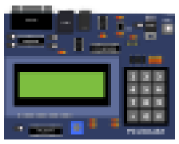


How to Compile PICsimLab and Create New Boards

Luis Claudio Gambôa Lopes <lcgamboa@yahoo.com>

http://sourceforge.net/projects/picsim/

November 11, 2015

Contents

Chapter 1

How to Compile PICsimLab

Code to compile:

- liblxrad (lxrad.dll)
- libpicsim (picsim.dll)
- PicsimLab (picsimlab.exe)

Tools needed:

- make
- gcc
- g++
- autoconf
- doxygen

Dependencies:

- wxwidgets-3.0

1.1 Install Compilers and Tools

For Linux based in debian distro:

```
sudo apt-get install make gcc g++ autoconf doxygen libwxgtk3.0-dev
```

For Windows:

- Install MinGW MSYS
- Download and compile wxwidgets-3.0 with MSYS.

1.2 Compiling and Install LXRAD Library

Link for source code download lxrad-0.8.tgz.

To build lxrad, open a shell (MSYS in windows) and type the following commands in the folder of downloaded file:

```
tar xvfz lxrad-0.8.tgz <enter>
cd lxrad-0.8 <enter>
autoconf <enter>
./configure <enter>
make <enter>
sudo make install <enter>
```

1.3 Compiling and Install PICsim Library

Link for source code download picsim-0.6.tgz.

To build picsim library, open a shell (MSYS in windows) and type the following commands in the folder of downloaded file:

```
tar xvfz picsim-0.6.tgz <enter>
cd picsim-0.6 <enter>
make <enter>
sudo make install <enter>
sudo ldconfig <enter>
```

1.4 Compiling and Install PICsimLab

Link for source code download PICsimLab-0.6.

To build PICsimLab, open a shell (MSYS in windows) and type the following commands in the folder of downloaded file:

```
tar xvfz picsimlab-0.6.tgz <enter>
cd picsimlab-0.6 <enter>
make <enter>
sudo make install <enter>
```

You can use netbeans IDE with C/C++ plugin to compile PICsimLab using MSYS. After extract files, open netbeans and create a C/C++ project using existing files and search for picsimlab-0.6 directory.

Chapter 2

Creating a New Board

The first step is get the schematic and all information about the board hardware. The second step is the creation of five files in PICsimLab dir (consider replace the 'x' of board_x for a number or name in your case):

- Board Picture (share/board_x.png);
- Board input map (share/input_boardx.map);
- Board output map (share/input_boardx.map);
- Board header (board_x.h);
- Board C++ code (board_x.cc);

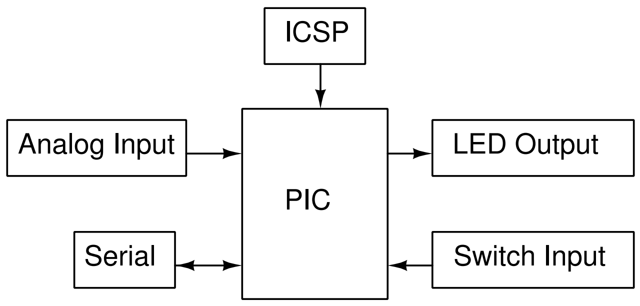
The third step is modify two files in PICsimLab dir to include the new board:

- Makefile
- boards_defs.h

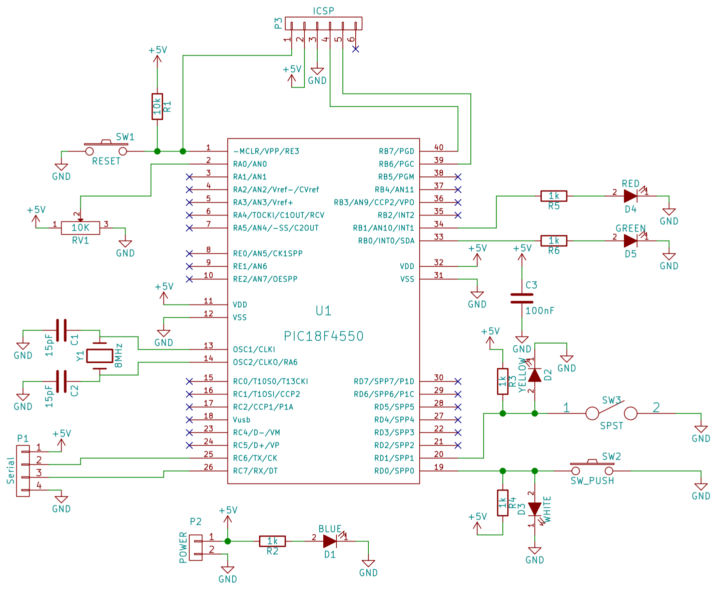
The four and last step is recompiling PICsimLab with new board support.

2.1 Board Hardware and Schematic

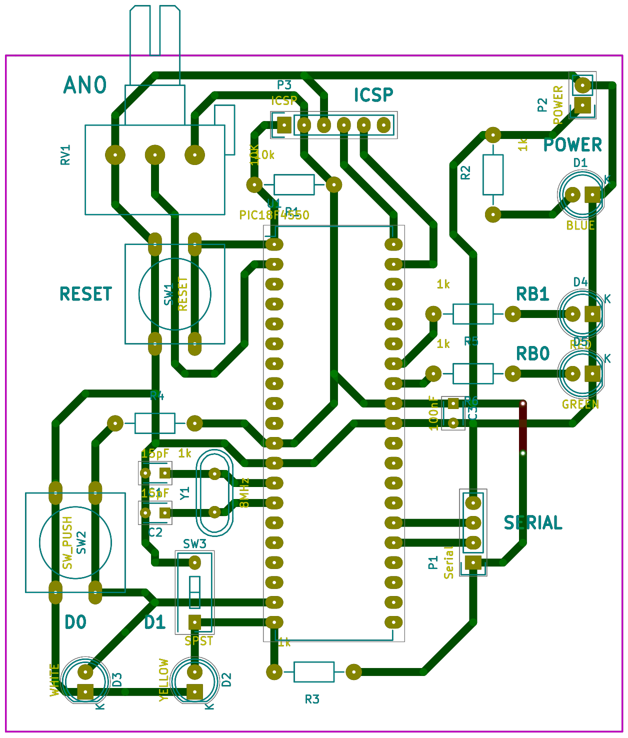
For this tutorial, the board created have the hardware shown in diagram below:



The schematic for the tutorial board made in Kicad.

