Création d'un pilote PWM
Pour la raspberry pi 2

La raspberry pi 2 dispose de deux canaux PMW (channel 1 et channel 2) permettant le pilotage de moteur ou de servomoteur. La PWM est configurables pour les deux canaux.

Pour configurer la modulation de largeur d'impulsions, trois étapes sont nécessaires :

- Programmer les broches du GPIO pour permettre l'utilisation de la PWM sur des broches externes
- configurer une horloge afin de la fournir au module PWM
- Programmer les différents registres du module PWM

1) Programmation des ports d'entrées sorties

La raspberry PI 2 dispose de 54 broches d'entrée sorties. Chaque broche peut être partagée par différents périphériques.

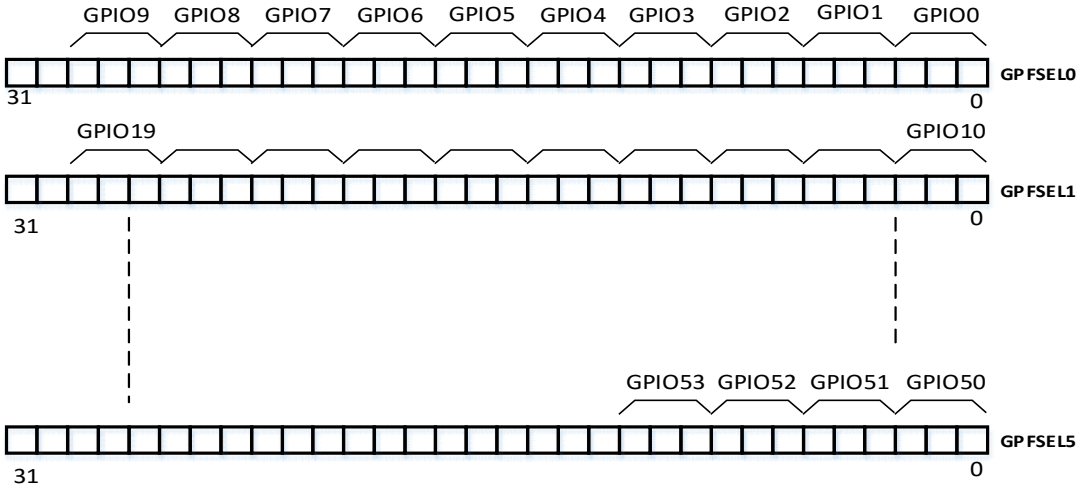
GPIO41	Low	PWM1	SD5	<reserved>	<reserved>	SPI2_MOSI	RXD1
GPIO42	Low	GPCLK1	SD6	<reserved>	<reserved>	SPI2_SCLK	RTS1
GPIO43	Low	GPCLK2	SD7	<reserved>	<reserved>	SPI2_CE0_N	CTS1
GPIO44	-	GPCLK1	SDA0	SDA1	<reserved>	SPI2_CE1_N	
GPIO45	-	PWM1	SCL0	SCL1	<reserved>	SPI2_CE2_N	
GPIO46	High	<Internal>					
GPIO47	High	<Internal>					
GPIO48	High	<Internal>					
GPIO49	High	<Internal>					
GPIO50	High	<Internal>					
GPIO51	High	<Internal>					
GPIO52	High	<Internal>					
GPIO53	High	<Internal>					

