Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Ingegneria e Scienze Informatiche

JAVASCRIPT E INTERNET OF THINGS
-
L'utilizzo di Espruino sulla board ESP88266

Relatore:
Chiar.mo Prof.
ALESSANDRO RICCI

Presentata da:
RICCARDO SOLAZZI

Sessione II
Anno Accademico 2016/2017

Alle mie nonne Argentina e Rovalda,
sperando di averle rese orgogliose da qua

Introduzione

L'obiettivo di questa tesi è quello di introdurre il concetto di Internet of Things e di mostrare alcuni dispositivi che possono aiutare alla realizzazione di questo concetto. Per fare ciò, dopo aver analizzato ciò che viene rappresentato come Internet of Things e le sue caratteristiche, sono stati trattati diversi dispositivi sia per quanto riguarda l'hardware sia per quanto riguarda il software. Successivamente si è deciso di analizzare il linguaggio di programmazione JavaScript in quanto offre innumerevoli vantaggi anche nell'ambito della programmazione embedded. Infine si è analizzato il funzionamento di un firmware che permette l'utilizzo del linguaggio analizzato precedentemente su una board nata, appunto, per la programmazione di sistemi embededd.

Indice

Elenco delle figure

Capitolo 1

Internet of Things

1.1 L'essenza di Internet of Things

Internet of things è un neologismo coniato da Kevin Ashton nel 1999 utilizzato come titolo di una presentazione fatta alla "Procter and Gamble". L'autore del suddetto termine decise di tornare sulla propria definizione dieci anni dopo in un articolo del portale "RFID Journal" in cui spiegò ciò che lui intendesse con questo concetto.
Internet of Things sarebbe stata la via futura per risolvere i limiti di tempo, precisione e attenzione di quello che era l'attuale mezzo di acquisizione delle informazioni: l'essere umano.
L'idea dietro a questo concetto era quella di non fermarsi però all'acquisizione dei dati, ma anche di elaborarli e capirli, riuscendo a far riconoscere l'ambiente esterno agli elaboratori senza dover ricorrere al fattore umano.
Il concetto di Internet of Things vide le luci della ribalta quando nel 2005 l'"International Telecommunications Union" pubblicò una relazione in cui analizzò ciò che IoT fosse in quel momento e ciò che sarebbe diventato. Questa relazione redatta da Lara Srivastava analizzò come la tecnologia attuale di quegli anni fosse più pervasiva di quanto si sarebbe potuto immaginare dieci anni prima e che il progresso tecnologico avrebbe continuato in quella direzione.

Un altro aspetto che la relazione diede alla luce era il fatto che Internet of Things avrebbe puntato a connettere gli oggetti di tutti i giorni in maniera intelligente e sensoriale combinando la tecnologia di identificazione ("tagging things"), reti di sensori ("feeling things"), sistemi integrati ("thinking things") e la nanotecnologia ("shrinking things").
In questo preciso momento storico, con Internet of things, si intende una rete di oggetti interconnessi e individuati in modo univoco che possono comunicare tra loro o interagire con il mondo reale. Internet of Things è quindi il mezzo tramite il quale è possibile avere un collegamento tra mondo reale e mondo virtuale andando a creare dei dispositivi in grado di interagire con l'ambiente che li circonda.
I dispositivi che fanno parte della rete di oggetti sono chiamati **smart things** e hanno la peculiarità di poter interagire all'interno del sistema di comunicazione in cui sono inseriti risultando come dei dispositivi attivi. Questi dispositivi sono caratterizzati da diversi fattori:

- Sono degli oggetti fisici;
- Hanno risorse limitate in termini di capacità di elaborazione, memoria, approvvigionamento energetico;
- Sono identificati da un codice alfanumerico univoco che permette il riconoscimento del dispositivo, spesso il codice è accompagnato da un nome;
- Possono essere influenzati e influenzare la realtà che li circonda.

L'obiettivo dell'internet delle cose è fare in modo che il mondo elettronico tracci una mappa di quello reale dando un'identità elettronica alle cose e ai luoghi dell'ambiente fisico. Gli oggetti e i luoghi muniti di tag (come rfid o qr code) sono in grado di comunicare informazioni in rete o a dispositivi mobili come i telefoni cellulari.

1.2 Architetture di Internet of Things

Nell'ambito dell'Internet of Things non esiste un'architettura che è stata definita come standard ma esistono più architetture che sono state proposte dai propri sviluppatori.

1.2.1 Architetture a tre e a cinque livelli

L'architettura a tre e a cinque livelli è l'architettura più semplice sviluppata per Internet of Things ed è stata una delle prime ad essere introdotta. Ha tre livelli nominati:

1. **Perception layer**: livello fisico che ha sensori che si occupano delle rilevazioni del dati dell'ambiente circostante. Si occupa di percepire i parametri fisici o di identificare gli altri oggetti intelligenti presenti nell'ambiente in analisi

2. **Network layer**: livello responsabile della connessione agli altri smart object, alla rete e ai server. Le sue caratteristiche permettono di utilizzarlo anche per la trasmissione e l'elaborazione dei dati.

3. **Application layer**: livello della fornitura di servizi specifici per l'utente. Definisce diverse applicazioni in cui l'Internet of Things può essere impiegato.

Nonostante l'architettura a tre livelli definisca l'idea principale di Internet of Things, essa non risulta essere sufficiente. L'aggiunta di due livelli permette di colmare le lacune dell'architettura a tre livelli, andando così a creare l'architettura a cinque livelli. In questa architettura sono stati aggiunti il **business layer** e il **processing layer** e il network layer viene sostituito dal **transport layer**. I ruoli dei livelli application e perception rimangono invariati. I nuovi layer saranno quindi:

1. **Transport layer**: si occupa del trasferimento dei dati acquisiti dai sensori del perception layer al processing layer e vice versa attraverso la rete utilizzando tecnologie come 3g, LAN, bluetooth, RFID e NFC.

2. **Processing layer**: è il livello che viene riconosciuto anche come livello middleware. Immagazzina, analizza, e processa tutti i dati che vengono dal transport layer.

3. **Business layer**: si occupa della gestione dell'intero sistema IoT, incluse le applicazioni e la privacy degli utenti.

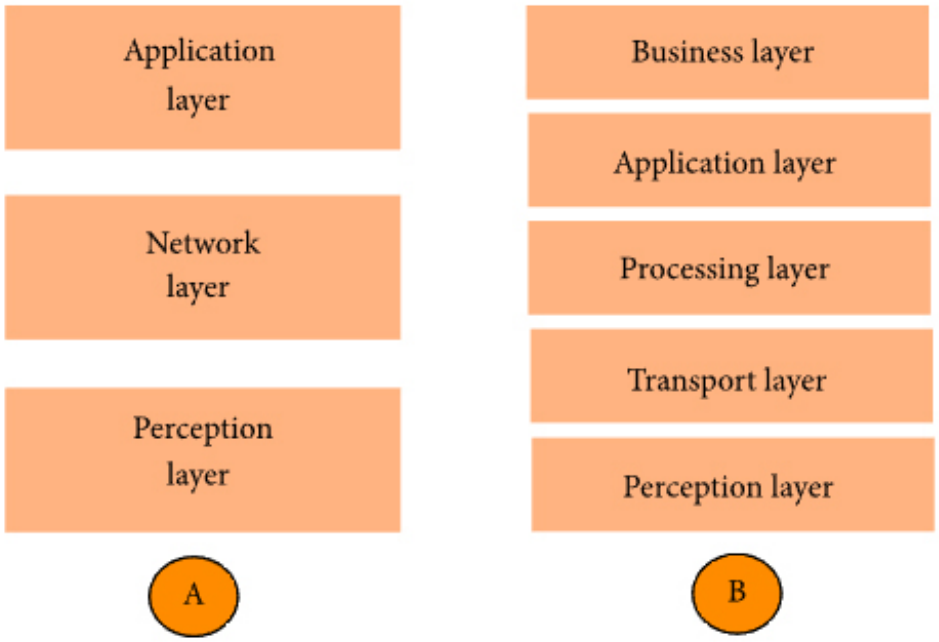

Figura 1.1: Architettura dei modelli a tre e cinque livelli

1.2.2 Cloud and Fog Based Architecture

In alcune architetture di IoT l'elaborazione dei dati è effettuata in una rete di computer organizzati tramite cloud centralizzati. Questo tipo di architettura permette una grande flessibilità e una grande scalabilità e offre servizi quali l'infrastruttura principale, la piattaforma, il software e la memoria che possono essere integrati dagli strumenti degli sviluppatori. Si è passati poi da un'architettura basata sul cloud computing a una basata sul

fog computing.
In questa architettura i sensori e i network gateway fanno parte dell'elaborazione e analisi dei dati. Quest'architettura presenta un approccio a livelli che inserisce dei livelli di **monitoring, preprocessing, storage** e **security** tra il physical layer e il transport layer. Il monitoring layer controlla l'energia, le risorse, le risposte e i servizi. Il preprocessing layer si occupa del filtraggio, dell'elaborazione e dell'analisi dei dati sensoriali. Lo storage layer fornisce le funzionalità legate alla memoria come la replicazione dei dati, la distribuzione e il salvataggio. Il security layer si occupa di decifrare e cifrare i dati per assicurare l'integrità e la sicurezza dei dati.

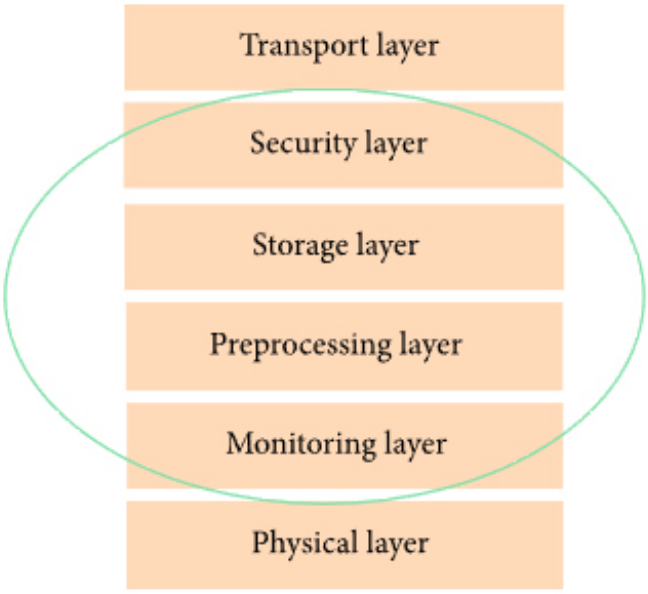

Figura 1.2: Architettura cloud and fog based

1.2.3 Social IoT

L'architettura chiamata **Social IoT - SIoT** è un'architettura che si basa sul trattare le relazioni tra gli oggetti della rete come gli essere umani trattano

i rapporti con gli altri esseri umani.
I tre aspetti principali di questo tipo di architettura sono:

1. **Navigabilità**: questo tipo di architettura permette la navigazione attraverso tutti i dispositivi connessi alla rete.

2. **Attendibilità**: La presenza di una forte relazione tra i device presenti nella rete permette di creare un rapporto attendibile.

3. **Modello social**: il modello social di quest'architettura permette di poter analizzare e studiare il rapporto tra i device analogamente a come funziona una rete social tra esseri umani.

L'architettura social presenta anche un'architettura server side. Il server si connette a tutti i componenti presenti nella rete, compone i servizi e funge da singolo punto per l'erogazione dei servizi per l'utente.
L'architettura server side ha tipicamente tre livelli:

1. **Base layer**: livello che contiene un database con tutti i dettagli dei device, dei loro attributi, le loro metainformazioni e le loro relazioni.

2. **Component layer**: questo livello ha i codici che servano per interagire con i device, per interrogare lo status dei device stessi e utilizza un loro sottoinsieme per eseguire un servizio.

