Da Uno a Yún

Francesco Potortì

<Potorti@isti.cnr.it>

Contatore elettronico

- LED superiore: Wh
- LED inferiore: Varh
- Lampeggio 20 ms
  - uno al secondo a 3600W
- Intervallo massimo 20', poi LED acceso fisso
- Intervallo minimo dipendente dalla potenza massima

Potenza assorbita

- Uso civile: contratti da 3 – 4,5 - 6 kW
  - lampeggio ogni 1,2 – 0,9 - 0,6 s
- Per il 3 kW, massimo prelevabile indefinitamente: 3,3 kW
- Massimo per tre ore: 4 kW
- A 10 kW di consumo lampeggio ogni 360 ms
- Massimo per due minuti 64 kW, che è il limite tipico dell'interruttore magnetotermico incorporato nel contatore

Vincoli di progetto

- Software e hardware liberi
- Lettura di potenza attiva e reattiva
- Autocontenuto, quindi web server integrato
- Banale da installare
- Storico completo
- Tempo reale

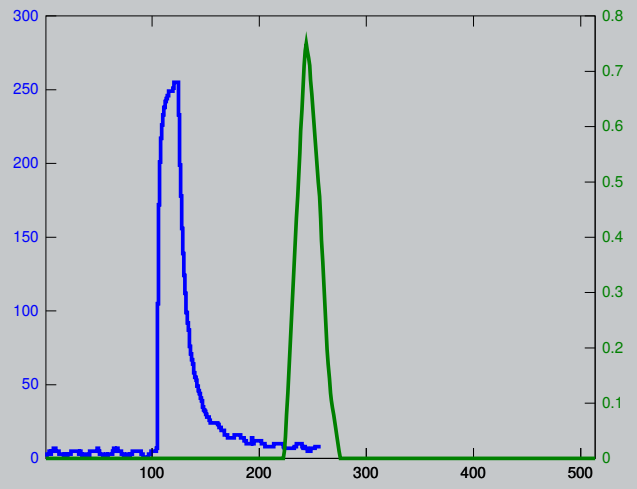
Licenze libere

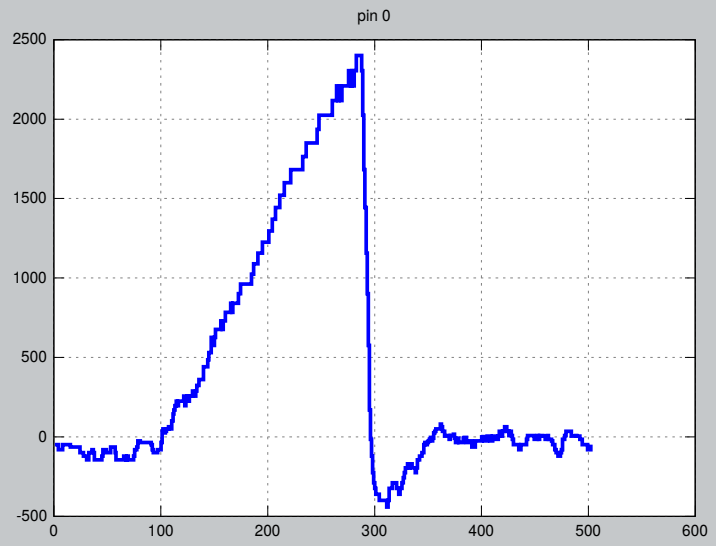
- Licenza di copyright per il software
  - licenza *libera* o open source o FOSS o FLOSS o F/OSS
  - distribuzione con sorgente e istruzioni
  - permette modifica e redistribuzione in forma originale o modificata
- Licenza di copyright per l'hardware
  - distribuzione con schema e firmware
  - permessi come sopra

Architettura

- Software semplice
  - compatibile con librerie Arduino
  - no interrupt: scheduler cooperativo
    - gestione campionamento
    - invio dati (WiFi shield)
- Hardware semplice
  - una o due fotoresistenze
- Espandibile
  - altri sensori possibili

