

## Objectifs

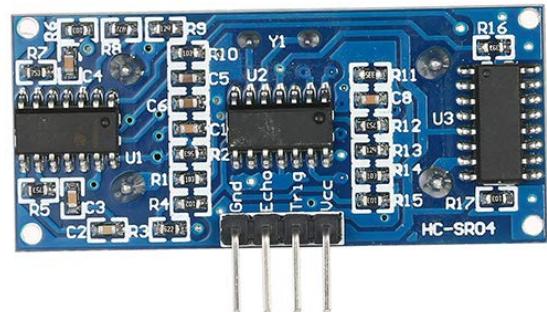
1. Savoir les caractéristiques du capteur ultrason HC-SR04
2. Savoir mesurer la distance avec le capteur ultrason
3. Savoir varier la fréquence de la luminosité des **LED** en fonction de la distance
4. Savoir calculer filtrer un signal avec le filtre **moyenneur**
5. Savoir créer et utiliser des nouvelles fonctions
6. Etc.

## Vidéo démonstration

## Principe de fonctionnement

Le mini-projet est un radar de recul utilisé par les voitures. Il permet d'indiquer au

conducteur qu'il s'approche d'un obstacle (mur de son garage, une autre voiture, etc.). L'indicateur de proximité peut être une sonnerie ou par l'intermédiaire des LEDs (trois LEDs rouge, vert et bleue qui s'allument au même temps). Ici nous varions la fréquence de luminosité en fonction de la distance : lorsqu'un objet s'approche du capteur, les LEDs clignotent rapidement (fréquence haute). Les LEDs clignotent plus lentement lorsque l'objet s'éloigne du capteur. Dans le présent projet, nous fixons la distance maximale à 50 cm (500 mm). Autrement dit lorsque la distance est supérieure à 50 cm, les LED s'arrêtent de clignoter. Elles commencent à s'allumer à partir des 50 cm. Le projet est constitué principalement d'un capteur à ultrasons HC-SR04, trois LED et une [carte Arduino](#). Capteur à ultrason HC-SR04



Voir le projet [Sèche-mains ultrason avec Arduino](#) pour plus des détails

## Câblage des composants

- Signal TRIG: Pin 3
- Signal ECHO: Pin 2



- LED Rouge: 5
- LED Vert : 6
- LED bleu: 7

## Comment mesurer la distance ?

La carte Arduino envoie une impulsion d'une durée égale à  $10\mu s$  dans le pin TRIG. L'impulsion génère une onde ultrason par l'émetteur ultrason. Le récepteur écoute le retour de l'onde ou l'écho via le pin ECHO. La durée de l'impulsion de retour est proportionnelle à la distance allée +retour du signal ultrason et la distance de propagation d'une onde ultrason (voir la formule dans la fonction ci-dessous et le projet sèche main ultrason). Par conséquent, on peut déduire la distance entre l'émetteur et l'objet. Il faut bien noter que le capteur renvoie une valeur nulle lorsque l'objet est très proche ou trop loin (absence de l'objet). La fonction GetDisCM() renvoie la distance en mm.



## Radar de recul ultrason avec Arduino



```
float GetDisCM(int Echo_in_pinn, int Trig_out_pinn, double Speed_soundd)
{
    float Dist_milm=0.0;
    long HC_val=0;
    // Déclanchement mesure avec l'envoie d'une impulsion de 10µs
    digitalWrite(Trig_out_pin, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig_out_pin, LOW);
    // Lecture du temps aller/retour de l'onde
    HC_val = pulseIn(Echo_in_pin, HIGH, 30000);
    // Calcul de distance d(mm), V=d/t==> v*t
    // Le coefficnet 2 pour la distance aller
    Dist_milm = (HC_val/2.0) * Speed_sound;

    return Dist_milm;
```



Radar de recul ultrason avec Arduino

}



## Comment faire clignoter une LED en fonction de la distance ?

La zone utile pour le capteur est fixée à 500 mm (50 cm). Lorsque la distance > 50 cm, on positionne à zéro les trois LED. Lorsque la distance =50 cm, a varié la luminosité en fonction de la distance. L'astuce utilisée consiste à utiliser la fonction `delay()` qui varier avec la distance. L'exemple ci-dessous montre comment clignoter une LED avec une fréquence égale à 100 hz (période 10 ms).



