Mouse

Documentazione Tecnica

Calcaterra Simone, Negri Matteo

Febbraio 2018

Indice

Capitolo 1

Introduzione

L'idea del progetto è quella di realizzare un mouse motorizzato in grado di compiere tragitti specificati a priori.
La particolarità di questo progetto sta nel fatto che viene utilizzata la tecnologia di un mouse PS2 in maniera tale da compiere tragitti con distanze precise.
Per fare questo utiliziamo la tecnologia del mouse sottostante.

1.1 Riferimenti

Per la realizzazione della documentazione e del progetto abbiamo fatto uso di risorse presenti in internet.
Per quanto riguarda la documentazione del protocollo PS2 il sito di riferimento Ã¨ stato http://www.computer-engineering.org/ps2protocol/.
La libreria utilizzata per l'implementazione del progetto l'abbiamo scaricata dal seguente link https://github.com/jazzycamel/PS2Mouse.

Capitolo 2

Il protocollo PS2 del mouse

2.0.1 Introduzione

Per la realizzazione del progetto abbiamo studiato ed analizzato il protocollo dal punto di vista della comunicazione tra Host e mouse e dal punto di vista delle risposte ad un dato input.

2.0.2 Il Protocollo

Il formato di un pacchetto per la comunicazione del mosue é composto da 3 byte.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X movement							
Byte 3	Y movement							

Figura 2.1: Pacchetto Mouse

Grazie alla presenza del bit di segno, sia di y che di x e grazie al fatto che la coordinata, qualsiasi essa sia, é di 2^8 possiamo rappresentare valori che vanno da -256 a +256.
Possiamo quindi mediante il frame atto a far comunicare Host e Mouse ottenere oltre alle coordinate anche le informazioni riguardanti i tasti del mouse, grazie ai primi tre bit del primo byte.

2.0.3 Le modalitá operative

- RESET L'host invia un segnale di reset 0xFF al mouse, il mouse legge tale segnale anche come un comando di diagnostica, questa operazione si chiama BAT, il mouse risponde con 0xAA se tutto é aposto o con 0xFC altrimenti.

Ottenuta la risposta del mouse l'host invierá 0xAA per dire al mouse che é tutto printo per la comunicazione e il mouse risponderá prima con il suo id, dopo di che inizierá a trasmettere (modalitá di stream).

- STREAM é la modalitá operativa, da questo momento il mouse e l'host comunicano a tutti gli effetti.

- REMOTE Fa polling per ottenere istruzioni.

- WRAP modalitá diagnostica dove viene fatto un echo di tutti i pacchetti ricevuti.

2.0.4 I principali comandi

- Host invia 0xFF(reset) $\rightarrow$ Il mosue risponde con 0xFA

- Host invia 0xFE(resend) $\rightarrow$ (Viene inviato quando il mouse risponde con un valore diverso da 0xFA, gli viene quindi reinviato il medesimo pacchetto) Se il mouse funziona correttamente risponde con 0xFA se risponde ancora con un valore diverso l'host invia un segnale di riavvio del dispositivo con 0xFC

- Host invia 0xF6(set default) $\rightarrow$ Il mouse risponde con 0xFA dopo di che configura i suoi parametri, sensibilitá, rapporto, ecc.. con valori di default.

- Host invia 0xF5 $\rightarrow$ Disabilita gli ack

- Host invia 0xF4 $\rightarrow$ Abilita gli ack.

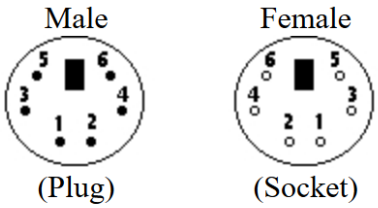
- Host invia 0xF2(get device) $\rightarrow$ Il mouse risponde con 0xFA poi risponde con il proprio ID

- Host invia 0xF0(Stream Mode) $\rightarrow$ Il mosue risponde con 0xFA

- Host invia 0xEB (read) $\rightarrow$ Il mouse risponde con 0xFA e poi inizia la condivisione dei dati

- Host invia 0xEA $\rightarrow$ Il mouse risponde con 0xFA e poi azzera i suoi registri e si mette in modalitá stream.

2.0.5 Electrical Interface



6-pin Mini-DIN (PS/2):
1 - Data
2 - Not Implemented
3 - Ground
4 - Vcc (+5V)
5 - Clock
6 - Not Implemented

Figura 2.2: Interfaccia

Vcc = +4.5V to 5.5V
Max Current = 275 mA

2.0.6 Comunicazione

Possono verificarsi tre tipologie di configurazioni di segnali sul bus dati e clock.
Data = HIGH, Clock = HIGH → Idle state
Data = HIGH, Clock = LOW → comunicazione inibita
Data = LOW, Clock = HIGH → l'host richiede di trasmettere
La frequenza del clock va da 10 a 16 kHz.
Tutti i dati vengono trasmessi un byte alla volta e ogni byte viene inviato in un frame costituito da 11-12 bit.

