

# 生成AIと考える ロボットプログラミング

RDS-X23

RDC-ESP32\_Type3

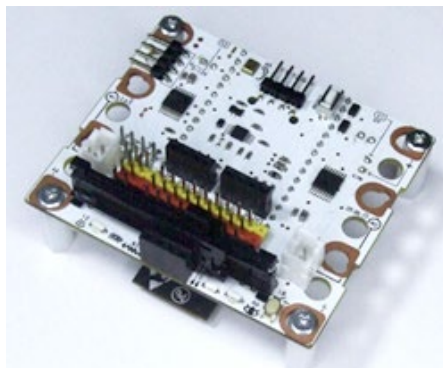
# 今回の目的

生成AIとの会話(チャット)を通じて、  
ロボット動作プログラムを育てよう！

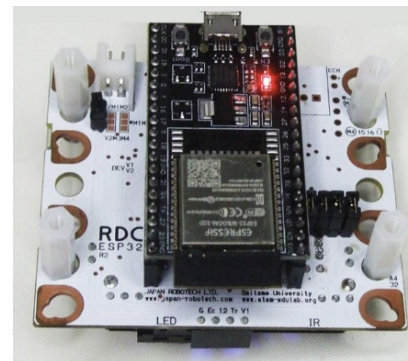
1. 生成AIの現状を知ろう
2. 生成AIにロボットの事を教えてあげよう
3. ロボットに何をさせたいか？整理しよう
4. ロボット動作プログラムを育てよう



RDS-X23



RDC-ESP32\_Type3



# 1. 生成AIの現状

2022年11月 OpenAI社が「ChatGPT」を公開  
世界中で生成AIがはじまりました



無料公開で世界中の  
多様な利用者が増加します



言語・文脈・目的の異なる  
データが大量に集まります



これらのデータが生成AIの  
改良や新機能の開発に活用

整理された  
データは財産！



# 何で無料なの？

## 1. 大規模な利用者データの収集

改善や新機能の開発に必要な整理された**データ回収**

## 2. 市場シェアの獲得とブランド認知の向上

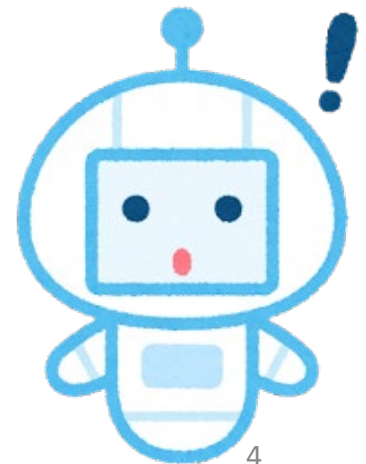
競合他社より早く普及させ「**標準ツール**」の地位確立

## 3. 将来的な収益化への布石

無料プランで利用者獲得 → **有料プランへの移行**

## 4. 社会的な周知と規制への対応

社会的な影響力や信頼性の向上から、  
倫理・安全性・規制に関する**議論をリード**できる！



# どの位？

## 世界の生成AIサービス市場規模(R6情報通信白書)

年	世界の生成AI市場規模	世界の自動車産業規模
2022	約187億ドル	約2兆7,000億ドル
2023	約670億ドル	約2兆7,000億ドル
...		
2032	約1兆3,040億ドル	約3兆ドル

2025年の調査では、  
29%の日本人が  
生成AIを利用した

注意: 予測に物価上昇を考慮

## 国内の生成AIサービス利用者数の推移(ICT総研など)

年度	利用者数(推定)	備 考
2022	約300万人	生成AIを公開
2023	約1,000万人	教育・ビジネス分野での導入を模索
2024	約1,924万人	アプリ改善によりnet利用者の3割が経験
2025	約2,537万人	利用者層の拡大・日常利用の定着

2024年6月, Webアンケート(4,290名)

ChatGPT(18.3%), Copilot (8.9%), Gemini (5.4%)

# 日本語でも利用できる生成AI

OpenAI



2023年時点で、  
数十億ドルの投資



ChatGPT



ChatGPTの技術が、  
copilot に利用



copilot

Google



Gemini

生成AIサービスには、  
莫大な電力と  
多くの専用コンピュータ  
の準備、開発・管理人材  
の確保が必要です。  
資金力が無ければ、  
事業の維持は困難です

複数サービスを同時利用し、優れた回答を出すAIを用いる。  
得意不得意が顕著に表れます

# 生成AIの本質



人間の創造性を模倣して、  
新しいコンテンツを創造する知的ツール

- ①画像、文章、数値などデジタル化された膨大なデータから、統計的な予測に基づきコンテンツを創造する。  
→俳句から小説まで、文字数の変化にも柔軟に対応できる。(スケーラビリティ)
- ②人間の意図をチャットからフィードバックすることで、コンテンツは成長する。初回の回答は「**一般論**」  
チャット(会話)を重ねる → 良き回答へ成長

# スケーラビリティ

生成AIの大きな利点



アレクサンドリア置かれた  
普遍的な図書館  
世界中の「智」を収集

コンピュータは

- ・膨大なデータを「記憶」
- ・必要なデータを瞬時に「呼出し」



人間が学び修得する知識・経験  
より広く・多くの事例にアクセスできる

生成AIは必要に応じて回答に使う学習データサイズを  
変更できること → スケーラビリティ

私たちが学んで修得する知識・経験より、  
広い範囲のデータから回答を出せる利点！



# 相手は機械 → 何でも聞こう

チャット中のAIは、

- ・疲れない
- ・ケンカしない
- ・同じ事を聞いても怒らない
- ・無知を笑わない
- ・話掛けられるのを待っている



恥ずかしがらず、  
気を使わず、  
何でも、何回でも、  
**チャットを繰り返そう！**



乳幼児の疑問

- ・ミルクの量はどの位？
- ・こんな症状の原因は？




いつでも**前向き**  
「**一般論**」を回答



# ハルシネーション hallucination

生成AIが、  
「間違ったことを自信満々に回答する」こと

生成AIへ、  
>>「理科」をテーマにした高校生二人の対談を、  
100字程度で作ってください

 と指示した結果

リョウ:「水って100度で凍るって知ってた？高校の化学で習った気がする。だから冷凍庫の温度って100度以下に設定されてるんだよね。」  
ナ ナ:「えっ、そうだったっけ？なんか記憶が曖昧だな...」

間違っている所を指摘してください！

# ハルシネーション hallucination

生成AIが、  
「間違っことを自信満々に回答する」こと

リョウ:「水って**100度で凍る**って知ってた？ **高校の化学で習った気がする**。だから冷凍庫の温度って**100度以下に設定**されてるんだよね。」

ナ ナ:「えっ、そうだったっけ？なんか記憶が曖昧だな...」

ハルシネーションのポイント:  
水は0度で凍り、100度で沸騰する。  
冷凍庫の温度設定も100度ではない。

AIが温度に関する**基本的な科学知識**を誤って生成した例

# なぜ起こるの？

- ・生成AIは「言葉のつながり」を学習しているだけ！  
本当かどうかわからない！
- ・デジタル化し保存されている情報には、  
間違い・倫理的に問題があるモノが混ざっている
- ・質問があいまいだったり、情報が少ないと  
それっぽい回答を作ってしまう

ハルシネーションは生成AIの「思い込み」  
使う人間の確認が大切！



- ・複数の情報源から確認すること！
- ・質問は具体的に！

## 2. 生成AIに ロボットのことを教えてあげよう

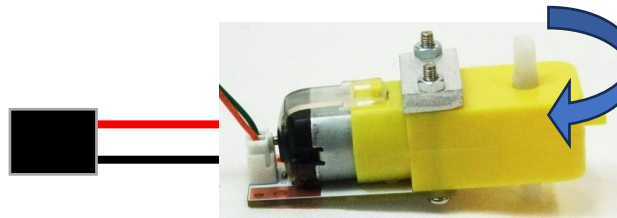
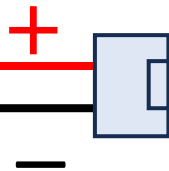
これから皆さんが動かすロボットには、  
どんな特徴がありますか？

- ・プログラミング言語の種類
- ・マイコンボードの仕様
- ・モータの種類・個数と接続方法
- ・センサーなどの種類と接続方法
- ・ロボットの速度
- ・タイヤの円周など

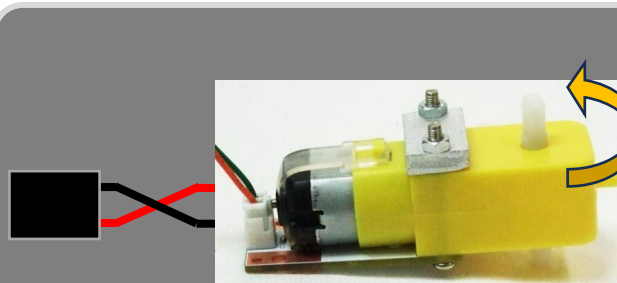


# モータを回すために

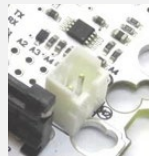
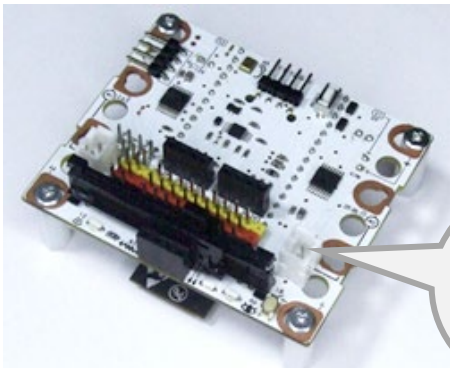
電源にモータケーブルを接続する。電源の極性(+)と、  
モータの極性を合わせる: **時計回り(CW)**  
モータの極性を入れ替える: **反時計回り(CCW)** に回転する



時計回り  
Clock Walk : 英語  
↓  
時計回り: **CW**  
と表記



反時計回り  
Counter Clock Walk  
↓  
反時計回り: **CCW**  
と表記



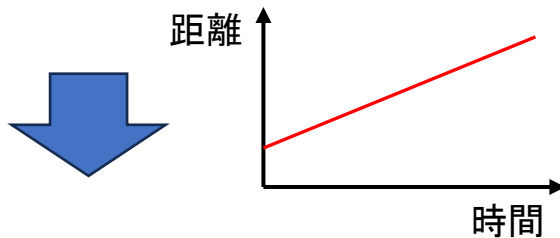
モータが意図しない方向に回転したら、  
モータケーブルの方向を入れ替えて接続！

