

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | کنترل دست رباتیک با استفاده از EMG | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | هانیه روزمند |  | | | |
|  | | | | اصول رباتیکزاستاد: محمد زارعاردیبهشت 1403 |  | | | |
|  | | |  | | |  | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  | | |  |
|  | چکیده | | | | | | |  |
|  |  | | |  |  | | |  |
|  |  | |  | | |  | |  |
|  |  |  | این پروژه با عنوان “کنترل دست رباتیک با استفاده از EMG”” توسط Dmytro Dziuba منتشر شده است. در این پروژه، از سیگنال‌های الکتریکی عضلات (EMG) برای کنترل انگشتان دست رباتیک استفاده می‌شود. از سه دستگاه uECG برای دریافت سیگنال EMG استفاده می‌شود. این سیگنال‌ها توسط میکروکنترلر Arduino Nano R3 پردازش می‌شوند. برای تشخیص حرکات انگشتان، از مقادیر میانگین سه کانال EMG استفاده می‌شود. این پروژه به کنترل دست رباتیک inMoov متصل شده است. اگر به دنبال راه‌ حلی ساده ‌تر و ارزان‌تر هستید، تیم ما یک حسگر EMG به نام uMyo توسعه داده است. این حسگر نیز نتایج مشابه را با روشی ساده‌تر ارائه می‌دهد. | | |  |  |  |
|  | | | | |
|  |  |  |  |
|  |  |

**EMG**

در عبارت ساده‌تر، EMG یا الکترومیوگرافی یک روش تشخیصی است که برای ارزیابی سلامت عضلات و سلول‌های عصبی کنترل‌کننده آن‌ها (نورون‌های حرکتی) استفاده می‌شود. در این روش، با استفاده از الکترودها، فعالیت الکتریکی عضلات و اعصاب اندازه‌گیری می‌شود. نتایج EMG می‌توانند نقص عصبی، نقص عضلانی یا مشکلات در انتقال سیگنال از عصب به عضله را نشان دهند.

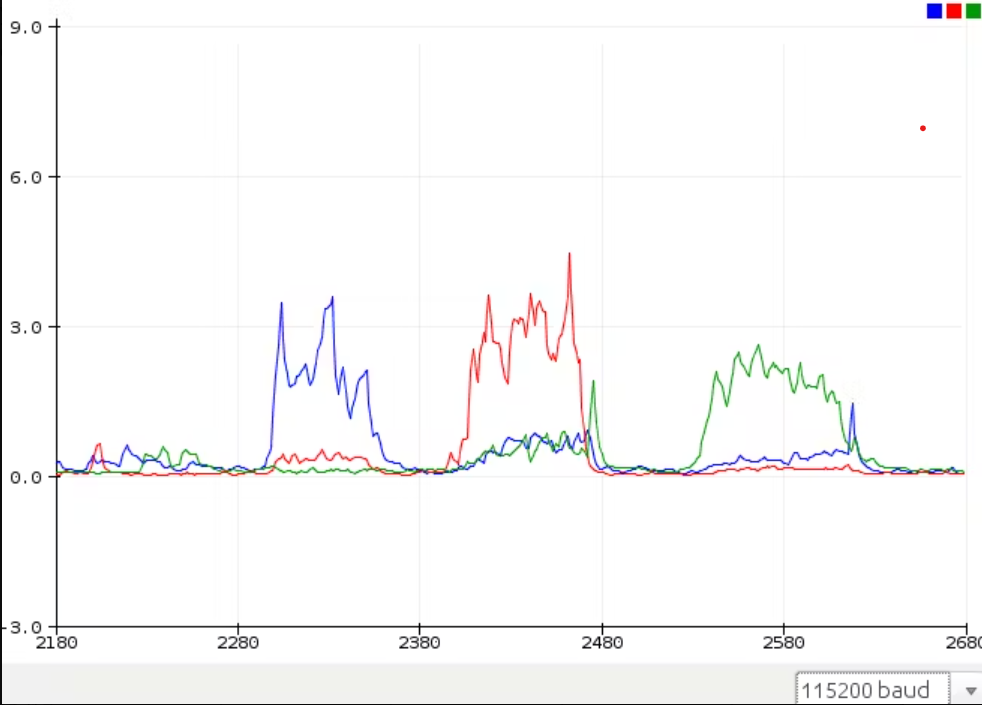
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Decorative | |  |  |  |  | |
|  | | THE PROCESS | | |  | |
|  | |  |  |  |  | |
|  | مقدمه:  از زمانی که این پروژه توسعه یافته است، تیم ما یک حسگر EMG به نام uMyo را توسعه داده است که قیمتی ارزان‌تر دارد و به راحتی قابل استفاده است. این حسگر نتایج مشابه را با روشی ساده‌تر ارائه می‌دهد. حالا به پروژه برگردیم: تیم ما تجربه‌ی طولانی در زمینه دست‌های رباتیک دارد. مدتی سعی کردیم دست مصنوعی قابل اعتمادی بسازیم، اما در این پروژه از یک مثال خوب از دست مصنوعی متن ‌باز به نام inMoov استفاده می‌کنیم. | | | | |  |

