ACTUALIZACIÓN DE BRAZO ROBÓTICO CRS F3 A ROBOT COLABORATIVO

José Guadalupe Zavala Villalpando

Tecnológico Nacional de México/Instituto Tecnológico de Celaya

jg.zavala@itcelaya.edu.mx

Luis Gerardo Esparza Díaz

Tecnológico Nacional de México/Instituto Tecnológico de Celaya

gerardo.esparza@itcelaya.edu.mx

José Manuel Martínez Aguilera

Tecnológico Nacional de México/Instituto Tecnológico de Celaya

11030712@itcelaya.edu.mx

Marco Antonio Martínez Alfaro

Tecnológico Nacional de México/Instituto Tecnológico de Celaya

11030544 @itcelaya.edu.mx

Resumen

En este trabajo se presenta la realización del control de un robot CRS F3 desde un dispositivo móvil en sus modos “world” y “joint”, por medio de bluetooth y un Arduino Mega para la transmisión de datos desde el celular hasta la computadora, donde un programa realizado en Visual Studio lee los datos y posteriormente los manda a un programa llamado ActiveRobot el cual finalmente mueve al robot.

Para poder realizar lo previamente mencionado, fue necesario reparar el brazo robótico CRS F3, ya que no se encontraba en operación. Para ello, se necesitó acudir a los manuales del robot y realizar diferentes pruebas eléctricas en él.

Palabra(s) Clave(s): Brazo robótico, Arduino, app inventor, relé.

Abstract

This paper presents the realization of the control of a CRS F3 robot from a mobile device in its "world" and "joint" modes, by means of bluetooth and an Arduino Mega for the transmission of data from the cell phone to the computer, where a program made in Visual Studio reads the data and then sends it to a program called ActiveRobot which finally moves the robot.

In order to carry out the aforementioned, it was necessary to repair the CRS F3 robotic arm, since it was not in operation. For this, it was necessary to go to the manuals of the robot and perform different electrical tests on it.

Keywords: Robotic arm, Arduino, app inventor, relay.

1. Introducción

Un brazo robótico es un brazo mecánico y programable, que tiene funciones parecidas a las de un brazo humano. Las partes están interconectadas a través de articulaciones que permiten movimientos rotacionales y prismáticos (Wikipedia, 2016).

Actualmente el diseño de brazos robóticos en la industria ha avanzado radicalmente debido al constante deseo de innovar y automatizar los procesos industriales. Las empresas están invirtiendo constantemente en la investigación y desarrollo para optimizarlos. Es por ello que las instituciones educativas reconocen la importancia de que los alumnos generen las habilidades para controlarlos partiendo del conocimiento de su modo de funcionamiento. Además de generar nuevas técnicas innovadoras para el control.

2. Desarrollo

El proyecto se encuentra dividido en tres secciones:

- Características del brazo CRS F3: Características físicas, eléctricas y de sus elementos.
- Reparación del brazo robótico CRS F3: Implementación de acciones sugeridas por los manuales y realización de pruebas eléctricas. También,

información básica con respecto al programa Robcomm 3 utilizado para monitorear el robot.

- Creación de la aplicación para mover el robot: Consiste en toda la programación en los lenguajes de Arduino, Visual Studio y app Inventor, así como una breve explicación de cada uno.

Características del brazo CRS F3

En su configuración más básica, el sistema del robot CRS F3 consiste en el brazo robótico CRS F3, el controlador C500C y un cable umbilical que provee la energía y la comunicación al brazo (figura 1).

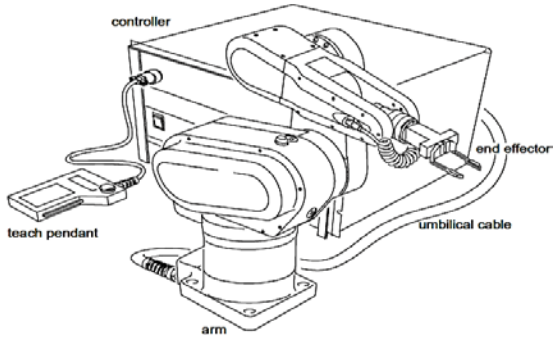

Figura 1 Configuración básica del robot.

El brazo (figura 2), transporta cargas y desempeña otras tareas de movimiento en su espacio de trabajo. Una placa de montado en su base asegura al brazo a una plataforma fija.

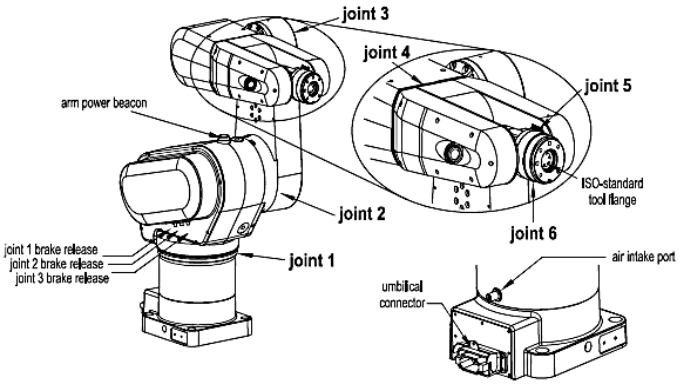

Figura 2 Brazo robótico.

