TSRR

IFAC-TSRR vol. 3 (2011) 1-10

Realizzazione di un sistema di controllo degli accessi con Arduino

A. Agostini ([1]), D. Andreuccetti ([1]), S. Cardone ([1]), R. Calzolai ([1]),

([1]) IFAC-CNR, Via Madonna del Piano 10, 50019 Sesto Fiorentino (FI), Italy

 ISSN 2035-5831

1 - Introduzione

Nella società attuale uno dei problemi che si pone frequentemente è il controllo degli accessi a tutti quei luoghi che, per qualche motivo, devono rimanere riservati soltanto agli autorizzati.

Questo problema coinvolge sia il settore privato che quello pubblico, ma è particolarmente sentito negli istituti di ricerca dove l'accesso ai laboratori di personale non addetto, e non qualificato, può originare problemi di sicurezza personale.

Proprio per questo motivo alcuni laboratori di utilità generale devono oggi poter essere utilizzati, anche in modalità "unattended", da personale generico (ancorché adeguatamente preparato), senza la necessità di prevedere un tecnico di laboratorio dedicato al servizio.

Il lavoro che viene illustrato in questo documento tratta di un particolare laboratorio dell'Istituto di Fisica applicata "Nello Carrara" (IFAC): l'officina meccanica.

La maggior parte dei laboratori presenti all'IFAC un tempo disponeva di un proprio addetto di profilo tecnico, incaricato di svolgere un servizio rivolto a tutto l'istituto o eventualmente ad un solo gruppo di ricerca (come nel caso di laboratori dedicati a specifiche attività scientifiche).

A causa della carenza di questi "addetti", l'officina meccanica deve adesso poter essere utilizzata anche da personale diverso dall'eventuale meccanico addetto, per svolgere piccoli lavori di pre-officina. È emersa quindi la necessità di predisporre un sistema di controllo che consenta l'ingresso all'officina solo al personale autorizzato dalla direzione.

L'edificio che ospita l'IFAC è situato all'interno di un'Area di Ricerca dotata di un sistema centralizzato di controllo degli accessi, ma integrarsi in tale sistema sarebbe risultato molto complesso e costoso, perché avrebbe richiesto di sostituire alcune centraline e di stendere i necessari cavi fino al punto interessato; inoltre, questo approccio non ci avrebbe dato la possibilità di gestire a livello di istituto le autorizzazioni all'accesso ed avrebbe comportato anche difficoltà per la gestione dei tabulati storici.

Poiché l'istituto è dotato di un proprio sistema di rilevamento delle presenze, agganciato ad un database del personale, si è pensato di sviluppare un dispositivo in grado di accedere a tali dati per consentire l'ingresso in officina (o in altri locali che in futuro si dovesse decidere di controllare) solo ai dipendenti presenti in istituto e abilitati alla funzione.

2 - Materiale utilizzato

Dopo aver preso in considerazione varie ipotesi di soluzione ed averle valutate dal punto di vista tecnico ed economico, alla fine la scelta è caduta su una implementazione basata sul progetto italiano "Arduino". Tale progetto mette a disposizione un ambiente di sviluppo completamente *open-source* in linguaggio C/C++ (si tratta, per l'esattezza, di una versione semplificata di tale linguaggio) con cui programmare la scheda hardware "Arduino" nelle varie versioni e le eventuali schede aggiuntive ad essa connesse.

In particolare, si sono utilizzati i seguenti componenti hardware:

- scheda principale di controllo "Arduino UNO", che è risultata adatta allo scopo perché molto compatta, semplice da programmare e da interfacciare con le altre schede;
- scheda di rete "Ethernet shield", per la connessione TCP/IP al server aziendale dell'istituto;
- scheda USB Host, per il collegamento del lettore di schede a banda magnetica;
- alimentatore in continua 5V - 250mA;
- lettore di schede a banda magnetica "WBT-1200" con interfaccia USB;
- elettroserratura per porte antincendio.

