Laboratorio di Segnali e Sistemi
introduzione ad Arduino

Esempi utilizzo dei microcontrollori
in Fisica delle alte energie (LHCb)
e del rivelatore ArduSiPM in:
Fisica Medica (Chirone)
Monitor di Fascio CERN (UA9)
CERN Beam Line for School (2017 winner)
Rivelazione di radiazioni e Raggi Cosmici
Progetto EOS (Rivelazione raggi cosmici in stratosfera)
Progetto Maya

Valerio Bocci A.A. 2018-2019

Introduzione: microprocessore e microcontrollore

Microprocessore

Un microprocessore integra sul chip la logica di elaborazione ma richiede sempre delle unità esterne (memorie, gestori di segnali e dispositivi periferici) per poter scambiare informazioni e interagire con l'esterno.

Microcontrollore

Un microcontrollore è invece un sistema completo, che integra in uno stesso chip il processore, la memoria permanente, la memoria volatile e i canali (pin) di I/O, oltre ad eventuali altri blocchi specializzati.
È progettato per interagire direttamente con il mondo esterno tramite un programma residente nella propria memoria interna e mediante l'uso di pin specializzati o configurabili dal programmatore. Sono disponibili in 3 fasce di capacità elaborativa (ampiezza del bus dati): 8 bit, 16 bit e 32 bit.

Introduzione (2)

Il primo microchip ottimizzato per applicazioni di controllo è stato il modello TMS 1000 Texas instruments commercializzato nel 1975, con RAM e ROM sullo stesso chip. Questo componente è stato utilizzato nei giochi Speak and Spell e Simon.

Oggi si producono annualmente decine di miliardi di pezzi. Principali produttori:

- Freescale Semiconductor (USA)
- Atmel (now Microchip) (USA)
- Microchip Technology (USA)
- Infineon Technologies (Germania)
- Texas Instruments Incorporated (USA)
- Fujitsu (Giappone)
- NXP Semiconductors (Paesi Bassi)
- STMicroelectronics (Francia, Italia)
- Samsung Electronics (Corea del Sud)

Introduzione (3):

Componenti di un microcontrollore:

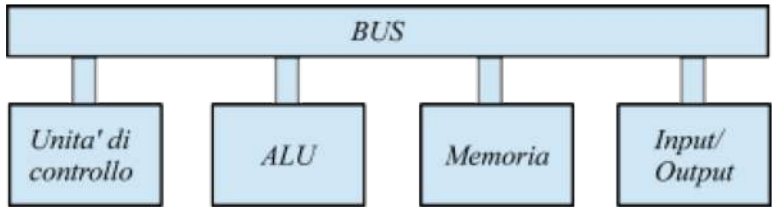
- CPU
- Memoria di programma: ROM, EPROM, FLASH
- Memoria dati: RAM, EPROM
- Oscillatore
- Porte di I/O
- Gestione di Interrupt
- Timer, contatori
- Moduli di comunicazione (SPI, I2C, USB....)
- ADC, DAC, PWM

Sistema di sviluppo:

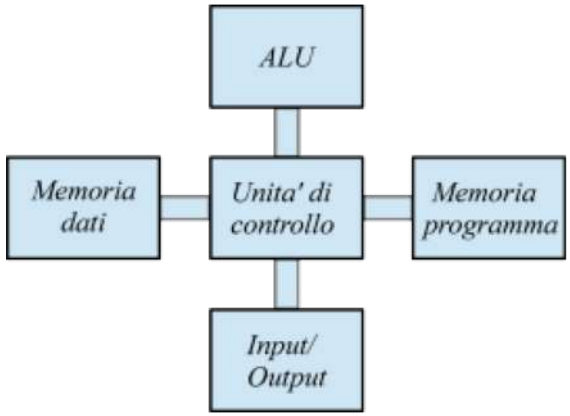
L'insieme di strumenti (hardware e software) necessari per generare il codice che deve essere eseguito dal processore, metterlo a punto e collaudarlo.

Introduzione (4): Architettura

- Von Neumann: Programma e dati sono ospitati sulla stessa memoria;
- Harvard: La memoria di programma e' separata da quella dei dati. Si utilizzano bus diversi.



(a)



(b)

Microcontrollori ATMEL AVR

AVR e' una famiglia di microcontrollori RISC ad architettura Harvard sviluppati dalla Atmel a partire dal 1996. L'AVR utilizza una memoria flash interna per il programma: questo permette di cancellare e riscrivere una nuova versione in pochi secondi e anche senza rimuovere il microcontrollore dalla scheda su cui è montato, velocizzando enormemente il processo di correzione e messa a punto del codice.

Caratteristiche:

- Memoria di programma flash (riprogrammabile almeno 10000 volte)
- Memoria EEPROM (riscrivibile almeno 100000 volte)
- Memoria RAM statica
- Clock interno calibrato
- Porte di I/O
- Gestione di Interrupt
- Timer, contatori
- ADC, DAC, PWM

Ci sono 4 gruppi di microcontrollori, con caratteristiche differenti (ATtiny, ATMega, ATXMega, AT90).

ATMega328

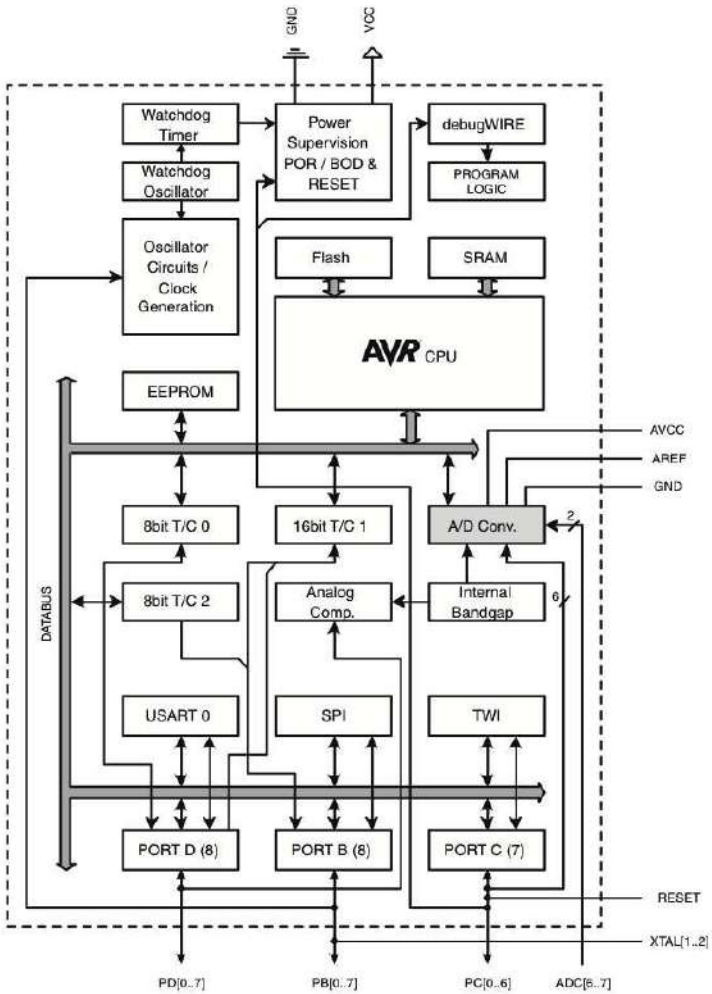
Caratteristiche principali:

- Tecnologia CMOS
- Flash memory: 32KB
- EEPROM: 1 KB
- SRAM: 2KB
- Clock 16 MHz
- I/O digitali 14 (di cui 6 PWM)
- Ingressi analogici: 6
- Alimentazione: 5V

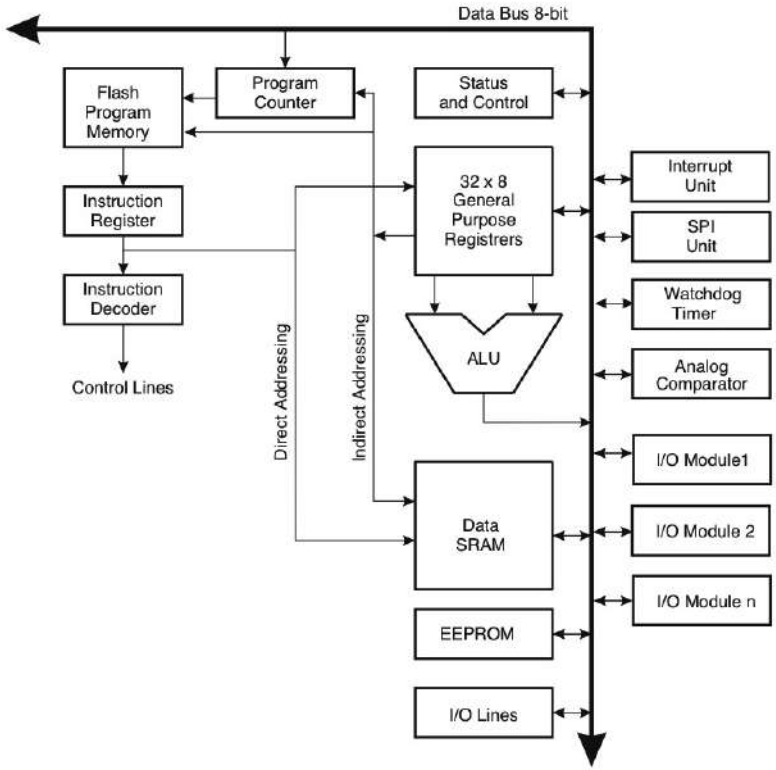
ATMega328: schema funzionale(1)

Il μC contiene inoltre:

- Circuiti per la comunicazione seriale;
- Timers, oscillatori;
- V_{ref} interna (1.1 V);
- Gestione dell'alimentazione;
- Gestione di interrupts;
- ....



ATMega328: schema funzionale(2)



ATMega328: Timing

Figure 6-4. The Parallel Instruction Fetches and Instruction Executions

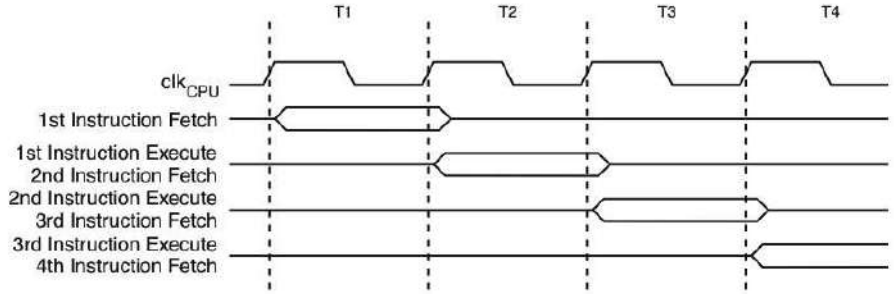
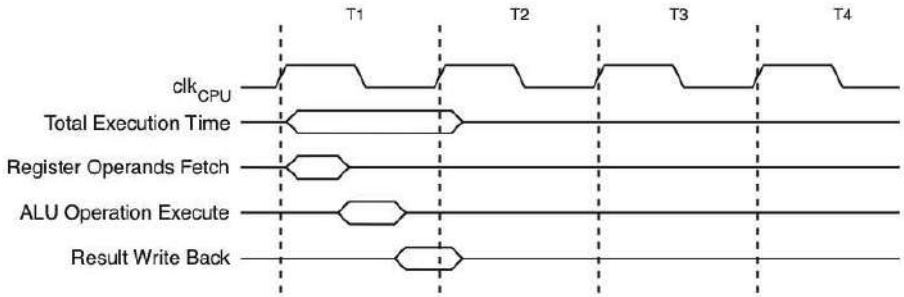


Figure 6-5 shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 6-5. Single Cycle ALU Operation



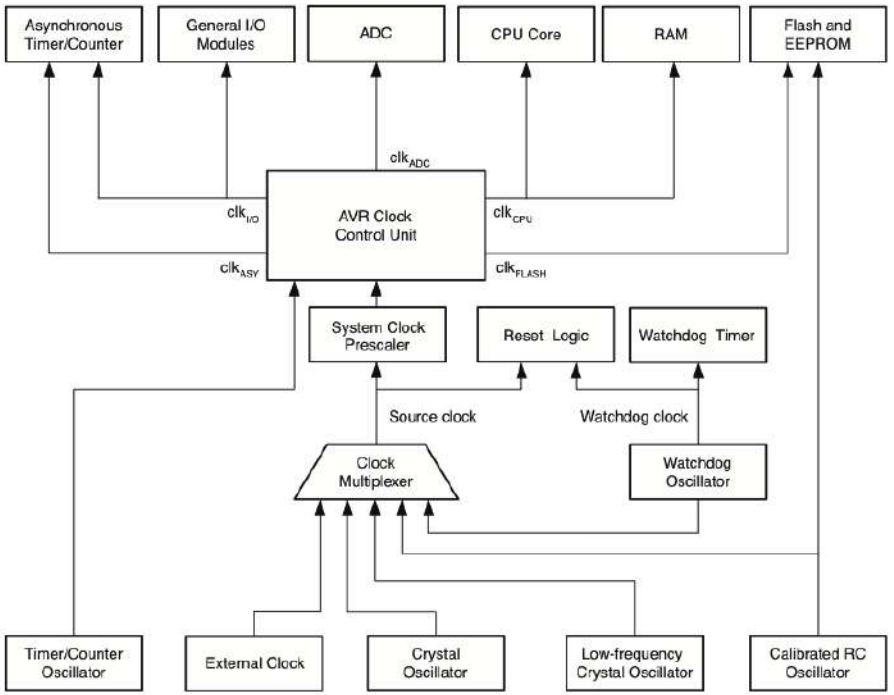
ATMega328: Interrupts

Ci sono varie possibili sorgenti di interrupts, interne o esterne e per ciascuna di esse un interrupt vector. Gli interrupt vectors sono contenuti nella memoria di programma e ordinati in base alla priorita'. Il Reset e' l'interrupt con priorita' piu' alta. Per ogni interrupt e' previsto 1 bit individuale di abilitazione, ma nello Status Register vi e' un bit (Global interrupt enable) di abilitazione globale. Quando arriva un interrupt il Global interrupt enable viene disabilitato e si avvia l'esecuzione della routine di servizio corrispondente. All'uscita di questa il Global bit viene nuovamente abilitato. Ci sono sostanzialmente due tipi di interrupt: nel primo tipo, l'evento viene memorizzato in un bit (interrupt flag) e viene servito appena possibile. Ovvero, se una o piu' condizioni di interrupts avvengono quando il Global enable e' disabilitato, non vengono perse e vengono servite non appena il Global enable e' abilitato di nuovo. Nel secondo tipo invece l'interrupt verra' servito solo se la condizione che lo ha causato persiste quando il Global enable e' abilitato di nuovo. Quando l'AVR esce da un interrupt ritorna sempre al programma principale ed esegue una istruzione prima di occuparsi di eventuali altri interrupt in attesa.

ATMega328: Clock

Sistema di gestione e distribuzione del clock:

Figure 8-1. Clock Distribution



Sono possibili diverse sorgenti di clock:

- interno (8 *MHz*, ma modificabile);
- esterno (0 ÷ 20 *MHz*), di vario tipo.

ATMega328: Alimentazione

- vari *sleep mode* selezionabili da programma per ridurre i consumi;
- *Brown out detector*: invia un segnale di reset quando la tensione di alimentazione scende sotto un livello prefissato;
- *Power on reset*: invia il segnale di reset finche' la tensione di alimentazione e' al di sotto di una certa soglia.

ATMega328: Watchdog

Il *watchdog* e' un mecccanismo di sicurezza che puo' essere utilizzato per evitare che il microcontrollore resti bloccato a causa di errori inaspettati nel programma o comunque altre cause. il sistema e' composto da un oscillatore a 128 *kHz* e un contatore con varie uscite. Sulla base di questo si puo' costruire una logica di *time-out* che puo' essere utilizzata per inviare un segnale di reset al microcontrollore.

ATMega328: Reset

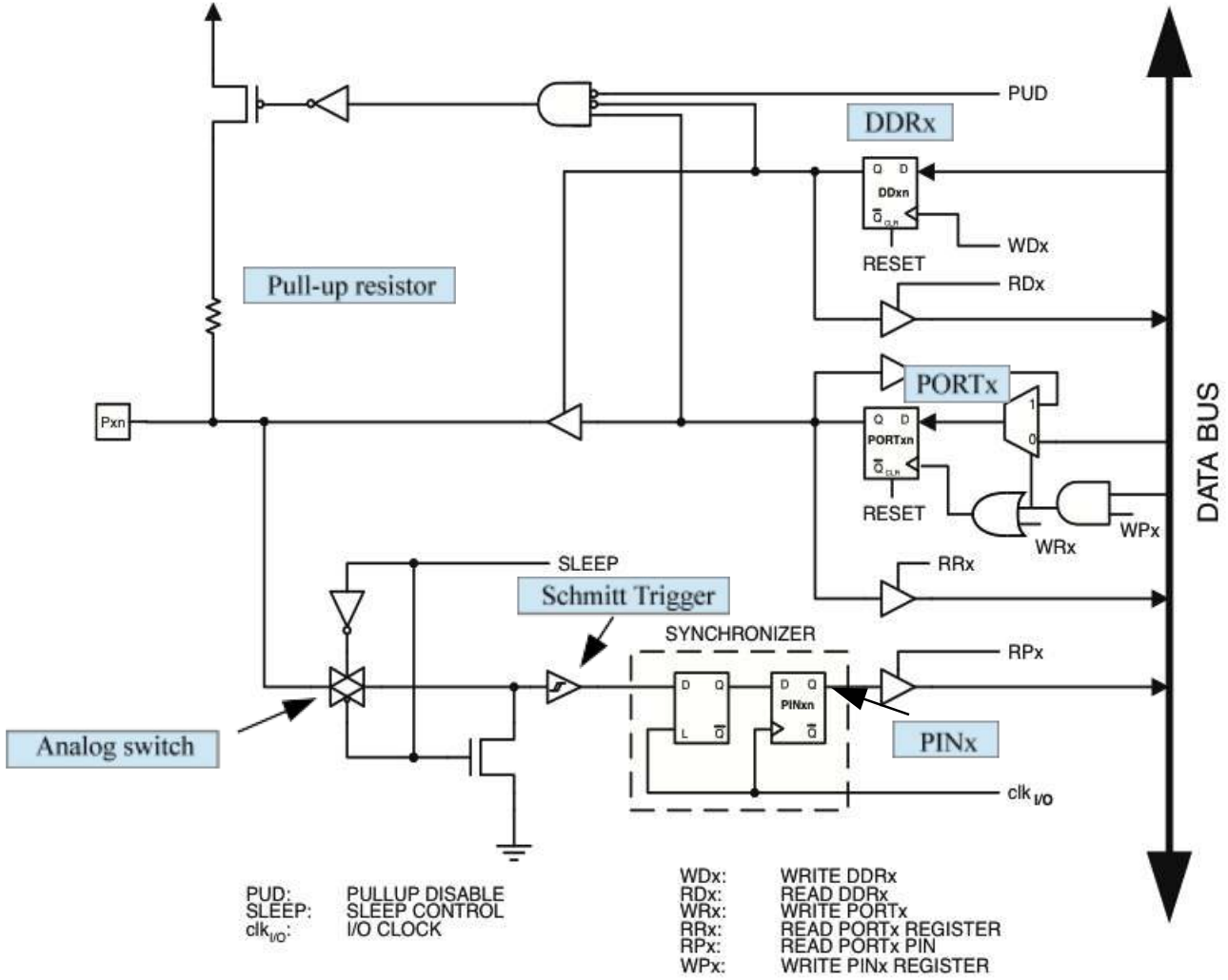
Ci sono quindi varie sorgenti che provocano il reset del microcontrollore:

- Un livello basso sul pin di reset;
- Power-on Reset;
- Brown-out-Reset;
- Watchdog Reset

ATMega328: Timers

Il microcontrollore comprende 2 timers a 8 bit e 1 timer a 16 bit che possono essere utilizzati in vari modi dal programma. Utilizzano un clock interno ma possono anche essere collegati ad un clock esterno.