|  |  |  |
| --- | --- | --- |
|  | روش کار  1. **قطعات سخت افزاری**:    * دستگاه uECG (سه عدد)    * دست inMoov    * آردوینو نانو R3    * درایور 16 کانال PWM Adafruit PCA9685    * ماژول nRF24 (عمومی) 2. **مراحل اجرا:**   **.1 پردازش سیگنال:**  کنترل بر اساس فعالیت الکتریکی عضلات (EMG) انجام می‌شود. سیگنال EMG توسط سه دستگاه uECG (که در واقع یک مانیتور ECG است، اما به دلیل استفاده از یک ADC عمومی، می‌تواند هر نوع سیگنال زیستی را اندازه‌گیری کند - از جمله EMG) به دست می‌آید. برای پردازش EMG، uECG یک حالت ویژه دارد که در آن داده‌های طیفی 32 بینی و میانگین “پنجره عضله” (میانگین شدت طیفی بین 75 و 440 هرتز) را ارسال می‌کند. تصاویر طیفی به این شکل هستند:    در این تصاویر، فرکانس در محور عمودی قرار دارد (در هر یک از 3 نمودار، فرکانس پایین در پایین، و فرکانس بالا در بالا - از 0 تا 488 هرتز با گام‌های تقریبی 15 هرتز). زمان در محور افقی قرار دارد (داده‌های قدیمی در سمت چپ، کلیتاً حدود 10 ثانیه روی صفحه نمایش است). شدت با رنگ‌ها کد شده است: آبی - پایین، سبز – متوسط ، زرد - بالا، قرمز - حتی بیشتر. برای تشخیص حرکات قابل اعتماد، پردازش مناسب تصاویر توسط کامپیوتر لازم است. اما برای فعال‌سازی ساده انگشتان دست رباتیک، کافیست از مقدار میانگین در 3 کانال استفاده کنید - uECG این مقدار را به صورت مناسب در بایت‌های خاصی ارائه می‌دهد تا نرم‌افزار Arduino بتواند آن را تجزیه و تحلیل کند. این مقادیر به سادگی به نظر می‌آیند:    ​ |  |

نمودارهای قرمز، سبز و آبی مقادیر خام از دستگاه‌های uECG روی گروه‌های مختلف عضلات هنگام فشردن انگشت شست، انگشتان نیمرخ و انگشتان وسطی به ترتیب مربوط می‌شوند. برای چشم ما، این موارد به وضوح متفاوت هستند، اما ما باید این مقادیر را به نوعی به “امتیاز انگشت” تبدیل کنیم تا یک برنامه بتواند مقادیر را به سروموتورهای دست خروجی دهد. مشکل این است که سیگنال‌ها از گروه‌های عضلات “مخلوط” هستند: در مورد اول و سوم، شدت سیگنال آبی تقریباً یکسان است - اما قرمز و سبز متفاوت هستند. در مورد دوم و سوم، سیگنال‌های سبز یکسان هستند - اما آبی و قرمز متفاوت هستند. برای “جداسازی” آن‌ها، از یک فرمول نسبتاً ساده استفاده شده:

**S0=V0^2 / (( V1 \*a0 +b0)( V2 \* c0+d0))**

در این فرمول، (S\_0) نمره برای کانال 0، (V\_0)، (V\_1)، (V\_2) مقادیر خام برای کانال‌های 0، 1 و 2، و (a)، (b)، (c)، (d) ضرایبی هستند که به صورت دستی تنظیم شده‌اند .مقادیر (a) و (c) از 0.3 تا 2.0 بودند، و (b) و (d) به ترتیب 15 و 20 بودند. شما باید آن‌ها را برای تنظیم موقعیت خاص حسگر خود تغییر دهید). همین نمره برای کانال‌های 1 و 2 نیز محاسبه شد.( پس از این، نمودارها تقریباً به طور کامل از هم جدا شدند.



**2.شماتیک:**

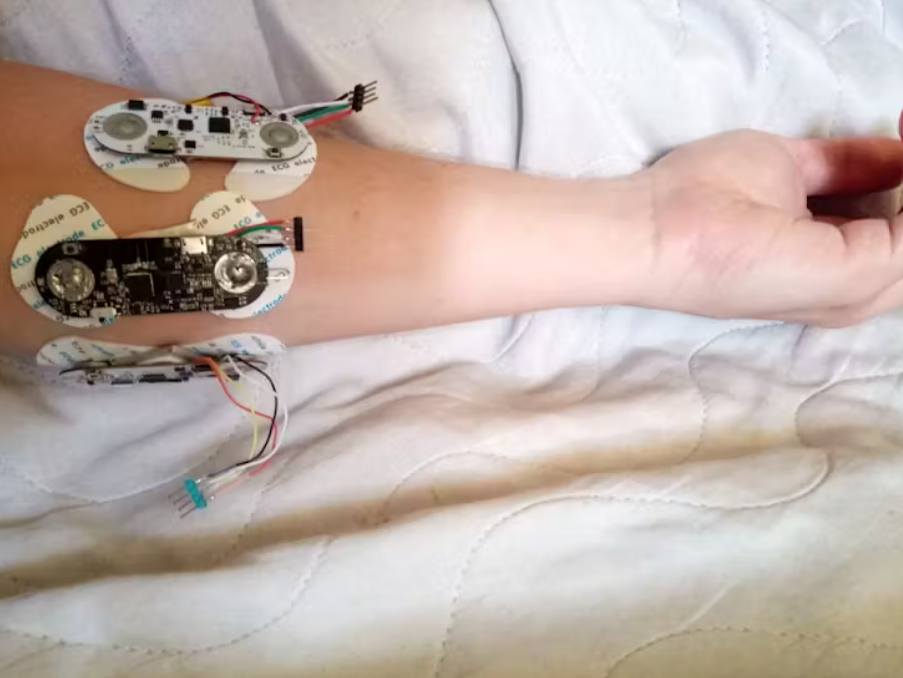
شماتیک بسیار ساده است. شما فقط به یک ماژول nRF24، کنترل‌کننده PCA9685 یا مشابه آن با رابط I2C، و منبع تغذیه 5 ولت با جریان بالا نیاز دارید که برای حرکت همه‌ی این سرو موتورها به طور همزمان کافی است (بنابراین حداقل توان مورد نیاز برای عملکرد پایدار حداقل 5 آمپر است).

**لیست اتصالات:**

* nRF24 pin 1 (GND) - Arduino's GND
* nRF24 pin 2 (Vcc) - Arduino's 3.3v
* nRF24 pin 3 (Chip Enable) - Arduino's D9
* nRF24 pin 4 (SPI:CS) - Arduino's D8
* nRF24 pin 5 (SPI:SCK) - Arduino's D13
* nRF24 pin 6 (SPI:MOSI) - Arduino's D11
* nRF24 pin 7 (SPI:MISO) - Arduino's D12
* PCA9685 SDA - Arduino's A4
* PCA9685 SCL - Arduino's A5
* PCA9685 Vcc - Arduino's 5v
* PCA9685 GND - Arduino's GND
* PCA9685 V+ - high amp 5V
* PCA9685 GND - high amp GND
* Finger servos: to PCA channels 0-4, in my notation thumb - channel 0, index finger - channel 1 etc.

