# MANUAL DE CONSTRUCCIÓN Y OPERACIÓN DE “JUANITO”

Raúl Alejandro Zamora Zamora
Estudiante de Ingeniería Mecatrónica

# Índice





# Manual de construcción y operación de “JUANITO”

## INTRODUCCIÓN

Juanito es un robot tipo SumoBot autónomo diseñado específicamente para competencias de sumo en la clasificación de 1 kilogramo. Su estructura robusta está pensada para confrontarse de manera efectiva, utilizando material de impresión 3D. Equipado con motores de alto desempeño, puede generar la fuerza necesaria para empujar a sus oponentes fuera de la arena. Además, sus sensores de respuesta rápida permiten que Juanito reaccione de una manera eficaz durante su funcionamiento. Juanito utiliza una XMotion V3 como microcontrolador que se programa mediante IDE Arduino.

Este manual proporciona las instrucciones necesarias para la configuración, operación y mantenimiento del robot Juanito. Asegúrese de leer detenidamente cada sección antes de utilizar el robot.

## ASPECTOS DE SEGURIDAD

Para garantizar la seguridad antes y durante la operación del robot, siga estas recomendaciones:

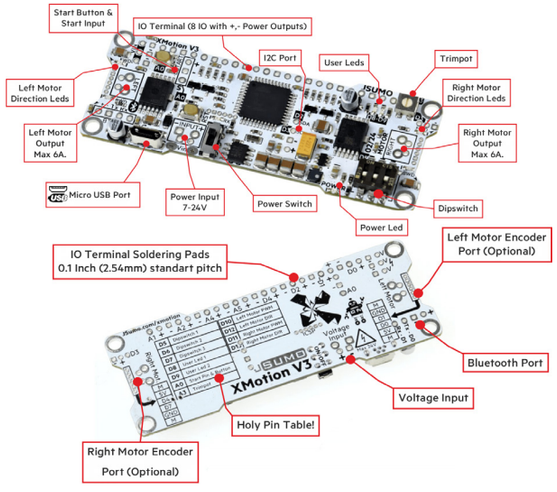
- Mantenga el robot alejado de líquidos.
- Mantenga los componentes fuera de riesgo estático.
- No lo opere cerca de superficies inflamables, altas temperaturas o altas corrientes.
- Verifique que todos los cables y conexiones se encuentren bien ajustados antes de encenderlo.
- Utilice equipo de protección antiestático.
- No transporte la batería en áreas despresurizadas.
- Revise la ficha técnica de carga de la batería LIPO antes de conectarla a la corriente.
- Mantenga el robot lejos de superficies altas.
- Mantenga precaución con el uso de la navaja.

## ELEMENTOS

### *UNIDAD DE CONTROL*

La unidad de control en Juanito es una placa de desarrollo XMotion V3 compartible con Arduino IDE, desarrollada por la compañía JSUMO. Esta placa está diseñada para proyectos de robótica, como los SumoBots, ofreciendo un rendimiento fiable y una excelente capacidad de procesamiento. Su rango de voltaje se encuentra entre los 7 y los 28 voltios de alimentación, con un voltaje regulado de 1,600 miliamperios en el circuito. Uno de los puntos fuertes de esta placa es su capacidad para gestionar motores de alta potencia a través de sus drivers integrados, soportando hasta 6 amperios por canal, lo que garantiza que Juanito pueda utilizar motores de gran torque sin sobrecalentarse ni perder eficiencia durante su funcionamiento. Además, la XMotion V3 tiene protecciones integradas contra sobre corrientes y corto circuitos, lo que añade una capa de seguridad durante las competencias de alto impacto.

Figura 1. Partes de la placa XMotion V3




Nota. Adaptado de *XMotion All In One Controller V3* [Imagen], por JSUMO, 2024, XMotion Features 2 (https://www.jsumo.com/xmotion-robot-controller). CC BY 4.0

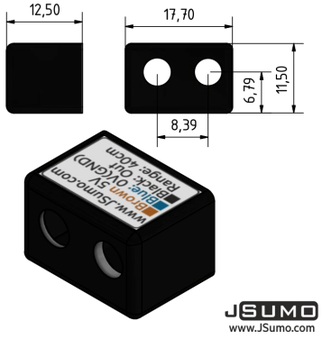


### *SENSORES*

Juanito utiliza dos tipos de sensores, el primero un sensor digital de distancia que lo ayuda a identificar la posición de su rival, y el segundo un sensor infrarrojo que mediante la lectura analógica de cambio de color en la arena lo ayuda a identificar su área de operación para no salirse de la misma.

