Capteur de distance infrarouge

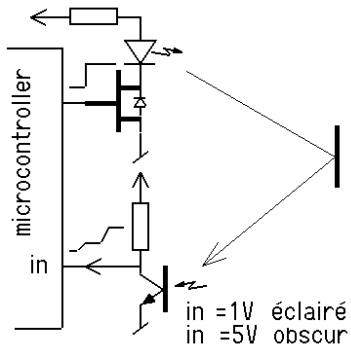
Voir www.didel.com/xbot/Dist2Ir.pdf, plus récent et complet

Les capteurs par réflexion infrarouge ont comme avantage d'être petits, bon marché et faciles à mettre en œuvre. Mais ils sont sensible à la lumière ambiante (surtout les spots) et sont difficiles à calibrer. Ils ne conviennent aussi que pour des courtes distances, qui dépendent de la taille du capteur, de son optique, de la puissance émise, de filtres éventuels. Des explication générales sont données sous www.didel.com/doc/sens/DocIr.pdf

Principe

Le principe d'un capteur par réflexion est d'éclairer l'obstacle avec une LED infrarouge, et de mesurer la lumière réfléchie avec une photodiode ou un phototransistor. L'objet éclairé retransmet une énergie inversément proportionnelle au carré de la distance.

Le schéma de câblage d'un détecteur d'obstacle est évident. Une résistance fixe le courant dans la LED qui éclaire l'obstacle. La résistance du photo-transistor est mesurée soit avec un diviseur de tension, soit par mesure du temps de charge ou décharge d'un condensateur, ce qui couvre une gamme de distance plus grande.



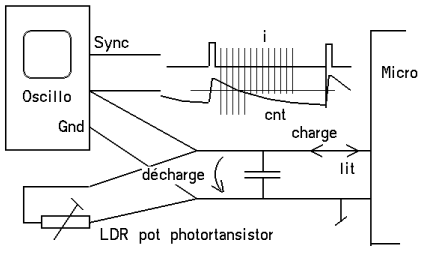
Une photodiode, un phototransistor, laissent passer un courant proportionnel à l'énergie lumineuse reçue. Dans l'obscurité totale, il y a un courant de fuite (dark current). Une diode travaille avec des niveaux de courants faibles (surtout les diodes de type PIN), alors qu'un phototransistors génère un courant plus élevé, tout en ayant un courant de fuite proportionnellement plus élevé. La solution d'emploi facile, avec de nombreux composants à disposition, est le phototransistor. Le LIT301/xx?515 a de meilleures performances puisque les diodes ont un boitier qui fait optique pour concentrer la lumière.

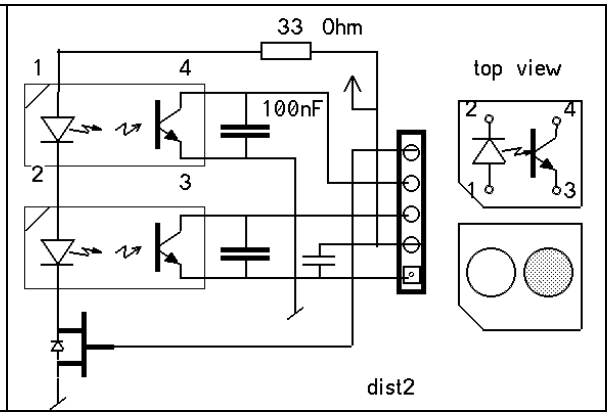
réf pyr

réf jd sol analo avec pot

Mesure par décharge d'un condo

La solution préférée pour la mesure est de charger le condensateur de l'interface en mettant la ligne en sortie du processeur, à l'état 1. On commute en entrée, et on mesure le temps de décharge, d'autant plus rapide qu'il y a plus de lumière. La valeur du condensateur est telle que le processeur compte avec suffisamment de précision. Dans l'obscurité, le condensateur se décharge pas, et un compteur limite la mesure à par exemple une valeur de 100. Si on compte toutes les 100us, la durée de la mesure est de 10 ms.