3. **Application layer**: livello che si occupa dell'erogazione dei servizi per gli utenti

L'architettura per i device invece presenta due livelli:

1. **Object layer**: livello che si occupa di permettere al device di connettersi e di comunicare con gli altri device.

2. **Social layer**: livello che si occupa di gestire e eseguire le applicazioni degli utenti e interagire con l'application layer dell'architettura del server.

1.3 Wireless Sensor Networks - WSN

La realizzazione dell'obiettivo prefissato dall'Internet of Things è possibile grazie a tecnologie come **WSN – Wireless Sensor Networks** utilizzate per operazioni di sensing. Con il termine wireless sensor network si indica una determinata tipologia di rete informatica caratterizzata da un'architettura distribuita e realizzata da un insieme di dispositivi autonomi in grado di prelevare dati dall'ambiente circostante e di comunicare tra loro.
I nodi di questa particolare rete sono chiamati **nodi sensore** e sono formati da dispositivi in grado di rilevare grandezze fisiche tramite l'utilizzo di sensori, di elaborare i dati ottenuti e di comunicare tra loro.
La rete di sensori è una rete dinamica che riesce a supportare anche un incremento del numero di sensori connessi oppure anche una diminuzione degli stessi dovuta a dei guasti; inoltre la topologia di questa rete, oltre a variare spesso, dipende dall'utilizzo che se ne fa. La wsn è una rete ad-hoc con alcune limitazioni dovute sia al tipo di comunicazione usata sia ai sensori impiegati, che però risulteranno avere delle caratteristiche comuni come il basso costo, la bassa potenza e con la possibilità di eseguire varie funzioni nei limiti delle loro caratteristiche computazionali e di memoria.
La progettazione di WSN dipende da alcuni fattori quali:

- **Tolleranza ai guasti**: è la capacità di mantenere le funzionalità della WSN senza interruzioni dovute ai guasti dei nodi. La tolleranza di un nodo k è modellata tramite una distribuzione di Poisson ed è vista come la probabilità di non avere malfunzionamenti nell'intervallo di tempo (0,t). Il tasso di affidabilità è quindi $R_k(t) = e^{-\lambda_k t}$ dove λ_k è il tasso di fallimento del nodo sensore k;

- **Scalabilità**: è la capacità della rete di poter gestire un aumento del numero di sensori connessi, è spesso utilizzata anche la densità di sensori all'interno del raggio di trasmissione di ciascun nodo in una data regione X. La densità è ottenuta dalla seguente formula: $\mu(R) = \frac{NxR^2}{X}$

dove R indica il range del raggio di trasmissione dei nodi e N il numero di dispositivi della regione X;

- **Costi di produzione**: visto il numero elevato di sensori in una rete WSN il costo di ogni singolo nodo è un fattore rilevante per la progettazione della rete stessa. L'obiettivo è quello di utilizzare sensori a basso costo;

- **Ambiente**: i sensori devono adattarsi agli ambienti in cui verranno utilizzati riuscendo a funzionare in maniera appropriata senza la supervisione dell'uomo anche in condizioni svantaggiose, come temperature estreme o ambienti molto rumorosi;

- **Topologia della rete**: la quantità elevata di sensori presenti nella rete obbliga la rete stessa a essere dinamica e fa si che la struttura della rete cambi frequentemente;

- **Vincoli hardware**: ogni nodo deve consumare una quantità limitata di energia; per fare ciò si devono imporre dei vincoli per quanto riguarda la capacità computazionale del nodo stesso. Spesso alcuni sensori vengono progettati in modo da poter ricavare energia da fonti esterne per permettergli di adattarsi meglio all'ambiente in cui verranno utilizzati.

- **Mezzi di comunicazione**: le reti di sensori prevedono l'utilizzo di tecnologie wireless per la comunicazione tra i nodi. L'assenza di cablaggio permette il posizionamento anche in luoghi difficili da raggiungere dai cavi e permette una certa dinamicità nella struttura della rete. Tra le tecnologie predilette per la comunicazione wireless della rete di sensori troviamo **Bluetooth** e **zigBee** che permettono una comunicazione wireless con un dispendio energetico limitato.

- **Consumo di energia**: i sensori utilizzati hanno una limitata sorgente di energia, per esempio una batteria, e questo limita le prestazioni

dell'intera rete, ecco perchè può risultare utile progettare i sensori con pannelli fotovoltaici o altri tipi di tecnologia in modo che essi siani indipendenti per quanto riguarda l'approvvigionamento energetico.

1.4 Web Semantico

Tenendo conto dell'immensa mole di device connessi a internet, fornire un'interoperabilità tra i device stessi che formano ciò che chiamiamo "Internet of things" è uno dei requisiti fondamentali per supportare l'indirizzamento, il monitoraggio, l'acquisizione, la rappresentazione, lo stoccaggio e lo scambio di informazioni. Per fare ciò è stata sviluppata un'estensione del World Wide Web chiamata **"Web Semantico"**.
Secondo il W3C il Web Semantico offre un framework comune che permette ai dati di essere condivisi e riutilizzati attraverso le applicazioni, le imprese e i limiti della comunità.
L'idea è nata grazie a Tim Berners-Lee e consisterebbe nell'applicare tecniche avanzate di conoscenza al fine di ovviare al problema del **knowledge gap**; ovvero la differenza che c'è tra le conoscenze che il calcolatore utilizza ai fini della computazione e quelle possedute e utilizzate dall'utente. Si tratterebbe di rendere la conoscenza in formati standard facilmente interpretabili per i computer, in modo tale che questi ultimi possano essere in grado di processarla e di ragionare in maniera automatizzata sui dati. Le macchine, grazie a tali standard, sarebbero in grado di operare ragionamenti anche complessi, in quanto in grado di riconoscere le risorse.
L'idea che permetterebbe la realizzazione del Web Semantico sarebbe quella di creare uno strato di informazioni per i documenti. Tale livello risulterebbe invisibile all'utente in quanto avrebbe unicamente lo scopo di venire interpretato dai calcolatori in modo tale che essi possano processare le risorse, non si andrebbe quindi a sostituire il web come lo conosciamo, ma soltanto ad estenderlo.
Uno dei presupposti essenziali è costituito dall'utilizzo di linguaggi che han-

no introdotto la possibilità di strutturare i dati, come XML. Tali linguaggi favorirebbero il processo di attribuzione di significati alle risorse in modo da renderle comprensibili ai calcolatori. Sulla base del linguaggio XML è stato sviluppato il linguaggio RDF che rappresenta lo standard proposto da W3C per la codifica, lo scambio e il riutilizzo di metadati strutturati. Esso inquadra le strutture di dati secondo tassonomie gerarchiche rendendo possibile l'esecuzione di procedure inferenziali.
Fornire descrizioni semantiche però non fornisce interoperabilità semantica e non risolve i problemi riguardanti la scoperta, la gestione e l'interazione con i dati. Le descrizioni semantiche hanno bisogno di essere condivise, processate e interpretate dai diversi metodi e servizi attraverso differenti domini. Definire un'ontologia per la descrizione semantica dei dati permette di creare un'interoperabilità per gli utenti che condividono la stessa ontologia. Attualmente sono definite diverse ontologie in base al contesto dei progetti nel quale vengono utilizzate, ma l'obiettivo dietro alla creazione delle ontologie stesse è quello di riuscire a crearne una globale che sia compatibile con tutti i progetti che le utilizzano.
Nell'ambito IoT usare un'ontologia comune permette di creare delle note semantiche che andranno a definire i metadati collegati ai dati stessi, permettendo di avere dei dati leggibili e interpretabili dalle macchine.

1.5 I linguaggi cardine del Web Semantico

Per permettere lo sviluppo del web semantico vengono utilizzati diversi linguaggi che permettono di individuare, unire e mettere in relazione tra loro le informazioni all'interno del web. Ciò che il web semantico descrive è detta risorsa ed essa viene rappresentata utilizzando sia il legame che c'è tra le varie risorse, sia utilizzando le proprietà della risorsa stessa. I linguaggi cardine del web semantico sono:

- **Resource Description Framework (RDF)**: linguaggio utilizzato per rappresentare metadati all'interno del World Wide Web in modo

machine-oriented, ovvero orientato verso una comunicazione Machine-To-Machine(M2M). RDF è composto da due componenti: **RDF Model and Syntax** che espone la struttura del modello RDF e **RDF Schema** che espone la sintassi per definire schemi e vocabolari per i metadati. La codifica dell'informazione avviene attraverso l'utilizzo dell'Uniform Resource Identifier (URI) che permette di individuare in modo univoco, attraverso una stringa, la risorsa da descrivere.
L'informazione è descritta attraverso RDF utilizzando tre parti: il **soggetto** che indica la risorsa, il **predicato** che identifica la caratteristica del soggetto/risorsa e l'**oggetto** che identifica il valore della proprietà.

- **Ontology Web Language (OWL)**: linguaggio di markup utilizzato per rappresentare esplicitamente il significato e la semantica di termini con vocabolari e relazioni tra gli stessi. Lo scopo è quello di descrivere delle basi di conoscenze, effettuare delle deduzioni su di esse e integrarle con i contenuti delle pagine web.

- **SPARQL Protocol and RDF Query Language (SPARQL)**: linguaggio query che permette, attraverso delle interrogazioni, di individuare i dati in formato RDF presenti all'interno del web, il quale viene considerato come un database.

Capitolo 2

L'hardware di IoT

2.1 Introduzione

La definizione del termine "Internet of Things" mette in risalto come l'obiettivo sia quello di creare una relazione tra il mondo fisico e quello digitale. Ci sono oggetti che vorremmo tracciare, controllare e con cui vorremmo interagire, questi oggetti rappresentano le "things" nell'Internet of things e vengono chiamate **entità d'interesse**. Qualsiasi cosa che include degli attributi che lo descrivono e che descrivono il suo stato e che risulta essere rilevante per un utente o per un applicazione può essere considerata un'entità d'interesse.
Per interagire con le entità d'interesse è necessario che esse siano connesse a internet e ciò è possibile grazie all'utilizzo di determinati device. Questi device possono essere integrati o allegati all'entità stessa (andando a creare ciò che viene chiamata **smart thing**) oppure possono essere installati nell'ambiente che circonda le identità d'interesse. I device possono, ad esempio, includere lettori RFID, sensori, attuatori e addirittura computer integrati.

2.2 System on a Chip

I **system on a chip** sono circuiti integrati che contengono i componenti di un computer o di altri sistemi elettronici. La popolarità dei system on a chip nel mobile computing è data dal fatto che questi sistemi sono ottimizzati per lavorare con un fabbisogno energetico limitato e hanno dimensioni contenute. La loro area di applicazione maggiore è quella dei sistemi integrati. La struttura di un system on a chip comprende:

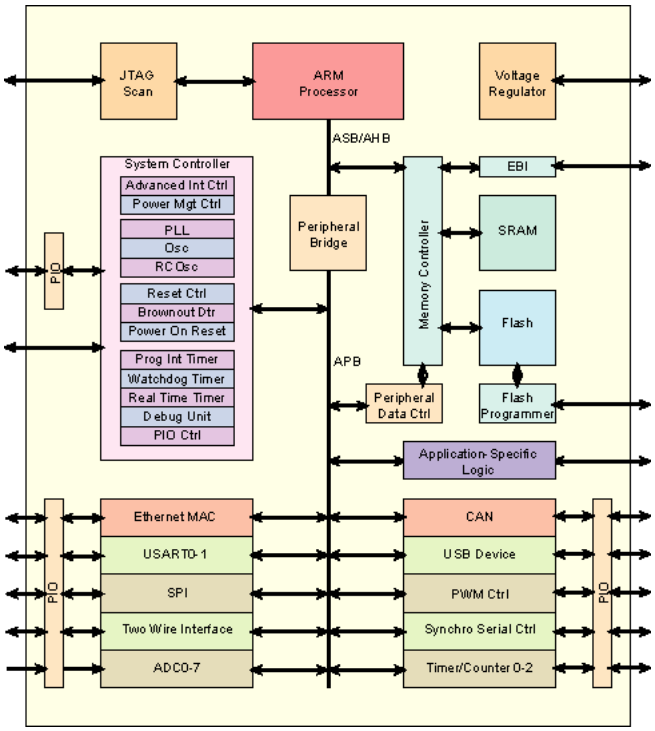

Figura 2.1: Struttura del SOC

- Uno o pù core di microcontrollori o microprocessori;
- Un modulo di memoria contenente uno o più blocchi di tipo ROM, RAM, EEPROM o memoria Flash;
- Un generatore di clock e un PLL
- Periferiche come contatori, orologi e altro;

- Connettori per interfacce standard come USB, FireWire, Ethernet, USART e SPI
- Convertitori digitali-analogici e convertitori analogici-digitali
- regolatori di tensione e circuiti di gestione dell'alimentazione.

