# Simple RC Mecanum Wheels Robot Wifi Arduino

**Materials:**

### **Robot/ Car Body**

* Arduino Nano x1
* nRF24L01 module x1
* L293D motor driver x2
* Breadboard x1
* N20 motors wif mount (6V 300rpm) x4
* 7.4V 900mah Li-Po Battery x1
* Mecanum wheels x4

### **Joystick Controller**

* Arduino Pro-Mini x1
* USB to TTL dongle x1 (for programming Arduino Pro-Mini)
* nRF24L01 module x1
* Joystick module x1
* Breadboard x1
* Tactile switch x2
* AA 2 cells battery box x1
* 2 AA batteries

**Circuit :**



**Codes: Joystick**

|  |
| --- |
| #include <SPI.h>#include <printf.h>#include <RF24.h>#if 1 #define DEBUG\_P(...) Serial.print(\_\_VA\_ARGS\_\_) #define DEBUG\_PLN(...) Serial.println(\_\_VA\_ARGS\_\_)#else #define DEBUG\_P(...) #define DEBUG\_PLN(...)#endif#define CE\_PIN A0 // The pins to be used for CE and SN#define CSN\_PIN 10#define JOYSTICK\_X A1 // The Joystick potentiometers connected to Arduino Analog inputs#define JOYSTICK\_Y A2#define JOYSTICK\_SW A3 // The Joystick push-down switch, will be used as a Digital input#define SW\_1 0 //The first tactile switch: One side to DPIN 0, the other side to GND pin#define SW\_2 3 //The second tactile switch: One side to DPIN 3, the other side to DPIN 5 which config as GND#define D\_GND 5 RF24 RF\_Joystick(CE\_PIN, CSN\_PIN);byte addresses[][6] = {"1Node", "2Node"}; // These will be the names of the "Pipes"unsigned long timeNow; // Used to grab the current time, calculate delaysunsigned long started\_waiting\_at;boolean timeout; // Timeout? True or Falsestruct dataStruct { unsigned long \_micros; // to save response times int Xposition; // The Joystick position values int Yposition; bool switch0; // The Joystick push-down switch bool switch1; // Rotate Left bool switch2; // Rotate Right} myData;  void(\* resetFunc) (void) = 0; //declare reset function @ address 0void setup() { Serial.begin(115200);  pinMode(JOYSTICK\_X, INPUT); pinMode(JOYSTICK\_Y, INPUT); pinMode(JOYSTICK\_SW, INPUT\_PULLUP);  pinMode(SW\_1, INPUT\_PULLUP);  pinMode(SW\_2, INPUT\_PULLUP);  pinMode(D\_GND, OUTPUT);  digitalWrite(D\_GND, LOW); printf\_begin(); // Needed for "printDetails" Takes up some memory RF\_Joystick.begin(); // Initialize the nRF24L01 Radio RF\_Joystick.setChannel(108); // Above most WiFi frequencies RF\_Joystick.setDataRate(RF24\_250KBPS); // Fast enough.. Better range RF\_Joystick.setPALevel(RF24\_PA\_MIN); //If RF not working, try set this to RF24\_PA\_MIN RF\_Joystick.openWritingPipe(addresses[0]); RF\_Joystick.openReadingPipe(1, addresses[1]); RF\_Joystick.printDetails(); //Uncomment to show LOTS of debugging information if(!RF\_Joystick.isChipConnected()) resetFunc(); DEBUG\_PLN(F("Start"));}void loop() { RF\_Joystick.stopListening();  myData.Xposition = analogRead(JOYSTICK\_X); myData.Yposition = analogRead(JOYSTICK\_Y); myData.switch0 = !digitalRead(JOYSTICK\_SW); // Invert the pulldown switch myData.switch1 = !digitalRead(SW\_1); // Invert the pulldown switch myData.switch2 = !digitalRead(SW\_2); // Invert the pulldown switch myData.\_micros = micros(); // Send back for timing DEBUG\_P(myData.Xposition); DEBUG\_P(","); DEBUG\_P(myData.Yposition); DEBUG\_P(","); DEBUG\_P(myData.switch0); DEBUG\_P(","); DEBUG\_P(myData.switch1); DEBUG\_P(","); DEBUG\_PLN(myData.switch2);  DEBUG\_P(F("Now sending - ")); int now = millis(); if (!RF\_Joystick.write( &myData, sizeof(myData))) { // Send data, checking for error ("!" means NOT) DEBUG\_PLN(F("Transmit failed ")); }  RF\_Joystick.startListening(); // Now, continue listening  started\_waiting\_at = micros(); // timeout period, get the current microseconds timeout = false; // variable to indicate if a response was received or not  while ( ! RF\_Joystick.available() ) { // While nothing is received if (micros() - started\_waiting\_at > 200000 ) { // If waited longer than 200ms, indicate timeout and exit while loop timeout = true; break; } }  if ( timeout ) { // Describe the results DEBUG\_PLN(F("Response timed out - no Acknowledge.")); } else { // Grab the response, compare, and send to Serial Monitor RF\_Joystick.read( &myData, sizeof(myData) ); timeNow = micros();  // Show it DEBUG\_P(F("Sent ")); DEBUG\_P(timeNow); DEBUG\_P(F(", Got response ")); DEBUG\_P(myData.\_micros); DEBUG\_P(F(", Round-trip delay ")); DEBUG\_P(timeNow - myData.\_micros); DEBUG\_PLN(F(" microseconds ")); } // Send again after delay. When working OK, change to something like 100 delay(100); if(!RF\_Joystick.isChipConnected()) resetFunc();} |

