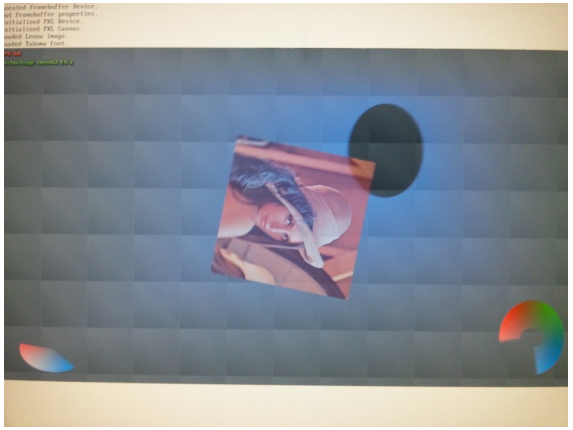
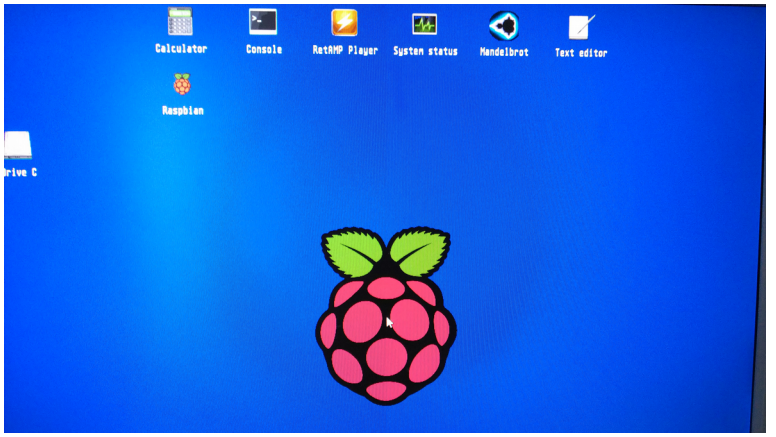
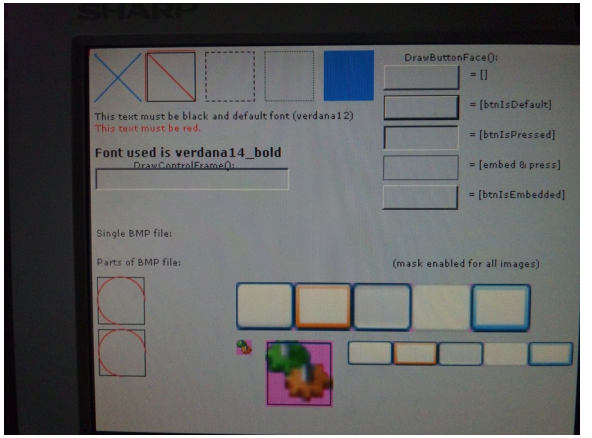
ultibo
.ORG

You + Ultibo = Pure Innovation







https://ultibo.org/

ベアメタル

- 主にRasberryPiで用いられる(サーバは別物）

  LinuxなどのOSを組み込んだシステムと、区別するために使われる
- OS無しでアプリを動かす
- OSのロード、起動時間が無いので起動が速い
- OSの恩恵も受けられない

  ネットワーク、マルチメディア、データベースアクセス

  GUIアプリケーションなどの作成は大変

Ultiboとは

- Raspberry Pi用組込み、ベアメタル開発環境
- OSではないが、OSで提供される多くのサービスを提供（メモリ管理、ネットワーク、ファイルシステム、スレッド…）
- オーストラリアのソフト開発会社SoftOzで作られた
- SoftOzディレクタとUltibo共同クリエイタのGarry Woodがこれまでのところ、殆どのコードを作成
- 用途は製品作成、試作、教育まで…様々
-

