Ordine degli Ingegneri di Pisa
T.A.G. - Talent Garden Pisa
via Umberto Forti 6, Montacchiello (PI)
23 novembre 2017 – 14:00 – 19:00

IISFA European Chapter

Introduzione alla programmazione IoT
Come funzionano e come si progettano oggetti "smart"

Marco A. Calamari
marco.calamari@ordineingegneripisa.it

IISFA – International Information Systems Forensics Association: Italian Chapter

Il vostro anfitrione

https://www.linkedin.com/in/marcocalamari/

- **Marco Calamari, classe 1955, ingegnere nucleare, si cimenta a rotazione tra attivita' di consulenza tecnica informatica, Computer Forensics, editoriali e di formazione.**
- **Membro di: IISFA, AIP, Opsi, Hermes Center, PWS**
- **Appassionato di privacy e crittografia, ha contribuito ai progetti FOSS Freenet, Mixmaster, Mixminion, Tor e Globaleaks.**
- **Fondatore del *Progetto Winston Smith* e del *Centro Hermes per la Trasparenza ed i diritti digitali*.**
- **Dal 2003 scrive su Punto Informatico ed altre riviste la rubrica "Cassandra Crossing", che ha superato le 400 uscite. (www.cassandracrossing.org)**

Di cosa parleremo

Smartphone, tablet, laptop, automobili, lavatrici, ferri da stiro, tutti questi oggetti possiedono una singola funzione, che usiamo quando vogliamo per fare cio' che vogliamo.

Per questo noi percepiamo l'oggetto come "semplice" perche' lo identifichiamo con la singola funzione che svolge.

Tutto questo e' cambiato da quando questi "oggetti" hanno iniziato a contenere software "embedded"..

Contenere software e' una caratteristica essenziale affinche' un oggetto sia parte dell'IoT, ed e' la fonte di tutte le opportunita' che l'IoT offre. E' anche la causa di tutti i problemi e le preoccupazioni che l'avvento dell'IoT ha causato.

Il tema del corso e' "comprendere per sviluppare".

Di cosa parleremo

Prima parte
Breve storia dell'IoT e delle tecnologia hardware e software che la popolano

Seconda parte
Il software dell'IoT; caratteristiche, tipologie di software e protocolli di comunicazione

Terza Parte
Schede di sviluppo e risorse per la realizzazione di un progetto IoT
Full IoT stack service providers: Particle.io

Quarta parte
Un esempio di sviluppo: controllare un carico elettrico casalingo da Internet

Storia dell'IoT e delle sue tecnologie

Eventi importanti

1991: Mark Weiser pubblica su "Scientific American" l'articolo *Il computer per il 21mo Secolo* che descrive, pur senza dargli un nome, l'avvento del "Disappearing Computer" - il computer che scompare.

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

1998: sempre Mark Weiser costruisce una fontana davanti il suo ufficio il cui getto rappresenta l'andamento del mercato azionario.

Ma fatto ben piu' importante, quello stesso anno introduce la definizione mancante e che ci guidera'

"*Ubiquitous computing*

Eventi importanti - 2


Credit: RFID Journal

1999 – il termine "Internet of Things" - Internet delle Cose viene (sembra) coniato da Kevin Ashton, direttore esecutivo di "MIT *Auto-ID Center*":

"I could be wrong, but I'm fairly sure the phrase "Internet of Things" started life as the title of a presentation I made at Procter & Gamble (P&G) in 1999. Linking the new idea of RFID in P&G's supply chain to the then-red-hot topic of the Internet was more than just a good way to get executive attention. It summed up an important insight which is stil often misunderstood."

Eventi importanti - 3

2002:

The Ambient Orb, viene creato da David Rose ed altri in una spin-off del MIT Media Lab, e lanciato sul mercato.

Orb visualizza l'indice Dow Jones, cambiando il colore e la dinamica della sua luce.

Incredibilmente Amazon lo lista ancora oggi, ma nessuno pare averlo mai comprato.

Non ha quindi inciso piu' di tanto sul mondo reale.

Eventi importanti -Նարեկ

2005: nasce Nabaztag

E’ la traslitterazione (non la Traduzione) del termine armeno

Lepre - “Նարեկ".

Concepito da Rafi Haladjian e Olivier Mével, e prodotto in oltre 100,000 esemplari dall’azienda francese Violet; Violet dopo poco falli’, vittima del suo stesso successo, e venne acquistata da Mindscape. Successivamente anche Mindscape fu ceduta, ed i suoi asset (hw & sw di Nabaztag), furono acquistati da Aldebaran Robotics e divennero abandonware.

Il mio coniglio (anzi i miei 6) sono la sola cosa computerizzata che in 30 anni di convivenza mi abbia fatto fare bella figura con la mia signora.

Նախագիծ

Eventi importanti - 5

Nabaztag ha un bottone sulla testa, due orecchie con fissaggio magnetico mosse da motori passo-passo e dotati di encoder per leggerne la posizione via Software, 5 LED RGB, un lettore RFID, una scheda audio con microfono ed altoparlante ed una scheda WiFi. E' controllato da un server remoto, originariamente programmato in Ruby on Rails, su cui ogni proprietario di Nabaztag deve caricare plugin e "programmarli".

Nabaztag puo' muovere orecchie e fare coreografie con i LED, puo' leggere oroscopi, previsioni del tempo ed indici azionari dal web e dirvi l'ora, anche con battute spiritose.

E' possibile "sposare" due conigli cosicche' se si muovono le orecchie ad dei due uno, l'altro canta, lampeggia e sposta le orecchie nella stessa posizione.

Eventi importanti - 6

2015: Amazon Echo / Alexa

Anonimo come il monolite di "2001: Odissea nello spazio", costa ai suoi acquirenti circa 150 euro e permette di ospitare un'intelligenza artificiale controllata da Amazon in casa, mettendola quindi in grado di ascoltare, con 6 orecchi sopraffini, tutto quello che viene detto.

Intelligenza Artificiale e Internet delle Cose si fondono nel vostro soggiorno allo scopo di vendervi piu' cose possibile, ma anche di carpirvi piu' dati personali possibile, e quest'ultima cosa passa largamente inosservata...

Problematiche dell'IoT

I 5 eventi che abbiamo visto, tutti avvenuti nel non breve periodo dal 1991 al 2015, sono abbastanza per raccontarci la storia dell'IoT.

In effetti l'IoT e' meglio descritta non da una "storia" ma dalle opportunita' che offre e dai problemi che causa.

La piu' importante opportunita' dell'IoT e' data senz'altro dall'essere formata da oggetti complessi e potenti, contenenti sensori, potenza di calcolo e molto software, che rendono non percepibile la propria complessita' interna.

Ma non sorprendentemente, i problemi derivano appunto dall'essere formata di oggetti complessi e potenti, contenenti sensori, potenza di calcolo e molto software, che rendono non percepibile la propria complessita' interna.

Il software dell'IoT

Il software dell'IoT

Che gli oggetti IoT siano definiti dal loro software "embedded" e' evidente.

Meno evidente forse e' che ogni oggetto visibile dell'IoT e' (molto) meno della meta' di quello del super-oggetto di cui fa parte.

L'altra meta' si trova "nel cloud".

"Nel cloud" normalmente vuol dire "nel computer di qualcun'altro"; comprare l'oggetto, non possederlo.

In questo caso invece parliamo sempre di software sviluppato dal fabbricante; doppio lavoro quindi!

Complessivamente il software lato server di un oggetto IoT e' costituito da un protocollo di comunicazione e da un server che contiene la parte applicativa ma soprattutto la parte di gestione dei dati della "flotta" di oggetti che supporta.