## Radar de recul ultrason avec Arduino



```
while(1)
{
    LED=1;
    delay(5);
    LED=0
    delay(5);
```



Radar de recul ultrason avec Arduino

}



Nous constatons que la fonction `delay()` prend en entrée par uniquement une constante, mais aussi une variable ! cette particularité permet de faire varier le temps d'attente en fonction de la distance. Il suffit d'utiliser la distance comme argument d'entrée de la fonction : `delay(Distance)` de la façon suivante :



## Radar de recul ultrason avec Arduino



## Radar de recul ultrason avec Arduino

```
while(1)
{
    Distant= lire la distance;
    LED=1;
    delay(Distant);
    LED=0
    delayDistant);
```



Radar de recul ultrason avec Arduino

}



La fonction SetLED() positionne les LED en fonction de la distance. La fonction tiens en compte également la limitation de 50 cm de la distance. Elle ignore aussi les valeurs nulles induites par l'absence de l'objet ou la présence d'un objet plus proche.



## Radar de recul ultrason avec Arduino



```
void SetLED(float Dist_mmm, int LED_rr, int LED_gg, int LED_bb)
{
    if((Dist_mmm < 500) && (Dist_mmm != 0.0))
    {
        digitalWrite(LED_rr, HIGH);
        digitalWrite(LED_gg, HIGH);
        digitalWrite(LED_bb, HIGH);
        delay(floor(Dist_mmm));
        digitalWrite(LED_rr, LOW);
        digitalWrite(LED_gg, LOW);
        digitalWrite(LED_bb, LOW);
        delay(floor(Dist_mmm));
    }
}
```



Radar de recul ultrason avec Arduino

}



## Comment calculer la valeur moyenne ?

La fonction GetMean() permet de renvoyer la [valeur moyenne](#) de la distance. La taille du filtre moyenneur est variable (lorsque la taille augmente, le temps de réponse augmente, la précision augmente). Consultez le projet [Traceur série avec Arduino](#) pour en savoir plus.



## Radar de recul ultrason avec Arduino



```
float GetMean(long Taille, int Echo_in_pinn, int Trig_out_pinn, double Speed_soundd)
{
    long i;
    float Distance_tmp, Mean_val=0.0,Somme=0.0;
    for(i=0; i<Taille; i++)
    {
        Distance_tmp=GetDisCM( Echo_in_pinn,  Trig_out_pinn,
Speed_soundd);
        Somme=Somme+Distance_tmp;
    }
    Mean_val=Somme/Taille;
    return Mean_val;
```



