UART

Ingeniería Mecatrónica

Programación de sistemas embebidos

Ivan Alejandro Pasillas Gonzalez

Marco teórico:

UART, son las siglas en inglés de Universal Asynchronous Receiver-Transmitter, en español: Transmisor-Receptor Asíncrono Universal, es el dispositivo que controla los puertos y dispositivos serie. Se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo.

Un UART dual, o DUART, combina dos UART en un solo chip. Existe un dispositivo electrónico encargado de generar la UART en cada puerto serie. La mayoría de las computadoras modernas utilizan el chip UART 16550, que soporta velocidades de transmisión de hasta 921,6 Kbps (Kilobits por segundo). Las funciones principales de chip UART son: manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.

El controlador del UART es el componente clave del subsistema de comunicaciones series de una computadora. El UART toma bytes de datos y transmite los bits individuales de forma secuencial. En el destino, un segundo UART reensambla los bits en bytes completos. La transmisión serie de la información digital (bits) a través de un cable único u otros medios es mucho más efectiva en cuanto a costo que la transmisión en paralelo a través de múltiples cables. Se utiliza un UART para convertir la información transmitida entre su forma secuencial y paralela en cada terminal de enlace. Cada UART contiene un registro de desplazamiento que es el método fundamental de conversión entre las forma secuencial y paralela.

El UART normalmente no genera directamente o recibe las señales externas entre los diferentes módulos del equipo. Usualmente se usan dispositivos de interfaz separados para convertir las señales de nivel lógico del UART hacia y desde los niveles de señalización externos.

Las señales externas pueden ser de variada índole. Ejemplos de estándares para señalización por voltaje son RS-232, RS-422 y RS-485 de la EIA. Históricamente se usó la presencia o ausencia de corriente en circuitos telegráficos.

Algunos esquemas de señalización no usan cables eléctricos; ejemplo de esto son la fibra óptica, infrarrojo (inalámbrico) y Bluetooth (inalámbrico). Algunos esquemas de señalización emplean una modulación de señal portadora (con o sin cables); por ejemplo, la modulación de señales de audio con módems de línea telefónica, la modulación en radio frecuencia (RF) en radios de datos y la DC-LIN para la comunicación de línea eléctrica.

Introducción:

En esta práctica se hará un control UART usando la psoc, una laptop, y un motor a pasos con un driver, en esta práctica usaremos el programa putty el cual nos ayudara a cumplir la función la cual es dar la orden al motor de que gire tantos grados como el usuario desee, para eso necesitamos saber que puerto COM es en el caso de esta practica es el 4, a continuación se muestran las imágenes

