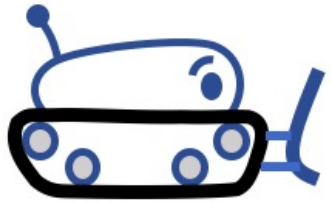
モデル駆動開発



自己紹介

- 浜名将輝
- 株式会社ゆめみ
- お仕事：スマホアプリ/サーバサイド
  - Kotlin/Spring Boot
- 大学院：バイナリ合成(高位合成)
  - RISC-V binary -> Verilog HDL

最近はおうち探しにハマっています

SUUMO楽しいです

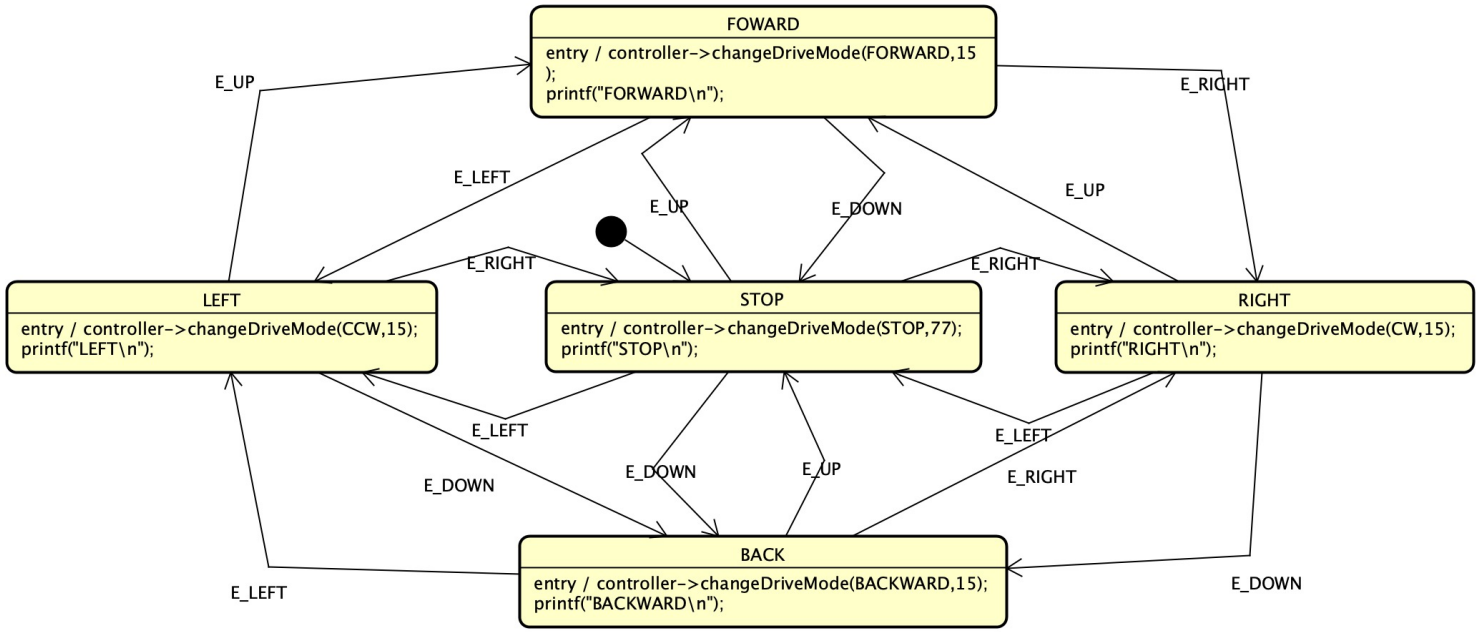
この講義の目的

- MDDの概要を掴む
- MDDのメリット/デメリットを知る
- MDDを体験する
- 実習の開発の流れをイメージできる

- この資料はお手元にダウンロードしてください

モデル駆動開発とは？

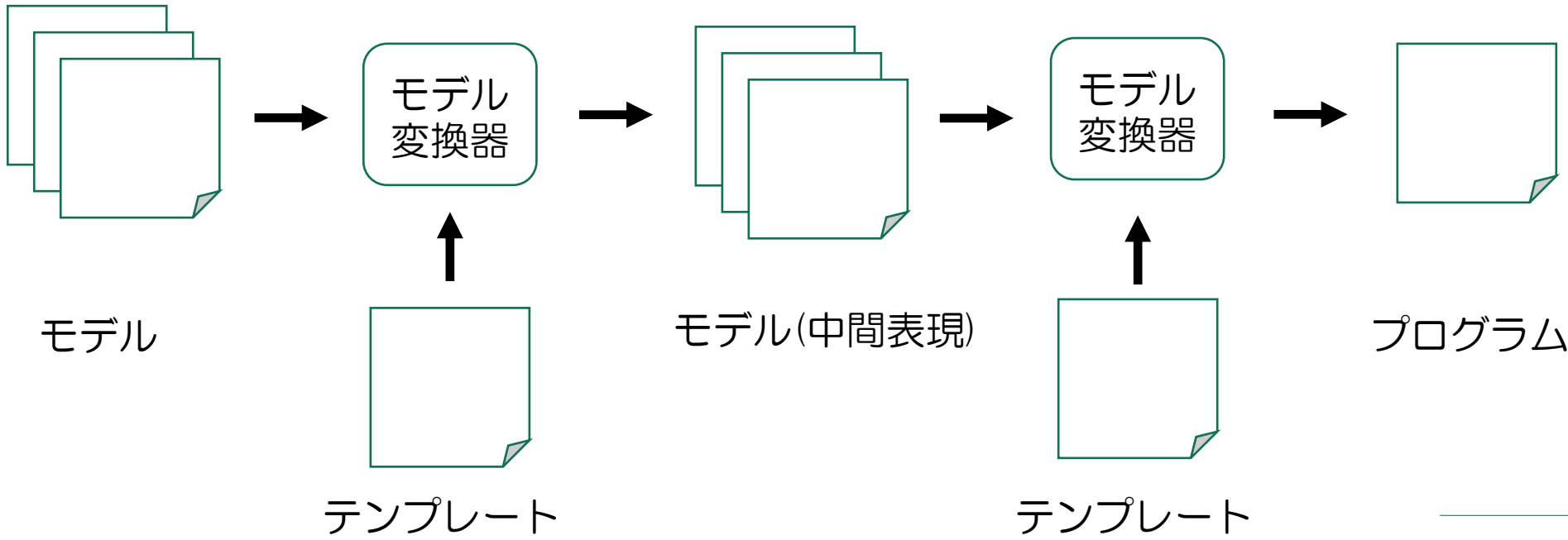
Model Driven Development
= 開発の抽象度を上げる手段の一つ
開発コストが減る




```
1  void LEDTank::doTransition(unsigned long event) {
2      this->_beforeState = this->state;
3      switch(this->state) {
4      case _STATE_INITIAL:
5          this->state = STATE_STOP;
6          // entry
7          controller->changeDriveMode(STOP, 77);
8          printf("STOP\n");
9          break;
10     case STATE_STOP:
11         if(((event & E_UP) != 0)) {
12             this->state = STATE_FOWARD;
13             controller->changeDriveMode(FORWARD, 15);
14             printf("FORWARD\n");
15             this->state = STATE_LEFT;
16             controller->changeDriveMode(CCW, 15);
17             printf("LEFT\n");
18         } else if(((event & E_DOWN) != 0)) {
19             this->state = STATE_BACK;
20             controller->changeDriveMode(BACKWARD, 15);
21             printf("BACKWARD\n");
22         } else if(((event & E_RIGHT) != 0)) {
23             this->state = STATE_RIGHT;
24             controller->changeDriveMode(CW, 15);
25             printf("RIGHT\n");
26         }
27         break;
28     }
29     ...
30 }
```

