

OSC Nagoya 2024
2024/05/25

C言語やROSでLEGOを動かそう

樋山一樹
(南山大学 / TOPPERS)

目次

- 自己紹介
- SPIKE-RTの紹介
- SPIKEをROS 2で動かすための開発環境の紹介

自己紹介

樋山一樹 (ひやま いつき)

- 愛知県在住
- 南山大学 理工学研究科1年
- 学部3年時に研究室に配属されて以来, 組込みシステムを中心に学習中
 - 研究室HP : <https://honda-lab1.sakura.ne.jp>
- 研究室配属後にTOPPESの活動に参加

LEGO Education SPIKE Primeとは

- SPIKE

- LEGO社とMITが共同で開発
 - プログラミング教育キット
- HubとPUPデバイスを組み合わせてロボットを制作
- 公式ではScrachやPythonでのプログラミングをサポート

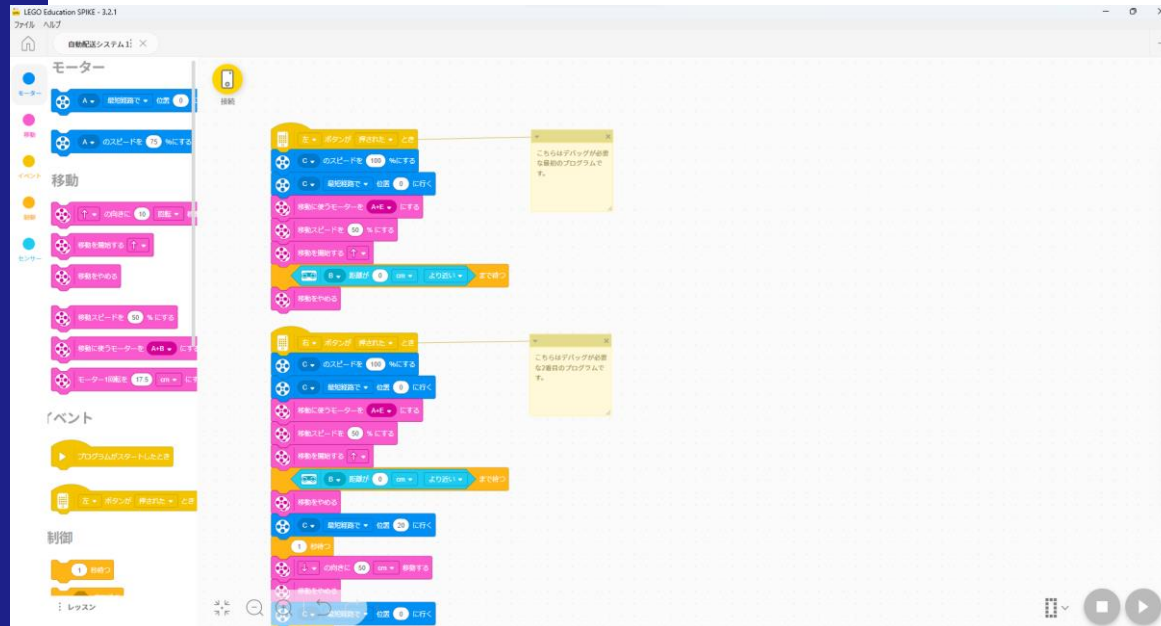
- Hub

- STM32F413 (Cortex-M4)

- EV3よりスペックダウン
- Linuxの実行は難しい



プログラミング可能なHub



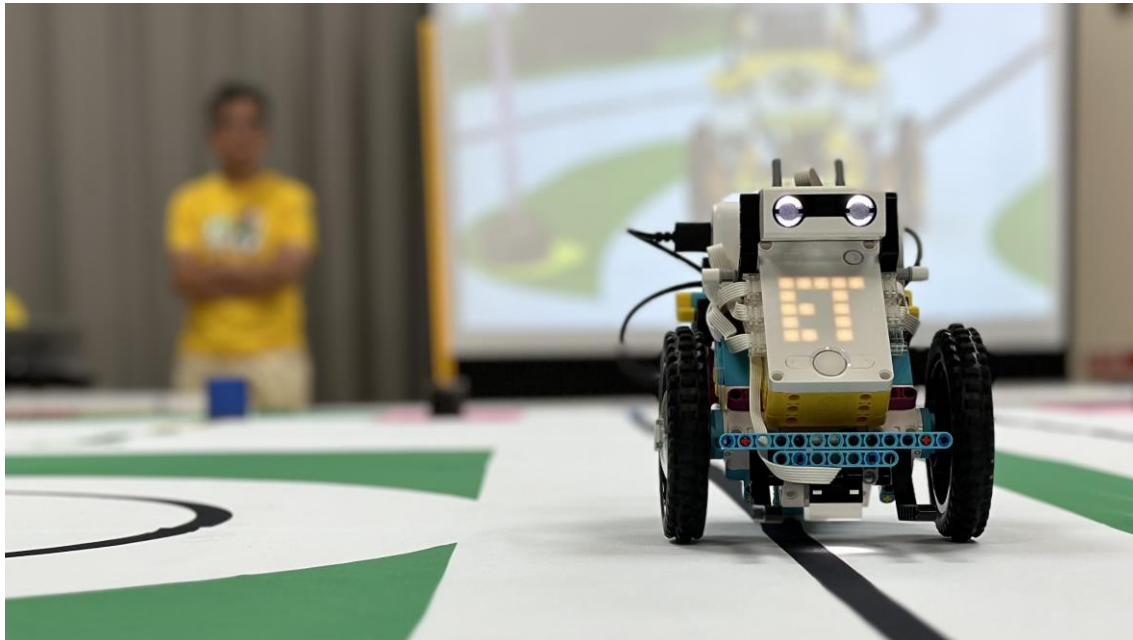
SPIKE Prime App



SPIKE Prime

SPIKEの活用

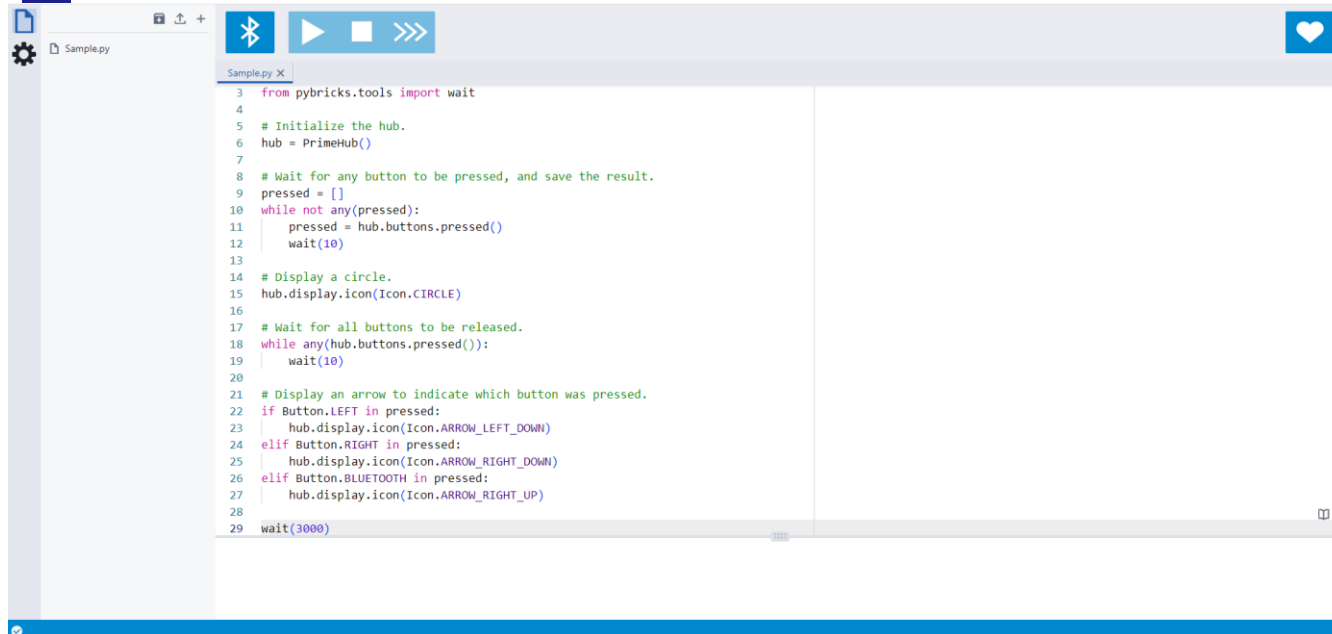
- 活用場面
 - 教育現場
 - ロボットコンテスト
 - ETロボコン
 - WRO (World Robot Olympiad)
 - など



ETロボコン(<https://www.etrobo.jp/>)

OSSコミュニティとSPIKE

- 様々なOSSコミュニティによりSPIKE向けSWプラットフォームが開発されている
 - Pybricks
 - ブラウザ上でPythonプログラミングが可能
 - SPIKE-RT
 - TOPPERSプロジェクト
 - この後紹介



```
3 from pybricks.tools import wait
4
5 # Initialize the hub.
6 hub = PrimeHub()
7
8 # Wait for any button to be pressed, and save the result.
9 pressed = []
10 while not any(pressed):
11     pressed = hub.buttons.pressed()
12     wait(10)
13
14 # Display a circle.
15 hub.display.icon(Icon.CIRCLE)
16
17 # Wait for all buttons to be released.
18 while any(hub.buttons.pressed()):
19     wait(10)
20
21 # Display an arrow to indicate which button was pressed.
22 if Button.LEFT in pressed:
23     hub.display.icon(Icon.ARROW_LEFT_DOWN)
24 elif Button.RIGHT in pressed:
25     hub.display.icon(Icon.ARROW_RIGHT_DOWN)
26 elif Button.BLUETOOTH in pressed:
27     hub.display.icon(Icon.ARROW_RIGHT_UP)
28
29 wait(3000)
```