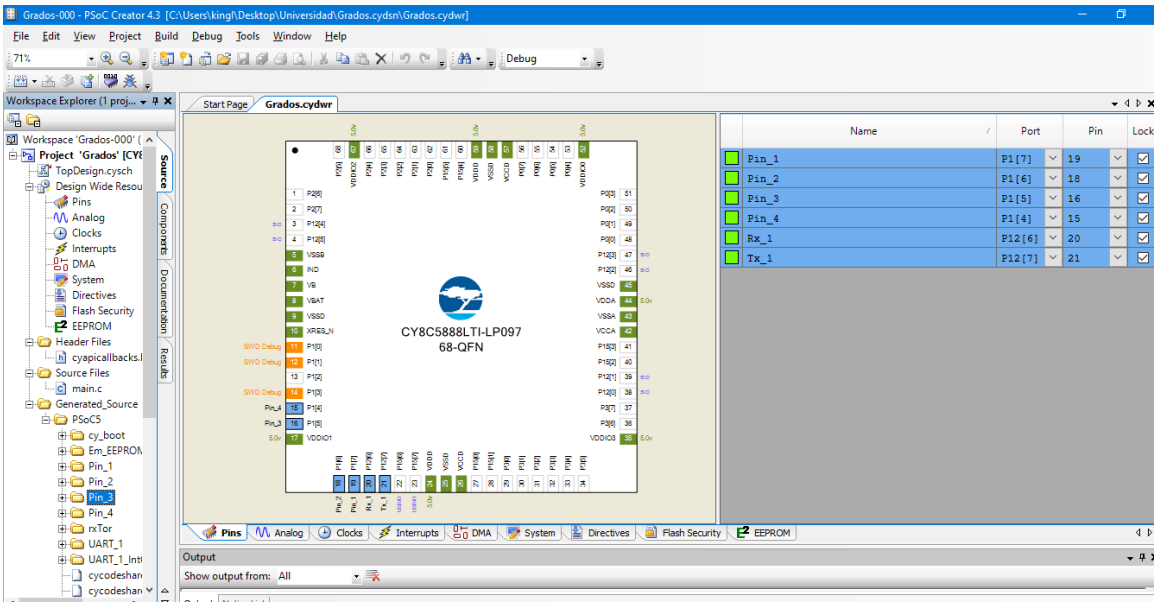

Figura 1: distribución de pines

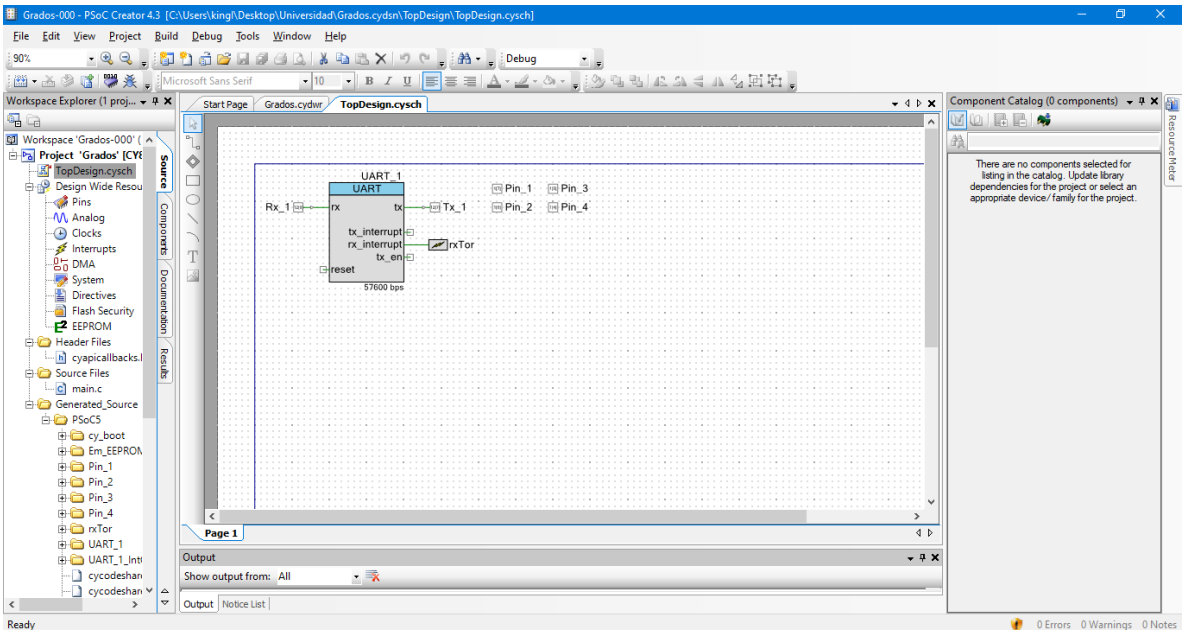

figura 2: Top desing

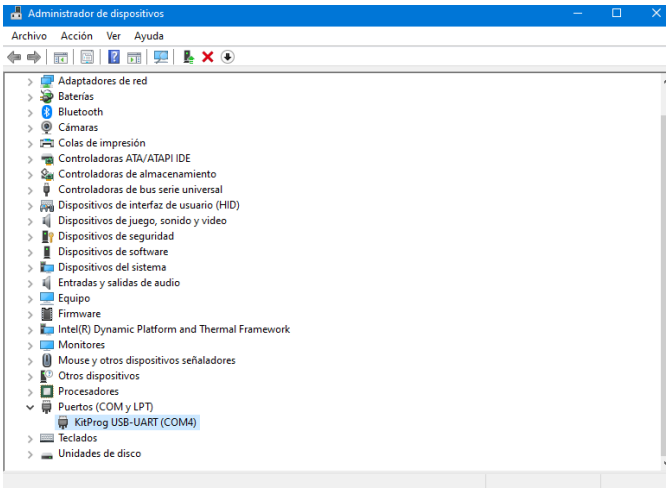

Figura 3: administrador de dispositivos

En el administrador de dispositivos buscaremos los puertos COM en la parte de abajo y como se muestra en la figura nuestro puerto es el 4, ahora abrimos putty

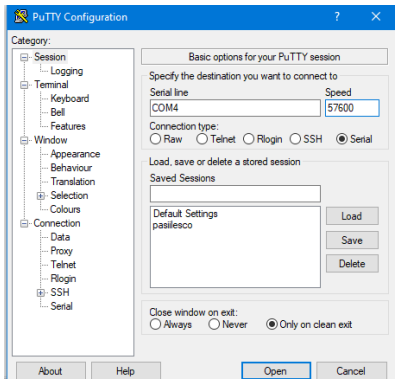

figura 4: interfaz de putty

una vez abierto putty se configuran las acciones necesarias, en el puerto COM ponemos que es el 4, en