ATMega328: Input-output digitale



ATMega328: Input-output digitale

Tutte le porte hanno le seguenti caratteristiche:

- resistore di pull-up selezionabile;
- logica 3-state;
- trigger di Schmitt in ingresso;
- sincronismo con il clock di sistema;
- in uscita possono erogare o assorbire corrente.

Tutte le porte degli AVR hanno una funzione di *true Read-Modify-Write* quando vengono usate come I/O digitali. Questo significa che la direzione di un singolo pin puo' essere cambiata al volo senza interferire con gli altri pin.

ATMega328: Input-output digitale

Il controllo di ogni porta e' effettuato con 3 registri:

- DDRx: Data Direction Register;
- PORTx: Output Register;
- PINx: Input Register

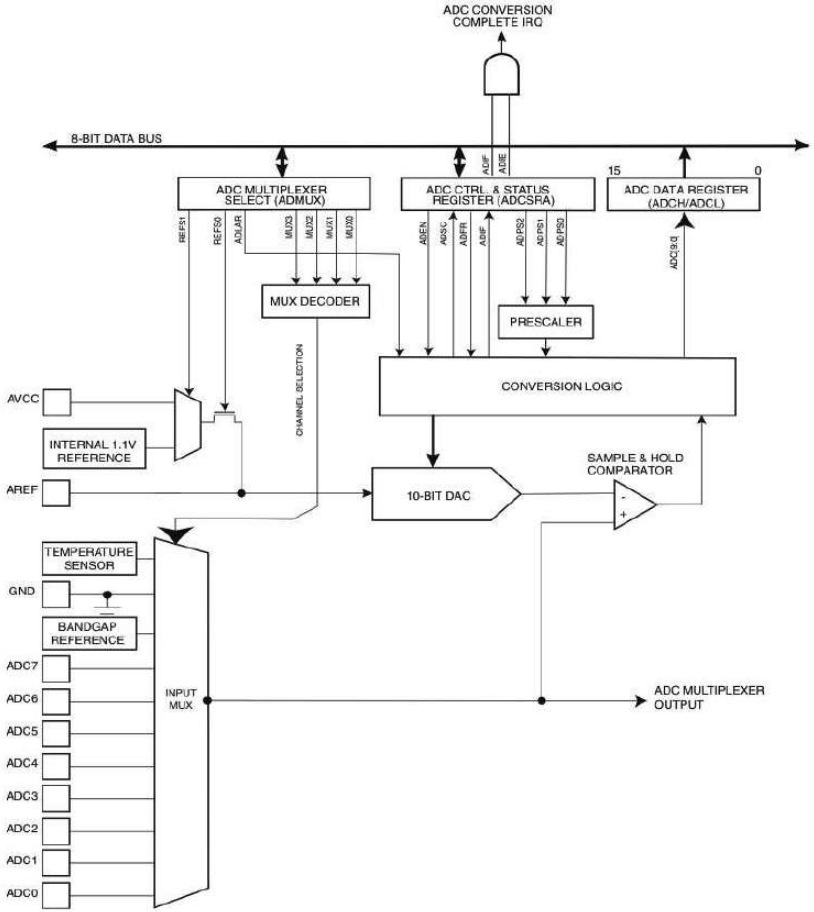
Ad esempio, PORTB3 contiene il bit in uscita per il pin 3 del Port B.

DDx	PORTx	PUD	I/O	Pull-up	Note
0	0	x	Input	No	Alta impedenza
0	1	0	Input	SI	
0	1	1	Input	No	Alta Impedenza
1	0	x	Output	No	
1	1	x	Output	No	

ATMega328: Uscite analogiche

Questa famiglia di microcontrollori non dispone di uscite analogiche. Tuttavia, l'uscita di alcuni pin digitali puo' essere modulata, ovvero utilizzata per emettere un'onda rettangolare (fra 0 e 5 V) con duty cycle variabile. Con un semplice integratore si puo' quindi avere una tensione *quasi* analogica.

ATMega328: Ingressi analogici



ATMega328: Ingressi analogici

Un ADC a 10 bit (sample and hold, successive approximation). Il tempo di conversione e'compreso tra 13 e 260 μs.
E' collegato tramite un multiplexer a 8 ingressi (solo 6 effettivamente utilizzabili, collegati ai pin $PC0 \div PC5$). L'ADC ha un'alimentazione separata (pin AV_{CC}) per migliore immunita' ai disturbi. L'uscita digitale e':

$$\text{Output} = \frac{2^{10} - 1}{V_{ref}} V_{in}$$

Ci sono 3 possibili opzioni per definire V_{ref}:

- AV_{CC};
- 1.1 V (Riferimento di tensione interno a bandgap);
- A_{ref} (Tensione sul pin A_{ref}).

La gestione dell'ADC e' delegata a 4 registri a 8 bit:

- ADMUX: Contiene le informazioni sulla scelta del canale d'ingresso e della V_{ref};
- ADCSRA: bits di status e controlli;
- ADCL e ADCH: contengono i 10 bit del risultato.

ATMega328: Comparatore analogico

Il comparatore analogico compara i valori in ingresso sull'ingresso positivo *AIN*0 e su quello negativo *AIN*1 (ovvero i pin *PD*6 e *PD*7); quando *AIN*0 > *AIN*1 l'uscita del comparatore *ACO* va ad 1. Questo segnale puo' essere usato per generare un Interrupt (sul fronte di salita o di discesa), oppure come trigger per il Timer/Counter 1. E' possibile anche utilizzare, all'ingresso non invertente, la tensione di riferimento interna (1.1 *V*).
E' anche possibile selezionare uno qualunque dei pin analogici di ingresso (*ADC*0 … *ADC*7) come ingresso negativo del comparatore (poiche' utilizza lo stesso multiplexer dell'ADC, quest'ultimo deve essere off)

ATMega328: Comunicazione seriale

Il microcontrollore supporta vari meccanismi di comunicazione con dispositivi esterni:

- USART (Universal Synchronous Asynchronous Receive Transmit): Pin *PD*0 e *PD*1;
- SPI (Serial Peripheral Interface): Pin *PB*2, *PB*3, *PB*4, *PB*5;
- I2C/TWI (Inter Integrated Circuit /Two Wire Interface: Pin *PC*4 e *PC*5.

Sono anzitutto necessari per consentire all'utilizzatore di caricare e gestire il programma che deve essere eseguito sul microcontrollore, ma consentono anche di gestire dispositivi esterni (integrati) ovvero di trasferire dati verso l'esterno.
Ne discuteremo in dettaglio piu' avanti.

IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 57, NO. 6, DECEMBER 2010

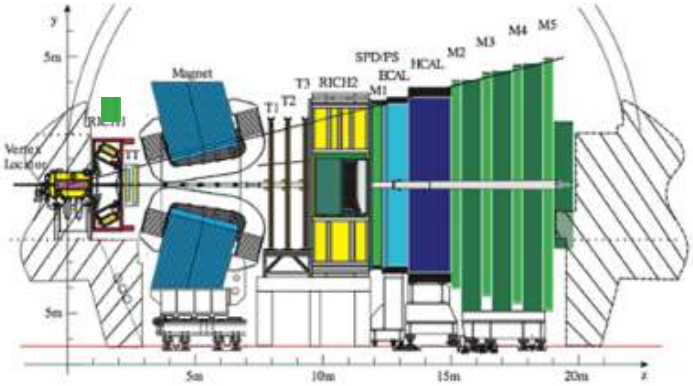
The Muon Front-End Control Electronics of the LHCb Experiment

Valerio Bocci, Giacomo Chiodi, Francesco Iacoangeli, Francesco Messi, and Rafael A. Nobrega

***Abstract*—The LHCb muon readout apparatus is made of 1368 Multi-Wire Proportional Chambers (MWPC) and 24 Gas Electron Multiplier (GEM) chambers connected to 7632 16-channel front-end boards, resulting in 122.112 channels to be read out.**

The large-scale of the system and the time constrains naturally led to the development of a custom and complex control system made of about 600 microcontrollers (μC) and 150 flash-based FPGAs which are directly connected to the front-end electronics and handled by six computers.

Muon Chambers



ELMB the Arduino of HEP (ATMega128 MCU)

Henk Boterenbrood
NiKhef ELMB

Björn Hallgren
CERN

156 x Service Board (SB)

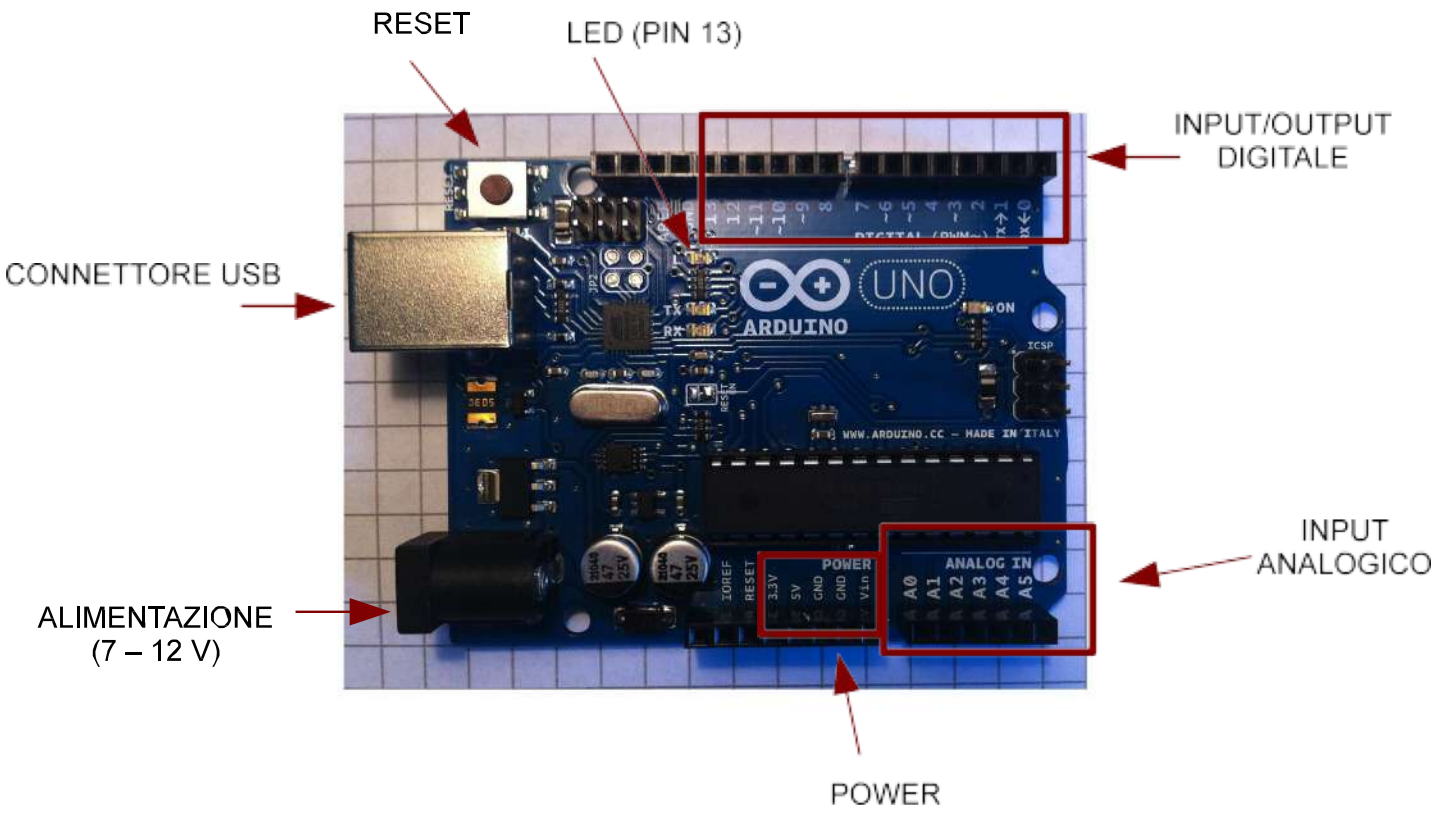
Complex Software
only for real expert.
C programming.
Automotive CANBus in
radiation environment.

Arduino

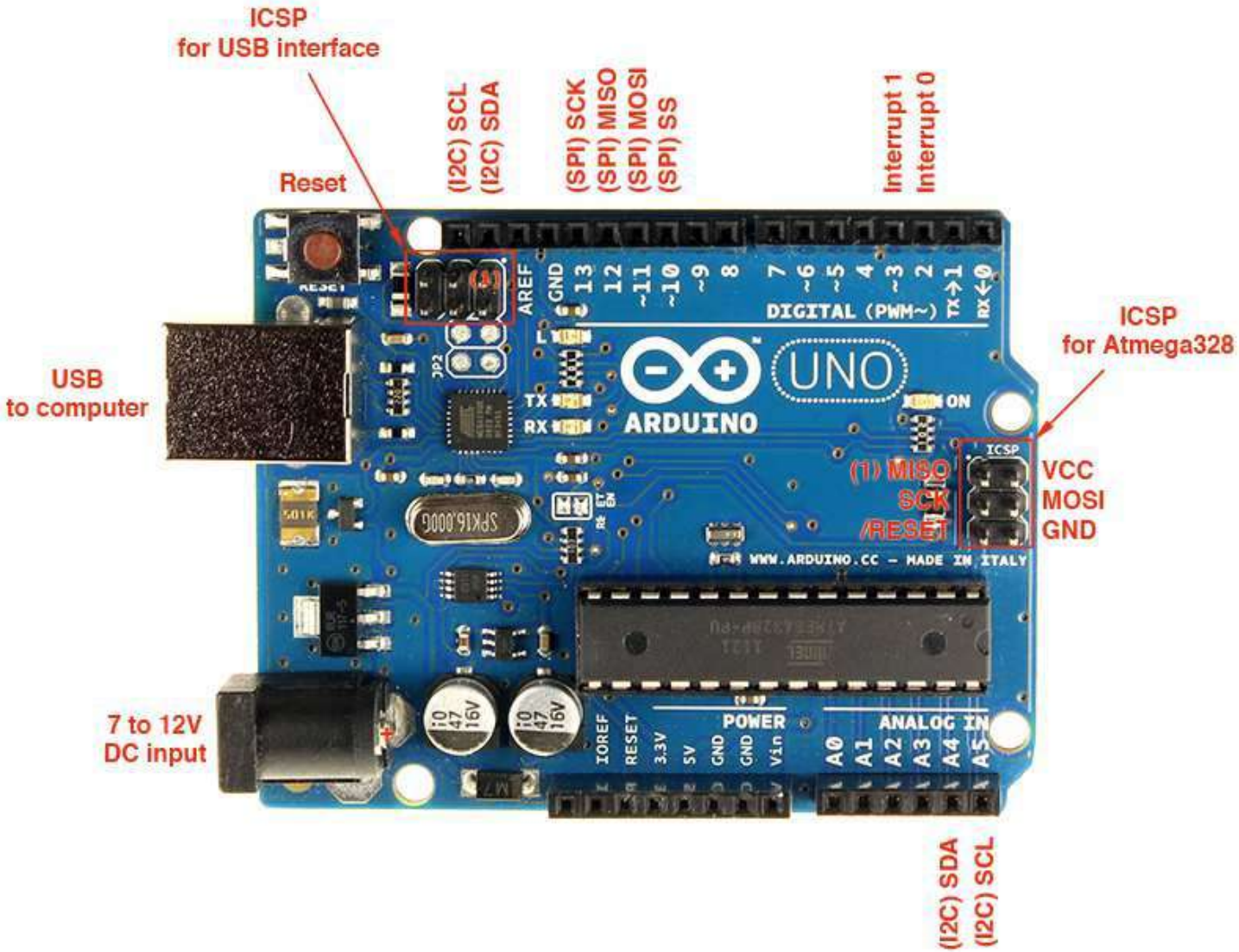
E’ un sistema di sviluppo di piccole dimensioni basato su Microcontrollori ATMEL, adatto per sviluppo di prototipi e per scopi didattici.
E’ integrato con un ambiente di sviluppo software (open source) che permette un facile e rapido utilizzo del microcontrollore.
Esistono varie versioni, noi utilizzeremo la scheda Arduino Uno, basata sul microcontrollore ATMEL ATMEGA328.

