**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FIEC**

**Laboratorio de Microcontroladores**

**Proyecto Final de Microcontroladores**

**Auto Controlado a Control Remoto.**

**Estudiantes**

**Andrés Sebastián Cáceres Valdiviezo**

**Víctor Ricardo Bayas Agurto**

**Paralelo # 8**

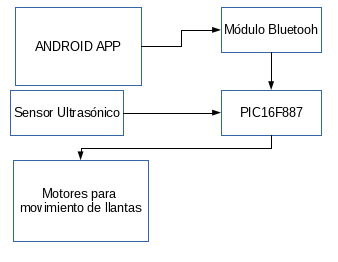
**Fecha de Presentación:**

**24 de febrero de 2016**

**1.- ENUNCIADO DEL PROBLEMA**

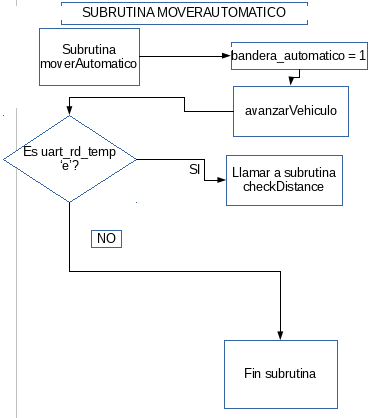
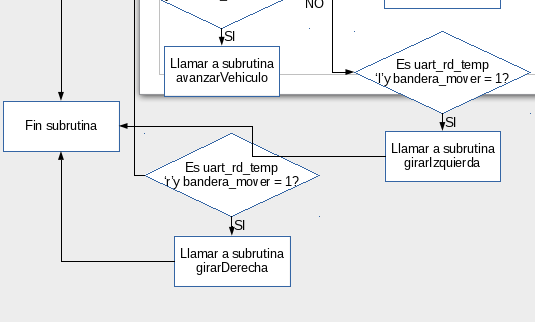
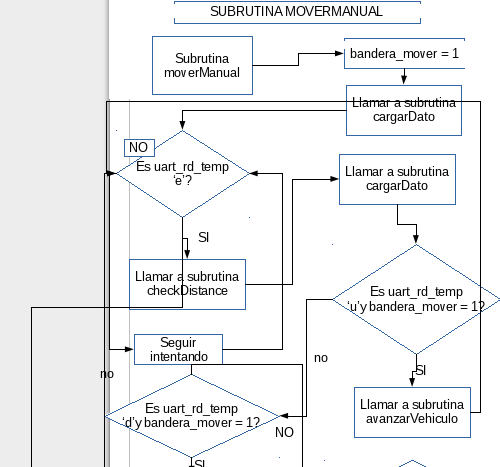
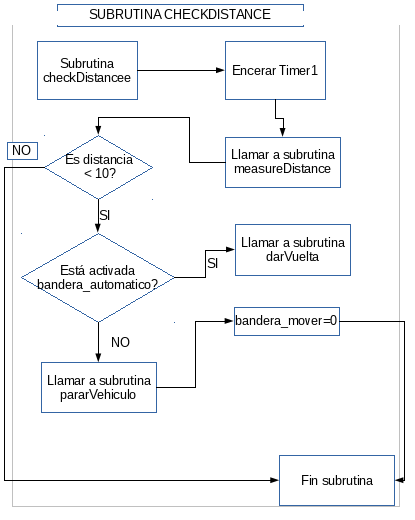
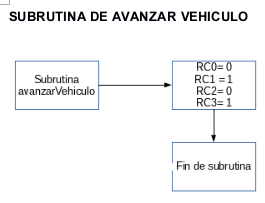
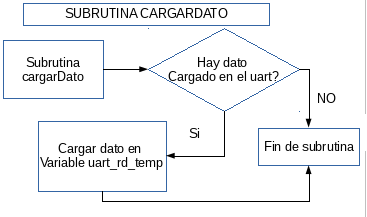
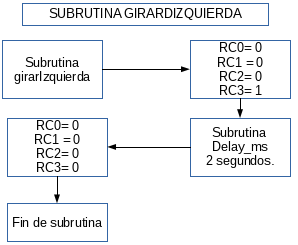
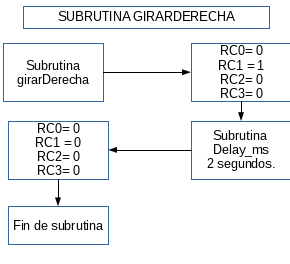
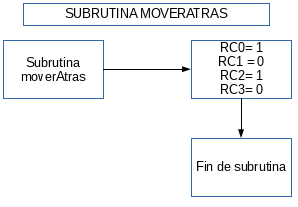
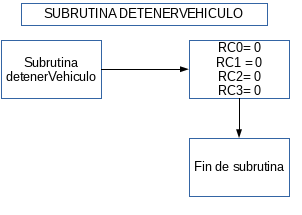
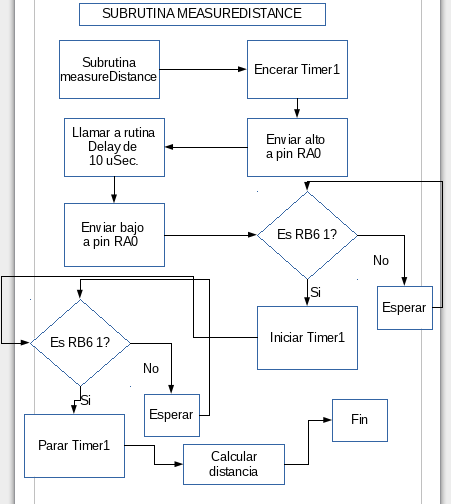
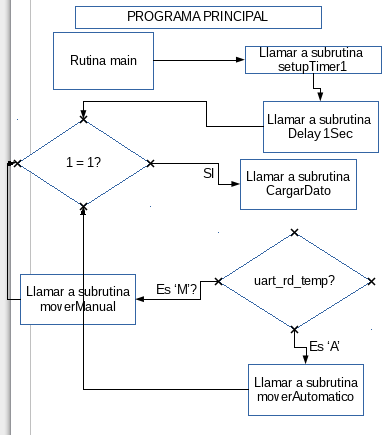
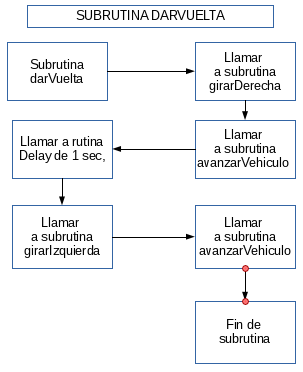
El proyecto consiste en dar movimiento a un auto controlado por medio de una aplicación en Android del cuál se envían los comandos a realizar por el auto. Se tienen 2 modos de operación: automático y manual. El modo automático es aplicado para la evasión de obstáculos en frente del auto en movimiento, para lo cual se le dará la vuelta.

