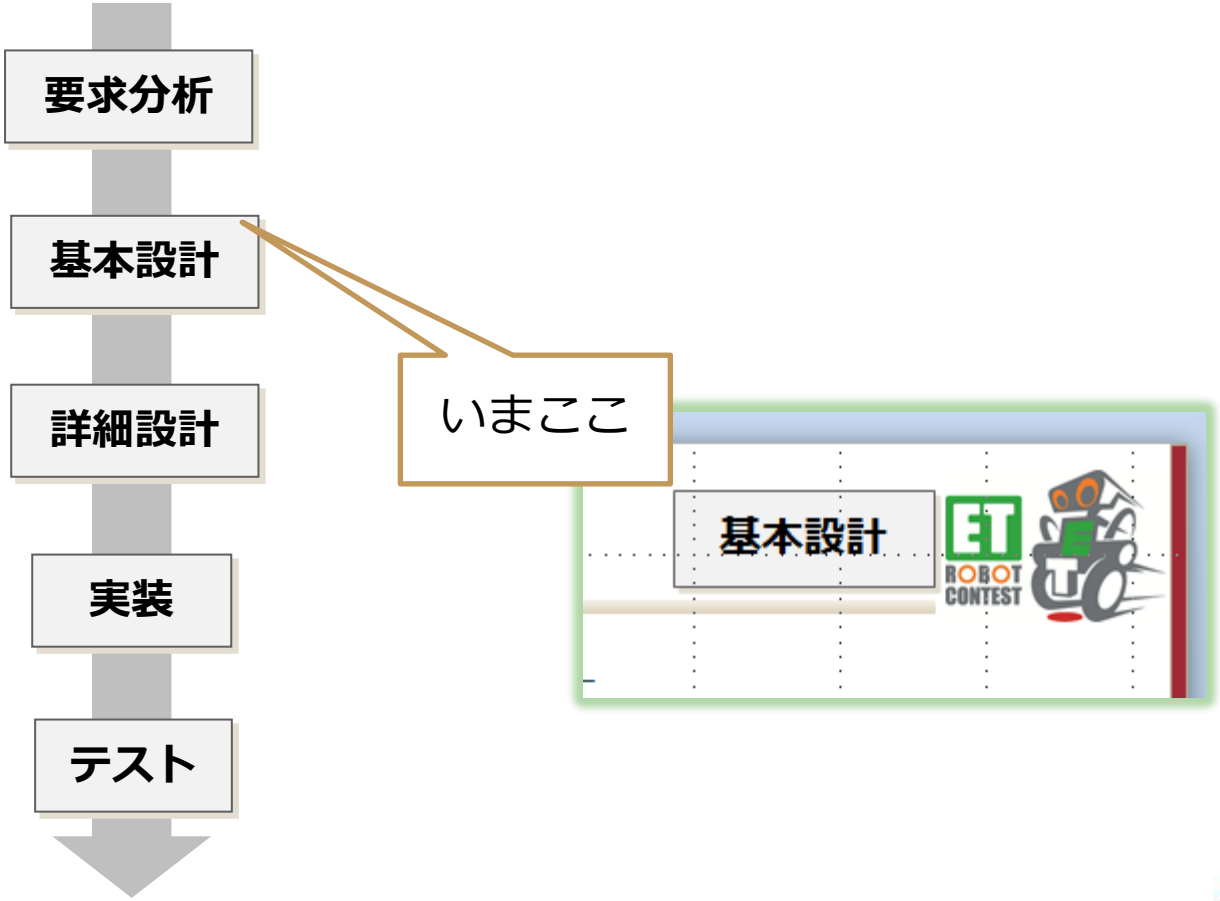
- ETロボコンの走行体を用いて演習しますので、開発に使う環境と、演習に使う環境とライントレースの方法について簡単に説明しておきます
- ライントレースを題材にして、モデルと部品を使ったソフトウェア設計の手順を紹介します

3. モデルと部品を使ってソフトウェアを設計する

3-0.　演習と開発工程の対応について

- 2章の開発工程の説明とスライドに開発工程名をつけました
  - これからやる演習や作成するモデル図と、2章の開発工程で説明した開発工程との対応関係がわかるようになっています



3-1. 演習に使う機器と環境の確認（１）

EV3

走行体

- 2輪倒立振子型ロボット
  - 短くはロボットと呼称します
- 右図のEV3way
  - 尻尾機構は使いません

EV3way

搭載しているデバイス

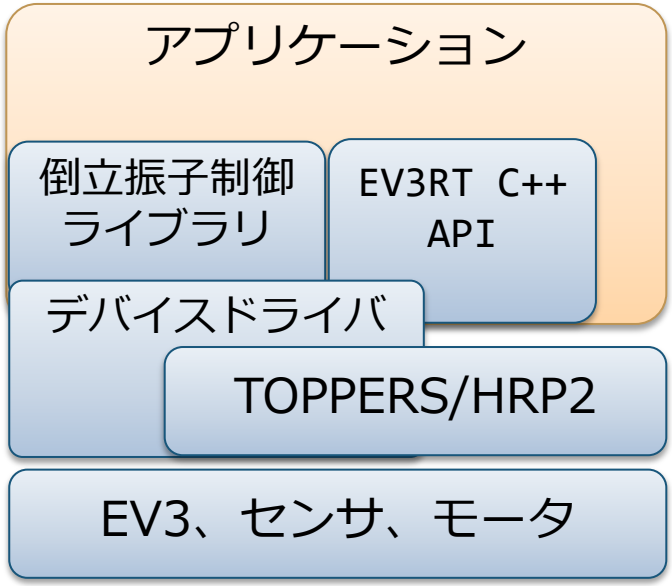
- 走行面の明暗を測る光センサ
- ユーザが押せるタッチセンサ
- 正面を測距する超音波センサ
- 制振に使うジャイロセンサ
- Bluetooth通信機構
- 回転角度センサ付きサーボモータ

※ この教育では、尻尾機構は使いません
尻尾が地面に触れないように先端部をモータ筐体に付けるようにたたんでおいてください

3-1. 演習に使う機器と環境の確認（２）

EV3

- システム分析を一部だけ済ませておきましょう
  - 自分たちが作るものと、入手して利用するもの識別します
  - 入手して利用する機器や環境、技術等を調査、評価します
    - 入手したものを呼び出すレベルまで詳細化が進めば、設計から実装に移れます
- 下図のアプリケーション部分を開発します
  - 提供される部分をこの演習の開発におけるプラットフォームとしましょう



ロボット内部のソフトウェアの構成（EV3用）

技術教育1の「開発環境・要素技術教育」で獲得した知識などが、この演習の開発におけるプラットフォームの構成要素になっていることを確認しましょう

自分たちが作るところ

入手して使うもの

3-1. 演習に使う機器と環境の確認（３）

EV3

- 使用するリアルタイムOS
  - EV3RT（EV3way用）
    - TOPPERS/HRP2が移植されており、μiTRON4.0のAPIが利用できます
- 演習に使うコース
  - 白地に黒い線を引いた周回コースを想定します
    - 難所は取り扱いませんので不要です
- 開発時の留意点
  - 各デバイス（センサ、モータ）と倒立振子制御ライブラリを使うには、OSと共に提供しているライブラリAPIを使用します
  - 走行体に搭載している各デバイスと倒立振子制御ライブラリは、動作開始時に初期化が必要です

3-1. 演習に使う機器と環境の確認（４）

- ０章「トレーニングの準備」に書いてある作業は済んでいますか
  - 準備が済んでいない人は、この演習後に環境を作って自分でやってみましょう
- 開発環境は構築できていますか
  - 使うロボットキットの構築ガイドに合わせて開発環境を構築しておきましょう
- モデリングツールはインストールしてありますか
  - 演習ではモデル図を作成しますので、astah* Professionalと参加者向けライセンスをインストールしましょう
- テキストエディタを用意しましょう
  - 演習で使うテキストエディタを決めて、使い方に慣れておきましょう
- Unixのコマンドに慣れておきましょう
  - Unixの基本コマンド（cd, lsなど）は使えるようになっていますか
  - WindowsやCygwinでのファイルやディレクトリ、ドライブ名の表し方を知っていますか
- 演習用のサンプルコード、サンプルモデルは入手しましたか
  - 実際にソースコードやモデル図を編集しながら演習しますので、入手しておきましょう
- 自分のロボットのジャイロセンサのオフセット値を調べておきましょう