Per l'attivazione dell'elettroserratura è stato realizzato un semplice circuito di temporizzazione e comando (CTC) dotato di relè (NB: nella foto il CTC è stato simulato con un led di colore verde).

Si noti che la scheda principale "Arduino UNO" dispone già di una propria porta USB; tuttavia questa porta è di tipo "Device" e non può quindi essere utilizzata per il collegamento del lettore di schede magnetiche (funzione che richiede una porta USB di tipo "Host"). Per questo motivo è stato prevista la scheda USB aggiuntiva, mentre la porta presente nell'unità di base viene utilizzata per il collegamento con un personal computer di supporto, utilizzato nelle fasi di sviluppo, programmazione, debug ed eventualmente gestione del dispositivo.

2.1 - Software utilizzato e suo funzionamento di massima

Versione del software utilizzato: "Arduino 0022"
Librerie standard utilizzate: Ethernet, SPI
Libreria per USB MAX3421E "USB_Host_Shield" prelevata dal sito:

http://www.circuitsathome.com/arduino_usb_host_shield_projects

Il software sviluppato per questo progetto prevede un colloquio diagnostico tramite la seriale USB nativa di Arduino, il cui livello di dettaglio può essere controllato tramite una "define". L'applicazione utilizza inoltre un approccio *client-server* per dialogare con il database centrale dell'IFAC, a sua volta controllato da un *application server* dedicato alla gestione delle presenze. Su ques'ultimo è stata prevista una nuova funzione specifica, attivata attraverso un comando "CHAC" (CheckAccess).

Portata a termine l'inizializzazione delle varie schede, il software client rimane in attesa di un evento dal lettore di schede magnetiche.

Al momento del passaggio di una scheda, viene letto il codice presente sulla banda magnetica (in traccia 2) e viene attivato il colloquio TCP/IP, inviando al server una richiesta "CHAC" corredata dei parametri necessari: l'identificativo della risorsa controllata ("OFFICINA" nel caso in esame), l'identificativo del dipendente ed una password di sicurezza.

Il server, al momento della richiesta, interroga il database del personale (MySQL) verificando la presenza in istituto del dipendente interessato ed analizzando, nel profilo di quest'ultimo, il flag di abilitazione alla risorsa a cui si chiede l'accesso; nel caso di accesso consentito, il server invia un messaggio di autorizzazione alla scheda Arduino, a seguito del quale questa provvede ad inviare un impulso di circa 1 secondo su una apposita uscita digitale. Questa uscita a sua volta viene utilizzata dalla schedina CTC, che provvede ad attivare l'elettroserratura (aprendo quindi la porta) per un tempo adeguato (regolabile fino ad un massimo di circa 10 secondi).

Attualmente, per motivi di sicurezza, il sistema prevede solo la gestione dell'ingresso in officina, mentre l'uscita è garantita tramite un maniglione antipanico. Si sono già previsti sistemi di rilevamento tramite RFID, che eviterebbero lo strisciamento della scheda e potrebbero forse controllare il transito anche in uscita dall'officina.

3 - Modifiche hardware e software

Nello sviluppo del sistema ci siamo imbattuti in un'incompatibilità della scheda Ethernet con quella USB, che usano la stessa uscita digitale (PIN 10).

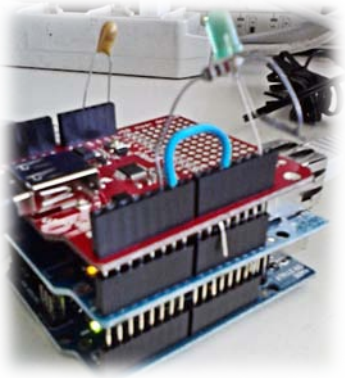
Per risolvere questo problema, si sono seguite alcune indicazioni disponibili in rete, che prevedono di modificare la libreria USB MAX3421E "USB_Host_Shield" e in particolare il file "Max3421e_constants.h":

```
/*
Modificato il PIN utilizzato da 10 a 6
come indicato da Felis nel suo blog
a causa di una incompatibilita' tra
USB HOST SHIELD e Ethernet Shield.
A.Agostini Giugno 2011

#define MAX_SS    10
*/
#define MAX_SS    6
```