	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO0	High	SDA0	SA5	<reserved>			
GPIO1	High	SCL0	SA4	<reserved>			
GPIO2	High	SDA1	SA3	<reserved>			
GPIO3	High	SCL1	SA2	<reserved>			
GPIO4	High	GPCLK0	SA1	<reserved>			ARM_TDI
GPIO5	High	GPCLK1	SA0	<reserved>			ARM_TDO
GPIO6	High	GPCLK2	SOE_N / SE	<reserved>			ARM_RTCK
GPIO7	High	SPI0_CE1_N	SWE_N / SRW_N	<reserved>			
GPIO8	High	SPI0_CE0_N	SD0	<reserved>			
GPIO9	Low	SPI0_MISO	SD1	<reserved>			
GPIO10	Low	SPI0_MOSI	SD2	<reserved>			
GPIO11	Low	SPI0_SCLK	SD3	<reserved>			
GPIO12	Low	PWM0	SD4	<reserved>			ARM_TMS
GPIO13	Low	PWM1	SD5	<reserved>			ARM_TCK
GPIO14	Low	TXD0	SD6	<reserved>			TXD1
GPIO15	Low	RXD0	SD7	<reserved>			RXD1
GPIO16	Low	<reserved>	SD8	<reserved>	CTS0	SPI1_CE2_N	CTS1
GPIO17	Low	<reserved>	SD9	<reserved>	RTS0	SPI1_CE1_N	RTS1
GPIO18	Low	PCM_CLK	SD10	<reserved>	BSCSL SDA / MOSI	SPI1_CE0_N	PWM0
GPIO19	Low	PCM_FS	SD11	<reserved>	BSCSL SCL / SCLK	SPI1_MISO	PWM1
GPIO20	Low	PCM_DIN	SD12	<reserved>	BSCSL / MISO	SPI1_MOSI	GPCLK0
GPIO21	Low	PCM_DOUT	SD13	<reserved>	BSCSL / CE_N	SPI1_SCLK	GPCLK1
GPIO22	Low	<reserved>	SD14	<reserved>	SD1_CLK	ARM_TRST	
GPIO23	Low	<reserved>	SD15	<reserved>	SD1_CMD	ARM_RTCK	
GPIO24	Low	<reserved>	SD16	<reserved>	SD1_DAT0	ARM_TDO	
GPIO25	Low	<reserved>	SD17	<reserved>	SD1_DAT1	ARM_TCK	
GPIO26	Low	<reserved>	<reserved>	<reserved>	SD1_DAT2	ARM_TDI	
GPIO27	Low	<reserved>	<reserved>	<reserved>	SD1_DAT3	ARM_TMS	
GPIO28	-	SDA0	SA5	PCM_CLK	<reserved>		
GPIO29	-	SCL0	SA4	PCM_FS	<reserved>		
GPIO30	Low	<reserved>	SA3	PCM_DIN	CTS0		CTS1
GPIO31	Low	<reserved>	SA2	PCM_DOUT	RTS0		RTS1
GPIO32	Low	GPCLK0	SA1	<reserved>	TXD0		TXD1
GPIO33	Low	<reserved>	SA0	<reserved>	RXD0		RXD1
GPIO34	High	GPCLK0	SOE_N / SE	<reserved>	<reserved>		
GPIO35	High	SPI0_CE1_N	SWE_N / SRW_N		<reserved>		
GPIO36	High	SPI0_CE0_N	SD0	TXD0	<reserved>		
GPIO37	Low	SPI0_MISO	SD1	RXD0	<reserved>		
GPIO38	Low	SPI0_MOSI	SD2	RTS0	<reserved>		
GPIO39	Low	SPI0_SCLK	SD3	CTS0	<reserved>		
GPIO40	Low	PWM0	SD4		<reserved>	SPI2_MISO	TXD1
	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5

Six registres permettent de déterminer le périphérique qui va partager une des broches du GPIO.

Address	Field Name	Description	Size	Read/ Write
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	R/W
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	R/W

Pour chaque broche du GPIO trois bits déterminent le périphérique qui partage la broche



Chaque tribit permet de définir la fonction associée à chaque broche du GPIO :

000 = GPIO Pin 49 est une entrée
001 = GPIO Pin 49 est une sortie
100 = GPIO Pin 49 est une alternate function 0
101 = GPIO Pin 49 est une alternate function 1
110 = GPIO Pin 49 est une alternate function 2
111 = GPIO Pin 49 est une alternate function 3
011 = GPIO Pin 49 est une alternate function 4
010 = GPIO Pin 49 est une alternate function 5

Pour notre application on souhaite associer la broche GPIO19 au module PWM. Le tableau nous donne la fonction alternée 5 pour le module PWM. La programmation de GPIO19 se fait par l'intermédiaire du registre GPFSEL1.

La fonction suivante permet de programmer pour chaque broche du GPIO (g) les huit configurations possibles (a) :

```
void SET_GPIO_ALT(int g,int a) {

*(gpio+ (((g)/10))) |= (((a)<=3?(a)+4:(a)==4?3:2)<<(((g)%10)*3));

}
```

Gpio est un pointeur sur les périphériques du gpio.