**Codes: Mecanum**

|  |
| --- |
| #include <math.h>#include <SPI.h>#include <printf.h>#include <RF24.h>#define ANALOG\_SPEED 1 //1 for Analog Speed; 0 for Full Speed#define MIN\_MOTOR\_SPD 100 //Minium Speed(PWM) for motors#define MAX\_MOTOR\_SPD 255 //Maximum Speed(PWM) for motors#if 1 #define DEBUG\_P(...) Serial.print(\_\_VA\_ARGS\_\_) #define DEBUG\_PLN(...) Serial.println(\_\_VA\_ARGS\_\_)#else #define DEBUG\_P(...) #define DEBUG\_PLN(...)#endif#define CE\_PIN A0 // The pins to be used for CE and SN#define CSN\_PIN 10RF24 RF\_Mecan (CE\_PIN,CSN\_PIN); //create RF24 object called transmitbyte addresses[][6] = {"1Node", "2Node"}; // These will be the names of the "Pipes"struct dataStruct { unsigned long \_micros; // to save response times int Xposition; // The Joystick position values int Yposition; bool switch0; // The Joystick push-down switch bool switch1; // Rotate Left bool switch2; // Rotate Right} myData; #define M\_STOP 0#define M\_FORWARD 1#define M\_BACKWARD 2#define CMD\_Stop 0#define CMD\_moveForward 1#define CMD\_moveBackward 2#define CMD\_moveLeft 3#define CMD\_moveRight 4#define CMD\_moveLeftForward 5#define CMD\_moveRightForward 6#define CMD\_moveLeftBackward 7#define CMD\_moveRightBackward 8#define CMD\_rotateLeft 9#define CMD\_rotateRight 10#define CMD\_TIMEOUT\_MS 1000byte curr\_cmd = CMD\_Stop, last\_cmd = CMD\_Stop, curr\_spd = 255, last\_spd = 255;int cmd\_last\_update = 0, timeout\_cnt = 0;typedef enum{ E\_Center = 0, E\_Up, E\_Down, E\_Left, E\_Right, E\_UpLeft, E\_UpRight, E\_DownLeft, E\_DownRight} J\_DIR;J\_DIR joystickDIR;#define JOYSTICK\_CENTER\_X 512 //ADC reading when X is centered#define JOYSTICK\_CENTER\_Y 512 //ADC reading when Y is centered#define JOYSTICK\_CENTER\_ADC\_TOL 100 //ADC reading tolerance for center detection#define JOYSTICK\_MAX\_RADIUS sqrt(2\*pow(JOYSTICK\_CENTER\_X,2)) //Max Radius#define JOYSTICK\_MIN\_RADIUS sqrt(2\*pow(JOYSTICK\_CENTER\_ADC\_TOL,2)) //Min Radiusclass MOTOR { private: byte IN1, IN2, EN; public: MOTOR(byte in1, byte in2, byte en); void set\_speed(byte en, byte spd);};MOTOR::MOTOR(byte in1, byte in2, byte en) { IN1 = in1; IN2 = in2; EN = en; pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT); pinMode(EN, OUTPUT);}void MOTOR::set\_speed (byte dir, byte spd) { if(spd<0||spd>255) return; switch(dir) { case M\_FORWARD: digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW); break; case M\_BACKWARD: digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH); break; case M\_STOP: default: digitalWrite(IN1, LOW); digitalWrite(IN2, LOW); break; } analogWrite(EN,spd);}MOTOR M\_FrontLeft(2, 7, 3), //Setup the Pins for four motors M\_FrontRight(8, 4, 5), M\_RearLeft(A4, A3, 9), M\_RearRight(A2, A1, 6); void(\* resetFunc) (void) = 0; //declare reset function @ address 0void setup() { Serial.begin(115200); printf\_begin(); // Needed for "printDetails" Takes up some memory RF\_Mecan.begin(); // Initialize the nRF24L01 Radio RF\_Mecan.setChannel(108); // 2.508 Ghz - Above most Wifi Channels RF\_Mecan.setDataRate(RF24\_250KBPS); // Fast