Relazione laboratorio di making
Rete di sensori per auto

Gianluca Lutero

February 2020

Introduzione

Il progetto consiste nella realizzazione di una rete di sensori da poter utilizzare in automobile. La rete è realizzata da un sensore di parcheggio e da un sensore ambientale che rilevano i rispettivi dati e li inviano al controller che li elabora e li visualizza su un display.

Componenti utilizzate

Di seguito vengono mostrati i componenti usati per ogni modulo della rete.

Controller

- 1 Arduino Uno
- 1 modulo nRF24L01
- 1 led RGB
- 1 buzzer
- 1 display lcd
- 3 resistenze da 2.7 kΩ
- 1 bottone

Sensore di parcheggio

- Arduino Uno
- 1 modulo nRF24L01
- 3 moduli HC-SR04

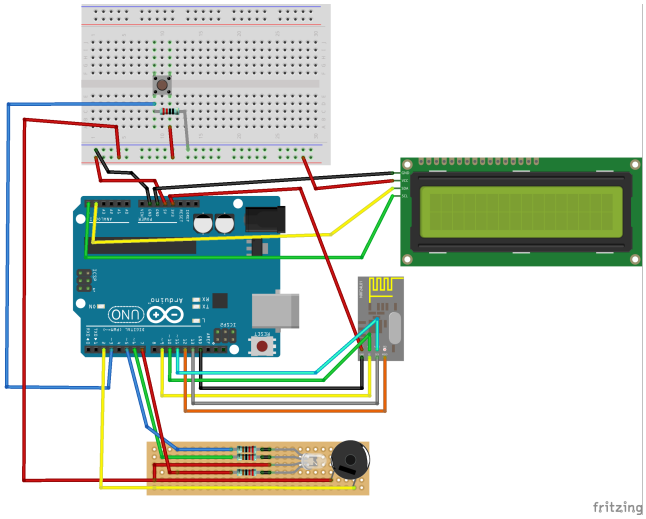
Sensore ambientale

- Arduino NANO
- 1 modulo nRF24L01
- 1 fotoresistore
- 1 resistenza da 10 kΩ
- 1 modulo DTH11

Realizzazione dei moduli

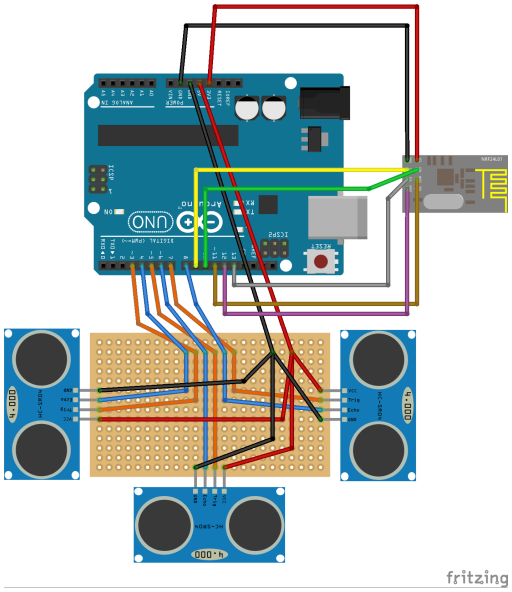
Controller

Il controller è il modulo centrale della rete, il suo compito è quello di ricevere i dati dal sensore di parcheggio e dal sensore ambientale attraverso il modulo radio nRF24L01 elaborarli e visualizzare i risulti sul display. I dati che possono essere visualizzati sono, alternativamente, le distanze istantanee dei sensori di parcheggio e la temperatura, l'umidità e la luminosità circostante del sensore ambientale. Per passare da una schermata ad un'altra del display si usa un bottone. Per avvisare di un imminente collisione rimane sempre attivo un led che varia il proprio colore da verde, che indica nessun ostacolo in prossimità, blu, che indica che vi è un ostacolo vicino, e rosso, che indica che l'ostacolo si trova ad una distanza inferiore ai 10cm. Inoltre è stato posto assieme al led un buzzer che inizia a suonare con maggiore frequenza nel momento in cui ci si avvicina all'ostacolo. Di seguito viene riportato lo schema elettrico del modulo.



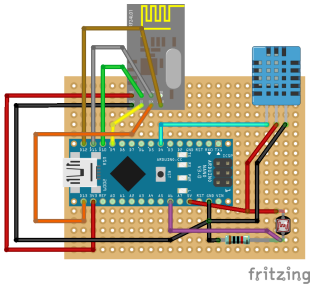
Sensore di parcheggio

Il sensore di parcheggio ha il compito di rilevare la distanza da possibili ostacoli. Il modulo è dotato di tre sensori ad ultrasuoni disposti su tre lati di una basetta millefori per coprire una maggiore area. Di seguito lo schema elettrico del sensore.



