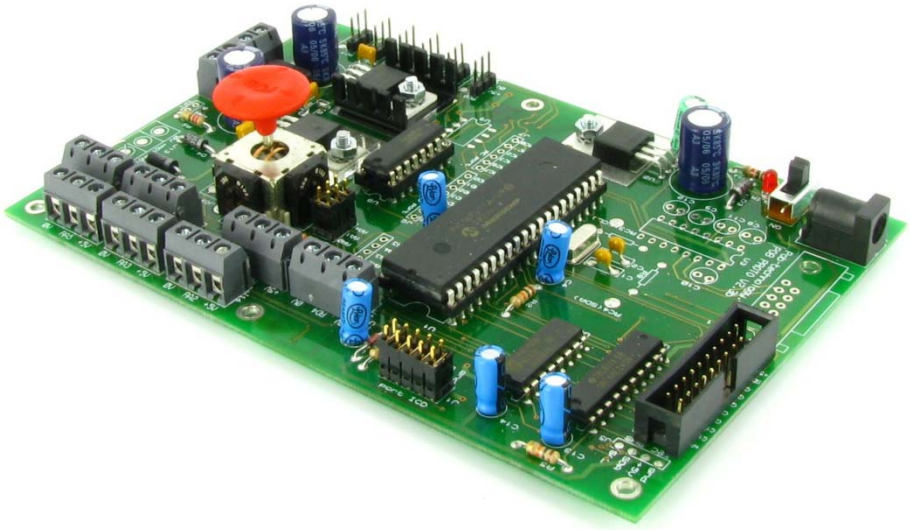
POB TECHNOLOGY

EXEMPLES DE CODE
POB-EYE ET POB-PROTO



Aout 2009 | POB Technology

TABLE DES MATIERES

INFORMATIONS LEGALES

Entreprise	**POB-Technology**
Type	SARL
SIRET	483 594 386 000 28
APE	742C

CONTACTS

Adresse	**POB-Technology** **11, avenue Albert Einstein** **69 100 VILLEURBANNE** **France**
Adresse mail	contact@pob-technology.com
Téléphone	+33 (0)4 72 43 02 36
Fax	+33 (0)4 83 07 50 89

GESTION DU DOCUMENT

Nom de Fichier	**Ezamples_for_POB_bot_fr.pdf**
Date de création	03/08/2009
Auteur	Alisson Foucault
Modification	03/08/2009

1. GENERALITE

1.1 LA LIBRAIRIE POB

Les programmes POB sont développés à partir d'une librairie propre à nos produits, facilitant leur mise en œuvre. L'essentiel des fonctions nécessaires à la programmation de votre robot s'y trouvent. Vous trouverez tous les renseignements nécessaires dans le fichier API Documentation du répertoire de votre Pob-tools. Il est important de vous familiariser avec la librairie POB afin d'optimiser votre utilisation du POB-bot, c'est pourquoi nous vous incitons fortement à le faire à travers ces quelques exemples.

1.2 MATERIEL ET OUTILS NECESSAIRES

Afin de réaliser les exemples présentés dans ce document, vous aurez besoin du matériel ci-dessous :

- Un POB-Bot et son câble série
- POB-Tools
- Paint ou un autre logiciel de traitement d'image
- Tera-term
- De quoi imprimer ou reproduire les formes à faire reconnaitre par votre robot

Figure 1.2.1 - POB-Eye

Figure 1.2.2 - POB-Proto

2. LIBRAIRIE D'EXEMPLES POB-EYE

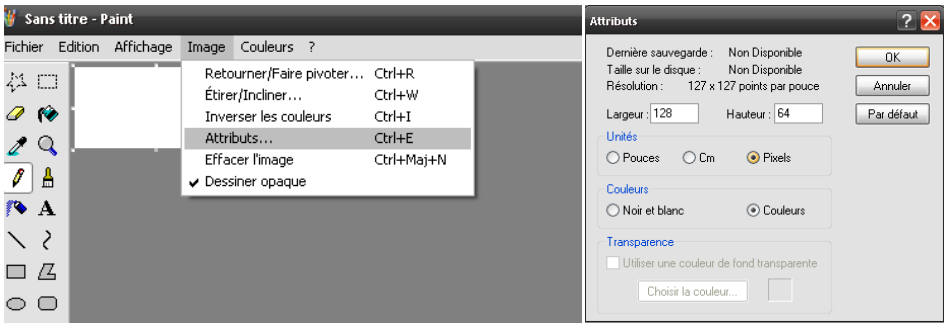
2. 1 HELLO WORLD

Traditionnellement le premier programme de toute chose, le Hello World permet ici au développeur d'afficher un fichier image sur l'écran LCD du robot POB.

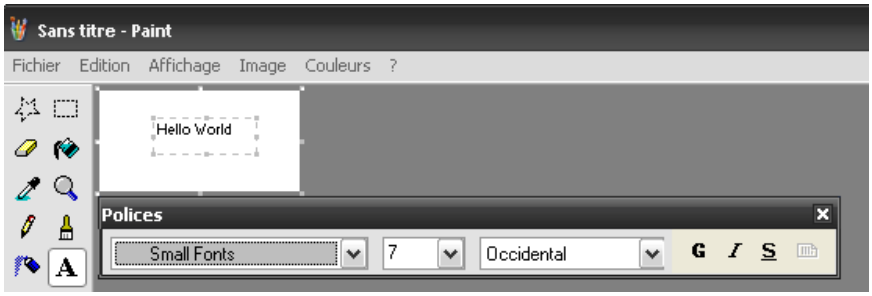
A. CREER LE FICHIER IMAGE

L'écran LCD Pob est apte à afficher des fichiers bitmaps 256 couleurs de résolution 128x64 pixels. Afin de procéder le plus facilement possible à la création de ce fichier, il est recommandé d'utilisé un logiciel très basique et gratuit capable de gérer tous ces paramètres : Paint.

Afin de dimensionner votre fichier une fois le logiciel lancé, cliquez sur Image, puis Attributs. Déclarez la largeur et la hauteur de votre fichier en pixels (respectivement 128 et 64), puis cliquez sur OK.



Il vous suffit maintenant d'utiliser l'outil texte et de sélectionner votre zone de travail afin d'y placer le commentaire de votre choix, « Hello World » dans le cas présent. Nous vous recommandons fortement l'utilisation de la police d'écriture Small Font, en taille 7, afin d'optimiser l'affichage sur l'écran LCD du robot.