SPIKE-RT

- SPIKE-RT

- SPIKEで利用することを目的として開発されたRTOS

- 軽量

- メモリ使用量が搭載量の2割以下

- C言語でのアプリケーション開発が可能

- マルチタスクプログラミング

- アプリケーションのリアルタイム性を確保

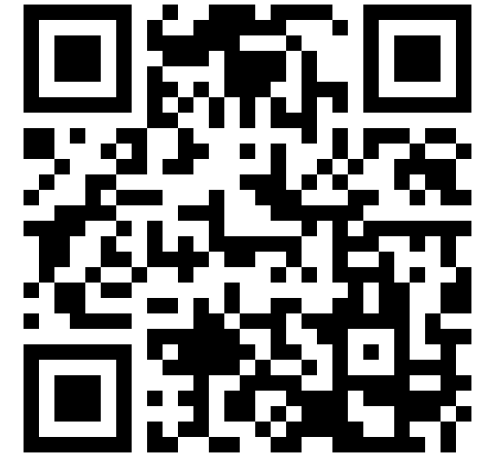
- TOPPERS/ASP3カーネルがベース

- ITRON系

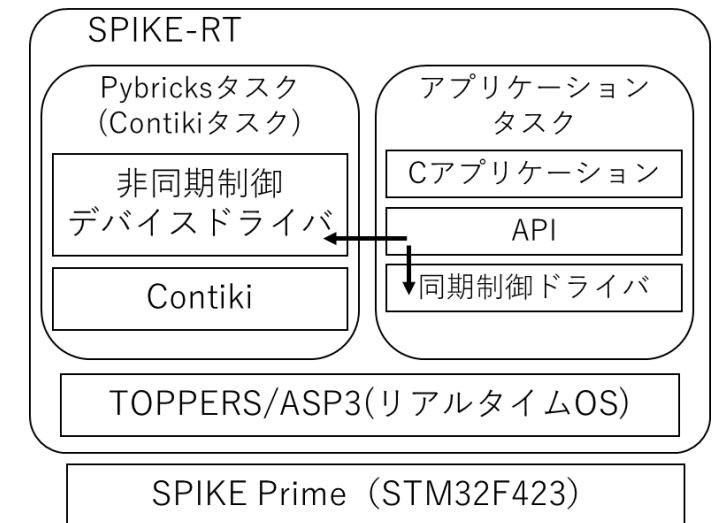


C言語

- 処理速度が高速
- HW制御
- 低レイヤの開発で多く活用
- 自動車を始めとした多くの組み込み機器で活用



SPIKE-RT



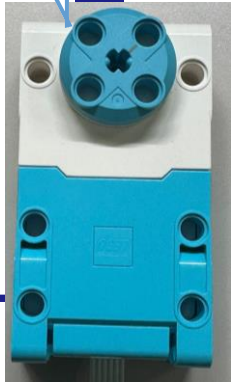
SPIKE-RTの内部構成

SPIKE-RTプログラミング

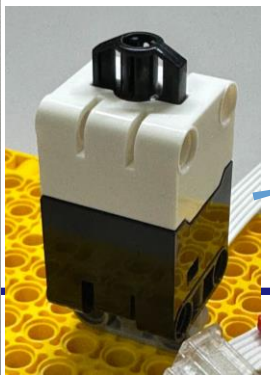
- APIリファレンス : <https://spike-rt.github.io/spike-rt/ja/html/modules.html>



モータ



フォースセンサ

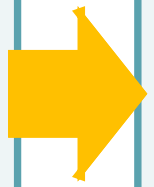


<フォースセンサを押す力に応じてモータを駆動する>

初期化处理

```
pbio_error_t err;  
pup_motor_t *motor;  
pup_device_t *force;  
  
dly_tsk(3000000);  
  
// Get pointer to device  
motor = pup_motor_get_device(PBIO_PORT_ID_A);  
force = pup_force_sensor_get_device(PBIO_PORT_ID_D);
```

デバイスへの
ポインタ



駆動処理

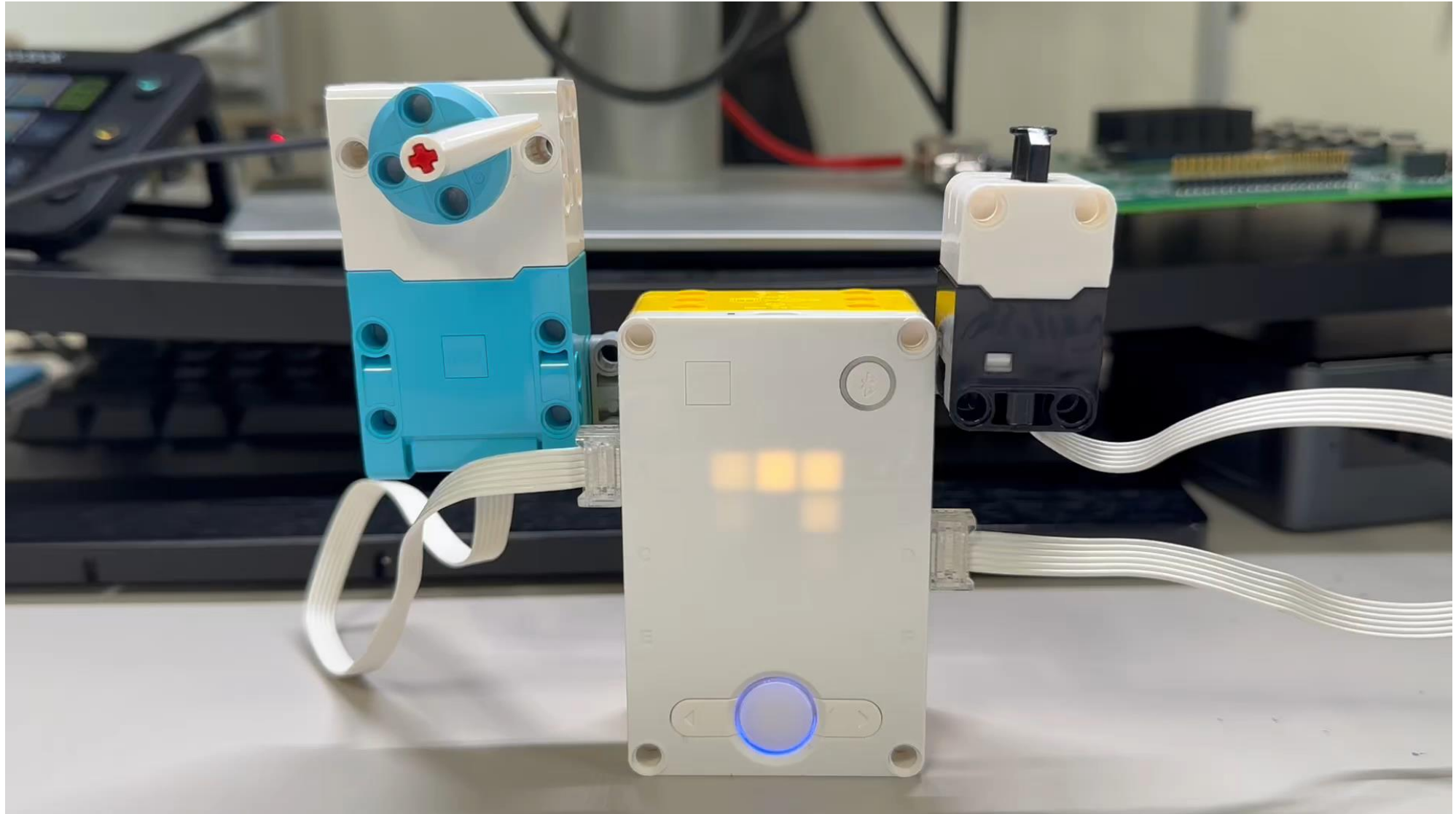
```
int force_val;  
  
while (1)  
{  
    force_val = pup_force_sensor_force(force);  
    pup_motor_set_speed(motor, force_val * 100);  
  
    dly_tsk(10000);  
}
```

力の大きさを取得

接続モータへの
ポインタ

指令値

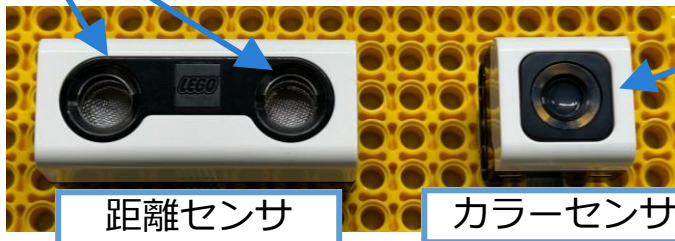
SPIKE-RTプログラミング



SPIKE-RTプログラミング

ライト

＜カラーセンサで取得した周囲の明るさに応じて距離センサのライトを点灯する＞



物体の色や周囲の明るさなどを検出

距離センサ

カラーセンサ

初期化処理

```
pup_device_t *col;  
pup_device_t *ult;  
  
dly_tsk(3000000);  
  
// Get pointer to device  
col = pup_color_sensor_get_device(PBIO_PORT_ID_A);  
ult = pup_ultrasonic_sensor_get_device(PBIO_PORT_ID_B);
```

デバイスへの
ポインタ

ポインタ取得

駆動処理

```
while (1)  
{  
    amb = pup_color_sensor_ambient(col);  
    hub_display_off();  
  
    if(amb > 10 && amb <= 40)  
        pup_ultrasonic_sensor_light_set(ult, 0, 30, 0, 30);  
    else if(amb <= 10)  
        pup_ultrasonic_sensor_light_set(ult, 30, 30, 30, 30);  
    else  
        pup_ultrasonic_sensor_light_off(ult);  
  
    dly_tsk(100000);  
}
```

明るさを取得

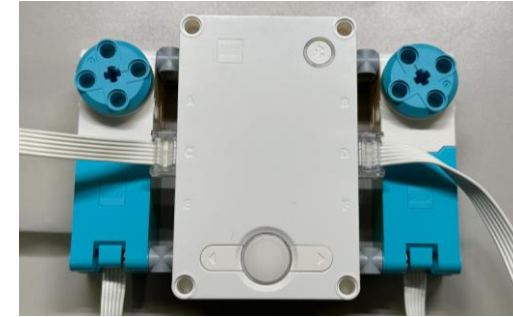
距離センサ
ライト点灯

SPIKE-RTプログラミング



マルチタスクプログラミング

- 教材 : NCES Education Program / 「組み込みソフトウェア開発技術の基礎」 など
 - <https://www.nces.i.nagoya-u.ac.jp/NEP/materials/about.html>
- データキューを使用してタスク間通信を行う
 - 本体ボタンの押下状態に応じてデータをデータキューに送信 (but_cyc_handler)
 - 受信側は受信データに応じてモータを駆動する (motor_task)
- 周期ハンドラを使用してボタンの押下状態を確認する



生成時にタスクを起動

コンフィギュレーションファイル (.cfg)

```
CRE_TSK(MAIN_TASK, { TA_ACT, 0, main_task, MAIN_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(MOTOR_TASK, { TA_NULL, 0, motor_task, MOTOR_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(BUT_CYC_HANDLER, { TA_NULL, 0, but_cyc_handler, BUTTON_CYC_PRIORITY, STACK_SIZE, NULL });

CRE_CYC(BUT_CYC, {TA_NULL, {TNFY_ACTTSK, BUT_CYC_HANDLER}, 1000, 0});

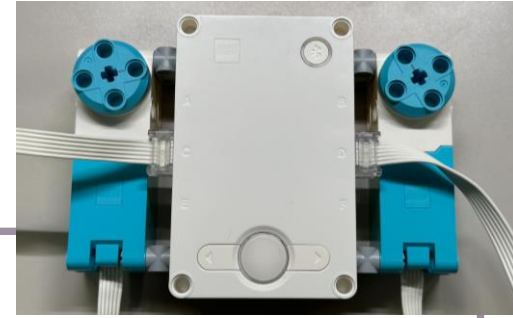
CRE_DTQ(MOTOR_DTQ, {TA_NULL, 10, NULL}); // NULL -> データキューの管理領域をカーネル等が確保
```

