Parking sensor

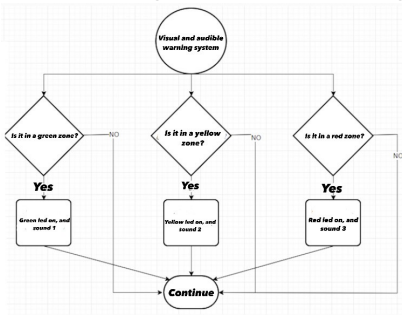
Introduction

Carlos Javier del Nero Castón

The parking sensor with Arduino allows us to measure distances using ultrasound. The operation is very simple. The sensor sends an ultrasonic wave through the trigger, it bounces against the object and the receiver or echo detects the wave. Knowing how long the wave has taken to travel, we can know the distance.

General description

COMPUTATIONAL THINKING The first thing is a brief description of what we want to achieve. This will help us to state the general problem and then break it down into smaller chunks. This is called computational thinking and if you are a regular user of this blog, you will already be familiar with this concept. The parking system consists of detecting an object through the ultrasonic sensor and alerting with light and sound signals. Therefore, we already have the first division, detecting the obstacle and alerting with sound and lights. The first thing I will do is to set out the algorithm of the obstacle detection system.



1. Check the distance of objects

```
1. Is it within range to warn?
  1.Yes
    1. Trigger visual and audible alarm
    2. Continue
  2. No
    1. Continue
```

The algorithm of the visual and audible warning system would be as follows.

1. Are you in the green zone?

```
1. Yes
  1. Turn on green LED
  2. Emit sound 1
  3. Exit
2. No
  1. Continue
```

2. Are you in the yellow zone?

```
1. Yes
  1. Turn on yellow LED
  2. Sound 2
  3. Exit
2. No
  1. Continue
```

3. Are you in the red zone?

```
1. Yes
  1. Turn on red LED
  2. Emit sound 2
  3. Exit
2. No
  1. Continue
```

From the two previous algorithms, we deduce that we will need several decision thresholds, one for each situation.

1. Threshold 1: is in the green zone from 30 cm to 20 cm.
2. Threshold 2: is in the yellow zone, from 20 cm to 10 cm.
3. Threshold 3: is in the red zone, less than 10 cm.

Hardware Design

- Arduino ultrasonic sensor (HC-SR04)

To measure distances with Arduino we can do it in different ways. There is the infrared sensor, which uses the properties of light to calculate distance, and the Arduino ultrasonic sensor uses the propagation properties of sound to measure distances. More specifically it uses ultrasound. This type of sound waves are above the spectrum audible to humans. The operation is very simple. The sensor sends an ultrasonic wave through the trigger, it bounces off the object and the receiver or echo detects the wave. By knowing how long the wave has taken to travel, we can find out the distance.

- Buzzer

To correctly simulate the distance sensor we are going to use an Arduino buzzer. These components use piezoelectricity, a physical phenomenon that affects certain crystals (quartz is the most common). When a crystal of this type is subjected to piezoelectricity, it deforms and vibrates. If we get that vibration to have a frequency within the audible spectrum, we get a sound.

- 1 green LED, 1 yellow LED, 1 red LED

Finally, we incorporate the visual alert system for the Arduino ultrasonic sensor. This allows us to visualise whether we are near or far from an obstacle. With 3 LEDs (green, yellow and red).

- Arduino UNO
- Protoboard

Where we will connect the components.

- Cables

To make the connections.

- 3 Resistors

Of 220 Ω

ELECTRIC SCHEMATIC

Software Design

- **VARIABLES AND CONSTANTS**

Through the ultrasonic sensor we are going to detect the obstacle. We start programming by declaring the variables and constants.

We define the pins for the LEDs, for the ultrasonic sensor and for the Arduino buzzer. Then we declare 4 constants. The first one is the speed of sound converting from metres per second to centimetres per second. We do this by multiplying by 100. The next constants are the decision thresholds we marked earlier.

- **SET UP FUNCTION**

In the setup function we start the serial monitor and set the pins to the corresponding mode. The LEDs, the Trigger of the ultrasonic sensor and the buzzer are in output mode (OUTPUT). The Echo pin of the ultrasonic sensor is in INPUT mode.

- **LOOP FUNCTION**

The loop() function contains the code that will be repeated over and over again. This is where we are going to put all our algorithm, the one we have detailed above. I have split this function into several functions to make the code more readable.

- **START ULTRASONIC SENSOR**

The first thing we do is to prepare the ultrasonic sensor. We do this with the function initiateTrigger() which sends a pulse. It starts in the low state for 2 milliseconds, then 10 milliseconds in the high state and finally we put it in the low state. This indicates that the signal will then be sent to be picked up by echo.

- **CALCULATE DISTANCE OF OBJECTS**

Once the sensor is ready, we can use it to calculate the distance. We do this with the function

calculateDistance(). It is a particular function as it will return a value. This is done by putting at the end (or wherever you want) the reserved word return followed by the value you want to return. All code below the return is not executed, so be careful. In this case I return the calculated distance inside the function which is a float variable. The pulseIn function detects just that, and returns the elapsed time in microseconds. For this reason, it must be converted to seconds by multiplying by 0.000001, which is the same as dividing by 1,000,000. With this information we can now apply the formula to calculate the distance as a function of time and speed.

- **TRIGGER ALERTS**

Once we have the distance calculated, we can decide whether we are in the situation to send an alert or not. Whenever the distance is below the first threshold (green LED threshold and the least restrictive), we will launch the corresponding visual and audible alert.

- **FINAL CODE**

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Conclusions

Testing the car sensor sensor car makes it easy for car drivers to park the car.

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună .

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF

Senzor de parcare

Introducere

Acest proiect are ca scop relizarea unui senzor de parcare. Cu ajutorul unui buzzer va fi semnalizata apropierea de un obiect. Ecranul LCD-ul va afisa distanta fata de cel mai apropiat obiect detectat.

Descriere generală

O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o descriere a acestora precum şi a modului în care interacţionează.

Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html

Hardware Design

List of components:

- Arduino UNO - Protoboard where we will connect the components - Cables to make the connections - 3 resistors of 220 Ω - 1 green LED - 1 yellow LED - 1 red LED - 1 Arduino ultrasonic sensor (HC-SR04) - 1 buzzer.

Software Design

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare (if any) (e.g. AVR Studio, CodeVisionAVR)
- librării şi surse 3rd-party (e.g. Procyon AVRlib)
- algoritmi şi structuri pe care plănuiţi să le implementaţi
- (etapa 3) surse şi funcţii implementate

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

===== Jurnal =====

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

===== Bibliografie/Resurse =====

<note>