I vari blocchi funzionali sono collegati tramite BUS proprietari o BUS standard.

2.3 Arduino

2.3.1 La scheda

Arduino è una piattaforma hardware open-source composta da una serie di schede elettroniche dotate di un microcontrollore, ovvero un dispositivo elettronico integrato su un unico chip, progettato appositamente per interagire con input esterni, analogici o digitali, e restituire output analogici o digitali derivati dalle operazioni di processamento interno determinate da un programma caricato nella memoria del chip.

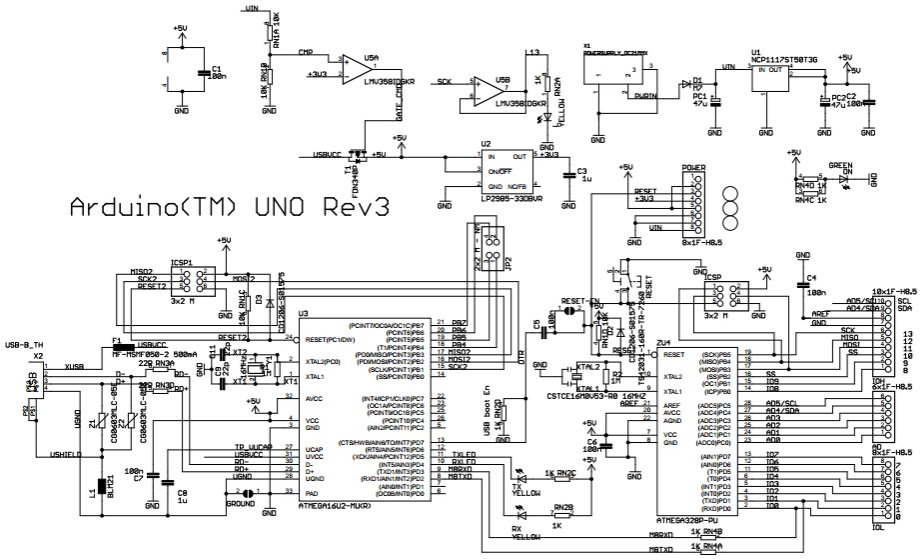

Figura 2.2: Struttura di un Arduino UNO

Il prodotto di riferimento della linea Arduino è **Arduino UNO** che presenta un microcontrollore ATmega328P. Per permettere ad Arduino di comunicare con un computer tramite USB e renderne semplie la programmazione ogni scheda integra un ulteriore chip per la conversione del segnale digitale da USB a seriale.

Nell'arduino UNO il chip programmato come convertitore USB-seriale è il chip ATmega16U2. I modelli più recenti chiamati **Arduino Leonardo** o **Arduino Esplora** non hanno bisogno di un ulteriore chip per la conversione USB-seriale poichè il microcontrollore che di cui sono dotati integra già questa funzione.

Per quanto le schede Arduino possano avere diverse componenti in base al modello preso in considerazione, tutte le schede hanno dei pin di input e di output attraverso i quali il microcontrollore riceve le informazioni in ingresso e tramite i quali comunica con l'ambiente esterno.

2.3.2 Le Shield

I vari modelli di Arduino possono essere estese in maniera semplice grazie all'utilizzo delle **shield**, ovvero schede compatibili con alcuni modelli di Arduino che permettono ad esempio di connettere la scheda ad una rete Wi-Fi, di collegare Arduino alla rete GSM, di controllare motori elettrici e servomotori in maniera più efficace e stabile o ancora di collegare più Arduino tra loro. Proprio come Arduino, le shield sono hardware open-source: le schematiche di ogni scheda si possono scaricare e modificare a piacimento. Le shield realizzate dal team di Arduino sono:

- **GSM Shield**: per la connessione alla rete GSM;
- **Motor Shield**: consente di guidare motori DC, stepper, relè e solenoidi.
- **Wifi Shield**: permette la connessione alla rete Wi-Fi;
- **Ethernet Shield**: consente alla scheda di essere connessa tramite ethernet
- **Wireles SD Shield**: consente di comunicare in modalità wireless utilizzando un modulo wireless Xbee o simile più uno slot per schede micro-SD;
- **Proto Shield**: permette la realizzazione di circuiti elettronici da interfacciare direttamente con Arduino, in questa shield è possibile saldare direttamente altri componenti hardware;
- **Wireless Proto Shield**: consente di comunicare in modalità wireless utilizzando un modulo wireless Xbee o simile.

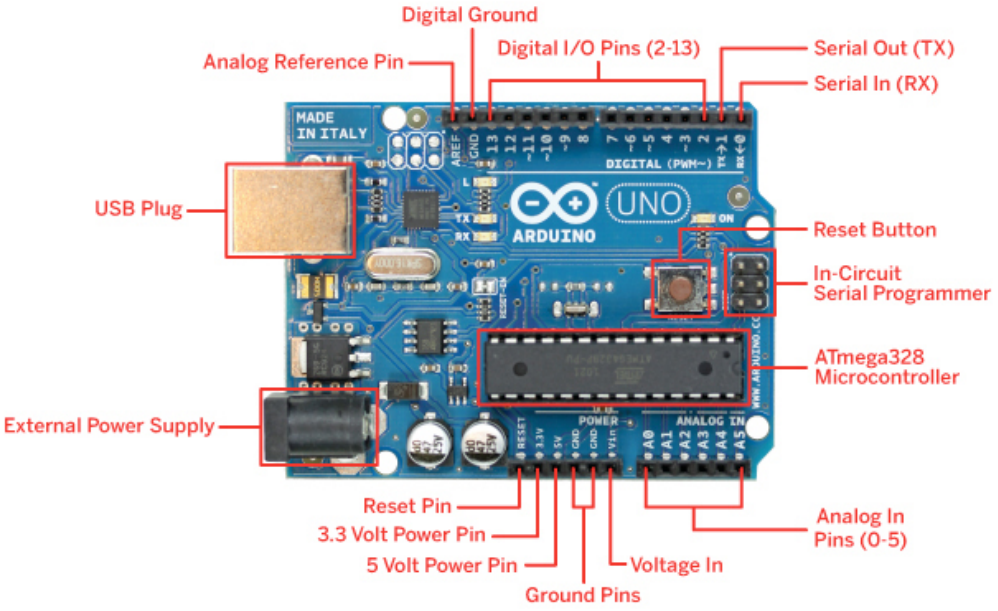

Figura 2.3: Scheda Arduino UNO

2.4 Raspberry PI

2.4.1 La scheda

Raspberry PI è un calcolatore implementato su una sola scheda elettronica lanciato sul mercato a inizio 2012. Ne sono stati prodotti diversi modelli con l'idea di realizzare un dispositivo economico concepito per stimolare l'insegnamento di base dell'informatica e della programmazione nelle scuole.
Raspberry risulta essere un'ottima piattaforma per la prototipazione di sistemi embedded avanzati, che devono svolgere compiti per i quali le soluzioni basate su microcontrollori non forniscono tipicamente sufficienti capacità di elaborazione/memoria.
La scheda presenta un SoC Broadcom che incorpora un processore ARM, una GPU VideoCore IV e una quantità di memoria ram che varia in base al modello, da un minimo di 245mb fino a 1Gb. La scheda non prevede una memoria non volatile come hard disk o ssd, ma presenta un lettore di schede

SD per la memoria e per il boot della scheda stessa.
Proprio come Arduino il Raspberry PI è in grado di comunicare con l'ambiente che lo circonda grazie all'utilizzo di sensori e attuatori, elaborando l'input e restituendo dati in output.

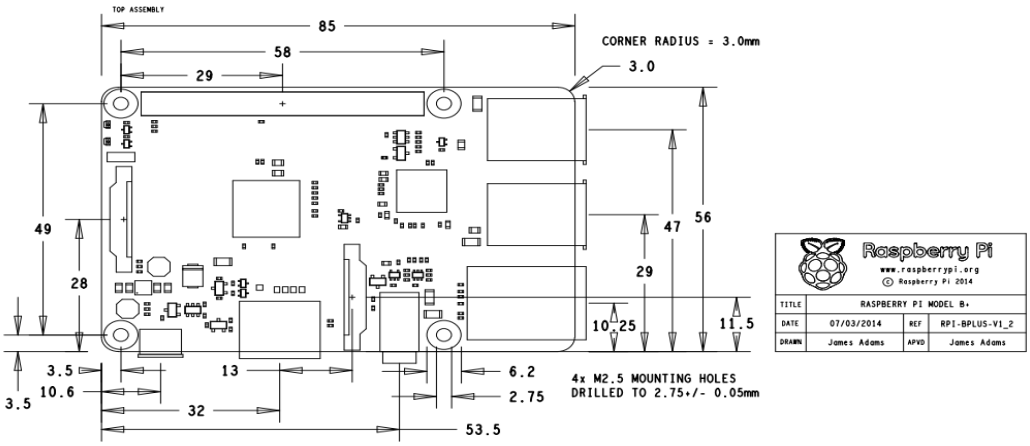

Figura 2.4: Struttura della scheda Raspberry Pi modello B+

La differenza che c'è tra le due board riguarda il fatto che mentre Arduino è una board programmabile tramite pc, Raspberry è un pc a se stante sul quale è possibile montare un sistema operativo ed è in grado di eseguire le applicazioni che ci vengono installate.

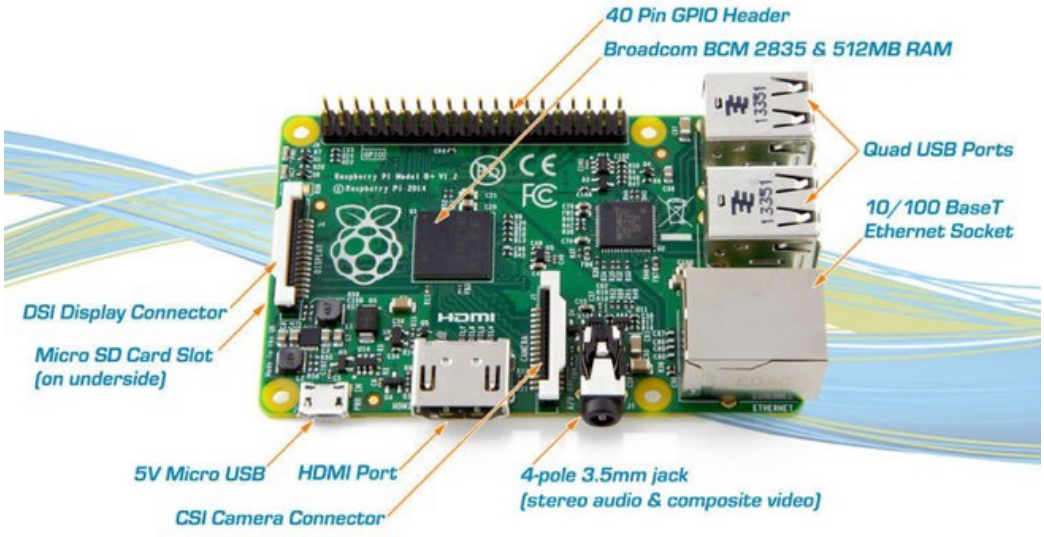

Figura 2.5: Scheda Raspberry Pi

2.4.2 Il sistema operativo di Raspberry Pi

Ci sono diversi sistemi operativi compatibili con il Raspberry Pi tra cui RISC OS, Pidora, Arch Linux ma il più utilizzato risulta essere **Raspbian**. Raspbian è un sistema operativo open-source basato su Debian. Raspbian ha tutte le caratteristiche principali di Debian con alcune modifiche effettuate per rendere il Raspberry più semplice da utilizzare e include diversi software pensati proprio per la piattaforma Raspberry. Raspbian è stato progettato per essere utilizzato in maniera semplice e intuitiva per chi si approccia per la prima volta a Raspberry Pi.

