Comunicación Serial

Socasi Bryan[1], Viera Khaterine[2], Yánez Danilo[3]

***Abstract*— This project consists of applying a serial communication between computers to control sensors and actuators that are connected to Arduino cards. The communication must be bidirectional and crossed, this means that the software of the sensors must be able to receive the data to activate the sensors and the softaware for the sensors must be able to receive the data to activate the sensors.**

I. INTRODUCCIÓN

La evolución de la tecnología en las últimas décadas ha permitido generar nuevos protocolos y dispositivos para la transmisión de datos entre una computadora y un periférico,en sus inicios alrededor de los años 60 esta transmisión de información se daba a través de puertos seriales enviando datos bit a bit con tiempo de espera prolongado, después en1998 fue reemplazada por puertos USB's que tenían mejores características y mayor eficiencia. Actualmente los individuos no tienen pleno conocimiento sobre los fundamentos o la estructura base que permitió que la comunicación pueda darse como se la conoce ahora,sin embargo, es muy importante entender los inicios en esta área especialmente para los estudiantes de ingeniería en electrónica, pues de esta manera se logrará entender la diversidad que ha existido a lo largo del tiempo y cómo funciona cada "cosa" que utilizamos.Relacionando los temas aprendidos en clase se desarrolla un trabajo de investigación que involucra la comunicación serial entre dos PC a través de una interfaz gráfica desarrollada con lenguaje de programación Java, controlada por un arduino.

II. CONCEPTOS

A. Arduino UNO

Arduino UNO es una placa electrónica basada en el microcontrolador ATmega 328. Cuenta con 14 pines digitales que pueden ser entradas o salidas de los cuales 6 pueden ser usados como salidas PWM, 6 entradas analógicas, un cristal oscilador de 16 MHz, una conexión USB,un conector de alimentación, una cabecera ICSP y un botón de reinicio.

Fig 1. Arduino Uno

B. Definición de pines

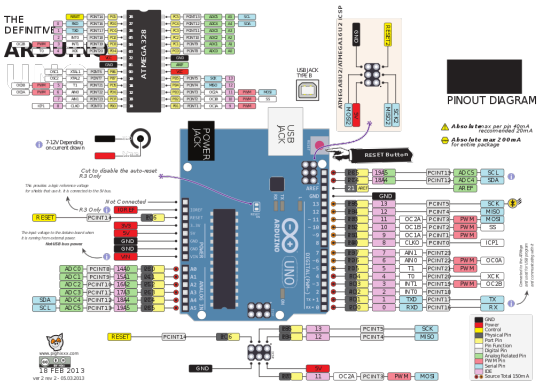

Fig 1. Definición de pines Arduino Uno

C. Java

Es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web.

Fig 2. Lenguaje de Programación JAVA

D. API Java Communications

La API de Java Communications es una extensión de Java que facilita el desarrollo de aplicaciones de comunicaciones independientes de plataforma para tecnologías como Smart Cards, sistemas integrados y dispositivos de punto de venta, dispositivos de servicios financieros, fax, módems, terminales de pantalla y equipos robóticos. La API de comunicaciones de Java también conocida como javax.comm permite a las aplicaciones el acceso al hardware

RS-232 (puertos serie) y un acceso limitado a IEEE-1284 (puertos paralelos), modo SPP. (Itcea, 2017)

1) Características:

- Enumeración de puertos (mapeo de puertos configurables por el administrador y el usuario)
- Configuración del puerto (velocidad en baudios, velocidad, bits de parada, paridad)
- Acceso a señales estándar DTR, CD, CTS, RTS y DSR EIA232
- Transferencia de datos a través de puertos RS-232
- Opciones de control de flujo de hardware y software.
- Control de umbral de buffer de recepción.
- Opción de evento asincrónico para la notificación de:
  - Datos disponibles en un puerto RS232.
  - Cambios de nivel de línea de hardware de puerto.
  - Cambios en la propiedad de puerto de una solo JVM.

E. Sensores

Es un concepto genérico que hace referencia a diferentes tipos de sensores, esto se entiende tanto las unidades que emite una señal analógica, como las unidades que emite una señal binaria (encendido o apagado).

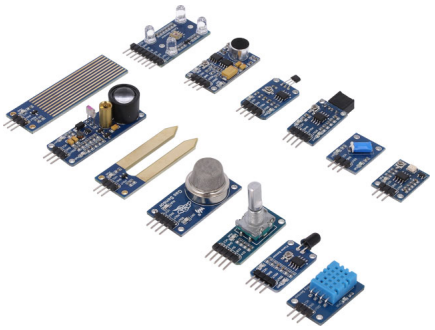

Fig 3. Sensores Electrónicos.

Convierte una magnitud física en una magnitud eléctrica, existe una gama de diferentes productos de sensores para diferentes magnitudes físicas.