El controlador C500C (figura 3), provee circuitos de seguridad, poder y control de movimiento para el brazo. Éste maneja los motores de cada articulación, calcula trayectorias y almacena en la memoria aplicaciones del robot. También detecta condiciones peligrosas como colisiones severas, sobrecalentamiento o sobrecorriente y errores de comunicación. Si una de estas condiciones es detectada, el controlador de inmediato activa un paro de emergencia o se apaga (Thermo, 2004).

Las características del brazo robótico, las especificaciones eléctricas y de la consola C500C se muestran en tabla 1, tabla 2 y tabla 3, respectivamente.

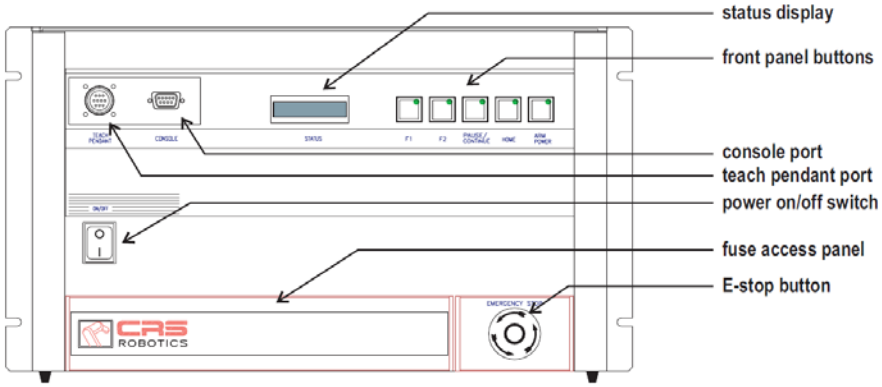

Figura 3 Controlador C500C.

Tabla 1 Especificaciones del brazo robótico.

Numero de ejes	6
Peso	52 kg [115 lb]
Carga nominal	3 kg [6.6 lb]
Velocidad lineal máxima	4 m/s
Sistema	Electromecánico, motores sin escobillas Encoders absolutos en cada articulación.
Frenos	Frenos en articulaciones 1, 2 y 3.
Modos de movimiento	Teach Automático ISO-9409

Tabla 2 Especificaciones eléctricas.

Voltaje de entrada AC	100/115/230 VAC ± 10%
Frecuencia	50-60 Hz
Consumo de potencia	1000 W

Tabla 3 Características del controlador C500C.

Microprocesador dual	133 MHz i486DX (procesador del sistema) 60 MHz TMS320C31 DSP (control de movimiento)
Memoria	4 MB RAM memoria de usuario. 512 kB NVRAM para almacenar aplicaciones 1 MB memoria flash para el firmware.
Usuario I/O	16 entradas digitales 12 salidas digitales 1 entrada analógica 4 salidas a relé.
Dimensiones	482.6 mm [19 pulg] x 266.7 mm [10.5 pulg]
Peso	31 kg [68 lb]

Reparación del brazo robótico

El brazo robótico cuenta con un programa llamado Robcomm 3 (figura 4), el cual sirve para poder saber que errores tiene el sistema mediante una ventana de terminal y comandos previamente establecidos.

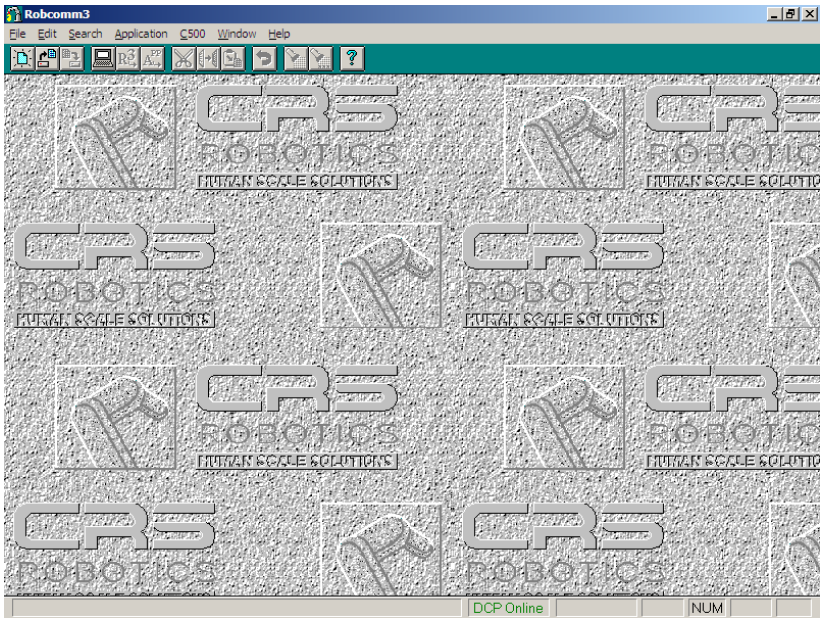

Figura 4 Robcomm 3.

