پروژه ی درس کنترل خطی

اعضای گروه:
نرگس مبینی کشه 401413182
زهرا قربانعلی 401413092
علیا عمرانی 401412466

عنوان پروژه:
کنترل موقعیت توپ بر روی میله (Ball and Beam) با استفاده از کنترل‌کننده PID و سنسور فاصله اولتراسونیک

هدف پروژه:
طراحی و پیاده‌سازی یک کنترلر بازخوردی با الگوریتم PID برای تنظیم موقعیت توپ روی یک میله با استفاده از سروو موتور و سنسور فاصله (ultrasonic). هدف نهایی، حفظ توپ در یک نقطه خاص روی میله است.

قطعات استفاده‌شده:
آردوینو (UNO)
سنسور اولتراسونیک HC-SR04
سروو موتور (مثلاً SG90 یا MG996R)
میله (Beam) و یک توپ
سیم‌کشی

توضیحات مربوط به اجزای استفاده شده در سیستم:
سنسور التراسونیک:
برای اندازه گیری فاصله از سنسور تا توپ استفاده میشود، برای تعیین فاصله موجی ارسال میکند و زمان بازگشت موج را اندازه می گیرد و زمان را با استفاده از سرعت موج به فاصله برحسب سانتی متر تبدیل میکند تا در کد استفاده شود.

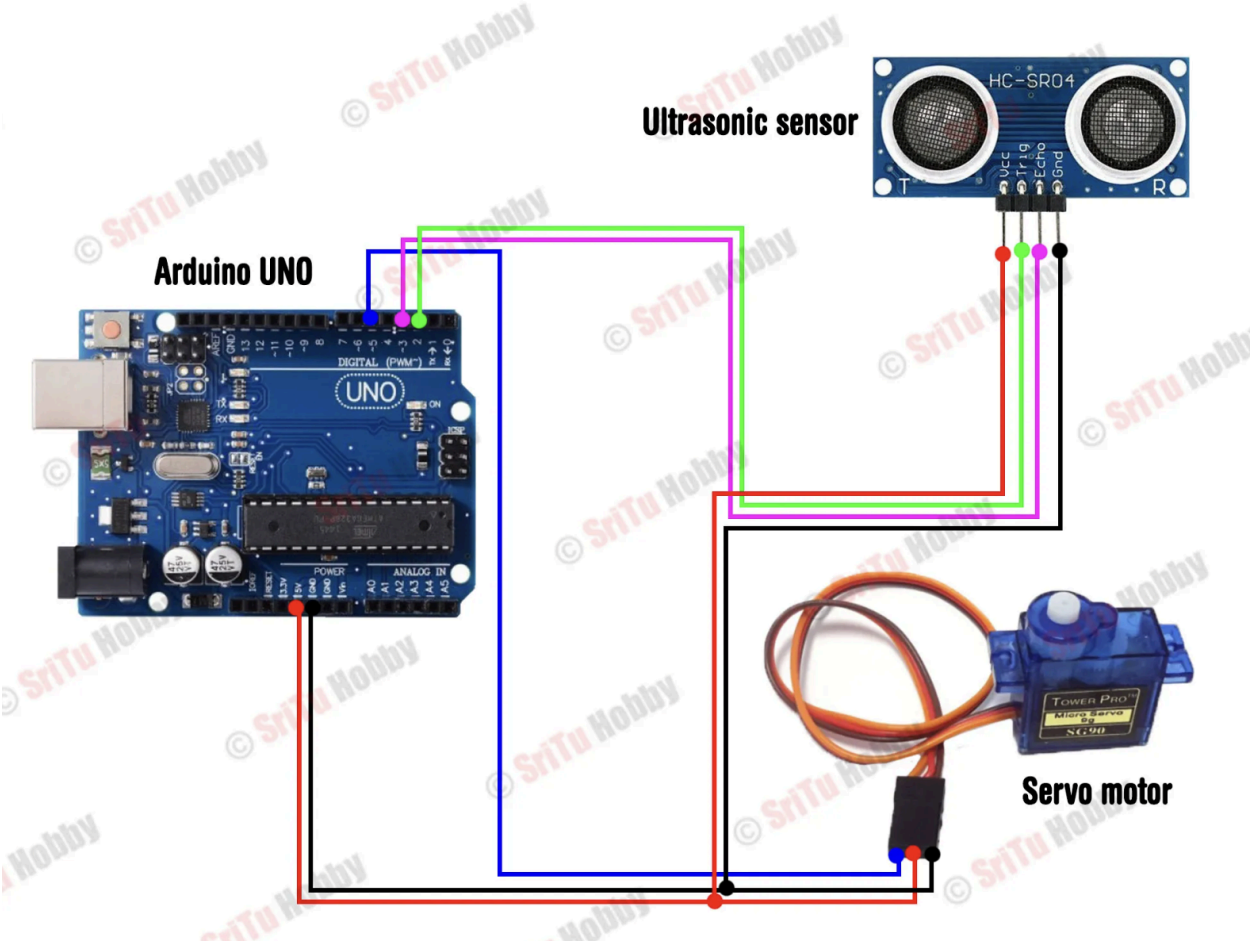
سروو موتور:
بعد از اندازه گیری فاصله توسط التراسونیک و تبدیل آن به سانتی متر با استفاده از کد، فاصله را به یک زاویه برای servo motor مپ میکنیم و به سروو میدهیم تا با توجه به فاصله میله ی تعادل را متناسب با مقادیر pid