Il software in oggetti “storici”

- **Nel 1969 siamo andati sulla Luna con meno di 10.000 linee di software, e per lo Shuttle negli anni ‘90 ne sono bastate 400.000.**
- **Un pacemaker ci salva la vita con 100.000 linee, tante quante ne aveva Photoshop 1.0, che pero’ oggi e’ cresciuto a 3.500.000.**
- **Nel 1971 la prima versione di Unix aveva 10.000 linee, mentre Debian 5.0 (Lenny) nel 2009 ne aveva 65.000.000 (incluse pero’ le applicazioni).**
- **Nel 1991 Windows 3.1 contava 2.000.000 di linee, nel 2001 Windows XP 43.000.000**
- **Un “vecchio” caccia supersonico F22 “Raptor” si contentava di 2.000.000**

Il software in oggetti “storici” - 2

- **Il modernissimo caccia F35 conta 45.000.000 di linee di codice**
- **Il rover "Curiosity" ha esplorato Marte con “solo” 5,000,000 di liee di programma.**
- **Il Large Hadron Collider, il piu’ grande strumento mai costruito dall’uomo, per trovare il bosone di Higgs ha avuto bisogno di 50,000,000 di linee.**
- **Ma una moderna autovettura di fascia alta contiene certamente piu’ di 100,000,000 (dicasi centomilioni) di linee di codice.**
- **Per fare un confronto, si consideri che il DNA di un topolino puo’ essere descritto con “solo” 120,000,000 di "linee di codice".**

Componenti di un oggetto IoT

Un generico oggetto IoT e' formato da:

- **Hardware meccanico ed involucro**
- **Hardware analogico (alimentazione, encoder...)**
- **Hardware digitale (CPU, DAC...)**
- **Hardware di comunicazione**
- **Sensori**
- **Attuatori**
- **Software server e protocollo di comunicazione**
- **Software client**

"Nabaztag ha un bottone sulla testa, due orecchie con fissaggio magnetico mosse da motori passo-passo e dotati di encoder per leggerne la posizione via software, 5 LED RGB, un lettore RFID, una scheda audio con microfono ed altoparlante,una scheda WiFi."

Nabaztag: il software Client

Nabaztag:tag (v2)

Il firmware e' memorizzato in 128 KB di flash ROM "sicura" ed in 8KB di Boot Flash ROM.

Il firmware e' crosscompilato esternamente in un blob binario.

Contiene una macchina virtuale capace di eseguire fino a 64 kB di bytecode.

Esiste un linguaggio pseudo-Assembler dedicato per la programmazione di movimenti, azioni e coreografie.

Coreografie & plugin possono essere anche scaricati dal server ed eseguiti nella VM.

Nabaztag: il software Server

Nabaztag:tag (v2)

Dopo la scomparsa di Violet e Mindscape, parecchie comunita' spontanee iniziarono a sviluppare cloni del software lato server e dei relativi plugin.

Per il Nabaztag:tag (v2) i software piu' diffusi sono OpenNab, scritto in PHP, and openJabNab, scritto in PHP & C++.

Ad oggi oltre una dozzina di questi server sono in funzione, e tengono desta una buona parte della popolazione dei conigli "smart", come il server italiano OpenZNab.it
Sono manutenuti da comunita' spontanee, che spesso gestiscono anche maillist e forum informativi.

Nabaztag: protocollo di rete

Nabaztag:tag (v2) + OpenJabNab

A causa della mancanza di documentazione, il protocollo di rete e' stato in parte sniffato e sottoposto a reverse-engineering.

Si e' verificato che tutte le comunicazioni usano il protocollo XMPP Jabber, con crittografia TLS.
Tuttavia quando un blob binario, ad esempio un file MP3, deveessere trasferito, questo viene fatto encodando l'oggetto in Base64, e poi trasferendolo con HTTP in chiaro.

Inoltre, per la scarsa potenza di calcolo lato client, quando un messaggio di testo deve essere letto dal coniglio, viene prima inviato al server che lo rasterizza in un file MP3, poi trasferito fuori dalla sessione XMPP usando HTTP in chiaro.

Hardware meccanico

Un oggetto IoT possiede una struttura ed un aspetto. Queste caratteristiche sono realizzate tramite:

- **una struttura meccanica, che assembla sensori attuatori, parte elettronica e parte elettrica**
- **un involucro ad essa fissato che fornisce l'aspetto desiderato all'oggetto**

I software liberi di CAD tridimensionale e le tecniche di stampa 3D, ormai economicissime, forniscono grandi possibilita' di prototipazione a costi molto ridotti rispetto al recente passato.

Hardware analogico

Un oggetto IoT contiene sia una parte elettrica che una parte digitale, di solito (ma non sempre) realizzate su due schede PCB separate.

La parte analogica di solito raggruppa al massimo possible l'alimentazione elettrica, la sensoristica, gli attuatori, l'eventuale raffreddamento; talvolta e' addirittura un componente della struttura meccanica stessa.

Ove possibile puo' essere realizzata sulla scheda digitale.

Attenzione alla normativa di sicurezza ed alle eventuali, anzi certe, necessita' di certificazioni.

Hardware digitale

Un oggetto IoT contiene certamente una CPU.

In passato si trattava spesso di PLC, microcontrollori programmabili di basse prestazioni e scarsa memoria (Ram/Programma/Rom).

La disponibilita' a prezzi bassi di computer su singola scheda potenti dotati di I/O programmabili, DAC, Ethernet, Bluetooth e WiFi (Raspberry Zero, Raspberry PI3, Cubieboard, Premoboard) permette nella maggioranza dei casi di utilizzarli direttamente come scheda digitale di un oggetto IoT.

Aziende orientate all'IoT mettono a disposizione schede di sviluppo che integrano in maniera piu' specializzata questi componenti, aggiungendo elettronica di potenza, interfacce 3/4G e funzionalita' di standby a bassissimo assorbimento.

Hardware di comunicazione

Un oggetto IoT contiene certamente uno o piu' canali di comunicazione in rete, tipicamente di tipo wireless (Bluetooth, BLE, WIFi, GSM 3/4G, Zigbee).

Nella maggioranza dei casi il canale od i canali necessari sono gia' disponibili o comunque integrati nella scheda digitale, che spesso proprio in base a questa necessita' viene selezionata tra i tanti modelli sul mercato.

Una grossa preoccupazione in meno!

Sensori

Un oggetto IoT contiene uno, ma spesso molti, sensori, anche di tipi diversi, con cui monitora l'ambiente che lo circonda

Questi sensori generano flussi di dati che vengono utilizzati sia internamente all'oggetto IoT, che scambiati con il server che anima e/o controlla e/o monitora la flotta.

Contatti, sonde di temperatura, accelerometri, girobussole, bussole, sensori infrarossi, sensori ad ultrasuoni, sonde di vibrazione, GPS, microfoni, telecamere, RFID, NFC, lettori SIM, cardiofrequenzimetri, rivelatori di particelle, rivelatori di radiazioni . . . . .

Attuatori

Un oggetto IoT contiene attuatori, con cui agisce sull'ambiente che lo circonda, e che sono spesso alimentati da una scheda analogica di "potenza".

Motori passo-passo, motori in CC, LED, LED RGB, matrici di LED, altoparlanti, schermi LCD, laser ...

Software server

Non e' evidente, e questo spesso e' un problema per l'utente finale, ma ogni oggetto IoT funziona relazionandosi con un server applicativo via connessione di rete; il server gli fornisce una parte di intelligenza applicativa e gestisce e memorizza i dati da lui prodotti.

