"MANEJO DE COMUNICACIÓN SERIAL I2C EN ARDUINO"

Trabajo Preparatorio N°9
Laboratorio de Sistemas Embebidos

Melanny Cecibel Dávila Pazmiño
Ingeniería en Telecomunicaciones
Facultad de Eléctrica y Electrónica
Quito, Ecuador
melanny.davila@epn.edu.ec

***Abstract*—En el siguiente documento se presenta el sustento teórico de la comunicación serial I2C en el microcontrolador Arduino Uno.**

***Index Terms*—Arduino, I2C, esclavo, maestro.**

I. Introducción

El estándar I2C (Inter-Integrated Circuit) fue desarrollado por Philips en 1982 para la comunicación interna de dispositivos electrónicos en sus artículos. Posteriormente fue adoptado progresivamente por otros fabricantes hasta convertirse en un estándar del mercado [1].

II. Objetivos

- Relacionar al estudiante con el uso y manejo de comunicación serial I2C en Arduino Uno.
- Establecer comparaciones entre los diferentes tipos de comunicación serial en Arduino Uno [2].

III. Cuestionario

A. Consultar las características relacionadas al uso de la comunicación serial I2C en la placa Arduino Uno.

El bus I2C requiere únicamente dos cables para su funcionamiento, uno para la señal de reloj (CLK) y otro para el envío de datos (SDA), lo cual es una ventaja frente al bus SPI. Por contra, su funcionamiento es un poco más complejo, así como la electrónica necesaria para implementarla [1].

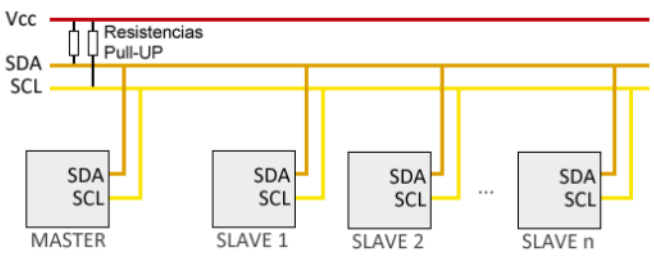

Fig. 1. Esquema de comunicación de I2C

En el bus Cada dispositivo dispone de una dirección, que se emplea para acceder a los dispositivo de forma individual. Esta dirección puede ser fijada por hardware (en cuyo caso, frecuentemente, se pueden modificar los últimos 3 bits mediante jumpers o interruptores) o totalmente por software. En general, cada dispositivo conectado al bus debe tener una dirección única.

Arduino dispone de soporte I2C por hardware vinculado físicamente a ciertos pines. También es posible emplear cualquier otro grupo de pines como bus I2C a través de sofware, pero en ese caso la velocidad será mucho menor [3].

Los pines a los que está asociado varían de un modelo a otro. La siguiente tabla muestra la disposición en alguno de los principales modelos.

TABLA I
Principales modelos del microcontrolador Arduino

Modelo	SDA	SCK
Uno	A4	A5
Nano	A4	A5
Mini Pro	A4	A5
Mega	20	21

B. Consultar y describir detalladamente el proceso de comunicación I2C con otros dispositivos y la asignación de direcciones.

El bus I2C tiene una arquitectura de tipo maestro-esclavo. El dispositivo maestro inicia la comunicación con los esclavos, y puede mandar o recibir datos de los esclavos. Los esclavos no pueden iniciar la comunicación (el maestro tiene que preguntarles), ni hablar entre si directamente [1].

Es posible disponer de más de un maestro, pero solo uno puede ser el maestro cada vez. El cambio de maestro supone una alta complejidad, por lo que no es algo frecuente.

El bus I2C es síncrono. El maestro proporciona una señal de reloj, que mantiene sincronizados a todos los dispositivos

del bus. De esta forma, se elimina la necesidad de que cada dispositivo tenga su propio reloj, de tener que acordar una velocidad de transmisión y mecanismos para mantener la transmisión sincronizada (como en UART).

