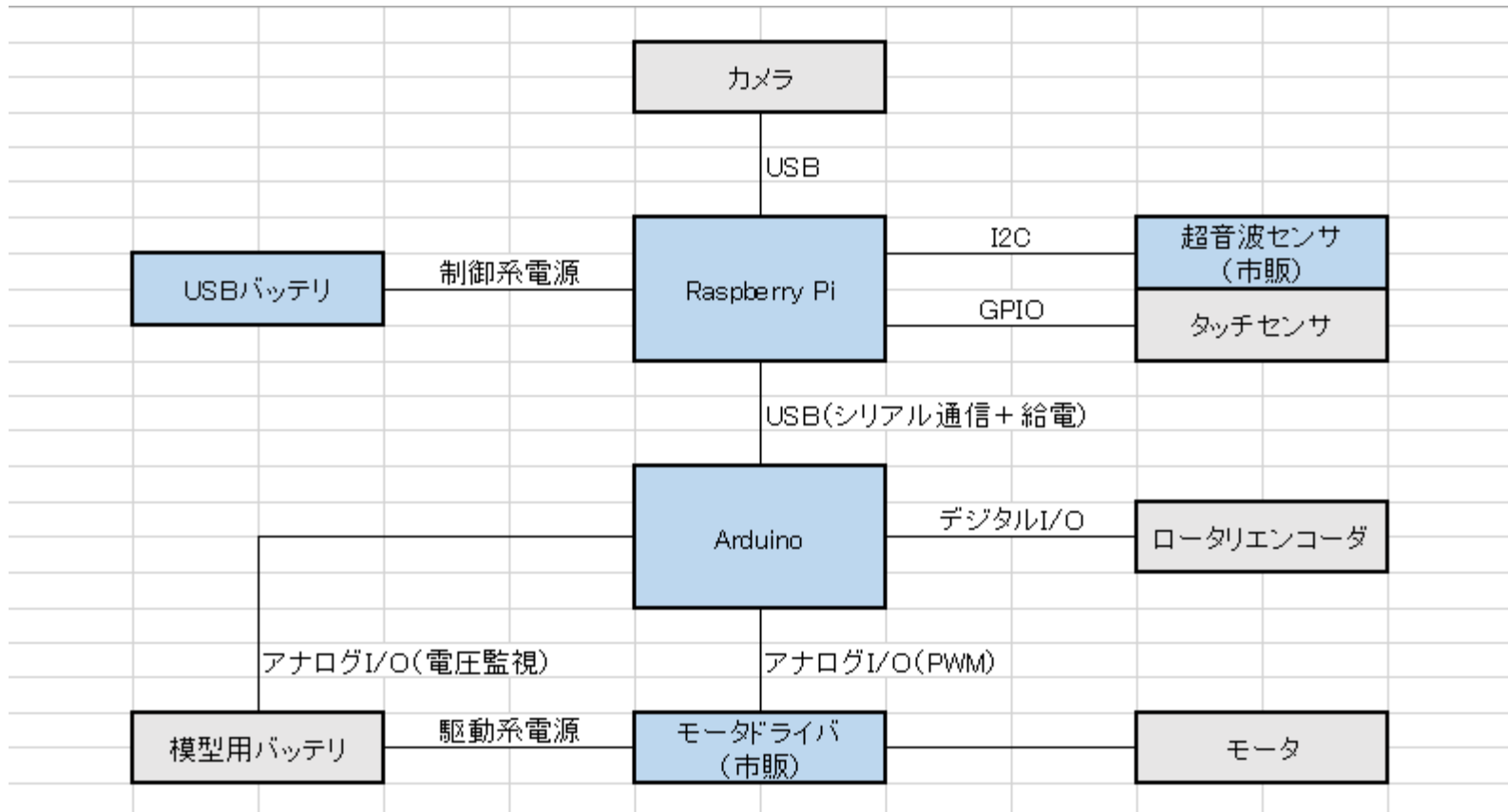


MIRSソフトウェア解説

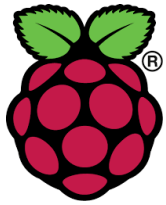
MIRS標準機の開発環境と 標準プログラム

2023/5/19

MIRS標準機の構成

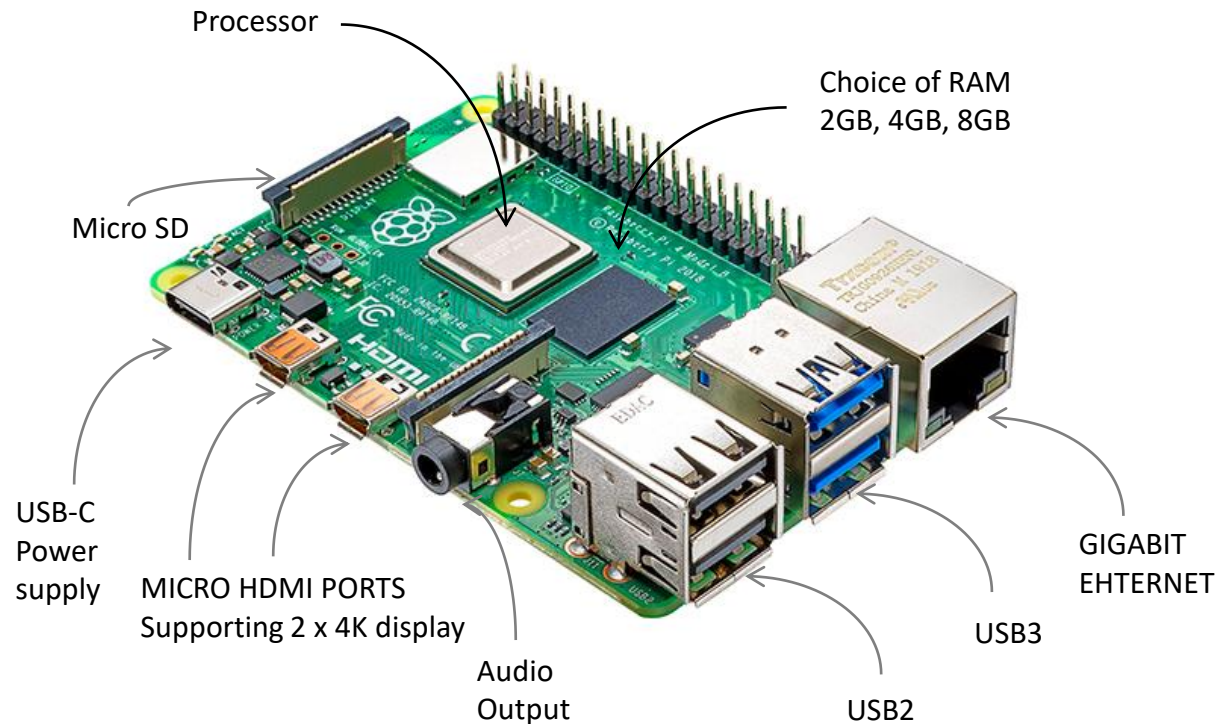


Arduino が走行制御を担い、それ以外の処理は Raspberry Pi が行っている。
Raspberry Pi と Arduino はシリアル通信により命令・データを送受信する。



Raspberry Pi 4 (Model B)

- らずべりーぱい
- シングルボードコンピュータ
- PCと遜色ない機能・性能
 - CPU
 - 1.5GHz
 - クアッドコア
 - ARM Cortex-A72
 - メモリ 4GB
 - USB, HDMI, LAN
- 40pin GPIO



Arduino

- Arduino UNO
 - シングルボードマイコン
 - [Arduino 製品群](#)の中で最もスタンダード
 - 5V駆動（USB供給可）
 - IO
 - 信号レベル: 5V
 - アナログピン 5
 - デジタルピン 14
 - 割込み 2ピン
 - [PWM出力](#) 6ピン
 - CPU: ATmega328P



MIRS標準機のソフトウェア開発

- MIRS標準機
 - Raspberry Pi と Arduino の2つのマイコンで制御系を構成している。
 - Arduino が走行制御を担い、それ以外の処理は Raspberry Pi が行っている。
 - Raspberry Pi と Arduino はシリアル通信により命令・データを送受信する。
 - それぞれに**標準プログラム**が用意されている。
- 開発環境
 - Raspberry Pi
 - 開発言語: C言語
 - ソースコードはエディタがあれば可
 - GCC がインストールされていればコンパイルまで可
 - Arduino
 - 開発言語: Arduino言語(≒C言語)
 - Arduino IDE がインストールされているPC (Raspberry Pi含む)