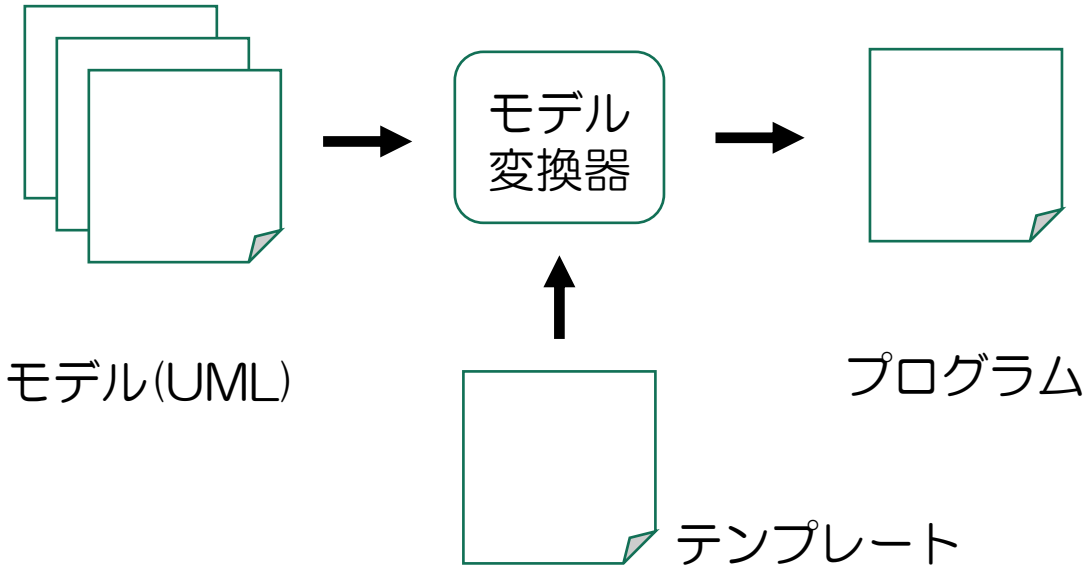
MDDでの開発工程

- モデルとは？
  - 成果物を説明するための文章や図や表の総称(ドキュメント)
- MDDとは？
  - モデルの変換を繰り返してやがてコードに落としていく



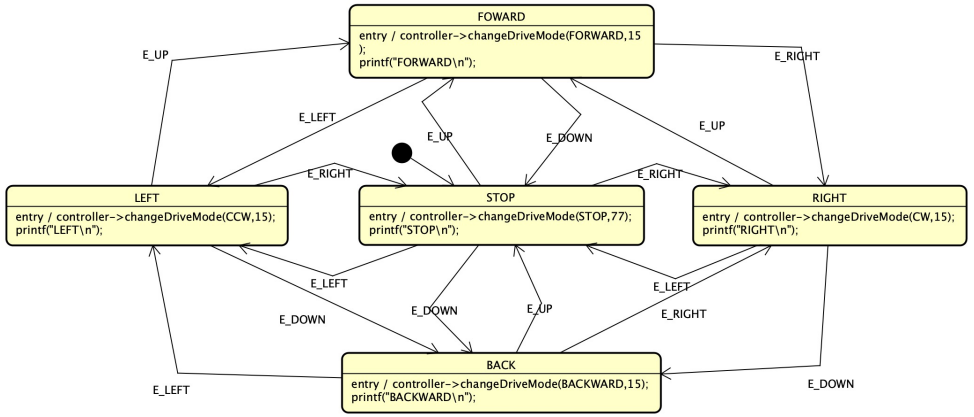
UMLを用いたMDD

- ただの文章や図、表を別の形式に変換することは難しい
  - 人によっていろんな書き方がある
- モデルにUMLを用いることで変換を容易にする
  - 成果物の設計記述を統一するために作られた
  - 一定の変換規則を当てはめやすい



MDDのメリット

- 開発の抽象度が上がる
  - コードよりも直感的に理解しやすい
- 開発時の作業量が削減できる
  - 作ったモデルがそのまま資料(ドキュメント)になる



MDDのデメリット

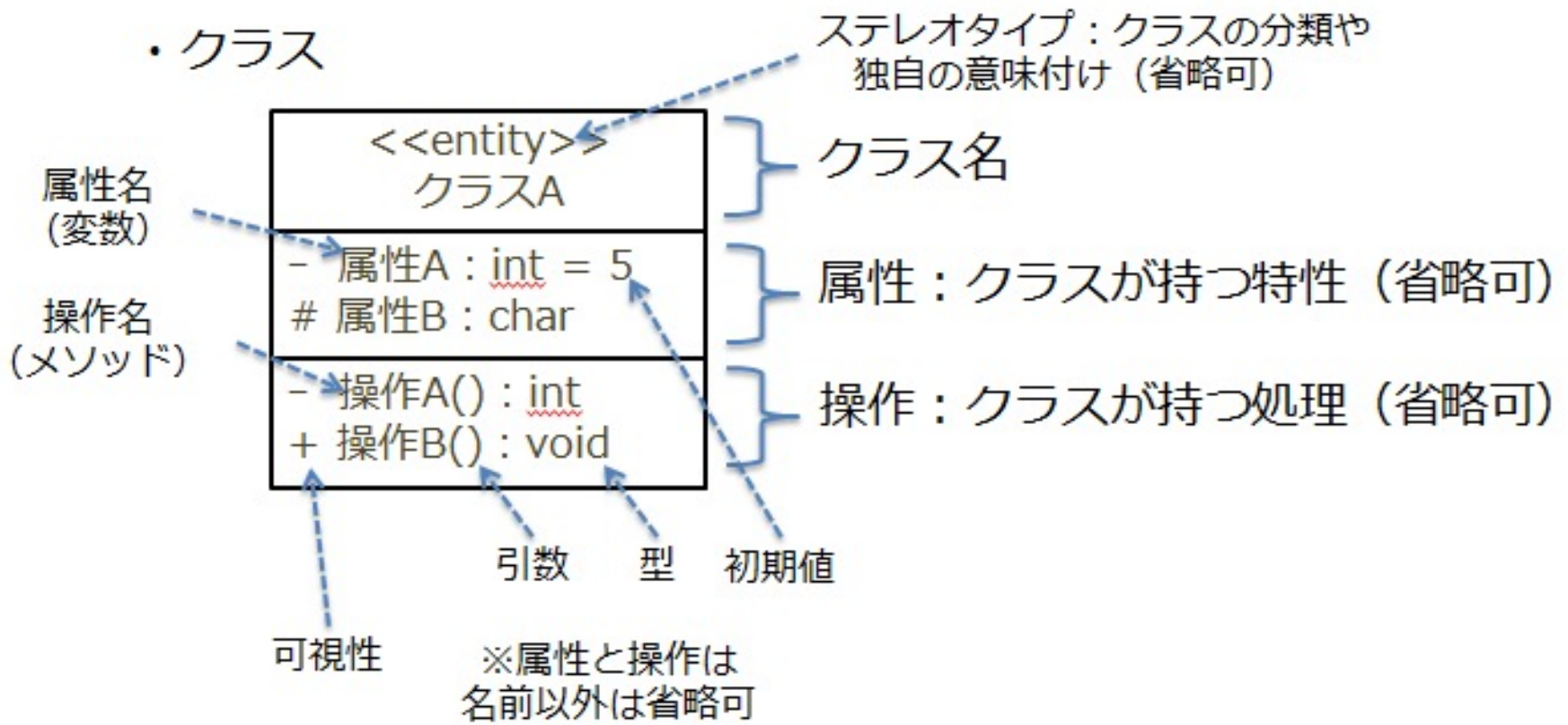
- 抽象度が上がることによる弊害
  - バグの特定が困難になる
  - パフォーマンスチューニングがしづらくなる
- モデルからコードを変換する仕組みを作るのに工数がかかる

メリット/デメリットを考えて採用するかどうかは要検討

LED-Camp9で行うMDD

- クラス図でロボットのオブジェクト情報を記述
- ステートマシン図でロボットの振る舞いを記述
- クラス図とステートマシン図からC++コードを自動生成！
- C++コードを他のライブラリとリンクして動きをシミュレート
- 最終的には競技会で高得点を取れるロボットを開発する

UML:クラス図


・クラス
ステレオタイプ：クラスの分類や
独自の意味付け（省略可）
<<entity>>
クラスA
クラス名
属性名
（変数）
－ 属性A：int ＝ 5
属性B：char
属性：クラスが持つ特性（省略可）
操作名
（メソッド）
－ 操作A()：int
＋ 操作B()：void
操作：クラスが持つ処理（省略可）
引数
型
初期値
可視性
※属性と操作は
名前以外は省略可

