Cody Hum
Alberto Rosas
Dang Tran
CMPE 344
3 May 2024

DIY Hand Gestures Controlled Car Using Arduino

Semester Project Report

In our DIY RC car project, we have assembled various parts to make an excellent and versatile vehicle. The sturdy Robot Smart Car Chassis forms the base, keeping it stable. We use an 18650 rechargeable battery for power; we store the battery securely with an 18650 Battery Holder. The L298N Motor Drive Controller Board drives the motors precisely. We also built the transmitter, which uses a wireless NRF24L01+ 2.4GHz module for communication and an MPU6050 3-axis accelerometer to read the acceleration and position of the transmitter. The UNO R3 Board ATmega328P and Mini Nano V3.0 ATmega328P handle the brain work. We used other small parts like wires, buttons, LEDs, and resistors to connect the more significant pieces. All these pieces work together to make a smart, fun DIY RC car that gives us the challenge to put together.

Soldering wires in a project can be challenging, demanding precision and patience to establish reliable connections. However, these challenges also present valuable learning opportunities. A key obstacle involves ensuring the proper preparation of wire ends and soldering surfaces, as any debris or oxidation may impede solder adhesion. Managing the soldering iron's heat to prevent wire overheating or component damage is essential, particularly with delicate parts. Attaining robust and enduring solder joints requires skillful control of solder flow and application, a task that can be daunting for novices. Furthermore, soldering within tight spaces or intricate circuits adds complexity, necessitating steady hands and meticulous attention to detail. Despite these hurdles, mastering wire soldering not only enhances project quality and reliability but also cultivates invaluable skills in precision craftsmanship and problem-solving.

While the overall expenses for our project were considerably high, we implemented several cost-saving strategies. The primary contributors to its high cost were the DIY Robot Smart Car Chassis, the 18650 rechargeable battery, and the NRF24L01+ 2.4GHz Wireless RF module, each priced between ten to twenty dollars. However, we offset these expenses by employing resourcefulness and creativity to acquire other essential components at no cost. By delving into the reservoir of electronic components tucked away in our household, we combed through forgotten drawers and storage spaces, unearthing items such as the Mini Nano V3.0 ATmega328P, wires, a button, LED, a 1k resistor, and a few AA batteries. This scavenging effort allowed us to repurpose these found materials for our projects, exemplifying the wealth of untapped potential within our homes, ripe for innovative exploration.

In our RC car project, Arduino/C++ played a pivotal role in programming the vehicle to respond to specific motions for control. We integrated an accelerometer into our design, reading the acceleration and position and sending the data to the Arduino Mini Nano to interpret motion signals accurately. Using C++, we coded the car's microcontroller to interact seamlessly with this sensor, ensuring precise movement detection. By analyzing accelerometer data through Arduino IDE, we translated it into commands for the car's motors. Four basic functions were programmed forward, backward, left, and right turns, similar to tank steering. After programming the basic functions we also implemented a secret mode that added preprogrammed functions on the RC car. The transmitter sends over a Mode 1 instead of Mode 0 when the button is pressed and held. Unfortunately, this is where we ran into our biggest obstacle. The receiver programming looped reading the data sent by the transmitter and then executing the correct command. However, after the last command was received the loop kept executing it. Our first idea to correct the problem was to send an all-0 command to stop all moments. This did not work because when doing a preprogrammed action the action ended after the transmitter sent the 0 command so it reads no new signal and kept going with the previous command. Our second attempt involved using the nRF24 library command that checks if the radio is available. Adding logic that checks if radio is not available and if the command is 0 as given by the transmitter works in stopping the commands from looping after being transmitted only once. This facilitated intuitive control methods such as steering by tilting the controller or accelerating by shaking it. The versatility and efficiency of Arduino have facilitated swift development and refinement, enabling us to fine-tune the car's responsiveness effectively. Consequently, our RC car emerged as a dynamic and interactive creation, showcasing the power of C++ in robotics projects.

Participating in the DIY RC car project provided practical skills that are useful in various careers. Building and programming the vehicle improved our problem-solving abilities, which was helpful in engineering roles like product design and manufacturing. Understanding electronics and mechanics is beneficial for jobs involving system optimization. Proficiency in programming, especially in languages like C++, is valuable in software development. Resourcefulness and adaptability in material sourcing are advantageous for project management and entrepreneurship. Collaboration during the project enhanced teamwork and communication skills, essential in any professional setting requiring cooperation. Overall, skills learned in the project are applicable across different fields, setting participants up for success in engineering, software development, project management, and entrepreneurship.