Raspberry Pi の開発環境

- OS : Raspberry Pi OS
 - 2020年5月、[Raspbian](#) (らずびあん) から改名
 - Debian系 Linux を Raspberry Pi にカスタマイズしたもの
 - 32bit (8GBメモリ搭載用には64bitベータ版)
 - Buster (1つ前のバージョン、現在のImagerのデフォルトはBlluseye)
- ライブラリ
 - [Wiring Pi](#) : Raspberry Pi のGPIO、シリアル通信 (I2C、RS232C)
(Blluseye を入れた場合は追加インストールの必要あり)
 - [OpenCV](#) : 画像処理 (APIはC++, Python に対応) バージョン3 or 4
 - Pthread : マルチスレッド (マルチタスク)
- 開発言語
 - 標準で C, C++, python (2 or 3) による開発が可能。
 - C, C++ のコンパイラは GCC (gcc, g++)
 - 標準プログラムはC言語で記述 (一部に C++で記述)

Arduino の開発環境

- Arduino IDE
 - Arduino の統合開発環境
 - Windows, PC, Linux (Raspberry Pi OS 含む) 版あり
 - Version 1.x と 2.x があるが、ソースレベルでは互換性あり(上位互換)
- Arduino 言語
 - C/C++をベースにしており、C言語のすべての構造と、いくつかのC++の機能をサポートしている。
 - [Arduino 日本語リファレンス](#)
 - 必須関数
 - loop関数
 - 名前のおり loop() 内の処理を繰り返す。
 - main関数の中の while(1) のようなもの
 - setup 関数
 - 起動時(loop関数実行前)に一度だけ実行する。
 - ピンモード(入出力、PWM、割込み)の設定など行う

標準プログラム

- RaspberryPi にはC言語、Arduino には Arduino 言語でコーディングされた標準プログラムを用意している。
- Raspberry Pi 用
 - MIRS2015の巡回警備の競技会に必要な機能モジュール群とそのテストプログラム
- Arduino 用
 - 直進・回転の走行制御を行う必要なモジュールと RaspberryPi との通信機能モジュールとそのテストプログラム