Arduino Uno: la scheda

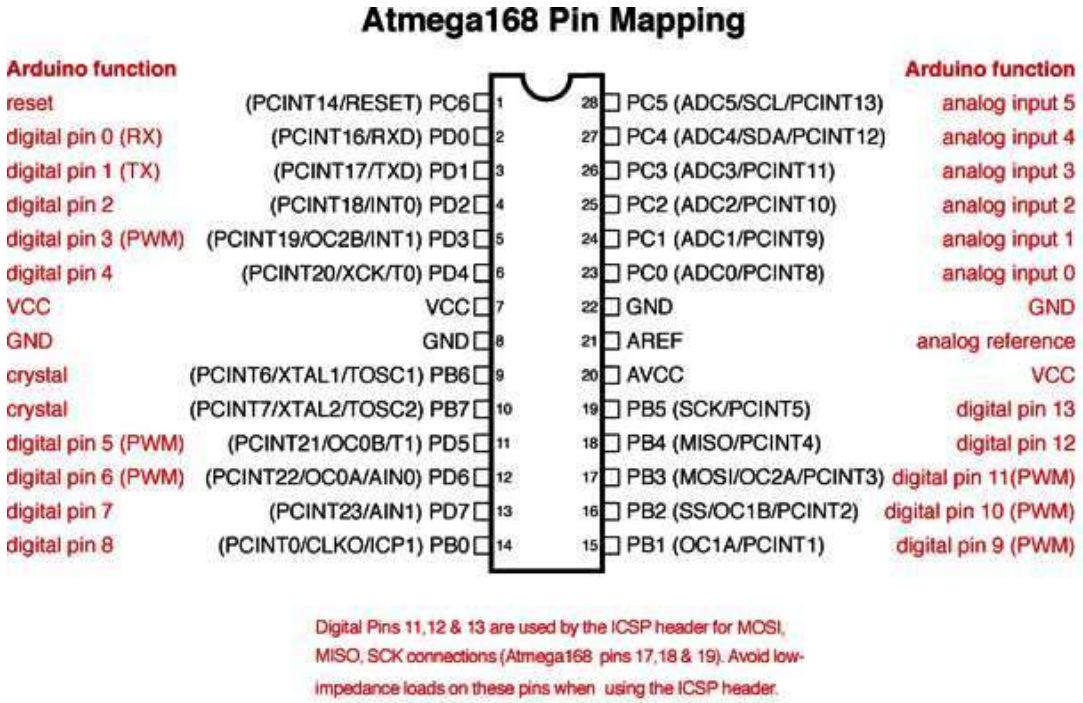
Arduino Uno: la scheda



Arduino Uno: la scheda



Arduino: pin mapping del ATMega328



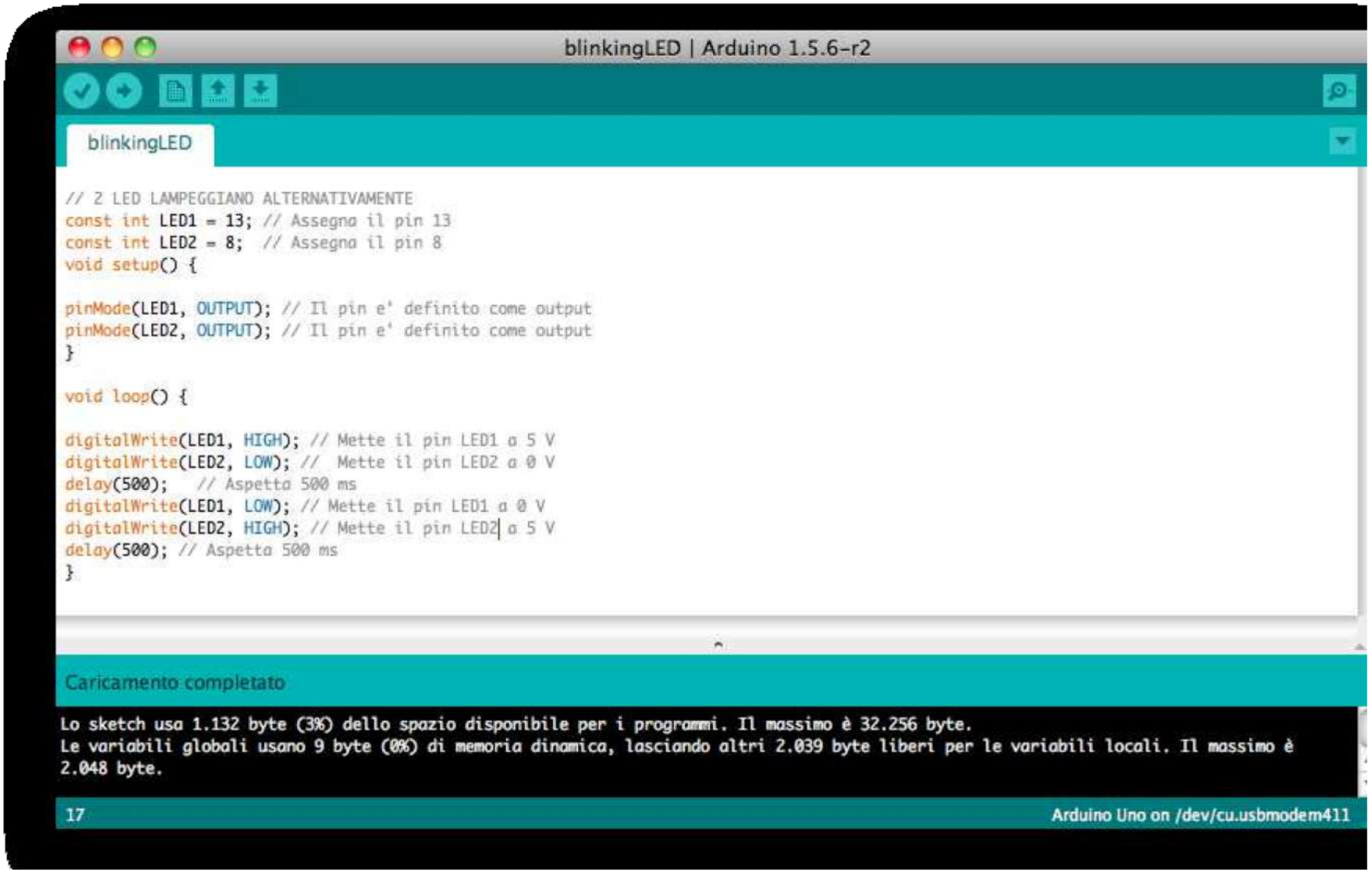
Ambiente di sviluppo

- Arduino dialoga con un computer host (PC Windows, Mac, Linux) tramite la porta USB;
- E' necessario installare sull'host il programma Arduino (open source). E' scritto in Java e quindi in grado di funzionare su molte piattaforme.
- La programmazione del microcontrollore e' in linguaggio C.

La finestra di scrittura del programma

Offre un *framework* per la scrittura dello *sketch* (come e' chiamato nel gergo di Arduino)

Semplice esempio



Semplice esempio (2)

Prima di compilare Arduino trasforma lo sketch creando il programma in *c* completo:

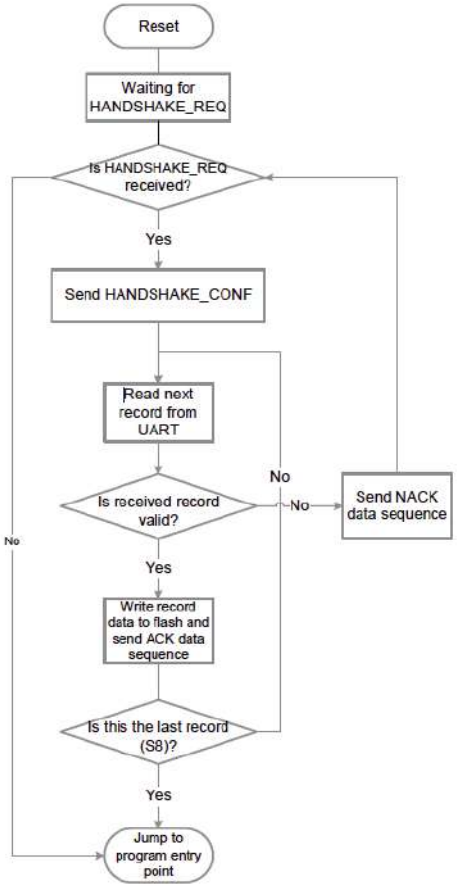
```
#include "Wprogram.h";
void setup();
void loop();
void setup() {
pinMode(LED1, OUTPUT);//Il pin e' definito come output
pinMode(LED2, OUTPUT);//Il pin e' definito come output
}
void loop() {
digitalWrite(LED1, HIGH);//Mette il pin LED1 a 5V digit
alWrite(LED2, LOW);//Mette il pin LED2 a 0V delay(500)
;//Aspetta 500ms
digitalWrite(LED1, LOW);//Mette il pin LED1 a 0V
digitalWrite(LED2, HIGH);//Mette il pin LED2 a 5V delay
(500);//Aspetta 500 ms
}
int main() {
setup();
for(;;){loop();}
return 0;
}
```

Semplice esempio (3)

Il compilatore (residente sul PC) crea il codice eseguibile. Il codice viene poi trasferito sulla memoria flash del *μC* (attraverso la connessione USB) ed eseguito.
Dal lato del *μC* il trasferimento e la scrittura in memoria sono eseguiti da un programma residente (BootLoader) che occupa gli ultimi 512 bytes della memoria flash.
Il programma e' memorizzato permanentemente e viene eseguito ogni volta che si accende la scheda (anche senza aprire Arduino sul PC), finche' non e' sostituito da un altro programma.

Il bootloader

Al momento dell'accensione (o al reset) il boot-loader si avvia e verifica se c'e' una richiesta di comunicazione proveniente dalla porta seriale. Se non c'e' passa il controllo all'applicazione esistente.
Se invece c'e' avvia il dialogo tramite la porta seriale, scarica l'applicazione nuova e poi la avvia.
La presenza del bootloader non è indispens-abile. Il programma da eseguire puo' essere caricato connettendo il *μC* ad una apposita scheda (programmatore).
Questa soluzione e' adatta per situazioni in cui non si prevede che l'applicazione debba subire modifiche o aggiornamenti.



Linguaggio

Un sommario del linguaggio di Arduino puo' essere reperito su:

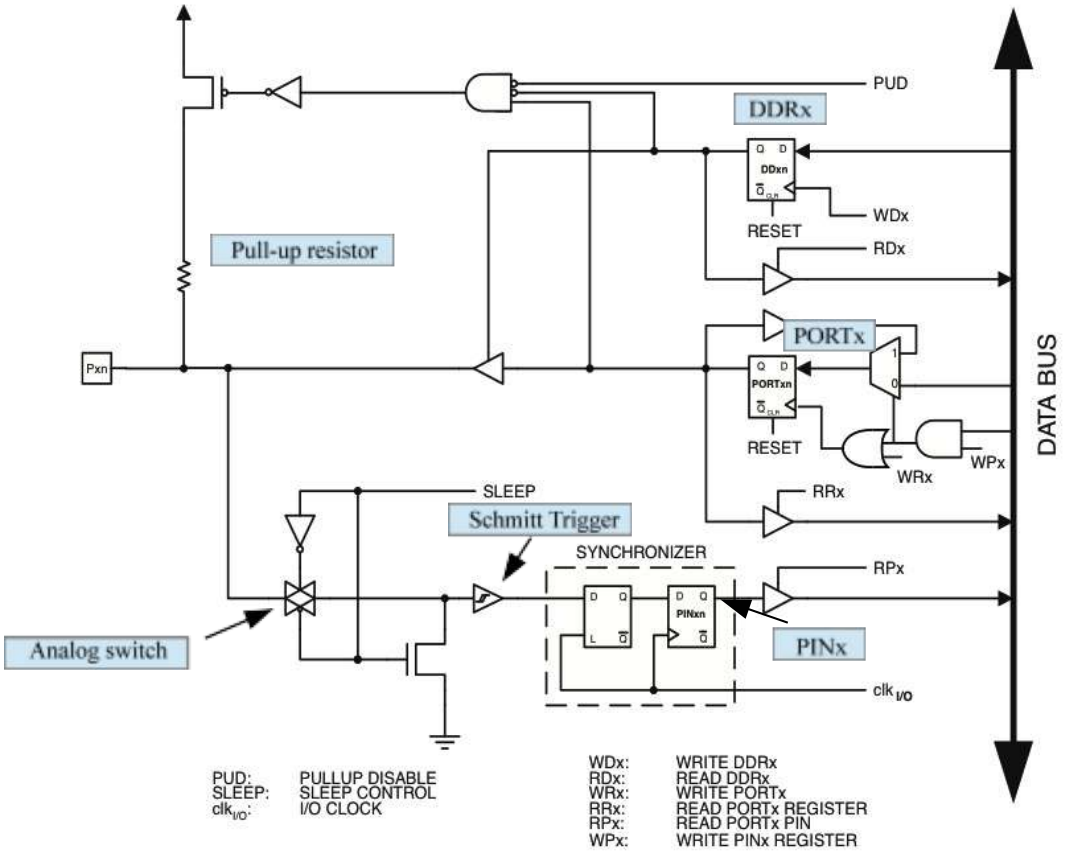
http://arduino.cc/en/Reference/HomePage

E' disponibile un'ampia libreria software per molteplici applicazioni.
Qui vedremo solo alcune cose fondamentali per iniziare.

I/O Digitale

- pinMode(pin,mode); (mode: *INPUT*, *OUTPUT* , *INPUT _PULLUP*)
- digitalWrite(pin, value); (value: *HIGH*, *LOW*)
- digitalRead(pin);

ATMega328: PULL UP



I/O Digitale: esempio

```
int led Pin = 13;
in t i n Pin = 7 ;
in t  val  = 0;
void  setup ()
{
   pinMode ( led Pin , OUTPUT) ;
   pinMode ( in Pin , INPUT ) ;
}
void loop ()
{
   val = digitalRead ( inPin ) ;
   digitalWr ite ( ledPin , val ) ;
}
```

Output Analogico

I pin 3,5,6,9,10,11 possono, in uscita, essere utilizzati in modo “analogico” (PWM). (Pin 5,6: 980 Hz; Pin 3,9,10,11: 490 Hz)

Esempio:

```
int led Pin = 9;              //LED connected to digital pin 9
in t val = 128 ;                 //val : 0−255 ( duty cycle )
void setup ()
{
   pinMode ( led Pin , OUTPUT) ;       //sets the pin as output
}
void loop ()
{
   analogWrite ( led Pin , val ) ;     //
}
```

ADC

Il microcontroller contiene un ADC a 10 bit (sample and hold, successive approximation).
E' collegato tramite un multiplexer a 8 ingressi (6 effettivamente utilizzabili su Arduino).

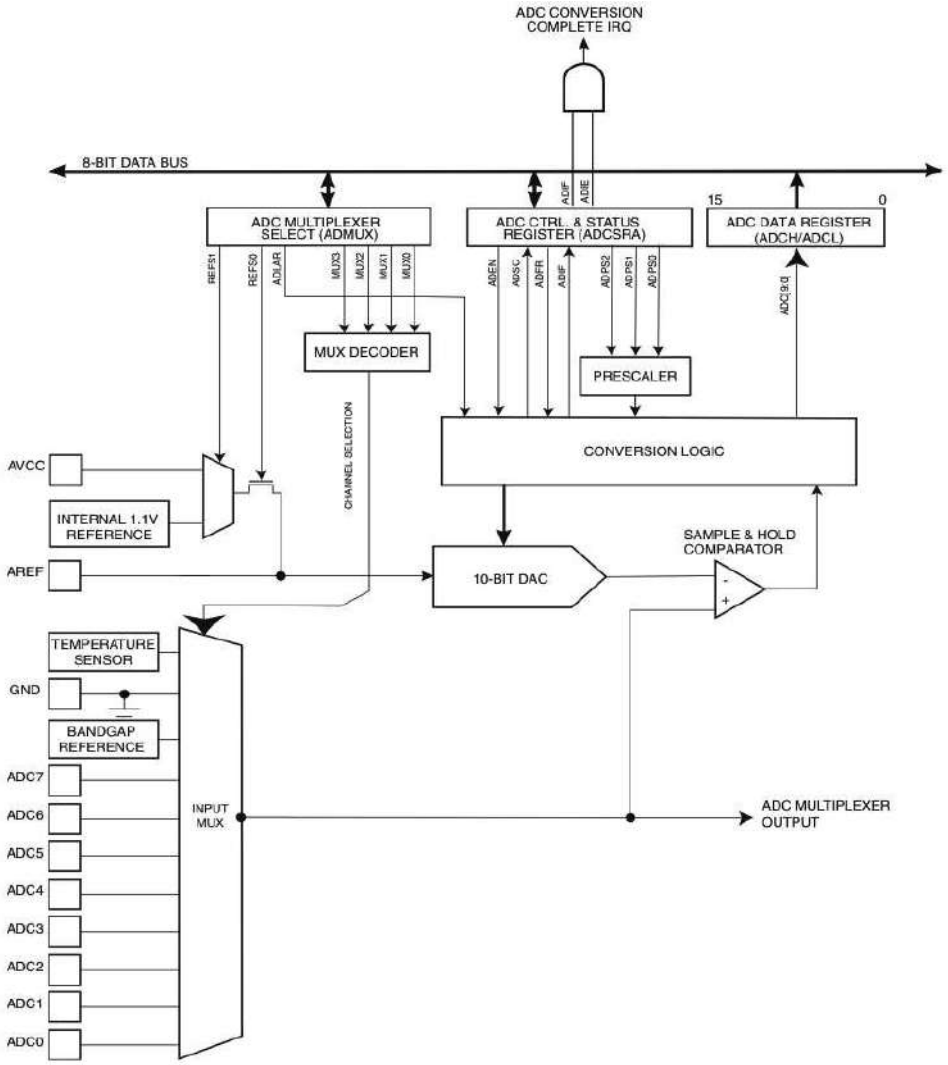
$$Output = \frac{2^{10}}{V_{ref}} V_{in}$$

