VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

FACULTY OF COMPUTER ENGINEERING

Lê Văn Hiển-21522060

Phan Trọng Nhân-21522408

CAPSTONE PROJECT

TELECONTROL CAR USING ACCELEROMETER

XE ĐIỀU KHIỂN TỪ XA SỬ DỤNG CẢM BIẾN GIA TỐC

FINALTERM CAPSTONE PROJECT REPORT

HO CHI MINH CITY, 2023

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY **UNIVERSITY OF INFORMATION TECHNOLOGY**	**SOCIALIST REPUBLIC OF VIETNAM** **Independence – Freedom - Happiness**

DETAILED TOPICS

VIETNAMESE PROJECT NAME: xe điều khiển từ xa sử dụng cảm biến gia tốc
ENGLISH PROJECT NAME: telecontrol car using accelerometer
Instructor PhD. Tri Nhut Do, Faculty of Computer Engineering
Implementation time: From: 28/09/2023 To: 18/11/2023
Student Perform:
Overview of the topic: **The goal of the subject:** **Methods of implementation:**

Main contents of the topic:	
Certification of Instructor (Sign and clearly state full name)	**HCM city,Year 2023 Month 11 Day 18** **Student** (Sign and clearly state full name)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

Traditionally, objects can be moved by some external forces which are generally human efforts. In our project we aim to find a way to control physical objects by just using a small amount of motion. Specifically, we use an angle sensor detection to detect the angle in a particular direction and turn it into the direction of a car. This concept can bring much comfort to users and reduce the physical cost in daily life as well as industries.

First , what is acceleration? Acceleration is the name we give to any process where the velocity changes. Since velocity is a speed and a direction, there are only two ways for you to accelerate: change your speed or change your direction—or change both

Nowadays, controlling a machine from a distance is a concept we are all familiar with. And more robots are now replacing manual workers in some dangerous tasks in the industry. A telecontrol car using an accelerometer is one kind of robot that is capable of handling such tasks. Because it can be controlled remotely far away from humans. Therefore when entering dangerous areas it's safer for humans because they don't have to directly go in. In circumstances such as entering a constructed area, dealing with concentrated hazardous chemicals, treating patients with fatal diseases like ebola and corona, defusing bombs, carrying heavy objects in factories,.... An accelerometer controlled car can be very helpful. An accelerometer car is just a simple example of remotely moving an object. By sticking to this basic idea and example, humans can create many possibilities and improve life comfort.

Chapter2: Overview

2.1 Research Direction:

2.1.1 Research Direction in World Wide:

Research has shown that gesture controls for vehicle navigation is a viable option and is an accepted navigation solution by users (Qian et al., 2020). A study conducted by Parada-Loira et al. 2014 found that for in-vehicle controls, although participants found touchscreen interactions to be easier, gesture controls were less distracting and more useful. Another solution to decrease distracted driving involved the implementation of haptic feedback with in-vehicle control. Richter etal. 2010 integrated haptic feed-back with touchscreens, finding a decrease in error rates with input tasks. By combining gesture controls and haptic feedback, the best of both worlds—gestures as a preferred navigation method and haptic feedback to improve task performance and user experience—can be achieved to ultimately reduce distraction times (Gaffary & Lécuyer, 2018).

2.1.2 Research Direction in Vietnam:

In vietnam, research shows that gesture controlled cars have been imported widely in vietnam. With the growth of the middle class and expansion of urban population as well as Vietnam automation industry has been growing steadily, tele controlled cars emerge as a promising technology which can solve some current difficulties. Vietnam is a young country with a lot of young people in urban areas particularly Ho Chi Minh city. Thus technology is likely to be interested in adopting new transportation methods.

2.2 Problem:

Despite the benefits of telecontrol cars, there are few problems that make it uncommonly used and may be impractical in some countries. Firstly, the cost of a well-made, accurate tele control car design is relatively high, not to mention the cost of energy to supply the system which in some countries it's hard to afford. Secondly , for tele control cars to function well in traffic, road infrastructure as well as traffic systems must be well-thought, which is a big problem for developing countries.

chapter 3: Theory Basis

3.1 Overview Theory:

In theory, this project uses the MEM sensor (MPU 6050) to measure the acceleration. From the detected value from MPU 6050, the car will be able to move in the desired direction. In our project , users can move the controller forward, backward, left and right to tell the car which way to go.For example, If users turn the controller forward that means the car will move forward, and if they turn backward the car will move backward, the same logic applies to left and right. To communicate remotely between our controller and the car we use nRF24l01 as known as wireless transceiver.

3.2 Movement Detection:

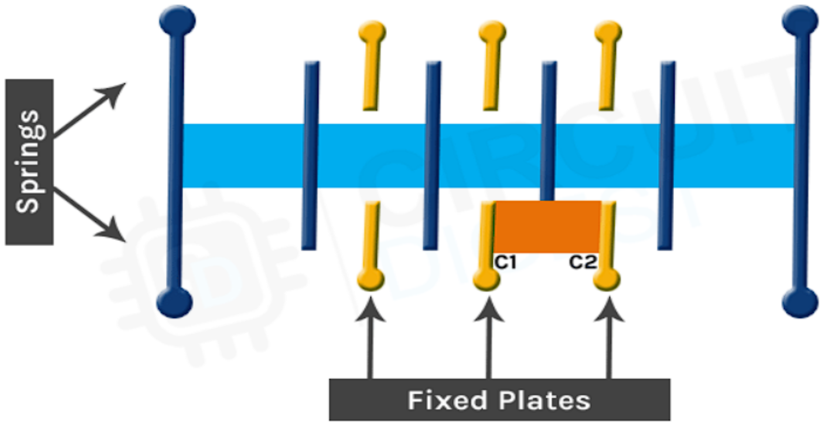


Figure 3.2: Simulation img of MPU

For this project, to detect direction movement we use a small device known as MPU 6050. MPU6050 is a very popular accelerometer gyroscope chip that has six axes sense (3 accelerometer values and 3 gyro values) with a 16-bit measurement resolution. But since our project the car only needs to move forward, backward, right and left. Therefore only 2 accelerometers (x-aisis, y-aisis) are needed. x -asis will return the value of whether the controller is moving right or left and y-aisis will return the value of whether the controller is moving front or back. All accelerometers work on the principle of a mass on a spring, when the thing they are attached to accelerates, then the mass wants to remain stationary due to its inertia and therefore the spring is stretched or compressed, creating a force which is detected and corresponds to the applied acceleration. In mems accelerometer, precise linear acceleration detection in 2 orthogonal axes is achieved by a pair of silicons MEMS detectors formed by spring ‘’proof” masses and electrodes are structures fixed in the substrate and remain stationary.When the sensor experiences acceleration, the mass is displaced relative to its surroundings, and this displacement is converted into an electrical signal, which can be used to measure linear acceleration.

3.3 Controlling car r]Remotely:

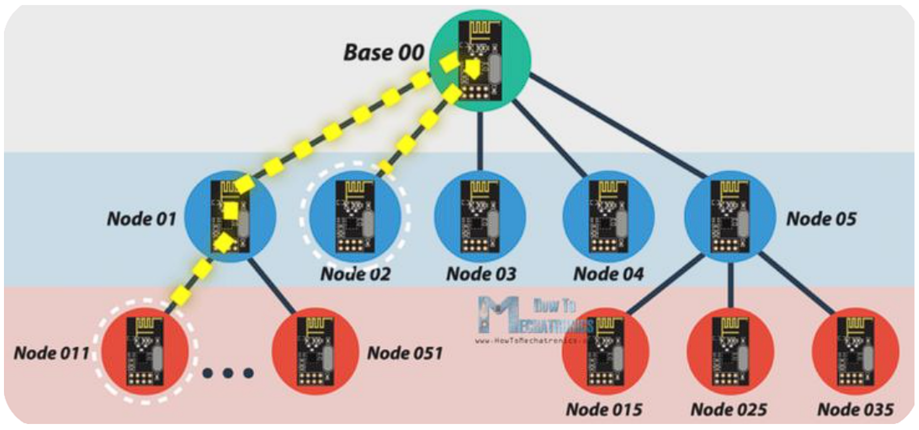


Figure 3.3: Wireless network of nRF24L01

Inorder for our car to be controlled without any wire connected, we use a device called nRF24l01 also known as a transceiver. The nRF24l01 comes in two parts: a transmitter(TX) and a receiver(RX). Each transmitter can transmit to six receivers(one receiver can listen to only one transmitter) but in this project we only need to control one car therefore only one transmitter and one receiver can meet our needs(two must have the same address). For a transceiver to become a transmitter we need to use commands such as (start writing , open writing pipe), the nRF24L01 remains in TX mode until it finishes the transmission of the current packet . In contrast, a transmitter can become a receiver when we use commands such as(start listening , open reading pipe), in this mode the RX continuously demodulates the signal from the RF channel and sends that to the baseband protocol engine.

3.4. Control wheel motor using L298N:

L298N is a popular dual H-bridge and commonly used to control motors. The L298N is designed in an H-bridge configuration, which consists of four switching transistors, they can control the direction of current flow through a load thus can make a motor go front or back and also change speed of a motor (each H-bridge can control one motor). In our project, we use 1 L298N to control 4 wheel-motors. 2 wheel-motors on the same side connect to 1 H-bridge because they need to move in the same direction.