3-2. 倒立振子制御ライブラリついて（１）

- 倒立振子制御ライブラリ（Balancerライブラリ）の役割
  - ロボットの倒立状態を維持する制御則を持ち、モータに与える操作量を計算するライブラリです（操作量をモータに与えた動作結果が制御量です）
    - 操作量の計算までが役割で、このライブラリの内部ではモータを動かしていないことに注意
- 倒立振子制御ライブラリが提供する関数
  - バランス制御関数(balance_control)、初期化関数(balance_init)
  - 使用するときは、balancer.h ファイルやパラメータファイルをインクルードします
- 倒立振子制御ライブラリの制約
  - バランス制御関数は一定の周期で操作量を更新する前提で設計されています
    - 現在のライブラリは4ms周期に調整されています（周期より長過ぎても短か過ぎてもダメ）
- 倒立振子制御C API の解説書
  - http://lejos-osek.sourceforge.net/jp/NXTway_GS_C_API.pdf
- 倒立振子制御ライブラリの場所
  - EV3RT：サンプルプログラムに含まれています

3-2. 倒立振子制御ライブラリついて（2）

- 倒立振子制御 C APIを使うときの留意点
  - ジャイロセンサには通電ドリフトや個体差があるので、補正が必要になります
    - ジャイロセンサは温度変化の影響を受けやすく、通電時間と共に出力値が変動する現象が発生し、ドリフトと呼ばれる
  - モータには個体差や劣化があるので、バランス制御関数に渡すエンコーダ値またはバランス制御関数が返すPWM値には、補正が必要になります
    - 両輪に同じPWM値を与えて前進させても、まっすぐ進まないことからも個体差は確認できる

API 関数	パラメータの説明

バランス制御関数
`void balance_control(`
`F32 cmd_forward,`
`F32 cmd_turn,`
`F32 gyro_value,`
`F32 gyro_offset,`
`F32 theta_m_l,`
`F32 theta_m_r,`
`F32 battery,`
`S8 *ret_pwm_l,`
`S8 *ret_pwm_r)`
	引数:
	cmd_forward: 前進値、100(前進最大値)～-100(後進最大値)
	cmd_turn: 旋回値、100(右旋回最大値)～-100(左旋回最大値)
	gyro_value: ジャイロセンサ値
	gyro_offset: ジャイロセンサオフセット値
	theta_m_l: 左モータエンコーダ値
	theta_m_r: 右モータエンコーダ値
	battery: バッテリ電圧値(mV)
	戻り値:
	ret_pwm_l: 左モータPWM出力値
	ret_pwm_r: 右モータPWM出力値

バランス制御初期化関数 `void balance_init(void)`	この関数の呼び出し時に、左右の車輪駆動モータのエンコーダ値も0にリセットする

EV3では、signed charの代わりに、コンパイラのstdint.hが提供するint8_tを使います

cf. http://lejos-osek.sourceforge.net/jp/nxtway_gs.htm

3-2. 倒立振子制御ライブラリついて（3）

- バランス制御関数をC++で使うときに考えたいこと
  - 4ms周期という制約はバランス制御関数の再計算の周期で、4msごとに値を取得し直すほど変化が急峻ではないパラメータもある
    - 走行のために与える前進値や旋回値、ジャイロセンサのオフセット値、バッテリの残量など
  - 操作量として得たPWM値の授受には、引数以外にクラスの属性を媒介する方法も使える
  - バランス制御関数を使う側がジャイロセンサの通電によるドリフトや個体差の補正処理を意識しないで使えるよう、補正するしくみが持てるようにしたい
  - バランス制御関数を使う側がモータの個体差や劣化に対処する補正処理を意識しないで使えるよう、補正するしくみが持てるようにしたい

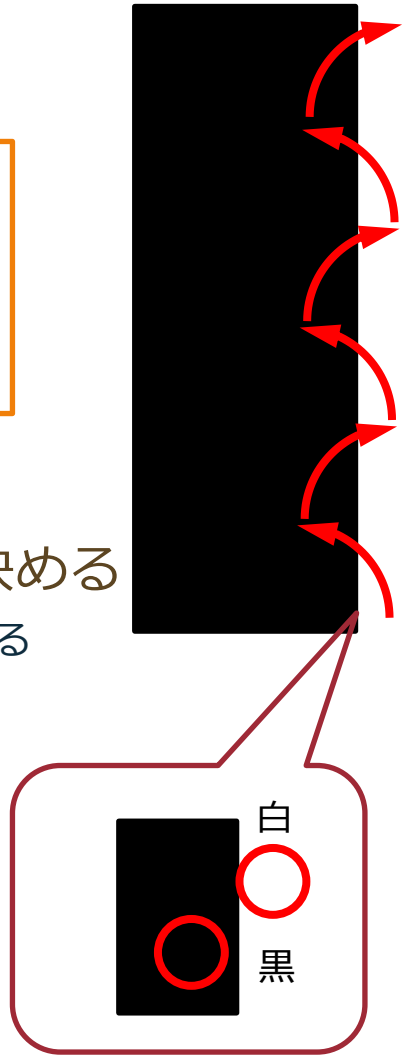
- バランス制御初期化関数をC++で使うとき考えたいこと
  - 呼び出し時に、左右の車輪駆動モータのエンコーダ値も0にリセットする処理を使う側が意識しないで使えるようにしたい

- C言語のライブラリをそのまま使わずにクラス化する余地がありそう
  - どんな方式で、どこまで対処するか、システムの構造を設計するとき考えてみましょう

3-3. 例題1：シンプルなライントレーサ

- 次の手順で経路に沿って走行するロボットです
  1. デバイスやライブラリを初期化する
     - ジャイロセンサのオフセット値を取得する
     - 左右モータの回転角度をリセットする
     - 倒立振子制御ライブラリを初期化する
  2. 走行している場所を調べる
     - 光センサから取得した値を使ってライン上かそうでないかを調べる
  3. （ラインの縁に沿って進むための）走行体の向きを決める
     - ライン上を検出した場合、ラインの縁に近づくために、向きを右にする
     - ライン外である場合、ラインの縁に近づくために、向きを左にする
  4. 決定した向きに従って倒立走行する
     - ジャイロセンサの値を取得する
     - 左右モータの回転角度（回転センサの値）を取得する
     - 本体バッテリーの電圧を取得する
     - 倒立振子制御APIを使って、倒立走行用のモータ出力値を計算する
     - 得られた出力値で左右モータを回転させる
  5. 2～4を繰り返すことで経路に沿って走行する