**3.قرار دادن سنسورهای EMG:**

برای به دست آوردن خوانش های معقول، مهم است که دستگاه های uECG را که فعالیت ماهیچه ها را ثبت می کنند، در مکان های مناسب قرار دهید. در حالی که بسیاری از گزینه های مختلف در اینجا امکان پذیر است، هر کدام به رویکرد پردازش سیگنال متفاوتی نیاز دارند :



ممکن است غیر قابل درک باشد، اما سیگنال عضله شست در طرف مقابل بازو بهتر قابل مشاهده است، بنابراین یکی از حسگرها در آنجا قرار می گیرد و همه آنها نزدیک به آرنج قرار می گیرند (عضلات بیشتر بدن خود را در آن ناحیه دارند. ، اما شما می خواهید بررسی کنید که دقیقاً در کجا قرار دارد - تفاوت فردی بسیار زیادی وجود دارد)

**4.کد:**

قبل از اجرای برنامه اصلی، باید شناسه‌های واحد دستگاه‌های uECG خاص خود را بیابید (این کار با حذف کامنت خط 101 و روشن کردن دستگاه‌ها یکی یکی انجام می‌شود) و آنها را در آرایه unit\_ids پر کنید (خط 37).

#include <SPI.h>  
#include <RF24.h>  
#include <RF24\_config.h>  
#include <nRF24L01.h>  
#include <Adafruit\_PWMServoDriver.h>  
#include <Wire.h>  
#define SERVOMIN 150 // this is the 'minimum' pulse length count (out of 4096)  
#define SERVOMAX 600 // this is the 'maximum' pulse length count (out of 4096)  
Adafruit\_PWMServoDriver pwm = Adafruit\_PWMServoDriver();  
int rf\_cen = 9; //nRF24 chip enable pin  
int rf\_cs = 8; //nRF24 CS pin  
RF24 rf(rf\_cen, rf\_cs);  
//pipe address - hardcoded on uECG side  
uint8\_t pipe\_rx[8] = {0x0E, 0xE6, 0x0D, 0xA7, 0, 0, 0, 0};  
uint8\_t swapbits(uint8\_t a){ //uECG pipe address uses swapped bits order  
// reverse the bit order in a single byte  
uint8\_t v = 0;  
if(a & 0x80) v |= 0x01;  
if(a & 0x40) v |= 0x02;  
if(a & 0x20) v |= 0x04;  
if(a & 0x10) v |= 0x08;  
if(a & 0x08) v |= 0x10;  
if(a & 0x04) v |= 0x20;  
if(a & 0x02) v |= 0x40;  
if(a & 0x01) v |= 0x80;  
return v;  
}  
long last\_servo\_upd = 0; //time when we last updated servo values - don't want to do this too often  
byte in\_pack[32]; //array for incoming RF packet  
unsigned long unit\_ids[3] = {4294963881, 4294943100, 28358}; //array of known uECG IDs - need to fill with your own unit IDs  
int unit\_vals[3] = {0, 0, 0}; //array of uECG values with these IDs  
float tgt\_angles[5]; //target angles for 5 fingers  
float cur\_angles[5]; //current angles for 5 fingers  
float angle\_open = 30; //angle that corresponds to open finger  
float angle\_closed = 150; //angle that corresponds to closed finger  
void setup() {  
//nRF24 requires relatively slow SPI, probably would work at 2MHz too  
SPI.begin();  
SPI.setBitOrder(MSBFIRST);  
SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI\_MODE0));  
for(int x = 0; x < 8; x++) //nRF24 and uECG have different bit order for pipe address  
pipe\_rx[x] = swapbits(pipe\_rx[x]);  
//configure radio parameters  
rf.begin();  
rf.setDataRate(RF24\_1MBPS);  
rf.setAddressWidth(4);  
rf.setChannel(22);  
rf.setRetries(0, 0);  
rf.setAutoAck(0);  
rf.disableDynamicPayloads();  
rf.setPayloadSize(32);  
rf.openReadingPipe(0, pipe\_rx);  
rf.setCRCLength(RF24\_CRC\_DISABLED);  
rf.disableCRC();  
rf.startListening(); //listen for uECG data  
//Note that uECG should be switched into raw data mode (via long button press)  
