero se genera la base de datos y se diseñan las tablas en las que se guardaran los datos, una vez que están listas las tablas procedemos a preparar el MATLAB para que con simples instrucciones podamos almacenar datos obtenidos del resultados o procedentes del exterior de la computadora.

Seguiremos los siguientes pasos para poder realizar la coneccion de SQL y MATLAB

Primero en el cuadro de comandos de MATLAB el comando :

querybuilder

Después de haber escrito el comando y se le haya dado aceptar, aparecerá una nueva ventana (figura 3.14) en la cual aremos las configuraciones necesarias para que MATLAB pueda comunicarse con las tablas que anteriormente generamos.

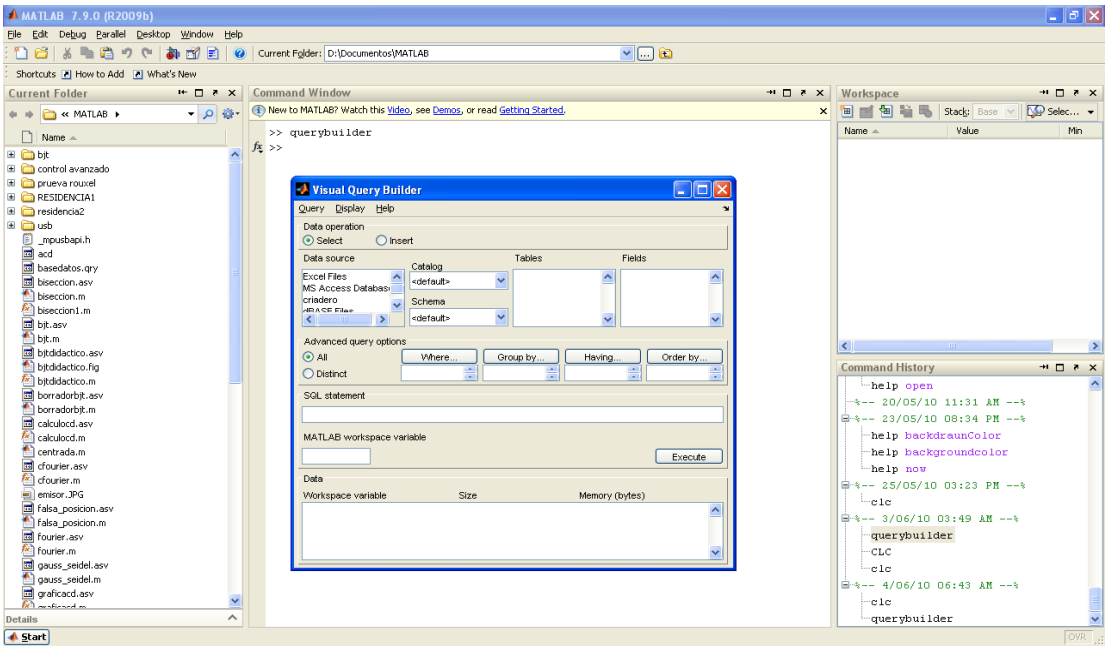


Figura 3.14

Como siguiente paso daremos un click en la pestaña que dice **Query,** esto hará que se despliegue un menú, enseguida daremos click sobre la opción que dice

define ODBC data source

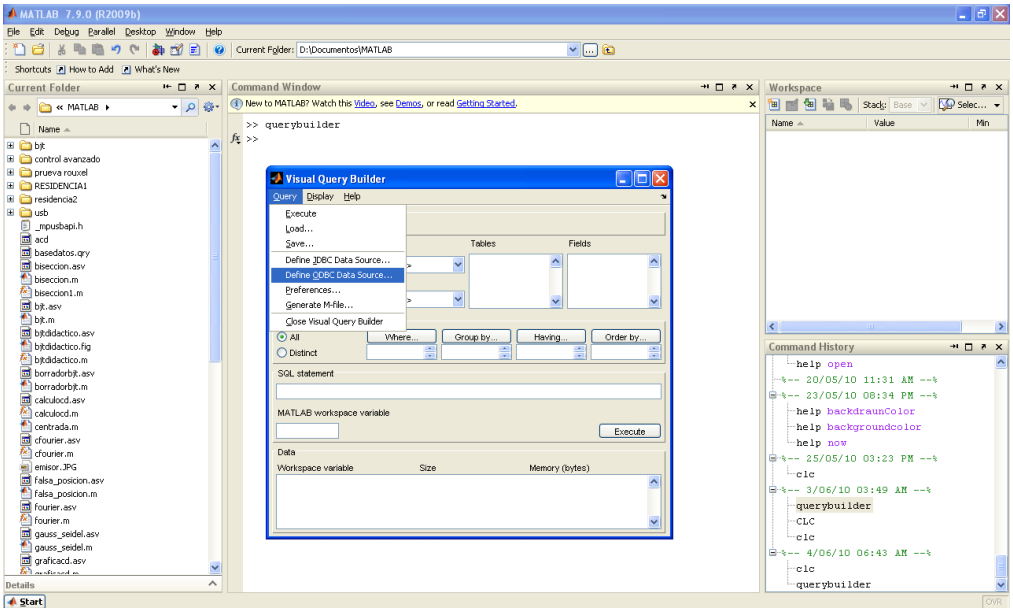


Figura 3.15

Posteriormente parecerá una nueva ventana que se llama administrador de orígenes de datos ODBC.

