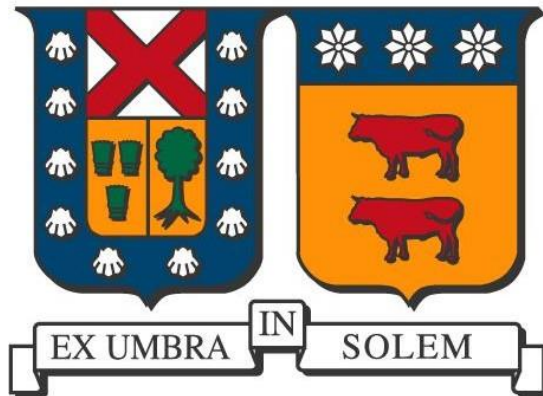


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
SEDE DE CONCEPCIÓN - REY BALDUINO DE BÉLGICA



**MANIPULACIÓN REMOTA DE BRAZO ROBÓTICO A TRAVÉS DE
MICROCONTROLADOR**

Tesis para optar al Título de Técnico
Universitario en

Automatización y control

Alumnos: Ignacio Medina

Diego Ruz

Profesor Guía: Helmut Contreras

2019

RESUMEN

En el presente trabajo de título se plantea la actualización del control de un brazo robótico ubicado en el laboratorio de Automatización de la Universidad Federico Santa María sede concepción. El brazo a intervenir es un Mitsubishi EX RV-M1, actualmente dicho brazo robótico ya cuenta con su control remoto local pero el mejoramiento tiene como objetivo principal diseñar un software y hardware para lograr el control del brazo a través de un microcontrolador (Arduino), generalmente usado por el alumnado de la Universidad, usando su propio lenguaje de comunicación, además, esto será comunicado remotamente a través de un módulo bluetooth hacia un dispositivo Android sin necesidad de cableado entre android y microcontrolador.

Desde la base del estudio del brazo, tanto de sus componentes, como su funcionamiento básico y su control, se lograrán enviar datos seriales a través de la comunicación rs-232 que posee el controlador del robot. Por lo tanto, en síntesis, lo que se logrará será enviar datos desde el celular, para que el microcontrolador lea el dato enviado por Android y lo reenvíe hacia el controlador del robot, donde este lo analizara, lo interpretará y ordenará al robot a operar.

INDICE DE MATERIAS

RESUMEN	2
INTRODUCCIÓN.....	6
CAPÍTULO 1. ROBÓTICA INDUSTRIAL	9
1.2 Clasificación de los tipos de robots industriales	9
1.3 Los elementos que forman parte de la totalidad del robot son:	10
1.4 Configuración de movimiento del robot	11
CAPÍTULO 2. ROBOT MELFA RV-M1	13
2.1 Características generales.....	13
2.2 Características del movimiento del robot	14
2.3 Controlador del brazo.....	15
2.4 Switch's de configuración	16
CAPÍTULO 3. COMUNICACIÓN RS-232	21
3.1 Especificaciones eléctricas de la interface.....	22
3.2 Tipo de Datos a enviar ASCII	23
3.3 Comunicación Serie	24
3.4 Comunicación RS-232 Arduino-Brazo	26
3.4.1 Puerto RS-232 brazo	27
3.4.2 Puerto RS-232 Arduino	29
3.5 Señales y función de cada conector DB-9 Y DB-25.....	30
3.6 Cable de comunicación Brazo-Arduino.....	31

CAPÍTULO 4. PROGRAMACIÓN	30
4.1 Programación arduino	32
4.1.1 interfaz programacion arduino	33
4.1.2 Programa de arduino	34
4.2 Programación módulo bluetooth	36
4.2.1 Bluetooth.....	36
4.2.2 Configuración bluetooth	37
4.2.3 Programa para configurar Bluetooth HC-05	38
4.2.4 Comandos AT para módulo HC-05	39
4.2.5 Características que ofrece este protocolo:	39
4.3 aplicación Móvil.....	40
4.3.1 scratch	40
4.3.2 APP inventor	41
4.2.3 Creación app para Android.....	42
5. ESQUEMA DE MONTAJE	45
5.1 Esquema de conexionado.....	45
6. CONCLUSIONES.....	46

INDICE DE FIGURAS

Capítulo 1: ROBOTICA INDUSTRIAL

Figura 1.1 Simulación Brazo- Robot	9
Figura 1.2 Elementos del brazo	10
Figura 1.3 Configuración Cartesiana con su volumen de trabajo.....	11
Figura 1.4 Configuración cilíndrica con su volumen de trabajo.....	11
Figura 1.5 Configuración Polar con su volumen de trabajo.....	12
Figura 1.6 Configuración angular con su volumen de trabajo.....	12

Capítulo 2: ROBOT MELFA RV-M1

Figura 2.1 Movimientos del brazo con sus respectivos nombres.....	13
Figura 2.2 Dimensiones y volumen de trabajo.....	14
Figura 2.3 Controlador del brazo.....	14
Figura 2.4 Parte posterior y lateral del controlador.....	15
Figura 2.5 Switch de configuración.....	16
Figura 2.6 Explicación Bit Switch WS2.....	19

Capítulo 3: COMUNICACIÓN RS-232

Figura 3.1 Conector RS-232(DB-9 hembra).....	21
Figura 3.2 Niveles de tensión RS-232.....	22
Figura 3.3 Caracteres ASCII.....	23
Figura 3.4 Señales de comunicación serie.....	24
Figura 3.5 Conexión del convertidor TTL- RS-232.....	26
Figura 3.6 Numerología de pines DB-25.....	27
Figura 3.7 Convertidor de señal.....	29
Figura 3.8 Conector DB-9 hembra.....	29
Figura 3.9 Cable de DB-25 a DB-9.....	31

Capítulo 4: PROGRAMACION

Figura 4.1 Interfaz programación arduino.....	32
Figura 4.2 Modulo bluetooth.....	32
Figura 4.3 Esquema conexionado bluetooth-arduino.....	36
Figura 4.4 Ejemplo Interfaz scratch.....	39
Figura 4.5 Ejemplo interfaz app inventor.....	40

INDICE DE TABLAS

Tabla 1 Interruptor ST1 y ST2.....	16
Tabla 2 Explicación bits Switch SW1.....	18
Tabla 3 Definición de posición perillas switch SW1.....	18
Tabla 4 Definición de posición perillas switch SW2.....	19
Tabla 5 Explicación switch SW3.....	20
Tabla 6 Definición de posición perillas switch SW3.....	20
Tabla 7 Asignación de pines DB-25 BRAZO.....	27
Tabla 8 Asignación pines de DB-25 a DB-9.....	30

PLANTEAMIENTO DEL PROBLEMA

En el taller de automatización de la universidad Federico santa maría sede concepción se encuentra un brazo robótico Melfa RV-M1, el cual está prácticamente sin uso, sin intervención y/o trabajo del alumnado de la misma universidad, con aplicaciones básicamente ajustadas mediante su control local. El proyecto llega buscando una finalidad académica en directa relación con los estudiantes de la especialidad de automatización, ya que se quiere comunicar este brazo con uno de los microcontroladores más utilizados por el estudiantado (Arduino), y así darle un uso académico donde los estudiantes puedan manipular este brazo de una forma más amigable y familiar respecto a sus conocimientos.

OBJETIVO GENERAL

El objetivo general del proyecto es lograr comunicar el brazo remotamente a través de Android en conjunto con un microcontrolador Arduino Mega, con esto, enviarle datos a través del celular al microcontrolador y este mande órdenes hacia el brazo, donde el brazo analizará y ejecutará.

OBJETIVOS ESPECIFICOS

- Comprender y estudiar el brazo Melfa RV-M1 de Mitsubishi.
- Comprender las capacidades del puerto RS-232 presente en el brazo y la forma en que interactúa con otros dispositivos.
- Desarrollar la comunicación microcontrolador-brazo.
- Desarrollar aplicación para manipular el brazo.

ALCANCE

El alcance se define como un anteproyecto, ya que se realizará el estudio, tanto a nivel de brazo, Arduino y comunicaciones, donde se dejarán datos de hardware y software con todo lo necesario para una posterior implementación.

INTRODUCCIÓN

A lo largo de la historia del ser humano, el hombre ha venido creando herramientas y artefactos que lo ayudan a realizar tareas que tal vez para el solo resulte complicado y/o peligroso; y así se llegó a grandes inventos de máquinas que tienen varias ventajas entre ellas: maximizar costos, reducir riesgos para obreros, disminuir tiempos de operación, producción en serie, etc.

Por lo mismo con el avance de las tecnologías hoy en día hay tareas que se pueden sustituir fácilmente por un tipo de robot o brazo robótico, el cual pueda ser programado para lograr ejecutar con perfección y continuamente una tarea repetitiva en algún proceso productivo.

Está de más decir que grandes industrias han ido creando diversos tipos de robots, cada uno con su propio lenguaje de programación (software) y hardware de ejecución. En base a esto, también existen o se pueden inventar opciones para poder controlar dicho robot de una manera más universal, a través de la comunicación rs-232 que la mayoría de los robots tienen un puerto en su controladora, esto, para poder enviarle datos a través de este puerto serial. Con esta ventaja los profesionales de automatización y control les abre una ventana por la cual acceder al control de cualquier robot, aferrándose a los comandos de dicho robot y al estudio de la comunicación rs-232, como también el estudio y conocimiento de los microcontroladores que se pueden usar para dicho control se puede llegar a la manipulación total de dicha máquina.

En el siguiente proyecto se muestra el control de un brazo Mitsubishi Melfa RV-M1, el control se logra enviando datos seriales a través de un celular Android a un puerto serial bluetooth conectado a nuestro microcontrolador ArduinoMega, para que este pueda recibir los datos y entregarlos hacia el controlador del brazo, para la conexión entre arduino y brazo, se usará un conversor de salida TTL a RS-232 y un cable rs-232 (db25) a rs232 (db9), para hacer la conexión entre microcontrolador y controlador brazo.