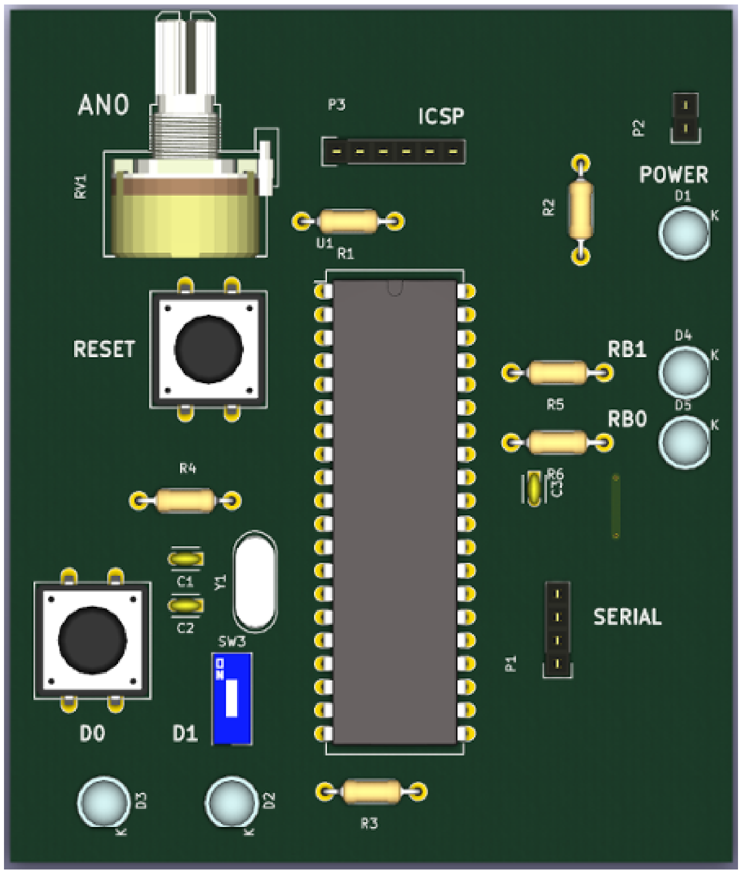


And the PCB layout was made in Kicad too. The PCB is not necessary if you have a real board.



2.2 Board Picture

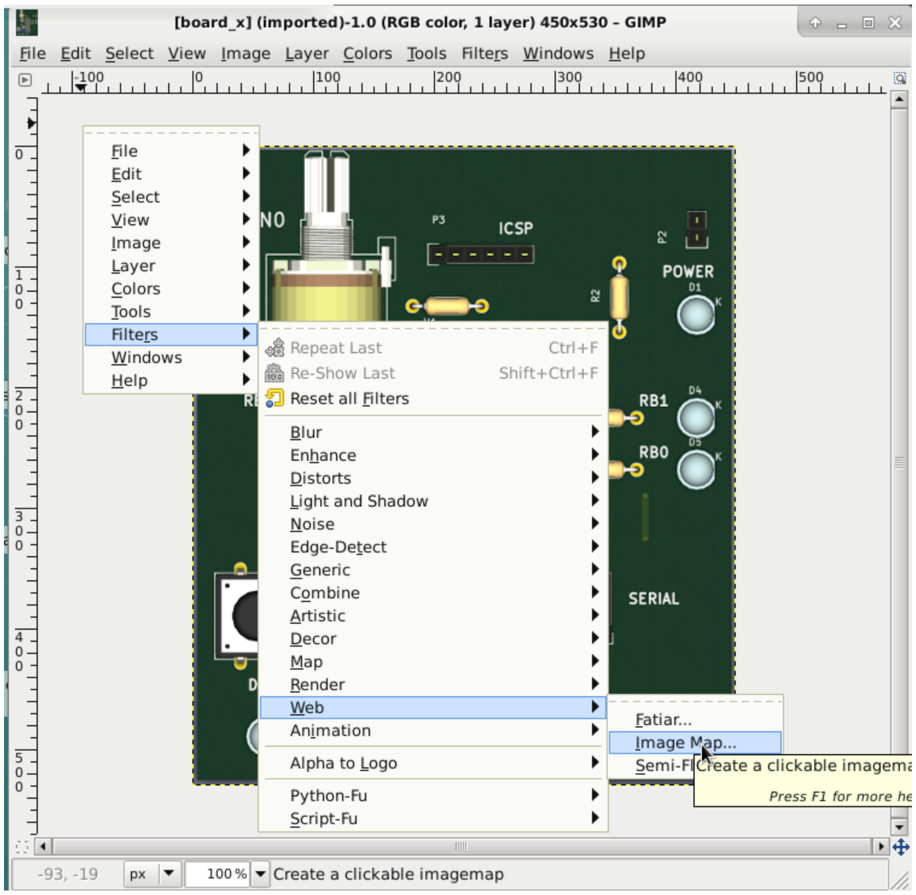
Because the real board of this tutorial never has been built, the board picture was taken from Kicad 3D viewer. The picture image is saved as “share/board_x.png”.



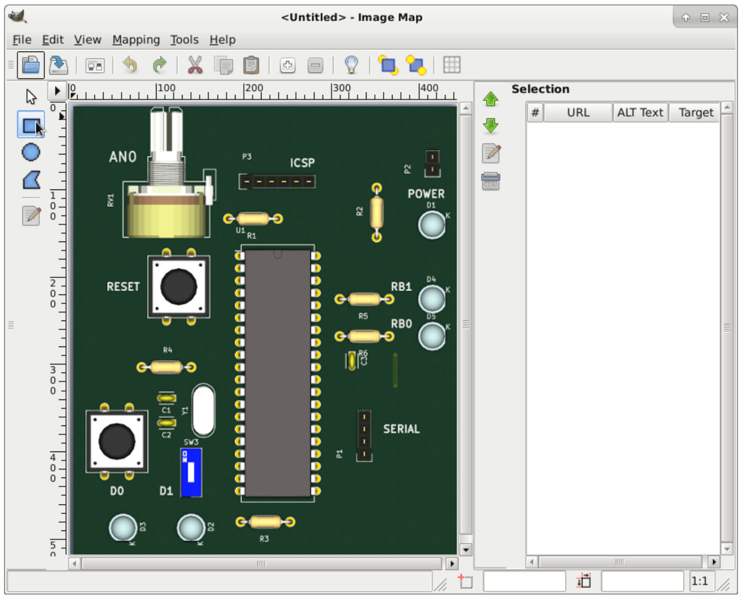
2.3 Picture maps

The PICsimLab use two type of image maps. The input map mark the areas in board picture which user can interact (by mouse click). The output map mark the areas in board picture to be redraw according simulator status. The picture maps used for PICsimLab are normal HTML image-map. They can be made by hand or using any software which can handle image maps. The original PICsimLab maps are made using Gimp image editor.

