Artículo **Revista de Tecnología Informática**
Marzo, 2018 Vol.2 No.4 11-15

Aplicación para dispositivos RC con Java y Arduino

Application for RC devices with Java and Arduino

ABRIL-GARCIA, José†*, MEZA-IBARRA, Iván, ALCÁNTAR-MARTÍNEZ, Adelina y LOPEZ-ROMO, Alonso

Universidad Tecnológica de Hermosillo, Blvd de Los Seris final sur s/n., Hermosillo, Sonora, México

ID 1er Autor: *José, Abril-Garcia* / **ORC ID:** 0000-0003-3494-6817, **Researcher ID Thomson**: F-4252-2018, **arXiv Author ID**: Jhabril, **CVU CONACYT ID**: 204935

ID 1er Coautor: *Iván, Meza-Ibarra* / **ORC ID:** 0000-0001-6139-032X, **Researcher ID Thomson**: F-3550-2018, **arXiv Author ID:** imeza, **CVU CONACYT ID**: 769494

ID 2do Coautor: *Adelina, Alcántar-Martínez* / **ORC ID:** 0000-0003-2715-9209, **Researcher ID Thomson**: F-6771-2018, **CVU CONACYT ID**: 640868

ID 3er Coautor: *Alonso, Lopez-Romo* / **ORC ID:** 0000-0001-7428-1480, **Researcher ID Thomson**: R-5616-2018, **arXiv Author ID**: alonsolopezr, **CVU CONACYT ID**: 944227

Recibido: 13 de Enero, 2018; Aceptado 25 de Febrero, 2018

Resumen

Este artículo propone una aplicación que puede ser usada para el control de un dispositivo radio control (RC), interactuando en tiempo real usando el lenguaje Java y Arduino. Se presenta en forma práctica el código usado para su implementación, el seguimiento del ensamblaje de hardware, y la electrónica para aislar eléctricamente la placa de Arduino del dispositivo de RC, y la forma de integrar las comunicaciones que se necesitan para la correcta operación. La aplicación en Java cuenta con componentes visuales de tipo Swing, que son usadas para desarrollo rápido de aplicaciones, las cuales son muy flexibles para hacer cambios de diseño visual, sin mucho esfuerzo en cuanto al desarrollo y la codificación, es decir, se expone en forma guiada el desarrollo tanto hardware como software, con el objetivo de lograr una metodología clara de la integración de múltiples tecnologías que puede ser usada como base para aplicaciones más complejas.

Java, Arduino, RC, GUI, PCB

Abstract

This article proposes an application that can be used to control a radio control device (RC), interacting in real time using the Java language and Arduino. The code used for its implementation, the tracking of the hardware assembly, and the electronics to electrically isolate the Arduino board from the RC device, and the way to integrate the communications needed for the correct operation are presented in practical form. The application in Java has visual components of Swing type, which are used for rapid development of applications, which are very flexible to make visual design changes, without much effort in terms of development and coding, that is, it is exposed in Guided development of both hardware and software, with the aim of achieving a clear methodology of the integration of multiple technologies that can be used as a basis for more complex applications.

Java, Arduino, RC, GUI, PCB

Citación: ABRIL-GARCIA, José, MEZA-IBARRA, Iván, ALCÁNTAR-MARTÍNEZ, Adelina y LOPEZ-ROMO, Alonso. Aplicación para dispositivos RC con Java y Arduino. Revista de Tecnología Informática. 2018. 2-4: 11-15

* Correspondencia al autor (correo electrónico: abril@uthermosillo.edu.mx)
† Investigador contribuyendo como primer autor.

Introducción

Hoy en día existe una gran tendencia a usar dispositivos RC para vigilancia y exploración, tanto por el bajo costo, riesgo y fácil uso. La mayoría de los pequeños dispositivos RC como los que se usan en este proyecto se puede intervenir el control manual para ser controlados por una PC remotamente. Este proyecto permite a la PC controlar la placa de Arduino que a su vez se comunica con el control RC. Proponemos una aplicación que pueda controlar dispositivos RC, mediante una GUI en Java y que sea adaptativa, es decir con modificaciones mínimas pueda controlar dispositivos similares.

En la figura uno vemos el diagrama general del proyecto, en el cual en el vemos una aplicación en java que se conecta a una placa Arduino (Arduino, 2018) a través de un puerto serial USB, la cual se conecta, través de los GPIO's, a un circuito de opto acopladores y este al control remoto que envía las señales al carro de juguete.