Revisando los manuales del robot CRS F3 y el programa del Robcomm, se realizó la puesta en marcha, la cual consiste en un conjunto de procesos para calibrar el brazo robótico con la utilización de comandos escritos en la terminal del Robcomm. Pero a pesar de la realización de esto, no se pudo rehabilitar el robot, debido a que, en una parte del manual, se encontraba una advertencia de si al

activar el botón de *arm power* no prendía la luz del faro del brazo robótico, implicando que era un error que no tenía que ver con la calibración.

Entonces se iniciaron pruebas de continuidad sobre el cable umbilical que se conecta entre la consola y el brazo robótico. En las siguientes ilustraciones se muestra cómo es el tipo de conector (figura 5) y la configuración de pines del mismo (tabla 4).

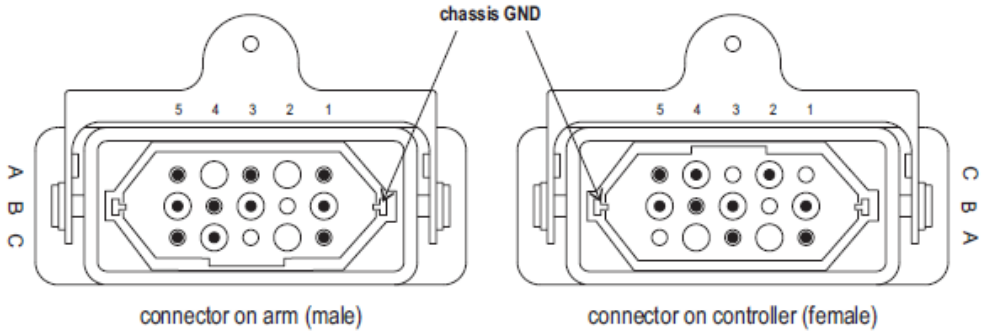

Figura 5 Cable umbilical.

Tabla 4 Configuración de pines del cable umbilical.

Pin	Signal Name	Signature	Description
A1	Return GND	10A	Signal and power ground return
A2	--	--	--
A3	Motor Power	+77 VDC, 10A	Power supply for amplifiers in the arm
A4	--	--	--
A5	Battery Test	0 to 4.5 VDC	Use to check battery voltage level
B1	TX+ to arm	RS-485 0 to 5 VDC, 50mA	Serial communication to the arm
B2	--	--	--
B3	BRAKE_V	24 VDC, 1A	Power to disengage brakes (only enables when arm power is enabled)
B4	OPT_PWR	+24 VDC, 1A	Power supply for end-of-arm options and brake release
B5	SYS_PWR	+12 VDC, 3A	Power supply for amplifiers and support electronics
C1	TX- to Robot	RS-485 0 to 5 VDC, 50mA	Serial communication to the arm
C2	--	--	--
C3	--	--	--
C4	RX+ from arm	RS-485 0 to 5 VDC, 50mA	Communication line from the arm
C5	RX+ from arm	RS-485 0 to 5 VDC, 50mA	Communication line from the arm

Tras hacer estas pruebas, se encontraron cinco leds indicadores de voltaje (figura 6), situados en la parte interior donde va conectado el cable umbilical al motor (figura 7). De los leds dos no prendían, uno es el que indicaba que estaban presentes los 77 V de alimentación del robot y el otro es un voltaje de Brake que sirve para desacoplar los frenos del motor. Las siguientes ilustraciones muestran los leds indicadores.

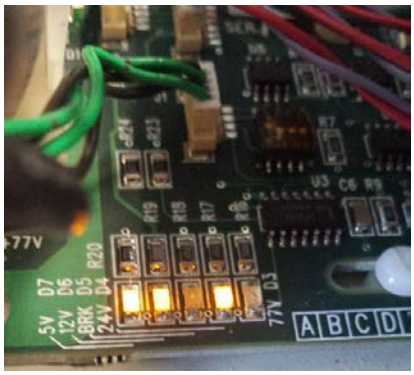

Figura 6 Leds indicadores apagados.

Figura 7 Localización de los leds indicadores.

Para comprobar que lo que indicaban los leds era cierto, se hicieron mediciones eléctricas del robot para observar el voltaje encontrado entre estos pines. En las siguientes ilustraciones, se muestran las mediciones del pin de los 77 V, pero uno en un robot que funciona (figura 8) y un robot que se encontraba deshabilitado (figura 9).

