

“MANEJO DE COMUNICACIÓN SERIAL I2C EN ARDUINO”

Informe N°9

Laboratorio de Sistemas Embebidos

Melanny Dávila

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

2nd Jonathan Álvarez

Ingeniería en Telecomunicaciones
Facultad de Eléctrica y Electrónica
Quito, Ecuador
jonathan.alvarez@epn.edu.ec

Abstract—En el presente informe se presentará comparaciones entre los distintos protocolos de comunicación serial que maneja Arduino Uno seguido de las ventajas y desventajas que presenta el uso de protocolos de comunicación UART frente a I2C. Finalmente se presentará un circuito que permitirá controlar el sentido de giro y velocidad de un motor DC.

Index Terms—I2C, comunicación serial, maestro, esclavo.

I. INTRODUCCIÓN

El bus I2C es un bus de 2 hilos o líneas: SCL (línea del reloj de sincronización de las transferencias de datos) y SDA (línea de datos). Ambos hilos se conectan a todos los dispositivos que estén conectados al bus. Además, existirá un tercer hilo con la masa (GND) común a todos los dispositivos. El microcontrolador principal (Master) se encarga de hacer las peticiones a las direcciones físicas de los dispositivos y con ordenes concretas a los accesos de memoria y de cómo esté programado internamente cada dispositivo.

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. Establecer una tabla comparativa de los protocolos de comunicación serial (SPI, UART, I2C) que maneja Arduino Uno, comparar tasas de transmisión, modos de comunicación, cantidad de dispositivos, asignación de direcciones, instrucciones y librerías asociadas, entre otros.

La tabla comparativa se encuentra en el Anexo 1.

B. Escribir ventajas y desventajas del uso de los protocolos de comunicación UART frente a I2C.

• Ventajas:

- UART es muy simple, permite conectar de forma rápida dos dispositivos, suele usarse con RS232 o RS485.

- UART puede enviar información a distancias más largas.
- Se puede conectar de forma rápida dos dispositivos mediante el uso de UART.
- No requiere mucha complejidad en el hardware.
- I2C sólo puede trabajar en un esquema half-duplex, sin embargo, se puede implementar más de un dispositivo maestro.
- El software de I2C puede sobrecargar al procesador pero este protocolo trabaja con control de flujo.

• Desventajas:

- I2C puede enviar información a tasas de bit más altas y únicamente necesita dos cables para la comunicación.
- UART usa comunicación asincrónica, por lo que añade sobrecarga a las tramas que envía y pierde eficiencia.
- En I2C se incrementa la complejidad del circuito cuando aumentan los másteres y los esclavos.
- UART sólo permite la comunicación entre dos dispositivos.
- I2C usa acuses de recibo, que podrían disminuir la eficiencia de transmisión.
- UART tiene un máximo de dispositivos que intervienen en su comunicación.

C. Realizar un programa en Tinkercad que permita controlar el sentido de giro y velocidad de un motor DC empleando el circuito de puente H L293D. El circuito utilizará la comunicación serial I2C entre dos Arduinos Uno. El Arduino Maestro controlará la velocidad y sentido de giro por medio de tres pulsadores (Aumentar, Disminuir y Cambio de Giro), mientras que el Arduino Esclavo utilizará el motor DC y el puente H.

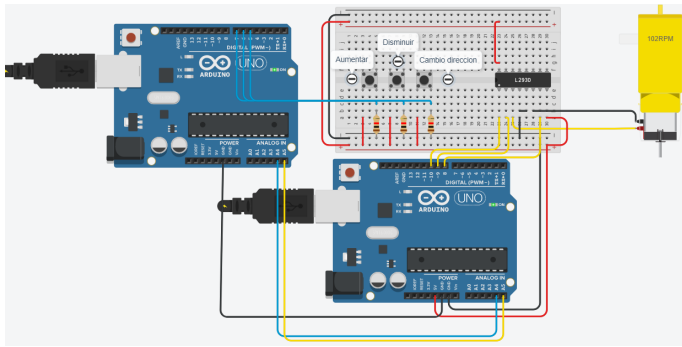


Fig. 1. Esquemático del circuito de control de velocidad y dirección de un motor usando I2C.

```

1 // CP Sistemas Embebidos
2 // Control de velocidad y direccion de un motor
3 // Arduino maestro
4 #include <Wire.h>
5 //Ingreso de los pulsadores
6 int aumentar = 7;
7 int disminuir = 6;
8 int cambio = 5;
9
10 void setup()
11 {
12     Wire.begin(1);
13     Serial.begin(9600);
14     pinMode(aumentar, INPUT);
15     pinMode(disminuir, INPUT);
16     pinMode(cambio, INPUT);
17 }
18
19 void loop()
20 //Dependiendo del pulsador enviara los datos
21 //al otro arduino
22 {
23     Wire.beginTransmission(1);
24     for(int i=5; i<8; i++){
25         if(digitalRead(i)>0){
26             Serial.println(i);
27             Wire.write(i);
28         }
29     }
30     Wire.endTransmission();
31     delay(200);
32 }

