Hand Gesture Controlled robot

(ربات کنترلی با حرکات دست)

محمد ایمانیه

موسسه آموزش عالی آپادانا - 1400120280015

—

مبانی رباتیک

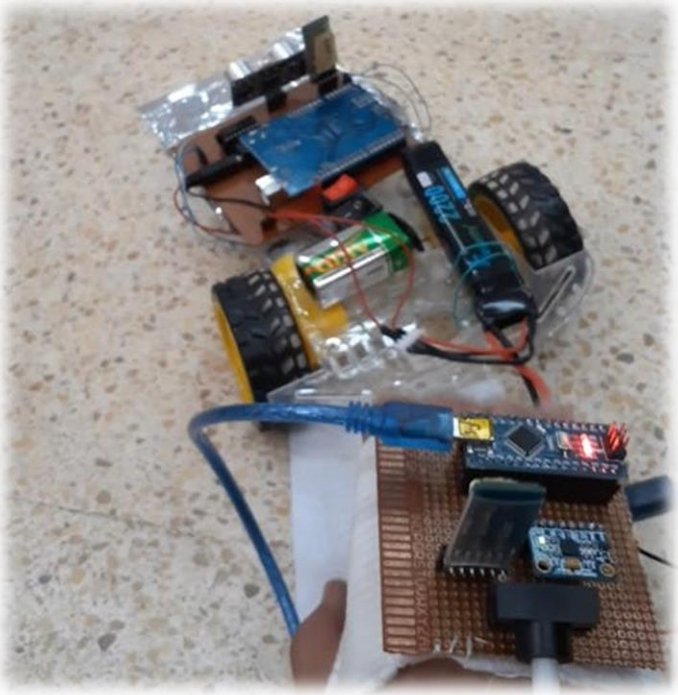
—

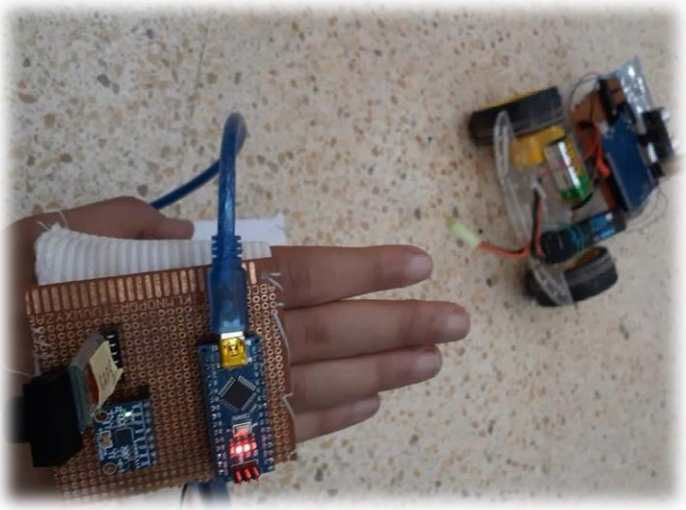
استاد: محمد زارع

تیر 1403

چکیده

این پروژه، یک ربات کنترلی با حرکات دست است. این پروژه شامل دو بخش اصلی است: واحد ارسال که شامل آردوینو نانو، سنسور mpu6050 ، ماژول بلوتوث HC-05 ، دکمه فشاری و باتری 9 ولت است که همه قطعات روی یک برد PCB جوش داده شده‌اند.
واحد دریافتی شامل آردوینو UNO ، ماژول بلوتوث، درایور موتور H-bridge l293d، باتری 9 ولت برای تغذیه آردوینو و سایر قطعات، و باتری 12 ولت برای تغذیه موتورها، به همراه سنسور اولتراسونیک برای جلوگیری از تصادفات است.
واحد ارسال با آردوینو نانو و سایر قطعات برای تشخیص حرکات دست، و **واحد دریافتی** که ربات خودرویی است مجهز به آردوینو UNO و سنسورها. داده‌های حرکتی از سنسورها توسط آردوینو نانو پردازش شده و به کمک ماژول بلوتوث به ربات ارسال می‌شوند تا جهت حرکت ربات را کنترل کنند. کاراکترهای 'L', 'R', 'B', 'F', و 'S' برای کنترل جهات مختلف حرکتی ربات استفاده می‌شوند.





THE PROCESS

مقدمه:

این پروژه یک ربات کنترلی است که با حرکات دست هدایت می‌شود و شامل دو بخش اصلی می‌باشد:

1. واحد ارسال: این بخش شامل یک آردوینو نانو است که به عنوان مغز مرکزی عمل می‌کند و داده‌های حرکتی را از سنسور MPU6050 دریافت می‌کند. این سنسور قادر به تشخیص حرکات و جهت‌گیری دست است. همچنین، یک ماژول بلوتوث HC-05 برای ارسال داده‌ها به واحد دریافتی، یک دکمه فشاری برای فعال یا غیرفعال کردن سیستم، و یک باتری 9 ولت برای تأمین انرژی تمام قطعات موجود بر روی یک برد PCB جوش داده شده‌اند.