Capítulo 1. Robótica industrial

Antes que todo, es conveniente dar un concepto sobre lo que significa un Robot Industrial, en el mundo existen varias definiciones, las cuales son aceptadas por diferentes personas, pero tal vez la más clara y aceptada a nivel mundial es la definición dada por la Asociación Internacional de Estándares (ISO): “Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas”.

Resulta interesante el observar la analógica que se hace de un brazo robótico con la anatomía de un brazo humano. (fig1.1).

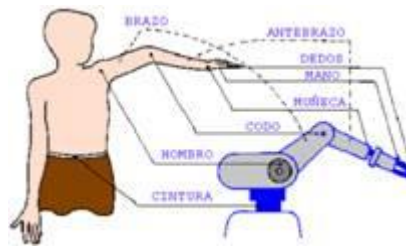


Fig1.1 Simulacion brazo-robot

1.2 Clasificación de los tipos de robots industriales

- **Tipo A. Manipuladores:** son robots multifuncionales con sistemas mecánicos básicos, por lo que deben ser utilizados en tareas sencillas y repetitivas, pueden ser controlados por una o más personas mediante control remoto, y cuenta con tres o cuatro grados de libertad.
- **Tipo B. Computarizados de precisión por controlador lógico programable (PLC):** es un manipulador pre ajustable que cuenta con sensores de regulación,

por lo tanto tienen mayor precisión y fuerza, se regulan mediante un PLC y tiene 4 grados de libertad.

- **Tipo C. Computarizados por CNC:** robot programable con trayectoria continua, son equipos más avanzados programados por CNC, con mayor fuerza y realizan trabajos más exigentes, posee seis grados de libertad.
- **Tipo D. Sensoriales:** robot que mediante sensores adquiere información de su entorno y es capaz de adaptarse a las condiciones del mismo. Poseen seis grados de libertad con una precisión de $\pm 0,04$ mm.

1.3 Los elementos que forman parte de la totalidad del robot son:

- **Manipulador:** es el elemento mecánicamente más importante, formado por varios eslabones unidos mediante articulaciones que permiten el movimiento de los mismos.
- **Controlador:** es el que regula cada uno de los movimientos del manipulador, las acciones, cálculos y procesamiento de la información.
- **Dispositivos de entrada y salida de datos:** permiten ingresar y leer información presente en los controladores.

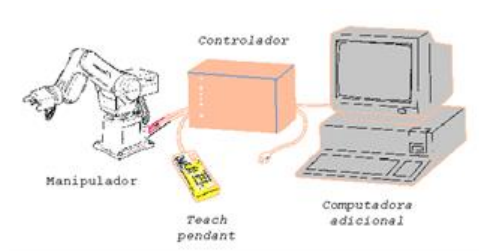


Fig1.2 Elementos del brazo

Antes de entrar en lleno a lo que nos importa, el robot Mitsubishi RV M1 primero debe conocerse las diferentes formas físicas que puede tener el brazo robótico, a lo cual se conoce como configuraciones para robots industriales.

1.4 Configuración de movimiento del robot

Configuración Cartesiana: posee tres grados de libertad con movimientos lineales, los cuales corresponden a los tres ejes X, Y y Z. Los movimientos que realiza este tipo de robot los hace mediante interpolaciones lineales. Presenta volúmenes de trabajo regulares.

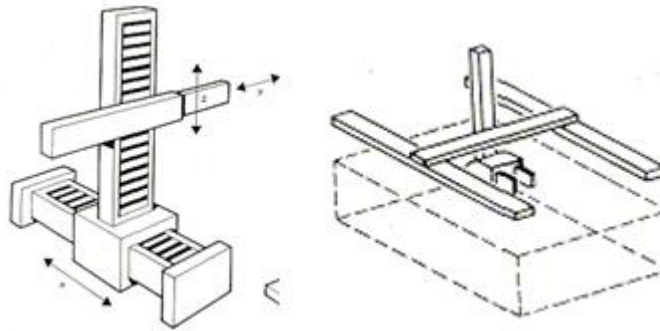


Fig 1.3. Configuración Cartesiana con su Volumen de Trabajo

Configuración Cilíndrica: posee tres grados de libertad con dos movimientos lineales y dos de rotación, por lo tanto, los movimientos se basan en interpolación lineal e interpolación por articulación. Presenta un volumen de trabajo parecido a un cilindro.

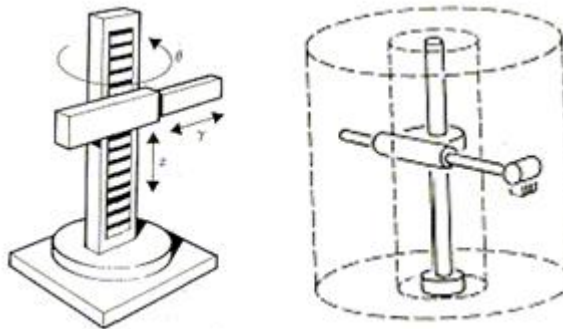


Fig 1.4. Configuración Cilíndrica con su Volumen de Trabajo

Configuración Polar: posee tres grados de libertad con movimientos de rotación y uno lineal, utiliza la interpolación por articulación para sus dos primeros movimientos y la lineal para movimientos de extensión. Volumen de trabajo irregular.

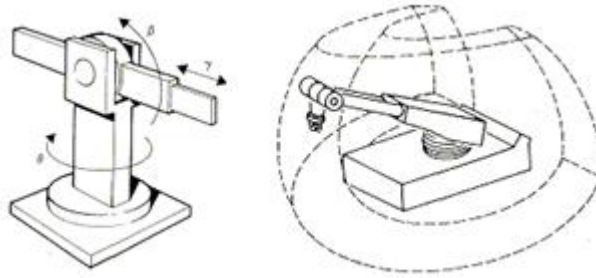


Fig1.5. Configuración Polar con su Volumen de Trabajo

Configuración Angular: posee una articulación con movimientos rotacionales y dos angulares. El movimiento se basa en interpolación por articulación. El volumen de trabajo es irregular, por lo que suele revisar el plano del robot

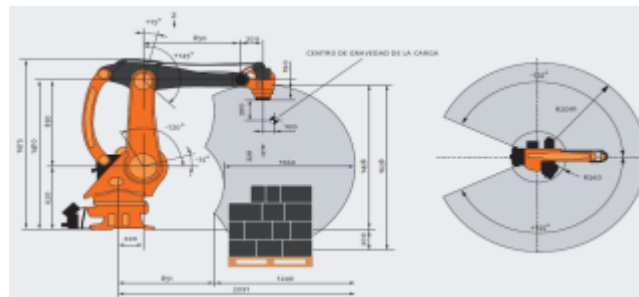


Fig1.6. Configuración Angular

Capítulo 2. ROBOT MELFA RV-M1

2.1 Características generales

El manipulador robot Mitsubishi RV-M1

- cuenta con cinco grados de libertad (DOF).
- Su capacidad de carga es de 1.2 Kg sin incluir el peso del efector final adaptado a él.
- Fuente de alimentación 120V AC, monofásica, 50/60 Hz
- Agarre manual estándar y fuerza de apertura programable

El sistema que permite operar el robot, se encuentra constituido por:

- Brazo articulado.
- Efector final (gripper, intercambiador de herramientas o sensor).
- Teaching box.
- Controlador.
- Cables de conexión.
- Computador con software para establecer comunicación con el robot.

El brazo cuenta con cinco articulaciones, siendo todas ellas rotacionales dentro de un Angulo limitado. Ver figura1.7

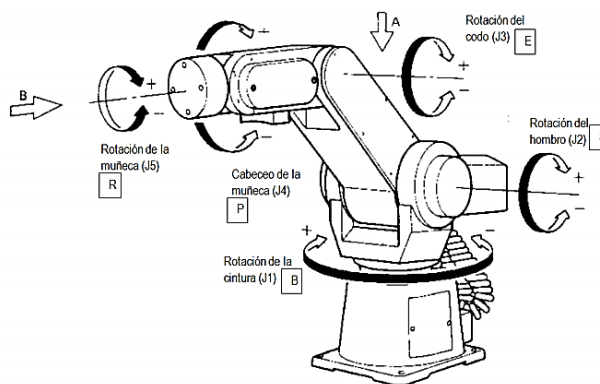


Fig 2.1. Movimientos del brazo con sus respectivos nombres

2.2 Características del movimiento del robot

Cada articulación puede rotar de forma limitada, para ofrecer al robot la posibilidad de posicionarse en cualquier lugar de un espacio confinado al que se le denomina volumen de trabajo del robot. La figura 1.8 presenta los límites de movimiento para cada articulación y el volumen de trabajo para el manipulador robot

