Caroline Praetzel, Tilt remote control:

The goal for my final project was to build a tilt remote to wirelessly control my robot's motion via my own physical motion. This project required a RF antenna to transmit X & Y accelerometer coordinates to my robot's Arduino, which were then used to create conditional statements to turn on corresponding direction control pins.

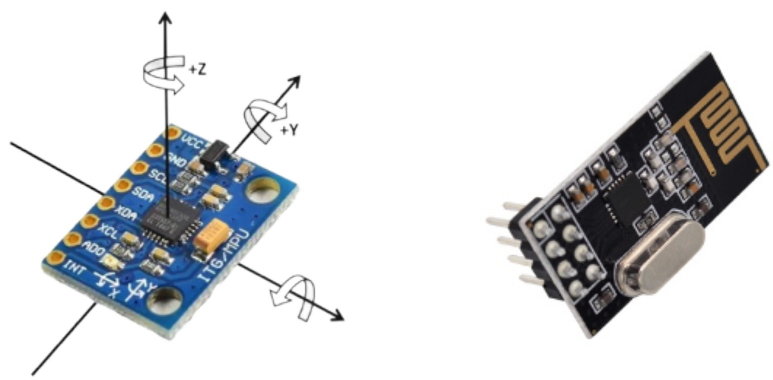


Figure 3: MPU6050 accelerometer, nRF24L01 transceiver

Supplies Needed:
MPU6050 Accelerometer/gyroscope.
NRF24L01 RF Transceiver module (x2)
Arduino Nano Every (x2)
9V Battery
10 µF capacitor (x2)
Mini Breadboard

Links:
RF tutorial:
https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/
Project inspiration: https://www.youtube.com/watch?v=RTJ33EWmTRI&t=11s
MPU6050 Library: https://goo.gl/uHB7jX
nRF24L01 Library: RF24 by TMRh20 available in Arduino library manager
I2Cdev Library: https://goo.gl/Ke1Wg1

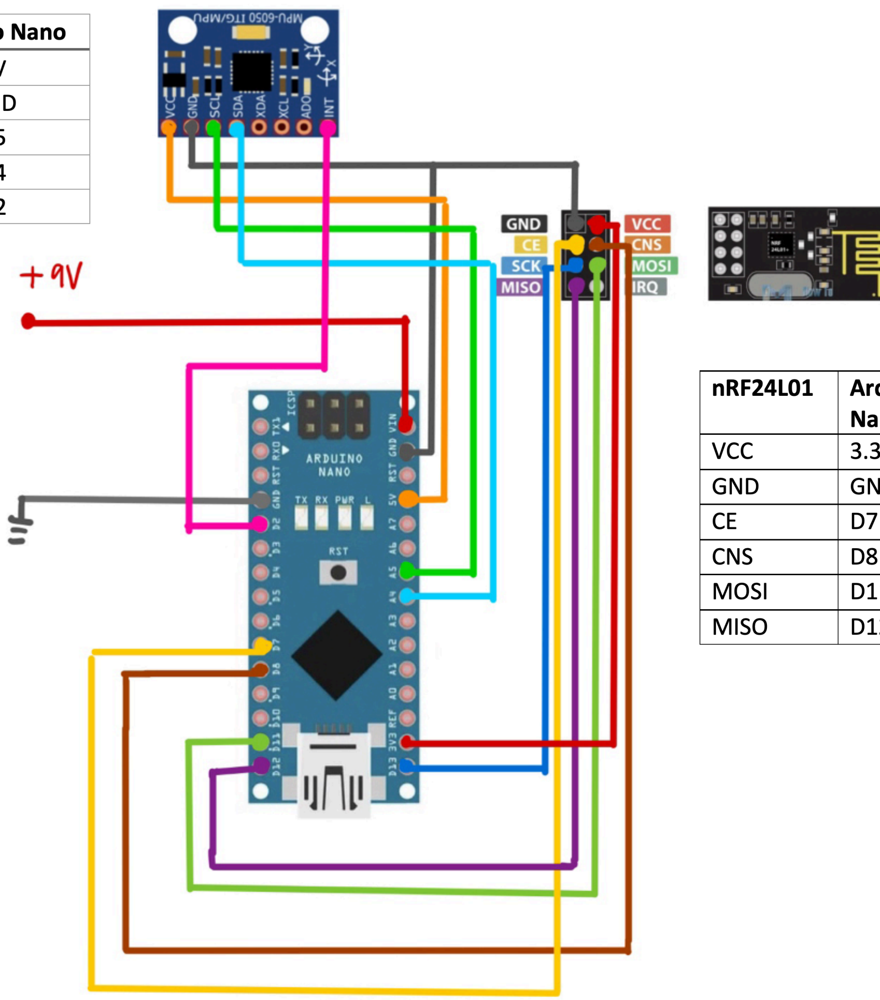
Upon receiving the supplies, I first completed an online tutorial for the NRF24L01 transceiver so that I could gain familiarity with the necessary Arduino libraries and ensure that my devices could communicate with each other. This tutorial was very helpful, and allowed me to send a simple string message, "Hello world" message between the RF chips. After completing this I knew my modules were not defective.