2. واحد دریافتی: این بخش شامل یک آردوینو UNO است که داده‌های ارسالی از واحد ارسال را دریافت می‌کند. یک درایور موتور H-bridge L293D برای کنترل موتورهای ربات، یک باتری 9 ولت برای تغذیه آردوینو و سایر قطعات، و یک باتری 12 ولت برای تغذیه موتورها به کار رفته است. علاوه بر این، یک سنسور اولتراسونیک برای جلوگیری از برخورد ربات با موانع نصب شده است.

داده‌های حاصل از سنسور MPU6050 توسط آردوینو نانو پردازش شده و بر اساس زاویه‌های حرکت دست، کاراکترهای خاصی که جهت حرکت ربات را مشخص می‌کنند، از طریق ماژول بلوتوث به آردوینو UNO در ربات ارسال می‌شوند. برای مثال، اگر مقدار محور **x** کمتر از **-17** باشد، کاراکتر 'F' برای حرکت به جلو ارسال می‌شود، اگر بیشتر از **20** باشد، کاراکتر 'B' برای حرکت به عقب، اگر مقدار محور **y** بیشتر از **30** باشد، کاراکتر 'R' برای حرکت به راست، و اگر کمتر از **-30** باشد، کاراکتر 'L' برای حرکت به چپ ارسال می‌شود. در صورتی که هیچ‌کدام از این شرایط برآورده نشوند، کاراکتر 'S' برای توقف ربات ارسال می‌شود.

کاربرد:

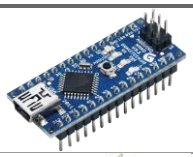
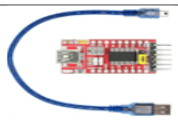
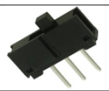
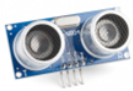
این پروژه ربات کنترلی با حرکات دست می‌تواند در زمینه‌های مختلفی کاربرد داشته باشد، از جمله:

- آموزشی: به عنوان یک ابزار آموزشی برای دانش‌آموزان و دانشجویان در زمینه‌های الکترونیک، رباتیک و برنامه‌نویسی.
- توانبخشی: برای کمک به افراد دارای معلولیت جسمی که نیاز به تمرینات حرکتی دست دارند.
- سرگرمی: به عنوان یک اسباب‌بازی پیشرفته که می‌تواند تجربه‌ای جذاب و تعاملی برای کودکان و بزرگسالان فراهم کند.
- پژوهشی: در پروژه‌های تحقیقاتی برای بررسی رابطهای کاربری جدید و ارتباط بین انسان و ماشین.
- صنعتی: در کارخانه‌ها و محیطهای صنعتی برای کنترل دستگاه‌ها و ماشین‌آلات از راه دور.
- پزشکی: ممکن است در آینده بتوان از این تکنولوژی برای کنترل دقیق ابزارهای جراحی استفاده کرد.
- نظامی: در سیستم‌های کنترل از راه دور برای هدایت وسایل نقلیه بدون سرنشین یا ربات‌های اکتشافی.

این پروژه نمونه‌ای از نوآوری در تکنولوژی است که می‌تواند در بسیاری از بخش‌ها تأثیرگذار باشد و زمینه‌های جدیدی را برای کاربردهای عملی و تحقیقاتی باز کند. امکانات استفاده از آن بسیار گسترده است و با پیشرفت تکنولوژی، کاربردهای بیشتری برای آن شناسایی خواهد شد.

روش کار:

1. مواد و قطعات مورد نیاز:

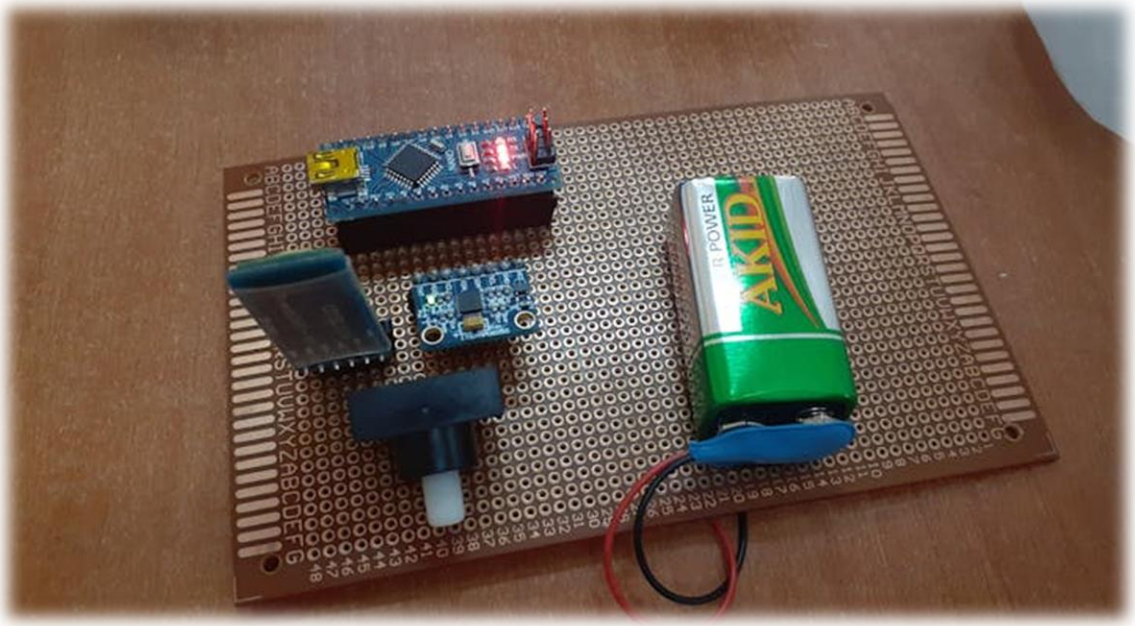
نام قطعه	تعداد مورد نیاز	بخش به کار رفته	توضیحات	عکس
آردوینو نانو R3	×1	واحد ارسال (روی دست)	در واحد ارسالی به کار می‌رود و داده‌های حرکتی دست را از سنسور MPU6050 دریافت و پردازش می‌کند	
سنسور MPU6050	×1	واحد ارسال (روی دست)	سنسور MPU6050 حرکات و جهت‌گیری دست را اندازه‌گیری می‌کند و اطلاعات را برای کنترل جهت حرکت ربات به آردوینو نانو ارسال می‌کند.	
دکمه فشاری (کلید فشاری، سری APEM A01)	×1	واحد ارسال (روی دست)	به عنوان دکمه فعال/غیرفعال سازی سیستم در واحد ارسالی به کار می‌رود.	
مبدل USB به سریال FTDI) FT232RL (Converter	×1	واحد ارسال (روی دست)	برای بارگذاری کد به آردوینو نانو استفاده می‌شود استفاده می‌شود، زیرا آردوینو نانو R3 به طور مستقیم پورت USB برای اتصال به کامپیوتر ندارد	
آردوینو UNO	×1	واحد دریافتی (ربات)	به عنوان مغز اصلی واحد دریافتی عمل می‌کند و دستورات را برای کنترل ربات پردازش می‌کند.	
درایور موتور دوگانه H-Bridge L293D	×1	واحد دریافتی (ربات)	این درایور در واحد دریافتی به کار می‌رود و به آردوینو UNO اجازه می‌دهد تا جریان الکتریکی موتورها را کنترل کند	
باتری قابل شارژ، لیتیوم یون (برای تغذیه موتورها)	×1	واحد دریافتی (ربات)	برای تغذیه موتورهای ربات در واحد دریافتی به کار می‌رود.	
کلید لغزنده E-SWITCH EG1218	×1	واحد دریافتی (ربات)	ممکن است برای تغییر حالت‌های عملیاتی ربات در واحد دریافتی استفاده شود.	
چرخ‌های ربات	×2	واحد دریافتی (ربات)	برای ساخت اسکلت حرکتی ربات در واحد دریافتی به کار می‌روند	
شاسی ربات	×1	واحد دریافتی (ربات)	برای ساخت بدنه ربات در واحد دریافتی به کار می‌روند	
سنسور اولتراسونیک HC-SR04 (عمومی)	×1	واحد دریافتی (ربات)	در واحد دریافتی به کار می‌رود و به ربات کمک می‌کند تا از برخورد با موانع جلوگیری کند.	
ماژول بلوتوث HC-05	×2	هر دو واحد	در هر دو واحد به کار می‌رود و ارتباط بی‌سیم بین واحد ارسالی و دریافتی را برقرار می‌کند.	
برد PCB	×2	هر دو واحد	برای نصب و اتصال قطعات الکترونیکی در هر دو واحد استفاده می‌شود	
باتری 9 ولت (عمومی)	×2	هر دو واحد	برای تأمین انرژی آردوینو نانو و یونو در هر دو واحد استفاده می‌شود.	

2.مراحل اجرا:

برای ساخت ربات کنترلی با حرکات دست، مراحل زیر را طی میکنیم:

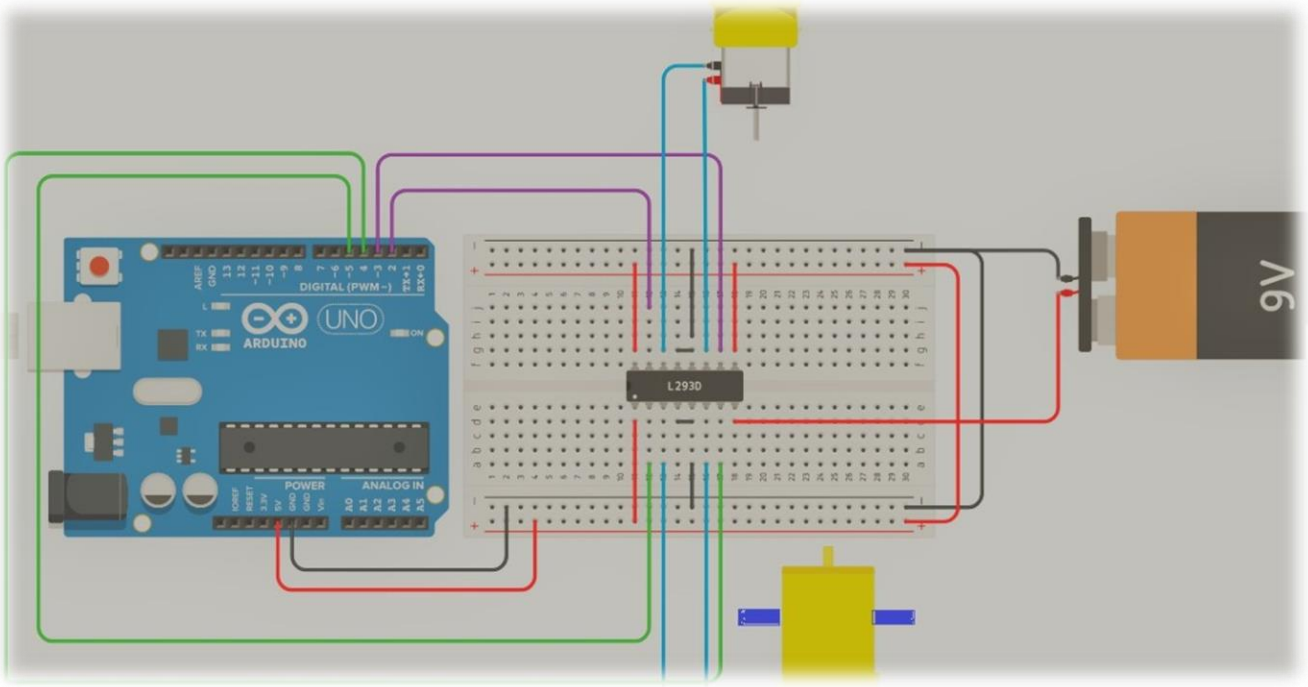
1) ساخت واحد ارسالی:

- **مونتاژ آردوینو نانو:** ابتدا آردوینو نانو را روی برد PCB قرار دهید وپایه های آن را لحیم کنید.
- **نصب سنسور MPU6050:** سنسور MPU6050 را به آردوینو نانو متصل کنید.
  - VCC به V5 آردوینو
  - GND به GND آردوینو
  - SCL به A5 آردوینو
  - SDA به A4 آردوینو
- **افزودن ماژول بلوتوث HC-05 :** ماژول بلوتوث را به آردوینو نانو متصل کنید تا بتوانید داده‌ها را به ربات ارسال کنید.
  - TX آردوینو به RX ماژول بلوتوث
  - RX آردوینو به TX ماژول بلوتوث
  - GND به GND
  - VCC به V3.3 یا V5 (بسته به ماژول)
- **قرار دادن دکمه فشاری:** دکمه فشاری را برای فعال یا غیرفعال کردن سیستم نصب کنید.
  - یک سر دکمه به پین دیجیتال آردوینو
  - سر دیگر دکمه به GND
- **اتصال باتری 9 ولت:** برای تغذیه سیستم، باتری 9 ولت را متصل کنید.
  - مثبت به VIN آردوینو
  - منفی به GND



2) ساخت واحد دریافتی:

- **مونتاژ آردوینو UNO:** آردوینو UNO را روی برد PCB قرار دهید آن را لحیم کنید.
- **نصب ماژول بلوتوث:** ماژول بلوتوث را برای دریافت داده‌ها از واحد ارسالی نصب کنید.
  - TX آردوینو به RX ماژول بلوتوث
  - RX آردوینو به TX ماژول بلوتوث
  - GND به GND و VCC به V5
- **افزودن درایور موتور H-bridge L293D:** این درایور را برای کنترل موتورهای ربات نصب کنید.
  - پین‌های کنترل موتور آردوینو به پین‌های ورودی درایور موتور
  - پین‌های تغذیه درایور موتور به باتری 12 ولت
  - GND درایور موتور به GND آردوینو
- **نصب سنسور اولتراسونیک:** این سنسور را برای جلوگیری از برخورد ربات با موانع نصب کنید.
  - VCC به V5 آردوینو
  - GND به GND آردوینو
  - Trig و Echo به پین‌های دیجیتال تعیین شده در آردوینو
- **اتصال باتری‌ها:** یک باتری 9 ولت برای تغذیه آردوینو و یک باتری 12 ولت برای تغذیه موتورها متصل کنید.
  - باتری 9 ولت به جک تغذیه آردوینو UNO
  - باتری 12 ولت به پین‌های تغذیه درایور موتور



