```
#include <Servo.h>

/*
  ========================================
  TANK STEERING WITH JOYSTICK
  ========================================

  WIRING INSTRUCTIONS
  -------------------

  JOYSTICK to ARDUINO:

    Joystick GND  →  Arduino pin A3
    Joystick +5V  →  Arduino pin A2
    Joystick X    →  Arduino pin A1  (left/right turning)
    Joystick Y    →  Arduino pin A0  (forward/back)

  Note: We are using A2 and A3 as power pins for the joystick.
  This is a trick - we set A2 to HIGH (+5V) and A3 to LOW (GND)
  so we don't need extra wires going to the Arduino's 5V and GND pins.

  RIGHT SERVO to ARDUINO:

    Signal wire (usually white or yellow)  →  Arduino pin 2
    Power wire  (usually red)              →  External power or Arduino 5V
    Ground wire (usually black or brown)   →  Arduino GND

  LEFT SERVO to ARDUINO:

    Signal wire (usually white or yellow)  →  Arduino pin 3
    Power wire  (usually red)              →  External power or Arduino 5V
    Ground wire (usually black or brown)   →  Arduino GND

  IMPORTANT NOTE ABOUT SERVO POWER:
  If both servos are twitching or behaving strangely, they may be
  drawing too much power from the Arduino. In that case, power the
  servos from an external battery pack and connect the grounds together.

  ========================================
  HOW TANK STEERING WORKS
  ========================================

  A tank (and this robot) steers by spinning its two wheels at
  different speeds or in different directions. Unlike a car, there
```

```
  is no steering wheel - turning is done entirely by the wheels.

    Both wheels forward          = drives straight ahead
    Both wheels backward         = drives straight back
    Right wheel forward only     = turns left
    Left wheel forward only      = turns right
    Right forward, Left backward = spins in place

  The Y axis of the joystick controls forward and backward.
  The X axis of the joystick controls left and right turning.
  Mixing the two together gives us smooth tank steering.

  ========================================
*/

// --- PIN ASSIGNMENTS ---

const int SERVO_R_PIN = 2;   // Right servo signal wire connects here
const int SERVO_L_PIN = 3;   // Left servo signal wire connects here

const int JOY_GND = A3;      // Joystick GND pin connects here
const int JOY_5V  = A2;      // Joystick +5V pin connects here
const int JOY_X   = A1;      // Joystick X axis (left/right turning)
const int JOY_Y   = A0;      // Joystick Y axis (forward/back)

// --- DEADZONE SETTINGS ---
// Real joysticks are not perfect. Even when you are not touching
// the joystick it rarely reads exactly 512 at center. This small
// error makes the robot drift or turn slightly when it should be still.
// The deadzone fixes this by ignoring small errors near center.
//
// *** TUNE ME ***
// Increase these values if the robot still drifts when centered.
// Decrease these values if the robot feels sluggish to respond.
// Start with 10 and adjust from there.

const int SPEED_DEADZONE = 10;  // Deadzone for forward/back
const int TURN_DEADZONE  = 10;  // Deadzone for left/right turning

// --- CREATE SERVO OBJECTS ---
// These lines create two servo objects we can control in our code

Servo servoR;   // This object controls the right wheel
Servo servoL;   // This object controls the left wheel
```

```
// ========================================
void setup()
{
  Serial.begin(9600);   // Start Serial Monitor so we can see values

  // Set up joystick power pins
  // We are using analog pins as power pins for the joystick
  // This saves us from needing extra wires to 5V and GND
  pinMode(JOY_5V,  OUTPUT);
  pinMode(JOY_GND, OUTPUT);
  digitalWrite(JOY_5V,  HIGH);  // A2 acts as +5V for the joystick
  digitalWrite(JOY_GND, LOW);   // A3 acts as GND for the joystick

  // Connect the servo objects to their pins
  servoR.attach(SERVO_R_PIN);
  servoL.attach(SERVO_L_PIN);

  // Stop both wheels at startup
  // For continuous rotation servos:
  //   write(93)  = stop  (may need small adjustment per servo)
  //   write(0)   = full speed one direction
  //   write(180) = full speed other direction
  //
  // *** TUNE ME ***
  // If your robot creeps forward or backward when the joystick
  // is centered, change 93 to a slightly higher or lower number
  // until the wheels actually stop. Try 90, 91, 92, 94, 95.
  servoR.write(93);
  servoL.write(93);

  delay(1000);   // Give servos time to settle before we start
}

// ========================================
void loop()
{
  // --- READ THE JOYSTICK ---
  // analogRead gives us a number from 0 to 1023
  // When the joystick is centered it reads around 512
  // Pushed all the way forward or back it reads near 0 or 1023
  int joyY = analogRead(JOY_Y);   // Forward/back
  int joyX = analogRead(JOY_X);   // Left/right
```

```
// --- CONVERT JOYSTICK READINGS TO SERVO RANGE ---
// map() converts a number from one range to another range.
// Here we convert the joystick range (0 to 1023)
// into the servo range (0 to 180).
