Universidad Iberoamericana Puebla

Departamento de Ciencias e Ingenierías

Laboratorio de Redes Digitales

Alvaro Zaid Gallardo Hernández

Patricia Mercedes Cortés Garcia

Rosa Elena Mar Larios

Date: 23rd February 2023 Maestro

Word count: 2,999 Huber Giron Nieto

Practica 1: "Protocolo de Comunicación – Contador de pulsos y contador de tiempo"

Contents

1 Resumen

A lo largo de este trabajo se elaboraran algunos ejercicios sobre un protocolo de comunicación relacionados al pulso en Aruino.

2 Introducción

El protocolo de comunicación es un conjunto de reglas y convenciones que se utilizan para permitir la comunicación entre dos dispositivos electrónicos. En el contexto de los contadores de pulsos y contadores de tiempo en Arduino, el protocolo de comunicación se refiere a las reglas y convenciones que se utilizan para transmitir los datos medidos desde el Arduino a otro dispositivo.

En este caso, el Arduino se encarga de medir ya sea la frecuencia de una señal (contador de pulsos) o el tiempo durante el cual un pin se mantiene en estado alto (contador de tiempo).

Estos datos medidos pueden ser transmitidos a otro dispositivo, como una computadora o un dispositivo móvil, utilizando un protocolo de comunicación.

3 Objetivos

3.1 General

Conocer, identificar y comprobar el funcionamiento al desarrollar un protocolo de comunicación, en esta primera parte de un contador de pulsos y un contador de tiempo en un pulso.

3.2 Específicos

Contador de pulsos

1. Utilizando 1 Arduino, diseñar una función que permita enviar un número determinado de pulsos (Variación de 0 a 1) a una frecuencia específica, la función debe tener la siguiente estructura void pulsos(int cantidad, int duración). (Comprobar el correcto funcionamiento utilizando el osciloscopio).
2. Utilizando un segundo Arduino, diseñar una función que permita contar el número de pulsos que ha recibido (Variación de 0 a 1) y mostrarlos en el monitor serial.
3. Utilizando ambos arduinos, conectar un pin de los Arduinos para comprobar el funcionamiento del sistema enviando un numero de pulsos específicos a la velocidad de 1 pulso por segundo,10, 100 y 1000 pulsos por segundo.

Contador de duración de pulsos

1. Utilizando 1 Arduino, diseñar una función que permita enviar un pulso con un tiempo en High determinado (Comprobar el correcto funcionamiento utilizando el osciloscopio).
2. Utilizando un segundo Arduino, diseñar una función que permita contar el tiempo de la duración de un pulso que ha recibido del primer Arduino (Variación de 0 a 1) y mostrarlos en el monitor serial.

4 Materiales

- 2 Arduinos
- Protoboard
- Botones

- LED's

5 Procedimiento

Contador de pulsos

Primer Arduino

```
1
2  \\Envia pulsos
3  void pulsos(int cantidad, int frecuencia)
4  {
5    for(int i=0; i<cantidad; i++)
6    {
7          digitalWrite(8, HIGH);
8          delay(1000/frecuencia);
9          digitalWrite(8, LOW);
10         delay(1000/frecuencia);
11   }
12 }
13
14 void setup()
15 {
16   pinMode(8, OUTPUT);
17   pulsos(86,100);
18 }
19 void loop()
20 {
21 }
```

Este código de Arduino crea una función llamada "pulsos" que envía un número especificado de pulsos a una frecuencia especificada en el pin 8 del microcontrolador. La función utiliza un bucle for que se ejecuta la cantidad especificada de veces. En cada iteración del bucle, el pin 8 se establece en un estado alto usando digitalWrite, seguido de un retraso calculado como 1000 dividido por la frecuencia especificada. Luego, el pin 8 se establece en un estado bajo usando digitalWrite y se espera otro retraso.

