TP20 – DETERMINER A L'AIDE D'UN MICROCONTROLEUR UNE DISTANCE APPLICATION AU RADAR DE RECUL

Un télémètre est un appareil qui permet de mesurer des distances à l'aide d'une onde ultrasonore ou lumineuse. Pour cela, l'émetteur envoie une brève impulsion vers la cible et en chronométrant la durée mise par l'onde pour parcourir l'aller-retour on peut calculer la distance entre l'émetteur-récepteur et la cible.

OBJECTIF DU TP

Reconstituer un radar de recul automobile simplifié. Le cahier des charges est le suivant : le système devra émettre un son ainsi qu'un signal lumineux rouge si la distance est inférieure à 100cm

DOCUMENTS

Doc.1

Qu'est-ce qu'un microcontrôleur ?

Un microcontrôleur est un microprocesseur que vous programmer : vous allez écrire ou modifier quelques lignes de programmes, il va les interpréter et appliquer ce que vous lui avez demander de faire. Par exemple : « s'il fait trop chaud ou trop ensoleillé, j'aimerai que tu baisses les rideaux ! » ; « Si le niveau d'eau est bas, rajoute de l'eau » etc…

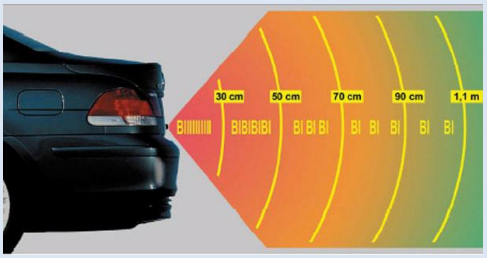
Pour ce faire, il faudra que vous communiquiez avec le même langage, rajouter à ce microcontrôleur des capteurs (de température, d'éclairement) et des actionneurs (moteurs, buzzer, LED)

HTML, Java, php, python, C+,Cobol,.. les langages informatiques sont nombreux . Vous avez eu l'occasion de programmer en Python en Math et en Physique. Les cartes Arduino comprennent et exécutent un langage qui est proche de Python, c'est le langage C++

Doc.2

Les radars de recul :

Les radars de recul pour les véhicules automobiles sont des systèmes d'assistance qui avertissent de la présence d'un obstacle à l'arrière du véhicule, lors d'une opération de marche arrière. Les principaux constituants de ce type de système sont :



- Les capteurs à ultrasons, composés d'émetteurs et de récepteurs
- Le calculateur, qui estime la distance entre l'obstacle et l'arrière du véhicule, à partir des informations délivrées par les capteurs à ultrasons
- Un système d'émission sonore (parfois accompagné de signal lumineux) qui avertit le conducteur.

Habituellement, un bip sonore est émis lorsque la distance obstacle/véhicule devient inférieure à une distance limite, puis, plus l'obstacle se rapproche, plus la fréquence des bips est élevée, jusqu'à devenir une émission sonore continue lorsque le véhicule est très proche de l'obstacle

Doc.3

Materiel à disposition

Carte Arduino, capteur ultrasons, buzzer, LED rouge, ou logiciel de simulation tinkercad

Doc.4

Programme permettant d'émettre un son : la3 (440Hz)

La fonction tone permet de produire un son. Entre les parenthèses il faut indiquer la sortie, la fréquence de la note, combien de temps jouer la note.

```
void setup() {
pinMode(6,OUTPUT);
}
void loop() {
tone(6,440,1000);
delay(3000);
}
```

La fonction noTone : stoppe la génération d'impulsion produite par l'instruction tone(). N'a aucun effet si aucune impulsion n'est générée. Il faut juste indiquer la sortie entre les parenthèses **noTone(6);**

Fréquence des notes :

Do2	Ré2	Mi2	Fa2	Sol2	La2	Si2	Do3	Ré3	Mi3	Fa3	Sol3	La3	Si3	Do4	Ré4	Mi4	Fa4	Sol4	La4	Si4
132	148,5	165	176	198	220	247,5	264	297	330	352	396	440	495	528	594	660	704	792	880	990

Doc.5

Programme permettant d'allumer une LED :