enough.. Better range RF\_Mecan.setPALevel(RF24\_PA\_MIN); RF\_Mecan.openWritingPipe(addresses[1]); RF\_Mecan.openReadingPipe(1, addresses[0]); RF\_Mecan.printDetails(); //Uncomment to show LOTS of debugging information RF\_Mecan.startListening(); Stop(); delay(500); if(!RF\_Mecan.isChipConnected()) resetFunc(); DEBUG\_PLN(F("Start"));}void loop() { int now = millis(); if (RF\_Mecan.available()) { timeout\_cnt = now; while (RF\_Mecan.available()) // While there is data ready to be retrieved from the receive pipe { RF\_Mecan.read( &myData, sizeof(myData) ); // Get the data } RF\_Mecan.stopListening(); // First, stop listening so we can transmit RF\_Mecan.write( &myData, sizeof(myData) ); // Send the received data back. RF\_Mecan.startListening(); // Now, resume listening so we catch the next packets. DEBUG\_P(F("Packet Received - Sent response ")); // Print the received packet data DEBUG\_P(myData.\_micros); DEBUG\_P(F("uS X= ")); DEBUG\_P(myData.Xposition); DEBUG\_P(F(" Y= ")); DEBUG\_P(myData.Yposition); if ( myData.switch0 == 1) DEBUG\_P(F(" Switch0 ON")); else DEBUG\_P(F(" Switch0 OFF")); if ( myData.switch1 == 1) DEBUG\_P(F(" Switch1 ON")); else DEBUG\_P(F(" Switch1 OFF"));  if ( myData.switch2 == 1) DEBUG\_PLN(F(" Switch2 ON")); else DEBUG\_PLN(F(" Switch2 OFF")); joystickDIR = xyToDir(myData.Xposition,myData.Yposition); if(joystickDIR==E\_Center) { if(myData.switch1 == 1 && myData.switch2 != 1) { curr\_cmd = CMD\_rotateLeft; curr\_spd = MAX\_MOTOR\_SPD; } else if(myData.switch2 == 1 && myData.switch1 != 1) { curr\_cmd = CMD\_rotateRight;  curr\_spd = MAX\_MOTOR\_SPD; } else curr\_cmd = CMD\_Stop; } else  curr\_cmd = joystickDIR; } else if(now-timeout\_cnt>CMD\_TIMEOUT\_MS) //If nothing is received after CMD\_TIMEOUT\_MS, set CMD to CMD\_Stop { DEBUG\_PLN(F("Timeout")); timeout\_cnt = now; curr\_cmd = CMD\_Stop; Stop(); resetFunc(); //call reset } if(!RF\_Mecan.isChipConnected()) { curr\_cmd = CMD\_Stop; Stop(); resetFunc(); } cmdHandle();}J\_DIR xyToDir(int xo, int yo) { //Translate XY coordinate to Joystick Driection long radius; float angle; long x = xo - JOYSTICK\_CENTER\_X; long y = yo - JOYSTICK\_CENTER\_Y; x = 0-x; if(x == 0 && y == 0 ){ radius = 0; angle = 0; } else { radius = sqrt(y\*y+x\*x); angle = atan2(y,x)\*180/PI; } if(angle<0) angle+=360; DEBUG\_P("x:"); DEBUG\_P(x); DEBUG\_P(", y:"); DEBUG\_PLN(y); DEBUG\_P("angle:"); DEBUG\_P(angle); DEBUG\_P(", r:"); DEBUG\_PLN(radius); if(radius<sqrt(2\*pow(JOYSTICK\_CENTER\_ADC\_TOL,2))) return E\_Center; if(ANALOG\_SPEED) { curr\_spd = map(radius,JOYSTICK\_MIN\_RADIUS,JOYSTICK\_MAX\_RADIUS,MIN\_MOTOR\_SPD,255); DEBUG\_P("spd:"); DEBUG\_PLN(curr\_spd); } if(angle<0+22.5||angle>=360-22.5) return E\_Right; if(angle<45+22.5) return E\_UpRight; if(angle<90+22.5) return E\_Up; if(angle<135+22.5) return E\_UpLeft; if(angle<180+22.5) return E\_Left; if(angle<225+22.5) return E\_DownLeft; if(angle<270+22.5) return E\_Down; if(angle<315+22.5) return E\_DownRight;}void cmdHandle() { if(curr\_cmd==last\_cmd && curr\_spd==last\_spd) return; switch(curr\_cmd) { case