# アナログとデジタル

生成AIに、アナログとデジタルの違いを訪ねてみよう！

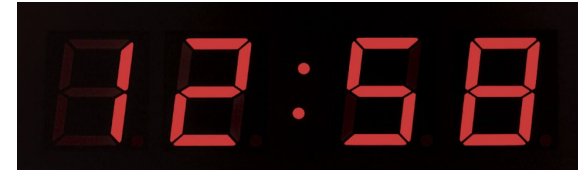


時を伝えるために長針と短針が  
円盤に配置された数字の上を、  
**連続して**回転する




「量」を**連続して**変化する表現

## アナログ



コンピュータは、  
デジタルが得意！

時を伝えるために配置された  
数字が、**1分毎に**変化する  
12:58 の次は 12:59



10, 20, 30, ...  
**10毎に**変化

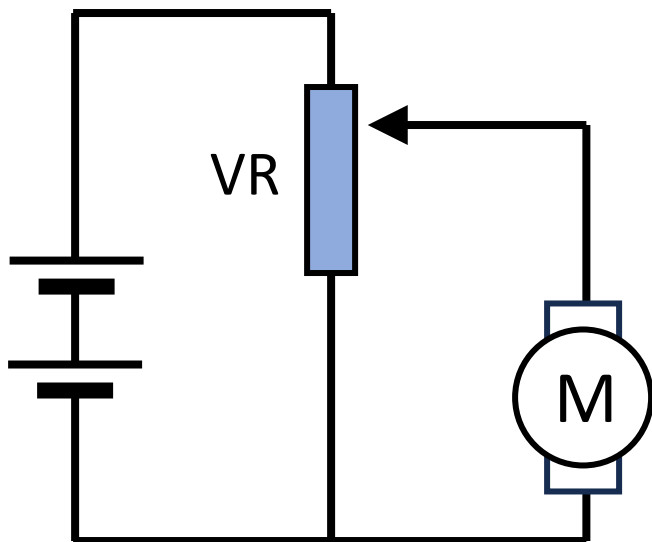
「量」を**非連続に**、**離散的**(かけ離れた)な  
数字や記号でする表現

## デジタル

# モータの出力(回転数)を変える方法

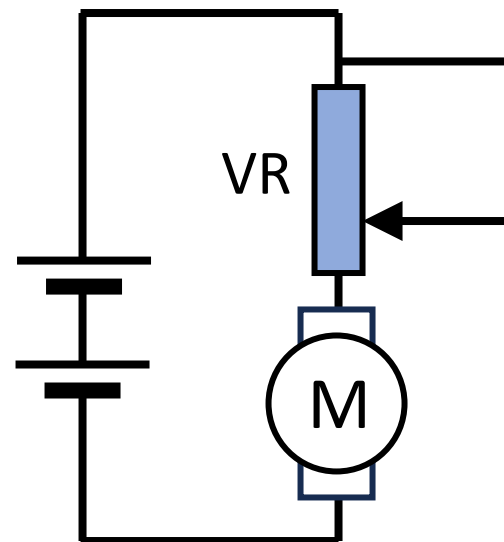
アナログ

## 電圧調整



可変抵抗器  
Variable  
Resistance

## 電流調整



オームの法則に従って、可変抵抗器 VR の抵抗値を変えることで、流れる電圧(V)・電流(A)を連続的に変化させる → 狙ったモータの回転数に調整する！



# Pulse Width Modulation (PWM)

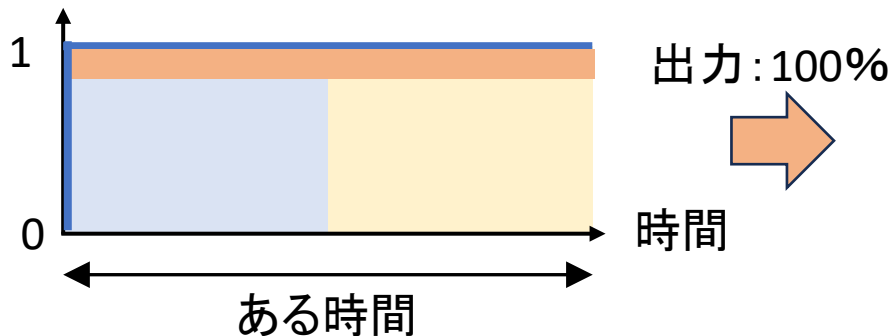
パルス 幅 変調

コンピュータはとても高速に、  
信号のON/OFFを切り替えるのが得意

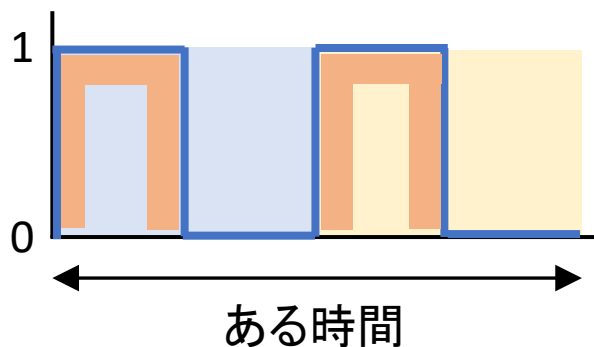


ON/OFF, 1か0, HIGH or LOW  
これは離散的なデジタル信号  
と言えます

信号出力



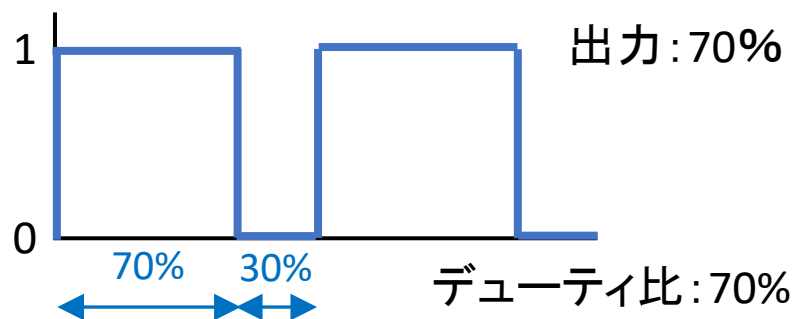
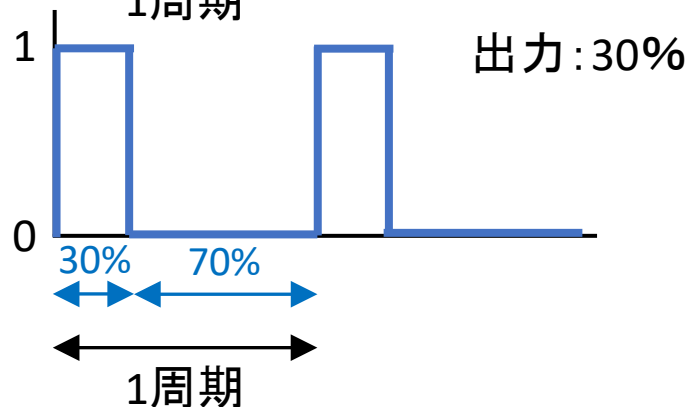
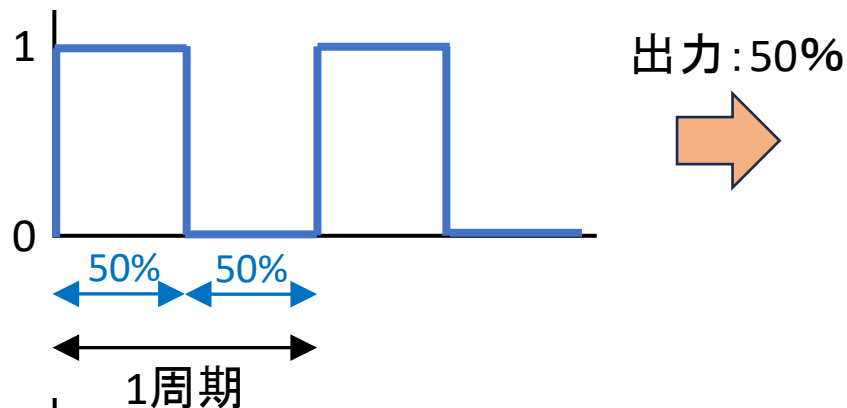
横軸: 時間, 縦軸: 信号出力  
ある時間の間, 信号は「1」つまり  
ONの状態でした.  
このときの信号出力を100%と考えます