タスク生成

周期ハンドラ生成

データキュー

マルチタスクプログラミング (タスク間通信)



送信データ型 (構造体)

```
struct data_packet{
  pup_motor_t *motor;
  int speed;
  int idx;
  bool is_setup;
};
```

main_task()

```
void
main_task(intptr_t exinf)
{
  act_tsk(MOTOR_TASK);
  sta_cyc(BUT_CYC);

  <省略>
  while (1)
  {
    slp_tsk();
  }
}
```

起動

タスク・周期
ハンドラ起動

自身は休
止状態に

データ送信

but_cyc_handler()

```
void
but_cyc_handler(intptr_t exinf) //1ms周期
{
  struct data_packet send_pkt;
  static hub_button_t pressed_ptn, pre_ptn;
  intptr_t send_data;

  hub_button_is_pressed(&pressed_ptn);

  if ((pressed_ptn & HUB_BUTTON_LEFT)
      && pre_ptn == 0) {
    create_pkt(&send_pkt, MOTOR_A);
    send_data = &send_pkt;

    snd_dtq(MOTOR_DTQ, send_data);
  }
  else if ((pressed_ptn & HUB_BUTTON_RIGHT)
           && pre_ptn == 0){
    create_pkt(&send_pkt, MOTOR_B);
    send_data = &send_pkt;

    snd_dtq(MOTOR_DTQ, send_data);
  }
  pre_ptn = pressed_ptn;
}
```

データ(構
造体)生成

Queue

データ(構
造体)生成

motor_task()

```
void
motor_task(intptr_t exinf)
{
  intptr_t rev_datta_pkt;
  struct data_packet *receive_pkt;
  pbio_error_t m_err;

  while (1) {
    rcv_dtq(MOTOR_DTQ, &rev_datta_pkt);
    receive_pkt = rev_datta_pkt;

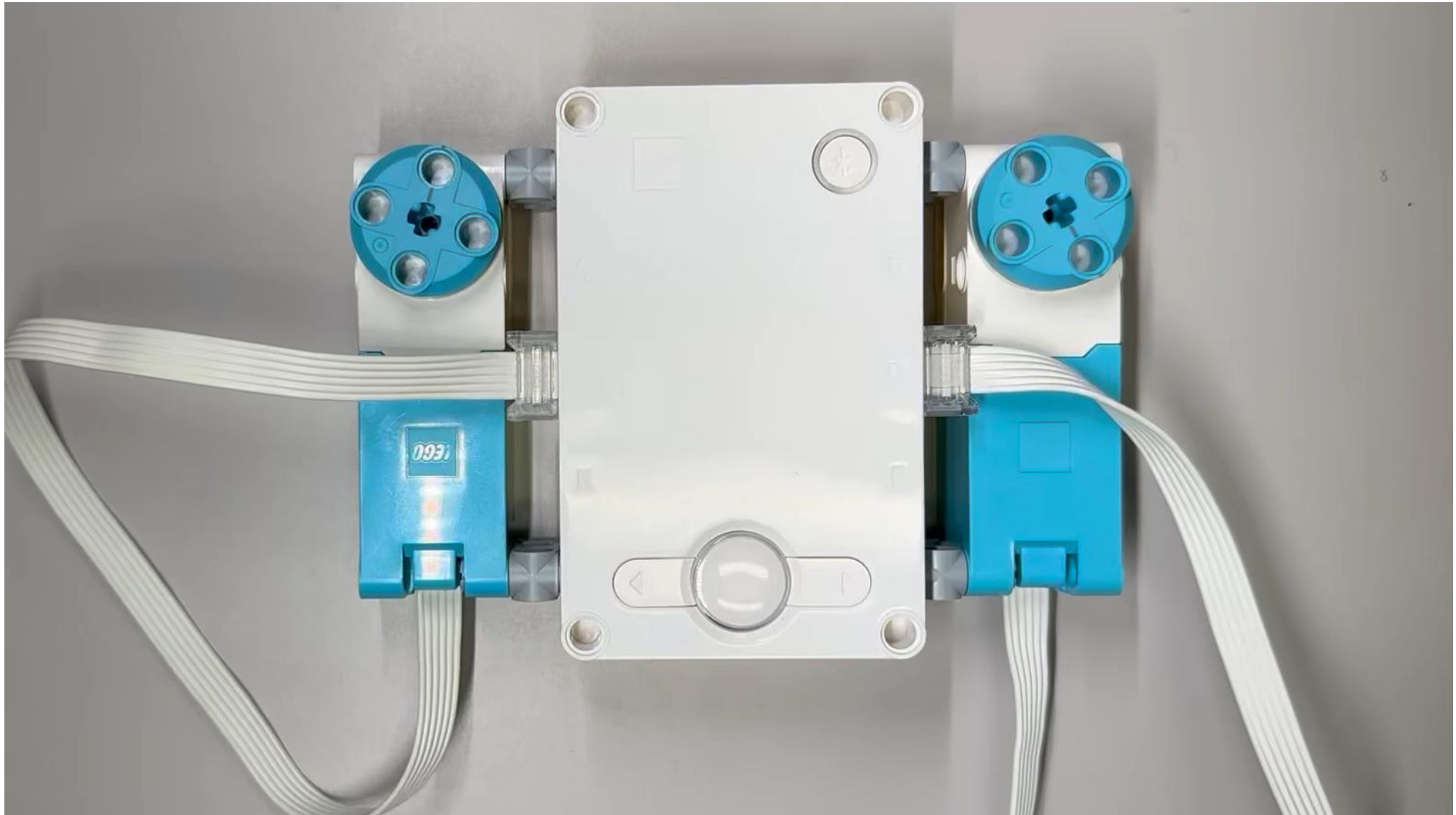
    if (!receive_pkt->is_setup)
      <省略(モータセットアップ処理)>

    if ((receive_pkt->idx % 2) == 0)
      pup_motor_set_speed(receive_pkt->motor,
                          receive_pkt->speed);
    else
      pup_motor_brake(receive_pkt->motor);
  }
}
```

データ受信

モータ出力

マルチタスクプログラミング (タスク間通信)



SPIKE-RTについてのまとめ

- このような人におすすめ
 - C言語の学習をしたい
 - 組み込みシステムに興味がある
 - RTOS(マルチタスク)を学習したい
 - SPIKEを使用したロボットコンテストに出場する



SPIKE-RTのサンプルを公開中

目次

- 自己紹介
- SPIKE-RTの紹介
- SPIKEをROS 2で動かすための開発環境の紹介

ROS 2

- ROS 2とは

- Robot Operating Systemの略
- ロボットや自動運转向けの分散処理フレームワーク
 - 通信ミドルウェア
 - publish/subscribe通信が可能
- UNIX系OS上での稼働を想定

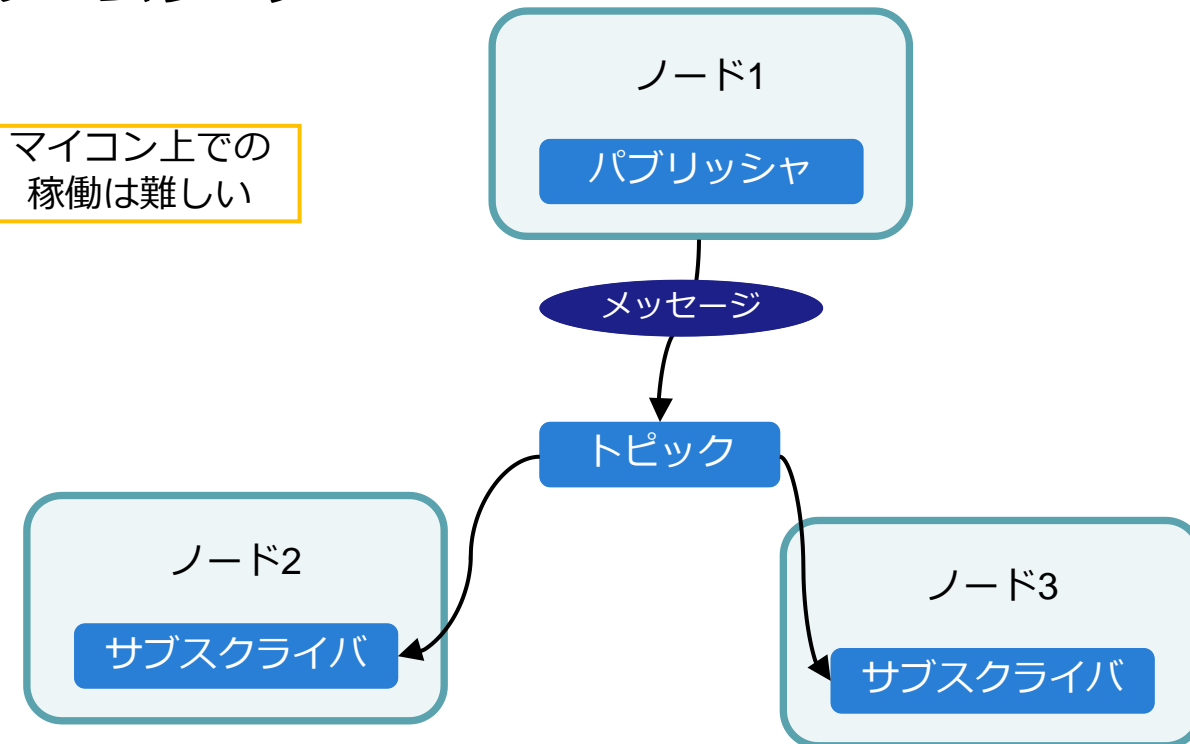
マイコン上での稼働は難しい

- 活用例

- Amazon Roboticsの物流補助ロボット
- aibo (SONY)