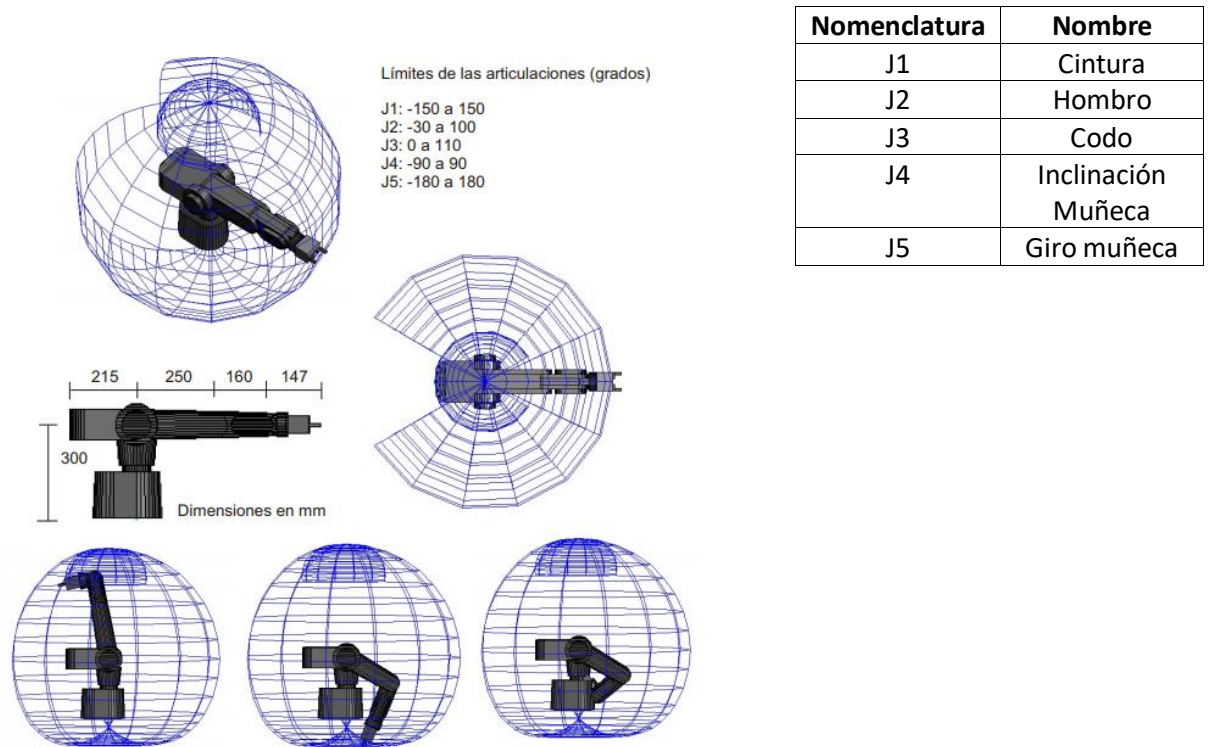


Fig2.2 Dimensiones y volumen de trabajo

2.3 Controlador del brazo

El brazo cuenta con una caja controladora (ver imagen 2.3), en la cual se basará la conexión del proyecto, esta controladora cuenta con el encendido y apagado del brazo y también, no menos importante, un puerto de conexión rs-232(db-25) el cual usaremos para lograr la comunicación arduino-brazo (ver imagen 2.4).

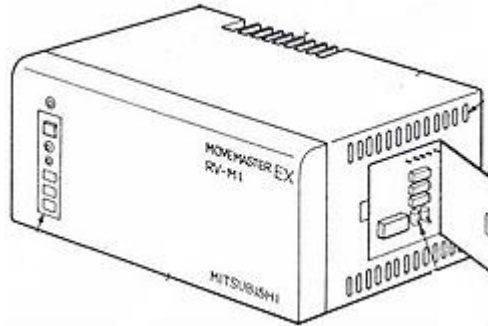


Fig 2.3 Controlador del brazo

Siguiendo con la controladora del brazo Melfa, es importante mencionar para la configuración, que esta controladora posee 5 switch en su parte lateral (ver imagen 2.4), los cuales tienen que ser configurados según lo que se busque realizar con el robot.

A continuación, se explicará cada switch y la configuración que tendrá cada uno de estos para poder llevar a cabo de manera efectiva dicho control.

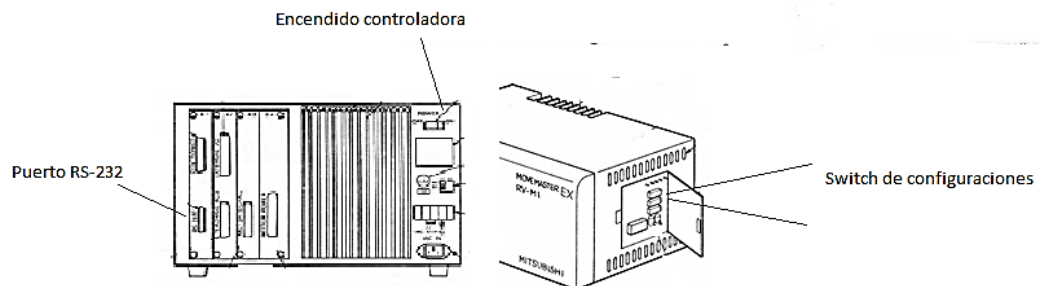


Fig 2.4 Parte posterior y lateral del controlador

2.4 Switch's de configuración

La controladora del brazo posee en su parte lateral una puerta la cual contiene 5 switch (ver imagen 2.5) cada uno de estos switch es importante configurarlos de manera correcta. Tres de estos switch como se puede ver en pantalla consta de una agrupación de 8 perillas, y dos de estos son interruptores que funcionan similar a un switch On-Off.



Fig 2.5 Switch de configuración

A continuación, se explicará el funcionamiento de los 2 interruptores que se pueden ver en la parte inferior de la imagen 2.5:

Interruptor ST1	Establece el modo de control. Es decir, si se encuentra en la posición superior, es el microprocesador del propio controlador quien tiene el control, por el contrario, si la posición es inferior, la computadora tendrá el control total del brazo.
Interruptor ST2	En la posición superior, los datos de la EPROM son transferidos a la memoria RAM de la unidad controladora. En posición inferior, los datos no son transferidos a la RAM de la unidad.

Tabla 2.1 Interruptor ST1 Y ST2

Justo sobre de estos dos interruptores se encuentran las otras 3 agrupaciones de interruptores o switch, (SW1, SW2 Y SW3), donde arriba es activado y abajo es desactivado, a continuación serán explicados:

SW1: Compuesto por 8 bits, enumerados de izquierda a derecha las funciones de este interruptor son las siguientes.

Bit 1	Selecciona la terminación para la transmisión de datos desde la conexión RS-232. Para la posición superior: CR+LF(“r”+) y para la posición inferior :CR(“”). Se pondrá en posición inferior (a no ser que este usandoMULTI16)
Bit 2	Indica si se revisa o no que los contenidos de la memoria RAM son retenidos cuando no está encendido; si se encuentra en la posición superior se revisa. En cambio, si está en l aposición inferior, no se revisará. Se pondrá en posición inferior si no se usa la batería.
Bit 3	Selecciona el tipo de tarjeta de entrada/salida usada, para la posición superior es del tipo A16 o B16, en la posición inferior es del tipo A8 o B8
Bit 4	Selecciona si se establecen, se cambian o se borrar las referencias a las posiciones de los datos en un sistema de coordenadas cartesiano, en la posición superior se activa, mientas que en la posición inferior se desactiva. Se activará cuando se haya establecido un sistema de coordenadas y

	se quiera enviar los datos escritos de la EPROM a la RAM
Bit 5	Selecciona el uso de los interruptores de la parte frontal del controlador o el de las señales externas para cargar el programa mientras la tarjeta de tipo A16 o B16 está siendo usada; para la posición superior selecciona las señales externas. Mientras que la posición inferior señala los interruptores de la parte frontal del controlador. Si se usa una tarjeta A8 o B8 deberá ponerse en la posición inferior.
Bit 6	Permite desactivar la tecla ENT del “teaching box” para liberar los frenos del robot. Normalmente ese bit se encontrará en su posición inferior.
Bit 7	No se usa
Bit 8	Permite activar o desactivar una alarma sonora. En la posición superior se emite un pitido cuando ocurre un error, mientras que en la posición inferior no se emite ningún sonido en caso de algún error.

Tabla 2.2 explicación Bits SW1

Para el proyecto la configuración a usar es la siguiente

1	2	3	4	5	6	7	8
•	•	•					•
			•	•	•	•	

Tabla 2.3 definición de posición de perillas SW1

SW2: esta agrupación de interruptores establecerá el formato para la transmisión de datos asíncrona. Las posiciones de este interruptor determinarán el número de bit de parada, el número de bits de datos, el bit de paridad y el factor de velocidad de transferencia. La siguiente figura simulará y aclarará el uso de cada bit del SW2.

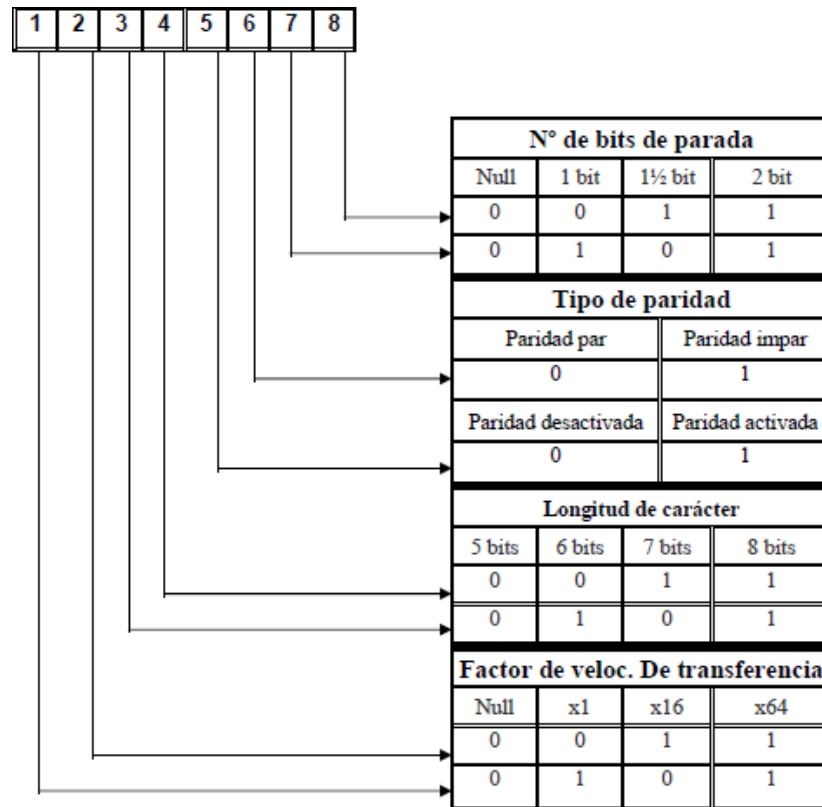


Fig 2.6 explicación bits SW2

Para el proyecto la configuración a usar es la siguiente:

1	2	3	4	5	6	7	8
	•	•	•		•	•	
•				•			•

Tabla 2.4 definición de posición de perillas SW2

Lo que significa que trabajamos con un factor de velocidad de transferencia x16, con una longitud de carácter de 8bits, paridad desactivada y un bit de parada.