Sensore ambientale

Il sensore ambientale ha il compito di rilevare le condizioni ambientali esterne. I dati rilevati dal modulo sono la temperatura, l'umidità e la luminosità esterna. Di seguito lo schema elettrico del sensore.



Protocollo di comunicazione tra i moduli

I moduli sopra descritti comunicano tra di loro attraverso i chip nRF24L01 settati rispettivamente come solo ricevente, per il controller, e solo come trasmit-

tente per gli altri due moduli. I moduli trasmittenti inviano al ricevente i dati raccolti e un id univoco utilizzato per distinguere da chi sono stati inviati i dati.

Software

Per programmare i microcontrollori Arduino è stato usato il software Arduino IDE. Per poter utilizzare il codice seguente è necessario prima installare le seguenti librerie per Arduino:

- **RF24**: per utilizzare i chip nRF24L01
- **DHT11**: per utilizzare il chip dht11
- **LiquidCrystal_I2C**: per utilizzare il display

Di seguito viene mostrato il codice dei moduli contenuto anche in questo repository.

Controller

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#include <LiquidCrystal_I2C.h>

// Modulo nrf24l01
#define CE 9
#define SCN 10
#define SCK 13
#define MOSI 11
#define MISO 12

// Display I2C
#define SDA A4
#define SCL A5

// LED RGB
#define RED 7
#define GREEN 6
#define BLUE 5

// Buzzer
#define BUZZER 2

// Button
#define BUTTON 3

RF24 radio(CE, SCN);
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
// Imposto l'indirizzo di comunicazione
const byte address[6] = "00001";

// Imposto il numero massimo di trasmittenti
const int n_transmitter = 2;

int button_state = 0;
int select = 0;

float parking[3];
float ambiental[3];

// Funzione per impostare il colore del led
void setColor(int redValue, int greenValue, int blueValue) {
  analogWrite(RED, redValue);
  analogWrite(GREEN, greenValue);
  analogWrite(BLUE, blueValue);
}

void setup() {

  Serial.begin(9600);

  // Inizializzo il display lcd
  lcd.init();
  lcd.backlight();

  // Inizializzo il modulo radio
  radio.begin();
  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();

  // Inizializzo il led
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);

  // Inizializzo il buzzer
  pinMode(BUZZER, OUTPUT);

  // Inizializzo il bottone
  pinMode(BUTTON,INPUT);

  // Inizializzo le variabili dati
  for(int i=0;i<3;++i){
    parking[i] = 9999;
    ambiental[i] = 9999;
  }
```

```
}

void loop() {
  int id;

  button_state = digitalRead(BUTTON);

  if(button_state == HIGH){
    select = (select + 1) % n_transmitter;
    Serial.print("Mostro i dati relativi alla trasmittente: ");
    Serial.println(select);
  }

  // Verifico che ci siano dei dati in arrivo
  if (radio.available()) {
    float data[4];

    // Ricevo i dati
    radio.read(&data, sizeof(data));

    id = data[0];

    Serial.print("Id sensore: ");
    Serial.println(data[0]);

    switch (id) {
      case 1:
        // Aggiorno i dati ambientali con quelli ricevuti
        ambiental[0] = data[1];
        ambiental[1] = data[2];
        ambiental[2] = data[3];

        Serial.print("Temperatura: ");
        Serial.println(data[2]);

        Serial.print("Umidita' : ");
        Serial.println(data[3]);

        Serial.print("Luminosita' : ");
        Serial.println(data[1]);

        break;

      case 2:

        // Aggiorno le distanze con quelli ricevuti
        parking[0] = data[1];
        parking[1] = data[2];
```

```
      parking[2] = data[3];

      // Accendo il led in base alla distanza
      float min = data[1];

      // Cerco tra i valori la distanza pi piccola
      for (int i = 2; i <= 3; ++i) {
        if (data[i] <= min) {
          min = data[i];
        }
      }

      if (min >= 60) {
        setColor(255, 0, 255);
      }

      if (min >= 10 && min < 60) {
        setColor(255, 255, 0);
        tone(BUZZER, 1000, 50 + min);
      }

      if (min < 10) {
        setColor(0, 255, 255);
        tone(BUZZER, 1000, 500);
      }

      Serial.print("Dist 00: ");
      Serial.println(data[1]);
      Serial.print("Dist 01: ");
      Serial.println(data[2]);
      Serial.print("Dist 02: ");
      Serial.println(data[3]);
  }
}

