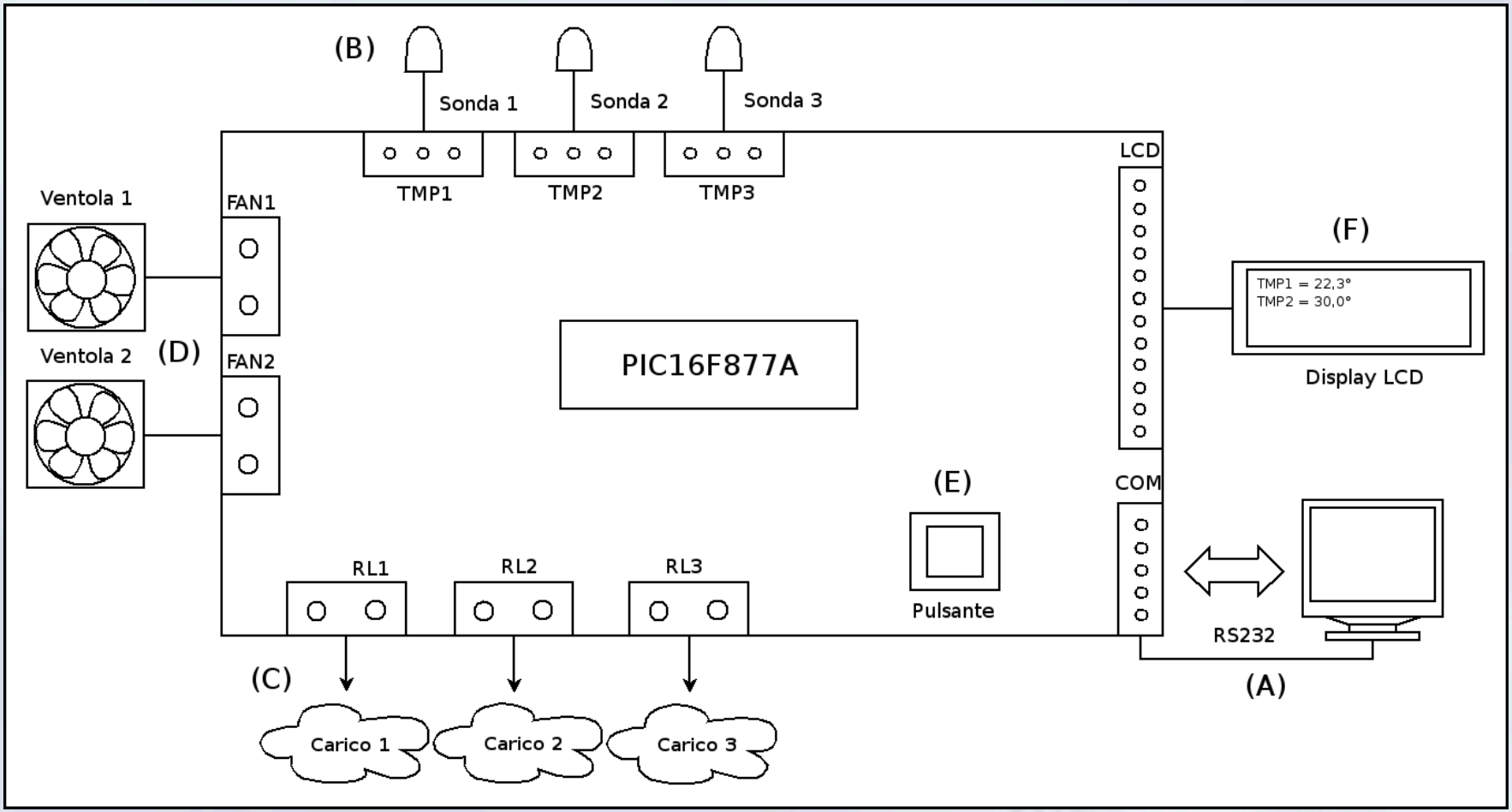

UNIVERSITA' DEGLI STUDI DI ROMA
TorVergata
Laureando: Andrea Sarro
Relatore: Daniel Pierre Bovet
A.A.: 2004/2005
Facoltà: Ingegneria
Dipartimento: Informatica
ORIGIN
'warn'
monitor
thing 2
duties 3
picture
Hardware Monitor & Protection System
Progettazione e realizzazione di un sistema di protezione e controllo dell'hardware