En la pestaña DSN de usuario le daremos click al botón agregar, que posteriormente nos aparecerá una nueva ventana, donde crearemos un nuevo origen de datos, en esta ventana nos aparece una lista de opciones de las cuales seleccionaremos la ultima opción de la lista SQL Server y daremos click en finalizar

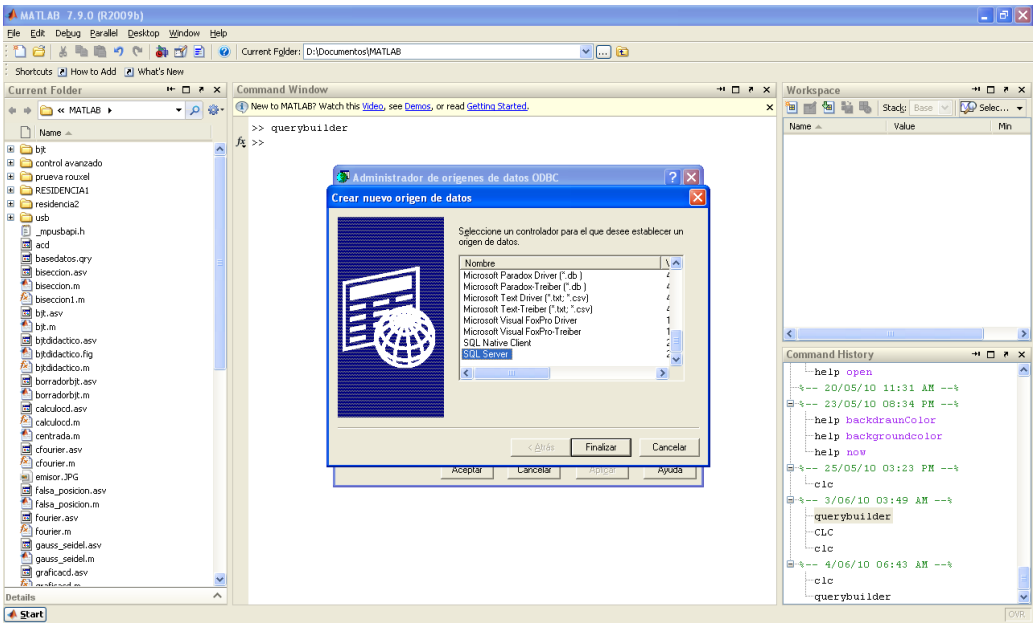


Figura 3.16

Después de dar finalizar nos aparece una nueva ventana con el nombre de crear un nuevo origen de datos para SQL server.

En nombre escribiremos el nombre con el que nos referiremos a la base de datos en este caso será: datos_criadero y en servidor escribiremos localhost como se muestra en la figura 3.17 y daremos siguiente

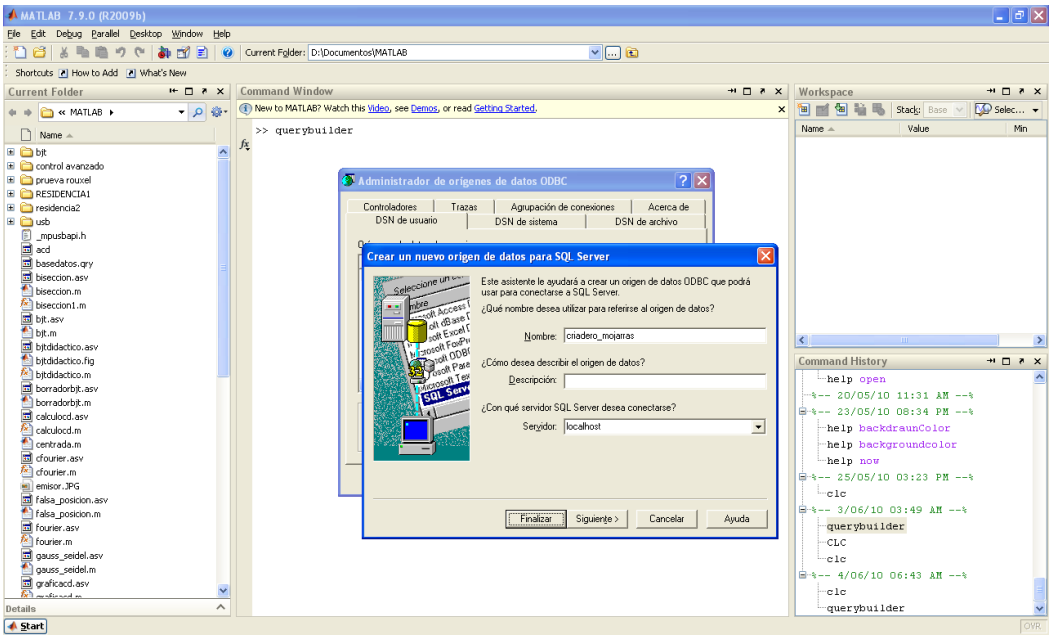


Figura 3.17

Aparecerá una nueva ventana figura 3.18 con el siguiente nombre: crear un nuevo origen de datos para SQL Server, en esa ventana habrá una pregunta que dirá lo siguiente:
¿Cómo desea que SQL Server compruebe la autenticidad del id. De inicio de sesión?
Seleccionaremos la segunda opción que dice:
Con la autentificación de SQL Server. Y en la id de inicio de secion pondremos “sa” y en contraseña pondremos la contraseña que se le asigna al SQL cuando se instala, en este caso “rouxel” posteriormente damos click en siguiente.