Our DIY RC car project was a thrilling exploration of teamwork, creativity, and problem-solving. From the initial brainstorming sessions to the final assembly, each step was a testament to our collective ingenuity and determination. As we navigated challenges like perfecting soldering techniques and debugging complex code, we discovered new depths of resilience and resourcefulness within ourselves. The integration of Arduino and C++ empowered us to create a sophisticated vehicle capable of responding to hand gestures, elevating our project to new heights of interactivity and innovation. Moreover, our commitment to cost-saving measures underscored our ability to think creatively and adapt to constraints, demonstrating the practical skills and entrepreneurial mindset that will serve us well in future endeavors. With these skills we can add modules to the RC car such as a camera, and depth sensor. Ultimately, our DIY RC car project not only provided us with a tangible showcase of our technical abilities but also fostered a sense of camaraderie and excitement as we worked together to bring our vision to life.

Transmitter Source Code

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include "Wire.h"
#include "MPU6050.h"

#define CSN_GPIO    8
#define CE_GPIO     7

// Hardware configuration
RF24 radio(CE_GPIO, CSN_GPIO);                           // Set up
nRF24L01 radio on SPI bus plus pins 7 & 8
MPU6050 accelgyro;

int16_t ax, ay, az; // x-axis and y-axis values, for this application
z-axis is not needed.
int16_t gx, gy, gz; // these values are not used, just declaring to pass
in the function.

#define TX_ENABLE_KEY     2 //Define led
#define TX_LED            3

const byte Address[6] = "00001";
int Pot_Val_Y = 0,Pot_Val_X = 0, Up_key = 0, Dn_key = 0, Left_key = 0,
Right_key = 0;
unsigned char Tx_command = 0,Speed_index = 0,Tx_Enable_Flag =
0,TX_Key_Pressed = 0;
unsigned char Tx_Array[3];
int Tx_count = 0;
```

```
void setup() {
  Serial.begin(115200);
  pinMode(TX_ENABLE_KEY, INPUT_PULLUP);
  pinMode(TX_LED, OUTPUT);

  radio.begin();
  radio.openWritingPipe(Address);
  radio.setPALevel(RF24_PA_MAX);
  radio.stopListening();
  radio.write(&Tx_command, sizeof(Tx_command));

  Wire.begin();
  Serial.println("Initializing I2C devices...");
  accelgyro.initialize();

  // verify connection
  Serial.println("Testing device connections...");
  Serial.println(accelgyro.testConnection() ? "MPU6050 connection
successful" : "MPU6050 connection failed");

  Tx_Array[0] = 0;
  Tx_Array[1] = 0;
  Tx_Array[2] = 0;

}

void loop()
{
  accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // we are getting
all 6 value from MPU6050 i.e. 3 Accelerometer values and 3 Gyro values
viz. ax,ay,az,gx,gy and gz.
  if(!(PIND & 0x04)&&(TX_Key_Pressed==0))
  {
      TX_Key_Pressed = 1;
      if(Tx_Enable_Flag==0)
      {
        Tx_Enable_Flag = 1;
        PORTD |= 0x08;
        //Serial.println("A");
```

```
    }
    else
    {
      Tx_Enable_Flag = 0;
      PORTD &= 0xF7;
      //Serial.println("B");
      Tx_Array[0] = 0;
      Tx_Array[1] = 0;
      Tx_Array[2] = 0;
      radio.write(Tx_Array, 3);
    }
    //PORTD |= 0x08;
    Serial.println("Pressed");
    Tx_count++;
    Serial.println(Tx_count);
}
if(Tx_count){
  Tx_count++;
}
if((PIND & 0x04)&&(TX_Key_Pressed==1))
{
  TX_Key_Pressed = 0;
  Serial.println("Released");
  if(Tx_count > 15 && digitalRead(TX_LED) == 1){
    Serial.println("Cheat mode enabled");
    Tx_Array[2] = 1;
    //Serial.print("Zero Check: ");
    //Serial.println(Tx_Array[2]);
  }
  else {
    Tx_Array[2] = 0;
    //Serial.print("Zero Check: ");
    //Serial.println(Tx_Array[2]);
  }
  Serial.println(Tx_count);
  Tx_count = 0;
}