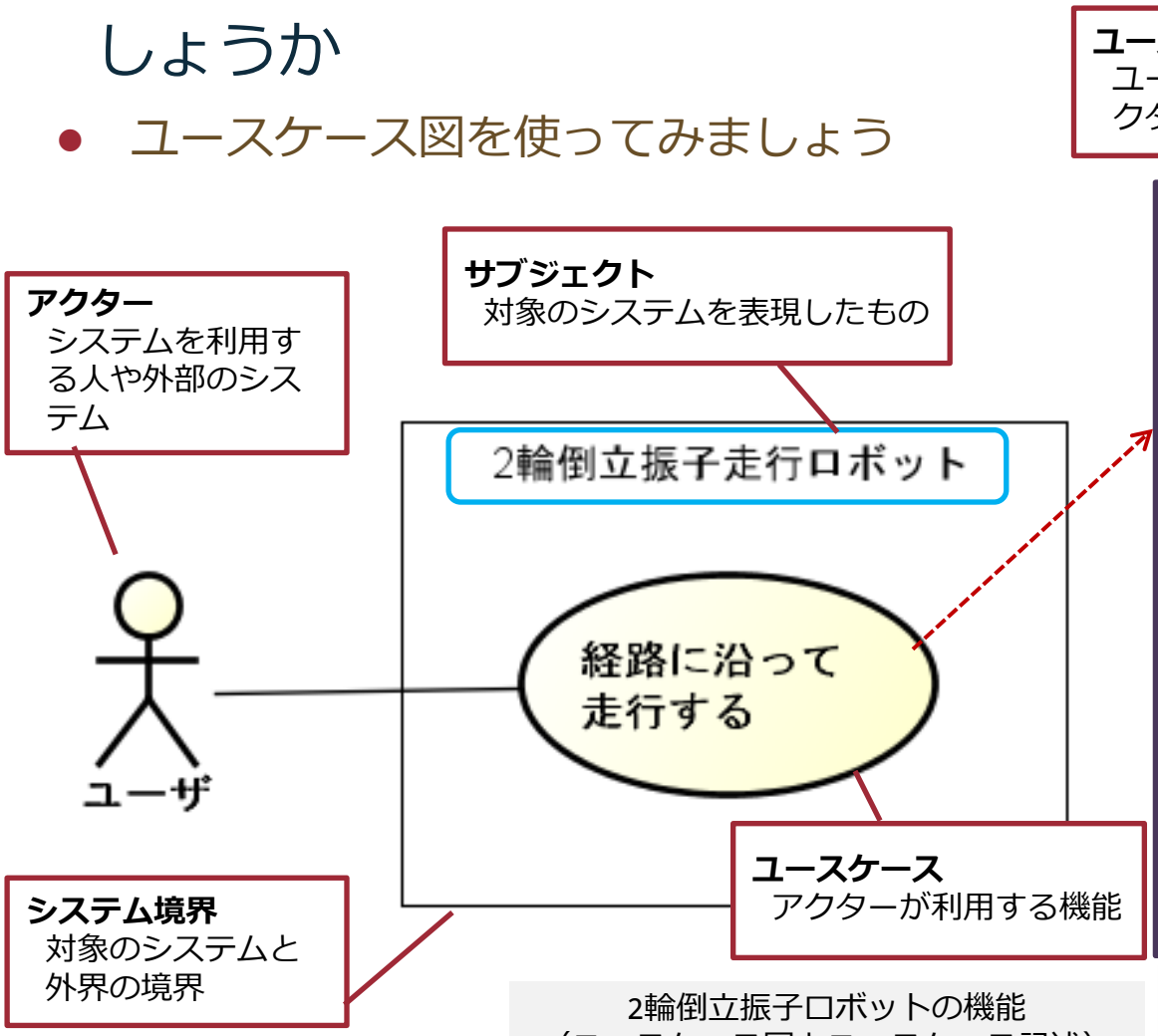
線に沿って走るロボットは、一般にライントレーサ、ラインフォロワー、線の端に沿う場合にはエッジトレーサなどと呼ばれています



3-4. システムが提供する機能を決める

要求分析

- 対象となるシステムが利用者に提供する「機能」はなんでしょうか
  - ユースケース図を使ってみましょう



ユースケース記述
ユースケースごとに、システムとアクターのやりとりを整理したもの

「経路に沿って走行する」の基本ユースケース

1. ユーザがロボットを起動すると、ロボットは動作を開始する
2. ロボットは動作を開始すると、デバイスやライブラリを初期化する
3. ロボットは、経路上にいるか外れているかを調べ、走行する向きを計算する
4. ロボットは決定した向きに従って倒立走行する
5. 3~4を繰り返すことで経路に沿って走行する

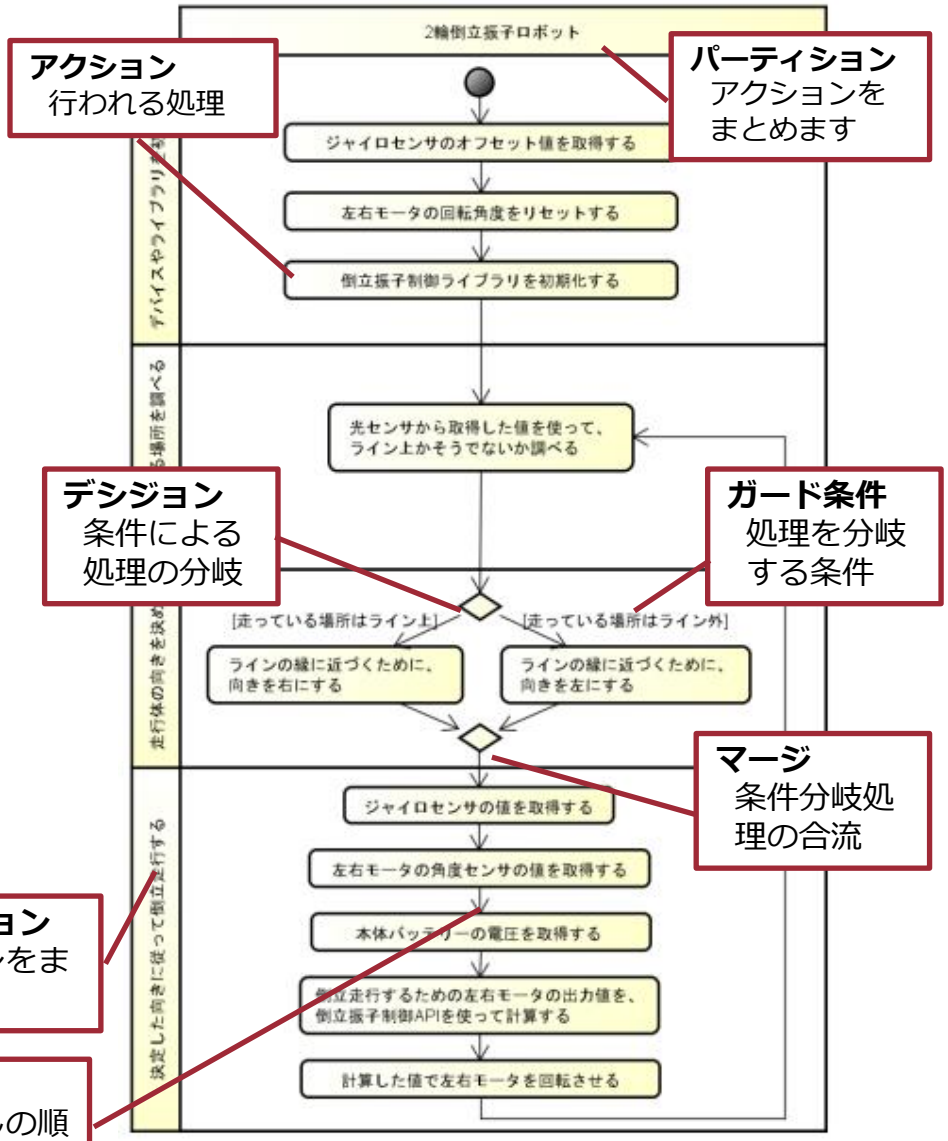
2輪倒立振子ロボットの機能
（ユースケース図とユースケース記述）

要求モデル・機能「2輪倒立振子ロボットの機能」を開いて、ユースケース図を確認しましょう

3-5. システムの振舞いを整理する

- システムがどのような動作をするか整理します
  - アクティビティ図を使って、ユースケース記述で書いた処理の流れを表してみましょう
    - ここではユースケースを実現するのに必要となる処理やその順序を表すのに利用しています
    - ユースケースが複数あるときは、ユースケースごとに別のアクティビティ図を作成するとよいでしょう
    - 「どうやって動かすのか」よりも「どんな風に動くのか」に着目して表してみましょう

要求モデル・振舞い「経路に沿って走行する」を開いて、アクティビティ図を確認しましょう



ユースケース「経路に沿って走行する」の振舞い（アクティビティ図）

3-6. 部品の候補を見つける（1）

- 機能を実現するのに必要そうな「部品」を探します
  - ユースケースやアクティビティ図のアクションから探します
  - 動作関わる「働き」や利用する「情報」に関わる機器やライブラリを候補とします