حرکت دهد، سروو با توجه به زاویه ی داده شده به آن می چرخد و میله ی متعادل که با استفاده از سیم فلزی له سروو متصل است حرکت میکند.

برد آردوینو:
با استفاده از برد آردوینو کد نوشته را اجرا میکنیم و مدار مربوط به این سیستم را نیز به pin های آردوینو متصل میکنیم.

جامپر:
برای متصل کردن اجزای مدار استفاده میشود.

سیستم مکانیکی:
شامل بدنه و میله ی متعادل میشود.



توضیحات مربوط به PID:

هر سیستم pid ای از 3 بخش اصلی تشکیل شده است:
بخش اول فیدبک است که در این سیستم با استفاده از سنسور التراسونیک sr04 داده میشود، بخش دوم یک عمل کننده است که در این سیستم servo motor است و در نهایت یک کد با یک ست پوینت نیاز داریم که در این جا نقطه ی میانی میله ی تعادل است که در نقطه ی 15 سانتی متر است.

ابتدا حرکت نسبی رو بررسی می کنیم که برای آن این رابطه را داریم: PID_p = Kp*error
برای تست کردن این پارامتر دو پارامتر دیگر را در کد صفر می کنیم و میدانیم که پارامتر p با توجه به فاصله ی توپ از نقطه ی تعادل واکنش نشان میدهد، و در صورت افزایش فاصله ی توپ نسبت به نقطه ی تعادل سعی میکند آن را در خلاف جهت حرکت دهد، سنسور فاصله تا توپ را اندازه گیری میکند، اگر فاصله ی توپ تا نقطه ی تعادل صفر نباشد سروو موتور را با نسبتی حرکت میدهد تا فاصله تا نقطه ی تعادل به صفر نزدیک تر شود با توجه به این توضیحات پارامتر p را تا جایی تغییر میدهیم که سیستم به صورت مناسبی اطراف نقطه ی تعادل نوسان کند، که برای این کار مقدار p باید حدودا 10 باشد.

سپس واکنش نسبت به تغییرات سرعت را بررسی میکنیم که برای آن داریم: PID_pd = Kp*error + Kd*(error - previous_error)/time
این پارامتر باعث میشود سیستم نسبت به تغییرات سرعت واکنش نشان دهد و در صورت قرار دادن عدد بزرگی برای این پارامتر سعی میکند توپ را در هر جایی از میله که قرار دارد ثابت کند که ما این را نمیخواهیم، برای این که توپ را در نقطه ی میانی ثابت کنیم باید با استفاده از پارامتر p کاری کنیم که توپ نسبت به نقطه ی میانی به طور مناسبی نوسان کند و با استفاده از d آن را ثابت کنیم.

برای دقیق تر کردن مکان ایستادن توپ باید از پارامتر انتگرالی استفاده کنیم: PID_i = PID_i + Ki * error
این پارامتر هر لوپ در کد را جمع میکند و به همین دلیل مقدار افزایش می یابد، به همین دلیل به این صورت عمل می کند که در صورتی که توپ در نقطه ای گیر کند میله ی متعادل را به آرامی بالا می برد تا توپ حرکت کند، برای مثال اگر توپ را با استفاده از شی ای متوقف کنیم میله بالا میرود و سعی میکند توپ را حرکت دهد.
در نهایت مقادیر PID به این صورت جمع این سه عبارت مربوط به پارامترها تعیین میشود.

توضیح کد:

```
#include <Servo.h>
Servo servo;
```

کتابخانه Servo برای کنترل سروو موتور استفاده می‌شود و یک شیء از آن ساخته شده است.

```
#define trig 2
#define echo 3
```

پین‌های trig و echo به ترتیب به پایه‌های Trigger و Echo سنسور HC-SR04 متصل هستند.

```
#define kp 20
#define ki 0.05
#define kd 15
```

با استفاده از دستور define ضرایب pid را تعریف میکنیم.

```
double priError = 0;
double toError = 0;
```

خطای قبلی را تعریف میکنیم که برای محاسبه ی مشتق استفاده میشود و مقدار اولیه ی آن را صفر قرار میدهیم.
مجموع خطاها را تعریف میکنیم که برای محاسبه ی انتگرال استفاده میشود و مقدار اولیه ی آن را برابر با صفر قرار میدهیم.

```
const int setPoint = 15;
const int centerAngle = 90;
const int minAngle = 30;
const int maxAngle = 150;
const int maxPIDrange = 60;
```