Tra le caratteristiche che Raspbian eredita da Debian troviamo una repository di software molto estesa che permette l'installazione semplificata

dei software richiesti per ampliare quelli già presenti. Raspbian ha diversi componenti chiave, questi componenti sono:

- **Raspberry Pi bootloader**: si occupa di inizializzare l'hardware e di far partire l'esecuzione del Kernel Linux;
- **Kernel Linux**: gestisce ogni parte del sistema operativo, dalla gestione della tastiera a quella dello schermo;
- **Daemons**: software che si occupa di fornire al sistema operativo diverse funzionalità come Apache web server, Cron e Autofs;
- **La Shell**: interfaccia che permette all'utente di interagire con il Raspberry Pi tramite dei comandi testuali
- **Il server grafico X.Org**: fornisce all'utente una piattaforma comune dalla quale viene costruita la GUI;
- **L'ambiente desktop**: si occupa di permettere all'utente di interagire utilizzando mouse e finestre, non soltanto la tastiera come la shell.

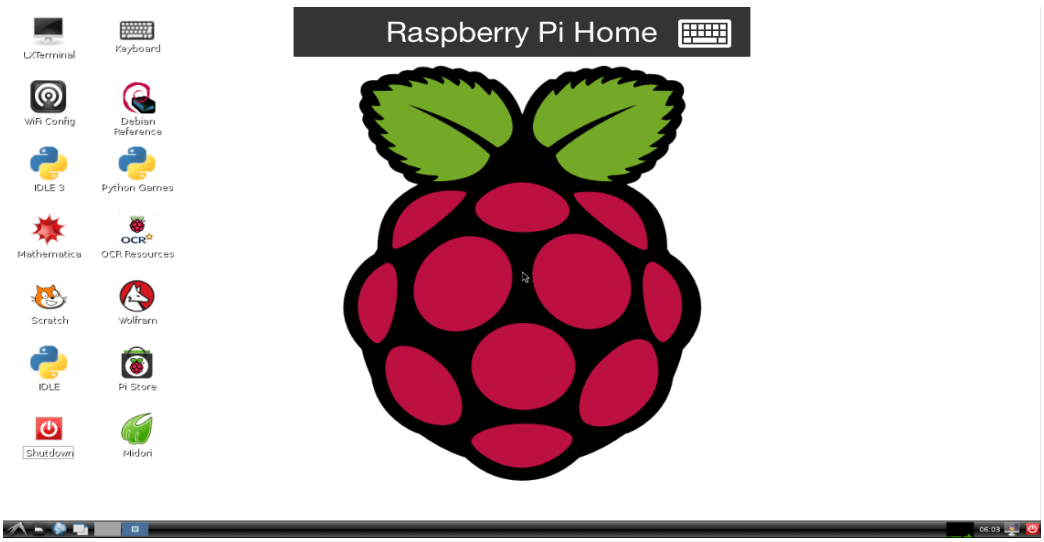

Figura 2.6: Screen del sistema operativo Raspbian

2.5 NodeMcu

2.5.1 La scheda

NodeMcu è una piattaforma open-source sviluppata per l'ambiente IoT che include un firmware basato sulla scheda **Esp8266**, ovvero un chip Wi-Fi a basso costo con supporto completo a tcp/ip e funzionalità da microcontrollore prodotto dall'azienda cinese Espressif System programmabile tramite LUA. Tra i componenti di questo chip troviamo:

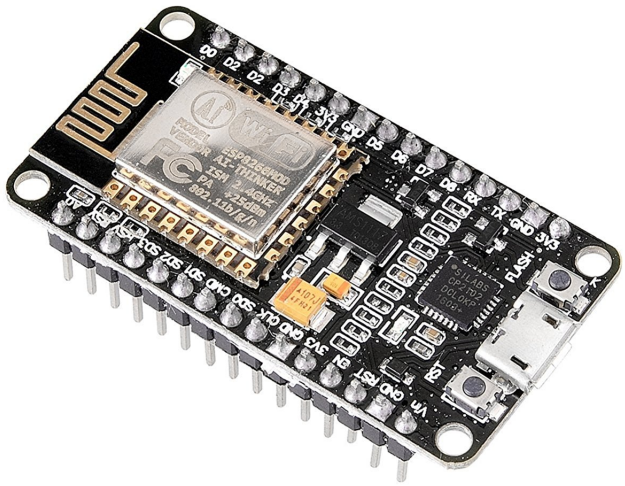

Figura 2.7: Scheda ESP8266

- Microcontrollore a 32 bit Tensilica L106(MCU) che utilizza un basso consumo energetico e un RSIC a 16 bit, raggiungendo una velocità massimo di clock di 160MHz;
- 64KiB di RAM dedicati alle istruzioni;

- 96KiB di RAM dedicati ai dati;
- Memoria Flash QSPI esterna da 512KiB a 4MiB;
- IEEE 802.11 b/g/n Wi-Fi;
- 16 GPIO pin.

Tra le interfacce per la comunicazione con le periferiche supportate troviamo:

- UART (Universal Asynchronous Receiver-Transmitter) su alcuni pin dedicati;
- SDIO;
- I2C (Inter Integrated Circuit);
- I2S (Integrated Interchip Sound);
- IR Remote Control;
- GPIO (General Purpose Input Output);
- ADC (Analog to Digital Converter);
- PWM (Pulse-width modulation);

2.5.2 Firmware

La scheda nasce per funzionare utilizzando il firmware **NodeMCU**, ovvero un firmware basato su **eLua**.
ELua è un progetto che offre l'implementazione del linguaggio **Lua** nel mondo embedded, estendendolo con specifiche funzionalità per andare a creare un ambiente efficiente e portatile per lo sviluppo dei software relativi al mondo embedded.
Alcune caratteristiche di eLua sono le seguenti:

- **Controllo completo della piattaforma**: eLua non prevede l'utilizzo di un sistema operativo, quindi i programmi funzionano direttamente grazie al microcontrollore;
- **Portabilità del codice**: è possibile programmare in Lua, in C o in un linguaggio ibrido tra i due senza doversi preoccupare della compatibilità con tutte le piattaforme e le architetture che sono supportate da eLua;
- **Dinamicità dell'hardware**: il codice risulta essere completamente indipendente dall'hardware, quindi è possibile effettuare modifiche all'hardware per il quale il programma è stato scritto senza andare a danneggiare il funzionamento del codice;
- **Sviluppo diretto**: non necessita dell'installazione di un ambiente di sviluppo sul pc per la programmazione nè di un emulatore di terminale o di seriale. Si può utilizzare un qualsiasi editor di testo, salvare il programma in una sd/mmc e farlo girare sulla scheda;
- **Flessibilità**: essendo un linguaggio di scripting ad alto livello risulta essere altamente adattabile e riconfigurabile permettendo delle modifiche future economiche ed efficienti;
- **RAD integrato**: permette di prototipare e sperimentare i progetti direttamente, senza l'utilizzo di simulatori;
- **Longevità**: possibilità di aggiungere configurazioni personalizzate ai progetti, rendendoli adattabili ai contesti che cambiano sempre;
- **Licenza**: eLua è un software open-source e gratuito che è possibile utilizzare anche in ambiti commerciali e in prodotti con codice privato senza il dover pagare royalty o diritti di utilizzo.

Tuttavia eLua non è l'unica alternativa per lavorare utilizzando la piattaforma NodeMCU: **Espruino**, al quale verrà dedicato il quarto capitolo di questa tesi, è una piattaforma che permette di utilizzare Javascript nella scheda

ESP8266, andando a creare un ambiente dinamico e modificabile anche da remoto.

Capitolo 3

Il linguaggio Javascript

3.1 Introduzione a Javascript

Javascript è un linguaggio di scripting orientato agli oggetti ed agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso.
Tali funzioni di script possono essere opportunamente inserite in file HTML, in pagine JSP o in appositi file separati con estensione .js che verranno richiamati. Il suo campo di utilizzo è stato esteso alle *Hybrid App* con le quali è possibile creare applicazioni per più sistemi operativi utilizzando un unico codice sorgente basato appunto su JavaScript, HTML e CSS.
Proprio grazie a questa particolarità JavaScript ha aumentato la propria popolarità riuscendo a permettere la creazione di applicazioni per più device.

3.2 La storia di JavaScript

JavaScript fu creato nel maggio del 1995 da Brendan Eich, ai tempi dipendente della Netscape, con il nome **Mocha**, scelto da Marc Anderssen, fondatore dell'azienda per la quale lavorava Brendan Eich. Nel settembre del

1995 il nome venne di nuovo cambiato in LiveScript per risaltarne la dinamicità. Nel dicembre dello stesso anno il nome fu di nuovo cambiato in quello con cui è attualmente riconosciuto grazie una mossa di marketing, infatti il nome JavaScript fu scelto per cavalcare l'onda della popolarità che aveva Java a quei tempi in seguito alla licenza ricevuta dalla Sun. Nel biennio 1996-1997 JavaScript fu preso in analisi dall'organizzazione **ECMA - European Computer Manufacturers Association** per avviare il processo di standardizzazione del linguaggio, standard che verrà chiamato **ECMAScript**.
Il processo di standardizzazione continuò ciclicamente, permettendo il rilascio di **ECMAScript 2** nel 1998 e **ECMAScript 3** nel 1999. Questa versione è quella che compone le fondamenta del JavaScript moderno come noi lo conosciamo. L'attuale versione aggiornata è **ECMAScript 6** rilasciata nel giugno 2015.
Questa versione implementa diversi cambiamenti sintattici che permettono di scrivere applicazioni più complessi, incluse le classi e i moduli definendoli semanticamente. Attualmente il supporto da parte dei browser di ECMAScript 6 rimane incompleto.
L'utilizzo di JavaScript ebbe un'impennata a metà anni 2000, specialmente quando **Jesse James Garrett** introdusse al mondo **Ajax**: una tecnica di sviluppo software per la realizzazione di applicazioni web interattive che utilizza JavaScript come spina dorsale. Questo portò alla creazione di diverse librerie come **Prototype**, **JQuery**, **Dojo** e **Moototools**.

3.3 I vantaggi di JavaScript

JavaScript offre un notevole numero di vantaggi tanto da renderlo presente in molti siti, anche per operazioni semplici che il linguaggio **HTML** non permette.
Tra i vantaggi che si possono avere dall'utilizzo di JavaScript ci sono:

- il fatto che JavaScript offra ai progettisti uno strumento di programmazione HTML con una sintassi di scrittura molto semplice;

- il fatto che sia in grado di reagire ad eventi, come ad esempio agli input creati dall'utente tramite tastiera e mouse;
- il fatto che possa interpretare, leggere e modificare il contenuto di un elemento HTML
- il fatto che possa essere utilizzato per la creazione dei cookie;
- il fatto che, essendo in grado di capire quale browser è utilizzato dall'utente, permette di gestire il caricamento di pagine create appositamente per quel browser.