// Stampo i dati sul display in base alla scelta
lcd.clear();
lcd.setCursor(0, 0);

switch(select){
  case 0:

      lcd.print("T: ");
      lcd.print(ambiental[1]);

      lcd.print("U: ");
      lcd.print(ambiental[2]);

      lcd.setCursor(0,1);
```

```
      if(ambiental[0] < 200)
         lcd.print("Buio");
      else
         lcd.print("Luminoso");
    break;
  case 1:

      if (parking[0] <= 10 || parking[1] <= 10 || parking[2] <= 10) {
        lcd.print("   Pericolo   ");
        lcd.setCursor(0, 1);
        lcd.print("   Ostacolo   ");
      } else {
        lcd.print("L:");
        lcd.print(parking[0]);
        lcd.print(" C:");
        lcd.print(parking[1]);
        lcd.setCursor(0, 1);
        lcd.print("R:");
        lcd.print(parking[2]);
      }
    break;
 }

 delay(1000);

}
```

Sensore di parcheggio

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

// Moduli ad ultrasuoni
#define TRIG00 8
#define ECHO00 7
#define TRIG01 6
#define ECHO01 5
#define TRIG02 4
#define ECHO02 3

// Modulo nrf24l01
#define CE 9
#define SCN 10
#define SCK 13
#define MOSI 11
#define MISO 12
```

```
RF24 radio(CE,SCN);

// Imposto l'indirizzo di comunicazione
const byte address[6] = "00001";

// Imposto l'id univoco del sensore
const int id = 2;

// Funzione per calcolare la distanza rilevata dal modulo ad ultrasuoni
float distance(int trig, int echo){
  int duration = 0;
  float distance = 0;

  digitalWrite(trig,LOW);
  delayMicroseconds(2);
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);

  duration = pulseIn(echo,HIGH);
  distance = (duration*0.034)/2;

  return distance;
}

void setup() {

  // Left sensor
  pinMode(TRIG00,OUTPUT);
  pinMode(ECHO00,INPUT);

  // Center sensor
  pinMode(TRIG01,OUTPUT);
  pinMode(ECHO01,INPUT);

  // Right sensor.h
  pinMode(TRIG02,OUTPUT);
  pinMode(ECHO02,INPUT);

  // Imposto il modulo nRF24L01
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();

  Serial.begin(9600);
  Serial.println("Distanze misurate dal sensore ad ultrasuoni:");

}
```

```
void loop() {
  float dist00 = 9999;
  float dist01 = 9999;
  float dist02 = 9999;
  float data[4];

  // Rilevo le distanze dai tre sensori
  Serial.print("Distanza sensore 00:");
  dist00 = distance(TRIG00,ECHO00);
  Serial.println(dist00);

  Serial.print("Distanza sensore 01:");
  dist01 = distance(TRIG01,ECHO01);
  Serial.println(dist01);

  Serial.print("Distanza sensore 02:");
  dist02 = distance(TRIG02,ECHO02);
  Serial.println(dist02);

  data[0] = id;
  data[1] = dist00;
  data[2] = dist01;
  data[3] = dist02;

  // Invio i dati
  radio.write(&data,sizeof(data));
  delay(1000);

  if (dist00 <= 10 || dist01 <= 10 || dist02 <= 10){
    Serial.println("Troppo vicino!!!");

  }

}
```

Sensore ambientale

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

#include <dht11.h>

// Modulo nrf20l01
#define CE 9
#define SCN 10
#define SCK 13
#define MOSI 11
```

```
#define MISO 12

// Sensore DTH11
#define S 4

RF24 radio(CE,SCN);

// Imposto l'indirizzo di comunicazione
const byte address[6] = "00001";

// Imposto l'id univoco
const int id = 1;

dht11 DHT11;


void setup() {

  Serial.begin(9600);

  // Inizalizzo il modulo radio
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();

  // Inizializzo modulo temperatura
  pinMode(S,INPUT);

  // Inizializzo fotoresistore
  pinMode(A6,INPUT);
}

void loop() {
  float temp;
  float umidity;
  float light;

  // Leggo la luminosit ambientale
  light = analogRead(A6);

  // Leggo la temperatura e l'umidit dal sensore dht11
  int chk = DHT11.read(S);

  float data[4];

  data[0] = id;
  data[1] = light;
  data[2] = (float)DHT11.temperature;
  data[3] = (float)DHT11.humidity;
```

```
  // Invio i dati
  radio.write(&data,sizeof(data));

  Serial.print("Luminosita': ");
  Serial.println(light);

  Serial.print("Temperatura : ");
  Serial.println((float)DHT11.temperature);

  Serial.print("Umidita' : ");
  Serial.println((float)DHT11.humidity);

  delay(5000);


}
```