En el caso del primer sensor, en este proyecto se utilizan cinco sensores del tipo *JS40F Digital Distance Sensor* fabricado por la compañía JSUMO. Este sensor está diseñado para identificar presencia y distancia para proyectos de robótica de pequeña escala, detectando objetos hasta 80 centímetros de distancia.

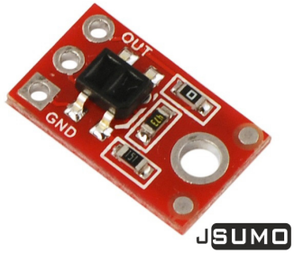
Figura 2. Sensor JS40F




Nota. Adaptado de *JS40F Digital Distance Sensor* [Imagen], por JSUMO, 2024, js40f digital infrared ir distance sensor (https://www.jsumo.com/js40f-digital-infrared-ir-distance-sensor-min-40-cm-range). CC BY 4.0

Para el caso del segundo sensor, en este proyecto se utilizan dos sensores del tipo *QTR1A Contrast (Edge) Sensor* fabricado por la compañía JSUMO. Este sensor está diseñado para reconocer cambios en la frecuencia de color, lo que lo ayuda específicamente en este proyecto a identificar su área de operación (arena) y no salirse de la misma mediante la configuración analógica en la programación.

Figura 3. Sensor QTR1A Contrast (Edge) Sensor




Nota. Adaptado de QTR1A Contrast (Edge) Sensor [Imagen], por JSUMO, 2024 QTR1A Contrast Edge Sensor (https://www.jsumo.com/qtr1a-contrast-edge-sensor). CC BY 4.0

*MOTORES*

Los motores que se utilizan son dos del tipo *Titan Dc Gearhead Motor 12V 1000 RPM HP* fabricados por la compañía JSUMO. Estos motores son diseñados para proyectos que requieren de una potencia y torque muy eficiente. Su alimentación es de 12 voltios de corriente directa, tiene un diámetro de 37mm y una longitud de 75mm, eje de 6mm de diámetro y 17mm de longitud, con 6 agujeros para tornillos M3 en el frente; corriente sin carga de 240mA y corriente de bloqueo de 5.9A; todos los engranajes son metálicos y el torque de bloqueo es de 7.5 kg-cm.

Figura 4. Titan Dc Motor

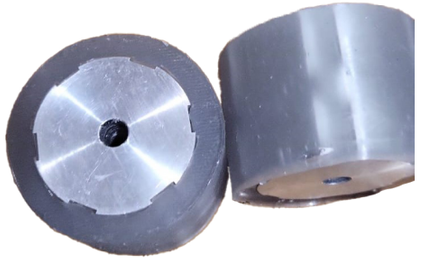

Nota. Adaptado de Titan Dc Gearhead Motor 12V 1000 RPM HP [Imagen], por JSUMO, 2024 Titan Dc Gearhead Motor 12V 1000 RPM HP (https://www.jsumo.com/jsumo-titan-dc-gearhead-motor-12v-1000-rpm-hp). CC BY 4.0



*RUEDAS*

Las ruedas que utiliza Juanito son dos, fabricado con rin de aluminio y neumático de caucho de silicón. Se utiliza una medida estándar de 45 mm de diámetro y 30 mm de ancho y se ajustan al eje del motor mediante tornillos M3.