- SPIKE Prime Hubもマイコン
- ROS2の稼働は難しい

ROS 2™

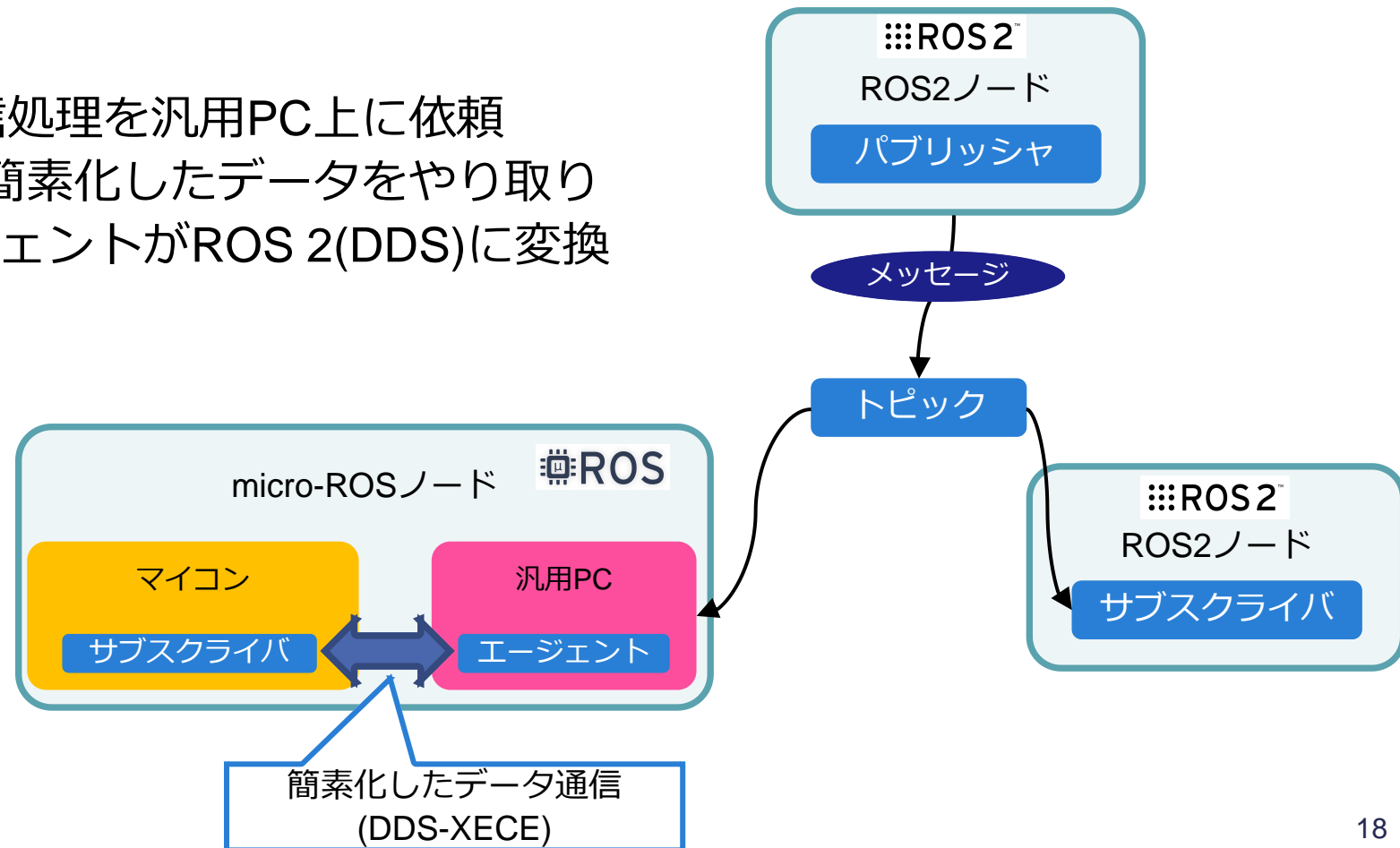


micro-ROS



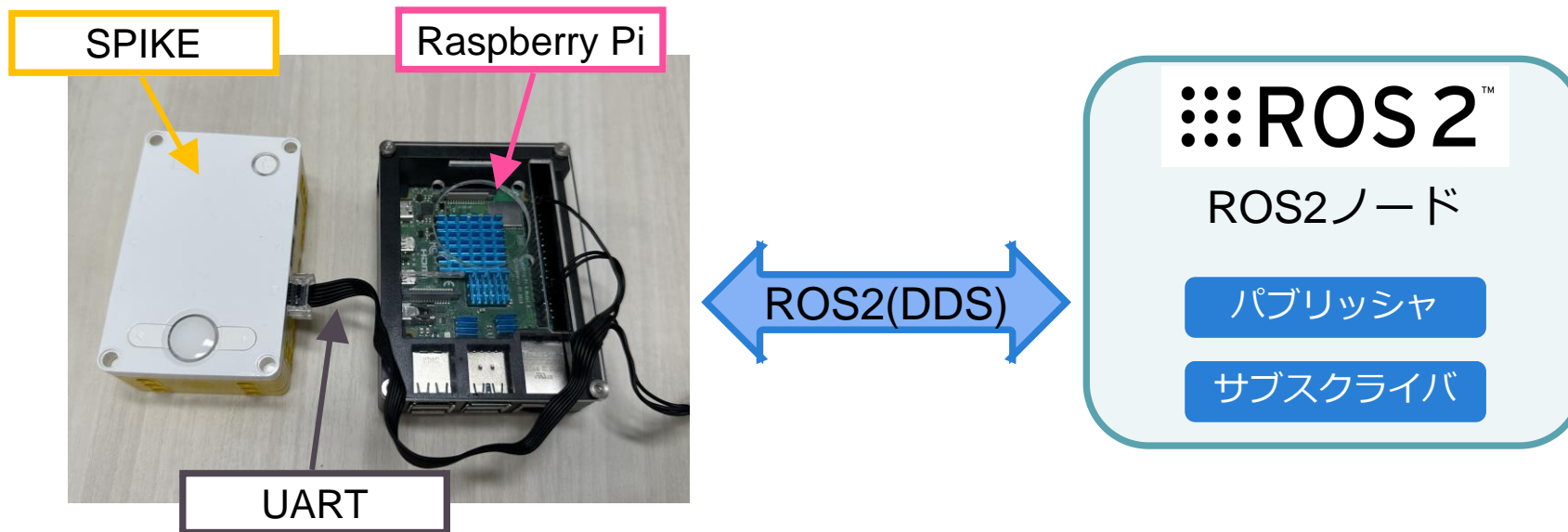
- ROS 2との関係性
 - ROS 2のマイコン上での稼働は厳しい
 - マイコンをROS2に接続する為の機構
- micro-ROSの構成
 - 処理性能を要する通信処理を汎用PC上に依頼
 - マイコン↔汎用PCは簡素化したデータをやり取り
 - 汎用PC上のエージェントがROS 2(DDS)に変換

micro-ROSであればSPIKE上で稼働できる！！



TOPPERSのROSへの取り組み

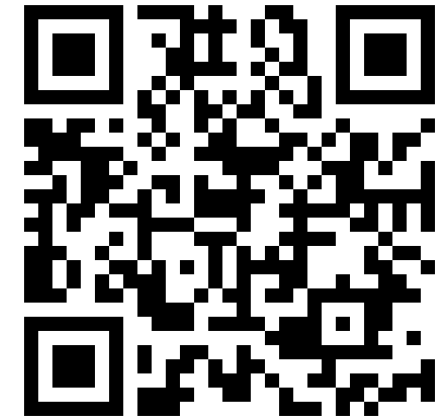
- micro-ROS_ASP3
 - TOPPERS/ASP3カーネル上で動作するmicro-ROSミドルウェア
 - LEGO SPIKE (SPIKE-RT) 上での利用もサポート
 - SPIKE上でmicro-ROSプログラミングが可能
 - SPIKEをROS 2に接続する事が可能
 - ROSの教材として活用可能



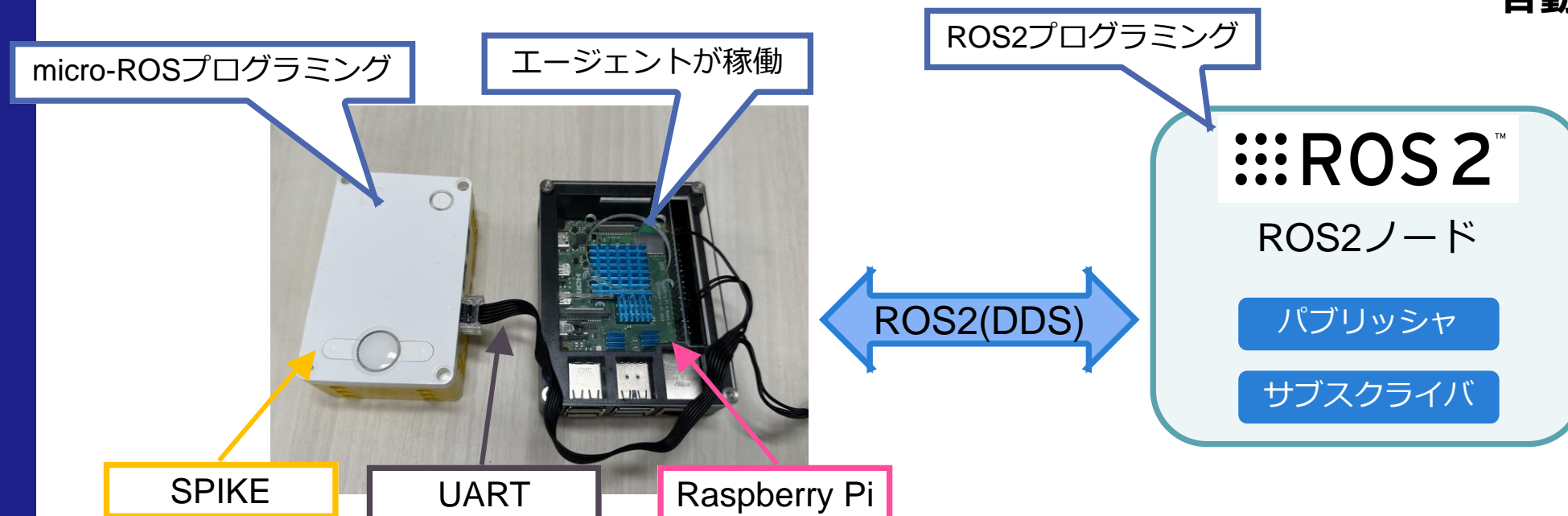
micro-ROSファームウェアの自動生成ツール

- 自動生成ツール

- micro-ROSプログラミングをせずにLEGOをROS 2で動かす
- ユーザはHub側の構成を設定ファイルに記述
- micro-ROSファームウェアを自動生成



