自律型ロボットe-Puck用イベント駆動型ライブラリ

落合剛†　鈴木徳一郎†　松原裕人†　大谷真†
湘南工科大学　情報工学科†

1. はじめに

e-Puck はスイス連邦工科大学ローザンヌ校で開発され 2006 年に一般リリースされた自律型の研究用小型移動ロボットである。移動用モータ、赤外線センサ、LED、カメラ、音源探知マイクなどを装備し、マイクロプロセッサ内のソフトウェアでこれらを制御する。現在モータやセンサを個別に制御するための下位の関数は公開されているが、一連の動作を連携させるための総合的ライブラリはなく、アプリケーション内で個別に処理しなければならない。これを解決するためのイベント駆動型ライブラリを、走行にかかわる動作を中心に、開発した。

2. 開発方針

e-Puck への動作指示を、直進走行、カーブ走行、センサ実行などの行動(アクション)とその実行条件(イベント)によってモデル化した。イベントには、走行停止、センサ反応、即実行などがある。ユーザプログラムで、実行したいアクション群と実行契機のイベントを事前に登録してメインループ関数を実行すると、メインループ関数の中で適宜これらを自動実行するようにすることとした。これによって、ユーザはアプリケーションに係る部分だけを記述すればよく、コード量が減る。またイベントごとに処理が分かれるのでプログラムが分かり易くなる。

e-Puck のプロセッサ(dsPIC30F6014)は比較的高度な割込み機構を持つが、e-Puck 搭載デバイスは、タイマ割り込みを除いては割込み型制御ではなく受動型制御であるため、メインループ関数実装内でのイベント監視はポーリングで行うこととした。

3. ライブラリの外部仕様

3.1 概要

主要な関数は、アクション登録関数、イベント登録関数、およびメインループ実行関数の 3 つである。図 1 に示すように、ユーザプログラムではアクションとイベントを登録した後、メインループ実行を指示する。イベント登録時にはそのイベントが発生したときどのアクションを実行するかも指示しておく。イベント駆動ライブラリの中では、アクションとイベントは、それぞれを管理するリストに登録される。その後、メインループ実装部分内でイベント発生を監視し、イベントが発生するとそれにリンクされたアクションを実行する。表 1 に具体的な関数名およびそれぞれの引数と戻り値を示す。

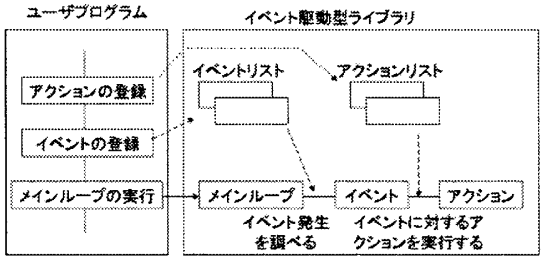


図 1 ライブラリ概要

表 1 ユーザプログラムで使用する関数

関数名	引数	戻り値	機能
register_action	void *action_func void *action_param	int action_id	アクションを登録する
register_event	void *event_func void *event_param int action_id	int event_id	イベントを登録する
cancel_action	int action_id	なし	アクションを削除する
cancel_event	int event_id	なし	イベントを削除する
mainloop	なし	なし	メインループ

3.2 アクションに関する関数

1) register_action()

アクションを登録する。アクション関数には標準アクション(プログラミング不要)と非標準アクション(アクションの内容を自由にプログラミングできる)の 2 種類がある。表 2 に今回サポートしたアクション関数の一覧を示す。register_action() は戻り値としてそのアクションの ID(action_id)を返す。action_id はイベント登録時およびアクションキャンセル時に使用する。

表 2 アクション関数

	アクション名	パラメタ	パラメタの内容
標準	action_start_straight	speed, distance	速度,距離
	action_start_curve	radius, degrees, speed	半径,角度,速度
	action_stop_motor	なし	
	action_set_timer	なし	
	action_reset_timer	なし	
	action_start_sensor	なし	
	action_stop_sensor	なし	
	action_exit_main	なし	
非標準	(他の関数と違う限り自由)	任意	

2) cancel_action()

action_id を指定して登録したアクションをキャンセルする。

3.3 イベントに関する関数

1) register_event()

イベントを登録する。引数にはイベントの種類

Event-driven Library for Autonomous Robot e-Puck
†Tsuyoshi Ochiai , Tokuichirou Suzuki , Hiroto Matsubara , Makoto Oya – Shonan Institute of Technology