Figura 5. Ruedas

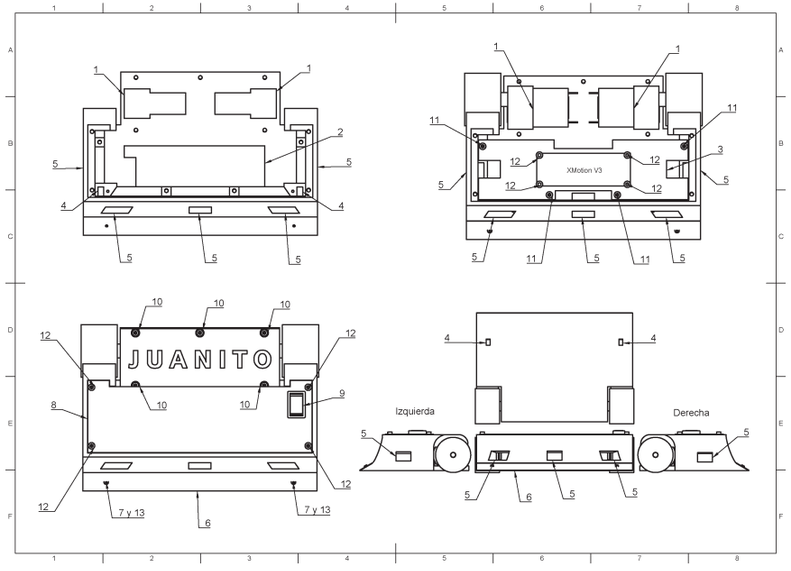


*CUERPO*

El diseño del cuerpo está realizado en Fusion 360 (Eagle). Su diseño robusto y practico a su vez está pensado para su confrontamiento de una manera efectiva. La posición de los motores tipo *Titan DC 1,000 RPM 12V* (1, Figura 6) se encuentra de tal manera que al montar las llantas de 45 x 30 mm de caucho de silicón el chasis se plante sobre la superficie. Sobre la parte central se creó un espacio para una batería LIPO de 110 x 35 x 22 milímetros (2, Figura 6). Sobre la batería se monta una superficie que sirve de sostén para la placa XMotion V3 (3, Figura 6), su diseño permite el paso del cableado de los motores, la batería y los sensores. En la parte frontal inferior se encuentra el espacio para montar los sensores infrarrojos tipo *QTR1A* (4, Figura 6). Así mismo en la parte frontal superior se encuentran cinco ranuras diseñadas para montar los sensores digitales de distancia tipo *JSF40* (5, Figura 6). En la parte frontal exterior se encuentra un espacio de 200 x 18.5 x 0.5 mm para montar una navaja de SumoBot (6, Figura 6) con una rosca (para tornillos de 2 mm) a 20 mm de distancia del extremo izquierdo y derecho (7, Figura 6). Sobre la parte superior se monta una tapadera (8, Figura 6) que cubre la parte interna del SumoBot, la cual contiene un switch (9, Figura 6) que

facilita la manipulación del estado de alimentación (on/off) para evitar la necesidad de tener expuesto los cables de la batería LIPO o la placa de desarrollo. El total de tornillos que utiliza es de 5 tornillos M3x40 mm cabeza hexagonal (10, Figura 6), 4 tornillos M3x20 mm cabeza hexagonal (11, Figura 6), 8 tornillos M3x8 mm cabeza hexagonal (12, Figura 6) y 2 tornillos M2x4 mm cabeza plana de cruz (13, Figura 6).

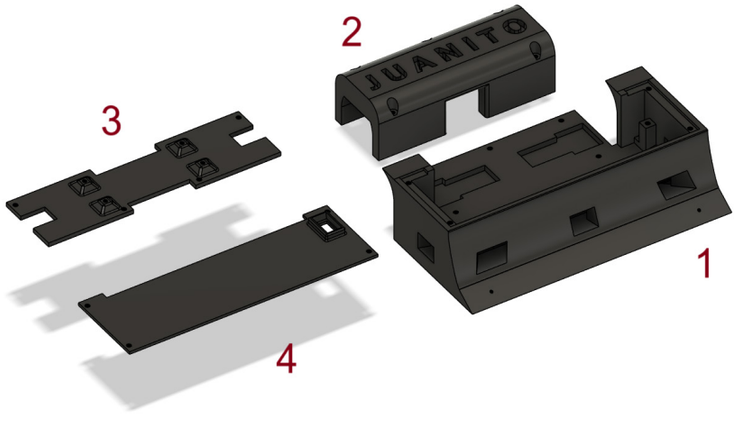
Figura 6. Componentes




Así mismo el diseño del cuerpo en bruto está dividido en cuatro partes, las cuales se encuentran fabricadas mediante impresión 3D con material PETG, las cuales están nombradas: Chasis (1, Figura 7), Cubre motores (2, Figura 7), Porta placa (3, Figura 7) y Cubierta (4, Figura 7).