1. **- DIAGRAMA DE BLOQUES**



1. **- DIAGRAMAS DE FLUJO**

**PROGRAMA**



**DESCRIPCIÓN DEL ALGORITMO O ESTRATEGIA**

**SETUPTIMER1**

1. Editar registro T1CON para usar el oscilador externo y configurar TIMER1 como temporizador. **TICON= 0X10**
2. Encerar TMR1H y TMR1L
3. Fin de subrutina

**AVANZAR VEHICULO**

1. Mandar a pines de puerto C 0,1,2,3 los valores de 0,1,0,1 respectivamente.
2. Fin de subrutina.

**-PARAR\_VEHICULO**

1. Mandar a pines de puerto C 0,1,2,3 los valores de 0,0,0,0 respectivamente.
2. Fin de subrutina.

**-RETROCEDER\_VEHICULO**

1. Mandar a pines de puerto C 0,1,2,3 los valores de 1,0,1,0 respectivamente.
2. Fin de subrutina.

**-GIRARDERECHA**

1. Mandar a pines de puerto C 0,1,2,3 los valores de 0,1,0,0 respectivamente.
2. Usar delay de 2 segundos.
3. Llamar a subrutina **PARAR\_VEHICULO**
4. Fin de subrutina.

**-GIRARIZQUIERDA**

1. Mandar a pines de puerto C 0,1,2,3 los valores de 0,0,0,1 respectivamente.
2. Usar delay de 2 segundos.
3. Llamar a subrutina **PARAR\_VEHICULO**
4. Fin de subrutina.