Une fois le fichier fini, enregistrez le sous format bitmap (.bmp) 256 couleurs (et non 24 bits par défaut).

⚠ Attention, ne pas laisser d'espace dans le nom de votre bitmap.

Par la suite il vous faudra créer un fichier Bitmap.h à inclure dans votre code. Nous verrons comment dans la partie C abordant la compilation.

B. LE CODE

Il s'agit de quelques lignes très basiques. Dans un premier temps, il vous faudra inclure la librairie <pob-eye.h> ainsi que le fichier Bitmap.h que vous créerez un peu plus tard. Maintenant que vous avez accès aux fonctions POB, voyons leur utilité.

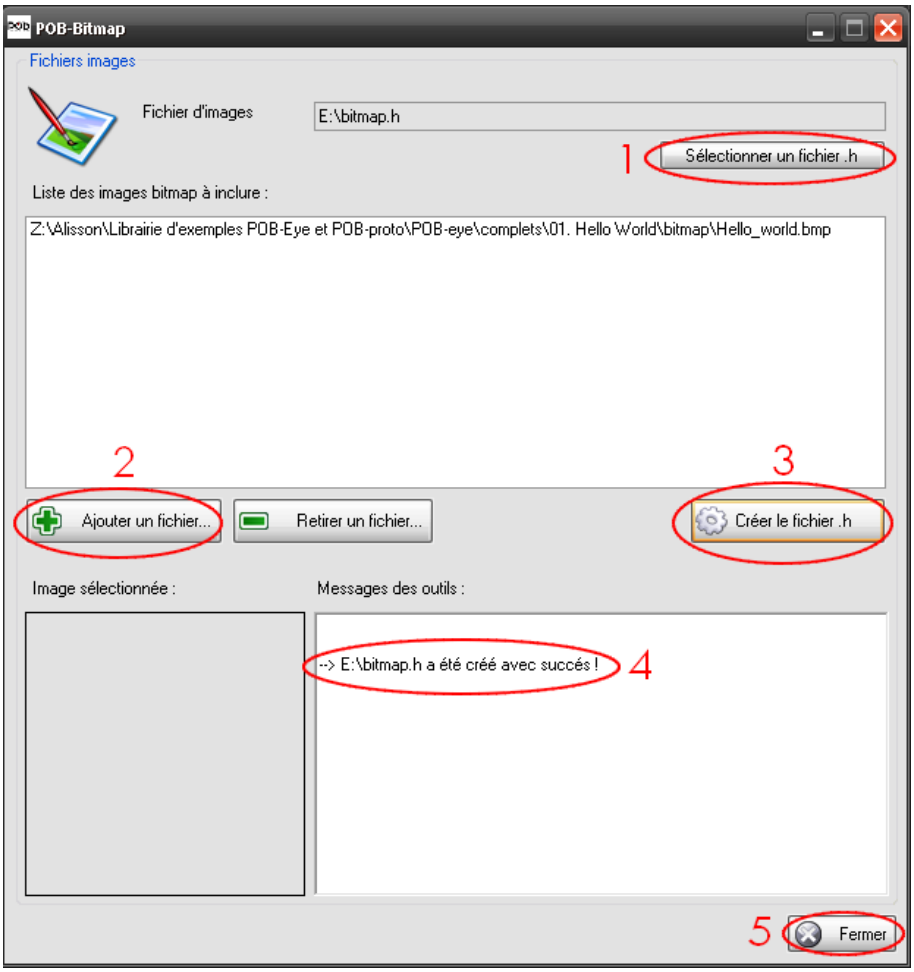
```
#include <pob-eye.h>
#include "Bitmap.h"  // include bitmap list

int main (void)
{
        //Initialize Buffer
        UInt8 LCD_Buffer [LCD_WIDTH*LCD_HEIGHT*BITS] ;
        GraphicBuffer ScreenBuffer ;

        //Initialize System
        InitPOBEYE();
        InitLCD();

        // Init the Graphic buffer with 128 per 64, one pixel per bit and LCD_Buffer
        InitGraphicBuffer( &ScreenBuffer, LCD_WIDTH,LCD_HEIGHT,ONE_BIT,LCD_Buffer);

        // Clear the graphic buffer
        ClearGraphicBuffer(&ScreenBuffer);

        //Draw the picture on the buffer
        DrawBitmap(0,0,IDB_HELLO_WORLD,bitmap,&ScreenBuffer);

        //Draw buffer in the LCD screen.
        DrawLCD(&ScreenBuffer);

        return 0;
}
```

Cette fonction suffit à afficher le bitmap hello_world.bmp sur l'écran LCD de votre robot. Vous pouvez aussi établir une forme de communication basique via port série entre votre POB-bot et votre PC en utilisant l'exemple *hello_world_text*, qui renvoit la chaine « Hello World » dans un terminal adéquat (Tera Term).

C. COMPILATION ET CHARGEMENT

Une fois votre fichier .c enregistré, il est temps de le compiler. Pour cela, ouvrez POB-tools et créez un nouveau projet (exemple : Hello_world.pobtools).

A *fichiers .h d'images*, faites *détail*, *sélectionner un fichier .h (1)* et créez en un que vous appellerez bitmap.h. Ajouter ensuite votre fichier bitmap « hello_world.bmp » (2), et créez le fichier .h (3). Un message confirmant la création de votre fichier apparait (4). Vous disposez maintenant de votre image en fichier .h. Place au fichier C.



Afin de compiler votre programme, toujours dans POB-tools, à *fichier de sortie .hex* , cliquez sur *détails*, *sélectionner un fichier .h*, par exemple hello_world.hex que vous créez, puis ajouter votre fichier .c et compilez. Votre programme est prêt à être chargé sur le POB-bot. Reste à connecter celui-ci à votre PC via le câble série, à sélectionner dans les options POB-tools le port COM correspondant, et à télécharger votre programme dans votre POB-bot en mode programmation.

Un reset plus tard, le POB-Bot affiche sur son écran le fameux « Hello World1 ».

2.2 DIALOGUER AVEC LE POB-EYE

Nous allons maintenant voir comment établir une communication entre le POB-bot et l'utilisateur, via un ordinateur et l'émulateur de terminal gratuit: Tera Term (http://ttssh2.sourceforge.jp/). Nous allons donc créer un programme ping_pong.c qui permet d'envoyer et de recevoir des informations octet par octet, sous la forme d'un écho de texte envoyé au POB-Eye.

A. LE CODE

Ce programme très simple ne comprend que quelques lignes de code.