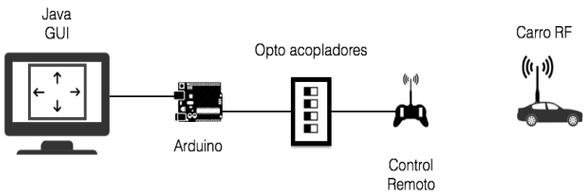

Figura 1 Diagrama general del proyecto.

Como resultado tenemos una aplicación que integra hardware y software que puede ser usada para cualquier dispositivo RC.

Marco Teórico

A continuación, describimos las herramientas utilizadas para el desarrollo del proyecto.

Java es un lenguaje muy popular de programación de propósito general, orientado a objetos, diseñado para tener las menores dependencias de implementación, y para ser ejecutada en cualquier dispositivo electrónico que tenga una máquina virtual Java (JVM). Con una sintaxis muy similar a la de C y C++.

Existen una serie de IDEs (Interface Development Environment) por sus siglas en inglés, para facilitar el desarrollo en Java. En nuestro caso usaremos el IDE NetBeans de Oracle, la misma compañía que adquirió Java.

NetBeans contiene un marco genérico para aplicaciones y componentes visuales Swing, que proporciona la estructura que, antes, cada desarrollador tenía que escribir por sí mismo.

Arduino IDE (Arduino, 2017), es un software de código abierto que hace que sea fácil escribir código y cargarlo en la placa electrónica de Arduino. Se ejecuta en Windows, Mac OS X y Linux. El entorno está escrito en Java y licencia de código abierto (Herger, 2015).

Especificaciones de Software:

- macOS Sierra Versión 10.12.5 (16F73)
- JRE build 1.8.0_101-b13
- JDK 1.8.0_101
- NetBeans IDE 8.2
- Arduino.cc 1.8.3
- Proteus 8 Release 8.3 SP1

Arduino es una plataforma electrónica muy popular, basada en hardware y software fáciles de usar. Las tarjetas Arduino son capaces de leer entradas digitales y análogas como: sensores de luz, de temperatura e interruptores, también soporta salidas, ya sea para activar un motor, encender un LED o comunicarse con otro dispositivo.

Los Opto-acopladores también llamado un opto-aislador o aislador óptico, es un componente que transfiere señales eléctricas entre dos circuitos aislados usando luz. Los opto-aisladores evitan que los altos voltajes afecten al sistema que recibe la señal.

El control de radio (a menudo abreviado a R/C o simplemente RC) es usado manualmente para enviar señales de radio para controlar algún dispositivo remotamente. El RC se utiliza para el control de vehículos modelo o a escala. Cuenta con un transmisor de radio con controles manuales. En nuestro caso el transmisor de RC estará intervenido para ser controlado por la placa Arduino y a su vez por la PC.

Especificaciones de Hardware:

- MacBook Pro 2.7 GHz Intel Core i5 8 GB 1867 MHz DDR·
- Placa RobotDyn Uno CH340/ATmega328PA
- Placa opto acopladores PCB

ISSN 2531-2197

- Radio control
- Carro RC

Desarrollo del proyecto

1. Diseño y programación de aplicación en NetBeans (Froufe, 2006) (Deitel, 2004). En esta etapa se desarrolló un GUI que simula el control en pantalla y permite mover al objeto con las flechas del teclado, se programaron todas las validaciones necesarias y las funciones para comunicarse con la placa Arduino a través del puerto USB, en la figura 2 podemos observar la pantalla principal de NetBeans.

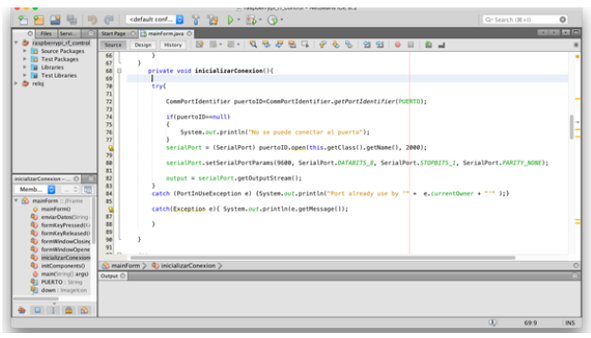

Figura 2 NetBeans IDE

2. Codificación del programa Arduino. En esta etapa se codificaron las funciones necesarias para comunicar a la GUI con la placa de opto-acopladores, se realizaron pruebas con el Monitor Serie, finalmente se descargó el código a la placa, en la figura 3 vemos el IDE de Arduino.