3 possibili V_{ref}:

- 5 V (default)
- 1.1 V
- A_{ref} (esterna)

Nota bene:

$$0 \leq A_{ref} \leq +5\ V$$



Esempio

```
int analog Pin = 3;              //analog pin  3
in t val = 0;                    //v a r i a b l e to s tore the value read
void  setup ()
{
   analogReference (EXTERNAL) ; //La Vref e ' f o r n i t a esternamente  Se
   ria l . begin (9600);                 //    setup s e r i a l
}
void loop ()
{
   val = analogRead ( analog Pin ) ;     //read the i nput pin
   Se ria l . p r i n t In ( val ) ;     //debug value
}
```

Comunicazione

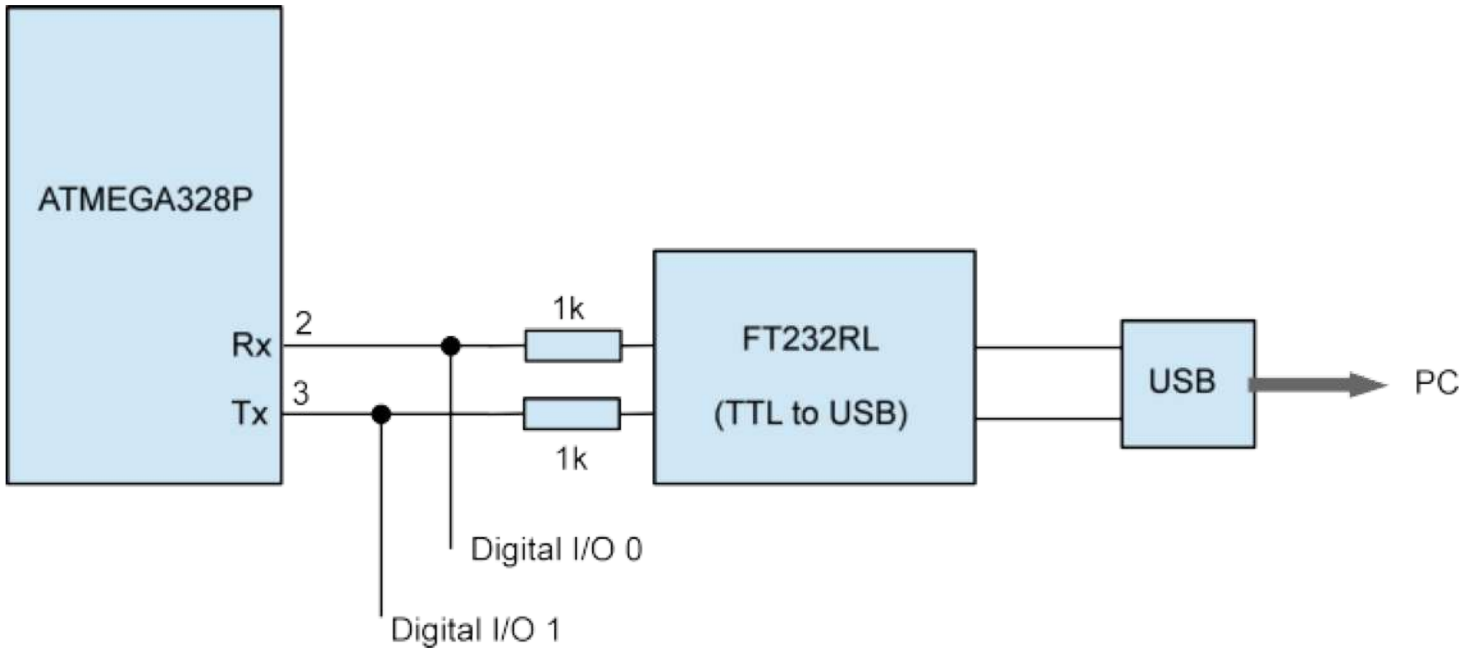
Il microcontrollore supporta vari meccanismi di comunicazione con dispositivi esterni:

- USART (Universal Synchronous Asynchronous Receive Transmit)
  Digital I/O 0 e 1
- SPI (Serial Peripheral Interface)
  Digital I/O 10, 11, 12, 13
- I2C/TWI (Inter Integrated Circuit /Two Wire Interface
  Analog pin A4, A5

Arduino fornisce librerie che facilitano l'uso di questi protocolli.

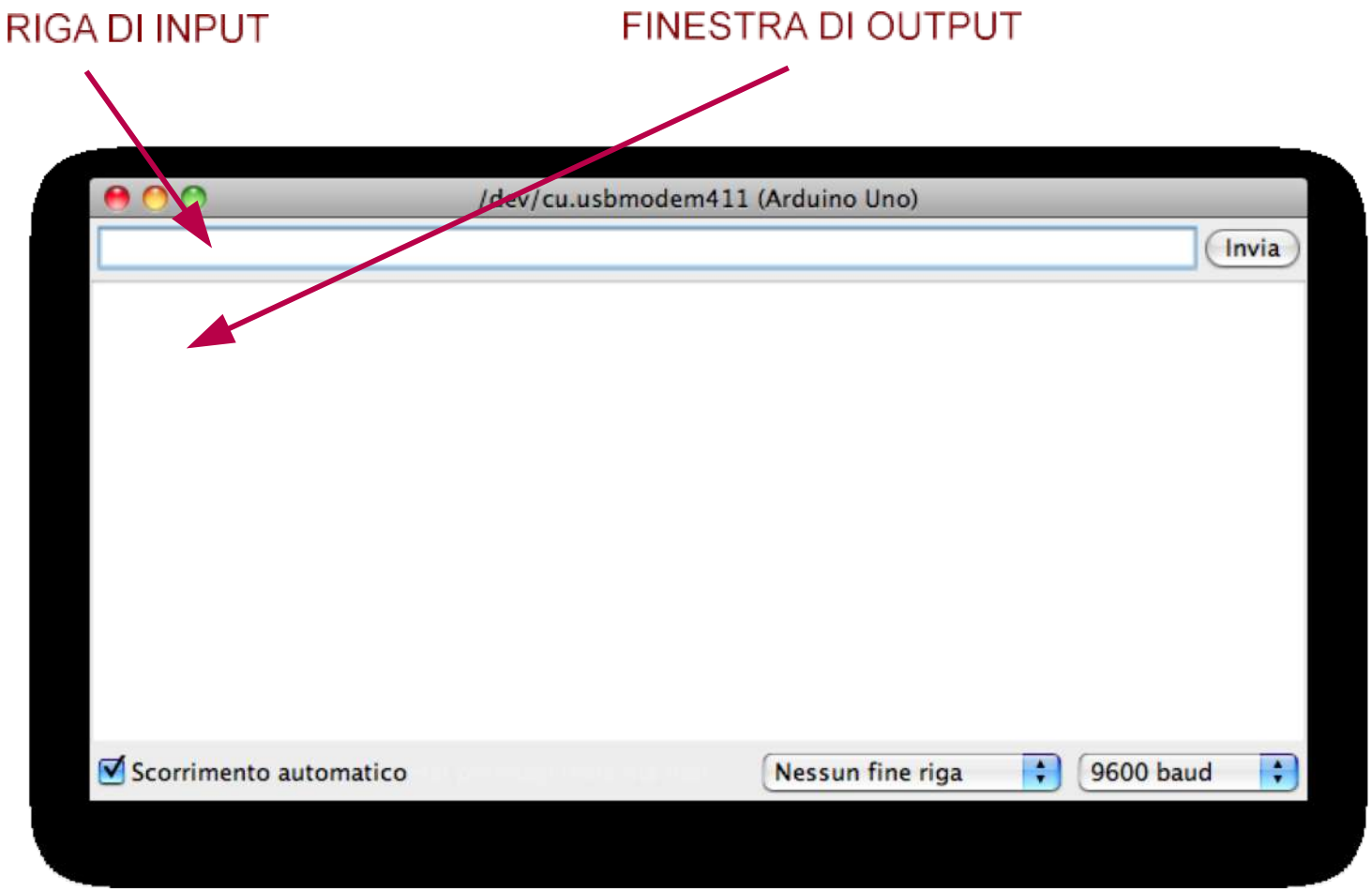
Comunicazione seriale USART

E' utilizzata da Arduino per connettere il μC al PC, tramite la porta USB: da qui viene fatto il download del programma.



Puo' liberamente essere poi utilizzata dall'utente.

Comunicazione seriale; finestra di monitor



Comunicazione seriale: esempio

```
int data = 0;
void setup () {
Serial.begin(9600); // Inizializzazione baud rate
}

void loop () {
   if (Serial.available() > 0) // verifica se esiste un INPUT
   {
data = Serial.parseInt();       //legge come numero intero
Serial.println(data);           // scrive sulla finestra di OUTPUT
delay(200);
   }
}
```

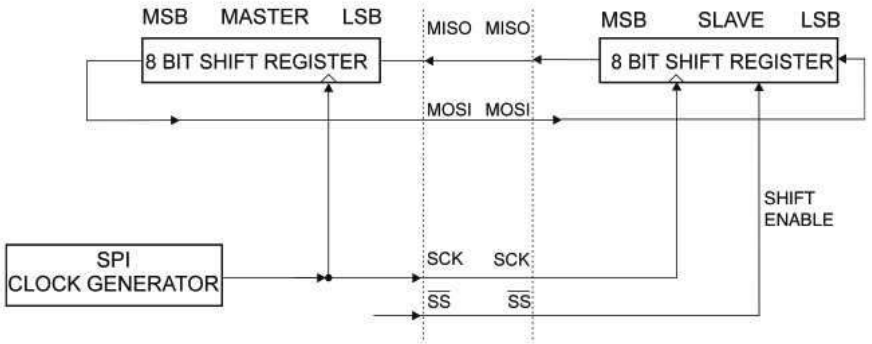
NOTA BENE:
Se si usa la comunicazione seriale i pin I/O 0 *e* 1 *non possono essere usati per altri scopi!*

SPI (Serial Peripheral Interface)

E' un sistema di comunicazione tra un microcontrollore e altri circuiti integrati o tra più microcontrollori.
La trasmissione avviene tra un dispositivo detto master e uno o più slave. Il master controlla il bus, emette il segnale di clock, decide quando iniziare e terminare la comunicazione.

Richiede 4 linee di comunicazione:

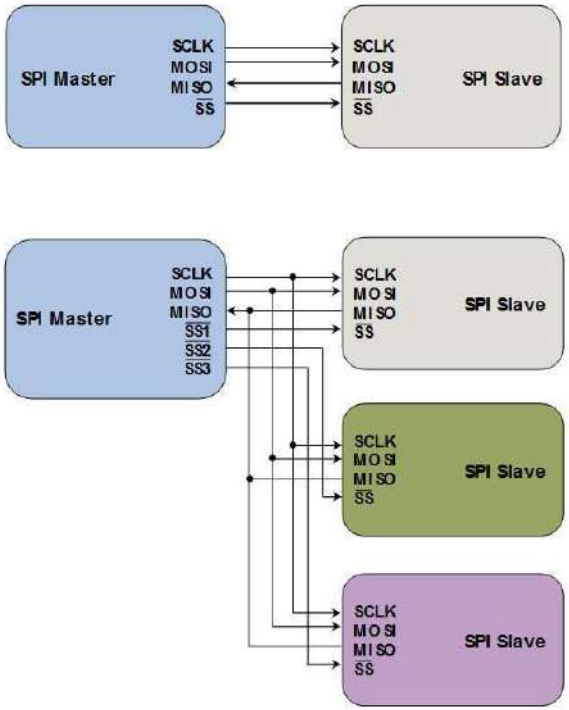
- SCK: Serial Clock;
- MOSI: Master Output - Slave Input;
- MISO: Master Input - Slave Ouput;
- SS: Slave Select



I vari dispositivi si scambiano dati utilizzando Shift Register.
(Libreria Arduino: SPI library)

SPI (Serial Peripheral Interface)-2

Esempi di connessione SPI



SPI (Serial Peripheral Interface)-3

Temporizzazione

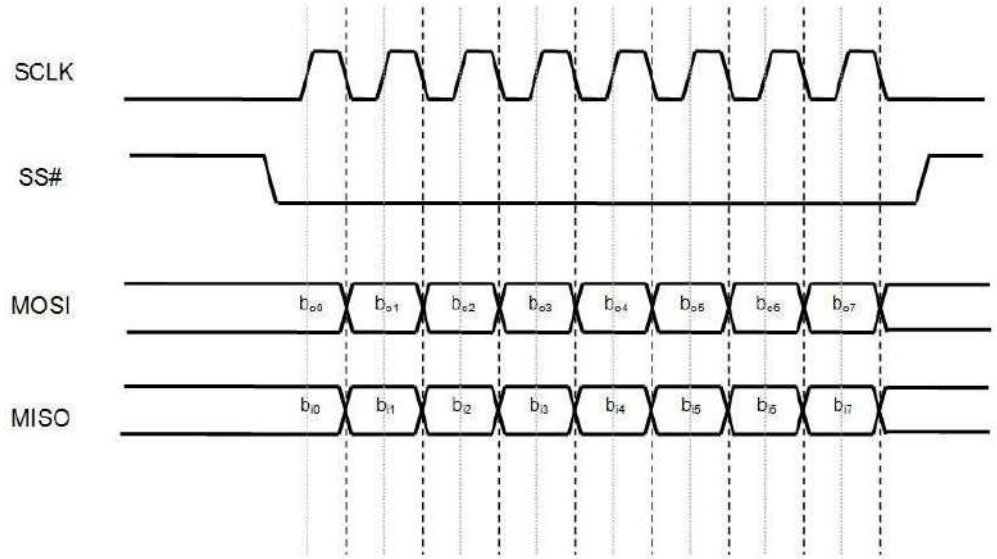


Figure 2 : A simple SPI communication. Data bits on MOSI and MISO toggle on the SCLK falling edge and are sampled on the SCLK rising edge. The SPI mode defines which SCLK edge is used for toggling data and which SCLK edge is used for sampling data.

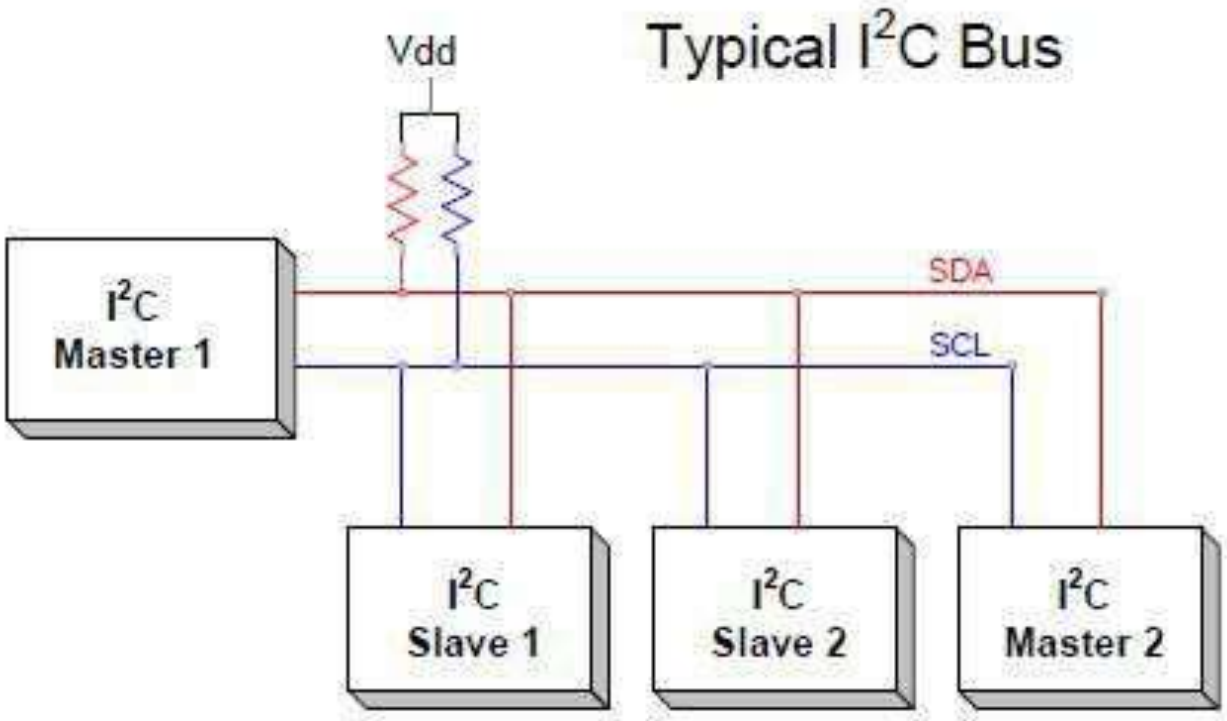
I^2C / TWI (Inter Integrated Circuit)

E' un sistema di comunicazione seriale bifilare utilizzato tra circuiti integrati.
Richiede un master e uno o piu' slave.
Linee

- SDA: Serial DAta Line, per i dati;
- SCL: Serial Clock Line, segnale di Clock (emesso dal master)

(Occorre poi anche una linea di GND comune e una linea di alimentazione per i pull-up).
Si possono collegare vari slave (ognuno con indirizzo diverso).
(Libreria Arduino: Wire library)

I^2C / TWI (Inter Integrated Circuit)-2



I^2C / TWI (Inter Integrated Circuit)-3

START	Slave address	Rd/nWr	ACK	Data	ACK	Data	ACK	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Example 1: writing 2 byte to a slave. The data put on the bus by the master are shaded.

START	Slave address	0	0	Data	0	Data	0	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Example 2: reading 2 bytes from a slave. The data put on the bus by the master are shaded.

START	Slave address	1	0	Data	0	Data	1	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Figure 5: Typical I^2C transfer, with 2 bytes of data. The master initiates the transfer with a START condition, followed by the slave address and the transfer type (read or write) bit. The slave acknowledges its address. Each data byte is then transmitted and acknowledged by the receiver. When it receives data, the master can issue a not-acknowledge condition (NACK) when it has received enough data. The bus is released when the master issues a STOP condition.