3.5 Explanation of tele control car system:

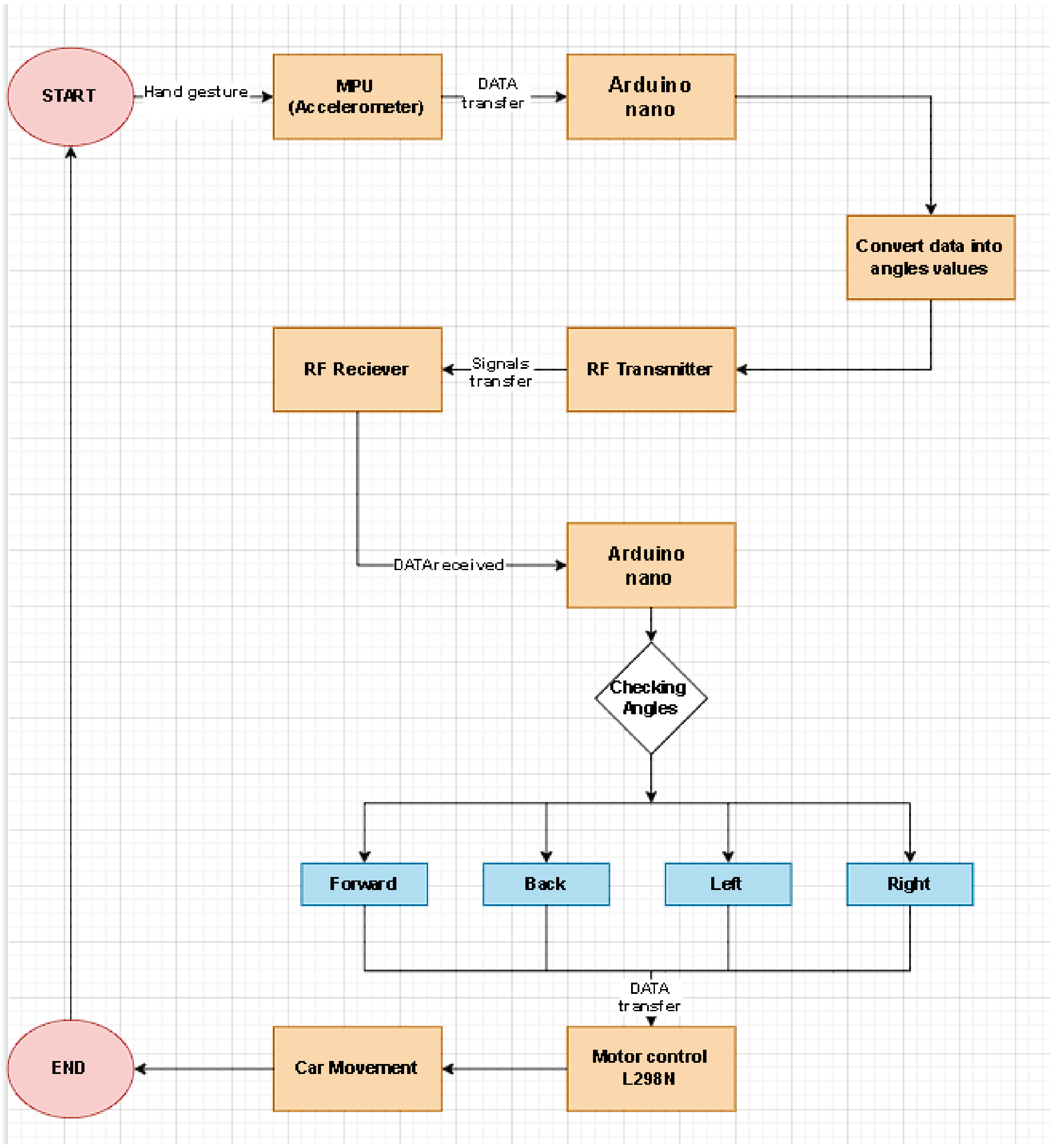


Figure 3.5: Diagram of full system

The system contains two main parts: a controller and a car. The controller consists of: 1 MPU, 1 transceiver and 1 microcontroller(arduino nano). A car includes: 1 L298N, 1 transceiver, 4 wheel motor and 1 microcontroller(arduino nano). The controller is used for sending commands to the car. Therefore, in our controller there is one transceiver which serves as TX to send data to the receiver(which is placed in the car to control it). First, The input is the rotation of the MPU (MEM sensor) . Microcontroller will collect data and put those in an array then command the transmitter to send packets of data to the receiver. Secondly , when the receiver receives data ,according to the input a microcontroller from the car will then control the L298N based on the algorithm that we set, so the system can recognize when to move forward, backward, right and left. Lastly, L298N will control 4 wheel-moto.

Chapter 4: System implementation process(mid-term)