3) برنامه‌ریزی آردوینو نانو (کد واحد ارسالی):

- **کدنویسی:** برنامه‌ای بنویسید که داده‌های سنسور را بخواند و زاویه‌های حرکت دست را تعیین کند.
- **تعریف کاراکترهای کنترلی:** برای هر جهت حرکت (جلو، عقب، راست، چپ) یک کاراکتر تعریف کنید.
- **اتصال آردینو نانو به کامپیوتر:** برای اتصال آردوینونانو به کامپیوتر (برای بارگذاری کد)، از FTDI Converter استفاده میکنیم ، باید پایه‌های مربوط به TX و RX را به هم متصل کنید، به این صورت که:
  - FTDI -TX را به RX آردوینو نانو.
  - FTDI - RX را بهTX آردوینو نانو.
  - FTDI -GND را به GND آردوینو نانو.
  - FTDI - VCC به V3.3 یا V5 (بسته به آردوینو نانو)

- همچنین، اگر آردوینو نانو شما نسخه V 3.3است، مطمئن شوید که جامپر تغذیه روی FTDI Converter را روی V 3.3 تنظیم کرده‌اید. این اتصالات اجازه می‌دهند تا کد برنامه‌نویسی از طریق ارتباط سریال به آردوینو نانو بارگذاری شود. این مرحله برای توسعه و آزمایش کد برنامه‌نویسی شما قبل از قرار دادن آن در محیط واقعی پروژه ضروری است.