En la función de configuración (setup), se comenta la inicialización de la comunicación serial utilizando Serial.begin(). Luego, se establece el pin 8 como salida usando pinMode y se llama a la función "pulsos" con una cantidad de 86 pulsos y una frecuencia de 100 Hz. Esto enviará 86 pulsos a una frecuencia de 100 Hz en el pin 8.

La función de loop se deja vacía, lo que significa que el programa continuará ejecutando la función "pulsos" una sola vez y luego no hará nada más. Este código puede ser útil para enviar una serie de pulsos en una frecuencia específica, lo que puede ser útil en aplicaciones como la generación de señales de activación para componentes externos o la prueba de circuitos.

Segundo Arduino

```
int c =0;

void setup()
{
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), count, RISING);
}

void loop()
{
  Serial.println(c);
}

void count()
{
  c++;
}
```

Este código de Arduino crea un contador de pulsos utilizando una interrupción en el pin 2. En la función de configuración (setup), se inicia la comunicación serial a una velocidad de 9600 baudios y se configura el pin 2 como entrada con resistencia de pull-up interna utilizando la función pinMode().

Luego, se llama a la función attachInterrupt() para asociar la interrupción con el pin 2, el tipo de interrupción (RISING) y la función que se llamará cuando se detecte la interrupción (count).

En la función de loop, se imprime el valor actual de la variable "c" a través de la comunicación serial utilizando la función Serial.println().

La función count() se llama cada vez que se detecta un flanco de subida en el pin 2, lo que significa que la señal de entrada ha cambiado de bajo a alto. En la función count(), se incrementa la variable "c" en 1.