UML:ステートマシン図

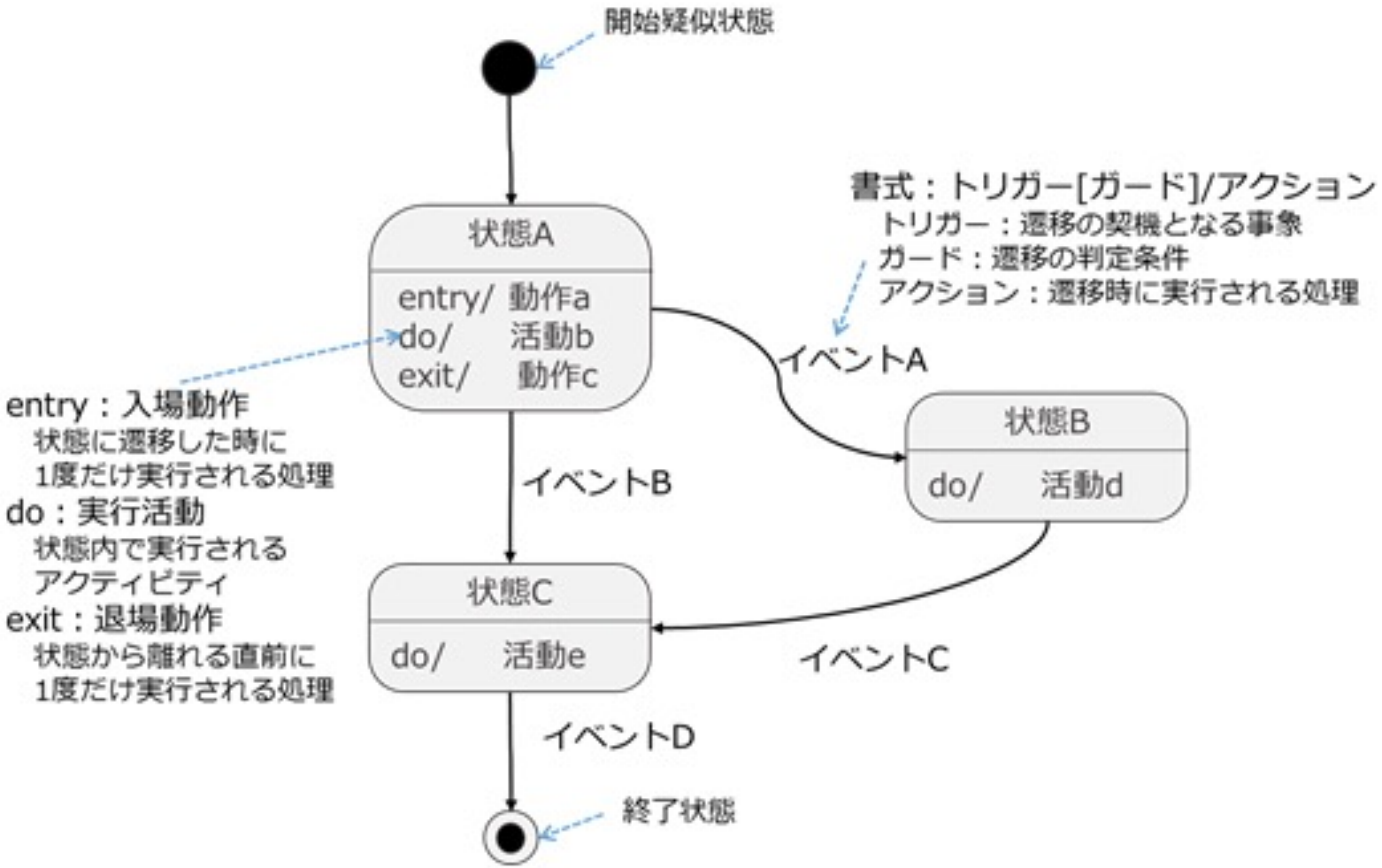
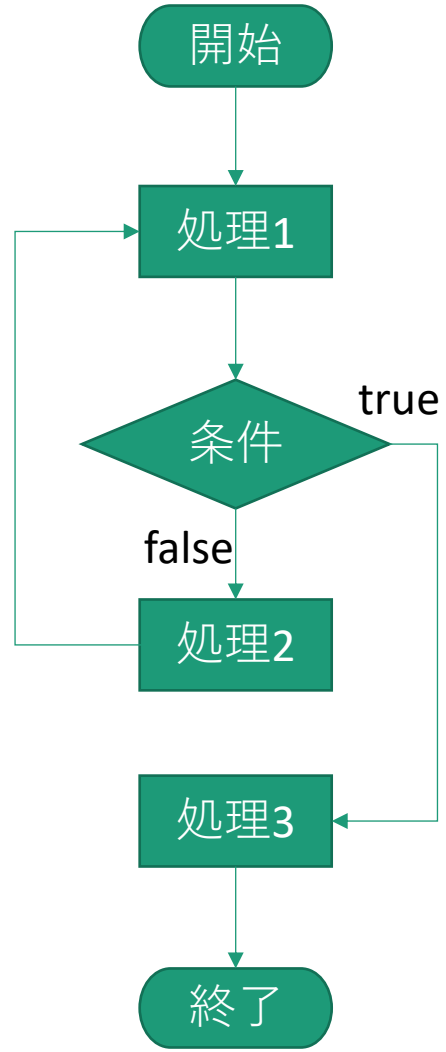


Figure 1 ステートマシン図

(余談) フローチャート図との違い

- ステートマシン図
  - 単一オブジェクトの振る舞い
  - オブジェクトの状態と活動
  - 外からのイベントによる状態遷移
- フローチャート図(flow chart)
  - システム全体(または一部)の処理の流れ(flow)、順番
  - 分岐はシステム内部の処理結果に基づく



扇風機を表現してみよう

- 仕様
  - 出力は2段階の設定が可能
  - ON/OFF/低/高 と書かれたボタンがある
    - ON/OFFボタンで運転開始/停止を制御する
    - ONを押すと出力低で運転する
    - 運転中に低/高ボタンを押すことで出力を制御する
  - ボタンが押されると「ピッ」と音がなる

扇風機クラス図

- (属性)level：byte
  - モータへの出力レベルを保存する変数
  - 0,1,2のどれかしか取らない
- (操作)outputPulse(void)
  - levelの値を読みその値に応じたPWM信号をモータへ送出する
- (操作)changeLevel(byte value)
  - levelの値を変更する
  - levelの制約外の値が入力された場合は何もしない
- (操作)sound(void)
  - 音を鳴らす

扇風機ステートマシン図

- 状態はOFF/低運転/高運転の3つ
- OFF状態でONボタンが押されると低運転に遷移(音鳴らす)
- 低運転状態の時は低出力でPWM信号を送出

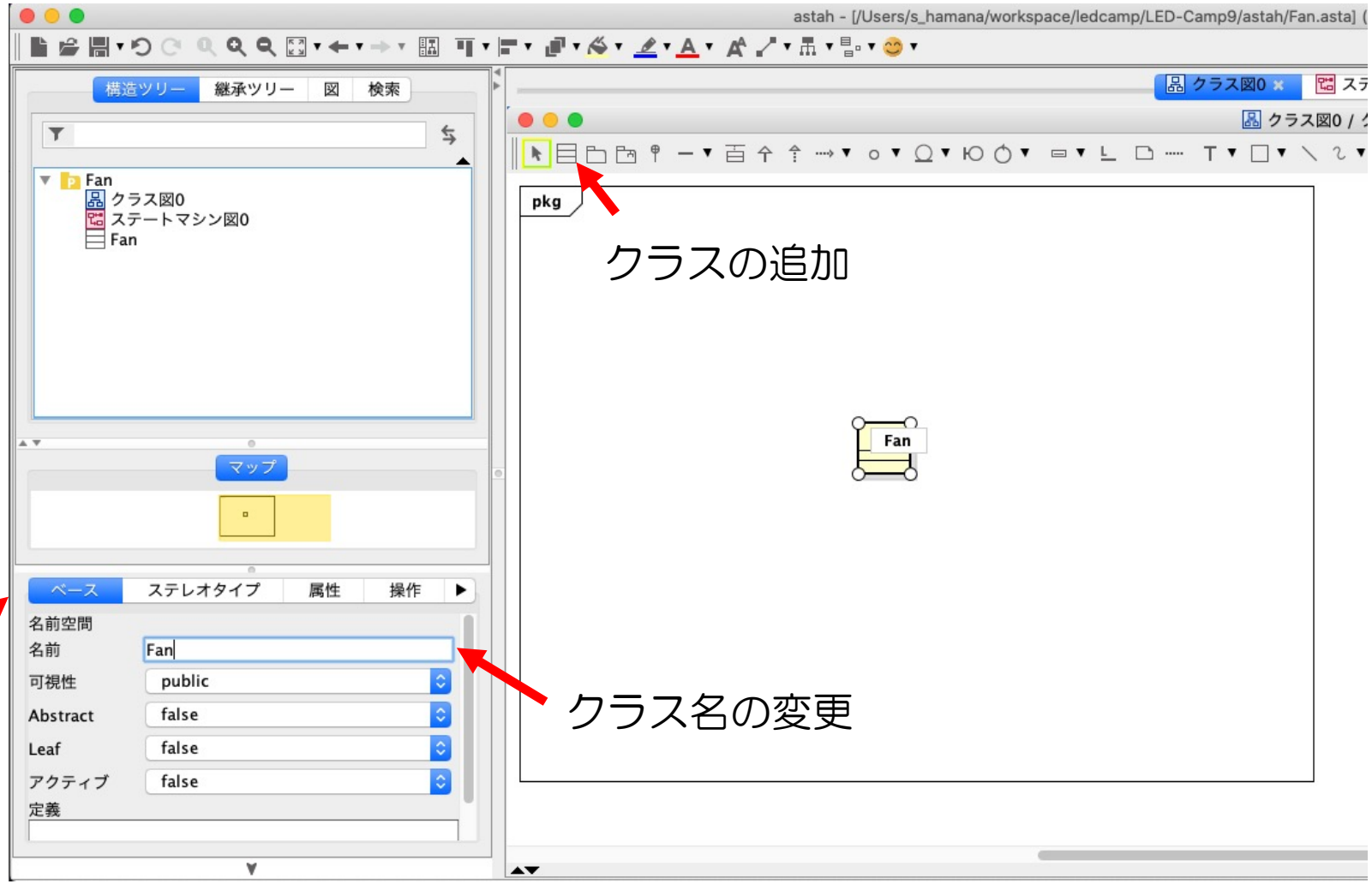
(これ以降はブレイクアウトルームで考えてみてください！)