ست پوینت تعریف میکنیم که نقطه ی تعادل است و آن را برابر با 15 سانتی متر قرار میدهیم.
زاویه ی میانی سروو موتور را تعیین میکنیم که برابر با 90 درجه است (زیرا دامنه ی حرکتی سروو موتور از 0 تا 180 درجه است)
میخواهیم فیدبکی را که از سنسور دریافت کردیم را به یک زاویه بین 30 تا 150 درجه map کنیم و دلیل انتخاب 30 و 150 درجه به جای استفاده از کل ظرفیت سروو موتور این است که از فشار مکانیکی و داغ شدن و کاهش عمر مفید سروو موتور جلوگیری میشود، اگر سروو به حداقل یا حداکثر زاویه برسد سیستم کنترل از کار می افتد و واکنش سیستم به تغییرات کند میشود.
پارامتر maxPIDrange را برابر با 90 - 30 = 60 قرار میدهیم. که یعنی مثبت یا منفی 60 درجه به صورت ماکسیمم میتواند تغییر کند.

```
void setup() {
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  servo.attach(5);
  Serial.begin(9600);
  servo.write(centerAngle);
}
```

پین echo را به عنوان ورودی تعریف میکنیم و پارامتر trig را به عنوان خروجی تعیین میکنیم.
سروو موتور را به پین 5 متصل میکنیم. ارتباط سریال با نرخ 9600 بیت بر ثانیه شروع میشود. سروو با زاویه ی مرکزی که تعریف کرده بودیم تنظیم میکنیم.

```
void loop() {
  PID();
  delay(50);
}
```

در هر حلقه ای که تابع اجرا میشود تابع PID فراخوانی میشود.
تاخیر 50 میلی ثانیه ای را برای عملکرد سریع تر سیستم تعیین میکنیم.

```
double distance() {
  digitalWrite(trig, LOW);
  delayMicroseconds(4);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
```

این تابع از طریق پالس‌های التراسونیک، فاصله‌ی توپ از سنسور را محاسبه می‌کند. برای اینکه سنسور امواج التراسونیک تولید کند پالس 10 میکرو ثانیه ای به سنسور ارسال می شود.

```
long t = pulseIn(echo, HIGH, 20000);
if (t == 0) return 999;
```

مدت زمان بازگشت پالس اندازه گیری می شود.
در صورت عدم دریافت پاسخ، عدد غیر معتبری (۹۹۹) بازمی‌گرداند.

```
  double cm = t / 29.0 / 2.0;
  return cm;
}
```

با توجه به سرعت صوت زمان به فاصله برحسب سانتی متر تبدیل میشود.

```
void PID() {
  double dis = distance();
  if (dis > 300) return;
```

فاصله اندازه گیری میشود، در صورتی که فاصله بیش از 300 سانتی متر باشد محاسبات انجام نمیشود.

```
double error = setPoint - dis;
```

خطا به عنوان تفاوت فاصله هدف و فاصله اندازه‌گیری شده محاسبه می‌شود.

```
double Pvalue = error * kp;
double Ivalue = toError * ki;
double Dvalue = (error - priError) * kd;
```

محاسبه ی مولفه های PID

```
double PIDvalue = Pvalue + Ivalue + Dvalue;
```

```
priError = error;
toError += error;
```

خطای فعلی برای محاسبات بعدی ذخیره میشود.

خطا به مجموع خطاها اضافه میشود.

```
PIDvalue = constrain(PIDvalue, -maxPIDrange, maxPIDrange);
```

مقدار pid را به ماکس و مینی که تعیین کرده بودیم محدود می کنیم.

```
int angle = centerAngle + PIDvalue;
angle = constrain(angle, minAngle, maxAngle);
```

زاویه ی سروو را با اضافه کردن مقدار pid به مقدار مرکزی 90 درجه تعیین میکنیم.

سپس زاویه را با توجه به زاویه ی ماکس و مینی که تعیین کرده بودیم محدود میکنیم.

```
  servo.write(angle);
  Serial.print("Dist: "); Serial.print(dis);
  Serial.print(" | Angle: "); Serial.println(angle);
}
```

زاویه ی محاسبه شده را به سروو می دهیم تا با توجه به این زاویه حرکت کند.

زاویه و فاصله را با استفاده از سریال مانیتور نمایش میدهیم.

نتیجه نهایی:

این پروژه فاصله ی توپ روی میله تا سنسور را اندازه‌گیری می‌کند و با الگوریتم PID، زاویه‌ی میله را طوری تنظیم می‌کند که توپ به نقطه‌ی هدف (۱۵ سانتی‌متر) برسد و در آنجا باقی بماند.