Interrupts

La scheda Arduino Uno puo' gestire 2 fonti di interrupts

- Interrupt 0: Digital I/O 2;
- Interrupt 1: Digital I/O 3;

Quando arriva un interrupt viene eseguita una routine specificata. Alla fine il controllo torna al programma precedentemente in esecuzione.
Esempio:

```
void setup ()
{
   Serial.begin(9600);
   attachInterrupt(0, myprog, CHANGE);      //interrupt se I/O 2 cambia
}
void loop ()
{
     ....
     ....
}
void myprog ()
{
// InterruptService Routine
  Serial.println("E' arrivato un interrupt");
}
```

Interrupts:funzioni

attachInterrupt(interrupt, ISR, mode);

- *interrupt:* 0 *o* 1*;*
- *ISR: nome della routine di servizio;*
- *mode: LOW,CHANGE, RISING, FALLING*

detachInterrupt(interrupt);

Elimina l'interrupt precedentemente abilitato.

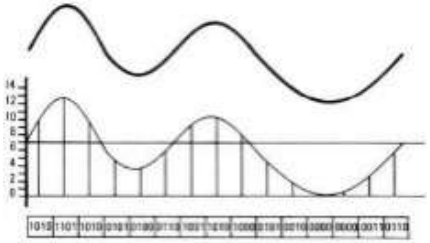
noInterrupts();

Disabilita tutti gli interrupts.

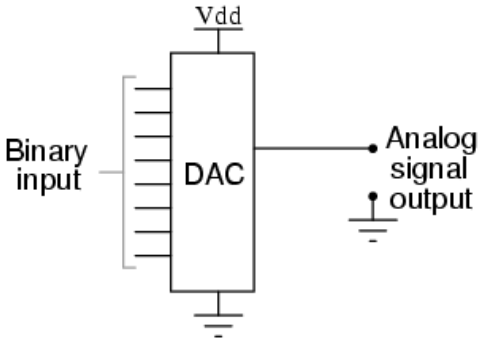
interrupts();

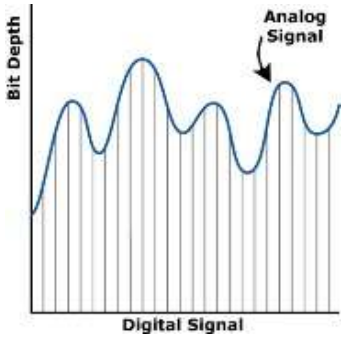
Abilita gli interrupts (precedentemente disabilitati).

Arduino DUE ha tutto il necessario per realizzare un sistema di acquisizione per interfacciarsi al mondo analogico.



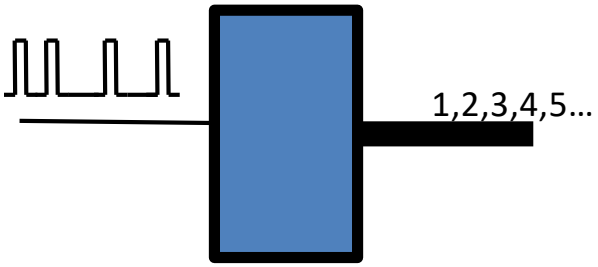
Convertitori Analogico Digitali (ADC):
Consentono di trasformare i segnali analogici in numeri.



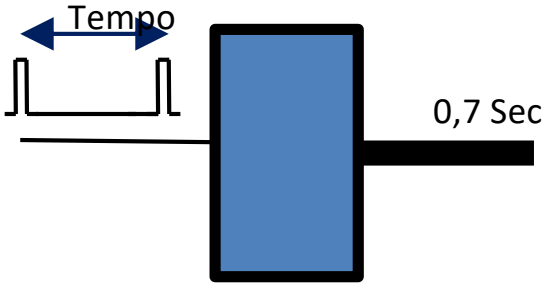


Convertitori Digitale Analogico (DAC):
Consentono di trasformare i numeri in segnali analogici.

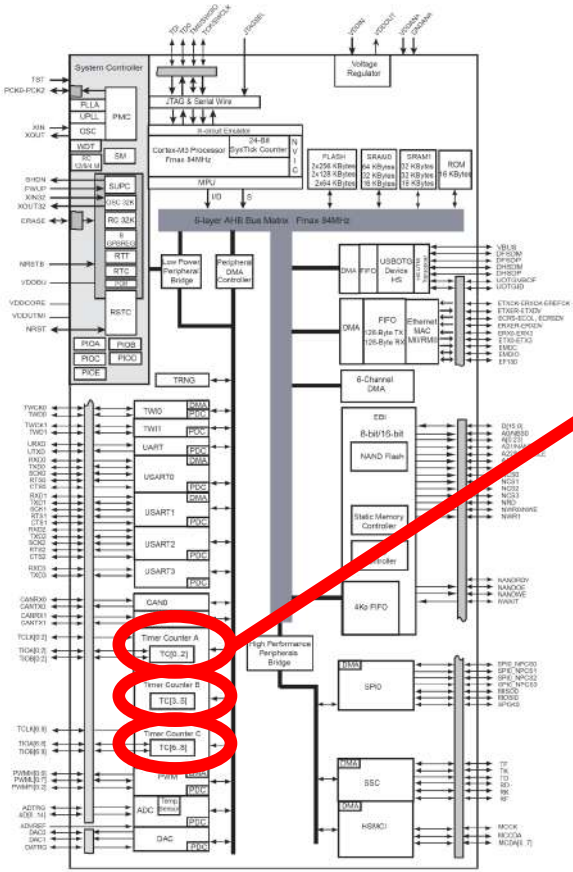
Arduino DUE sistema di acquisizione completo (2/2).

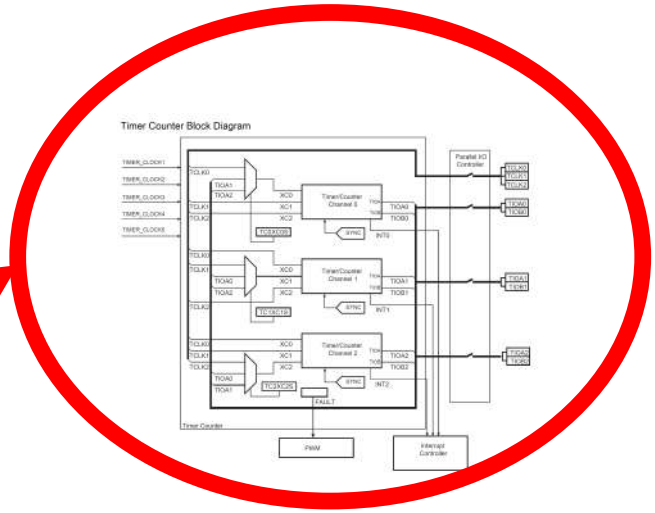


Contatori Digitali
Consentono di contare il numero di eventi.

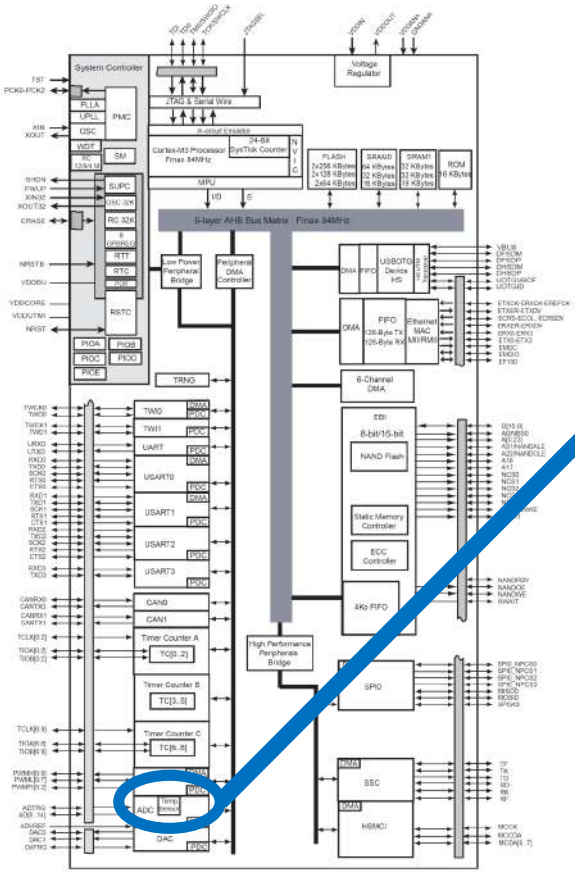


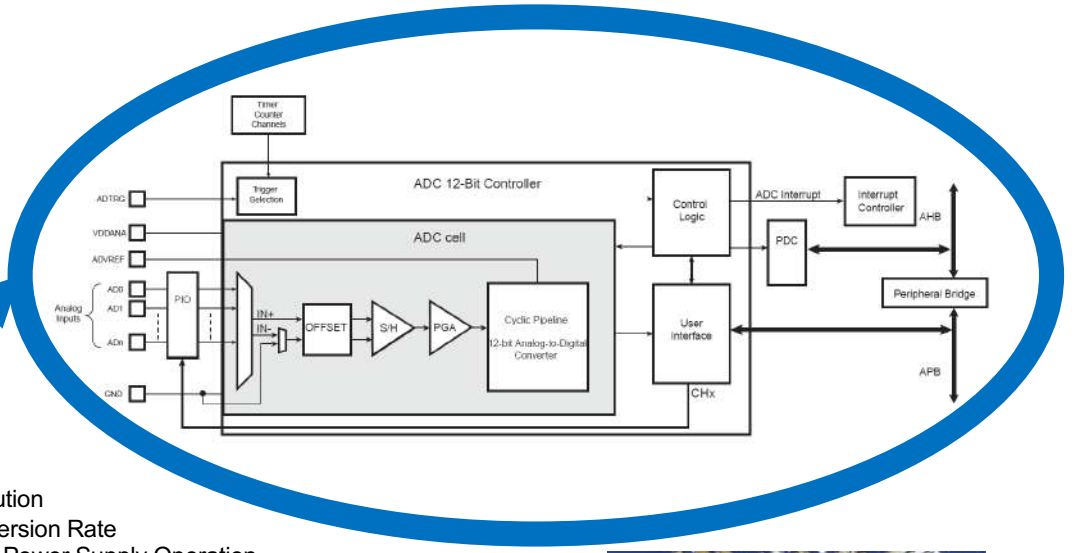
Misuratori di tempo digitali (TDC)
Consentono di misurare il tempo tra due eventi.



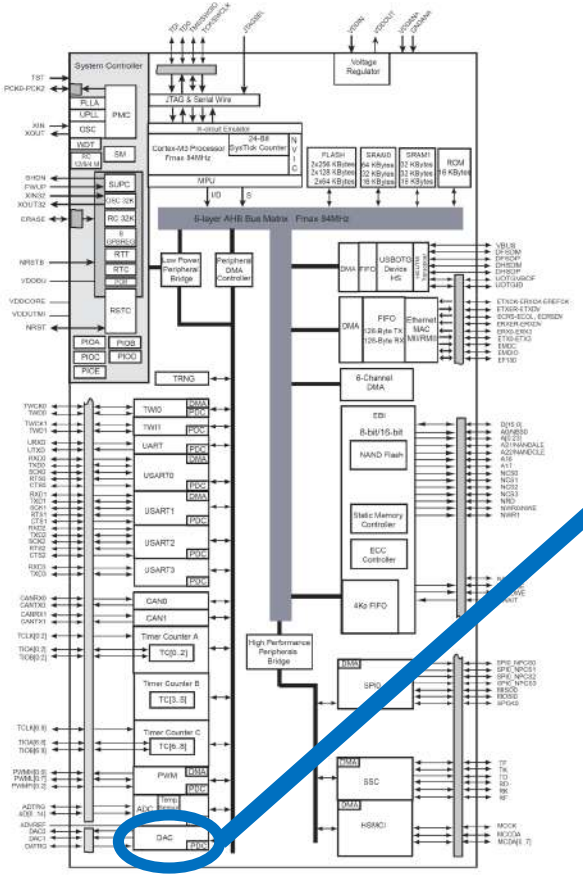


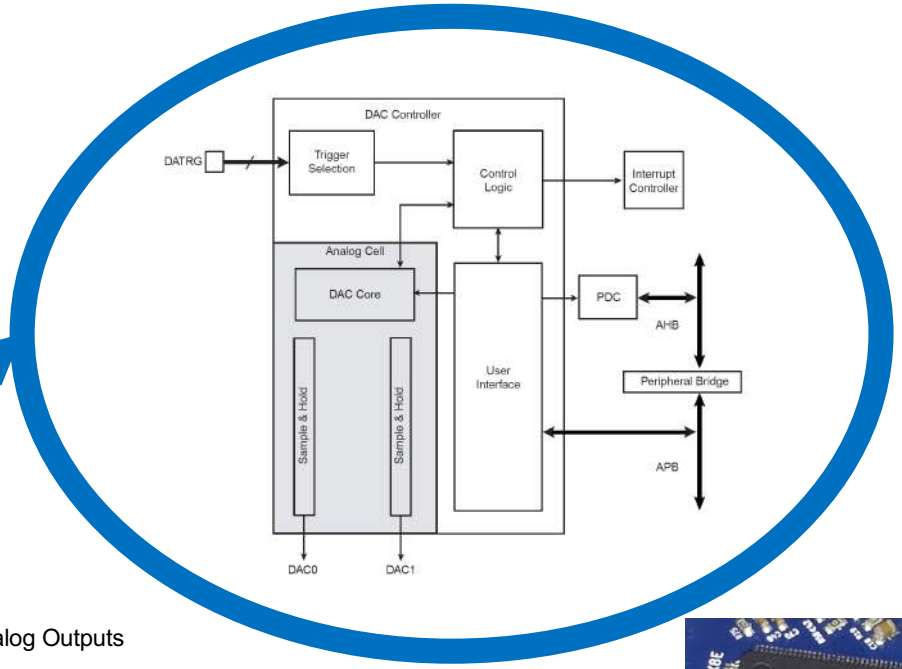
- Three 32-bit Timer Counter Channels
- A Wide Range of Functions Including:
  - Frequency Measurement
  - Event Counting
  - Interval Measurement
  - Pulse Generation
  - Delay Timing
  - Pulse Width Modulation
  - Up/down Capabilities
- Each Channel is User-configurable and Contains:
  - Three External Clock Inputs
  - Five Internal Clock Inputs
  - Two Multi-purpose Input/Output Signals
- Internal Interrupt Signal



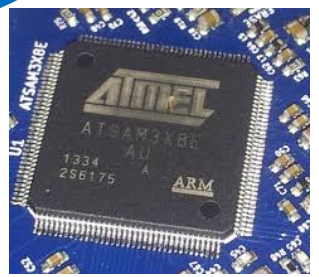


- 12-bit Resolution
- 1 MHz Conversion Rate
- Wide Range Power Supply Operation
- Selectable Single Ended or Differential Input Voltage
- Programmable Gain For Maximum Full Scale Input Range 0 - VDD
- Integrated Multiplexer Offering Up to 16 Independent Analog Inputs
- Individual Enable and Disable of Each Channel
- Hardware or Software Trigger
  – External Trigger Pin
  – Timer Counter Outputs (Corresponding TIOA Trigger)
  – PWM Event Line
- Drive of PWM Fault Input
- PDC Support
- Possibility of ADC Timings Configuration
- Two Sleep Modes and Conversion Sequencer





- Two Independent Analog Outputs
- 12-bit Resolution
- Individual Enable and Disable of Each Analog Channel
- Hardware Trigger
  – External Trigger Pins
- PDC Support
- Possibility of DACC Timings and Current Configuration
- Sleep Mode
  – Automatic Wake-up on Trigger and Back-to-Sleep Mode after Conversions of all Enabled Channels
- Internal FIFO



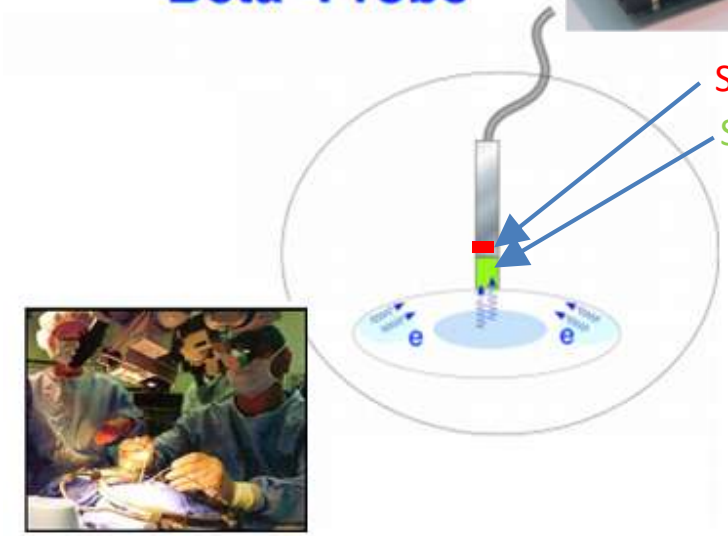
Application Example 1: Intraoperative β- Detecting Probe

A novel radioguided surgery technique exploiting β^- decays

E. Solfaroli Camillocci, G. Baroni, F. Bellini, V. Bocci, F. Collamati, M. Cremonesi, E. De Lucia, P. Ferroli, S. Fiore, C. M. Grana, M. Marafini, I. Mattei, S. Morganti, G. Paganelli, V. Patera, L. Piersanti, L. Recchia, A. Russomando, M. Schiariti, A. Sarti, A. Sciubba, C. Voena & R. Faccini

ArduSiPM

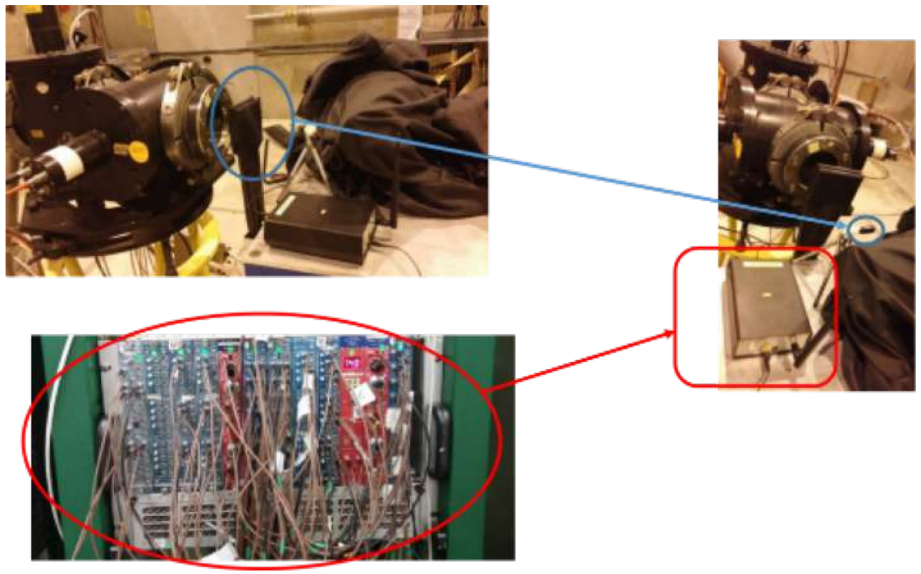
Control and readout
Android App



- Radioguided **intraoperative beta probe**, with scintillation material coupled with SiPM detector.