SW3: Esta agrupación establece la velocidad media de transferencia, hay que tomar en cuenta el factor de velocidad que se configura en el SW2 debido a que todo va a depender del factor por el que se esté multiplicando (x1, x16, x64), los switch del interruptor SW3 serán explicados a continuación.

SW3	Factor de velocidad de Transferencia.		
	X1	X16	X64
1	1200	75	-
2	2400	150	-
3	4800	300	75
4	9600	600	150
5	-	1200	300
6	-	2400	600
7	-	4800	1200
8	-	9600	2400

Tabla 2.5 Explicación Switch SW3

Debido a que nuestra velocidad es de 9600 Y factor de velocidad es x16, el SW3 adoptará la siguiente configuración.

1	2	3	4	5	6	7	8
							•
•	•	•	•	•	•	•	

Tabla 2.6 definición de posición de perillas SW3

Con estas combinaciones de los interruptores, los parámetros de la programación serán los siguientes:

- 9600 Baudios
- 8 bits de datos
- Sin paridad
- 1 bit de parada

Capítulo 3. Comunicación RS-232

- La norma EIA/TIA RS-232 fue definida por la asociación de industrias en electrónica (EIA) en 1969. También se denomina estándar 24V.
- Creada inicialmente para la comunicación de un computador con un modem
- A nivel industrial utilizado ampliamente para comunicar equipos punto a punto, como plc, variadores de frecuencia, etc.

La comunicación Rs-232 también conocido como EIA/TIA RS-232C, es una interfaz que designa una norma para el intercambio de datos binarios serie entre un DTE (Data Terminal Equipment, "Equipo Terminal de Datos"), como por ejemplo una computadora, y un DCE (Data Communication Equipment, "Equipo de Comunicación de Datos"), por ejemplo un módem. Existen otros casos en los que también se utiliza la interfaz RS-232. Una definición equivalente publicada por la UIT se denomina V.24.



Fig3.1 Conector RS-232 (DB-9 hembra).

El RS-232 consiste en un conector tipo DB-25 (de 25 pines), debido a que muchos de los pines del DB-29 no son utilizados es normal encontrar la versión de 9 pines (DE-9, o popularmente mal denominados DB-9), más barato e incluso más extendido para cierto tipo de periféricos (como el ratón serie de la PC)

3.1 Especificaciones eléctricas de la interface

- Velocidad máxima según la norma =20kbps
- Capacidad del cable máxima según la norma debe ser <2.500pF
- Capacidad típica de los cables usados están entre 130 y 170pF por metro.

En nuestro caso, se utilizará en la salida de arduino un convertidor de señal, debido a un problema por la transmisión de datos en los niveles de voltaje. Ya que la salida de voltaje de arduino son 5V. En la siguiente imagen se ven los niveles eléctricos de transmisión de los datos de la interfaz RS-232

Especificaciones de los niveles eléctricos de la interface RS-232

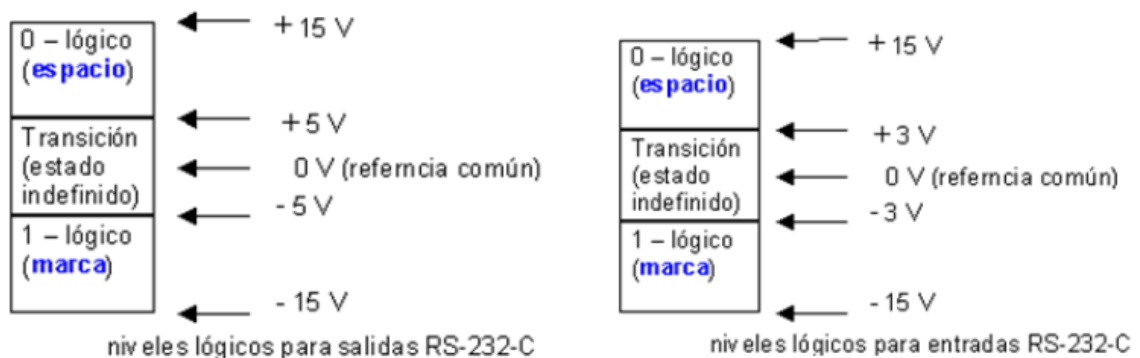


Fig 3.2 niveles de tensión RS-232

3.2 Tipo de Datos a enviar ASCII

Los datos a enviar a través de la comunicación son datos llamados ASCII que básicamente es un código de siete bits, lo que significa que usa cadenas de bits representables con siete dígitos binarios (que van de 0 a 127 en base decimal) para representar información de caracteres. En el momento en el que se introdujo el código ASCII muchas computadoras trabajaban con grupos de ocho bits (bytes u octetos), como la unidad mínima de información; donde el octavo bit se usaba habitualmente como bit de paridad con funciones de control de errores en líneas de comunicación u otras funciones específicas del dispositivo.

Este sistema convierte a dígitos binarios en caracteres imprimibles y no imprimibles conforme a un estándar establecido, tal y como se muestra en la tabla anterior. Los usuarios pueden realizar estos procesos de cálculo incluso sin herramientas adicionales, pero para ello deben comprender cómo se hacen cálculos **binarios o hexadecimales**. En un sistema numérico, cada posición de una potencia se corresponde con una base determinada, que en el sistema decimal es 10 y que en los sistemas binarios y hexadecimales es 2 o 16. En este caso se multiplica el valor de la cifra por el valor de la posición.

Caracteres de control ASCII			Caracteres ASCII imprimibles								ASCII extendido															
DEC	HEX	Simbolo ASCII	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
00	00h	NULL (carácter nulo)	32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	à	192	C0h	À	224	E0h	Ó			
01	01h	SOH (inicio encabezado)	33	21h	!	65	41h	A	97	61h	a	129	81h	Ù	161	A1h	á	193	C1h	Á	225	E1h	Ô			
02	02h	STX (inicio texto)	34	22h	"	66	42h	B	98	62h	b	130	82h	Ú	162	A2h	â	194	C2h	Â	226	E2h	Õ			
03	03h	ETX (fin de texto)	35	23h	#	67	43h	C	99	63h	c	131	83h	Û	163	A3h	ã	195	C3h	Ã	227	E3h	Ö			
04	04h	EOT (fin transmisión)	36	24h	\$	68	44h	D	100	64h	d	132	84h	Ü	164	A4h	ä	196	C4h	Ä	228	E4h	ö			
05	05h	ENQ (enquiry)	37	25h	%	69	45h	E	101	65h	e	133	85h	Ý	165	A5h	å	197	C5h	Å	229	E5h	ø			
06	06h	ACK (acknowledgement)	38	26h	&	70	46h	F	102	66h	f	134	86h	ÿ	166	A6h	æ	198	C6h	Æ	230	E6h	ù			
07	07h	BEL (timbre)	39	27h	'	71	47h	G	103	67h	g	135	87h	ÿ	167	A7h	ø	199	C7h	Ø	231	E7h	þ			
08	08h	BS (retroceso)	40	28h	(	72	48h	H	104	68h	h	136	88h	ÿ	168	A8h	ç	200	C8h	Ç	232	E8h	ÿ			
09	09h	HT (tab horizontal)	41	29h	)	73	49h	I	105	69h	i	137	89h	ÿ	169	A9h	¸	201	C9h	¸	233	E9h	ÿ			
10	0Ah	LF (salto de línea)	42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	ÿ	170	AAh	¸	202	CAh	¸	234	EAh	ÿ			
11	0Bh	VT (tab vertical)	43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ÿ	171	ABh	¼	203	CBh	¼	235	EBh	ÿ			
12	0Ch	FF (form feed)	44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	ÿ	172	ACH	½	204	CHh	½	236	EBh	ÿ			
13	0Dh	CR (retorno de carro)	45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ÿ	173	ADh	¾	205	CDh	¾	237	EDh	ÿ			
14	0Eh	SO (shift Out)	46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	ÿ	174	ACh	¸	206	CEh	¸	238	EBh	ÿ			
15	0Fh	SI (shift In)	47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	ÿ	175	AFh	¸	207	CFh	¸	239	EFh	ÿ			
16	10h	DLE (data link escape)	48	30h	0	80	50h	P	112	70h	p	144	90h	ÿ	176	B0h	¸	208	D0h	¸	240	F0h	ÿ			
17	11h	DC1 (device control 1)	49	31h	1	81	51h	Q	113	71h	q	145	91h	ÿ	177	B1h	¸	209	D1h	¸	241	F1h	±			
18	12h	DC2 (device control 2)	50	32h	2	82	52h	R	114	72h	r	146	92h	ÿ	178	B2h	¸	210	D2h	¸	242	F2h	±			
19	13h	DC3 (device control 3)	51	33h	3	83	53h	S	115	73h	s	147	93h	ÿ	179	B3h	¸	211	D3h	¸	243	F3h	±			
20	14h	DC4 (device control 4)	52	34h	4	84	54h	T	116	74h	t	148	94h	ÿ	180	B4h	¸	212	D4h	¸	244	F4h	±			
21	15h	NAK (negative acknowle.)	53	35h	5	85	55h	U	117	75h	u	149	95h	ÿ	181	B5h	¸	213	D5h	¸	245	F5h	±			
22	16h	SYN (synchronous idle)	54	36h	6	86	56h	V	118	76h	v	150	96h	ÿ	182	B6h	¸	214	D6h	¸	246	F6h	±			
23	17h	ETB (end of trans. block)	55	37h	7	87	57h	W	119	77h	w	151	97h	ÿ	183	B7h	¸	215	D7h	¸	247	F7h	±			
24	18h	CAN (cancel)	56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	¸	216	D8h	¸	248	F8h	±			
25	19h	EM (end of medium)	57	39h	9	89	59h	Y	121	79h	y	153	99h	ÿ	185	B9h	¸	217	D9h	¸	249	F9h	±			
26	1Ah	SUB (substitute)	58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	ÿ	186	BAh	¸	218	DAh	¸	250	FAh	±			
27	1Bh	ESC (escape)	59	3Bh	;	91	5Bh	[	123	7Bh	{	155	9Bh	ÿ	187	BBh	¸	219	DBh	¸	251	FBh	±			
28	1Ch	FS (file separator)	60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	ÿ	188	BCh	¸	220	DDh	¸	252	FDh	±			
29	1Dh	GS (group separator)	61	3Dh	=	93	5Dh	]	125	7Dh	}	157	9Dh	ÿ	189	BDh	¸	221	DDh	¸	253	FDh	±			
30	1Eh	RS (record separator)	62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	ÿ	190	BEh	¸	222	DEh	¸	254	FEh	±			
31	1Fh	US (unit separator)	63	3Fh	?	95	5Fh	_				159	9Fh	ÿ	191	BFh	¸	223	DFh	¸	255	FFh	±			
127	20h	DEL (delete)																								