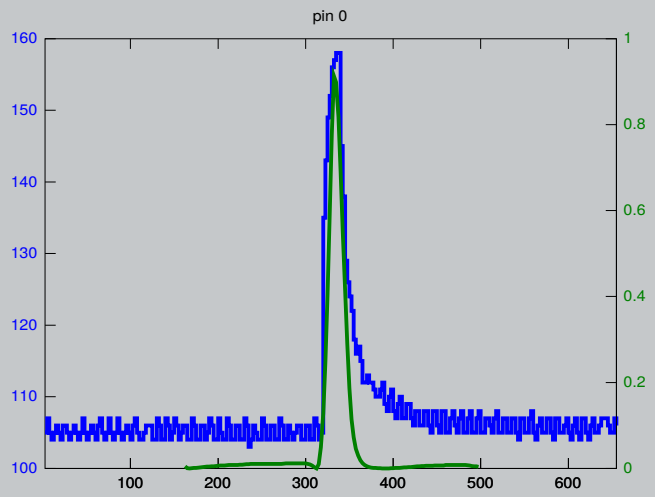
Impulso e correlazione

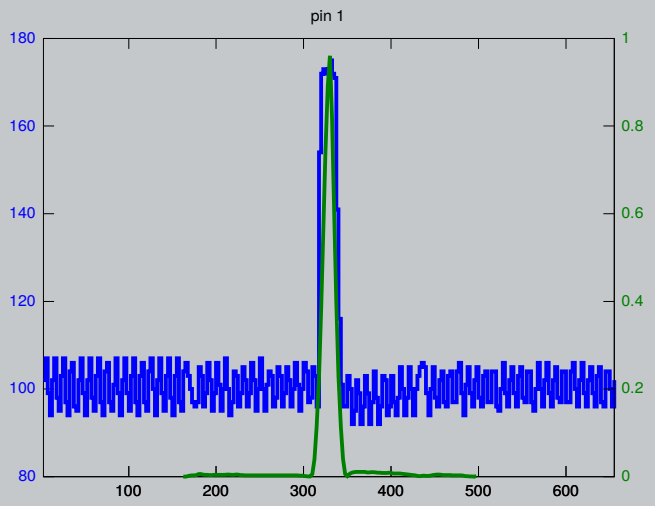




- L'impulso luminoso è lungo 20 ms
  - ma ha una coda
  - la risposta del correlatore è in verde
- La risposta del correlatore a uno scalino quando la correlazione non è normalizzata

Più correlatori





- Risposta del correlatore con due diverse fotoresistenze
- La risposta è normalizzata in 0..1
- La risposta normalizzata allo scalino è quasi nulla
- Sopporta forte rumore

Multitasking

- Cooperativo, switch solo con yield()
- yield è chiamato da dentro delay()
- Il task principale è il campionatore
  - un loop con un ciclo di 2500 µs
  - dentro ci sono tre campionamenti
- L'altro task è un server HTTP
  - risponde a un GET /
  - risponde a un GET /ajaxReq

Il task campionatore

- Campionamento ogni 2,5 ms
  - otto campioni per impulso
- Correlatore lungo 90 campioni
  - finestra 235 ms, max 3600/0,235 = 15 kW
- Un campionatore per ogni LED
- Campionamento più correlazione ~420 µs
- Rimangono 2500-2*420 = 1660 µs
- yield() solo se rimangono 720 µs

Il task HTTP server

- GET /
  - restituisce una breve pagina statica
  - la pagina contiene un programma Javascript
- GET /ajaxReq
  - restituisce le ultime letture in XML

```
<?xml version='1.0'>
  <_>
    <P1>251</P1>
    <T1>0x0144AB9C56B8</T1>
    <P0>592</P0>
    <T0>0x0144AB9C774F</T0>
    <T>0x0144AB9C94E6</T>
    <UT>7782</UT>
    <WS>8</WS>
    <DC>36</DC>
    <DY>8</DY>
    <RL>0</RL>
    <CM>249</CM>
  </_>
```

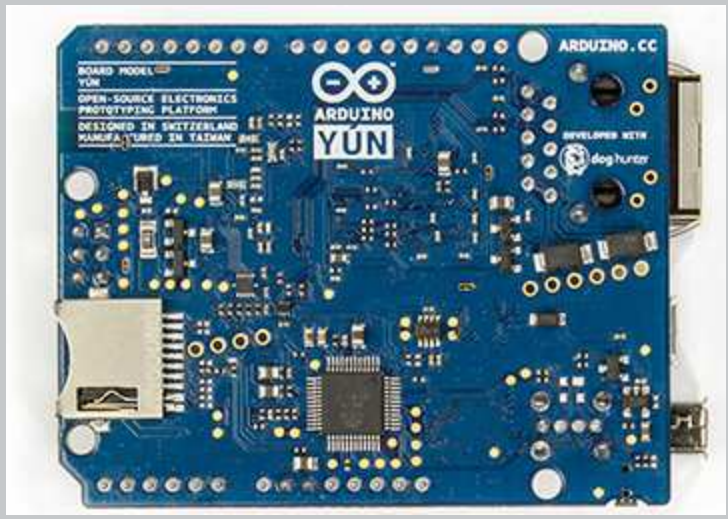
Problemi

- Siamo al limite del codice (32 KB)
  - la libreria SD standard non entra
- Siamo al limite della memoria (2 KB)
  - la libreria SD richiede buffer di 512 B
- La libreria WiFi richiede patch
  - inseriti yield() nelle routine più lente
- Lo shield WiFi è instabile
  - watchdog resetta tutto dopo 3 minuti
  - modifica hardware per reset da software