El protocolo I2C prevé resistencias de Pull-UP de las líneas a Vcc. En Arduino frecuentemente no se instalan estas resistencias, ya que la librería Wire activa las resistencias internas de Pull-UP. Sin embargo las resistencias internas tienen un valor de entre 20-30 k Ohmnios, por lo que son unas resistencias de Pull-UP muy blandas.

Usar unas resistencias blandas implica que los flancos de subida de la señal serán menos rápidos, lo que implica que se podrá usar velocidades menores y distancias de comunicación inferiores. Si se desea emplear velocidades o distancias de transmisión superiores, se deberá poner físicamente resistencias de Pull-UP de entre 1k a 47k Ohmios.

Para poder realizar la comunicación con solo un cable de datos, el bus I2C emplea una trama (el formato de los datos enviados) amplia. La comunicación costa de:

- 7 bits a la dirección del dispositivo esclavo con el que queremos comunicar.
- Un bit restante indica si queremos enviar o recibir información.
- Un bit de validación.
- Uno o más bytes son los datos enviados o recibidos del esclavo.
- Un bit de validación.

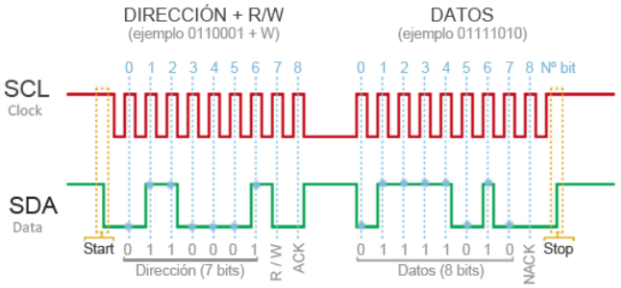

Fig. 2. Funcionamiento de las direcciones del bus I2C

Con estos 7 bits de dirección es posible acceder a 112 dispositivos en un mismo bus (16 direcciones de las 128 direcciones posibles se reservan para usos especiales)

Este incremento de los datos enviados (18bits por cada 8bits de datos) supone que, en general, la velocidad del bus I2C es reducida. La velocidad estándar de transmisión es de 100kHz, con un modo de alta velocidad de 400kHz.

C. Consultar la librería Wire.h, para el manejo del bus I2C, en un cuadro, liste y describa las funciones relacionadas a esta.

Para usar el bus I2C en Arduino, el IDE Standard proporciona la librería "Wire.h", que contiene las funciones necesarias para controlar el hardware integrado.

Algunas de las funciones básicas son las siguientes:

- Wire.begin(): Inicializa el hardware del bus.
- Wire.beginTransmission(address): Comienza la transmisión hacia la dirección "address".
- Wire.endTransmission(): Finaliza la transmisión.
- Wire.requestFrom(address,nBytes): Solicita un número de bytes al esclavo en la dirección "address".
- Wire.available(): Detecta si hay datos pendientes por ser leídos.
- Wire.write(): Envía un byte.
- Wire.read(): Recibe un byte.
- Wire.onReceive(handler): Registra una función de callback al recibir un dato.
- Wire.onRequest(handler): Registra una función de callback al solicitar un dato.

D. Realizar un programa en Tinkercad que permita la creación y visualización de los 3 caracteres personalizados mediante el LCD 16X2. El conjunto de caracteres a representar serán los símbolos de "Dinosaurio" que se presenta en Google cuando no existe conexión a Internet.

A continuación se presenta el esquemático de circuito que permite la visualización de los 3 caracteres personalidados.