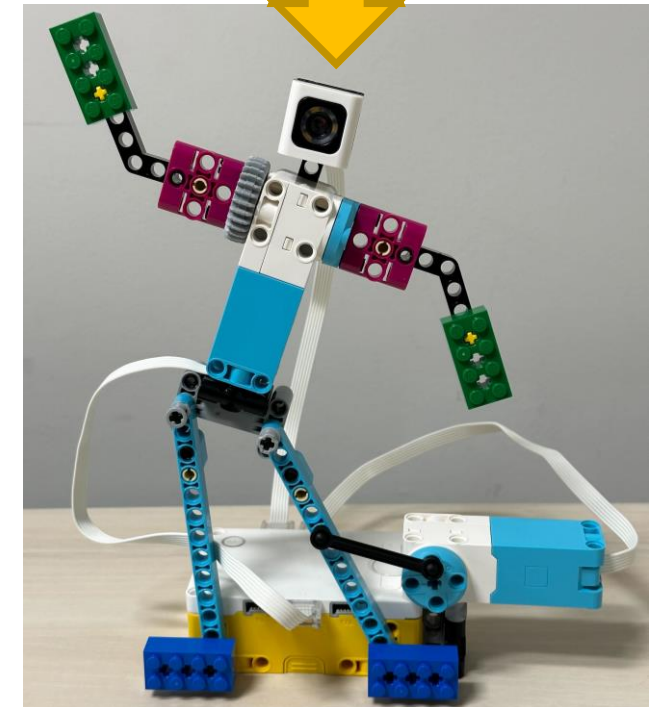
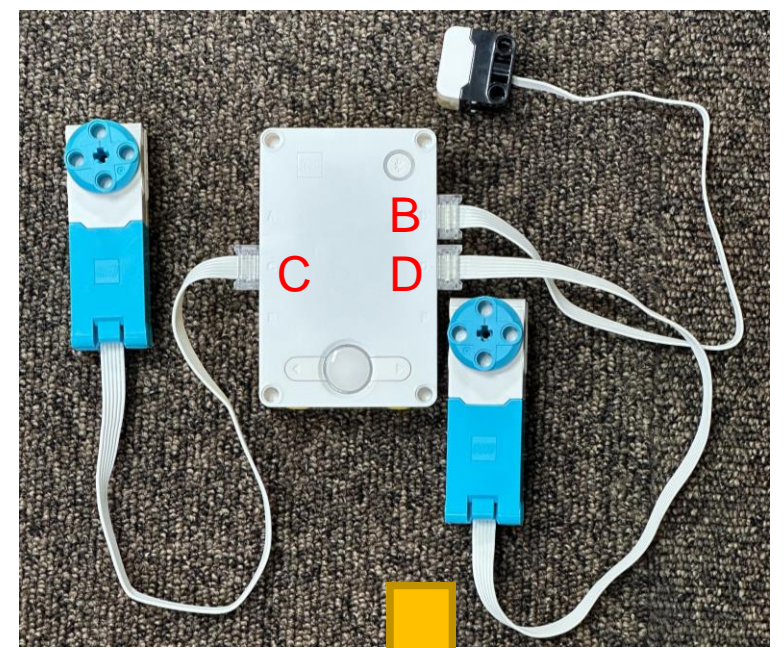
自動生成ツール



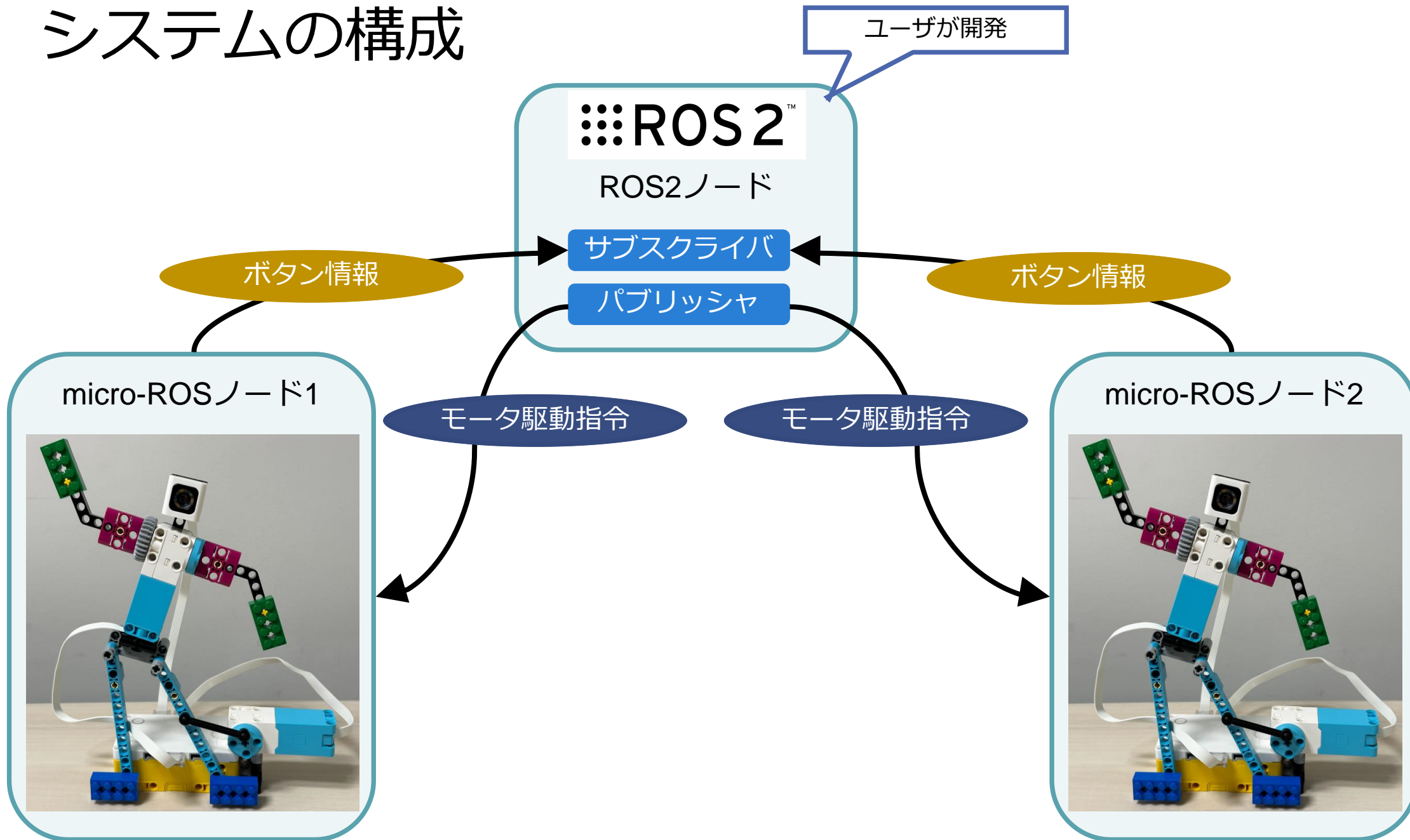
サンプル「ブレイクダンサー」

Yaml設定ファイル

```
PortB:  
  device: color-sensor  
  qos: best-effort  
  enable_lights: True  
  light_qos: best-effort  
  
PortC:  
  device: motor  
  qos: best-effort  
  wise: counter-clock  
  run_mode: set-speed  
  
PortD:  
  device: motor  
  qos: best-effort  
  wise: clock  
  run_mode: set-speed  
  
hub:  
  hub_program_cycle: 10  
  enable_imu: False  
  enable_battery_management: False  
  enable_button: True  
  enable_speaker: False  
  opening: True
```