**-CARGARDATO**

1. Verificar que dato haya sido llenado en registros de UART
2. Si hay dato entonces asignarlo a la variable global **uart\_rd\_temp.**
3. Fin de subrutina.

**-MEASUREDISTANCE**

1. Encerar TMR1H Y TMR1L
2. Enviar alto al **Trigger** (pin 0 del puerto A).
3. Esperar a que el sensor ultrasónico envíe 1 al pin **RB6**
4. Al ocurrir esto enviar un bajo al **Trigger**
5. Medir distancia usando fórmula dada, suponiendo que se utilice un oscilador de 8MHz.
6. Fin de subrutina.

**-CHECKDISTANCE**

1. Llamar a subrutina **measureDistance.**
2. Si la distancia del ultrasónico al objeto es < 10 y se encuentra en modo automático llamar a subrutina **darVuelta.**
3. Si distancia < 10 y está en modo manual entonces llamar a subrutina **pararVehiculo.**
4. Caso contrario llamar a subrutina **avanzarVehiculo** y activar banderas de movimiento.
5. Fin de subrutina.

**-MOVERMANUAL**

1. Encerar la bandera de mover\_automatico
2. Asignar 1 a variable mover\_manual.
3. Encerar variable **uart\_rd\_temp**
4. Llamar a subrutina **cargarDato.**
5. Mientras no se envíe caracter de salida de modo ‘e’ realizar switch entre los demás caracteres enviados.
6. Si el caracter enviado es ‘u’: llamar a subrutina **avanzarVehiculo**
7. Si el caracter enviado es ‘d’: llamar a subrutina **retrocederVehiculo**
8. Si el caracter enviado es ‘l’: llamar a subrutina **girarIzquierda**
9. Si el caracter enviado es ‘r’: llamar a subrutina **girarDerecha**
10. Si se sale mandando ‘e’ encerar bandera\_manual.
11. Fin de subrutina.

**-MOVERAUTOMATICO**

1. Encerar la bandera de mover\_manual
2. Asignar 1 a variable mover\_automatico.
3. Encerar variable **uart\_rd\_temp**
4. Llamar a subrutina **cargarDato.**
5. Llamar a subrutina **avanzarVehiculo**
6. Mientras no se envíe caracter de salida de modo ‘e’ realizar switch entre los demás caracteres enviados.
7. Si se sale mandando ‘e’ encerar bandera\_manual.
8. **Fin de subrutina.**

**-PRINCIPAL**

1. Setear el puerto A como salidas TRISA = 0
2. Setear el puerto B como salidas: TRISB = 0b01000000; menos el pin 6.
3. Encerar ANSELH y ANSEL.
4. Desabilitar comparadores.
5. Poner en TRISC los valores: TRISC = 0b10000000
6. Llamar a subrutina **setupTimer1**
7. Iniciar loop infinito.
8. Llamar a subrutina **cargarDato.**
9. Si valores de uart\_rd\_temp = ‘A’, entonces llamar subrutina **moverEnAutomatico**
10. Si valores de uart\_rd\_temp = ‘M’, entonces llamar subrutina **moverManual**
11. Fin de programa.
12. **- CÓDIGO FUENTE EN LENGUAJE C**

#define Trig PORTA.RA0

#define Echo PORTB.RB6

// LCD module connections

sbit LCD\_RS at RB4\_bit;

sbit LCD\_EN at RB5\_bit;

sbit LCD\_D4 at RB0\_bit;

sbit LCD\_D5 at RB1\_bit;

sbit LCD\_D6 at RB2\_bit;

sbit LCD\_D7 at RB3\_bit;

sbit LCD\_RS\_Direction at TRISB4\_bit;

sbit LCD\_EN\_Direction at TRISB5\_bit;

sbit LCD\_D4\_Direction at TRISB0\_bit;

sbit LCD\_D5\_Direction at TRISB1\_bit;

sbit LCD\_D6\_Direction at TRISB2\_bit;

sbit LCD\_D7\_Direction at TRISB3\_bit;

// End LCD module connections

/\* -----------------------LCD TEXTO--------------\*/

char txtDefault[] = "Esperando Seleccion";

char txtManual[] = "Modo Manual";

char txtArriba[] = "Arriba";

char txtIzquierda[] = "Izquierda";

char txtDerecha[] = "Derecha";

char txtAbajo[] = "Abajo";

char txtAutomatic[] = "Automatico";