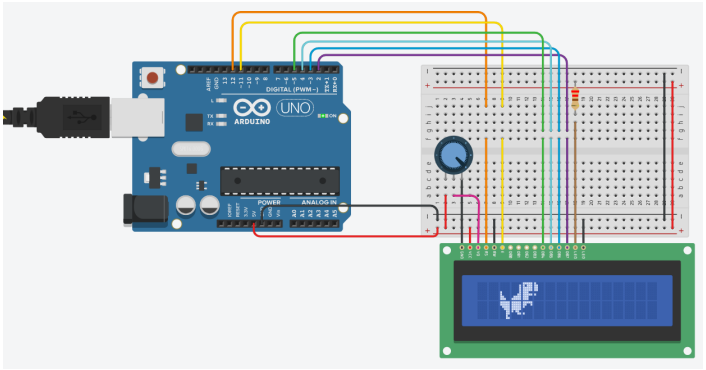

Fig. 3. Esquemático del circuito implementado

El código 1, permite obtener lo solicitado.

```
/* Trabajo preparatorio Nro. 9
Implementacion del simbolo de Dinosaurio de navegador de
Google cuando existe desconexion*/

//Biblioteca del LCD
#include <LiquidCrystal.h>

//Objeto LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

/* Arreglo de 8 Bytes para crear al simbolo de Dinosaurio, tal
que se tiene:
  A B C D
  E F G H*/

//Descripcion de los segmentos encendidos por bloques

byte A[8] = {B00000, B00000, B00000, B00000,
    B10000, B10000, B11000, B11100};
```

```
byte B[8] = {B00000, B00001, B00001, B00001,
    B00001, B00011, B01111, B11111};
byte C[8] = {B11111, B10111, B11111, B11111,
    B11100, B11111, B11100, B11100};
byte D[8] = {B10000, B11000, B11000, B11000,
    B00000, B10000, B00000, B00000};
byte E[8] = {B11111, B11111, B01111, B00111,
    B00011, B00011, B00010, B00011};
byte F[8] = {B11111, B11111, B11111, B11111,
    B10110, B00010, B00010, B00011};
byte G[8] = {B11111, B11001, B10000, B00000,
    B00000, B00000, B00000, B00000};
byte H[8] = {B11111, B11111, B01111, B00111,
    B00011, B00011, B00000, B00000};
byte I[8] = {B11111, B11111, B11111, B11111,
    B00110, B10010, B00010, B00011};
byte J[8] = {B11111, B11111, B11111, B11111,
    B10110, B00011, B00000, B00000};
byte K[8] = {B11111, B11001, B10000, B00000,
    B00000, B10000, B00000, B00000};

int aux = 0; //Variable auxiliar

void setup()
{
  //Control del LCD 16x2
  lcd.begin(16, 2);
}

//Funcion que crea y mueve el dinosaurio
void Movimiento()
{

  while(aux <16)
  {

    lcd.clear(); //Pantalla sin mostrar nada
  Dinosaurio1(); //se muestra el dinosaurio 1
    delay(500); //retardo
    lcd.clear(); //Pantalla sin mostrar nada
  Dinosaurio2();//Se muestra el dinosaurio 2
    delay(500);//retardo
    lcd.clear();//Pantalla sin mostrar nada
    Dinosaurio3();//Se muestra el dinosaurio 3

    delay(500);//Retardo

    aux = aux + 3;/*Incremento de la variable en
    3 porque se mostraron 3 tipos de dinosarios
    */

    if(aux == 15)
    {
      aux = 0;
    }
  }
}

void loop()
{
  Movimiento();
}


void Dinosaurio1(){
  //Caracteres de escritura
  lcd.createChar (0, A);
  lcd.createChar (1, B);
  lcd.createChar (2, C);
  lcd.createChar (3, D);
  lcd.createChar (4, E);
  lcd.createChar (5, F);
  lcd.createChar (6, G);

  /*Escritura en la pantalla
  Se indica la  posicion del cursor
  tal que lcd.setCursor(Columna,Fila)
  Primera fila*/
  lcd.setCursor(aux, 0);
  lcd.write((byte)0);
  lcd.write((byte)1);
  lcd.write((byte)2);
  lcd.write((byte)3);

  //Segunda Fila
  lcd.setCursor(aux, 1);
  lcd.write((byte)4);
  lcd.write((byte)5);
  lcd.write((byte)6);
}

void Dinosaurio2(){
  //Caracteres para escritura
  lcd.createChar (0, A);
  lcd.createChar (1, B);
  lcd.createChar (2, C);
  lcd.createChar (3, D);
  lcd.createChar (4, H);
  lcd.createChar (5, I);
  lcd.createChar (6, G);

  /*Escritura en la pantalla
  Se indica la  posicion del cursor
  tal que lcd.setCursor(Columna,Fila)
  Primera fila*/
  lcd.setCursor(aux+1, 0);
  lcd.write((byte)0);
  lcd.write((byte)1);
  lcd.write((byte)2);
  lcd.write((byte)3);

  //Segunda Fila
  lcd.setCursor(aux+1, 1);
  lcd.write((byte)4);
  lcd.write((byte)5);
  lcd.write((byte)6);
}

void Dinosaurio3(){
  //Caracteres para escritura
  lcd.createChar (0, A);
  lcd.createChar (1, B);
  lcd.createChar (2, C);
  lcd.createChar (3, D);
  lcd.createChar (4, E);
  lcd.createChar (5, J);
  lcd.createChar (6, K);

  /*Escritura en la pantalla
  Se indica la  posicion del cursor
  tal que lcd.setCursor(Columna,Fila)
  Primera fila*/
  lcd.setCursor(aux+2, 0);
  lcd.write((byte)0);
  lcd.write((byte)1);
  lcd.write((byte)2);
  lcd.write((byte)3);

  //Segunda Fila
  lcd.setCursor(aux+2, 1);
  lcd.write((byte)4);
  lcd.write((byte)5);
  lcd.write((byte)6);
}
```