Fig3.3 Caracteres ascii

3.3 Comunicación Serie

La comunicación de datos en un puerto serie se usa normalmente para efectuar comunicaciones asíncronas, es decir, sin tiempo preestablecido para, los datos llegan en ráfagas o paquetes de información; normalmente cada paquete es de 8 bits = 1 byte (el equivalente a un carácter en código ASCII). Algunos equipos envían carácter por carácter, mientras que otros guardan muchos caracteres en la memoria y cuando les toca enviarlos, los envían uno tras otro. El número de bits que se emplean para cada paquete.

Un envío de datos de un bit a la vez, de forma secuencial, sobre un canal de comunicación o un bus. Ver fig 2.4

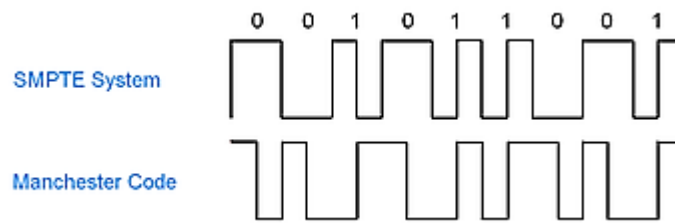


Fig3.4 Señales de comunicación Serie

¿Por qué se utiliza la transmisión de datos serie?

Si la velocidad de transferencia de datos en paralelo es mucho más rápida,

- Para realizar la comunicación de datos en paralelo se requiere gran cantidad de hilos conductores, pues debe ser establecido un hilo para cada bit de datos, además de las señales de control. Esto encarece notablemente la comunicación en función de la distancia. La comunicación serie requiere 2, 3 ó 4 hilos.
- Una entrada salida/serie puede ser transmitida a través de pares de cobre, cable coaxial, fibra óptica, vía radio o vía satélite, lo que proporciona comunicación con equipos remotos (redes locales) o muy remotos (Internet a través de las redes telefónicas y de datos).

- La comunicación paralela no posee el alto grado de estandarización que ha alcanzado la comunicación serie, lo que permite la intercomunicación entre equipos, por ejemplo, mediante RS232, USB o FireWire.

Los equipos de comunicaciones serie se pueden dividir entre:

- Simplex: Una comunicación serie simplex envía información en una sola dirección (por ejemplo, una emisora de radio comercial).
- Half-duplex: Significa que los datos pueden ser enviados en ambas direcciones entre dos sistemas, pero en una sola dirección al mismo tiempo.
- Full-duplex: En una transmisión full-duplex cada sistema puede enviar y recibir datos al mismo tiempo.

3.4 Comunicación RS-232 Arduino-Brazo

Como sabemos los niveles de salidas desde la placa arduino son de 5V, por ende, para lograr esta transmisión de datos, es necesario poner a las salidas seriales de arduino, un convertidor de señal, que en este caso es de TTL a RS-232, a continuación, se presentara una imagen de la conexión. Ver fig2.3

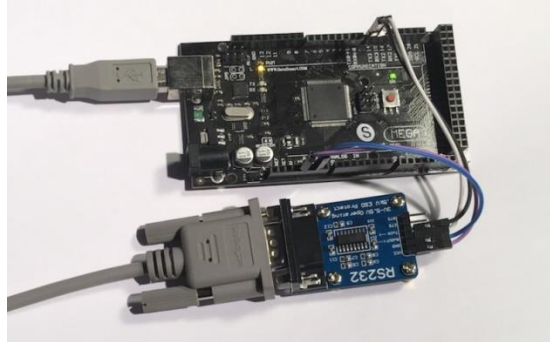


Fig 3.5 Conexión del convertidor TTL-RS-232

Para entender con mayor profundidad el funcionamiento del puerto RS-232, es importante saber que éste trabaja entre +12 voltios y -12 voltios, de manera que un cero lógico es cuando el terminal esté entre +9 y +12 voltios, y un uno lógico cuando esté entre -9 y -12 voltios, por lo que en un puerto serie que no está transmitiendo mantiene la terminal de transmisión en un 1 lógico, es decir, entre -9 y -12 voltios. Es por esto que es necesario el conversor, ya que como se mencionó anteriormente Arduino tiene salidas de voltaje de los 5V entonces la transmisión de datos no sería exitosa.

Lo que hace la tecnología TTL (Transistor to transistor logic) es subir esos niveles de voltajes a través de un circuito integrado llamado Max232 que contiene el conversor y así poder entregar los datos en niveles de voltajes optimos para la comunicación RS-232.

3.4.1 Puerto RS-232 brazo

Es importante decir que el brazo robótico cuenta con un Terminal DB-25, pero muchos de sus pines como se mencionó anteriormente no son utilizados.

En la siguiente imagen se puede ver con claridad que pines usa nuestro brazo robótico a intervenir

Asignación de los pines del conector RS-232 del robot RV

<i>N° DE PIN</i>	<i>Señal</i>	<i>N° de pin</i>	<i>Señal</i>
1	FG	14	No conectado
2	SD(TxD)	15	No conectado
3	RD(RxD)	16	No conectado
4	RS(RxD)	17	No conectado
5	CS(CTS)	18	No conectado
6	DR(DRS)	19	No conectado
7	SG	20	ER(DTR)
8	No conectado	21	No conectado
9	No conectado	22	No conectado
10	No conectado	23	No conectado
11	No conectado	24	No conectado
12	No conectado	25	No conectado
13	No conectado		

Tabla 3.1 Asignacion de pines DB-25 Brazo

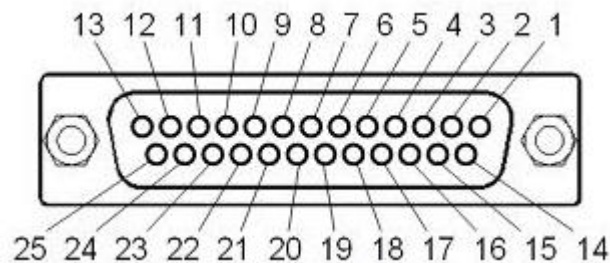


Fig3.6 Numerología de pines DB-25

A continuación, nombrará cada pin con su principal función:

- **FG:** Tierra conectada al terminal FG de la controladora
- **SD(TxD)- Salida:** Proporciona las líneas sobre las cuales la controladora envía datos al ordenador
- **RD(RxD)-Entrada:** Proporciona las líneas sobre las cuales el ordenador envía datos a la controladora
- **RS(RTS)- Salida:** Deberá ser activada si el ordenador desea transmitir datos
- **CT(CTS)-Entrada:** Señal usada para autorizar a la controladora a transmitir datos
- **DR(DRS)-Entrada:** Indica que el ordenador está listo para transmitir y recibir datos
- **SG:** Señal de referencia para datos y líneas de control (Tierra lógica)
- **ER(DTR)-Salida:** indica que la unidad controladora esta lista para transmitir y recibir datos

3.4.2 Puerto RS-232 Arduino

Como se mencionó anteriormente para la salida de datos serie desde el arduino, se tiene que conectar un convertidor de señal (ver fig 3.6), esto porque las salidas del arduino son en niveles de voltaje de +5V y como ya sabemos la transmisión de datos en el protocolo de comunicación RS-232 se logra con niveles de voltaje de los +-15V .

El convertidor de señal tiene por una parte pines machos los cuales corresponden a las señales que van a provenir desde el arduino es decir:

- 5V
- GND
- TX
- RX

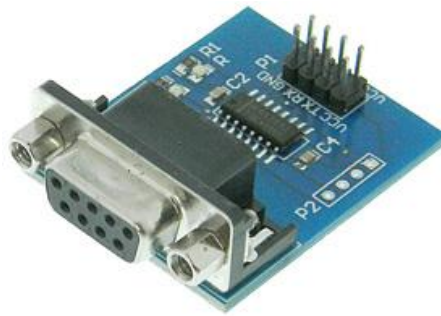


Fig3.7 Convertidor de señal

Por el otro extremo, el convertidor tiene un conector DB9 hembra para la señal de datos Seriales, el cual debe ir conectado al conector del propio controlador del brazo que es un conector de la señal, pero db25.