velocidad 57600, si no es esa velocidad al ejecutar no funcionara, y en las casillas de selección antes que los primero 2 pasos se ponen en serial, una vez teniendo eso le damos en open y se abrirá la interfaz

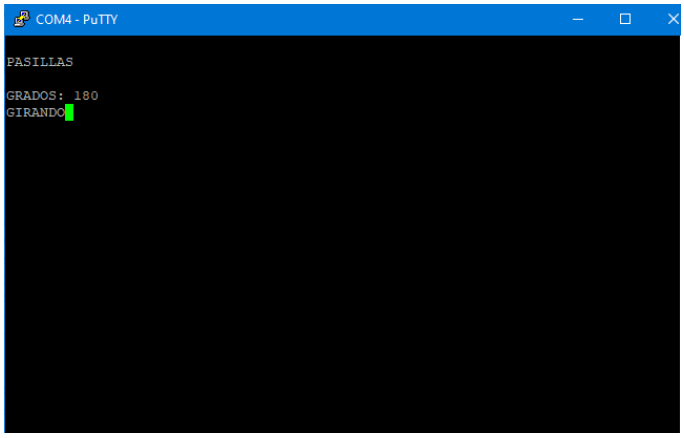

Figura 5: interfaz del motor

En esta ventana ya se abre la terminal donde se ejecutarán los grados a mover, como se puede observar en la sección grados se puso 180 al darle enter, el motor gira loa grados que se le ordenan y en este caso no hay limite de grados en cierto punto puede girar hasta 620 grados

