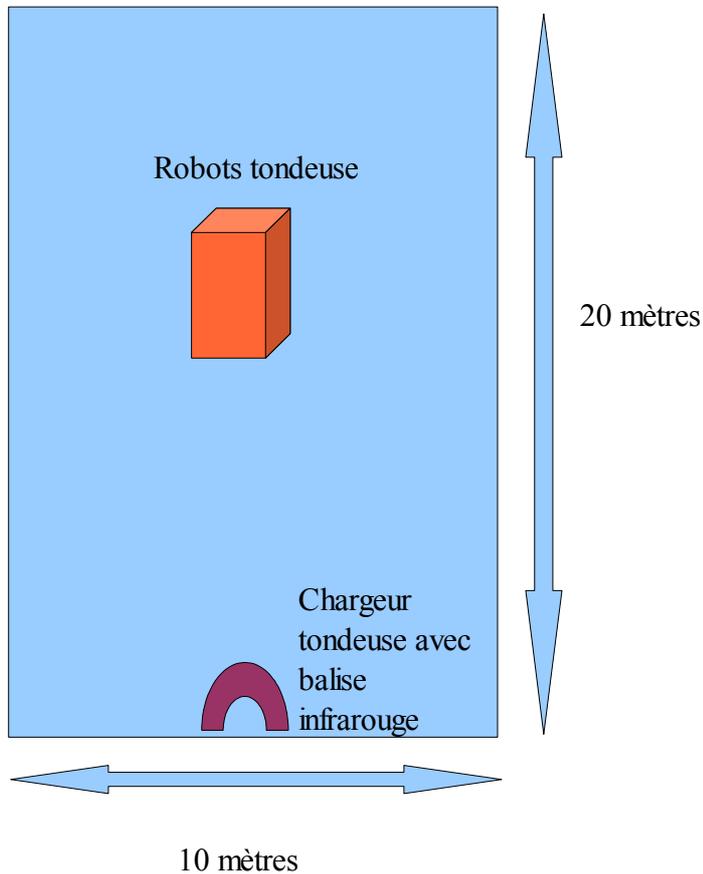


# Robots tondeuse.

## 1. Schémas du terrain:



## 2. Nomenclature du matériel utilisé pour le robots tondeuse:

- moteur utiliser pour la propulsion visseuse;devisseuse;foreuse a batterie .
- contrôleur moteur ( pont en H a mosfet a pwm ) (fabrication maison).
- détecteur de proximité a ultrason (fabrication maison).
- contrôleur pic16F877A cadencer à 20 MHZ .
- codeur pour contrôle odométrique et vitesse (fabrication maison).
- détecteur infrarouge (pour le retour a la base).
- batterie (batterie des 2 foreuses démontées)
- détecteur de pluie (fabrication maison).
- détecteur de blocage de lame de coupe .
- détecteur de soulèvement .
- détection de charge et décharge de la batterie.
- détection jour et nuit.

-détecteur de choc ( style boumper)

**3. Nomenclature pour la base de rechargement avec balise:**

-émetteur infrarouge avec codage (fabrication maison)

-chargeur récupération de la visseuse .

**4. Nomenclature divers pour les 2 système:**

-fibre de verre .

-aluminium divers formes.

-vis et écrous.

**5. Comportement de la tondeuse:**

la tondeuse partira de sa base.

A partir de ce moment elle devra tondre en ligne droite en faisant des aller et retour sur la pelouse .

Pour cela une mémoire eeprom extérieur est utiliser pour une cartographie en 2D ,

Cela déterminera les limites ou la tondeuse ne pourra pas aller et ou aussi ne devra pas tondre (exemple : fleur ; grillage ; passage en béton ou gravillon ; etc...) .

Maintenant quand les batteries seront déchargées ,il y aura l'arrêt du moteur de coupe; mémorisation du dernier emplacement tondu et rotation de la tondeuse pour la détection de la balise infrarouge qui est sur la base pour la recharge .