Es importante saber que pines de cada terminal se deben conectar, ya que el cambio de db25 A db9 debe ser de manera correcta para que los datos se puedan transmitir

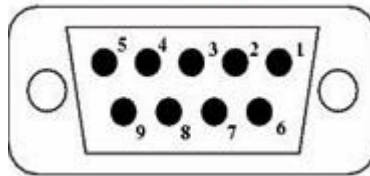


Fig3.8 Conector DB-9 Hembra

3.5 Señales y función de cada conector DB-9 Y DB-25

En base a lo anterior es importante hacer de manera correcta el cableado de conectores y las señales que utiliza el brazo para satisfacer las señales.

Por ende, en la siguiente tabla se representarán las señales de cada conector con su respectivo pin para conectar.

Numero de pin		Señal	Descripción
DB-25	DB-9		
1	1	-	GN o Tierra
2	3	TxD	Datos transmitidos
3	2	RxD	Datos recibidos
4	7	RTS	Peticion para enviar
5	8	CTS	Listo para enviar
6	6	DSR	Paquete de datos listo
7	5	SG	Señal de referencia (Tierra lógica)
20	4	DTR	Terminal de datos preparada

Tabla 3.2 Asignación pines de DB-25 A DB-9

3.6 Cable de comunicación Brazo-Arduino

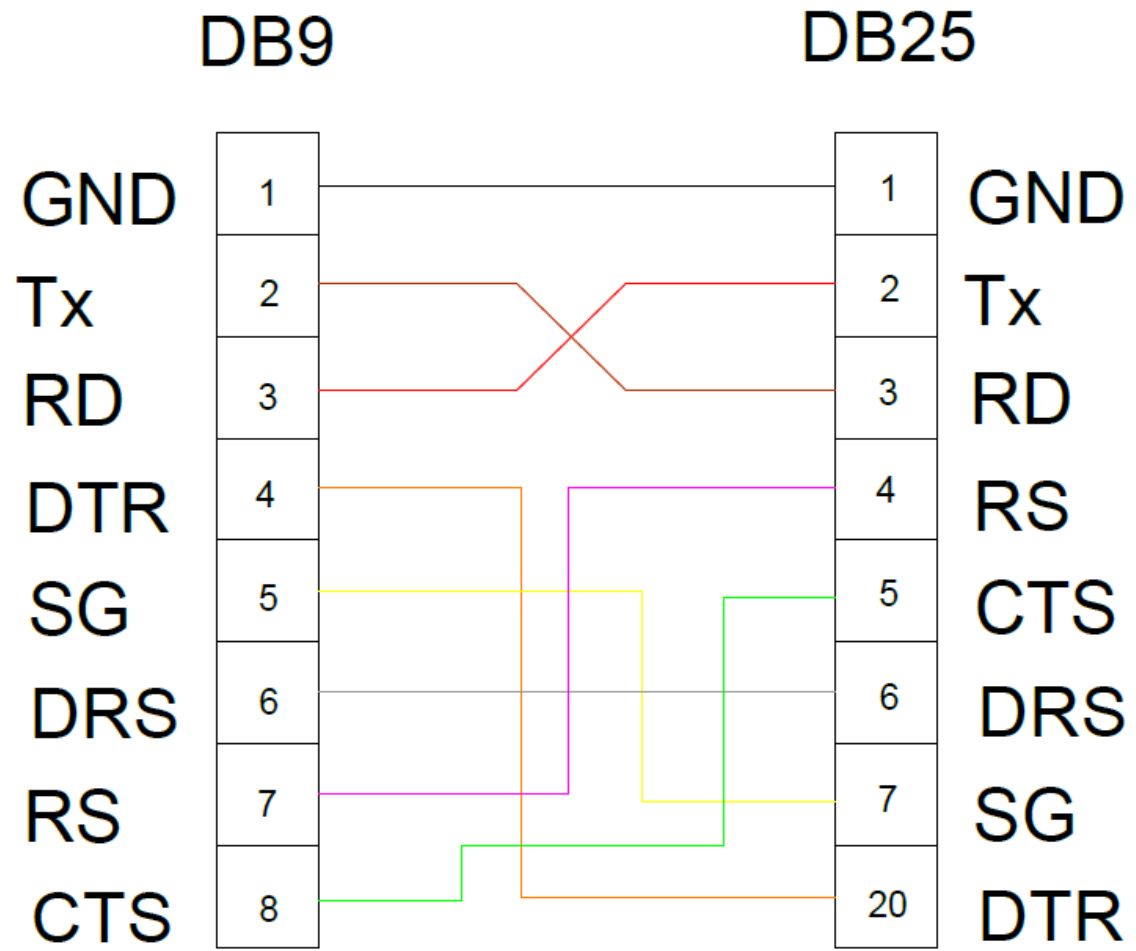


Fig 3.9 Cable de DB-25 a DB-9

Capítulo 4. Programación

Para llevar a cabo cualquier proyecto de manipulación de robot, una parte esencial y sino la más importante es la etapa de programación de los dispositivos de control, donde correspondiendo a cada software de programación se debe manejar de manera correcta utilizando minuciosamente cada comando para la total realización. Una vez dicho esto es importante saber que dispositivos vamos a programar.

Los dispositivos o software a programar son:

- **Arduino**
- **Módulo bluetooth**
- **Aplicación móvil (Android)**

4.1 Programación arduino

Para programar un Arduino, el lenguaje estándar es C++, aunque es posible programarlo en otros lenguajes. No es un C++ puro, sino que es una adaptación que proviene de avr-libc que provee de una librería de C de alta calidad para usar con GNU Compiler Collection (colección de compiladores GNU o GCC para abreviar) en los microcontroladores AVR de Atmel y muchas funciones específicas para los MCU AVR de Atmel.

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

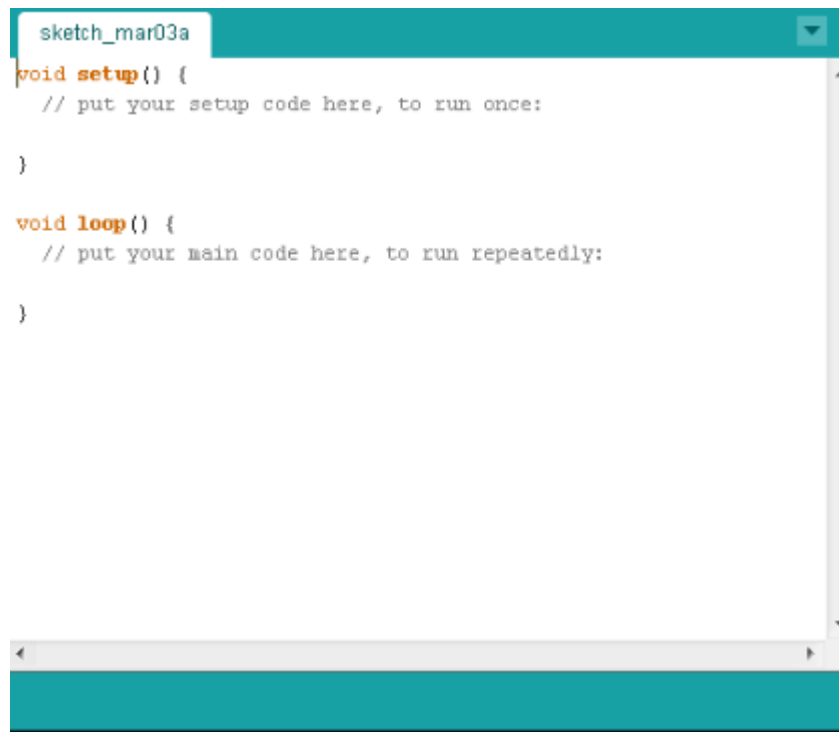
Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma. Actualmente existe un estándar, denominado ISO C++

4.1.1 interfaz programacion arduino

La interfaz de programacion de arduino es bastante sencilla, tiene 2 areas para la programacion

Setup: Donde se declaran las variables que se usarán

Void loop: El area donde se ejecuta el programa.

A screenshot of the Arduino IDE's code editor window. The title bar at the top shows 'sketch_mar03a'. The editor contains the following code:

```
void setup() {  
  // put your setup code here, to run once:  
  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

The code is color-coded: 'void' is blue, 'setup' and 'loop' are orange, and comments are grey. The editor has a light grey background and a dark teal border at the bottom.

Fig 4.1 Interfaz programación arduino

4.1.2 Programa de arduino

Lo que está a continuación es la programación con la cual se trabajará, explicada en cada línea con un comentario en formato Arduino.

```
char bt;

String brazo = ""; //Almacena los datos en forma de palabra

void setup () {
  Serial. Begin (9600); //Velocidad de comunicación puerto serie 0
  Serial1.begin(9600); //Velocidad de comunicación puerto serie 1 (modulo Bluetooth)
  (38400)
  Serial2.begin(9600); //Velocidad de comunicación puerto serie 2 (brazo RV-M1)
}

void loop () {
  while (Serial1.available() ==0); //cuando la disponibilidad del puerto serie 1 es igual a 0
  bt = Serial1.read(); //igual a la variable "bt" a la lectura del puerto serie 1
  mov (); //llama a la sub rutina "mov"
  Serial2.println(brazo); //Imprime el valor de la variable "brazo" el el puerto serie 2
}

void mov () {
  switch (bt) {
  case 'Z':
    brazo = "NT"; //Sitúa el brazo a la posición inicial
    break;