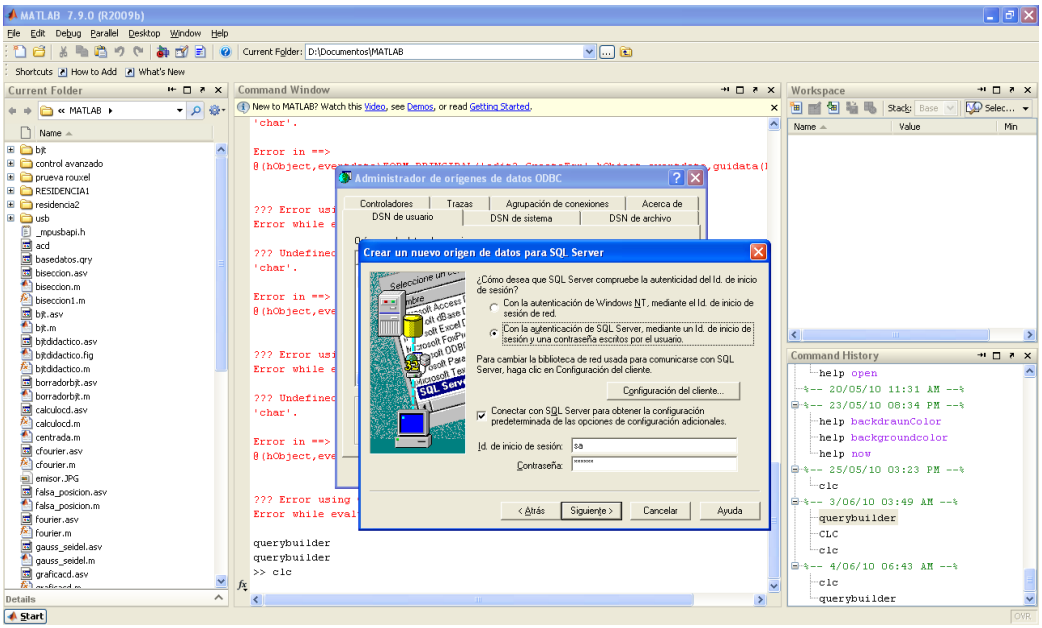


figura 3.18

Después de dar siguiente nos aparecerá una nueva ventana que se llama crear un nuevo origen de datos para SQL Server como se muestra en la figura 3.19

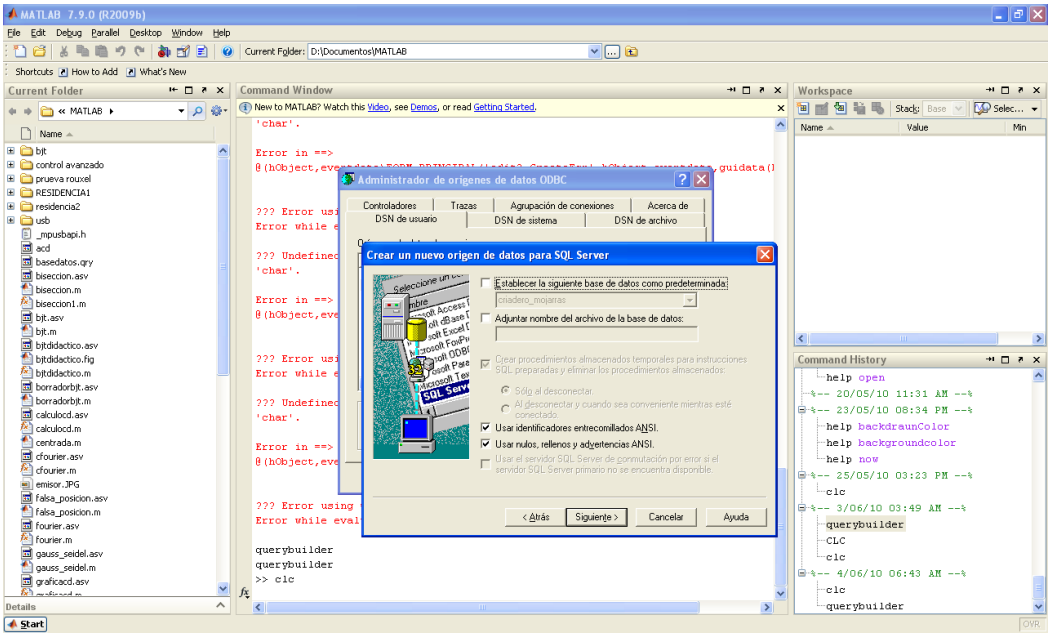


figura 3.19

Seleccionaremos el recuadro que dice: establecer la siguiente base de datos como predeterminada y seleccionaremos la que se llama datos_criadero, le

damos click a siguiente y nos aparecerá otra ventana a la cual le daremos finalizar como se muestra en la siguiente figura 3.20