Remote Transmitter:

The RF module uses a Serial Peripheral Interface (SPI) (Figure 4). A 10 µF capacitor should first be soldered to between VCC and GND of the RF module; I initially skipped this step and was experiencing lots of difficulty maintaining communication until I added in this bypass capacitor. The Arduino Nano has designated MISO/MOSI/SCK pins, and the CE/CNS can be connected to any digital pins. The logic voltage for these pins can be 5V, whereas the operating voltage (VCC) is 1.9V-3.6V, so I connected the module VCC to the 3.3V Arduino output pin.

The MPU6050 has a 3-axis accelerometer & 3 axis-gyroscope, but I only used the X & Y accelerometer values to obtain left/right/forward/backward motion options. I connected the SCL & SDA to the corresponding Arduino pins A5 & A4, respectively. I then connected the INT pin to D2, and VCC to the Arduino's 5V pin.

MPU6050	Arduino Nano
VCC	5V
GND	GND
SCL	A5
SDA	A4
INT	D2



nRF24L01	Arduino Nano
VCC	3.3V
GND	GND
CE	D7
CNS	D8
MOSI	D11
MISO	D12

Figure 4: Remote transmitter schematic

I then made these connections using jumper wires on a mini breadboard, and powered them with the 9V battery in the ECEN 2270 kit. Finally, I mounted the entire breadboard to an extra nintendo switch remote base that I had at home, and taped the antenna to the front. The overall design was quite messy, and an obvious improvement to this project would be using a protoboard or PCB for a cleaner look.