4.1. Hardware system

4.1.1. Block diagram

- **Block diagram of transmitting system:**

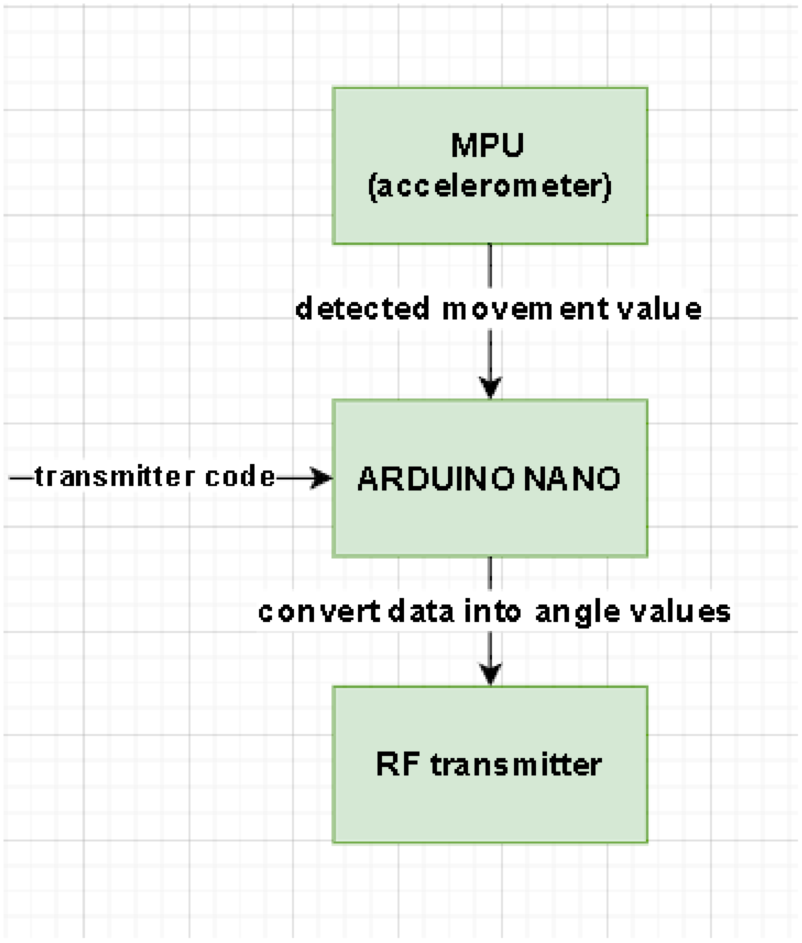


Figure 4.1.1.1: Transmitter Block Diagram

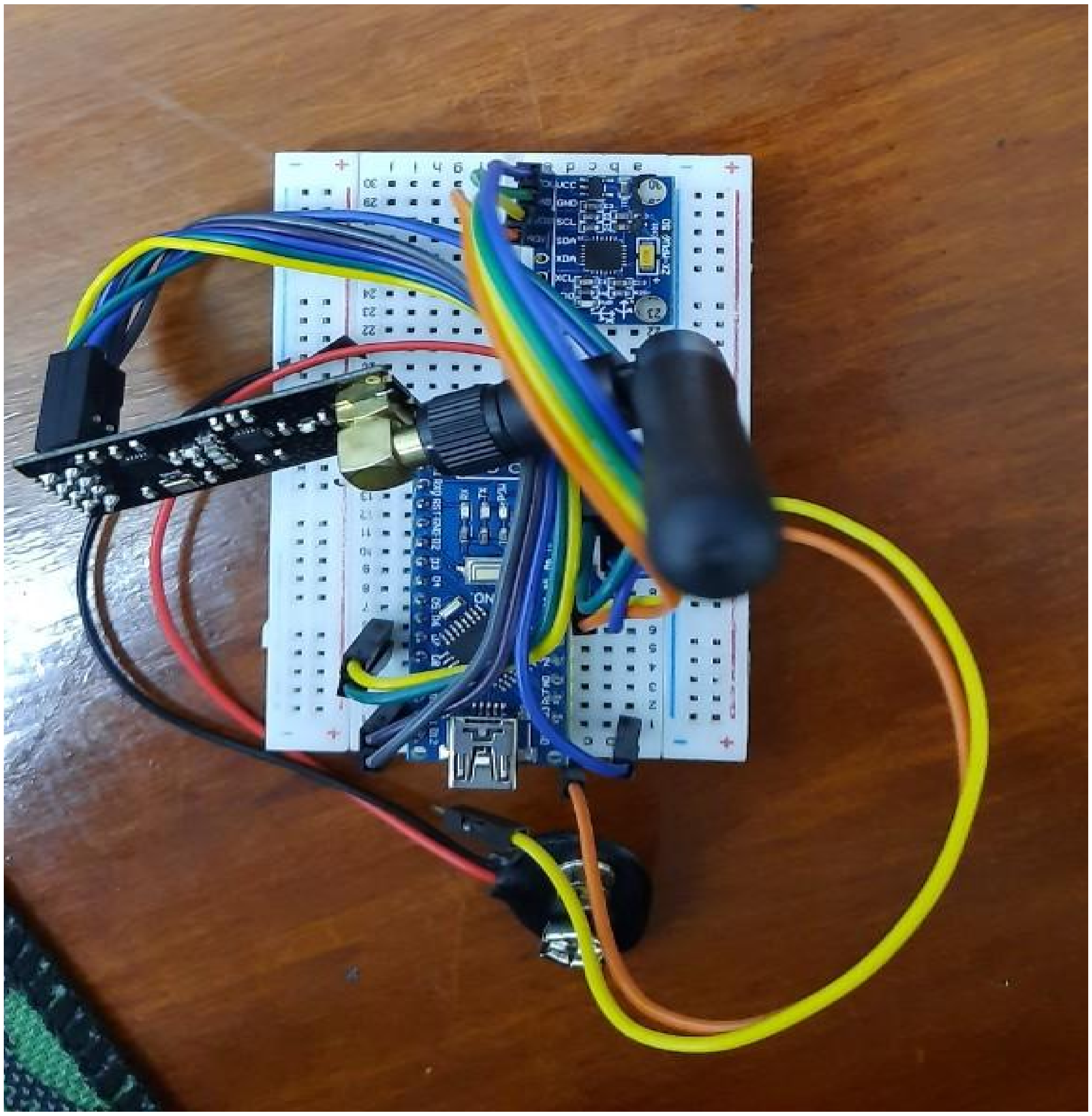


Figure 4..1.1.2: Transmitter Block

- **Block diagram of receiving system:**

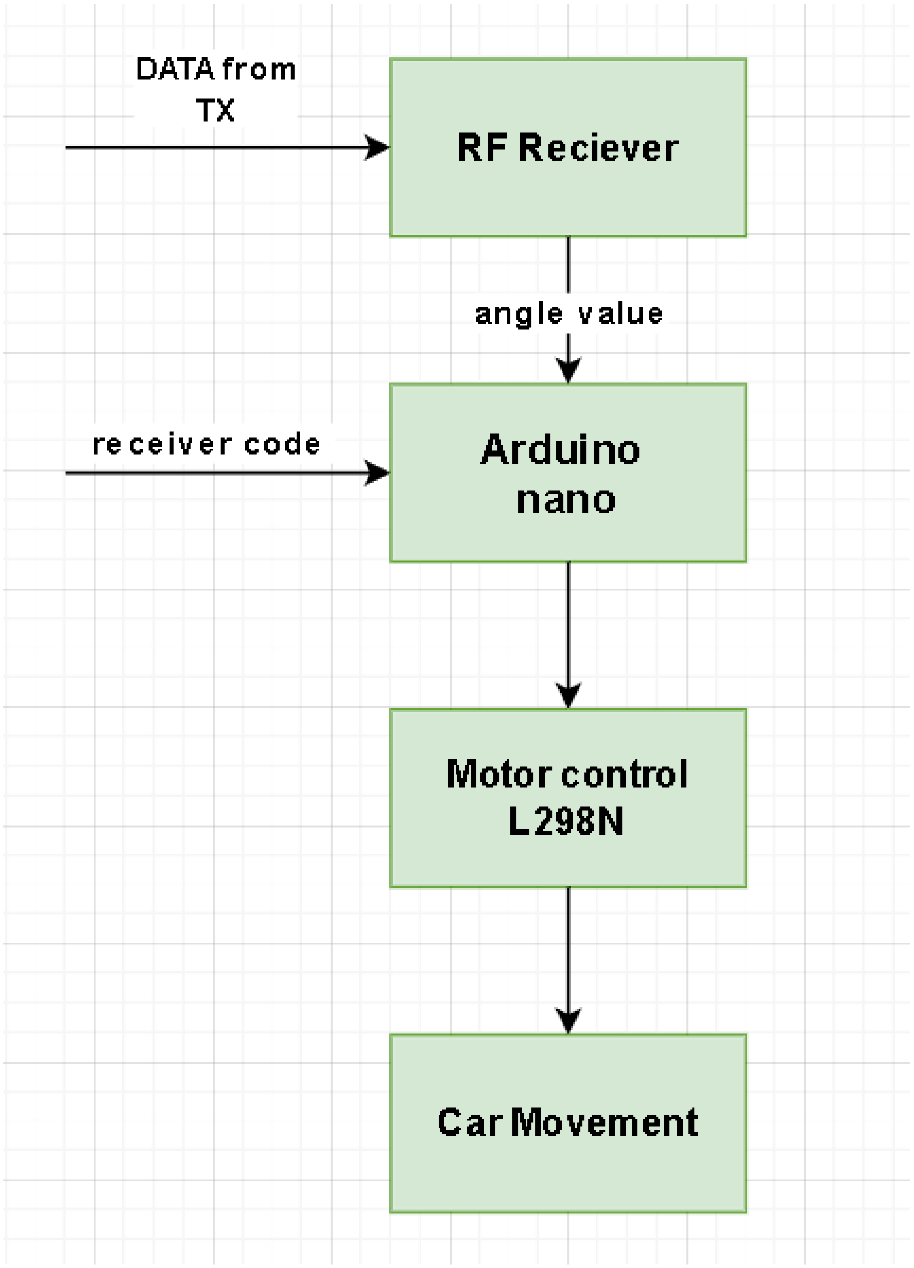


Figure 4.1.1.3: Receiver Block Diagram

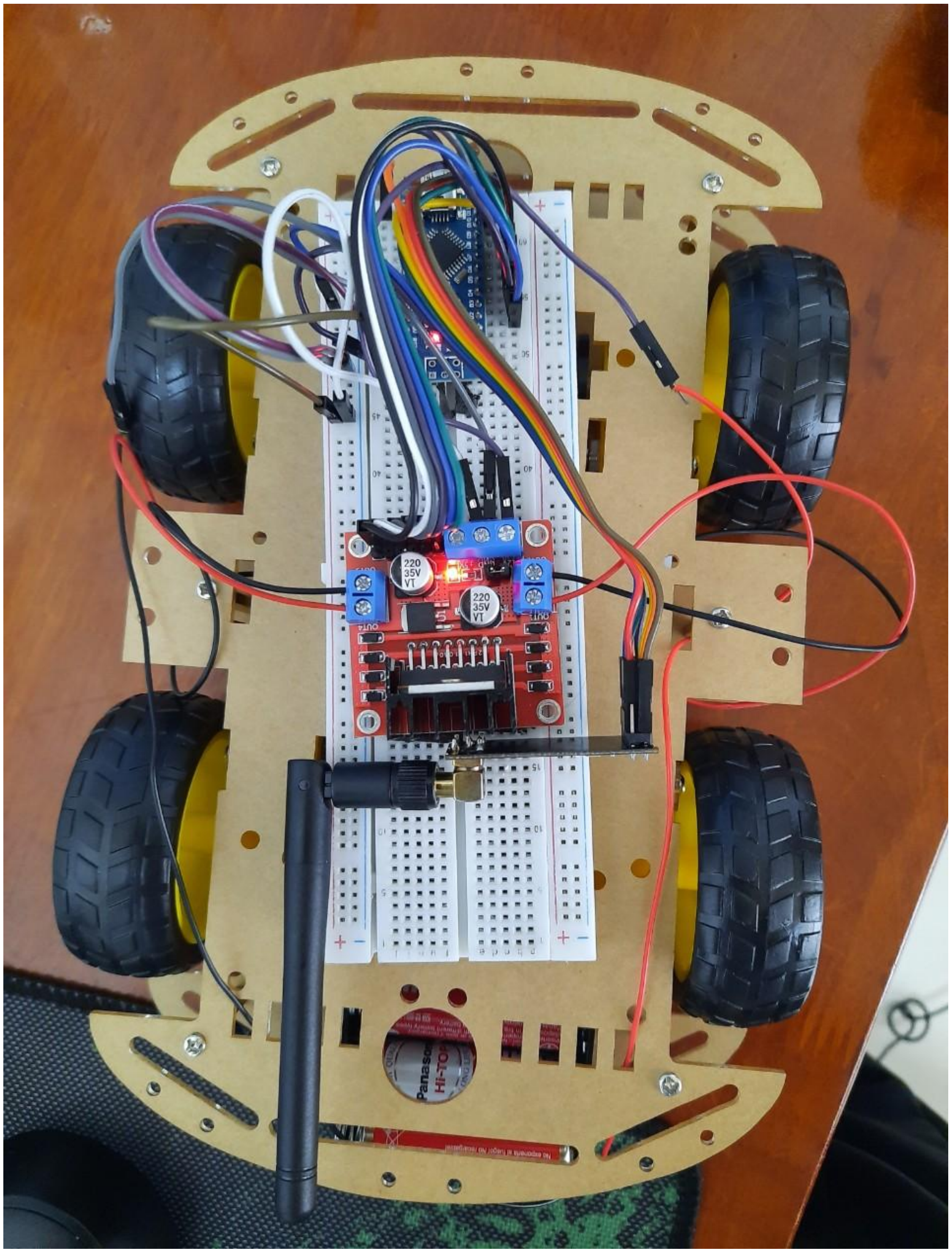


Figure 4.1.1.4: Receiver Block