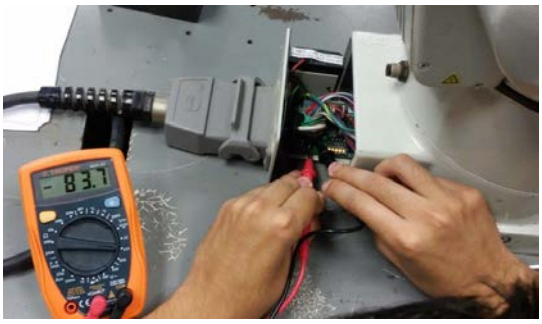

Figura 8 Medición de voltaje en robot funcional.

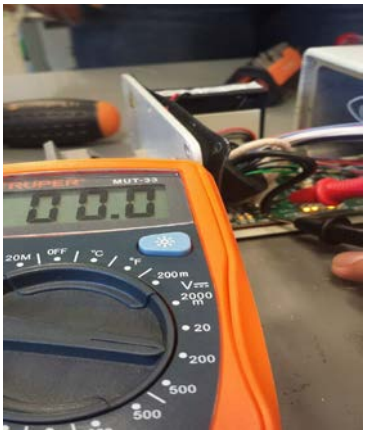

Figura 9 Medición de voltaje en el robot descompuesto.

Tras esta prueba, se observó que sí deben estar presentes los 77 V o algo aproximado. Así que, para continuar, se analizó la placa de potencia del brazo robótico (figura 10).

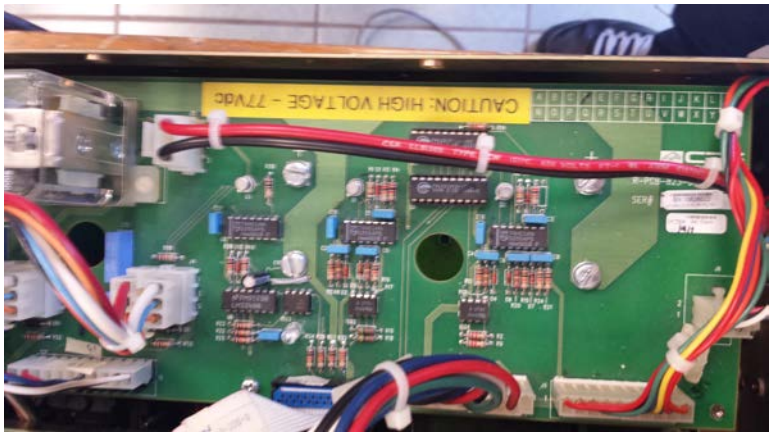

Figura 10 Placa de potencia del brazo robótico.

Se checaron los elementos del circuito de potencia al analizar las líneas de conducción de la placa, encontrando que constaban de:

- Un puente de diodos: Que pasa de AC a DC.
- 2 Fusibles: Para la protección del circuito
- 3 capacitores: Para la conversión de AC a DC y mantener el voltaje.
- 3 Relevadores: Activados y desactivados por los paros de emergencia y por el botón de arm power.

Sabiendo esto, se procedió a realizar mediciones eléctricas del circuito observándose que uno de los relevadores no se estaba activando porque no pasaba el voltaje para llevarlo a cabo. Entonces se desmontó la placa de potencia de la consola llegando a la parte donde se encontraba el relé inactivo (figura 11).

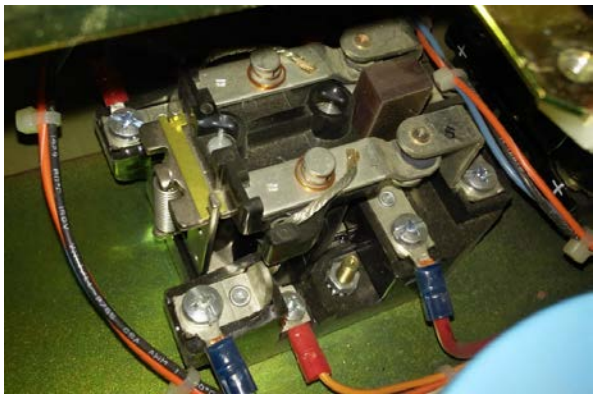

Figura 11 Relevador descompuesto.

Se empezaron a realizar pruebas con una fuente de voltaje externa al robot para ver si se activaba el relé, pero siguió sin funcionar. Entonces se pudo observar que las placas donde hace contacto el relé estaban “atoradas” entre ellas o incluso unidas. Lo que se hizo fue darle unos golpecitos para desatorarlas y, una vez sueltas, se procedió a conectar todo el robot. Esta vez funcionó adecuadamente y se volvieron a checar los voltajes en la línea de potencia, observándose que esta vez sí daban los resultados adecuados e incluso el robot ya había empezado a funcionar.

Este problema se le puede atribuir a dos causas:

- Como el robot es viejo, es probable que sea parte del desgaste mecánico presentado en el relé.
- Hubo una corriente que pudo hacer que las placas de unión de los relés se calentaran al grado de unirse térmicamente entre ellas.