Scopo del progetto

- Realizzazione di un dispositivo stand-alone in grado di rilevare il funzionamento anomalo di un apparato elettronico (ad esempio un PC) e di evitarne il danneggiamento

- Realizzazione del software di gestione, in grado di:
  - Comunicare con il dispositivo
  - Configurare i parametri operativi del dispositivo
  - Ricevere informazioni sullo stato dell'apparato elettronico controllato

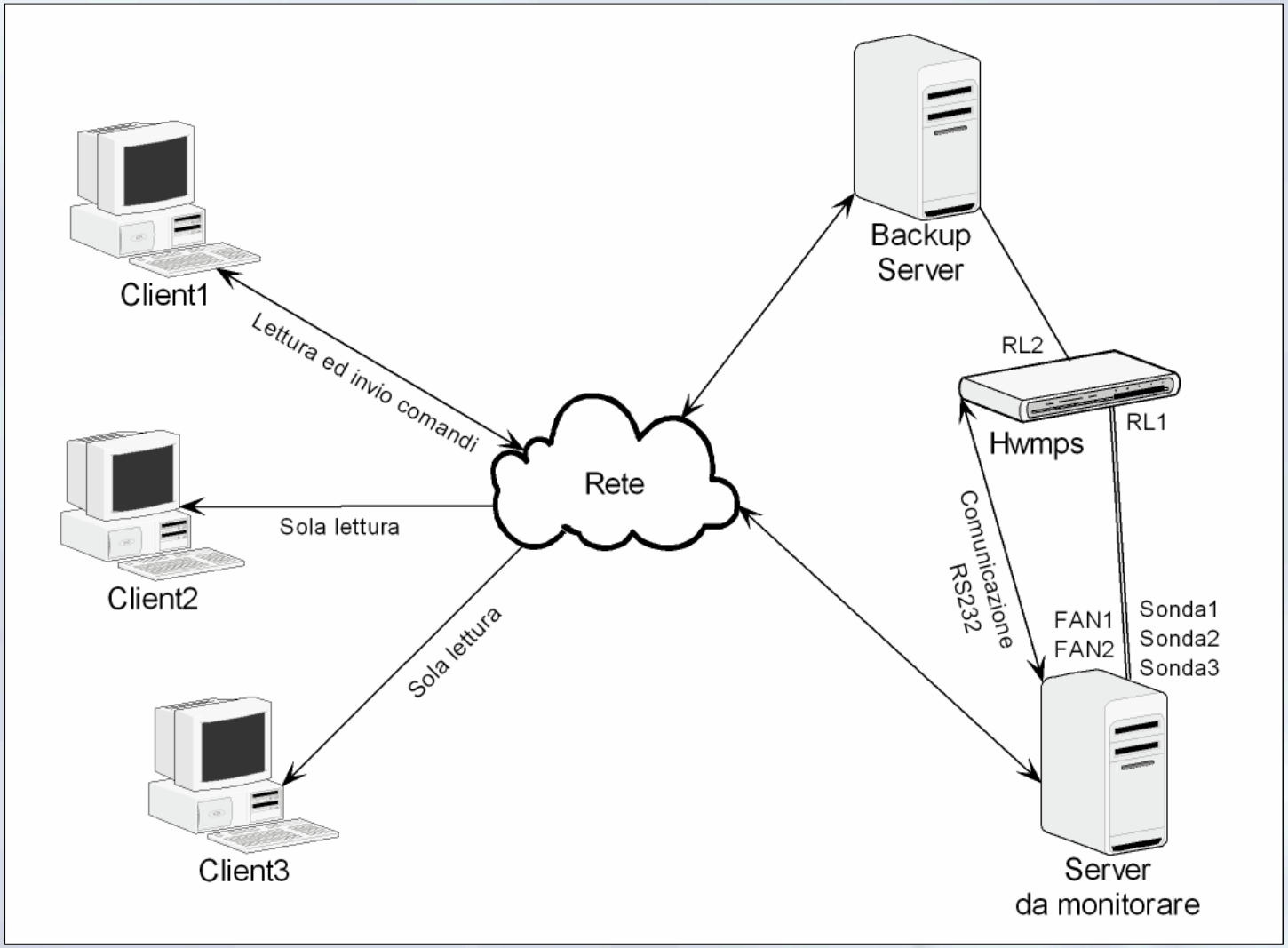
Il dispositivo di controllo (HWMPS)