MIRS標準ソフトウェアのドキュメント

- MIRSMG4D 管理台帳



MIRSMG4D ドキュメント管理台帳

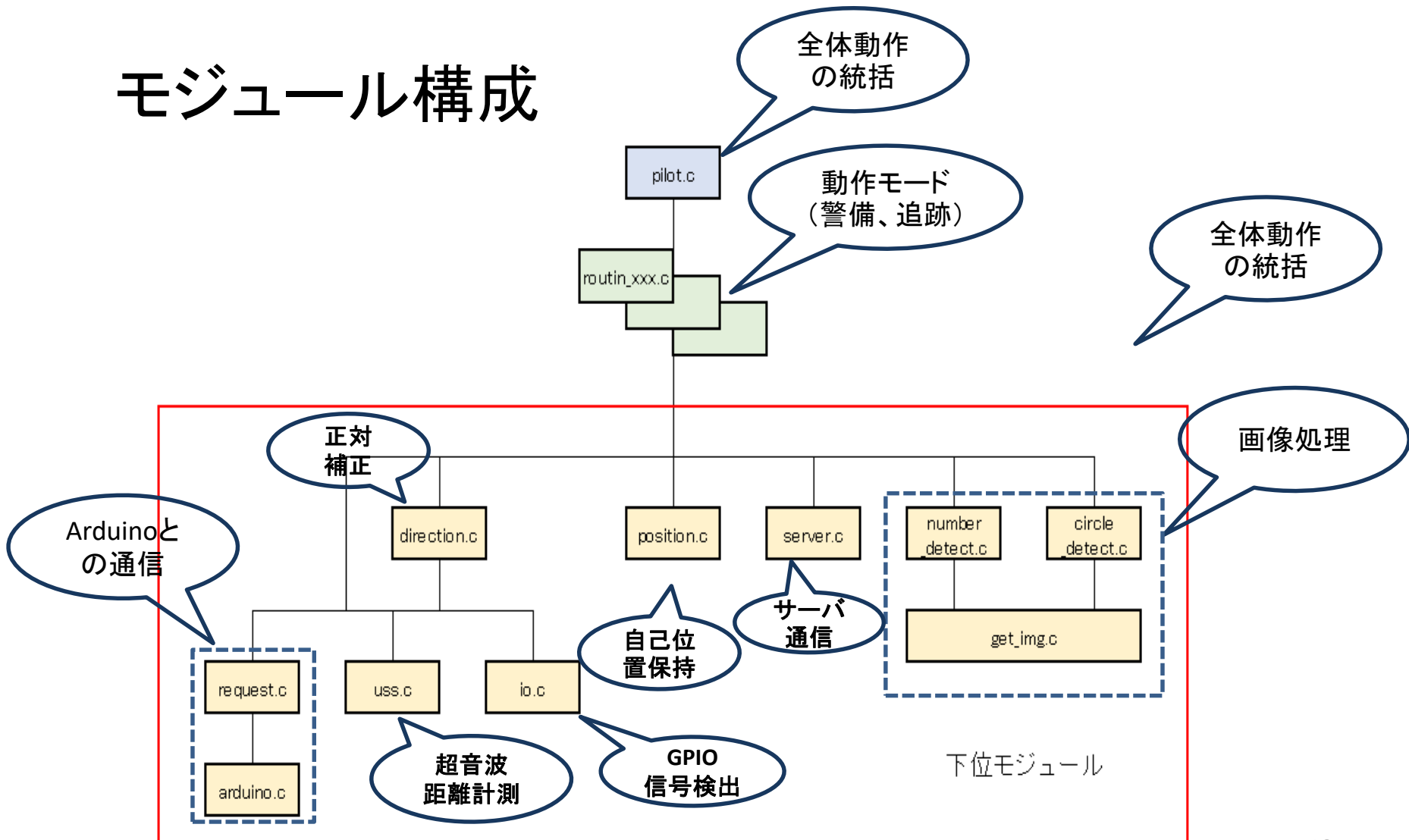
保護されていない通信 | www2.denshi.numazu-ct.ac.jp/mirsd2/mirsmg4d/