Ultibo
他のベアメタルとの違い

- IDEまで含めたフレームワーク

  操作が簡単、コンパイルボタンをクリックするだけ
- 実用アプリを作るためのフレームワーク

  libcなどの低位の関数でなく、高レベルな関数、クラスによるプログラミング
- ネットワーク通信、シリアル通信をサポート
- SQLite3をサポート
- 商用利用が可能（修正LGPLライセンス）

Ultiboは速い

- 起動が速い
  - OSが無いため、直ぐに起動（Lチカなら2秒程度）
- 実行速度が速い
  - ネイティブコードなのでPythonに比べると高速
  - ユーザ空間、カーネル空間がないのでオーバヘッドがない
  - 単一メモリアドレスのため、I/O制御が高速
- ビルドが速い
  - Lazarusは起動時間がEclipseの1/10の以下で起動
  - Pascalは1パスコンパイラなため、C++よりビルドが速い

Ulitiboの基本構成

- Ultibo Coreライブラリ

  Ultiboの基本ライブラリで、プラットフォーム、ハードウェアサポートだけでなく、メモリ、スレッド、マルチCPU、ロック/同期などをサポート

- FPC(Free Pascal)

  オブジェクト指向Pascalコンパイラ

  ARM, i368, x86-64や、JVMにも対応

- Lazarus(IDE)

  FPC用の統合開発環境

  Window, Mac, Linuxで稼働

  Rasberry Pi3でも使用可能

Ultibo Core

- 実行の単位はスレッド（数十～数百）
- 単一メモリ空間
- マルチプロセッサ　サポート
- メモリ管理、スレッド、スケージュール管理、割り込み処理、デバイス管理それにFPCランタイムライブラリのためのコア機能
- ネットワークインターフェース、USB, MMC/SD、ハードウェア特有のドライバなど

Pascalは古い…けど

- Pascal誕生は1970年

  でも、現在はObject Pascal
- p-コードマシン

  UCSD Pascalなんてのもあった
- Android, iPhone対応も

  2010年頃から対応…Android2.xの頃

  （ARMネイティブコード、JavaVMコード両方が使用可）

  Object-C以外の言語で、最初に登録されたiPhoneアプリは FPC+Lazarusで開発された
- FPCは各種プロセサ、OSプラットフォームに対応

Lazarus

- GUIグラフィカルデザインをサポートするIDE
- Delphi, VB, C#ライクな操作
- Window, Linux, Macサポート
  （RasberryPiでも使用可能）
- Win32, Gtk2+,Gtk3, Qt4, Qt4, Cocoa対応
- 豊富なユニット（ライブラリ）

  ビジュアルなDBアプリ作成コンポーネント、スプレッドシート、Google APIなど
- クロスコンパイル、リモートデバッグサポート

  Linuxで、Windows32/64bitアプリ、RasberryPiのアプリ開発可能
- Ultiboでは専用にカスタマイズしたLazarusを使用

  GUIデザイン機能などが止められている

UltiboとC

- Newlib
  - C言語用の標準Cライブラリ
  - PascalからCの変数のアクセス、関数の呼び出しが可能
  - Pascalからスタティックリンクしたプログラムを実行可能
- LazarusでC言語ソースの編集
  - Lazarus IDEでC言語のソースの編集も可能（スマートではないが）
  - Makefileは問題あり（TAB文字入力）
- Lazrusからビルド
  - CのMakefileを用意し、プロジェクトオプションのビルドでmakeを記述すれば、ビルドボタンを押すだけ