Figura 7. Partes que componen el cuerpo

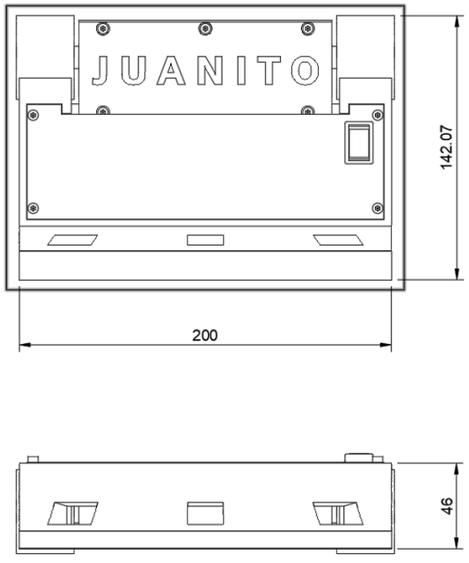



Los archivos STL de las partes que componen el cuerpo se pueden encontrar en el siguiente enlace de GitHub: https://github.com/rauulzamoora/Juanito-Sumo

## ESPECIFICACIONES TÉCNICAS

- *Dimensiones:* 200 mm de largo (x), 142.07 mm de ancho (y) y 46 mm de alto (z).
- *Peso:* 1,000 gramos.
- *Batería:* LIPO 11.7 voltios 2,200 miliamperios (110 x 35 x 22 milímetros).
- *Controlador:* XMotion V3 using Arduino IDE.

Figura 8. Dimensiones.

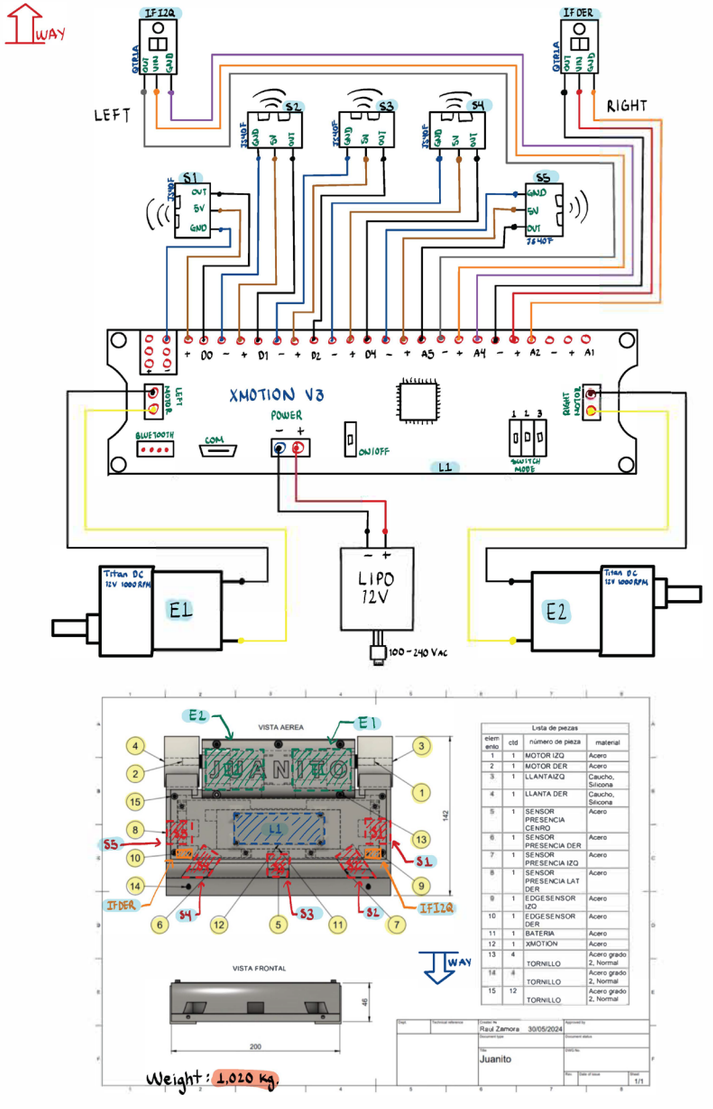



## DIAGRAMA DE CONEXIONES

En la Figura 9 se presenta el diagrama representativo de conexiones entre los componentes y el microcontrolador. Los colores mostrados en las conexiones están apegados al cableado físico del robot.



Figura 9. Diagrama de conexiones.