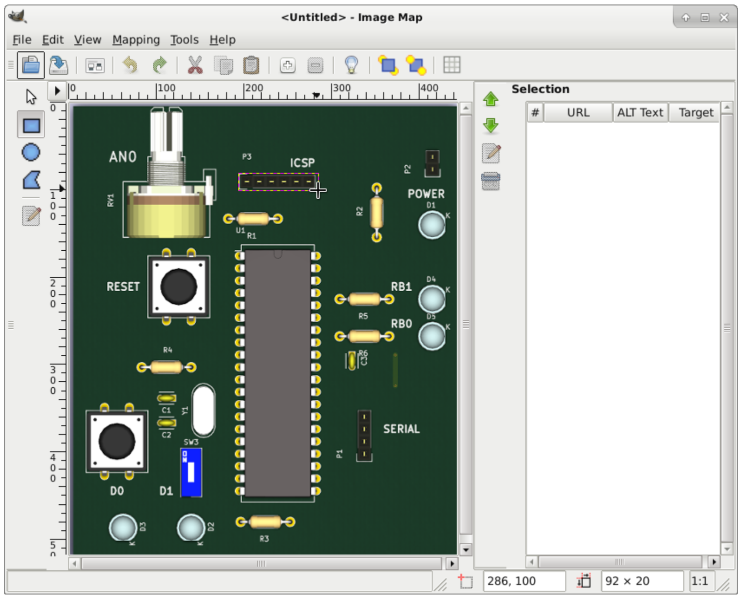
To start, in the GIMP, use the Filters->Web->Image Map to open image map editor window.



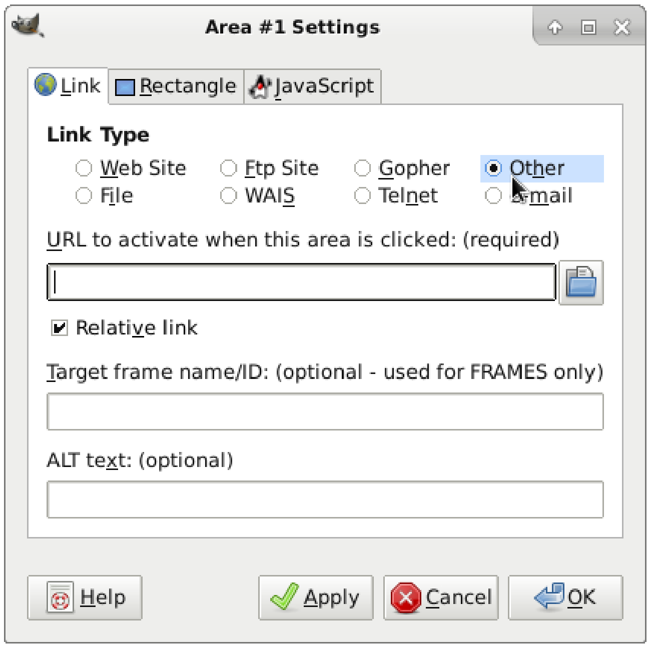
Then select rectangle or circle map on toolbar.



And mark the area in picture.



After area is select, in the settings windows select the link type for "Other".



Area #1 Settings

Link | Rectangle | JavaScript

Link Type

( ) Web Site ( ) Ftp Site ( ) Gopher (● Other)
( ) File ( ) WAIS ( ) Telnet ( ) e-mail

URL to activate when this area is clicked: (required)

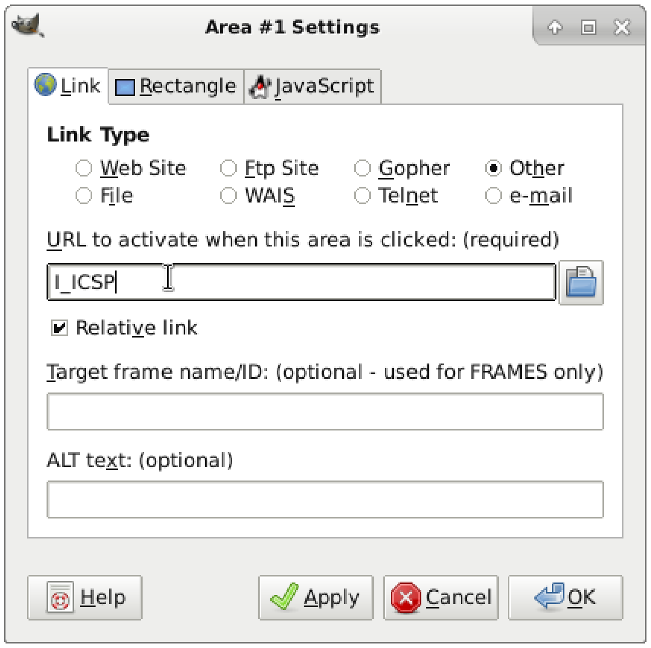
[✓] Relative link

Target frame name/ID: (optional - used for FRAMES only)

ALT text: (optional)

Help Apply Cancel OK

And write the name of area. The name must describe the area function on the board.