/\* ----------------------ADC--------------------- \*/

unsigned int distance\_cm = 0, distance\_inc = 0, TMR = 0;

int a = 0;

char txt[7];

/\*-----------------------UART-------------------- \*/

char uart\_rd; /\* char leido enviado desde aplicacion Android \*/

char uart\_rd\_temp;

int bandera\_mover = 1; /\* Por defecto, se puede mover, si sensor detecta tropiezo,

no mueve \*/

int bandera\_automatico = 1;

/\* HABILITACION INTERRUPCION \*/

//int contadorInterrupt = 0;

void setupTimer1(){

/\* Configurando timer 1 a preescalador 1:4 GIE, PORTB6 interrupcion \*/

// TImer1 Contador 1x4x preescalador

T1CON = 0b00001010; // 0b00001010 --> bit 1 internal clock en 0, bit

TMR1H = 0; //Sets the Initial Value of Timer

TMR1L = 0; //Sets the Initial Value of Timer

// Interrupcion en cambio de estado de PORTB6

/\* IOCB.IOCB6 = 1; \*/

}

void waitSignal(){

TMR1H = 0; // Clear Timer1

TMR1L = 0;

if(Echo == 0){

Trig = 0;

Delay\_us(2);

Trig = 1;

Delay\_us(10); // Send LOW-to-HIGH Pulse of 10us to Ultrasonic

Trig = 0;

}

}

/\* Avanzar vehiculo \*/

void avanzarVehiculo(){

RC0\_bit = 0;

RC1\_bit = 1;

RC2\_bit = 0;

RC3\_bit = 1;

}

/\* Logica para detener el vehiculo. \*/

void pararVehiculo(){

RC0\_bit = 0;

RC1\_bit = 0;

RC2\_bit = 0;

RC3\_bit = 0;

}

/\* 2 Motores: motor derecha es ccp2, motor izquierda es ccp1

giroderecha: detener motor derecha, mover izquierdo\*/

void girarDerecha(){

RC0\_bit = 0;

RC1\_bit = 1;

RC2\_bit = 0;

RC3\_bit = 0;

Delay\_ms(2000); /\* 2 segundos de giro \*/

/\* Se para el carro con la posicion ya girada. \*/

RC0\_bit = 0;

RC1\_bit = 0;

RC2\_bit = 0;

RC3\_bit = 0;

}

/\* giroizquierda: mover motor derecha, detener izquierdo\*/

void girarIzquierda(){

RC0\_bit = 0;

RC1\_bit = 0;

RC2\_bit = 0;

RC3\_bit = 1;

Delay\_ms(1400); /\* 2 segundos de giro \*/

/\* Se para el carro con la posicion ya girada. \*/

RC0\_bit = 0;

RC1\_bit = 0;

RC2\_bit = 0;

RC3\_bit = 0;

}

/\* giroizquierda: mover motor derecha, detener izquierdo\*/

void moverAtras(){

RC0\_bit = 1;

RC1\_bit = 0;

RC2\_bit = 1;

RC3\_bit = 0;

}

/\* Funcion usada en modo automatico para esquivar obstaculo \*/

void darVuelta(){

girarDerecha();

avanzarVehiculo();

Delay\_ms(1000);

girarIzquierda();

avanzarVehiculo();