- 1 bit di partenza, di solito 0.
- 8 bit di dati.
- 1 bit di paritá.
- 1 bit di stop, di solito 1.
- 1 bit di ack, hosto → device.

Device to Host

Quando il device vuole inviare dati, prima di tutto verifica se il clock é a livello alto, se lo fosse invia, altrimenti salva sul buffer quello che deve inviare fino a quando non lo ha inviato. Il tutto avviene con lo schema appena visto.

- 1 bit di partenza, di solito 0.

- 8 bit di dati.
- 1 bit di paritá.
- 1 bit di stop, di solito 1.

Host to Device

L'invio dei dati da un host ad un device, é molto simile al precedente cambia in alcuni passaggi.
Il device al quale vogliamo inviare i dati possiede un suo clock, é quindi necessario portare le linee di clock e data in modalitá ascolto, modalitá "Requesto to send".
Le fasi che svolge l'host per comunicare con il device sono:

1. Porta la linea clock a segnale basso per almeno 100 microsecondi.
2. Porta la linea data a segnale basso.
3. Rilascia il clock.
4. Asetta che il device porti a 0 logico il segnale del clock.
5. Invia il dato al device, il primo bit.
6. Aspetta che il segnale del clock torni alto
7. Aspetta che il segnale del clock torni basso
8. Ripeti il punto 5,6,7
9. Rilascia la linea dati
10. Ascpetta che il device porti la linea data a segnale basso.
11. Aspetta che il device porti la linea del clock a segnale basso
12. Aspetta che il device rilasci la linea data e la linea clock.

Capitolo 3

La libreria

3.1 introduzione

Il seguente capitolo ad analizza il codice della libreria utilizzata per la realizzazione del progetto. Analizziamo funzione per funzione andando a capire perché vengono svolte alcune operazioni e come esse trovano riscontro nella documentazione del capitolo precedente.

3.2 La libreria

3.2.1 setup...

Invio un segnale di reset 0xFF al device mediante il metodo write leggo i 3 byte di risposta, del quale non mi preoccupo dato che sono solo di diagnostica Con il comando 0xF0 setto il mouse in modalitÃ streram Legge l'ack Aspetta 100 microsecondi in quanto essendo il clock del device di 10 - 16.7 kHz, significa che il clock resterá per 30-50 microsecondi ad un valore basso e 30-50 microsecondi ad un valore alto formando cosí un periodo che va dai 60 ai 100 microsecondi

```
1     void PS2Mouse::begin(void){
2         write(0xFF);
3         for(int i=0; i<3; i++) read();
4           write(0xF0);
5         read();
6         delayMicroseconds(100);
7   }
8
```

3.2.2 write...

Il compito di questo metodo Ã¨ quello di scrivere, ossia di effettuare una comunicazione tra Host e device. Per prima cosa settiamo i gpio di data e clock come gpio di lettura (input) e a valore HIGH, dopo di che aspettiamo un tempo di 300 microsecondi, tempo arbitrario in quanto sono ammessi tutti i valori purchÃ© superiori a

100 microsecondi. Successivamente effettuiamo i punti 1,2,3 della documentazione, ossia portiamo il segnale del clock e dati a livello LOW, dopo di che rilasciamo il clock. Fatto questo aspettiamo che il segnale del clock torni LOW. A questo punto possiamo "scrivere" sul dispositivo il comando desiderato Dato che bisogna inviare 8 bit effettuiamo un for nel quale: Se il bit Ã¨ da mandare, l'if Ã¨ verificato ed invia un segnale HIGH sulla linea dati, LOW altrimenti aspetta un periodo di clock. Effettua uno xor del dato e il bit di paritÃ Shifta di un bit il dato da scrivere. Aspetto quindi che il device porti la linea dati e clock a livello LOW, dopo di che aspetto rilasci la linea clock

```
1     void PS2Mouse::write(uint8_t data){
2
3         uint8_t parity=1;
4
5         gohi(_ps2data);
6         gohi(_ps2clk);
7         delayMicroseconds(300);
8         golo(_ps2clk);
9         delayMicroseconds(300);
10        golo(_ps2data);
11        delayMicroseconds(10);
12        gohi(_ps2clk);
13
14        while(digitalRead(_ps2clk)==HIGH);
15
16        for(int i=0; i<8; i++){
17          if(data&0x01) gohi(_ps2data);
18          else golo(_ps2data);
19          while(digitalRead(_ps2clk)==LOW);
20          while(digitalRead(_ps2clk)==HIGH);
21          parity^=(data&0x01);
22          data=data>>1;
23        }
24
25        if(parity) gohi(_ps2data);
26        else golo(_ps2data);
27
28        while(digitalRead(_ps2clk)==LOW);
29        while(digitalRead(_ps2clk)==HIGH);
30
31        gohi(_ps2data);
32        delayMicroseconds(50);
33
34        while(digitalRead(_ps2clk)==HIGH);
35        while((digitalRead(_ps2clk)==LOW)||(digitalRead(_ps2data)==LOW))
      ;
36
37        golo(_ps2clk);
38      }
39
```

3.2.3 read...

