UNIVERSITÀ DEGLI STUDI
DI GENOVA

DIPARTIMENTO DI INGEGNERIA NAVALE, ELETTRICA, ELETTRONICA E DELLE TELECOMUNICAZIONI

CORSO DI STUDIO IN INGEGNERIA ELETTRONICA E TECNOLOGIE DELL'INFORMAZIONE

Tesi di Laurea Triennale

Marzo 2020

Progettazione e implementazione su modulo Arduino di un datalogger per il monitoraggio di parametri cardiaci

Candidato: Giorgio Demarzi

Relatore:Prof. Riccardo Berta

Sommario

Lo scopo di questo lavoro di tesi è quello di illustrare il modulo progettato per monitorare parametri cardiaci quali: numero di battiti del cuore al minuto (Beats Per Minute - BPM), l' intervallo di tempo fra due battiti (Interbeat interval - IBI) e l'ampiezza massima della forma d'onda del battito (Pulse Amplitude - PA). Questi valori vengono salvati su una scheda di memoria e in un secondo momento vengono poi inviati ad una API (Application Programming Interface) chiamata "Atmosphere" che permette di gestire grandi quantità di misure provenienti da sensori. Infine, i dati vengono ordinati e incasellati in una interfaccia web basata su HTML, CSS e JavaScript.

Il progetto è stato sviluppato su piattaforma Arduino in un linguaggio di programmazione C-like che si basa sull'Arduino IDE, un software open-source che permette di scrivere e caricare codice sulla scheda omonima e altre compatibili.

1. Introduzione

Internet of Things (IoT) è un neologismo utilizzato in telecomunicazioni, nato dall'esigenza di dare un nome agli oggetti reali connessi ad internet. Nel concreto si indicano un insieme di tecnologie che permettono di collegare a Internet qualunque tipo di apparato. Lo scopo di questo tipo di soluzioni è sostanzialmente quello di monitorare e controllare e trasferire informazioni per poi svolgere azioni conseguenti.

L'IoT è una delle nuove frontiere dell'uso della rete internet. Non più solo le persone o le "persone giuridiche" o le imprese, sono riconoscibili sulla rete Internet, ma anche gli oggetti possono esserlo. Strumenti che acquisiscono intelligenza, ovvero capacità di rilevare informazioni e di comunicarle. L'IoT è una vera e propria Nuova Internet proprio perché apre prospettive un tempo inimmaginabili, in cui gli oggetti assumono un ruolo attivo grazie al fatto di essere in rete e di inviare e ricevere dati sulla rete.

Una videocamera, ad esempio, non è più solo nella condizione di inviare dati e immagini, ma è nella condizione di farlo in modo intelligente, in funzione delle immagini riprese, della temperatura o della luminosità. In questo modo è in grado di adattare il proprio comportamento in funzione di parametri di interesse che possono evolvere nel corso del tempo. Il nostro orologio può ricordarci appuntamenti e può verificare se effettivamente li rispettiamo, se siamo o non siamo in un certo luogo a una certa ora. Le confezioni di prodotti alimentari possono trasferire importanti informazioni sulla qualità del prodotto, sul modo in cui è stato realizzato e sui tutti coloro che hanno partecipato alla produzione. La confezione di un farmaco ci può avvertire se non lo stiamo assumendo come stabilito e ci può dire come ovviare e a una eventuale dimenticanza. Per non parlare poi delle automobili che possono dialogare costantemente con l'ambiente circostante e possono facilitare la nostra guida, aumentando comodità e sicurezza.

Le maggiori società di ricerca, come Accenture tra le altre, sostengono che si arriverà a oltre 25 miliardi di apparati IoT entro il 2020. Molti operatori del settore ritengono che il numero sarà ampiamente superato e già questo rappresenta una straordinaria opportunità di business per tutti gli operatori del settore. Autori quali Adrian McEwen (con il libro "Designing the Internet of Things") parlano di creatività e IoT, e di come le prossime idee e prodotti vincenti avranno bisogno di collegare oggetti della vita di ogni giorno con internet e con la tecnologia.[1]

La realizzazione di questo lavoro di tesi nasce dalla volontà di generare un oggetto "intelligente" in grado di monitorare alcuni parametri fisiologici del corpo umano, in particolare quelli cardiaci. La misura considerata è principalmente la frequenza cardiaca ovvero il numero di battiti del cuore al minuto (BPM). Rappresenta insieme alla temperatura corporea, la pressione sanguigna e il ritmo respiratorio una delle funzioni vitali ovvero quei valori che nell'individuo rappresentano la funzionalità dell'organismo.

Da qui ha origine l'idea dell'implementazione di un modulo capace di essere un data logger, ovvero un dispositivo in grado di registrare la frequenza cardiaca attraverso un sensore esterno, ed anche in grado di poter inviare questi valori ad un server per renderli disponibili per futuri utilizzi.

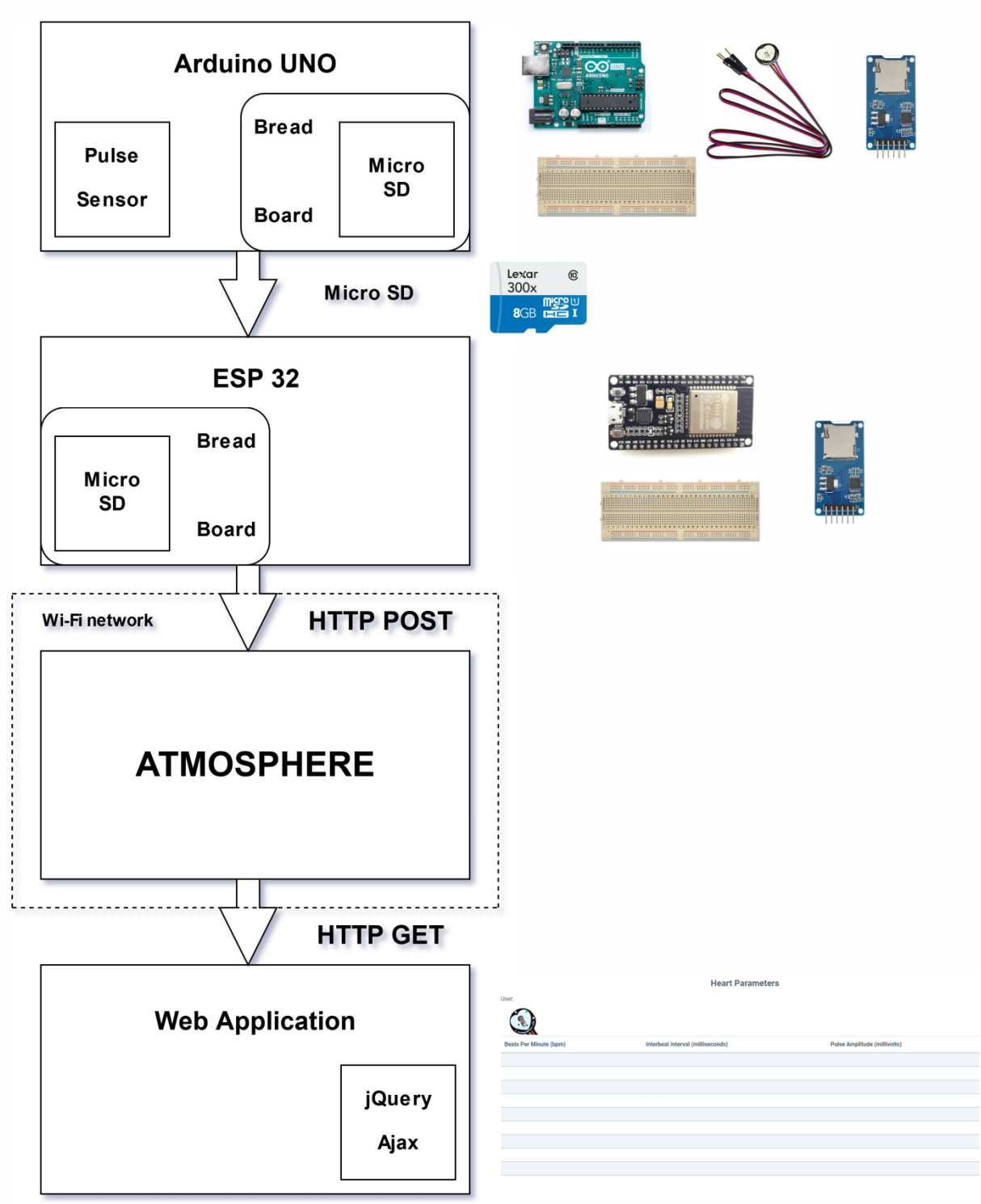
Il dispositivo hardware che meglio risponde alle esigenze richieste è la piattaforma open-source Arduino, sulla quale è montato un Pulse Sensor, in grado di rilevare l'informazione richiesta, ed anche un modulo per la lettura e scrittura dati su microSD, per poter memorizzare i valori acquisiti.