if((ay<=-4000)||((ay>=4000)))
```

```
  {
    //Serial.print("A , ");          // uncomment A,B,C..... if you are
debugging
    if((ax>=-4000)||((ax<=4000)))
    {
      //Serial.print("B , ");
      if((ay<=-4000))
      {
          //Serial.print("C , ");
          Tx_command = 1;                      // forward
          Speed_index = (ay + 4000)/-2000 + 1; // more the negative
value more the speed. ay valu varies from -16384 to +16384 same for ax
and others.
          if(Speed_index>5)
          {
            Speed_index = 5;
          }
      }

      if((ay>=4000))
      {
          //Serial.print("D , ");
          Tx_command = 2;                      // Backward
          Speed_index = (ay - 4000)/2000 + 1; // more the positive value
more the speed. ay valu varies from -16384 to +16384 same for ax and
others.
          if(Speed_index>5)
          {
            Speed_index = 5;
          }
      }
    }
    else
    {
        //Serial.print("E , ");
        Tx_command = 0;
        Speed_index = 0;
    }
  }
  else if((ax<=-4000)||((ax>=4000)))
  {
    //Serial.print("F , ");
    if((ay>=-4000)||((ay<=4000)))
    {
      //Serial.print("G , ");
      if((ax<=-4000))
      {
          //Serial.print("H , ");
          Tx_command = 4;                     // Right
          Speed_index = (ax + 4000)/-2000 + 1; // more the negative
value more the speed. ax valu varies from -16384 to +16384 same for ay
and others.
          if(Speed_index>5)
          {
            Speed_index = 5;
          }
      }