Prima di tutto rendo i gpio di data e clock gpio d'ascolto, fatto questo aspetto metá periodo e attendo che il segnale del clock sia LOW dopo di che aspetto torni HIGH, compiendo cosí un periodo. fatto questo leggo il byte con un ciclo for: aspetto che il clock raggiunga il segnale LOW bit = legge il dato dallla linea data aspetto che il clock torni HIGH mediante il metodo bitWrite, nella libreria standard di arduino, scrivo in data i bit appena letti. aspetto due cicli di clock e rilascio il clock Il metodo si conclude con la restituzione del campo data.

```
uint8_t PS2Mouse::read(void){
    uint8_t data=0, bit=1;

    gohi(_ps2clk);
    gohi(_ps2data);
    delayMicroseconds(50);
    while(digitalRead(_ps2clk)==HIGH);

    delayMicroseconds(5);
    while(digitalRead(_ps2clk)==LOW);

    for(int i=0; i<8; i++){
      while(digitalRead(_ps2clk)==HIGH);
      bit=digitalRead(_ps2data);
      while(digitalRead(_ps2clk)==LOW);
      bitWrite(data,i,bit);
    }

    while(digitalRead(_ps2clk)==HIGH);
    while(digitalRead(_ps2clk)==LOW);
    while(digitalRead(_ps2clk)==HIGH);
    while(digitalRead(_ps2clk)==LOW);

    golo(_ps2clk);

    return data;
  }

```

3.2.4 getPosition...

Scriviamo il comando 0xEB, comando che comunica al mouse la necessitá di leggere i dati Il mouse risponderá con un messaggio di ack, 0xFA ma che a noi non interessa. Successivamente leggiamo i primi 8 bit, contenenti le informazioni presentate nella documentazione leggiamo il successivo byte, ossia quello contenente il valore della coordianta x. leggiamo il successivo byte, ossia quello contenente il valore della coordianta y. fatto ció verifichiamo la presenza o meno del segno della coordianta, prima x poi y. Infine poiniamo o meno il segno al valore mediante il metodo twos.

```
void PS2Mouse::getPosition(uint8_t &stat, int &x, int &y){
    write(0xEB);
```

```
3         read();
4         stat=read();
5         uint8_t _x=read();
6         uint8_t _y=read();
7
8         bool negx=bitRead(stat,4);
9         bool negy=bitRead(stat,5);
10        x=twos(_x, negx);
11        y=twos(_y, negy);
12      }
13
```

Capitolo 4

Il progetto

4.1 codice

Di seguito viene presentato il codice del progetto. L'idea é quella di far eseguire al mouse un tragitto preimpostato, in questo caso un quadrato.

```
1
2       #include "PS2Mouse.h"
3
4       PS2Mouse mouse(6,5);
5
6       int pos = 0;
7
8       void avviaMotore(int pV, int pG){
9         digitalWrite(pV,HIGH);
10        digitalWrite(pG,LOW);
11      }
12      void arrestaMotore(int pV, int pG){
13        digitalWrite(pV,LOW);
14        digitalWrite(pG,LOW);
15      }
16
17      void setup(){
18        Serial.begin(9600);
19        pinMode(8,OUTPUT);
20        pinMode(9,OUTPUT);
21        pinMode(10,OUTPUT);
22        pinMode(16,OUTPUT);
23        while(!Serial);
24        Serial.print("Setup...");
25
26        mouse.begin();
27        Serial.println("complete!");
28      }
29
30      void curvadx(){
```

```
31          uint8_t stat;
32          int x,y;
33          mouse.getPosition(stat,x,y);
34          pos = 0;
35          while(pos < 150){
36            uint8_t stat;
37            int x,y;
38            mouse.getPosition(stat,x,y);
39            Serial.print("\tx=");
40            Serial.print(x, DEC);
41            Serial.print("\ty=");
42            Serial.println(y, DEC);
43            avviaMotore(9,8);
44            avviaMotore(10,16);
45            pos += abs(y);
46          }
47          arrestaMotore(9,8);
48          arrestaMotore(10,16);
49
50        }
51        void dritto(int dist){
52          pos = 0;
53          while(pos < dist){
54            uint8_t stat;
55            int x,y;
56            mouse.getPosition(stat,x,y);
57            Serial.print("\tx=");
58            Serial.print(x, DEC);
59            Serial.print("\ty=");
60            Serial.println(y, DEC);
61            avviaMotore(8,9);
62            avviaMotore(10,16);
63            pos += abs(y);
64          }
65          arrestaMotore(8,9);
66          arrestaMotore(10,16);
67
68        }
69
70        void quadrato(int lato){
71          dritto(lato);
72          delay(1000);
73          curvadx();
74          delay(1000);
75          dritto(lato);
76          delay(1000);
77          curvadx();
78          delay(1000);
79          dritto(lato);
80          delay(1000);
81          curvadx();
82          delay(1000);
83          dritto(lato);
84          delay(1000);
```

```
85          curvadx();
86          delay(1000);
87        }
88
89
90        void loop(){
91          uint8_t stat;
92          int x,y;
93          mouse.getPosition(stat,x,y);
94
95          Serial.print(stat, DEC);
96          Serial.print("\tx=");
97          Serial.print(x, DEC);
98          Serial.print("\ty=");
99          Serial.println(y, DEC);
100
101         quadrato(500);
102         delay(10000);
103       }
104
```