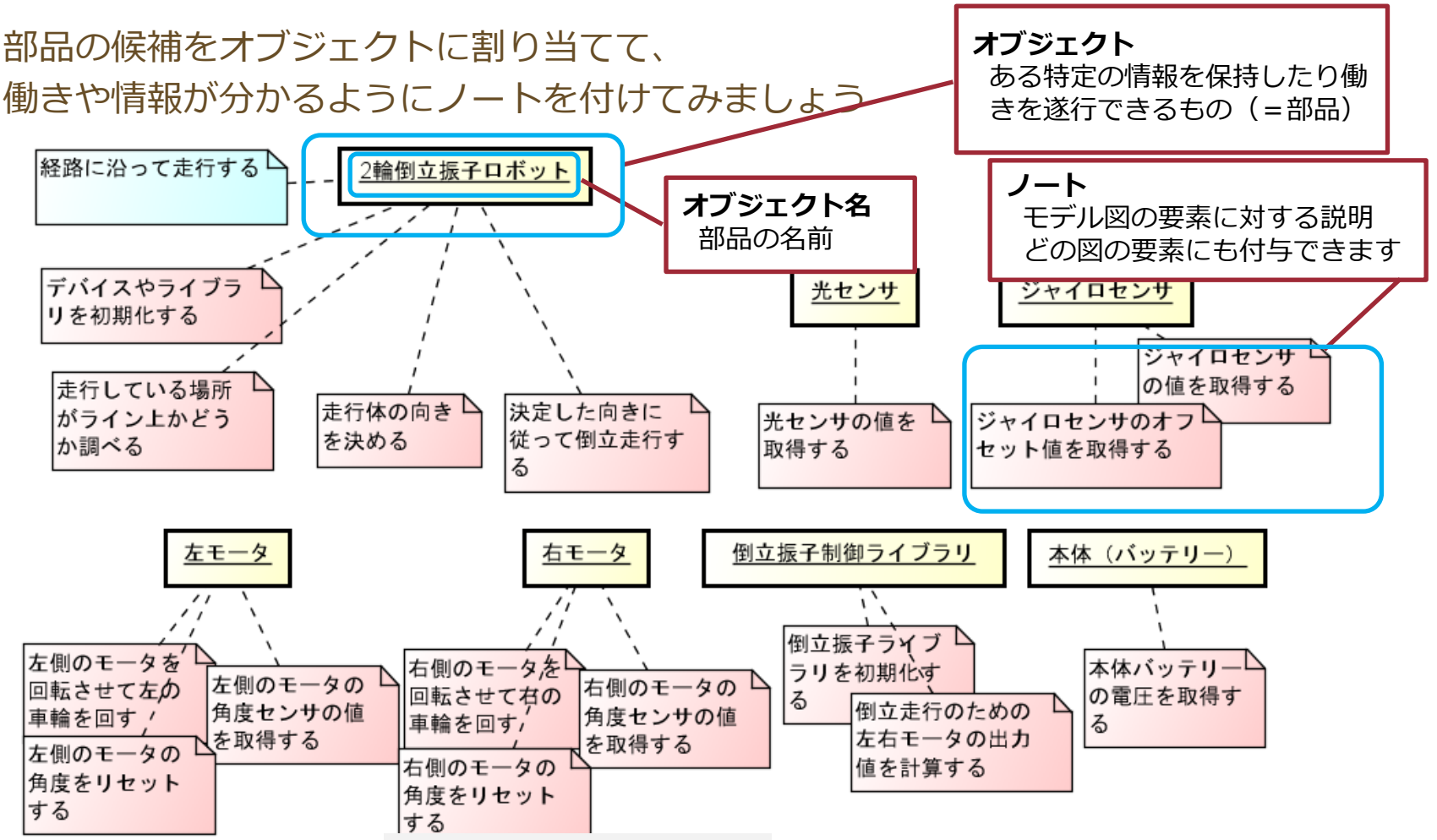
「働き」や「情報」	部品の候補
経路に沿って走行する	2輪倒立振子ロボット
デバイスやライブラリを初期化する	2輪倒立振子ロボット
走行している場所（ライン上かどうか）を調べる	2輪倒立振子ロボット
光センサの値を取得する	光センサ（EV3ではカラーセンサを使う）
走行体の向きを決める	2輪倒立振子ロボット
ジャイロセンサのオフセット値を取得する	ジャイロセンサ
ジャイロセンサの値を取得する	
本体バッテリーの電圧を取得する	本体（インテリジェントブロック）のバッテリー
左右モータの角度センサの値を取得する	左モータ、右モータ （モータを回転させると車輪が回転しロボットは走行する）
左右モータを回転角度をリセットする	
左右モータを回転させる	
倒立振子ライブラリを初期化する	倒立振子制御ライブラリ
倒立走行のための左右モータの出力値を計算する	
決定した向きに従って倒立走行する	2輪倒立振子ロボット

ロボットの部品の候補と働き

3-6. 部品の候補を見つける（2）

- 部品の候補を図で表してみましょう
  - オブジェクト図を使ってみましょう
  - 部品の候補をオブジェクトに割り当てて、働きや情報が分かるようにノートを付けてみましょう

設計モデル・構造「ロボットの部品の候補と働き」を開いて、オブジェクト図を確認しましょう



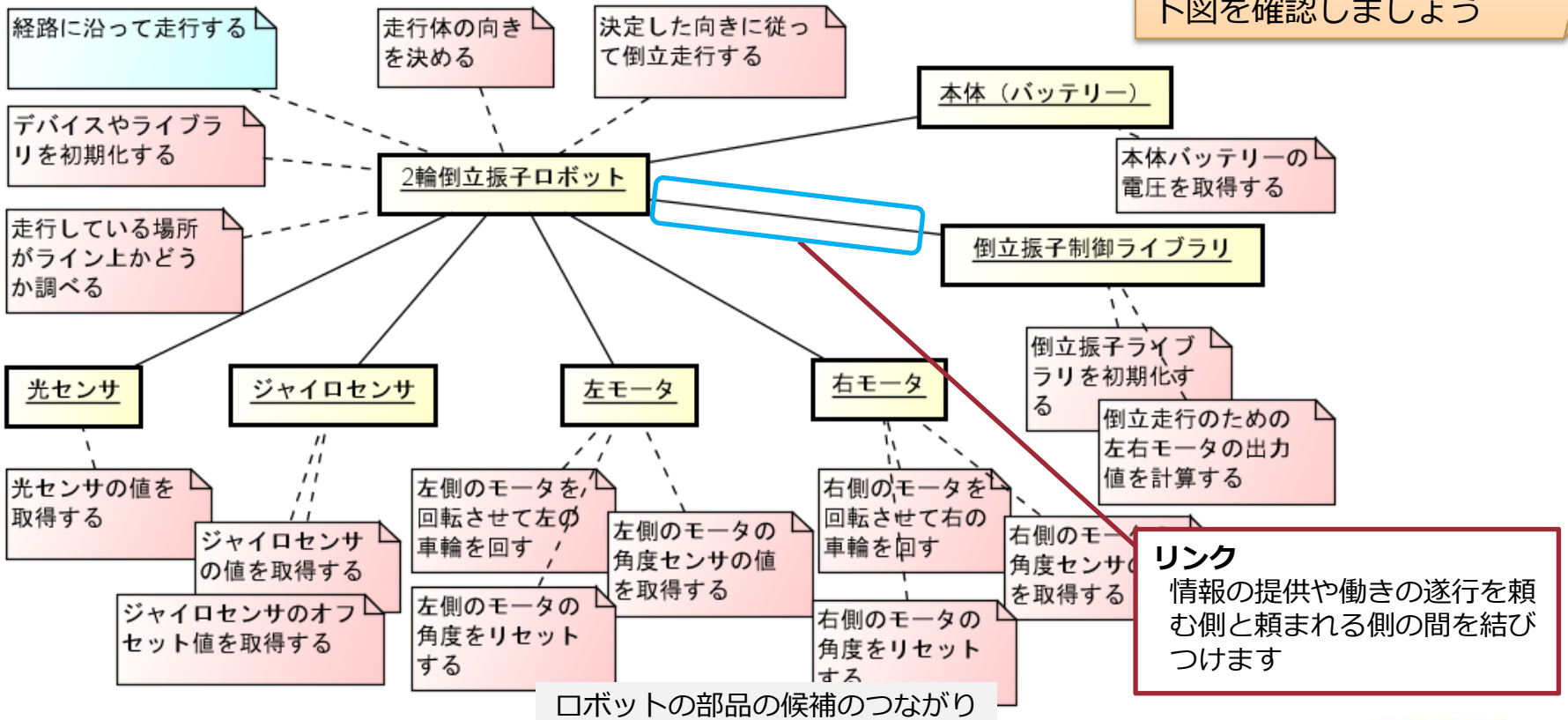
ロボットの部品の候補と働き（オブジェクト図）

3-7. 部品のつながりを整理する

基本設計

- 処理を頼む側と頼まれる側の間を線（リンク）でつなぎます
  - リンクが引けないオブジェクトが見つかったら、吟味します
    - そのオブジェクトは、他の部品の働きや情報かもしれない
    - そのオブジェクトは、部品の候補から外れるかもしれない