## CODIGO APLICADO

El código utilizado en este proyecto es el que se muestra a continuación.

```
// Los sensores de JS40F dan 1 cuando ven al oponente.

#include <xmotionV3.h> // Biblioteca XMotion incluida
int RightSensor = A5; // Pin del sensor de oponente derecho
int RightDiagonalSensor = 4; // Pin del sensor de oponente diagonal derecho
int MiddleSensor = 2; // Pin del sensor de oponente central
int LeftDiagonalSensor = 1; // Pin del sensor de oponente diagonal izquierdo
int LeftSensor = 0; // Pin del sensor de oponente izquierdo

int LeftLine = A2; // Pin del sensor de línea izquierdo
int RightLine = A4; // Pin del sensor de línea derecho
int Start = A0; // Pin del botón de inicio

int Led1 = 8;
int LastValue = 0;

void setup() {
  xmotion.StopMotors(100); // Detener motores, en caso de que los pines de salida estén abiertos.
  xmotion.ToggleLeds(100); // Función de parpadeo para hacer parpadear 2 LEDs de usuario con un retardo.
  pinMode(Led1, OUTPUT);
  pinMode(RightSensor, INPUT); // Declaramos las entradas y salidas digitales.
  pinMode(RightDiagonalSensor, INPUT);
  pinMode(MiddleSensor, INPUT);
  pinMode(LeftDiagonalSensor, INPUT);
  pinMode(LeftSensor, INPUT);
  pinMode(Start, INPUT);
  Serial.begin(9600);
}
```

Continúa sg. pag.



```
void loop() {
  while (digitalRead(Start) == 0) // Se espera que se pulse el botón. Cuando se presiona, da un valor de 1.
  { // Esta instrucción if
    if (digitalRead(LeftSensor) == 1 || digitalRead(MiddleSensor) == 1 || digitalRead(RightSensor) == 1 || digitalRead(RightDiagonalSensor) == 1 || digitalRead(LeftDiagonalSensor) == 1)
    {
      digitalWrite(Led1, HIGH);
    } else {
      digitalWrite(Led1, LOW);
    }

    xmotion.UserLed2(100); // Parpadeo del LED de usuario 2 en intervalos de 100 milisegundos.
  }
  xmotion.CounterLeds(1000, 5);
  while (1) {
    if (analogRead(LeftLine) < 800 && analogRead(RightLine) > 800 ) // El sensor de línea izquierdo detecta la línea
    {
      xmotion.Backward(100, 100); // Retroceder al 100% de velocidad, retroceso de 100 ms.
      xmotion.Right0(100, 200); // Giro a la derecha al 100% de velocidad, duración de 200 ms.
    } else if (analogRead(LeftLine) > 800 && analogRead(RightLine) < 800 ) // El sensor de línea derecho detecta la línea
    {
      xmotion.Backward(100, 100); // Retroceder al 100% de velocidad, retroceso de 100 ms.
      xmotion.Left0(100, 200); // Giro a la izquierda al 100% de velocidad, duración de 200 ms.
    } else if (analogRead(LeftLine) < 800 && analogRead(RightLine) < 800 ) // Ambos sensores detectan la línea
```

Continúa sg. pag.

```
  {
    xmotion.Backward(100, 200); // Retroceder al 100% de velocidad, retroceso de 200 ms.
    xmotion.Right0(100, 300); // Giro a la izquierda al 100% de velocidad, duración de 200 ms.
  }

  else if (digitalRead(MiddleSensor) == 1 ) // El sensor central ve al oponente (0 No visto, 1 Visto)
  {
   xmotion.Forward(100, 1); // Ambos motores hacia adelante al 100% de velocidad, 1 milisegundo
    LastValue = 0;
  }
  else if (digitalRead(RightSensor) == 1) // El sensor derecho ve al oponente
  {
   xmotion.Right0(70, 1); // Giro a la derecha al 70% de potencia durante 1 milisegundo
    LastValue = 1;
  }
  else if (digitalRead(LeftSensor) == 1) // El sensor izquierdo ve al oponente
  {
   xmotion.Left0(70, 1); // Giro a la izquierda al 70% de potencia durante 1 milisegundo
    LastValue = 2;
  }
  else if (digitalRead(LeftDiagonalSensor) == 1) // El sensor diagonal izquierdo ve al oponente
  {
   xmotion.ArcTurn(20, 70, 1); // Motor izquierdo al 20% de velocidad, derecho al 70% de velocidad durante 1 ms.
    LastValue = 2;
  }
  else if (digitalRead(RightDiagonalSensor) == 1) // El sensor diagonal derecho ve al oponente
```

Continúa sg. pag.



```
  {
   xmotion.ArcTurn(70, 20, 1); // Motor izquierdo al 70% de velocidad, derecho al 20% de velocidad durante 1 ms.
    LastValue = 1;
  } else if (LastValue == 0) { // El sensor central vio al oponente
   xmotion.Forward(20, 1);
  } else if (LastValue == 1) { // El sensor derecho o diagonal derecho vio al oponente
   xmotion.Right0(30, 1);
  } else if (LastValue == 2) { // El sensor izquierdo o diagonal izquierdo vio al oponente
   xmotion.Left0(30, 1);
  }
 }
}
```

También puede encontrar el código en el siguiente enlace: https://github.com/rauulzamoora/Juanito-Sumo

## CONFIGURACIÓN

*Paso 1: Verificación de componentes.*

Asegúrese de que todos los componentes se encuentren presentes.