Congiuntamente a questo intevento sul lato software, è stato necessario apportare una modifica anche ai collegamenti hardware tra i vari moduli del sistema Arduino, come indicato in:

http://arduino.cc/forum/index.php?topic=51874.0



Con la modifica software, indichiamo ad Arduino di pilotare la scheda USB tramite l'uscita 6 invece che l'uscita 10.

Con la modifica hardware, viene scollegato il piedino 10 della scheda USB per evitare di prelevare tale segnale da Arduino e viene intradato il segnale dell'uscita 6 di Arduino al pin 10 della scheda USB tramite un ponticello (in azzurro nella foto). Non è necessario intervenire in alcun altro modo all'interno delle schede.

Un altro problema che si è dovuto affrontare riguarda il *boot* di Arduino all'accensione che, in assenza della connessione USB/Seriale verso il PC utilizzato per la programmazione ed il monitoraggio, porta al blocco del sistema in attesa di una richiesta di *upload* dal PC stesso.

Anche in questo caso la soluzione è stata suggerita dalla consultazione di alcuni forum su Internet, tra cui in particolare:

http://arduino.cc/forum/index.php?topic=62636.0

Poiché si è visto che la pressione del tasto di RESET forza Arduino ad avviare il programma già presente a bordo, si è pensato di ottenere lo stesso risultato in modo automatico, simulando la pressione del pulsante per mezzo di un condensatore (10 uF 25V al tantalio) connesso tra i pin di RESET e GND.

In questo modo, appena si accende Arduino il condensatore – inizialmente scarico – tiene a zero il pin RESET provocando l'inizializzazione hardware del sistema; poi il condensatore si carica e permette ad Arduino di eseguire regolarmente il programma a bordo.

Questa modifica garantisce anche il corretto riavvio del sistema in caso di mancanza di alimentazione, senza necessità di intervento manuale.