Area #1 Settings

Link | Rectangle | JavaScript

Link Type

( ) Web Site ( ) Ftp Site ( ) Gopher (● Other)
( ) File ( ) WAIS ( ) Telnet ( ) e-mail

URL to activate when this area is clicked: (required)

I_ICSP

[✓] Relative link

Target frame name/ID: (optional - used for FRAMES only)

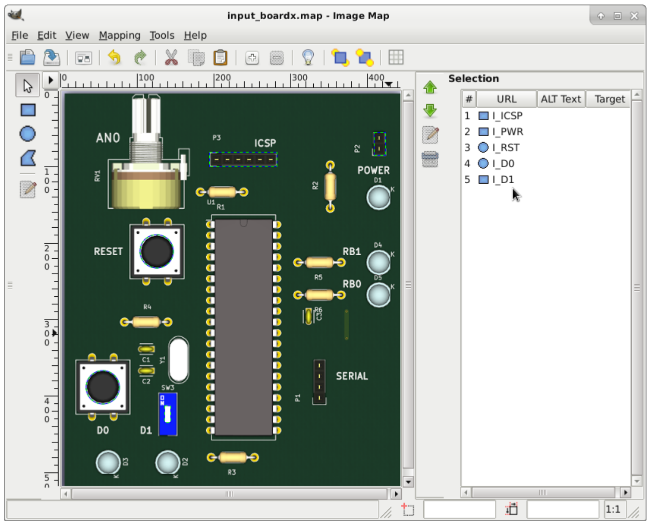
ALT text: (optional)

Help Apply Cancel OK

2.3.1 Input map

For this tutorial board, five input areas are marked:

- I_ICSP - where user click to load hexfile.
- I_PWR - where user click to turn on/off the board.
- I_RST - Button to reset board.
- I_D0 - Button connected in RD0.
- I_D1 - Switch connected in RD1.


input_boardx.map - Image Map
File Edit View Mapping Tools Help
AN0
ICSP
POWER
RESET
RB1
RB0
SERIAL
D0
D1
Selection

#	URL	ALT Text	Target
1	I_ICSP		
2	I_PWR		
3	I_RST		
4	I_D0		
5	I_D1		

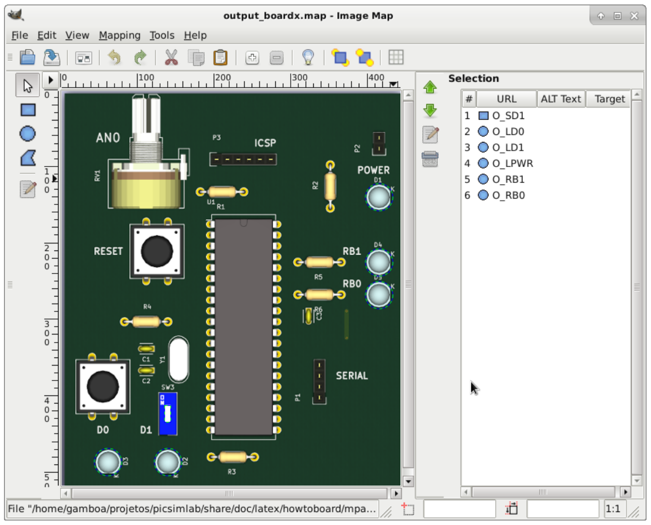
Input map generated by Gimp image map editor and saved as “share/input_boardx.map”.

```
<img src="[board_x](imported)" width="450" height="530" border="0" usemap="#map" />

<map name="map">
<!-- #$-:Image map file created by GIMP Image Map plug-in -->
<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->
<!-- #$-:Please do not edit lines starting with "#$" -->
<!-- #$VERSION:2.3 -->
<!-- #$AUTHOR:lcgamboa@yahoo.com   -->
<area shape="rect" coords="194,80,283,99" href="I_ICSP" />
<area shape="rect" coords="411,54,426,84" href="I_PWR" />
<area shape="circle" coords="125,211,22" href="I_RST" />
<area shape="circle" coords="54,390,22" href="I_D0" />
<area shape="rect" coords="135,414,143,436" href="I_D1" />
</map>
```

2.3.2 Output map

For this tutorial board, six output areas are marked:

- O_SD1 - draw the switch on/off.
- O_LD0 - draw LED connected in button.
- O_LD1 - draw LED connected in switch.
- O_LPWR - draw power LED indicator.
- O_RB0 and O_RB1 - draw LEDs connected in RB0 and RB1.


output_boardx.map - Image Map
File Edit View Mapping Tools Help
Selection
ANO
ICSP
POWER
RESET
RB1
RB0
SERIAL
D0
D1

#	URL	ALT Text	Target
1	O_SD1		
2	O_LD0		
3	O_LD1		
4	O_LPWR		
5	O_RB1		
6	O_RB0		

File "/home/gamboa/projetos/picsimlab/share/doc/latex/howtoboard/mpa... 1:1

Output map generated by Gimp image map editor and saved as “share/output_boardx.map”.

```
<img src="[board_x](imported)" width="450" height="530" border="0" usemap="#map" />

<map name="map">
<!-- #$-:Image map file created by GIMP Image Map plug-in -->
<!-- #$-:GIMP Image Map plug-in by Maurits Rijk -->
<!-- #$-:Please do not edit lines starting with "#$" -->
<!-- #$VERSION:2.3 -->
<!-- #$AUTHOR:lcgamboa@yahoo.com   -->
<area shape="rect" coords="135,414,143,436" href="O_SD1" />
<area shape="circle" coords="61,489,17" href="O_LD0" />
<area shape="circle" coords="140,489,17" href="O_LD1" />
<area shape="circle" coords="418,140,17" href="O_LPWR" />
<area shape="circle" coords="418,226,17" href="O_RB1" />
<area shape="circle" coords="418,269,17" href="O_RB0" />
</map>
```

2.4 Board code

The header file and c++ code file with comments are listed in the next two subsections. This files control the behavior of board in simulator.