設計モデル・構造「ロボットの部品の候補のつながり」を開いて、オブジェクト図を確認しましょう

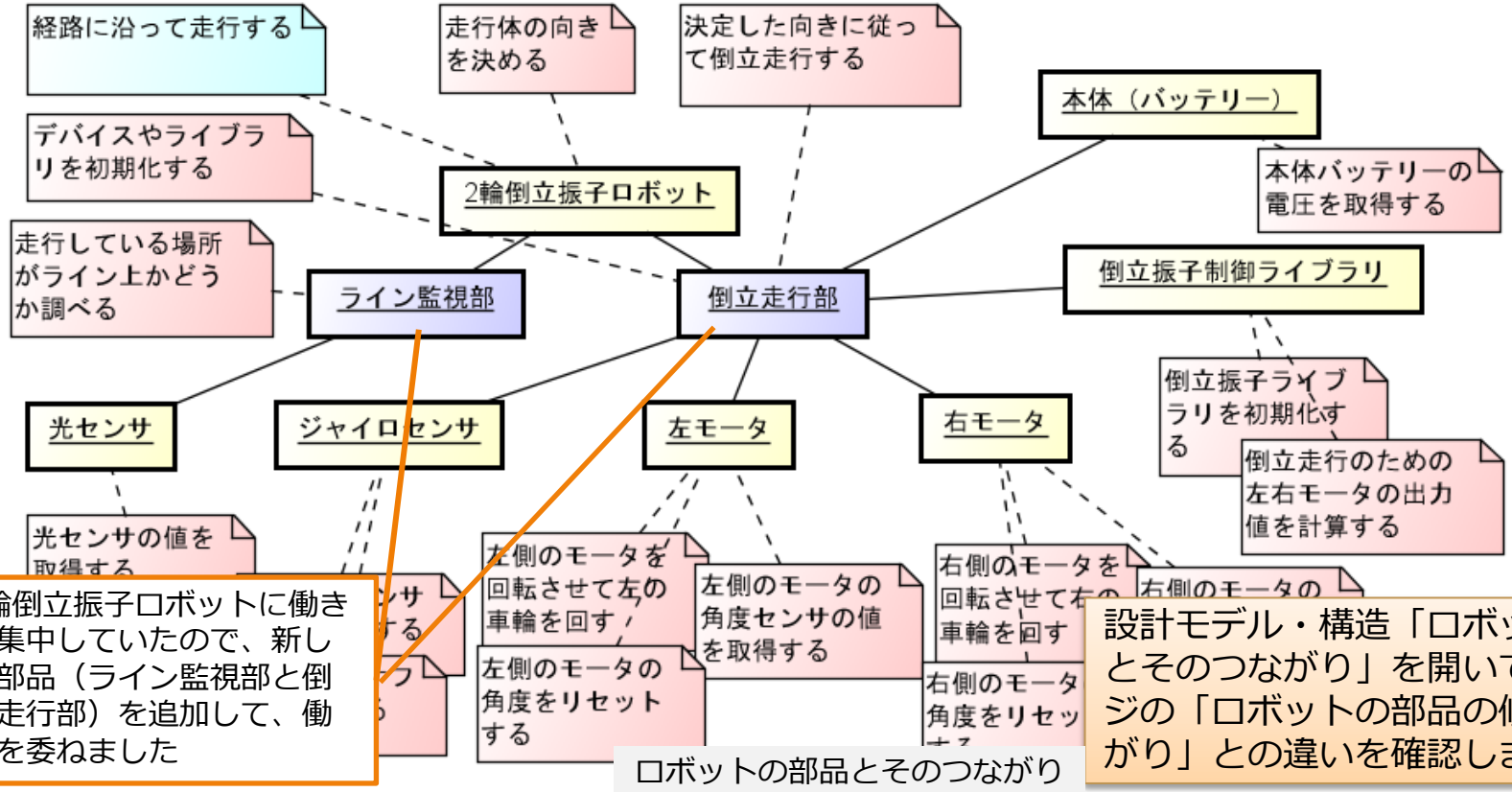


ロボットの部品の候補のつながり（オブジェクト図）

3-8. 大きな部品を分割する

基本設計

- 働きが集中している部品を探して分割を考えます
  - 各部品が本来やるべき働きを残し、他の働きは別の部品に手分けします
  - 手分けした働きを担当する新しい部品を用意して、名前を付けます
  - ひとつの部品が担う働きが少なくなり、再利用しやすくなります