Application Example 2:
Use of ArduSiPM in the CERN UA9 and CRYSBEAM activity

(substitute old Scintillator and electronics for PM)

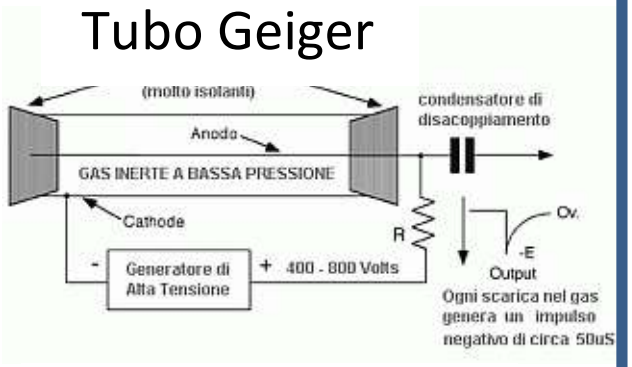


- As beam trigger @ extracted beam line H8 (CERN)

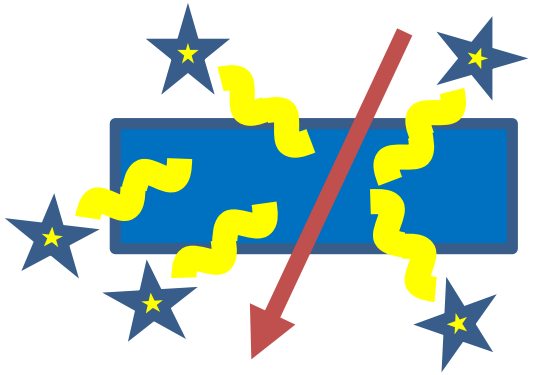
- As beam losses counter @ SPS

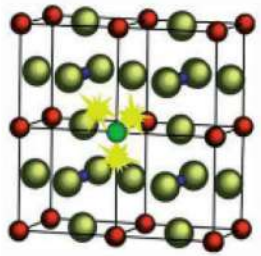
- *This work has been supported by the ERC Ideas Consolidator Grant*
- *No.615089 "CRYSBEAM".*

Tubo Geiger

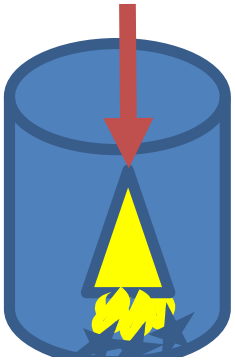


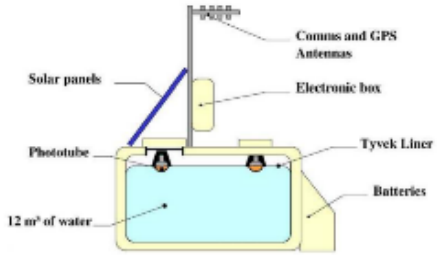
Materiali scintillanti



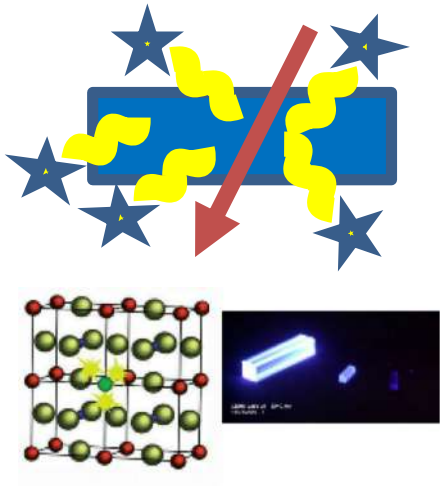


Effetto Cherenkov

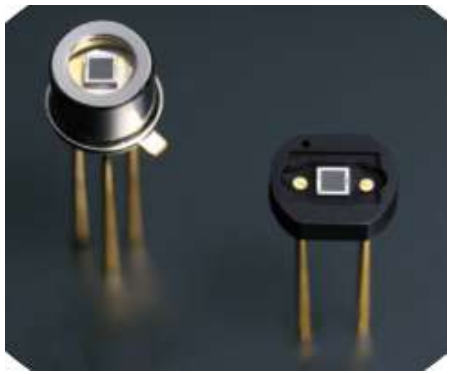


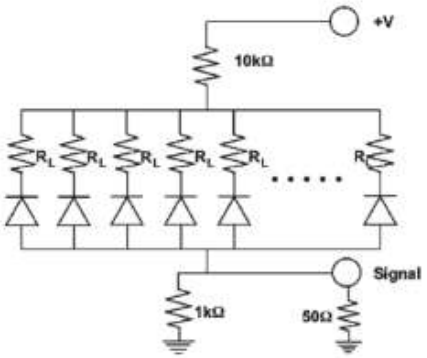


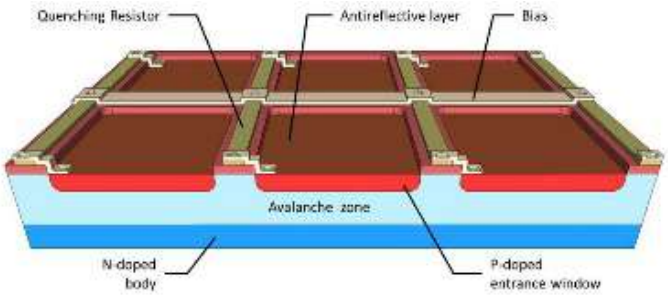
L’utilizzo degli scintillatori in passato non era alla portata di tutti.
Per rilevare i fotoni emessi era necessario uno rivelatore di luce chiamato FOTOMOLTIPLICATORE.

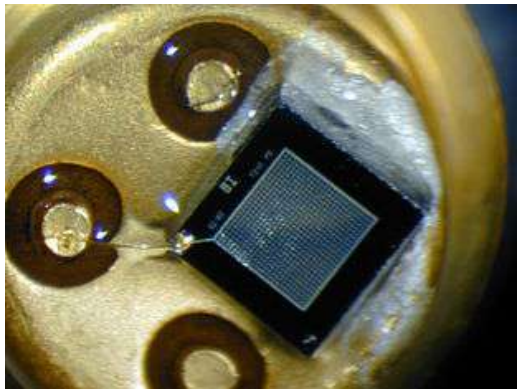


La tecnologia del fotomoltiplicatore, nata nel 1934 si basa sull’effetto fotoelettrico (1921 Einstein Premio Nobel) , e sull’emissione secondaria di elettroni. I fotomoltiplicatori sono oggetti costosi (Keuro) e da laboratorio, necessitano di tensioni dell’ordine dei 1000 Volt e sono molto fragili.







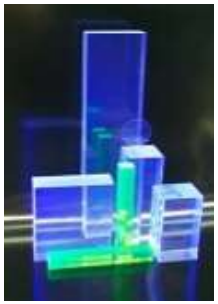


SiPM

Il SiPM o Silicon Photomultiplier è un nuovo tipo di fotomoltiplicatore al silicio.
Diversamente dai fotomoltiplicatori tradizionali (PMT o Photomultiplier Tubes), costruiti con tubi a vuoto, i SiPM sono prodotti direttamente da un wafer di silicioimpiantando in esso matrici di microcelle lette in parallelo ciascuna delle quali è un diodo (Avalanche Photodiode o APD) che lavora in modalità Geiger.
Le dimensioni tipiche di un SiPM sono da (1mm x 1mm) fino a (3mm x 3mm) ma in linea di principio sarebbe possibile produrre geometrie arbitrarie. Le singole microcelle hanno dimensioni tipiche che vanno dai 20x20 µm ai 50x50 µm.

Rispetto ai tradizionali PMT presentano numerosi vantaggi quali ad esempio la bassa tensione di funzionamento (da 30 a 80 V a seconda del modello e del costruttore).

Is it possible to build a complete particle detector and data acquisition system using Arduino microcontroller and Arduino Language ?

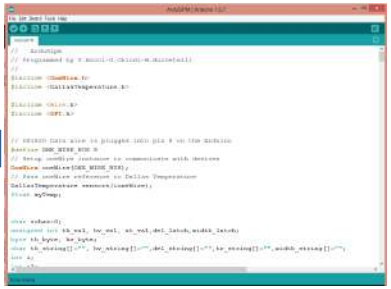


Scintillator

Photons Sensor (SiPM)

Custom Electronics (ArduSiPM Shield)

Arduino DUE

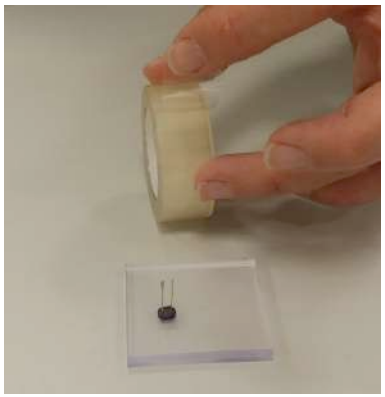
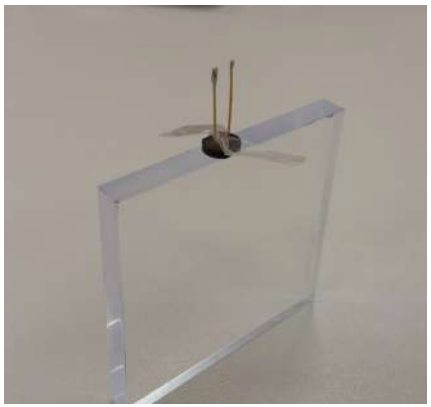
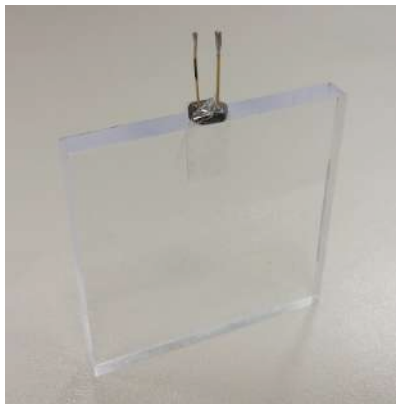


ArduSiPM Software

Particle Detector

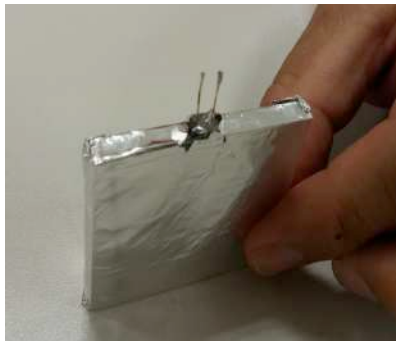
ArduSiPM

Dr. Valerio Bocci
17 Maggio 2017

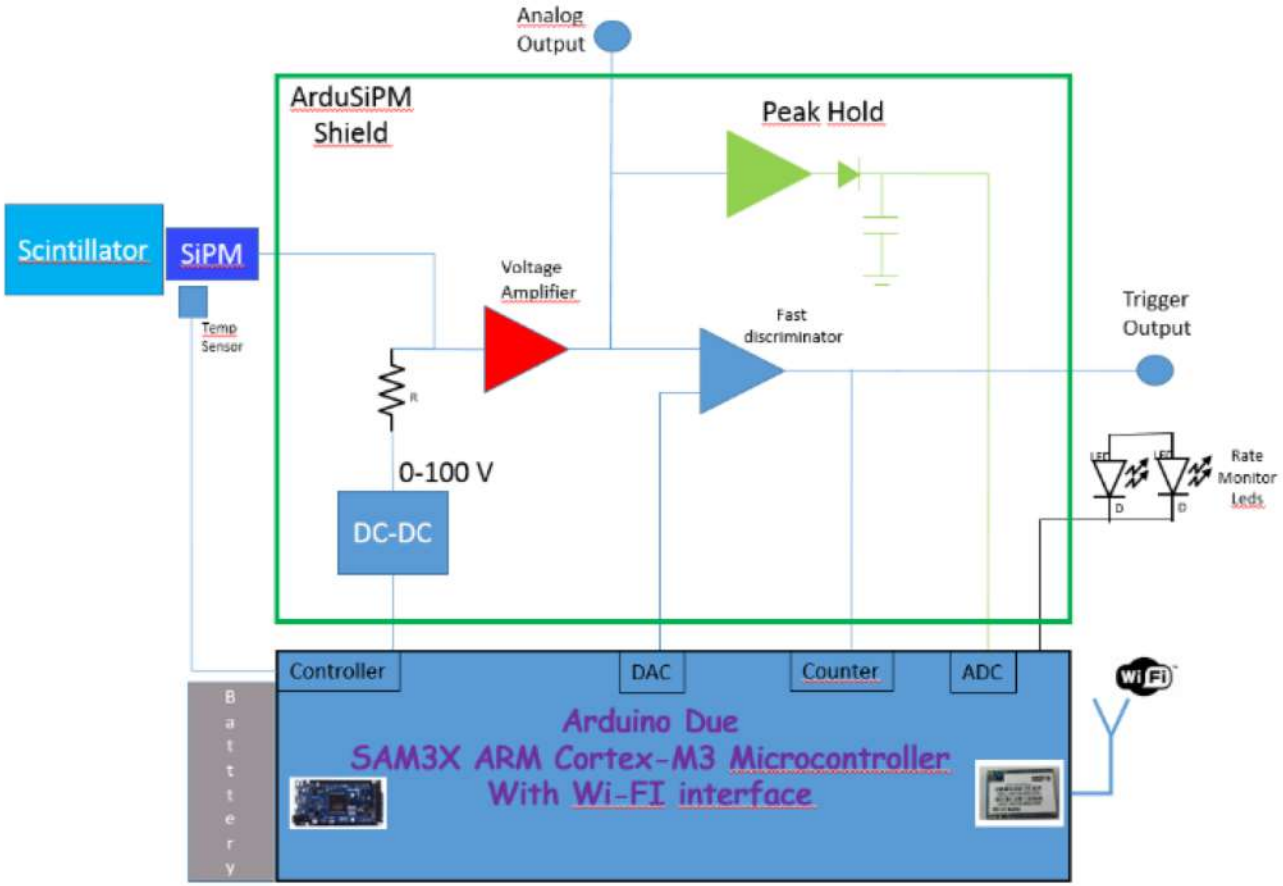
Attaccare il SiPM allo scintillatore usando dello scotch



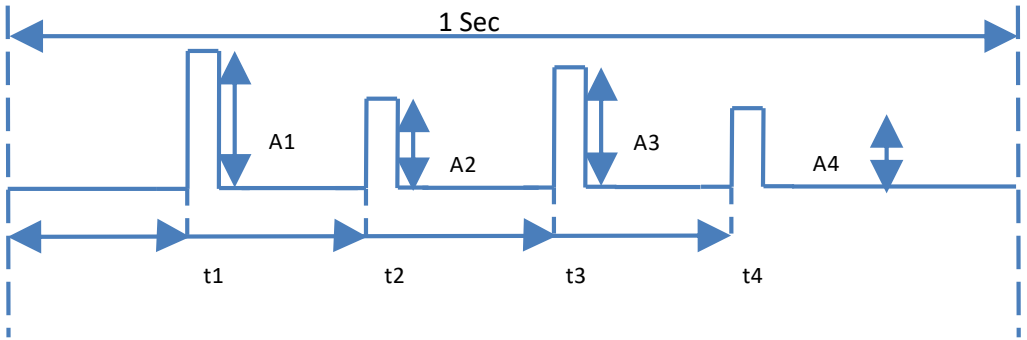


Ricoprirlo con una superficie riflettente (foglio di allumino da cucina)

ArduSiPM Block Diagram



ArduSiPM measuraments



We split the measuraments in 1 second windows,
acquiring number of pulses, amplitude and time of each one.