Tuttavia il modulo precedentemente descritto non è in grado di interfacciarsi ad una rete Wi-Fi e dialogare con un server. Si utilizza pertanto un ulteriore dispositivo, l'ESP32, pensato per essere un "ponte" per collegare Arduino alle reti Wi-Fi, sul quale viene montato il modulo microSD precedente. In questo modo è possibile instaurare una comunicazione HTTP fra il dispositivo e la Web Application Programming Interface "Atmosphere" ed è possibile inviare i dati rilevati, memorizzati nella microSD, sulle API.

Così facendo è possibile raccogliere queste informazioni in una web application ovvero un'applicazione fruibile via web per mezzo di un network, come ad esempio attraverso la Rete Internet, in una architettura tipica di tipo client-server, che offre determinati servizi all'utente client. In questo caso si parla di un'interfaccia web che comunica via HTTP con la Web API "Atmosphere" e che quindi permetta di vedere le misure raccolte incasellate e ordinate.

2. Metodi e strumenti utilizzati

Digramma del sistema completo:

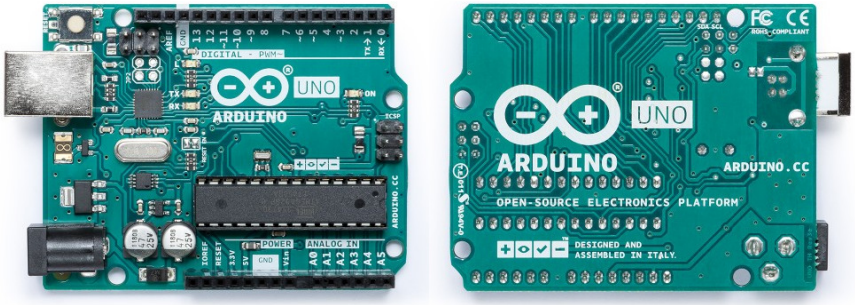


Il progetto è costituito da due moduli fisici e una interfaccia Web:

- Modulo Arduino UNO con Pulse Sensor e MicroSD Card Adaptor adibito alla rilevazione e alla memorizzazione dei dati fisiologici di interesse ovvero il numero di battiti del cuore al minuto (Beats Per Minute - BPM), l'intervallo di tempo fra due battiti (Interbeat interval - IBI) e l'ampiezza massima della forma d'onda del battito (Pulse Amplitude - PA).
- Modulo ESP32 con MicroSD Card Adaptor adibito a stabilire una comunicazione HTTP fra il dispositivo e la Web Application Programming Interface "Atmosphere".
- Interfaccia Web basata su HTML, CSS e JavaScript adibita a ordinare, incasellare e rendere disponibili i dati raccolti.

Descrizione dei componenti:

Arduino UNO



Arduino è una piattaforma open-source basata su risorse hardware e software pensati per essere flessibili e di immediato impiego. È stata ideata e sviluppata nel 2005 da alcuni membri dell'Interaction Design Institute di Ivrea come strumento per la prototipazione rapida e per scopi hobbistici, didattici e professionali.

La scheda Arduino UNO si basa su un circuito stampato che integra un microcontrollore con dei pin connessi alle porte I/O, un regolatore di tensione e un' interfaccia USB che permette la comunicazione con il computer utilizzato per programmare. A questo hardware viene affiancato un ambiente di sviluppo integrato (IDE) multipiattaforma. Questo software permette anche ai novizi di lavorare con Arduino, in quanto i programmi sono scritti in un linguaggio di programmazione semplice e intuitivo, chiamato Wiring, derivato dal C e dal C++, liberamente scaricabile e modificabile.

Un programma Wiring viene chiamato sketch, ed è normalmente costituito da tre parti principali:

- Introduzione: La prima parte è adibita alla dichiarazione delle librerie, delle variabili, delle costanti e per la codifica delle routine ovvero delle porzioni di codice che vengono eseguite solo quando vengono richiamate da una specifica istruzione. Vi è inoltre la dichiarazione degli oggetti utilizzati, ad esempio display a cristalli liquidi, servomotori o come in questo caso un sensore di pulsazioni. Infine possono essere presenti note esplicative sulle finalità e sul funzionamento del programma.
- void setup(): La seconda parte o parte di setup, viene eseguita solo all'avvio del programma. È adibita ad eventuali istruzioni da eseguire una sola volta o a definire delle impostazioni che non verranno più cambiate nel corso della sua esecuzione, ad esempio l'inizializzazione delle variabili o la dichiarazione delle porte di input e di output .
- void loop() : La terza parte o parte di loop, rappresenta la parte principale del programma, che viene eseguita e ripetuta fino al termine dell'alimentazione o fino alla pressione del pulsante reset. Contiene le istruzioni e la logica del programma.[3]

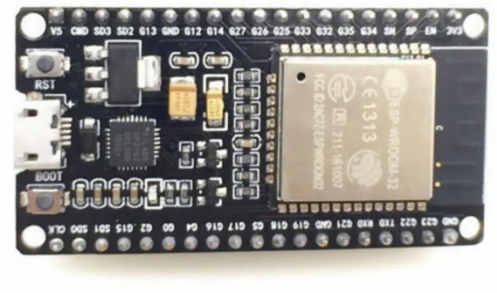
Pulse Sensor

Il Pulse Sensor è un sensore plug-and-play progettato e realizzato per Arduino dalla WorldFamous Electronics. Viene utilizzato per poter incorporare i dati relativi alla frequenza cardiaca all'interno del progetto.

Il sensore deve essere applicato sul polpastrello della mano e si collega direttamente all'Arduino attraverso i cavetti. Può lavorare con un'alimentazione sia a 3 Volt sia a 5 Volt e nonostante sia pensato per Arduino è compatibile con qualsiasi MCU con un ADC (ADC - Analog to Digital Conversion).

Il Pulse Sensor presenta una ben documentata libreria per l'Arduino IDE che ne permette di sfruttare le sue features: in particolare vengono utilizzate quelle relative al monitoraggio del numero di battiti del cuore al minuto (Beats Per Minute - BPM), dell' intervallo di tempo fra due battiti (Interbeat interval - IBI) e dell'ampiezza massima della forma d'onda del battito (Pulse Amplitude - PA).[4]

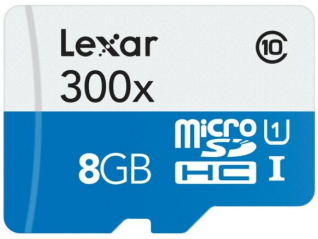
ESP32



ESP32 è una serie di microcontrollori a basso costo e a basso consumo con Wi-Fi integrato e Bluetooth dual-mode. Sviluppato da Espressif Systems, ha avuto successo come "ponte" per collegare diversi microcontrollori (principalmente Arduino) alle reti Wi-Fi grazie ad un costo molto più accessibile rispetto alle alternative.

