Laboratorio de Principios de Mecatrónica
Práctica 3. Protocolos de Comunicación

Jorge Alejandro Ramírez Gallardo
Ingeniería Mecatrónica e Ingeniería Industrial
ITAM
Ciudad de México, México
rmrez.alejandro.g@gmail.com

Ricardo Edward Meadowcroft
Ingeniería en Computación y Lic. en Matemáticas Aplicadas
ITAM
Ciudad de México, México
rmeadowc@itam.mx

***Abstract*—En esta práctica se realizó una conexión entre dos dispositivos Arduino mediante los protocolos I2C y SPI, y por comunicación inalámbrica mediante módulos xBee, para transmitir distinta información, de origen analógico y digital, entre ellos.**

I. Introducción

Para una variedad de funciones en que es importante el uso de sistemas embebidos, no es suficiente la información de sensores y la acción de dispositivos conectados directamente al microcontrolador de manera física para llevar a cabo su trabajo correctamente. Es entonces necesario reconocer la aplicación de los protocolos de comunicación entre microcontroladores para poder extender la utilidad de tales sistemas.

En esta práctica, se compara entre protocolos existentes para tal propósito, y se aplica el uso de la emisión de ondas electromagnéticas de radio mediante módulos XBee para la comunicación inalámbrica.

II. Marco teórico

A. Comparación entre I2C y SPI

Ambos sistemas permiten la transferencia de datos entre dos sistemas embebidos de manera similar, sin embargo, la manera en que cada uno es implementado tiene implicaciones en la eficiencia y costo de la comunicación llevada a cabo. Concretamente, I2C usa 2 'líneas' de comunicación, uno que lleva datos de maestro a esclavo, o viceversa, y uno que lleva pulsos de reloj para sincronizar ambos; por otro lado, SPI tiene cuatro líneas, con un reloj, líneas separada para transferencia esclavo-maestro y maestro-esclavo, y una línea para seleccionar qué esclavo a transmitir. [1]

De este modo, SPI puede leer y escribir simultáneamente al contrario de I2C, y su arquitectura permite conectar múltiples esclavos a un maestro; sin embargo, con I2C también es permitido conectar múltiples maestros. [2]

En general el SPI permite una comunicación más rápida, pero el I2C tiene además posibilidad de 'estiramiento de reloj', en donde al superar la velocidad transmisión de información la capacidad disponible, se transmite el reloj más lentamente, distorsionando la velocidad, pero permitiendo que toda la información llegue. SPI es generalmente más eficiente en poder, pero no manda bits que indican el inicio y fin de una transmisión de bits, al contrario de I2C, además de otros mecanismos de verificación de transmisión, y entonces suele contar con mayor ruido informático. [3]

III. Desarrollo

La presente práctica contiene tres secciones para su realización En primera instancia, se introduce a los protocolos de I2C y SPI de para la comunicación Master-Slave entre dos microcontroladores AVR para regular el modo de giro de un motor a través de un puente H.

En segunda instancia, se implementa el uso de acelerómetros para regular la velocidad con la que el motor gira de acuerdo con el ángulo obtenido.

Por último, se implementa la comunicación inalámbrica entre dos dispositivos XBee para introducir a los alumnos a la comunicación sin cableado que ocupará necesariamente para el posterior desarrollo del proyecto final.

A. I2C vs. SPI

La comunicación entre dispositivos es posible a través de protocolos de comunicación. En el desarrollo de la presente práctica, destacamos el uso de dos tipos de protocolos de comunicación para conectar los AVR ATMEGA2560: el protocolo I2C y el protocolo SPI.

En el caso del protocolo I2C, la organización de las conexiones y la programación de los dispositivos se llevó a cabo de la siguiente forma. Se conectaron las dos vías de comunicación entre el AVR controlador (Master) y el AVR controlado (Slave) a través de los puertos del reloj serial (SCL) y de los datos en serie (SDA), siendo en el último por donde se envía la información correspondiente a la transferencia de datos. La conexión de ambos dispositivos se ilustra a continuación.

