Stationnement parfait **avec LiDAR**

Rob Reynolds (États-Unis)

Quoi de plus génial que les capteurs de distance ? Ils sont souvent le premier pas dans la réalisation d'un robot autonome, mais servent aussi bien dans d'autres applications divertissantes, éducatives ou utilitaires. Quand j'ai eu l'idée d'un système d'assistance pour garer ma voiture, j'ai déménagé le projet avec ses capteurs de distance de ma paillasse dans le garage et suis parti pour de nouvelles bidouilles !

D'abord, inventer un besoin

Je n'ai jamais eu ni balle de tennis, ni planche de surf ou ni aucun autre repère accroché au plafond de mon garage pour m'indiquer l'endroit précis où arrêter ma voiture lorsque je la rentre au garage. Maintenant que j'ai un adolescent d'âge à se mettre au volant, il est temps d'y songer. Comme je ne joue pas au tennis, mais à l'électronique, c'est d'elle que viendra donc la solution. Le plus simple serait de suspendre un RPi au plafond du garage à la place de la balle de tennis, mais je veux utiliser un capteur de distance associé à un petit feu tricolore. À l'entrée du garage, la voiture est accueillie par une LED verte ; lorsqu'elle s'avance, le feu passe au jaune puis au rouge lorsqu'elle atteint le point d'arrêt idéal. Ce feu tricolore, je vais le concevoir et l'imprimer en 3D. Ça fera un coffret sympa !

Quel est le capteur adéquat ?

Avec cette profusion de capteurs de distance/proximité (**fig. 1**), comment savoir lequel convient le mieux à votre projet ? Il y a un bon nombre de caractéristiques à évaluer, dont la portée, la résolution, le type d'interface, la fréquence de mesure et le coût. Il y en a qui ne sont pas critiques (faut-il vraiment 635 mesures par seconde ?) et d'autres qui sont incontournables (mon professeur affirme qu'il faut utiliser des composants I²C). Entre porte et mur du fond, mon garage fait environ 8,6 m . Je pourrais probablement m'en tirer avec l'un de nos appareils à ultrasons XL-MaxSonar, d'une portée d'environ 8,3 m, mais, pour cette réalisation, je vais utiliser le module TFMini - Micro LiDAR [1]. Je l'utilise avec le kit Qwiic, car il comprend une carte amplificatrice. Comme le TFMini fonctionne sous 5 V mais communique sous 3,3 V, la carte amplificatrice fournie avec le kit Qwiic élimine le besoin de conversion de niveau logique si l'on utilise une carte 5 V, ou d'une alimentation bi-tension si l'on utilise une carte 3,3 V. En l'associant au Redboard, avec son connecteur Qwiic intégré, je peux faire de la moitié capteur de cet assemblage un dispositif *brancher-jouer* simple (**fig. 2**). Ajoutez quelques LED à forte luminosité, des résistances et une alimentation, et le tour sera joué !

Qwiic = assemblage rapide et facile des composants I²C

Le système Qwiic simplifie cette réalisation. La version TFMini Qwiic est livrée avec la carte amplificatrice et une paire de câbles. Un câble relie le module à la carte amplificatrice, l'autre la carte amplificatrice

au connecteur Qwiic de votre RedBoard. Les LED verte, jaune et rouge sont sur les broches 8, 9 et 10. Pour le code, j'ai légèrement retouché le croquis *LidarTest.ino* disponible sur la page du *Guide de branchement du TFMini Qwiic* [2]. Le **listage 1** montre le programme retouché. J'ai également réalisé un modèle vite fait de feu tricolore que vous pouvez télécharger, ainsi que le croquis Arduino, depuis mon dépôt Github. [3]

JE VOUS AURAI PRÉVENU ! En raison de la consommation du TFMini et du système de gestion de l'énergie du RedBoard, si vous essayez d'alimenter votre projet final par la prise jack, il se figera, la LED verte restera allumée et votre chauffeur – trompé par sa confiance dans votre assistant – foncera droit dans le mur. Alors qu'avec l'alimentation par le connecteur microUSB, ce problème ne se pose pas. Oubliez donc cette prise jack si vous ne voulez pas d'ennuis !

À vous de jouer !