CMD\_Stop: Stop(); DEBUG\_PLN("Stop"); break; case CMD\_moveForward: moveForward(curr\_spd); DEBUG\_PLN("moveForward"); break; case CMD\_moveBackward: moveBackward(curr\_spd); DEBUG\_PLN("moveBackward"); break; case CMD\_moveLeft: moveLeft(curr\_spd); DEBUG\_PLN("moveLeft"); break; case CMD\_moveRight: moveRight(curr\_spd); DEBUG\_PLN("moveRight"); break; case CMD\_moveLeftForward: moveLeftForward(curr\_spd); DEBUG\_PLN("moveLeftForward"); break; case CMD\_moveRightForward: moveRightForward(curr\_spd); DEBUG\_PLN("moveRightForward"); break; case CMD\_moveLeftBackward: moveLeftBackward(curr\_spd); DEBUG\_PLN("moveLeftBackward"); break; case CMD\_moveRightBackward: moveRightBackward(curr\_spd); DEBUG\_PLN("moveRightBackward"); break; case CMD\_rotateLeft: rotateLeft(curr\_spd); DEBUG\_PLN("rotateLeft"); break; case CMD\_rotateRight: rotateRight(curr\_spd); DEBUG\_PLN("rotateRight"); break; } last\_cmd = curr\_cmd;  last\_spd = curr\_spd; }void Stop() { M\_FrontLeft.set\_speed(M\_STOP,0); M\_FrontRight.set\_speed(M\_STOP,0); M\_RearLeft.set\_speed(M\_STOP,0); M\_RearRight.set\_speed(M\_STOP,0);}void moveForward(byte spd) { M\_FrontLeft.set\_speed(M\_FORWARD,spd); M\_FrontRight.set\_speed(M\_FORWARD,spd); M\_RearLeft.set\_speed(M\_FORWARD,spd); M\_RearRight.set\_speed(M\_FORWARD,spd);}void moveBackward(byte spd) { M\_FrontLeft.set\_speed(M\_BACKWARD,spd); M\_FrontRight.set\_speed(M\_BACKWARD,spd); M\_RearLeft.set\_speed(M\_BACKWARD,spd); M\_RearRight.set\_speed(M\_BACKWARD,spd);}void moveLeft(byte spd) { M\_FrontLeft.set\_speed(M\_BACKWARD,spd); M\_FrontRight.set\_speed(M\_FORWARD,spd); M\_RearLeft.set\_speed(M\_FORWARD,spd); M\_RearRight.set\_speed(M\_BACKWARD,spd);}void moveRight(byte spd) { M\_FrontLeft.set\_speed(M\_FORWARD,spd); M\_FrontRight.set\_speed(M\_BACKWARD,spd); M\_RearLeft.set\_speed(M\_BACKWARD,spd); M\_RearRight.set\_speed(M\_FORWARD,spd);}void moveLeftForward(byte spd) { M\_FrontLeft.set\_speed(M\_STOP,0); M\_FrontRight.set\_speed(M\_FORWARD,spd); M\_RearLeft.set\_speed(M\_FORWARD,spd); M\_RearRight.set\_speed(M\_STOP,0);}void moveRightForward(byte spd) { M\_FrontLeft.set\_speed(M\_FORWARD,spd); M\_FrontRight.set\_speed(M\_STOP,0); M\_RearLeft.set\_speed(M\_STOP,0); M\_RearRight.set\_speed(M\_FORWARD,spd);}void moveLeftBackward(byte spd) { M\_FrontLeft.set\_speed(M\_BACKWARD,spd); M\_FrontRight.set\_speed(M\_STOP,0); M\_RearLeft.set\_speed(M\_STOP,0); M\_RearRight.set\_speed(M\_BACKWARD,spd);}void moveRightBackward(byte spd) { M\_FrontLeft.set\_speed(M\_STOP,0); M\_FrontRight.set\_speed(M\_BACKWARD,spd); M\_RearLeft.set\_speed(M\_BACKWARD,spd); M\_RearRight.set\_speed(M\_STOP,0);}void rotateLeft(byte spd) { M\_FrontLeft.set\_speed(M\_BACKWARD,spd); M\_FrontRight.set\_speed(M\_FORWARD,spd); M\_RearLeft.set\_speed(M\_BACKWARD,spd); M\_RearRight.set\_speed(M\_FORWARD,spd);}void rotateRight(byte spd) { M\_FrontLeft.set\_speed(M\_FORWARD,spd); M\_FrontRight.set\_speed(M\_BACKWARD,spd); M\_RearLeft.set\_speed(M\_FORWARD,spd); M\_RearRight.set\_speed(M\_BACKWARD,spd);} |