Il dispositivo - Funzioni

- Funzioni di monitoring
  - Lettura di temperature da tre sonde
  - Visualizzazione dello stato del sistema su display LCD
  - Comunicazione seriale con un PC
- Funzioni di protezione e controllo
  - Pilotaggio e protezione di carichi tramite tre relè
  - Messaggi di avviso via LCD e software di gestione
  - Regolazione del regime di rotazione di due ventole

Scenario applicativo



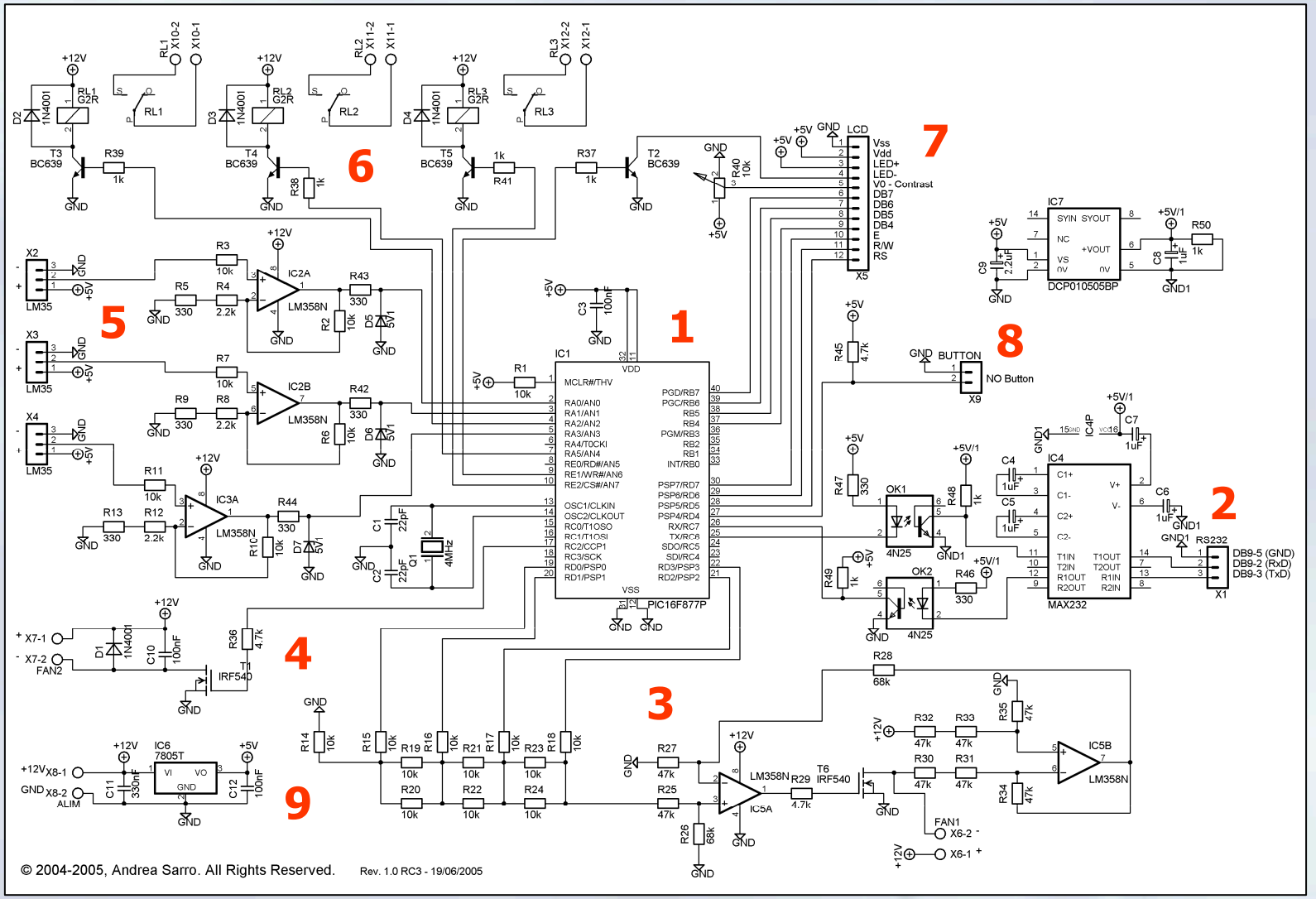
Le fasi del progetto

1. Progettazione e disegno del circuito elettronico tramite software CAD:
   - Schema elettrico ⇨ EAGLE Schematic Editor
   - Circuito stampato ⇨ EAGLE Layout Editor
2. Realizzazione del firmware ⇨ Compilatore HI-TECH PICC
3. Simulazione del firmware (codice macchina) ⇨ MPLAB SIM
4. Programmazione del firmware sul microcontrollore
5. Sviluppo di una libreria cross-platform per la comunicazione seriale
6. Sviluppo del software client-server di gestione e configurazione del dispositivo

Schema elettrico

- Approccio modulare (divisione in 9 sezioni autonome):
  1. Microcontrollore
  2. Comunicazione seriale
  3. DAC
  4. Modulazione PWM
  5. Sonde di temperatura
  6. Relè
  7. Display LCD
  8. Pulsante
  9. Alimentazione

Schema elettrico



Il microcontrollore