ある時間の半分だけ, 信号を「1」に  
しました. このときの信号出力の合計  
は, 半分の50%になります

# Pulse Width Modulation (PWM)

パルス 幅 変調



パルス信号の幅 (HIGHの時間) を  
変えることで, ON/OFF制御でも  
任意の信号出力を実現する

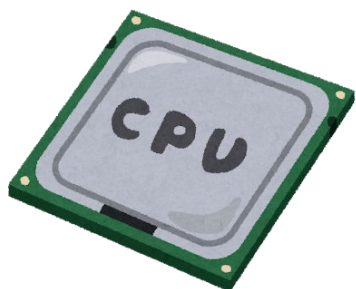
1周期に対する  
ONの周期の割合 (%) を  
デューティ比と言う

# モータの制御\_概要

CPU → モータドライバIC  
との接続準備



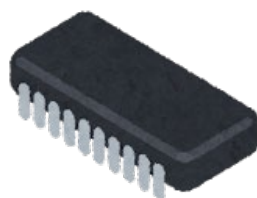
RDC-ESP32TYPE3の内部



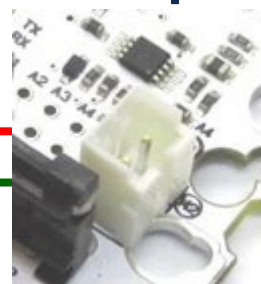
pin = 25番

pin = 26番

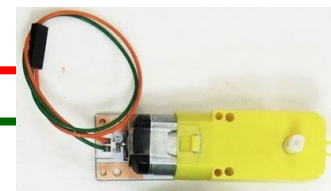
pin = 27番



M2用の  
モータドライバIC

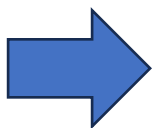


M2



CPUと各モータドライバICを  
何番のpinで接続するか？

プログラム上で指示  
する必要がある



デジタル (digital)  
pin = 25  
pin = 26  
アナログ (analog)  
pin = 27

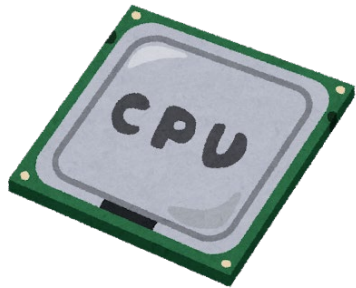
M2モータドライバICは、

- ・デジタル信号  
→ 正転・逆転・停止
- ・アナログ信号  
→ 回す力  
(電力)を制御

# モータの制御\_回転方向



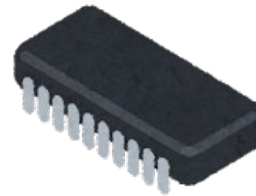
RDC-ESP32TYPE3の内部



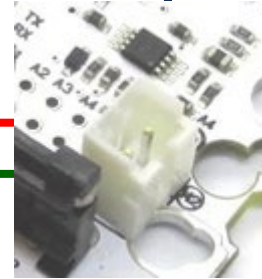
25番→HIGH or LOW

26番→HIGH or LOW

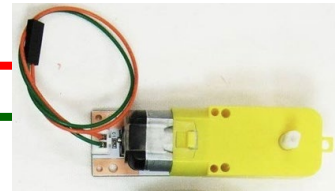
27番



M2用の  
モータドライバIC



M2



## プログラムから

M2用モータドライバICに繋がる  
デジタル2本のpinへ、

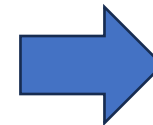
HIGH(高い)かLOW(低い)を指定

M2モータの回転方向

正転(CW)・逆転(CCW)・停止(STOP)の指示



STOP



組み  
合わせ

正転(CW)  
25番←HIGH  
26番←LOW

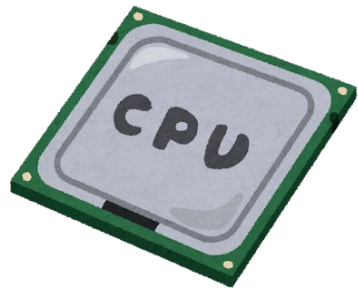
逆転(CCW)  
25番←LOW  
26番←HIGH

停止(STOP)  
25番←LOW  
26番←LOW

# モータの制御\_出力調整



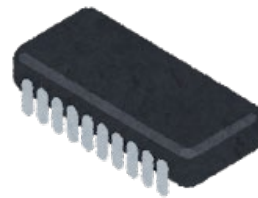
RDC-ESP32TYPE3の内部



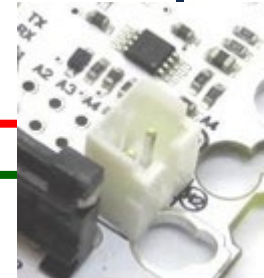
25番

26番

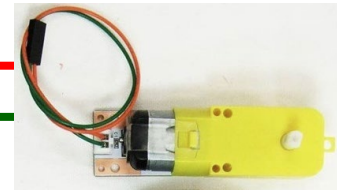
27番 → PWM



M2用の  
モータドライバIC



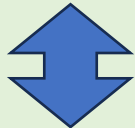
M2



プログラムから

PWMの数値を指定し、モータ出力を調整

PWM: 0 ~ 255



出力: 0 % ~ 100%

256段階 =  $2^8$

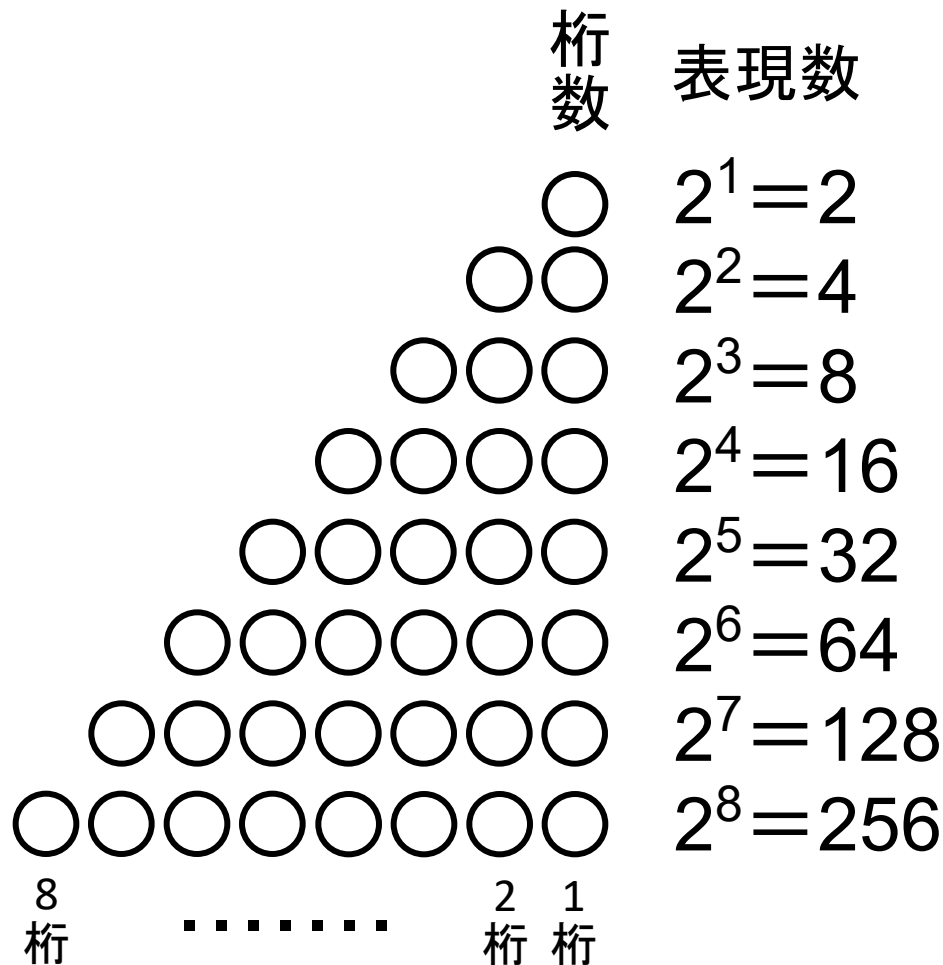
デジタルは、

ONかOFF, 1か0, HIGHかLOWなど  
2つの状態で表現する

→ 2進数と相性がよい

# デジタルと2進数の表現数

ONかOFF, 1か0, HIGHかLOWなど2つの状態で表現する → 2進数と相性がよい



$$2^n \text{ 指数: 桁数} = \text{bit数}$$

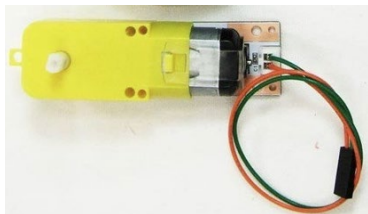
$$2^1 \rightarrow \begin{matrix} 0 \\ 1 \end{matrix} \left. \vphantom{\begin{matrix} 0 \\ 1 \end{matrix}} \right\} \begin{matrix} \text{表現数} \\ \text{(表現できる数)} \\ 0 \text{ と } 1 \text{ の } 2 \text{ つ} \end{matrix}$$

$$2^8 \rightarrow 256 \text{ 表現数}$$

# ロボットの仕様を整理



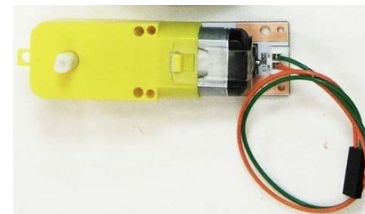
M1(左側)



正面



M2(右側)

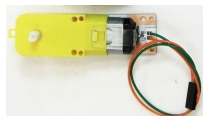


RDC-ESP32TYPE3  
(実際の方向を確認)



# マイコンボードと pin番号

M1

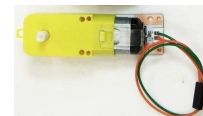


デジタル (digital)  
pin = 2番  
pin = 4番  
アナログ (PWM)  
pin = 5番



RDC-ESP32TYPE3  
(実際の方向を確認)

M2



デジタル (digital)  
pin = 25番  
pin = 26番  
アナログ (PWM)  
pin = 27番

マイコンボードは、  
各pinの信号を変化させ、それに繋がる  
各モータを自由に動かします

1個のモータは、  
デジタル信号 (2本) → 正転・逆転・停止  
アナログ信号 (1本) → 回す力  
計3本の信号でモータを制御します

ROBO DESIGNER®

RDCのマニュアル  
マイコンボードガイド : ピンアサイン



各モータを担当する  
デジタル・PWMのpin番号を整理



# 今回のプログラミング方法

## 動作プログラムを作る

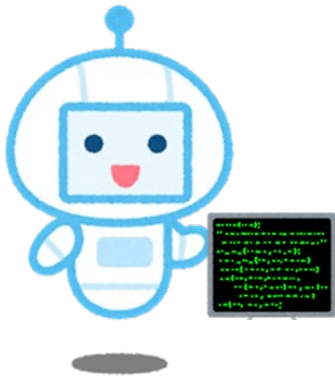
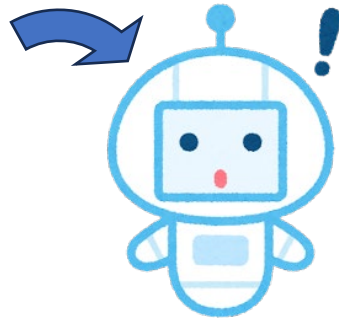
ロボットの仕様や実現したい要望(動作)を生成AIへ伝え、会話を重ねて実現する