La scheda ESP32 si basa, analogamente all' Arduino UNO, su un circuito stampato che integra un microcontrollore con dei pin connessi alle porte I/O, un regolatore di tensione e un' interfaccia USB che permette la comunicazione con il computer utilizzato per programmare. È in grado di garantire, inoltre, una connettività Wi-Fi e una Bluetooth. L'hardware presente sulla scheda è compatibile con il già citato Arduino IDE e quindi il linguaggio di programmazione per questo ambiente di sviluppo è lo stesso Wiring.[5]

Scheda MicroSD



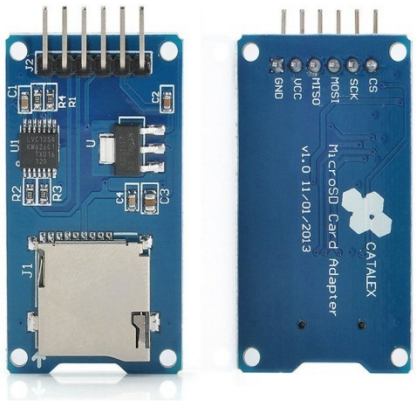
La microSD (Micro Secure Digital) è una scheda di memoria dalle dimensioni estremamente ridotte. Sono state realizzate pensando soprattutto ad espandere la memoria degli smartphone i quali necessitano leggerezza e bassi consumi. In questo progetto la microSD viene utilizzata per poter memorizzare i dati rilevati dal Pulse Sensor.

La scheda SD presenta nove contatti striscianti che permettono di trasferire alla microSD l'alimentazione e tutti i segnali necessari per la scrittura e la lettura dei dati.

In riferimento alla comunicazione SPI (Serial Peripheral Interface), che è quella utilizzata dal modulo microSD (Catalex MicroSD Card Adaptor), i vari pin hanno la seguente funzione:

- PIN SS: È il PIN di selezione del dispositivo, letteralmente indica Slave Select; quando si vuole andare a leggere o scrivere sulla SD si va a porre questo PIN a livello logico basso. È molto utile quando vi sono più schede SD sullo stesso bus di comunicazione SPI.
- PIN MOSI, SCK e MISO: Sono i tre PIN usati per la comunicazione SPI e sono rispettivamente il PIN dei dati in ingresso al modulo (Master Output Slave Input - dove il master è il microcontrollore, lo slave il modulo SD), il PIN di clock del protocollo SPI (Serial ClocK, fornito dal master) e il PIN di output del modulo (Master Input Slave Output).
- PIN GND: È il PIN dell'alimentazione negativa.
- PIN VCC: È il PIN dell'alimentazione positiva.[6]

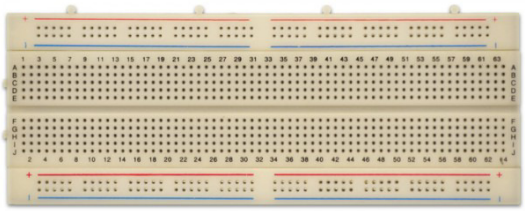
Catalex MicroSD Card Adaptor



Il Catalex MicroSD Card Adaptor è un modulo per la lettura e scrittura dati su microSD. Si tratta di un dispositivo relativamente economico che può essere utilizzato per qualsiasi microcontrollore, come Raspberry PI o Arduino.

La scheda presenta l'alloggiamento della scheda SD, la quale viene connessa con i contatti striscianti verso il basso, e rende disponibile i PIN nella parte inferiore del chip. Sono presenti un PIN di alimentazione positiva (VCC) e uno negativa (GND), i tre PIN della comunicazione SPI (MISO, MOSI, SCK) e infine vi è il PIN di Slave Select (SS). La scheda è fornita di un regolatore di tensione LDO per abbassare i 5 Volt ai 3.3 Volt necessari per il funzionamento del chip. [7]

Breadboard



La Breadboard o basetta sperimentale è uno strumento utilizzato per creare prototipi di circuiti elettrici. A differenza della basetta millefori, che è un circuito stampato su cui vengono saldati i componenti e i collegamenti che formano il prototipo, la breadboard non richiede saldature ed è completamente riutilizzabile.

La semplicità, la velocità di impiego e la buona affidabilità ne fanno uno strumento indispensabile in un laboratorio elettronico, in fase di elaborazione e studio del progetto, per provare il corretto funzionamento dei circuiti in esame.

Con questa tecnica di montaggio è possibile realizzare circuiti semplici, ma anche complessi.[8]

Web Application Programming Interface "Atmosphere"

Atmosphere è una Application Programming Interface (API) per un web server: si tratta di un insieme di interfacce che consentono l'accesso alle risorse del server garantendo sicurezza e controllo.

In generale le API sono degli strumenti di programmazione che le maggiori software house e industrie del mondo informatico, come ad esempio Microsoft e Google, mettono a disposizione degli sviluppatori per facilitare il loro compito nella realizzazione di applicazioni di vario genere. Possono essere delle librerie di funzioni che permettono al programmatore di interagire con un programma o una piattaforma software o semplicemente una serie di "chiamate" a parti di un programma che uno sviluppatore può utilizzare per abbreviare il suo lavoro.

Le API, quindi, sono delle interfacce grafiche che programmatori terzi possono utilizzare per espandere le funzionalità di programmi, applicazioni e piattaforme di vario genere. Rappresentano, quindi, il portale attraverso il quale interagire con programmi altrimenti inaccessibili.

Si può affermare quindi che utilizzando questi “artifici” di programmazione, si possono estendere le funzionalità di una risorsa ben oltre le reali intenzioni dello sviluppatore o della software house che l'ha realizzato.

Sulla base di quanto detto le API rappresentano, di fatto, un materiale davvero prezioso nel campo della programmazione sia per gli sviluppatori che le utilizzano sia per le software house che le rilasciano. Per questo la loro distribuzione avviene solamente attraverso canali ufficiali e, in alcuni casi, in forma ristretta.

Atmosphere è una Web API basata sullo stile software Representational State Transfer (REST). REST supporta le operazioni CRUD (Create, Read, Update and Delete) su HTTP ed è approvato dall'organizzazione Internet Engineering Task Force (IETF). Un'API RESTful separa l'interfaccia utente dal server e dall'archiviazione dei dati, il che migliora la portabilità dell'interfaccia, aumenta la scalabilità dei progetti e consente l'evoluzione indipendente dei componenti di sviluppo.

Come tutte le API Web RESTful, si basa su metodi HTTP per accedere alle risorse URL-encoded e utilizza il linguaggio JSON (JavaScript Object Notation) per trasmettere dati. Si tratta di una API pensata per gestire grandi quantità di dati provenienti da sensori. Presenta quattro risorse principali:

- THING: Rappresenta l'entità (una persona, un ambiente, un dispositivo ecc) oggetto di misura.
- FEATURE: Rappresenta una grandezza fisica misurata da un dispositivo (la temperatura, l'umidità, il peso, ecc).
- DEVICE: Rappresenta un dispositivo hardware o software che può misurare una certa grandezza fisica (FEATURE) su una determinata entità (THING).
- MEASUREMENT: Rappresenta una misura effettuata da un dispositivo (DEVICE) per una grandezza fisica (FEATURE) su una determinata entità (THING).

In memoria, le risorse corrispondono alle tabelle nel database. Ogni risorsa costituisce un numero di campi, alcuni sono obbligatori mentre altri sono facoltativi. I campi corrispondono agli attributi della tabella nel database. All'interno delle API, le risorse sono modellate in due fasi: lo schema che definisce la struttura delle risorse e il controller che ne definisce la sua funzionalità. Uno schema di risorse descrive i campi inclusi i loro tipi, i valori predefiniti ed i riferimenti ad altri campi di risorse. I campi che fanno riferimento ad altre risorse hanno funzioni di convalida implementate nel relativo schema. Esso include inoltre definizioni di plug-in e opzioni di indicizzazione. Il controller definisce i metodi HTTP supportati dalla risorsa. I metodi principali sono GET, POST, PUT e DELETE. Il metodo GET è rappresentato nel controller da due funzioni asincrone: "get" che supporta le query di filtro per recuperare più record e "getone" che recupera un record dal suo ID. Il metodo POST è responsabile dell'inserimento di nuovi record, mentre il metodo PUT permette di aggiornare i record esistenti. Infine, il metodo DELETE rimuove i record dal database.

Ogni funzione , fra quelle precedentemente elencate, è asincrona, accetta una richiesta HTTP come input, esegue una funzione specifica e quindi invia una risposta all'IP richiedente.