Para poder solucionar la segunda causa, se propuso al encargado de laboratorio que se compraran reguladores de voltaje para cada robot para evitar los picos de corriente.

Control de movimiento del robot

Esta parte del proyecto se centra en la investigación, programación e implementación de los programas y componentes electrónicos que fueron necesarios para lograr la comunicación humano-computadora-robot. Para explicar la totalidad del procedimiento que se siguió, este se dividirá en las siguientes secciones: hardware, donde mencionamos los componentes electrónicos utilizados; software, que engloba todos los programas que fueron requeridos y programación muestra cada uno de los códigos implementados.

Hardware

Arduino Mega 2560. Placa para aplicaciones electrónicas que contiene un microcontrolador ATmega2560 (figura 12) con la función de adquirir las señales provenientes desde la aplicación móvil y transmitirlas hacia la computadora.

Figura 12 Placa Arduino Mega2560.

Las características principales de este dispositivo son:

- Microcontrolador ATmega2560.
- Voltaje de entrada de 7 a 12 V.
- 54 pines digitales de Entrada/Salida (14 de ellos son salidas PWM).
- 16 entradas análogas.
- 256 k de memoria flash.
- Velocidad del procesamiento de 16 Mhz.
- Posibilidad de usar uno o más puertos seriales (max 4).

Módulo HC-05. Debido a la simplicidad de su conexión el módulo bluetooth HC-05 (figura 13) nos permite establecer comunicación de manera muy sencilla y nos sirve como comunicador entre la placa arduino y el dispositivo móvil a usar.

Figura 13 Modulo Bluetooth HC-05.

Características:

- Alcance máximo de señal: 10 m.
- Velocidad de procesamiento: Síncrono 1 Mbps, Asíncrono 2.1 Mbps (Max).
- Voltaje de alimentación de 3.3–5 V

Software

Sistema Operativo. Windows XP es una versión de Microsoft Windows la cual nos sirvió como base para soportar los programas necesarios dentro de un CPU debido a que este ofrece la compatibilidad necesaria con cada uno de los softwares utilizados para la comunicación con el robot.

Active Robot. Este programa permite la creación de aplicaciones para el total acceso y control del sistema de un robot CRS F3 mediante una computadora (figura 14).

Figura 14 Ventana de inicio de Active Robot.

Este programa presenta las siguientes características:

- Programación de interfaces compiladas con el modelo de objetos componente de Microsoft.
- Compatibilidad con un gran rango de herramientas de desarrollo ActiveX como Microsoft Visual Basic 6 y Visual Basic C++ 6 así como versiones posteriores, y compatible con aplicaciones como Microsoft Access 97/2000 y Labview de National Instruments.
- Acceso al control de todas las capacidades de la consola C500C tales como movimiento, capacidades de control y módulos de entradas y salidas.
- Aplicaciones de ejemplo detalladas en Visual Basic y Visual C++.

Visual Studio 2005. Es un entorno de desarrollo integrado para sistemas operativos Windows, soporta múltiples lenguajes tales como C++, C#, Visual Basic.Net entre otros, unas las características más importantes para el proyecto es la capacidad de comunicación con las librerías proporcionadas por el Active Robot, la compatibilidad con el sistema operativo y la más importante la capacidad de implementar hilos dentro de su código para poder realizar procesos simultáneos durante la ejecución esto con el fin de que la comunicación entre el dispositivo móvil y la computadora se realice lo más rápido posible (figura 15).

Figura 15 Logo Visual Studio.

IDE Arduino. Entorno de desarrollo que maneja un lenguaje C con la versatilidad para realizar aplicaciones electrónicas de forma sencilla y con una gran variedad de posibilidades (figura 16). Principal mente se utilizó para establecer un medio de comunicación entre la aplicación móvil y Visual Basic.

Figura 16 Logo de Arduino.

MIT APP Inventor. Software libre en línea que permite el desarrollo de aplicaciones para dispositivos móviles a manera de las necesidades de cada programador, ofrece emuladores en tiempo real y la utilización de varios componentes presentes en los teléfonos más actuales tales como el uso de módulos bluetooth (figura 17).

Figura 17 Logo de MIT APP inventor.

Programación

A continuación, se muestra cada uno de las secuencias de programación usadas dentro de cada software para la comunicación entre el dispositivo móvil y el Robot CRS:

- **Comunicación App Inventor–Modulo HC-05–Arduino.** Para la comunicación entre estos componentes se diseñó una interfaz gráfica dentro de APP Inventor (figura 18) que fuera acorde a los parámetros con lo que se puede manejar el robot y el diseño final.

Figura 18 Aplicación desarrollada en APP Inventor.

- El primer paso en la programación fue poder mandar datos desde el dispositivo móvil hacia el modulo bluetooth, para que esto fuera posible ambos componentes debían estar previamente vinculados. Dentro de app inventor los bloques utilizados para la comunicación (figura 19) nos permiten que antes de ser presionado el botón **conectar dispositivo** éste ya debe almacenar los nombres de los aparatos que han sido vinculados al celular o tablet, y después de presionarse mande una solicitud de comunicación con el que hayamos seleccionado.