```

Script 1. Código de control y dirección de motor en el arduino maestro

```

1 // Practica 9
2 // Se presenta un circuito de control usando
3 // comunicacion I2C
4 // En un arduino se recibe las seniales de
5 // botonos y se envia
6 // hacia otro arduino que controlara un motor
7 #include <Wire.h>
8 //Definición de pines
9 int izq = 9; //Controla la direccion del motor
10 int der = 8;
11 int motor = 10; //Pin que dara la se al PWM al
12 //puente H
13 int velocidad = 0;
14 int direccion = 0;
15 int i;
16 void setup()
17 {
18     //Codigo correspondiente a la comunicacion I2C
19     Wire.begin(1);

```

```

17 Wire.onReceive(receiveEvent);
18 Serial.begin(9600);
19 pinMode(8, OUTPUT);
20 pinMode(9, OUTPUT);
21 pinMode(10, OUTPUT);
22 }
23 // Secuencia de codigo para recibir y procesar
24 // el dato
25 void receiveEvent(int howMany){
26     int recep = 0;
27     //Si existe comunicacion entra a la condicion
28     if(Wire.available() > 0){
29         recep = Wire.read();
30         Serial.println(recep);
31         //Tomando el caso respectivo ejecutamos la
32         //accion de los botones
33         switch(recep){
34             case 7:
35                 velocidad = velocidad + 30;
36                 if(velocidad > 255){
37                     velocidad = 255;
38                 }
39                 break;
40             case 6:
41                 velocidad = velocidad - 30;
42                 if(velocidad < 0){
43                     velocidad = 0;
44                 }
45                 break;
46             case 5:
47                 //Cambiamos el valor de la variable que
48                 //controla la direccion
49                 if(direccion){
50                     direccion = 0;
51                 }
52                 else{
53                     direccion = 1;
54                 }
55                 break;
56         }
57     }
58     //Funcion para que el motor gire en sentido
59     //positivo
60     void izquierda(int velocidad){
61         analogWrite(motor, velocidad);
62         digitalWrite(izq, HIGH); //Control de sentido
63         //puente H
64         digitalWrite(der, LOW);
65     }
66     //Funcion para que el motor gire en sentido
67     //negativo
68     void derecha(int velocidad){
69         analogWrite(motor, velocidad);
70         digitalWrite(der, HIGH); //Control de sentido
71         //puente H
72         digitalWrite(izq, LOW);
73     }
74     void loop(){
75         //A partir de la variable direccion llamamos a
76         //la funcion correspondiente
77         if(direccion){
78             derecha(velocidad);
79         }
80         else{
81             izquierda(velocidad);
82         }
83         // El potenciómetro controlara los pasos
84         Serial.println(velocidad);
85         delay(500);
86     }
87 }

```

Script 2. Código de control y dirección de motor en el arduino esclavo

D. Conclusiones:

Jonathan Álvarez

- Los protocolos de comunicación serial nos permiten seleccionar varias cantidades de velocidades que pueden ir desde muy bajas a muy altas lo que permiten la comunicación con gran cantidad de dispositivos.
- I2C maneja un principio de maestro esclavo al igual que el protocolo SPI, mientras que el protocolo UART no lo maneja.
- I2C tuvo la mas facil implementación de todos los protocolos utilizados y el que menos fallas presento a la hora del funcionamiento del circuito y manejo de los datos lo que indica por que es popular para comunicación con sensores

Melanny Dávila

- La comunicación I2C es fácil de implementar en comparación a otros protocolos. En el IDE de Arduino se requiere únicamente de una librería específica y de una función principal para establecer la comunicación entre los dispositivos que serán designados como maestro y esclavos.
- I2C es uno de los protocolos más utilizados para comunicarse mediante el uso de sensores digitales, ya que permite controlar el flujo de datos y obtener confirmación de los mismos.
- La implementación de aplicaciones más amigables con el usuario es posible mediante el uso de LCD, los mismos que permiten implementar diseños más dinámicos mediante el uso de sensores.