3.4 Caratteristiche di JavaScript

3.4.1 Imperativo e strutturato

JavaScript supporta la maggior parte della programmazione strutturata ereditata dal C (come gli if e i cicli). Come nel linguaggio C, JavaScript distingue le espressioni dagli statement. Una differenza sintattica tra C e JavaScript è il fatto che quando si programma in JavaScript si può omettere il punto e virgola a fine dichiarazione.

3.4.2 Typing dinamico

Come molti linguaggi di scripting, JavaScript è un linguaggio in cui il tipo è scelto dinamicamente, quindi i tipi sono associati ai valori e non alle variabili. Ad esempio una variabile che è dichiarata come un numero poi può essere dichiarata nuovamente come stringa. JavaScript include una funzione **eval** che può eseguire istruzioni fornite come stringhe a run-time.

3.4.3 Basato sui prototipi

JavaScript è un linguaggio basato sull'utilizzo degli oggetti estesi dai prototipi, utilizzati dove altri linguaggi orientati a oggetti usano le classi per

l'ereditarietà.
JavaScript supporta sia la dot notation che la notazione classica e la maggior parte delle proprietà dei valori possono essere aggiunte, modificate o eliminate a run-time.

3.4.4 Le Funzioni come oggetti

In JavaScript le funzioni vengono considerate oggetti e, come tali, possono avere proprietà e metodi. Le funzioni possono essere **innestate**, ovvero possono essere definite all'interno di altre funzioni andandole a creare ogni volta che viene invocata la funzione più esterna.

3.4.5 Esecuzione del codice

In JavaScript lato client, il codice viene eseguito direttamente sul client e non sul server, andando a non sovraccaricare il server anche con la presenza di script particolarmente complessi. Questo approccio porta però anche degli svantaggi, in caso di script con codice sorgente particolarmente grande il tempo per lo scaricamento può diventare troppo lungo e può andare a rovinare la user experience. Un altro svantaggio è quello che ogni informazione che presuppone un accesso a dati memorizzati in un database remoto deve essere rimandata ad un linguaggio che effettui in maniera esplicita la transazione, per poi restituire i dati ad una variabile JavaScript. Questi limiti sono stati superati grazie all'avvento di AJAX.

3.5 La programmazione in JavaScript

In questa sezione verrà analizzato il linguaggio JavaScript e le meccaniche che lo caratterizzano.

3.5.1 Le Variabili

In JavaScript, grazie alla tipizzazione dinamica dei dati, il tipo delle variabili viene determinato in base al loro valore. Ogni valore assume un determinato tipo che determina il ruolo del dato stesso. Esistono i seguenti tipi primitivi:

- **numero**: il valore di tipo numerico ha un valore numerico, senza distinzione fra valori numerici interi e reali;
- **stringa**: il valore di tipo stringa ha un valore letterale e consiste nell'essere una sequenza di carattere;
- **booleano**: il valore di tipo booleano è un tipo di dato che può assumere solo due valori;
- **null**: tipo di dato particolare, segnala che la variabile è vuota;

3.5.2 Il controllo di Flusso

Un **controllo di flusso** è un controllo che viene eseguito durante l'esecuzione di un programma, esso è caratterizzato da un'istruzione che controlla il verificarsi di una determinata condizione. In JavaScript le funzioni per il controllo del flusso sono:

- **If...Else**: operatore che controlla una determinata condizione: se è verificata esegue una determinata porzione di codice, se invece non è verificata non esegue nulla o un'altra determinata porzione di codice.
- **Ciclo For**: il ciclo for esegue ciclicamente le istruzioni al sui interno finchè non viene raggiungo il limite indicato dalla condizione di fine ciclo.
- **Ciclo While**: il controllo di flusso While esegue le istruzioni finchè la condizione è verificata.

- **Ciclo do...While**: il ciclo do...While è simile al ciclo While, solo che le istruzioni all'interno del ciclo vengono eseguite sicuramente almeno una volta.

3.5.3 Le Funzioni

La **funzione** è un particolare costrutto sintattico che permette di raggruppare, all'interno di un programma, una sequenza di istruzioni in un unico blocco, compiendo così una specifica operazione, azione (o elaborazione) sui dati del programma stesso in modo tale che, a partire da determinati input, restituisca determinati output. Il codice all'interno di una funzione sarà quindi eseguito ogni volta che verrà richiamata la funzione stessa. Una parte fondamentale di tutte le funzioni è la parola chiave **return** che determina il valore che la funzione restituisce e che corrisponde alla fine della funzione stessa, infatti ogni volta che il codice deve eseguire l'operazione return esce dalla funzione corrente e restituisce il valore determinato dall'espressione.

3.5.4 Gli Oggetti

Un **oggetto** è un insieme di valori che vanno a formare una struttura dati unica e tale da avere una particolare identità. Solitamente un oggetto viene utilizzato per rappresentare un'entità specifica tramite un'aggregazione di dati e di funzionalità. Tipicamente un oggetto possiede dei dati, detti **proprietà** e rappresentati da coppie nome-valore e delle funzionalità dette **metodi** e rappresentate da funzioni. In JavaScript gli oggetti non sono vincolati dalle classi e non sono quindi limitate a livello di struttura e comportamento.
L'ereditarietà in JavaScript è gestita tramite il **prototyping**. Ogni funzione JavaScript possiede un attributo prototype che si riferisce ad un oggetto prototype al quale è possibile aggiungere attributi e metodi. La funzione di questo oggetto è quella di fare da modello per la creazione di altri oggetti e ogni attributo e metodo aggiunto ad un oggetto prototype viene reso dispo-

nibile a tutti gli oggetti della funzione costruttore a cui l'oggetto prototype è attribuito.

3.5.5 Gli Eventi

JavaScript è un linguaggio di programmazione **Event driven**, il che significa che supporta il meccanismo degli eventi. Questa peculiarità permette di avere una reale interattività con il mondo esterno a differenza di un linguaggio che non li supporta, che segue percorsi fissi previsti dal programmatore. Utilizzando questo tipo di programmazione i browser generano eventi ogni volta che accade qualcosa ad un suo elemento, come il caricamento di una determinata risorsa o l'interazione con l'utente che clicca in un determinato punto o passa il mouse sopra un collegamento. Grazie alla diffusione dello standard **DOM (Document Object Model)** l'accesso agli elementi HTML da parte del codice JavaScript funziona in modo molto simile in tutti i browser, il che permette, tramite la combinazione dell'uso di DOM e della gestione degli eventi di JavaScript, la realizzazione delle applicazioni Web alle quali siamo abituati. Gli eventi sono caratterizzati da diversi aspetti, tra cui:

- **event type**: rappresenta la stringa che specifica il tipo di evento che si verifica. Ad esempio la string *keydown* serve a rappresentare l'evento della pressione di un tasto della tastiera;

- **event target**: rappresenta l'oggetto/elemento su cui si è verificato l'evento o anche a cui è associato l'evento. Quando si parla di un evento è necessario specificare quale sia l'event target associato a quell'evento;

- **event handler**: è una funzione che gestisce o risponde a un determinato evento andando a specificare come ci si dovrà comportare in base all'evento rivelato;

- **event object**: è un oggetto associato ad un evento il quale contiene informazioni sull'evento stesso. Gli event object vengono passati alle

funzioni per la gestione degli eventi e hanno una proprietà che specifica l'event type e una che specifica il destinatario dell'evento stesso;

- **event propagation**: è il processo mediante il quale il browser decide quale oggetto ha attivato l'event handler.

JavaScript utilizza il pattern **Observer** per la gestione degli eventi. L'aspetto fondamentale del funzionamento di questo pattern sta nell'uso delle funzioni di callback ogni volta che si verifica un evento su un determinato oggetto. Ciò significa che il codice delle applicazioni sarà affidato ad azioni attivate dal thread principale che ha anche la funzione di Observer.

Capitolo 4

ESP8266 e JavaScript

4.1 Introduzione a Espruino

Espruino è una soluzione per la programmazione della scheda ESP8266 che comprende un firmware interprete di JavaScript open-source per microcontrollori, creato per rendere lo sviluppo semplice e veloce. Il firmware interprete di Espruino è supportato da molti microcontrollori diversi e oltre ad aver la possibilità di flashare questo firmware sulle board supportate, è possibile anche comprare delle board con questo firmware già installato. Espruino non è però solamente il firmware interprete o la parte hardware, Espruino è composto anche da un IDE Web, dalla documentazione, da tutorial, da moduli e da uno strumento a riga di comando che vanno a formare una soluzione completa per lo sviluppo di software destinati all'utilizzo di ambienti embedded. Espruino risulta essere un'ottima alternativa a Raspberry Pi e ad Arduino nonostante le marcate differenze che ci sono tra i tre sistemi.
La differenze tra Espruino e Raspberry Pi sono diverse: Raspberry pi risulta essere molto flessibile e potente, ma la sua richiesta di un fabbisogno energetico relativamente alto lo rende difficile da utilizzare solamente con una fonte energetica come una batteria. Inoltre non è possibile pianificare delle azioni in real-time su un Raspberry Pi, il che significa che un tempismo preciso nel

quando effettuare determinate azioni risulta essere difficile. La potenza di Raspberry pi permette anche, ad esempio, la visualizzazione di video, cosa che Espruino non è in grado di fornire, in quanto non abbastanza potente. Espruino rende possibile l'utilizzo di JavaScript, ciò lo rende ottimo per chi è neofita risultando utile per chi è alle prime armi. D'altro canto Raspberry pi permette di scegliere tra diversi linguaggi di programmazione, rendendolo più versatile e potente ma scoraggiante.
Raspberry pi inoltre non supporta l'I/O analogico, cosa che invece Espruino fa. Un'ulteriore differenza tra Espruino e il Raspberry Pi è che il Raspberry necessita di una SD card contenente il sistema operativo mentre Espruino no.
Anche tra Espruino e Arduino ci sono alcune differenze: Epsruino risulta essere più piccola della maggior parte delle schede Arduino e risulta essere di più semplice utilizzo per chi si approccia per la prima volta a questo mondo. Nonostante Arduino consumi meno energia per funzionare, Espruino è stato creato tenendo sempre in considerazione l'ottimizzazione del consumo, infatti Espruino utilizza un sistema che permette di risparmiare energia: Espruino richiede fino a dieci volte meno energia quando viene impostato in sleep mode. L'utilizzo di un interprete di JavaScript significa che Espruino non necessita di essere resettato ogni volta che vengono effettuati dei cambiamenti al codice, ciò però significa anche che la velocità d'esecuzione risulta essere leggermente lenta.
Espruino risulta essere altamente efficiente per quanto riguarda l'ottimizzazione del consumo, essendo basato sugli eventi, l'interprete presente in Epsruino può autonomamente impostarsi in sleep-mode quando capisce che non ci sono azioni da effettuare. Questo significa che il codice scritto per Espruino sarà sostanzialmente più efficiente dello stesso codice scritto in C. Attualmente, quando è in sleep-mode, Espruino utilizza all'incirca un terzo della potenza che utilizza quando è occupato, invece quando è in modalità deep sleep arriva ad utilizzare addirittura solamente 20 microamps. Espruino attualmente è compatibile con JavaScript al 95% infatti implementa la

maggior parte dei subset delle specifiche JavaScript. Alcune funzionalità non sono implementate in quanto gli sviluppatori di Espruino sperano che non vengano mai utilizzate in quanto vanno contro ai principi della buona programmazione, altre invece verranno implementate in futuro, come le **regular expression** e **Unicode** (che non è stato implementato in quanto necessita di più memoria di quella presente). L'interprete di JavaScript presente in Epsruino implementa la maggior parte delle librerie standard, dovendo però escludere quelle meno utilizzate. Ciò è necessario in quanto non è presente abbastanza memoria per poterle includere tutte.