Le schéma inclut un transistor IRLML2502 dpour pulser le courant dans les leds. Avec une résistance de 33 Ohm le courant est de ~70 mA.		

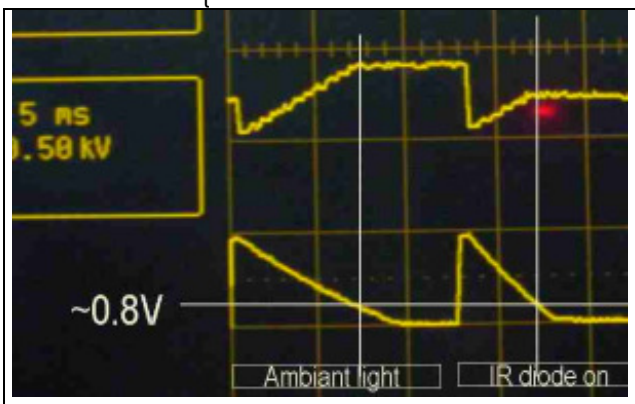
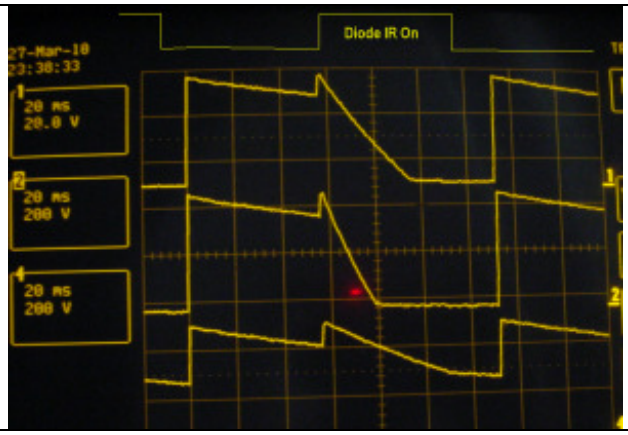
Pour diminuer l'effet de la lumière ambiante, on fait une première mesure sans éclairage IR, puis une 2e mesure avec éclairage, La différence corrige un peu, mais ce qui est important, c'est de vérifier que la lumière ambiante donne une valeur très différente. Si non, la capteur est mal dirigé ou doit être protégé par des caches bien placés.

Pour une mesure, on précharge 100 us, ou commute en entrée, et on lit la valeur qui décroit exponentiellement. Tant que le seul est supérieur à l'état 1 (environ 1.9Và 5V, 0.8V à 3V) on compte.

```
cnt = 0 ;
for (i=0; i<100; i++) {
delayMicroseconds (100);
if (Dist0 == On)  cnt0++ ;
distance0 = cnt0 ;
```

La distance max est de 100 ce qui fait une durée d'acquisition de 20ms.

```
{
```

	
Visualisation du compteur, de la décharge et du seuil	Le premier cycle a la diode d'éclairage IR éteinte, donc décharge lente. Le 2e cycle mesure la lumière réfléchie par l'obstacle.

La routine est facile à écrire. Après avoir chargé les condensateur on exécute une boucle toutes les 500 microsecondes (période d'interruption) dans laquelle on incrémente les compteurs associés à chaque capteur. Un décompteur détermine la valeur maximum mesurée (quand c'est obscur) et définit la durée de la mesure.

`//DistIrX.ino`
`#include "DefNewRobot.h"`
`#define bDisG = 0 //PORTC condos`
`#define bDisD = 1`
 	
`#define bIrLed = 2 //PORTC IrLed actif à 1`
`#define IrLedOn PORTC | 1<< bIrLed`
`#define IrLedOff PORTC & !(1<< bIrLed)`
 	