E' normalmente anche piu' complesso della parte client, anche perche' deve gestire non solo un oggetto ma un'intera flotta di oggetti

Fa questo utilizzando un protocollo applicativo di comunicazione, spesso derivato da protocolli standard quali HTTP, XMPP, REST ...

Scambia dati descritti in formati standard, ASCII, XML, SOAP ...

Software client

E' la parte che, insieme all'involucro, piu' definisce un particolare oggetto IoT.

Se sviluppato da zero e' spesso molto complesso, ma oggi sono disponibili, anche con licenza libera, librerie molto potenti per gestire le piu' svariate necessita' degli oggetti IoT.

Negli oggetti piu' semplici, dotati di PLC, e' spesso implementato come applicazione monolitica (firmware) senza il supporto di un sistema operativo; con l'introduzione di schede digitali come Raspberry, Cubieboard e simili, che girano GNU/Linux od Android nativamente ed hanno alta potenza di calcolo, gli oggetti che contengono un intero sistema operativo ed una vera applicazione sono diventati molto comuni.

Si utilizzano spesso IDE standard (Arduino, VB, etc.).

Ambienti di sviluppo per l'IoT

Il sofware libero ed Open Source

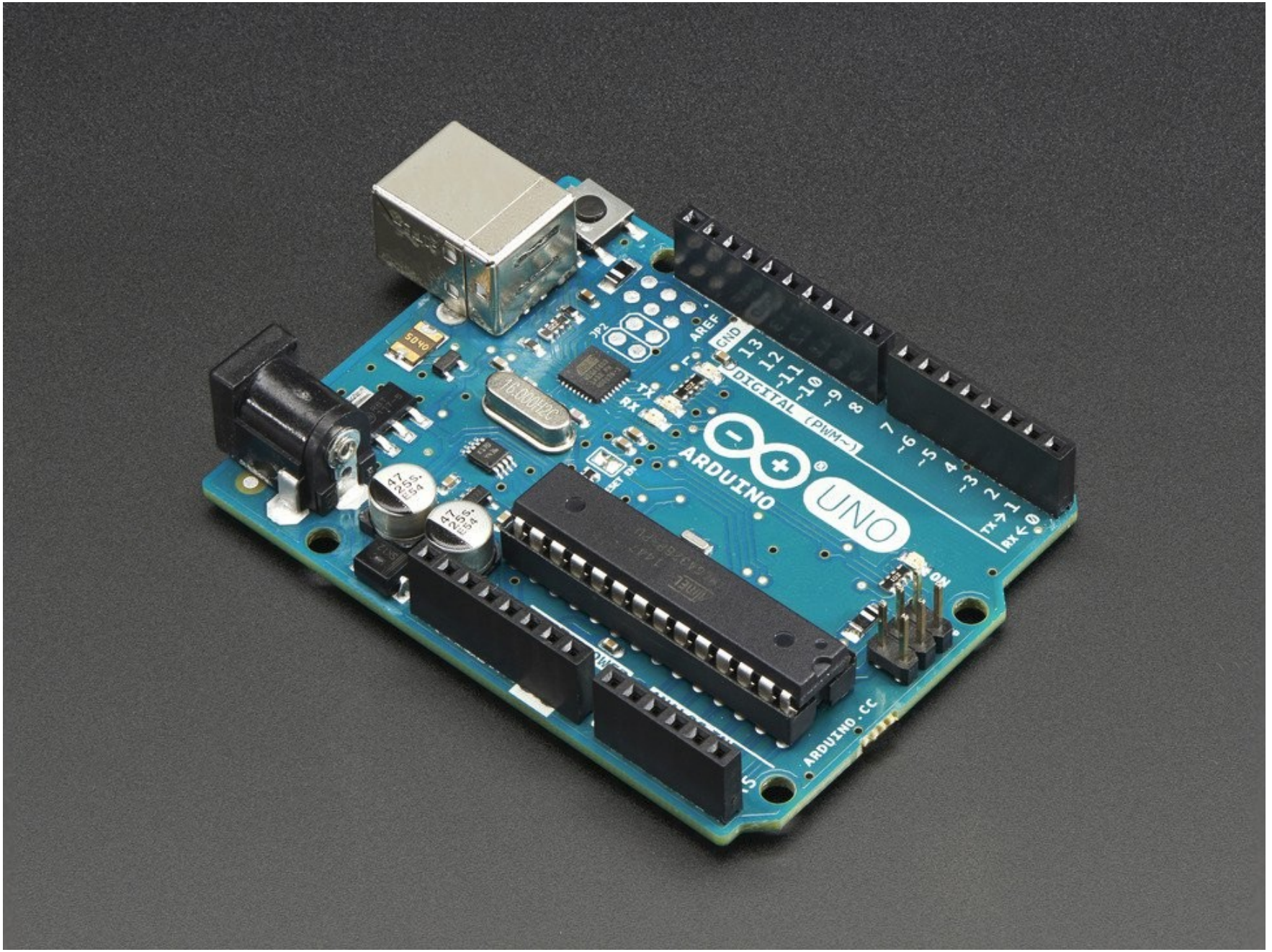
Come per il fenomeno Linux ed altre rivoluzioni, se non tecnologiche almeno di "mercato", il software libero ed open source (FOSS) ha cambiato il "mondo" tecnologico anche nel settore dei firmware e dei microcontroller.

Ha permesso a un grande numero di non addetti ai lavori di cimentarsi pari a pari con gli esperti, ha innescato i circoli virtuosi che producono librerie, software e programmi gratutuiti ed esplorabili.

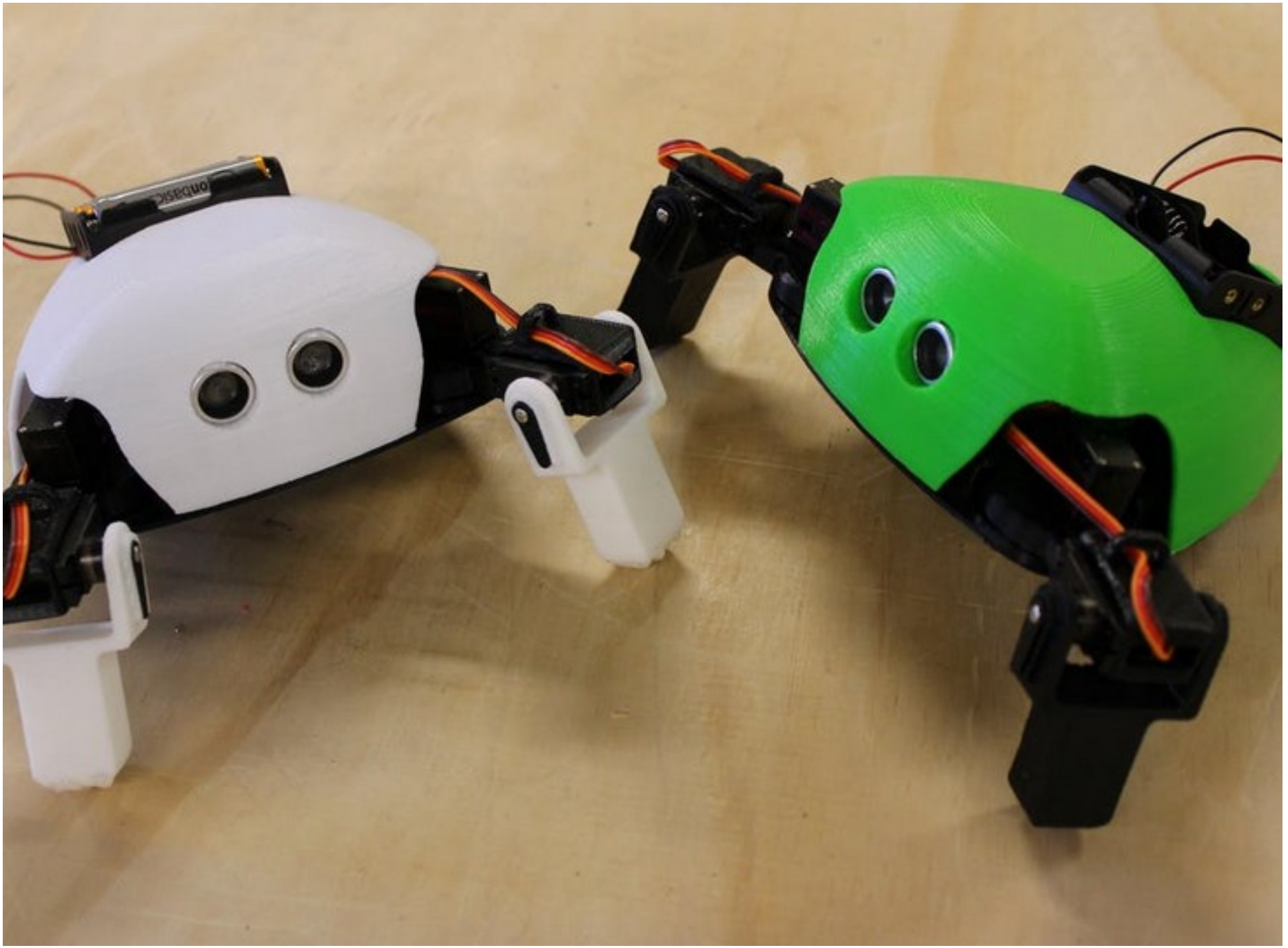
Ha creato un nuovo ecosistema, anche commerciale; l'economia del dono produce anche questi piccoli miracoli.