4.2 Impostare l'ambiente di lavoro

4.2.1 Installazione dei driver

Il primo passo da effettuare consiste nel rendere la board riconoscibile dal sistema operativo del proprio computer, per fare ciò è necessario installare i giusti driver. I driver utilizzati sono quelli sviluppati da ***Silicon Labs***, al momento della scrittura della tesi sono stati utilizzati i driver nella versione 6.7.5 su un sistema dotato di Windows 10 a 64 bit.
Questi driver sono necessari nello specifico per la comunicazione tra UART e USB, rendendo possibile la comunicazione tra board e computer. I driver sono reperibili al seguente link: `https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers`.

4.2.2 Installazione di Python

Il secondo passo da effettuare corrisponde nell'installare ***Phyton*** scaricabile dal sito `https://www.python.it/download/`. La versione installata è la 3.6.3 e dopo averlo installato è fondamentale impostare le variabili d'ambiente in modo che sia utilizzabile direttamente da linea di comando.

4.2.3 Installazione della libreria Pyserial

Il passo successivo consiste nell'installare ***Pyserial***, ovvero una libreria phyton che fornisce supporto per le connessioni seriali ("RS-232") su una varietà di dispositivi diversi: porte seriali vecchio stile, dongle Bluetooth, porte a infrarossi e così via. Supporta anche le porte seriali remote tramite RFC 2217 (dal V2.5). Pyserial è scaricabile al seguente url: `https://github.com/pyserial/pyserial`. Da prompt dei comandi bisogna spostarsi nella cartella in cui è stato scaricato Pyserial e poi procedere con il seguente comando:

```
python setup.py install
```

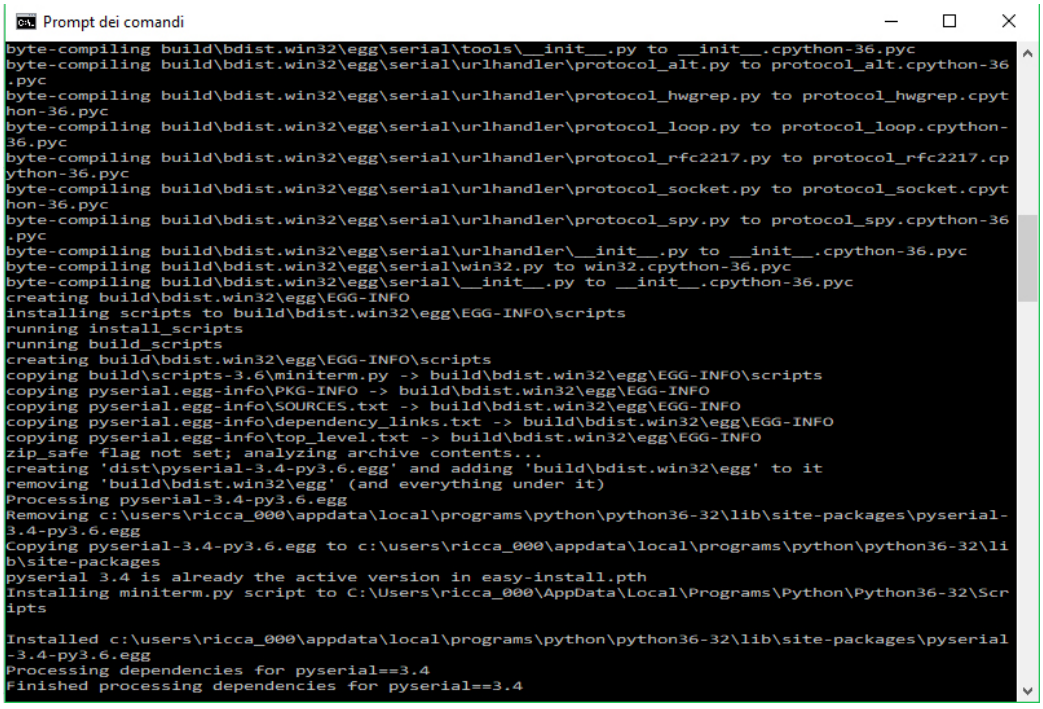

Figura 4.1: Schermata del risultato dell'installazione

4.2.4 Installazione di Espruino

Il prossimo passo prevede il download di Epsruino vero e proprio. Espruino può essere scaricato all'indirizzo: `https://www.espruino.com/Download` e attualmente la versione più recente risulta essere la versione 1.94. Successivamente è necessario scaricare uno strumento che verrà utilizzato per il flashing del firmware nella nostra board. Questo strumento si chiama ***EspTool*** ed è disponibile al seguente link: `https://github.com/espressif/esptool`. Una volta scaricati sia il firmware che il tool, tramite prompt dei comandi, bisogna spostarsi nella directory del firmware. A questo punto, prima di flashare il firmware, sarà necessario cancellare la memoria flash della board. Per fare ciò si utilizza il seguente comando:

```
python esptool/esptool.py --port COM4 erase_flash
```

Questo comando è diviso in diverse parti:

- **esptool.py**: seleziona lo strumento da utilizzare per effettuare questa operazione. Siccome l'operazione viene effettuata dentro un'altra cartella, è necessario specificare la directory del tool utilizzato;
- – **port COM4**: specifica la porta in cui verrà effettuata l'operazione di cancellazione della memoria;
- **erase_flash**: operazione che consiste nella cancellazione della memoria flash della board.

Una volta effettuata questa operazione, si passerà al flashing del firmware vero e proprio. Questo comando risulta essere più complicato dei precedenti ed è il seguente:

```
esptool.py --port COM4 --baud 115200 write_flash --flash_freq 80m
--flash_mode qio --flash_size 32m
0x0000 "boot_v1.6.bin" 0x1000 espruino_esp8266_user1.bin
0x3FC000 esp_init_data_default.bin 0x3FE000 blank.bin
```

Anche questo comando, come il precedente, è diviso in diverse parti:

- **esptool.py**: seleziona lo strumento da utilizzare per l'operazione di flashing;
- **–port COM4**: parametro che segnala la porta in cui verrà effettuata l'operazione;
- **–baud 115200**: specifica il baudrate con il quale stiamo flashando il firmware;
- **write_flash**: operazione che stiamo eseguendo
- **–flash_freq 80m flash_mode qio –flash_size 32m**: alcuni parametri che riguardano l'operazione di scrittura;
- la parte restante del comando riguarda gli indirizzi e i file che verranno scritti sulla board.

4.2.5 Installazione di Espruino Web IDE

L'ultimo passaggio da effettuare per iniziare la programmazione utilizzando JavaScript e Espruino riguarda l'utilizzo di un IDE. Espruino fornisce un IDE Web chiamato **Espruino Web IDE** che funziona tramite il browser Web **Google Chrome**. L'installazione dell'IDE avviene tramite l'aggiunta di un'estensione al browser chiamata come l'IDE stesso.
L'estensione è disponibile al seguente link: `https://chrome.google.com/webstore/detail/espruino-web-ide/bleoifhkdalbjfbobjackfdifdneehpo` ed è facilmente installabile.

4.3 Primi programmi

La prima operazione da fare, dopo aver aperto Espruino Web IDE consiste nell'impostare il baud rate a 11520 per permettergli di comunicare correttamente con la board. Questa operazione è possibile utilizzando il menù delle impostazione dell'IDE. Per effettuare il collegamento dei componenti alla board è necessario conoscere lo schema dei pin della board, che è diverso da quello che è scritto nella board. Ad esempio il pin **GPIO4** è collegato al pin 2 e viene chiamato nella board **D2**. Nel codice però, per comunicare a un determinato pin, bisogna usare la sigla formata da **D + numero del GPIO**. Ad esempio sempre per comunicare con un componente collegato nel pin D2 dovrò utilizzare la dicitura **D4**.

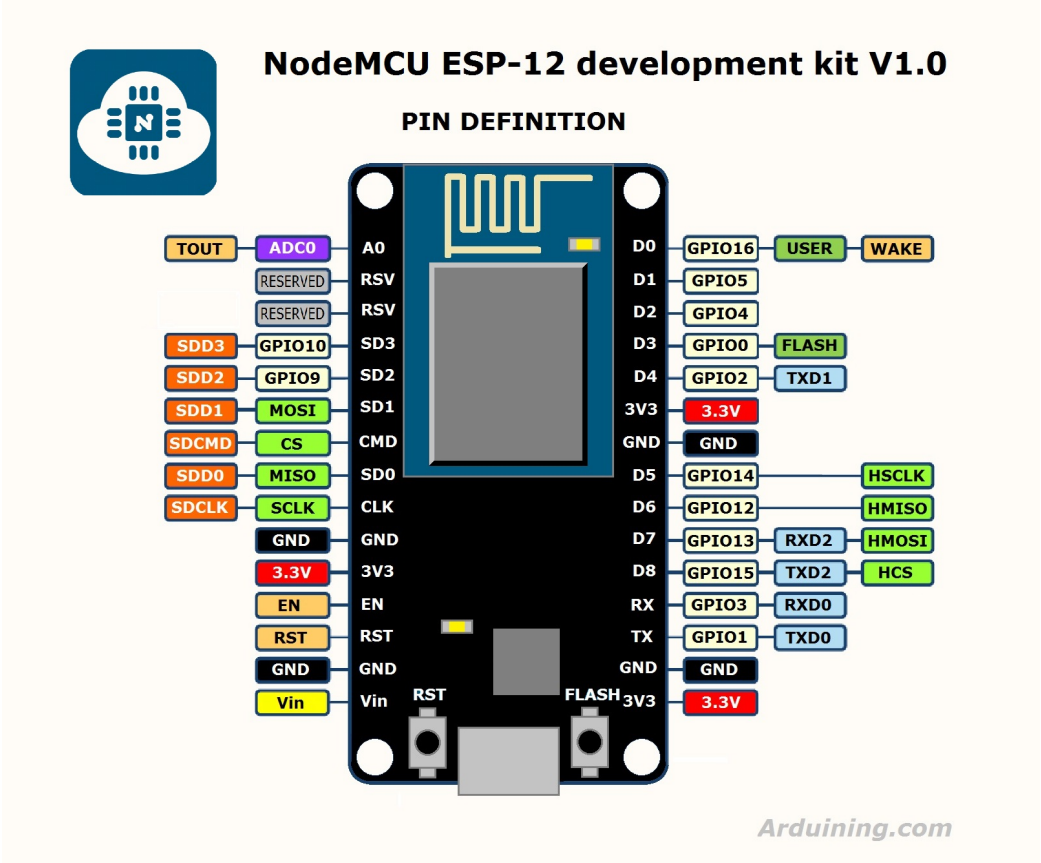

Figura 4.2: Schema dei pin della board ESP8266 con i relativi nomi

4.3.1 Lampeggiamento alternato di due led

Il primo programma per testare il funzionamento della board è quello che consiste nel lampeggiamento di due led.

