POLITECNICO DI TORINO

Corso di Laurea Magistrale

in INGEGNERIA ELETTRICA

Tesi di Laurea Magistrale

Impianti semaforici innovativi per la comunicazione infrastruttura-utente

Relatori:
Prof. Paolo Di Leo
Dott. Ing. Alessandro Ciocia

Candidato:
Mattia Ghione

firma dei relatori:

firma del candidato:

Anno accademico 2016-2017

Sommario

Premessa

L'obiettivo di questa tesi, riguarda lo studio e la creazione di dispositivi elettronici in grado di rilevare e comunicare lo stato in cui si trova una lanterna semaforica a diverse tipologie di utenti.

La tesi è stata svolta nell'ambito di un contratto di ricerca tra il *DENERG – Dipartimento Energia del Politecnico di Torino* e *SOGEN s.r.l.*, società di ingegneria, relativo allo sviluppo di nuove applicazioni innovative per impianti semaforici.

In accordo con la suddetta società, si è optato per soluzioni innovative a basso costo, pur garantendo un'elevata affidabilità, in modo da poter facilmente introdurre le nuove tecnologie nel settore.

Lo sviluppo è stato suddiviso in più parti, prima fra tutte la ricerca e lo studio allo stato attuale degli impianti semaforici, delle tecnologie elettriche, elettroniche e di comunicazione disponibili. A questa prima fase di ricerca, è seguita l'attività in laboratorio dove è stato ideato, costruito, programmato e collaudato il primo prototipo di questa nuova tecnologia. Dopo di che, grazie alla collaborazione con *IREN S.p.a.*, sono stati eseguiti dei test su un impianto semaforico funzionante in modo da ottenere una valutazione generale del lavoro svolto. Dai dati ottenuti è stato possibile valutarne i pregi ed i difetti in modo da migliorare ed affinare i prototipi realizzati successivamente.

Cap. 1: Innovazione tecnologica dei sistemi semaforici e prospettive

1.1 Generalità

Verso la metà del *XIX* secolo in Inghilterra, si decise di utilizzare un segnale luminoso per poter controllare il traffico sempre più crescente delle carrozze presenti nella capitale inglese. Il primo semaforo fece la sua comparsa a **Londra** nel ***1868*** quando venne installato in un incrocio vicino al parlamento britannico. Derivato dai segnali ferroviari del tempo, era costituito da una lanterna a gas rotativa che emetteva una luce rossa ed una luce verde durante la notte e da un sistema di bracci meccanici che davano indicazioni agli autisti dei mezzi se potevano o meno attraversare questo incrocio. La prima installazione non ebbe buon esito, in quanto dopo un incidente in cui venne coinvolto un poliziotto intento nell'avvio della lanterna venne dismesso definitivamente nel *1872* [1].

Il primo semaforo moderno vero e proprio, per certi versi ancora differente da quelli di oggi, ma alimentato per la prima volta dalla rete elettrica, comparve a **Cleveland** nell'agosto del ***1914***; infatti aveva solamente due lanterne (il rosso ed il verde) ed era comandato a distanza da un vigile. Per i primi esempi di lanterne a tre luci si deve attendere il *1920*, a New York City dove due anni più tardi vennero installati i primi semafori controllati automaticamente. Da questo momento in poi questo sistema semaforico venne esportato anche in Europa, la prima città fu Parigi nel *1922* seguita poi da Amburgo, Berlino e **Milano** nel ***1925*** in piazza Duomo tra via Orefici e via Torino, dove però l'installazione porto un notevole aumento degli incidenti stradali [2].

Un altro passo importante avvenne nel ***1952***, sempre a **New York** dove venne installato il primo attraversamento pedonale con le famose scritte: "*Walk/Don't Walk*" collocato all'angolo della 44esima strada [3].

I semafori moderni sono molto cambiati rispetto a quelli del passato, oggi si avvalgono infatti delle più moderne tecnologie come ad esempio l'illuminazione led, i segnalatori acustici per aiutare gli utenti non vedenti ed il countdown del tempo disponibile.