Quando l'API viene inizializzata per la prima volta, si connette al server di archiviazione e crea il database "Atmosphere-DB" con una raccolta all'interno, definito gruppo "users". È presente un utente amministratore, predefinito nel codice sorgente, per poter creare altri "user" e altre raccolte. L'utente "admin" può creare altri due tipi di utenti: "authority" e "regular". Mentre l' "admin" ha pieno accesso al DB con tutti i metodi senza restrizioni, l'utente "authority" ha un accesso limitato ai record di sua proprietà e l'utente "regular" ha accesso limitato che include solo i metodi GET. Queste tre differenti tipologie sono adatte all'utente previsto: un "admin" è il principale operatore dell'API, l' "authority" è un fornitore di servizi o un partner selezionato che utilizza l'API mentre l'utente "regular" rappresenta il pubblico interessato. Questo meccanismo di accesso viene chiamato "Authorization".

Un'altra misura di sicurezza supportata dall'API è l'autenticazione. È basata su un JSON Web Token (JWT) come pass di sicurezza che scade ogni 30 minuti. Per ottenere un JWT, gli utenti devono effettuare un POST la risorsa "login" con le credenziali fornite. L'API invierà una risposta contenente un JWT con il livello di autorizzazione specifico dell'utente richiedente. Il JWT deve quindi essere utilizzato come attributo di intestazione per eventuali ulteriori richieste all'API.

In questo progetto vengono misurate tre FEATURES: il numero di battiti del cuore al minuto (Beats Per Minute - BPM), l'intervallo di tempo fra due battiti (Interbeat interval - IBI) e l'ampiezza massima della forma d'onda del battito (Pulse Amplitude - PA). Vengono chiamate rispettivamente "heart-rate" (misurata in "beats-per-minute"), "interbeat-interval" (misurata in "milliseconds") e "pulse-amplitude" (misurata in "millivolts").

È presente un DEVICE, il cui nome è "pulse-sensor" in grado di fare delle misure sulle tre FEATURES create in precedenza ed inoltre vi sono due THINGS chiamate "giovanni.bianchi" e "carlo.verdi", taggati come "human", che rappresentano gli utenti che usando il dispositivo acquisiscono le misure interessate.[9]

HyperText Transfer Protocol (**HTTP**)

L'HyperText Transfer Protocol (HTTP) è un protocollo usato come principale sistema per la trasmissione d'informazioni sul web ovvero in un'architettura tipica client-server. HTTP è senza stato: la connessione viene chiusa una volta terminata lo scambio richiesta/risposta e non vengono mantenuti dati di sessione o altro. La comunicazione tra il browser e il server avviene in due tempi attraverso un HTTP request e un HTTP response.

Una richiesta HTTP è composta dalle seguenti parti:

- Metodo HTTP: I metodi HTTP definiscono i modi in cui il client può interagire con la risorsa. Ne esistono due principali: il metodo GET che recupera una risorsa dal server e il metodo POST che invia una risorsa al server. Ne esistono altri definiti dalle specifiche del protocollo HTTP, ma molti di essi sono poco utilizzati o non supportati infatti, molti browser moderni supportano solo GET e POST.
- Request header: Le intestazioni di richiesta forniscono molte informazioni supplementari come l'host (Host), i formati di risposta accettati dal client (Accept) e l'applicazione utilizzata dal client per effettuare la richiesta (User-Agent).
- URL: L'URL è l'indirizzo o il percorso univoco che identifica la risorsa che il client desidera. Il metodo HTTP definisce ciò che il client vuole fare con la risorsa.
- Body:Il Body rappresenta il corpo della richiesta.

Una risposta HTTP è composta dalle seguenti parti:

- Versione: Definisce la versione di HTTP utilizzata.
- Codice di stato: Il codice di stato comunica il risultato della richiesta al client. Esistono diversi codici di stato che indicano il successo, un errore o un'azione che il client deve compiere, ad esempio reindirizzare l'utente a un'altra pagina. Ne esistono quattro principali: il codice 200 indica che la richiesta è andata a buon fine, il codice 301 indica che la risorsa richiesta è stata spostata definitivamente ad un nuovo URI, il codice 404 indica che la risorsa richiesta non è stata trovata ed infine il codice 500 indica un errore generico causato solitamente da una configurazione errata del server.
- Response header: Le intestazioni di risposta forniscono molte informazioni supplementari come il tipo di codifica utilizzata sui dati dal server (Content-Encoding), il tipo dei dati (Content-Type), la data e l'orario in cui i dati sono stati inviati (Date) e il nome del server web utilizzato (Server).
- Body: Il Body rappresenta il corpo della risposta

In questo progetto viene formulata un HTTP POST nel codice dell'ESP32 col fine di inviare i dati rilevati dal Pulse Sensor alla Web Application Programming Interface "Atmosphere". L'HTTP Request è formulata con questo template:

Metodo HTTP: POST

Headers:

Content-Type: application/json

Authorization: Token di autorizzazione di "Atmosphere"

URL: http://test.atmosphere.tools/v1/measurements

Body:

```
{
        "thing": "mario.rossi", (in a
        "feature": "pulse-amplitude",
        "device": "pulse-sensor",
        "values": [             { "value": ["15"], "delta": "0" },
                                { "value": ["20"], "delta": "0" },
                                { "value": ["17"], "delta": "0" },
                                { "value": ["18"], "delta": "0" }
                ]
}
```

Ovviamente il valore del campo "thing" varia a seconda della persona oggetto della misura (“giovanni.bianchi” e “carlo.verdi”) mentre il valore di "feature" varia con la grandezza fisica misurata ("heart-rate", "interbeat-interval" e "pulse-amplitude").

Successivamente viene formulata un HTTP GET nel codice JavaScript della Web Application col fine di ottenere le misure rilevate per poterle incasellare all'interno di una tabella. L'HTTP Request è formulata con questo template:

Metodo HTTP: GET

Query Params:

filter: {"thing":"mario.rossi"}

limit: 10

page: 1

Headers:

Authorization: Token di autorizzazione di "Atmosphere"

charset: UTF-8

URL: http://test.atmosphere.tools/v1/measurements

Ovviamente il valore del campo "filter" dei Query Params varia a seconda della persona oggetto della misura (“giuseppe.bianchi” e “mario.rossi”).[10]

Web Application

Le Web Application sono applicazioni accessibili via web grazie ad un network, si tratta quindi di software che non necessitano di essere installati nel nostro computer**.** Non si tratta infatti di software che risiedono fisicamente nel pc, ma sono raggiungibili attraverso internet da qualsiasi terminale o dispositivo.

L'utilizzo delle Web Application è ormai ampiamente radicato, infatti un qualsiasi Social Network come Facebook, Youtube, oppure Gmail rappresentano esempi comuni di applicazioni web.

Un'applicazione di questo genere porta con se numerosi vantaggi:

- L'uso dell'applicativo non dipende dal sistema operativo (Windows, Mac, Linux, Android,...ecc)
- Gli aggiornamenti sono disponibili senza dover fare installazioni o upgrade
- L'accesso è immediato: non si deve scaricare alcun software
- I dati sono centralizzati: i database sono su un server a cui si accede con una connessione internet o intranet e i dati si raggiungono tramite un semplice browser anche device (smartphone o tablet)
- Massima sicurezza: i dati risiedendo su un server esterno non rischiano di essere attaccati da virus presenti sul client
- Maggiore risparmio su eventuali costi infrastrutturali
- Estrema flessibilità e semplicità: l'utente può interagire da qualsiasi sede, senza doversi limitare ad un un'unica postazione di lavoro.
- Il software può essere utilizzato da più utenti contemporaneamente[11]

Quando un utente accede ad una Web Application, l'impaginazione del testo e delle immagini che compaiono nel browser è spesso creata utilizzando un semplice linguaggio noto come Hyper Text Markup Language (HTML). La porzione di testo che è "delimitata" dall'HTML viene trasferita dal sito web al browser dell'utente. Il browser interpreta questo testo, mostrando testo ed immagini all'utente. La porzione di testo che viene trasferita è tipicamente chiamata pagina web.

La formattazione del documento HTML è definita dal linguaggio Cascading Style Sheets (CSS). L'introduzione del CSS si è resa necessaria per separare i contenuti delle pagine HTML dalla loro formattazione o layout e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine stesse sia per gli utenti, garantendo contemporaneamente anche il riutilizzo di codice ed una sua più facile manutenzione.

Lo scripting del documento HTML è definito dal linguaggio JavaScript. La sua introduzione si è resa necessaria per introdurre uno script all'interno di un'altro programma, ovvero permettere la scrittura di funzioni integrate per compiere determinate azioni altrimenti non possibili con il solo HTML statico.