- **نرم افزار کد نویسی:** برای برنامه‌نویسی و بارگذاری کد بر روی بردهای آردوینو مانند آردوینو نانو R3، از محیط توسعه مجتمع آردوینو (Arduino IDE) استفاده می‌شود. Arduino IDE یک پلتفرم کدنویسی متن‌باز است که به شما امکان می‌دهد به راحتی کد بنویسید، آن را کامپایل کنید و سپس به برد آردوینو خود بارگذاری کنید.

```
// کتابخانه‌های مورد نیاز
#include <Wire.h>
#include <SoftwareSerial.h>

// تعریف پورت‌های سریال برای ارتباط با ماژول بلوتوث
SoftwareSerial BTSerial(10, 11); // RX -> PIN 11 | TX -> PIN 10

// آدرس I2C سنسور MPU6050
const int MPU_ADDRESS = 0x68;

// متغیرهای برای ذخیره داده‌های خام شتاب‌سنج و ژیروسکوپ
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;

// متغیرهای برای ذخیره زاویه‌های محاسبه شده
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;

// متغیرهای برای ذخیره خطاهای سنسور
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;

// متغیرهای برای محاسبه زمان
float elapsedTime, currentTime, previousTime;
```

```
// تابع setup برای راه‌اندازی اولیه
void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600); // شروع ارتباط با ماژول بلوتوث

  Wire.begin(); // شروع ارتباط I2C
  Wire.beginTransmission(MPU_ADDRESS); // آغاز ارتباط با MPU6050
  Wire.write(0x6B); // ارسال به رجیستر 6B
  Wire.write(0x00); // ریست کردن MPU6050
  Wire.endTransmission(true); // پایان ارتباط
}

// تابع loop که در هر چرخه اجرا می‌شود
void loop() {
  MPU_read_accel_data(); // خواندن داده‌های شتاب‌سنج
  MPU_read_gyro_data(); // خواندن داده‌های ژیروسکوپ
  // محاسبه زمان گذشته از آخرین بار
  previousTime = currentTime;
  currentTime = millis();
  elapsedTime = (currentTime - previousTime) / 1000; // تبدیل میلی‌ثانیه به ثانیه
  // فیلتر ترکیبی - ترکیب داده‌های شتاب‌سنج و ژیروسکوپ
  roll = 0.92 * (roll + (GyroX * elapsedTime)) + 0.08 * accAngleX;
  pitch = 0.92 * (pitch + (GyroY * elapsedTime)) + 0.08 * accAngleY;
  // محاسبه زاویه‌ها با استفاده از داده‌های ژیروسکوپ
  gyroAngleX += GyroX * elapsedTime;
  gyroAngleY += GyroY * elapsedTime;
  gyroAngleZ += GyroZ * elapsedTime;
  // چاپ زاویه‌ها بر روی مانیتور سریال
  Serial.print("roll: ");
  Serial.print(roll);
  Serial.print("    ");
  Serial.print("pitch: ");
  Serial.println(pitch);
  // ارسال دستورات به ماژول بلوتوث بر اساس زاویه‌های محاسبه شده
  if(pitch < -17) {
    BTSerial.write('F');
  } else if(pitch > 20) {
    BTSerial.write('B');
  } else if(roll > 30) {
    BTSerial.write('R');
  } else if(roll < -30) {
    BTSerial.write('L');
  } else if(yaw > 30) {
    BTSerial.write('r');
  } else if(yaw < -30) {
    BTSerial.write('l');
  } else {
    BTSerial.write('S');
  }
}
```

```
// توابع برای خواندن داده‌های شتاب‌سنج و ژیروسکوپ
void MPU_read_accel_data() {
  // === خواندن داده‌های شتاب‌سنج === //
  Wire.beginTransmission(MPU_ADDRESS); // آغاز ارتباط با MPU
  Wire.write(0x3B); // 0 شروع با رجیستر x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false); // پایان ارتباط بدون توقف ارتباط
  Wire.requestFrom(MPU_ADDRESS, 6, true); // خواندن 6 رجیستر به طور کامل، هر مقدار محور در 2 رجیستر ذخیره شده است
  // 2-+ برای دامنه gبر اساس دیتاشیت باید مقادیر خام را بر 16384 تقسیم کرد ،
  AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // مقدار محور X
  AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // مقدار محور Y
  AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // مقدار محور Z
  // محاسبه زاویه رول و پیچ از داده‌های شتاب‌سنج
  accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - (-0.40); // AccErrorX تقریبا (0.58) برای
مراجعه کنید calculate_IMU_error() جزئیات بیشتر به تابع سفارشی
  accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ, 2))) * 180 / PI) - (-3.75); // AccErrorY تقریبا (-
1.58)
}
//----------------------------------------
void MPU_read_gyro_data(){
  // === خواندن داده‌های ژیروسکوپ === //
  Wire.beginTransmission(MPU_ADDRESS); // آغاز ارتباط با MPU
  Wire.write(0x43); // 0 آدرس اولین رجیستر داده‌های ژیروسکوپ x43
  Wire.endTransmission(false); // پایان ارتباط بدون توقف ارتباط
  Wire.requestFrom(MPU_ADDRESS, 6, true); // خواندن 4 رجیستر به طور کامل، هر مقدار محور در 2 رجیستر ذخیره شده است
  // برای دامنه 250 درجه بر ثانیه باید ابتدا مقدار خام را بر 131.0 تقسیم کرد، بر اساس دیتاشیت
  GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // مقدار محور X
  GyroY = (Wire.read() << 8 | Wire.read()) / 131.0; // مقدار محور Y
  GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0; // مقدار محور Z
  // تصحیح خروجی‌ها با مقادیر خطای محاسبه شده
  GyroX = GyroX - (-0.68); // GyroErrorX تقریبا (-2.12)
  GyroY = GyroY - (-2.48); // GyroErrorY تقریبا (4.12)
  GyroZ = GyroZ - (-0.12); // GyroErrorZ تقریبا (1.20)
}
//----------------------------------------
void calculate_IMU_error() {
// فراخوانی کنیم تا خطای داده‌های شتاب‌سنج و ژیروسکوپ را محاسبه کنیم. از اینجا می‌توانیم مقادیر خطا استفاده شده در معادلات بالا را روی مانیتور سریال چاپ کنیم setup ما می‌توانیم این تابع را در بخش
  // را به صورت صاف قرار دهیم تا مقادیر صحیح را دریافت کنیم، بنابراین می‌توانیم مقادیر صحیح را بدست آوریم IMU توجه داشته باشید که باید
  // خواندن مقادیر شتاب‌سنج 200 بار
  while (c < 200) {
    Wire.beginTransmission(MPU_ADDRESS);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_ADDRESS, 6, true);
    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;
    // جمع کل خواندن‌ها
    AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) * 180 / PI));
    AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) * 180 / PI)); c++; }
```

```
// تقسیم جمع بر 200 برای بدست آوردن مقدار خطا
  AccErrorX = AccErrorX / 200;
  AccErrorY = AccErrorY / 200;
  c = 0;
  // خواندن مقادیر ژیروسکوپ 200 بار
  while (c < 200) {
    Wire.beginTransmission(MPU_ADDRESS);
    Wire.write(0x43);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_ADDRESS, 6, true);
    GyroX = Wire.read() << 8 | Wire.read();
    GyroY = Wire.read() << 8 | Wire.read();
    GyroZ = Wire.read() << 8 | Wire.read();
    // جمع کل خواندن‌ها
    GyroErrorX = GyroErrorX + (GyroX / 131.0);
    GyroErrorY = GyroErrorY + (GyroY / 131.0);
    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
    c++;
  }
  // تقسیم جمع بر 200 برای بدست آوردن مقدار خطا
  GyroErrorX = GyroErrorX / 200;
  GyroErrorY = GyroErrorY / 200;
  GyroErrorZ = GyroErrorZ / 200;

  // چاپ مقادیر خطا روی مانیتور سریال
  Serial.print("AccErrorX: ");
  Serial.print(AccErrorX);
  Serial.print(" | AccErrorY: ");
  Serial.print(AccErrorY);
  Serial.print(" | GyroErrorX: ");
  Serial.print(GyroErrorX);
  Serial.print(" | GyroErrorY: ");
  Serial.print(GyroErrorY);
  Serial.print(" | GyroErrorZ: ");
  Serial.println(GyroErrorZ);
  delay(1000);}
```