Nel presente capitolo si intende analizzare quali sono le aziende presenti sul mercato e quali prodotti offrono in questo ambito, in modo da aver presente qual è la situazione attuale. Si vogliono inoltre studiare quali sono i protocolli di comunicazioni più adatti ai nostri scopi facendo un confronto tra essi, valutando pregi e difetti di ognuno, ma tenendo sempre presente i principi su cui si basa tutto questo elaborato.

1.1.1 Sistemi semaforici in Italia

Dall'indagine condotta si è scoperto che sono molte le società che operano sul mercato dei sistemi semaforici ed il nostro scopo è analizzare quale è lo schema di base per ogni impianto e quali sono le soluzioni tecnologiche offerte da queste aziende. Gli elementi principali che compongono un semaforo sono: la lanterna con le tre luci, una centralina contenente un regolatore semaforico (*PLC*), gli elementi che permettono di garantire la sicurezza elettrica dell'impianto (interruttori magnetotermici-differenziali) e la morsettiera per la connessione della lanterna stessa e dell'impianto alla rete elettrica. Da questa base di partenza, il mercato offre molte soluzioni ed accessori che permettono di personalizzare il sistema a seconda delle necessità. Infatti si possono installare regolatori di nuova generazione dotati di display touch-screen e di sistemi di comunicazione Bluetooth che permettono di interagire con il PLC senza dover aprire il quadro, ma direttamente dal proprio smartphone o tablet; inoltre sono disponibili diversi sistemi di comunicazione (*Ethernet, Wi-Fi, Modem 2G/3G/4G*) che permettono di eseguire tutte le operazioni necessarie direttamente dalla sede operativa [4]. Oltre a diversi tipi di regolatori, sono disponibili molte varietà di lanterne semaforiche con luci ad incandescenza o led, con diversa tensione di alimentazione; tra gli accessori i più importanti sono: i moduli addizionali con il "*countdown*" dei tempi [5] e i dispositivi acustici per non vedenti per la sicurezza nell'attraversamento [6][7]. Negli ultimi tempi stanno prendendo piede i servizi di monitoraggio del traffico basati su software multifunzione e applicazioni per smartphone [8]. La ricerca è stata svolta tra le aziende più attive su questo tipo di settore: *SCAE s.p.a.* [9], *EMI s.r.l.* [10], *INCES s.r.l.* [11] e *La Semaforica s.r.l.* [12].

1.2 Applicazione della tecnologia bus ai sistemi semaforici

In informatica ed in elettronica la tecnologia ***BUS*** è un canale di comunicazione che permette alle periferiche ed ai componenti di comunicare tra di loro scambiandosi informazioni e dati attraverso la trasmissione di segnali, con la differenza che a dispetto delle connessioni point to point un solo bus può collegare più dispositivi. Le connessioni possono avvenire su circuito stampato oppure tramite cavo. Nel primo caso se la connessione è di tipo parallelo la scheda presenta un alto numero di piste compatte in parallelo che vanno a collegare i diversi componenti della scheda. L'altra modalità di connessione invece è di tipo seriale. Con l'avanzare della tecnologia sembra che quest'ultima sia la soluzione più apprezzata, perché offre meno ingombri e costi più contenuti. Tra i più importanti esempi di bus seriali si possono citare: *SPI*, *SATA*, P*CI Express*, *USB*, *LonWorks*, *Konnex* e *CAN*.

Durante lo studio della tecnologia BUS sono state effettuate le analisi di costi e benefici tra i protocolli più diffusi, in particolare si sono prese in considerazione per il confronto le tecnologie *LonWorks* (*Echelon*), *Konnex* (*KNX*) e *CAN*. La prima di queste si basa su un protocollo sviluppato da *Echelon Corporation* per dispositivi che utilizzano diversi tipi di connessioni fisiche (doppino, onde convogliate, fibra ottica, radio e TCP/IP) per comunicare tra loro. Il protocollo è denominato *LonTalk* ed è diffuso nei sistemi di automazione di edifici come ad esempio gli impianti di illuminazione, di controllo della temperatura in impianti che utilizzano sensori di movimento e circuiti di sicurezza interna che fanno uso di telecamere [12-15]. Questa soluzione presenta però costi relativamente alti dei componenti.