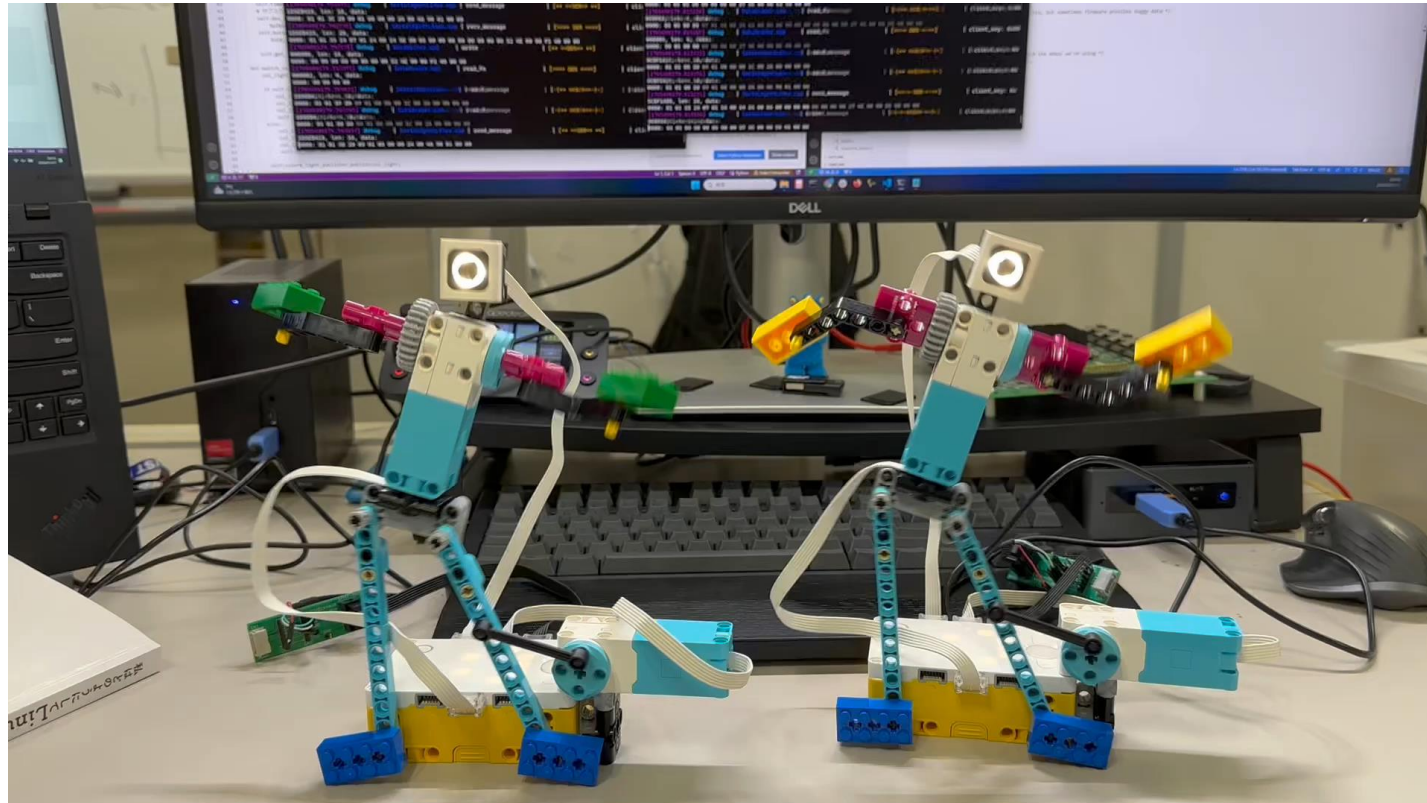


システムの構成



実行の様子

- TOPPES展示ブースにて実行中！！
- 3F : マーケットプラザ



まとめ

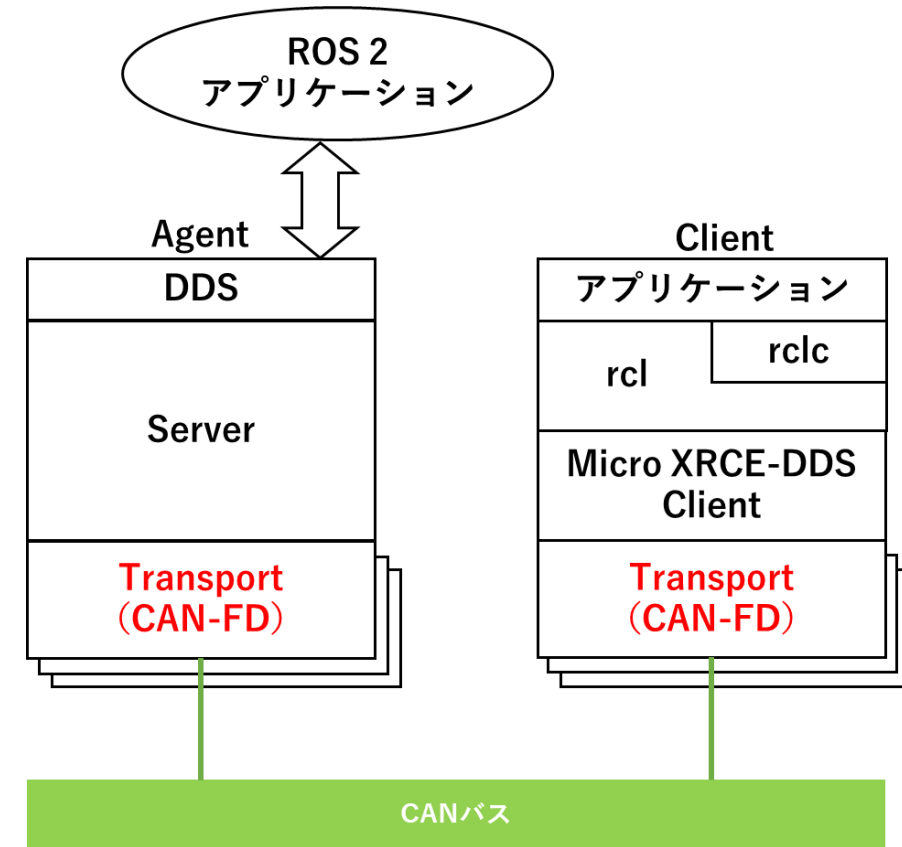
- SPIKE-RTはこのような人におすすめ
 - C言語の学習をしたい
 - 組み込みシステムに興味がある
 - SPIKEを使用したロボットコンテストに出場する
- micro-ROS_ASP3
 - SPIKEでmicro-ROSプログラミングが可能
 - SPIKEをROS 2の世界に接続可能
 - 各種ツールを使用してROS 2教材として活用可能

組み込みシステム向け軽量ROS環境 micro-ROSの産業ネットワーク (CAN-FD) 対応

南山大学 竹内 結斗

micro-ROS, CAN-FD

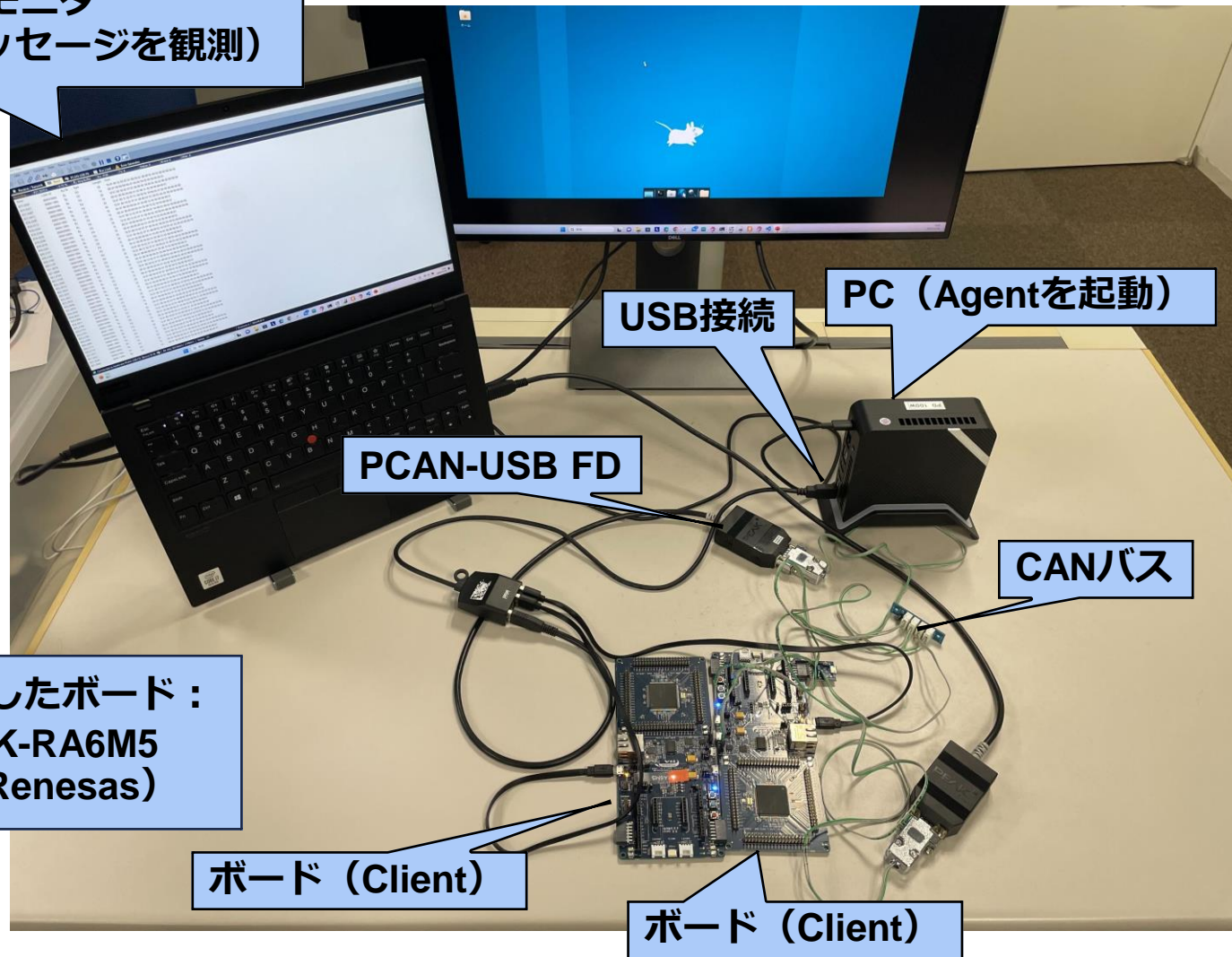
- micro-ROS
 - 小規模な組み込みシステムなどでROS 2を使用可能にする
 - オープンソースとして公開されている
 - 通信を実現するトランスポート層が独立している
 - 組み込みシステムは機器間の通信に様々な方法が使われる
 - Serial, Ethernet (TCP/UDP), CAN-FDなどのサポートがある
- CAN-FD
 - 主に車載やFAのネットワークとして使われている
 - メッセージ固有のCAN-IDによる優先度制御が可能



トランスポート層にCAN-FDを用いた構成図

機器の接続

CANモニタ
(メッセージを観測)



PCAN-USB FD

USB接続

PC (Agentを起動)

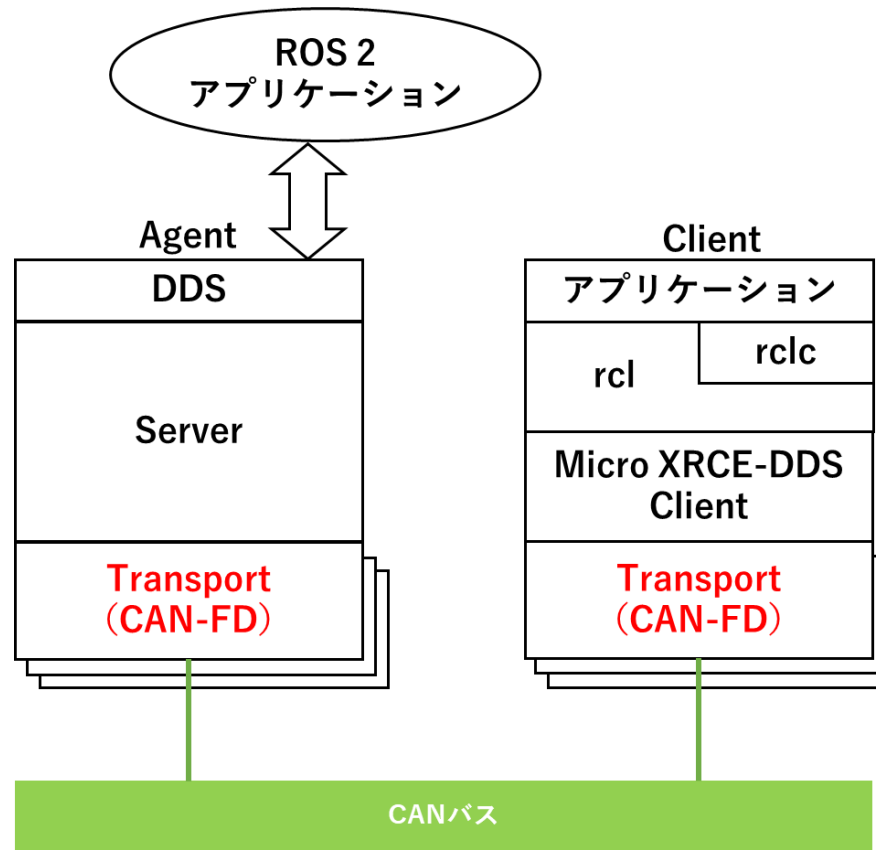
CANバス

使用したボード：
EK-RA6M5
(Renesas)

ボード (Client)

ボード (Client)