4.1.2 : Circuit Schematic

- Circuit Schematic of transmitting system

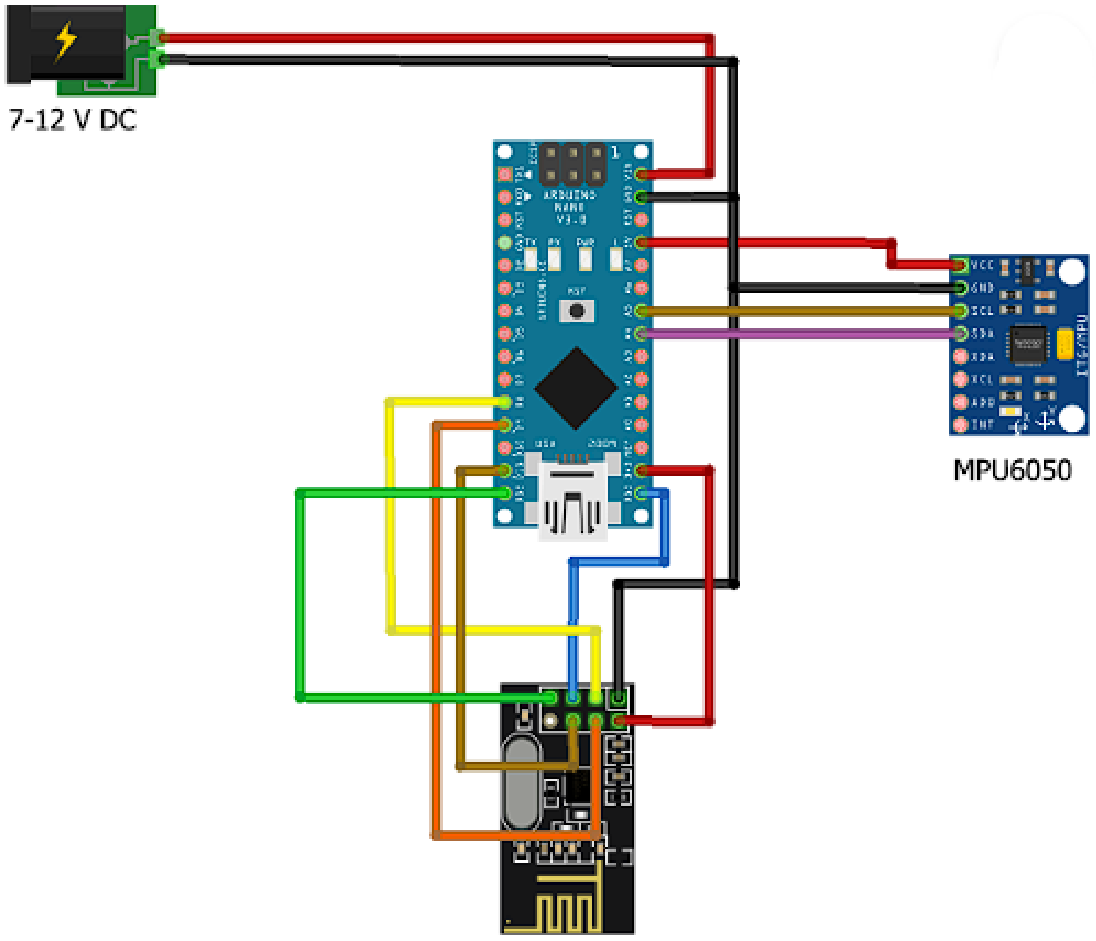


Figure 4.1.2.1: Transmitting System Schematic

- **Circuit Schematic of receiving system**

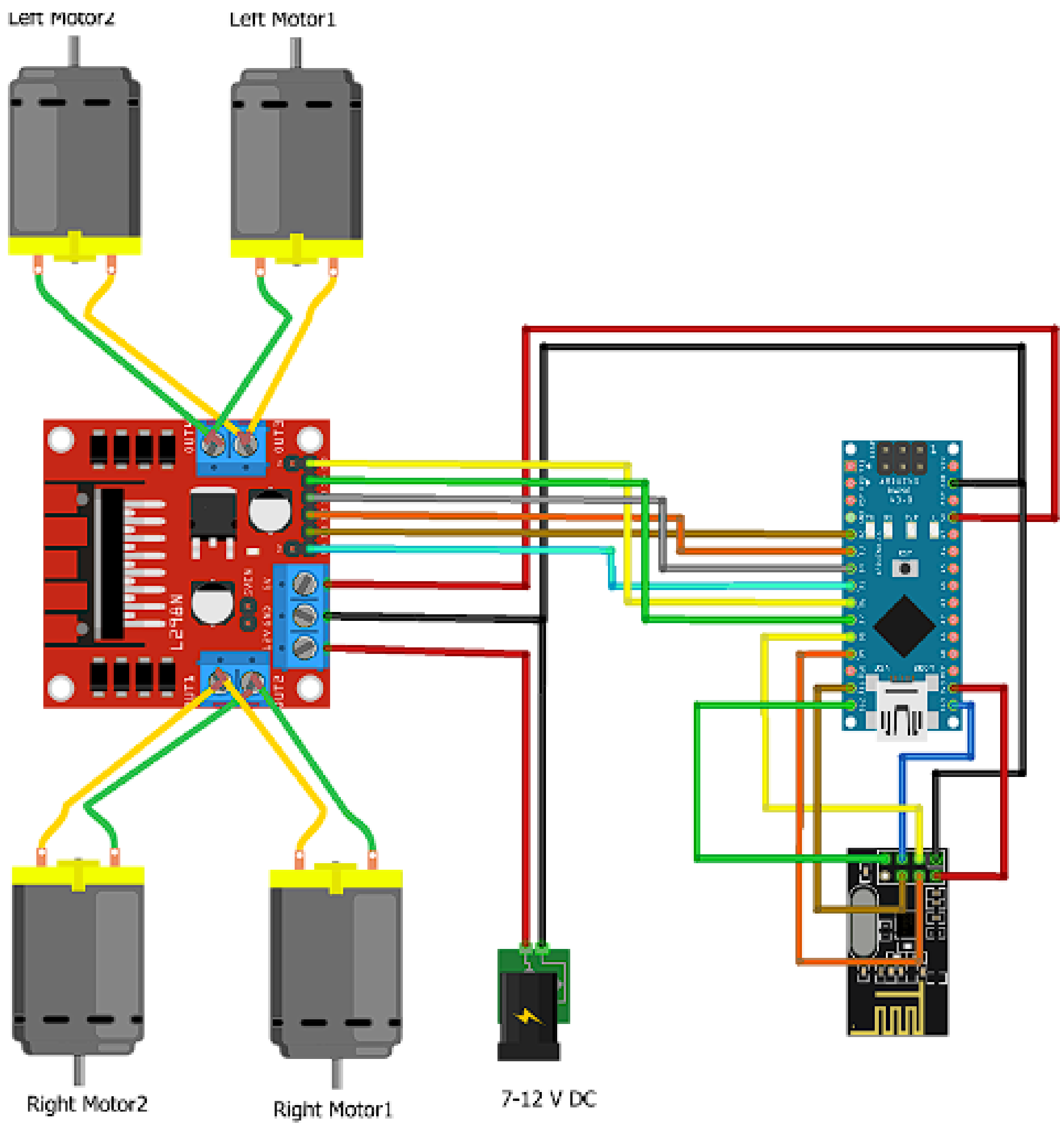


Figure 4.1.2.2: Receiving System Schematic

4.2. Software System

4.2.1 Software System flowchart

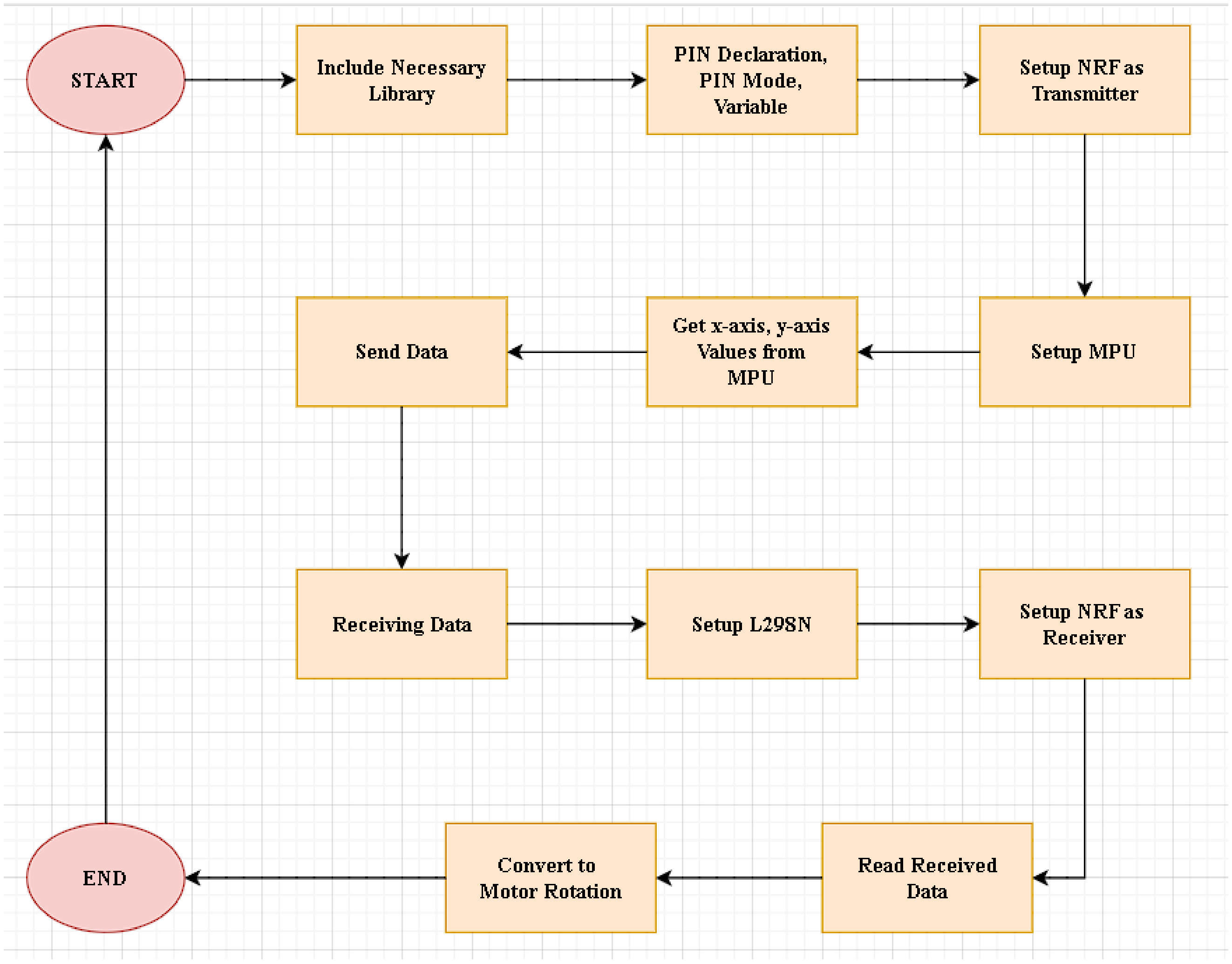
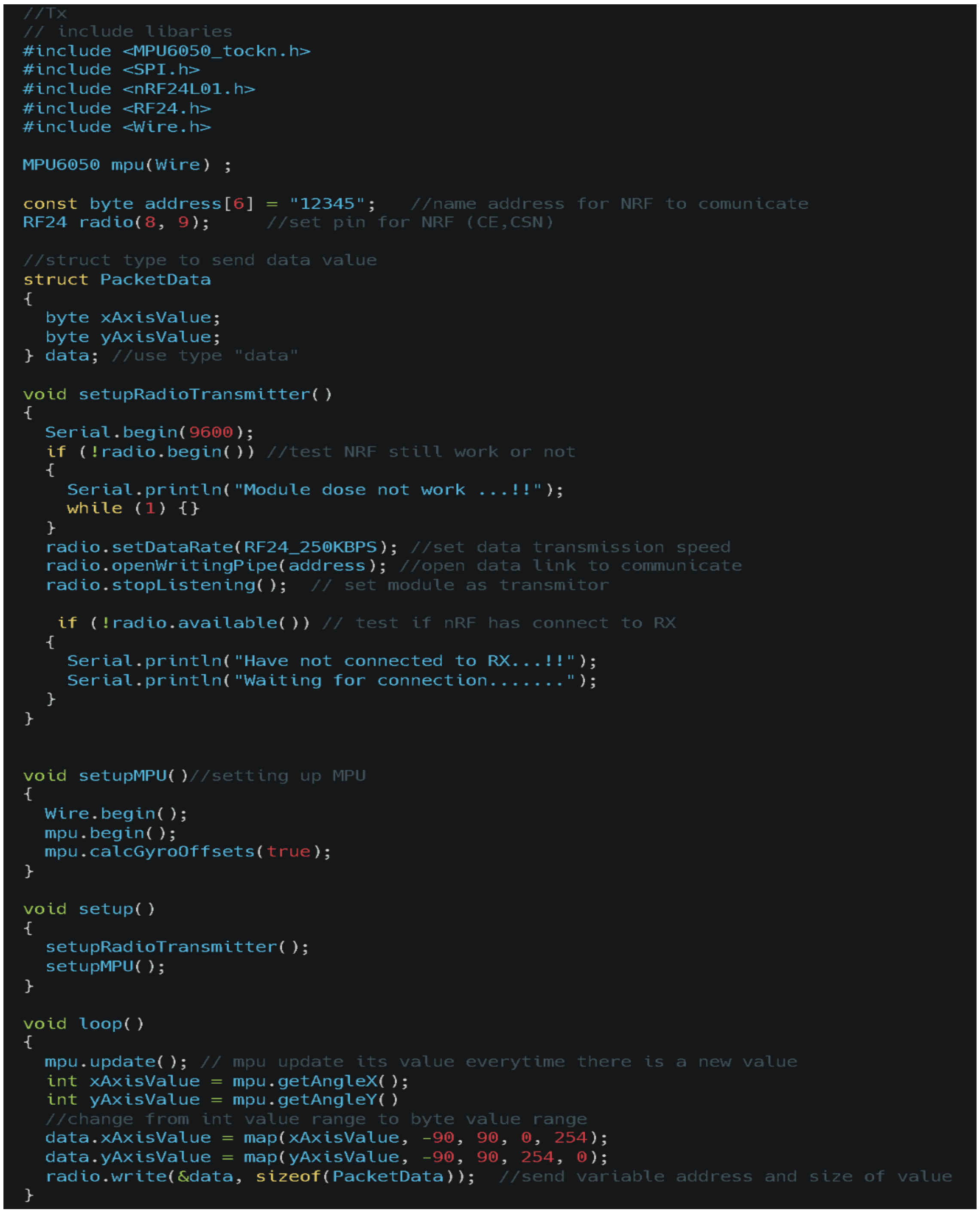