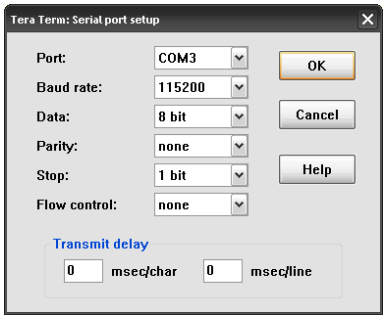
```
#include <pob-eye.h>

int main(void)
{
	//Initialize byte value
	UInt8 byte = 0;

	// Initialization of the POB-Eye and the Serial
	InitPOBEYE();
	InitUART((UInt16)(BR_115200|NO_PARITY|ONE_STOP_BIT|LENGTH_8_BIT));

	while (1)
	{
		//Get the byte from the UART
		byte = GetByteFromUART();

		//Send the byte to the UART
		SendByteToUART(byte);
	}
	return 0;
}
```

Comme toujours, on commence par initialiser les systèmes utilisés, dans ce cas de figure le POB-Eye et le port série. Puis le robot récupère chaque byte envoyé sur le port série afin de le renvoyer à son tour, ce qui crée une forme basique de communication via le port série.

B. CHARGER ET EXECUTER

De la même manière que pour l'exemple 1, compilez et chargez votre programme dans votre POB-bot via POB- Tools. Vous pouvez maintenant lancer Tera Term. Ce dernier requiert une petite configuration lors du lancement. Choisissez le mode Serial, et sélectionnez le port COM correspondant à votre connexion avec le POB-bot. Dans Setup, Serial port, mettez le taux de transfert à 115 200 puis validez.

Chacun des caractères que vous tapez dans le terminal Tera Term vous est renvoyé par le POB-bot.



2.3 AFFICHER L'IMAGE DE LA CAMERA SUR L'ECRAN

Dans cet exemple, il s'agit de récupérer l'image de la caméra du POB-Eye et de la retranscrire sur l'écran LCD.

CODE & TRANSFERT

```
#include <pob-eye.h>

int main (void)
{
	Int16 i=0,j=0,k=0,i_Cam=0,i_LCD=0;

	//LCD buffer, 128*64 pixels, 1 byte per pixel
	UInt8 LCD_Buffer [LCD_WIDTH*LCD_HEIGHT*BYTES];

	// Graphic buffer
	GraphicBuffer LCD_Screen;

	// Frame of camera
	RGBFrame FrameFromCam;

	//Initialize POB-Eye and LCD screen
	InitPOBEYE();
	InitLCD();

	// Init the graphic buffer : 128 per 64, 1 bytes per pixel and LCD_Buffer
	InitGraphicBuffer(&LCD_Screen,LCD_WIDTH,LCD_HEIGHT,EIGHT_BITS,LCD_Buffer);

	// Clear the graphic buffer
	ClearGraphicBuffer(&LCD_Screen);

	// Get the pointer of the red,green and blue video buffer
	GetPointerOnRGBFrame(&FrameFromCam);

	while (1)
	{
		// Grab the RGB components
		GrabRGBFrame();

		// Binary the RGB buffer
		BinaryRGBFrame(&FrameFromCam);

		// Put the pixel from the red component witch is binary, to the lcd buffer
		for (k=0,i=63 ; i ; i--)
		{
			for (j=0;j<120;j++,k++)
				LCD_Screen.buffer[k] = FrameFromCam.red[i+(j*88)];
			k+=8;
		}
		// draw the buffer on the screen
		DrawLCD(&LCD_Screen);
	}
	return 0;
}
```

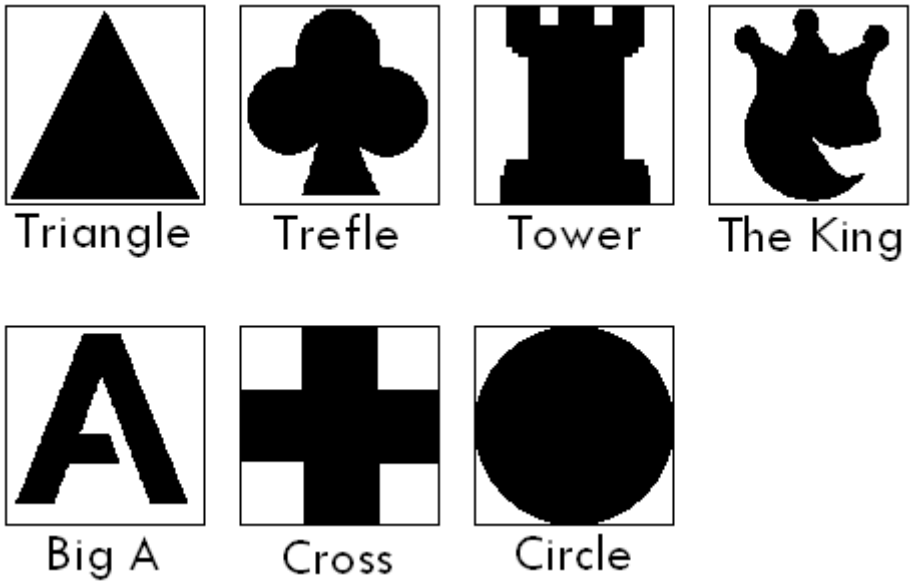
Après initialisation de tous les composants et de toutes les variables utilisées, on récupère l'image envoyée par la caméra que l'on envoi sous forme binaire à l'écran LCD qui la stocke et la dessine. Le programme se transfère de la même manière que celle évoquée ci-dessus.

2.4 RECONNAISSANCE DE FORMES

Cet exemple nous permet d'utiliser la caméra du POB-Eye afin de lui faire reconnaitre des formes préalablement définies par l'utilisateur.