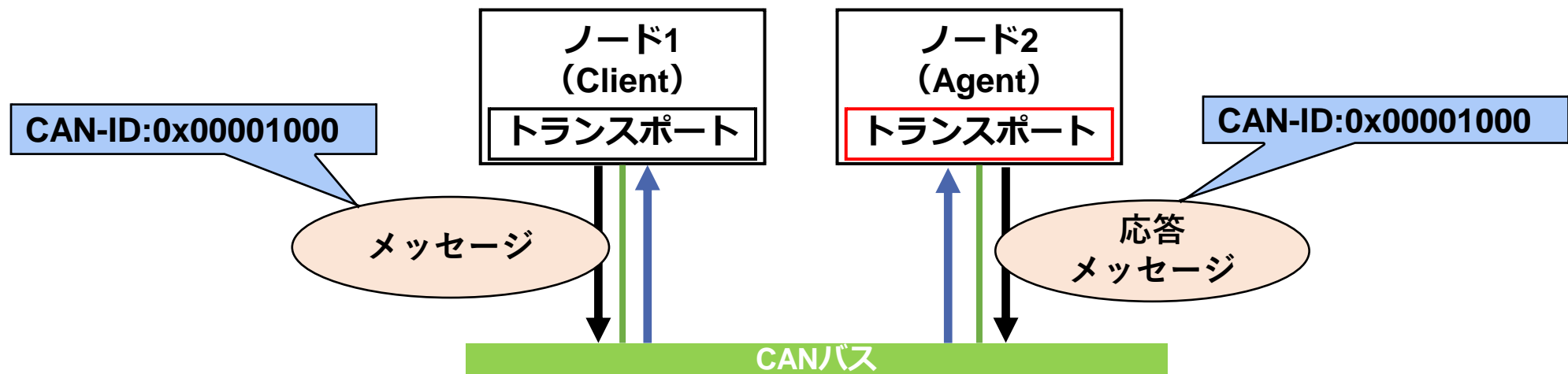
使用する機器の接続



ソフトウェア構成図

CAN-FD トランスポートの問題点

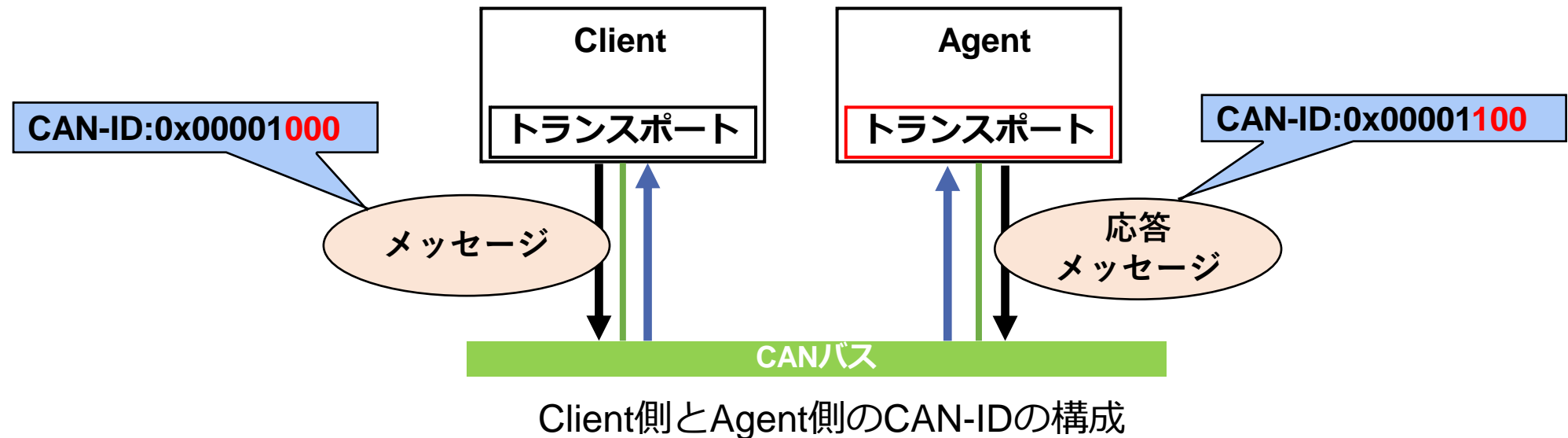
- 問題点：ClientとAgentが送信するメッセージのCAN-IDが同じである
 - ClientがAgentへメッセージ送信後，AgentはClientへ応答メッセージを送信する
 - Agentが送信する応答メッセージはClientが送信するメッセージのCAN-IDを使用する
 - **異なるノードからのメッセージが同一CAN-IDを使用することはCAN-FDでは違反**
 - 異なるノードから同一CAN-IDのメッセージが同時に送信されると，両方のメッセージがCANバス上に流れ，メッセージがミックスされることにより，正しいメッセージが受信されない
 - ClientとAgentの双方がpublishする場合にこの問題が発生する可能性がある



Client側とAgent側で同一CAN-IDを使用するイメージ図

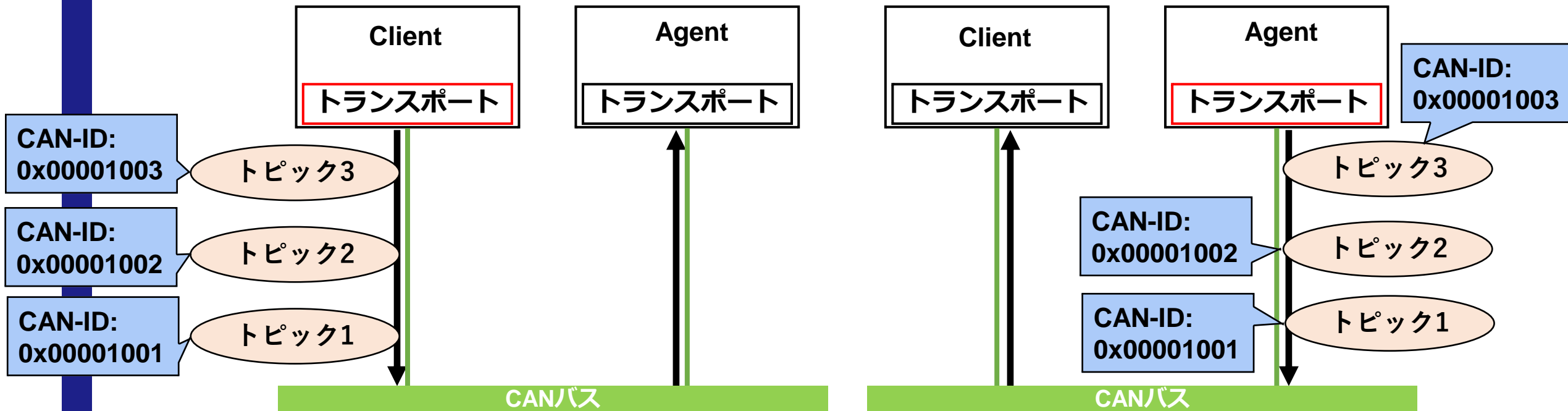
CAN-FD トランスポートの問題点の解決方法

- AgentがClientに送信する応答メッセージには, そのClientが送信するメッセージのCAN-IDの下位ビットに0x100を付与する仕組みを追加
 - AgentのCAN-FDトランスポートに追加
- AgentがClientに送信する応答メッセージのCAN-IDの上位ビットはClientの送信先のAgentかを判断するために使用する



CAN-FD トランスポートの課題と改善

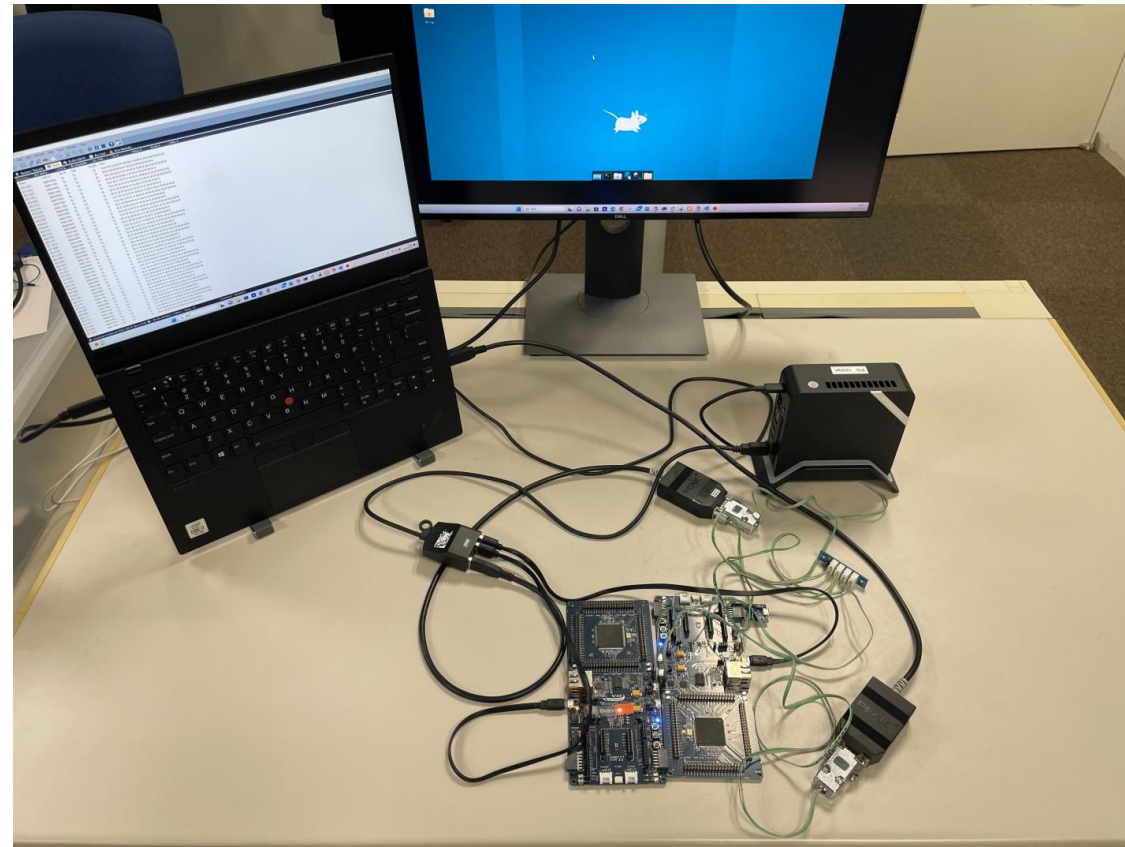
- 課題：すべてのトピックでCAN-IDが同じである
 - すべてのトピックに対して同じCAN-IDを使用しているため、CAN-IDによる優先度制御に対応できない
 - **トピックの種類ごとに、異なるCAN-IDを割り当て、優先度制御に対応させた**
 - Client, AgentそれぞれのCAN-FD トランスポートに追加