START
Include Necessary Library
PIN Declaration, PIN Mode, Variable
Setup NRF as Transmitter
Setup MPU
Get x-axis, y-axis Values from MPU
Send Data
Receiving Data
Setup L298N
Setup NRF as Receiver
Read Received Data
Convert to Motor Rotation
END

Figure 4.2.1: Software Flowchart

4.2.2. Software for Controller



```
//Tx
// include libaries
#include <MPU6050_tockn.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>

MPU6050 mpu(Wire) ;

const byte address[6] = "12345";   //name address for NRF to comunicate
RF24 radio(8, 9);     //set pin for NRF (CE,CSN)

//struct type to send data value
struct PacketData
{
  byte xAxisValue;
  byte yAxisValue;
} data; //use type "data"

void setupRadioTransmitter()
{
  Serial.begin(9600);
  if (!radio.begin()) //test NRF still work or not
  {
    Serial.println("Module dose not work ...!!");
    while (1) {}
  }
  radio.setDataRate(RF24_250KBPS); //set data transmission speed
  radio.openWritingPipe(address); //open data link to communicate
  radio.stopListening();  // set module as transmitor

   if (!radio.available()) // test if nRF has connect to RX
  {
    Serial.println("Have not connected to RX...!!");
    Serial.println("Waiting for connection.......");
  }
}


void setupMPU()//setting up MPU
{
  Wire.begin();
  mpu.begin();
  mpu.calcGyroOffsets(true);
}

void setup()
{
  setupRadioTransmitter();
  setupMPU();
}

void loop()
{
  mpu.update(); // mpu update its value everytime there is a new value
  int xAxisValue = mpu.getAngleX();
  int yAxisValue = mpu.getAngleY()
  //change from int value range to byte value range
  data.xAxisValue = map(xAxisValue, -90, 90, 0, 254);
  data.yAxisValue = map(yAxisValue, -90, 90, 254, 0);
  radio.write(&data, sizeof(PacketData));  //send variable address and size of value
}
```

Figure 4.2.2.1:Controller Software Code

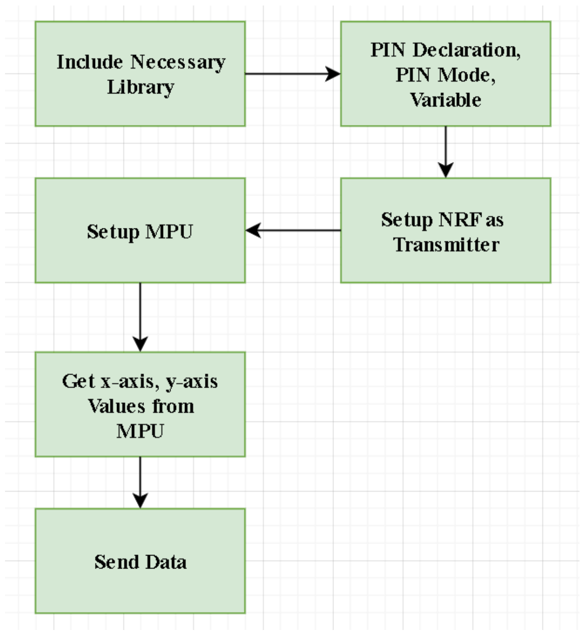
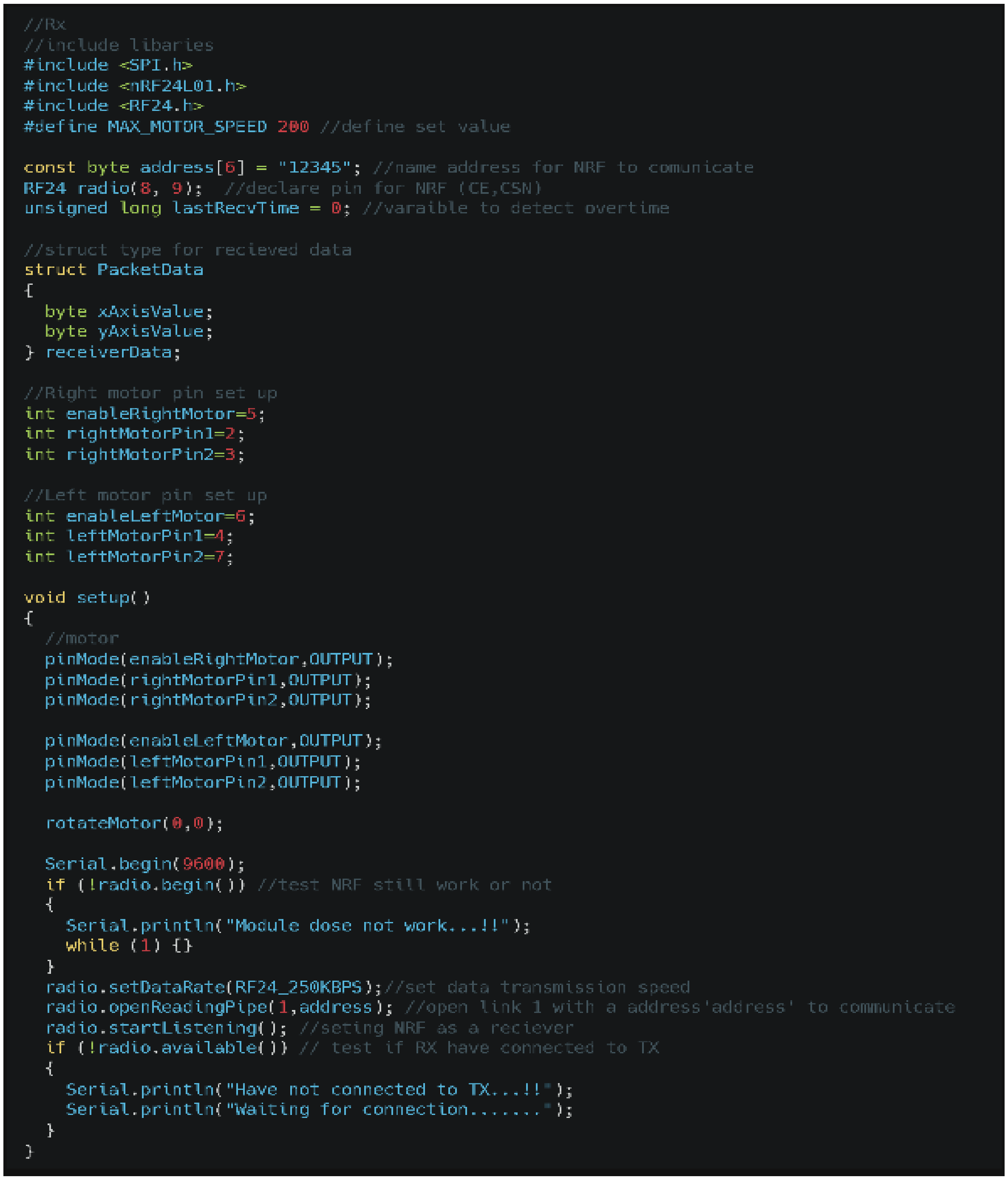