`#define DirCha  DDRC |= 1<<bDistG | 1<< bDistD | 1<< bIrLed`
`#define CapaCha PORTC |= 1<<bDistG | 1<< bDistD`
`#define DirMes  DDRA  &= ~(1<<bDistG | 1<< bDistD)`
`#define CapaGHigh PINC & 1<<bDistG`
`#define CapaDHigh PINC & 1<<bDistD`
	Pour précharger, puis lire, il faut agir sur la directions et sur la valeur en sortie
	
	définitions
	
	variables globales
	nom de la procédure
	
	On se mets en sortie
	On charge les condos
	Environ 100 us pour une bonne charge

```
byte distance0 ; byte distance1 ;
void GetDist ()  // para distance global
{
 byte cnt0, cnt1, tmp, i ;
       CapaCha ;  DirCha ;  // precharge
       delayMicroseconds (100) ;
       DirMes ;
       IR01 = On ;
       cnt0 = 0 ;  cnt1 = 0 ;
       for (i=0; i<100; i++) {
              delayMicroseconds (100) ;
              if (Dist0 == On)    cnt0++ ;
              if (Dist1 == On)    cnt1++ ;
       }
       distance0 = cnt0 ;
       distance1 = cnt1 ;
       IR01 = Off ;
}
```

Dans le programme principal on apelle la procédure qui mets à jour les variables `distance0 et distance1`

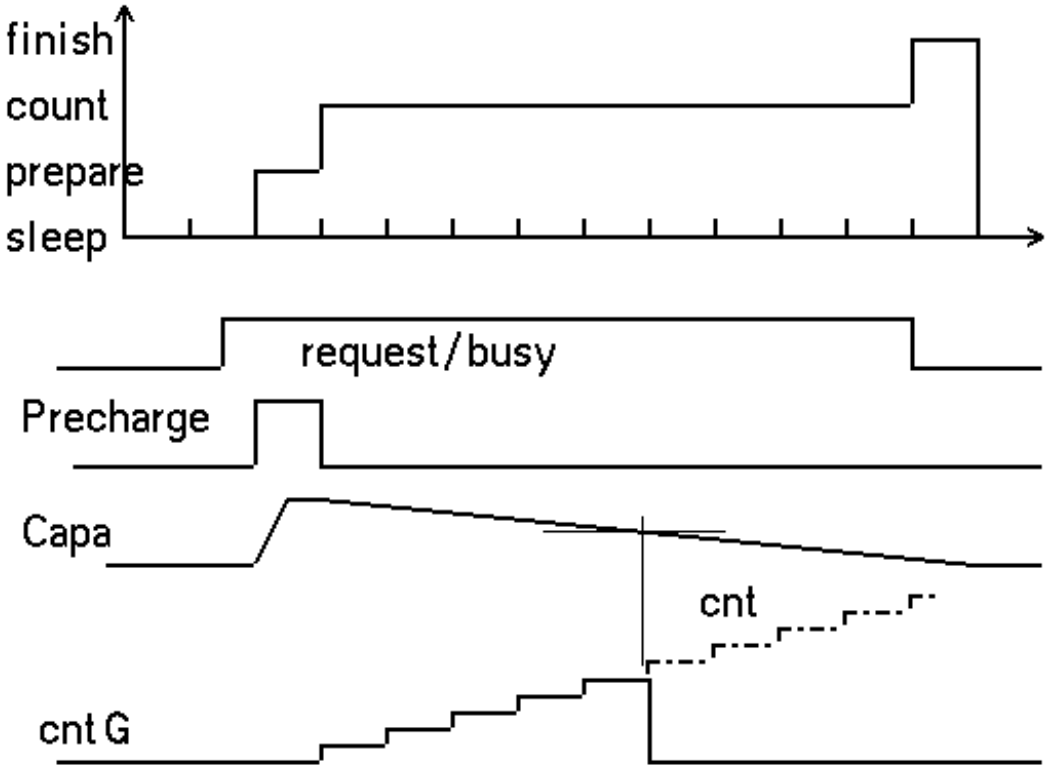
```
       GetDist () ;
       if (distance0 < 40) {..faire ceci.. }
```

On se remets en entrée
On active le transistor qui allume l'éclairage IR
On initialise les compteurs

La boucle de mesure

On sauve les valeurs calculées

On coupe l'infrarouge



jdn 131206