**6. Comportement de la base de rechargement:**

Nous utiliserons le chargeur de la foreuse avec quelques transformation .

La balise infrarouge sera codée en cas que nous utiliserions plusieurs balises (pour les très grand terrain).

## 7. les schémas utiliser pour le robot et la base:

### 1. le pont en H :

L'idéal est donc d'utilisé des transistors de type MOS, qui introduisent moins de pertes, lors de la conduction.

En effet la résistance drain source (  $R_{dson}$  est de quelques 10m Ohms).

Le transistor de chez Motorola modèle MTP75N06HD supporte un courant de 75 A.

Le problème des transistors MOS est que l'équivalent canal P n'existe pas, en effet ils sont plus difficile à réaliser ( cf un cours de physique des composants).

On va donc utiliser 4 transistors canal N pour réaliser le variateur.

Pour la commande des transistors du bas il n'y a pas de problème car la tension de grille est supérieur à la tension de source.

Pour ceux du haut, il faut mettre un artifice afin d'avoir une tension de commande de grille au dessus de la tension d'alimentation du montage.

En effet le potentiel de la source du transistor du haut se retrouve quasiment au potentiel  $V_{cc}$ . (Q1 et Q2) :  $V_{ds} \sim 0$

On met en place une pompe pour élever la tension ( $V_{bost}$  sur le montage) à presque 2 fois la tension  $V_{cc}$ .

Enfin la commande de l'ensemble peut être confié à un PIC qui fournira le 20khz nécessaire à la pompe.

Avec 4 autres sorties, il commandera la marche avant, et la marche arrière. une version plus évolué peut contenir une fonction de frein.

Pour cela il suffit de faire conduire les deux transistors du bas, en gardant ceux du haut bloquer. Voila pour la partie électrique, il faut cependant prévoir des fils assez gros pour la partie puissance 2.5mm de section multibrin.

Au niveau logiciel on peut intégrer tout un tas de fonction:

Control du courant via un shunt de faible résistance,

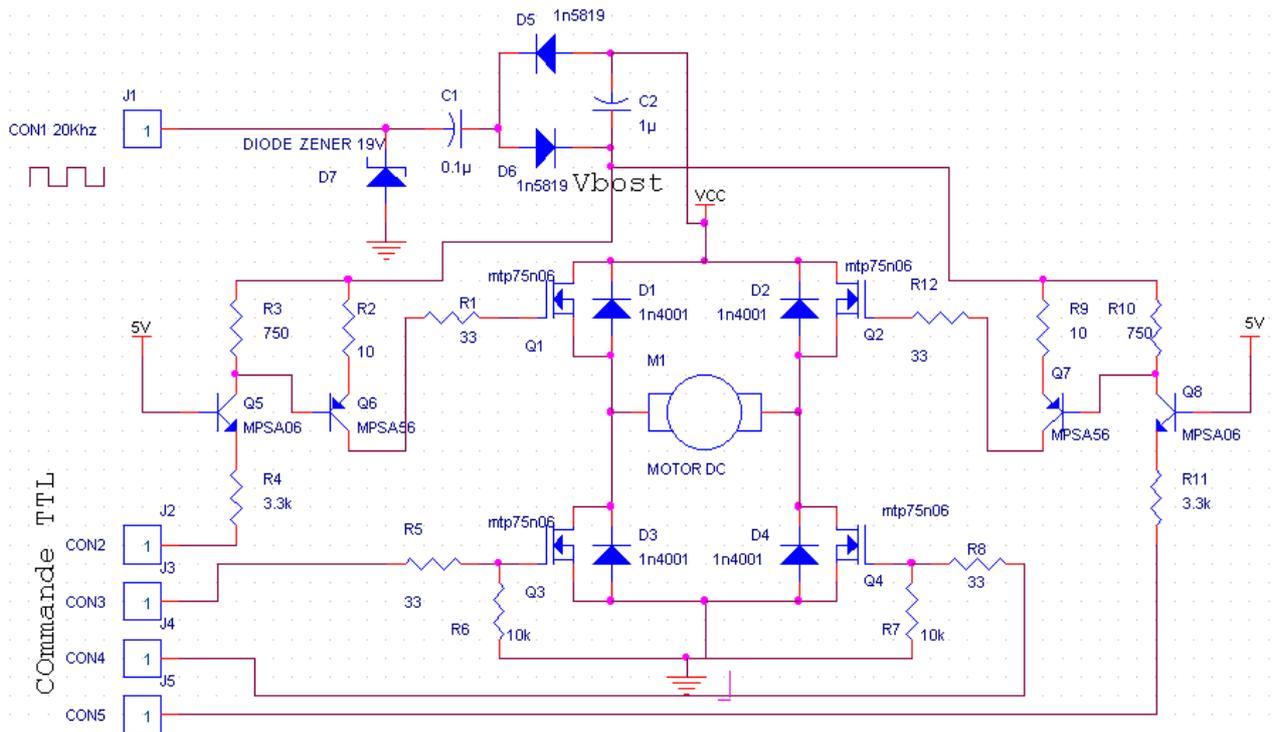
Courbe de démarrage progressive,

Asservissement de la vitesse par contre réaction type P, PI, PID numérique, avec des infos venant d'un incrémental de chez HP par exemple ( HEDS 5640 A12).

etc...

Rappel des caractéristiques:

- Commande en 5V
- Vitesse variable de 0% à 100%
- Avant, arrière
- Tension moteur max 50 V
- Courant continu max 75 A
- Courant crête max 200 A



Commande autorisée:

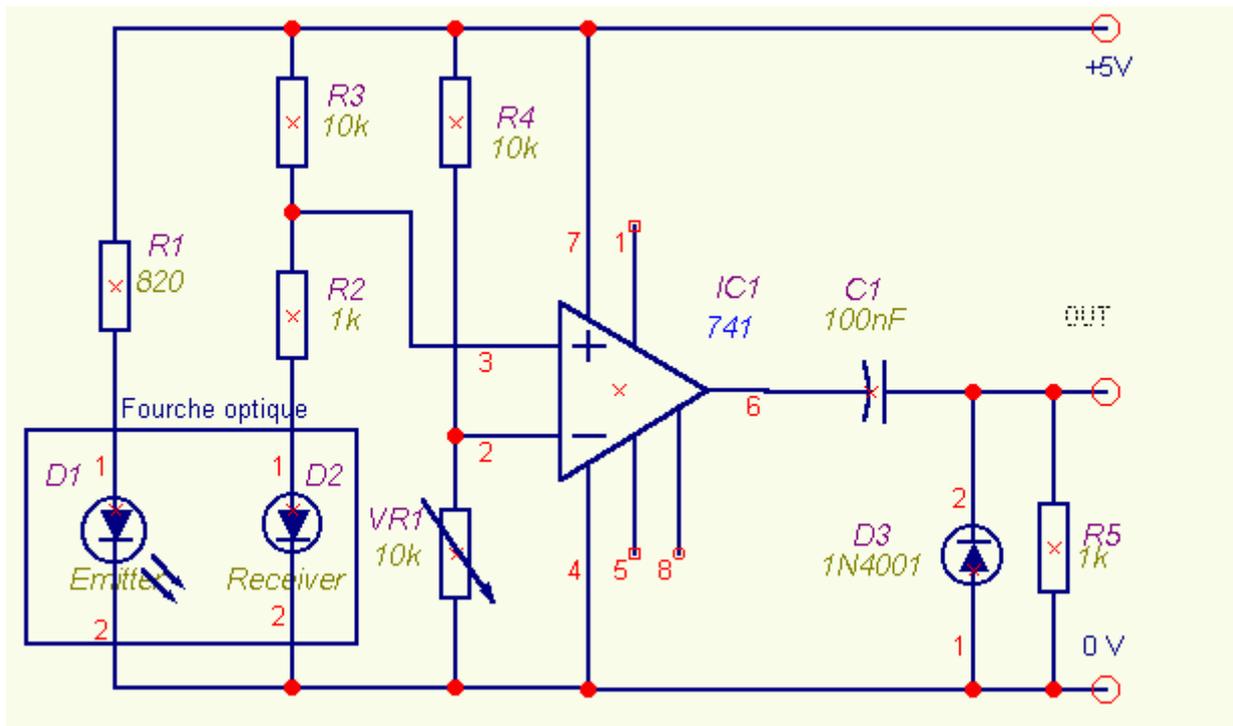
	Con2	Con3	Con4	Con5
Avant	0	0	1	1
Arrière	1	1	0	0
Frein	1	1	1	1
Arrêt	1	0	0	1