- 4 partes del cuerpo (ver Figura 7).
- 1 placa de desarrollo XMotion V3.
- 5 sensores tipo JS40F.
- 2 sensores tipo QTR1A.
- 2 motores Titan Dc Gearhead Motor 12V 1000 RPM HP.
- 2 ruedas 45 x 30 mm (ver Figura 5)
- 1 Batería LIPO 11.7 V 2,200 mAh.
- 1 switch on/off (9, Figura 6).
- 1 navaja de 200 x 18.5 mm.
- 5 tornillos M3x40 mm cabeza hexagonal.
- 4 tornillos M3x20 mm cabeza hexagonal.
- 8 tornillos M3x8 mm cabeza hexagonal.

- 2 tornillos M2x4 mm cabeza plana de cruz.
- Cable de conexión de datos micro USB.
- Cargador de batería LIPO.

*Paso 2: Ensamble.*
Monte los componentes en su respectiva posición, tome como referencia la Figura 6 y guíese de los siguientes pasos:

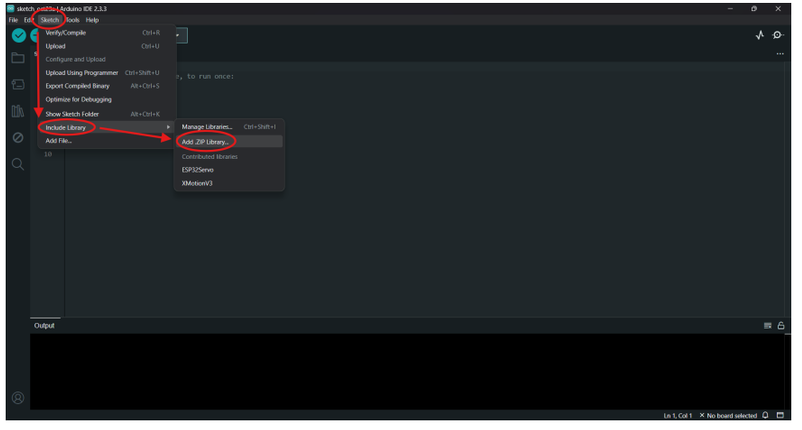
1. Coloque los motores en su posición (1, Figura 6), asegúrese de mantener el eje en la posición correcta, para ello tomaremos de referencia la rueda y comprobaremos que en la posición en la que se encuentra el eje la rueda apenas toque la superficie.
2. Colocaremos el Cubre motores (2, Figura 7) sobre los motores y lo atornillaremos utilizando los tornillos M3x40 mm (ver Figura 6).
3. Coloque los sensores infrarrojos QTR1A en su posición (4, Figura 6).
4. Coloque los sensores de distancia JS40F en su posición (6, Figura 6).
5. Coloque la batería LIPO en su posición (2, Figura 6).
6. Monte el Porta placa (3, Figura 7) y utilice 4 tornillos M3x20mm (ver Figura 6).
7. Sobre el Porta placa coloque la placa de desarrollo XMotion V3 (3, Figura 6) y utilice los tornillos M3x8 mm para asegurar su posición (ver Figura 6).
8. Realice las conexiones eléctricas y de comunicación. Asegúrese de realizar una buena conexión y de mantener el mejor orden posible.
9. Colocamos el switch on/off en el espacio sobre la Cubierta (4, Figura 7) y realizamos la conexión entre el switch y la batería LIPO.
10. Montamos la Cubierta (4, Figura 7) sobre el Chasis (1, Figura 7) y utilizamos 4 tornillos M3x20 mm para fijar (ver Figura 6).
11. Colocamos la navaja en su respectiva posición (6, Figura 6) y fiamos utilizando los tornillos M2x4 mm (13, Figura 6).
12. Colocamos las ruedas en los ejes del motor y fijamos con los tornillos M3 respectivo de cada rueda.
13. Antes de encender verificar una vez más las conexiones y que todo se encuentre bien montado.



