**Manual: Carrito controlado por celular con ESP32 y Python**

**Ing. Sandra Patricia González**

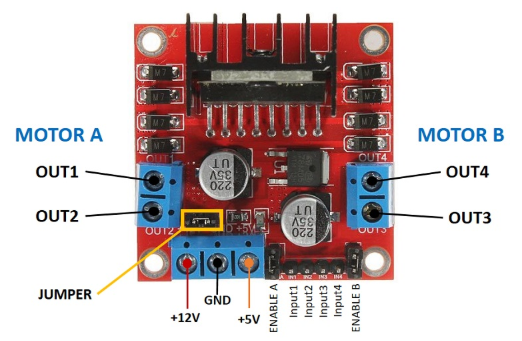
**1. Materiales y explicación de cada componente**

* **ESP32**



Tomado de: <https://images.theengineeringprojects.com/image/webp/2024/03/esp32-pinout.jpg.webp?ssl=1>  
Es una placa de desarrollo con un microcontrolador que incluye **WiFi y Bluetooth integrados**.  
Será el **cerebro del carrito**, encargado de recibir las órdenes enviadas desde el celular y controlar el movimiento de los motores.

* **Puente H L298N**

**Video:** [**https://www.youtube.com/watch?v=bIUswhZ4xBQ&ab\_channel=AprendiendoconGustavo**](https://www.youtube.com/watch?v=bIUswhZ4xBQ&ab_channel=AprendiendoconGustavo)  


Es un módulo electrónico que permite **controlar la dirección y velocidad** de los motores de corriente continua.  
Cada L298N puede controlar **dos motores**, por eso se necesitan dos para manejar los cuatro motores del carrito.

* **Motorreductores de 6V con llantas (x2)**



**Video:** [**https://www.youtube.com/watch?v=PuFb6B7g\_PU&ab\_channel=Profe.Alencaste**](https://www.youtube.com/watch?v=PuFb6B7g_PU&ab_channel=Profe.Alencaste)

**Video:** [**https://www.youtube.com/watch?v=sFv9Su5vvm8&ab\_channel=FARADAYPERU**](https://www.youtube.com/watch?v=sFv9Su5vvm8&ab_channel=FARADAYPERU)  
Son pequeños motores de corriente continua con una caja reductora que disminuye la velocidad pero aumenta la fuerza (**torque**).  
Los reductores permiten que el carrito tenga la potencia suficiente para moverse.

* **Baterías de litio 3.7V 1800mAh (x2)**



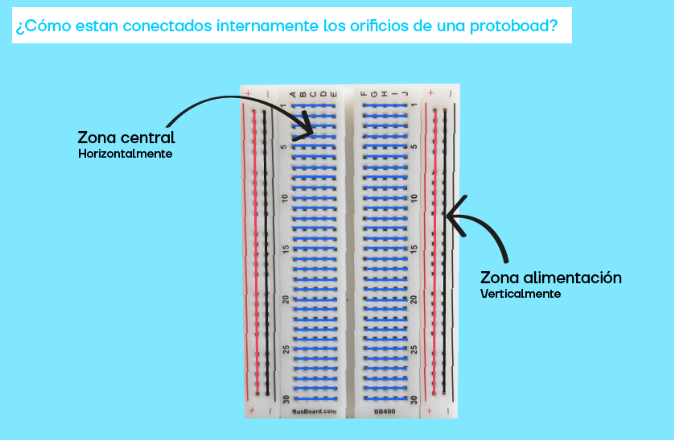
Son la fuente de energía del proyecto.

* + Al conectarlas en **serie**, aumenta el voltaje.
  + Al conectarlas en **paralelo**, aumenta la duración.
  + Con ellas se alimentan los puentes H y el ESP32.
* **Porta baterías (x1)**
*   
  Es el soporte donde se colocan las baterías para conectarlas de forma segura y ordenada.
* **Cables Dupont**



Son cables de conexión rápida que permiten unir la ESP32, los módulos L298N y los demás componentes.  
Se utilizan para hacer las conexiones eléctricas sin necesidad de soldar.