作業中質問があれば実行委員にお気軽にどうぞ！

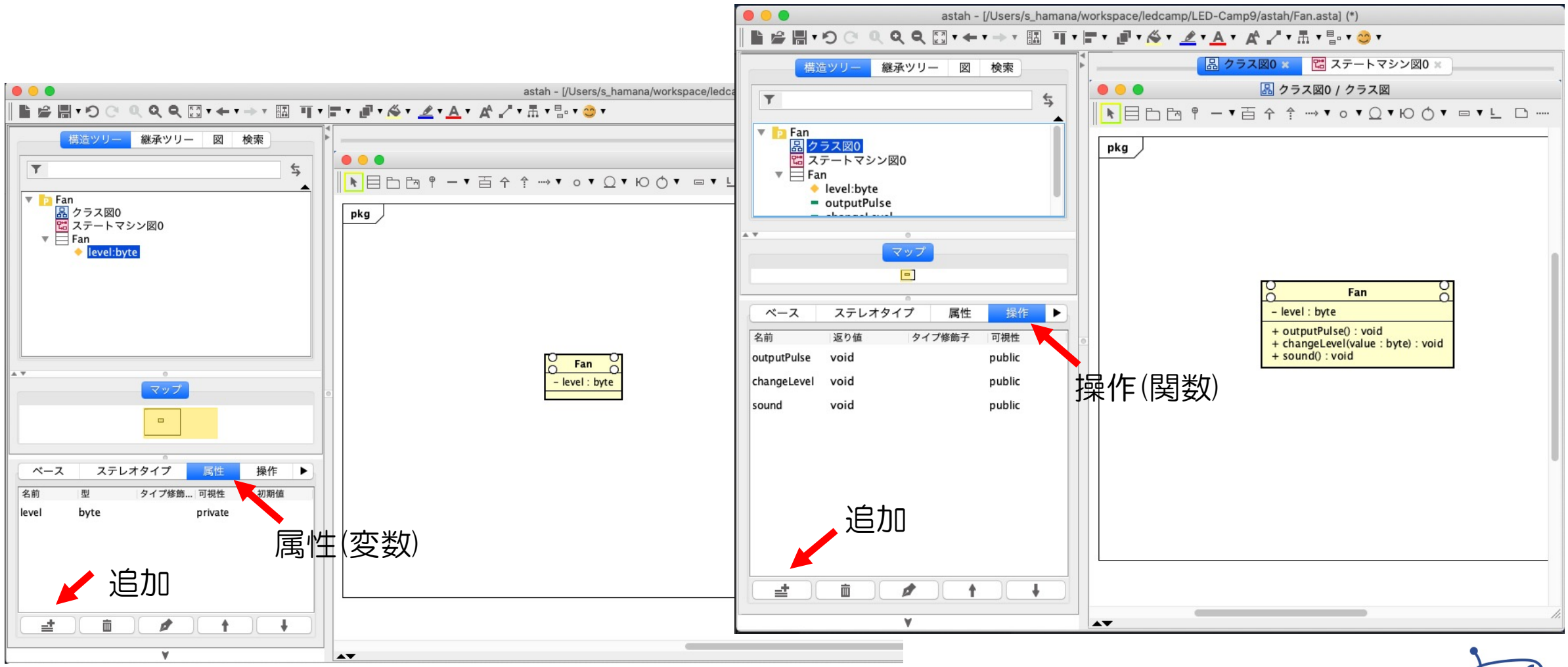
ハンズオン:準備

- astah*を起動
- [ファイル] -> [新規作成]
- no_titleを右クリック
  - 図の追加(クラス図/ステートマシン図を追加)
- いったんプロジェクトを保存
  - 保存時にFanと入力するとno_titleに変わる