E. Recomendaciones:

Jonathan Álvarez

- Comprobar que la conexión este con los tres cables correspondientes, revisando que las tierras se encuentren conectadas entre las placas
- Utilizar la comunicación I2C o SPI si necesita que un dispositivo controle varios sensores

Melanny Dávila

- Es importante implementar el comando `lcd.clear` para poder borrar los datos presentados previamente en la pantalla y de esta manera reiniciar a a pantalla para evitar confusiones.
- Se recomienda verificar si existe comunicación entre el dispositivo esclavo y maestro mediante la impresión de datos en el monitor serial del IDE de Arduino

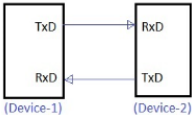
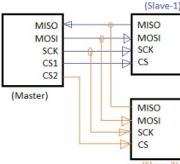
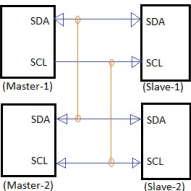
REFERENCES

- [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. 03, 2021).
- [2] E. Espinosa, E. Tatayo, “MANEJO DE COMUNICACIÓN SERIAL I2C EN ARDUINO”. C.P. SISTEMAS EMBEBIDOS, Accedido: feb. 10, 2020. [En línea].
- [3] “UART vs SPI vs I2C — Difference between UART,SPI and I2C”. <https://www.rfwireless-world.com/Terminology/UART-vs-SPI-vs-I2C.html> (accedido feb. 03, 2021).

- [4] “Understanding the Difference Between UART vs SPI”, Total Phase Blog, jun. 29, 2016. <https://www.totalphase.com/blog/2016/06/spi-vs-uart-similarities-differences/> (accedido feb. 03, 2021).

“ANEXO 1”

TABLA I
PRINCIPALES DIFERENCIAS ENTRE LOS PROTOCOLOS UART Y SPI

Parámetro	UART	SPI	I2C
Acrónimo	Universal Asynchronous Receiver/Transmitter	Serial Peripheral Interface	Inter-Integrated Circuit
Diagrama de Interfaz			
Designación de pines	TxD: Transmisor Rx D: Receptor	SCLK: reloj serial MOSI: entrada esclava, salida maestra MISO: entrada maestra, salida esclava SS: selección de esclavo	SDA: Datos en serie SCL: Reloj serial
Velocidad de datos	La velocidad de datos entre dos dispositivos debe tener el mismo valor.	La velocidad máxima admitida está entre 230kbps y 460kbps	Admite 100kbps, 400kbps, 3.4 Mbps. Algunas variantes también admiten 10Kbps y 1Mbps
Alcance mínimo	Hasta 50 pies [1]	Más de 50 pies	Más de 50 pies
Tipo de comunicación	Asíncronica	Síncronica	Síncronica
Número de maestros	No aplica	Uno [2]	Uno o más de uno
Reloj	No hace uso de ninguna señal de reloj común. Cada dispositivo usa una señal de reloj independiente.	Existe una señal de reloj común entre el maestro y el esclavo.	Existe una señal de reloj común entre varios maestros y varios esclavos.
Protocolo	Hace uso de un bit de inicio y uno de parada para 8 bits de datos.	Cada fabricante tiene sus propias especificaciones acerca de los protocolos de comunicación con los periféricos.	Utiliza bits de inicio y parada. Utiliza el bit ACK para cada 8 bits de datos, lo que indica si los datos se han recibido o no.
Direccionamiento por software	Dado que es una conexión uno a uno entre dos dispositivos, no es necesario el uso de direccionamiento.	Las líneas de selección de esclavos se utilizan para direccionar cualquier esclavo conectado a el maestro.	Habrà múltiples esclavos y múltiples maestros y todos los maestros pueden comunicarse con todos los esclavos. Se pueden conectar o direccionar hasta 27 dispositivos esclavos en el circuito de interfaz I2C.
Ventajas	*Comunicación simple. *Muy popular. *Casi todos los dispositivos presentan soporte UART con conector de 9 pines, conocido como interfaz RS232 [1].	*Simple y no requiere gran procesamiento. *Comunicación full duplex. *Alcance de mayor velocidad de datos. *Uso de menor energía en comparación con I2C.	*Se pueden utilizar más de un maestro en el diseño. *Necesita sólo 2 cables para la comunicación. *Fácil direccionamiento y agregación de dispositivos no requiere líneas CS.
Desventajas	*Adecuado únicamente para comunicación entre solo dos dispositivos. *Si la velocidad de transmisión no se fija previamente entre los dispositivos existirá distorsión.	*A mayor número de esclavos, aumenta la complejidad de hardware, mayor número de pines usados [2]. *Al agregar nuevos dispositivos, se debe agregar líneas CS lo que genera cambios en software *No existe control de flujo.	*Aumenta la complejidad del circuito cuando aumenta el número de esclavos y maestros. *La interfaz es semidúplex. *Requiere una pila de software para controlar el protocolo y mayor trabajo en el microcontrolador [3].