2.4.1 board_x.h

```
/* ########################################################################

   PICsimLab - PIC laboratory simulator

   ########################################################################

   Copyright (c) : 2015  Luis Claudio Gamboa Lopes

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2, or (at your option)
   any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

   For e-mail suggestions :  lcgamboa@yahoo.com
   ######################################################################## */

#ifndef BOARD_x_H
#define        BOARD_x_H

#include<lxrad/lxrad.h>

//new board class must be derived from board class defined in board.h
class cboard_x:public board
{
  private:
     int p_BT1;           //first board switch in RD0
     int p_BT2;           //second board switch in RD1
     unsigned int lm[40]; //pins mean value (for PWM outputs)

     //controls to be added in simulator window
     CScroll *scroll1; //scroll for analog input AN0
     CGauge *gauge1;   //gauge to show mean value of RB0
     CGauge *gauge2;   //gauge to show mean value of RB1
     CLabel *label1;   //label of scroll AN0
     CLabel *label2;   //label of gauge RB0
     CLabel *label3;   //label of gauge RB1

   public:
      //Constructor called once on board creation
      cboard_x(void);
      //Destructor called once on board destruction
      ~cboard_x(void);
      //Called ever 100ms to draw board
      void Draw(_pic *pic, CDraw *draw,double scale);
      //Return a list of board supported microcontrollers
      String GetSupportedDevices(void){return wxT("PIC18F4550,PIC16F877A,");};
      //Return the filename of board picture
      String GetPictureFileName(void){return wxT("board_x.png");};
      //Return the filename of board picture input map
      String GetInputMapFile(void){return wxT("input_boardx.map");};
      //Return the filename of board picture output map
      String GetOutputMapFile(void){return wxT("output_boardx.map");};
      //Reset board status
      void Reset(_pic *pic);
      //Event on the board
      void MouseButtonPress(_pic *pic, uint button, uint x, uint y,uint state);
      //Event on the board
      void MouseButtonRelease(_pic *pic, uint button, uint x, uint y,uint state);
      //Event on the board
      void KeyPress(_pic *pic, uint key, uint x, uint y,uint mask);
      //Event on the board
      void KeyRelease(_pic *pic, uint key, uint x, uint y,uint mask);
      //Called ever 1s to refresh status
      void RefreshStatus(_pic *pic);
      //Called to save board preferences in configuration file
      void WritePreferences(void);
      //Called whe configuration file load  preferences
      void ReadPreferences(char *name,char *value);
      //return the input ids numbers of names used in input map
      unsigned short get_in_id(char * name);
      //return the output ids numbers of names used in output map
      unsigned short get_out_id(char * name);
};

#endif        /* BOARD_x_H */
```

2.4.2 board_x.cc

```
/* ########################################################################

   PICsimLab - PIC laboratory simulator

   ########################################################################

   Copyright (c) : 2015  Luis Claudio Gamboa Lopes

   This program is free software; you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation; either version 2, or (at your option)
   any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

   For e-mail suggestions :  lcgamboa@yahoo.com
   ######################################################################## */

//include files
#include"picsimlab1.h"
#include"board_x.h"


/* ids of inputs of input map*/
#define I_ICSP        1  //ICSP connector
#define I_PWR        2  //Power button
#define I_RST        3  //Reset button
#define I_D0        4  //RD0 push button
#define I_D1        5  //RD1 switch

/* ids of outputs of output map*/
#define O_SD1        1  //switch position (On/Off)
#define O_LD0        2  //LED on RD0 push button
#define O_LD1        3  //LED on RD1 switch
#define O_LPWR        4  //Power LED
#define O_RB0        5  //LED on RB0 output
#define O_RB1        6  //LED on RB1 output

//return the input ids numbers of names used in input map
```

```
unsigned short
cboard_x::get_in_id(char * name)
{
  if(strcmp(name,"I_ICSP")==0)return I_ICSP;
  if(strcmp(name,"I_PWR")==0)return I_PWR;
  if(strcmp(name,"I_RST")==0)return I_RST;
  if(strcmp(name,"I_D0")==0)return I_D0;
  if(strcmp(name,"I_D1")==0)return I_D1;

  printf("Erro input '%s' don't have a valid id! \n",name);
  return -1;
};

//return the output ids numbers of names used in output map
unsigned short
cboard_x::get_out_id(char * name)
{

  if(strcmp(name,"O_SD1")==0)return O_SD1;
  if(strcmp(name,"O_LD0")==0)return O_LD0;
  if(strcmp(name,"O_LD1")==0)return O_LD1;
  if(strcmp(name,"O_LPWR")==0)return O_LPWR;
  if(strcmp(name,"O_RB1")==0)return O_RB1;
  if(strcmp(name,"O_RB0")==0)return O_RB0;

  printf("Erro output '%s' don't have a valid id! \n",name);
  return 1;
};

//Constructor called once on board creation
cboard_x::cboard_x(void)
{
  proc=P18F4550;  //default microcontroller if none defined in preferences
  ReadMaps();     //Read input and output board maps

  //controls propierties and creation
  //scroll1
  scroll1=new CScroll();
  scroll1->SetFOwner(&Window1);
  scroll1->SetName(wxT("scroll1_px"));
  scroll1->SetX(12);
  scroll1->SetY(273);
  scroll1->SetWidth(140);
  scroll1->SetHeight(22);
  scroll1->SetEnable(1);
  scroll1->SetVisible(1);
  scroll1->SetRange(100);
  scroll1->SetPosition(50);
```

```
scroll1->SetType(4);
Window1.CreateChild(scroll1);
//gauge1
gauge1=new CGauge();
gauge1->SetFOwner(&Window1);
gauge1->SetName(wxT("gauge1_px"));
gauge1->SetX(13);
gauge1->SetY(382);
gauge1->SetWidth(140);
gauge1->SetHeight(20);
gauge1->SetEnable(1);
gauge1->SetVisible(1);
gauge1->SetRange(100);
gauge1->SetValue(0);
gauge1->SetType(4);
Window1.CreateChild(gauge1);
//gauge2
gauge2=new CGauge();
gauge2->SetFOwner(&Window1);
gauge2->SetName(wxT("gauge2_px"));
gauge2->SetX(12);
gauge2->SetY(330);
gauge2->SetWidth(140);
gauge2->SetHeight(20);
gauge2->SetEnable(1);
gauge2->SetVisible(1);
gauge2->SetRange(100);
gauge2->SetValue(0);
gauge2->SetType(4);
Window1.CreateChild(gauge2);
//label1
label1=new CLabel();
label1->SetFOwner(&Window1);
label1->SetName(wxT("label1_px"));
label1->SetX(12);
label1->SetY(249);
label1->SetWidth(60);
label1->SetHeight(20);
label1->SetEnable(1);
label1->SetVisible(1);
label1->SetText(wxT("AN0"));
label1->SetAlign(1);
Window1.CreateChild(label1);
//label2
label2=new CLabel();
label2->SetFOwner(&Window1);
label2->SetName(wxT("label2_px"));
label2->SetX(12);
```

```
  label2->SetY(306);
  label2->SetWidth(60);
  label2->SetHeight(20);
  label2->SetEnable(1);
  label2->SetVisible(1);
  label2->SetText(wxT("RB0"));
  label2->SetAlign(1);
  Window1.CreateChild(label2);
  //label3
  label3=new CLabel();
  label3->SetFOwner(&Window1);
  label3->SetName(wxT("label3_px"));
  label3->SetX(13);
  label3->SetY(357);
  label3->SetWidth(60);
  label3->SetHeight(20);
  label3->SetEnable(1);
  label3->SetVisible(1);
  label3->SetText(wxT("RB1"));
  label3->SetAlign(1);
  Window1.CreateChild(label3);
};

//Destructor called once on board destruction
cboard_x::~cboard_x(void)
{
    //controls destruction
    Window1.DestroyChild(scroll1);
    Window1.DestroyChild(gauge1);
    Window1.DestroyChild(gauge2);
    Window1.DestroyChild(label1);
    Window1.DestroyChild(label2);
    Window1.DestroyChild(label3);
}

//Reset board status
void
cboard_x::Reset(_pic *pic)
{

    p_BT1=1;//set push button  in default state (high)

    //write button state to pic pin 19 (RD0)
    pic_set_pin(pic,19,p_BT1);
    //write switch state to pic pin 20 (RD1)
    pic_set_pin(pic,20,p_BT2);


  //verify serial port state and refresh status bar
#ifndef _WIN_
    if(pic->serialfd > 0)
#else
    if(pic->serialfd != INVALID_HANDLE_VALUE)
#endif
      Window1.statusbar1.SetField(2,wxT("Serial Port: ")+
        String::FromAscii(SERIALDEVICE)+wxT(":")+itoa(pic->serialbaud)+wxT("(")+
        String().Format("%4.1f",fabs((100.0*pic->serialexbaud-100.0*
        pic->serialbaud)/pic->serialexbaud))+wxT("%)"));
    else
      Window1.statusbar1.SetField(2,wxT("Serial Port: ")+
        String::FromAscii(SERIALDEVICE)+wxT(" (ERROR)"));


};

//Called ever 1s to refresh status
void
cboard_x::RefreshStatus(_pic *pic)
{
   //verify serial port state and refresh status bar
#ifndef _WIN_
    if(pic->serialfd > 0)
#else
    if(pic->serialfd != INVALID_HANDLE_VALUE)
#endif
      Window1.statusbar1.SetField(2,wxT("Serial Port: ")+
        String::FromAscii(SERIALDEVICE)+wxT(":")+itoa(pic->serialbaud)+wxT("(")+
        String().Format("%4.1f",fabs((100.0*pic->serialexbaud-100.0*
        pic->serialbaud)/pic->serialexbaud))+wxT("%)"));
    else
      Window1.statusbar1.SetField(2,wxT("Serial Port: ")+
        String::FromAscii(SERIALDEVICE)+wxT(" (ERROR)"));

};

//Called to save board preferences in configuration file
void
cboard_x::WritePreferences(void)
{
    char line[100];
    //write selected microcontroller of board_x to preferences
    Window1.saveprefs(wxT("px_proc"),getnamebyproc(proc,line));
    //write switch state of board_x to preferences
    Window1.saveprefs(wxT("px_bt2"),String::Format("%i",p_BT2));
};