On doit Remapper les adresses d'entrées sorties physiques dans l'espace d'adresse virtuelle du noyau.

gpio= ioremap(GPIO_BASE, 4096);

avec

#define BCM2709_PERI_BASE 0x3F000000 //pour le SOC BCM2836
#define GPIO_BASE (BCM2709_PERI_BASE + 0x200000) // GPIO contrôleur

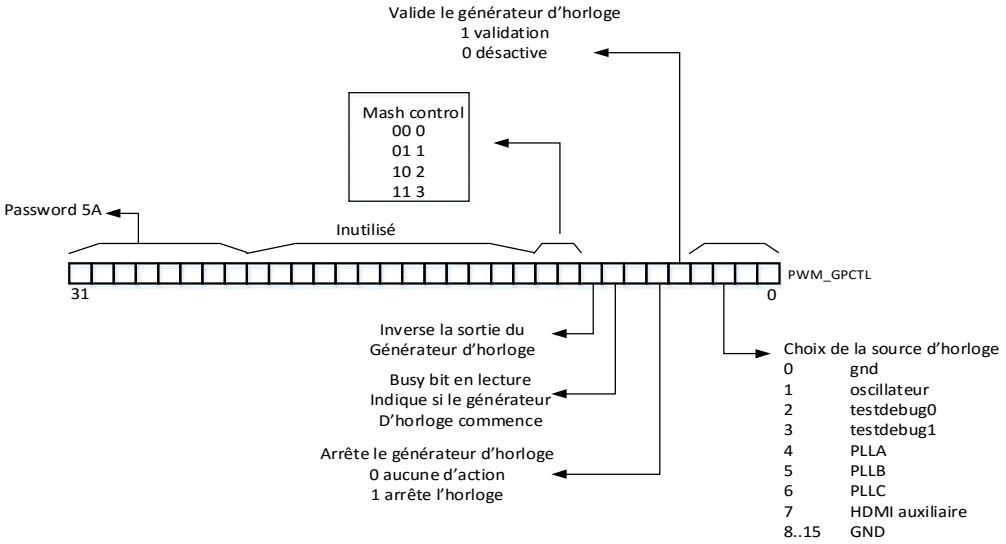
Le module PWM sans horloge ne pourrait fonctionner. On doit donc lui fournir une horloge

2) La génération de l'horloge

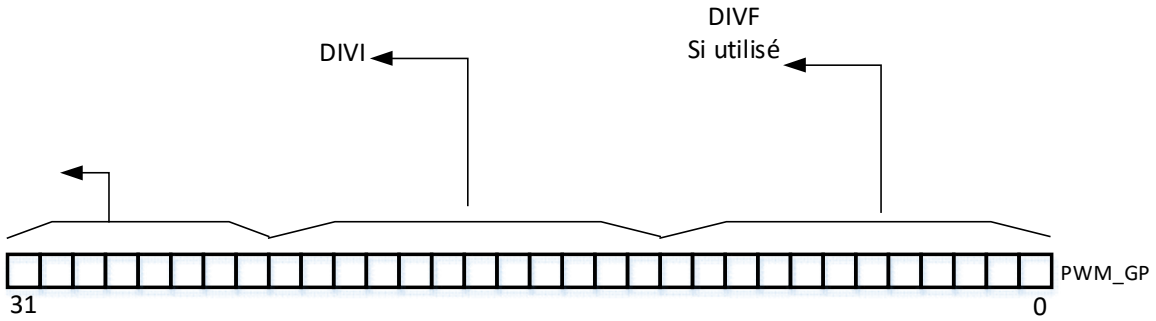
Il y a plusieurs sources possibles pour générer une horloge (oscillateur, PLL ….). Afin d'établir des fréquences très précises quatre modes de fonctionnement ont été choisis en utilisant deux diviseurs de fréquence. Le diviseur DIVI pour la partie entière et DIVF pour la partie fractionnaire :

MASH	min DIVI	min output freq	FREQ sortie moyenne	max output freq
0 (int divide)	1	source / (DIVI)	source / (DIVI)	source / (DIVI)
1	2	source / (DIVI)	source / (DIVI + DIVF / 1024)	source / (DIVI + 1)
2	3	source / (DIVI - 1)	source / (DIVI + DIVF / 1024)	source / (DIVI + 2)
3	5	source / (DIVI - 3)	source / (DIVI + DIVF / 1024)	source / (DIVI + 4)