3.1 - Codice sviluppato lato client (Arduino)

```
/*
Ethernet module.
*/
#include <SPI.h>
#include <Ethernet.h>

/*
MAX3421E USB Host controller communication.
*/
#include <Max3421e.h>
#include <Max3421e_constants.h>
```

```
#include <Usb.h>

/*
Data taken from configuration descriptor.
*/
#define KBD_ADDR       1
#define KBD_EP         1
#define EP_MAXPKTSIZE  8
#define EP_POLL        0x0a

/*
Endpoint record structure.
*/
EP_RECORD ep_record[2];

#define LINEFEED   10
#define RETURNCH   13
#define ENDTXTFILE 26

/*
 0 nessun debug;
 1 debug normale;
>1 per avere debug dettagliato.
*/
#define DEBUG 1

MAX3421E Max;
USB Usb;
String idutn="";

/*
Uscita digitale per il controllo dell'elettroserratura.
*/
int ctcPin=5;

/*
Enter a MAC address and IP address for your controller below.
The IP address will be dependent on your local network.
*/
byte mac[] = {  0x90, 0xA2, 0xDA, 0x00, 0x11, 0x83 };
byte ip[] = { 149,139,32,113 };

/*
Enter the IP address of the server you're connecting to.
*/
byte server[] = { 149,139,32,55};
byte subnet[] = {255, 255, 252, 0};
byte gateway[] = {149, 139, 32, 1};

/*
Initialize the Ethernet client library.
*/
Client client(server, 5122);

void setup()
{
  Serial.begin(9600);
  Serial.println("Controllo accesso OFFICINA IFAC. @IFAC-CNR 2011 -
Agostini, Andreuccetti, Cardone ");
  /*
  Setto l'uscita digitale "ctcPin" come OUTPUT.
```

```
  */
  pinMode(ctcPin, OUTPUT);
  /*
  Start the Ethernet connection.
  */
  Ethernet.begin(mac, ip, gateway,subnet);
  /*
  Give the Ethernet shield a second to initialize.
  */
  if(DEBUG) Serial.println("Inizializzo la rete TCP/IP...");
  delay(5000);
  /*
  If you get a connection, report back via serial.
  */
  if(testTCP())
  {
    if(DEBUG>1) Serial.println("connesso.");
    /*
    Start USB Host controller.
    */
    Max.powerOn();
    if(DEBUG) Serial.println("Sistema attivato e pronto per la lettura di
schede!");
    delay(1000);
  }
  else
  {
    if(DEBUG) Serial.println("connessione al server fallita!");
    if(DEBUG) Serial.println("Sistema bloccato!");
    while(1);
  }
} /* setup */

void loop()
{
  /*
  If there are incoming bytes available
  from the server, read them and print them.
  */
  String row="";
  String testo="";
  String code="";
  /**/
  if(DEBUG>1) Serial.println("Inizio loop");
  Max.Task();
  if(DEBUG>1) Serial.println("Max Task");
  Usb.Task();
  if(DEBUG>1) Serial.println("Usb Task");
  if(Usb.getUsbTaskState() == USB_STATE_CONFIGURING )
  {
    /*
    Wait for addressing state.
    */
    if(DEBUG>1) Serial.println("init");
    crdrdr_init();
    Usb.setUsbTaskState( USB_STATE_RUNNING );
  }
  if(Usb.getUsbTaskState() == USB_STATE_RUNNING )
  {
    /*
    Poll the card reader.
```

```
    */
    if(DEBUG>1) Serial.println("poll");
    code=crdrdr_poll();
    if(DEBUG>1) Serial.println("exit poll");
  }
  if(code.length()==8)
  {
    if(DEBUG) Serial.println("Connessione al server.");
    if(connettiServer())
    {
      /*
      Invia il comando al server.
      */
      if(DEBUG>1) Serial.print("C:");
      testo="CHAC OFFICINA ";
      testo+=code;
      testo+=" ########";
      if(DEBUG) Serial.println(testo);
      client.println(testo);
      /*
      Attendi risposta.
      */
      row=netgets();
      if(row.startsWith("+OK"))
      {
        if(DEBUG>1) Serial.print("S:");
        if(DEBUG>1) Serial.println(row);
        if(DEBUG) Serial.println("ACCESSO CONSENTITO!");
        digitalWrite(ctcPin, HIGH);   // attiva l'uscita
        delay(1000);                  // aspetta 1 secondo
        digitalWrite(ctcPin, LOW);    // disattiva l'uscita
      }
      else
      {
        if(DEBUG>1) Serial.print("S:");
        if(DEBUG>1) Serial.println(row);
        if(DEBUG) Serial.println("ACCESSO NEGATO!");
      }
      /*
      Chiude la connessione.
      */