Wiring

Il wiring è quello mostrato dall'immagine seguente:

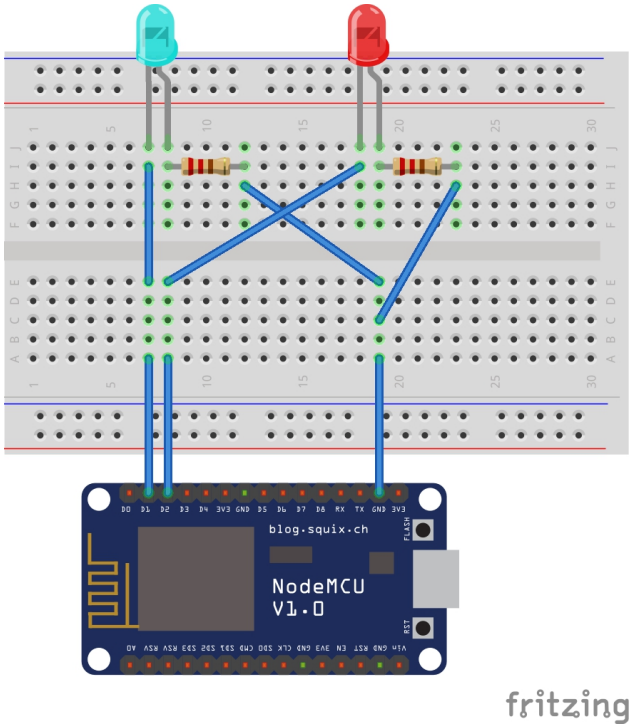

Figura 4.3: Wiring di due led con la board ESP8266

Codice

Il codice risulta essere molto semplice e permette a due led di lampeggiare in maniera alternata. Il codice è formato dalla dichiarazione di una funzione e dalla sua esecuzione.

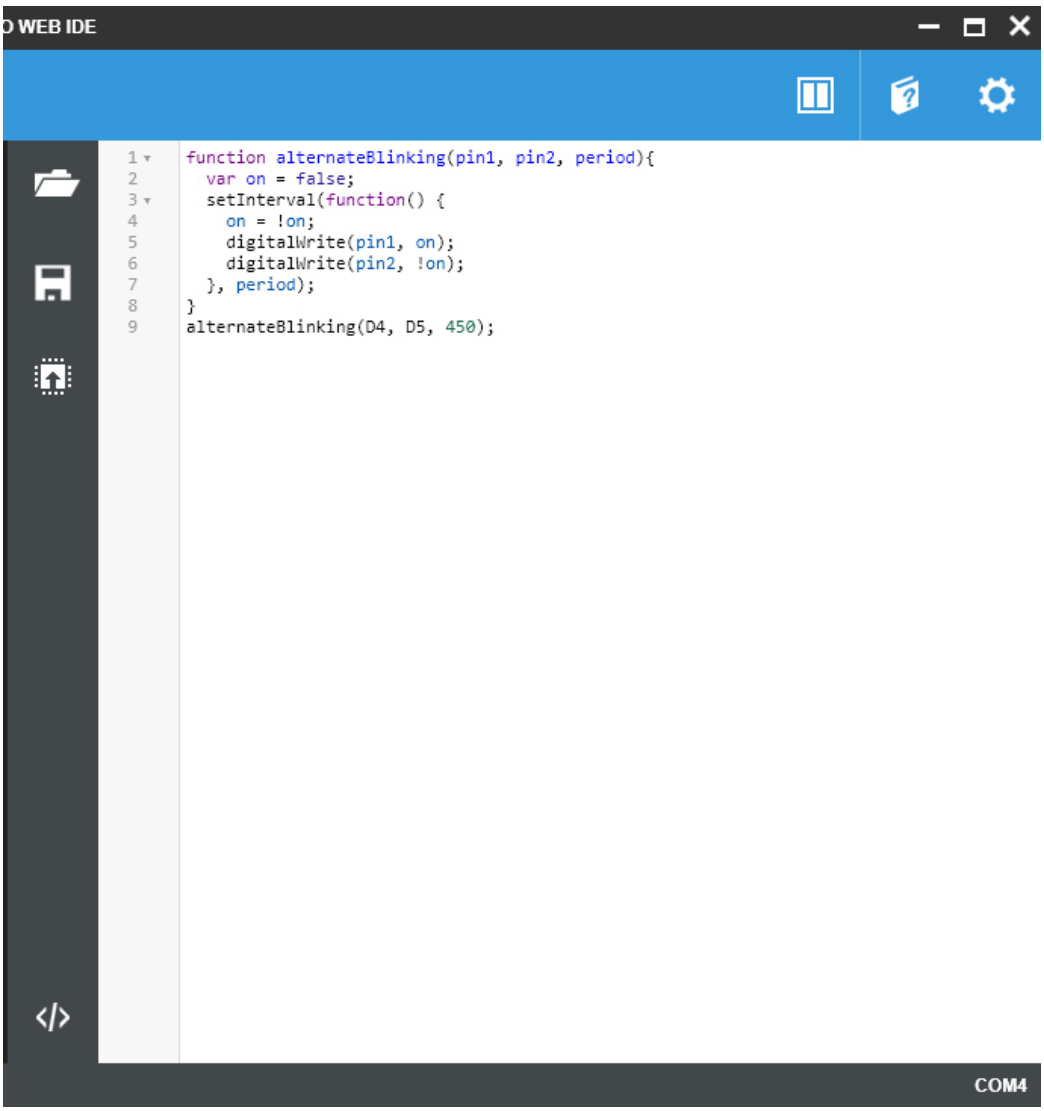

Figura 4.4: Codice per il lampeggiamento alternato di due led

La funzione dichiarata viene chiamata **alternateBlinking** e richiede tre argomenti: due pin e period. Gli argomenti pin che vengono passati alla fun-

zione corrispondono a quelli a cui sono collegati i due led. In questo caso sono i GPIO4 e GPIO5 che corrispondono a D1 e D2 nella board e a D4 e D5 nel codice. Il terzo argomento è period, un valore numerico che andrà a indicare il ritardo di esecuzione della funzione.
All'interno della funzione viene utilizzato ***SetInterval()***. Questa funzione permette di ritardare la funzione che gli viene passata come primo argomento per un determinato periodo.
All'interno della funzione passata a SetInterval viene richiamata per due volte la funzione ***DigitalWrite()***, funzione che si occupa di passare a un determinato pin un determinato valore. Questa funzione viene chiamata due volte in quanto vengono utilizzati due led e i valori passati saranno un booleano e il suo contrario. Facendo così ogni volta che un led sarà acceso l'altro risulterà spento.

4.3.2 Visualizzazione di messaggi su schermo LCD

In questo programma verrà collegato uno schermo LCD alla board ESP8266 che mostrerà un timer avanzare nel tempo di secondo in secondo e che illuminerà in alternanza due led.
Lo schermo verrà collegato utilizzato il protocollo **I2C**.

Wiring

Il wiring è quello mostrato dall'immagine seguente:

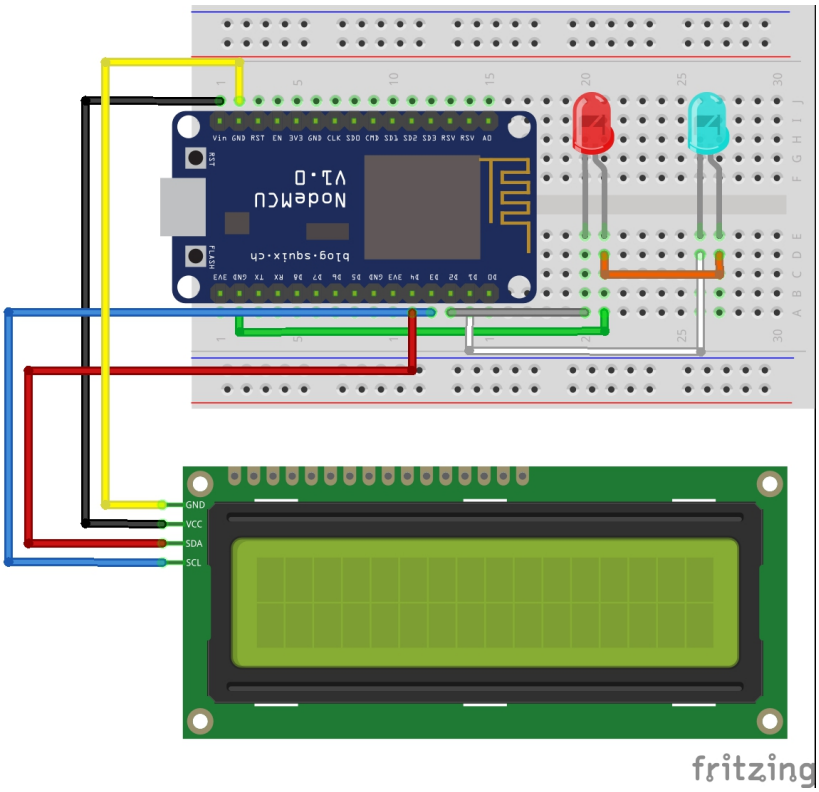

Figura 4.5: Wiring di due led e di uno schermo LCD con la board ESP8266

Codice

Il codice è molto semplice e intuitivo e simile a quello dell'esempio precedente.

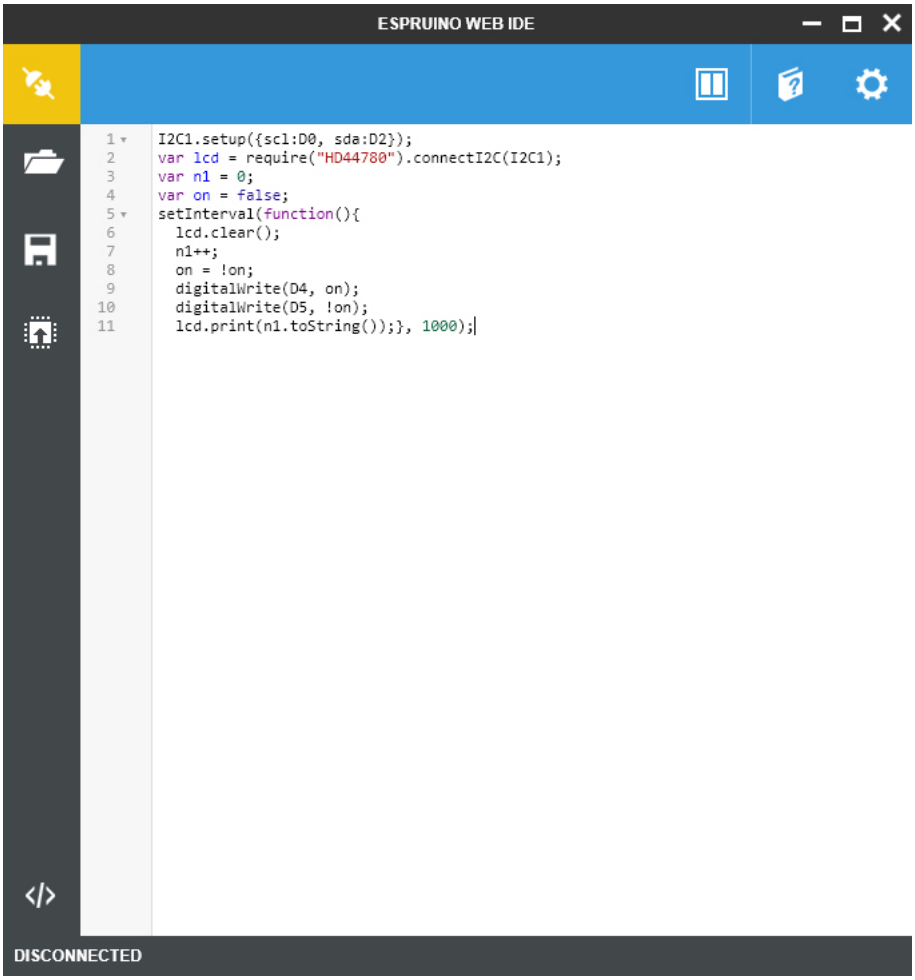

Figura 4.6: Codice per il lampeggiamento alternato di due led e conteggio dei secondi su schermo LCD

Nel codice viene usata la funzione ***require***. La funzione require accetta l'identificatore di un modulo (in questo caso "HD44780" che corrisponde allo schermo LCD) e ne restituisce l'API riguardante il modulo stesso.
Vengono utilizzate due funzioni che riguardano lo schermo: la prima è ***clear***

che si occupa di cancellare tutti i caratteri presenti nello schermo, la seconda è ***print*** che serve a stampare a video la stringa che gli viene passata.