Le registre de contrôle d'horloge PWM_GPCTL pour la PWM est donné ci-dessous



L'oscillateur a une fréquence de 19.2MHz . Les diviseurs de fréquence DIVI et DIVF se programment dans le registre PWM_GP



Pour une fréquence désirée de 1200Hz, on a le programme suivant :

```
// horloge validé mais source = gnd
*(clk1+ PWMCLK_CNTL)= 0x5A000000 | ( 1 << 5);// +40 CLK_PWM_CTL pour le channel 1 voir P 106 107
udelay(100);

        idiv= (int) (19200000.0f/ 16000.0f);//1200Hz
        idiv=192;
        if((idiv <1) || (idiv > 0x1000)){
        printk("idiv out of range: %x\n", idiv);
        return 0;
        }
        printk("set freq %d \n", idiv);

        *(clk1+ PWMCLK_DIV)= 0x5A000000 | ( idiv << 12);// programmation de la partie entière du diviseur de
frequence
        udelay(100);
        *(clk1+ PCMCLK_DIV)= 0x5A000000 | ( idiv << 12);// programmation de la partie entière du diviseur de
frequence
        udelay(100);
*(clk1+ PCMCLK_CNTL)= 0x5A000011;// valide l'horloge et l'oscillateur Mash=0
udelay(100);
```

Comme tout à l'heure clk1 est un pointeur :

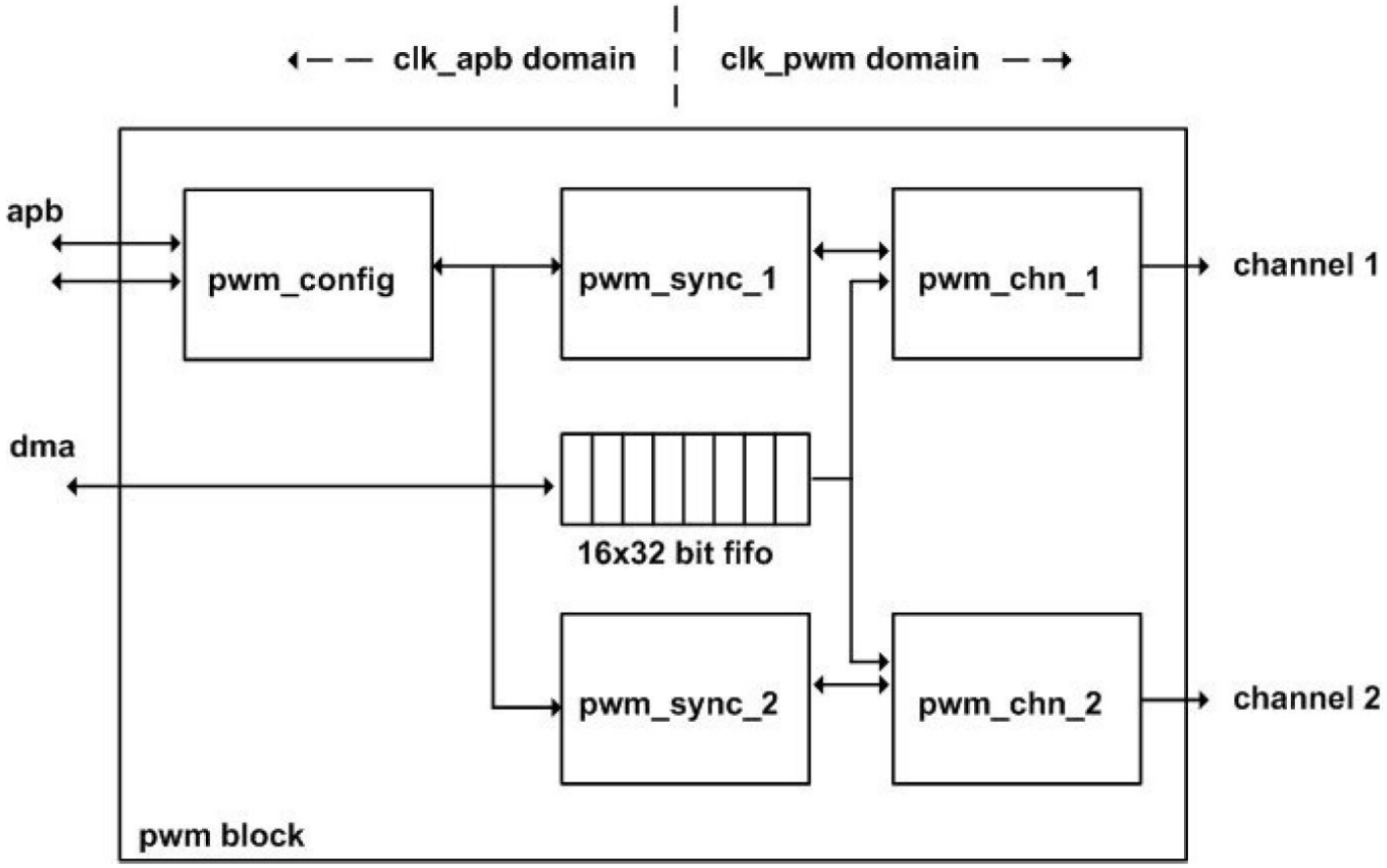
```
clk1= ioremap(CLOCK_BASE, 4096);
```