      scollegaServer();
      if(DEBUG>1) Serial.println("Esce da loop");
    }
    else
    {
      if(DEBUG) Serial.println("Errore nella connessione al server!");
    }
  }
  else if(code.startsWith("-ERR"))
  {
    if(DEBUG) Serial.println("Termina programma oppure ERRORE");
    /*
    Chiude una eventuale connessione.
    */
    Serial.flush();
    if(DEBUG>1) Serial.println("END");
    while(1);  //stop
  }
} /* loop */
```

```
int connettiServer()
{
  String row="";
  if(!client.connect())
  {
    return(false);
  }
  else
  {
    while(!row.startsWith("+OK"))
    row=netgets();
    return(true);
  }
} /* connettiServer */

void scollegaServer()
{
  String row="";
  if(DEBUG>1) Serial.print("C:");
  if(DEBUG>1) Serial.println("FINE");
  /*
  Chiude il collegamento col server.
  */
  client.println("FINE");
  while(!row.startsWith("+OK"))
  row=netgets();
  if(DEBUG>1) Serial.print("S:");
  if(DEBUG>1) Serial.println(row);
  /*
  Stop the client.
  */
  client.stop();
} /* scollegaServer */

int testTCP()
{
  String row="";
  if(!connettiServer())
  {
    if(DEBUG) Serial.println("..testconnessione...fallita!");
    return(false);
  }
  else
  {
    if(DEBUG) Serial.println("..testconnessione...ok!");
    scollegaServer();
    return(true);
  }
} /* testTCP */

String netgets()
{
  String s="";
  char ch=0;
  int i=0;
  /**/
  while(1)
  {
    ch=client.read();
    if(ch==RETURNCH)
    {
```

```
      ch=client.read();
      return(s);
    }
    if(ch!=-1) s+=ch;
  }
  return(s);
} /* netgets */

void crdrdr_init(void)
/*
Initialize card reader.
*/
{
  byte rcode = 0;  //return code
  /*
  Initialize data structures;
  Copy endpoint 0 parameters.
  */
  ep_record[ 0 ] = *( Usb.getDevTableEntry( 0,0 ));
  ep_record[ 1 ].MaxPktSize = EP_MAXPKTSIZE;
  ep_record[ 1 ].Interval  = EP_POLL;
  ep_record[ 1 ].sndToggle = bmSNDTOG0;
  ep_record[ 1 ].rcvToggle = bmRCVTOG0;
  /*
  Plug endpoint parameters to devtable.
  */
  Usb.setDevTableEntry( 1, ep_record );
  /*
  Configure device.
  */
  rcode = Usb.setConf( KBD_ADDR, 0, 1 );
  if(rcode)
  {
    if(DEBUG) Serial.print("Error attempting to configure card reader.
Return code:");
    if(DEBUG) Serial.println( rcode, HEX );
    while(1);  //stop
  }
  /*
  Set boot protocol.
  */
  rcode = Usb.setProto( KBD_ADDR, 0, 0, 0 );
  if(rcode)
  {
    if(DEBUG) Serial.print("Error attempting to configure boot protocol.
Return code:");
    if(DEBUG) Serial.println( rcode, HEX );
    while(1);  //stop
  }
} /* crdrdr_init */

int readCardReader(void)
{
  int val=0;
  byte rcode=0;             //return code
  char buf[8]={0};
  while(buf[2]==0) rcode=Usb.inTransfer(KBD_ADDR,KBD_EP,8,buf);
  if(rcode!=0)
  {
    return -1;
  }
```

```
  val=(int)buf[2]-29;
  if(val==10) val=0;
  if(DEBUG>1) Serial.println(val);
  return(val);
} /* readCardReader */

String crdrdr_poll(void)
/*
Poll card reader and print result.
*/
{
  int digit=readCardReader();
  if(DEBUG>1) Serial.println(".poll inizia switch");
  switch(digit)
  {
    case 22:
      if(DEBUG>1) Serial.println("Inizio lettura codice traccia 2");
      idutn=readCardReader();
      idutn+=readCardReader();
      idutn+=readCardReader();
      /*
      Controlla che siamo in presenza di un prefisso valido di tre cifre.
      */
      if(idutn!="204")
      {
        if(DEBUG>1) Serial.println(".poll errore 1"); return("-ERR1");
      }
      idutn+=readCardReader();
      idutn+=readCardReader();
      idutn+=readCardReader();
      idutn+=readCardReader();
      idutn+=readCardReader();
      if(readCardReader()==27)
      {
        return(idutn);
      }
      else
      {
        if(DEBUG>1) Serial.println(".poll errore 2 ");
        return("-ERR2");
      }
      break;
    default:
      if(DEBUG>1) Serial.println(".poll default");
      return("0");
  }
} /* crdrdr_poll */
```