توضیحات توابع:

- **setup ()**: در آردوینو برای ارسال داده‌ها، ارتباطات سریال و I2C را با سرعت 9600 بیت بر ثانیه برقرار و سنسور MPU6050 را پیکربندی می‌کند. این تابع در شروع برنامه فراخوانی شده و تنظیمات اولیه لازم برای عملکرد برنامه را تعیین می‌کند که تا زمان ریست یا بارگذاری مجدد دستگاه، ثابت باقی می‌مانند.
- **loop()**: این تابع در هر چرخه اجرا می‌شود و وظیفه خواندن داده‌های شتاب‌سنج و ژیروسکوپ، محاسبه زمان گذشته، و ارسال دستورات به ماژول بلوتوث را بر عهده دارد.
- **MPU_read_accel_data()** : این تابع داده‌های شتاب‌سنج را از MPU6050 می‌خواند و زاویه‌های محاسبه شده از شتاب را تعیین می‌کند
- **MPU_read_gyro_data()**:این تابع داده‌های ژیروسکوپ را از MPU6050 می‌خواند و خطاهای محاسبه شده را اصلاح می‌کند.
- **calculate_IMU_error()**:این تابع برای محاسبه خطای شتاب‌سنج و ژیروسکوپ استفاده می‌شود و مقادیر خطا را بر روی مانیتور سریال چاپ می‌کند.

4) برنامه‌ریزی آردوینو UNO: (کد واحد دریافتی):

- **کدنویسی:** برنامه‌ای بنویسید که داده‌های دریافتی از ماژول بلوتوث را بخواند و موتورها را بر اساس آن کنترل کند.
- **نرم افزار کد نویسی:** برای برنامه‌نویسی و بارگذاری کد بر روی بردهای آردوینو UNO، از محیط توسعه مجتمع آردوینو (Arduino IDE) استفاده می‌شود. Arduino IDE یک پلتفرم کدنویسی متن‌باز است که به شما امکان می‌دهد به راحتی کد بنویسید، آن را کامپایل کنید و سپس به برد آردوینو خود بارگذاری کنید.

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(10, 11); // RX -> PIN 11 | TX -> PIN 10

// تعریف پین‌ها برای سنسور فاصله و موتورها
int trigPin = 12;    // Trig
int echoPin = 13;    // Echo
long duration, cm, inches;
int direct;
int l1=3; // موتور چپ جلو
int r1=5; // موتور راست جلو
int l2=6; // موتور چپ عقب
int r2=9; // موتور راست عقب

// متغیرهای کمکی برای تصحیح حرکت
int forward_corr;

// متغیرهای کمکی برای تابع backward
int a = 0;
int s = 3;
int b = 0;
int c = 2;

// تابع setup برای راه‌اندازی اولیه
void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600); // شروع ارتباط با ماژول بلوتوث

  // تنظیم پین‌های موتور به عنوان خروجی
  pinMode(l1, OUTPUT);
  pinMode(r1, OUTPUT);
  pinMode(l2, OUTPUT);
  pinMode(r2, OUTPUT);

  // اطمینان از خاموش بودن موتورها در ابتدا
  digitalWrite(l1, LOW);
  digitalWrite(r1, LOW);
  digitalWrite(l2, LOW);
  digitalWrite(r2, LOW);

   // تنظیم پین‌های سنسور فاصله
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

```
// تابع loop که در هر چرخه اجرا می‌شود
void loop() {
  // اندازه‌گیری فاصله توسط سنسور فاصله
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  cm = (duration/2) / 29.1; // محاسبه فاصله به سانتی‌متر

  // چاپ فاصله بر روی مانیتور سریال
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();

  // دریافت داده‌ها از ماژول بلوتوث و اجرای دستورات مربوطه
  if(BTSerial.available()>0){
    direct = BTSerial.read();
    switch(direct) {
      case 'B': Serial.println("Backward"); backward(); break;
      case 'F': Serial.println("Forward"); forward(); break;
      case 'R': Serial.println("Right"); left(); break;
      case 'L': Serial.println("Left"); right(); break;
      case 'r': Serial.println("Rotate Right"); rotr(); break;
      case 'l': Serial.println("Rotate Left"); rotl(); break;
      case 'S': Serial.println("Stop"); stopCar(); break;
    }
  }
}
// توابع برای کنترل حرکت ربات
void forward() {
  // اگر فاصله کمتر از 15 سانتی‌متر باشد، ربات را متوقف کن
  if(cm <= 15){
    stopCar();
  } else {
    // در غیر این صورت، ربات را به جلو حرکت ده
    digitalWrite(l1, HIGH);
    digitalWrite(r1, LOW);
    digitalWrite(l2, LOW);
    digitalWrite(r2, HIGH);
  }
}
void backward() {
  // ربات را به عقب حرکت ده
  digitalWrite(l1, LOW);
  digitalWrite(l2, HIGH);
  digitalWrite(r1, HIGH);
  digitalWrite(r2, LOW);
}
```

```
void right() {
  // ربات را به راست حرکت ده
  digitalWrite(l1, HIGH);
  digitalWrite(r1, LOW);
  digitalWrite(l2, LOW);
  digitalWrite(r2, LOW);
}
void left() {
  // ربات را به چپ حرکت ده
  digitalWrite(l1, LOW);
  digitalWrite(r1, LOW);
  digitalWrite(l2, LOW);
  digitalWrite(r2, HIGH);
}
void stopCar() {
  // تمام موتورها را خاموش کن تا ربات متوقف شود
  digitalWrite(l1, LOW);
  digitalWrite(r1, LOW);
  digitalWrite(l2, LOW);
  digitalWrite(r2, LOW);
}
void rotr() {
  // ربات را به سمت راست بچرخان
  digitalWrite(l1, HIGH);
  digitalWrite(r1, LOW);
  digitalWrite(l2, HIGH);
  digitalWrite(r2, LOW);
}
void rotl() {
  // ربات را به سمت چپ بچرخان
  digitalWrite(l1, LOW);
  digitalWrite(r1, HIGH);
  digitalWrite(l2, LOW);
  digitalWrite(r2, HIGH);
}
```