Ne approfitteremo, anzi ne faremo man bassa . . .

Arduino



Critter: Crawling Arduino Robot



Gli ambienti di sviluppo

Gli ambienti di sviluppo del firmware tradizionalmente erano a linea comandi, Linux o dos-like.

Non e' piu' cosi'.

Dall'avvento di Arduino il panorama dello sviluppo di firmware e' cambiato, ed oggi sviluppare con qualcosa di diverso da un IDE e' divenuto un vezzo del passato, una situazione dominata dalla pigrizia o dal timore, tipo "quello che funziona non si cambia".

IDE di Arduino

L'IDE di Arduino, dietro la sua apparente semplicita', quasi "banalita'", e' importante perche' rappresenta una facile porta di accesso ad una quantita' incredibile di software e librerie, ed al supporto di una altrettanto incredibile quantita' di schede hardware e microcontroller.

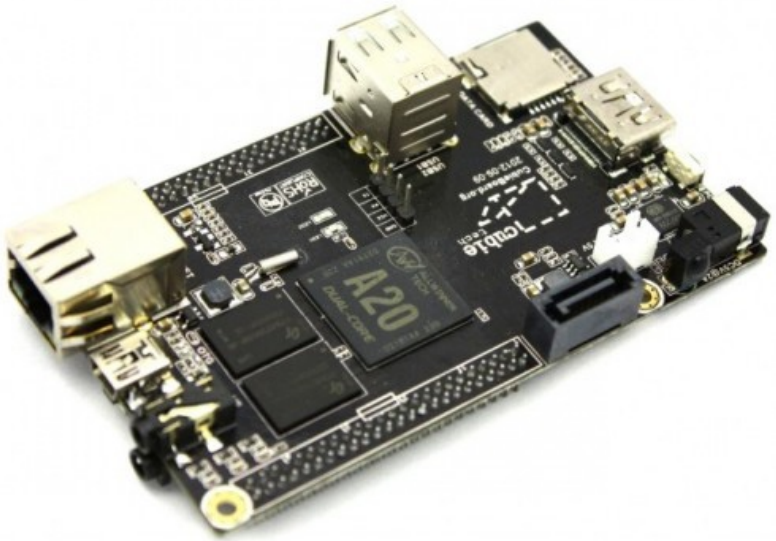
Come molte moderne IDE, ovviamente installabili sul proprio computer, ne esiste una versione completamente cloud, che permette anche di gestire file, firmware compilati e librerie senza che debbano risiedere sul computer da cui l'IDE viene acceduta.

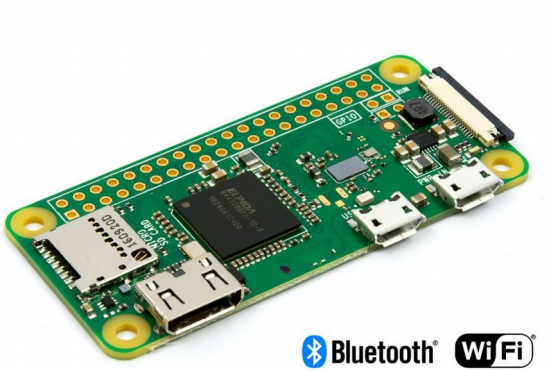
Se questo sia solo un vantaggio od anche un pericolo lo giudicheranno le singole persone secondo i rispettivi gusti e necessita'.

Hardware general purpose

Oltre, e forse piu', del software anche l'evoluzione dell'offerta commerciale di hardware potente o specifico per l'IoT apre grandi possibilita' per lo sviluppo di oggetti IoT.

Con prezzi da 15 a 45 euro per un pezzo, troviamo la Cubieboard 2, Raspberry PI3 e Raspberry Zero