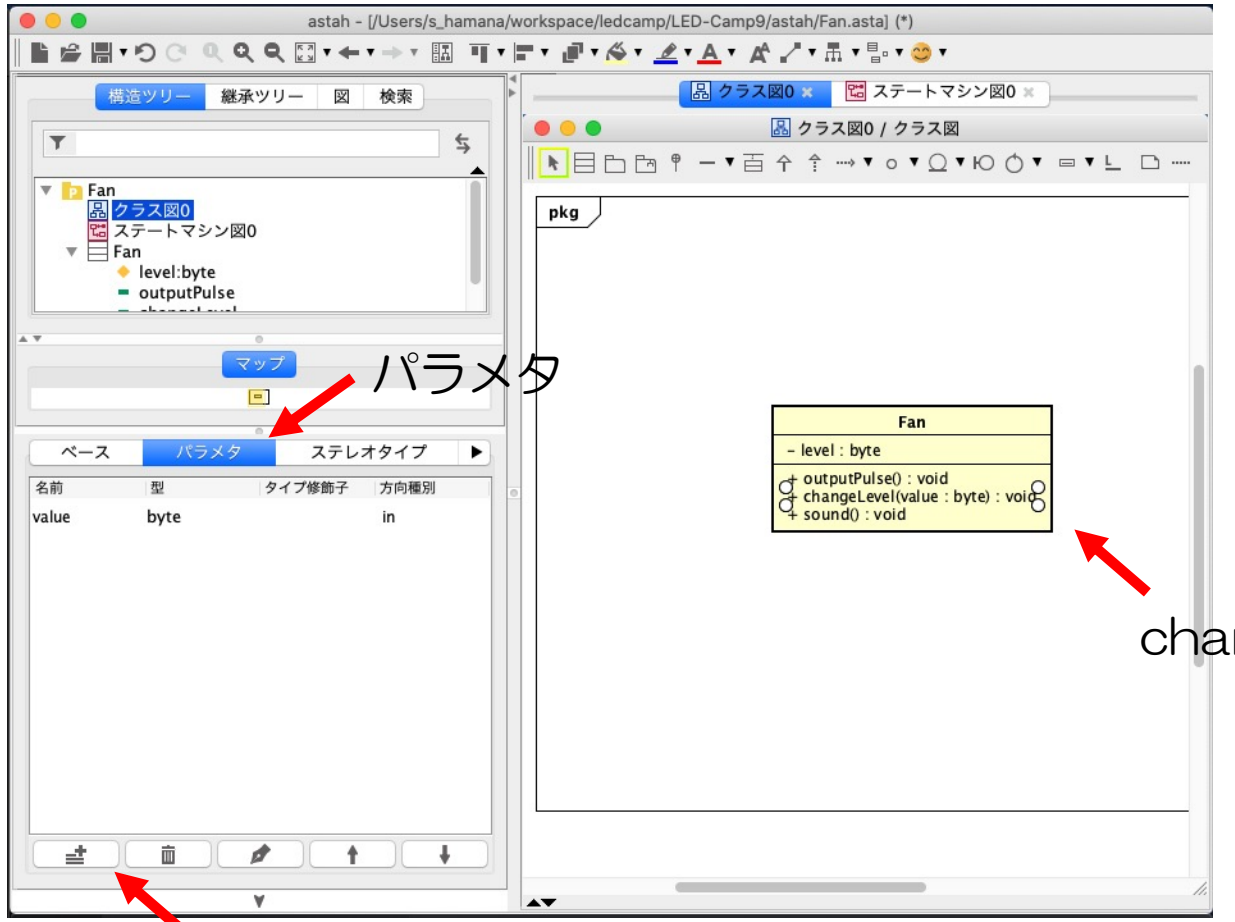
ハンズオン:クラス図の書き方1



ハンズオン:クラス図の書き方2

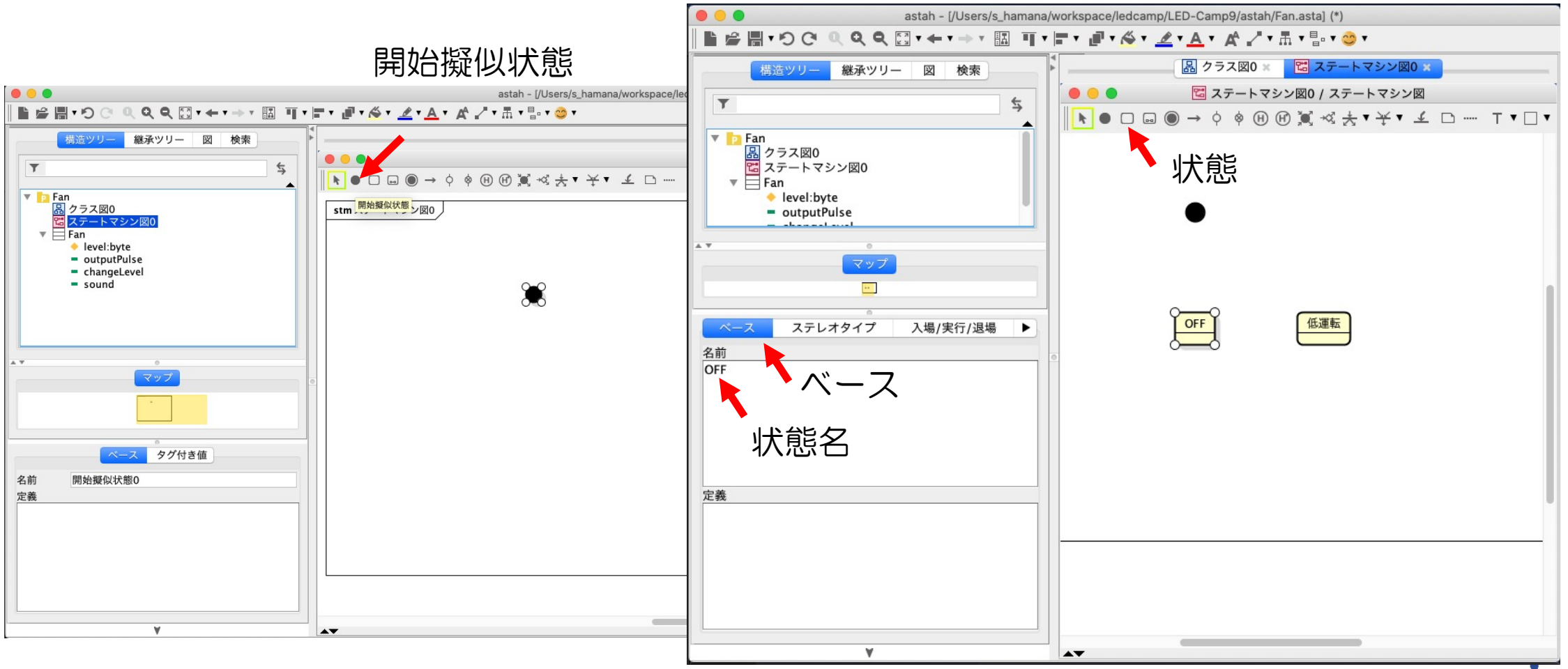


ハンズオン:クラス図の書き方3

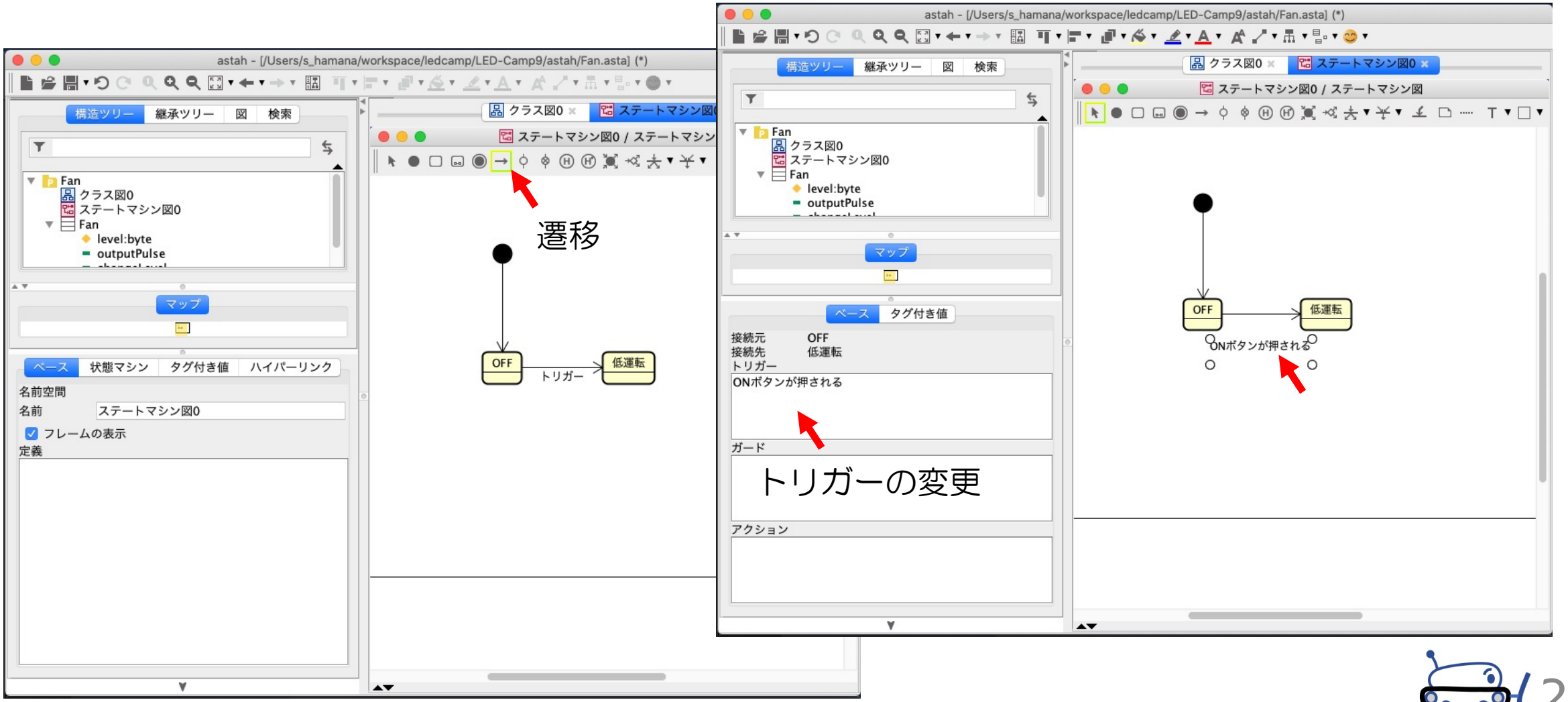


ハンズオン:ステートマシン図の書き方1

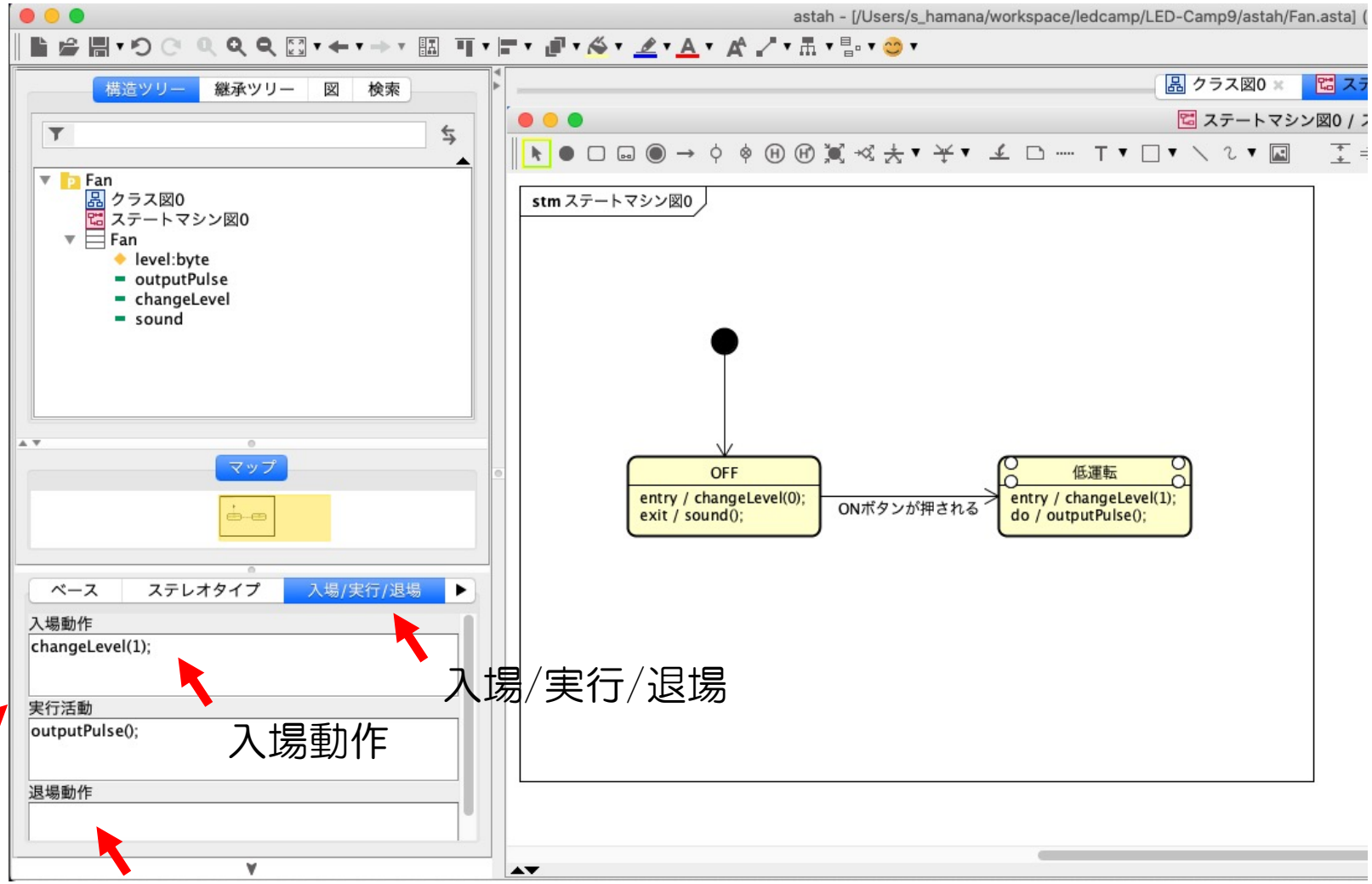
開始擬似状態



ハンズオン:ステートマシン図の書き方2



ハンズオン:ステートマシン図の書き方3



演習:扇風機ステートマシン図完成

- 状態はOFF/低運転/高運転の3つ
- OFF状態でONボタンが押されると低運転に遷移(音鳴らす)
- 低運転状態の時は低出力でPWM信号を送出

続きはブレイアウトルームで！

時間は15分とします

(再掲)LED-Camp9で行うMDD

- クラス図でロボットのオブジェクト情報を記述
- ステートマシン図でロボットの振る舞いを記述
- クラス図とステートマシン図からC++コードを自動生成！
- C++コードを他のライブラリとリンクして動きをシミュレート
- 最終的には競技会で高得点を取れるロボットを開発する

モデルをコードに変換する