Toppa temporanea

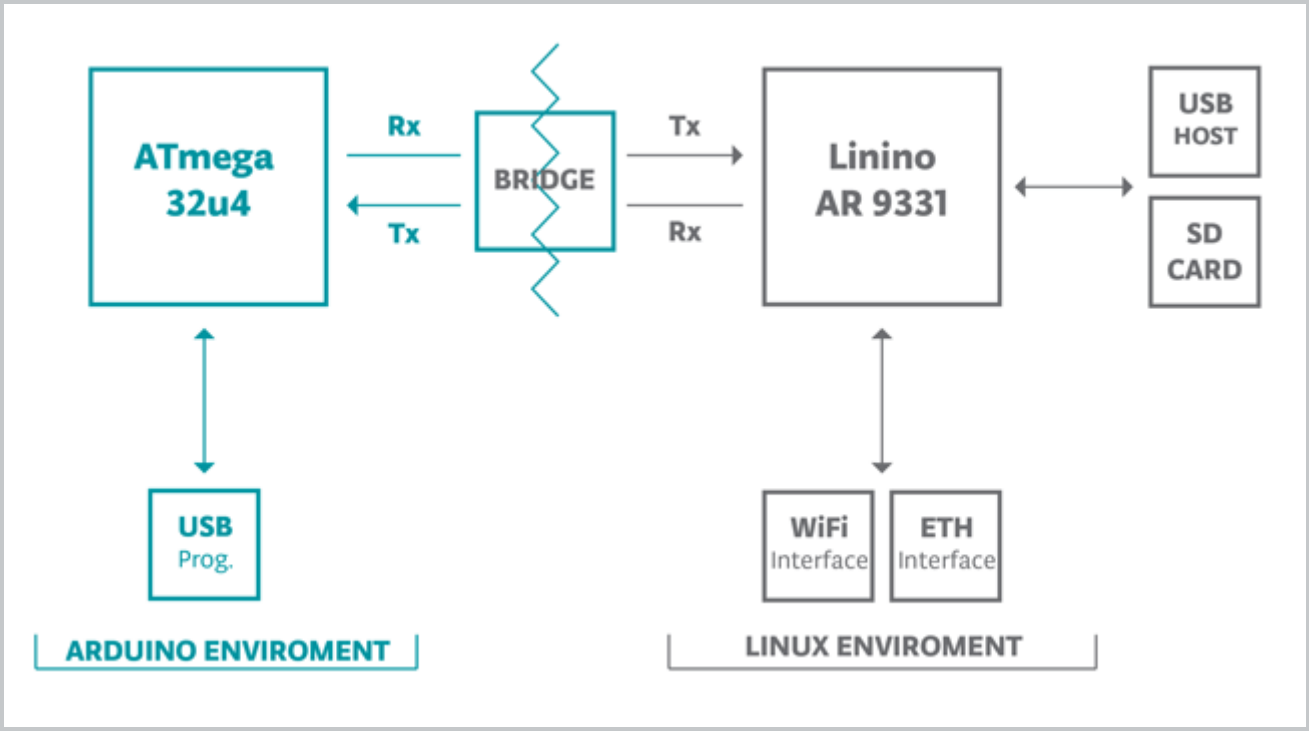
- Tutto il carico di lavoro all'esterno
  - il server HTTP interno risponde solo alle chiamate Ajax del javascript
  - la pagina web è su un server web esterno
  - il server esterno memorizza I dati su disco
- Il problema è che serve un server esterno sempre acceso...
- E l'installazione sull'Uno è roba da hacker
  - patch alle librerie, modifica hardware

Soluzione: passare a Yún



- Microcontrollore AVM
  - compatibile Leonardo
- Atheros 16 bit, 400 MHz
  - 64 MB RAM, 16 MB flash
  - Linuino, basato su OpenWrt
- Periferiche integrate
  - Ethernet, WiFi
  - SD, USB client, USB host

Architettura Yún



- Il microcontrollore è isolato dalle periferiche, niente librerie di gestione
  - accede alle risorse tramite libreria Bridge