Include Necessary Library
PIN Declaration, PIN Mode, Variable
Setup NRF as Transmitter
Setup MPU
Get x-axis, y-axis Values from MPU
Send Data

Figure 4.2.2.2:Controller Software Flowchart

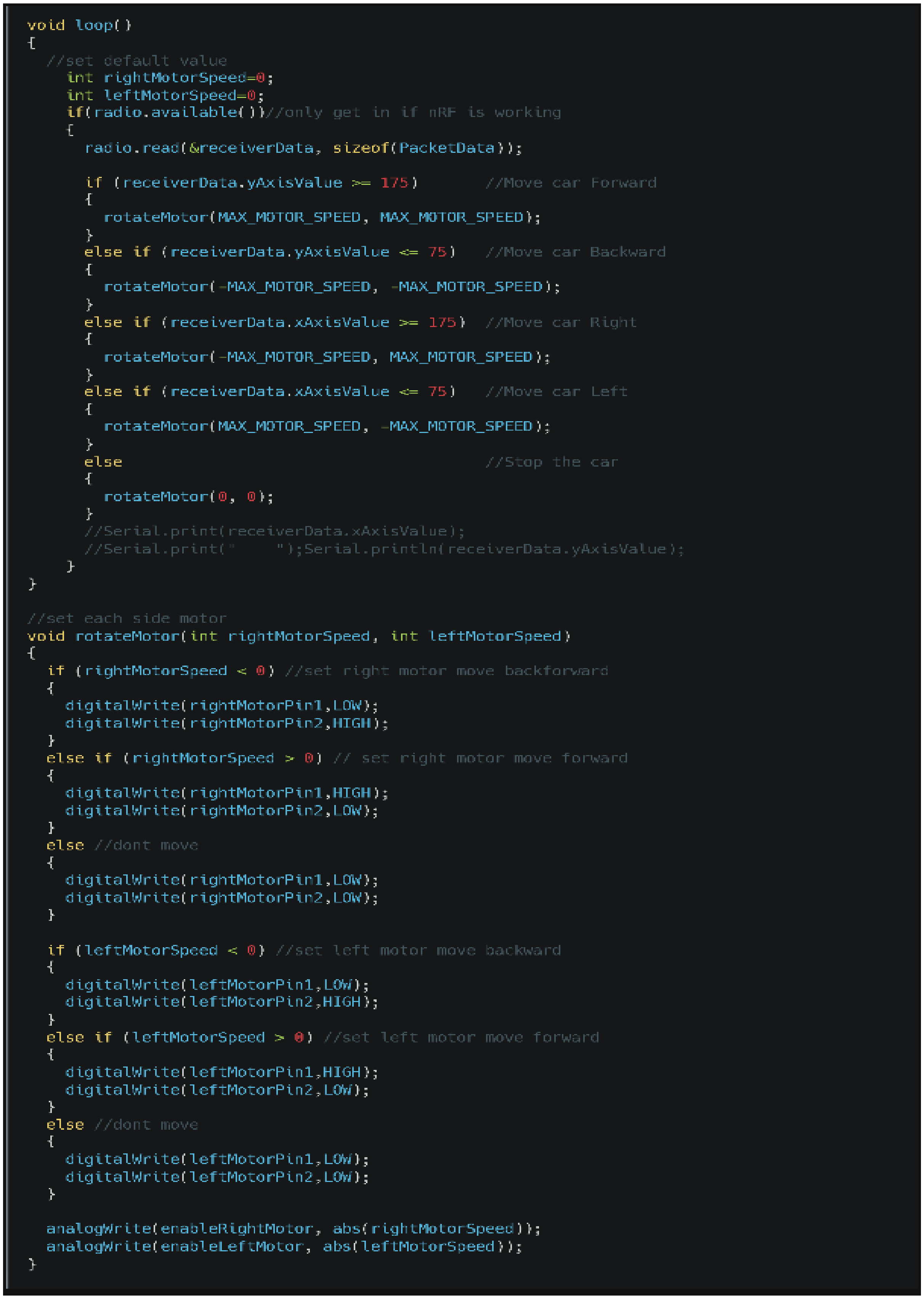
In our code after including all necessary libraries for the controller to function, we declare the pin we used and ports of the arduino as well as some needed variables. First, we have 2 void functions 1 void function to setup nRF the other is to setup MPU. The void function is to check if the components are working as well as setting necessary values. Second, we get the value of x-axis and value of y-axis in the MPU . Last, we put the values of x/y from MPU into an array of 2 elements and use an nRF transmitter to send the array as well as the size of the array to the nRF receiver.

4.2.3. Software for Car

Figure 4.2.3.1:Car Software Code



```
//Rx
//include libaries
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#define MAX_MOTOR_SPEED 200 //define set value

const byte address[6] = "12345"; //name address for NRF to comunicate
RF24 radio(8, 9);  //declare pin for NRF (CE,CSN)
unsigned long lastRecvTime = 0; //varaible to detect overtime

//struct type for recieved data
struct PacketData
{
  byte xAxisValue;
  byte yAxisValue;
} receiverData;

//Right motor pin set up
int enableRightMotor=5;
int rightMotorPin1=2;
int rightMotorPin2=3;

//Left motor pin set up
int enableLeftMotor=6;
int leftMotorPin1=4;
int leftMotorPin2=7;

void setup()
{
  //motor
  pinMode(enableRightMotor,OUTPUT);
  pinMode(rightMotorPin1,OUTPUT);
  pinMode(rightMotorPin2,OUTPUT);

  pinMode(enableLeftMotor,OUTPUT);
  pinMode(leftMotorPin1,OUTPUT);
  pinMode(leftMotorPin2,OUTPUT);

  rotateMotor(0,0);

  Serial.begin(9600);
  if (!radio.begin()) //test NRF still work or not
  {
    Serial.println("Module dose not work...!!");
    while (1) {}
  }
  radio.setDataRate(RF24_250KBPS);//set data transmission speed
  radio.openReadingPipe(1,address); //open link 1 with a address'address' to communicate
  radio.startListening(); //seting NRF as a reciever
  if (!radio.available()) // test if RX have connected to TX
  {
    Serial.println("Have not connected to TX...!!");
    Serial.println("Waiting for connection.......");
  }
}
```



```
void loop()
{
  //set default value
    int rightMotorSpeed=0;
    int leftMotorSpeed=0;
    if(radio.available())//only get in if nRF is working
    {
      radio.read(&receiverData, sizeof(PacketData));

      if (receiverData.yAxisValue >= 175)       //Move car Forward
      {
        rotateMotor(MAX_MOTOR_SPEED, MAX_MOTOR_SPEED);
      }
      else if (receiverData.yAxisValue <= 75)   //Move car Backward
      {
        rotateMotor(-MAX_MOTOR_SPEED, -MAX_MOTOR_SPEED);
      }
      else if (receiverData.xAxisValue >= 175)  //Move car Right
      {
        rotateMotor(-MAX_MOTOR_SPEED, MAX_MOTOR_SPEED);
      }
      else if (receiverData.xAxisValue <= 75)   //Move car Left
      {
        rotateMotor(MAX_MOTOR_SPEED, -MAX_MOTOR_SPEED);
      }
      else                                      //Stop the car
      {
        rotateMotor(0, 0);
      }
      //Serial.print(receiverData.xAxisValue);
      //Serial.print("    ");Serial.println(receiverData.yAxisValue);
    }
}

//set each side motor
void rotateMotor(int rightMotorSpeed, int leftMotorSpeed)
{
  if (rightMotorSpeed < 0) //set right motor move backforward
  {
    digitalWrite(rightMotorPin1,LOW);
    digitalWrite(rightMotorPin2,HIGH);
  }
  else if (rightMotorSpeed > 0) // set right motor move forward
  {
    digitalWrite(rightMotorPin1,HIGH);
    digitalWrite(rightMotorPin2,LOW);
  }
  else //dont move
  {
    digitalWrite(rightMotorPin1,LOW);
    digitalWrite(rightMotorPin2,LOW);
  }

  if (leftMotorSpeed < 0) //set left motor move backward
  {
    digitalWrite(leftMotorPin1,LOW);
    digitalWrite(leftMotorPin2,HIGH);
  }
  else if (leftMotorSpeed > 0) //set left motor move forward
  {
    digitalWrite(leftMotorPin1,HIGH);
    digitalWrite(leftMotorPin2,LOW);
  }
  else //dont move
  {
    digitalWrite(leftMotorPin1,LOW);
    digitalWrite(leftMotorPin2,LOW);
  }

  analogWrite(enableRightMotor, abs(rightMotorSpeed));
  analogWrite(enableLeftMotor, abs(leftMotorSpeed));
}
```