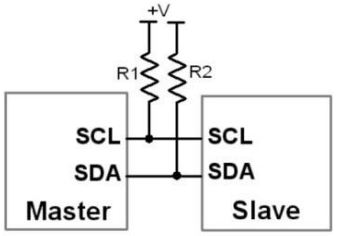

Imagen 1. Conexiones entre AVR para comunicación I2C.

La implementación del código referente a la habilitación y ejecución del protocolo I2C dentro del Master se muestra a

continuación, instanciando los dos botones utilizados para el proceso de comunicación conforme a lo descrito en el marco de trabajo de la práctica, logrando con ello la comunicación serial entre los dos dispositivos.

```
#include <Wire.h>

int button1 = 1;
int button2 = 2;

void setup() {
  Wire.begin(); // join i2c bus (address optional for
master)
  pinMode(button1,INPUT);
  pinMode(button2,INPUT);
  }
int x = 0;
int y = 0;

void loop() {
  // transmit to device #8
  x = digitalRead(button1);
  y = digitalRead(button2);
  Wire.beginTransmission(8);
  Wire.write(x);
  Wire.write(y);
  // sends one byte
  Wire.endTransmission();
  // stop transmitting
  delay(250);
}
```

En segundo lugar, mostramos la implementación del código referente a la habilitación y ejecución del protocolo I2C dentro del Slave, instanciando los dos datos utilizados para el proceso de cambio de uso del motor conectado al Slave mediante un puente H y conforme a lo descrito en el marco de trabajo de la práctica.

```
int pin1 = 8;
int pin2 = 9;
int c1 = 0;
int c2 = 0;

#include <Wire.h>

void setup() {
  Wire.begin(8); // join i2c bus with address #8
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600); // start serial for output
  pinMode(pin1,OUTPUT);
  pinMode(pin2,OUTPUT);
}

void loop() {
  if (c1 == 0){
     digitalWrite(pin1,LOW);
    }
    if (c1 == 1){
      digitalWrite(pin1,HIGH);
    }
    if (c2 == 0){
      digitalWrite(pin2,LOW);
    }
    if (c2 == 1){
      digitalWrite(pin2,HIGH);
    }
}

void receiveEvent(int howMany) {
    if (Wire.available()==2){
      c1 = Wire.read();
      Serial.print(c1); // print the character
    }
    if (Wire.available()==1){
      c2 = Wire.read();
      Serial.print(c2); // print the character
    }
}
```

El siguiente protocolo de comunicación es SPI, con el cual se implementan cuatro líneas: el envío de información del Master al Slave (MOSI), el envío de información del Slave al Master (MISO), el reloj (SCK) y el pin Slave Select (SS) que controla la autorización del traspaso de información del Master al Slave. El armado del cableado requerido para esta comunicación se ilustra a continuación para un Arduino UNO, aunque el concepto y aplicación es el mismo para el ATMEGA 2560, con la diferencia del número de los pines designados para el uso del protocolo.

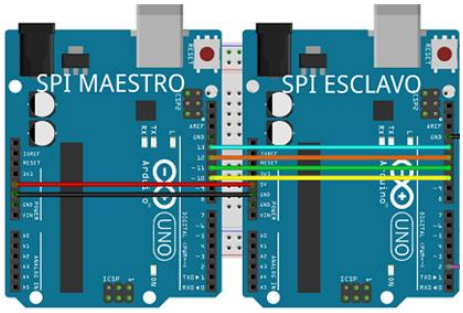

Imagen 2. Conexiones entre AVR (Arduino UNO) para comunicación SPI.