を示すイベント関数、およびこのイベントが発生したときに実行すべきアクションの action_id を指定する。アクション関数と同様にイベント関数には標準イベントと非標準イベントの 2 種類がある。表 3 にイベント関数の一覧を示す。register_event()はその ID(event_id)を返す。

表 3 イベント関数

	イベント関数	パラメタ	動作
標準	event_sensor	sense[8]	センサのイベント
	event_timer	time	タイマのイベント
	event_immediate	なし	無条件のイベント
	event_stop_motor	なし	モータ停止時のイベント
非標準	(他の関数と違う限り自由)	任意	

2) cancel_event()

event_id を指定して登録したイベントをキャンセルする。

3.4 メインループ実行関数

mainloop()

メインループを起動する。イベント、アクションをすべて登録した後に実行する。

3.5 使用例

図 2 に簡単な使用例を記す。e-Puck を速度 250 ステップ／秒で 100mm 直線で走行させ、停止したらメインループを終了させる例である。2 つのアクション(走行開始とメインループ脱出)と 2 つのイベント(即実行と走行停止)を登録した後メインループを実行している。なおここでステップとは移動用の車輪付ステッピングモータのステップのことで円周 257.61mm を 1 回転分のステップ数 1000 で割った値が 1 ステップで進む距離である。utility.h はこのファイルを読み込むと全てのライブラリを使用できるようになるものである。

```
#include "utility.h"
int main(void){
        int a1, a2, e1, e2;
        param_start_straight sample = {250, 100};
        a1 = register_action(start_straight, &sample);
        a2 = register_action(exit_main, NULL);
        e1 = register_event(immediate, NULL, a1);
        e2 = register_event(event_stop_motor, NULL, a2);
        mainloop();
        return 0;
}
```

図 2 使用例

4. ライブラリの実装

4.1 アクションリストとイベントリスト

アクションリストとイベントリストは図 3 に示す構造体の配列とした。リストの実現には他にポインタリストとする方法もあるがコードが複雑となり、またメモリの消費大のため配列とした。なお 2 つのリストともに action_id, event_id と配列のインデックスをそれぞれ対応させている。action_function と event_function は登録したアクション、イベントの関数ポインタを格納する。action_param, と event_param は登録した関数のパラメタを格納する。action_id はイベントが発生したときに実行すべきアクションの ID を格納する。valid_flag はイベントの有効／無効を示すフラグで最初は無効。イベントを登録すると有効になる。

```
typedef struct{
        void (*action_function)(void *a);
        void *action_param;
}action;
typedef struct{
        void (*event_function)(void *a, void *b);
        void *event_param;
        int action_id;
        int valid_flag;
}event;
```

図 3 イベントリストとアクションリストの要素

4.2 メインループ関数の実装

e-Puck のデバイスは割込みではなく受動的に制御するため、メインループ内部でポーリングによってイベントの監視を行う。図 4 にメインループのフローを示す。

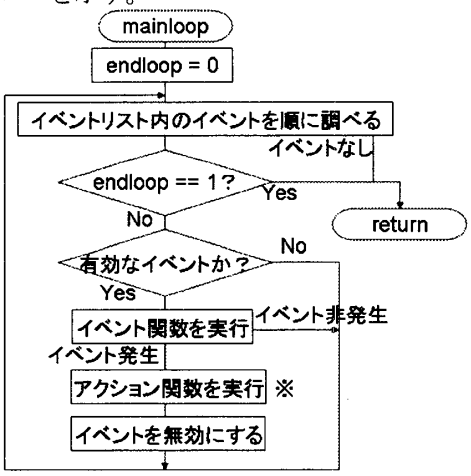


図 4 メインループのフロー

1 回のポーリングサイクルではイベントリスト内の要素について以下を実行する。

1)イベントが有効かどうか調べる。

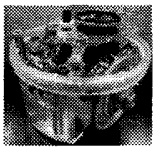
2)有効であればイベント関数を実行する。イベント関数はイベントが発生したかを調べる。

3)イベントが発生していた場合 action_id から実行すべきアクション関数を実行する。なお実行するアクション関数が exit_main の場合 endloop に 1 をセットする。

4)イベントを無効にする。

5. まとめ

これによって e-Puck の動作の基礎となるライブラリのプロトタイプが完成した。しかし、機能面でイベントに優先順位が付けられないなど、多くの問題がまだ残されている。今後は実用に耐え得るライブラリにするための改良を行いたい。



参考文献

e-Puck サイト

1. http://www.e-puck.org/