LES FORMES

Il vous faut vous procurer ou concevoir vous-même un dictionnaire de formes distinctives pour le robot. Le programme fournit en exemple utilise les formes suivantes :

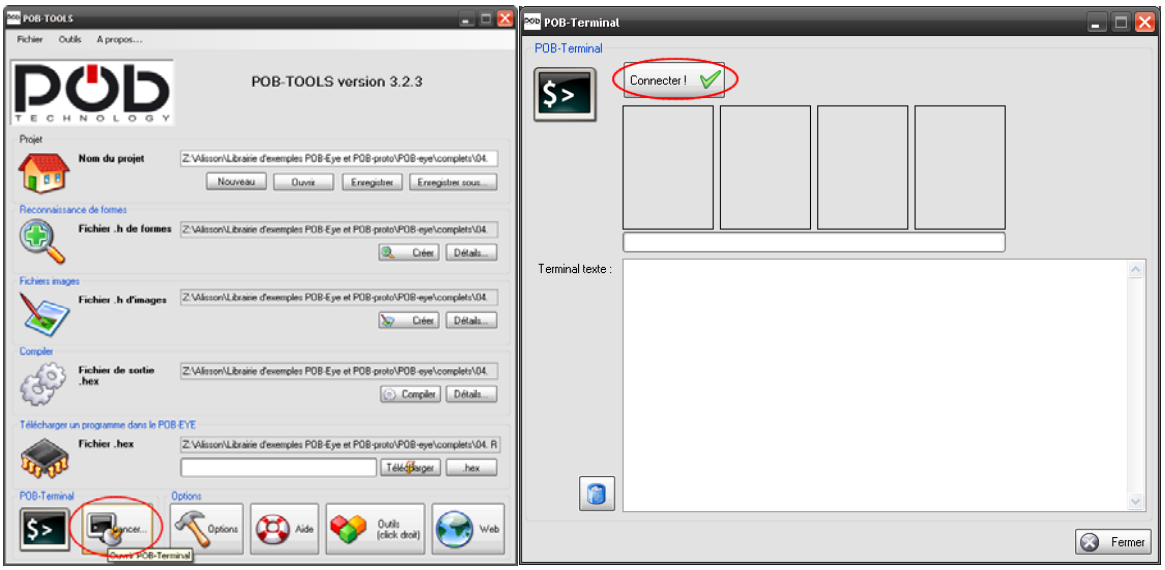


Tout comme les images à afficher sur un écran, les motifs à enregistrer doivent être au format bitmap. La résolution optimale est de 100x100 pixels. Afin d'améliorer la reconnaissance, utilisez un fond blanc derrière vos formes noires. De même, certaines formes, comme le cercle, ont tendance à se reproduire par contraste accru sur l'image de la caméra. Si vous souhaitez une détection précise, évitez ce genre de formes.

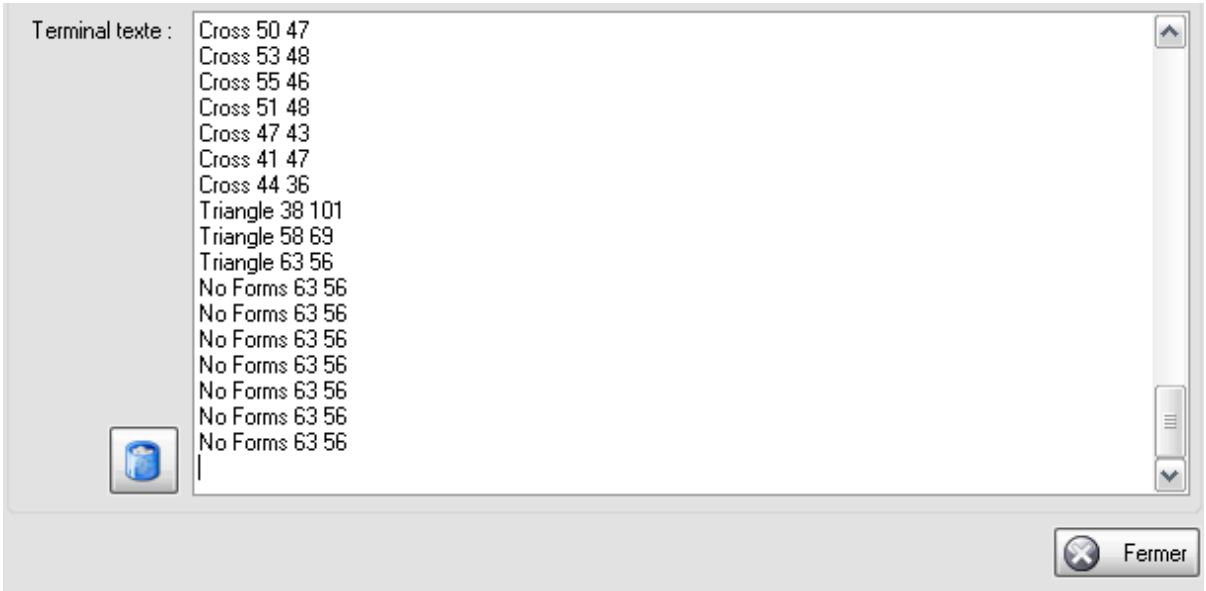
Une fois toutes vos formes créées, il ne vous reste plus qu'à créer le fichier .h de forme via Pob-tools. Lancez le logiciel. A *fichier .h de forme*, cliquez sur *détails,* puis *sélectionner un fichier .h*, et créez un nouveau fichier (exemple : pattern.h). Ajoutez tous vos bitmaps de forme, puis cliquez sur *créer le fichier .h*. Ce fichier sera à inclure dans votre programme.

L'AFFICHAGE

Afin de permettre à l'utilisateur de réaliser lorsque le robot reconnait une forme, le programme imprime le nom de la forme et sa position sur le terminal Pob-tools. Lorsque vous lancez votre robot en mode exécution, laissez le connecté à votre PC via le port série. Puis, dans Pob-tools, cliquez sur *Lancer le POB-Terminal,* puis sur *connecter !*.



Les formes reconnues s'affichent dans le terminal.



Vous pouvez aussi, comme dans l'exemple, utiliser des fichiers bitmaps pour afficher le nom, ou la forme, de la forme reconnue sur l'écran LCD de votre robot.