La implementación del código utilizado para la comunicación, así como para la activación del motor con la información recibida se muestra a continuación, comenzando por el código referente al Master.

```
#include <SPI.h>
int button1 = 1;
int button2 = 2;

int x=0;
int y=0;

void setup()
{
  Serial.begin (9600);
  pinMode(button1,INPUT);
  pinMode(button2,INPUT);
  digitalWrite(SS, HIGH);
  SPI.begin();
```

```
}

void loop()
{
  // Bus SPI de 2MHz, modo cero
  SPI.beginTransaction(SPISettings(2000000,
MSBFIRST, SPI_MODE0));
  digitalWrite(SS, LOW);
 x =  digitalRead(button1);
 y =  digitalRead(button2);

 SPI.transfer(x);
 SPI.transfer(y);
digitalWrite(SS, HIGH);
SPI.endTransaction();
SPI.end();
}
```

Por su parte, el código relacionado con la configuración del Slave de la comunicación es el siguiente.

```
volatile byte posicion;
volatile boolean control;
int led = 2;
char mensaje [2];

void setup ()
{
  pinMode(led,OUTPUT);
 pinMode (MISO, OUTPUT);
  SPCR |= bit (SPE);
  posicion = 0;
  control = false;
  SPI.attachInterrupt();
}

ISR (SPI_STC_vect)
{
  byte aux = SPDR;
  if (posicion < sizeof mensaje)
  {
    mensaje[posicion++] = aux;
     if (aux == '\0' | aux ==’1’)
      control = true;
  }
 else{
    posición=0;
  }
}

void loop ()
{
  if(control)
  {
    mensaje[posicion] = 0;
    int mensajeRecibido = mensaje;
    control = false;
    if(mensajeRecibido == "0"){
      // Encendemos el led en el Arduino esclavo
      digitalWrite(button1,HIGH);
      }
     if(mensajeRecibido == "1"){
      // Apagamos el led en el Arduino esclavo
      digitalWrite(button22,LOW);
      }
mensaje[posición++] = 0;
    int mensajeRecibido = mensaje;
    control = false;
    if(mensajeRecibido == "0"){
      // Encendemos el led en el Arduino esclavo
      digitalWrite(button1,HIGH);
      }
     if(mensajeRecibido == "1"){
      // Apagamos el led en el Arduino esclavo
      digitalWrite(button22,LOW);
      }

  }
}
```

B. Acelerómetro

Para esta sección de la práctica se creó un programa en el Arduino Master que tome lectura de un acelerómetro analógico, proporcionado dentro del laboratorio, con el fin de que mapeara los datos a un valor de posición angular para desplegarse apropiadamente en el monitor serial.

Posterior a dicha implementación, se configuraron ambos dispositivos Arduino para que la velocidad en el motor del Arduino Slave fuera proporcional a la posición angular del acelerómetro en un recorrido de -90 a 90 grados, mediante una conexión I2C. El código correspondiente al acelerómetro se incluye dentro del repositorio de GitHub, pues, considerando que la implementación de I2C puede observarse ´por

C. XBee

En la última sección se habilitó y configuró la comunicación inalámbrica entre dos módulos XBee mediante el software XCTU, el cual se tuvo que descargar en el equipo de cómputo para su uso. Se estableció un AVR Master y un Slave y se empleó el mismo canal de comunicación para que sólo entre estos dos dispositivos se comunicaran.

Una vez comunicados, se empleó este mecanismo para hacer que la velocidad en el motor del Arduino Slave sea proporcional a la posición angular del acelerómetro conectado al Master en un recorrido de -90 a 90 grados. El código utilizado para implementar dicha comunicación en el Arduino Master se muestra a continuación.

```
#include <Wire.h>

int freno=24;
int xpin=A3;
int valor=0;
int vel=0;
int dir=0;

void setup() {
  //Wire.begin();
  Serial.begin(4800);
  pinMode(freno,INPUT);
}
```

```
void loop() {
  valor= analogRead(xpin);

  if (digitalRead(freno)==HIGH){
    vel=255;
    dir=3;
  }else{
    //va de 0 a 255 o de 0 a -255
    if (valor>330.5 && valor<336.5){
      dir = 0;
      vel= 0;
    }else{
      dir=1;
      valor=valor-333.5;
      if (valor<0){
        valor=-valor;
        dir=2;
      }
      vel=(valor*255)/66.5;
      if (vel>255){
        vel=255;
      }
    }
  }

//  Wire.beginTransmission(8);
//  Wire.write(dir);
//  Wire.write(vel);
//  Wire.endTransmission();

  Serial.write(dir);
  delay(100);
  Serial.write(vel);
  delay(100);
}
```