La libreria Bridge: i processi

- Bridge.begin()
  - lancia run-bridge su Linino, il programma che riceve ed esegue i comandi di Bridge
- classe Process
  - lancia un processo sincrono o asincrono su Linuino, legge da stdout e scrive su stdin e stdout, controlla se gira, lo uccide
- classe HttpClient
  - estende Process, è un'interfaccia a `curl`

La libreria Bridge: I/O

- classe Console
  - funziona come la Serial su Uno: legge e scrive da una porta seriale virtuale su USB
- classe FileIO
  - tutte le funzioni per leggere e scrivere su SD
- classe Mailbox
  - gestisce un dizionario di coppie `nome:valore` in memoria volatile, direttamente accessibile dal server web dello Yún

La libreria Bridge: Internet server

- classi YunServer e YunClient
  - YunServer definisce un server Internet
  - YunServer.accept() restituisce un YunCLient per ogni connessione
  - YunCLient.read() e .write() permettono di usare la connessione
- L'interfaccia è analoga a quella degli shield Ethernet e WiFi, che era molto limitata: un solo client per server
  - la limitazione non ha più senso

Porting minimale

- Si usano solo le funzioni del Bridge
  - non si tocca nulla su Linino
  - le chiamate alla libreria WiFi diventano chiamate al Bridge
  - si aggiunge la memorizzazione su SD
  - si scrivono pagine web statiche su SD
- Vantaggi
  - veloce, stabile (si spera), semplice
  - nuovo sketch caricabile via rete
  - pagine web caricabili via rete (credo)

Porting ottimale

- Si elimina il server HTTP su AVM
- Bridge solo per mandare le ultime letture
  - usando per esempio la classe MailBox
- Tutto il resto fatto dentro Linino
  - memorizzazione su SD
  - server web
- Vantaggi
  - si sfrutta tutta la potenza di Linino
  - non si deve implementare un server web

E ora giochiamo

- Reset completo di Yún e configurazione
- Login root:password@ssh://...
- Esplorazione scheda SD
  - mkdir /mnt/sd/arduino/www, http://.../sd
- Il dizionario (classe Mailbox del Bridge)
  - /keystore_manager_example
  - /data/get → JSON, /data/put/mykey/myvalue
- Un'interfaccia REST realizzata col Bridge
  - Un web server che legge dai pin e il potbridge

Attività in corso al CNR

Proposte di tesi di laurea

Francesco Potortì (CNR-ISTI)

Ambiente di lavoro

- Un tirocinio sul porting da Uno a Yún
  - illustrato nei lucidi precedenti
- Due tesi specialistiche su localizzazione
  - illustrate di seguito
- Si lavora al CNR o a casa
  - anche tesi in coppia
  - al CNR opportunità di contatti esperti, ambiente giovane

Tecnologie usate

- Software: Internet of Things
  - reti di sensori
  - middleware basato su OSGi
  - web services, REST
- Hardware: dispositivi con radio
  - punti di accesso Wi-Fi
  - smartphone con Arduino
  - sensori ZigBee e Z-Wave

Localizzazione device-free interni il metodo

- Una rete “opportunistica” di radio
  - ZigBee, 6LoWPAN, Z-Wave
- Ognuno ascolta le trasmissioni di ogni altro e ne registra la potenza (RSS)
- Con vari metodi matematicamente sofisticati si identifica la posizione
  - classificazione preceduta da allenamento
  - RTI: radio tomographic imaging
- Funziona attraverso le pareti

Localizzazione device-free interni le applicazioni al CNR

- Un'architettura per utilizzare, comparare e fondere diversi metodi di localizzazione
  - definizione dell'architettura in OSGi
  - implementazione di tre metodi completa
  - definizione del metodo di fusione dati
  - progetto di sperimentazione, misure e miglioramento algoritmi

Localizzazione in interni con Wi-Fi il metodo

- Si usa la potenza ricevuta (RSS) da diversi punti di accesso
- Con vari metodi si deduce la posizione
  - fingerprinting
- Errore di alcuni metri
- La mappa di verosimiglianza ottenuta si fonde con altre informazioni
  - statiche: mappe
  - dinamiche: contesto, sensori indossati

Localizzazione in interni con Wi-Fi le applicazioni al CNR

- Sistema di navigazione all'interno dell'Area CNR
  - in corso di sviluppo
  - possibile estensione per l'ospedale
- Posizionamento in centri commerciali per l'identificazione delle aree più frequentate
  - trattative in corso per commessa
- Definizione interfaccia e implementazione
  - su middleware OSGi in progetto europeo IoT