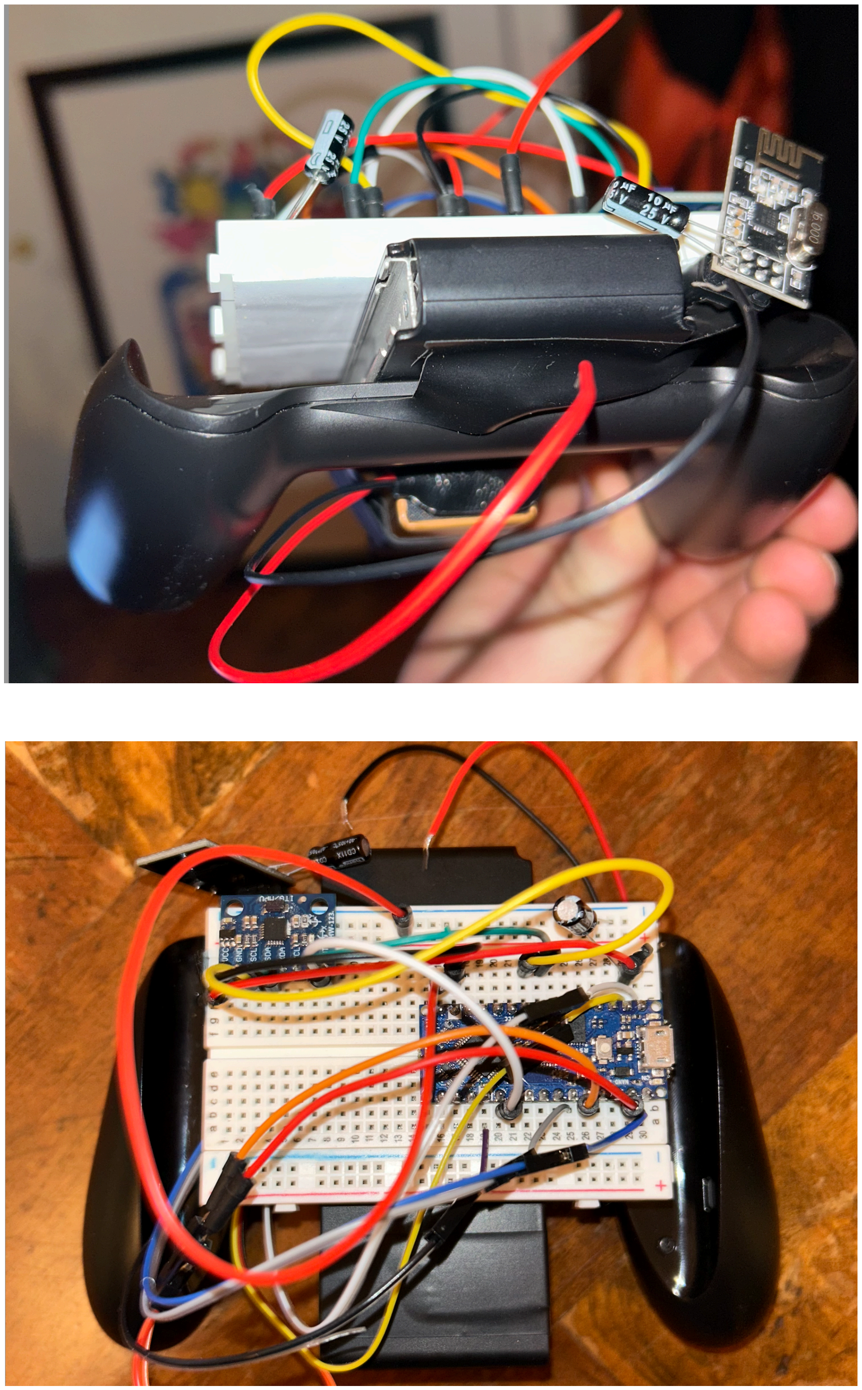
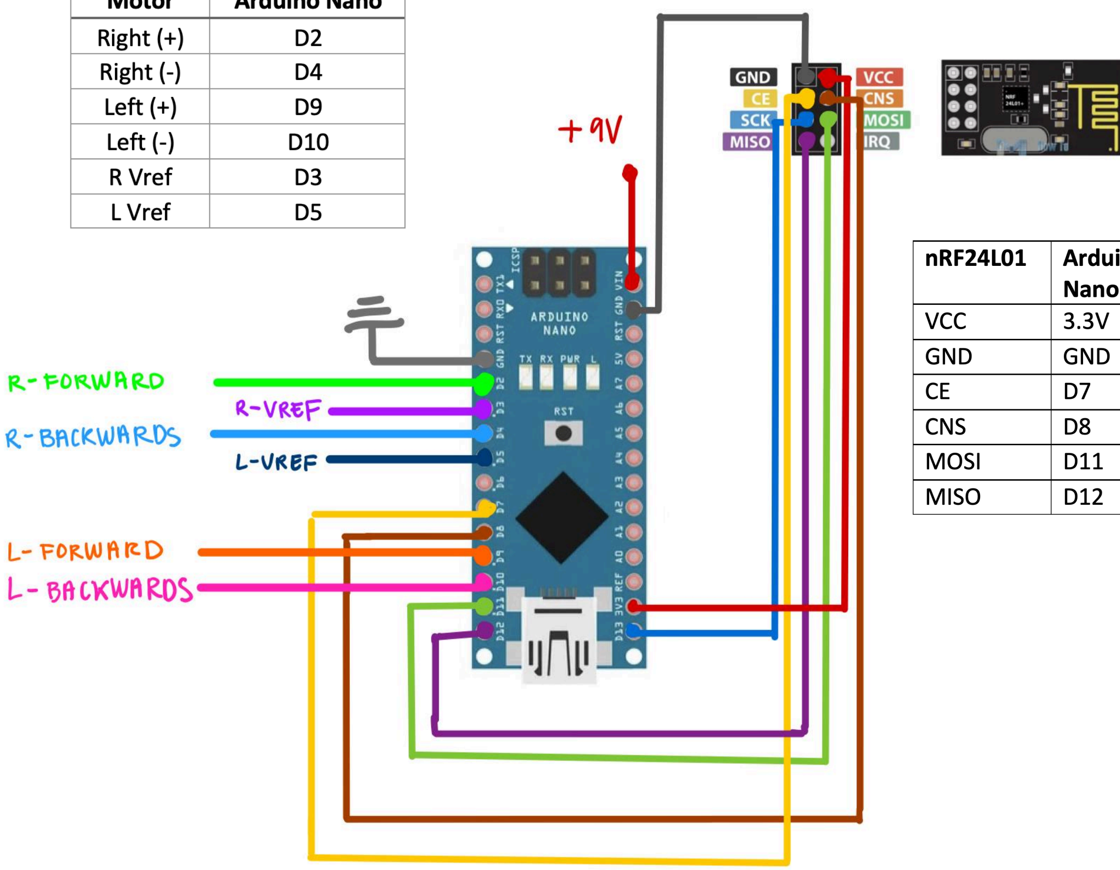