A su vez, se desarrolló el código pertinente para el Arduino Slave conectado al segundo Xbee.

```
#include <Wire.h>

int fward=7;
int bward=6;
int enable=2;
int vel=0;
int dir=0;

void setup() {
//  Wire.begin(8);
//  Wire.onReceive(reception);
  Serial.begin(4800);
  pinMode(fward,OUTPUT);
  pinMode(bward,OUTPUT);
  pinMode(enable,OUTPUT);
  analogWrite(bward,0);
  analogWrite(fward,0);
  digitalWrite(enable,LOW);
}

void loop() {
  if(Serial.available()>1){
    dir=Serial.read();
    vel=Serial.read();
  }
  switch (dir) {
  case 0:
    //freno pasivo
    digitalWrite(enable,LOW);
    break;
  case 1:
    //avanzar hacia delante
    analogWrite(fward,0);
    analogWrite(bward,vel);
    digitalWrite(enable,HIGH);
    break;
  case 2:
    //avanzar hacia atras
    analogWrite(fward,vel);
    analogWrite(bward,0);
    digitalWrite(enable,HIGH);
    break;
  case 3:
    //freno activo
    pinMode(bward,OUTPUT);
    pinMode(enable,OUTPUT);
    analogWrite(bward,vel);
    analogWrite(fward,vel);
    digitalWrite(enable,HIGH);
    break;
  }
}
```

IV. Resultados

Los resultados obtenidos resaltan lo siguiente. En primera instancia, la comunicación entre dispositivos es fundamental dentro del diseño de sistemas embebidos, pues, en ocasiones, los microcontroladores que poseemos para realizar una tarea no son lo suficientemente potentes y/o capaces para poder desarrollar una tarea particular, o bien, no se pueden incorporar en dos locaciones separadas y operar de forma simultánea.

A razón de lo anterior, la clave en poder solventar los problemas de implementación yace en el uso de dos dispositivos comunicados y conectados bajo una relación Master-Slave, con la finalidad de mantener el orden y control del sistema en un solo microcontrolador, utilizando protocolos de comunicación universales, como lo son I2C o SPI, para poder efectuar las tareas descritas.

Por otra parte, el uso de dispositivos adicionales al microcontrolador, tales como los sensores, permite que este pueda tener una amplia gama de aplicaciones y usos, a la vez que le permite conectar al procesamiento del microcontrolador con la información obtenida del medio que debe evaluar (sensores) o cambiar (actuadores).

Por último, considerando que la tendencia a una mayor movilidad, el uso de tecnologías inalámbricas para lograr una comunicación adecuada y sin necesidad de ocupar conexiones alámbricas entre los dispositivos que se desean comunicar, además de que, en la implementación de dichas comunicaciones, errores de comandos o de instalación pueden surgir, por lo que es relevante revisar meticulosamente cómo se estructura el programa desarrollado y en qué condiciones de hardware va a operar.

Los códigos desarrollados pueden consultarse por GitHub en https://github.com/RmrezG/PM_Ramirez_Meadowcroft.

V. Conlcusiones

Se pudo apreciar a lo largo de la práctica la manera en que la comunicación entre sistemas computacionales no es una tarea trivial, si no que lleva consigo una variedad de consideraciones que implican la estructuración de distintos protocolos, que deben entenderse mas a fondo para aplicar tales modos de comunicación al funcionamiento correcto de sistemas entrelazados.

Así, al tener comprehensión de los principios de comunicación y transferencia de datos entre microcontroladores, las posibilidades que presentan estos para el diseño de sistemas embebidos aumenta considerablemente para resolver una variedad de problemas pertinentes en la vida real, que pueden requerir accionar o recibir información de una variedad de fuentes separadas.

VI. Referencias

[1] Amlendra. (2019, January 30). *Difference between I2C and SPI (I2C vs SPI), you should know*. Obtenido de https://aticleworld.com/difference-between-i2c-and-spi/

[2] SPI versus I2C protocols. (n.d.). Obtenido de https://bitwizard.nl/wiki/SPI_versus_I2C_protocols