M1  
デジタル (digital)  
pin = 8番  
pin = 4番  
アナログ (analog)  
pin = 1番

M2  
デジタル (digital)  
pin = 7番  
pin = 10番  
アナログ (analog)  
pin = 1番

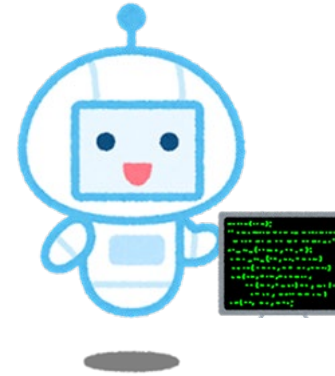
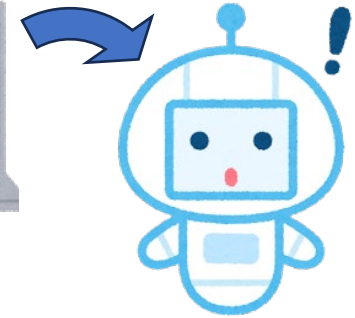
M3  
デジタル (digital)  
pin = 8番  
pin = 13番  
アナログ (analog)  
pin = 11番

M4  
デジタル (digital)  
pin = 11番  
pin = 13番  
アナログ (analog)  
pin = 11番



## プログラムのデバッグ

エラーのあるプログラムを生成AIへ読込ませ、会話を重ねることで問題点を明らかにし、対処法を試す



# お願いを考える

目的: DCモータを回すプログラムを作る

1秒ごとに正転・逆転を繰り返し, 3秒後に停止

生成AIへ伝える「ロボット」の仕様

- ・使う統合開発環境: Arduino IDE
- ・CPU: ESP32
- ・DCモータを1つ利用
- ・PWM制御でモータ出力を調整したい

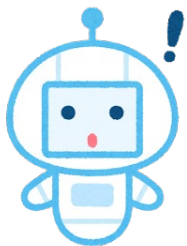


プログラマーへ  
どんな風にお願ひしたら  
伝わるだろう？



# 「お願い」を考える

Arduino IDEを使い開発します.  
CPUはESP32です. 使うDCモータは1個です.  
PWM制御で出力を変更したいです.  
DCモータの回転は, 正転・逆転を1秒ごとに  
切り替へ, 3秒後に停止する様にしてください.



文脈を読み取れる生成AIは,  
プログラマーへの「お願い」を理解できます!  
同じ「お願い」をプロンプトとして入力してみましょう

# Copilot とは

この説明は、Copilotで生成

「Microsoft Copilot(マイクロソフト コパイロット)」は、  
Microsoft社が提供している生成AIツールです  
文章・コード(コンピュータプログラム)・画像などを生成します

## 1. ブラウジング

従来の検索エンジンでは、明確なキーワードを入力する必要があった。  
Copilotでは、対話形式で検索することができる

## 2. 文章の生成

検索や質問だけでなく、ゼロから文章を作成することも可能  
アイデア出しやコンテンツの生成に活用できる

## 3. 画像の認識

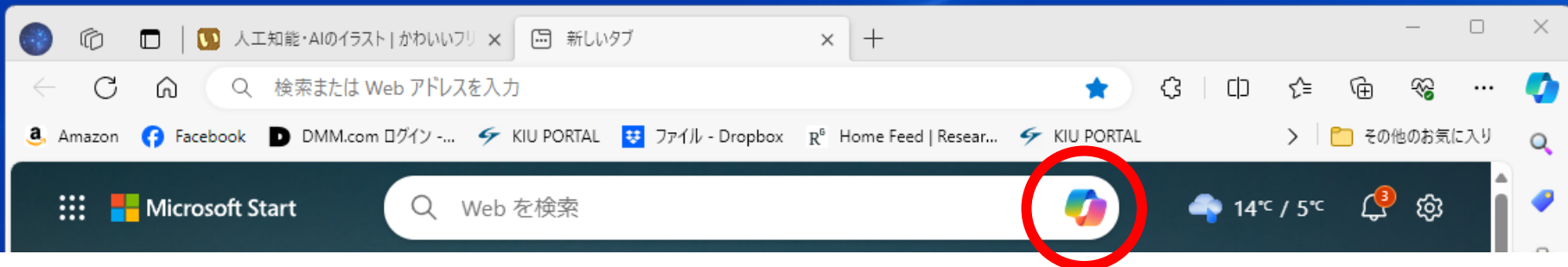
画像をアップロードすればその画像に関する情報を得たり、  
画像に基づいた質問をすることができる

## 4. 画像の生成

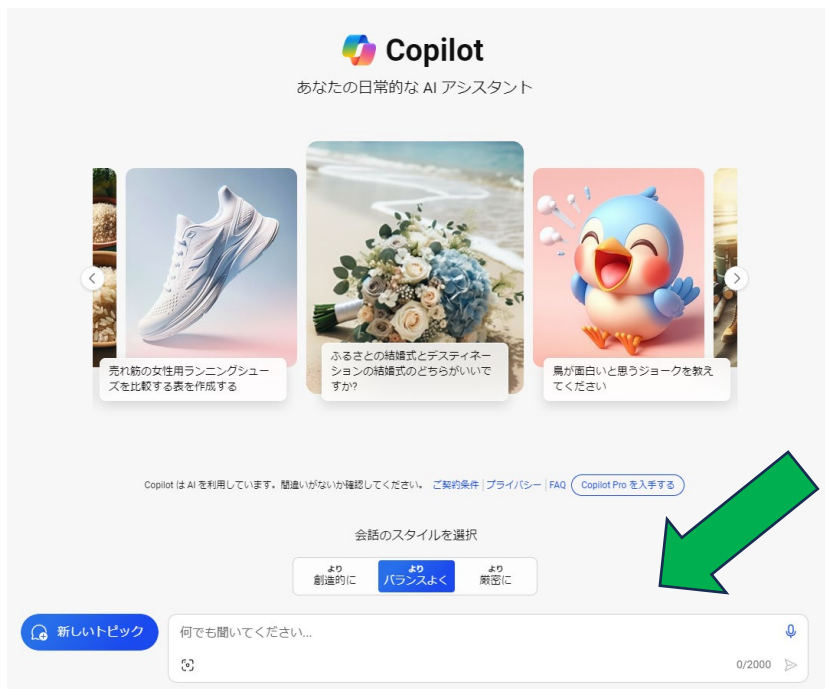
プロンプト(指示文)を入力することで、  
入力したテキストに合った画像を生成できる

有料版ではMicrosoft 365と連携することも可能(無料のブラウザ版で講義を進めます)

# 手軽なマイクロソフト「Copilot」



ココをクリック



語りかけるように、  
お願いを入力

この作業指示を  
「プロンプト」と呼びます

# Copilotへ「お願い」を入力

User登録 + サインインした場合

Arduino IDEを使い開発します。CPUはESP32です。使うDCモータは1個です。 | PWM制御で出力を変更したいです。DCモータの回転は、正転・逆転を1秒ごとに切り替へ、3秒後に停止する様にしてください。



AI で生成されたコンテンツは誤りを含む可能性があります。

入力が終わったら  をクリック！

サインインしなくても利用できます

Arduino IDEを使い開発します。CPUはESP32です。使うDCモータは1個です。 PWM制御で出力を変更したいです。 DCモータの回転は、正転・逆転を1秒ごとに切り替へ、3秒後に停止する様にしてください。

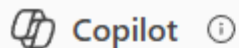


クイック応答 ▾



入力が終わったら  をクリック！

# Copilotの回答



ESP32を使ってDCモータをPWM制御し、1秒ごとに正転・逆転を切り替え、3秒後に停止するプログラムの例を以下に示します。モータドライバとしてL298Nなどを使うことを想定しています。

## 🔧 接続例（L298Nの場合）

- IN1 → ESP32のGPIO 18
- IN2 → ESP32のGPIO 19
- ENA (PWM) → ESP32のGPIO 5

RDC搭載のモータドライバはL298Nではありませんが、命令方法は同じです

```
1 const int IN1 = 18;
2 const int IN2 = 19;
3 const int ENA = 5;
4
5 void setup() {
6   pinMode(IN1, OUTPUT);
7   pinMode(IN2, OUTPUT);
8   pinMode(ENA, OUTPUT);
9 }
10
11 void loop() {
12   // 正転
13   digitalWrite(IN1, HIGH);
14   digitalWrite(IN2, LOW);
15   ledcAttachPin(ENA, 0); // チャンネル0にENAを割り当て
```

☰ その他の行を表示する

①回答をよく読み,「コード例」を確認

②ピン設定の内容を確認

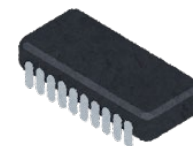
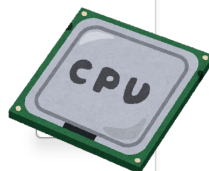
コード例では,

IN1に18番 ←デジタル

IN2に19番 ←デジタル

ENに5番 ←PWM

③動かしたいDCモータのピン設定を確認



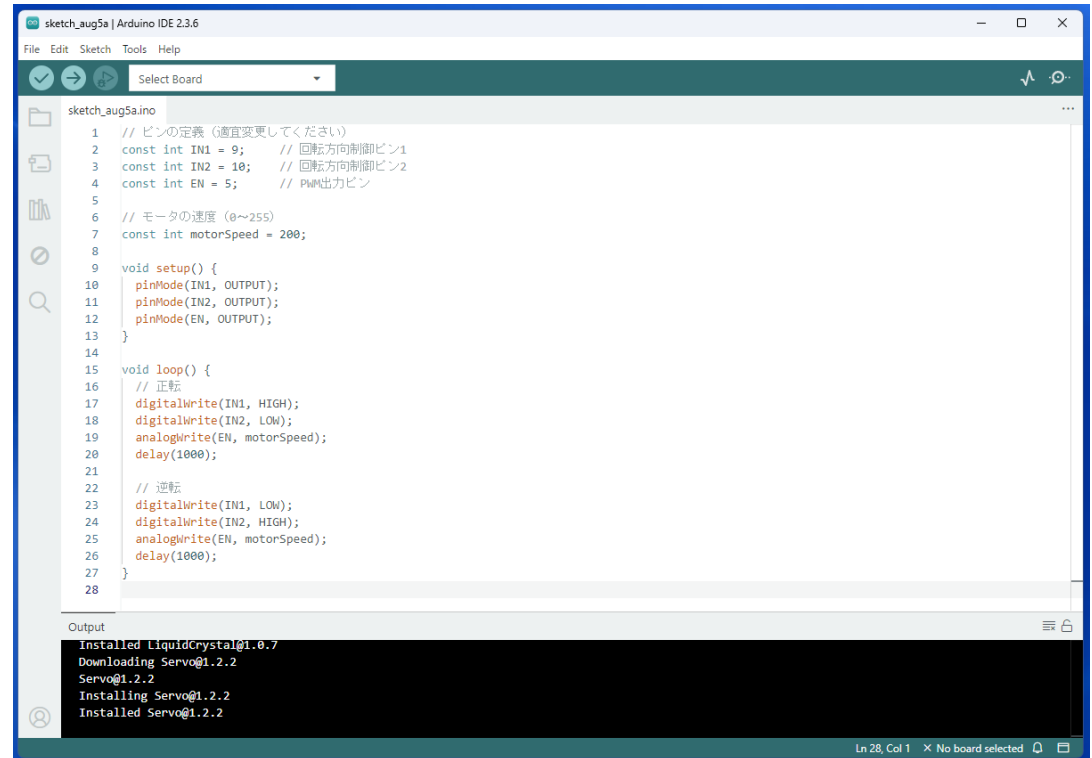
# コードをコピー＆ペースト(貼付け)