Radar de recul ultrason avec Arduino

}



# Programme principale

```
// Variables pour HC-SR04
const int Echo_in_pin = 2;
const int Trig_out_pin = 3;
float Dist_mm=0.0;

// Vitesse du son (mm/us)
const double Speed_sound = 340.0e-3;

// Paramètres
const int LED_r=5;
const int LED_g=6;
const int LED_b=7;

void setup()
{
    // Initialisation HC-SR04
    pinMode(Trig_out_pin, OUTPUT);
    pinMode(Echo_in_pin, INPUT);
    digitalWrite(Trig_out_pin, LOW);
    // Init interface série
    Serial.begin(115200);

    // LEDs
    pinMode(LED_r,OUTPUT);
    pinMode(LED_g,OUTPUT);
    pinMode(LED_b,OUTPUT);
    digitalWrite(LED_r, LOW);
    digitalWrite(LED_g, LOW);
    digitalWrite(LED_b, LOW);
```



```
}

void loop()
{

    /* Mesure de la distance */
    Dist_mm=GetDisCM( Echo_in_pin,  Trig_out_pin,  Speed_sound); // Valeur brute
    //Dist_mm=GetMean(8,  Echo_in_pin,  Trig_out_pin,  Speed_sound);
    // Valeur moyenne

    /* Affichage de la distance */
    Serial.println(Dist_mm);
    /* Activation des LEDs */
    SetLED(Dist_mm, LED_r, LED_g, LED_b);

}

// Calcul de la distance en mm
float GetDisCM(int Echo_in_pinn, int Trig_out_pinn, double Speed_soundd)
{
    float Dist_milm=0.0;
    long HC_val=0;
    // Déclanchement mesure avec l'envoie d'une impulsion de 10µs
    digitalWrite(Trig_out_pin, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig_out_pin, LOW);
    // Lecture du temps aller/retour de l'onde
    HC_val = pulseIn(Echo_in_pin, HIGH, 30000);
    // Calcul de distance d(mm), V=d/t==> v*t
    // Le coefficnet 2 pour la distance aller
    Dist_milm = (HC_val/2.0) * Speed_sound;
```



```
    return Dist_milm;
}

// Valeur Moyenne
float GetMean(long Taille, int Echo_in_pinn, int Trig_out_pinn, double Speed_soundd)
{
    long i;
    float Distance_tmp, Mean_val=0.0, Somme=0.0;
    for(i=0; i<Taille; i++)
    {
        Distance_tmp=GetDisCM( Echo_in_pinn, Trig_out_pinn,
Speed_soundd);
        Somme=Somme+Distance_tmp;
    }
    Mean_val=Somme/Taille;
    return Mean_val;
}

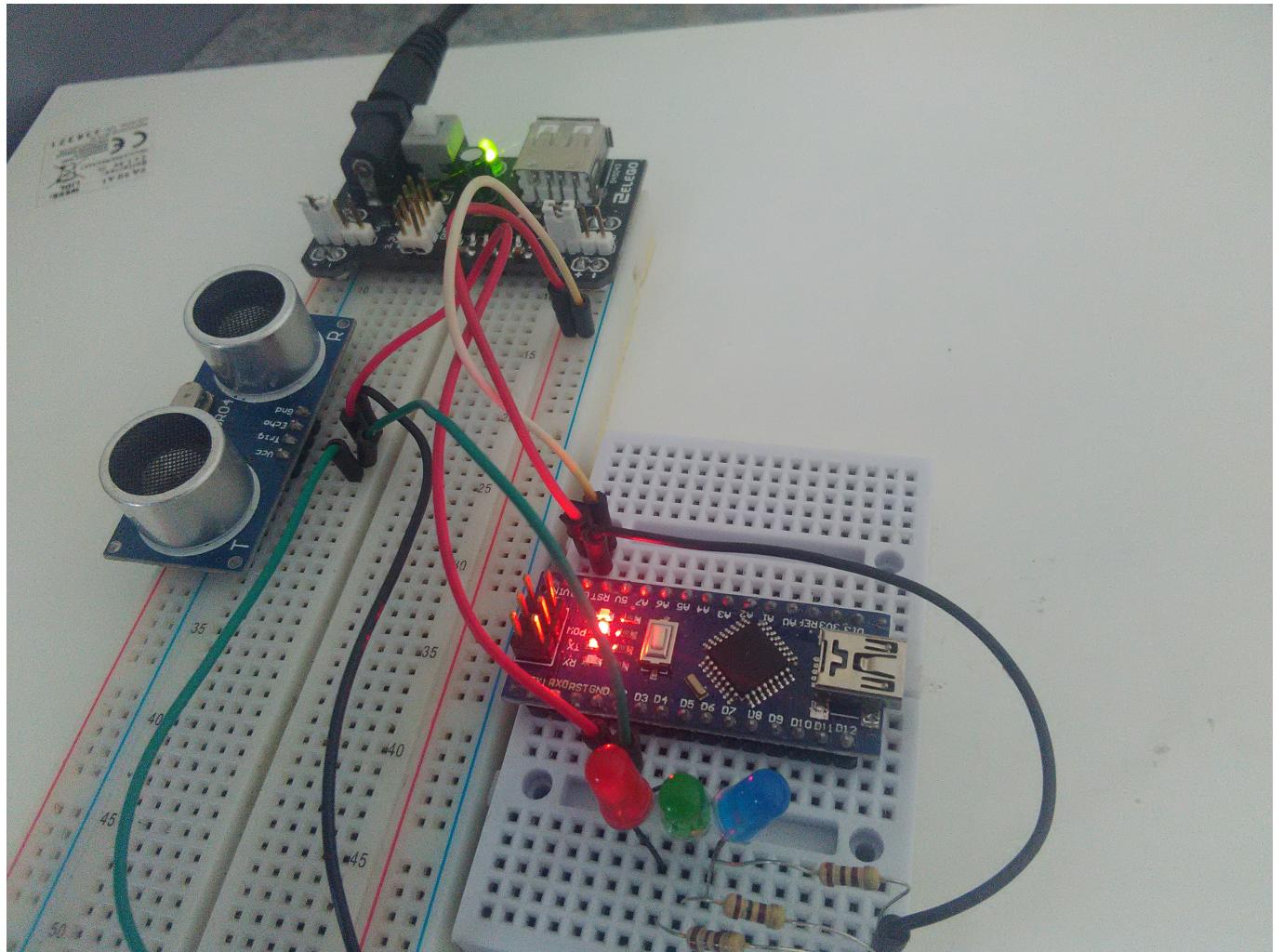
void SetLED(float Dist_mmm, int LED_rr, int LED_gg, int LED_bb)
{
    if((Dist_mmm < 500) && (Dist_mmm != 0.0))
    {
        digitalWrite(LED_rr, HIGH);
        digitalWrite(LED_gg, HIGH);
        digitalWrite(LED_bb, HIGH);
        delay(floor(Dist_mmm));
        digitalWrite(LED_rr, LOW);
        digitalWrite(LED_gg, LOW);
        digitalWrite(LED_bb, LOW);
        delay(floor(Dist_mmm));
    }
}
```