le reste est interdit...

Il suffit ensuite de faire varier par logiciel le rapport cyclique des signaux Con4 ou Con3 (en PWM) pour faire varier la vitesse du moteur.

Attention pour les moteurs qui seront fort sollicités ou de fortes puissances c'est d'utiliser des optocoupleurs entre la puissance et le µp.

## 2. Schémas de l'interface d'un capteur tachymétrique :



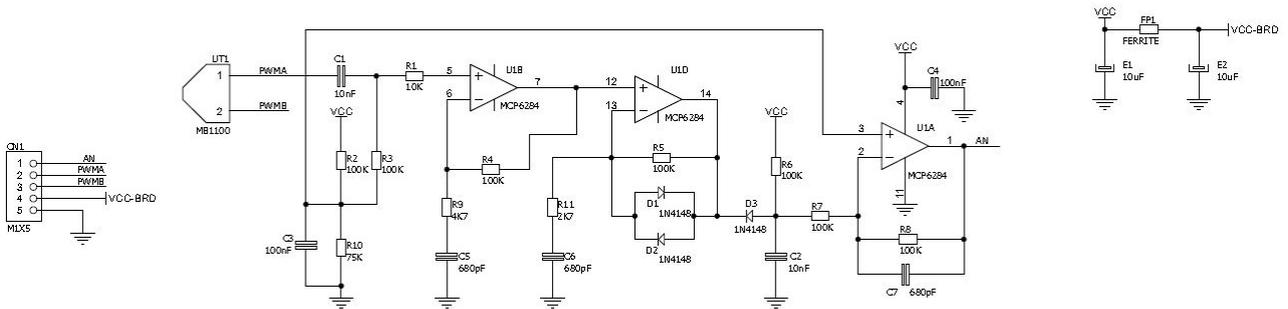
Le schémas ci-dessus devra être réalisé en double ( deux capteurs de vitesses ) : le premier capteur sera relié à la broche RB3 du PIC et le deuxième à RB2. La résistance R1 limite le courant qui circule dans la diode émettrice de la fourche optique. R2 et R3 polarise l'anode de la diode réceptrice aux +5V . Lorsque celle-ci reçoit un signal lumineux , son anode se trouve connecté à la masse. Ce signal est envoyé sur l'entrée non inverseuse de l'ampli op IC1 ( un vulgaire uA741 ) monté en comparateur de tension. R4 et VR1 forme un pont diviseur de tension variable grâce a l'ajustable VR1 . Cette tension variable est appliquée sur l'entrée inverseuse de IC1 et permet de choisir la tension de basculement du comparateur. Le signal de sortie de IC1 disponible sur la broche 6 est un signal carré dont la fréquence dépend directement du nombre de dents qui passent entre la fourche optique. Le condensateur C1 et la diode D3 permettent d'obtenir de courtes impulsions sur les fronts montants du signal carré ( la diode D3 court-circuitant les tensions négatives du aux fronts descendants ) . R5 polarise les entrées du PIC16F877A au 0V. La vitesse d'exécution du PIC16F877A à 20Mhz permet par un simple pooling logiciel de détecter les courtes impulsions sans perte .