* **Protoboard (opcional)**

  
Placa de pruebas que permite realizar conexiones temporales sin necesidad de soldadura. Facilita el montaje y pruebas previas.

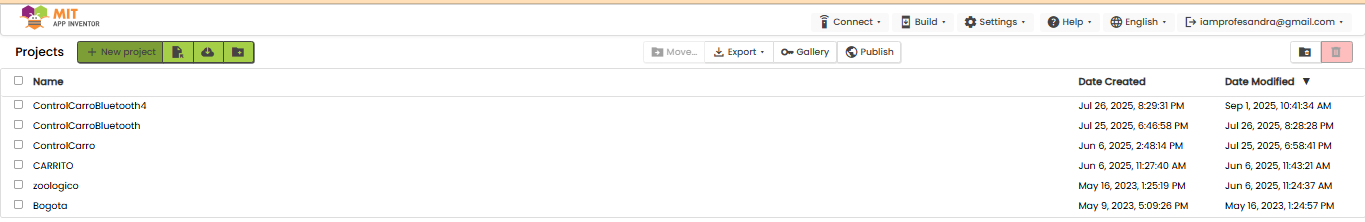
* **Celular Android con MIT App Inventor**



El celular será el **control remoto del carrito**.  
En MIT App Inventor se diseña una aplicación con botones que enviarán comandos al ESP32 por Bluetooth.

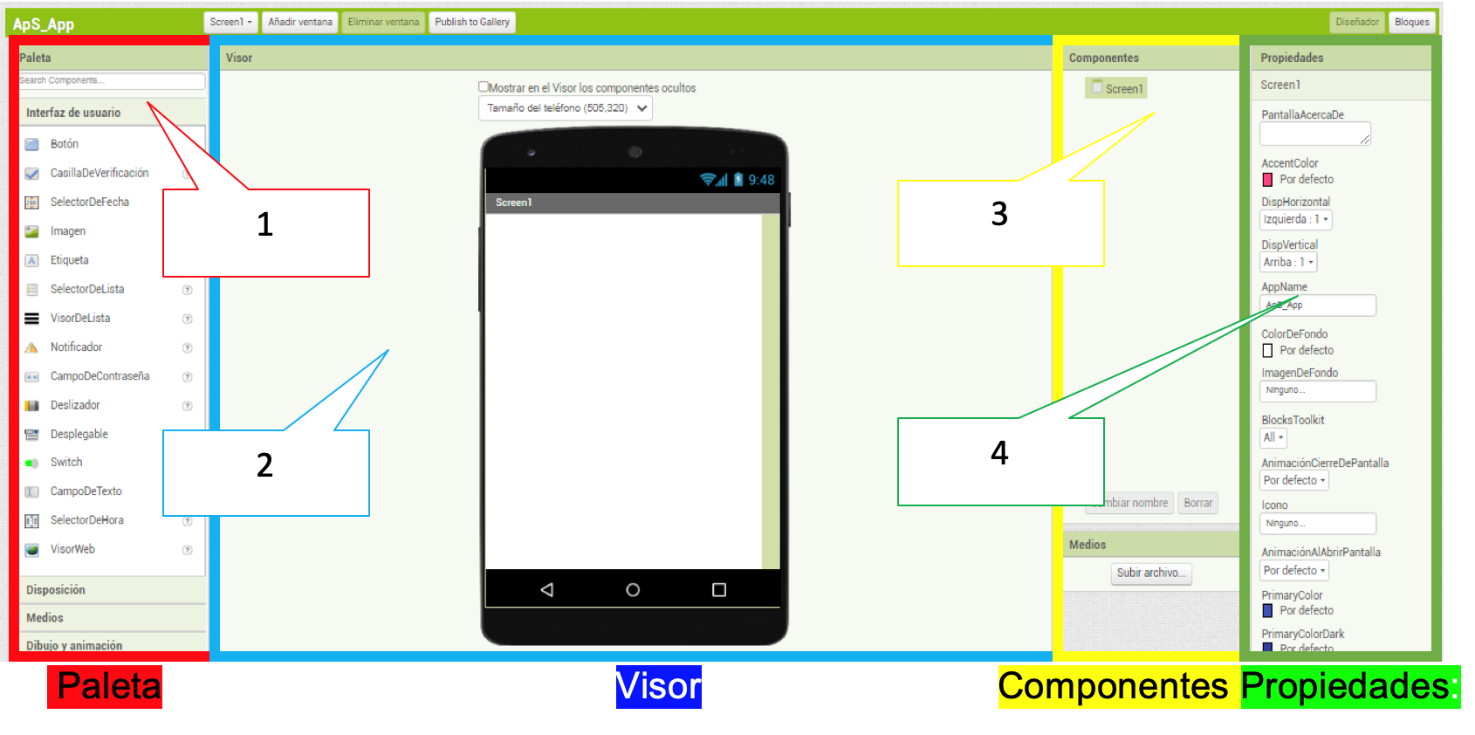
Pasos:

1. Ingresa a la página <https://appinventor.mit.edu/>
2. Crear una cuenta.
3. Se abre una ventana y debes cliquear en New Project:

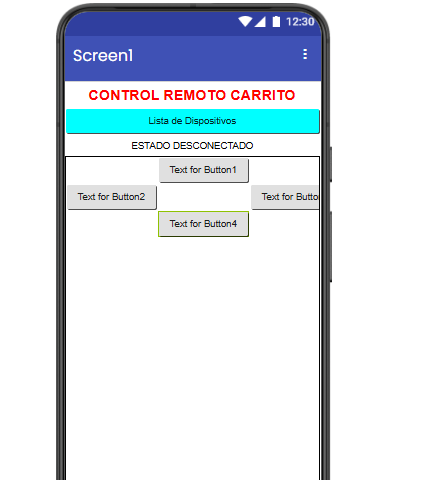


Dale un nombre al proyecto por ejemplo ControlCarritoDosLlantas.

1. Te va a aparecer una ventana como la siguiente:



1. **PALETA:**Agrupados en 12 categorías, ofrece multitud de elementos para integrarlos en nuestra aplicación arrastrándolos a la ventana Visor.
2. **VISOR**. Muestra una pantalla similar a un dispositivo móvil para ir alojando los elementos de la paleta y comprobar su disposición.
3. **COMPONENTES**.  Aparecen los elementos seleccionados en el visor en forma de árbol. Incluye una ventana inferior “Medios” para subir archivos multimedia.
4. **PROPIEDADES**. La cuarta columna permite la configuración de los componentes seleccionados anteriormente: color y alineación de texto, visible, etc. bien para la aplicación en general o bien para cada una de las pantallas de **nuestra** app.
5. En la sección **PALETA🡪User interface**, selecciona **Label** y arrástrala a la ventana **VISOR.** y en la ventana **PROPIEDADES:**
   1. **Width** selecciona **Fill parent**, esto hará que la etiqueta se vea en todo la pantalla.
   2. **Text** escribe CONTROL DE CARRITO.
   3. **Font Size**: 20
   4. **FontBold** Chequeado
   5. **Text Color**: Rojo
   6. **TextAlignment**: Center:1
6. En la sección **PALETA**, selecciona **ListPicker** y arrástrala a la ventana **VISOR.** y en la ventana **PROPIEDADES:**
   1. **Width** selecciona **Fill parent**, esto hará que la etiqueta se vea en todo la pantalla.
   2. **Text** escribe Lista de Dispositivos (al desplegar éste botón aparece la listas de dispositivos que se han emparejado previamente mediante bluethooth)
   3. **BackgroundColor** Azul
7. En la sección **PALETA**, selecciona **Label** y arrástrala a la ventana **VISOR.** y en la ventana **PROPIEDADES:**
   1. **Width** selecciona **Fill parent**, esto hará que la etiqueta se vea en todo la pantalla.
   2. **Text** escribe ESTADO DESCONECTADO.
   3. **Font Size**: 20
   4. **FontBold** Chequeado
   5. **Text Color**: Rojo
   6. **TextAlignment**: Center:1
8. En la sección **PALETA 🡪sección Layout**, selecciona **TableArrangement1** y arrástrala a la ventana **VISOR.** y en la ventana **PROPIEDADES:**
   1. Columns: 3
   2. Rows: 3
   3. Height: Fill parent
   4. Width: Fill parent
9. En la sección **PALETA 🡪User Interface**, selecciona **Button** y arrastra 4 botones a la ventana **VISOR** como lo muestra la imagen**:**