ソフトウェア

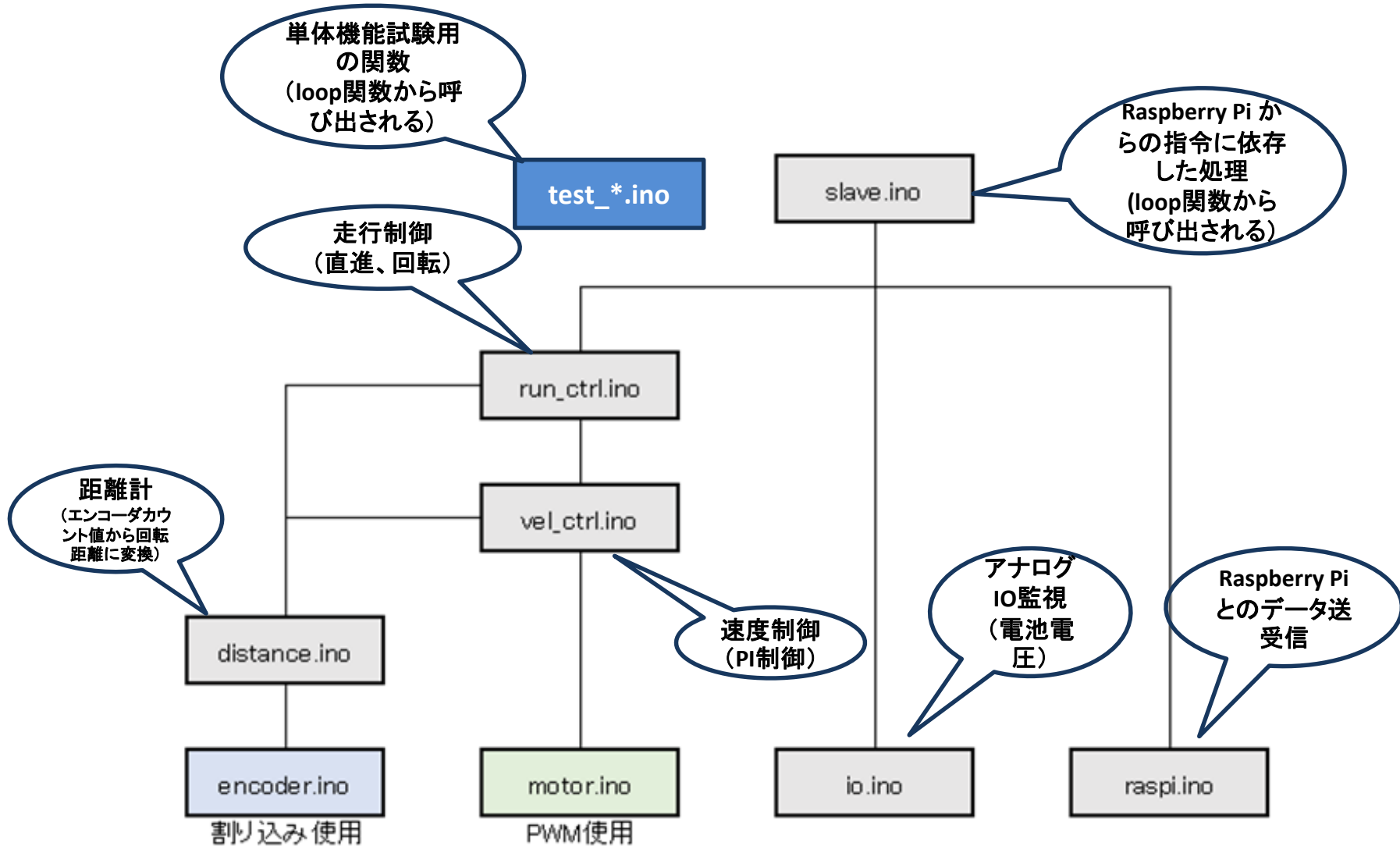
ドキュメント番号	ドキュメント名称	採番者	版数	作成者	登録日	備考
MIRSMG4D-SOFT-0001	Arduino ソフトウェア 解説	鈴木静男	A01	牛丸真司	2017/04/19	Ver 3.1
MIRSMG4D-SOFT-0002	Arduino ソフトウェア 関数レファレンス	鈴木静男	A01	牛丸真司	2017/04/19	Ver 3.1
MIRSMG4D-SOFT-0003	Arduino ソフトウェア ソースコード	鈴木静男	A01	牛丸真司	2019/05/01	Ver 3.1.3
MIRSMG4D-SOFT-0004	RaspberryPi ソフトウェア 解説	鈴木静男	A01	牛丸真司	2017/04/19	Ver 3.0
MIRSMG4D-SOFT-0005	RaspberryPi ソフトウェア 関数レファレンス	鈴木静男	A01	牛丸真司	2017/04/19	Ver 3.0
MIRSMG4D-SOFT-0006	RaspberryPi ソフトウェア ソースコード	鈴木静男	A01	牛丸真司	2019/05/01	Ver 3.0.1
MIRSMG4D-SOFT-0007	ソケット通信によるプロセス間通信	牛丸真司	A01	牛丸真司	2018/12/21	

標準プログラム (Raspberry Pi)