La fonction digitalWrite : Met un niveau logique HIGH (HAUT en anglais) ou LOW (BAS en anglais) sur une broche numérique. Si la broche a été configurée en SORTIE avec l'instruction pinMode(), sa tension est mise à la valeur correspondante : 5V (ou 3.3V sur les cartes Arduino 3.3V) pour le niveau HAUT, 0V (masse) pour le niveau BAS.

```
void setup() {
  pinMode(2,OUTPUT);
  }
void loop () {
    digitalWrite(2,HIGH);
    delay(1000);
  }
```

Doc.6

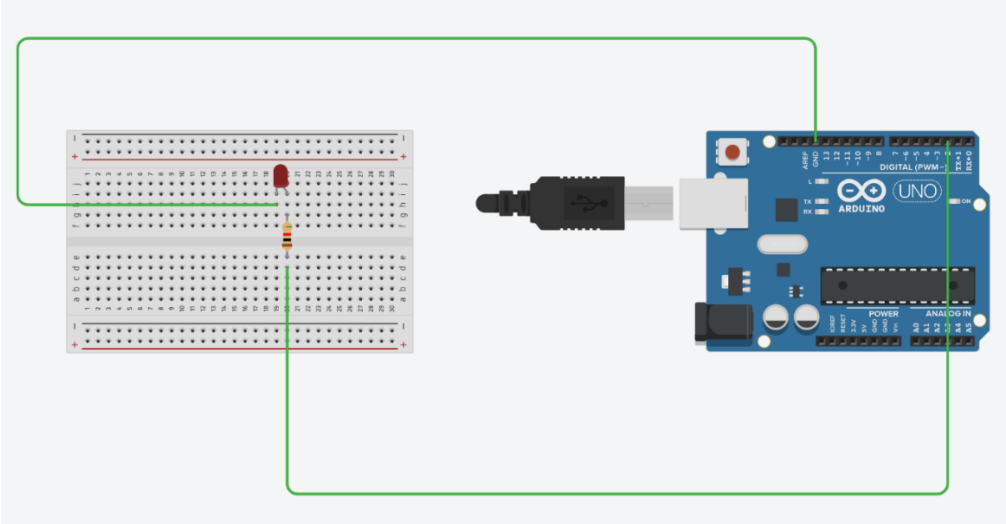
Boucle if… else :

La syntaxe d'une telle boucle est :

```
if (condition)
{
    Action à réaliser si la condition est vraie
}
else
{
     Action à réaliser sinon
}
```

TRAVAIL A FAIRE

1. Regarder la vidéo logiciel tinkercad : https://youtu.be/AFWlV1CLjqg
2. Dans le logiciel tinkercad, après vous avoir créer un compte, réaliser un premier circuit permettant de faire clignoter une led rouge. La led est branchée sur le pin n°2