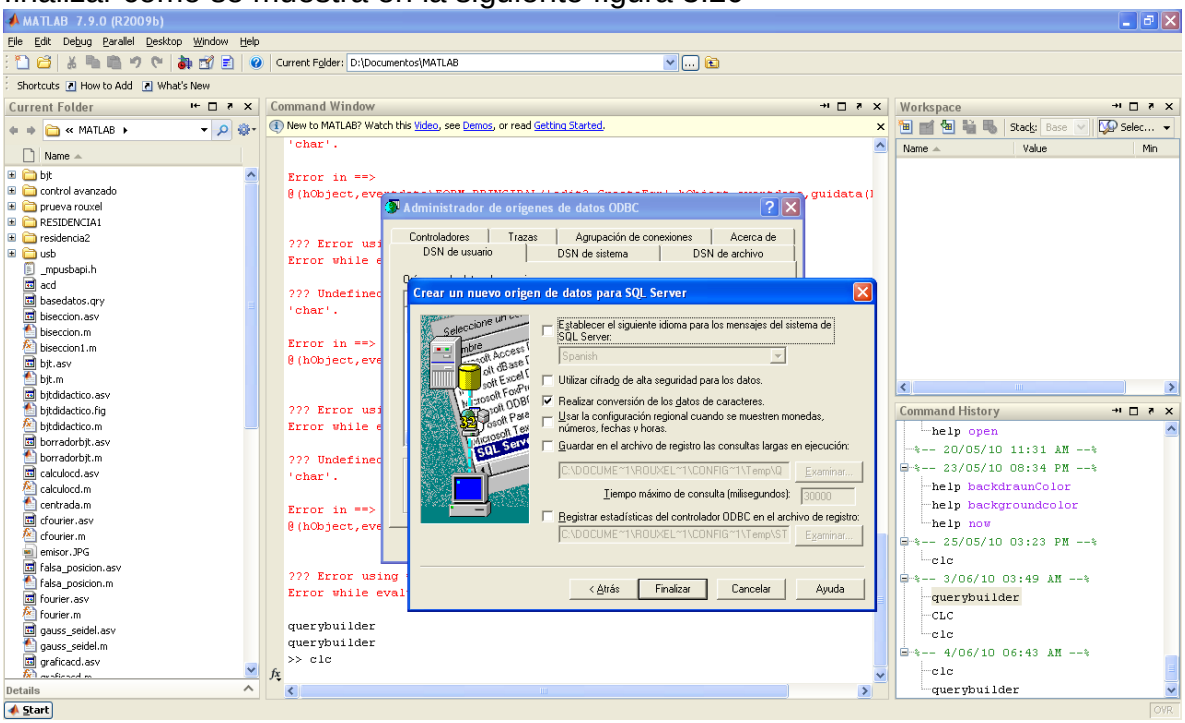


figura 3.20

Posteriormente quedara solo la ventana que se llama administrador de datos de orígenes de datos ODBC seleccionaremos la base de datos y le damos aceptar

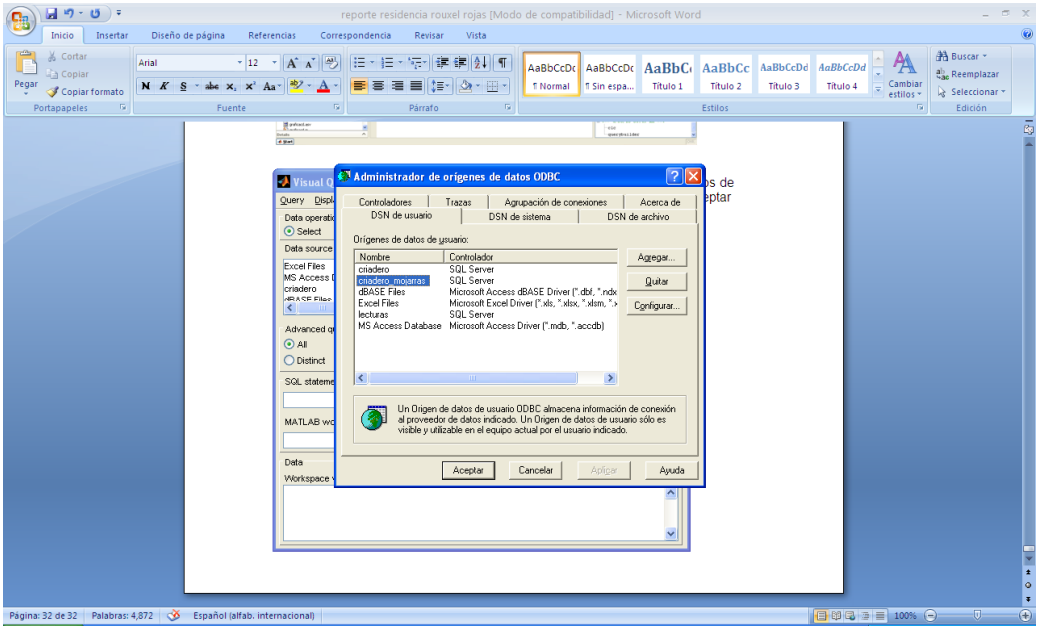


figura 3.21

Como siguiente paso volvemos a escribir en MATLAB el comando querybuilder donde nos abrirá nuevamente la ventana visual query builder y tiene varios submenús en data source seleccionaremos la base de datos datos_criadero y automáticamente nos pedirá un nombre de usuario y contraseña en el cual pondremos como nombre de usuario “sa” y en contraseña rouxel como se muestra en la figura 3.22