## ①Copilotのコードをコピー



## ②Arduino IDE にペースト(貼付け)

```
1 // ピンの定義 (適宜変更してください)
2 const int IN1 = 9; // 回転方向制御ピン1
3 const int IN2 = 10; // 回転方向制御ピン2
4 const int EN = 5; // PWM出力ピン
5
6 // モータの速度 (0~255)
7 const int motorSpeed = 200;
8
9 void setup() {
10   pinMode(IN1, OUTPUT);
11   pinMode(IN2, OUTPUT);
12   pinMode(EN, OUTPUT);
13 }
14
15 void loop() {
16   // 正転
17   digitalWrite(IN1, HIGH);
18   digitalWrite(IN2, LOW);
19   analogWrite(EN, motorSpeed);
20   delay(1000);
21
22   // 逆転
23   digitalWrite(IN1, LOW);
24   digitalWrite(IN2, HIGH);
25   analogWrite(EN, motorSpeed);
26   delay(1000);
27 }
28
```



## ③Arduino IDE 上でピンの設定を, RDC-ESP32\_Type3指定に変更し, コンパイル→Upload すれば動作確認できます



# プログラム(コード)を読む

## プログラムの構造

注意: 生成されるコードは,  
同じ回答にならないことが多い!

```
1 // モータ制御ピンの定義
2 const int motorPin1 = 9; // IN1 (PWM対応ピン)
3 const int motorPin2 = 10; // IN2 (PWM対応ピン)
4
5 // PWM出力値 (0~255)
6 const int pwmValue = 200;
7
8 void setup() {
9     pinMode(motorPin1, OUTPUT);
10    pinMode(motorPin2, OUTPUT);
11 }
12
13 void loop() {
14     // 正転
15     analogWrite(motorPin1, pwmValue);
16     analogWrite(motorPin2, 0);
17     delay(1000); // 1秒待つ
18
19     // 逆転
20     analogWrite(motorPin1, 0);
21     analogWrite(motorPin2, pwmValue);
22     delay(1000); // 1秒待つ
23 }
24
```

初期化処理部:

- ・利用するライブラリの宣言
- ・PIN番号の定義や宣言
- ・変数の宣言

起動処理部「void setup()」:  
CPUが起動してsetup() が  
一度だけ実行される

「宣言」, 「定義」, 「void」など, 判らない  
言葉の意味を生成AIに質問しよう!

動作処理部「void loop()」:  
CPUが起動している間,  
常に実行される処理部

# このコードが回答されていたら

このプログラムは「Hブリッジ制御」向けのコードです

```
1 // モータ制御ピンの定義
2 const int motorPin1 = 9; // IN1 (PWM対応ピン)
3 const int motorPin2 = 10; // IN2 (PWM対応ピン)
4
5 // PWM出力値 (0~255)
6 const int pwmValue = 200;
7
8 void setup() {
9   pinMode(motorPin1, OUTPUT);
10  pinMode(motorPin2, OUTPUT);
11 }
12
13 void loop() {
14   // 正転
15   analogWrite(motorPin1, pwmValue);
16   analogWrite(motorPin2, 0);
17   delay(1000); // 1秒待つ
18
19   // 逆転
20   analogWrite(motorPin1, 0);
21   analogWrite(motorPin2, pwmValue);
22   delay(1000); // 1秒待つ
23 }
24
```

初期化処理部のPIN番号の定義から、  
**2つ**のPWM対応PINで、  
モータ制御していることが判ります



デジタル(digital)  
**pin = 25**  
**pin = 26**  
アナログ(analog)  
**pin = 27**

RDC-ESP32TYPE3 は  
「PWM付きHブリッジ制御」  
です！

生成AIへ

PWM付きHブリッジ制御の  
コードへ変更してください

と指示してください

# プログラム(コード)を読む

回答として、

どちらかのコードが出力されたと思います

例1: if文とbool型変数による正転・逆転の切り替え

```
1 // モータ制御ピンの定義
2 const int IN1 = 2; // モータ方向制御ピン1
3 const int IN2 = 3; // モータ方向制御ピン2
4 const int PWM = 5; // モータPWM出力ピン
5
6 // 回転方向フラグ
7 bool forward = true;
8
9 void setup() {
10     pinMode(IN1, OUTPUT);
11     pinMode(IN2, OUTPUT);
12     pinMode(PWM, OUTPUT);
13 }
14
```

bool型(**bool**)  
の変数は  
true(真) もしくは  
false(偽)  
の二種類の値  
だけを取る

forward が true(真)  
もしくは false(偽)  
(true(真)以外か?)  
を if文で分岐

```
15 void loop() {
16     if (forward) {
17         // 正転: IN1 = HIGH, IN2 = LOW
18         digitalWrite(IN1, HIGH);
19         digitalWrite(IN2, LOW);
20     } else {
21         // 逆転: IN1 = LOW, IN2 = HIGH
22         digitalWrite(IN1, LOW);
23         digitalWrite(IN2, HIGH);
24     }
25
26     // PWMで速度制御 (0~255)
27     analogWrite(PWM, 200);
28
29     // 1秒待機
30     delay(1000);
31
32     // 回転方向を切り替え
33     forward = !forward;
34 }
35
```

生成AIへ

IF文を使わなコードへ  
変更してください

と指示してください

感嘆符「！」で forward 内の  
true(真) と false(偽) を入替

# プログラム(コード)を読む

回答として, どちらかのコードが出力されたと思います

例2: 正転(1000ms) → 逆転(1000ms) → 正転(1000ms) → 停止

```
1  const int IN1 = 18;
2  const int IN2 = 19;
3  const int ENA = 5;
4
5  void setup() {
6      pinMode(IN1, OUTPUT);
7      pinMode(IN2, OUTPUT);
8      pinMode(ENA, OUTPUT);
9  }
10
11 void loop() {
12     // 正転
13     digitalWrite(IN1, HIGH);
14     digitalWrite(IN2, LOW);
15     ledcAttachPin(ENA, 0); // チャンネル0にENAを割り当て
16     ledcSetup(0, 5000, 8); // チャンネル0, 周波数5kHz, 分解能8bit
17     ledcWrite(0, 200); // PWMデューティ比 (0~255)
18     delay(1000);
19
20     // 逆転
21     digitalWrite(IN1, LOW);
22     digitalWrite(IN2, HIGH);
23     ledcWrite(0, 200); // 同じPWMで逆転
24     delay(1000);
25
26     // 正転
27     digitalWrite(IN1, HIGH);
28     digitalWrite(IN2, LOW);
29     ledcWrite(0, 200);
30     delay(1000);
31
32     // 停止
33     digitalWrite(IN1, LOW);
34     digitalWrite(IN2, LOW);
35     ledcWrite(0, 0); // PWM出力を0に
36     while (true); // ループ停止
37 }
38
```

次のページで, PIN番号を変更します.  
判らないことは, なんでも気軽に生成AIへ質問してください!

# PIN定義の変更

右側のM2モータを回して、プログラム(コード)を試みましょう！

```
1  const int IN1 = 18;  
2  const int IN2 = 19;  
3  const int ENA = 5;  
4  
5  void setup() {  
6      pinMode(IN1, OUTPUT);  
7      pinMode(IN2, OUTPUT);  
8      pinMode(ENA, OUTPUT);  
9  }  
10
```

初期化処理部: 生成AIが定義した  
(仮)のPIN番号の定義や宣言

モータ	IN1	IN2	EN(PWM)
(仮)	18	19	5



モータ	IN1	IN2	EN(PWM)
M2(右側)	25	26	27

初期化処理部:  
RDCマニュアルが指定する定義PIN番号へ,  
Arduino IDEのコードを変更する

# ESP32: PWMの設定1

ESP32ではPWMを利用するとき、特有の設定が必要です

```
11 void loop() {
12     // 正転
13     digitalWrite(IN1, HIGH);
14     digitalWrite(IN2, LOW);
15     ledcAttachPin(ENA, 0);           // チャンネル0にENAを割り当て
16     ledcSetup(0, 5000, 8);           // チャンネル0, 周波数5kHz, 分解能8bit
17     ledcWrite(0, 200);               // PWMデューティ比 (0~255)
18     delay(1000);
}
```

Led Control Setup の略

freq

PWMの周波数を5000Hz: 1秒間に5000回に設定

`ledcSetup(uint8_t channel, double freq, uint8_t resolution_bits);`

channel

ESP32は最大16チャンネルのPWMをサポート  
今回は最初の「0番」を利用

`uint8_t` は 符号(±)なし(unsigned) : 負の値は扱えない

resolution\_bits

PWMの分解能(ビット数)

分解能は,

元の数を何分割するか?