Il programma che ospita ed esegue lo script fornisce allo stesso un'API ben definita, che consente l'accesso ad operazioni specifiche, la cui implementazione è a carico del programma ospite stesso. Lo script, quando eseguito, utilizza riferimenti a questa API per richiedere al programma ospite l'esecuzione di operazioni specifiche, non previste dai costrutti del linguaggio JavaScript in sé. In effetti, questo è esattamente lo stesso meccanismo che viene adottato anche in un linguaggio quale il C o Java, nel quale il programma si affida a delle librerie, non previste dal linguaggio in sé, che permettono di effettuare operazioni quali l'I/O o l'esecuzione di chiamate a funzioni di sistema.

L'esempio tipico di programma ospite per uno script JavaScript è quello del browser. Un browser tipicamente incorpora un interprete JavaScript; quando viene visitata una pagina web che contiene il codice di uno script JavaScript, quest'ultimo viene portato in memoria primaria ed eseguito dall'interprete contenuto nel browser.

Le caratteristiche principali di JavaScript sono:

- l'essere un linguaggio interpretato: il codice non viene compilato, ma interpretato (in JavaScript lato client, l'interprete è incluso nel browser che si sta utilizzando).
- la sintassi è relativamente simile a quella del C, del C++ e di Java.
- definisce le funzionalità tipiche dei linguaggi di programmazione ad alto livello (strutture di controllo, cicli, ecc.) e consente l'utilizzo del paradigma object oriented.
- è un linguaggio debolmente tipizzato
- è un linguaggio debolmente orientato agli oggetti.[12]

jQuery AJAX

jQuery è una libreria JavaScript per applicazioni web. Nasce con l'obiettivo di semplificare la selezione, la manipolazione e la gestione degli eventi, nonché implementare funzionalità AJAX.

AJAX (Asynchronous JavaScript and XML) è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. AJAX è, per definizione, asincrono nel senso che i dati extra sono richiesti al server e caricati in background senza interferire con il comportamento della pagina esistente. Normalmente le funzioni richiamate sono scritte con il linguaggio JavaScript. Tuttavia, e a dispetto del nome, l'uso di JavaScript e di XML non è obbligatorio, come non è detto che le richieste di caricamento debbano essere necessariamente asincrone. L'utilizzo di questa tecnica di programmazione si è molto diffuso negli ultimi anni, soprattutto con l'avvento delle nuove applicazioni del Web 2.0 per la cui realizzazione viene chiesto agli sviluppatori di superare i limiti del "web tradizionale" realizzando pagine web sempre più simili alle applicazioni che vengono installate sui computer. Questo tipo di approccio di sviluppo richiede al programmatore un notevole sforzo e la scrittura di molto più codice rispetto al solito. Senza un framework di supporto, inoltre, questa pratica può dirsi proibitiva per chiunque non sia un vero esperto della materia.

Il metodo jQuery.ajax() (oppure in alternativa $.ajax()), che viene applicato direttamente all'oggetto jQuery, permette di effettuare una chiamata AJAX e di personalizzarla attraverso i molti parametri disponibili:

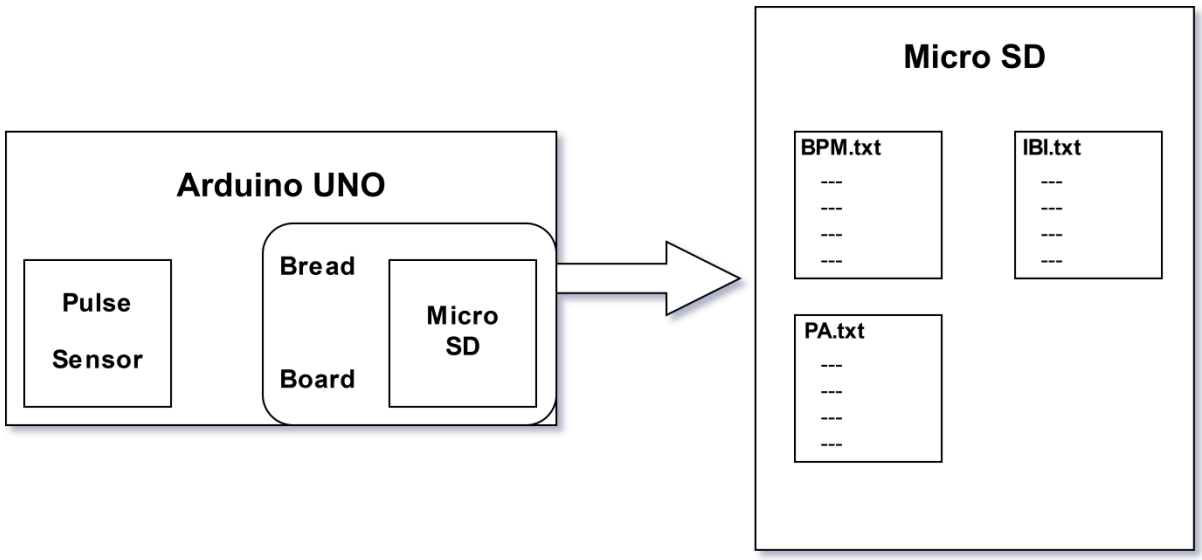
- async (default: true): Determina se la chiamata deve essere asincrona (true) o sincrona (false).
- complete: Consente di specificare una funzione che verrà eseguita al termine della chiamata indipendentemente che abbia dato successo o errore; è eseguita dopo success o error.
- data: Contiene i dati che devono essere inviati alla risorsa che elabora la richiesta; il formato può essere sotto forma di oggetto (contenente delle coppie chiave/valore) oppure sotto forma di semplice stringa
- dataType: Tipo di dato che si riceve di ritorno; Non è un parametro obbligatorio, ma è sempre consigliabile specificarlo. I tipi possibili sono: "xml", "html", "script", "json", "jsonp" e "text".
- error: Consente di specificare una funzione che verrà eseguita in caso di errore nell'effettuare la chiamata.
- success: Consente di specificare una funzione che verrà eseguita al successo della chiamata.
- timeout: E' possibile impostare una durata massima (in millisecondi) della chiamata; se la risorsa non risponde entro il imite fissato la chiamata viene abortita.
- type (default: GET): E' utilizzato per specificare il tipo di richiesta da effettuare, principalmente POST o GET; sono utilizzabili anche altri metodi HTTP (come ad es. PUT, DELETE, ...) ma non tutti i browser li supportano.
- url: URL della risorsa alla quale viene inviata la richiesta.
- headers: Consente di specificare gli headers per il tipo di richiesta HTTP da effettuare

L'ordine dei parametri passati al metodo $.ajax() non è assolutamente rilevante.[13]

3. Sperimentazione e risultati

Per garantirne il corretto funzionamento è necessario produrre un codice compatibile con le necessità del progetto. La programmazione dei microcontrollori rappresenta il cuore del lavoro di tesi il cui scopo è quello di eseguire ripetutamente il compito prefissato, svolgendo la funzione di datalogger.

Modulo Arduino UNO con Pulse Sensor e MicroSD Card Adaptor



Il codice caricato sull'Arduino UNO si occupa di rilevare, ad ogni battito cardiaco percepito dal sensore, i valori di interesse e di memorizzarli su tre differenti file .txt sulla scheda SD.

Introduzione:

- Viene inclusa la libreria `PulseSensorPlayground.h`, necessaria per poter utilizzare il Pulse Sensor.
- Viene inclusa la libreria `SPI.h`, un protocollo dati seriale e sincrono, utilizzato dal microcontrollore per comunicare con le periferiche vicine.
- Viene inclusa la libreria `"SdFat.h"`, necessaria per memorizzare i dati nella MicroSD e viene istanziato un oggetto `SdFat SD`.
- Si definisce una variabile `const int PulseWire` e la si inizializza a "0" poichè il PURPLE WIRE del Pulse Sensor è connesso all' ANALOG PIN 0 di Arduino.
- Si definisce una variabile `int Threshold` e la si inizializza a "550". Si tratta di un valore di default consigliato nei codici d'esempio del Pulse Sensor. Determina quale segnale viene "contato come un battito" e quale viene ignorato.
- Viene creata un'instanza dell'oggetto del tipo `PulseSensorPlayground` chiamato `pulseSensor`.
- Vengono definiti i tre file BPMFile, IBIFile e PAFile che conterranno i valori misurati.

void setup ():

- Viene aperta la comunicazione seriale ed il porto seriale.
- Viene inizializzata la scheda MicroSD.
- Vengono creati i file BPM.txt, IBI.txt e PA.txt sulla MicroSD e ne viene controllata l'esistenza.