*Paso 3. Entorno de programación.*
Para poder programar en la placa de desarrollo XMotion V3 en Arduino IDE primero debemos descargar la librería en el siguiente repositorio de GitHub: https://github.com/rauulzamoora/Juanito-Sumo.

Una vez descargada la librería en formato ZIP procederemos a instalarlo en el entorno de Arduino IDE (versión 2.3.3) siguiendo la ruta: Sketch / Include Library / Add ZIP Library, como se muestra en la Figura 10.

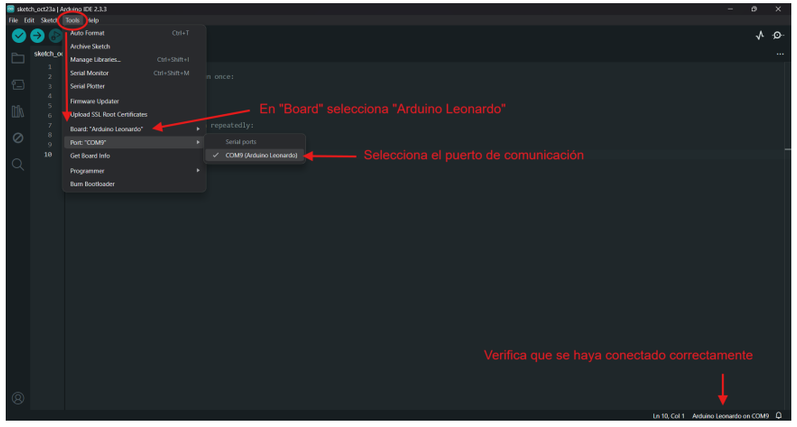
Figura 10. Instalación de librerías




Una vez que se abra el explorador de archivos seleccionaremos en la ruta de descargas el archivo nombrado “XMotionV3 Library”. Al finalizar reiniciamos el entorno Arduino IDE.
Para obtener comunicación entre el software y la XMotion V3 haremos los siguientes pasos:

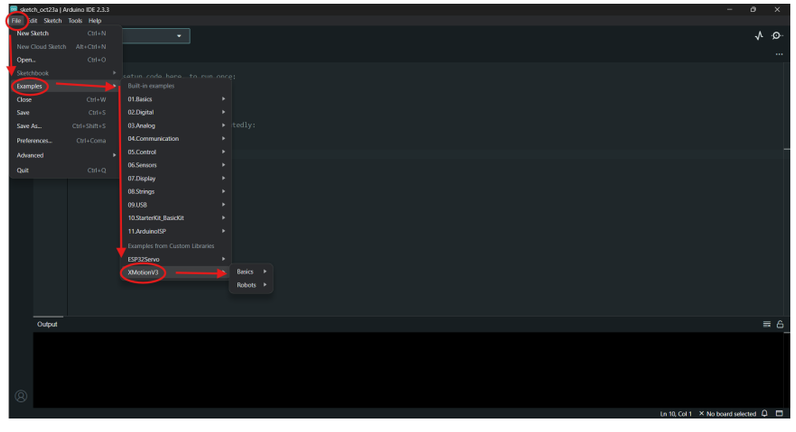
1. Verificamos que las conexiones del robot con el microcontrolador se encuentren debidamente conectados.
2. Conectamos mediante un cable micro USB la XMotion V3 con el ordenador.
3. Una vez conectados, en el entorno de Arduino IDE seleccionaremos el tipo de tarjeta (Arduino Leonardo) y el puerto de comunicación serial (COM) como se muestra en la Figura 11.

Figura 11. Selección de puerto y tarjeta.




4. En este paso ya deberíamos poder compilar y cargar nuestro programa de manera satisfactoria. Puedes consultar ejemplos de códigos dentro de la misma librería en la siguiente ruta: (File / Examples / XMotionV3), como se muestra en la Figura 12.

Figura 12. Códigos de ejemplo.





Puedes consultar más ejemplos y guías de apoyo en el Blog de JSUMO entrando al siguiente enlace: https://blog.jsumo.com/xmotion-basics-xmotion-101/

## OPERACIÓN DEL ROBOT

### *ENCENDIDO Y APAGADO*

Para encender el robot primero conectamos la batería LIPO, después encienda el power switch de la XMotion V3 (ver Figura 1), por último, encienda el switch on/off que se encuentra en la cubierta (9, Figura 6), este último será el que utilizaremos intermitentemente siempre y cuando se cumplan los dos primeros pasos. El robot debe trabajar de manera autónoma.