Ultiboをパワーアップ

- Synapse/Synaser

  FPC, Delphiなどで使用されるネットワーク通信、シリアル通信用のライブラリ

- fpGUI

  FPC用GUIツールキット

  Win32, GTK, Qt, Cocoaよりも軽量

  だがしかし、当然機能は限られる…

- Video Core IV

  OpenGLを使用したサンプル例

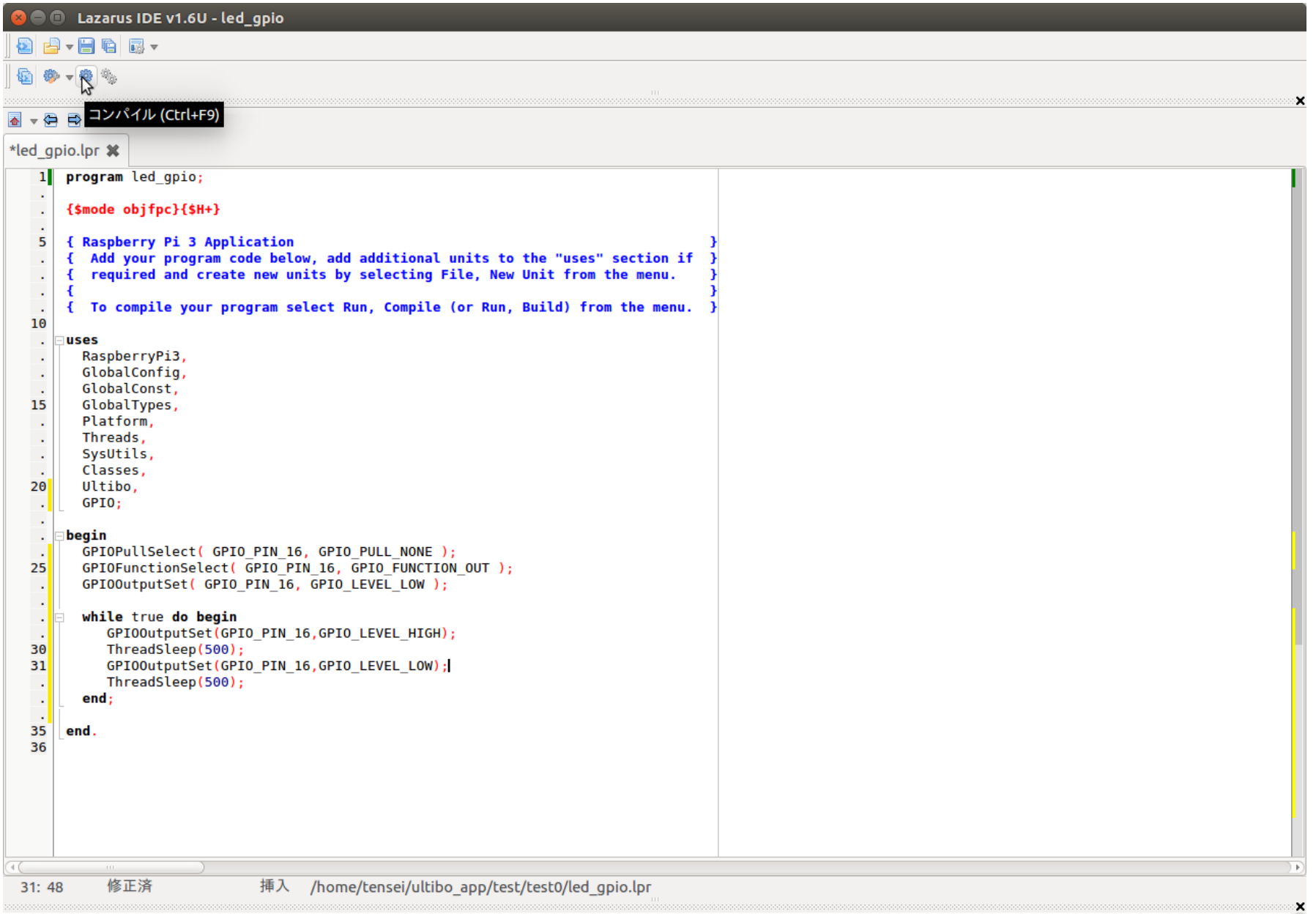
  Video Core Ⅳ用サンプルをスタティックリンクして稼働例

ライセンス

- Ultiboはスタテックリンクしてkernel*.imgを作成
  - 主にC言語のライブラリを使用する場合ライセンスに注意（LGPLは静的リンクの場合、ソース公開の必要性がある）
- Modefied LGPL
  - FPC/Lazarusで多く使用されているライセンス
  - スタティックリンクを可能にしている
- SQLite3
  - SQLite3はパブリックドメイン