- Si configura l'oggetto `pulseSensor`, inserendo i valori precedentemente definiti:
```
pulseSensor.analogInput(PulseWire);
pulseSensor.setThreshold(Threshold);
```

- Viene verificata l'esistenza dell' oggetto `pulseSensor`.

void loop ():

- Viene chiamata la funzione sul nostro oggetto `pulseSensor` che ritorna il valore BPM come un int.
```
int myBPM = pulseSensor.getBeatsPerMinute();
```
`myBPM` contiene questo valore BPM.

- Viene chiamata la funzione sul nostro oggetto `pulseSensor` che ritorna il valore IBI come un int.
```
int myIBI = pulseSensor.getInterBeatIntervalMs();
```
`myIBI` contiene questo valore IBI.

- Viene chiamata la funzione sul nostro oggetto `pulseSensor` che ritorna il valore PA come un int.
```
int myPA = pulseSensor.getPulseAmplitude();
```
`myPA` contiene questo valore PA.

- Si controlla costantemente se "è avvenuto un battito"
```
if (pulseSensor.sawStartOfBeat())
```

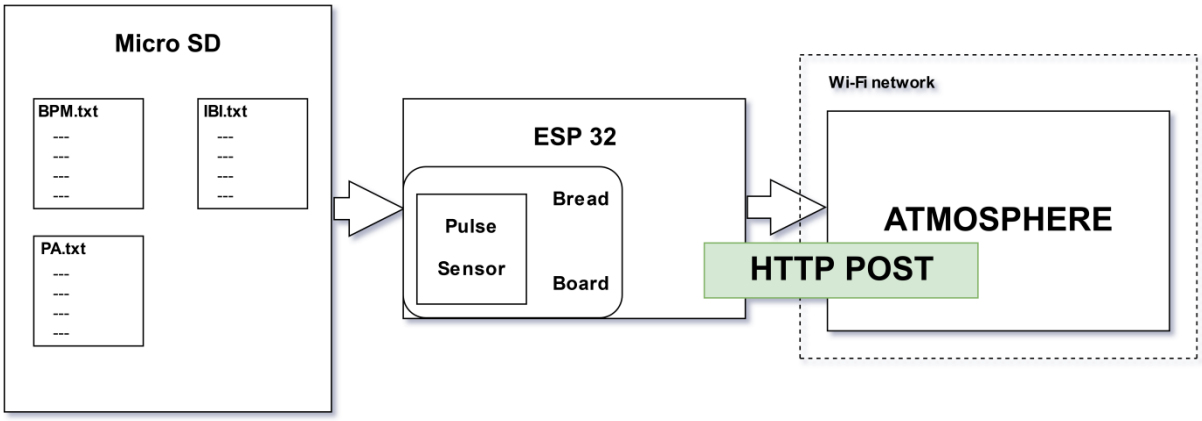
- Se il test è "true" I valori di BPM, IBI e PA vengono formattati come String per essere salvati nei file .txt
```
String stringBPM = "";
String stringIBI = "";
String stringPA = "";
stringBPM = String(myBPM);
stringIBI = String(myIBI);
stringPA = String(myPA);
```

- Vengono aperti tutti i file .txt
```
 BPMFile = SD.open("BPM.txt", FILE_WRITE);
IBIFile = SD.open("IBI.txt", FILE_WRITE);
PAFile = SD.open("PA.txt", FILE_WRITE);
```

- Se i file sono aperti allora vengono salvati i valori BPM, IBI e PA sui file .txt
```
if (BPMFile && IBIFile && PAFile) {
 BPMFile.println(stringBPM + "\n");
 delay(50);
 IBIFile.println(stringIBI + "\n");
 delay(50);
 PAFile.println(stringPA + "\n");
 (...)
```
- Vengono chiusi I file .txt
```
BPMFile.close();
IBIFile.close();
PAFile.close();
```

- Se i file non si aprono, viene stampato un errore sul monitor seriale
```
Serial.println("error opening files");
```

Modulo ESP32 con MicroSD Card Adaptor



Il codice caricato sull'ESP32 si occupa di rilevare le misure presenti nei file .txt, di generare e inviare la richiesta HTTP Post alla Web API "Atmosphere".

Introduzione:

- Viene definito con un `#define` il TOKEN di autorizzazione necessario per la comunicazione con "Atmosphere"
  ```
  #define TOKEN "JWT (...)_cNNVqMeeNpX0"
  ```
- Viene inclusa la libreria `HTTPClient.h` per eseguire richieste HTTP POST utilizzando l'ambiente ESP32 e Arduino
- Viene inclusa la libreria `mySD.h`, analoga alla precedente `"SdFat.h"`, necessaria per leggere i dati dalla MicroSD, adatta per essere utilizzata nell'ambiente ESP32 e Arduino
- Viene inclusa la libreria **`SPI`**`.h`, un protocollo dati seriale e sincrono, utilizzato dal microcontrollore per comunicare con le periferiche vicine
- Viene inclusa la libreria **`WiFi`**`.h` che permette all'ESP32 di connettersi ad internet
- Vengono definiti i tre file BPMFile, IBIFile e PAFile che contengono i valori misurati
- Vengono definiti tre array `BPMResult[10]`, `IBIResult[10]`e `PAResult[10]` che conterrano i valori memorizzati nella scheda MicroSD
  ```
  String BPMResult[10];
  String IBIResult[10];
  String PAResult[10];
  ```
- Si definisce una variabile `const char* ssid` contenente il nome della rete Wifi
- Si definisce una variabile `const char* password` contenente la password per accedere alla rete Wifi
- Si definisce una variabile `const char server[]` contenente l'url del server dove postare le misure

void setup ():

- Viene aperta la comunicazione seriale ed il porto seriale
- Viene inizializzata la scheda MicroSD

- Vengono visualizzati i file e le directory presenti nella scheda MicroSD sul monitor seriale dell'Arduino IDE, attraverso la funzione `printDirectory()`

```
void printDirectory(File dir, int numTabs) {

while(true) {
File entry =  dir.openNextFile();
if (! entry) {
      break;
}
for (uint8_t i=0; i<numTabs; i++) {
      Serial.print('\t');
}
// Print the name
Serial.print(entry.name());

if (entry.isDirectory()) {
      Serial.println("/");
      printDirectory(entry, numTabs+1);
} else {

      Serial.print("\t\t");
      Serial.println(entry.size());
}
entry.close();
  }
}
```

- Viene avviata la connessione WiFi

void loop ():

- Viene controllata la presenza della connessione WiFi, altrimenti viene visualizzato un errore sul monitor seriale.

- Se la connessione è presente, viene costruita la richiesta HTTP POST per inviare i valori BPM al server. Si utilizza la funzione `BPMFgets()`.