avec

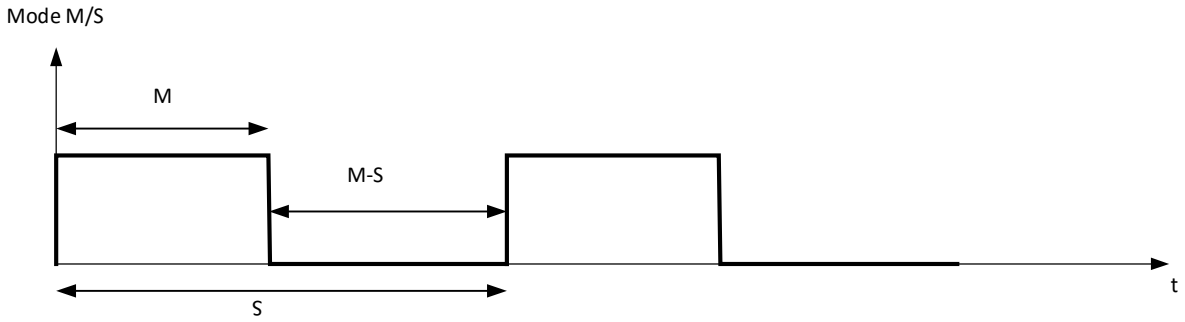
```
#define BCM2709_PERI_BASE 0x3F000000
#define CLOCK_BASE (BCM2709_PERI_BASE + 0x101000)
```

3) Le module PWM

Le module pwm permet de générer une modulation de largeur d'impulsion dont le rapport cyclique est programmable. Il est possible de générer grâce à une pile FIFO et les canaux DMA des ondes de forme complexe afin de réduire au maximum le taux d'harmonique. Ce module est constitué de deux canaux PWM (channel 1 et channel 2)

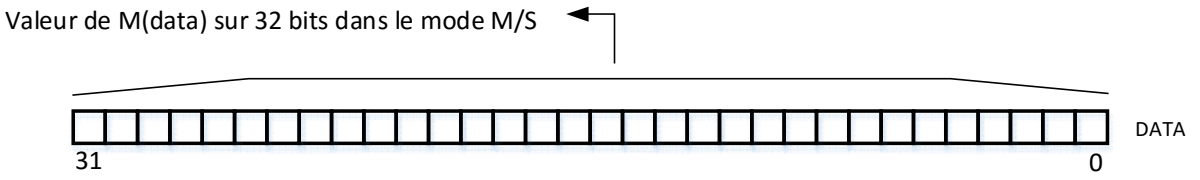


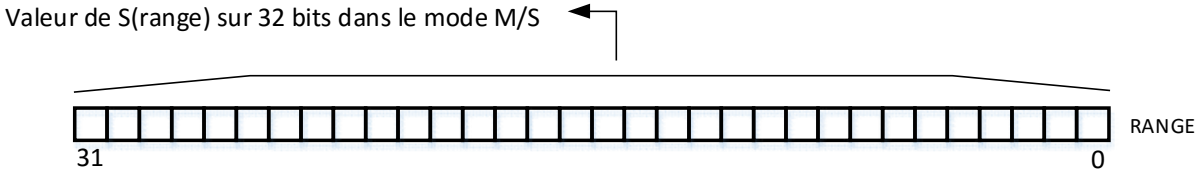
L'horloge est commune aux deux canaux. On peut travailler suivant deux modes le mode M/S (PWM naturel) et le mode PWM. Le mode M/S est décrit par le chronogramme ci-dessous :