Ma réalisation est simple, voire expéditive, vous pourrez donc faire autrement et mieux. Lancez-vous avec le code et les fichiers .STL [3] ! Quelques suggestions :

- ajouter des boutons pour faciliter le réglage du point d'arrêt optimal ;
- améliorer le coffret ;
- augmenter le nombre de LED pour accroître la visibilité.

Avec de l'ambition, vous pouvez même remplacer le RedBoard par un RPi et programmer ce projet en Python, en ajoutant un écran pour plus d'interaction visuelle.
Y a-t-il un avantage à utiliser un ordinateur mono-carte plutôt qu'un microcontrôleur pour ce projet ? Absolument aucun. Mais pourquoi faire simple quand on peut faire compliqué ? Alors, au travail, faites-vous plaisir, les amis !

(200698 VF Helmut Müller)

[suite au verso]

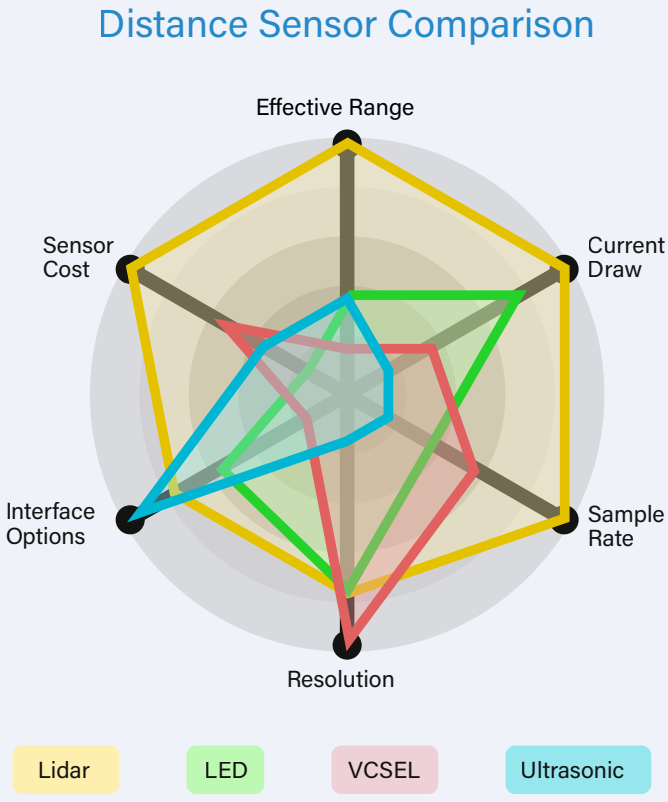

Figure 1. Pour choisir le bon capteur de distance pour votre application, évaluez les performances des technologies disponibles par rapport à vos objectifs de conception.