}

void cargarDato(){

if (UART1\_Data\_Ready()) // If data is received,

uart\_rd\_temp = UART1\_Read(); // leer el dato recibido del celular

}

void measureDistance(){

TMR1H = 0; //Sets the Initial Value of Timer

TMR1L = 0; //Sets the Initial Value of Timer

Trig = 1; //TRIGGER HIGH

Delay\_us(10); //10uS Delay

Trig = 0; //TRIGGER LOW

while(!Echo); //Waiting for Echo

T1CON.F0 = 1; //Timer Starts

while(Echo); //Waiting for Echo goes LOW

T1CON.F0 = 0; //Timer Stops

a = (TMR1L | (TMR1H<<8)); //Reads Timer Value

a = a/58.82; //Converts Time to Distance

a = a + 1; //Distance Calibration

}

void checkDistance(){

/\* char txt[7]; \*/

TMR1H = 0; //Sets the Initial Value of Timer

TMR1L = 0; //Sets the Initial Value of Timer

Trig = 1; //TRIGGER HIGH

Delay\_us(10); //10uS Delay

Trig = 0; //TRIGGER LOW

while(!Echo); //Waiting for Echo

T1CON.F0 = 1; //Timer Starts

while(Echo); //Waiting for Echo goes LOW

T1CON.F0 = 0; //Timer Stops

a = (TMR1L | (TMR1H<<8)); //Reads Timer Value

a = a/58.82; //Converts Time to Distance

a = a + 1; //Distance Calibration

if(a<10) //Check whether the result is valid or not

{

if(bandera\_mover && bandera\_automatico) {/\* Si hay obstaculo y esta en Automatico\*/

Lcd\_Out(2,3,"Dando vuelta....");

darVuelta();

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(2,3,"Termino la vuelta....");

}else if(bandera\_mover && !bandera\_automatico){

pararVehiculo();

bandera\_mover = 0;

}

}else {

bandera\_mover = 1;

avanzarVehiculo();

}

}

void moverManual(){

bandera\_mover = 1;

Lcd\_Out(1,2,txtManual);

uart\_rd\_temp = ' ';

cargarDato();

bandera\_automatico = 0; /\* Se limpia la bandera de automatico por si acaso. \*/

while (uart\_rd\_temp != 'e') { /\* Si se le envía e se sale de modo manual \*/

checkDistance();

cargarDato(); /\* se verifica si se tiene que cambiar o no la variable uart\_rd\_temp \*/

if (uart\_rd\_temp == 'u' && bandera\_mover) { /\* Up: En la interrupción para el auto \*/

Lcd\_Out(2,3,txtArriba);

avanzarVehiculo();

}

else if (uart\_rd\_temp == 'd') { /\* Down: \*/

Lcd\_Out(2,3,txtAbajo);

moverAtras();

}

else if (uart\_rd\_temp == 'l') { /\* left: \*/

Lcd\_Out(2,3,txtAbajo);

girarIzquierda();

avanzarVehiculo();

}

else if (uart\_rd\_temp == 'r') { /\* right: \*/

Lcd\_Out(2,3,txtDerecha);

girarDerecha();

avanzarVehiculo();

}

}

pararVehiculo(); /\* como se manda e entonces, se detiene el movimiento del auto y se sale a esperar seleccion de otro modo. \*/

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(2,3,"Se salio de modo manual...");

Delay\_ms(500);

uart\_rd\_temp = ' ';

}

void moverEnAutomatico(){

uart\_rd\_temp = ' ';

bandera\_automatico = 1; /\* Con esta bandera se valida en la interrupcion que hacer cuando ocurre un choque. \*/

Lcd\_Cmd(\_LCD\_CLEAR);

bandera\_mover = 1;

Lcd\_Out(2,3,txtAutomatic);

/\* Inicio de movimiento en automatico hacia adelante. \*/

avanzarVehiculo();

cargarDato();

while (uart\_rd\_temp != 'e') { /\* Si se le envía e se sale de modo automatico. \*/

checkDistance();

/\* waitSignal(); \*/

cargarDato();

/\* Al salir de modo automatico paramos los motores.\*/

if(uart\_rd\_temp == 'e')

{

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(2,3,"Se paro el vehiculo");

pararVehiculo();

}

}

pararVehiculo();

bandera\_automatico = 0; /\* al enviar e se sale de modo automatico.\*/

uart\_rd\_temp = ' ';

}

void printDistance(){

IntToStr(a,txt);

Ltrim(txt);

Lcd\_Out(1,12,txt);

Lcd\_Out(1,15,"cm");

}

void main() {

TRISA = 0b00000000; // set PORTA -> salidas

TRISB = 0b01000000; // set PORTB -> salida menos el pin 6.