      if((ax>=4000))
      {
          //Serial.print("I , ");
          Tx_command = 3;                     // Left
          Speed_index = (ax - 4000)/2000 + 1; // more the positive value
more the speed. ax valu varies from -16384 to +16384 same for ay and
others.
          if(Speed_index>5)
          {
            Speed_index = 5;
          }
      }
    }
    else
    {
        //Serial.print("J , ");
        Tx_command = 0;
        Speed_index = 0;
    }
  }
  else
  {
      //Serial.print("K , ");
      Tx_command = 0;
      Speed_index = 0;
  }
  Serial.print(Tx_command);
  Serial.print(" , ");
  Serial.println(Speed_index);
  if(Tx_Enable_Flag)
  {
    Tx_Array[0] = Tx_command;
    Tx_Array[1] = Speed_index;
    radio.write(&Tx_Array, 3);     // 1st byte = Direction , 2nd Byte = 
Speed
  }
  delay(100);
}
```

Receiver Source Code

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#define CSN_GPIO    8
#define CE_GPIO     7

#define RIGHT_FORWARD     5  // Left side motor forward
#define RIGHT_BACKWARD    4  // Left side motor backward
#define LEFT_FORWARD      3  // Right side motor forward
#define LEFT_BACKWARD     2  // Right side motor backward

// Hardware configuration
RF24 radio(CE_GPIO, CSN_GPIO);                           // Set up 
nRF24L01 radio on SPI bus plus pins 7 & 8

const byte Address[6] = "00001";
unsigned char Received_Command = 0,Speed_index = 0,Run_Stop_Mode = 0,
Mode=0;   //  Run_Stop_Mode -> 0 = Stop , 1 = Run;
unsigned char Rx_Array[3];
unsigned int Run_Stop_Counter = 0;

void setup() {
  Serial.begin(115200);
  pinMode(RIGHT_FORWARD,OUTPUT);   //left motors forward
  pinMode(RIGHT_BACKWARD,OUTPUT);   //left motors reverse
  pinMode(LEFT_FORWARD,OUTPUT);   //right motors forward
  pinMode(LEFT_BACKWARD,OUTPUT);   //right motors reverse
  radio.begin();
  radio.openReadingPipe(0, Address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
  Serial.println("START");
}

void loop()
{
  if (radio.available())    // If the NRF240L01 module received data
  {
    delay(1);
    radio.read(&Rx_Array, 3);
    Received_Command = Rx_Array[0];
    Speed_index = Rx_Array[1];
    Mode = Rx_Array[2];
    Serial.print("Mode: ");
    Serial.print(Mode);
    Serial.print(Received_Command);
    Serial.print(" , ");
    Serial.println(Speed_index);
  }
  else if (!radio.available() && Received_Command==0)
//STOP (all motors stop) , If any other command is received.
  {
    Serial.println("Stop");
    Received_Command = 0;
    Speed_index = 0;
    Mode = 0;
    digitalWrite(RIGHT_FORWARD,LOW);
    digitalWrite(RIGHT_BACKWARD,LOW);
    digitalWrite(LEFT_FORWARD,LOW);
    digitalWrite(LEFT_BACKWARD,LOW);
  }

/***************** Speed control Logic *******************/
  if (Mode==1){
    if(Received_Command == 1)            //move forward(all motors
rotate in forward direction)
    {
      digitalWrite(RIGHT_FORWARD,HIGH);
      digitalWrite(RIGHT_BACKWARD,LOW);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,HIGH);
      delay(500);
      digitalWrite(RIGHT_FORWARD,LOW);
      digitalWrite(RIGHT_BACKWARD,HIGH);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,HIGH);
      delay(500);
    }
    else if(Received_Command == 2)      //move reverse (all motors
rotate in reverse direction)
    {
      digitalWrite(RIGHT_FORWARD,HIGH);
      digitalWrite(RIGHT_BACKWARD,HIGH);
      digitalWrite(LEFT_FORWARD,HIGH);
      digitalWrite(LEFT_BACKWARD,HIGH);
      delay(500);
      digitalWrite(RIGHT_FORWARD,LOW);
      digitalWrite(RIGHT_BACKWARD,HIGH);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,HIGH);
      delay(1000);
    }
    else if(Received_Command == 3)      //circle left (all motors rotate
in reverse direction)
    {
      digitalWrite(RIGHT_FORWARD,HIGH);
      digitalWrite(RIGHT_BACKWARD,HIGH);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,LOW);
      delay(500);
    }
    else if(Received_Command == 4)      //circle left (all motors rotate
in reverse direction)
    {
      digitalWrite(RIGHT_FORWARD,LOW);
      digitalWrite(RIGHT_BACKWARD,LOW);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,HIGH);
      delay(500);
    }
    else                                 //STOP (all motors stop) , If
any other command is received.
    {
      digitalWrite(RIGHT_FORWARD,LOW);
      digitalWrite(RIGHT_BACKWARD,LOW);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,LOW);
    }

  }

  if(Run_Stop_Mode==0)    // Stop
  {
    digitalWrite(RIGHT_FORWARD,LOW);
    digitalWrite(RIGHT_BACKWARD,LOW);
    digitalWrite(LEFT_FORWARD,LOW);
    digitalWrite(LEFT_BACKWARD,LOW);
    Run_Stop_Counter++;
    if(Run_Stop_Counter>=((5-Speed_index)*100))   // Speed_index = 1 ->
Min speed,   5 -> Max Speed.
    {
      Run_Stop_Counter = 0;
      Run_Stop_Mode = 1;
    }
  }
  else if(Run_Stop_Mode==1)    // Run
  {
    Run_Stop_Counter++;
    if(Run_Stop_Counter>=((Speed_index)*100))   // Speed_index = 1 ->
Min speed,   5 -> Max Speed.
    {
      Run_Stop_Counter = 0;
      Run_Stop_Mode = 0;
    }

    if(Received_Command == 1)            //move forward(all motors
rotate in forward direction)
    {
      digitalWrite(LEFT_FORWARD,HIGH);
      digitalWrite(RIGHT_FORWARD,HIGH);
    }
    else if(Received_Command == 2)      //move reverse (all motors
rotate in reverse direction)
    {
      digitalWrite(LEFT_BACKWARD,HIGH);
      digitalWrite(RIGHT_BACKWARD,HIGH);
    }
    else if(Received_Command == 3)      //turn right (left side motors
rotate in forward direction, right side motors rotates in backward
direction)
    {
      digitalWrite(LEFT_FORWARD,HIGH);
      digitalWrite(RIGHT_BACKWARD,HIGH);
    }
    else if(Received_Command == 4)      //turn left (right side motors
rotate in forward direction, left side motors rotates in backward
direction)
    {
      digitalWrite(RIGHT_FORWARD,HIGH);
      digitalWrite(LEFT_BACKWARD,HIGH);
    }
    else if(Received_Command == 0)      //STOP (all motors stop)
    {
      digitalWrite(RIGHT_FORWARD,LOW);
      digitalWrite(RIGHT_BACKWARD,LOW);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,LOW);
    }
    else                                  //STOP (all motors stop) , If
any other command is received.
    {
      digitalWrite(RIGHT_FORWARD,LOW);
      digitalWrite(RIGHT_BACKWARD,LOW);
      digitalWrite(LEFT_FORWARD,LOW);
      digitalWrite(LEFT_BACKWARD,LOW);
    }
  }
  delay(10);
}
```