LE PROGRAMME

```
#include <pob-eye.h>

// Include bitmap list and dictionnary of forms
#include "Bitmap.h"
#include "pattern.h"

int main (void)
{
	UInt8 i=0,Nb_Identify=0;

	// List of form
	Form ListOfForm[MAX_OF_FORM];

	// Struct of three pointers on the RGB components
	RGBFrame FrameFromCam;

	// System and LCD screen initialization
	InitPOBEYE();
	InitLCD();

	// Get the pointer of the red,green and blue video buffer
	GetPointerOnRGBFrame(&FrameFromCam);

	//The pixels will be stocked in the LCD buffer
	UInt8 LCD_Buffer [LCD_WIDTH*LCD_HEIGHT*BITS] ;
	GraphicBuffer ScreenBuffer ;

	// Init the Graphic buffer with 128 per 64, one pixel per bit and LCD_Buffer
	InitGraphicBuffer( &ScreenBuffer, LCD_WIDTH,LCD_HEIGHT,ONE_BIT,LCD_Buffer);

	// clear the graphic buffer
	ClearGraphicBuffer(&ScreenBuffer);

	while(1)
	{
		// grab the RGB components
		GrabRGBFrame();

		// Binary the three RGB Buffer
		BinaryRGBFrame(&FrameFromCam);

		// Try to identify the forms and make a list of it
		Nb_Identify=IdentifyForm(&FrameFromCam,ListOfForm,pattern);

		// Parse the list of the form and print result on the Pob-Terminal and the LCD Screen
		for (i=0;i<Nb_Identify;i++)
		{
			switch (ListOfForm[i].id)
			{
			case IDP_0_CROSS:
				// Draw bitmap on the buffer and the LCD screen
				DrawBitmap(0,0,IDB_CROSS,bitmap,&ScreenBuffer);
				DrawLCD(&ScreenBuffer);
				// Print text on the terminal
				PrintTextOnPobTerminal("Cross %d %d",ListOfForm[i].x,ListOfForm[i].y);
			break;
```

```
                        case IDP_1_BIGA:
                                DrawBitmap(0,0,IDB_BIGA,bitmap,&ScreenBuffer);
                                DrawLCD(&ScreenBuffer);
                                PrintTextOnPobTerminal("A Big A %d %d",ListOfForm[i].x,ListOfForm[i].y);
                        break;

                        case IDP_2_KING:
                                DrawBitmap(0,0,IDB_KING,bitmap,&ScreenBuffer);
                                DrawLCD(&ScreenBuffer);
                                PrintTextOnPobTerminal("The King %d %d",ListOfForm[i].x,ListOfForm[i].y);
                        break;

                        case IDP_3_TOWER:
                                DrawBitmap(0,0,IDB_TOWER,bitmap,&ScreenBuffer);
                                DrawLCD(&ScreenBuffer);
                                PrintTextOnPobTerminal("Tower %d %d",ListOfForm[i].x,ListOfForm[i].y);
                        break;

                        case IDP_4_TREFLE:
                                DrawBitmap(0,0,IDB_BIGA,bitmap,&ScreenBuffer);
                                DrawLCD(&ScreenBuffer);
                                PrintTextOnPobTerminal("Trefle %d %d",ListOfForm[i].x,ListOfForm[i].y);
                        break;

                        case IDP_5_TRIANGLE:
                                DrawBitmap(0,0,IDB_TRIANGLE,bitmap,&ScreenBuffer);
                                DrawLCD(&ScreenBuffer);
                                PrintTextOnPobTerminal("Triangle %d %d",ListOfForm[i].x,ListOfForm[i].y);
                        break;
                        default:
                        break;
                        }
                }
        if (Nb_Identify == 0)
                {
                DrawBitmap(0,0,IDB_NOFORMS,bitmap,&ScreenBuffer);
                DrawLCD(&ScreenBuffer);
                PrintTextOnPobTerminal("No Forms %d %d",ListOfForm[i].x,ListOfForm[i].y);
                }
        }
        return 0;
}
```

Le programme commence comme d'ordinaire par une initialisation complète de l'ensemble des cartes et capteurs utilisés. Dans la boucle, on utilise la fonction IdentifyForm pour comparer l'image reçue par la caméra avec les formes à reconnaitre. Si une forme est identifiée, son nom est imprimée sur le terminal, ainsi que dessinée sur l'écran LCD via des fichiers bitmaps correspondant. (Bitmap.h)

2.5 RECONNAISSANCE DE FORMES AVEC SUIVI SUR L'ECRAN LCD

Ce programme a pour but non seulement de projeter les images perçues par la camera sur l'écran LCD, mais aussi de procéder en temps réel à l'analyse de forme, soit une combinaison des exemples 2.3 et 2.4. Pour cette raison, nous ne détailleront pas les méthodes, qui sont les mêmes, mais nous vous fourniront le code de cet exemple.

LE PROGRAMME

```
#include <pob-eye.h>

//bitmap and form list
#include "bitmap.h"
#include "pattern.h"


#define BUFFER_WIDTH64
#define BUFFER_HEIGHT          64

//function to draw picture on both screen, when a form is recognize
static void DrawWhatIsKnown(Form What);
//function to draw rectangle arround the form known
static UInt8 DrawRect (Rect Rectangle,const GraphicBuffer *Target);


UInt8 LCD_Left_Buffer [BUFFER_WIDTH*BUFFER_HEIGHT*BYTES];  // array to store the left lcd screen, one byte per pixel
UInt8 LCD_Right_Buffer [BUFFER_WIDTH*BUFFER_HEIGHT*BITS];  // array to store the right lcd screen one bit per pixel.

GraphicBuffer LCD_Left ;  // left graphic buffer
GraphicBuffer LCD_Right ; // right graphic buffer
UInt8 X_Pic=5,Y_Pic=5;    // x,y to draw pictures of form known on the right LCD screen

int main (void)
{
        Int16 i=0,j=0,k=0,tmp=0;

        // LCD Buffer
        UInt8 LCD_Buffer[BUFFER_WIDTH*2*BUFFER_HEIGHT*BITS];
        // Number of form
        Int16 Nb_Identify=0 ;
        // List of forms
        Form  ListOfForm[MAX_OF_FORM];
        // RGB Frame of camera
        RGBFrame FrameFromCam ;

        //Initialize POB-EYE and LCD Screen
        InitPOBEYE();
        InitLCD();

        // Init Left and Right screens buffer
        InitGraphicBuffer( &LCD_Left,BUFFER_WIDTH,BUFFER_HEIGHT,EIGHT_BITS,LCD_Left_Buffer);
        InitGraphicBuffer( &LCD_Right,BUFFER_WIDTH,BUFFER_HEIGHT,ONE_BIT,LCD_Right_Buffer);

        // Clear the Left and Right screen
        ClearGraphicBuffer(&LCD_Left);
        ClearGraphicBuffer(&LCD_Right);

        // Get the pointer of the red,green and blue video buffer
        GetPointerOnRGBFrame(&FrameFromCam);
```

```
	while(1)
	{
		// Init the x,y of the first picture
		X_Pic=8;
		Y_Pic=8;

		// draw a frame in right buffer
		DrawBitmap(0,0,IDB_7_FRAME,bitmap,&LCD_Right);

		// Grab the RGB components
		GrabRGBFrame();

		// Binary the three RGB Buffer
		BinaryRGBFrame(&FrameFromCam);

		// draw red component in left buffer
		DrawComponentInABufferVideo(FrameFromCam.red, &LCD_Left );

		// try to identify the forms and make a list of it.
		Nb_Identify=IdentifyForm(&FrameFromCam,ListOfForm,pattern);

		// Parse the list of the form
		for(i=0;i<Nb_Identify;i++)
			DrawWhatIsKnown(ListOfForm[i]);

		// Draw the twis LCDs screens
		DrawLeftLCD(&LCD_Left);			//the left for the real time video
		DrawRightLCD(&LCD_Right);	//the right for the result
	}
	return 0;
}
```