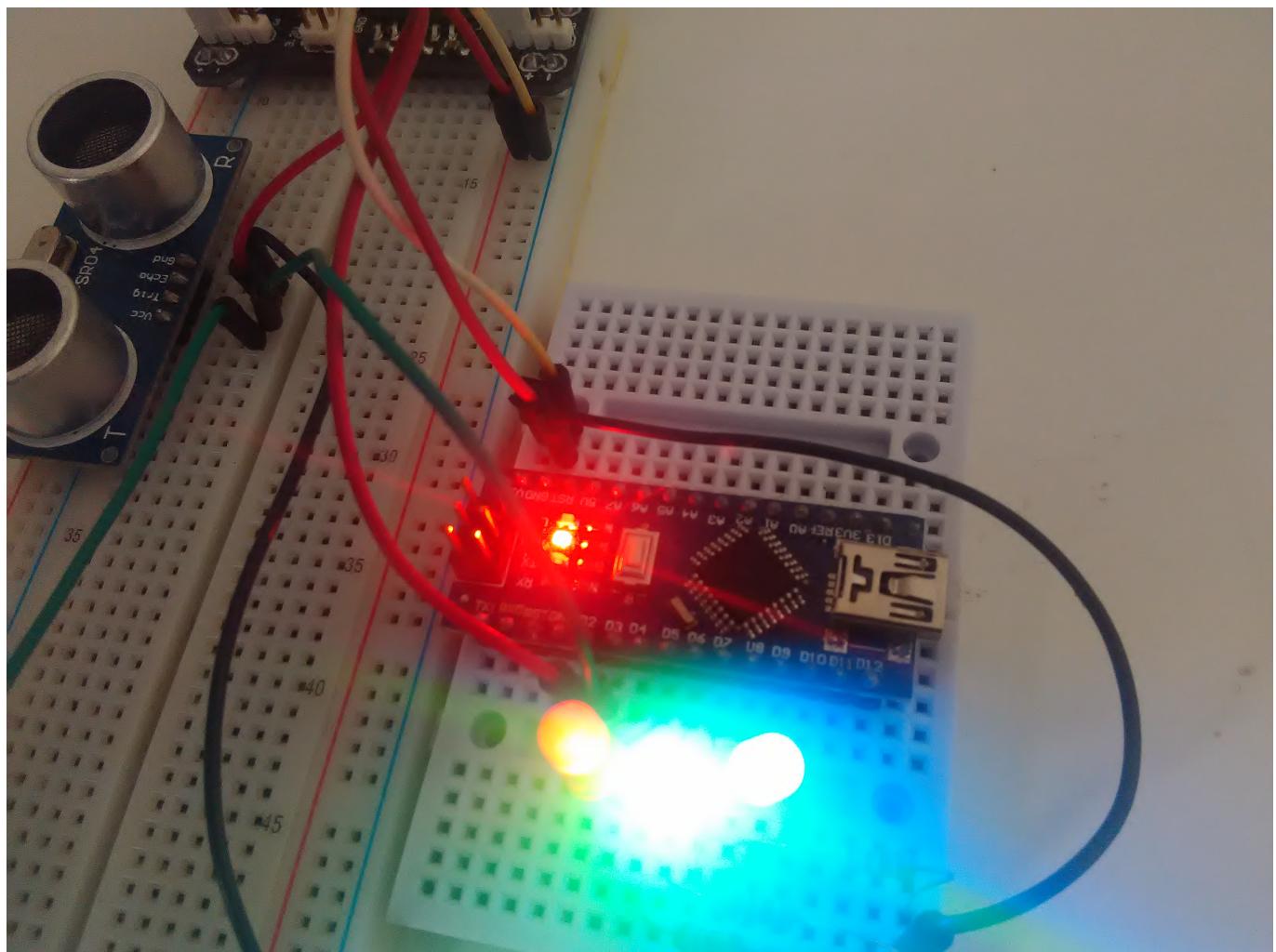


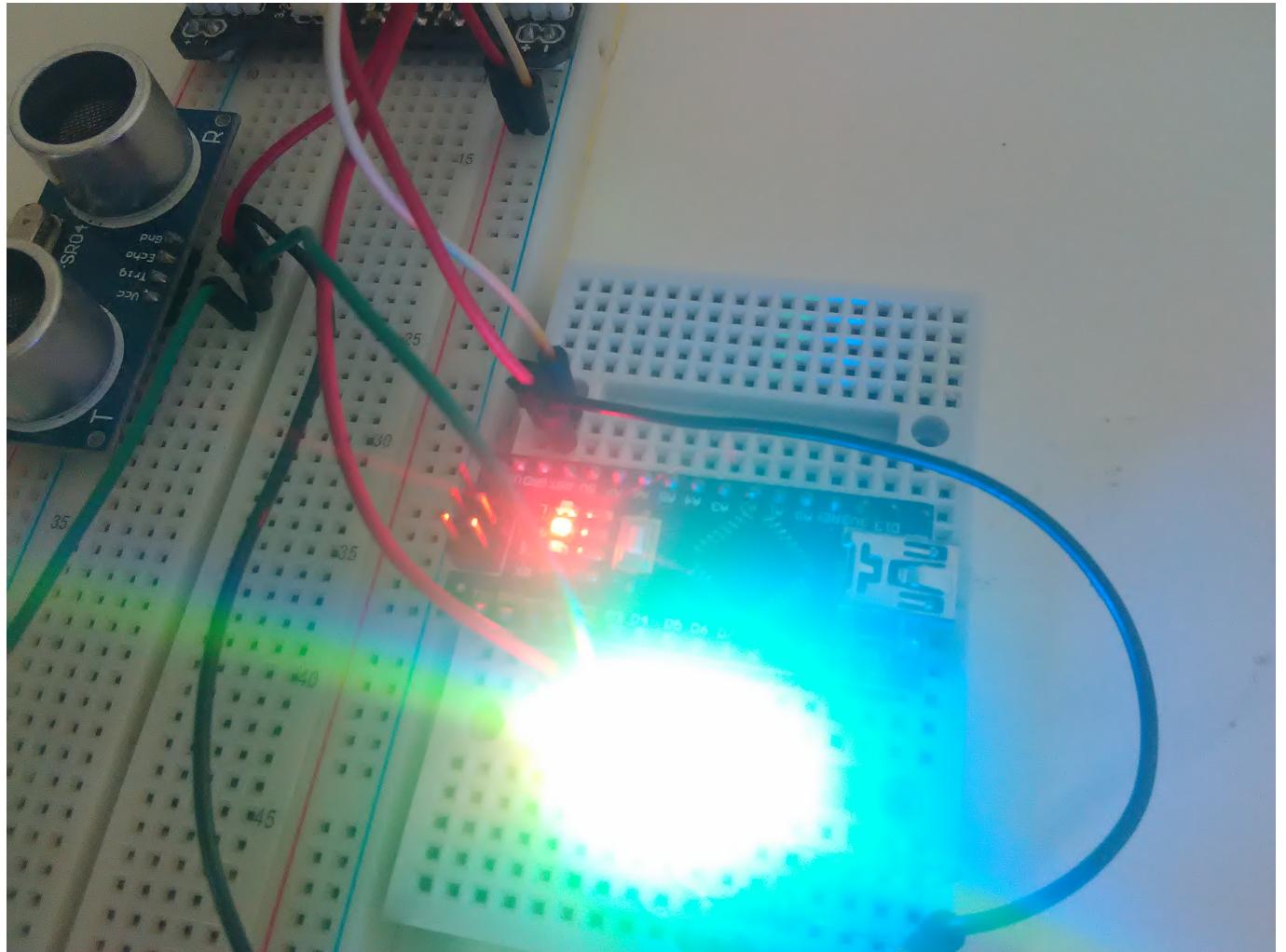
Radar de recul ultrason avec Arduino

}

## Photos du projet







## Téléchargement

- Programme Arduino: Radar de recul ultrason avec Arduino

[Tout les projets microcontrôleur](#)