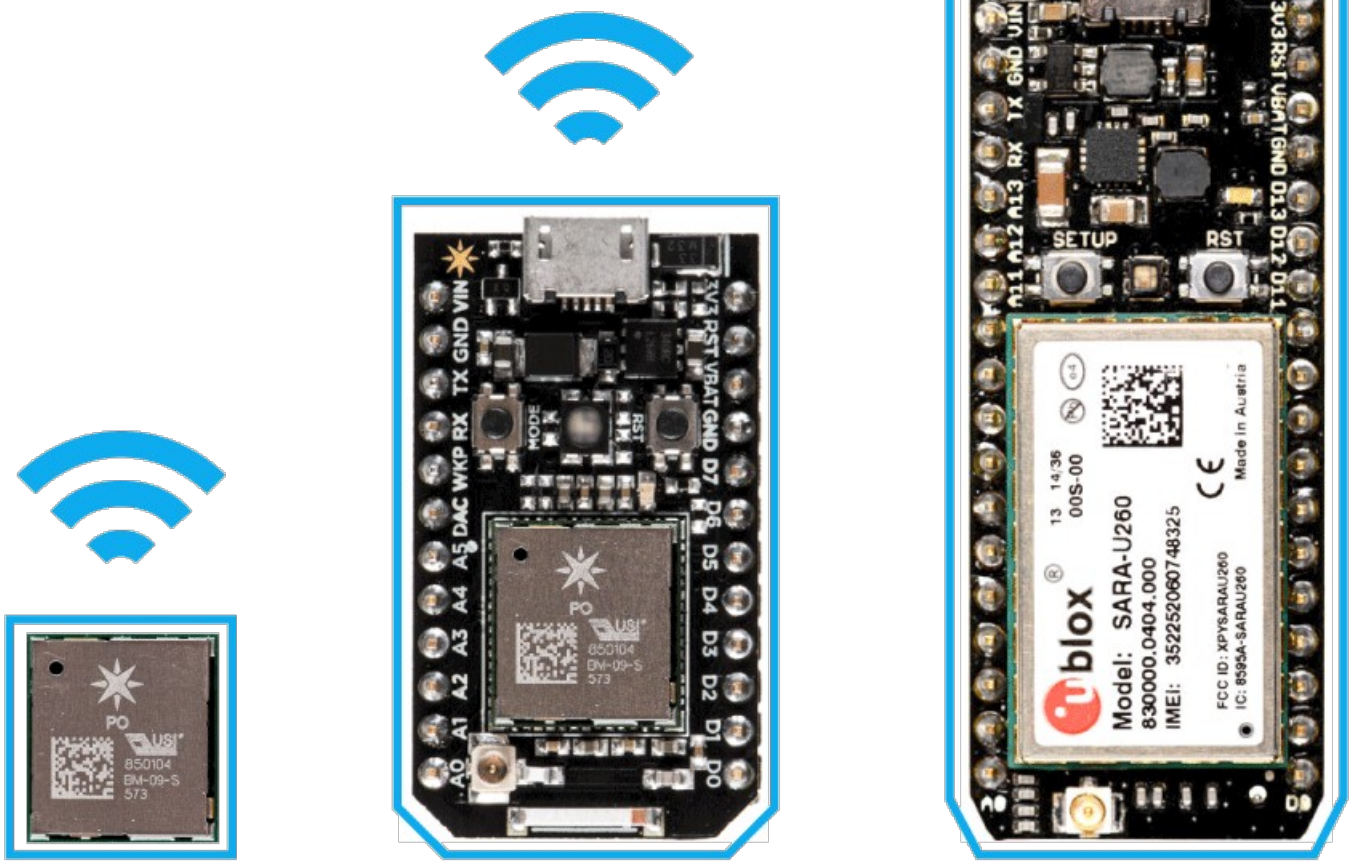




Hardware specializzato IoT

Molto interessanti sono le schede "proprietarie" di fornitori di servizi IoT completi (full IoT stack service provider) come le Photon e la Electron di Particle Industries.

Utilissime anche le SIM 3G dati "globalizzate"



Fornitori di Servizi IoT

Negli ultimi due anni sono nati fornitori di servizi e soluzioni IoT che forniscono un insieme completo di servizi per lo sviluppo, la prototipazione, la vendita e la gestione di oggetti IoT, e di "flotte" di oggetti IoT.

Particle.io (Particle Industries, inc.) ad esempio fornisce un sevizio completo per lo sviluppo e la commercializzazione di oggetti IoT.

L'ambiente di sviluppo software IDE è pubblico e gratuito, accessibile solo via cloud; fornisce librerie di sviluppo, esempi di software ed accurata documentazione per utilizzarlo con una curva di apprendimento brevissima.

Fornitori di Servizi IoT - 2

Vengono fornite, a prezzi bassi, schede standardizzate complete e programmabili per le principali tipologie di oggetti. Sono disponibili anche le corrispondenti schede di sviluppo per l'ottimizzazione del progetto.

Gli oggetti sviluppati possono interagire tra di loro e con i sistemi di controllo utilizzando il Cloud di servizi di Particle.io, quindi lato server non è necessario sviluppare l'intero sistema di gestione ma solo, se necessaria, l'applicazione di controllo.

La compilazione, il deployment e l'aggiornamento del software/firmware sono interamente svolti nel cloud e via internet; nessuna necessità di programmare flash EEPROM, o di accedere fisicamente ogni singolo oggetto via scomodi connettori seriali, USB o JTag.

Fornitori di Servizi IoT - 3

Un oggetto IoT necessita di un canale wireless di comunicazioni: Bluetooth, WiFi, GSM o 3G/4G.
I vari tipi di schede standard forniscono uno di questi canali. Sono disponibili SIM internazionali, a funzionamento garantito in un intero continente.

Terminata la fase di prototipazione, anche la produzione può utilizzare le schede standardizzate di Particle, approvvigionate sia a listino che sulla base di accordi commerciali (oltre 100 pezzi).
La gestione di intere flotte di oggetti, a cominciare dall'aggiornamento dei firmware, utilizza servizi esistenti ed economici forniti via cloud.

Grazie a questo insieme di servizi, la barriera economica, di apprendimento e di tempo necessario per sviluppare su IoT si abbassa drasticamente.

Fornitori di Servizi IoT - 4

I servizi iniziali per una attività di sviluppo sono gratuiti.

La scheda di sviluppo e' acquistabile ad un costo basso; tipicamente scheda ed accessori richiedono un impegno economico complessivo di meno di 100 euro.

All'interno dell'ecosistema di Particle.io è possibile anche utilizzare la diffusissima e standardizzatissima scheda Raspberry PI3, famosa quanto Arduino ma ordini di grandezza più potente.

Lock-in, Sicurezza e Privacy

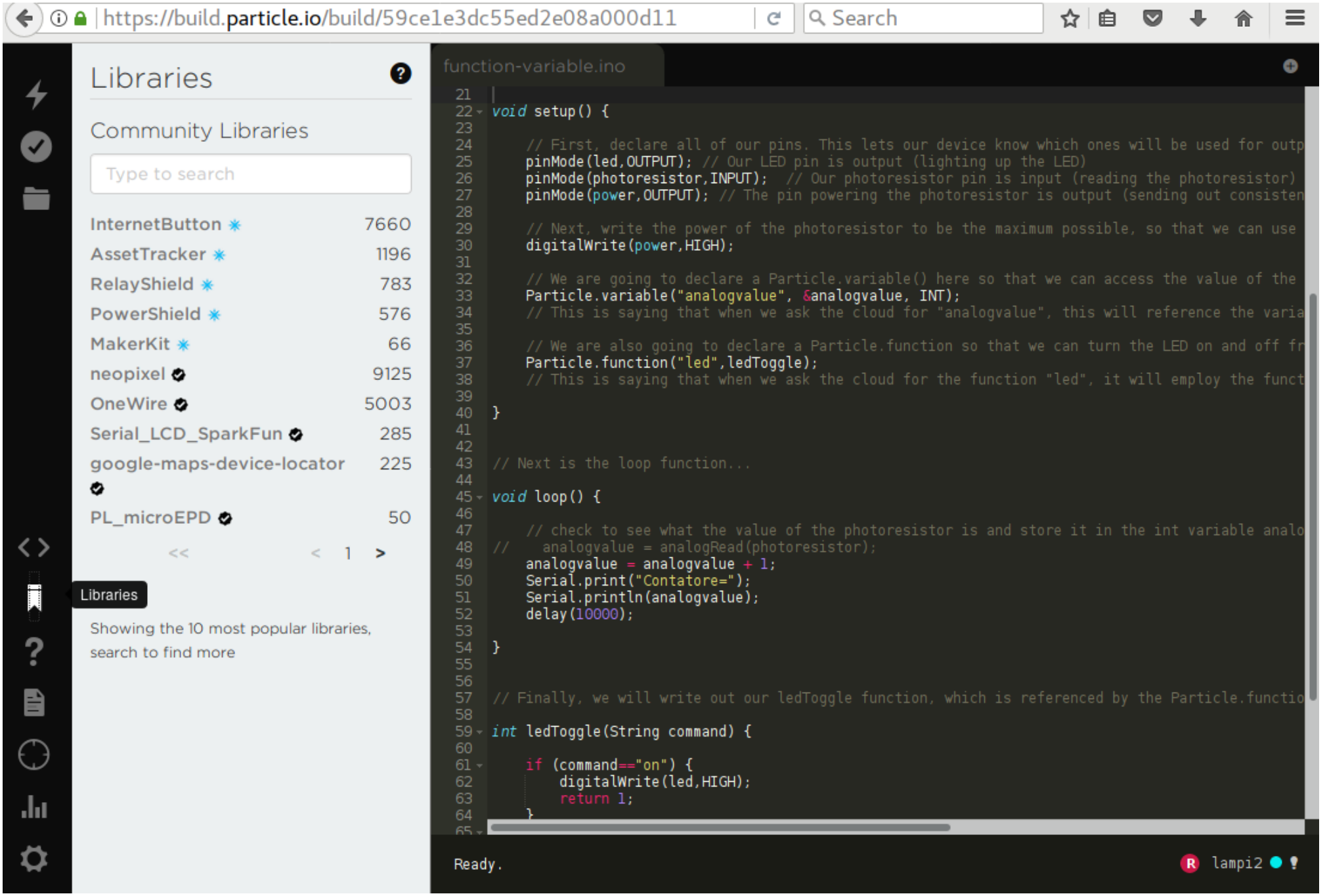
A chiusura di questa parte del corso, tre considerazioni importanti per ogni attivita' di sviluppo software/hardware, in particolare per l'IoT.