- Microcontrollore =
  Microprocessore + Memorie + Periferiche
- Microchip PIC16F877A Mid-Range MCU:
  - Microcontrollore ad 8 bit in package PDIP / 40 pin
    - 8192 words di Memoria Programma
    - 368 byte di Memoria Dati (RAM)
    - 256 byte di EEPROM
  - Periferiche utilizzate nel progetto:
    - USART (Universal Synchronous Asynchronous Receiver Transmitter)
    - ADC (Analog/Digital Converter)
    - Modulo CCP (Capture/Compare/PWM)

Il firmware del PICmicro

- Composto da circa 2000 righe di codice (linguaggio C), strutturato in 4 moduli:
  - Delay Library ⇨ Introduzione di pause nell'esecuzione del firmware
  - Lcd Library ⇨ Visualizzazione su display LCD
  - Serial Library ⇨ Gestione della comunicazione seriale
  - Firmware ⇨ Gestione del comportamento del dispositivo HWMPS

HWMPS Serial Lib

UNIVERSITA' DEGLI STUDI DI ROMA TorVergata

- Libreria cross-platform (Linux/Windows) per la comunicazione seriale
- Opera in modalità asincrona, non bloccante
- Sviluppata in C++, sfruttando l'ereditarietà e l'astrazione. Definisce i metodi per:
  - Aprire la porta seriale al baud-rate specificato
  - Chiudere la porta seriale
  - Inviare dati
  - Ricevere dati in modalità non bloccante
  - Verificare lo stato della porta

HWMPS Serial Lib (cont.)