/\* TRISB.f6 = 1; \*/

ANSEL = 0; // Configure AN2 pin as analog

ANSELH = 0;

C1ON\_bit = 0; // Disable comparators

C2ON\_bit = 0;

TRISC = 0b10000000; // PUERTOC COMO SALIDA exepto pin 7

/\* TRISC.f7 = 1; \*/

PORTC = 0; // LIMPIAR PUERTOC

setupTimer1();

Delay\_us(10); /\* wait for acquisition time\*/

Lcd\_Init(); // Initialize LCD

/\* Programa principal \*/

Lcd\_Out(2,0,txtDefault); // Write text in second row

measureDistance();

printDistance();

UART1\_Init(19200); // initialize UART1 module

Delay\_ms(100); // Wait for UART module to stabilize

/\* moverEnAutomatico(); \*/

while (1) {

/\* waitSignal(); \*/

if (UART1\_Data\_Ready()) { // If data is received,

uart\_rd = UART1\_Read(); // leer el dato recibido del celular

/\* waitSignal(); \*/

switch (uart\_rd) {

case 'A': { /\* Modo automático \*/

Lcd\_Out(2,3,txtAutomatic);

moverEnAutomatico();

uart\_rd = ' '; /\* esperamos nuevo envío de modo. \*/

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(2,3,txtDefault); // Write text in second row

measureDistance();

break;

}

case 'M': {

moverManual();

uart\_rd = ' '; /\* esperamos nuevo envío de modo. \*/

Lcd\_Cmd(\_LCD\_CLEAR);

Lcd\_Out(2,3,txtDefault); // Write text in second row

measureDistance();

break;

}

default:{

uart\_rd = ' ';

break;

}

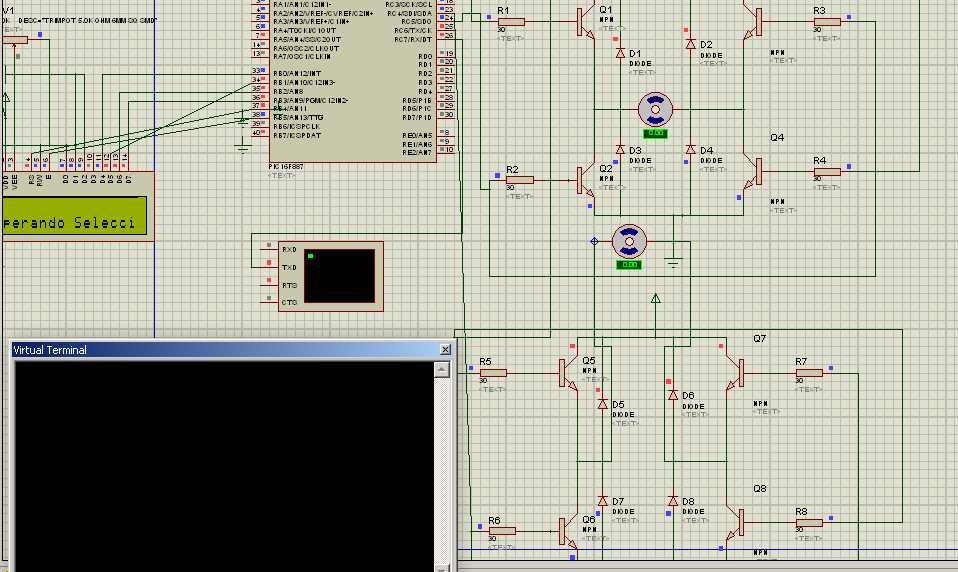
}

}

}

}

**6.- GRÁFICOS DE CIRCUITOS IMPRESOS**



Gŕafico mostrando la funcionalidad de los puentes H y la pantalla LCD esperando en el menú principal.

**7.- CONCLUSIONES**

Se aprendió como aplicar los conocimientos obtenidos en la materia para poder implementar un proyecto real y funcional abarcando los componentes más utilizados en el mundo de los microcontroladores. (UART, LCD)

Se aprendió a poder interconectar los pines del microcontrolador de manera adecuada en el protoboard, gracias a ello se pudieron realizar pruebas que no eran factibles de realizar en el proteus como la medida de la distancia usando el HC - SR04 o probar bien la comunicación serial con el módulo de bluetooh.

Se logró ingeniar ideas para lograr el movimiento pedido al auto, acudiendo a los conocimientos de lenguaje C y de los microcontroladores.

**8.- RECOMENDACIONES**

Utilizar placa después de tener todos los resultados esperados en el protoboard, si se realiza directo a la placa se aumenta la dificultad de reparar errores o cambios de lógica en los pines del microcontrolador.

Utilizar guías de equipos usados tales como el RS232, el HC -SR04 los cuáles tienen amplia documentación y ejemplos de como poder usarlos y funcionar en un programa.