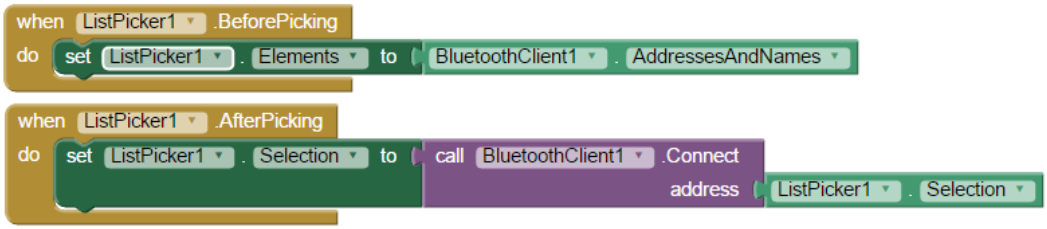

Figura 19 Código de programación para solicitud de comunicación en APP Inventor.

Una vez que está conectado, el siguiente paso para el móvil es transmitir datos que puedan ser recibidos por el módulo HC-05 e interpretado por el Arduino, desde la app el procedimiento usado para mandar datos es el siguiente, figura 20.

En la figura 20 se observa la programación realizada para el botón J1P el cual maneja tanto la rotación positiva del joint 1 como el movimiento en X dependiendo de las siguientes condiciones:

- ✓ Si el botón de paro se encuentra activado.
- ✓ Y cuál modo de movimiento ha sido seleccionado.
- ✓

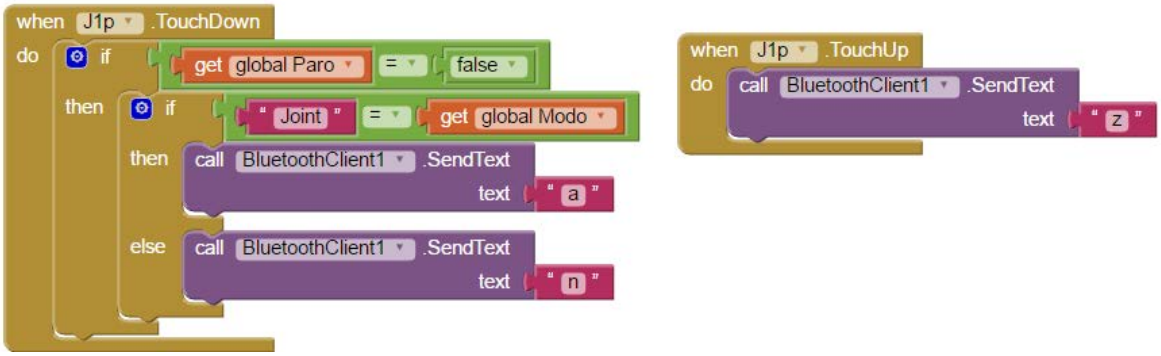

Figura 20 Procedimiento de envió de datos.

Desde la aplicación lo que se busca es el envió de datos sencillos los cuales no tengan problemas para su identificación y manejen un procedimiento único tanto para el Arduino como para Visual Basic. En este caso, observamos que en modo joint (default) se manda un carácter “a” que permite la rotación positiva en el motor número 1 y en modo world transmite una “n” que activa el movimiento en el eje X en dirección positiva. A

continuación, se muestra cada uno de los caracteres (tabla 5) que maneja la aplicación para activar los diferentes joints o movimientos en los ejes coordenados.

Tabla 5 Caracteres Enviados desde la aplicación.

Botón	Joint	World	Botón	Joint	World
J1+	a	n	J4+	g	t
J1-	b	o	J4-	h	u
J2+	c	p	J5+	i	v
J2-	d	q	J5-	j	w
J3+	e	r	J6+	k	x
J3-	f	s	J6-	l	y

Como se muestra anteriormente, el carácter m no se encuentra debido a que este está reservado para el botón de *home* el cual inicia una secuencia en el brazo para llevarlo a su posición de seguridad. Usando el mismo procedimiento, modificamos parámetros tanto de la velocidad del robot como el avance por pulso que tiene.

- **Conexión Arduino–HC-05.** El diagrama de conexión para estos dos componentes es el estándar para este tipo de dispositivos (figura 21), la única variante es que se conectó en el puerto serial 1 del Arduino Mega.

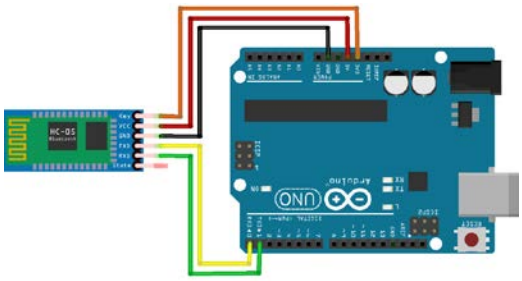

Figura 21 Conexión Arduino y Modulo HC-05.