モジュール構成



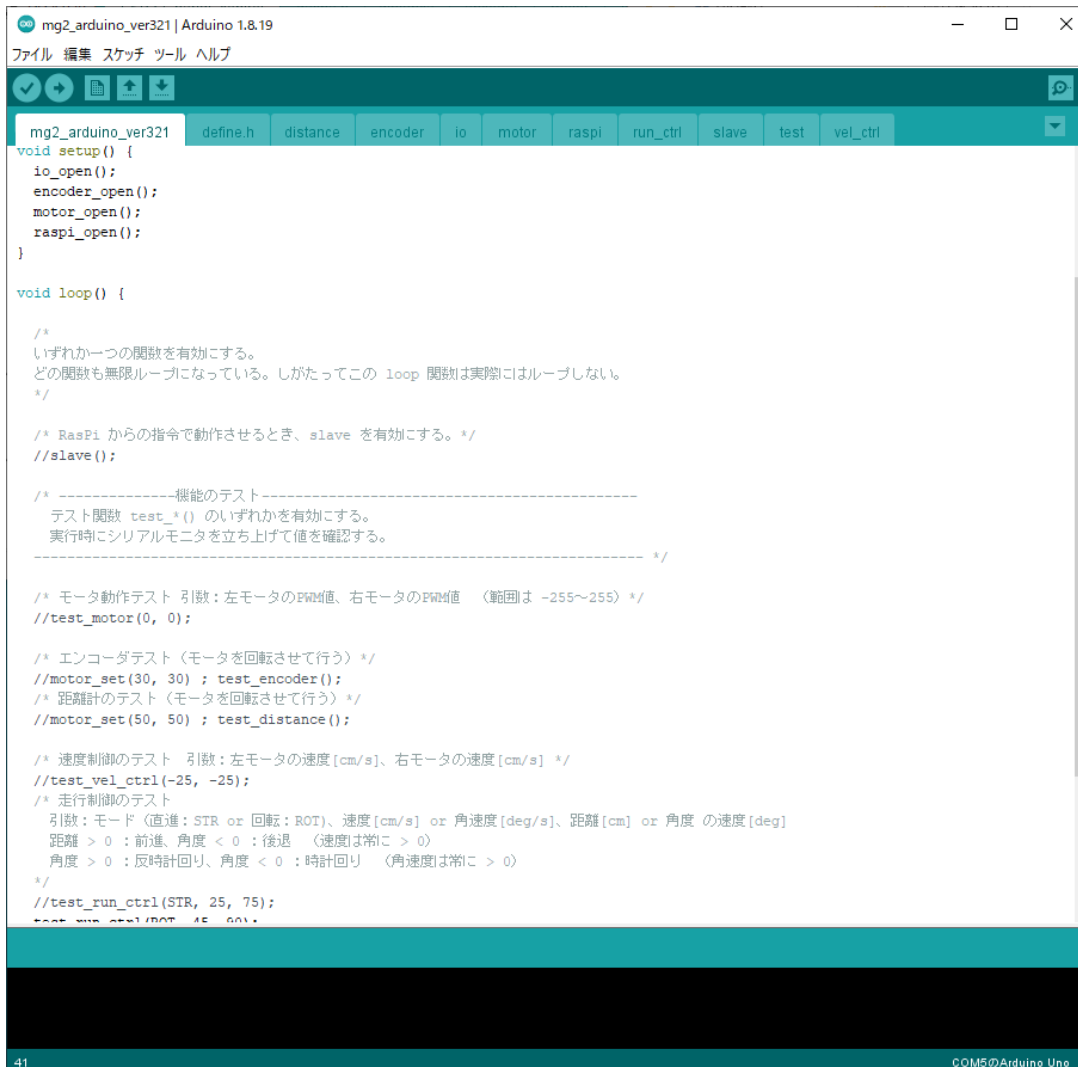
標準プログラム (Arduino) のモジュール構成



Raspberry Pi の標準プログラム

- モジュール毎に独立なファイルとしている(プログラム分割)
- ソースファイルに対応したヘッダファイルで、ソース内に定義している関数、共有変数の `extern` 宣言する。
- モジュール単体テスト用のテストプログラムを `test_*.c` として用意している。
- コンパイルはソースプログラムのトップで `make` する(トップディレクトリに `Makefile` を用意)。実行ファイルはトップディレクトリに作成
- ソースプログラム(*.c)、ヘッダファイル(*.h)、オブジェクトモジュール(*.o) を、`src`, `include`, `build` ディレクトリに分けて置いている。
- 最新版は 4.1.0