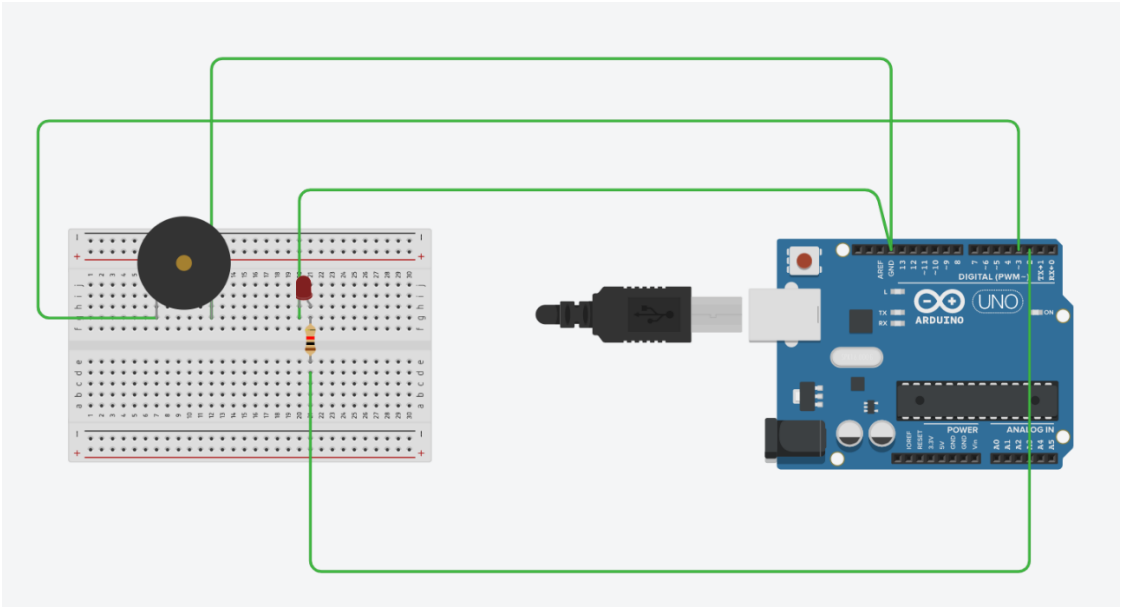
3. Dans la partie code, faire un copier/coller du programme ci-dessous :

```
void setup()
{
  pinMode(2, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(2, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

4. Changer le programme pour faire clignoter la led plus vite

5. Réaliser un second circuit permettant d'émettre un son ainsi que de faire clignoter une led rouge (la led est toujours sur le pin2 et le buzzer sur le pin3)

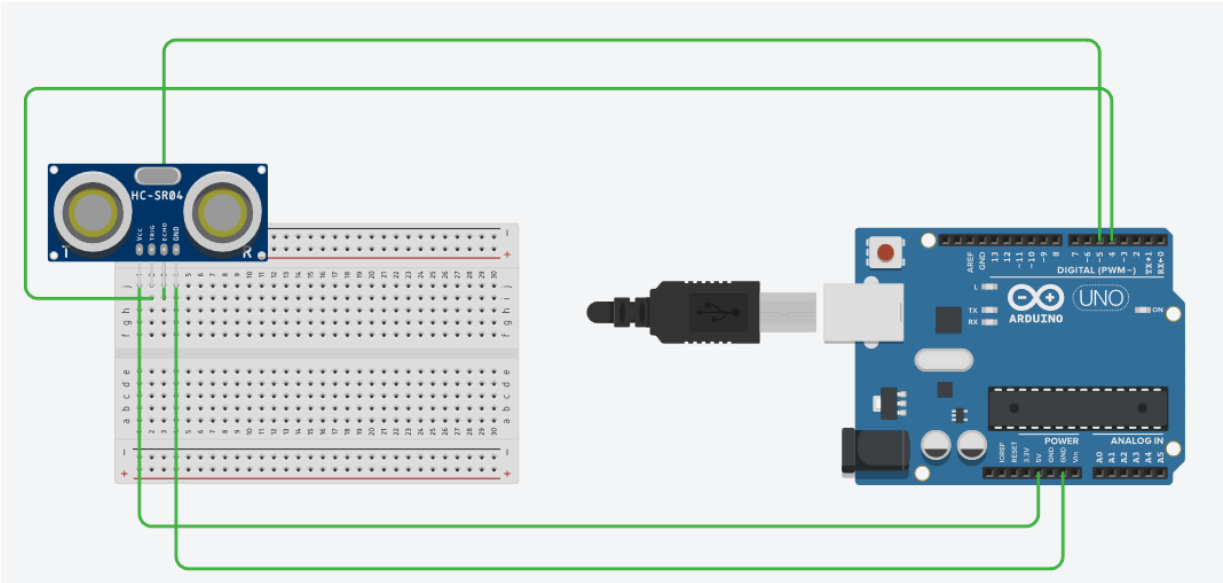


6. Dans la partie code, faire un copier/coller du programme ci-dessous :

```
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
}

void loop()
{
  digitalWrite(2, HIGH);
  tone(3,528,2000);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(2, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

7. Changer le programme pour jouer une note La3
8. Faire le montage télémètre sur tinkercad (VCC est sur le pin 5V, TRIG est sur le pin4 , ECHO est sur le pin 5 et GND est le pin GND)



9. Dans la partie code, faire un copier/coller du programme ci-dessous :
   Lancer la simulation. Cliquer en bas sur le moniteur de série pour voir les distances ainsi mesurées apparaitre
   Cliquer sur l'émetteur-récepteur d'ultrasons et changer la distance de l'obstacle. Vérifier que les mesures de distance changent dans le moniteur de série

```
int trigPin = 4;  // Pins utilisés dans ce projet
int echoPin = 5;
long retard ; // définition des variables
int distance;

void setup() {
pinMode(trigPin, OUTPUT); // Règle le trigPin en sortie
pinMode(echoPin, INPUT); // Règle le echoPin entrée
Serial.begin(9600); // Starts the serial communication
}

void loop() {
digitalWrite(trigPin, LOW); //  Ces 5 lignes fabriquent une impulsion de 10ms
```