Deux fonctions appelées dans le main ne font pas partie de la librairie : DrawWhatIsKnow et DrawRect que voici .

```
void DrawWhatIsKnown (Form What)
{
	Rect Target;

	// draw the frame
	DrawBitmap(X_Pic,Y_Pic,IDB_8_LITTLE_FRAME,bitmap,&LCD_Right);

	// the picture of what is known in the frame
	DrawBitmap(X_Pic+3,Y_Pic+3,What.id -1,bitmap,&LCD_Right);

	//manage the (x,y) To draw the next known form on the right lcd screen
	X_Pic+=28;
	If ((X_Pic+28)>BUFFER_WIDTH)
	{
		Y_Pic+=26;
		X_Pic=8;
	}

	// draw a rectangle arround the known form on the left lcd screen

	Target.x=What.x - (What.width/2);		// manage the upper left corner of the form
	Target.y=What.y - (What.height/2);
	Target.width=What.width;
	Target.height=What.height;

	DrawRect(Target,&LCD_Left);			// Draw the rectangle arround the form known.
}
```

```
UInt8 DrawRect (Rect Rectangle,const GraphicBuffer *Target)
{
	UInt8 i=0,X=0,Y=0,W=2,H=2;
	UInt16 tmp=0,SizeOfBuffer=0;

	SizeOfBuffer=Target->width*Target->height;

	/*****************************************************************/
	/* we have to manage a ratio between the video buffer of the frame	*/
	/* from the cam (88*120 pixels), and the LCD screen (64*64 pixels)	*/
	/*****************************************************************/

	X=(Rectangle.x*186)>>8;
	Y=(Rectangle.y*135)>>8;
	W+=((Rectangle.width)*186)>>8;
	H+=((Rectangle.height)*135)>>8;


	//draw the 2 horizontals lines
	for (i=0;i<W;i++)
	{
		tmp=X+Y*Target->width+i;
		if (tmp>(SizeOfBuffer)) return 0;
		Target->buffer[tmp]=0xFF;

		tmp+=H*Target->width;
		if (tmp>(SizeOfBuffer)) return 0;
		Target->buffer[tmp]=0xFF;
	}

	//draw the 2 verticals lines
	for (i=0;i<H;i++)
	{
		tmp=X;
		tmp+=((i+Y)*Target->width);
		if (tmp>(SizeOfBuffer)) return 0;
		Target->buffer[tmp]=0xFF;

		tmp+=W;
		if (tmp>(SizeOfBuffer)) return 0;
		Target->buffer[tmp]=0xFF;
	}

	/*****************************************************************/
	/* Drawing Horizontal line or vertical with a "for" instruction */
	/* it's more faster than using the DrawLine function		*/
	/*****************************************************************/

	return 1;
}
```

3. LIBRAIRIE D'EXEMPLES POUR POB-PROTO

Après avoir eu un aperçu des fonctions réalisables sur le POB-Eye, voici un bref catalogue d'exemples pour POB-Proto.

3.1 INITIALISER LA POB-PROTO

Le premier exemple, et non le moindre, est l'initialisation de la POB-Proto, qui ne fait pas partie de la librairie. Afin de faciliter sa réutilisation nous vous conseillons par la suite de créer un fichier .c à part contenant les fonctions nécessaires que vous joindrez à chacun de vos programmes utilisant la POB-Proto.

LE CODE

```
#include <stdlib.h>
#include <pob-eye.h>

//Function to initialize the POB-PROTO board
void InitPobProto (void)
{
	// struct to set the pob-proto
	PobProto	Proto;

	//to get the position of the analogic joystick, you have to set the PORTA as analogic input
	Proto.porta=ALL_PORTA_AS_ANA;

	//all pin of PORTC are configured to manage servomotors
	Proto.portc=RC7_AS_SERVO	| RC6_AS_SERVO |RC3_AS_SERVO |RC2_AS_SERVO|RC1_AS_SERVO |RC0_AS_SERVO;

	//RD0 RD1 RD2 RD3 are configured as digitals output to gear DC motor, RD4 RD5 RD6 RD7 are configured as digitals input
	Proto.portd=RD7_AS_DI| RD6_AS_DI	|RD5_AS_DI |RD4_AS_DI|RD3_AS_DO	|RD2_AS_DO	|RD1_AS_DO
	|RD0_AS_DO;

	//set the pob proto
	SetPobProto(&Proto);
}

int main(void)
{
	//init POB-EYE and Serial
	InitPOBEYE();
	InitUART((UInt16)(BR_115200|NO_PARITY|ONE_STOP_BIT|LENGTH_8_BIT));

	//pob-proto init
	InitPobProto ();

	return 0;

}
```

La fonction InitPobProto dispose des ports A, C et D. Le port A obtient la position du joystick analogique. Le port C est dédié aux servomoteurs. Le port D regroupe les entrées et sorties digitales.

3.2 LIRE LA VALEUR DU JOYSTICK

Maintenant que la POB-Proto est initialisée, nous pouvons en découvrir l'exploitation possible via quelques exemples. Commençons par un programme qui permet de lire la position du joystick et de l'afficher.
Le joystick se déplace selon deux axes : horizontal et vertical. On utilise la fonction PrintTextOnPobLCD pour afficher les valeurs en temps réels, ainsi que l'axe auquel elles correspondent.