```

```
//Called whe configuration file load  preferences
void
cboard_x::ReadPreferences(char *name,char *value)
{
     //read switch state of board_x of preferences
     if(!strcmp(name,"px_bt2"))
      {
        if(value[0] == '0')
          p_BT2=0;
        else
          p_BT2=1;
      }
    //read microcontroller of preferences
    if(!strcmp(name,"px_proc"))
    {
      proc=getprocbyname(value);
    }
};


//Event on the board
void
cboard_x::KeyPress(_pic *pic, uint key, uint x, uint y,uint mask)
{
  //if keyboard key 1 is pressed then activate button (state=0)
  if(key == '1')
  {
    p_BT1=0;
  }

  //if keyboard key 2 is pressed then toggle switch state
  if(key == '2')
  {
    p_BT2^=1;
  }

};

//Event on the board
void
cboard_x::KeyRelease(_pic *pic, uint key, uint x, uint y,uint mask)
{
  //if keyboard key 1 is pressed then deactivate button (state=1)
  if(key == '1')
  {
    p_BT1=1;
  }

```

```
};

//Event on the board
void
cboard_x::MouseButtonPress(_pic *pic, uint button, uint x, uint y,uint state)
{

  int i;

  //search for the input area which owner the event
  for(i=0;i<inputc;i++)
  {
    if(((input[i].x1 <= x)&&(input[i].x2 >= x))&&((input[i].y1 <= y)&&
       (input[i].y2 >= y)))
    {

      switch(input[i].id)
      {
        //if event is over I_ISCP area then load hex file
        case I_ICSP:
          Window1.menu1_File_LoadHex_EvMenuActive(NULL);
          break;
        //if event is over I_PWR area then toggle board on/off
        case I_PWR:
          if(Window1.Get_picpwr()) //if on turn off
          {
            Window1.Set_picrun(0);
            Window1.Set_picpwr(0);
            pic_reset(pic,1);
            Reset(pic);
            p_BT1=1;
            Window1.statusbar1.SetField(0,wxT("Stoped"));
          }
          else //if off turn on
          {
            Window1.Set_picpwr(1);
            Window1.Set_picrun(1);
            pic_reset(pic,1);
            Reset(pic);
            Window1.statusbar1.SetField(0,wxT("Running..."));
          }
          break;
        //if event is over I_RST area then turn off and reset
        case I_RST:
          if(Window1.Get_picpwr())//if powered
          {
            Window1.Set_picpwr(0);
            Window1.Set_picrst(1);
          }
          break;
        //if event is over I_D0 area then activate button (state=0)
        case I_D0:
          p_BT1=0;
          break;
        //if event is over I_D1 area then toggle switch state
        case I_D1:
          p_BT2^=1;
          break;
      }
    }
  }

};

//Event on the board
void
cboard_x::MouseButtonRelease(_pic *pic, uint button, uint x, uint y,uint state)
{
  int i;

  //search for the input area which owner the event
  for(i=0;i<inputc;i++)
  {
    if(((input[i].x1 <= x)&&(input[i].x2 >= x))&&((input[i].y1 <= y)&&
       (input[i].y2 >= y)))
    {
      switch(input[i].id)
      {
        //if event is over I_RST area then turn on
        case I_RST:
          if(Window1.Get_picrst())//if powered
          {
            Window1.Set_picpwr(1);
            Window1.Set_picrst(0);

            if(pic_reset(pic,-1))
            {
              Reset(pic);
            }
          }
          break;
        //if event is over I_D0 area then deactivate button (state=1)
        case I_D0:
          p_BT1=1;
          break;
      }
    }
  }

};


//Called ever 100ms to draw board
//This is the critical code for simulator running speed
void cboard_x::Draw(_pic *pic, CDraw *draw,double scale)
{
  int i;
  int j;
  unsigned char pi;
  const picpin * pins;

  int JUMPSTEPS = Window1.GetJUMPSTEPS(); //number of steps skipped
  long int NSTEPJ=Window1.GetNSTEPJ();  //number of steps in 100ms

  draw->Canvas.Init(scale,scale); //initialize draw context

//board_x draw
  for(i=0;i<outputc;i++) //run over all outputs
  {
    if(!output[i].r)//if output shape is a rectangle
    {
      if(output[i].id == O_SD1)//if output is switch
      {
       //draw a background white rectangle
       draw->Canvas.SetBgColor (255, 255, 255);
       draw->Canvas.Rectangle (1, output[i].x1, output[i].y1,
          output[i].x2-output[i].x1,output[i].y2-output[i].y1 );

       if(!p_BT2) //draw switch off
       {
         //draw a grey rectangle
         draw->Canvas.SetBgColor (70, 70, 70);
         draw->Canvas.Rectangle (1, output[i].x1,output[i].y1+
         ((int)((output[i].y2-output[i].y1)*0.35)),output[i].x2-output[i].x1 ,
         (int)((output[i].y2-output[i].y1)*0.65) );
       }
       else //draw switch on
       {
         //draw a grey rectangle
         draw->Canvas.SetBgColor (70, 70, 70);
         draw->Canvas.Rectangle (1, output[i].x1,
          output[i].y1,output[i].x2-output[i].x1 ,
          (int)((output[i].y2-output[i].y1)*0.65));
       };
```

```
        }
      }
      else //if output shape is a circle
      {

          draw->Canvas.SetFgColor (0, 0, 0);//black

          switch(output[i].id)//search for color of output
          {
            case O_LD0: //White using pin 19 mean value (RD0)