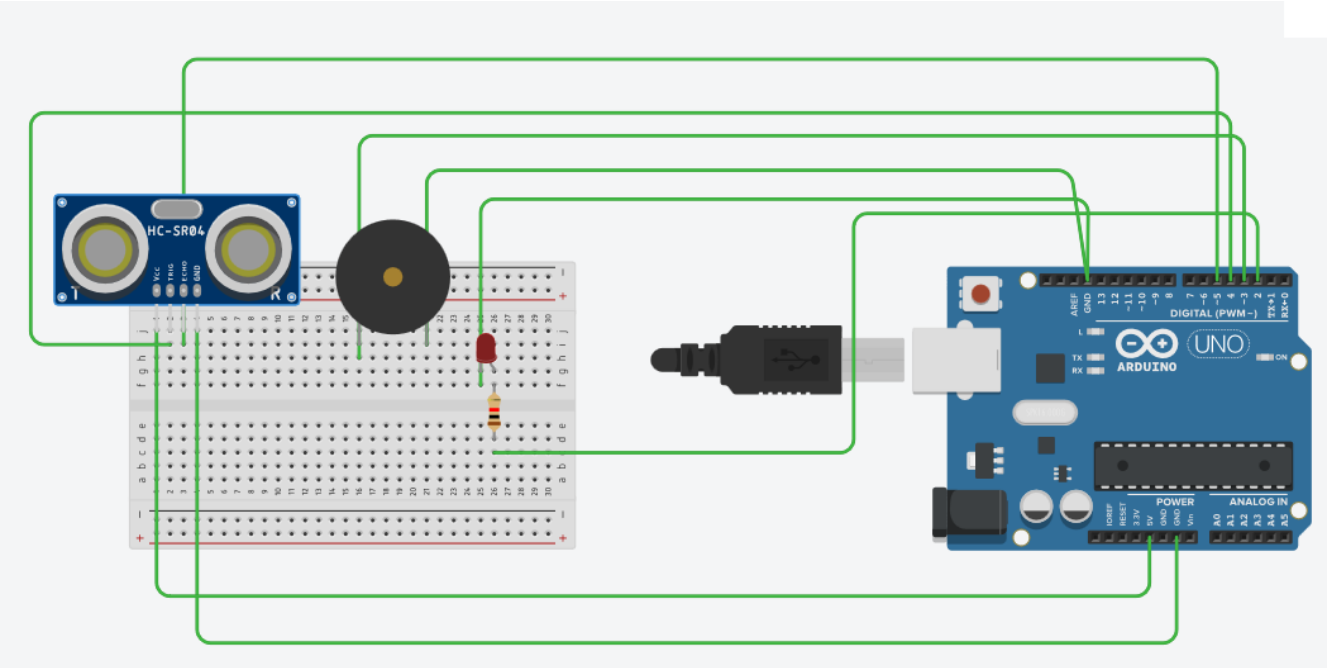
```
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

retard = pulseIn(echoPin, HIGH);    //  Mesure le retard à la réception, détection d'un pulse
distance = retard * 1e-6 * 34000 /2;  // calcule la distance correspondante

Serial.print("Distance en cm : ");
Serial.println(distance);
delay(500);
}
```

10. Modifier votre circuit et votre programme actuels afin de répondre au cahier des charges du TP

Correction :




```
int trigPin = 4;  // Pins utilisés dans ce projet
int echoPin = 5;
long retard ; // définition des variables
int distance;


void setup() {
pinMode(trigPin, OUTPUT); // Règle le trigPin en sortie
pinMode(echoPin, INPUT); // Règle le echoPin entrée
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
Serial.begin(9600); // Starts the serial communication
}

void loop() {
digitalWrite(trigPin, LOW); //  Ces 5 lignes fabriquent une impulsion de 10ms
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
```

```
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

retard = pulseIn(echoPin, HIGH);    //  Mesure le retard à la réception, détection d'un pulse
distance = retard * 1e-6 * 34000 /2;  // calcule la distance correspondante

if(distance<100)
{
  digitalWrite(2,HIGH);
  tone(3,440,1000);
}
else
{
  digitalWrite(2,LOW);
  noTone(3);
}
Serial.print("Distance en cm : ");
Serial.println(distance);
delay(500);
}
```