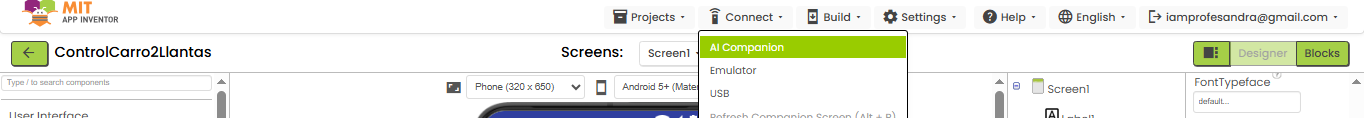


1. Por cada botón haz lo siguiente:

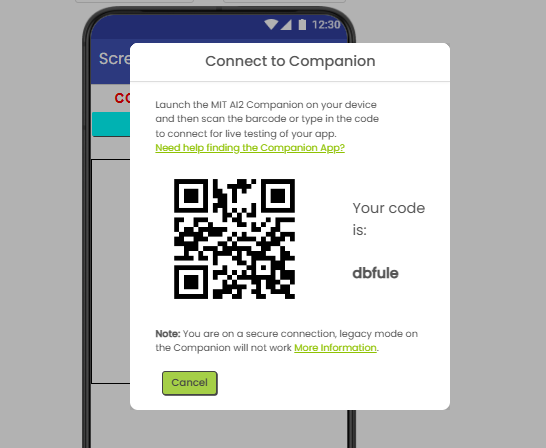
En la ventana **PROPIEDADES:**

* 1. **Text** copia y pega por cada botón las siguientes imágenes 🔼, 🔽, ◀️, ▶️
  2. **Font Size**: 60
  3. **Shape**: rounded

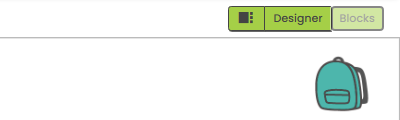
1. En la sección **PALETA 🡪Connectivity** selecciona BluetoothCliente y arrástralo dentro de la ventana, no debe quedar visible.
2. En la sección **PALETA 🡪Sensor** selecciona AccelerometerSensor y arrástralo dentro de la ventana, no debe quedar visible.
3. Para ver vista previa debes:
4. Descargar la App MIT AI2 Companion.
5. En el pc selecciona Connect 🡪AI Companion



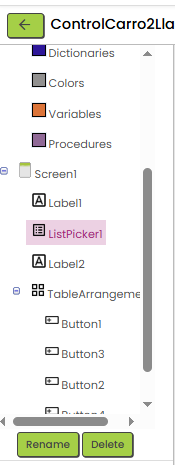
1. Debe aparecer algo como esto:



1. Escanea el código QR con la aplicación de App Inventor.
2. Ya debes poder ver en tu celular lo que estás viendo en tu pantalla, los cambios que realices se verán reflejados en tiempo real en tu celular.
3. Programación: Haz clic en la esquina superior derecha sobre el botón Blocks, como se muestra en la siguiente figura:

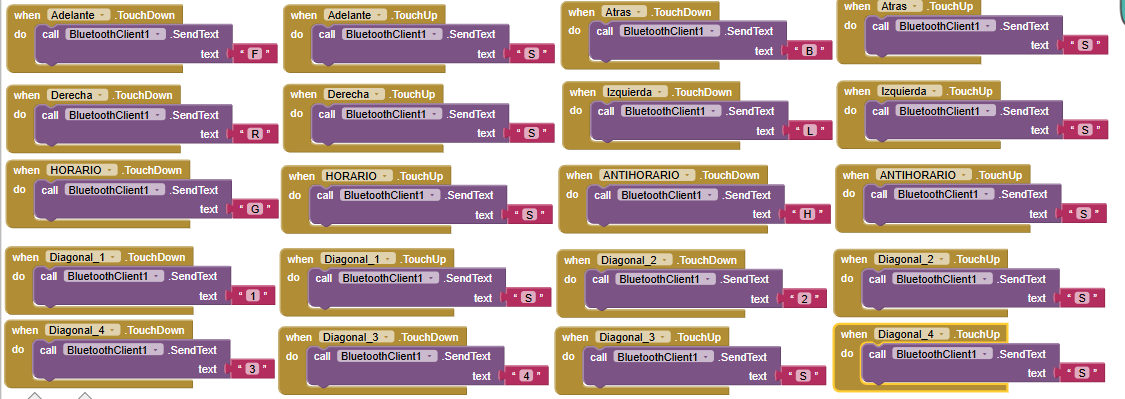


Busca en el panel Izquierdo ListPicker1

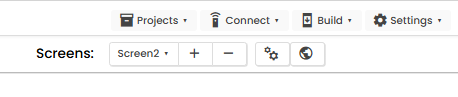


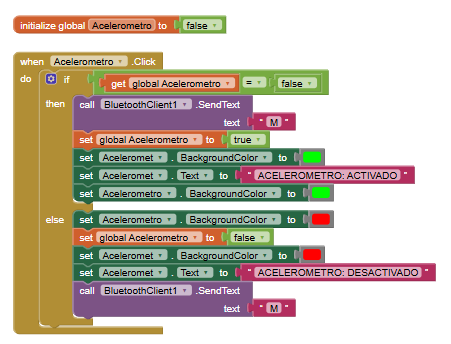
Programa los siguientes módulos:

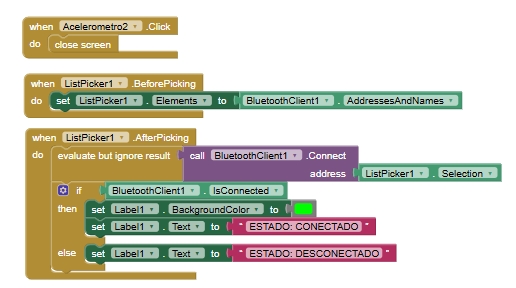


Programación de los Botones:

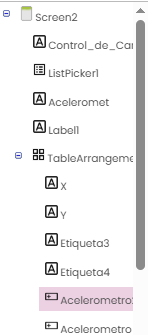
Ahora crea otro screens: como se muestra en la imágen

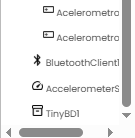


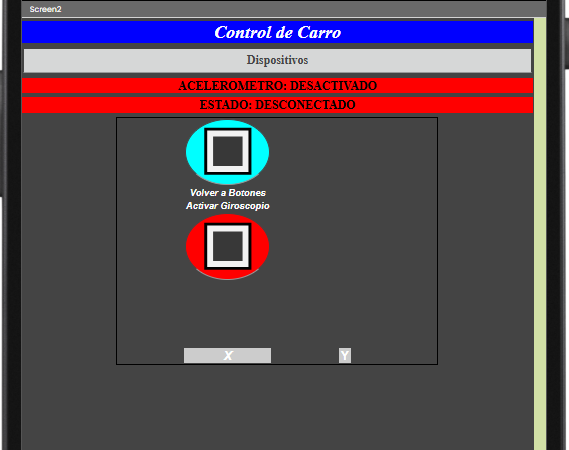




Y en Dessing debe quedar así:







**2. Alistamiento de la ESP32:**

**💻 Preparar Arduino IDE para ESP32**

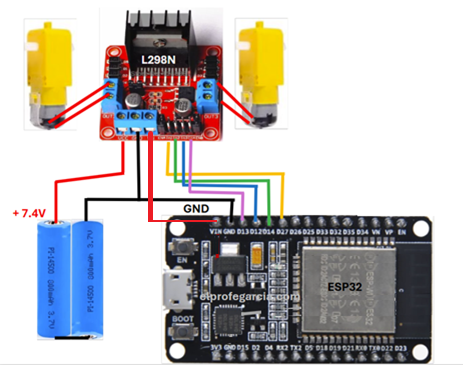
1. Instala Arduino IDE (versión 2.x recomendada).
2. Abre Archivo → Preferencias. En *URL adicionales de gestor de tarjetas* pega:

https://dl.espressif.com/dl/package\_esp32\_index.json

1. Ve a Herramientas → Placa → Gestor de tarjetas, busca ESP32 y haz clic en *Instalar*.
2. Selecciona Herramientas → Placa → ESP32 Arduino → ESP32 Dev Module.
3. Conecta tu ESP32 por USB, en Herramientas → Puerto elige el puerto COM correspondiente.
4. Copia el código que ya tienes. Guarda el archivo como carrito\_bluetooth.ino.
5. Haz clic en Verificar (✓) para compilar y en Subir (→) para cargarlo.
6. Abre el Monitor Serial (115200 baud) para comprobar que dice:
7. ESP32-Carro\_ByMike listo

**📱 Crear App en MIT App Inventor**

**. Esquema de conexión eléctrico**

****

**Nota: del GND de la L298N deben salir dos cables, uno para el polo negativo de la batería y otro para el GND de la ESP32**

* Cada **L298N** controla dos motores. cada L298N tiene 4 pines de control (IN1, IN2, IN3, IN4), que permiten manejar dos motores.  
  Como el carrito tiene dos módulos L298N, necesitamos 8 pines de control en el ESP32.
* **Según el esquema:**

Primer L298N (Motores delanteros, por ejemplo)

IN1 → D27

IN2 → D14

IN3 → D12

IN4 → D13

* **Eso significa que tenemos:**

Motor A → IN1 (D27), IN2 (D14)

Motor B → IN3 (D12), IN4 (D13)

* **🔧 Código:**

#include <BluetoothSerial.h>

BluetoothSerial SerialBT;

// Pines motores (solo 2 motores: derecho e izquierdo)

#define IN1 32 // Motor Derecho

#define IN2 33

#define IN3 25 // Motor Izquierdo

#define IN4 26

bool modoAcelerometro = false;

void setup() {

pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);

SerialBT.begin("ESP32-Carro\_ByMike");

Serial.begin(115200);

}

void loop() {

if (SerialBT.available()) {

char comando = SerialBT.read();

Serial.println(comando);

if (comando == 'M') {

modoAcelerometro = !modoAcelerometro;

Serial.println(modoAcelerometro ? "Modo ACELERÓMETRO" : "Modo MANUAL");

detenerMotores();

return;

}

if (modoAcelerometro) {

// Solo comandos simples en acelerómetro

switch (comando) {

case 'F': moverAdelante(); break;

case 'B': moverAtras(); break;

case 'L': girarIzquierda(); break;

case 'R': girarDerecha(); break;

case 'S': detenerMotores(); break;

default: detenerMotores(); break;

}

} else {

// Modo manual

switch (comando) {

case 'F': moverAdelante(); break;

case 'B': moverAtras(); break;

case 'L': girarIzquierda(); break;

case 'R': girarDerecha(); break;

case 'S': detenerMotores(); break;

default: detenerMotores(); break;

}

}

}

}

// ===== FUNCIONES DE MOVIMIENTO =====

void moverAdelante() {

adelante(IN1, IN2); // Motor derecho

adelante(IN3, IN4); // Motor izquierdo

}

void moverAtras() {

atras(IN1, IN2);

atras(IN3, IN4);

}

void girarDerecha() {

atras(IN1, IN2); // Motor derecho hacia atrás

adelante(IN3, IN4); // Motor izquierdo hacia adelante

}

void girarIzquierda() {

adelante(IN1, IN2); // Motor derecho hacia adelante

atras(IN3, IN4); // Motor izquierdo hacia atrás

}

void detenerMotores() {

detener(IN1, IN2);

detener(IN3, IN4);

}

// ===== FUNCIONES AUXILIARES =====

void adelante(int in1, int in2) {

digitalWrite(in1, HIGH); digitalWrite(in2, LOW);

}

void atras(int in1, int in2) {

digitalWrite(in1, LOW); digitalWrite(in2, HIGH);

}

void detener(int in1, int in2) {

digitalWrite(in1, LOW); digitalWrite(in2, LOW);

}