              draw->Canvas.SetColor (lm[18], lm[18], lm[18]);
              break;
            case O_LD1: //Yelllow using pin 20 mean value (RD1)
              draw->Canvas.SetColor (lm[19], lm[19], 0);
              break;
            case O_LPWR: //Blue using picpwr value
              draw->Canvas.SetColor(0,0,225*Window1.Get_picpwr()+30);
              break;
            case O_RB0: //Green using pin 33 mean value (RB0)
              draw->Canvas.SetColor (0, lm[32], 0);
              break;
            case O_RB1: //Red using pin 34 mean value (RB1)
              draw->Canvas.SetColor (lm[33],0 , 0);
              break;
          }

          //draw a circle
          draw->Canvas.Circle (1,output[i].x1, output[i].y1,output[i].r );
      };

    };

    //end draw
    draw->Canvas.End();
    draw->Update ();


    //reset mean value
    for(pi=0;pi < pic->PINCOUNT;pi++)
    {
      lm[pi]=0;
    };

   //read pic.pins to a local variable to speed up
   pins = pic->pins;


   j=JUMPSTEPS+1;//step counter
```

```
 if(Window1.Get_picpwr()) //if powered
   for(i=0;i<Window1.GetNSTEP();i++) //repeat for number of steps in 100ms
      {

        if(j > JUMPSTEPS)//if number of step is bigger than steps to skip
        {
          pic_set_pin(pic,19,p_BT1);//Set pin 19 (RD0) with button state
          pic_set_pin(pic,20,p_BT2);//Set pin 20 (RD1) with switch state
        }

        //verify if a breakpoint is reached if not run one instruction
        if(!mplabxd_testbp(pic))pic_step(pic,0);

        if(j > JUMPSTEPS)//if number of step is bigger than steps to skip
        {
          //increment mean value counter if pin is high
          for(pi=0;pi < pic->PINCOUNT;pi++)
          {
           lm[pi]+=pins[pi].value;
          }

          //set analog pin 2 (AN0) with value from scroll
          pic_set_apin(pic,2,((5.0*(scroll1->GetPosition()))/
            (scroll1->GetRange()-1)));

          j=0;//reset counter
        }
        j++;//counter increment
      }


      //RB0 mean value to gauge1
      gauge1->SetValue((100.0*lm[33])/NSTEPJ);
      //RB1 mean value to gauge2
      gauge2->SetValue((100.0*lm[32])/NSTEPJ);

      //calculate mean value
      for(pi=0;pi < pic->PINCOUNT;pi++)
      {
       lm[pi]= (int)(((225.0*lm[pi])/NSTEPJ)+30);
      }

};
```

2.5 Integration with PICsimLab

To integration of the new board in PICsimLab, are necessary edit two files.

The first is file boards_def.h. The changes to be made are:

1. The board_x.h header must be added (in line 36);
2. The definition BOARDS_LAST must be incremented, the original value 4 is incremented to 5 (in line 39);
3. The new case of switch for board_x must be added (in lines 59 to 61).

```
1  /* ########################################################################
2
3     PICsimLab - PIC laboratory simulator
4
5     ########################################################################
6
7     Copyright (c) : 2015  Luis Claudio Gamboa Lopes
8
9     This program is free software; you can redistribute it and/or modify
10    it under the terms of the GNU General Public License as published by
11    the Free Software Foundation; either version 2, or (at your option)
12    any later version.
13
14    This program is distributed in the hope that it will be useful,
15    but WITHOUT ANY WARRANTY; without even the implied warranty of
16    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17    GNU General Public License for more details.
18
19    You should have received a copy of the GNU General Public License
20    along with this program; if not, write to the Free Software
21    Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
22
23    For e-mail suggestions :  lcgamboa@yahoo.com
24    ######################################################################## */
25
26 #ifndef BOARDS_DEFS_H
27 #define        BOARDS_DEFS_H
28
29 #include"picsimlab1.h"
30
31 //includes of boards
32 #include"board_1.h"
33 #include"board_2.h"
34 #include"board_3.h"
35 #include"board_4.h"
36 #include"board_x.h"   //included for new board
37
38 //number of last board
39 #define BOARDS_LAST 5 //incremented for new board
```

```

//boards object creation
board * create_board(int *lab,int *lab_)
{
   board * pboard;
        switch(*lab)
        {
          case 1:
            pboard= new cboard_1();
            break;
          case 2:
            pboard= new cboard_2();
            break;
          case 3:
            pboard= new cboard_3();
            break;
          case 4:
            pboard= new cboard_4();
            break;
          case 5:                   //included for new board
            pboard= new cboard_x(); //included for new board
            break;                  //included for new board
          default:
            mprint(wxT("Invalid Board! Using Default!\n"));
            *lab=1;//default
            *lab_=1;//default
            Window1.combo2.SetText(wxT("1"));
            pboard= new cboard_1();
          break;
        }

    return pboard;
}

#endif        /* BOARDS_DEFS_H */
```