今回は 8bit  $\rightarrow 2^8 = 256$

つまり, 0  $\rightarrow$  0%

255  $\rightarrow$  100%

型名	意味	値の範囲
uint8_t	unsigned int 8-bit	0 ~ 255
int8_t	signed int 8-bit	-128 ~ +127

# ESP32: PWMの設定2

ESP32ではPWMを利用するとき、特有の設定が必要です

```
11 void loop() {  
12     // 正転  
13     digitalWrite(IN1, HIGH);  
14     digitalWrite(IN2, LOW);  
15     ledcAttachPin(ENA, 0);  
16     ledcSetup(0, 5000, 8);  
17     ledcWrite(0, 200);  
18     delay(1000);  
}
```

// チャンネル0にENAを割り当て  
// チャンネル0, 周波数5kHz, 分解能8bit  
// PWMデューティ比 (0~255)

`ledcSetup(uint8_t channel, double freq, uint8_t resolution_bits);`

`ledcWrite(uint8_t channel, uint32_t duty);`

Write は書き込み  
: 出力値を書き込みますの意味

resolution\_bits

PWMの分解能(ビット数)

分解能は,

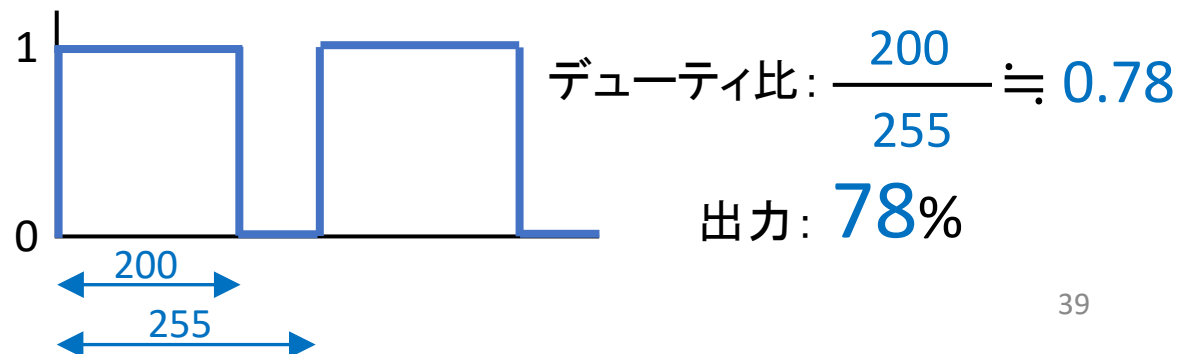
元の数进行何分割するか?

今回は 8bit  $\rightarrow 2^8 = 256$

つまり, 0  $\rightarrow$  0%

255  $\rightarrow$  100%

1周期に対する  
ONの周期の割合(%)を  
duty(デューティ比)と言う



# プログラム(コード)を読む

例3: 正転(1000ms) → 逆転(1000ms) → 正転(1000ms) → 停止

```
1 // ピン定義 (適宜変更してください)
2 const int IN1 = 25;      // モータ方向制御ピン1
3 const int IN2 = 26;      // モータ方向制御ピン2
4 const int ENA = 27;      // PWM出力ピン (モータ速度制御)
5
6 const int PWM_FREQ = 5000; // PWM周波数
7 const int PWM_CHANNEL = 0; // PWMチャンネル
8 const int PWM_RESOLUTION = 8; // PWM分解能 (0~255)
9
10 void setup() {
11     pinMode(IN1, OUTPUT);
12     pinMode(IN2, OUTPUT);
13
14     // PWM設定
15     ledcSetup(PWM_CHANNEL, PWM_FREQ, PWM_RESOLUTION);
16     ledcAttachPin(ENA, PWM_CHANNEL);
17 }
18
```

```
19 void loop() {
20     // 1秒目: 正転
21     digitalWrite(IN1, HIGH);
22     digitalWrite(IN2, LOW);
23     ledcWrite(PWM_CHANNEL, 200); // PWM出力 (0~255)
24     delay(1000);
25
26     // 2秒目: 逆転
27     digitalWrite(IN1, LOW);
28     digitalWrite(IN2, HIGH);
29     ledcWrite(PWM_CHANNEL, 200);
30     delay(1000);
31
32     // 3秒目: 正転
33     digitalWrite(IN1, HIGH);
34     digitalWrite(IN2, LOW);
35     ledcWrite(PWM_CHANNEL, 200);
36     delay(1000);
37
38     // 停止
39     digitalWrite(IN1, LOW);
40     digitalWrite(IN2, LOW);
41     ledcWrite(PWM_CHANNEL, 0);
42     while (true); // ループ停止
43 }
44
```

動作確認を進めましょう！



# チャットで修正1

## 正転・逆転を半分の0.5秒で切り替える様にしたいとき

正転・逆転を半分の0.5秒で切り替える様に修正してください.



AIで生成されたコンテンツは誤りを含む可能性があります。

と、対話する様に生成AIへ投げかけてください



0.5秒ごとに正転・逆転を切り替えるようにするには、`delay(1000)`を`delay(500)`に変更するだけでOKです。以下が修正後のコードです。

1秒 = 1,000m秒 (ミリ秒)    なので    0.5秒 = 500m秒 (ミリ秒)

このコードで、DCモータは0.5秒ごとに正転・逆転を繰り返します。

もし、モータの加減速や停止時間を入れたい場合も対応できます。その他の制御方法やセンサ連携などもご希望があれば教えてください！

# チャットで修正2

## DCモータを全速で回したいとき

DCモータを全速で回す様に修正してください。



AI で生成されたコンテンツは誤りを含む可能性があります。

と、対話する様に生成AIへ投げかけてください

 Copilot ⓘ

DCモータを**全速で回す**には、PWMの出力値を最大（255）に設定します。以下が修正後のコードです。

コードも生成されます

このコードで、DCモータは0.5秒ごとに正転・逆転を切り替えながら、常に最大速度で回転します。

※ もしモータの起動時に負荷が大きい場合や、スムーズな切り替えが必要な場合は、加減速処理（ソフトスタート）も追加できます。必要であればその方法もご紹介します！

必要なら追加で指示を出してください

# ショートカットキー

Ctrl + C

選択した範囲をコピー

Ctrl + V

選択したデータを貼付け

Ctrl + X

選択した範囲を切り取り

Ctrl + Z

直前の操作を元に戻す



直前の操作へ戻し  
出来具合の確認を  
行います  
そんなとき便利です

Alt + Tab

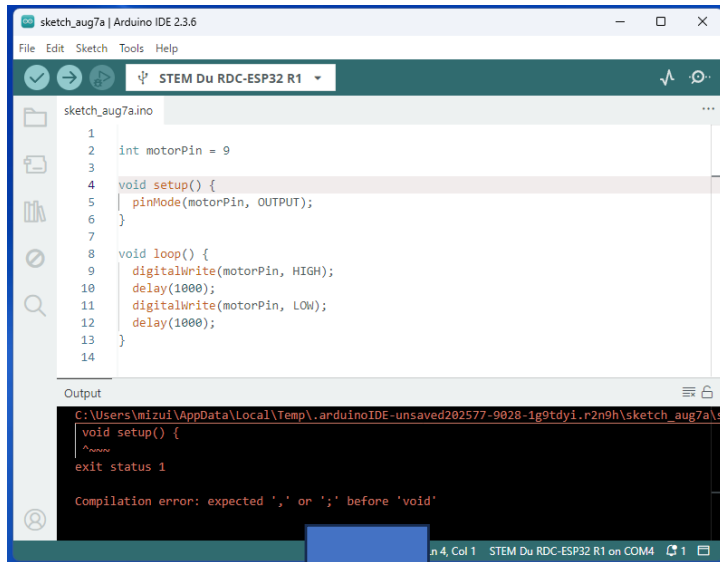
アクティブなウィンドウの切り替え

Ctrl + N

新規の文章作成ウィンドウを開く

Windows上のアプリで  
共通利用できる事も多いので覚えましょう

# エラーとデバッグ



```
1 int motorPin = 9
2
3
4 void setup() {
5   pinMode(motorPin, OUTPUT);
6 }
7
8 void loop() {
9   digitalWrite(motorPin, HIGH);
10  delay(1000);
11  digitalWrite(motorPin, LOW);
12  delay(1000);
13 }
14
```

Output

C:\Users\mizui\AppData\Local\Temp\.arduinoIDE-unsaved202577-9028-1g9tdy1.r2n9h\sketch\_aug7a\sketch\_aug7a.ino

void setup() {  
 ^~~~~~  
exit status 1

Compilation error: expected ',' or ';' before 'void'



Output

C:\Users\mizui\AppData\Local\Temp\.arduinoIDE-unsaved202577-9028-1g9tdy1.r2n9h\sketch\_aug7a\sketch\_aug7a.ino

```
void setup() {  
  ^~~~~~  
exit status 1
```

Compilation error: expected ',' or ';' before 'void'

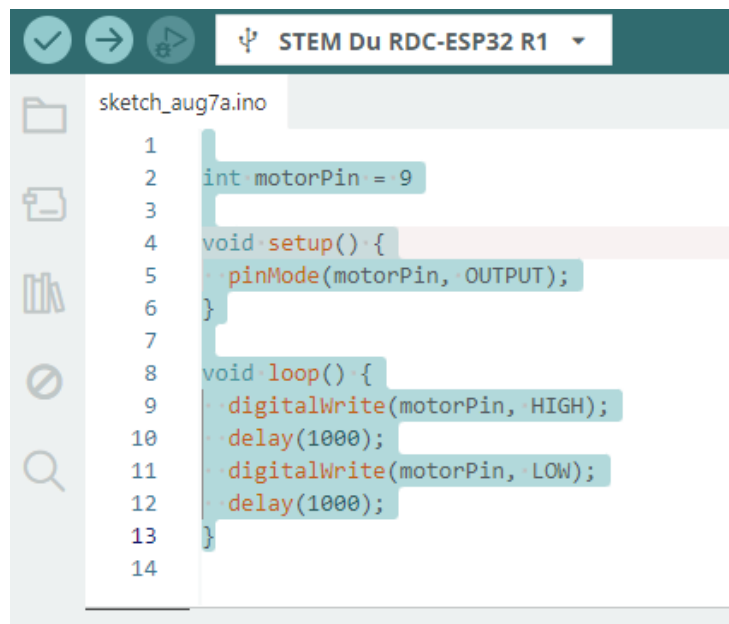
エラーメッセージを理解してデバッグ！

何を「間違った」  
のかな？

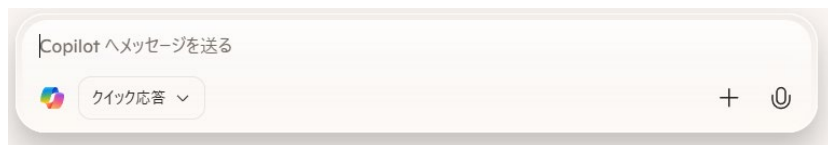


エラーの原因は、  
Arduino IDEの「Output」へ  
「エラーメッセージ」として出力  
されますが、生成AIと一緒に  
原因究明と対応（デバッグ）を  
試してみましょう！