4.2 schema

Il circuito Ã¨ composto dai seguenti dispositivi:

- Una board Pro Micro con integrato ATMEGA 32U4.
- Mouse con connessione PS2 cablato nel seguente modo:
  - Vcc connesso a Vcc
  - Gnd connesso a Gnd
  - Data line connessa al gpio 5
  - Clock line connessa al gpio 6
- due motori DC da 3.3V collegati nel seguente modo:
  - Vcc del motore 1 connesso al gpio 9
  - Gnd del motore 1 connesso al gpio 8
  - Vcc del motore 2 connesso al gpio 10
  - Gnd del motore 2 connesso al gpio 16
- quettro led per scopo diagnostico.

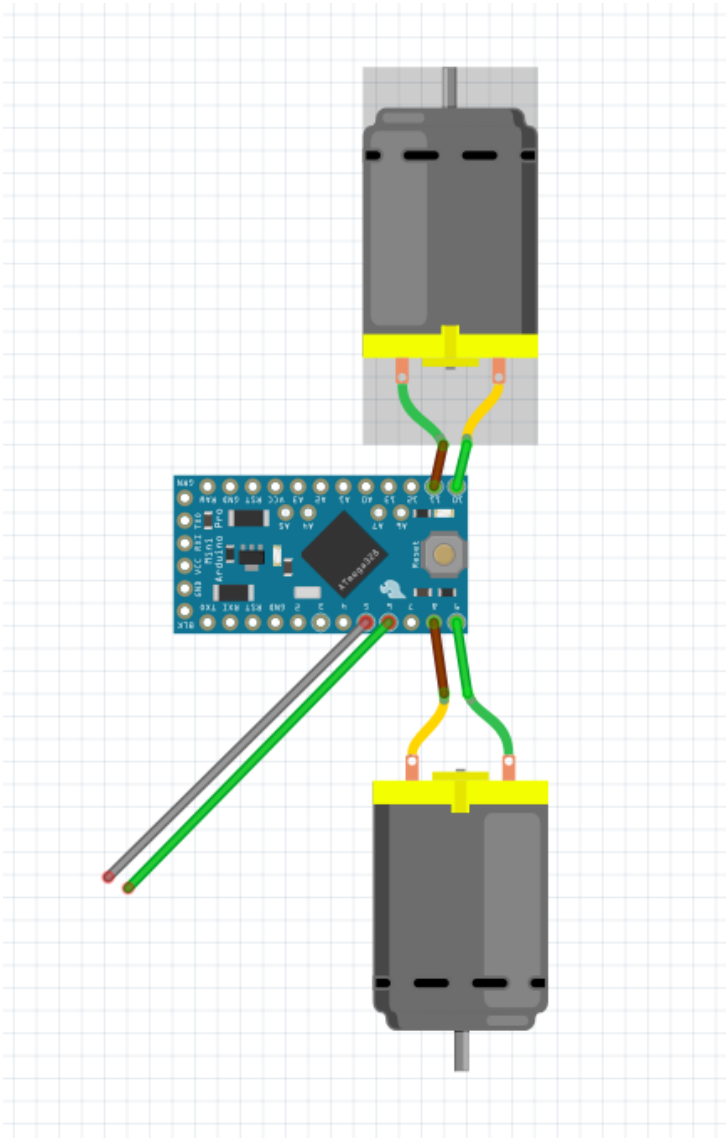


Figura 4.1: circuito

In figura vengono rappresentati i collegamenti per il funzionamento dei motori e due cavi che non sono collegati a nulla, in realtÃ rappresentano i cavi derivanti dal mouse, clock e data.