Para apagar el robot sólo presione de retorno el switch on/off que se encuentra en la cubierta (9, Figura 6). Consecuentemente si se pretende desmontar apague el power switch de la XMotion V3 (ver Figura 1) y después desconecte la batería LIPO.

### *MODOS DE OPERACIÓN*

La placa de desarrollo XMotion V3 cuenta con tres modos de operación manipulados mediante un DIP switch de tres pasos (ver Figura 1) mismos que puedes configurar mediante la programación de esta. Los pines del interruptor DIP están conectados a D5, D6, D7.

A continuación, se muestra un ejemplo de configuración de los modos de operación.

```
#define DipSwitch1 5 // Dipswitch 1 tied to Digital 5
#define DipSwitch2 6 // Dipswitch 2 tied to Digital 6
#define DipSwitch3 7 // Dipswitch 3 tied to Digital 7
#define Start 10 // Button tied to Digital 10

void setup() {
pinMode(DipSwitch1, INPUT); //Dipswitch 1 Declared as Input
pinMode(DipSwitch2, INPUT); //Dipswitch 2 Declared as Input
pinMode(DipSwitch3, INPUT); //Dipswitch 3 Declared as Input
pinMode(Start, INPUT); //Button Declared as Input

digitalWrite(DipSwitch1, HIGH); // Dipswitch Inputs are High (Pull-up
made)
digitalWrite(DipSwitch2, HIGH);
digitalWrite(DipSwitch3, HIGH);

Serial.begin(9600); //Serial Interface started with 9600 bits per sec.
}

void loop() {
Serial.print(“Button State:”); //We are writing this statement to serial
Monitor
Serial.print(digitalRead(Start)); //digital reading of button
Serial.print(” “);
Serial.print(“Dipswitch Inputs:”);
Serial.print(digitalRead(DipSwitch1)); // digital reading of dipswitches
Serial.print(digitalRead(DipSwitch2));
Serial.println(digitalRead(DipSwitch3));
delay(100);
}
```

Nota. Adaptado de *XMotion Basics – XMotion 101* [Imagen], por JSUMO, 2024. How to use built-in button and dipswitch? Digital Input Elements (https://blog.jsumo.com/xmotion-basics-xmotion-101/).



## MANTENIMIENTO

- Recargue completamente la batería después de un uso prolongado. Verificar el tiempo de carga de su batería.
- Limpie los cuidadosamente sensores con un paño antiestático para evitar lecturas incorrectas.
- Limpie cuidadosamente la placa de desarrollo XMotion V3 con material especializado.
- Remueva la suciedad que se pueda encontrar en el chasis.

## SOLUCIÓN DE PROBLEMAS

Si el robot presenta fallas, consulte esta tabla de problemas comunes:

| *Problema* | *Posible Causa* | *Solución* |
|---|---|---|
| El robot no enciende | Batería agotada | Recargar la batería |
| No responde al control | Problema en el código o en las conexiones físicas | Realizar una inspección utilizando el diagrama de conexiones (Figura 9) y verificar el código (Puedes realizar una prueba programando un bloque por cada tipo de sensor) |
| Los sensores no detectan | Conexión defectuosa | Realizar una inspección utilizando el diagrama de conexiones (Figura 9) y verificar el código (Puedes realizar una prueba programando un bloque por cada tipo de sensor) |

Para soporte adicional, contacte a los desarrolladores:
Correo:


- raul.zamora3045@alumnos.udg.mx
- ricardo.bermudez3857@alumnos.udg.mx

## REFERENCIAS

JSUMO. (n.d.). *XMotion robot controller.* Recuperado el 22 de octubre de 2024, de https://www.jsumo.com/xmotion-robot-controller

JSUMO. (2024). *Librería XMotionV3*. Recuperado el 22 de octubre de 2024, de https://blog.jsumo.com/wp-content/uploads/2024/03/XMotionV3.zip

JSUMO. (n.d.). *XMotion basics: XMotion 101*. Recuperado el 22 de octubre de 2024, de https://blog.jsumo.com/xmotion-basics-xmotion-101/

Arduino. (n.d.). *Arduino IDE (versión 2.3.3)*. https://www.arduino.cc/en/software