Using a 200KBits/s serial stream

Data Stream example:

```
Only rate:
$10
$50
$244

ADC+Rate:
v1Fv1Dv22v27v1Dv19v20v23v20v1Cv19v1F$12
v18v1Ev1Ev1Bv19v1Bv29v19v1Av1Dv1Bv1Dv2Av18v1B$15
v15v20v21v21v1Dv1Fv1Av1Av1A$9
v19v17v1Bv18v1Cv1Dv1D$7

TDC+ADC+RATE:
taedvataf0v7tv9v3$3
```

Legend:
vXXX ADC Value in HEX MSB zero suppressed
tXXXXXXXX TDC value in HEX MSB zero suppressed
$XXX rate in Hz

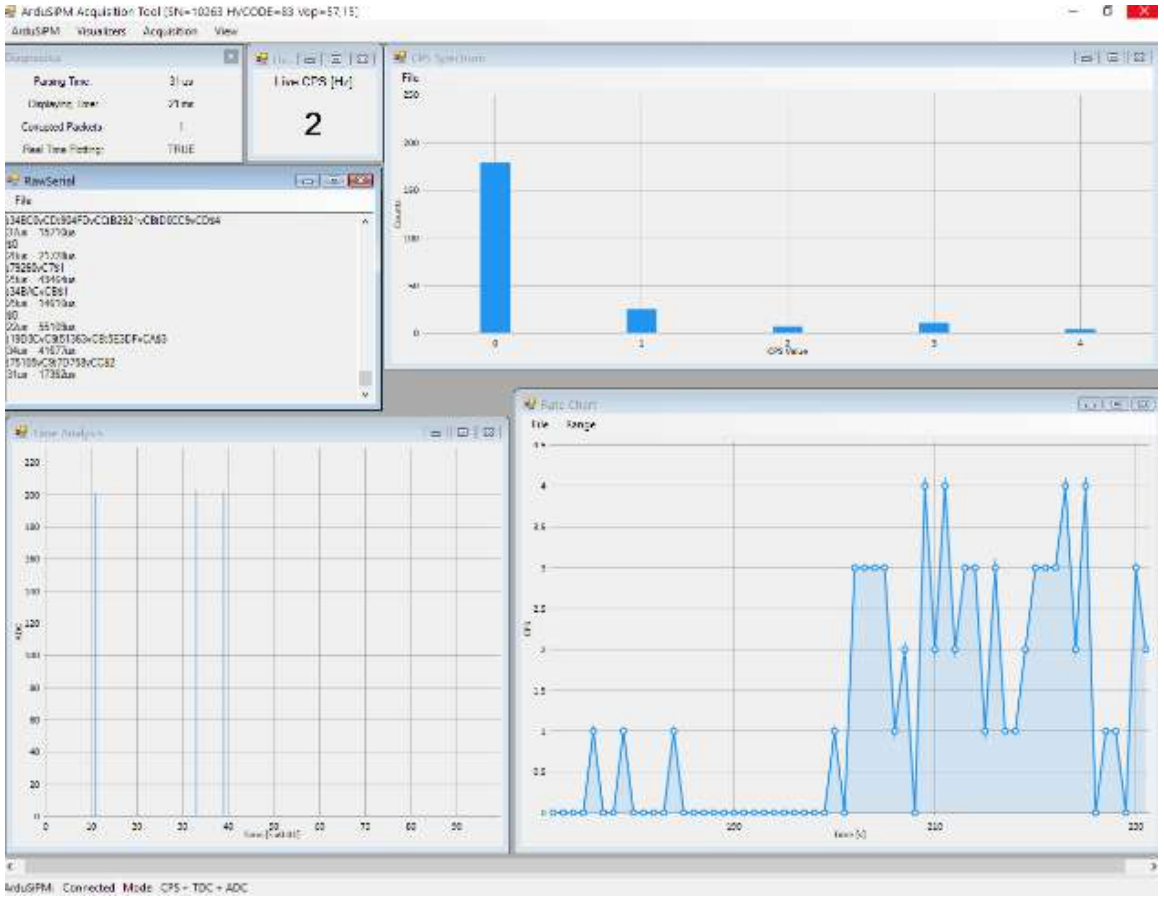
We can meausure and dump (depending from amplitude and distribution of pulses):

- Only the frequency up to 40 MHz
- ADC value up to 4-6 KHz
- ADC,TDC and rate 1 -2 KHz

Using the SAM3X8 built-in ethernet it is possible to increase data acquisition performance.

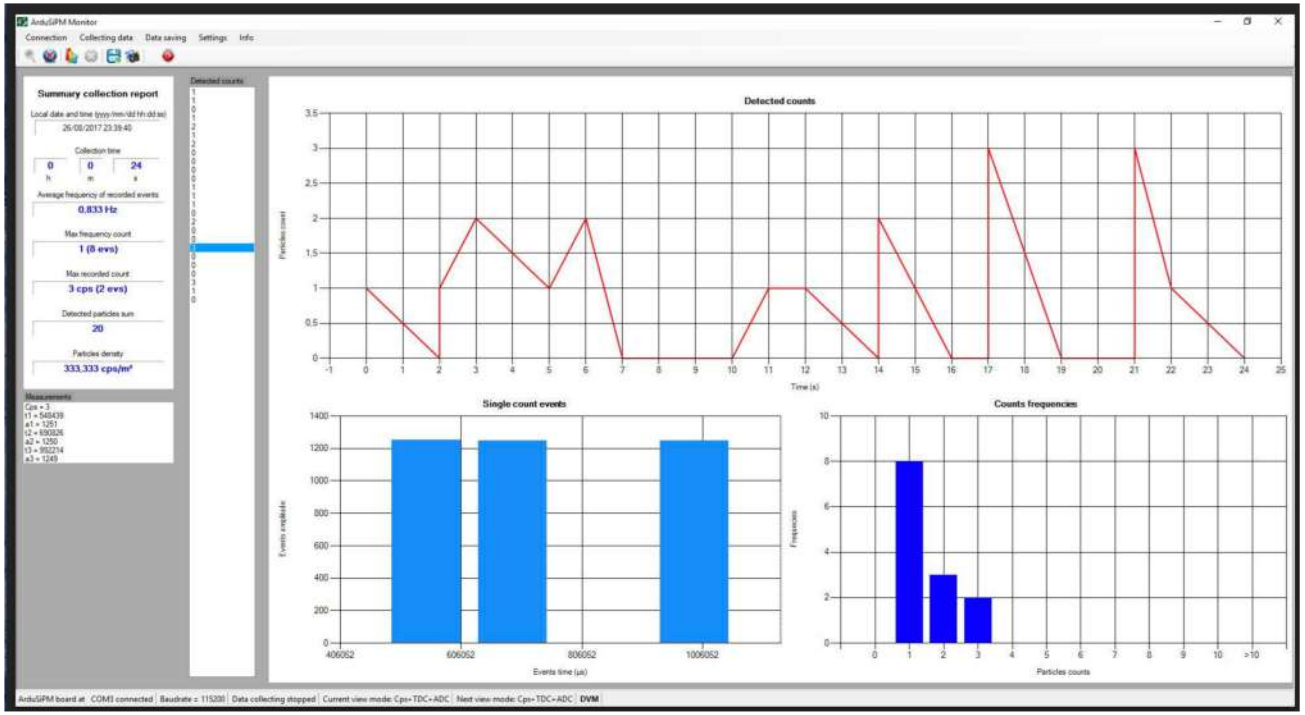
The ArduSiPM Data format is open…Users can write custom programs for data aquisition and visualization.

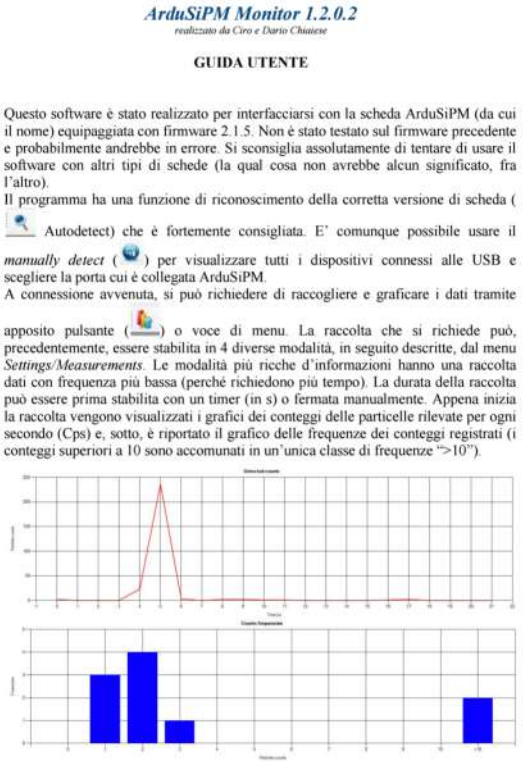
Example 1. **ArduSiPM Acquisition Tool** by Filippo Curti (Filippo.curti1@gmail.com)
(written in C#, fast running also with high rate, at the moment poor in documentation freeware)



Valerio Bocci 2017

Example 2. **ArduSiPM Monitor** by Ciro e Dario Chiaiese (cirochiaiese@gmail.com)
Written in VisualBasic (slow good for cosmic and low rate source) well documented.





ArduSiPM Monitor 1.2.0.2

realizzato da Ciro e Dario Chiaiese

GUIDA UTENTE

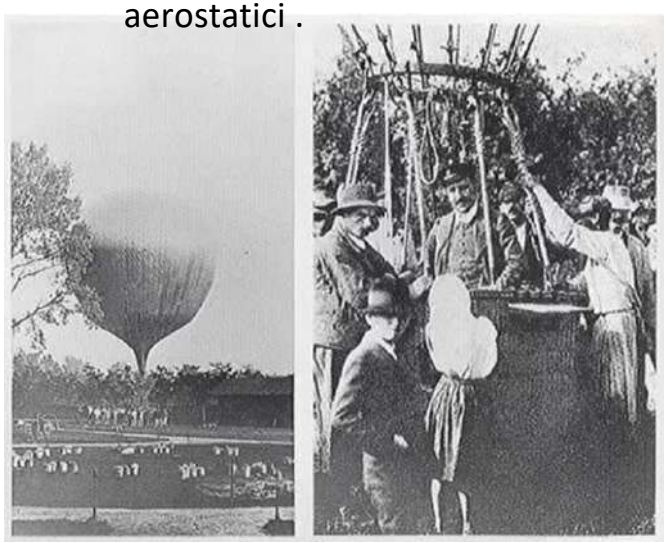
Questo software è stato realizzato per interfacciarsi con la scheda ArduSiPM (da cui il nome) equipaggiata con firmware 2.1.5. Non è stato testato sul firmware precedente e probabilmente andrebbe in errore. Si sconsiglia assolutamente di tentare di usare il software con altri tipi di schede (la qual cosa non avrebbe alcun significato, fra l'altro).

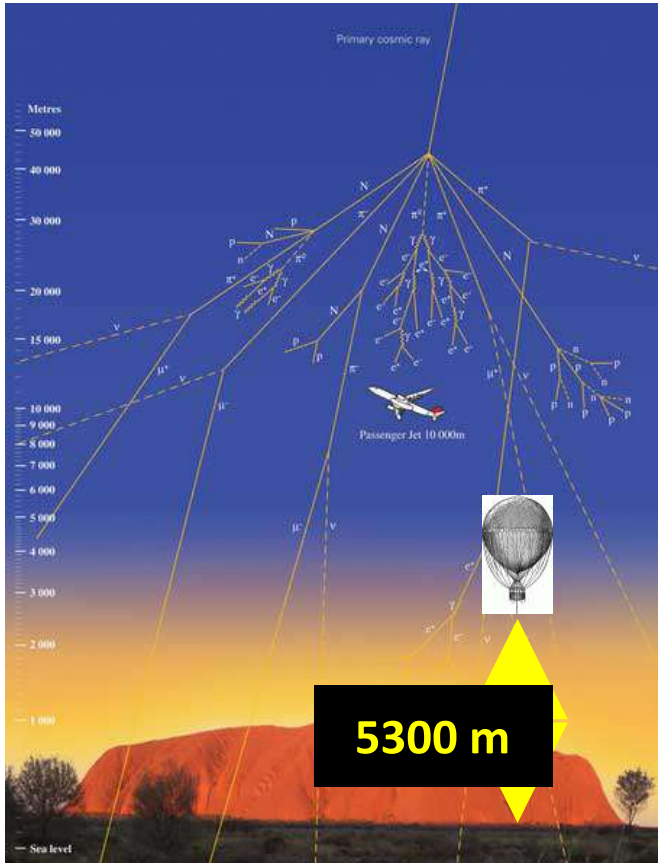
Il programma ha una funzione di riconoscimento della corretta versione di scheda (Autodetect) che è fortemente consigliata. E' comunque possibile usare il *manually detect* ( ) per visualizzare tutti i dispositivi connessi alle USB e scegliere la porta cui è collegata ArduSiPM.

A connessione avvenuta, si può richiedere di raccogliere e graficare i dati tramite apposito pulsante ( ) o voce di menu. La raccolta che si richiede può, precedentemente, essere stabilita in 4 diverse modalità, in seguito descritte, dal menu *Settings/Measurements*. Le modalità più ricche d'informazioni hanno una raccolta dati con frequenza più bassa (perché richiedono più tempo). La durata della raccolta può essere prima stabilita con un timer (in s) o fermata manualmente. Appena inizia la raccolta vengono visualizzati i grafici dei conteggi delle particelle rilevate per ogni secondo (Cps) e, sotto, è riportato il grafico delle frequenze dei conteggi registrati (i conteggi superiori a 10 sono accomunati in un'unica classe di frequenze ">10").

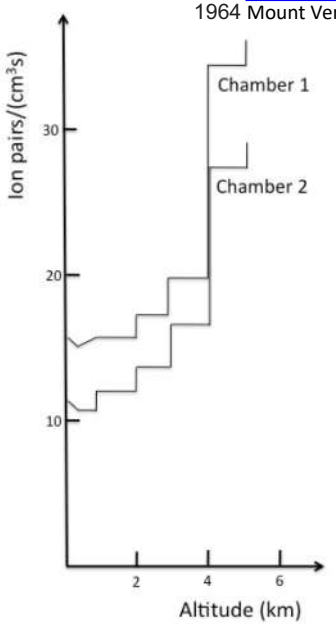
Gli esperimenti in quota di Victor Hess

Tra il 1911 e il 1912 l'Austriaco Victor Hesse misurò la radiazione penetrante in quota utilizzando dei palloni aerostatici .

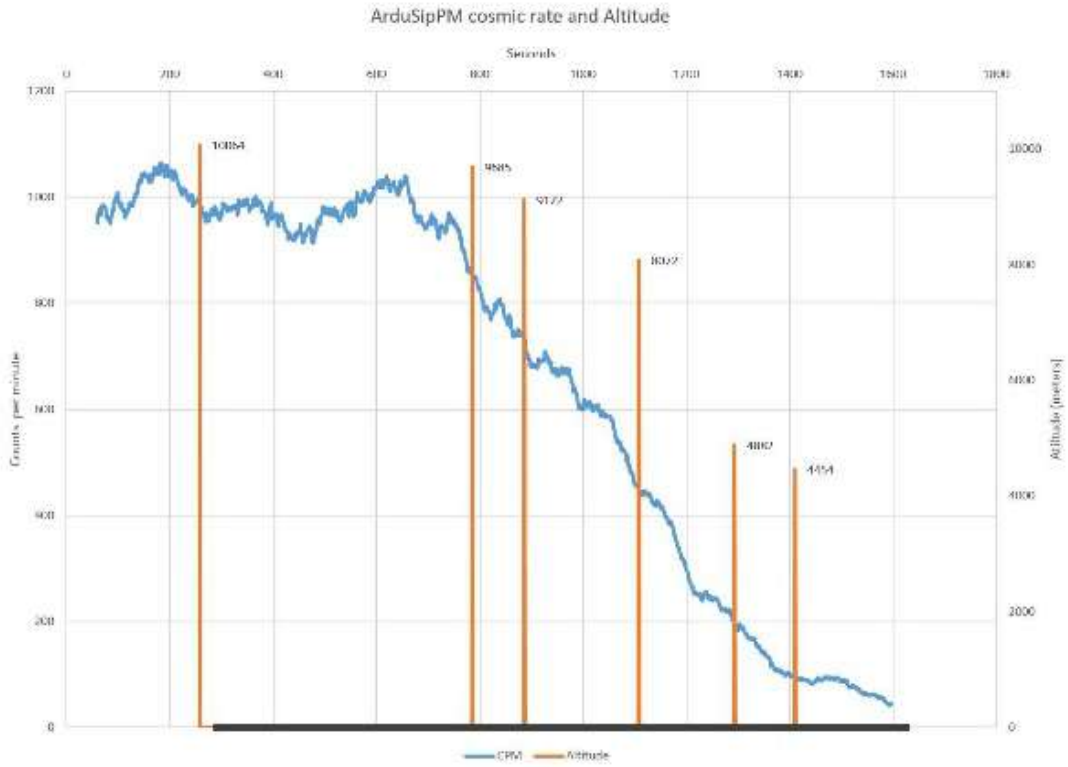




Victor Franz Hess
1883 Schloss Waldstein, Peggau
1964 Mount Vernon, New York



Misura dei raggi cosmici su un aereo di linea

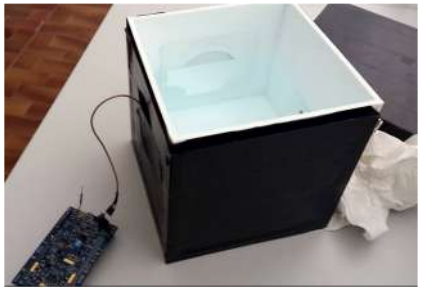


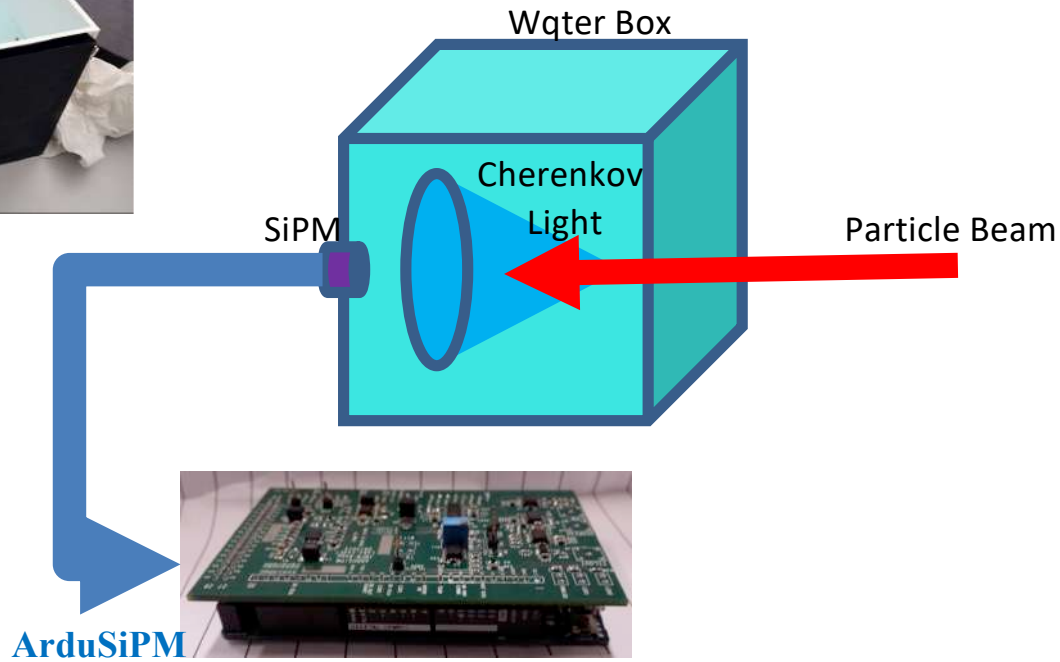
Valerio Bocci International Cosmic Day 2017

