RPI_GPIO	Raspberry : Mise en œuvre des Ports Entrées / Sorties	
Système : Raspberry	Durée : 2 heures	Travail individuel

G.COLIN

Compétences : Tester et valider un module logiciel et matériel.

Contexte et démarche

On demande de mettre en œuvre les ports d'entrées / sorties sur la Raspberry. La programmation se fait en C/C++ et en utilisant l'IDE CODEBLOCK et la librairie WiringPi.

La démarche :

- Configurer Codeblock avec la librairie WiringPi
- Ecrire le programme permettant de faire clignoter une led sur une ligne de port.
- Câbler et tester le clignotement de la led.
- Ecrire un programme pour tester une ligne de port en entrée.
- Câbler et tester le programme avec des lignes programmées en entrée et en sortie

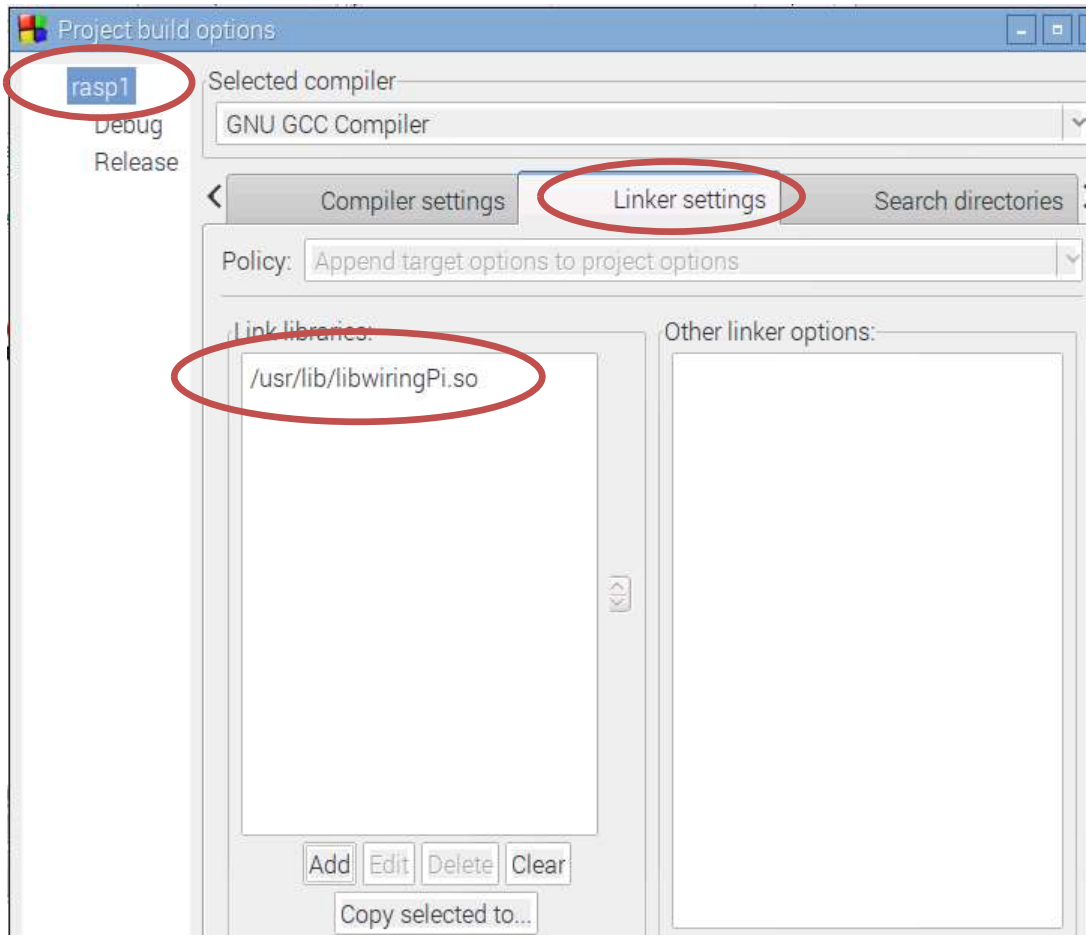
1 – Configuration de CodeBlock

⇨ La connexion sur la Raspberry se fait avec VNC, avec l'utilisateur pi (demander le mot de passe et l'IP).