UNIVERSITA' DEGLI STUDI DI ROMA TorVergata

- Nella classe *SerialPort* è dichiarato il metodo virtuale puro *SerialDataEvent*, che pertanto la rende astratta.

```
class SerialPort
{
public:
bool ComConnect(char* com_port, int speed_bps);
bool ComSendCommand(char* command, bool cr);
bool ComDisconnect();
[…]
protected:
[…]
virtual void SerialDataEvent(char* buffer, unsigned int dim) = 0;
private:
[…]
}
```

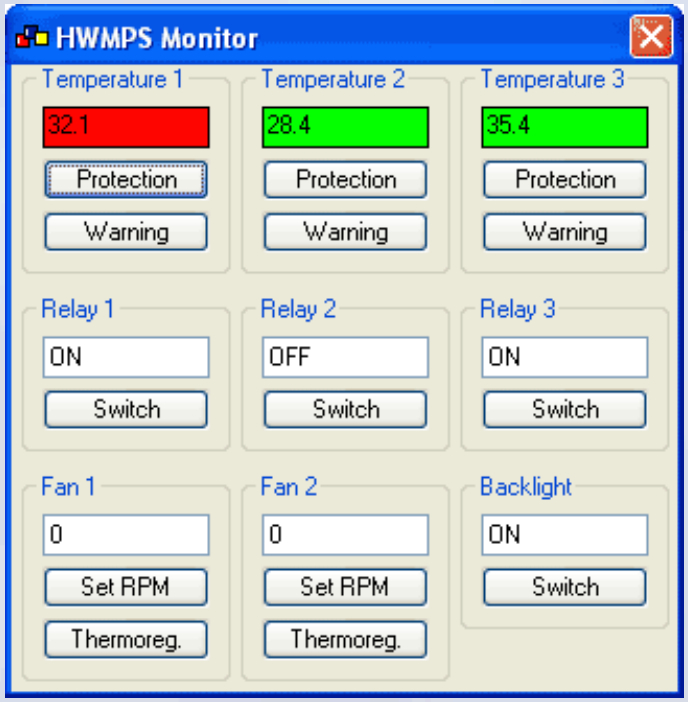
HWMPS Serial Lib (cont.)

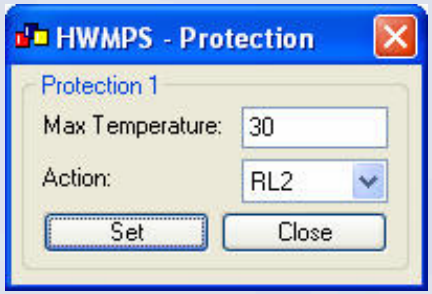
UNIVERSITA' DEGLI STUDI DI ROMA TorVergata

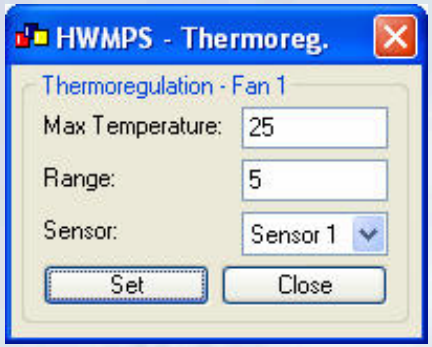
- Nella classe *SerialPort* è dichiarato il metodo virtuale puro *SerialDataEvent*, che pertanto la rende astratta.
- *SerialDataEvent* è il metodo che si occupa della ricezione dei dati dalla porta seriale e viene chiamato dal sistema operativo al verificarsi di un SIGIO.
- Per utilizzare la libreria il software finale eredita la classe astratta e definisce il metodo *SerialDataEvent*, ottenendo un comportamento specializzato per i dati in ingesso dall'interfaccia seriale.

HWMPS Monitor Client

- Permette la configurazione del dispositivo e la visualizzazione dello stato corrente del sistema







Conclusioni

Abbiamo raggiunto lo scopo di:

- Realizzare un dispositivo hardware autonomo basato su microcontrollore
- Realizzare una libreria cross-platform per la comunicazione seriale asincrona
- Realizzare un software client/server cross-platform con interfaccia grafica e supporto per la comunicazione seriale

Tutto il materiale prodotto è stato rilasciato sotto licenza opensource utilizzando la piattaforma per gli sviluppatori offerta da SourceForge.net:

http://hwmps.sourceforge.net