Código 1: Implementación síbolo de dinosaurio de Google

E. Realizar un programa en Tinkercad que permita controlar el encendido de 4 Bombillas. El circuito estará conformado por dos placas Arduino Uno conectadas en modo Maestro-Esclavo, utilizando comunicación serial I2C. En la placa Maestro se conectarán 4 pulsadores que controlan el encendido de las 4 bombillas. Las bombillas estarán conectadas en la placa Esclavo.

En la siguiente figura, se muestra el esquemático implementado que controla el funcionamiento de 4 bombillas.

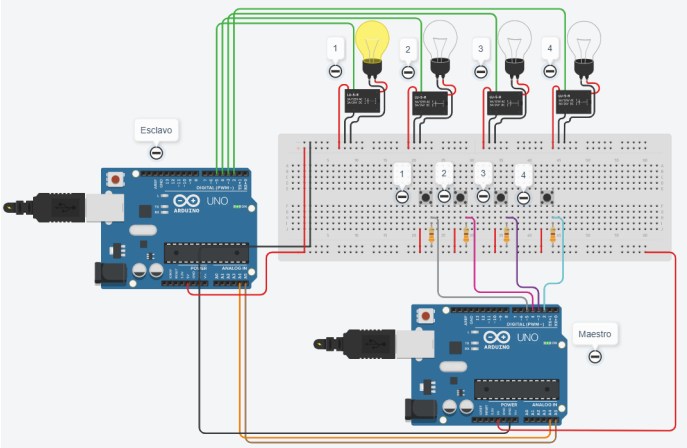

Fig. 4. Esquemático del circuito implementado

Los siguientes segmentos de código permiten controlar a los dispositivos: maestro y esclavo.

```
/*Maestro
En base a la lectura de los pushbuttons
accionados, recoge la senal de informacion
y envia dicha informacion al dispositivo esclavo
    */

//Libreria Wire para el uso de I2C
#include <Wire.h>

void setup()
{
  //Indico que esta placa sera la maestra
  Wire.begin(1);
  //Indico los pines para lectura
  pinMode(2, INPUT);
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
}

//Variables auxiliares
int b1,b2,b3,b4;

//Funcion Detectar, de los pulsadores
void Detectar()
{
  //Valor de los pulsadores de las bombillas 1, 2, 3 y 4
  b1= digitalRead(2);
  b2 = digitalRead(3);
  b3 = digitalRead(4);
  b4 = digitalRead(5);

  //Validacion de los pulsadores y envio de informacion
  if(b1 == HIGH)
  {
    //Comienzo de la transmision
    Wire.beginTransmission(1);
    //Enviamos el pin a encender
    Wire.write(2);
    // Paramos la transmisi n, se envia lo escrito con write
    Wire.endTransmission();
    b1 = 0;

  }
  else if(b2== HIGH)
  {
    //Comienzo de la transmision
    Wire.beginTransmission(1);
    //Encendido del pin
    Wire.write(3);
    //Fin de la transmision, escritura de la informacion
    Wire.endTransmission();
    b2= 0;
  }
  else if(b3 == HIGH)
  {
    //Comienzo de la transmision
    Wire.beginTransmission(1);
    //Encendido del pin
    Wire.write(4);
    //Fin de la transmision, escritura de la informacion
    Wire.endTransmission();
    b3= 0;
  }
  else if(b4== HIGH)
  {
    //Comienzo de la transmision
    Wire.beginTransmission(1);
    //Encendido del pin
    Wire.write(5);
    //Fin de la transmision, escritura de la informacion
    Wire.endTransmission();
    b4= 0;
  }
}

void loop()
{
  Detectar();
}
```