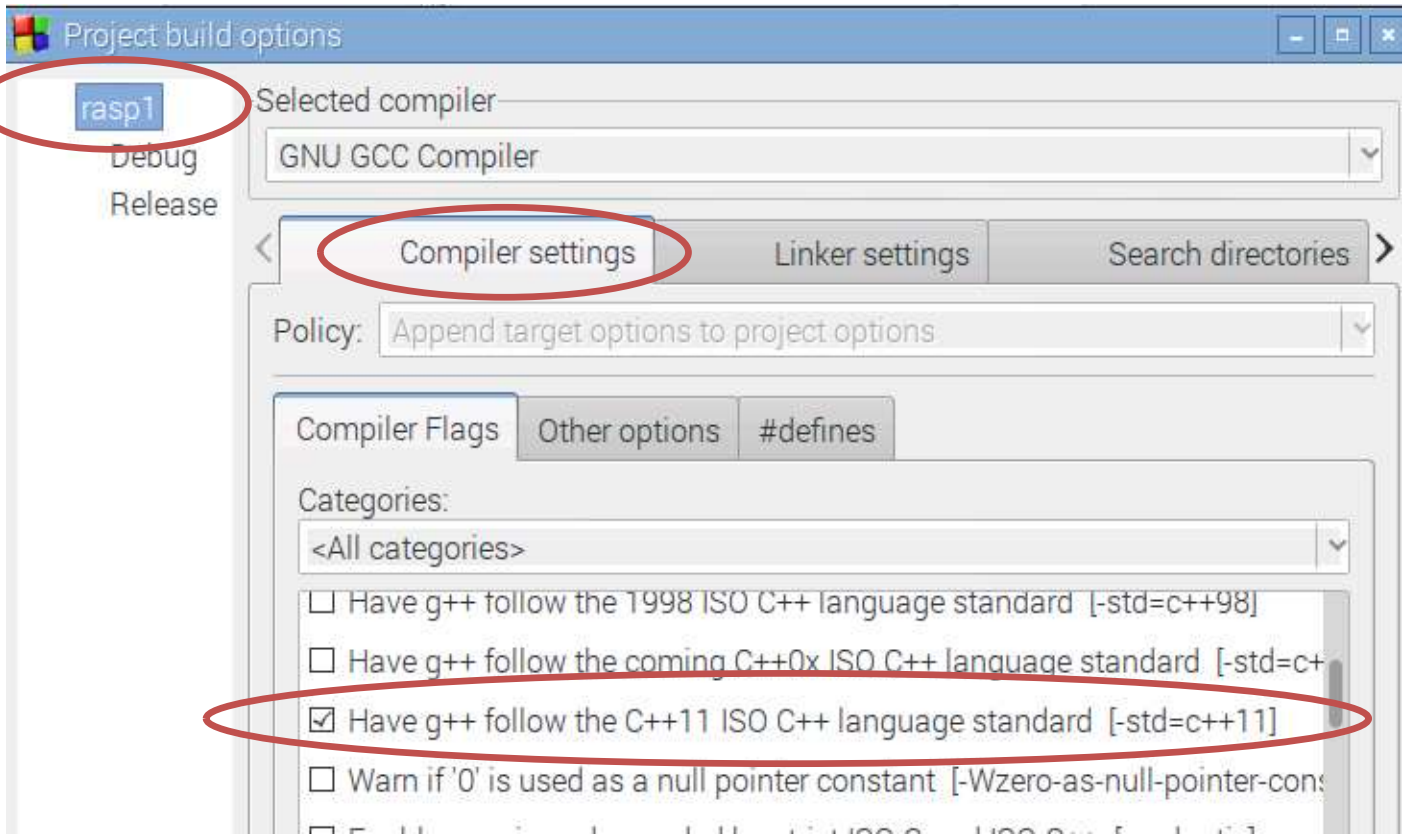
⇨ Créer un sous répertoire prog_cpp dans le répertoire pi de la raspberry

⇨ Créer un nouveau projet (en mode console) sous CodeBlocks. Choisir le langage C++. Le nommer rasp1 et le placer dans le répertoire prog_cpp créé.