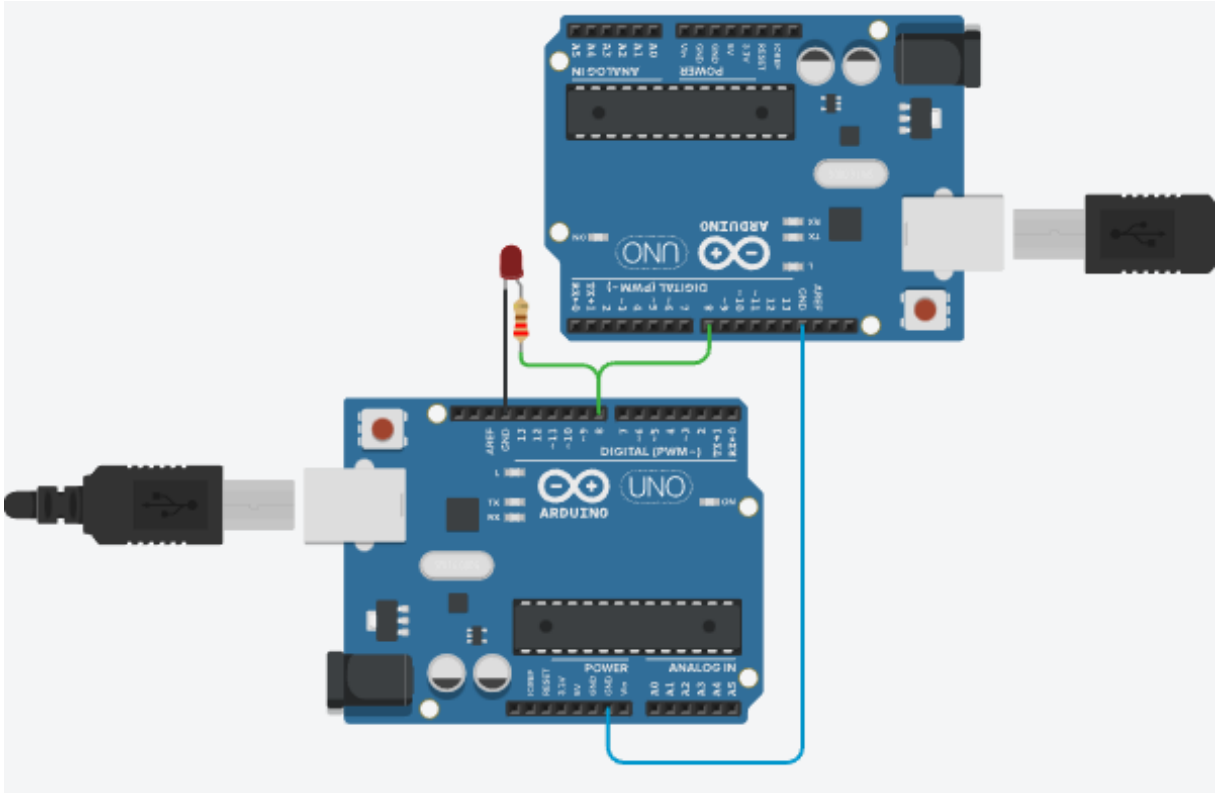

Figure 1: Esquema

****

Contador de duración de pulsos

Primer Arduino

```
1
2
3  \\Envia pulsos
4
5
6  void pulsot(int tiempo)
7  {
8          digitalWrite(8, HIGH);
9          delay(tiempo);
10         digitalWrite(8, LOW);
11         delay(tiempo);
12 }
13
14
15 void setup()
16 {
```

```
  delay(2000);
  pinMode(8, OUTPUT);
  pulsot(50);
}

void loop()
{
}
```

Este código de Arduino envía un pulso cuadrado de duración "tiempo" en milisegundos en el pin 8 utilizando la función pulsot(). En la función pulsot(), se establece el pin 8 como alto utilizando la función digitalWrite() y se retrasa el tiempo especificado en milisegundos utilizando la función delay(). Luego, el pin 8 se establece en bajo utilizando digitalWrite() y se retrasa de nuevo el tiempo especificado en milisegundos utilizando delay().

En la función de configuración (setup), se retrasa el inicio de la ejecución del programa por 2 segundos utilizando la función delay() y se configura el pin 8 como salida utilizando la función pinMode(). Luego, se llama a la función pulsot() para enviar un pulso cuadrado de 50 ms de duración.

En la función de loop(), no se realiza ninguna acción, por lo que el programa se ejecuta en un bucle infinito sin realizar ninguna operación. Este código puede ser útil para enviar pulsos cuadrados de duración determinada a dispositivos externos o para generar señales de reloj en circuitos digitales.

Segundo Arduino

```

// CONTADOR DE TIEMPO
int contador=0;

void setup()
{
  pinMode(8, INPUT);
  Serial.begin(9600);
}

void loop()
{
  if(digitalRead(8)==HIGH)
  {
    while(digitalRead(8)==HIGH)
```

```
16      {
17        delay(1);
18        contador = contador +1;
19      }
20      Serial.println(contador);
21    }
22  }
```

Este código de Arduino cuenta el tiempo en milisegundos durante el cual un pin se mantiene en estado alto. El pin se configura como entrada en la función de configuración (setup) utilizando la función pinMode(), y se inicia la comunicación serial a una velocidad de 9600 baudios utilizando la función Serial.begin().

En la función de loop(), se verifica si el pin está en estado alto utilizando la función digitalRead(). Si el pin está en estado alto, se inicia un ciclo while() que se ejecuta mientras el pin se mantiene en estado alto. Dentro de este ciclo while(), se retrasa la ejecución del programa por 1 milisegundo utilizando la función delay() y se incrementa la variable cuenta en 1 unidad en cada iteración.

Una vez que el pin vuelve a estado bajo, se imprime el valor actual de la variable cuenta utilizando la función Serial.println(). Este valor representa el tiempo en milisegundos durante el cual el pin se mantuvo en estado alto.

Este código puede ser útil para medir el tiempo de respuesta de un dispositivo externo o para detectar la duración de un pulso en una señal de entrada.