- 変換用のテンプレートと変換機構を独自に用意する必要がある
  - 言うなればコンパイラ
  - これに多大な工数がかかる
  - この工数と開発工数とのトレードオフも重要

LED-Camp9でのMDD

- 変換用のテンプレート
  - 用意しました
- 変換機構
  - astah* プラグイン「m2t」を使用

Webotsでロボットを動かしてみよう

- ハンズオン
  - 実際に既存モデルをWebotsで動かしてみる
- 演習
  - モデルに手を加えてWebotsで動かしてみる

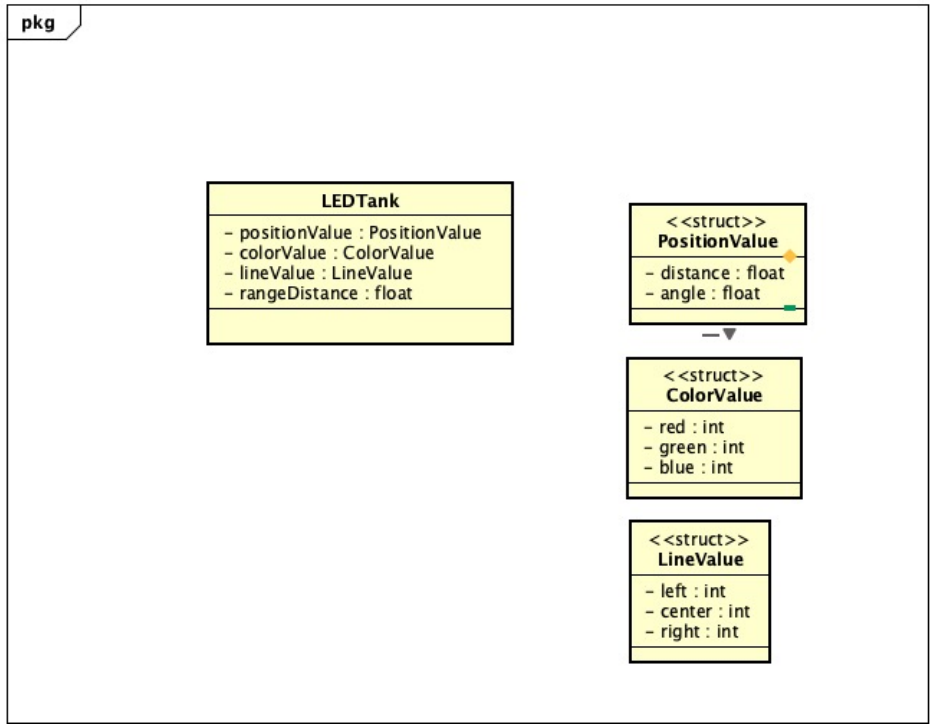
ハンズオン:準備

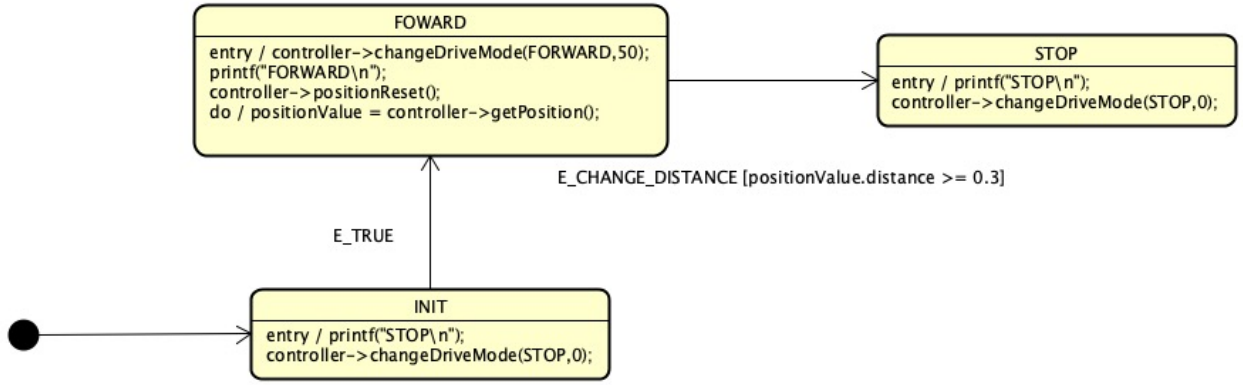
- 演習用リポジトリをclone
  - https://github.com/LED-Camp/LED-Camp9.git
- 演習用astahファイルを開く
  - astah/lecture.astah
- 演習用Worldをwebotsで開く
  - world/lecture_mdd.wbt

ハンズオン:クラス/ステートマシン図

- 30cm前方に進んで止まるだけ

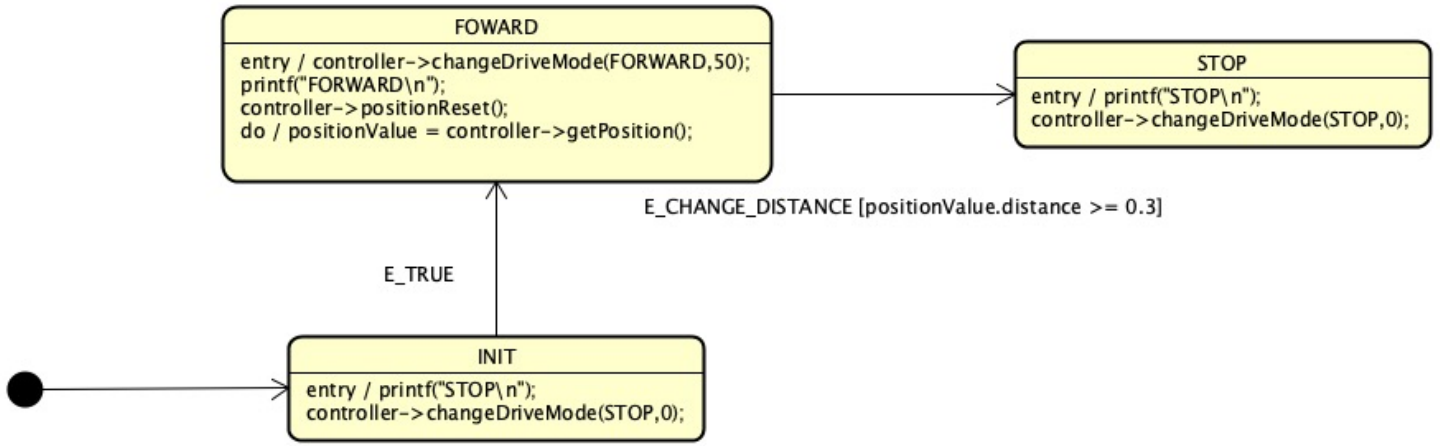
Controller* controller;