⇨ Installer la librairie wiringPi dans le projet, comme ci-dessous (menu Project/Build options)



⇨ Choisir la version 2011 du compilateur (minimum ou 2014)



2 – Clignotement d'une led

Le programme principal est donné ci-dessous. La syntaxe de la librairie wiringPi reprend la syntaxe d'Arduino.

```
#include <iostream>

#include <wiringPi.h>

using namespace std;

int main (void)
{
    setenv("WIRINGPI_GPIOMEM", "1", 1);

    wiringPiSetup () ;

    pinMode (0, OUTPUT) ;

    while (1)
    {
        digitalWrite (0, HIGH) ;
        delay (100) ;
        digitalWrite (0,  LOW) ;
        delay (100) ;
    }
    return 0 ;
}
```

setenv donne les droits d'accès au GPIO à l'utilisateur pi (sinon il n'y a que root)
wiringPiSetup initialise les fonctions de la librairie.

La sortie 0 de wiringPi correspond à la broche 11, conformément au tableau ci-dessous.

BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
		3.3v			1 || 2			5v		
2	8	SDA.1	IN	1	3 || 4			5V		
3	9	SCL.1	IN	1	5 || 6			0v		
4	7	GPIO. 7	IN	1	7 || 8	1	ALT0	TxD	15	14
		0v			9 || 10	1	ALT0	RxD	16	15
17	0	GPIO. 0	IN	0	11 || 12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13 || 14			0v		
22	3	GPIO. 3	IN	0	15 || 16	0	IN	GPIO. 4	4	23
		3.3v			17 || 18	0	IN	GPIO. 5	5	24
10	12	MOSI	OUT	0	19 || 20			0v		
9	13	MISO	IN	0	21 || 22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23 || 24	1	OUT	CE0	10	8
		0v			25 || 26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27 || 28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29 || 30			0v		
6	22	GPIO.22	IN	1	31 || 32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33 || 34			0v		
19	24	GPIO.24	IN	0	35 || 36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37 || 38	0	IN	GPIO.28	28	20
		0v			39 || 40	0	IN	GPIO.29	29	21
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM

⇨ Proposer un schéma de câblage (led et résistance) pour faire clignoter une led .

⇨ Demander le matériel au prof et faire valider votre proposition.

⇨ Tester le programme

⇨ Proposer une modification du programme pour utiliser l'étiquette LED à la place du numéro de broche.

⇨ Modifier le programme pour tester la sortie patte 35 et 37 du connecteur.

⇨Faire constater le fonctionnement au prof.

3 – Test d'une ligne en entrée

⇨ Proposer un programme pour tester la patte 13 du connecteur en entrée (en allumant la led suivant l'état de cette entrée).

Par mesure de sécurité, cette patte 13 sera mise à 0V ou à 3.3V par l'intermédiaire d'une résistance de 1kΩ, pour éviter des courts circuits en cas de mauvaise programmation.

⇨ Demander le matériel au prof. Demander la méthode pour réaliser le test.

⇨ Tester le programme et faire constater au prof.

4 – Sauvegarde des programmes

⇨ A l'aide du logiciel FileZilla (client) , en utilisant le protocole SFTP – SSH File Transfert Protocol , copier le répertoire prog_cpp dans le répertoire de travail personnel sur H :

⇨ Supprimer le répertoire prog_cpp sur la Raspberry.

⇨ Faire constater au prof.