III. Diagramas

A. Diagrama de bloques

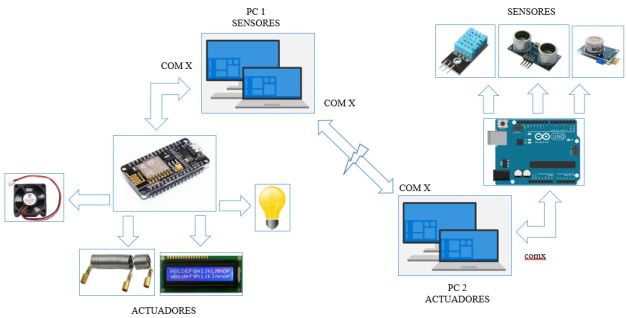

Fig 8. Diagrama de bloques de la Comunicación Serial

IV. ESTADO DEL ARTE

A. Java

En (Cisco systems, 2016) se indica qué es Java y su programación orientada a objetos, elementos del lenguaje Java, operadores y estructuras de control del lenguaje Java, fundamentos de definición y uso de clases, sistema, cadenas, cadena de buffer, math y otras clases de envoltura, arrays, clases y herencia, cómo comprender y usar los paquetes, creación de interfaces gráficas de usuario con AWT, applets y gráficos, excepciones, archivos, flujos, entrada y salida, colecciones, hilos.

B. Arduino

En (Albatish, Mosa, Abu , 2018) se encontró información sobre cómo enfrentar alguna dificultad al tratar con la plataforma Arduino mediante la descripción del diseño de un sistema de tutoría inteligente basado en escritorio. La idea principal de este sistema es una introducción sistemática al concepto de plataforma Arduino. El sistema muestra las placas de circuito de Arduino y un entorno de desarrollo de fuente abierta y una biblioteca para escribir código para controlar el tema de la placa de la plataforma Arduino. Lo que nos indica en (Pan Zhu, 2018) es que un sensor es ”un dispositivo que detecta una variación en la energía de entrada para producir una variación en otra o la misma forma de energía”. Por tanto, implementarlos con arduino, es una manera muy interesante de realizar este tipo de proyectos, de esta manera, en este artículo se describe aspectos importantes de su uso.

C. Comunicación Serial

En el artículo de (Padrón, Prieto, Herrera, Velazquez, 2018) se indica una aplicación importante con la comunicación serial pues debido a la naturaleza inherente a las comunicaciones móviles, diversos factores influyen en la calidad de los servicios que se desarrollan para los dispositivos de enlace cuando son fabricados, en los cuales no se encuentran técnicas y procedimientos de seguridad implementados en comparación con dispositivos fijos o de escritorio actuales. En este trabajo se presenta la implementación del algoritmo criptográfico por bloques ”Advanced Encryption Standard” (AES) de 128 bits, para una comunicación serial entre dos tarjetas Spartan 3E fabricadas por Xilinx. Este algoritmo público según la FIPS-197 es el estándar para muchas aplicaciones en Seguridad Informática tanto en Software como Hardware. Este método criptográfico sigue siendo un mecanismo para implementar los servicios de seguridad recomendados en la ISO 7498-2.

V. EXPLICACIÓN DEL CÓDIGO

A. Comunicación Serial

Usaremos el API Java Communications (COMM) que nos permitirá manejar el puerto serie, y así comunicarnos con el otro computador. También usaremos cables adaptadores USB a DB9 y el más importante, un cable DB9 cruzado para que el envió de bits se realice correctamente. Para usar el api de comunicaciones de java necesitaremos de una librería que contiene las clases para manejar los puertos, estos archivos se deben incluir en la carpeta donde se encuentra el Java instalado, siguiendo las siguientes direcciones:

- Comm.jar \jdk\jre\lib\ext
- Win32com.dll \jdk\bin y en \Windows\system32
- Javax.comm.properties \jdk\jre\lib

Para el código necesitaremos importar la librería mencionada anteriormente

```
import javax.comm.*;
```