Vendor lock-in - adottando un fornitore di servizi (come Particle.io) si diventa "prigionieri" della sua tecnologia, delle condizioni commerciali, degli SLA e della loro proprieta' intellettuale.

Sicurezza – per tutelarsi da possibili pesanti conseguenze legali (e fare un buon lavoro) il software di un oggetto IoT deve essere (almeno) allo stato dell'arte del software commerciale.

Privacy – gli oggetti IoT sono fonti di dati personali e sensibili che vengono esportati, gestiti e memorizzati sul server; attenzione ad informare gli utenti ed a valutare sempre le conseguenze legali.

IDE di Particle

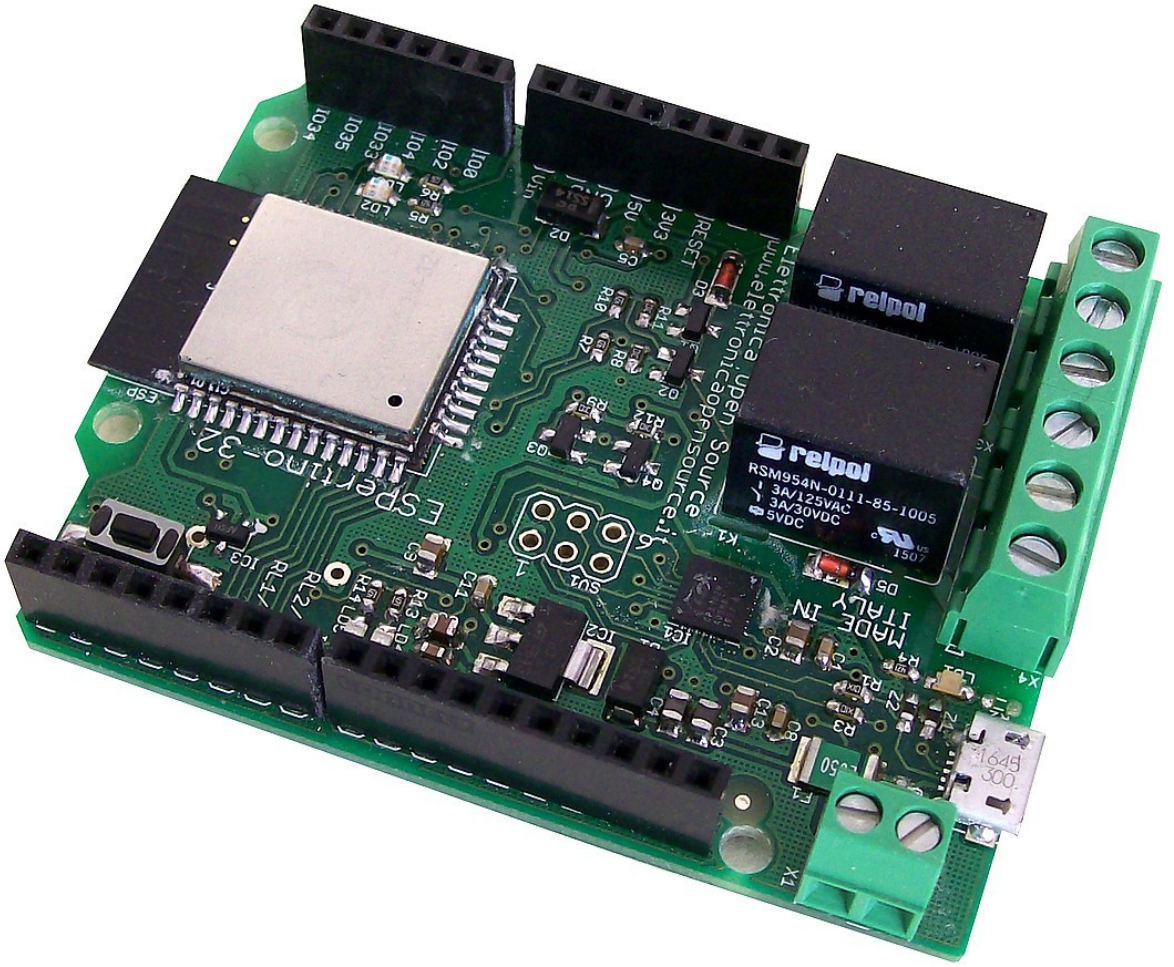


Esercitazione

Utilizzo dell'IDE di Particle

ESPertino

ESPertino, la scheda che useremo nel corso, non ha grosse capacita' di calcolo ma, oltre che WiFi e Bluetooth, possiede interessanti capacita' di stand-by ad assorbimento ridottissimo, e di controllo diretto di carichi elettrici alimentati a 220V.



ESPertino - 2

Uno degli oneri del progettista elettronico in generale e' di provvedere le semplici ma onerose funzionalita' di interfacciamento elettrico, protezione della scheda e condizionamento dei segnali, sempre richieste quando il mondo analogico e quello digitale si incontrano e devono convivere nello stesso oggetto.

ESPertino e' stato selezionato perche' si accontenta di una singola alimentazione a 5V, utilizza i due piu' importanti protocolli wireless e puo' controllare direttamente dei carichi elettrici grazie ai rele' onboard.

Sviluppiamo un oggetto IoT

Il progetto del corso

La cosa piu' semplice ed utile in un oggetto IoT e' il suo controllo wireless e via Internet; ad esempio una lampada da tavolo modificata che puo' essere accesa e spenta via internet.

Inserendovi in uno spazio vuoto disponible una scheda ESPertino alimentata da rete, il lavoro hardware si riduce a collegare dei fili; quello software invece, pur semplificato dalle sofisticate librerie disponibili, rimane non banale.

Proporvi lo schema elettrico dell'oggetto che andremo a realizzare sarebbe offensivo nei vostri confronti, quindi, dopo aver modificato una normale prolunga elettrica per inserirvi la nostra scheda, passiamo direttamente a parlare di software.

Specifiche del progetto

Il software del progetto deve permettere di controllare un carico elettrico da Internet.