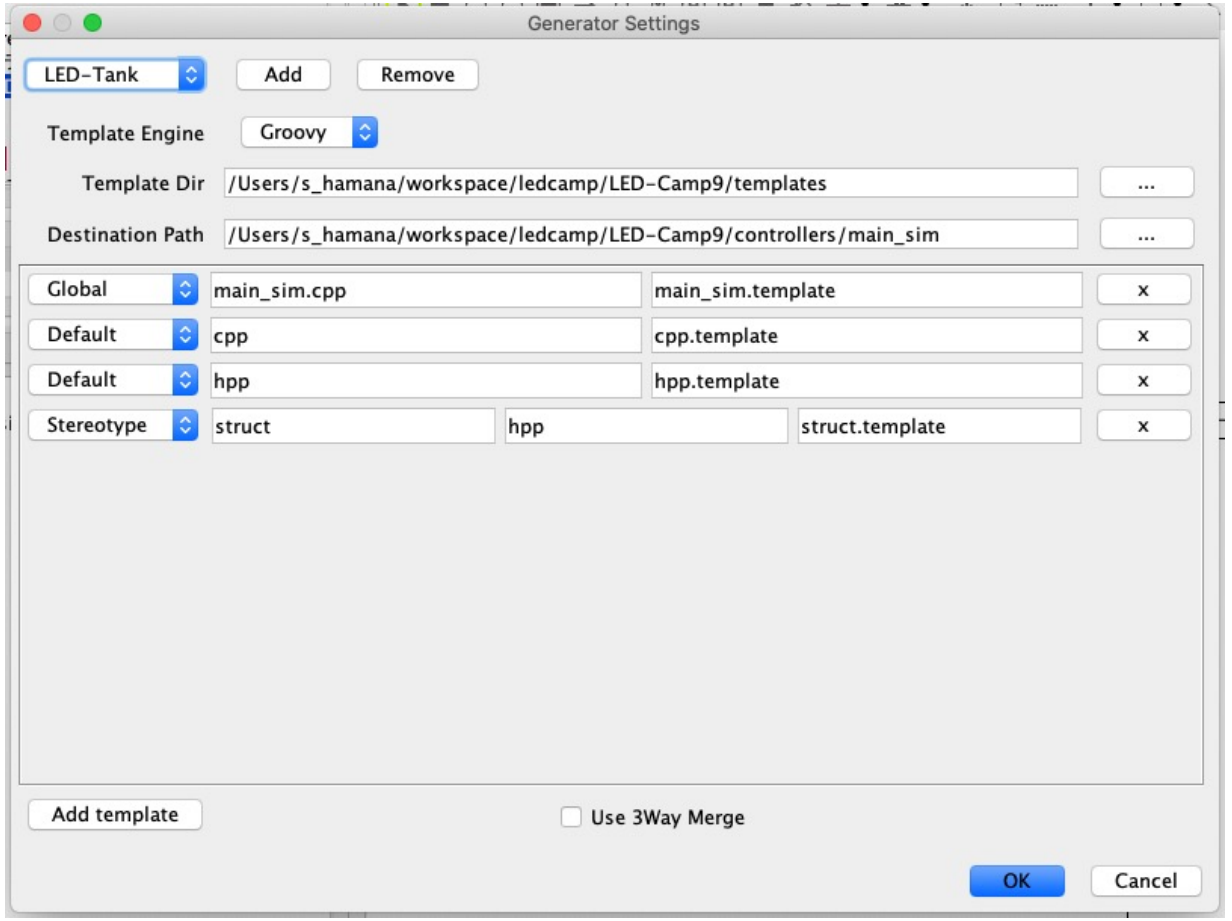


ハンズオン:ステートマシン図の説明

Controller::changeDriveMode()
　　前進/後退/左右旋回をする
Controller::getPosition()
　　進んだ距離/旋回した角度を取得する
Controller::positionReset()
　　進んだ距離/旋回した角度のカウントをリセットする