// Joystick center (~512) becomes servo center (~90).
int speed = map(joyY, 0, 1023, 0, 180);

// Turn is mapped to -90 to 90 so that:
//   joystick left  → negative number → robot turns left
//   joystick right → positive number → robot turns right
//   joystick center → 0              → no turning
int turn = map(joyX, 0, 1023, -90, 90);

// *** TRY ME ***
// If your robot turns in the wrong direction, change the
// turn line above to flip the range like this:
//   int turn = map(joyX, 0, 1023, 90, -90);

// --- APPLY DEADZONES ---
// abs() gives the absolute value so we can check both
// positive and negative numbers with one comparison.
//
// Speed deadzone:
// If speed is within SPEED_DEADZONE of 90 (stopped),
// force it to exactly 90 so the robot does not creep.
//
// Example: speed = 86 (joystick slightly off center)
//   abs(86 - 90) = 4, which is less than 10
//   so speed gets set to 90 → robot stays still
if (abs(speed - 90) < SPEED_DEADZONE)
{
  speed = 90;
}

// Turn deadzone:
// If turn is within TURN_DEADZONE of zero,
// force it to exactly 0 so the robot drives straight.
//
// Example: turn = -2 (joystick slightly off center)
//   abs(-2) = 2, which is less than 10
//   so turn gets set to 0 → robot drives straight
if (abs(turn) < TURN_DEADZONE)
{
  turn = 0;
```

```
}

// --- HANDLE OPPOSITE-FACING SERVOS ---
// The two servos are mounted facing opposite directions on the robot.
// If we send them the same value, one wheel goes forward
// while the other goes backward - spinning the robot in place.
// To make both wheels push the same direction we FLIP one side
// by subtracting from 180.
//
// We also ADD the turn value to one wheel and SUBTRACT it from
// the other. This speeds up one side and slows down the other,
// which steers the robot while it is driving.
//
//   rightSpeed = speed + turn
//   leftSpeed  = (180 - speed) - turn
//
// Example: driving forward (speed=0) and turning right (turn=45)
//   rightSpeed = 0  + 45 = 45   (right wheel slows down)
//   leftSpeed  = 180 - 45 = 135 (left wheel speeds up)
//   result: robot curves to the right
int rightSpeed = speed + turn;
int leftSpeed  = (180 - speed) - turn;

// --- KEEP VALUES INSIDE SAFE RANGE ---
// constrain() makes sure a value never goes below 0 or above 180.
// Without this, adding turn to speed could produce a number like
// 220 or -15, which would confuse the servo.
rightSpeed = constrain(rightSpeed, 0, 180);
leftSpeed  = constrain(leftSpeed,  0, 180);

// *** TRY ME ***
// If your robot drives backward when you push the joystick forward,
// swap the pin numbers at the top of the code:
//   SERVO_R_PIN = 3
//   SERVO_L_PIN = 2
// OR swap the rightSpeed and leftSpeed lines below:
//   servoR.write(leftSpeed);
//   servoL.write(rightSpeed);

// Send the speed values to each servo
servoR.write(rightSpeed);
servoL.write(leftSpeed);

// --- PRINT VALUES TO SERIAL MONITOR ---
```

```
  // Open Serial Monitor (magnifying glass icon, top right)
  // to see these values while the robot is running.
  // This is very helpful for troubleshooting. If the robot
  // is not behaving as expected, check these values first.
  Serial.print("Y: ");
  Serial.print(joyY);
  Serial.print("  X: ");
  Serial.print(joyX);
  Serial.print("  Speed: ");
  Serial.print(speed);
  Serial.print("  Turn: ");
  Serial.print(turn);
  Serial.print("  Right: ");
  Serial.print(rightSpeed);
  Serial.print("  Left: ");
  Serial.println(leftSpeed);

  delay(50);   // Short pause before reading joystick again
               // 50 milliseconds = 20 readings per second
               // which is smooth enough for driving
}

/*
  ========================================
  TROUBLESHOOTING GUIDE
  ========================================

  Robot creeps when joystick is centered:
    → Adjust the stop value in setup() from 93 up or down
      until the wheels stop. Each servo is slightly different.
    → Also try increasing SPEED_DEADZONE from 10 to 15 or 20.

  Robot drifts slightly to one side when driving straight:
    → Increase TURN_DEADZONE from 10 to 15 or 20.

  Robot is slow to respond when you push the joystick:
    → Decrease SPEED_DEADZONE or TURN_DEADZONE from 10 to 5.

  Robot drives backward when joystick pushed forward:
    → Swap SERVO_R_PIN and SERVO_L_PIN values at the top,
      OR swap the servoR.write and servoL.write lines.

  Robot turns the wrong way:
    → Change the turn map() range from (-90, 90) to (90, -90)
```

```
  One wheel not moving:
    → Check the signal wire connection for that servo
    → Check Serial Monitor to confirm the value being sent
      is not stuck at 93 (stop)

  Both wheels twitching or stuttering:
    → The Arduino may not be supplying enough power to both servos
    → Try powering the servos from a separate battery pack
    → Make sure all GND wires share a common ground

  ========================================
*/
```