Arduino の標準プログラム



```
mg2_arduino_ver321 | Arduino 1.8.19
ファイル 編集 スケッチ ツール ヘルプ

mg2_arduino_ver321  define.h  distance  encoder  io  motor  raspi  run_ctrl  slave  test  vel_ctrl
void setup() {
  io_open();
  encoder_open();
  motor_open();
  raspi_open();
}

void loop() {

  /*
  いずれか一つの関数を有効にする。
  どの関数も無限ループになっている。しがたってこの loop 関数は実際にはループしない。
  */

  /* RasPi からの指令で動作させるとき、slave を有効にする。*/
  //slave();

  /* -----機能のテスト-----
  テスト関数 test_*() のいずれかを有効にする。
  実行時にシリアルモニタを立ち上げて値を確認する。
  ----- */

  /* モータ動作テスト 引数: 左モータのPWM値、右モータのPWM値 (範囲は -255~255) */
  //test_motor(0, 0);

  /* エンコーダテスト (モータを回転させて行う) */
  //motor_set(30, 30); test_encoder();
  /* 距離計のテスト (モータを回転させて行う) */
  //motor_set(50, 50); test_distance();

  /* 速度制御のテスト 引数: 左モータの速度 [cm/s]、右モータの速度 [cm/s] */
  //test_vel_ctrl(-25, -25);
  /* 走行制御のテスト
  引数: モード (直進: STR or 回転: ROT)、速度 [cm/s] or 角速度 [deg/s]、距離 [cm] or 角度 の速度 [deg]
  距離 > 0 : 前進、角度 < 0 : 後退 (速度は常に > 0)
  角度 > 0 : 反時計回り、角度 < 0 : 時計回り (角速度は常に > 0)
  */
  //test_run_ctrl(STR, 25, 75);
  test_run_ctrl(ROT, 45, 60);
}
```

- モジュール毎に独立なファイルを作成
 - IDE上ではタブで分離
- テストプログラム
 - testのタブに記述
 - API は test_ で始まる関数
 - loop関数中の一つのtest_関数のみを使用する(他はコメントする)
- 最新版は 3.2.0

プログラムの追加(改変)

- Raspberry Pi
 - Makefile に変更を加える。
 - 具体的な変更方法はプログラミング応用で説明するが、標準プログラム内の Makefile から類推することも可能
- Arduino
 - Arduino IDE のタブを追加

標準プログラムのコーディングルールに従う場合は、ソースプログラム以外にヘッダファイル、テストプログラムを用意する。

ソフトウェア開発と試験

- 組み込みシステムの開発（MIRS開発を含む）では、ソフトとハードの並行開発が一般的
- コーディングは、ハードウェアの有無に関係なく可能
 - コーディングだけなら、エディタさえあれば可能
 - コンパイルチェックは必要なライブラリがある環境で可能
- 単体・機能試験
 - 試験を行いたいハードウェアがあれば、単体・機能試験は可能
 - ハードウェアは実機にマウントされている必要はない
- ハードウェアが完成していなくても（手元になくても）、**ハードウェアの仮想化**による動作試験が可能

仮想化による試験

- ハードウェアの仮想化

- 実際のハードウェアの別ハードウェアまたは事前に用意したデータで代用する。

- 例 タッチセンサ押下 → キーボード入力
 カメラ画像 → 事前に用意した画像の読み込み

- 通信 → 相手から受け取るデータを事前に用意

- ソフトウェアの仮想化

- 例：画像認識処理結果 → 事前に用意した処理結果情報を渡す。

他言語での開発

- 最近のMIRSオリジナル開発では、画像処理系や通信系など一部（または全て）を Python や C++で記述するチームが増えている。
 - Cで開発したプログラムと他言語で開発したプログラムを共存させることは可能。（データ・情報はプロセス間通信によって受け渡しできる）
 - 画像処理系は Python で書かれたソースコードのサンプルが多い。（特に DL 系）
 - OpenCV 3.xのAPIは C言語に非対応（C++, Python, Javaに対応）
 - TensorFlow, PiTorch などのDLフレームワークは自前で環境を作る必要がある。
 - RaspberryPi の標準プログラムを全て Python で書き換えたチームもあり（MIRS2101）

走行制御

- モータはPWM信号で実効電圧(=平均電圧)を変化させて制御する。(参考ページ)
- ロータリーエンコーダのA層信号を用いた割り込み処理により、エンコーダの矩形信号エッジのカウントを行う。
- ロータリーエンコーダのカウント値を利用して、左右のモータの速度制御をPI制御で行う。
- ロータリーエンコーダのカウント値を利用して、機体の直進・回転制御をPID(P/PD/PID)制御で行う。

直進制御ブロック線図