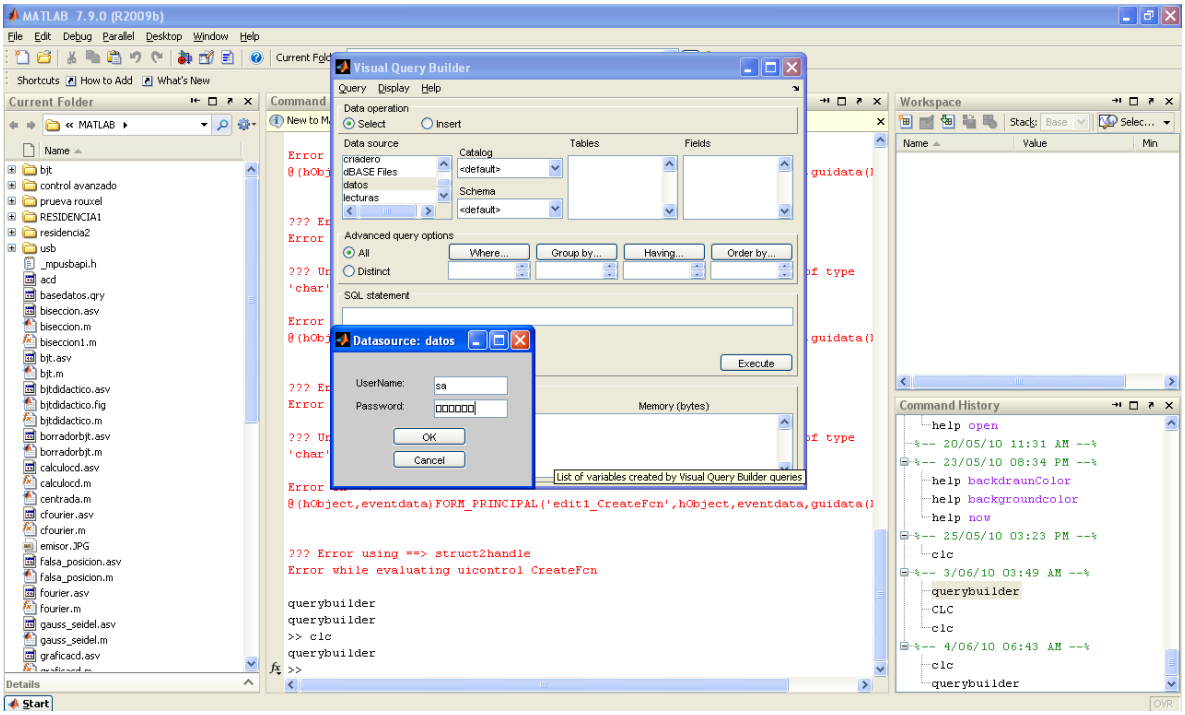


figura 3.22

Después de darle click a “ok” en el submenú tablas aparecerán todas las tablas existentes en esa base de datos y nosotros elegiremos la de lectura_del_criadero que fue la que creamos, automáticamente en el submenú fields aparecerá todos los nombres de las columnas de la taba, seleccionaremos todos. En opciones avanzadas de query seleccionaremos distinct y en MATLAB workspace variable le pondremos un nombre en este caso conexión y le daremos click en execute y cerraremos la ventana.

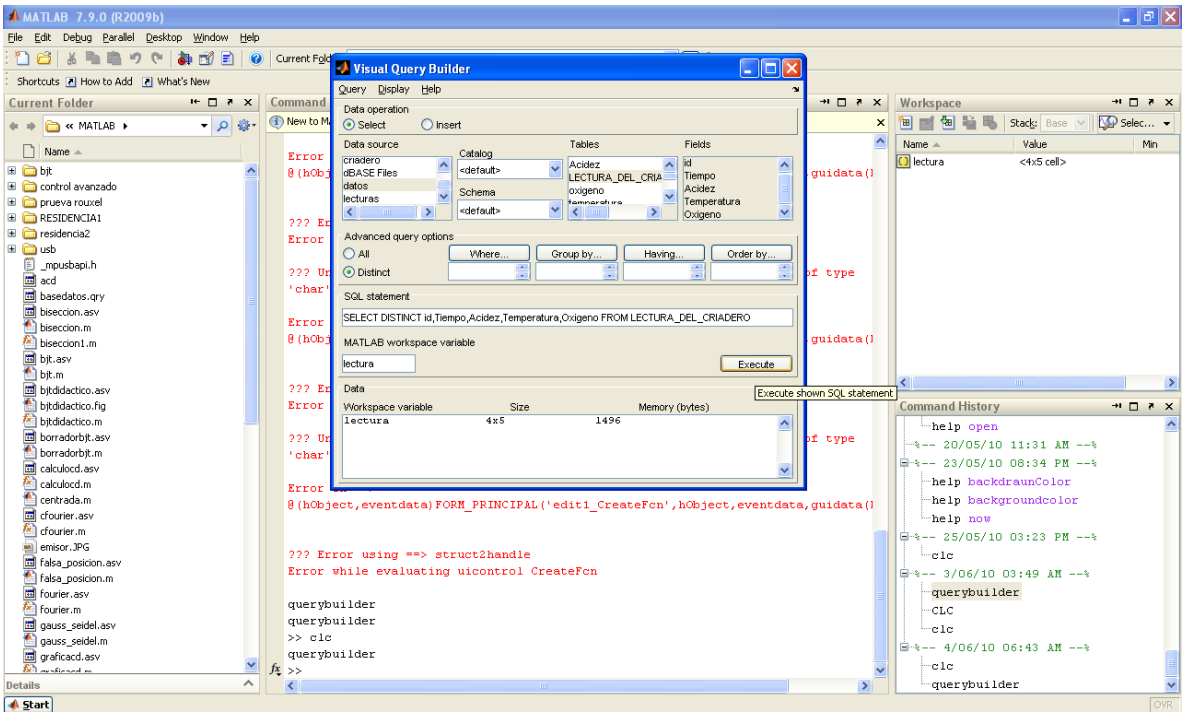


figura 3.23

De esta manera quedara listo nuestro MATLAB para poder conectarse con SQL Server mediante algunos comandos como los siguientes:

Con este comando nos conectamos a la base de datos, luego dentro del paréntesis escribimos primero el nombre de la base de datos enseguida el servidor y por último la contraseña.

```
conexion=database('lecturas','sa','rouxel')
```

Una vez que MATLAB se conecta con la base de datos insertamos los datos con el siguiente comando.

```
insert(conexion,'LECTURA_DEL_CRIADERO',{'Tiempo','Acidez','Temperatura'
,'
      Oxigeno'},{DATESTR(NOW) y1(n) y2(n) y(n)})
```

Después de ingresar los datos a SQL cerramos la comunicación, para evitar errores, con el siguiente comando.

```
 close (conexion)
```

3.6 CREACIÓN DEL SOFTWARE DE CONTROL IMPLEMENTADO EN MATLAB

Para la adquisición de datos a la computadora primero que nada tenemos que seleccionar un compilador adecuado que cumpla con las características que nosotros deseamos.

En este caso seleccionamos a MATLAB debido a que es un programa en el que contamos con las herramientas necesarias para poder adquirir datos, para poder guardarlos en una base de datos, graficar todas las variables entrantes y si la variable de oxigenación se sale de rango realizar una acción activar un motor en el exterior para oxigenar el agua.

Para que nuestro software de control fuera visual implementamos la guide en MATLAB así, podemos visualizar las graficas en tiempo real, visualizar si en el exterior de la computadora se está llevando a cabo una eventualidad como alimentación de las mojarras o de oxigenación al agua.