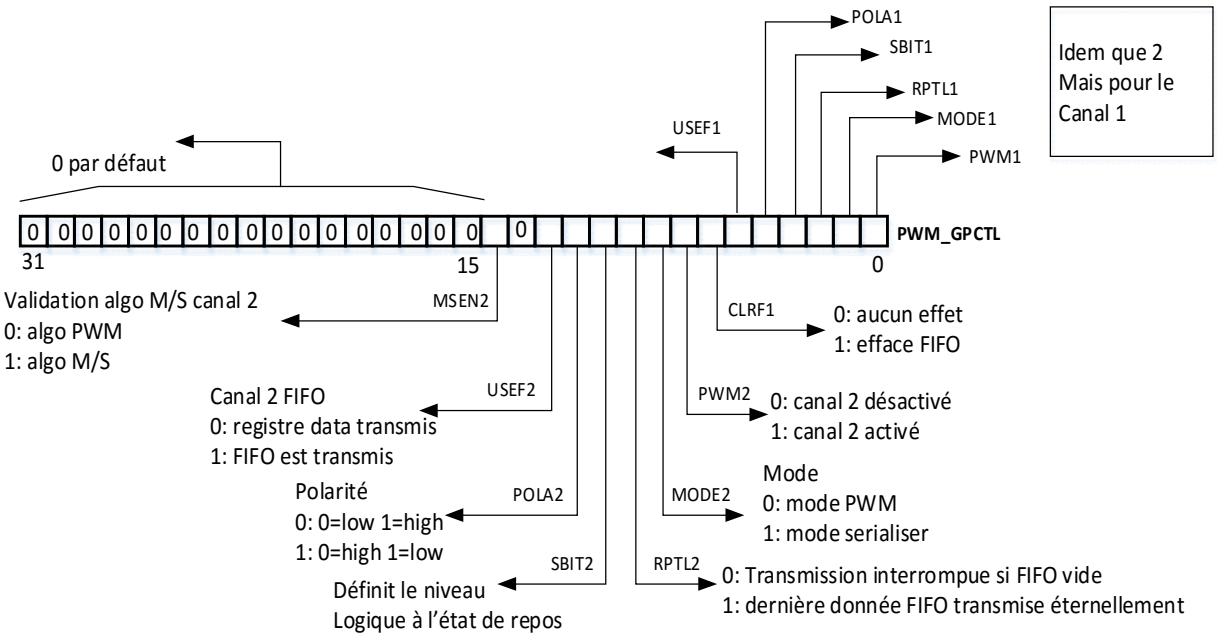
Le rapport cyclique est donc égal à M/S.

Les valeurs de M et S sont programmables par deux registres





Le point clé est la programmation du registre de contrôle PWM



La programmation reste la suivante :

```
pwm1= ioremap(PWM_BASE, 4096);
// désactive la pwm
*(pwm1+ PWM_CTL)= 0;
*(pwm1+ PWM_CTL)|= PWMCTL_CLRF;// efface la fifo
iowrite32(2000,pwm1+PWM_RNG1);// range
udelay(10);
iowrite32(1000,pwm1+PWM_DAT1);// data
udelay(10);
}
```

*(pwm1+ PWM_CTL)|= PWMCTL_PWEN1|PWMCTL_MSEN1 ; // Valide le mode M/S et le canal 1

Pwm1 est un pointeur sur les registres du module PWM

```
#define BCM2709_PERI_BASE 0x3F000000
#define PWM_BASE (BCM2709_PERI_BASE + 0x20C000)
#define PWM_CTL 0
#define PWM_RNG1 4
#define PWM_DAT1 5
#define PWMCTL_PWEN1 (1<<0)
#define PWMCTL_MSEN1 (1<<7)
```

Il nous reste à charger le module et observez la tension sur la broche GPIO19 :