The second file is Makefile. The only change to be made is include object board_x.o in list (in line 18).

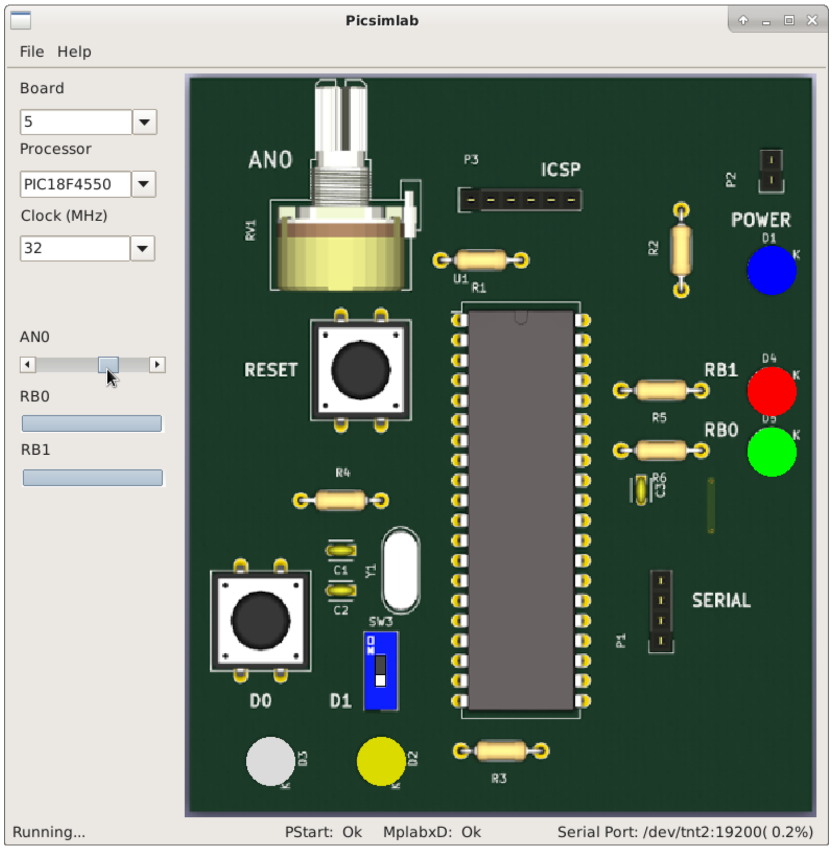
```
1  CC = g++
2
3  prefix=/usr
4
5  RM= rm -f
6  CP= cp
7  MKDIR = mkdir -p
8
9  appdir= ${prefix}/share/applications/
10 sharedir= ${prefix}/share/picsimlab/
```

```
execdir= ${prefix}/bin/

FLAGS = -D_VERSION_=\"0.6.0\" -Wall -O3 -D_SHARE_=\"${sharedir}\" \
        -fomit-frame-pointer `lxrad-config --cxxflags`

OBJS = ppicsimlab.o picsimlab1.o picsimlab2.o picsimlab3.o board.o \
       lcd.o mi2c.o rtc.o rtc2.o prog_psp.o board_1.o board_2.o \
       board_3.o board_4.o mplabxd.o board_x.o #new board object


all: $(OBJS)
        $(CC) $(FLAGS) $(OBJS) -opicsimlab -lpicsim `lxrad-config --libs`

%.o: %.cc
        $(CC) -c $(FLAGS) $<

install: all
        ${MKDIR} ${sharedir}
        $(CP) -dvf share/picsimlab.desktop ${appdir}
        $(CP) -drvf share/* ${sharedir}
        $(CP) -dvf picsimlab ${execdir}

uninstall:
        $(RM) -drvf ${sharedir}
        $(RM) -dvf ${execdir}picsimlab
        $(RM) -dvf ${appdir}picsimlab.desktop


clean:
        $(RM) picsimlab *.o core
```

After change this two files and include the five files created for new board, the PICsimLab can be recompiled, as described in first chapter.

2.6 Final Result

The PICsimLab board created for this tutorial are shown in the figure below.



Board: 5
Processor: PIC18F4550
Clock (MHz): 32
Running...
PStart: Ok MplabxD: Ok Serial Port: /dev/tnt2:19200(0.2%)

The sample program below can be used to test new board, this code is write for XC8 compiler:

```c
#include <xc.h>;

#include "config_4550.h"
#include "adc.h"
#include "serial.h"
#include "itoa.h"

void main()
{
  unsigned int  val;
  char buffer[10];

  ADCON1=0x02;
  TRISA=0xFF;
  TRISB=0xFC;
  TRISC=0xBF;
  TRISD=0xFF;
  TRISE=0x0F;

  adc_init();
  serial_init();


  while(1)
  {
      val=adc_amostra(0);

      if(PORTDbits.RD1)
      {
        if(val > 340)
           PORTBbits.RB0=1;
        else
          PORTBbits.RB0=0;

        if(val > 680)
          PORTBbits.RB1=1;
        else
          PORTBbits.RB1=0;
      }
      else
      {
          if(PORTDbits.RD0)
          {
              PORTBbits.RB0=1;
              PORTBbits.RB1=0;
          }
          else
          {
              PORTBbits.RB0=0;
              PORTBbits.RB1=1;
          }

      }

      serial_tx_str(itoa(val,buffer));
      serial_tx_str("\r\n");
  }

}
```

Chapter 3

License