トピック毎にCAN-IDを設定したときのイメージ図
(左図：Clientはpublisher, 右図：Clientはsubscriber)

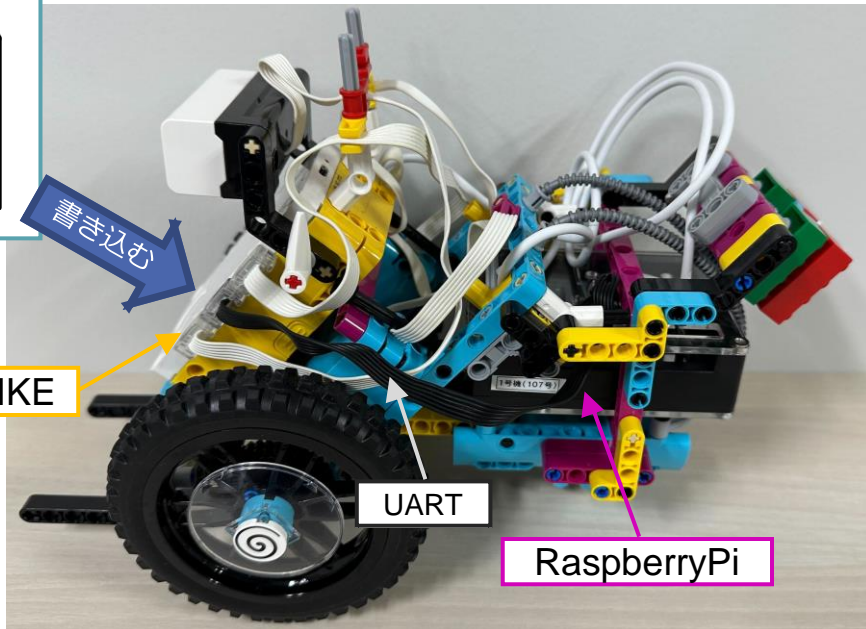
まとめ

- 現状のmicro-ROSの通信方法としてCAN-FDを使用する際に、以下の事を実施
 - ClientとAgentが送信するメッセージのCAN-IDが同じであるという問題点を解決した
 - トピック毎に異なるCAN-IDを割り当てる機能をCAN-FD トランスポートに追加した

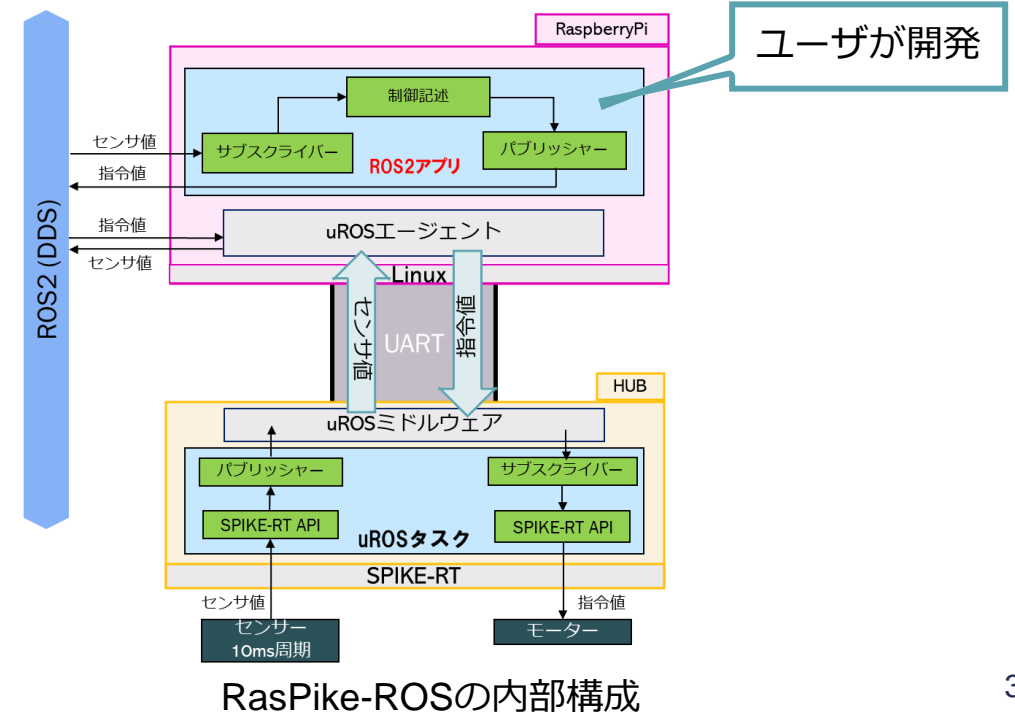


付録：ETロボコン走行体をROS 2で動かす

- ETロボコン走行体
 - SPIKEとRaspberryPiを使用
- RasPike-ROS
 - ETロボコン走行体向けソフトウェアプラットフォーム
 - ROS2とmicro-ROSを使用
 - SPIKEにmicro-ROSファームウェアを書き込む
 - ユーザはRaspberryPi上でROS2アプリケーションの開発を行う



ETロボコン走行体



付録：ETロボコン走行体をROS 2で動かす

- ライントレースプログラム

ROS 2™

Python

```
def line_trace_on_tick(self):
    self.steering_amount_calculation()
    self.motor_drive_control()

def timer_on_tick(self):
    # メッセージの生成
    motor_speed = MotorSpeedMessage()
    color_mode = Int8()
    color_mode.data = 3
    motor_speed.right_motor_speed = self.send_right_speed
    motor_speed.left_motor_speed = self.send_left_speed

# メッセージのパブリッシュ (送信)
self.motor_speed_publisher.publish(motor_speed)
self.color_mode_publisher.publish(color_mode)

def main(args=None):
    # ROS通信の初期化
    rclpy.init(args=args)
    # ノードの生成
    node = linetracerNode()

    rclpy.spin(node) # ROS2アプリ始動
    node.destroy_node()
    rclpy.shutdown()
```

タイマーコール
バック(周期呼
び出し)

ROS2 API
(パブリッシュ)

Python

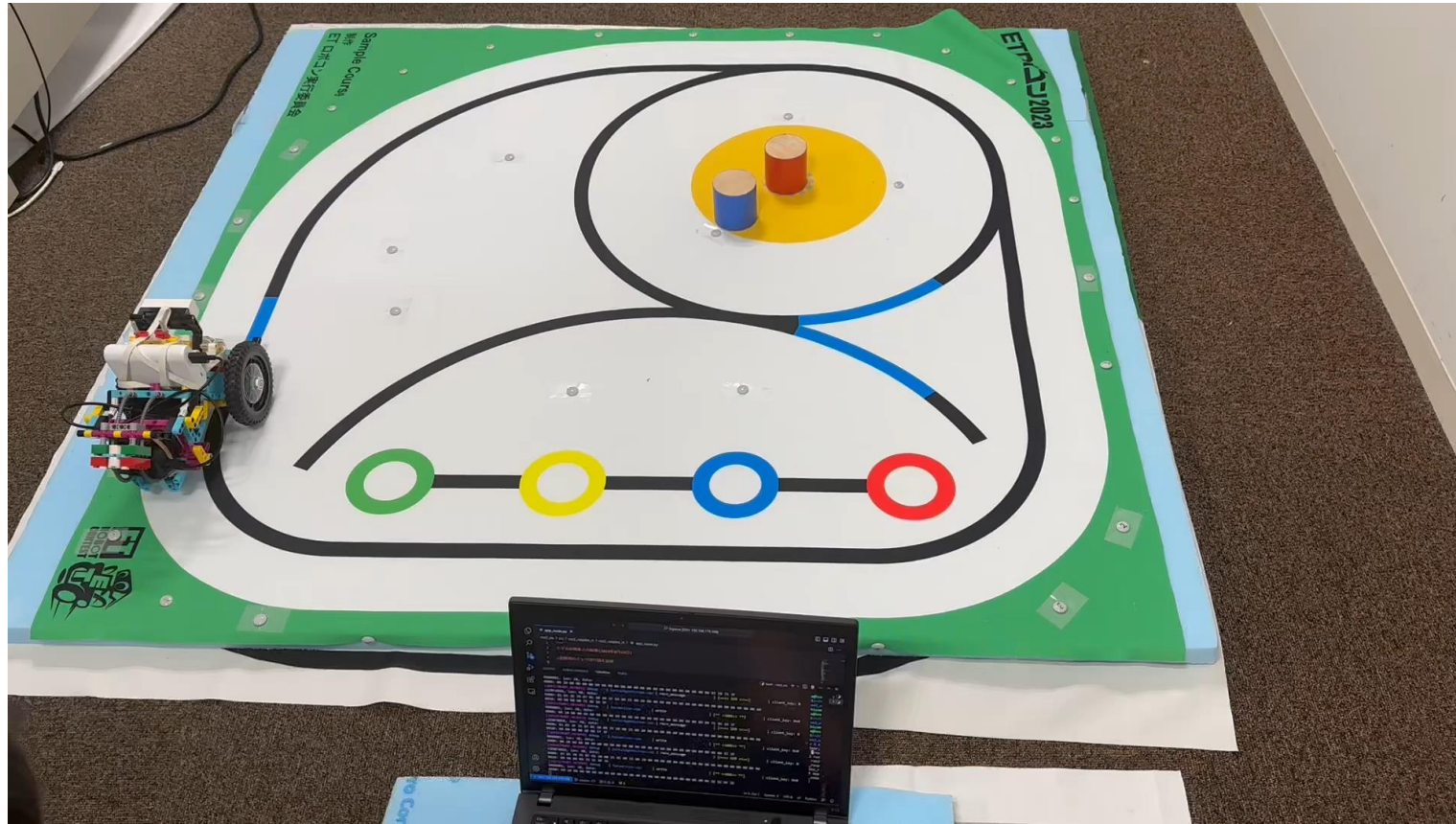
```
# ライントレース
def steering_amount_calculation(self):
    target_brightness = (white_brightness - brack_brightness) / 2
    diff_brightness = target_brightness - self.rev_color_sensor_refrection
    self.diff[1] = self.diff[0]
    self.diff[0] = int(diff_brightness)

    p_val = diff_brightness
    i_val = self.pre_i_val + (self.diff[0] + self.diff[1]) * delta_t / 2
    d_val = int((self.diff[0] - self.diff[1]) / 1)

self.steering_amount = (kp * p_val) + (kd * d_val) + (ki * i_val)

def motor_drive_control(self):
    self.send_left_speed = bace_speed + (self.steering_amount * left_edge)
    self.send_right_speed = bace_speed - (self.steering_amount * left_edge)
```

付録：ETロボコン走行体をROS 2で動かす



動画：<https://www.youtube.com/watch?v=RoaVhumuqcQ>