El código implementado dentro del Arduino (figura 22) funciona simplemente como un puente entre el dispositivo móvil y la computadora puesto que el carácter que recibe del celular o tablet es retransmitido hacia el puerto serial de la computadora.

```
void loop()
{
  byte entrada = Serial1.read();
  byte salida ="a";
  switch (entrada)
  {
  case '1':
  Serial.write("1");
  break;
case 'a':
while (salida != 'z')
{
Serial.write("a");
delay (50);
salida = Serial1.read();
}
break;
```

Figura 22 Código de Arduino para retransmisión de señales hacia la computadora.

Lo que se observa en la figura a es la parte donde comienza el ciclo para la detección de caracteres, donde el caso número 1 representa un procedimiento que el Arduino le dará a la computadora para variar la velocidad con la cual se va a mover el robot. En la parte b se muestra el código básico que utiliza el Arduino para comunicarle a la computadora que ejecute un movimiento del robot definido por el carácter sea (tabla 5).

Código Visual Basic. A continuación, se muestra el código implementado dentro del programa Visual Basic para el control del Robot CRS:

```
Imports System.IO.Ports
Imports System.Threading
Imports ACTIVEROBOTLib
Public Class Form1
   Private Recibidos As String
   Dim RobotObject As New CRSRobot
   Private AvanceG As Integer
   Private AvanceM As Integer
   Public Sub New()
      InitializeComponent()
// Procedimiemto para iniciar puerto serial
      If Not SerialPort1.IsOpen Then
         Try
            SerialPort1.Open()
            AvanceG = 15
            AvanceM = 10
            RobotObject.Speed = 10
         Catch ex As Exception
            MessageBox.Show(ex.ToString)
         End Try
```

```
            AddHandler SerialPort1.DataReceived, AddressOf Recepcion
        End If
    End Sub
    Private Sub Recepcion(ByVal sender As Object, ByVal e As
SerialDataReceivedEventArgs)
//recepcion de datos
        Recibidos += SerialPort1.ReadExisting()
        Invoke(New EventHandler(AddressOf Actualizar))
    End Sub
    Private Sub Actualizar(ByVal sender As Object, ByVal e As EventArgs)
//Condicion de espera para que no tome otra instrucción
//hasta que no termine la anterior
        If RobotObject.Finished Then
            Select Case Recibidos
// Seleccion de casos para comandos del robot
                Case "1"
                    RobotObject.Speed = 1
                    Recibidos = ""
                    Exit Select
                Case "2"
                    'RobotObject.AxisNegativeLimit(1) = 30
                    RobotObject.Speed = 5
                    Recibidos = ""
                    Exit Select
                Case "3"
                    RobotObject.Speed = 10
                    Recibidos = ""
                    Exit Select
                Case "4"
                    RobotObject.Speed = 25
                    Recibidos = ""
                    Exit Select
                Case "A"
                    AvanceG = 1
                    Recibidos = ""
                    Exit Select
                Case "B"
                    AvanceG = 5
                    Recibidos = ""
                    Exit Select
                Case "C"
                    AvanceG = 15
                    Recibidos = ""
                    Exit Select
```

```
Case "D"
   AvanceG = 30
   Recibidos = ""
   Exit Select
Case "E"
   AvanceM = 1
   Recibidos = ""
   Exit Select
Case "F"
   AvanceM = 5
   Recibidos = ""
   Exit Select
Case "G"
   AvanceM = 10
   Recibidos = ""
   Exit Select
Case "H"
   AvanceM = 25
   Recibidos = ""
   Exit Select
Case "I"
   AvanceM = 50
   Recibidos = ""
   Exit Select
Case "J"
   AvanceM = 100
   Recibidos = ""
   Exit Select
Case "a"
   Panel1.BackColor = Color.Green
   RobotObject.Joint(1, AvanceG)
   Recibidos = ""
   Exit Select
Case "b"
   Panel1.BackColor = Color.Red
   RobotObject.Joint(1, -AvanceG)
   Recibidos = ""
   Exit Select
Case "c"
   RobotObject.Joint(2, AvanceG)
   Recibidos = ""
   Exit Select
Case "d"
   RobotObject.Joint(2, -AvanceG)
```

```
    Recibidos = ""
    Exit Select
Case "e"
    RobotObject.Joint(3, AvanceG)
    RobotObject.Speed = 25
    Recibidos = ""
    Exit Select
Case "f"
    RobotObject.Joint(3, -AvanceG)
    Recibidos = ""
    Exit Select
Case "g"
    RobotObject.Joint(4, AvanceG)
    Recibidos = ""
    Exit Select
Case "h"
    RobotObject.Joint(4, -AvanceG)
    Recibidos = ""
    Exit Select
Case "i"
    RobotObject.Joint(5, AvanceG)
    Recibidos = ""
    Exit Select
Case "j"
    RobotObject.Joint(5, -AvanceG)
    Recibidos = ""
    Exit Select
Case "k"
    RobotObject.Joint(6, AvanceG)
    Recibidos = ""
    Exit Select
Case "l"
    RobotObject.Joint(6, -AvanceG)
    Recibidos = ""
    Exit Select
Case "m"
    RobotObject.Ready()
    Recibidos = ""
Case "n"
    RobotObject.JogWorld(1, AvanceM)
    Recibidos = ""
    Exit Select
Case "o"
    RobotObject.Speed = 25
```

```
        RobotObject.JogWorld(1, -AvanceM)
        Recibidos = ""
        Exit Select
    Case "p"
        RobotObject.JogWorld(2, AvanceM)
        Recibidos = ""
        Exit Select
    Case "q"
        RobotObject.JogWorld(2, -AvanceM)
        Recibidos = ""
        Exit Select
    Case "r"
        RobotObject.JogWorld(3, AvanceM)
        Recibidos = ""
        Exit Select
    Case "s"
        RobotObject.JogWorld(3, -AvanceM)
        Recibidos = ""
        Exit Select
    Case "t"
        RobotObject.JogWorld(6, AvanceM)
        Recibidos = ""
        Exit Select
    Case "u"
        RobotObject.JogWorld(6, -AvanceM)
        Recibidos = ""
        Exit Select
    Case "v"
        RobotObject.JogWorld(5, AvanceM)
        Recibidos = ""
        Exit Select
    Case "w"
        RobotObject.JogWorld(5, -AvanceM)
        Recibidos = ""
        Exit Select
    Case "x"
        RobotObject.JogWorld(4, AvanceM)
        Recibidos = ""
        Exit Select
    Case "y"
        RobotObject.JogWorld(4, -AvanceM)
        Recibidos = ""
        Exit Select
End Select
```

```
        Else
            Recibidos = ""
        End If
    End Sub
    Private Sub PictureBox1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    End Sub
    Private Sub Panel1_Paint(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.PaintEventArgs) Handles Panel1.Paint
    End Sub
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    End Sub
End Class
```