A continuación tenemos que declarar las clases necesarias para manejar los puertos con sus respectivas variables asignadas, hay que tener en cuenta que la comunicación que usaremos es FULL-DUPLEX, ya que enviaremos y recibiremos información entre ambos computadores.

```
public class PuertoSerie extends Thread{
    private CommPortIdentifier idPuerto;
    private SerialPort puertoSerie; //siempr
    private OutputStream flujoSalida;
    private InputStream flujoEntrada;

    private int longitud = -1;
    private byte[] buffer = new byte[1024];
    private String aux; //Para que el mensaj
```

```
public PuertoSerie(String puerto) throws PortInUseException,
        UnsupportedCommOperationException, IOException{
    try {
        idPuerto=CommPortIdentifier.getPortIdentifier(puerto);
        if(idPuerto.isCurrentlyOwned()){
            System.out.println("puerto en uso");
        } else{
                puertoSerie=(SerialPort) idPuerto.open("comunicacion",1000);
            System.out.println("Puerto Abierto");
            puertoSerie.setSerialPortParams(115200, SerialPort.DATABITS_8,
                    SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
            System.out.println("Puerto configurado");
            flujoSalida=puertoSerie.getOutputStream();
            System.out.println("Flujo de salida configurado");
            flujoEntrada=puertoSerie.getInputStream();
            System.out.println("Flujo de entrada configurado");
            this.start();
            }
        } catch (NoSuchPortException ex) {
            Logger.getLogger(PuertoSerie.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```

Para la transmisión de información, declararemos un método con un argumento para transmitir el mensaje deseado, nos ayudaremos de un método .write() que posee la clase SerialPort.

```
public void txSerial(String mensaje){
    try {
        flujoSalida.write(mensaje.getBytes());
        System.out.println(mensaje);
    } catch (IOException ex) {
        Logger.getLogger(PuertoSerie.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Ahora para recibir datos usaremos un método run() de la clase derivada de Thread que nos permitirá crear un subproceso en el programa y con esto poner al puerto en constante "escucha", por lo que necesitaremos crear un hilo hacia el constructor que contiene la información del puerto usado.

```
public void run() {
    System.out.println("hilo iniciado");
    while (true) {
        try {
            if((longitud=flujoEntrada.read(buffer))>-1){

                aux = new String(buffer,0,longitud);
                System.out.println(aux);
                Sensores.lblRecibir.setText(aux);
                    //CO2.lblCO2.setText(aux);
            }
        } catch (IOException ex) {
            Logger.getLogger(PuertoSerie.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
```

Como usaremos varios sensores y actuares es necesario comunicarse con estos a través del puerto y luego cerrar dicha comunicación, ya que si esta quedara abierta, seria imposible comunicarse con varios sensores o actuadores a la vez, por lo que es necesario crear un método para cerrar los puertos usados.

```
public void cerrar() throws IOException{
    flujoSalida.close();
    flujoEntrada.close();
    puertoSerie.close();
}
```

Ahora necesitamos crear un Jframe para el control de esta clase, para este proyecto es necesario dos GUIs una que controlara los sensores y otra para controlar los actuadores. Básicamente necesitamos colocar un ComboBox con el nombre de los puertos y un botón para activar la comunicación. En el botón agregaremos un evento para que realice una acción al momento de ser presionado, esto nos permitirá escribir el código para activar la comunicación, para lo cual primero debemos crear un objeto de la clase que contiene los parámetros del Puerto serie y luego llamar los métodos necesarios para iniciar la comunicación, llevando la información obtenida del ComboBox y el mensaje deseado al momento de presionar el botón.

```
public class Sensores extends javax.swing.JFrame {
    private PuertoSerie puerto;
    //private SerialArduino puertoArduino;
    SerialArduino puertoArduino=new SerialArduino();
```

```
private void btnPCActionPerformed(java.awt.event.ActionEvent evt) {
try {
        puerto=new PuertoSerie(String.valueOf(cbPuertoPC.getSelectedItem()));
    } catch (PortInUseException ex) {
        Logger.getLogger(Sensores.class.getName()).log(Level.SEVERE, null, ex);
    } catch (UnsupportedCommOperationException ex) {
        Logger.getLogger(Sensores.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(Sensores.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

VI. CÓDIGO ARDUINO

1) Actuadores:

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h> //configurar puertos base

const int foco=14;
const int ventilador=12;
const int niquelina=13;


LiquidCrystal_I2C lcd(0x27, 16, 2); //protocolo 0x27, 16x2

void setup() {
pinMode(foco, OUTPUT);
pinMode(ventilador, OUTPUT);
pinMode(niquelina, OUTPUT);
pinMode(2, OUTPUT);
Wire.begin(D2, D1); // D1 = SCL | D2 = SDA
lcd.begin(); // Iniciamos el LCD
lcd.backlight(); // Prendemos la Iluminacion del LCD
lcd.clear();
Serial.begin(115200);
}
```

```
void loop() {
if(Serial.available()>0){
  int input=Serial.read();
//foco
  if(input=='0'){
    digitalWrite(foco, HIGH);
  }
  else if(input=='1'){
    digitalWrite(foco, LOW);
  }
//ventilador
    if(input=='2'){
    digitalWrite(ventilador, HIGH);
  }
  else if(input=='3'){
    digitalWrite(ventilador, LOW);
  }
//niquelina
    if(input=='4'){
    digitalWrite(niquelina, HIGH);
  }
  else if(input=='5'){
    digitalWrite(niquelina, LOW);
  }
```