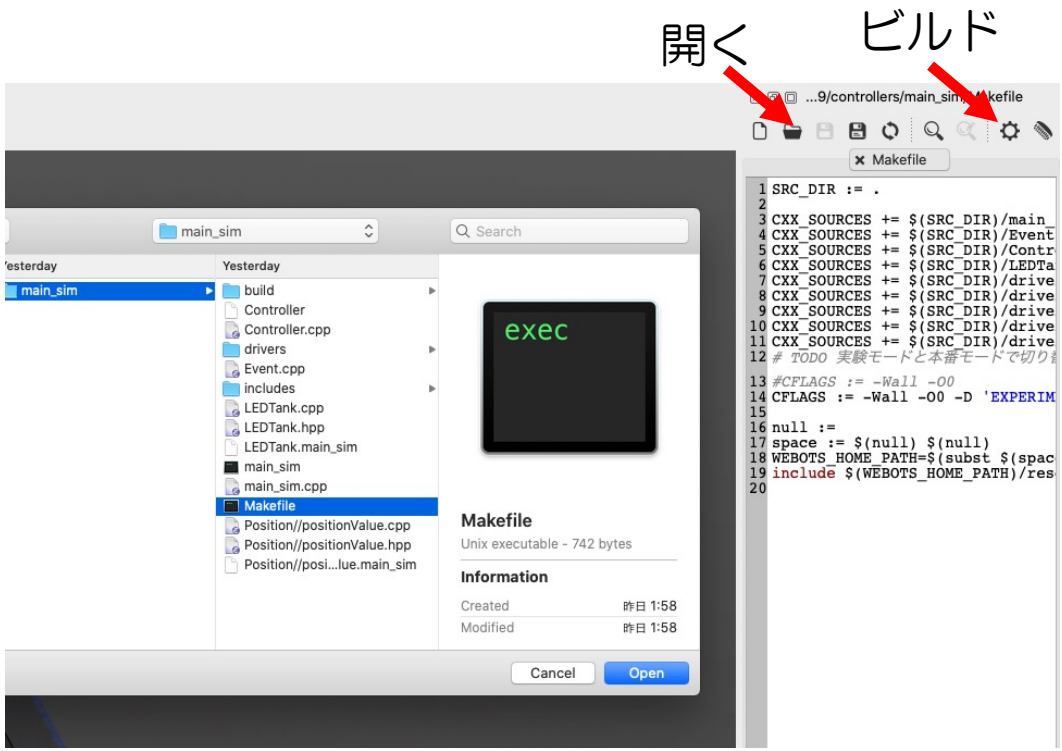


ハンズオン:コードの生成

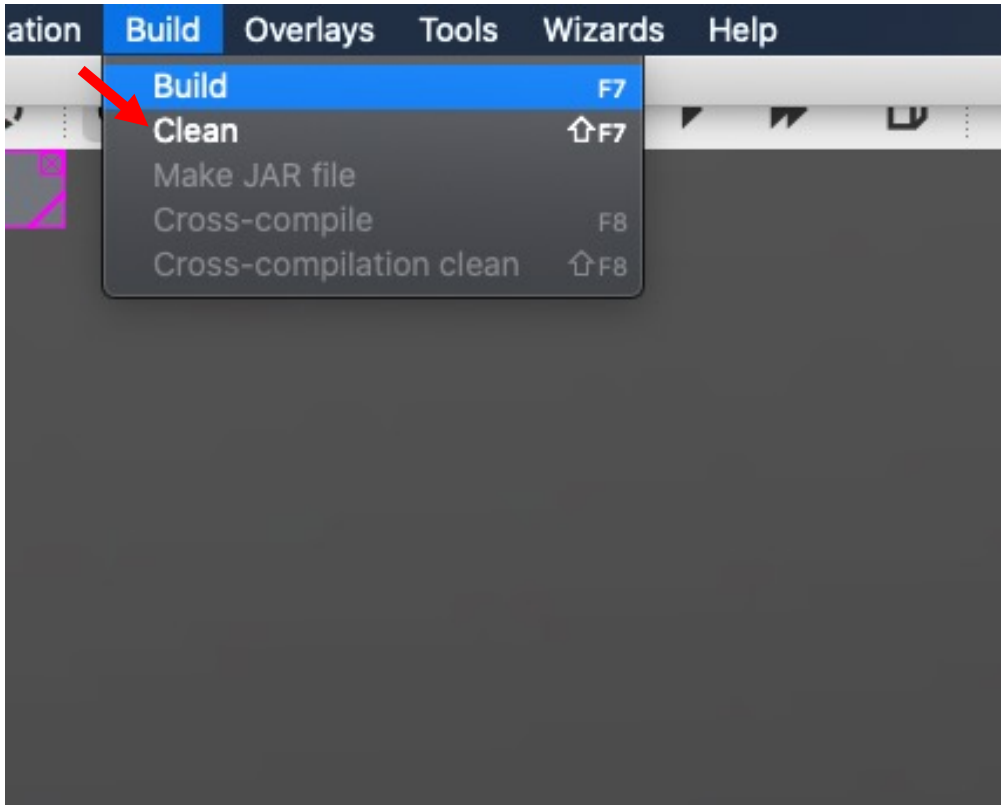


ハンズオン:コードのビルド

controller/main_sim/Makefile



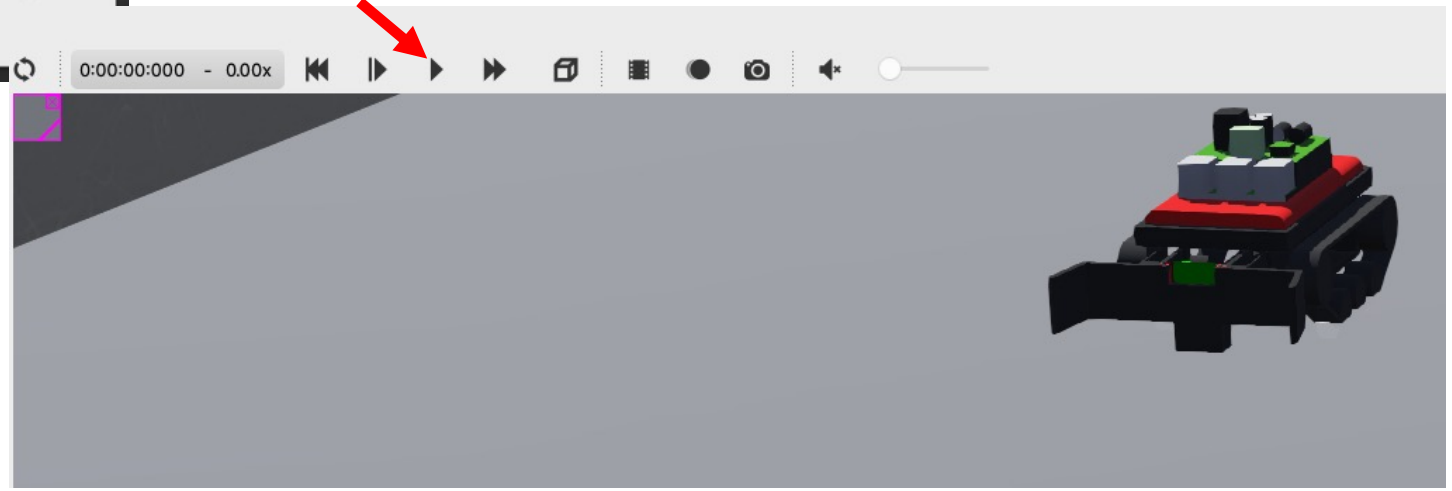
コードの修正がコンパイル時に反映されないとき
-> CleanしてからBuild



ハンズオン:シミュレーションの実行

ビルド後のダイアログ

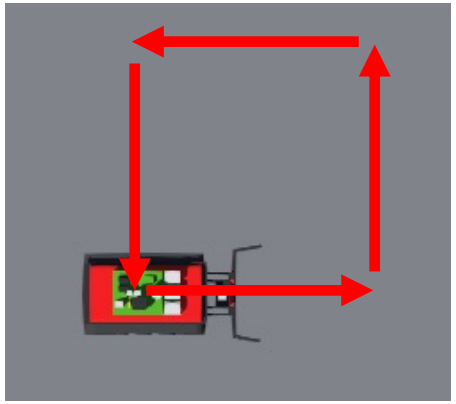
再生ボタンでシミュレート開始



演習: ステートマシン図を変える

- 以下の動きをするようにステートマシン図を変えてください
  - 30cm前方に進む
  - 90°左に旋回する

  を4回繰り返す
  - 停止する



使用するAPI/イベント

- API
  - Controller::changeDriveMode(CCW, [power: int (0-100)])
    - 前進/後退/左右旋回をする
  - Controller::getPosition()
    - 進んだ距離/旋回した角度を取得する
  - (任意)Controller::positionReset()
    - 進んだ距離/旋回した角度のカウントをリセットする
- イベント
  - E_CHANGE_DISTANCE
  - E_CHANGE_ANGLE
  - 複数ガード条件は&&, ||でつないでください(C++の条件文と同じ)

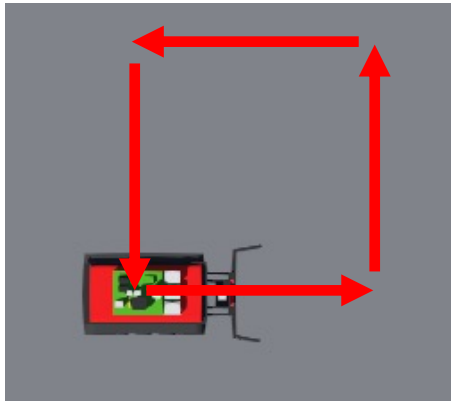
再掲)演習: ステートマシン図を変える

- 以下の動きをするようにステートマシン図を変えてください
  - 30cm前方に進む
  - 90°左に旋回する

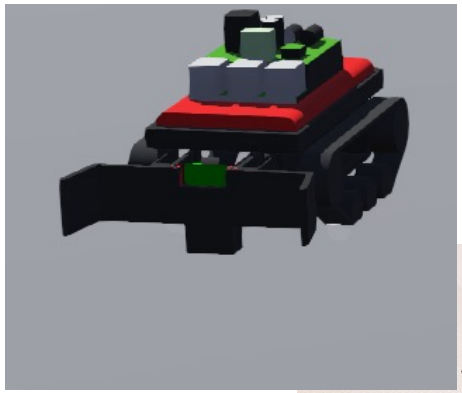
  を4回繰り返す
  - 停止する

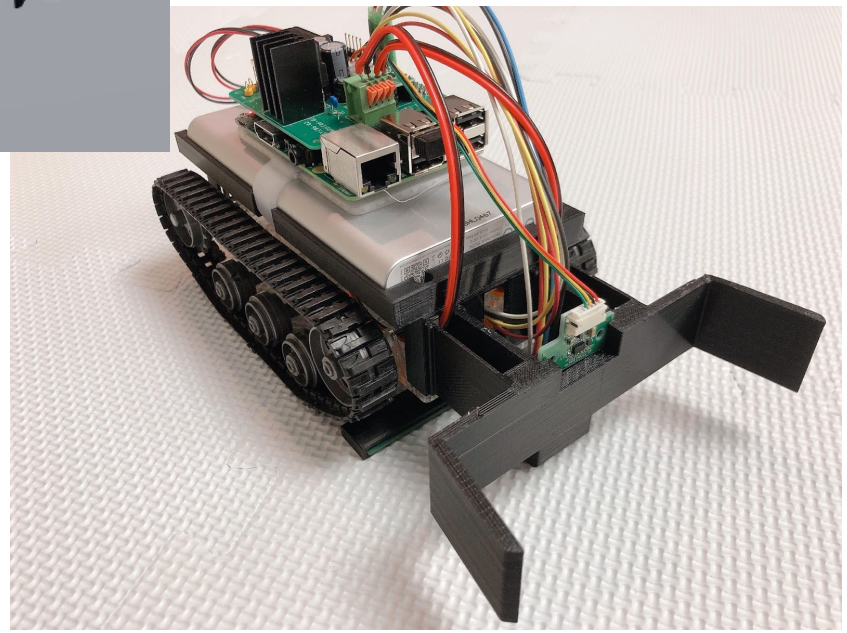
ブレイアウトルームで！

時間は25分とします

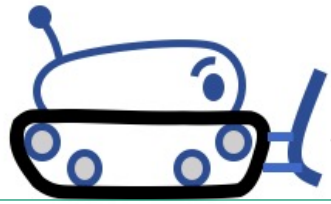


最後に





ちなみに！
このロボットカーの名前は「タンくん」
LED-Camp6, 7では右のように
物理ボディがありました！
コロナ禍によるオンライン開催
を受けてバーチャル転生しました。
可愛がってあげてください。



おわり