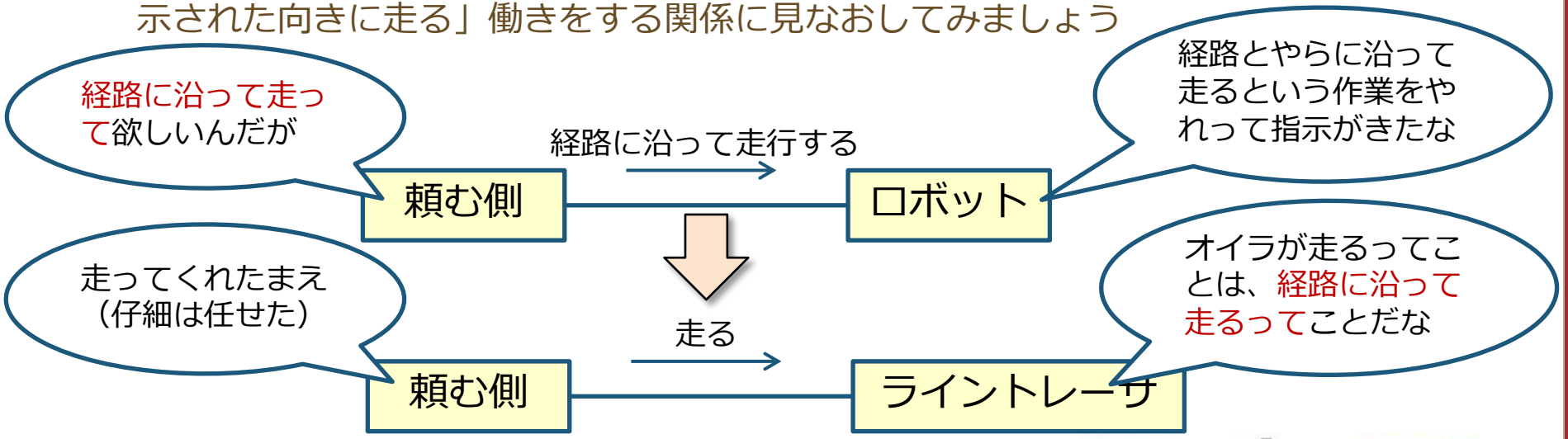


ロボットの部品とそのつながり
（オブジェクト図）

3-9. 役割が分かる名前と働きに変える

基本設計

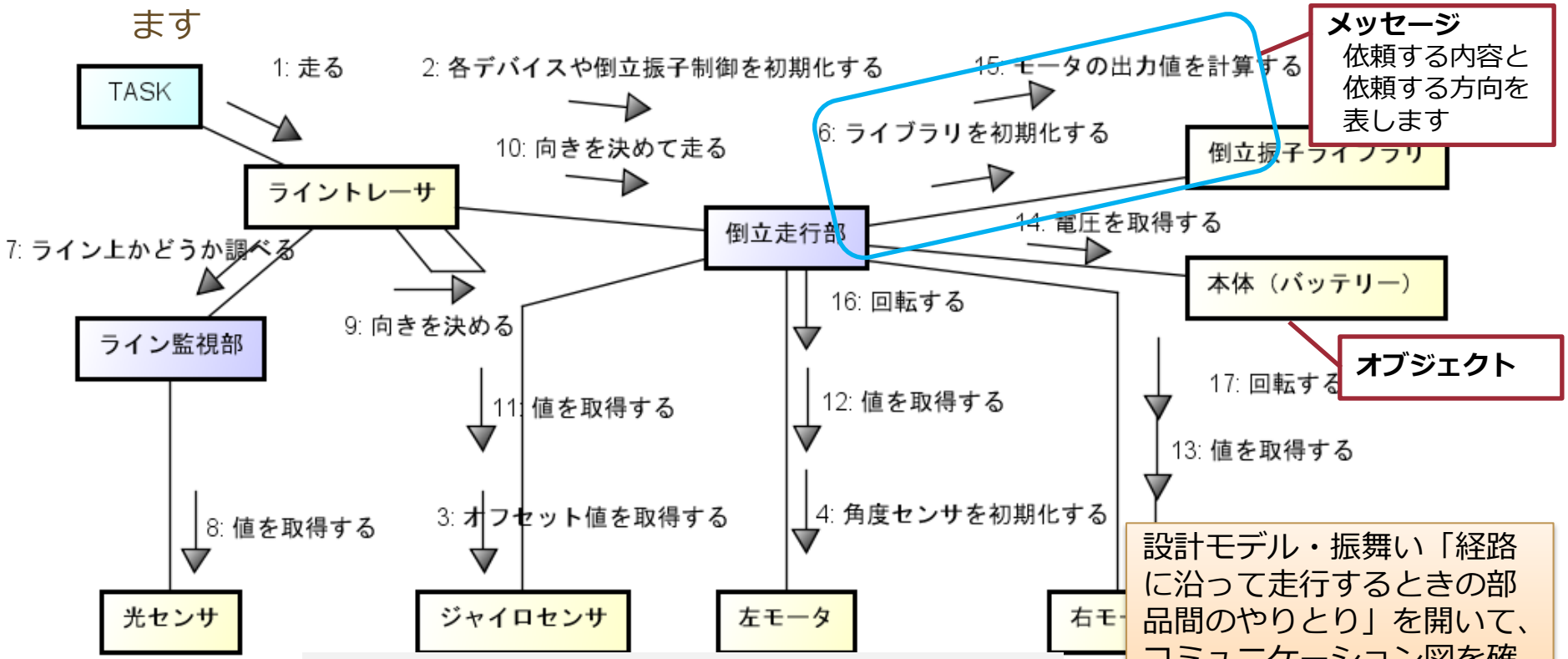
- ロボット自身も大きな部品と捉えて、役割が分かる部品名を考えましょう
  - 「経路に沿って走行する」ロボットで「ライントレーサ」としましょう
    - 「ライントレース」は「経路に沿って走行する」部品（ここではロボット自身）の働きの名前
    - 部品の名前には「ライントレースする」モノや係なので「ライントレーサ」と名づけてみました
- 部品として名前をつけたら働きの呼び出し方も見なおしてみましょう
  - ロボットに「経路に沿って走行する」と指示する関係から、ライントレーサに「走る」ことを頼むと、ライントレーサが「経路に沿って走行する」働きをする関係に見なおしてみましょう
  - 倒立走行部に「向きを決めて走る」ことを頼むと、倒立走行部が「倒立走行しながら指示された向きに走る」働きをする関係に見なおしてみましょう



3-10. 部品による機能実現を確認する

基本設計

- 部品を組合わせて動作させたときに、機能が実現できるか確認します
  - コミュニケーション図を使ってみましょう
  - 「経路に沿って走行する」アクティビティ図（3.5節）を参照しながら書きます
  - 頼まれる部品の「働き」をメッセージにして、「協調して動作」するように書きます
  - 部品の妥当性（働きの割り当てやつながり）や網羅性（ヌケモレが無いこと）を確認します



ユースケース「経路に沿って走行する」の部品間のやりとり（コミュニケーション図）

3-11. 部品の仕様を定義する

- これまでの検討結果を元に「部品の仕様（設計図）」を定義します
  - クラス図を使ってみましょう
  - UMLでは部品仕様をクラスと呼びます
  - クラス図で部品の仕様をまとめて定義しておくと、
    部品の仕様だけではなく、
    部品どうしのつながり（関連）も視覚的に定義できます
  - 次のクラス図は、倒立走行部とモータの部品仕様を表します

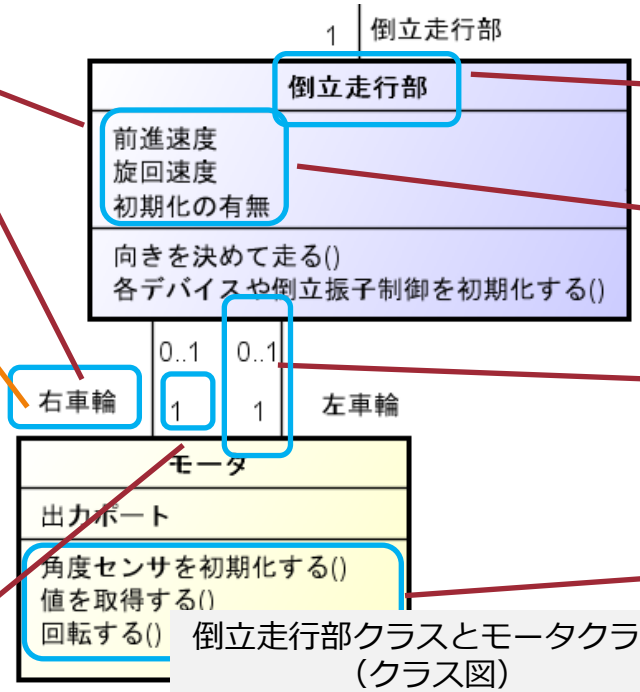
クラス間のつながりを示す線が関連で、
クラスのインスタンス（オブジェクト）の
間のつながりを示す線がリンクです
リンクは関連のインスタンスとみなすこともできます

