10_Pacchetto_RF24_Arduino_Relazione

Citterio Giorgio e Colombo Umberto

Lo scopo di quest'attività è convertire i programmi da seriale a programmi per interfacciarsi con il modulo radio RF24.

Attività preliminare

In questa prima parte abbiamo collegato il modulo radio alle breadboard su arduino seguendo il seguente link.

> Arduino e RF24
>
> Prima configurazione di Arduino con nRF24L01+
>
> https://www.vincenzov.net/tutorial/elettronica-di-base/Trasmissioni/laboratorio/RF24a.htm

Il programma testato è il seguente:

```
#include <RF24.h>
RF24 radio(7, 8); // Imposta CE e nCSN conformemente all'hardware

void setup() {
  Serial.begin(9600);
  radio.begin();
}

void loop() {
  if (radio.isChipConnected())
     Serial.println ("nRF24L01p presente");
  else
     Serial.println ("nRF24L01p non rilevato");
  delay(1000);
}
```

parte 1

In questa prima attività abbiamo modificato il programma del sensore su arduino per potersi interfacciare col modulo radio RF24, e inviare il pacchetto.

sketch sensore arduino:

```
#include <RF24.h>
RF24 radio(7, 8);
```

```
#define ID "BE"
#define TIPO "S1"
#define MITTENTE "M001"
#define DESTINATARIO "P001"
#define WRITINGPIPE "00001"
#define VUOTO "----------------"

struct pacchettoS1 {
  char id[2];
  char mittente[4];
  char destinatario[4];
  char tipo[2];
  char valoreSensore[4];
  char vuoto[16];
};

void setup() {
  Serial.begin(9600);

  radio.begin();
  radio.setPALevel(RF24_PA_MIN);
  radio.setPayloadSize(32);
  radio.setDataRate(RF24_2MBPS);
  radio.openWritingPipe((byte *)WRITINGPIPE);
  radio.stopListening();
}

void loop() {
  int num = analogRead(A0);
  char s[5];
  sprintf(s, "%04d", num);

  struct pacchettoS1 msg;
  memcpy(msg.id, ID, sizeof(msg.id));
  memcpy(msg.mittente, MITTENTE, sizeof(msg.mittente));
  memcpy(msg.destinatario, DESTINATARIO, sizeof(msg.destinatario));
  memcpy(msg.tipo, TIPO, sizeof(msg.tipo));
  memcpy(msg.valoreSensore, s, sizeof(msg.valoreSensore));
  memcpy(msg.vuoto, VUOTO, sizeof(msg.vuoto));

  Serial.write((char *)&msg, sizeof(msg));
  Serial.println();

  radio.write((char *)&msg, sizeof(msg));

  delay(2500);

}
```

---

parte 2

In questa seconda parte dell’attività abbiamo modificato il programma dell’attuatore su arduino per potersi interfacciare col modulo radio RF24, ricevere il pacchetto ed elaborarlo.

sketch attuatore arduino:

```
#include <RF24.h>
RF24 radio(7, 8);

#define ID "EP" //BE
```

```
#define TIPO "A1"
#define DESTINATARIO "P438"  //A001
#define READINGPIPE "00001"

struct pacchettoA1 {
  char id[2];
  char mittente[4];
  char destinatario[4];
  char tipo[2];
  char direzione[1];
  char velocita[3];
  char vuoto[16];
};

void setup() {
  Serial.begin(9600);
  pinMode(9, OUTPUT);
  pinMode(5, OUTPUT);

  radio.begin();
  radio.setPALevel(RF24_PA_MIN);
  radio.setPayloadSize(32);
  radio.setDataRate(RF24_2MBPS);
  radio.openReadingPipe(0, (byte *) READINGPIPE);
  radio.startListening();
}

void loop() {
  struct pacchettoA1 msg;
  if (radio.available())
  {
    radio.read((char*) &msg, sizeof(msg));
    int controlloId = memcmp(ID, msg.id, 2);
    int controlloDest = memcmp(DESTINATARIO, msg.destinatario, 4);
    char vel[4];
    if (controlloId == 0 && controlloDest == 0)
    {
      Serial.println((char *)&msg);
      memcpy(vel,msg.velocita,sizeof(msg.velocita));
      vel[3] = '\0';
      int velocita = atoi(vel);
      if (memcmp("A",msg.direzione, 1) == 0)
      {
        digitalWrite(5, LOW);
        digitalWrite(9, HIGH);
        analogWrite(3, velocita);
      }
      if (memcmp("I",msg.direzione, 1) == 0)
      {
        digitalWrite(5, HIGH);
        digitalWrite(9, LOW);
        analogWrite(3, velocita);
      }
      if (memcmp("S",msg.direzione, 1) == 0)
      {
        digitalWrite(5, LOW);
        digitalWrite(9, LOW);
        analogWrite(3, 0);
      }
    }
  }
}
```