  case 'X':
```

```
brazo = "OG"; //Sitúa las articulaciones a sus respectivos 0°  
break;
```

```
case 'Y':  
brazo = "GO"; //Acciona la apertura de la pinza  
break;
```

```
case 'W':  
brazo = "GC"; //Acciona el cerrado de la pinza  
break;
```

```
default:  
brazo = ""; //Anula el carácter erróneo  
break;  
}  
}
```

4.2 Programación módulo bluetooth

Siguiendo con el tema de la programación, es turno de programar y mostrar la configuración utilizada para poder hacer la relación de Arduino –Android, el cual es posible a través de un dispositivo electrónico llamado Modulo bluetooth.

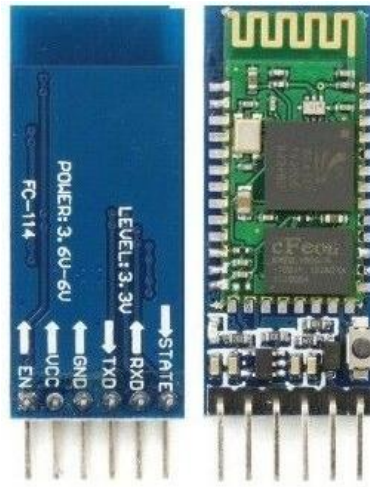


Fig 4.2 Modulo bluetooth

4.2.1 Bluetooth

Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo coste. Los dispositivos que incorporan este protocolo pueden comunicarse entre sí cuando se encuentran dentro de su alcance con rango óptimo de 10 metros (opcionalmente 100m con repetidores). Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión es suficiente mediante un canal de comunicación de máximo 720Kb/s (1Mbps de capacidad bruta) y la frecuencia de radio con la que trabaja está en el rango de 2.4 a 2.48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Duplex con un máximo de 1600 saltos/seg. Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1Mhz; esto permite dar seguridad y robustez.

4.2.2 Configuración bluetooth

Para que el hc-05 entre en modo AT, requiere que cuando se encienda el módulo, el pin KEY este en HIGH. Por eso hemos conectado la tensión Vcc del módulo Bluetooth al pin 8 de nuestro Arduino

El consumo del módulo es mínimo y nuestro Arduino es capaz de alimentarlo sin problemas, por eso el módulo se encenderá cuando pongamos HIGH en el pin 9. Esto nos permitirá poner en HIGH el pin digital 8, al iniciar nuestro programa y después levantar el pin 8, de este modo cuando arranque entrará sin más en el modo de comandos AT.

El resto de los pines se conectan de forma similar a lo que se realizó en la sesión anterior. Txd y Rxd se deben conectar cruzados con los pines de comunicación de Arduino, que usaremos mediante la librería software Serial.

El pin State refleja, supuestamente, la situación en la que se encuentra el módulo y por ahora no se realizaron las pruebas de su funcionamiento

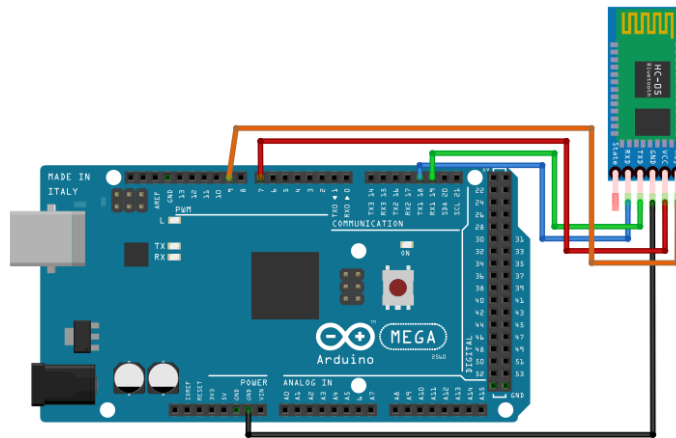


Fig4.3 Esquema conexionado bluetooth-arduino

4.2.3 Programa para configurar Bluetooth HC-05

```
#include <SoftwareSerial.h>
```

```
SoftwareSerial BT1(10, 11); // RX | TX
```

```
void setup()
```

```
{ pinMode(8, OUTPUT);    // Al poner en HIGH forzaremos el modo AT
  pinMode(9, OUTPUT);    // cuando se alimente de aquí
  digitalWrite(9, HIGH);
  delay (500) ;          // Espera antes de encender el modulo
  Serial.begin(9600);
  Serial.println("Levantando el modulo HC-05");
  digitalWrite (8, HIGH); //Enciende el modulo
  Serial.println("Esperando comandos AT:");
  BT1.begin(57600);
}
```