クラス
部品の仕様を表します

ロール名（関連端名）
相手の部品名を表します

これまで左右モータとしていましたが、同じ種類のモータが役割別に1つずつ使われていることがわかりました。そこで、役割ごとに別の関連とし、関連端名をつけました

多重度
相手の部品がつながる数を表します



倒立走行部クラスとモータクラス（クラス図）

クラス名
仕様の名前を表します

属性
部品が保持する情報を記述します

関連と多重度
両端の部品どうしがつながることを表します

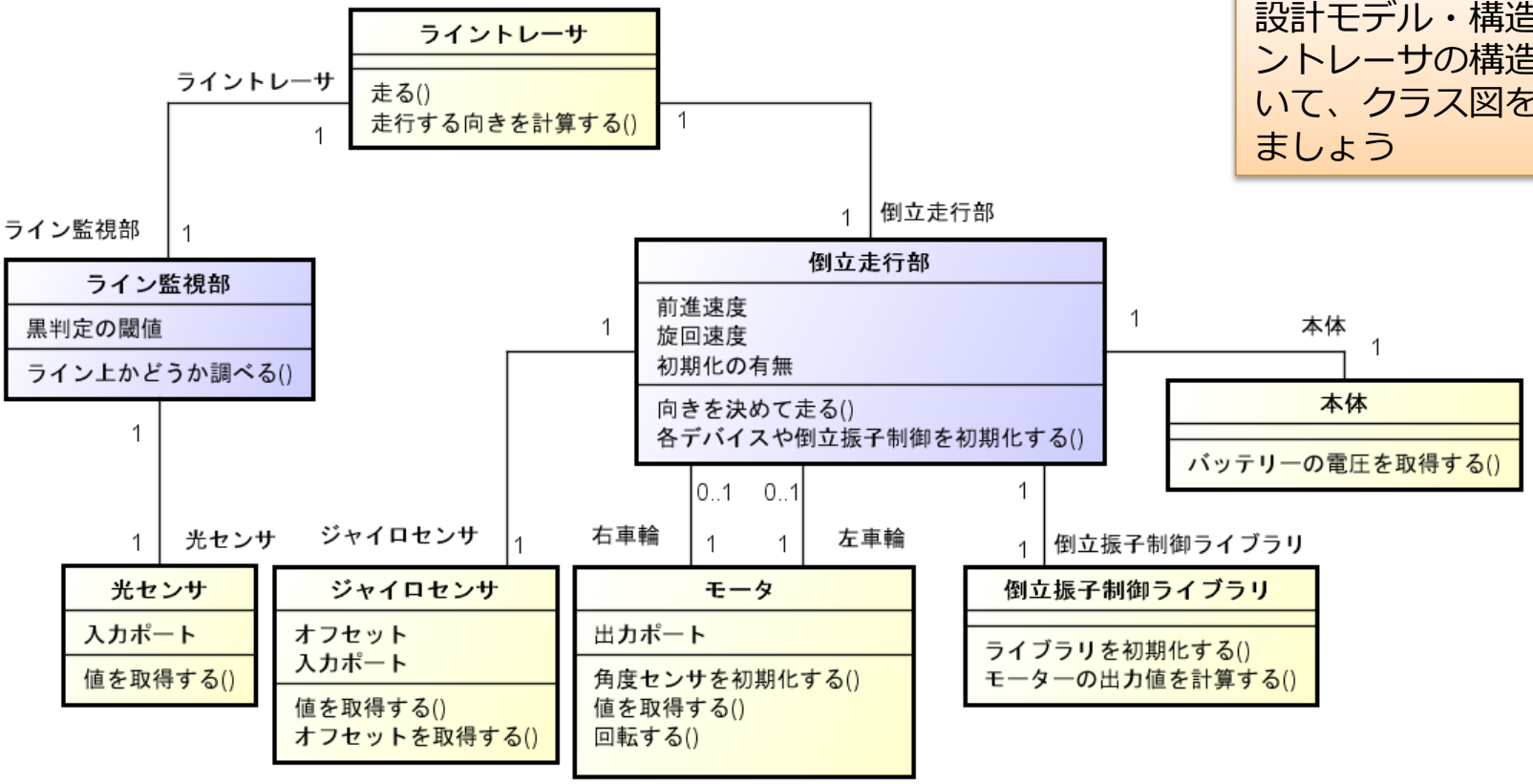
操作
部品の働きを記述します

3-12. システムの構造を決定する

基本設計

- 2輪倒立振子ロボットの構造を決定しましょう
  - クラス図を使ってみましょう
  - 部品をクラスに、働きをクラスの操作に割り当ててみましょう

設計モデル・構造「ライントレーサの構造」を開いて、クラス図を確認しましょう

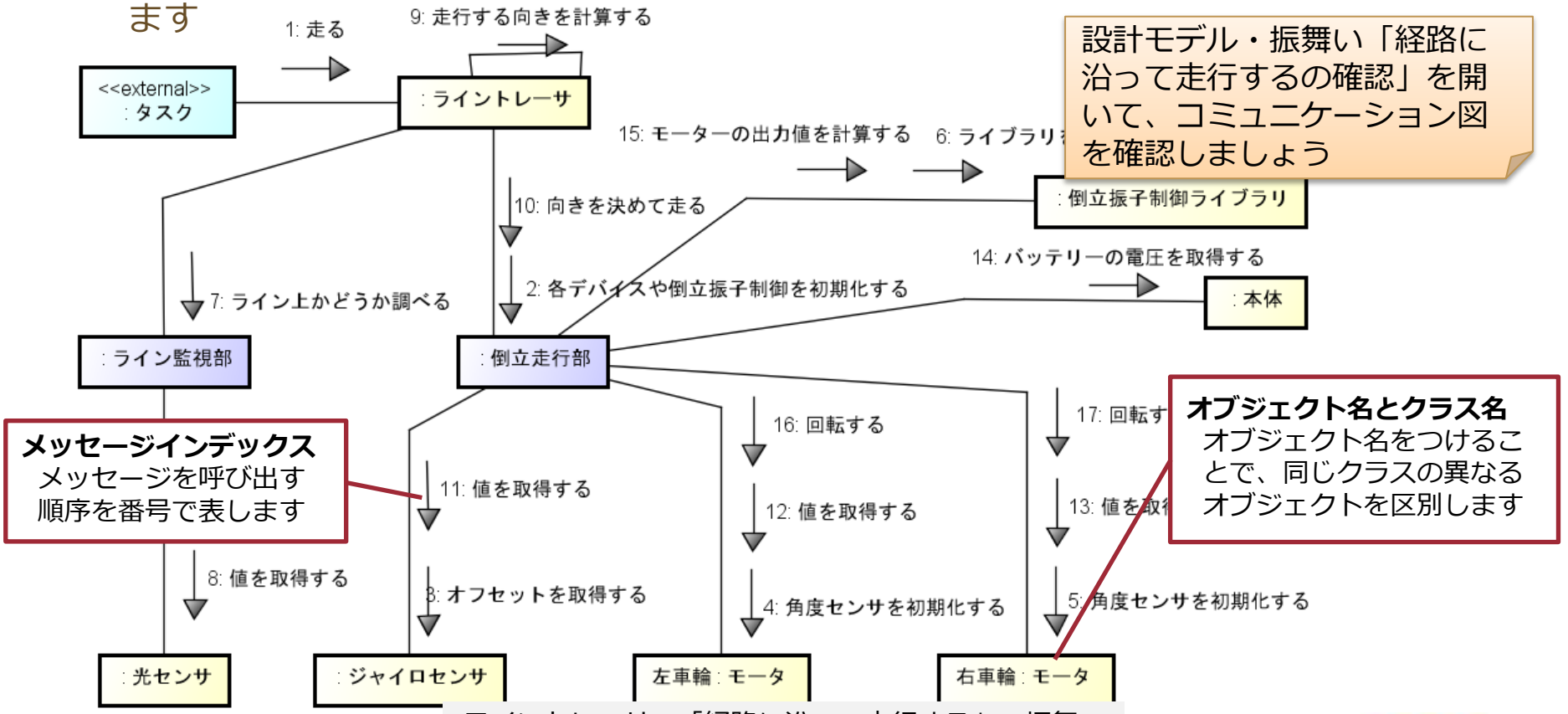


経路に沿って走行する2輪倒立振子ロボットの構造（クラス図）

3-13. システムの振舞いを確認する（1）

基本設計

- クラスに割り当てた働きのつながりを確認します
  - コミュニケーション図を使ってみましょう
  - システムのクラス図（3.11節）を元に書きます
  - 図が書けたら、ユースケース「経路に沿って走行する」が実現できていることを確認します