//in order to send compatible packets, by default it sends data in BLE mode  
//which cannot be received by nRF24  
Serial.begin(115200); //serial output - very useful for debugging  
pwm.begin(); //start PWM driver  
pwm.setPWMFreq(60); // Analog servos run at ~60 Hz updates  
for(int i = 0; i < 5; i++) //set initial finger positions  
{  
tgt\_angles[i] = angle\_open;  
cur\_angles[i] = angle\_open;  
}  
}  
void setAngle(int n, float angle){ //sends out angle value for given channel  
pwm.setPWM(n, 0, SERVOMIN + angle \* 0.005556 \* (SERVOMAX - SERVOMIN));  
}  
float angle\_speed = 15; //how fast fingers would move  
float v0 = 0, v1 = 0, v2 = 0; //filtered muscle activity values per 3 channels  
void loop()  
{  
if(rf.available())  
{  
rf.read(in\_pack, 32); //processing packet  
byte u1 = in\_pack[3];//32-bit unit ID, unique for every uECG device  
byte u2 = in\_pack[4];  
byte u3 = in\_pack[5];  
byte u4 = in\_pack[6];  
unsigned long id = (u1<<24) | (u2<<16) | (u3<<8) | u4;  
//Serial.println(id); //uncomment this line to make list of your uECG IDs  
if(in\_pack[7] != 32) id = 0; //wrong pack type: in EMG mode this byte must be 32  
int val = in\_pack[10]; //muscle activity value  
if(val != in\_pack[11]) id = 0; //value is duplicated in 2 bytes because RF noise can corrupt packet, and we don't have CRC with nRF24  
//find which ID corresponds to current ID and fill value  
for(int n = 0; n < 3; n++)  
if(id == unit\_ids[n])  
unit\_vals[n] = val;  
}  
long ms = millis();  
if(ms - last\_servo\_upd > 20) //don't update servos too often  
{  
last\_servo\_upd = ms;  
for(int n = 0; n < 5; n++) //go through fingers, if target and current angles don't match - adjust them  
{  
if(cur\_angles[n] < tgt\_angles[n] - angle\_speed/2) cur\_angles[n] += angle\_speed;  
if(cur\_angles[n] > tgt\_angles[n] + angle\_speed/2) cur\_angles[n] -= angle\_speed;  
}  
for(int n = 0; n < 5; n++) //apply angles to fingers  
setAngle(n, cur\_angles[n]);  
//exponential averaging: prevents single peaks from affecting finger state  
v0 = v0\*0.7 + 0.3\*(float)unit\_vals[0];  
v1 = v1\*0.7 + 0.3\*(float)unit\_vals[1];  
v2 = v2\*0.7 + 0.3\*(float)unit\_vals[2];  
//calcualting scores from raw values  
float scor0 = 4.0\*v0\*v0/((v1\*0.3 + 20)\*(v2\*1.3 + 15));  
float scor1 = 4.0\*v1\*v1/((v0\*2.0 + 20)\*(v2\*2.0 + 20));  
float scor2 = 4.0\*v2\*v2/((v0\*1.2 + 20)\*(v1\*0.5 + 15));  
//print scores for debugging  
Serial.print(scor0);  
Serial.print(' ');  
Serial.print(scor1);  
Serial.print(' ');  
Serial.println(scor2);  
//compare each score with threshold and change finger states correspondingly  
if(scor2 < 0.5) //weak signal - open finger  
tgt\_angles[0] = angle\_open;  
if(scor2 > 1.0) //strong signal - close finger  
tgt\_angles[0] = angle\_closed;  
if(scor1 < 0.5)  
{  
tgt\_angles[1] = angle\_open;  
tgt\_angles[2] = angle\_open;  
}  
if(scor1 > 1.0)  
{  
tgt\_angles[1] = angle\_closed;  
tgt\_angles[2] = angle\_closed;  
}  
if(scor0 < 0.5)  
{  
tgt\_angles[3] = angle\_open;  
tgt\_angles[4] = angle\_open;  
}  
if(scor0 > 1.0)  
{  
tgt\_angles[3] = angle\_closed;  
tgt\_angles[4] = angle\_closed;  
}  
}  
}