- La funzione `BPMFgets()` legge il file BPM.txt e memorizza le prime dieci misure presenti nell'array `BPMResult`

```
void BPMFgets() {
String line;
int i=0;
// open test file
if (!SD.exists("BPM.txt")) {      // Controlla se esiste il file
                                  BPM.txt contenente le misure
Serial.println("BPM.txt FOUND");
return;
}
Serial.println("BPM.txt FOUND");
File BPMFile = SD.open("BPM.txt",FILE_READ);
if (!BPMFile) {
return;
}
Serial.println("BPM.txt open");
```

```
while (BPMFile.available() != 0) {
line = BPMFile.readStringUntil('\n'); // Il file viene letto fino al
                                       carattere '\n', il dato viene
                                       quindi memorizzato nell'array
                                       BPMString
line.trim();

if (i<= 9 && line != "") {

BPMResult[i] = line;
Serial.println(BPMResult[i]);
i= i+1;
}
else {

  }
}
```

- Viene istanziato un oggetto `HTTPClient`
  `HTTPClient http;`

- Viene specificata la destinazione della richiesta HTTP
  `http.begin(server);`

- Si definisce l'header della richiesta HTTP: viene specificato il content-type ed il TOKEN di "Atmosphere" di autorizzazione.
  ```
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Authorization", TOKEN);
  ```

- Viene costruito il body della richiesta HTTP, tramite la funzione `getBPMBody()`
  `String BPMbody = getBPMBody();`
  La funzione `getBPMBody()` costruisce il body della richiesta HTTP
  ```
  String getBPMBody() {
  String body = "";
  body += " {\"thing\": \"mario.rossi\",";
  body += " \"feature\": \"heart-rate\",";
  body += " \"device\": \"pulse-sensor\",";
  body += " \"samples\": [ ";
  body += " { \"values\": [" + BPMResult[0] + "], \"delta\": 0 }, ";
  body += " { \"values\": [" + BPMResult[1] + "], \"delta\": 10 }, ";
  (...)
  body += " { \"values\": [" + BPMResult[9] + "], \"delta\": 90 } ";
  body += " ] }";
  return body;
  }
  ```

- Viene inviata la richiesta HTTP POST e viene salvato il codice di risposta nella variabile `httpResponseCodeBPM`
  `int httpResponseCodeBPM = http.POST(BPMbody);`

- Viene effettuato un controllo sul valore `httpResponseCodeBPM`; se è positivo si ottiene una stringa contentente la risposta alla richiesta, altrimenti viene mostrato sul monitor seriale un messaggio di errore
  ```
  if(httpResponseCodeBPM>0){

  String BPMresponse = http.getString();    // Viene ottenuta la
  risposta                                       alla richiesta
  ```

```
Serial.println(httpResponseCodeBPM);    // Stampa il codice di ritorno
Serial.println(BPMresponse);            // Stampa la risposta alla
                                        richiesta HTTP

}
else{

Serial.print("Error on sending POST (BPM): ");
Serial.println(httpResponseCodeBPM);

}
```

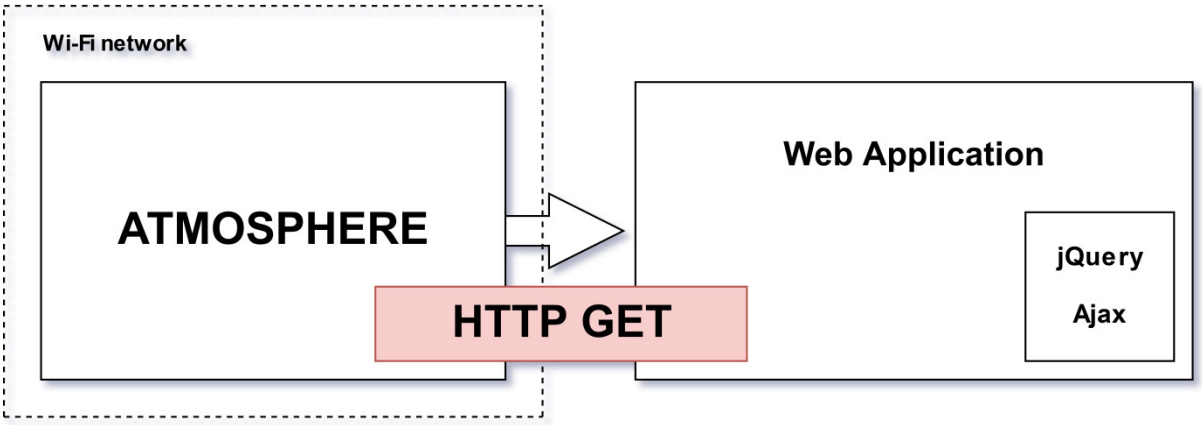
- Vengono liberate le risorse
  `http.end();`

- Viene costruita la richiesta HTTP per inviare i valori IBI.
  Vengono ripetute esattamente le stesse istruzioni precedenti, viste per le misure BPM ma con le analoghe funzioni `IBIFgets()` e `getIBIBody()`.

- Viene costruita la richiesta HTTP per inviare i valori PA.
  Vengono ripetute esattamente le stesse istruzioni precedenti, viste per le misure BPM ma con le analoghe funzioni `PAFgets()` e `getPABody()`.

- Il programma entra in un loop infinito e non vengono più inviate altre richieste HTTP, fino al prossimo RESET della scheda

Interfaccia Web basata su HTML, CSS e JavaScript



Il codice dell'interfaccia Web è distribuito su due files: index.html e data.js

Codice index.html

Il codice HTML si occupa di definire lo stile ed il layout della tabella.

- Definizione del titolo della pagina HTML

```
<!DOCTYPE html>
<html>
     <head>
<title>Heart Parameters</title>
<meta charset="UTF-8" />
```

- Viene importato lo stile CSS per la creazione della tabella messo a disposizione dal sito www.siimple.xyz

```
<link rel="stylesheet"href="https://cdn.jsdelivr.net/npm/siimple@3.3.1/dist
/siimple.min.css">
```

- Viene importata la risorsa Jquery necessaria per formulare la richiesta HTTP

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min
.js">
</script>
```

- Viene importato lo script dove viene formulata la richiesta HTTP

```
<script src="data.js">
</script>
</head>
```

- Vengono definiti i parametri della tabella: il titolo e la THING (l'oggetto della misura). Nella prima riga della tabella viene definito il tipo di misurazione effettuata

```
<body>
<h2 align="center">Heart Parameters</h2>
<p class="user"> User: </p> </div>

<td><img id="image" src="image.png" height="75" width="150"></td>
<div class="siimple-table siimple-table--striped">
<div class="siimple-table-header">
<div class="siimple-table-row">
<div class="siimple-table-cell">Beats Per Minute (bpm) </div>
<div class="siimple-table-cell">Interbeat interval (milliseconds) </div>
<div class="siimple-table-cell">Pulse Amplitude (millivolts)</div>
</div>
</div>
<div class="siimple-table-body">
<div class="siimple-table-row">
```

- Le altre righe vengono riempite dalle misure recuperate dal server. Vengono mostrate dieci misurazioni diverse. Le linee di codice vengono ripetute per dieci volte.

```
<div class="siimple-table-cell"><div>  <p class="value0BPM">
</p> </div></div>
<div class="siimple-table-cell"><div>  <p class="value0IBI">
</p> </div></div>
<div class="siimple-table-cell"><div>  <p class="value0PA">
</p> </div></div>
</div>
<div class="siimple-table-row">
<div class="siimple-table-cell"><div>  <p class="value1BPM">
```

```
</p> </div></div>
<div class="siimple-table-cell"><div>  <p class="value1IBI">
 </p> </div></div>
<div class="siimple-table-cell"><div>  <p class="value1PA">
</p> </div></div>
</div> (ecc...)
```

Codice data.js

Il codice JavaScript si occupa di effettuare la richiesta HTTP GET e di mettere a disposizione le misure all'HTML.

- Viene formulata la richiesta HTTP

```
$(document).ready(function() {
  $.ajax({
   type: "GET",
   url: "http://test.atmosphere.tools/v1/measurements",
   data: {filter: "{\"thing\":\"mario.rossi\"}", page : 1, limit :10},
   headers: {
      "Content-type" : "application/json",
      "Authorization": "Token di autorizzazione di Atmosphere",
      "charset": "UTF-8"
    },
      success: function(response){
    }
   })
```

- Vengono messe a disposizione all'HTML le misure recuperate dal server