```
void loop()
```

```
{ if (BT1.available())
  Serial.write(BT1.read());
  if (Serial.available())
    BT1.write(Serial.read());
}
```

4.2.4 Comandos AT para módulo HC-05

AT Comando de prueba, debe responder con OK

AT+BAUD<Numero> configurar la tasa de baudios con la que trabajará el bluetooth

AT+NAME<Nombre> configura el nuevo nombre que llevará el módulo HC-05

AT+PIN<pin de 4 dígitos> configura la clave de emparejamiento nueva

AT+PSWD=<número 4 dígitos> cambiar la contraseña del módulo bluetooth

Otros comandos

AT+ROLE=1 Comando para colocar el módulo en modo Maestro (Master)

AT+ROLE=0 Comando para colocar el módulo en modo Esclavo (Slave)

AT+VERSION Obtener la versión del firmware

AT+PIO=10,1\r\n Colocar el pin de IO de propósito general a nivel alto

4.2.5 Algunas características que ofrece este protocolo son:

- Establece conexiones con poco gasto de energía.
- Establece enlaces por lo general de corta duración.
- Otorga seguridad mediante diversas maneras de cifrado de datos, además de exigir el uso de un PIN para establecer conexiones entre equipos.
- Soporta voz y datos
- Tiene un bajo costo de producción e implementación, se planteó que no sobrepasara los US\$5 por dispositivo.

4.3 Aplicación Móvil

Para elaborar la aplicación móvil serán necesarias la comprensión de dos tipos de lenguajes directamente relacionados entre sí

4.3.1 scratch

Scratch es un lenguaje de programación que fue creado en la universidad de Massachusetts Instituto of Tecnology (MIT) y buscaba que el usuario promedio logre desarrollar un programa de una manera más didáctica sin necesidad de que este posea un conocimiento tan profundo sobre el tema y así logre comprender o ampliar sus conocimientos de este.

Este lenguaje consiste en la agrupación de bloques ya que de esta manera es mucho más visual para el usuario lograr una correcta estructura del programa (fig.4.1).

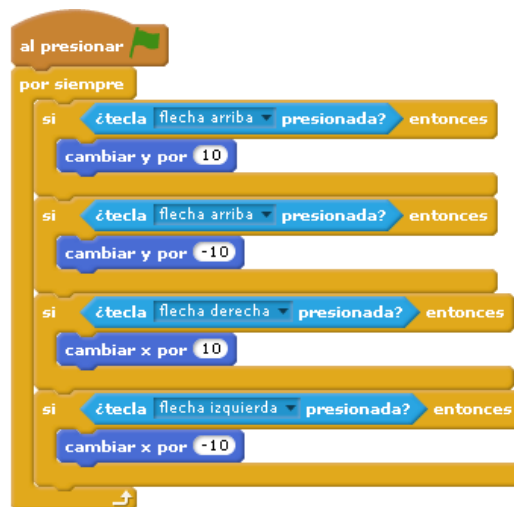


Fig 4.4 ejemplo interfaz scratch

4.3.2 APP inventor

Posee una interfaz gráfica basada en Scratch y el StarLogo , que permite a los usuarios arrastrar y soltar objetos visuales para crear una aplicación que puede ejecutarse en el sistema Android originalmente desarrollada por Google y mantenida ahora por el Instituto de Tecnología de Massachusetts (MIT)

El editor de bloques utiliza la biblioteca Open Blocks de Java para la creación de lenguajes de programación visuales.

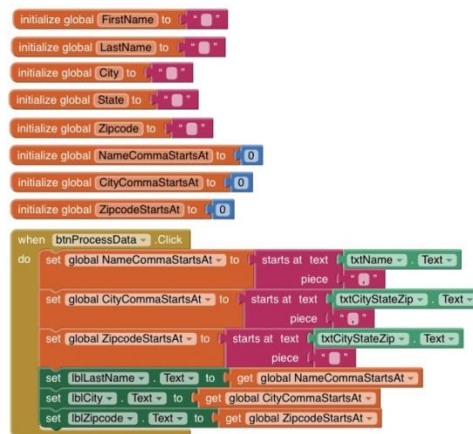
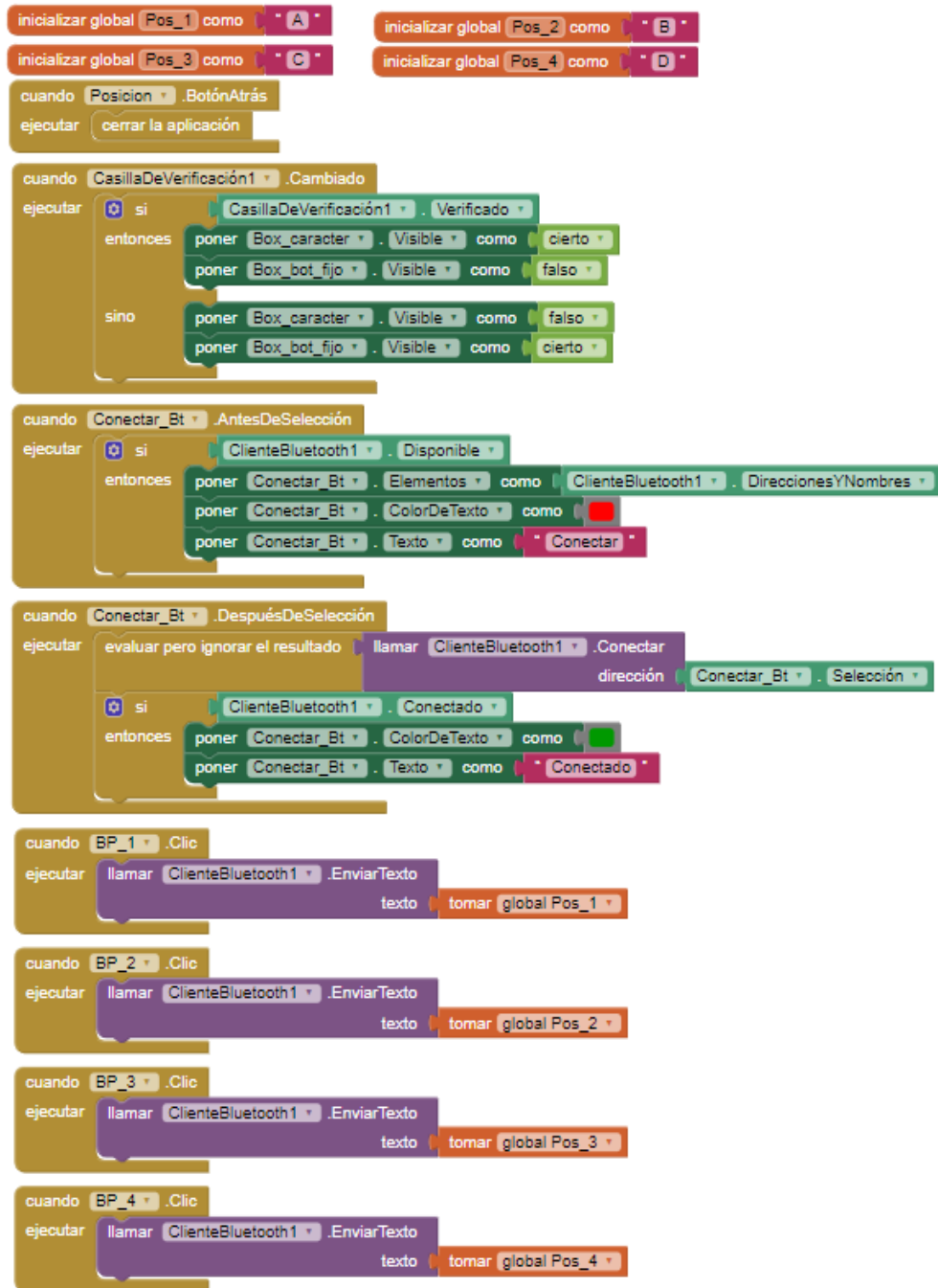


Fig4.5 ejemplo interfaz app inventor

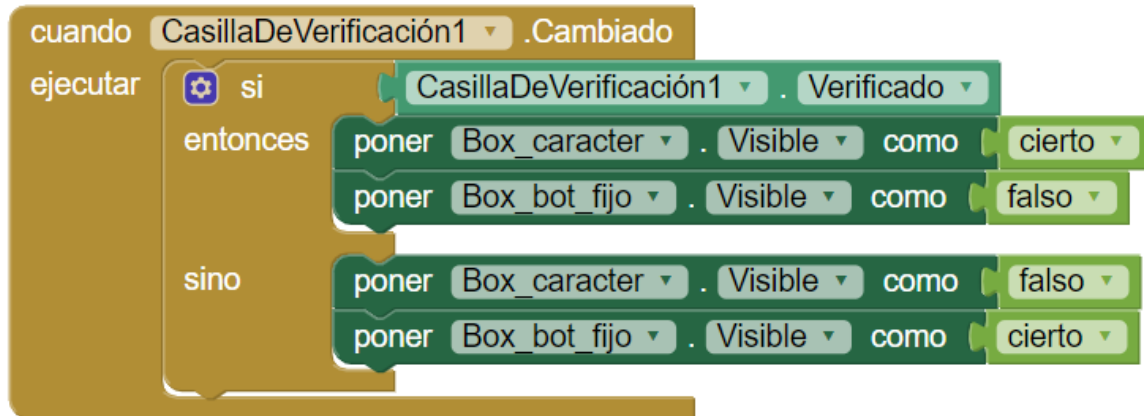
4.2.3 Creación app para Android



Declaración de variables



Visualización de caracteres enviados



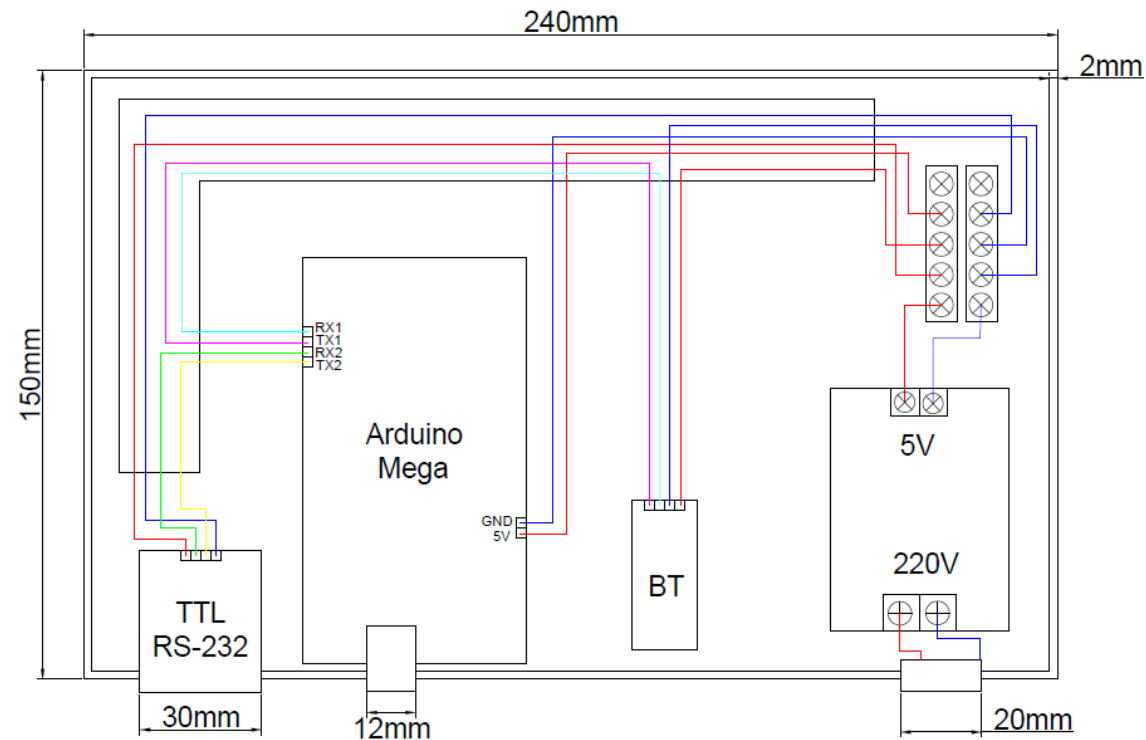
Establecimiento de conexión con bluetooth



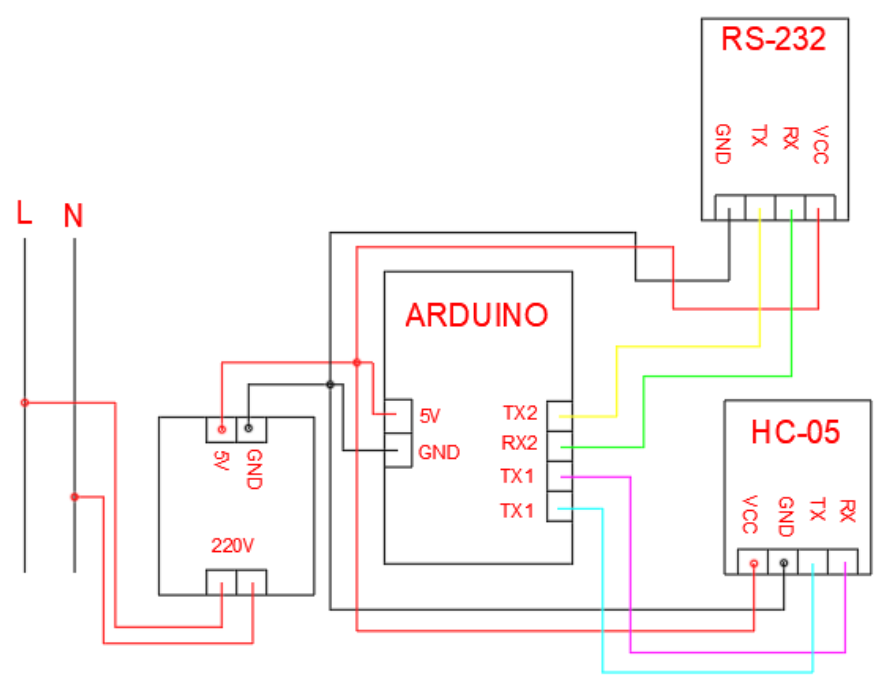


Esto nos dice que al pulsar un botón este envía un carácter del cual lo toma de su respectiva variable global

5. Esquema de montaje



5.1 Esquema de conexionado



6. Conclusiones

Durante la realización de este trabajo se aprecia la importancia que tiene el protocolo de comunicación RS-232 ya que abre una puerta de opciones para el control de máquinas, ya sean robot, o maquinarias grandes de control, es por esto que la mayoría de las maquinas poseen un puerto RS-232, ya sea DB-25 o DB-9 en sus controladoras. Es importante decir que este control a través de RS-232 no puede ser mas de 15m pero a pesar de esto y la ventaja que entrega es que si no se tiene el control directo de la máquina, un profesional de automatización y control, con habilidades en comunicación serial y el manual de la máquina, podría manejarla sin problemas a través de dicho puerto.

En este caso se trabajó con un brazo robótico Melfa de Mitsubishi el cual tenía su propio control local, se armó un sistema de hardware conectado a su entrada RS-232, en agrupación con un microcontrolador arduino y un celular Android, se estudió y se dejó cátedra de la programación a utilizar para cada componente, tanto como configuraciones y programación y/o creación de aplicaciones para manejarla.

No está de más decir que los sistemas de programación y control, es complicado saber en qué lugar del software o hardware se podría generar la falla, problemas tales como diferencia en la velocidad de comunicación, tanto emisor como receptor, longitudes de los datos enviados y/o diferencia en la cantidad de bits de control que puede provocar al momento de intercambio de datos.