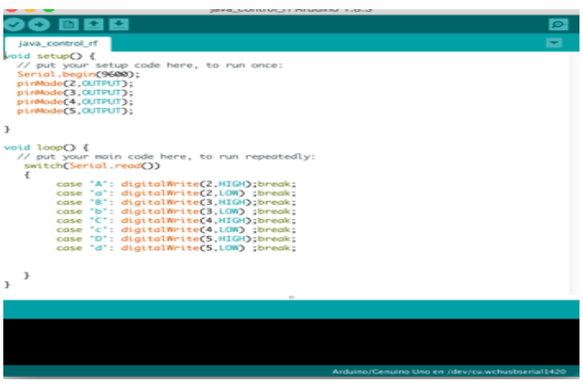

Figura 3 Arduino IDE

3. Integración de la aplicación en Java y Arduino. Aquí se probó la comunicación de la GUI con Arduino. Se utilizó una tableta de Leds (Figura 4) para verificar que la funcionalidad de la aplicación.

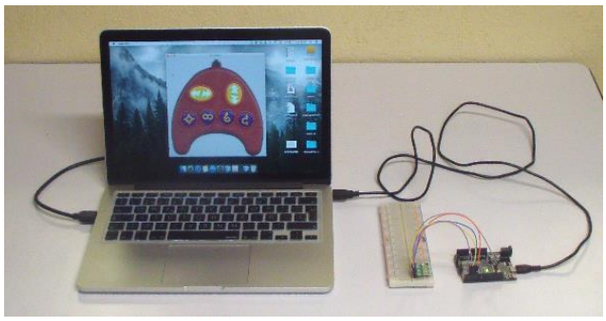

Figura 4 Java - Arduino y tableta de Leds

4. Diseño integración de placa con los opto-acopladores (Maloney, 2000). Debido a las diferencias de voltaje de los dispositivos Arduino (5 V) y control RC (9V), no es seguro conectarlos directamente por lo que se recurrió al diseño de una placa electrónica que mediante opto-acopladores permitiera una comunicación segura entre el Arduino y el RC, en la figura 5 vemos el diseño de la placa.

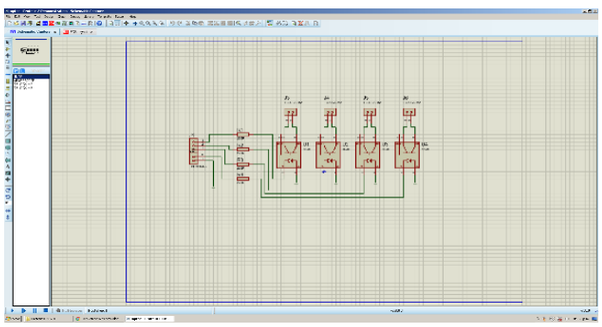

Figura 5 Diseño de la PCB

5. Integración final. Por último, se integró la aplicación Java, Arduino, la placa Opto y control RC. En esta etapa final del proyecto, se conectaron todos los dispositivos y se realizaron las pruebas necesarias para verificar el funcionamiento completo del sistema en la figura 6 vemos la versión final del proyecto operando.

Figura 6 Integración final

ABRIL-GARCIA, José, MEZA-IBARRA, Iván, ALCÁNTAR-MARTÍNEZ, Adelina y LOPEZ-ROMO, Alonso. Aplicación para dispositivos RC con Java y Arduino. Revista de Tecnología Informática. 2018. 2-4: 11-

Resultados y conclusiones

Se verificó el movimiento del carro RC de forma adecuada en tiempo real. En la figura 6, anteriormente descrita, se pueden observar los botones del control RC en pantalla, que representa el dispositivo manual real e interactúa con los dispositivos interconectados. Hemos realizado con éxito una integración de control de hardware manual (Control RC), con una interface gráfica que responde en tiempo real, y que puede ser operada con otro sistema de información que interactúe de manera remota. Es decir que, con unos pocos cambios y siguiendo esta metodología clara y sencilla, es posible realizar a futuro desarrollos de mayor complejidad, como una rutina de patrullaje incorporando un temporizador y una rutina automática. También de esta forma se pueden incorporar aplicaciones en Domótica, basadas en la integración del lenguaje de alto nivel y multiplataforma como lo es Java.