```
 var o = response;
 console.log(o);
$('.user').append(o.docs[0].thing);
$('.value0PA').append(o.docs[0].samples[0].values[0]);
$('.value1PA').append(o.docs[0].samples[1].values[0]);
(ecc...)
$('.value0IBI').append(o.docs[1].samples[0].values[0]);
$('.value1IBI').append(o.docs[1].samples[1].values[0]);
(ecc...)
$('.value0BPM').append(o.docs[2].samples[0].values[0]);
$('.value1BPM').append(o.docs[2].samples[1].values[0]);
(ecc...)
  });
}) ;
```

Verifica del funzionamento

Per poter verificare il corretto funzionamento del dispositivo è necessario constatare che i dati ottenuti siano riconducibili a dei valori reali. Questo tipo di analisi può essere effettuato sui valori di Beats Per Minute e Interbeat interval.

Il battito cardiaco del nostro cuore, varia in base alla nostra età: in linea di massima più aumenta la fascia di età e più andrà a diminuire il battito medio al minuto del nostro cuore.

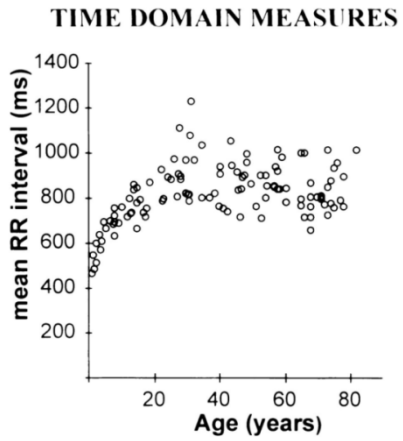
I valori normali dei battiti del cuore risultano essere:

- Per i neonati (80 – 180 bpm)
- Per i bambini (80 – 100 bpm)
- Per gli adolescenti (70 – 120 bpm)
- Per gli adulti (60 – 90 bpm)[14]

La frequenza cardiaca può avere valori diversi in tutto l'arco della giornata: anche in stato di quiete il battito cardiaco del corpo umano assume valori diversi in funzione del momento in cui ci si trova (diminuisce durante il riposo notturno e si alza dopo aver mangiato).

A modificare l'intensità del battito cardiaco possono essere anche le emozioni e lo stress: infatti se ci si trova davanti ad episodi che suscitano i precedenti, l'intensità del battito cardiaco s'innalzerà portando in alcuni casi (riguardanti individui affetti da patologie) anche a situazioni di pericolo.

La dinamica degli intervalli interbeat cardiaci varia anch'essa con l'età e si tratta di una variabile in relazione ai cambiamenti nei meccanismi regolatori. La maturazione del sistema nervoso autonomo e di altri sistemi di controllo durante l'infanzia, causano una maggiore variazione della frequenza cardiaca, mentre con l'invecchiamento vi è una riduzione della variabilità complessiva delle risorse umane e anche della complessità della dinamica fisiologica. Questo fatto può essere dovuto sia a fattori strutturali sia a cambiamenti funzionali. Attraverso un lungo monitoraggio effettuato su 114 individui in salute (con un'età compresa dagli 1 agli 82 anni) i valori Interbeat interval variano secondo questo grafico: [15]



Considerando quanto detto, sono state effettuate delle misure di BPM e IBI a due differenti users, a riposo: Giuseppe.bianchi, adolescente (19 anni) e mario.rossi, adulto (51 anni). Utilizzando le Web Application dedicate ai due users, è possibile osservarne le prime dieci rilevazioni ed è possibile constatare che i valori di BPM e IBI ottenuti rientrano pienamente nelle fasce d'età rispettive degli users; questo fatto dimostra l'effettiva efficacia del dispositivo ottenute

4. Contributo personale e considerazioni conclusive

Ho scelto di trattare questo argomento di tesi perché mi ha sempre suscitato particolare interesse il mondo dei microcontrollori e della programmazione. La realizzazione di questo progetto mi ha permesso mi mettermi in contatto, per la prima volta, con le principali problematiche che si possono incontrare durante l'organizzazione di un sistema con microcontrollori e reti internet.

Le parti su cui mi sono concentrato maggiormente sono state la scrittura del codice e l'implementazione del circuito dell'ESP32. La scrittura dei programmi ha richiesto una notevole quantità di tempo, per svariate ragioni: prima fra tutte, l'affacciarsi direttamente al mondo dei microcontrollori in modo pratico. È stato, infatti complesso ricercare il codice d'esempio da poter adattare al mio caso specifico, soprattutto nel caso della HTTP POST, e trovare delle librerie specifiche e compatibili con l'hardware a disposizione. La stesura del codice dell'Arduino è stata sufficientemente fluida: si è trattato di adattare uno dei codici di esempio del Pulse Sensor alle mie specfiche. La difficoltà maggiore è stata nel ricercare la libreria adatta allo scopo di memorizzare i dati in una microSD, all'interno di file .txt. La `SD.h` di default non è riuscita a garantirmi l'affidabilità necessaria e quindi ho deciso di optare per la `"SdFat.h"` che ad ogni esecuzione era in grado di restituirmi il risultato desiderato. Inizialmente, per inviare l'HTTP POST, era previsto l'utilizzo di un'Arduino Wifi Shield ma a causa della mancanza di un preciso codice d'esempio ho preferito utilizzare l'ESP32 che era dotato di una sua specifica libreria, l' `HTTPClient.h` che permette di generare con poche linee di codice una richiesta HTTP. La `"SdFat.h"` però non è compatibile con questo microcontrollore, pertanto è stato necessario utilizzare la `mySD.h`, una libreria simile alla precedente ma adatta a lavorare sull'ESP32. Per la creazione della Web Application mi sono basato su una semplice applicazione .html che esegue una HTTP GET, successivamente estesa alle mie esigenze. Per quanto riguarda il CSS, cioè lo stile di ciò che si vede sulla pagina, ho utilizzato quello messo a disposizione da www.siimple.xyz che presenta delle tabelle semplici ma funzionali alle esigenze. Ci sono state alcune difficoltà con la formattazione JSON della risposta del server all'HTTP GET che sono state superate utilizzando la funzione replace e sostituendo caratteri della risposta Atmosphere con i caratteri di un file JSON.

Il cercare di risolvere le problematiche citate, mi ha permesso di entrare nella logica della programmazione non più finalizzata alla risoluzione di un esercizio ma al compimento di un progetto più esteso e tangibile. Per questo motivo, l'aver compiuto questo lavoro di tesi, è stata un' esperienza positiva, che mi ha permesso di acquisire una serie di competenze pratico operative finora da me poco potenziate. È stato quindi un percorso decisamente utile per mettere in pratica le mie capacità e attitudini.

Questa tesi rappresenta l'esempio di una possibile estensione di funzionalità dei dispositivi, come ad esempio gli smartwatch, di largo utilizzo, in grado di rilevare parametri fisiologici. Infatti rendere facilmente disponibili le rilevazioni, tramite Web Application, ad un medico o ad un infermiere, può essere di estremo aiuto sia in caso di emergenza, sia in caso preventivo. La rilevazione dei parametri fisiologici fondamentali è infatti una componente importante dell'attività infermieristica; i segni vitali fanno parte della serie di dati raccolti dagli infermieri durante l'accertamento, o durante il monitoraggio dei pazienti che per svariate motivazioni necessitano di un continuo aggiornamento dei parametri, essi rappresentano una modalità veloce ed efficace per valutare le condizioni del paziente ed identificare la presenza di problemi o la risposta del paziente ad alcuni interventi. Una versione collaudata, ottimizzata ed estesa di questo progetto di tesi, porterebbe quindi notevoli agevolazioni in ambito medico al fine di rendere i possibili provvedimenti più efficaci.

5. Riferimenti bibliografici

[1] http://www.alianteit.it/iot-la-rivoluzione-internet-of-things/
[2] https://store.arduino.cc/arduino-uno-rev3
[3] https://www.alberti-porro.edu.it/wordpress/wp-content/uploads/2014/01/ProgrammareArduino.pdf
[4] https://pulsesensor.com/pages/open-hardware
[5] https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf
[6] https://www.ne555.it/modulo-microsd-card-adapter/
[7] https://www.arduitronics.com/product/210/microsd-card-adapter
[8] http://www.peduto.it/UsoBreadBoard/uso_breadboard.htm
[9] Ahmad Kobeissi "Atmosphere, a Measurement-oriented Data Framework for IoT"
[10] http://www.di-srv.unisa.it/~ads/corso-security/www/CORSO-9900/http/intro.htm
[11] https://www.cosmobile.com/ita/blog/a-cosa-servono-le-applicazioni-web-1001.html
[12] https://www.html.it/pag/16026/introduzione22/
[13] http://www.di.uniba.it/intint/people/progwebnicole/15-IntroAjax.pdf
[14] Daniel Limmer, Michael F. O'Keefe, *Emergency Care*, 10a ed, Upper Saddle River, New Jersey, Edward Pearson, Prentice Hall, 2005, p. 214.
[15] https://www.ahajournals.org/doi/full/10.1161/01.cir.100.4.393