Si le pooling ne fonctionne pas je prendrais un autre pic (par exemple: pic16F628A)

et la je mettrais les équations du contrôle de vitesse des deux roues et aussi peut-être une roue folle avec un autre codeur juste pour savoir si le robots patine .

Le  $\mu$ P auras donc 3 pins en entrées pour les 3 codeurs et 3 pins en sorties pour liaisons I2C a une vitesse de 400 kb/sec je pense que ça sera suffisant.

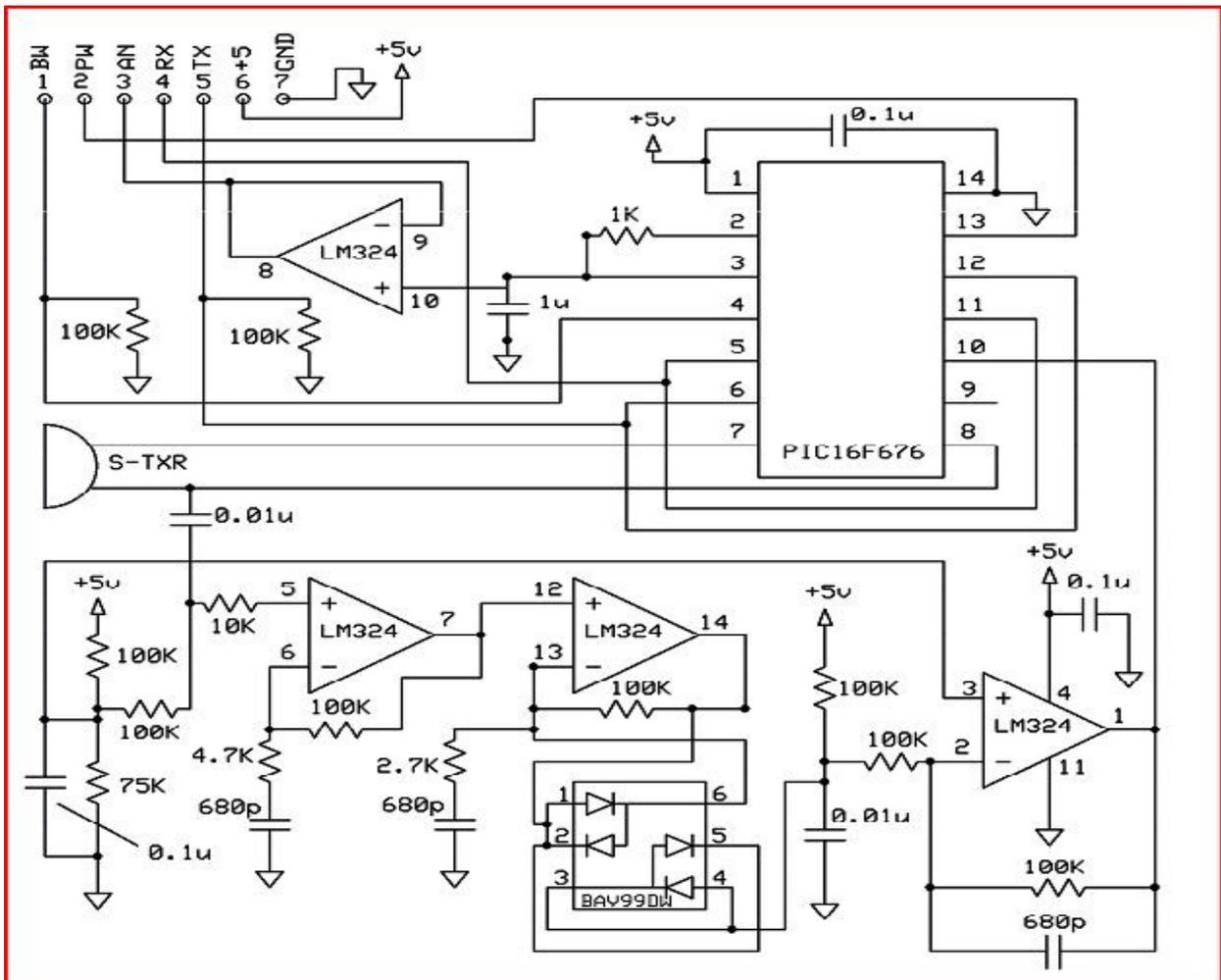
### 3. détecteur de proximité à ultrason:



AN - sorties analogiques tension avec un facteur d'échelle de  $(V_{cc}/512)$  par pouces. Une alimentation de 5V rendements 9.8mV/in ~. et 6.4mV/in 3.3V ~ rendements. La sortie est tamponné et correspond à la plage de données les plus récentes.

voici le schémas d'un détecteur a ultrason avec une cellule émettrice et réceptrice dans le même boîtier. A la sortie AN vous trouverais une tension qui sera proportionnelle a la distance reçue.

Il existe tous fait a cet adresse <http://www.mikroe.com/eng/products/view/437/distance-meter-proto-board/>



le schémas ci-dessus est le même que l'autre schémas sans le pic16F676 .





```

; The main loop controls the range finder. In response to a low going trigger
; input, its calls "burst" to send out 8 cycles of 40khz. It then raises the
; pulse line so the host can begin timing.
; There is a choice of two tone detect routines, the simplest is currently set.
; It then clears the output pulse so the host can complete timing, and loops
; around to wait for the next cycle.
; If an echo is not detected then the watchdog timer will reset the PIC after
; about 30mS, and the pulse line will be cleared. Therefore a very long pulse
; should be interpreted as "nothing detected"

```

```

main:   clrwdt
        btfss  trig           ; wait for trigger signal from user to go high
        goto   main          ; from previous measurement.

```

```

m2:     clrwdt
        btfsc  trig           ; wait for trigger signal from user
        goto   m2

```

```

        call   burst          ; send the ultra-sonic burst
        bsf    pulse         ; start the output timing pulse

```

```

; OK, here's the cheap-n-easy way to detect the echo, just wait for a transition
; on the echo line. Though not really detecting a tone, it is very effective.
; The transducers provide the selectivity.

```

```

m1:     btfsc  echo
        goto   m1            ; wait for low
        bcf    pulse         ; end the output timing pulse

```

```

; And here is the "proper" tone detector. It detects 3 cycles of 40khz to
; give a valid output. It works but is still experimental. It is not as
effective
; as just detecting the first edge, particually in the first few cm.
;

```

```

;     call tone              ; validate 3 cycles of 40khz
;     bcf     pulse          ; end the output timing pulse
;

```

```

        goto   main

```

```

;////////////////////////////////////
;
; The burst routine generates an accurately times 40khz burst of 8 cycles.
; Since a 4Mhz PIC (1uS instruction rate) cannot generate timings of less
; than 1uS, the high half cycle is 12uS and the low half cycle 13uS.
; That's good enough.

```

```

burst:  clrfl  loop
        movlw  8              ; number of cycles in burst
        movwf  loop

```

```

burst1: movlw  0x10           ; 1st half cycle
        movwf  GPIO

```

```

        movlw  3              ; (3 * 3inst * 1uS) -1uS = 8uS
        movwf  dlyctr         ; 8uS + (4*1uS) = 12uS

```

```

burst2: decfsz  dlyctr,f
        goto   burst2

```

```

        movlw  0x20
        movwf  GPIO
        movlw  2              ; (2 * 3inst * 1uS) -1uS = 5uS
        movwf  dlyctr         ; 5uS + (8*1uS) = 13uS