Código 2: Segmento del dispositivo maestro

```
/*Esclavo
Control del encendido de las bombillas en base a la
se al leida por parte de los pushbutton*/

//Libreria Wire para la implementacion de I2C
#include <Wire.h>

void setup()
{
  //Direccion del esclavo
  Wire.begin(1);

  //Funcion que lee los datos
  Wire.onReceive(Recibir);

  //Descripcion de los pines usados como salida
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
```

```
}

//Variables auxiliares
volatile int pin;
int aux1 = 0, aux2 = 0, aux3 = 0, aux4 = 0;

//Funcion para recibir los bytes en una comunicacion I2C
void Recibir(int num)
{
  //Recepcion de bytes
  if (Wire.available() >0)
  {
    // Lectura del byte
    pin = Wire.read();
  }
}

/*Funcion Encender, controla a las senales que se envian
a los pines para encender las bombillas*/
void Encender()
{
  //Pin en cero
  if (pin == 2)
  {
    //enceramos pin para evitar confusiones
    pin = 0;
    //Cambio del estado del pin
    if (aux1 == 0)
    {
      digitalWrite(2,HIGH);
      aux1 = 1;
    }
    else
    {
      digitalWrite(2,LOW);
      aux1 = 0;
    }
  }
  else if (pin == 3)
  {
    //Pin en cero
    pin = 0;
    //Cambio del estado del pin
    if (aux2 == 0)
    {
      digitalWrite(3,HIGH);
      aux2 = 1;
    }
    else
    {
      digitalWrite(3,LOW);
      aux2 = 0;
    }
  }
  else if (pin == 4)
  {
    //Pin en cero
    pin = 0;
    //Cambio del estado del pin
    if (aux3 == 0)
    {
      digitalWrite(4,HIGH);
      aux3 = 1;
    }
    else
    {
      digitalWrite(4,LOW);
      aux3 = 0;
    }
  }
  else if (pin == 5)
  {
    //Pin en cero
    pin = 0;
    //Cambio del estado del pin
    if (aux4 == 0)
    {
      digitalWrite(5,HIGH);
      aux4 = 1;
    }
    else
    {
      digitalWrite(5,LOW);
      aux4 = 0;
    }
  }
}

void loop()
{
  Encender();
}
```

Código 3: Segmento del dispositivo esclavo

REFERENCES

[1] J. M. — Arduino — 7, "How to Setup I2C Communication on the Arduino", Circuit Basics, abr. 24, 2020. https://www.circuitbasics.com/how-to-set-up-i2c-communication-for-arduino/ (accedido feb. 04, 2021).

[2] E. Espinosa, E. Tatayo, "MANEJO DE COMUNICACIÓN SERIAL I2C EN ARDUINO". C.P. SISTEMAS EMBEBIDOS, Accedido: feb. 02, 2021. [En línea].

[3] L. Llamas "El bus I2C en Arduino", https://www.luisllamas.es/arduino-i2c/ (accedido feb. 03, 2021).

[4] "arduino - I2C Communication — arduino Tutorial". https://riptutorial.com/arduino/topic/9092/i2c-communication (accedido feb. 03, 2021).

[5] "Arduino Uno- I2C Communication". https://sodocumentation.net/arduino/topic/9092/i2c-communication (accedido feb. 03, 2021).