Figure 5: Final remote design

Robot Receiver:
The SPI connections between the RF receiver and the Arduino were the same as the transmitter. I also added connections to the four existing direction control nodes, and also to the two low pass filters going to each Vref node in the compensator circuits. I then mounted the antenna at the rear of the robot with tape.

Motor	Arduino Nano
Right (+)	D2
Right (-)	D4
Left (+)	D9
Left (-)	D10
R Vref	D3
L Vref	D5



nRF24L01	Arduino Nano
VCC	3.3V
GND	GND
CE	D7
CNS	D8
MOSI	D11
MISO	D12

Figure 4: Receiver schematic

Conclusions:
Eventually I ended up getting the tilt controller to work for my robot, but overall this project was very challenging for me. I really learned how complicated it can be to try to pinpoint what's not working when you have multiple devices interacting, and my remote still had barely any devices, just three! I am glad that I took the time to make sure each device was working independently before trying to get them all working together, because I think I would have been completely stuck if I just built the whole remote, ran the code, and nothing happened. After struggling for a few weeks, finally getting this thing to work was one of the most satisfying project experiences I have had in college!

Initially, I was able to send the "Hello world" string between antennas, and print the accelerometer data to the serial module, but I could not get the gyro and antennas working at the same time. The coding was the hardest portion for me, I had to find alternative Arduino libraries when the ones I was initially using were not working. I don't have much experience coding in C or Arduino, so I was struggling with editing the code to work with the new libraries.

If I had more time, I would have really liked to improve the usability of the controller. While the remote completed my goal of causing the robot to move, it was not the easiest to control. I would want to continue to finetune the numeric values of the conditional statements so that the sensitivity of each directional motion matched. I also think it would have been cool to add portions into the code where a further angled tilt increased the robot's speed by changing the associated Vref value. One last idea I had was to experiment with using the X-Z accelerometer coordinates instead of X-Y, since the Z rotation is more similar to spinning a steering wheel.