La programación basada en c (MATLAB) queda de la siguiente manera, la forma final de la guide se mostrara al final de la programación en la figura 3.24.

```
function varargout = FORM_PRINCIPAL(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FORM_PRINCIPAL_OpeningFcn, ...
                   'gui_OutputFcn',  @FORM_PRINCIPAL_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before FORM_PRINCIPAL is made visible.
function FORM_PRINCIPAL_OpeningFcn(hObject, eventdata, handles,
varargin)
```

```
global t data_in data_out vid_pid_norm out_pipe in_pipe conectado
my_in_pipe my_out_pipe handles1 x y z n conteo latch t1 t2
y=0
x=0
n=1
z=0
t2=0
t1=0
latch=0
conteo=1
handles1 = handles;
% Choose default command line output for menu
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes menu wait for user response (see UIRESUME)
% uiwait(handles.figure1);


conectado = 0;

data_in = eye(1,64,'uint8');        % Se declara el vector de datos de
entrada (el que se recibe del PIC)
data_out = eye(1,64,'uint8');     % Se declara el vector de datos de
salida (el que se envia al PIC)
                                    % TODOS LOS DATOS SE DECLARAN COMO
                                    % UINT8 de lo contrario no hay
                                    % comunicación.

vid_pid_norm = libpointer('int8Ptr',[uint8('vid_04d8&pid_000b') 0]);
out_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);
in_pipe = libpointer('int8Ptr',[uint8('\MCHP_EP1') 0]);

%SE DECLARA LA RUTINA QUE DEBE DE EJECUTARSE AL CERRAR LA APLICACIÓN
set(handles.figure1,'CloseRequestFcn',@closeGUI);


loadlibrary mpusbapi _mpusbapi.h alias libreria
% Los archivos _mpusbapi.c y mpusbapi.dll deben de estar en la misma
% carpeta de trabajo y se obtienen de la descarga del driver en la
% pagina de microchip ("Microchip MCHPFSUSB v2.4 Installer.zip"),
% al instalarse queda ubicado en X:\Microchip Solutions\USB
% Tools\MCHPUSB Custom Driver\Mpusbapi\Dll\Borland_C, en caso de
descargar
% una version de driver más reciente, reemplace éstos archivos por
los más
% nuevos.

%libisloaded libreria               % Confirma que la librería ha
sido
```

```
                                    % cargada
%libfunctions('libreria', '-full')  % Muestra en la línea de comandos
las
                                    % funciones de la librería
%libfunctionsview libreria          % Muestra en un cuadro lo mismo
que la
                                    % instrucción anterior

%calllib('libreria','MPUSBGetDLLVersion');

while (conectado == 0)
    [conectado] =
calllib('libreria','MPUSBGetDeviceCount',vid_pid_norm);
    if (conectado ==0)
        selection = questdlg('La tarjeta de evaluación no se
encuentra',...
                        'Tarjeta no encontrada',...
                        'Buscar de nuevo','Cancelar','Cancelar');
        switch selection,
            case 'Cancelar',
                stop (t);
                return
            case 'Buscar de nuevo'
                conectado = 0;
        end
    end
end
if conectado == 1   % Es importante seguir ésta secuencia para
comunicarse con el PIC:
                    % 1. Abrir tuneles, 2. Enviar/Recibir dato
                    % 3. Cerrar tuneles

    [my_out_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0),
vid_pid_norm, out_pipe, uint8(0), uint8 (0)); % Se abre el tunel de
envio
    [my_in_pipe] = calllib('libreria', 'MPUSBOpen',uint8 (0),
vid_pid_norm, in_pipe, uint8 (1), uint8 (0)); % Se abre el tunel de
recepción
end

axes(handles.axes1)
title('Grafica de Nivel de Oxigeno')
ylabel('Oxigeno en %')
xlabel('Tiempo')
axes(handles.axes2)
title('Grafica de Nivel de Temperatura')
ylabel('Temperatura en °C')
xlabel('Tiempo')
axes(handles.axes3)
title('Grafica de Nivel de Acides')
ylabel('Acidez en %')
xlabel('Tiempo')
set(handles.text10,'string',datestr(now))
```

```
% --- Outputs from this function are returned to the command line.
function varargout = FORM_PRINCIPAL_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% es el boton de inicio del programa
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global conectado my_out_pipe my_in_pipe data_in data_out handles1  x
y z n aux x1 y1 x2 y2 latch t1 t2
%y=0
%x=0
x()
y()
x1()
y1()
x2()
y2()
z=0

while(z==0)
int16 data;
if conectado == 1
  calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out, uint8(64),
uint8(64), uint8(200)); % Se envia el dato al PIC
    [aa,bb,data_in,dd] = calllib('libreria', 'MPUSBRead',my_in_pipe,
data_in, uint8(64), uint8(64), uint8(200)); % Se recibe el dato que
envia el PIC
    %data_out (1) = data_out (1)+ 1;     % Se incrementa el primer
dato del vector que se envia
    %data_in(1)                          % Se muestra el primer valor
del vector recibido
End
%despues de recibir el dato del pic estos se almacenan en la variable