LE CODE

Commençons par vous fournir le code de la fonction PrintTextOnPobLCD, que nous vous recommandons de sauvegarder dans un fichier .c à son nom afin d'en refaire usage.

```
#include <pob-eye.h>
#include <stdlib.h>

#include "math.h"
#include "bitmap.h"

//array for display the graphic interface
static GraphicBuffer        LCD_Buffer_Video;
static UInt8                LCD_Buffer[LCD_WIDTH*LCD_HEIGHT*BITS];
static UInt8                ASCII_Buffer[LCD_WIDTH*LCD_HEIGHT*BITS]; // Buffer to stock the ascii table in bitmap format

// Set the bitmap  picture of the first 127 ASCII Char in ASCII_GRAPHIC_BUFFER
// ASCII_GRAPHIC_BUFFER will be used to get the pictures of characters to display on the LCD
void InitAsciiBuffer()
{
        GraphicBuffer ASCII_GRAPHIC_BUFFER;
        InitGraphicBuffer(&ASCII_GRAPHIC_BUFFER,LCD_WIDTH,LCD_HEIGHT,ONE_BIT,ASCII_Buffer);
        ClearGraphicBuffer(&ASCII_GRAPHIC_BUFFER);
        DrawBitmap(0,0, IDB_ASCII,bitmap,&ASCII_GRAPHIC_BUFFER); //bitmap is an array built by pob-tools
}

void PrintTextOnPobLCD(int row, int col, char *string, UInt8 *Screen_Buffer)
{
        int i,j,k=0; // i = ascii char starting buffer; j = 0 to 7 (8 times 8 points)

while(*string)
        {
                // starting point in the ASCII_Buffer of the char to display,128 (16charx8lines of pixels) is a complete row of text
                i = (string[0]/16)*128 + (string[0]%16);
        // display char on Screen_Buffer, 8 bytes
        for (j=0; j<8; j++)
                {
                  // 8 intergers to define a bitmap of a char
                  Screen_Buffer[col+row*128+k*16]=ASCII_Buffer[i+j*16];
                  k++;
                }
        string++;
        k=0;
        // 1 byte right
        col+=1;
        if (col%16==0)
                {
                // Manages end of line and go to the next line on the left
                row++;
                col=0;
                        // Go to the top of the screen if the bottom is reached
                        if (row%8 == 0)  row=0;
                }
        }
}
```

La fonction PrintTextOnPobLCD nécessite la conversion depuis le format bitmap au format .h afin d'effectuer l'affichage. L'image, tout comme le .h, vous sont tous deux fournis. Cette fonction accepte les arguments suivants :

• row : Ligne du premier caractère sur l'écran LCD(environs 1 à 8)

• col : Colonne du premier caractère sur l'écran LCD (environs 1 à 16)

• string : Chaine de caractères à afficher. Utilisez la fonction sprintf pour imprimer une chaine à partir d'entiers et de décimaux.

• Screen_Buffer : Buffer à afficher.

Ascii_Buffer est le buffer des 127 caractères de la table ASCII. Chaque caractère fait 8 pixels sur 8, et il est possible d'afficher 16 caractères par ligne, et 8 par colonne.

La fonction *InitPobProto* sera elle aussi employée pour la récupération de la valeur du joystick, et nous ne la détaillerons pas ici car elle était l'objet du précédent exemple. Le reste de la fonction fonctionne de la manière suivante :

Dans un premier temps, on initialise variables et systèmes, comme toujours. Puis on effectue un nettoyage de l'écran. Des variables récupèrent la valeur des axes horizontaux et verticaux, puis sont passées en chaines de caractère afin de pouvoir être affichées via la fonction PrintTextOnPobLCD. Elles sont pour finir affichées avec quelques informations complémentaires afin de clarifier leur position. On obtient ainsi un affichage en temps réel des coordonnées du joystick sur chaque axe de navigation. En cas de pression du joystick, un message s'affiche durant une seconde.

Le code de cette fonction est consultable ci-dessous. Les coordonnées ainsi récupérées pourront par la suite être utilisées, comme il sera fait dans le prochain exemple.

```
#include "pob-eye.h"
#include "pad.h"

void InitPobProto (void);
void IntroPOB(void);

// external functions declarations (for PrintTextOnPobLCD)
extern void InitAsciiBuffer();
extern void PrintTextOnPobLCD(int row, int col, char *string, UInt8 *Screen_Buffer);

int	main(void)
{
	//variable declaration (128 is the center of the LCD screen)
	int top_bottom_axe = 128;
	int right_left_axe = 128;
	int button = 0;
	char top_bottom[10];
	char right_left[10];

	//This buffers will stock the pixels to display,
	 GraphicBuffer		LCD_Buffer_Video;
	UInt8			LCD_Buffer[LCD_WIDTH*LCD_HEIGHT*BITS];
	// Buffer to stock the ascii table in bitmap format
	UInt8			 ASCII_Buffer[LCD_WIDTH*LCD_HEIGHT*BITS];

	//Initialize POB-EYE (lib), POB-LCD (lib) and POB-PROTO(source code at end of this file)
	InitPOBEYE();
	InitLCD();
	InitPobProto();
```

```
        // Initialize the Graphic buffer with 128 per 64, one pixel per bit and LCD_Buffer
        InitGraphicBuffer( &LCD_Buffer_Video, LCD_WIDTH,LCD_HEIGHT,ONE_BIT,LCD_Buffer);

        //Clear and draw the buffer to make clean the screen
        ClearGraphicBuffer(&LCD_Buffer_Video);
        DrawLCD(&LCD_Buffer_Video);

        // Init Ascii buffer, use to write in the LCD screen with PrintTextOnPobLCD function
        InitAsciiBuffer();

        while (1)
        {
                //get the values of pad
                top_bottom_axe = GetPortAnalog(UP_DOWN_AXE);
                right_left_axe = GetPortAnalog(RIGHT_LEFT_AXE);
                button = GetPortAnalog(BUTTON);

                if (button >=100)
                        {
                        ClearGraphicBuffer(&LCD_Buffer_Video);
                        //convert it from int to char*
                        sprintf(top_bottom, "%d", top_bottom_axe);
                        sprintf(right_left, "%d", right_left_axe);

                        //Display values
                        PrintTextOnPobLCD(1, 2, "Top", LCD_Buffer);
                        PrintTextOnPobLCD(2, 1, "Bottom", LCD_Buffer);
                        PrintTextOnPobLCD(5, 3, top_bottom, LCD_Buffer);
                        PrintTextOnPobLCD(1, 8, "Right", LCD_Buffer);
                        PrintTextOnPobLCD(2, 8, "Left", LCD_Buffer);
                        PrintTextOnPobLCD(5, 8, right_left, LCD_Buffer);
                        DrawLCD(&LCD_Buffer_Video);
                        }
                // If button is pressed
                if (button < 100)
                        {
                        ClearGraphicBuffer(&LCD_Buffer_Video);
                        DrawLCD(&LCD_Buffer_Video);
                        PrintTextOnPobLCD(4, 1, "Button pressed", LCD_Buffer);
                        DrawLCD(&LCD_Buffer_Video);
                        Wait(1000000);
                        ClearGraphicBuffer(&LCD_Buffer_Video);
                        DrawLCD(&LCD_Buffer_Video);
                        }
        }
 return 0;
}
```

3.3 BOUGER LE ROBOT AVEC LE JOYSTICK

Une fois ayant connaissance de la position du joystick, ses coordonnées peuvent trouver diverses utilités, notamment le contrôle des servomoteurs. Ci-dessous : deux exemples de manipulation de servomoteurs par Joystick. Le premier permet de faire se mouvoir les servomoteurs contrôlant la tête du POB-Bot, le second fera avancer votre robot en fonction de la position du joystick.