# コードをコピー＆ペースト



①コードを全てコピー



②コードをペーストして実行

✓ 修正済みコード

```
Cpp  コピー

int motorPin = 9; // セミコロンを追加

void setup() {
  pinMode(motorPin, OUTPUT);
}

void loop() {
  digitalWrite(motorPin, HIGH); // モーターをON
  delay(1000); // 1秒待つ
  digitalWrite(motorPin, LOW); // モーターをOFF
  delay(1000); // 1秒待つ
}
```

🔍 修正ポイント

- `int motorPin = 9` の行に セミコロン `;` が必要です。C++ (Arduinoの言語) では、文の終わりにセミコロンを付けるのがルールです。

③回答を確認して

- ・誤りを修正
- ・必要なアドバイスをメモ！



判らない言葉(単語)や  
疑問点についてチャット(会話)  
を続け「学び」を深めよう！

# 相手は機械 → 何でも聞こう

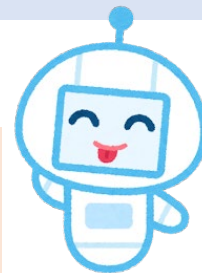
どんなプログラムを作りたいのか？

- ・具体的に指示する
- ・自分の持っている情報を伝える
- ・断片的な言葉でも大丈夫！
- ・細かな指示を何度でも！



恥ずかしがらず、  
気を使わず、  
何でも、何回でも、  
**チャットを繰り返そう！**

回答に「誤り」のある可能性は認識しておこう！  
「怪しい」と感じたら他の方法（書籍や検索）で確認！



# ローバー型ロボットを前進させよう

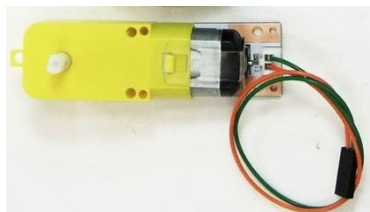
目標：ロボットを1秒間だけ前進させたい

モータの配置を生成AIへ伝える**文章**を考えよう！

正面

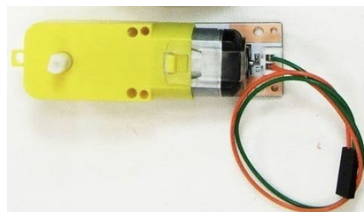


M1: 左側



RDC-ESP32TYPE3

M2: 右側



モータの配置が  
判ってないと、  
各モータの  
回転方向が  
定まらないよね



# ロボットの仕様を伝える

正面



M1: 左側



M2: 右側



RDC-ESP32TYPE3

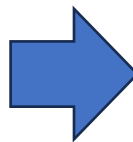


Arduino IDEを使い開発します。  
CPUはESP32です。使うDCモータは1個です。  
PWM制御で出力を変更したいです。  
DCモータの回転は、正転・逆転を1秒ごとに  
切り替へ、3秒後に停止する様にしてください。



例文:

2つのDCモータを動かします。  
左側にM1, 右側にM2,  
を配置します。



2つの文章を  
組合わせてみよう！



# プロンプト例：前進 1 秒間

Arduino IDEを使い開発します。  
CPUはESP32です。使うDCモータは1個です。  
PWM制御で出力を変更したいです。  
DCモータの回転は、正転・逆転を1秒ごとに  
切り替へ、3秒後に停止する様にしてください。



2つのDCモータを  
動かします。  
左側にM1, 右側にM2,  
を配置します。

簡単な国語力の  
応用がポイント！



Arduino IDEを使い開発します。ロボットはローバー型です。  
CPUはESP32です。使うDCモータは2個です。  
PWM制御で出力を変更したいです。各モータに車輪が付きます。

左側にM1, 右側にM2, を配置します。

ロボットが1秒間, 前進するプログラムを提示してください。

# プログラム(コード)を読む: 前進1秒

このプログラムは「Hブリッジ制御」向けのコードです

```
1 // PWMチャンネルとピンの設定
2 const int motor1PWM = 5; // 左モータ M1 のPWMピン
3 const int motor2PWM = 18; // 右モータ M2 のPWMピン
4
5 const int motor1Dir = 4; // 左モータ M1 の方向制御ピン
6 const int motor2Dir = 19; // 右モータ M2 の方向制御ピン
7
8 const int pwmChannel1 = 0; // PWMチャンネル (0~15の範囲で設定可能)
9 const int pwmChannel2 = 1;
10
11 const int pwmFreq = 5000; // PWM周波数 (Hz)
12 const int pwmResolution = 8; // PWM分解能 (8ビット: 0~255)
```

```
13
14 void setup() {
15     // ピンの初期化
16     pinMode(motor1Dir, OUTPUT);
17     pinMode(motor2Dir, OUTPUT);
18
19     // PWMチャンネルの設定
20     ledcSetup(pwmChannel1, pwmFreq, pwmResolution);
21     ledcSetup(pwmChannel2, pwmFreq, pwmResolution);
22
23     // PWMチャンネルとピンの紐付け
24     ledcAttachPin(motor1PWM, pwmChannel1);
25     ledcAttachPin(motor2PWM, pwmChannel2);
26
```

```
27 // 前進方向に設定 (HIGH:前進)
28 digitalWrite(motor1Dir, HIGH);
29 digitalWrite(motor2Dir, HIGH);
30
31 // PWM出力 (速度設定: 0~255)
32 ledcWrite(pwmChannel1, 200); // 左モータ M1
33 ledcWrite(pwmChannel2, 200); // 右モータ M2
34
35 // 1秒間前進
36 delay(1000);
37
38 // 停止
39 ledcWrite(pwmChannel1, 0);
40 ledcWrite(pwmChannel2, 0);
41 }
42
43 void loop() {
44     // 何もしない
45 }
46
```

初期化処理部のPIN番号の定義から、  
左右のモータが**2つの**  
PWM対応PINで、  
モータ制御していることが判ります

生成AIへ

PWM付きHブリッジ制御の  
コードへ変更してください

と指示してください

# プログラム(コード)を読む: 前進1秒

## //PWMチャンネルとピンの表示を統一

```
1 // モータ制御ピンの設定 (ESP32のPWM対応ピン)
2 const int motor1PWM = 5; // 左モータ M1 のPWMピン
3 const int motor2PWM = 18; // 右モータ M2 のPWMピン
4
5 const int motor1IN1 = 4; // 左モータ M1 の方向制御ピン1
6 const int motor1IN2 = 16; // 左モータ M1 の方向制御ピン2
7
8 const int motor2IN1 = 19; // 右モータ M2 の方向制御ピン1
9 const int motor2IN2 = 17; // 右モータ M2 の方向制御ピン2
10
```

初期化処理部のPIN番号の定義が,  
「PWM対応PIN番号」と  
「方向制御PIN番号」を  
分けて表記している

## 生成AIへ

ピン設定を各モータ毎に  
変更してください

と指示してください

Arduino IDE上で見やすく編集してもOK

# プログラム(コード)を読む: 前進1秒

参考: P.33

初期化処理部:

```
1 // 左モータ M1 の設定
2 const int M1_IN1 = 4;
3 const int M1_IN2 = 16;
4 const int M1_PWM = 5;
5
6 // 右モータ M2 の設定
7 const int M2_IN1 = 19;
8 const int M2_IN2 = 17;
9 const int M2_PWM = 18;
10
11 // PWMチャンネル設定
12 const int M1_PWM_CH = 0;
13 const int M2_PWM_CH = 1;
14 const int PWM_FREQ = 5000;
15 const int PWM_RES = 8; // 8ビット (0~255)
16
```

起動処理部「void setup()」:

```
17 void setup() {
18   // 左モータ方向制御ピンの初期化
19   pinMode(M1_IN1, OUTPUT);
20   pinMode(M1_IN2, OUTPUT);
21
22   // 右モータ方向制御ピンの初期化
23   pinMode(M2_IN1, OUTPUT);
24   pinMode(M2_IN2, OUTPUT);
25
26   // PWMチャンネルの初期化
27   ledcSetup(M1_PWM_CH, PWM_FREQ, PWM_RES);
28   ledcSetup(M2_PWM_CH, PWM_FREQ, PWM_RES);
29
30   // PWMピンとチャンネルの紐付け
31   ledcAttachPin(M1_PWM, M1_PWM_CH);
32   ledcAttachPin(M2_PWM, M2_PWM_CH);
33 }
34
```

動作処理部「void loop()」:

```
35 void loop() {
36   // 前進方向に設定 (IN1: HIGH, IN2: LOW)
37   digitalWrite(M1_IN1, HIGH); // 左モータ M1
38   digitalWrite(M1_IN2, LOW);
39   ledcWrite(M1_PWM_CH, 200); // PWM出力 (速度設定)
40
41   digitalWrite(M2_IN1, HIGH); // 右モータ M2
42   digitalWrite(M2_IN2, LOW);
43   ledcWrite(M2_PWM_CH, 200); // PWM出力 (速度設定)
44   delay(1000); // 1秒間前進
45
46   // 停止 (PWMを0に)
47   ledcWrite(M1_PWM_CH, 0);
48   ledcWrite(M2_PWM_CH, 0);
49   while (true); // 無限ループで停止状態を維持
50 }
```

Arduinoでは、  
setup() → loop() の順に自動的に  
呼び出されるため、  
loop() に while(true); を入れることで、  
1回だけ処理して停止する  
という構成が可能になります