data y se grafica dato por dato primero oxigenación luego acidez y
por ultimo la temperatura
    aux=100/1023;
data = (uint16(data_in(4)) * uint16(256)) + uint16(data_in(3));
set(handles1.edit1,'String',(num2str(data*aux)));
x(n)=n;
y(n)=data*aux;
```

```
axes(handles1.axes1)
 plot(x,y)
title('Grafica de Nivel de Oxigeno')
ylabel('Oxigeno en %')
xlabel('Tiempo')
 axis([x(n)-20 x(n)+5 -5 105])

pause(0.25)

 if conectado == 1
  calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out*2,
uint8(64), uint8(64), uint8(200)); % Se envia el dato al PIC
    [aa,bb,data_in,dd] = calllib('libreria', 'MPUSBRead',my_in_pipe,
data_in, uint8(64), uint8(64), uint8(200)); % Se recibe el dato que
envia el PIC
    %data_out (1) = data_out (1)+ 1;     % Se incrementa el primer
dato del vector que se envia
    %data_in(1)                          % Se muestra el primer valor
del vector recibido
end
    aux=100/1023;
data = (uint16(data_in(4)) * uint16(256)) + uint16(data_in(3));
set(handles1.edit2,'String',(num2str(data*aux)));
x1(n)=n;
y1(n)=data*aux;
axes(handles1.axes2)
 plot(x1,y1)
title('Grafica de Nivel de Acidez')
ylabel('Acidez en %')
xlabel('Tiempo')
 axis([x(n)-20 x(n)+5 -5 105])

 pause(0.25)
  if conectado == 1
  calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out*3,
uint8(64), uint8(64), uint8(200)); % Se envia el dato al PIC
    [aa,bb,data_in,dd] = calllib('libreria', 'MPUSBRead',my_in_pipe,
data_in, uint8(64), uint8(64), uint8(200)); % Se recibe el dato que
envia el PIC
    %data_out (1) = data_out (1)+ 1;     % Se incrementa el primer
dato del vector que se envia
    %data_in(1)                          % Se muestra el primer valor
del vector recibido
end
    aux=528/1023;
data = (uint16(data_in(4)) * uint16(256)) + uint16(data_in(3));
set(handles1.edit3,'String',(num2str(data*aux)));
x2(n)=n;
y2(n)=data*aux;
axes(handles1.axes3)
 plot(x2,y2)
title('Grafica de Nivel de Temperatura')
ylabel('Temperatura en °C')
xlabel('Tiempo')
 axis([x(n)-20 x(n)+5 -5 105])
```

```
 %condicion para el aranque del motor de oxigenacion

if (y(n)<=3 && latch==0)

      set(handles.text8,'BackgroundColor','green')
      latch=1
end
if (y(n)>=8 && latch==1)

      set(handles.text8,'BackgroundColor','red')
      latch=0
end

%condiciones para el motor de alimento

%n = n+1;
if (t1>=60 && t1<=63)


    set(handles.text9,'BackgroundColor','green')
else
    set(handles.text9,'BackgroundColor','red')
end
   if t1>63
       t1=0
   end

%en esta parte  nos conectamos a la base de datos con SQL Server
cada minuto y una vez guardado el dato cerramos la conexión y se
abrirá después de un minuto
 if(t2==60)
conexion=database('lecturas','sa','rouxel')

insert(conexion,'LECTURA_DEL_CRIADERO',{'Tiempo','Acidez','Temperatur
a','Oxigeno'},{DATESTR(NOW) y1(n) y2(n) y(n)})
 close (conexion)

 end
 if t2>61
     t2=0
 end

 n =n+1
 t1=t1+1
 t2=t2+1
 %pause(1)
pause(0.25);

end


function edit1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
conectarbase()


% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called



function tiempo(hObject, eventdata) %(source,eventdata)

global conectado my_out_pipe my_in_pipe data_in data_out handles1
conteo x y z n
x()
y()

int16 data;

if conectado == 1
```

```
    calllib('libreria', 'MPUSBWrite',my_out_pipe, data_out,
uint8(64), uint8(64), uint8(200)); % Se envia el dato al PIC
    [aa,bb,data_in,dd] = calllib('libreria', 'MPUSBRead',my_in_pipe,
data_in, uint8(64), uint8(64), uint8(200)); % Se recibe el dato que
envia el PIC
    %data_out (1) = data_out (1)+ 1;     % Se incrementa el primer
dato del vector que se envia
    %data_in(1)                          % Se muestra el primer valor
del vector recibido
end


data = (uint16(data_in(4)) * uint16(256)) + uint16(data_in(3));
set(handles1.edit1,'String',(num2str(data)));
x(n)=n;
y(n)=data;
axes(handles1.axes1)
 plot(x,y)
title('Grafica de Nivel de Oxigeno')
ylabel('Oxigeno en %')
xlabel('Tiempo')
 axis([x(n)-20 x(n)+5 -5 y(n)+5])
n = n+1;

function closeGUI(source,eventdata)
%src is the handle of the object generating the callback (the source
of the event)
%evnt is the The event data structure (can be empty for some
callbacks)
global t my_in_pipe my_out_pipe conectado z
z=1;
selection = questdlg('Desea cerrar la aplicación?',...
                     'Confirmación',...
                     'Si','No','Si');
switch selection,
   case 'Si',
 %    stop (t);
    if conectado == 1
        calllib('libreria', 'MPUSBClose', my_in_pipe);  % Se cierra
el tunel de recepción
        calllib('libreria', 'MPUSBClose', my_out_pipe); % Se cierra
el tunel de envio
     end
     unloadlibrary libreria      % Importante descargar la librería
de memoria, de lo contrario genera errores
     delete(gcf)
     close all
     clear all
  case 'No'
    return
end
```