A School made Cherenkov light detector

(Winner of CERN "A beamline for schools"2017)

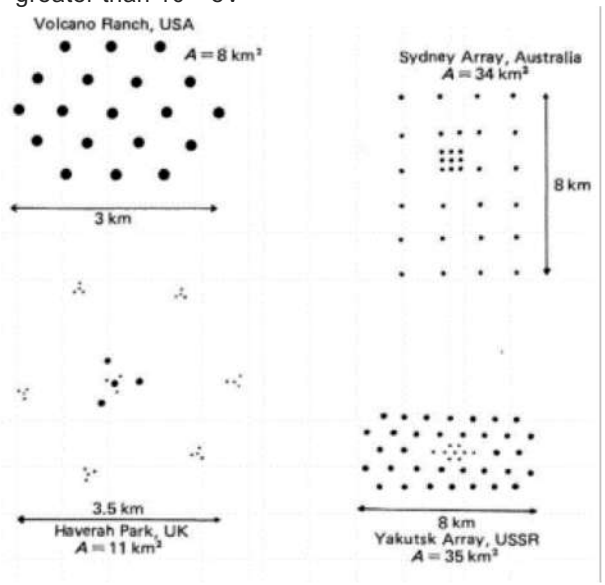
LICEO SCIENTIFICO STATALE T. C. ONESTI (prof Maria Rita Felici)



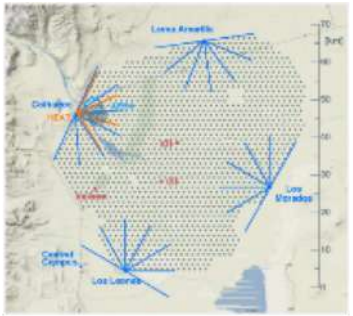


The search of Ultra Energetic Cosmic Ray
E> 10^{19} eV

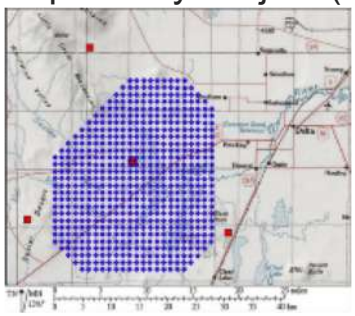
On February 22, 1962, John David Linsley observed an air shower at Volcano Ranch created by a primary particle with an energy greater than 10^{20} eV

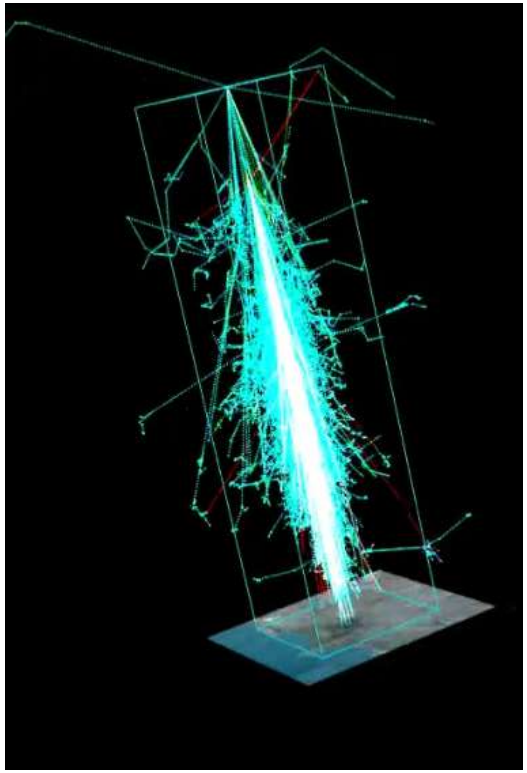


Pierre Auger Observatory (Argentina)



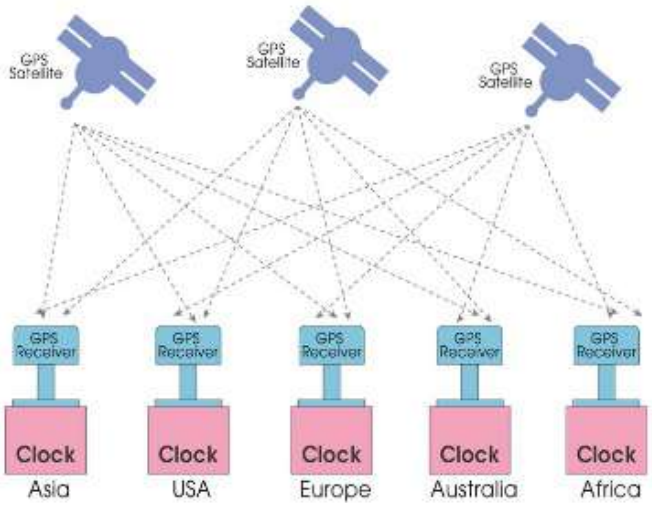
Telescope Array Project (Utah)





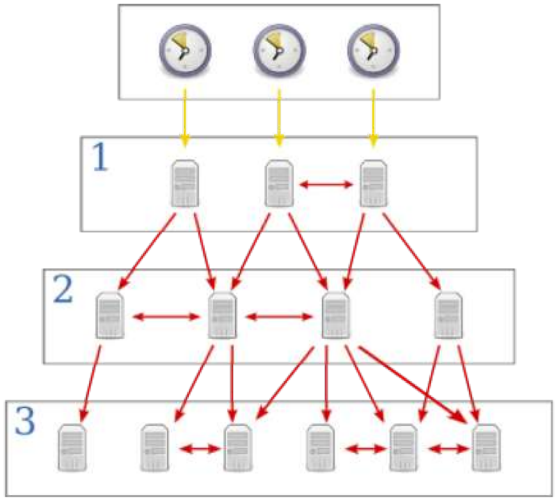
Precise time distribution methods

GPS time (from Global Position Systems satellite)



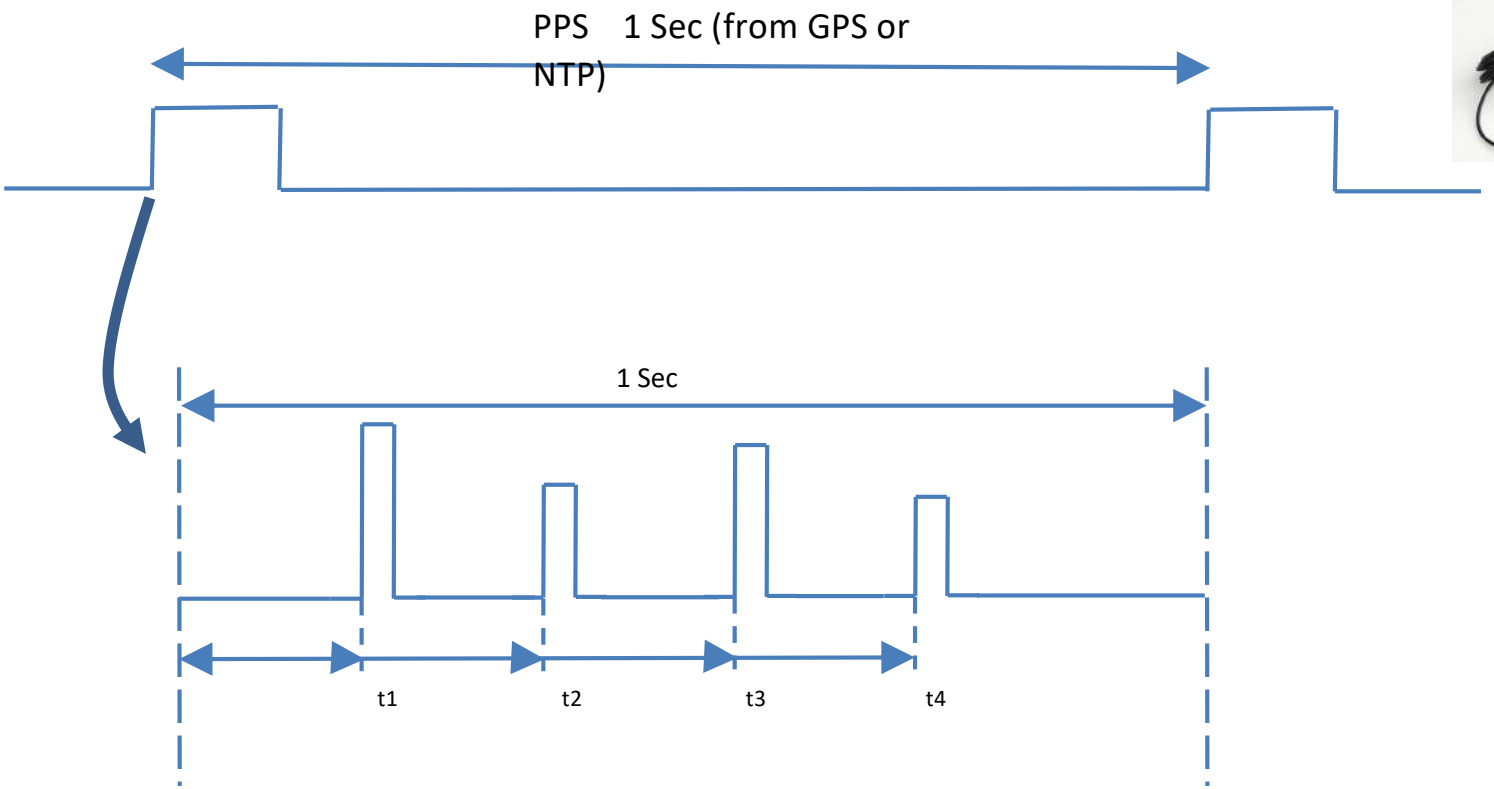
Low cost GPS module (<30 Euro) 25 ns precision

Network Time Protocol (from internet)



Low cost wi-fi internet processor precision <10 ms

Synchronization with ArduSiPM

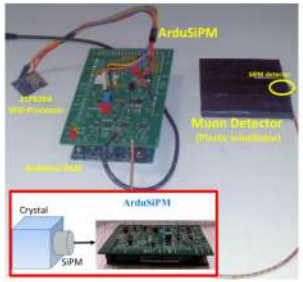


An educational distributed Cosmic Ray detector network based on ArduSiPM using microcontrollers as data acquisition node NTP protocol as time distribution and IoT technology for data aggregation.

Valerio Bocci , Giacomo Chiodi, Paolo Fresch, Francesco Iacoangeli, Luigi Recchia

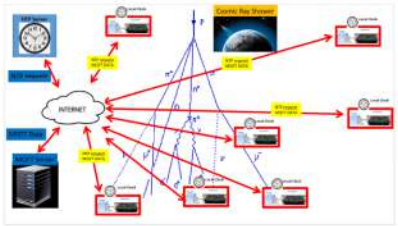
INFN Roma, Piazzale A.Moro, 2 - 00185 Roma
valerio.bocci@roma1.infn.it

The advent of microcontrollers with enough CPU power and with analog and digital peripherals give the possibility to design a complete acquisition system in one chip. The existence of an world wide data infrastructure as internet allows to think at distributed network of detectors capable to elaborate and send data or respond to settings commands. The internet infrastructure allow us to do things unthinkable a few years ago, like to distribute the absolute time with tens of milliseconds precision to simple devices far apart from a few meters to thousands of kilometers and to create a Crowdsourcing experiment platform using simple detectors.

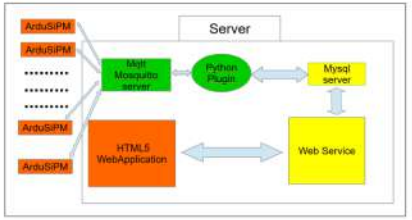


The ArduSiPM (1) is an easy hand-held battery operated data acquisition system based with an Arduino board, which is used to detect cosmic rays and nuclear radiation. The ArduSiPM uses an Arduino DUE (an open Software/Hardware board based on an ARM Cortex-M3 microcontroller) as processor board and a piggyback custom designed board (Shield), these are controlled by custom developed software and interface. The Shield contains different electronics features both to monitor, to set and to acquire the SiPM signal using the microcontroller board. The SiPM photon counting detector can be coupled to a cheap plastic scintillator to realize a cosmic ray detector (mainly muon particles). An ArduSiPM channel give informations about rate of events, arrival time and number of photons produced by muons, it contains all the feature from controls to data acquisition typical of High Energy Physics channel at a cost affordable for single user or school.

(1) **The ArduSiPM a compact trasportable Software/Hardware Data Acquisition system for SiPM detector** V. Bocci et al. IEEE NSS/MIC 2014 Pages: 1 - 5, DOI: 10.1109/NSSMIC.2014.7431252 arXiv:1411.7814

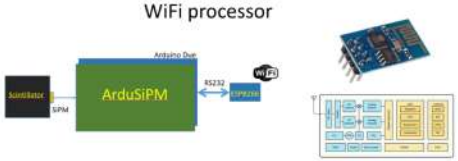


Using Network Time Protocol (NTP) the absolute time from the network, with a precision of tens milliseconds. The network time can be used from a cloud of ArduSiPMs to detect offline coincidence events linked to Ultra High Energy Cosmic Ray



The terms of IoT (Internet of Things) define a set of data communication protocols and the capability of single embedded electronics objects to communicate using the internet .
The MQTT (Message Queue Telemetry Transport) is one of the main protocol used in IoT device for data transmission over TCP/IP, the client version can run easily in nowadays microcontrollers, the MQTT broker (the server version) can run also in credit card–sized single-board computers as well in big server.

ESP8266 MQTT, NTP and WiFi processor



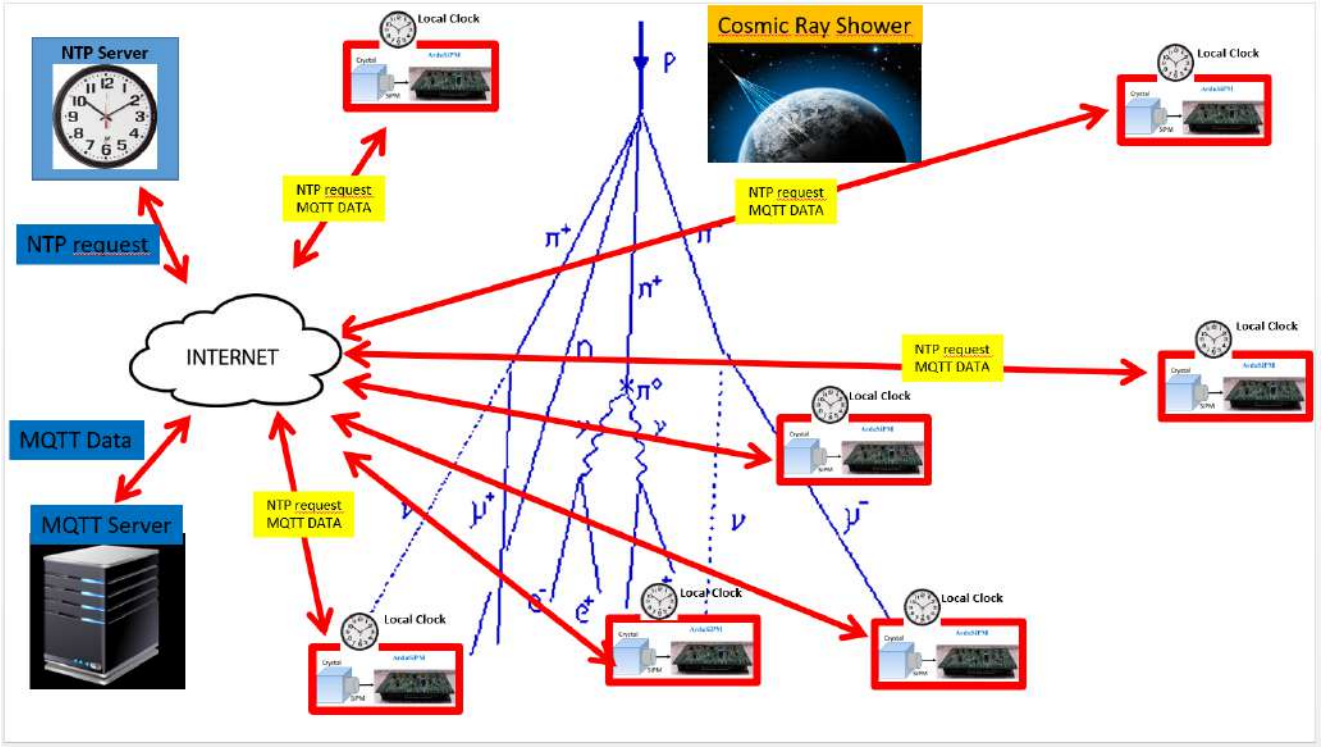
The ArduSiPM sends data over rs232, the Wifi Processor elaborate the data and send them using the MQTT protocol to the server. We use as network processor the Espressif ESP8266 a low-cost Wi-Fi chip with full TCP/IP stack and a 32-bit RISC CPU running at 80 MHz. The ESP8266 can be used to send and configure MQTT packets,NTP request and configure ArduSiPM device.

ArduSiPM web configuration pages.

Search of cosmic Airshower in a wide area using ArduSiPM

Multiple ArduSiPM can be used for the research extended AirShower

The advent of microcontrollers with enough CPU power and with analog and digital peripherals give the possibility to design a complete acquisition system in one chip. The existence of an world wide data infrastructure as internet allows to think at distributed network of detectors capable to elaborate and send data or respond to settings commands.
The internet infrastructure allow us to do things unthinkable a few years ago, like to distribute the absolute time with tens of milliseconds precision to simple devices far apart from a few meters to thousands of kilometers and to create a Crowdsourcing experiment platform using simple detectors.



Dr. Valerio Bocci INFN Roma
Matinees di scienza 21 Apr 2017 LNF