※これはコード例です。  
皆さんのコードが同じである必要はありません！

# モータとPIN割り当て

```
1 // 左モータ M1 の設定
2 const int M1_IN1 = 4;
3 const int M1_IN2 = 16;
4 const int M1_PWM = 5;
5
6 // 右モータ M2 の設定
7 const int M2_IN1 = 19;
8 const int M2_IN2 = 17;
9 const int M2_PWM = 18;
10
11 // PWMチャンネル設定
12 const int M1_PWM_CH = 0;
13 const int M2_PWM_CH = 1;
14 const int PWM_FREQ = 5000;
15 const int PWM_RES = 8; // 8ビット (0~255)
16
```

初期化処理部: 生成AIが定義した  
(仮)のPIN番号の定義や宣言

モータ	IN1	IN2	EN (PWM)
M1(左側)	4	16	5
M2(右側)	19	17	18

「EN」はEnable(イネーブル): 有効化を意味

初期化処理部: RDCマニュアルが  
指定する定義PIN番号へ,  
Arduino IDEのコードを変更する

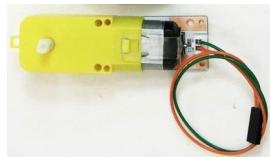


RDC-ESP32TYPE3

モータ	IN1	IN2	EN (PWM)
M1(左前)	2	4	5
M2(右後)	25	26	27

# 前進→ハード調整と動作確認

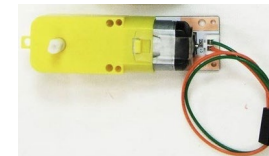
プログラム上での前進は、各モータに正転の指示  
→ 正転でロボットが前進するようコネクタの接続を調整！



M1: 左側



正面



M2: 右側

```
36 // 前進方向に設定 (IN1: HIGH, IN2: LOW)
37 digitalWrite(M1_IN1, HIGH); // 左モータ M1
38 digitalWrite(M1_IN2, LOW);
39 ledcWrite(M1_PWM_CH, 200); // PWM出力 (速度設定)
```

```
41 digitalWrite(M2_IN1, HIGH); // 右モータ M2
42 digitalWrite(M2_IN2, LOW);
43 ledcWrite(M2_PWM_CH, 200); // PWM出力 (速度設定)
44 delay(1000); // 1秒間前進
```

RDC-ESP32TYPE3  
(実際の方向を確認)



P.14を参考に、モータが意図しない方向に回転したら、モータケーブルの方向を入れ替えて接続！

# RDC指定のPIN設定をAIへ伝達

モータ	IN1	IN2	EN(PWM)
M1(左前)	2	4	5
M2(右後)	25	26	27



生成AIへ、以下を指示

M1\_IN1 = 2, M1\_IN2 = 4, M1\_EN = 5,  
M2\_IN1 = 25, M2\_IN2 = 26, M2\_EN = 27  
と定義します. コードを変更してください.

すると →

```
1 // 左モータ M1 の方向制御ピン
2 const int M1_IN1 = 2;
3 const int M1_IN2 = 4;
4 // 左モータ M1 のPWM出力ピン
5 const int M1_EN = 5;
6
7 // 右モータ M2 の方向制御ピン
8 const int M2_IN1 = 25;
9 const int M2_IN2 = 26;
10 // 右モータ M2 のPWM出力ピン
11 const int M2_EN = 27;
12
```

# 前進を基に右旋回を作る！

生成AIへ以下の指示を出す

右旋回を1秒間行うプログラムに変更してください。

+



AI で生成されたコンテンツは誤りを含む可能性があります。

## 生成AIの回答

### 右旋回の動作イメージ

右旋回では、左側の車輪（M1, M3）を前進、右側の車輪（M2, M4）を後退させることで、ロボットがその場で右に回転します。

- ①回答コードをコピーしてArduino IDEへ張り付け
- ②コードに誤りが無いか確認
- ③コードをコンパイルして転送
- ④動作確認



# 動作シナリオを作る

- ・前進1秒でロボットは何cm進みましたか？
- ・車輪の外径は何cmですか？
- ・右旋回1秒でロボットは何度( $^{\circ}$ )回りましたか？



これらの情報を生成AIへ伝達し、  
1m進むコード・ $90^{\circ}$  旋回するコード・  
1m進んで旋回して戻ってくるコードなどを、  
生成AIと一緒に考えてください。

生成AIと共に考えて目的を達成することが、  
この講座の目的です

# 生成AIと仲良く → 国語力を磨こう！

プロンプトから

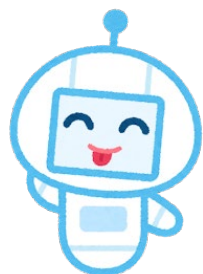
指示の意図を読解

- ・デジタル化された膨大な資料から一般論を提示
- ・断片的な言葉でも大丈夫！
- ・細かな指示を何度でも！



- ・本・新聞を読もう！  
(良い文章に触れよう)
- ・文章を書こう  
(日記, 記事の書き写し)

**毎日の小さな習慣を継続！**



回答に「誤り」や「違和感」を感じたら、  
他の方法(書籍や検索)で確認！  
生成AIへ「〇〇は誤りです」と教えてあげて！

# 今のAIとの理想の関係：共進化

私が書いた文章Aと文章Bを  
要約して、400字にまとめて

文章Cを校正し、  
修正箇所を太字で強調して

プランDとプランEの類似点を  
表示して  
また、同様のプランをwebか  
ら抽出して



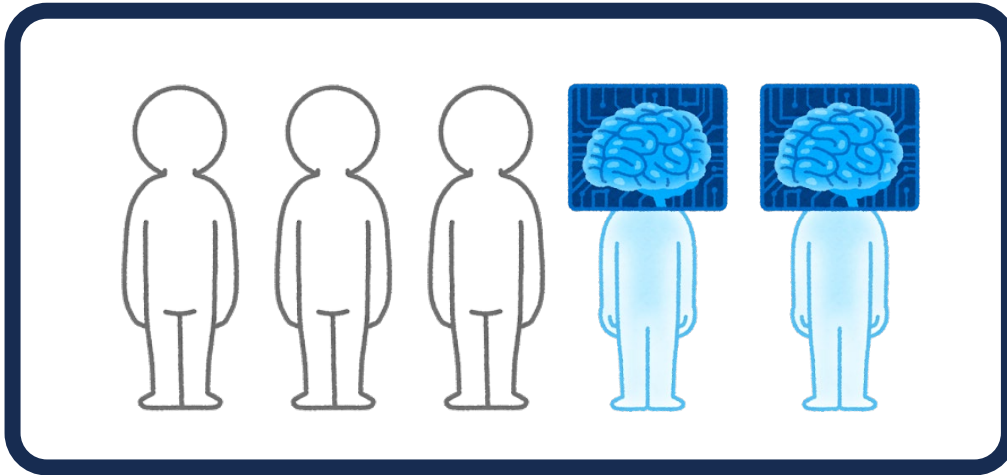
- ・自分の考えたアイデアや文章に対する要約や校正に用いる
- ・生成AIは自分のアイデアや文章を、練り直す道具として用いる

「こんな文章を作って」の丸投げでは、人間の成長が期待できない！

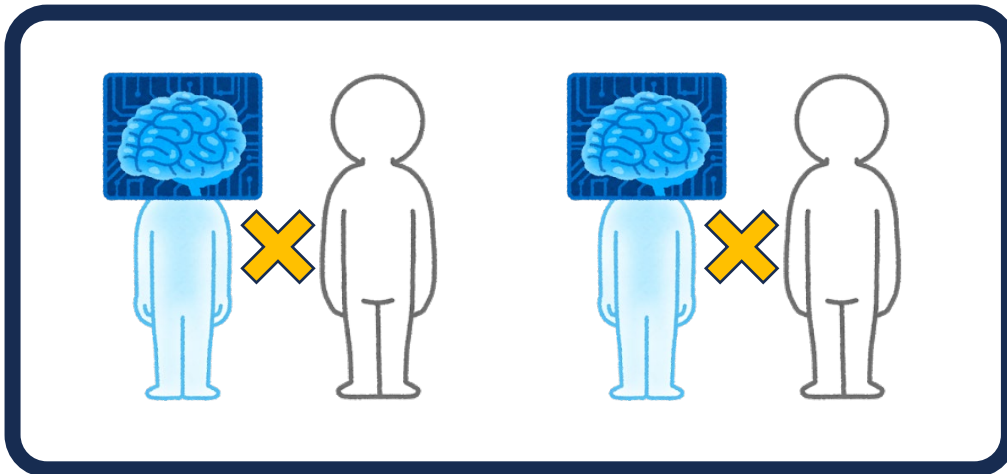


# 今起きている事

ICT化



AI化



AIを使いこなした人間が生産性を高め、  
AIを使いこなせていない人間の仕事を奪う

# AIハラスメント

## AI普及による新たな社会問題

### 情報リテラシーの差別



### プロンプト監視・出力監視



### AI万能主義の押し付け



### 業務の偏り・評価の不公平



# コンテンツとモデル

コンテンツ	モデル名	技術
文章（テキスト）	GPT-4, Claude, Gemini	Transformer
画像	DALL-E, Midjourney, Stable Diffusion	拡散モデル
音声	Whisper, VALL-E	WaveNet系
動画	Sora, Runway	拡散＋時系列処理

**モデル**とは大量のデータを学習し、人間の様に文章・画像・音声・動画など生成する「AIの仕組み」

# ファインチューニング

これが出来る人材の教育が  
社会で求められています！

一般的な知識や言語パターンを学習したモデル  
「GPT-4, Claude, Gemini」に対して、  
法律文書・医療文書など特定分野のデータを準備し、  
再度学習と回答調整を行うことで、  
法律アシスタント・医療AIなど専門特化させること

- ・有償の専門AIとして提供される
- ・ファインチューニングは個人でも行える！
- ・専門知識を持つ人間の確認が必要

# AIアシスタント → AIEージェント

AIアシスタント:

人間の指示に従って情報整理や情報提供を行う人工知能

AIEージェント:

自律的に判断・行動し目標達成を目指す人工知能

	AIアシスタント	AIEージェント
主な役割	利用者支援	自律的な目標達成
判断力	限定的	状況に応じて判断
応用範囲	対話から情報提供	対話、意思決定、行動など

様々な生成AIサービスも、人間の指示と確認が必要