ultibo プロジェクトの作成

コンパイル


Lazarus IDE v1.6U - led_gpio
コンパイル (Ctrl+F9)
*led_gpio.lpr
program led_gpio;
{$mode objfpc}{$H+}
{ Raspberry Pi 3 Application }
{ Add your program code below, add additional units to the "uses" section if }
{ required and create new units by selecting File, New Unit from the menu. }
{ }
{ To compile your program select Run, Compile (or Run, Build) from the menu. }
uses
RaspberryPi3,
GlobalConfig,
GlobalConst,
GlobalTypes,
Platform,
Threads,
SysUtils,
Classes,
Ultibo,
GPIO;
begin
GPIOPullSelect(GPIO_PIN_16, GPIO_PULL_NONE);
GPIOFunctionSelect(GPIO_PIN_16, GPIO_FUNCTION_OUT);
GPIOOutputSet(GPIO_PIN_16, GPIO_LEVEL_LOW);
while true do begin
GPIOOutputSet(GPIO_PIN_16,GPIO_LEVEL_HIGH);
ThreadSleep(500);
GPIOOutputSet(GPIO_PIN_16,GPIO_LEVEL_LOW);
ThreadSleep(500);
end;
end.
31: 48
修正済
挿入
/home/tensei/ultibo_app/test/test0/led_gpio.lpr

Hello, world

```
program helloworld;

{$mode objfpc}{$H+}

uses
  RaspberryPi3, GlobalConfig, GlobalConst, GlobalTypes, Platform,
  Threads, SysUtils, Classes, Ultibo,
  Console;

var
  WindowHandle:TWindowHandle;

begin
  WindowHandle:=ConsoleWindowCreate(ConsoleDeviceGetDefault,
                                    CONSOLE_POSITION_FULL,True);

  writeln( 'Hello, world' );

  ThreadHalt(0);
end.
```

Lチカ

```
program led_gpio;

{$mode objfpc}{$H+}

uses
  RaspberryPi3, GlobalConfig, GlobalConst, GlobalTypes, Platform,
  Threads, SysUtils, Classes, Ultibo,
  GPIO;

begin
  GPIOPullSelect( GPIO_PIN_16, GPIO_PULL_NONE );
  GPIOFunctionSelect( GPIO_PIN_16, GPIO_FUNCTION_OUT );
  GPIOOutputSet( GPIO_PIN_16, GPIO_LEVEL_LOW );

  while true do begin
     GPIOOutputSet( GPIO_PIN_16, GPIO_LEVEL_HIGH );
     ThreadSleep( 500 );
     GPIOOutputSet( GPIO_PIN_16, GPIO_LEVEL_LOW );
     ThreadSleep( 500 );
  end;

end.
```

SQLite3も簡単(1)

```
uses
  ～略～
  Console, FileSystem, FATFS, sqlite3,sqlite3db, strings;
var
  WindowHandle:TWindowHandle;
  MySQL: TSQLite;
  SQL: String;
  i, j: Integer;
  a: TStringList;
begin
  WindowHandle:=ConsoleWindowCreate(ConsoleDeviceGetDefault,
                                    CONSOLE_POSITION_FULL,True);

  ConsoleWindowWriteLn(WindowHandle,'Waiting for drive C:¥');
  while not DirectoryExists('C:¥') do
   begin
    Sleep(1000);
   end;

  ConsoleWindowWriteLn(WindowHandle,'Creating class');
  MySQL := TSQLite.Create('test.db');
  MySQL.BusyTimeout := 1000;
```