A. *Sensores*

```
#include <DHT.h>

const int sensorDHT=A0;
int temp,humedad;
DHT dht (sensorDHT,DHT11);

const int echo = 2;
const int trigger = 3;
const int co2D=4;
const int co2A=A1;
int valor;
int limite;
long duracion,distancia;
int i=0;

void setup() {
  Serial.begin(9600);
  dht.begin(); //Iniciamos nuestro sensor DHT11
  pinMode(trigger,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(temp,INPUT);
  pinMode(co2D,INPUT);
  pinMode(co2A,INPUT);
}
```

```
void loop() {
  delay(1000);
    char selector=Serial.read();
  for(int i=0;i<10;i++){
    if(selector=='A'){
   Temperatura();
  }
  if(selector=='B'){
    Humedad();
  }
  if(selector=='C'){
    Proximidad();
  }
  if(selector=='D'){
    CO2();
  }
```

```
void Temperatura(){
  temp= dht.readTemperature(); //Permite leer la temperatura.
  Serial.print(temp);
  Serial.print("°C"); //Tempertura: 29°C
  delay(3000);

}

void Humedad(){
  humedad= dht.readHumidity(); //Funcion incluida en la libreria. Permite leer la humedad.
  Serial.print(humedad);
  Serial.println("%");
  delay(3000);
}

void Proximidad(){
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);   // genera el pulso de triger por 10ms
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
```

VII. CONCLUSIONES

- La comunicación de puerto serial ha tenido una breve evolución en las últimas décadas, a través del estado del arte se comprobó su aplicación y funcionamiento con la transmisión de datos, en la creación del sistema de comunicación serial presentado en este informe, se procedió a la conexión entre dos computadoras con el Arduino y el uso del ESP8266, enviando datos para realizar la comunicación serial tanto para los sensores y los actuadores, por medio de una consola de java y Arduino.
- El Arduino se puede utilizar para diferentes ámbitos como desarrollar elementos autónomos, conectándose a dispositivos e interactuar tanto con el hardware como con el software; es una plataforma electrónica que en este trabajo nos permitió la interacción con una interfaz gráfica desarrollada con lenguaje de programación Java, con la finalidad de poder tener una interacción más fácil del Arduino con el computador.
- Se logro cumplir con cada uno de los objetivos, y entender el desarrollo de todos los programas, implementados en dos computadoras, con el debido funcionamiento de los Sensores y Actuadores.

VIII. RECOMENDACIONES

- Se debe considerar que los pines en los que se conecta cada uno de los sensores y actuadores, que están configurados para funcionar en un orden específico, para obtener los datos requeríos en el Producto de unidad.
- Se debe tener en consideración que los paquetes utilizados en Java, no son propios de ahí, pero se debe descargar las librerias, y el funcionamiento de código nos permita trabajar de manera eficaz en los programas implementados..
- Conectar correctamente nuestros elementos, tanto a la tarjeta de Desarrollo del ESP8266 y la tarjeta de desarrollo Arduino UNO, debido a que, aunque realicemos correctamente la programación y la configuración con el servidor, los Actuadores y sensores se encuentran en los pines incorrectos o con la polarización indebida estos no podrán funcionar como esperamos.

REFERENCES

[1] A., P. (17 de 10 de 2012). twenergy. Recuperado el 04 de 28 de 2018, de https://twenergy.com/a/como-se-genera-la-electricidad-666

[2] Albatish, I., Mosa, M., Abu , S. (01 de 2018). ResearchGate. Obtenido de $323322871_{A}RDUINO_{T}utor_{A}n_{I}ntelligent_{T}utoring_{S}ystem$

[3] BBVA Open4U. (15 de 08 de 2015). BBVA. Obtenido de https://bbvaopen4u.com/es/actualidad/el-internet-de-las-cosas-de-codigo-abierto-plataformas-y-aplicaciones-para

[4] Carlos, C. (12 de 06 de 2015). Galia,FC. Obtenido de http://galia.fc.uaslp.mx/

[5] Cisco systems. (03 de 2016). ResearchGate. Obtenido de https://www.researchgate.net/publication/

[6] Crespo, E. (25 de 09 de 2016). Aprendiendo Arduino. Obtenido de https://aprendiendoarduino.wordpress.com/2016/09/25/que-es-arduino/

[7] Facultad Tecnológica. (02 de 10 de 2014). Universidad Distrital Francisco José de Caldas. Recuperado el 28 de 04 de 2018, de http://gemini.udistrital.edu.co/comunidad/grupos/gispud/Circuitos-II/Capitulo-3/3.2.1.html

[8] García, J. A. (s.f.). AF. Obtenido de AF: http://www.asifunciona.com/fisica/af$_{d}iodos/af_{d}iodos_{6}.htm$

[9] N.O.Sadiku, C. K. (2013). En Fundamentos de Circuitos Eléctricos (pág. 866). México: Mc Graw Hill Education