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global z
z=1;



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text8,'BackgroundColor','green')
```

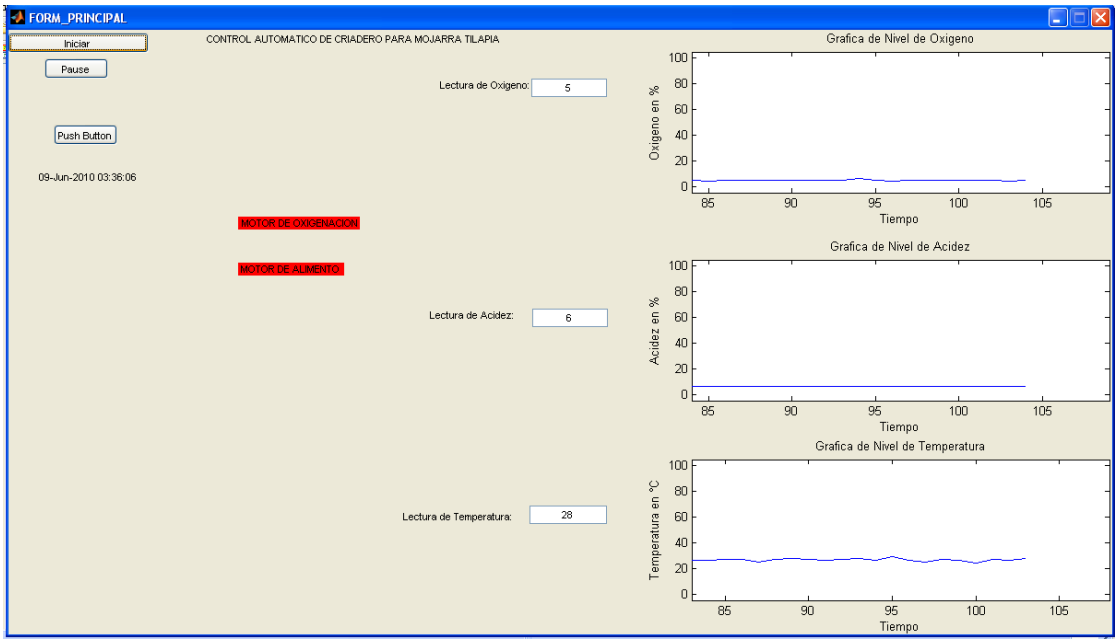


Figura 3.24

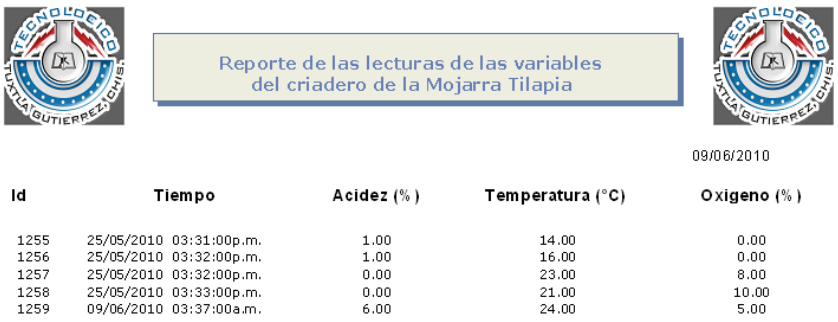
En la guide cuando los indicadores de los motores encargados de oxigenar y alimentar las mojarras están de color rojo, esto indica que los motores están apagados, al momento que alguno de los dos indicadores se pone de color verde o ya sea que los dos al mismo tiempo se pongan de color verde, esto nos estará indicando que los motores están encendidos.

El motor de la oxigenación trabaja condicionado con el nivel de oxigeno adquirido por medio de sensores si baja de 3 que es el nivel mínimo optimo requerido el motor se activara hasta llegar a 9 que es el nivel máximo optimo requerido para el nivel de oxigeno en el agua.

3.7 VISUALIZACION DE LA BASE DE DATOS EN UNA HOJA DE REPORTE CON VISUAL REPORT

Una vez que ya tenemos la información en nuestra base de datos de las variables de oxigenación, temperatura y acidez del agua, necesitamos implementar un programa para poder visualizar en forma de reporte los datos y poder analizarlos con mucha mayor facilidad. Es por esto que optamos por visualizar nuestro reporte atreves de un programa VISUAL REPORT que es un software especializado en la generación de reportes de bases de datos.

El reporte que se obtiene de nuestro criadero de mojarras quedara de la siguiente manera en la figura 3.25



Reporte de las lecturas de las variables del criadero de la Mojarra Tilapia

09/06/2010

Id	Tiempo	Acidez (%)	Temperatura (°C)	Oxigeno (%)
1255	25/05/2010 03:31:00p.m.	1.00	14.00	0.00
1256	25/05/2010 03:32:00p.m.	1.00	16.00	0.00
1257	25/05/2010 03:32:00p.m.	0.00	23.00	8.00
1258	25/05/2010 03:33:00p.m.	0.00	21.00	10.00
1259	09/06/2010 03:37:00a.m.	6.00	24.00	5.00

:

Figura 3.25

Al obtener el reporte de esta manera, de forma ordenada por fechas será mucho más fácil analizar cómo ha ido variando cada variable al transcurso de la crianza de las mojarras y poder hacer un análisis más a fondo de las mojarras.

CAPITULO 4: CONCLUSION

4.1 CONCLUSIÓN

Se realizó este proyecto gracias a los conocimientos adquiridos a través de los semestres cursados, este proyecto tendrá un gran impacto en el cultivo de mojarras ya que con este proyecto no solo se podrá mejorar la salud de las mojarras durante su desarrollo si no que también nos permitirá poder estudiar aun mas las condiciones de las mojarras, esto para poder impulsar a un mas el cultivo de esta especie en la región, el proyecto es viable para la obtención de alimentos de mayor calidad.

BIBLIOGRAFIA

- MANUAL DE PRODUCCIÓN DE TILAPIA.

AUTOR: BIÓL. FERNANDO CANTOR ATLATENCO

- DOMINE MICROSOFT SQL SERVER 2008

AUTOR: Pérez López, Cesar

- MANUAL EN ESPAÑOL DE PIC 18F4550

http://www.dtforum.net/index.php?topic=21401.0

- MATLAB Y SUS APLICACIONES EN LAS CIENCIAS Y LA INGENIERÍA

AUTOR: Pérez, César

EDITORIAL: Pearson – Prentice Hall, 2002