SQLite3も簡単(2)

```
 // writeln(MySQL.Version);
  ConsoleWindowWriteLn(WindowHandle,'Creating table');
  SQL := 'CREATE TABLE Test(No int, Nom varchar(32),Prenom
varchar(32));';
  MySQL.Query(sql, nil);
  SQL := 'INSERT INTO Test VALUES(1,''Coursiere'', ''Olivier'');';
  if MySQL.IsComplete(sql) then
    begin
    ConsoleWindowWriteLn(WindowHandle,'Inserting first row');
    MySQL.Query(sql, nil);
    end;
  SQL := 'INSERT INTO Test VALUES(2,''Jourde'', ''Eric'');';
  if MySQL.IsComplete(sql) then
    begin
    ConsoleWindowWriteLn(WindowHandle,'Inserting second row') ;
    MySQL.Query(sql, nil);
    end;
  ConsoleWindowWriteLn(WindowHandle,'Selecting rows') ;
```

SQLite3も簡単(3)

```
  SQL := 'SELECT * FROM Test;';
  MySQL.Query(sql, nil);
  ConsoleWindowWriteLn(WindowHandle,'Fields Names ----------------');
  for i:=0 to MySQL.List_FieldName.count-1 do
    writeln(i,' -> ',MySQL.List_FieldName.Strings[i]);
  ConsoleWindowWriteLn(WindowHandle,'Fields ----------------');
  for i:=0 to MySQL.List_Field.count-1 do
      begin
        a:=TStringList(MySQL.List_Field.items[i]);
        write(i,' -> ');
        for j:=0 to a.count-1 do
          write(a.Strings[j],'  ');
        ConsoleWindowWriteLn(WindowHandle, '');
      end;

//  SQL := 'DROP TABLE Test;';
//  MySQL.Query(sql, nil);
  MySQL.Free;
  ThreadHalt(0);
end.
```

Ultiboがまだまだなところ…

- SSL、SSHなど暗号化が絡んだライブラリ

  単に暗号化が難しいというよりは、スタティックリンクするためのライセンスの問題も関係し、WiFiなどの開発が遅れている理由の一つである
- WiFi, Bluetoothのサポート
- LazrusのGUIグラフィカルデザインによるアプリが書けない
- RaspberryPi以外の対応は保留

Ultiboアプリケーション

- CP/M on RasberryPi

  CP/MとZ80のエミュレータ

- Oberon

  チューリッヒ工科大学のニクラウス･ヴィルト率いるチームが設計開発したオペレーティングシステムとプログラミング言語

- ultibo_retro_gui

  GUI for Ultibo/Retromachine environment

  SIDプレーヤ

Youtube 動画デモ
Discovering Ultibo

- Episode 1 Getting Started
- Episode 2 Exploring USB
- Episode 3 Building the RTL
- Episode 4 GPIO Events
- Episode 5 LCD 16×2
- Episode 6 Remote LED

Ultibo サンプル

- ExamplesMaster
  HelloWorld, Blinker, ScreenOutput, KeybordInput, TimeDate,
  PascalObject, Exceptions, FileHandling, LogOutput,
  Multithredding, MultiCPU, WebServer, SerialConnection,
  GPIOHadling, MouseCursor, BouncingBoxes, TextEditor,
  PWMControl
  (Advanced) DedicatedCPU, RAMDisk, Sensormatic3000, UDPServer
  (Contributed) MouseDrawing
- Synapse
  EchoDaemon, FTPServ, HTTPProxy, HTTPServ, SNMPServ
  SNTP, Scan, SendMail, SerialEcho, TFTPServer

Ultiboサンプル

- Asphyre-ultibo

  Basic, Blinky, Combustion, DisplayFB, DisplaySPI

  DisplaySPIAndFB, Media, Plasma, Shapes, Tunnel

- fpGUI-ultibo

  aggcanvas, canvastest, eventtest, helloworld

Ultibo サンプル (Video Core Ⅳ)

- HelloPi(C)

  HelloAudio, HelloDispmanX, HelloEncode, HelloJPEG, HelloMMALEncode, HelloTeapot, HelloTiger, HelloTriangle, HelloTriangle2, HelloVideo, HelloVideocube
- OpenGLES(FPC)

  HelloGLES, HelloGLES2
- OpenVG(FPC)

  HelloVG, ShapesDemo
- RaspiCam(C)

  RaspiStill, RaspiVid