

Line following robot

- **lms** = Left Motor Speed PWM পিন — পিন নম্বর ৫
- **lmf** = Left Motor Forward Direction পিন — পিন নম্বর ৬
- **lmb** = Left Motor Backward Direction পিন — পিন নম্বর ৭
- **rmf** = Right Motor Forward Direction পিন — পিন নম্বর ৮
- **rmb** = Right Motor Backward Direction পিন — পিন নম্বর ৯
- **rms** = Right Motor Speed PWM পিন — পিন নম্বর ১০

lsp = Left Speed Parameter

rsp = Right Speed Parameter

tsp = Turn Speed Parameter

line_prop = Line Proportional Control Factor

Threshold:

threshold = 512 মানে হলো —

সেম্বর থেকে পাওয়া অ্যানালগ রিডিংয়ের মান যদি 512 এর উপরে হয়, তাহলে সেটা লাইন পাওয়া হিসেবে গণ্য করা হবে।

একটু বিস্তারিত:

- Arduino এর অ্যানালগ ইনপুট 0 থেকে 1023 পর্যন্ত মান দেয়।
- 0 মানে খুব কম আলো/রিফ্লেকশন, 1023 মানে বেশি আলো/রিফ্লেকশন।
- threshold = 512 মানে মাঝামাঝি মান।

ফিদি সেন্সর মান 512 এর উপরে হয় → লাইন আছে (1)

নিচে হয় → লাইন নেই (0)

Binary Sensor Readings for Line Detection

- 0b000010 → অর্থ ২য় সেন্সর (ডান থেকে ৫ম পজিশনে) লাইন পেয়েছে।
- 0b000100 → ৩য় সেন্সর লাইন পেয়েছে।
- 0b000011 → ২য় ও ৩য় সেন্সর দুটোই লাইন পেয়েছে।
- 0b001000 → ৪র্থ সেন্সর লাইন পেয়েছে।

Turn concept

 's' → **Straight / Stop / Stable**

- অর্থ: রোবট কোনো টার্ন নিচ্ছে না, মানে এখন সোজা চলছে বা কোন মোড় নিচ্ছে না।
- কোডে ব্যবহাত:

cpp

```
if (turn != 's') { ... turn = 's'; }
```

abc 'r' → Right Turn

- অর্থ: রোবট ডান দিকে ঘুরবে বা ঘুরছে।
- যখন সেন্সর দেখে যে ডানদিকে ঘুরতে হবে, তখন `turn = 'r';` করা হয়।
- কোডের অংশ:

cpp

```
if ((s[2] + s[3]) && s[0] && !s[5]) turn = 'r';
```

abc 'l' → Left Turn

- অর্থ: রোবট বাম দিকে ঘুরবে বা ঘুরছে।
- যখন বাম দিকের সেন্সর লাইন ধরে, তখন `turn = 'l';` সেট করা হয়।
- কোডের অংশ:

cpp

```
if ((s[2] + s[3]) && s[5] && !s[0]) turn = 'l';
```

PID কন্ট্রোল অ্যালগরিদম (PID Control Algorithm)

 PID এর পূর্ণরূপ:

P → Proportional (আনুপাতিক)

I → Integral (সমাকলিত)

D → Derivative (সঞ্চারক)

 PID কীভাবে কাজ করে?

PID কন্ট্রোলার সেন্সর থেকে ইনপুট পায় এবং তিনটি অংশ হিসাব করে:

1. ◆ Proportional (P):

- বর্তমান ভুল (error) এর উপর ভিত্তি করে কাজ করে।
- বড় error → বেশি করেকশন।
-  সূত্র:
 $P = kp * \text{error}$

2. ◆ Integral (I):

- সময়ের সাথে সাথে ছোট ছোট error এর যোগফল হিসাব করে।
- এটি ধীরে ধীরে সিস্টেমকে সঠিক অবস্থানে নিয়ে আসে।
-  সূত্র:
 $I = I + ki * \text{error}$

3. ▲ Derivative (D):

- error কত দ্রুত বাড়ছে বা কমছে, তা দেখে।
- হঠাতে পরিবর্তন হলে দ্রুত প্রতিক্রিয়া দেয়।
-  সূত্র:
 $D = kd * (\text{error} - \text{last_error})$

পূর্ণ PID ফর্মুলা:

```
PID_output = (kp * error) + (ki * Σerror) + (kd * Δerror)
```

যেখানে:

- $\text{error} = \text{desired_position} - \text{current_position}$
- $\text{kp}, \text{ki}, \text{kd} \rightarrow$ গেইন কনস্ট্যান্ট, যেগুলো টিউন করতে হয়।

লাইন ফলোয়ার রোবটে PID

সাধারণত,

- লাইন থেকে দূরে সরে গেলে \rightarrow বড় error \rightarrow বেশি কারেকশন।
- মাঝখানে থাকলে \rightarrow $\text{error} \approx 0 \rightarrow$ সোজা চলে।

ব্যবহার:

```
error = 3.5 - avg;  
PID = kp * error + kd * (error - last_error);  
// (I অংশ অনুপস্থিত এখানে)  
lmotor = base_speed + PID;  
rmotor = base_speed - PID;
```

উদাহরণ (বেসিক কন্ট্রোল):

মনের avg error = 3.5 - avg করেকশন (PID)

2.0	1.5	ডান দিকে ঘুরবে
3.5	0	সোজা চলবে
5.0	-1.5	বাম দিকে ঘুরবে

PID এর সুবিধা:

- খুব নির্ভুল কন্ট্রোল।
- বিশ্ময়করভাবে স্মৃথ লাইন ফলো।
- সহজেই টিউন করা যায়।

PID টিউনিং:

Parameter	বাড়ালে কী হয়
kp	দ্রুত প্রতিক্রিয়া, কিন্তু অস্থিরতা আসতে পারে
ki	ছোট error এর জন্য ভাল, তবে overshoot বাড়ে
kd	দ্রুত পরিবর্তনে স্থিতিশীলতা আনে