設計モデル・振舞い「経路に沿って走行するの確認」を開いて、コミュニケーション図を確認しましょう

メッセージインデックス
メッセージを呼び出す順序を番号で表します

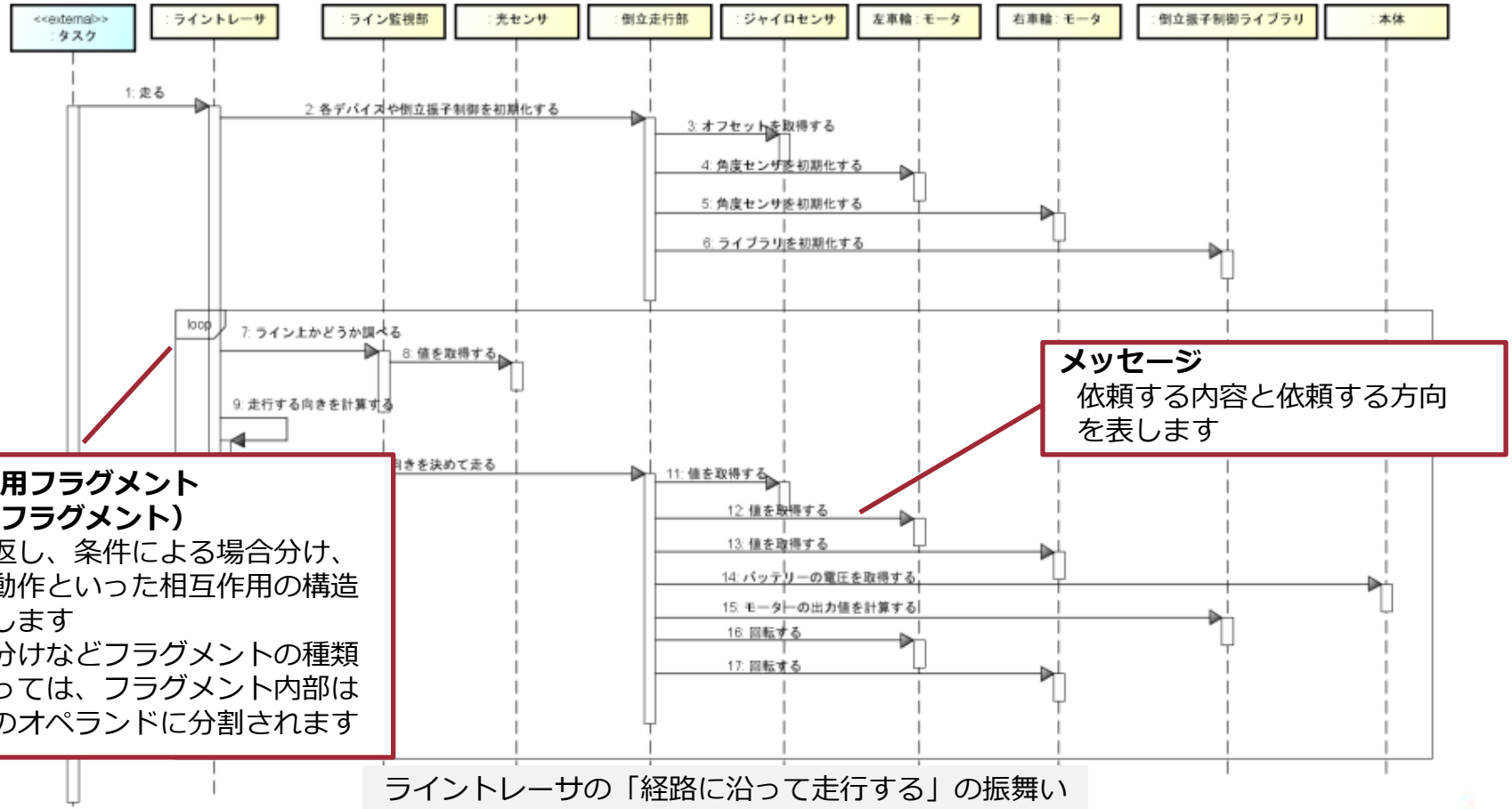
オブジェクト名とクラス名
オブジェクト名をつけることで、同じクラスの異なるオブジェクトを区別します

ライントレーサの「経路に沿って走行する」の振舞い（コミュニケーション図）

3-13. システムの振舞いを確認する（2）

基本設計

- 時間軸に沿って動作を確認します
  - シーケンス図を使ってみましょう
  - コミュニケーション図よりもメッセージの呼び出し順序が分かりやすくなります



メッセージ
依頼する内容と依頼する方向を表します

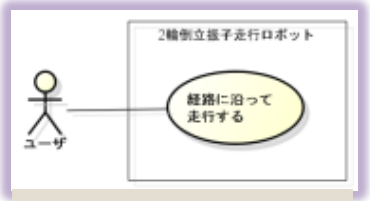
相互作用フラグメント（結合フラグメント）
繰り返し、条件による場合分け、並行動作といった相互作用の構造を表します
場合分けなどフラグメントの種類によっては、フラグメント内部は複数のオペランドに分割されます

ライントレーサの「経路に沿って走行する」の振舞い（シーケンス図）

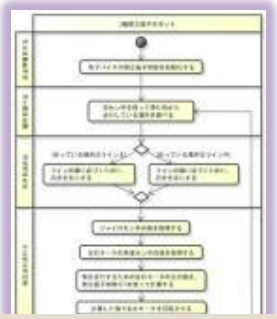
3-14. まとめ

- モデルを使った設計の一連の流れ

【機能の視点】何を提供するのか？どのようにすれば実現できるのか？



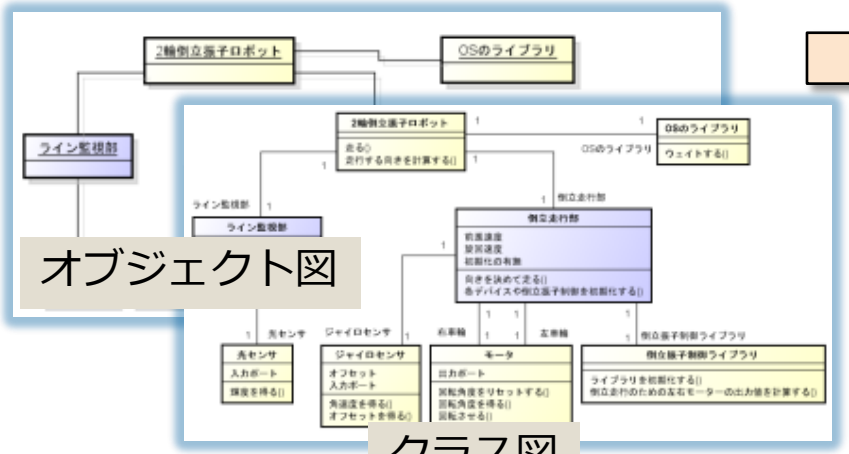
ユースケース図



アクティビティ図

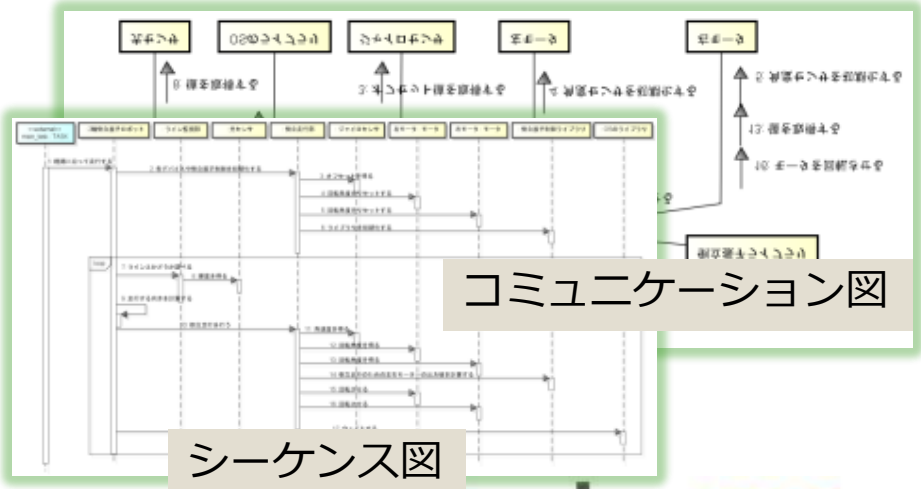
【構造の視点】
そのためには、どんな部品が必要か？

【振舞いの視点】
部品をどのように
動かせば良いのか？



オブジェクト図

クラス図



コミュニケーション図

シーケンス図

理解度チェック（3）

- モデルと部品を使ってソフトウェアを設計する

1. UMLでモデルを書くときの３つの視点とは何ですか？
2. 以下を表現するためのUML図は何ですか？
   a. 対象システムが提供する「機能」
   b. 対象システムの「振る舞い」の仕様
   c. 対象システムを分解したときの部品を定義
   d. 各部品の仕様
   e. 部品どうしの協調動作
   f. 部品の操作に対する制御フローや状態に応じた部品内部の動作
3. 走行体をきちんとライントレースさせるために
   a. 走行体をスタート位置に設置するときに気をつけなくてはならない事は何でしょうか？
   b. 例題１において、なぜ、２～４は4msで繰り返さないといけないのでしょうか？