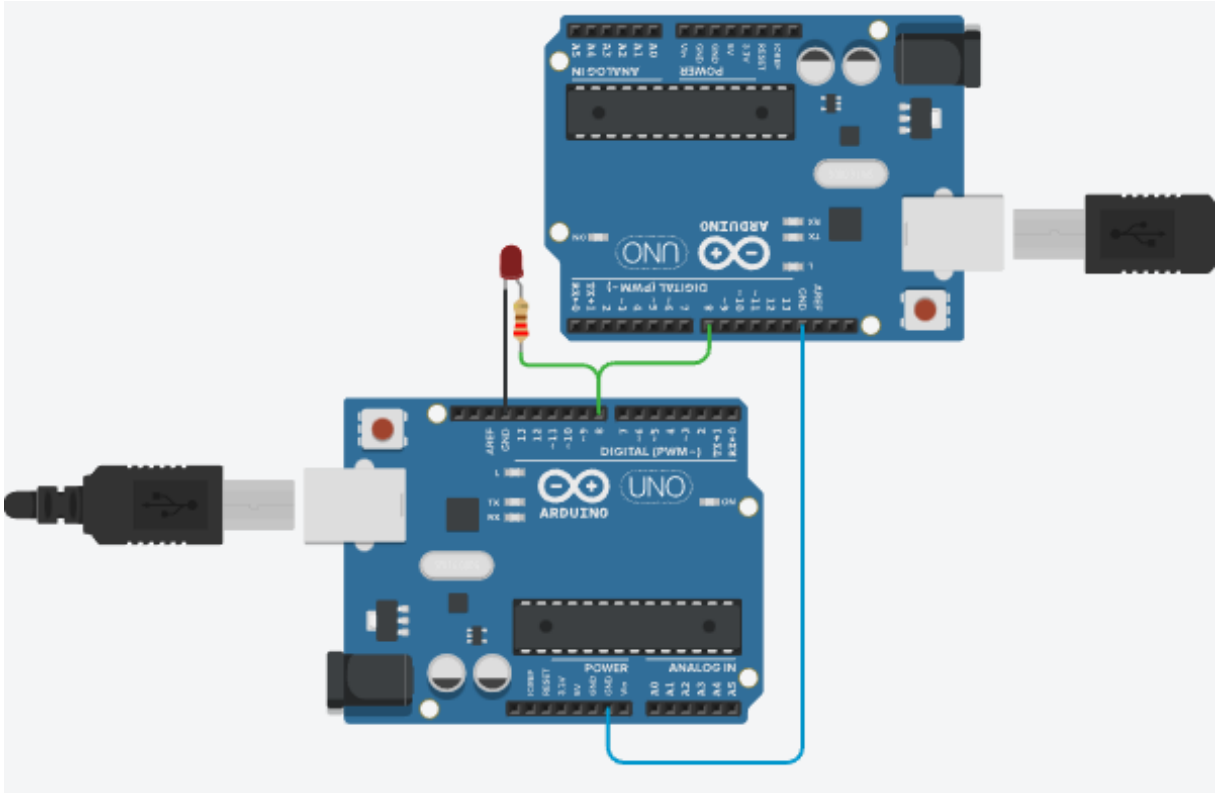

Figure 2: Esquema

6 Conslusión

A manera de conclusión se puede decir que los objetivos se han alcanzado y se ha podido entender de una mejor forma el concepto de protocolo, así como el uso de Arduino.

7 Evidencias

10 pulsos- frecuencia de 10 Hz- https://youtube.com/shorts/arJx5x5szQI?feature=share

1 pulso por segundo- https://youtube.com/shorts/H7PwIAzWNkc?feature=share

Contador de tiempo de 50 segundos- https://youtube.com/shorts/abbtBKLDztw?feature=share

Contador de pulsos, 10 pulsos por segundo- https://youtube.com/shorts/QhFiBIYqX8c?feature=share

Contador de pulsos, 100 pulsos por segundo- https://youtube.com/shorts/4fHaNvxYROs?feature=share

8 Referencias

Arduino, S. A. (2015). Arduino. Arduino LLC, 372. Banzi, M., Cuartielles, D., Igoe, T., Martino, G., Mellis, D. (2014). Arduino. The official Arduino web page at http://arduino.cc.