```

```

burst3: decfsz  dlyctr,f

```

```

        goto    burst3
        nop
        decfsz  loop,f
        goto    burst1

        movlw   0x00                ; set both drives low
        movwf   GPIO

        retlw   0

;////////////////////////////////////
;
; The timing for this routine is critical. Our little PIC is only chugging
; along at 4Mhz, or 1uS per instruction. The longest path through this code
; is 19uS, out of the 25uS available - thats tight and why I only wait for a
; low on the echo line and not a high as well.

tone:   clrf    TMR0

t1:     btfsc   echo
        goto    t1                ; wait for low

        movfw   TMR0
        clrf    TMR0
        movwf   period            ; store timer0 value

        movlw   21                ; if(period>22 && period<30)
        subwf   period,w
        btfss   _C
        goto    t2
        movlw   30
        subwf   period,f
        btfsc   _C
        goto    t2

        decfsz  tone_cnt,f        ; 25uS period OK, so
        goto    t1                ; if not yet 3 of them, keep looking
        retlw   0                ; else - success - return

t2:     movlw   3                  ; failed to detect 25uS period, so reset tone
detect  movwf   tone_cnt            ; to 3 and keep looking
        goto    t1

;////////////////////////////////////

        end

;////////////////////////////////////

```

et voici le fichier en HEX , il suffira de prendre notpad de copier/coller et enregistrer le fichier avec une extension HEX .

```

:100000002500890C02000D0C0600200026040400C7
:100010000607070A040006060A0A1309260546060B
:100020000F0A2604070A6700080C2700100C260098
:10003000030C2800E8021A0A200C2600020C2800F3

```

:10004000E802200A0000E702160A000C2600000859

:1000500061004606290A010261002A00150C8A0087

:100060000307390A1E0CAA000306390AE902290A05

:080070000008030C2900290A15

:021FFE00EE0FE4

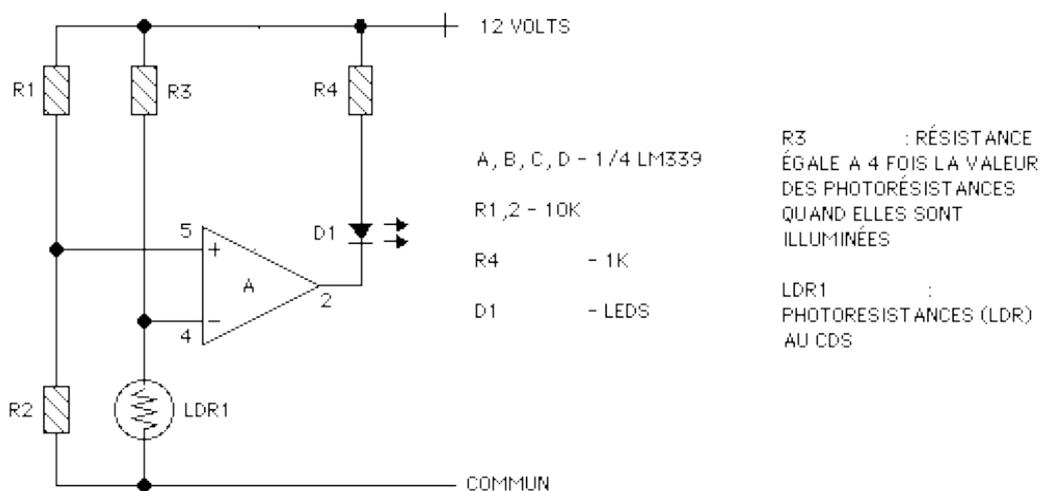
:00000001FF

pour une explication complète voici le lien:

<http://www.robotelectronics.co.uk/hm/srf04tech.htm>

#### 4. détecteur jour nuit:

DÉTECTEURS À PHOTORÉSISTANCES UTILISANT LE COMPAREUR LM339



voilà ce schémas permet de ce passer d'utiliser les timer du  $\mu P$  .

Donc on compte simplement les jours ce qui permettras une facilité de programmation .

Si maintenant on voudrais régler la détection de la lumière ,il suffit de placé un potentiomètre de  $4,7\text{ k}\Omega$  lin entre R1 et R2 et le pin 5 du lm339 .

Tout autre ampli op peut être utiliser par rapport a son stock.