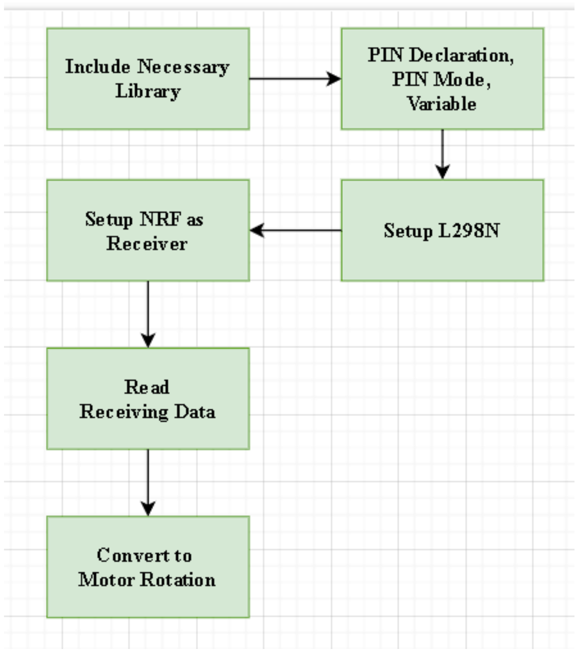

Include Necessary Library
PIN Declaration, PIN Mode, Variable
Setup L298N
Setup NRF as Receiver
Read Receiving Data
Convert to Motor Rotation

Figure 4.2.3.2:Car Software Flowchart

As for car software first we include all necessary libraries needed for the Car. We declare pins, ports and variables for the car. After that we set up L298N for motor control, the setting function for L298N comprises pin mode for L298N IO. Next is the setup function for nRf as a receiver. We declared a struct variable for received data from the transmitter. Finally , it is a set of if functions for converting the data from MPU to motor rotation so the car can rotate as the user wants.

4.3 Connect Car and Controller

The connection between car and controller comes down to two nRF. nRF in the controller has to work as the transmitter therefore it can transfer the values from the controller to the car. To set nRF as a transmitter is quite simple. After include nRF.h(nRF library) and nRF24l01.h(library for nRF24l01), we use function such as "radio.openWritingPipe()/radio.stopListening()" to make nRF works as the transmitter.

In Contrast, for the nRF24 to w
orks as the receiver in car system that to receive data from the controller and control the car we use functions such as "radio.openReadingPipe()/radio.startListening()" to command nRF works as the receiver.

Last is setting channels for two nRF to communicate to one another. In Order to communicate two nRF must connect in the same channel(pipe).

Chapter 5. TEST RESULTS AND EVALUATION :

5.1. Overview of the results:

Overall, the result has met our expectations and the car has been running smoothly with minor to non error.

5.2. Proposal system setting up

- **Car controller setup:**

+ Power supply: power supply we use battery 9v to supply power for the whole controller. When testing we use a power supply from a computer USB to debug and fix.
+ nRF: nRF was connected to arduino nano, with a respective port

  VCC -> 3.3V

  GND -> GND

  CE -> D7 (digital 7)

  CSN -> D8 (digital 8)

  SCK -> D13 (default SPI port for nRF to function)

  MO -> D11(default SPI port for nRF to function)

  MI -> D12(default SPI port for nRF to function)
+ MPU: MPU was connected to arduino nano, with a respective port

  VCC -> 5V

  GND -> GND

  SDA -> A4 (analog port cause MPU return array of numbers)

SCL ->A5 (analog port cause MPU return array of numbers)

- **Car setup:**

+ Power supply: to get enough power for all the IO and 4 motors to run, we use set of four batteries which provide a 12V power source.
+ nRF: nRF was connected to arduino nano, with a respective port

VCC -> 3.3V

GND -> GND

CE -> D7 (digital 7)

CSN -> D8 (digital 8)

SCK -> D13 (default SPI port for nRF to function)

MO -> D11(default SPI port for nRF to function)

MI -> D12(default SPI port for nRF to function)

+ L298N:

VCC -> 12V(battery) + 5V(Arduino nano)

GND -> GND

EN A -> D5

INPUT 1 ->D2

INPUT2 ->D3

OUT1 -> motor 1+2 positive

OUT2 -> motor 1+2 negative

EN B -> D6

INPUT3 ->D4

INPUT4 ->D7

OUT3 ->motor 1+2 positive

OUT4 ->motor 1+2 negative

5.3 Experimental Science

In the experiment for our project we tested three main categories, and found a few critical errors but soon found the science behind it and fixed them respectively.

Firstly, It is the connection between the transmitter and receiver NRF. At first, 2 modules did not work. The transmitter does not transmit and the receiver does not receive mainly because the modules did not start. We tried (serial.begin) function to know if the nRF was turned on and started working or not. But the problem was not the module or the adapter we plugged in; it was the wire and breadboard does not transfer enough electricity. After finding the science behind it, we were able to fix it and now 2 nRF are able to communicate wirelessly in the module.

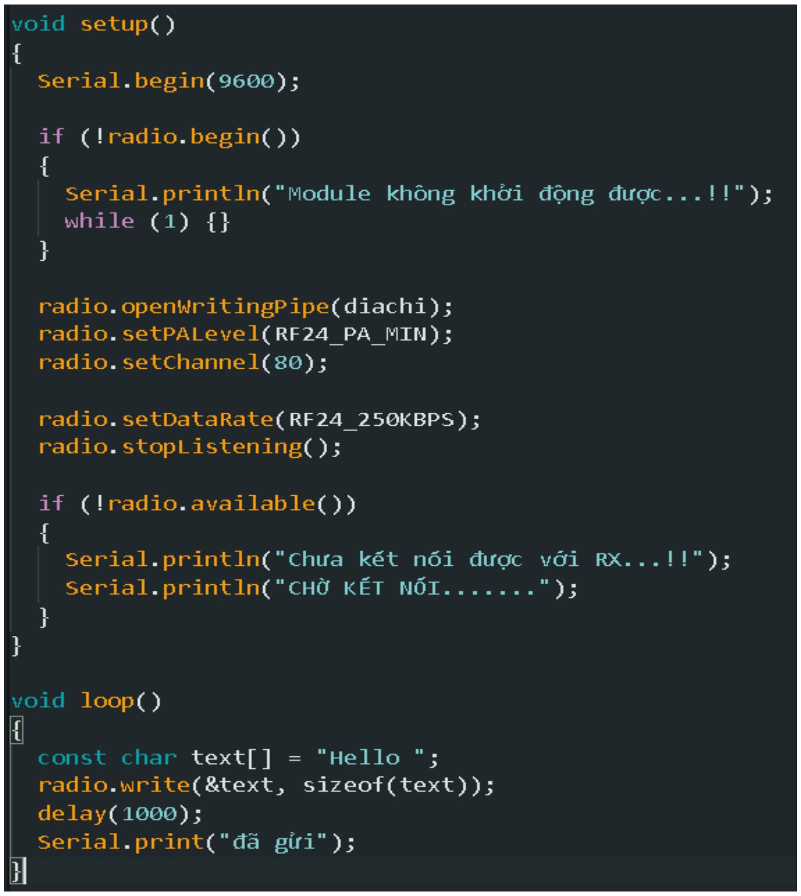
Secondly, we tried the performance of the MPU. The MPU we had did not have ports plugged in, therefore it was frustrating cause ports are easily disconnected. To deal with this problem we welded wire into the MPU port. Additionally , when plugged MPU into our transmitter system, the nRF did not turn on. And we found out it was because the power was not enough. The problems were soon fixed and MPU is running.

Lastly , we tried the full system. Because of full preparation the system is successfully run without any bugs.

5.4. Detailed Results on Experiments:

5.4.1. Transmit and Receive Result:

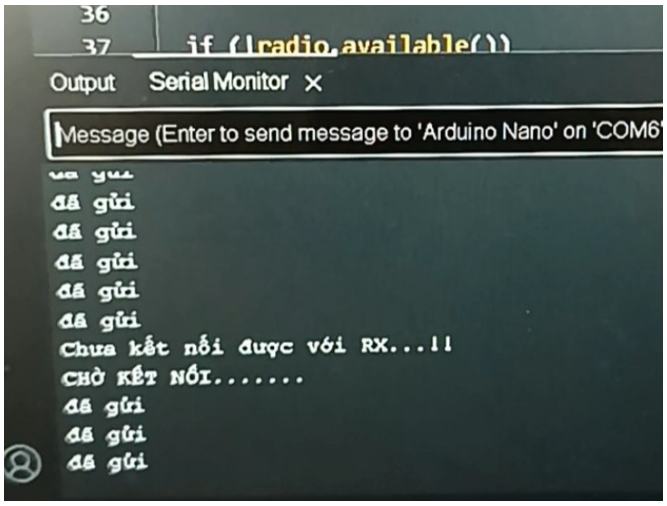
We are using this code to test the performance of nRF. The code is for the transmitter but we used it in both Tx and Rx to test if the nRF is working or not, since it is an easily damaged component.



```
void setup()
{
  Serial.begin(9600);

  if (!radio.begin())
  {
    Serial.println("Module không khởi động được...!!");
    while (1) {}
  }

  radio.openWritingPipe(diachi);
  radio.setPALevel(RF24_PA_MIN);
  radio.setChannel(80);

  radio.setDataRate(RF24_250KBPS);
  radio.stopListening();

  if (!radio.available())
  {
    Serial.println("Chưa kết nối được với RX...!!");
    Serial.println("CHỜ KẾT NỐI.......");
  }
}

void loop()
{
  const char text[] = "Hello ";
  radio.write(&text, sizeof(text));
  delay(1000);
  Serial.print("đã gửi");
}
```

Figure 5.4.1.1:nRF Testing Code

In this code first we use the function (serial.begin()) to test if it has begun or not, or if it is working. If the serial.begin()=1 which means it is working then it will go to set the basic channel/rate of the nRF. After that it will go to (serial.available) function which finds if the channel between Rx and Tx is available or not. If done all the testing and there is no bug, it will print (“đã gửi”) like in the picture below.