Todo el código está estructurado de tal manera que según el comando que reciba dentro desde el arduino active un caso para modificar variables como la velocidad, o ejecute una secuencia de movimiento dentro del robot. Acorde a esto tenemos también una pantalla que nos indica cuando el programa ha comenzado a funcionar (figura 23).

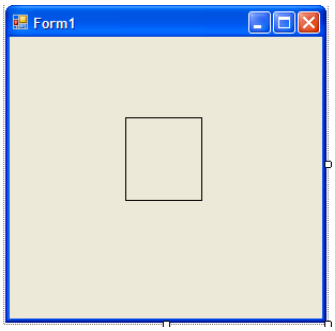

Figura 23 Pantalla de inicio de la aplicación en Visual Basic.

3. Resultados

Los resultados obtenidos durante el proyecto son los siguientes:

- Reparación y mantenimiento del al menos un robot de los dos que se encontraban inhabilitados.
- Establecimiento de comunicación entre un dispositivo móvil y el robot CRS.

- Control del robot CRS F3 mediante un dispositivo móvil.
- Realización de una aplicación móvil para el control remoto del robot CRS.

4. Discusión

Debido a que el proyecto fue desde un inicio demasiado ambicioso se buscó llegar al mayor desarrollo posible y ahora se le considera a este como exitoso.

Se logró crear una aplicación a base de software libre con la cual se puede manejar el Robot CRS.

Se había logrado reestablecer la funcionalidad de al menos un robot, sin embargo, por motivos desconocidos, uno comenzó a presentar fallas de nuevo.

El desempeño durante el proyecto de los dos integrantes del equipo fue suficiente para llevar a cabo las tareas designadas.

Se elaborarán manuales de uso para los robots y la aplicación, así como un apartado donde vengan detección y soluciones de fallas presentadas en la consola durante la realización del proyecto.

5. Bibliografía y Referencias

[1] Acaymo, A. (2015). Lectura de entradas digitales con Arduino y Visual Studio.

[2] Botello Mancera, C. (2016). Manual de deteccion de fallas y solucion en los robots CRS F3. Celaya.

[3] CRS. (2002). ActiveRobot User Guide.

[4] Definicion de Brazo robótico. (2016 de 09 de 2016). https://es.wikipedia.org /wiki/Brazo_rob%C3%B3tico.

[5] Network, M. D. (2014). SerialPort ReadLine Method. https://msdn.microsoft .com/en-us/library/system.io.ports.serialport.readline(v=vs.110).aspx?cs-save -lang =1&cs-lang=vb#code-snippet-1.

[6] Thermo. (2004). CRS F3 Robot System User Guide.

[7] Zapata., I. M. (26 de 04 de 2014). Barrido de Leds Arduino-VisualBasic 6.0. https://www.youtube.com/watch?v=C_rXP-1rWyg.