Código de programación:

```
#include "project.h"

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctype.h>


#define STR_LEN 32

#define RX_BUF_LEN 128


#define SPACE ' '

#define TAB '\t'

#define CR  '\r'

#define LF  '\n'


const int speed = 2;

const double AGrados = 1.435;


char          str[STR_LEN+1] ; /* print buffer */

int           str_index = 0 ;

int           str_ready = 0 ;


void horario(int grados) {
```

```
for (int x = 0; x<=grados * AGrados; x++){
    Pin_1_Write(0);
    Pin_2_Write(0);
    Pin_3_Write(0);
    Pin_4_Write(1);
    CyDelay(speed);

    Pin_1_Write(0);
    Pin_2_Write(0);
    Pin_3_Write(1);
    Pin_4_Write(0);
    CyDelay(speed);

    Pin_1_Write(0);
    Pin_2_Write(1);
    Pin_3_Write(0);
    Pin_4_Write(0);
    CyDelay(speed);

    Pin_1_Write(1);
    Pin_2_Write(0);
    Pin_3_Write(0);
    Pin_4_Write(0);
```

```
        CyDelay(speed);
    }
}

void antihorario(int grados){
     for (int x = grados * AGrados; x>=0; x--){
        Pin_1_Write(1);
        Pin_2_Write(0);
        Pin_3_Write(0);
        Pin_4_Write(0);
        CyDelay(speed);

        Pin_1_Write(0);
        Pin_2_Write(1);
        Pin_3_Write(0);
        Pin_4_Write(0);
        CyDelay(speed);

        Pin_1_Write(0);
        Pin_2_Write(0);
        Pin_3_Write(1);
        Pin_4_Write(0);
        CyDelay(speed);
```

```
        Pin_1_Write(0);
        Pin_2_Write(0);
        Pin_3_Write(0);
        Pin_4_Write(1);
        CyDelay(speed);
    }
 }

inline int is_delimiter(uint8_t c){
    int result = 0 ;
    switch(c) {
    case CR:
    case LF:
    case TAB:
    case SPACE:
        result = c ;
        break ;
    }
    return( result ) ;
}

void cls(){
```

```
 UART_1_PutString("\033[2J\033[H");
}

void print(char *str){
    UART_1_PutString(str) ;
}

int check_str(void){
    int result = 0 ;
    uint8_t c ;
    while((result == 0) && (UART_1_GetRxBufferSize())) {
        c = UART_1_GetByte() ;
        result = is_delimiter(c) ;
        if (result) { /* a string delimter was detected */
            str[str_index] = 0 ;
            str_index = 0 ;
        } else { /* still in the middle of a string */
            str[str_index++] = c ;
            if (str_index >= STR_LEN) { /* string is too long */
                str[STR_LEN] = 0 ;
                str_index = 0 ;
                result = -1 ;
            }
```

```
        }
    }
    return( result ) ;
}

int get_str(void){
    int result = 0 ;
    while(result == 0) {
        result = check_str() ;
    }
    return( result ) ;
}

int isNumber(const char * tmp){
    int isDigit = 0;
    unsigned int j=0;
    while(j<strlen(tmp) && isDigit == 0){
        isDigit = isdigit(tmp[j]);
        j++;
    }
    return isDigit;
}
```

```
int main (){
    CyGlobalIntEnable;
    UART_1_Start();
    //rxTor_StartEx(InterrupRX);
    int grados = 0;
    cls();
    for(;;){
        while (!check_str()) {
            cls();
            print("\r\nPASILLAS\r\n");
            print("\n\rGRADOS: ");
            print(str);
            CyDelay(50);
        }
        if(!isNumber(str)){
            print("\n\rERROR DATO NO VALIDO!!");
        } else {
            sscanf(str, "%d", &grados);
            if(!(grados<=720) || !(grados>=-720)){
                print("\n\rFUERA DE RANGO");
            } else {
            print("\n\rGIRANDO");
            if(grados <= 0){
```

```
            antihorario(-grados);
            grados = 0;
        } else {
            horario (grados);
        }
      }
    }
    CyDelay(1000);
    *str = '\0';
    memset(&str[0], 0, sizeof(str));
    UART_1_ClearTxBuffer();
    UART_1_ClearRxBuffer();
  }
}
```

Conclusión: en esta práctica se vio la importancia de los controladores UART en el mundo de la robótica ya que es esencial para los robots polares y cilíndricos ya que esos son lis robots que giran sobre su propio eje simultáneamente, y al llevar motores paso a paso son muy precisos en cuanto a los movimientos