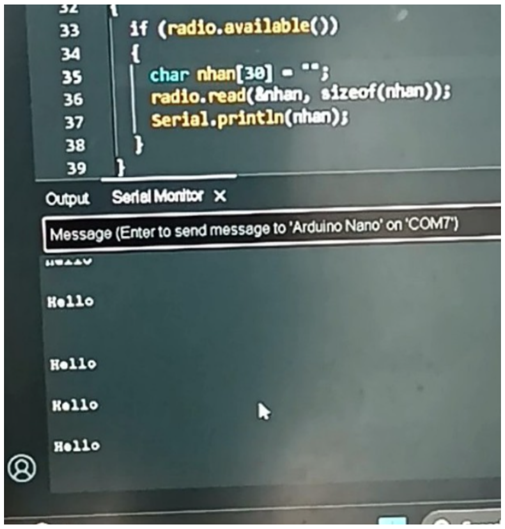


```
if (!radio.available())
```

```
đã gửi
đã gửi
đã gửi
đã gửi
đã gửi
Chưa kết nối được với RX...!!
CHỜ KẾT NỐI.......
đã gửi
đã gửi
đã gửi
```

Figure 5.4.1.2: Sending Result

After testing the both with transmitter code, we charge receiver code into the nRF works as the receiver. And plugged it with the USB to print the word (“Hello”) which we sended by the transmitter previously.

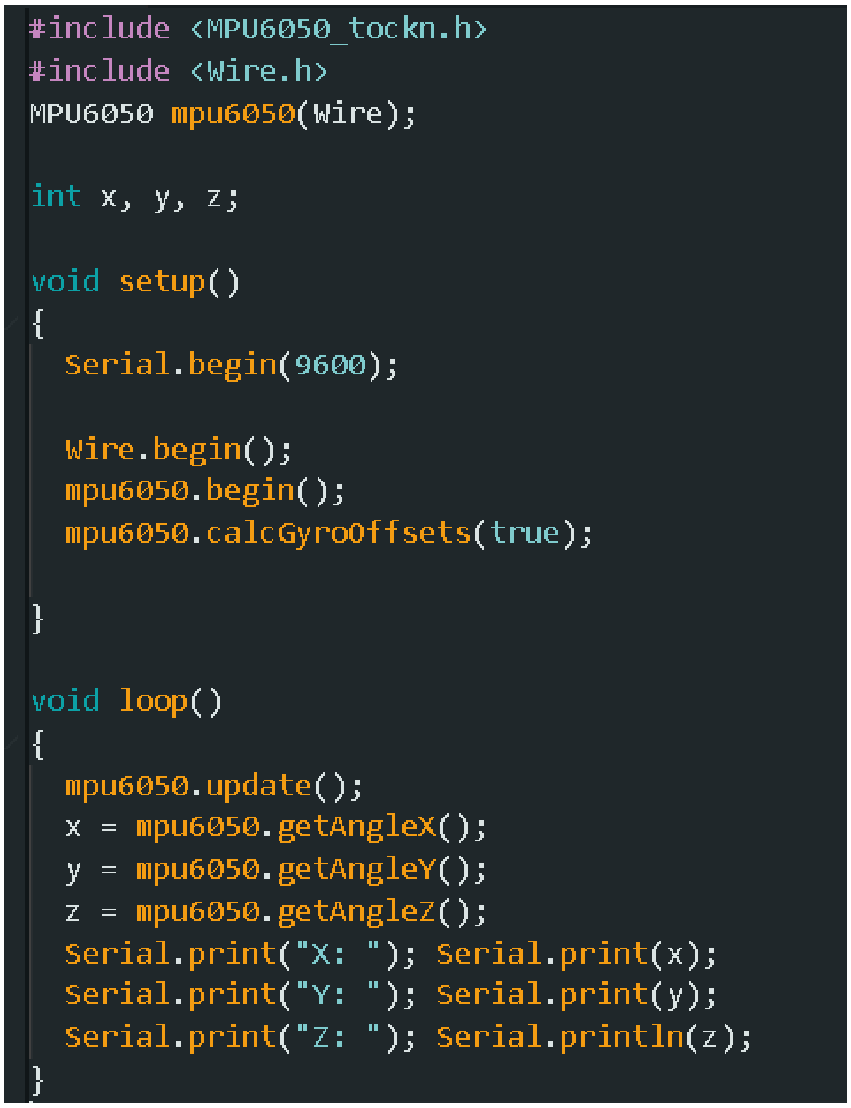


```
  if (radio.available())
  {
    char nhan[30] = "";
    radio.read(&nhan, sizeof(nhan));
    Serial.println(nhan);
  }
}
```

```
Hello

Hello
Hello
Hello
```

Figure 5.4.1.3: Full Testing Result

5.4.2. MPU Accuracy:



```
#include <MPU6050_tockn.h>
#include <Wire.h>
MPU6050 mpu6050(Wire);

int x, y, z;

void setup()
{
  Serial.begin(9600);

  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);

}

void loop()
{
  mpu6050.update();
  x = mpu6050.getAngleX();
  y = mpu6050.getAngleY();
  z = mpu6050.getAngleZ();
  Serial.print("X: "); Serial.print(x);
  Serial.print("Y: "); Serial.print(y);
  Serial.print("Z: "); Serial.println(z);
}
```

Figure 5.4.2: Testing MPU Code

In testing MPU we simply include needed libraries for MPU to work. After that we begin MPU and setup MPU. Lastly, we use the MPU library to get x,y,z and print it on the main screen.

5.4.3. Tele Control Car Performance:

Number	Controller	Expected value	Real value	Car movement
1	Forward	>=175	210	Forward
2	Backward	<=75	53	Backward
3	Right	>=175	197	Right
4	Left	<=75	26	Left
5	Forward	>=175	191	Forward
6	Right	>=175	211	Right
7	Forward	>=175	175	Forward
8	Left	<=75	70	Left
9	Forward	>=175	187	Forward
10	Backward	<=75	59	Backward
11	Right	>=175	188	Right
12	Left	<=75	74	Left
13	Backward	<=75	64	Backward
14	Right	>=175	185	Right

Table 5.4.3: Car Testing Case Table

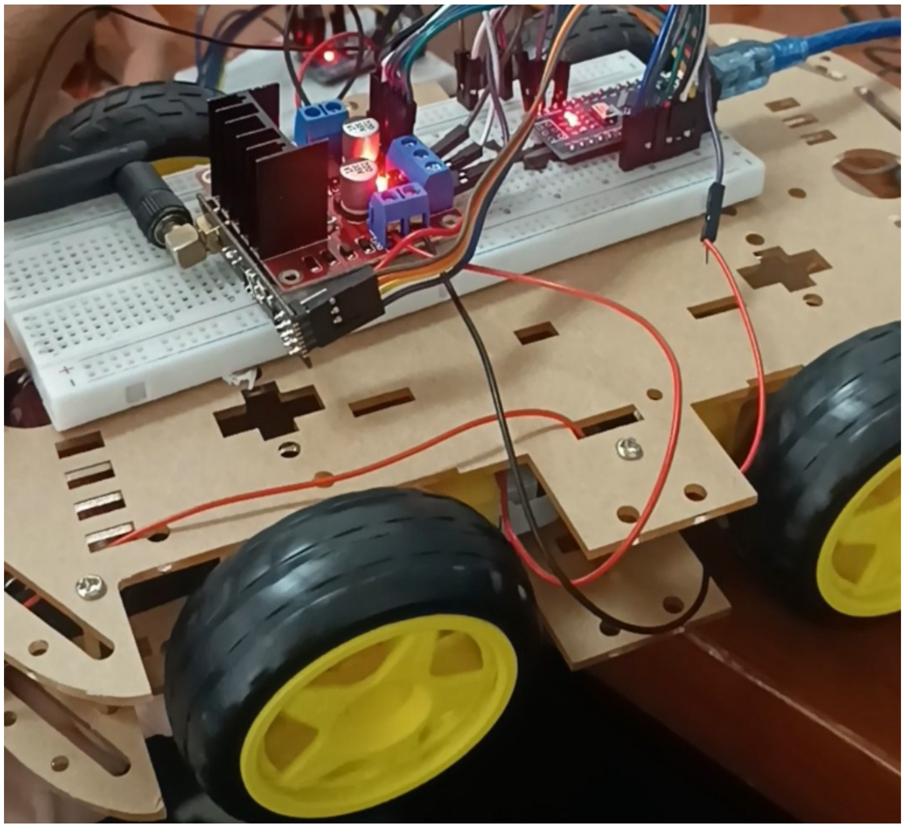


Figure 5.4.3: Car Performance

5.5. Result Evaluation:

In conclusion, because our project does not require precise value. Therefore, testing accuracy would be unnecessary. However, by testing table above and other testing we can say that the precision of this project is 100% correct since the car can respond fast and correctly in every test.

Chapter 6. SUMMARY, RESTRICTIONS AND DEVELOPMENT ORIENTATIONS:

6.1 SUMMARY

The tele control car using an accelerometer is a functional idea and can be useful in many situations,also the project has many potential and can be updated further more. Tele control car can be controlled by moving the controller in specific directions. The project is specifically tough in communicating process between 2 modules as well as the power supplies. But can be fixed by minimizing the code as well as change to more powerful power supplies.

6.2 RESTRICTIONS

The restriction of budget is probably the biggest restriction for us, because of that it causes a considerable amount of problems . It is because of limited amount of money we have to alternate some of the components with some cheaper more affordable options. The cheaper components are not as sustainable, quite easy to break and not reliable therefore we have to adjust and change a great amount of methods. Buses and energy resources are also a big restriction for us because the buses are easy to fail and power is not enough to function

6.3 DEVELOPMENT ORIENTATIONS

In future projects to continue to develop our car: There is no doubt that there are plenty of possibilities and ideas to improve and upgrade this project. Some of the ideas is that

- Embedded the camera into the system so the car can recognize more movement
- Use AI for a car to make the car self function
- Find more efficient energy and buses problems

- Instal more functions to make the car more useful