Elenchiamo delle semplici specifiche:

- **L'oggetto controllato non deve richiedere la connessione alla rete locale del luogo di installazione, quindi deve essere collegato via wireless;**
- **L'oggetto deve essere alimentato a 220V;**
- **Il controller che andremo a realizzare deve poter essere inserito nell'oggetto originale.**

Funzionalita' del software

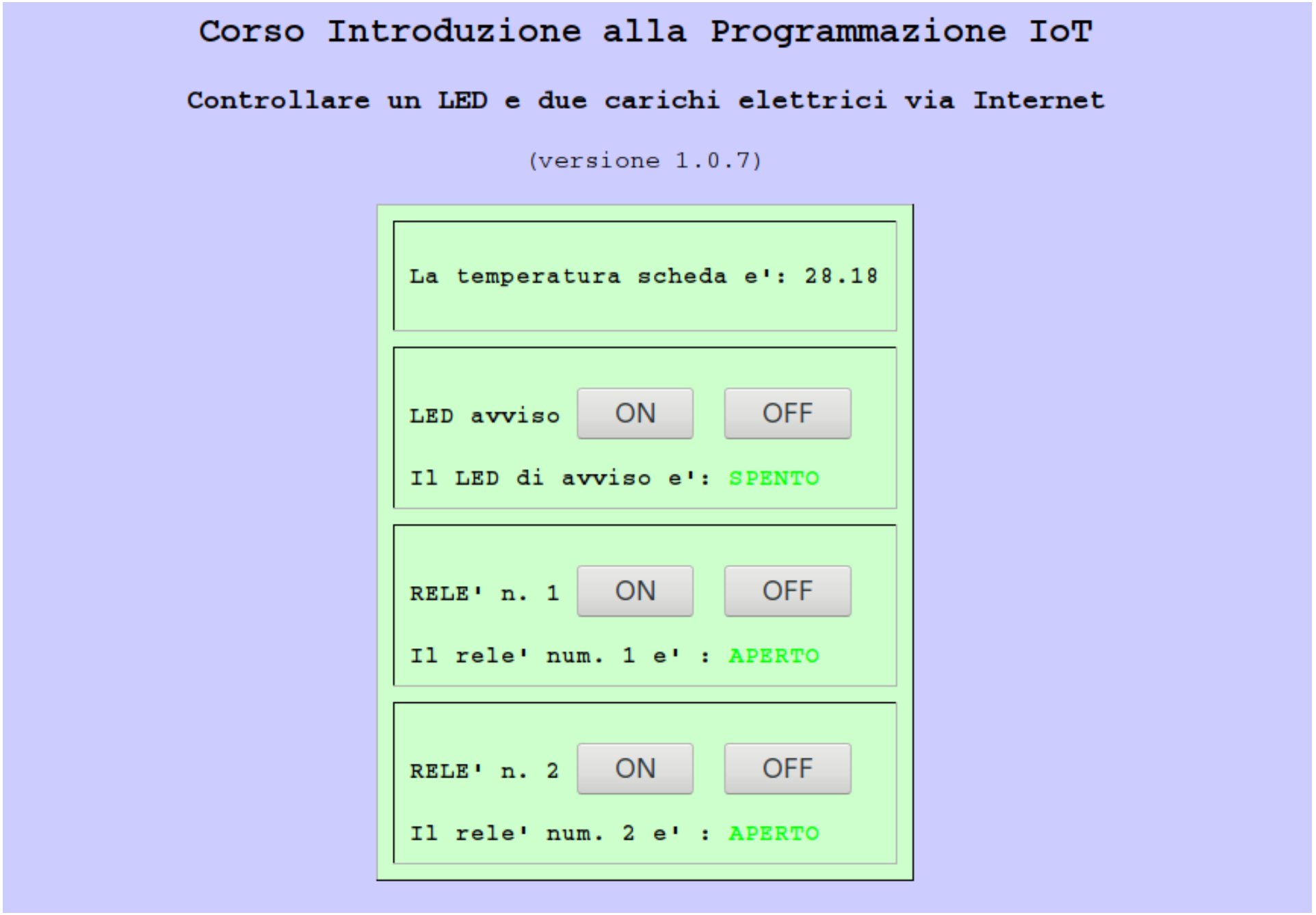
L'interfaccia di controllo sara' ovviamente realizzata con un web server, che dovra' risiedere sulla scheda stessa.

Ricordiamo che la scheda ESPertino permette di controllare due carichi elettrici, possiede un LED gestibile direttamente via software, ed un sensore di temperatura.

Estendiamo lo scopo del progetto al controllo anche al controllo di questi elementi supplementari, in modo da rendere il software sviluppato ancora piu' versatile e riutilizzabile.

Le due funzionalita' di rete, web server ed interfaccia WiFi saranno ottenute (per fortuna) utilizzando librerie preconfezionate.

Mockup dell’interfaccia web



Struttura del software

Come gia' accennato, il software di un microcontrollore, per quanto evoluto, anche oggi gira spesso sul Bare Metal, ossia direttamente sulla CPU, senza la presenza di un sistema operativo ne' di un monitor.

Deve essere quindi un eseguibile linkato staticamente, possibilmente compilato utilizzando librerie specializzate in modo da limitare la quantita' di software da scrivere (e da debuggare).

Ricordiamo che la memoria e' ridotta, non esiste un gestore di memoria virtuale e neppure la possibilita' di multitasking, quindi dovremo tornare a tempi ormai remoti e ad una programmazione attenta e parca nell'utilizzo delle poche risorse disponibili.

Un programma “nullafacente”

```
#include <WiFi.h>


void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Il software client

Il programma principale, ogni programma principale per un controller Arduino-like, e' scritto in un dialetto del C, ed e' composto da tre parti:

Parte dichiarativa – contiene dichiarazioni di variabili e caricamento di moduli di libreria;

Parte di setup – inizializza hardware & software, compiendo tutte quelle operazioni che devono essere eseguite solo all'accensione dell'oggetto IoT;

Main loop – dove tutto succede e dove tutto deve essere fatto.

Sezione inizializzazione

```
#include <WiFi.h>
// ssid e password richiesti per la connessione alla rete WiFi
const char* ssid     = "pippo";
// inserite l'ssid della vostra rete WiFi
const char* password = "123456";
// inserite la password della vostra rete WiFi

// definizione pin di I/O utilizzati
#define USER_LED   5
// pin a cui e' collegato il led a disposizione del programma utente
#define RELAY_1    12
// pin a cui è collegato il primo rele'
#define RELAY_2    14
// pin a cui è collegato il secondo rele'

// variabili di appoggio
char buffer_riga[80];
int contatore=0;
boolean riga_vuota;
char version_sw[6] = "1.0.6";

// istanza del web server, creata sulla porta 80
WiFiServer server(80);
```

Sezione setup

```
void setup()
{
  // inizializzazione dei pin (direzione e stato iniziale)
  pinMode(USER_LED, OUTPUT);
  pinMode(RELAY_1,  OUTPUT);
  pinMode(RELAY_2,  OUTPUT);


  // setup sensore temperatura e DAC
  // pinMode(36,  OUTPUT);
  analogReadResolution(12); //12 bits
  analogSetAttenuation(ADC_0db);  //For all pins

  // inizializzazione stato del controller
  digitalWrite(USER_LED, HIGH); // led spento
  digitalWrite(RELAY_1,  LOW);
  digitalWrite(RELAY_2,  LOW);


  // inizializzazione della porta seriale: baudrate 115200
  Serial.begin(115200);
```

Sezione setup - 2

```
  // connessione alla rete WiFi
  Serial.println("+----------------------------------------+");
  Serial.println("|  Benvenuti al corso : Introduzione all'IoT  |");
  Serial.println("|  Web server per controllo LED e 2 carichi   |");
  Serial.println("+----------------------------------------+");
  Serial.println(version_sw);
  Serial.print("  Connessione alla rete WiFi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  // loop di attesa sino a quando la connessione viene stabilita
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connessione alla rete WiFi OK!");
  Serial.println("Il DHCP mi ha dato l'indirizzo IP: ");
  Serial.println(WiFi.localIP());

  // avvio del server
  server.begin();
}
```

Main loop

```
void loop()
{
  // attesa della connessione da parte di un client
  WiFiClient client = server.available();
  if (client)
  {

  Serial.println("Richiesta da un nuovo client");