Transmitter code appendix:

```
//Add the necessary libraries
#include <SPI.h>        //SPI library for communicate with the nRF24L01+
#include "RF24.h"       //The main library of the nRF24L01+
#include "Wire.h"       //For communicate
#include "I2Cdev.h"     //For communicate with MPU6050
#include "MPU6050.h"    //The main library of the MPU6050
#include <nRF24L01.h>
#include "String.h"

//Define the object to access and control the Gyro and Accelerometer (We don't use the Gyro data)
MPU6050 mpu;
int16_t ax, ay, az;
int16_t gx, gy, gz;

//Define packet for the direction (X axis and Y axis)
int data[2];

//Define object from RF24 library - 7 and 8 are a digital pin numbers to which signals CE and CSN are
connected.
RF24 radio(7,8);

//Create a pipe addresses for the communicate
const byte address[6] = "00001"; //needs to match receiver address

void setup(void){
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();             //Initialize the MPU object
  radio.begin();                //Start the nRF24 communicate
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();   //Sets the address of the receiver to which the program will send data.
}

void loop(void){

  //With this function, the acceleration and gyro values of the axes are taken.
```

```
  //If you want to control the car axis differently, you can change the axis name in the map command.
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  // Serial.println(ax); // only for testing


  //In two-way control, the X axis (data [0]) of the MPU6050 allows the robot to move forward and
backward.
  //Y axis (data [0]) allows the robot to right and left turn.
  data[0] = map(ax, -17000, 17000, 300, 400 ); //Send X axis data to range between 300-400
  data[1] = map(ay, -17000, 17000, 100, 200);  //Send Y axis data to range between 100-200

  //For testing, make sure that values printed to serial monitor change when remote is tilted
  Serial.print("Data 1: ");
  Serial.println(data[0]);
  Serial.print("Data 2: ");
  Serial.println(data[1]);

  radio.write(&data, sizeof(data)); //send X-Y data via radio address
  delay(100); // update 10 times per second
}
```

Receiver code appendix:

```
//Add the necessary libraries
#include <SPI.h>        //SPI library for communicate with the nRF24L01+
#include "RF24.h"
#include <nRF24L01.h>   //The main library of the nRF24L01+

//Define enable pins of the Motors
const int enbA = 3; //Left Vref
const int enbB = 5;  //right Vref

//Define direction control pins of the Motors
const int IN1 = 2;      //Right Motor (+)
const int IN2 = 4;      //Right Motor (-)
const int IN3 = 9;      //Left Motor (+)
const int IN4 = 10;     //Left Motor (-)

//Define variable for the motors speeds
//This way you can synchronize the rotation speed difference between the two motors
//Set both to 150 if no speed difference between motors
int RightSpd = 150;
int LeftSpd = 150;


//Define object from RF24 library - 7 and 8 are digital pin numbers to which signals CE and CSN are
connected
RF24 radio(7,8); // radio CE CSN pins

//Create a pipe addresses for the communicate
const byte address[6] = "00001"; //can change to any address shared with transmitter

void setup(){
  //Define the motor pins as OUTPUT
```

```
  pinMode(enbA, OUTPUT);
  pinMode(enbB, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  Serial.begin(9600);
  radio.begin();                        //Start the nRF24 communicate
  radio.openReadingPipe(0, address); //Search for signal from defined address
  radio.setPALevel(RF24_PA_MIN);   //Sets the address of the transmitter to which the program will
receive data.
  radio.startListening();               //Starts receiving
  }

void loop(){
  if (radio.available()){
        int data[2];  //Define packet for the direction (X axis and Y axis)
        radio.read(&data, sizeof(data)); //read X-Y data
        Serial.print("Data 1: ");  //Print X-Y data to seriel monitior, this is helpful for testing
        Serial.println(data[0]); //X-data
        Serial.print("Data 2: ");
        Serial.println(data[1]); // Y-data

        if(data[0] > 380){  // X-condition for forward motion, change to alter sensitivity
        //forward
        analogWrite(enbA, RightSpd); //set Vref
        analogWrite(enbB, LeftSpd);  //set Vref
        digitalWrite(IN1, HIGH); //right forward on
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, HIGH); //left forward on
        digitalWrite(IN4, LOW);
        // Serial.println("1"); // for testing only
        }

        if(data[0] < 310){ // X-condition for backward motion, change to alter sensitivity
        //backward
        analogWrite(enbA, RightSpd); //set Vref
        analogWrite(enbB, LeftSpd); //set Vref
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH); //right backward on
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH); //left backward on
        // Serial.println("2"); //for testing only
        }

        if(data[1] > 180){ // Y-condition for left motion, change to alter sensitivity
        //left
        analogWrite(enbA, RightSpd); //set Vref
        analogWrite(enbB, LeftSpd); //set Vref
        digitalWrite(IN1, HIGH); //right forward on
        digitalWrite(IN2, LOW);
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH); // left backward on
        //  Serial.println("3");   //for testing only
        }

        if(data[1] < 110){ // Y-condition for left motion, change to alter sensitivity
```