Apéndices

Código NetBeans

```
public class mainForm extends javax.swing.JFrame
{
     ImageIcon up =new ImageIcon(new
ImageIcon("images/up.png").getImage().getScaledI
nstance(60, 60, Image.SCALE_DEFAULT))
     JLabel n = new JLabel(new ImageIcon(new
ImageIcon("images/control2.png").getImage().getS
caledInstance(600, 600, Image.SCALE_DEFAULT)));

    private final String
PUERTO="/dev/cu.wchusbserial1420";

    SerialPort serialPort;

    private OutputStream output=null;

    public mainForm() {

        this.setContentPane(n);
        inicializarConexion();

        initComponents();

        jLabel1.setIcon(t);
        jLabel2.setIcon(t);
        jLabel3.setIcon(t);
        jLabel4.setIcon(t);

    }
   private void enviarDatos(String datos){
        try{
            output.write(datos.getBytes());
        } catch(IOException e){
            System.out.println("ERROR" + e);
private
        }
    }
 void inicializarConexion(){

        try{
            CommPortIdentifier
puertoID=CommPortIdentifier.getPortIdentifier(PU
ERTO);

            if(puertoID==null){
                System.out.println("No se puede
conectar al puerto");
            }
            serialPort = (SerialPort)
puertoID.open(this.getClass().getName(), 2000);

            serialPort.setSerialPortParams(9600,
SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);

            output =
serialPort.getOutputStream();
        }
        catch (PortInUseException e)
{System.out.println("Port already use by '" +
e.currentOwner + "'" );}

        catch(Exception e){
System.out.println(e.getMessage());

        }

    }private void
formKeyPressed(java.awt.event.KeyEvent evt) {

            switch(evt.getKeyCode()){
                case 38:
                    if(jLabel2.getText()==null){
                        jLabel1.setIcon(up);
                        enviarDatos("A");
                        jLabel1.setText("");
                    }
                    break;

                case 40:
                    if(jLabel1.getText()==null){
                        jLabel2.setIcon(down);
                        enviarDatos("D");
                        jLabel2.setText("");
                    }
                    break;

                case 39:
                    if(jLabel3.getText()==null){
                        jLabel4.setIcon(right);
                        enviarDatos("B");
                        jLabel4.setText("");
                    }
                    break;

                case 37:
                    if(jLabel4.getText()==null){
                        jLabel3.setIcon(left);
                        enviarDatos("C");
                        jLabel3.setText("");
                    }
                    break;
            }
    }

    private void
formWindowClosing(java.awt.event.WindowEvent
evt) { serialPort.close(); }

}
```

Código Arduino.

```
void setup() {
  Serial.begin(9600);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);

}
```

ISSN 2531-2197

```
void loop() {
   switch(Serial.read())
   {
        case 'A': digitalWrite(2,HIGH);break;
        case 'a': digitalWrite(2,LOW) ;break;
        case 'B': digitalWrite(3,HIGH);break;
        case 'b': digitalWrite(3,LOW) ;break;
        case 'C': digitalWrite(4,HIGH);break;
        case 'c': digitalWrite(4,LOW) ;break;
        case 'D': digitalWrite(5,HIGH);break;
        case 'd': digitalWrite(5,LOW) ;break;
   }
}
```

Bibliografía

Arduino. (20 de 12 de 2017). *Download the Arduino IDE*. Obtenido de Download the Arduino IDE: http://arduino.cc/en/Main/Software

Arduino. (16 de 03 de 2018). Obtenido de http://arduino.cc/en/Main/ArduinoBoardUno

Deitel, H. y. (2004). *Como programar el C/C++ y JAVA* (4 ed.). México: Pearson Education.

Froufe, A. (2006). *JAVA 2: Manual y tutorial de JAVA.* España: Ra-Ma Editorial.

Herger, L. &. (2015). Engaging students with open source technologies and Arduino. *IEEE Integrated STEM Education Conference*, 27-32.

Maloney, J. T. (2000). *Electrónica industrial Dispositivos y Sistemas.* México: Editorial Prentice Hall Hispanoamericana.

ISSN 2531-2197