4.3.3 Connessione della board alla rete WiFi

La board ESP8266 ha il modulo per la connessione alle reti WiFi incorporato. Utilizzando Espruino è quindi possibile connettere la board a una rete WiFi e utilizzarlo per operazioni riguardanti la rete WiFi.

Wiring

Il collegamento dei componenti è lo stesso di quello del programma precedente. In questo caso lo schermo verrà utilizzato per visualizzare l'indirizzo ip assegnato dalla rete alla board, mentre i led serviranno per segnalare se la board è connessa oppure no alla rete.

Codice

Il codice risulta essere leggermente più complicato dei precedenti, ma risulta comunque semplice capirlo.

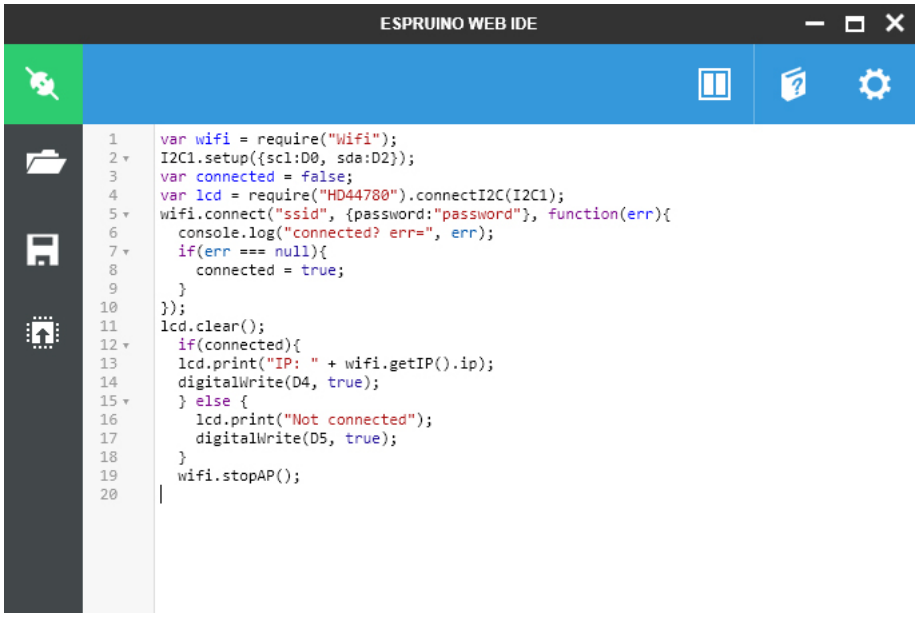

Figura 4.7: Codice per l'impostazione della connessione e relativo check tramite led e schermo LCD.

Conclusioni

In questo elaborato si è presentato il concetto di *Internet of Things*, assieme alla descrizione delle sue architetture e delle tecnologie che ne hanno permesso lo sviluppo e la popolarità che ha guadagnato nel tempo.
L'obiettivo dell'internet delle cose è fare in modo che il mondo elettronico tracci una mappa di quello reale dando un'identità elettronica alle cose e ai luoghi dell'ambiente fisico. Per fare ciò è fondamentale la Wireless Sensor Network, cioè una rete di sensori che permette di acquisire dati riguardanti l'ambiente circostante.
Un altro concetto che ha permesso lo sviluppo di Internet of Things è ciò che viene catalogato come Web Semantico.
Il Web Semantico, offrendo un framework comune che permette ai dati di essere condivisi e riutilizzati attraverso le applicazioni, permette di raggiungere il concetto di interoperabilità: ovvero la possibilità di utilizzare i dati in più sistemi, anche diversi tra loro. Ciò è possibile grazie all'utilizzo di determinati linguaggi, che formano le fondamenta del web semantico.
Questi linguaggi servono per strutturare i dati e permettono il processo di attribuzione di significati alle risorse, in modo di renderle comprensibili e interpretabili dai calcolatori.
Alcuni esempi di questi linguaggi risultano essere **RDF**, **OWL** e **SPARQL**.
Dopo aver analizzato il concetto di Internet of Things e le sue potenzialità, in questo elaborato sono stati analizzati le principali piattaforme hardware utilizzate in questo campo.
La diffusione di Internet of Things è possibile grazie al concetto di **System**

on a chip, ovvero circuiti integrati che contengono i componenti di un computer completo o di altri sistemi elettronici.
La popolarità di questi chip nel mobile computing è in continuo aumento anche grazie al fatto che i system on a chip richiedono un fabbisogno energetico limitato, ottimizzano le proprie risorse e hanno dimensioni contenute.
La prima piattaforma analizzata è stata **Arduino**.
Arduino è una piattaforma open-source composta da una serie di schede elettroniche dotato di un microcontrollore, ovvero un dispositivo elettronico integrato su un unico chip, progettato appositamente per interagire con input esterni, analogici o digitali, e restituire output analogici o digitali derivati dalle operazioni di processamento interno determinate da un programma caricato nella memoria del chip.
Arduino è una piattaforma estendibile tramite l'utilizzo delle **Shield**.
Le shield sono schede compatibili con alcuni modelli di Arduino che permettono di aggiungere funzionalità alla scheda stessa, come ad esempio la possibilità di connettere la scheda ad una rete WiFi, di collegare Arduino alla rete GSM o di controllare motori elettrici e servomotori. Come Arduino stesso, anche le Shield sono open-source ed è possibile scaricare e modificare a piacimento le shield.
La seconda piattaforma analizzata è stata **Raspberry Pi**.
Raspberry PI è un calcolatore implementato su una sola scheda elettronica ottima per la prototipazione di sistemi embedded avanzati.
La scheda è basata su un system on a chip Broadcom che incorpora un processore ARM, una GPU VideoCore IV e una quantità di memoria RAM che varia da modello a modello, partendo da un minimo di 254mb ed arrivando a un massimo di 1gb.
Come Arduino, Raspberry Pi è in grado di comunicare con l'ambiente che lo circonda grazie all'utilizzo di sensori e attuatori, elaborando input e restituendo output, ma, a differenza della board made in italy, Raspberry Pi, essendo un pc a se stante, ha la possibilità di montare un sistema operativo ed è in grado di eseguire le applicazioni che ci vengono intallate.

Raspberry pi è compatibile con molti sistemi operativi, ma il più utilizzato è **Raspbian**, un sistema operativo creato a hoc basato sulla distribuzione linux **Debian**.
La terza piattaforma analizzata in questo capitolo è la piattaforma chiamata **NodeMcu**, una piattaforma open-source sviluppata per l'ambiente IoT che include un firmware basato sulla scheda **ESP8266**.
ESP8266 è un chip WiFi a basso costo con supporto completo ai protocolli di internet TCP/IP e funzionalità da microcontrollore.
I componenti di questo chip sono: un microcontrollore a 32 bit, 64KiB di RAM dedicati alle istruzioni e 95KiB di RAM dedicati ai dati, una memoria flash che può andare da 512KiB a 4MiB e 16 pin GPIO.
Nonostante il firmware presente nella board preveda l'utilizzo del linguaggio ELua, è possibile cambiare firmware e utilizzare altri linguaggi come ad esempio JavaScript.
Il terzo capitolo è incentrato sul linguaggio **JavaScript**, un linguaggio che nasce come linguaggio web per rendere le pagine dinamiche, ma che ha riscosso molto successo anche fuori dall'utilizzo per il quale è stato pensato in quanto offre diversi vantaggi.
Essendo un linguaggio basato sugli eventi, JavaScript permette una certa interazione con l'utente andando a gestire i diversi input.
JavaScript supporta la maggior parte della programmazione strutturata ereditata dal C, ad esempio è possibile utilizzare i costrutti if e i cicli, ma a differenza del C, JavaScript è un linguaggio che utilizza il typing dinamico, ovvero ogni variabile dichiarata come numero poi può essere utilizzata come una stringa.
JavaScript è un linguaggio basato sugli oggetti che vengono estesi dai prototipi in maniera analoga a come gli altri linguaggi basati sugli oggetti utilizzano le classi. Questo permette a JavaScript di supportare l'ereditarietà. In JavaScript inoltre le funzioni vengono considerate oggetti e, come tali, possono avere proprietà e metodi.
L'ultimo capitolo di questo elaborato è dedicato a **Espruino** e al suo utilizzo.

Espruino è un firmware interprete di JavaScript open-source per microcontrollori supportato dalla board ESP8266 e permette la programmazione della board tramite l'utilizzo di JavaScript.
Espruino però non si limita ad essere un firmware, l'ambiente di programmazione è integrato grazie ad un IDE Web.
Espruino ha caratteristiche diverse in confronto ai sistemi analizzati precedentemente, la minor potenza di elaborazione in confronto al Raspberry Pi ne permette una versatilità maggiore, andando però a sacrificare alcune funzionalità. Un ulteriore differenza tra i due sistemi è quella che riguarda l'input/outuput analogico: mentre Raspberry non lo supporta, Espruino è in grado di supportarlo.
Anche tra Espruino e Arduino ci sono alcune differenze, nonostante Arduino consumi meno energia di Espruino, quest'ultimo utilizza un sistema per l'ottimizzazione delle risorse che gli permette di essere impostato in sleep-mode ed arrivare a consumare un decimo dell'energia utilizzata durante il normale utilizzo. Inoltre, grazie all'utilizzo di JavaScript, Espruino non necessita di essere resettato ogni volta che vengono effettuati dei cambiamenti al codice. Espruino al momento è compatibile con JavaScript al 95% e implementa la maggior parte delle librerie JavaScript.
In conclusione l'utilizzo di JavaScript in combinazione con dei sistemi embedded renderà possibile molte interazioni che, secondo me, permetteranno di raggiungere l'obiettivo stabilito da Internet of Things.

Bibliografia

[1] W3C - *http://www.w3.org/*

[2] Wikipedia - *http://it.wikipedia.org/*

[3] W3Schools.com - JavaScript - *http://www.w3schools.com/js/*

[4] David Flanagan: JavaSricpt The Definitive Guide, 6th Edition, O'Reilly, 2011

[5] JavaScript.HTML.it - *http://javascript.html.it/*

[6] RFID Journal - *http://www.rfidjournal.com/*

[7] W3C OWL - *https://www.w3.org/TR/owl2-overview/*

[8] W3C SPARQL - *https://www.w3.org/TR/sparql11-overview/*

[9] W3C RDF - *https://www.w3.org/TR/rdf-primer/*

[10] Arduino - *https://www.arduino.cc/*

[11] Raspberry PI - *https://www.raspberrypi.org/*

[12] Epsruino - *https://www.espruino.com/*

Ringraziamenti

Ci sono innumerevoli persone che vorrei ringraziare in quanto mi hanno sostenuto durante questo mio tortuoso percorso.
Per prima voglio ringraziare la mia famiglia: nonostante tutte le difficoltà incontrate e la voglia di mollare la mia famiglia mi ha sempre sostenuto, motivandomi e facendo sacrifici per permettermi di arrivare fino a qua.
Voglio ringraziare gli "Uomini a cavallo di draghi che lanciano lupi ai vermi" per tutti i momenti di svago che mi hanno regalato, permettendomi di non impazzire (del tutto).
Ringrazio anche chi ha condiviso con me questo percorso di studio e i viaggi della speranza in treno.
Ringrazio Beatrice Sharon per il supporto infinito nonostante la distanza, minacciandomi di botte ogni volta che pensavo di non potercela fare.
Ringrazio tutti i miei amici, che mi hanno davvero sostenuto e che non posso ringraziare uno ad uno, perchè se no i ringraziamenti diventerebbero più lunghi della tesi stessa.