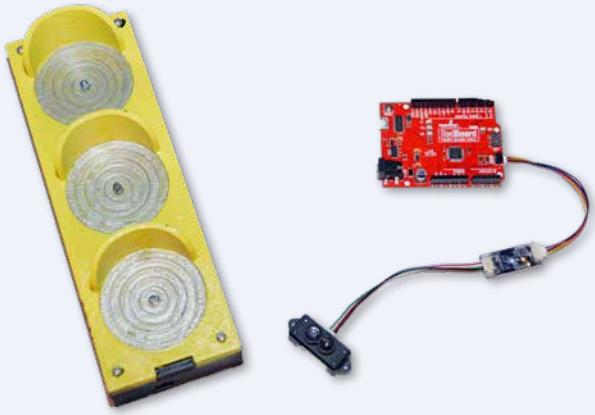

Figure 2. Les composants électroniques et le coffret imprimé en 3D de ce projet

Listage 1. Programme Perfect Parking with LiDar

```
/*
  TFMiniStopLight.ino
  Rob Reynolds, November 19, 2018

  This code is a small practical demonstration
application of the TFMini Lidar Module. The 3D
files can be found in the Github repository, here
  [ https://github.com/ThingsRobMade/TFMini_Stop_
Light ]
  Based heavily on the previous collaborative work
done by Nate Seidle and Benewake. The original
example sketch for the Qwiic Enabled TFMini can be
found here:
  (https://www.sparkfun.com/products/14786)

  This code is free, but if you find it useful,
and we meet someday, you can buy me a beer
(Beerware license).
*/

#include <Wire.h>

uint16_t distance = 0; //distance
uint16_t strength = 0; //signal strength
uint8_t rangeType = 0; //range scale
/*Value range:
  00 (short distance)
  03 (intermediate distance)
  07 (long distance) */

boolean valid_data = false;
//ignore invalid ranging data

const byte sensor1 = 0x10;
//TFMini I2C Address

// Define pins for LEDs
const int greenLED = 8;
const int yellowLED = 9;
const int redLED = 10;
int stopLimit = 160; //Change this number of cm to
adjust stop distance

void setup()
{
  Wire.begin();

  Serial.begin(115200);
  Serial.println("TFMini I2C Test");
  //For testing using Serial Monitor

  // Set LED pins as outputs
  pinMode(redLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
}

void loop()
{
  if (readDistance(sensor1) == true)
  {
    if (valid_data == true) {
      Serial.print("\stopLimit[");
/* These Serial.print lines remain for testing and
adjustment purposes */
      Serial.print(stopLimit);
      Serial.print("\tDist[");
      Serial.print(distance);
      Serial.println();

      if (distance <= stopLimit) {
        digitalWrite(redLED, HIGH);
        digitalWrite(yellowLED, LOW);
        digitalWrite(greenLED, LOW);
      }
      else if (distance > stopLimit && distance
< (stopLimit + 200) ) { //change this number to
increase distance yellow stays lit
        digitalWrite(redLED, LOW);
        digitalWrite(yellowLED, HIGH);
        digitalWrite(greenLED, LOW);
      }
      else if (distance > (stopLimit + 199) ) {
//change this number to adjust when yellow LED
illuminates
        digitalWrite(redLED, LOW);
        digitalWrite(yellowLED, LOW);
        digitalWrite(greenLED, HIGH);
      }
    }

    //  else {
    //   Serial.println("Read fail");
    //  }

    delay(50);
//Delay small amount between readings
  }
}


//Write two bytes to a spot
boolean readDistance(uint8_t deviceAddress)
{
```

Et si on remplaçait le RedBoard par un RPi pour programmer ce projet en Python ?

Rob Reynolds

```
  Wire.beginTransmission(deviceAddress);
  Wire.write(0x01); //MSB
  Wire.write(0x02); //LSB
  Wire.write(7); //Data length: 7 bytes for
distance data
  if (Wire.endTransmission(false) != 0) {
    return (false); //Sensor did not ACK
  }
  Wire.requestFrom(deviceAddress, (uint8_t)7); //
Ask for 7 bytes

  if (Wire.available())
  {
    for (uint8_t x = 0 ; x < 7 ; x++)
    {
      uint8_t incoming = Wire.read();

      if (x == 0)
      {
        //Trigger done
        if (incoming == 0x00)
        {
          //Serial.print("Data not valid: ");
          //for debugging
          valid_data = false;
          //return(false);
        }
        else if (incoming == 0x01)
        {
          Serial.print("Data valid:     ");
          valid_data = true;
        }
      }
      else if (x == 2)
        distance = incoming;
//LSB of the distance value "Dist_L"
      else if (x == 3)
        distance |= incoming << 8; //MSB of the
distance value "Dist_H"
      else if (x == 4)
        strength = incoming; //LSB of signal
strength value
      else if (x == 5)
        strength |= incoming << 8; //MSB of signal
strength value
      else if (x == 6)
        rangeType = incoming; //range scale
    }
  }
  else
  {
    Serial.println("No wire data avail");
    return (false);
  }

  return (true);
}
```

Accessoires

Vous trouverez chez Elektor et SparkFun les principaux articles mentionnés dans cet article.

- **RedBoard - Programmé avec Arduino**
  www.elektormagazine.fr/esfe-en-perfectparking1
- **TFMini - Micro Module LiDAR**
  www.elektormagazine.fr/esfe-en-perfectparking2

LIENS

[1] TFMini - Module Micro LiDAR : https://www.sparkfun.com/products/14588
[2] Guide de connexion de TFMini Qwiic : https://bit.ly/2Xzun4r
[3] Sketch de Perfect Parking, fichiers .STL du coffret : https://github.com/ThingsRobMade/TFMini_Stop_Light