    // inizializzazione variabili di appoggio
    memset(buffer_riga, 0, sizeof(buffer_riga));
    contatore=0;
    // ogni richiesta http termina sempre con una riga vuota
    boolean riga_vuota = true;

while (client.connected())
{
      if (client.available())
```

Main loop - 2

```
{
  char c = client.read();
  Serial.write(c);

  // lettura richiesta HTTP, carattere per carattere
  buffer_riga[contatore]=c;

  if (contatore < sizeof(buffer_riga)-1) contatore++;

  // se si e' arrivati a fine della riga, e riga vuota,
  // la richiesta http e' terminata. Quindi procedere
  // con l'invio della risposta

  if (c == '\n' && riga_vuota)

  {
    // invio dell'header standard di risposta HTTP
    int led = digitalRead(USER_LED);
    int rly1 = digitalRead(RELAY_1);
    int rly2 = digitalRead(RELAY_2);
```

Main loop - 3

```
// lettura e scaling della temperatura
int reading = analogRead(36);
// current voltage off the sensor
float voltage = reading * 3.3;
// using 3.3v input
voltage /= 4096.0;
// divide by 1024
float temp = (voltage - 0.5) * 100 ;
//converting from 10 mv per degree with 500 mV offset
float temperature = temp * 0.16;      // scaling

String header_risp = "HTTP/1.1 200 OK\r\n";
header_risp += "Content-Type: text/html\r\n";
header_risp += "Connection: close\r\n";
// la connessione verra' chiusa dopo la risposta
header_risp += "\r\n";

header_risp += "<!DOCTYPE HTML>\r\n<html>
                \r\n<head>\r\n";
...
header_risp += "<h2>Corso ""Introduzione alla
                     Programmazione IoT"" </h2>\r\n";
```

Main loop – 4

```
if (c == '\n')
          {
            // inizio di una nuova linea
            riga_vuota = true;
            if (strstr(buffer_riga, "GET /led_on") > 0)
            {
              Serial.println("USER LED ON");
              digitalWrite(USER_LED, HIGHT);
            }
            else if (strstr(buffer_riga, "GET /led_off") > 0)
        ...           }
            // inizio di una nuova riga
            riga_vuota = true;
            memset(buffer_riga, 0, sizeof(buffer_riga));
            contatore=0;
          }
          else if (c != '\r')
          {
            riga_vuota = false; // la riga ha 1 o + caratteri
          }
        }}
```

Main loop - 5

```
// pausa necessaria affinche' il browser possa
ricevere i dati

    delay(1);

    // chiusura della connessione
    client.stop();
    Serial.println("client disconnesso");
  }
}
```

Boot del firmware

```
ets Jun  8 2016 00:22:57
rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0x00
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,
wp_drv:0x00 mode:DIO, clock div:1
load:0x3fff0008,len:8
load:0x3fff0010,len:2016
load:0x40078000,len:7780
ho 0 tail 12 room 4
load:0x40080000,len:252
entry 0x40080034

+-----------------------------------------+
|  Benvenuti al corso : Introduzione all'IoT  |
|  Web server per controllo LED e 2 carichi   |
+-----------------------------------------+
1.0.7
  Connessione alla rete WiFi: marcowifi
....

Connessione alla rete WiFi avvenuta con successo!
Il DHCP mi ha dato l'indirizzo IP:
192.168.101.35
```

Output della seriale

```
Richiesta da un nuovo client

GET / HTTP/1.1
Host: 192.168.101.35
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:52.0) Firefox/52.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

client disconnesso
```

HTML generato

```
<!DOCTYPE HTML>
<html>
<head>
<meta name="viewport" content="width=device-width">
</head>
<body bgcolor=ccccff><font face=Courier>
<center>
<h2>Corso Introduzione alla Programmazione IoT</h2>
<h3>Controllare un LED e due carichi elettrici via
             Internet</h3>(versione 1.0.7)<br><br>
<b>
<table bgcolor=ccffcc border=1 cellpadding=10 cellspacing=10>
<tr><td>
<p>La temperatura scheda e': 28.18</tr></td>
<tr><td>
<p>LED avviso
<a href="led_on"><button>ON</button></a> 
<a href="led_off"><button>OFF</button></a></p>
Il LED di avviso e': <b><font color=00ff00>SPENTO</font></b>
</tr></td>
...
```

HTML generato - 2

```
...

<p>RELE' n. 2
<a href="relay2_on"><button>ON</button></a> 
<a href="relay2_off"><button>OFF</button></a></p>
Il rele' num. 2 e' : <b><font color=00ff00>APERTO</font></b>
</td></tr>

</table>

</b>
</center>

</body>

</html>
```

Grazie per l'attenzione

+ Marco A. Calamari marco.calamari@ordineingegneripisa.it --+
| |
| PGP RSA: ED84 3839 6C4D 3FFE 389F 209E 3128 5698 |
| DSS/DH: 8F3E 5BAE 906F B416 9242 1C10 8661 24A9 BFCE 822B |
| Tel: (+39) 050 576031 Cell: (+39) 347 8530279 |
| Fax: (+39) 050 7849817 Skype-Twitter: calamarim |
| |
+ P.E.C.: marcoanselmoluca.calamari@ingpec.eu ------------+

Link utili

The Ambient Orb su Amazon
https://www.amazon.com/AMBIENT-DEVICES-Stock-Orb-AORB02/dp/B000653KKQ

Nabaztag su Wikipedia
https://it.wikipedia.org/wiki/Nabaztag

Sito del progetto Arduino
https://www.arduino.cc/

IDE di Arduino
https://www.arduino.cc/en/Main/Software

Critter: Crawling Arduino Robot
https://create.arduino.cc/projecthub/slantconcepts/critter-crawling-arduino-robot-123670

Descrizione dei tipi di schede Arduino disponibili
https://it.wikipedia.org/wiki/Arduino_(hardware)

Uso dell'IDE
https://it.wikipedia.org/wiki/Arduino_(software)

Introduzione al linguaggio di Arduino
http://www.alberti-porro.gov.it/wordpress/wp-content/uploads/2014/01/ProgrammareArduino.pdf

Link utili - 2

Sito della rivista EMC Elettronica
https://it.emcelettronica.com/

Articolo dedicato ad ESPertino
https://it.emcelettronica.com/la-nuova-scheda-espertino-per-liot

La costruzione di un Web Server per il progetto del corso
https://it.emcelettronica.com/web-server-con-espertino

Sito di Particle Industries, inc.
https://www.particle.io/

Slide del corso
http://www.cassandracrossing.org/documents/OIP_Calamari_introduzione_programmazione_IoT.pdf

Sketch dei sorgenti per il progetto del corso
http://www.cassandracrossing.org/documents/OIP_Calamari_introduzione_programmazione_IoT_esempio.ino

Firmware del progetto, compilato per scheda ESP32
http://www.cassandracrossing.org/documents/OIP_Calamari_introduzione_programmazione_IoT_esempio.esp32.bin

Adafruitcom – sito di commercio elettronico per schede di sviluppo
https://www.adafruit.com/

Q&A time

+ Marco A. Calamari marco.calamari@ordineingegneripisa.it --+
| |
| PGP RSA: ED84 3839 6C4D 3FFE 389F 209E 3128 5698 |
| DSS/DH: 8F3E 5BAE 906F B416 9242 1C10 8661 24A9 BFCE 822B |
| Tel: (+39) 050 576031 Cell: (+39) 347 8530279 |
| Fax: (+39) 050 7849817 Skype-Twitter: calamarim |
| |
+ P.E.C.: marcoanselmoluca.calamari@ingpec.eu ---------------+