INCLINAISON DE LA TETE DU POB-BOT

A l'aide du Joystick analogique, vous pouvez maintenant incliner selon vos désirs la tête de votre POB-bot. La tête est théoriquement articulée sur 180 degrés, mais sa disposition limite cette angle à quelques degrés de moins. Afin de ne pas forcer inutilement sur des servomoteurs, nous partirons sur une base de mouvement du 30ème au 140ème degré. Tout comme au cours de l'exemple précédent, PrintTextOnPobTerminal et InitPobProto seront utilisés.

```
#include <string.h>
#include "pob-eye.h"
#include "pad.h"

// external functions declarations (for PrintTextOnPobLCD)
extern void InitAsciiBuffer();
extern void PrintTextOnPobLCD(int row, int col, char *string, UInt8 *Screen_Buffer);

int	main(void)
{
	//variable declaration (128 is the center of the LCD screen)
	int	top_bottom_axe = 128;
	int	right_left_axe = 128;
	char	top_bottom[10];
	char	right_left[10];
	int	position = 130;


	//This buffers will stock the pixels to display,
	 GraphicBuffer  LCD_Buffer_Video;
	UInt8			LCD_Buffer[LCD_WIDTH*LCD_HEIGHT*BITS];
	// Buffer to stock the ascii table in bitmap format
	UInt8			ASCII_Buffer[LCD_WIDTH*LCD_HEIGHT*BITS];

	//Initialize POB-EYE (lib), POB-LCD (lib) and POB-PROTO(file Functions.c)
	InitPOBEYE();
	InitLCD();
	InitPobProto();
	SwitchOnAllServo();

	// Initialize the Graphic buffer with 128 per 64, one pixel per bit and LCD_Buffer
	InitGraphicBuffer( &LCD_Buffer_Video, LCD_WIDTH,LCD_HEIGHT,ONE_BIT,LCD_Buffer);

	//Clear and draw the buffer to make clean the screen
	ClearGraphicBuffer(&LCD_Buffer_Video);
	DrawLCD(&LCD_Buffer_Video);
```

```
	while (1)
	{
		//get the values of pad
		top_bottom_axe = GetPortAnalog(UP_DOWN_AXE);
		right_left_axe = GetPortAnalog(RIGHT_LEFT_AXE);

		if (top_bottom_axe > 200)
		{
			if (position >= 30)
				position= position - 15;
			SetServoMotor(0, position);
		}
		else if (top_bottom_axe < 100)
		{
			if (position <=140)
				position = position + 15;
			SetServoMotor(0, position);
		}
		else if (right_left_axe >200)
			MoveBot(RIGHT);
		else if (right_left_axe < 100)
			MoveBot(LEFT);
		else
			MoveBot(STOP);

		//convert it from int to char*
		sprintf(top_bottom, "%d", top_bottom_axe);
		sprintf(right_left, "%d", right_left_axe);

		//display it
		PrintTextOnPobLCD(1, 2, "Top", LCD_Buffer);
		PrintTextOnPobLCD(2, 1, "Bottom", LCD_Buffer);
		PrintTextOnPobLCD(5, 3, top_bottom, LCD_Buffer);
		PrintTextOnPobLCD(1, 8, "Right", LCD_Buffer);
		PrintTextOnPobLCD(2, 8, "Left", LCD_Buffer);
		PrintTextOnPobLCD(5, 8, right_left, LCD_Buffer);
		DrawLCD(&LCD_Buffer_Video);
	}
return 0;

}
```

Un simple ajout de conditions relatives à la position du capteur permet ainsi de faire bouger les servomoteurs sur un axe.

MOUVEMENTS DU POB-BOT

De la même manière, le joystick peut aussi servir de contrôle directionnel au robot. Il suffit pour cela d'utiliser la position du joystick pour activer les servomoteurs du tank. Comme vous pouvez le constater par rapport à l'exemple ci-dessus, seuls quelques appels de fonctions changent, le reste du programme est strictement identique.

```
while (1)
{
        //get the values of pad
        top_bottom_axe = GetPortAnalog(UP_DOWN_AXE);
        right_left_axe = GetPortAnalog(RIGHT_LEFT_AXE);

        if (top_bottom_axe > 200)
                MoveBot(RUN);
        else if (top_bottom_axe < 100)
                MoveBot(BACK);
        else if (right_left_axe >200)
                MoveBot(RIGHT);
        else if (right_left_axe < 100)
                MoveBot(LEFT);
        else
                MoveBot(STOP);

         //convert it from int to char*
        sprintf(top_bottom, "%d", top_bottom_axe);
        sprintf(right_left, "%d", right_left_axe);

        //display it
        PrintTextOnPobLCD(1, 2, "Top", LCD_Buffer);
        PrintTextOnPobLCD(2, 1, "Bottom", LCD_Buffer);
        PrintTextOnPobLCD(5, 3, top_bottom, LCD_Buffer);
        PrintTextOnPobLCD(1, 8, "Right", LCD_Buffer);
        PrintTextOnPobLCD(2, 8, "Left", LCD_Buffer);
        PrintTextOnPobLCD(5, 8, right_left, LCD_Buffer);
        DrawLCD(&LCD_Buffer_Video);

}
```

En espérant que ces quelques exemples vous ont assisté dans votre familiarisation avec la libraire POB, et en vous souhaitant une bonne programmation.

CONTACTER POB-TECHNOLOGY

POB-TECHNOLOGY
11, avenue Albert Einstein
69 100 VILLEURBANNE
France

Site internet	http://www.pob-technology.com/
Adresse mail	contact@pob-technology.com
Téléphone	+33 (0)4 72 43 02 36
Fax	+33 (0)4 83 07 50 89