توضیحات توابع:

- **setup()**: این تابع برای راه‌اندازی اولیه و تنظیم پین‌ها و ارتباطات استفاده می‌شود. ارتباط سریال و بلوتوث را با سرعت 9600 بیت بر ثانیه شروع می‌کند و پین‌های موتور و سنسور فاصله را تنظیم می‌کند.
- **loop()**: این تابع در هر چرخه از اجرای برنامه فراخوانی می‌شود. فاصله را با استفاده از سنسور فاصله اندازه‌گیری می‌کند و دستورات دریافتی از ماژول بلوتوث را اجرا می‌کند.
- **Forward()**: ربات را به جلو حرکت می‌دهد، مگر اینکه فاصله کمتر از 15 سانتی‌متر باشد که در آن صورت ربات را متوقف می‌کند.
- **backward()**: ربات را به عقب حرکت می‌دهد.
- **right()**: ربات را به راست حرکت می‌دهد.
- **left()**: ربات را به چپ حرکت می‌دهد.

- **stopCar()**: تمام موتورها را خاموش می‌کند تا ربات متوقف شود.
- **rotr()**: ربات را به سمت راست می‌چرخاند.
- **rotl()**: ربات را به سمت چپ می‌چرخاند.

این توابع به شما امکان می‌دهند تا ربات را با دستورات ارسالی از طریق بلوتوث کنترل کنید. هر دستور با یک کاراکتر نمایش داده شده و باعث فراخوانی تابع مربوطه می‌شود تا عملیات مورد نظر را انجام دهد.

5) تست و اشکال‌زدایی:

- **تست حرکت:** ربات را در محیطی امن قرار دهید و حرکات دست را برای کنترل جهت‌های مختلف تست کنید.
- **اشکال‌زدایی:** اگر ربات به درستی کار نمی‌کند، کد و اتصالات خود را بررسی کنید.

نتیجه گیری:

پروژه ربات کنترلی با حرکات دست نمونه‌ای از نوآوری در ترکیب سخت‌افزار و نرم‌افزار است و به کاربر امکان می‌دهد تا با حرکات دست خود، یک ربات را کنترل کند. این پروژه، که از آردوینو نانو و ماژول MPU6050 برای تشخیص حرکات دست و ارسال دستورات به ربات استفاده می‌کند، نه تنها برای هواداران رباتیک و دانشجویان مهندسی جذاب است، بلکه پتانسیل کاربردهای عملی در زمینه‌هایی مانند کمک به افراد دارای معلولیت یا سرگرمی را نیز دارد.

این پروژه نشان می‌دهد که چگونه می‌توان با استفاده از قطعات مقرون‌به‌صرفه و دسترس، سیستم‌های کنترلی پیچیده‌ای را طراحی و پیاده‌سازی کرد. همچنین، این پروژه به عنوان یک ابزار آموزشی ارزشمند عمل می‌کند که می‌تواند مفاهیم اساسی الکترونیک، برنامه‌نویسی و رباتیک را به طور عملی به دانش‌آموزان و علاقه‌مندان آموزش دهد. در نهایت، این پروژه می‌تواند به عنوان بستری برای توسعه و اجرای ایده‌های خلاقانه‌تر در آینده به کار رود و الهام‌بخش ساخت پروژه‌های پیشرفته‌تر در حوزه رباتیک باشد.

منابع:

Hackster.io آدرس پروژه اصلی در سایت
https://projecthub.arduino.cc
https://www.arduino.cc
https://copilot.microsoft.com
https://youtu.be/Sj9lhk6dB1U?t=5