این کد برای کنترل دست رباتیک با استفاده از سیگنال‌های EMG (فعالیت الکتریکی عضلات) نوشته شده است. به طور خلاصه، این کد از حسگرهای uECG برای دریافت سیگنال‌های EMG استفاده می‌کند و با استفاده از آن‌ها می‌تواند انگشتان دست رباتیک را کنترل کند. در ادامه، توضیحی از هر یک از متدها و متغیرهای این کد آمده است:

1. **:**setup() در این متد، تنظیمات اولیه برای استفاده از حسگرها و سروموتورها انجام می‌شود. این شامل تنظیمات SPI، nRF24، PWM، مقادیر اولیه برای زاویه‌های انگشتان و غیره است.
2. :setAngle(int n, float angle) این متد برای ارسال زاویه به سروموتورها استفاده می‌شود.
3. loop(): در این متد، سیگنال‌های دریافتی از حسگرها پردازش می‌شوند. مقادیر EMG محاسبه می‌شوند و بر اساس آن‌ها زاویه‌های انگشتان تنظیم می‌شود. همچنین امتیازهای مرتبط با فعالیت عضلات محاسبه می‌شوند و بر اساس آن‌ها انگشتان باز یا بسته می‌شوند.
4. setDataRate(RF24\_1MBPS): این متد برای تنظیم نرخ انتقال داده‌ها (data rate) به 1 مگابیت بر ثانیه استفاده می‌شود.
5. setAddressWidth(4): این متد برای تنظیم عرض آدرس (address width) به 4 بایت استفاده می‌شود.
6. setChannel(22): این متد برای تنظیم کانال ارتباطی (communication channel) به کانال 22 استفاده می‌شود.
7. setRetries(0, 0): این متد برای تنظیم تعداد و تاخیر تکرارها در صورت عدم موفقیت در ارسال داده‌ها به تعداد صفر و تاخیر صفر می‌باشد.
8. setAutoAck(0): این متد برای غیرفعال کردن تأیید خودکار (auto-acknowledge) استفاده می‌شود.
9. disableDynamicPayloads(): این متد برای غیرفعال کردن اندازه‌های پویا (dynamic payloads) استفاده می‌شود.
10. setPayloadSize(32): این متد برای تنظیم اندازه پیام‌ها (payload size) به 32 بایت استفاده می‌شود.
11. openReadingPipe(0, pipe\_rx): این متد برای باز کردن یک لوله (pipe) برای دریافت داده‌ها با آدرس pipe\_rx استفاده می‌شود.
12. setCRCLength(RF24\_CRC\_DISABLED): این متد برای غیرفعال کردن کنترل خطا CRC استفاده می‌شود.
13. disableCRC(): این متد نیز برای غیرفعال کردن کنترل خطا CRC استفاده می‌شود.
14. startListening(): این متد برای شروع گوش دادن به داده‌های دریافتی استفاده می‌شود.

**5.نتیجه گیری:**

رفتار آن کاملاً خوب نیست و با این پردازش فقط می تواند انگشتان باز و بسته را تشخیص دهد (و نه حتی هر یک از 5، فقط 3 گروه عضلانی را تشخیص می دهد: شست، اشاره و وسط با هم، انگشت حلقه و انگشت کوچک با هم). اما "AI" که سیگنال را تجزیه و تحلیل می کند در اینجا 3 خط کد می گیرد و از یک مقدار واحد از هر کانال استفاده می کند. با تجزیه و تحلیل تصاویر طیفی 32 بین روی رایانه شخصی یا تلفن هوشمند می توان کارهای بیشتری انجام داد. همچنین این نسخه تنها از 3 دستگاه uECG کانال EMG استفاده می کند. با کانال های بیشتر، تشخیص الگوهای واقعاً پیچیده باید امکان پذیر باشد - اما خوب، این نکته پروژه است، تا نقطه شروعی برای هر کسی که علاقه مند است فراهم شود. کنترل دستی قطعاً تنها کاربرد چنین سیستمی نیست.