La tecnologia *Konnex* è il primo standard di domotica coperto da royalty approvato sia come standard europeo che mondiale sviluppato dalla *KNX Association*, ed anch'esso come il *LonWorks* è diffuso negli impianti di illuminazione, sicurezza, risparmio energetico e monitoraggio [16-19]. Come per la precedente questa soluzione presenta un alto costo dei componenti e a cui si aggiunge il costo delle royalties per ciascun controller programmato.

La tecnologia più promettente per le applicazioni semaforiche, nel caso di impiego di tecnologia BUS, tenendo presente le esigenze concordate con *SOGEN* risulta essere il protocollo CAN-BUS che consente di tenere i costi bassi, avere un'alta affidabilità ed avere componenti facilmente disponibili.

1.2.1 Protocollo di comunicazione CAN-BUS

Il protocollo **CAN-BUS**, dove CAN significa *Controller Area Network*, è uno standard seriale, di tipo Broadcast (ovvero che può distribuire informazioni verso tutti i destinatari contemporaneamente) sviluppato da *BOSCH* agli inizi degli anni '80. Questa tecnologia è diffusa principalmente nel campo dell'industria automobilistica per consentire la comunicazione fra i dispositivi elettronici intelligenti di cui gli autoveicoli sono dotati. Il suo successo è dovuto ai notevoli vantaggi tecnologici che offre:

- **Tempi di risposta rapidi:** la tecnologia CAN prevede molti strumenti hardware e software e sistemi di sviluppo per protocolli ad alto livello (il bus CAN implementa solo i primi due livelli della pila *ISO-OSI*) che consentono di connettere un elevato numero di dispositivi mantenendo stringenti vincoli temporali.

- **Semplicità e flessibilità del cablaggio:** il bus seriale è implementato su un doppino intrecciato (schermato o meno a seconda delle esigenze). I nodi non hanno un indirizzo che li identifichi e possono essere aggiunti o rimossi senza dover riorganizzare il sistema una sua parte. Teoricamente il numero di nodi che possono essere presenti è infinito.

- **Alta immunità ai disturbi:** lo standard *ISO11898* raccomanda che i chips di interfaccia possano continuare a comunicare anche in condizioni estreme, come l'interruzione di uno dei due fili o il cortocircuito di uno di essi con massa o con l'alimentazione. Un problema comune, nella trasmissione di segnali, sono i disturbi di tipo *RFI* (Radio Frequency Interferences). Questi sono generati involontariamente (o volontariamente) dagli apparati tecnologici che oggi ci circondano e la soluzione primaria da adottare contro le RFI è schermare i cavi con un conduttore metallico che poi viene messo a terra. Per compensare i limiti di questa tecnica, si utilizza la tecnica chiamata differencial. Questa tecnica sfrutta due cavi, dove viene inviato rispettivamente il segnale trasmesso come positivo e lo stesso segnale trasmesso in negativo. I disturbi saranno presenti in entrambi i cavi ed avranno lo stesso valore. Il ricevente non deve far altro che la differenza trai due segnali, in modo tale che la parte di rumore risulterà scomparsa. Questo metodo funziona bene sole se sui due cavi c'è lo stesso disturbo, ovvero occupino lo stesso spazio. Per arrivare a una legittima approssimazione, si avvolgono i due cavi insieme (Twisted Pair).

- **Elevata affidabilità:** la rilevazione degli errori e la richiesta di ritrasmissione viene gestita direttamente dall'hardware con cinque diversi metodi (due a livello di bit e tre a livello di messaggio). Il bus can ha un'incredibile capacità di riconoscere gli errori. La probabilità che un messaggio sia corrotto e non riconosciuto come tale, è praticamente nulla.
  É stato calcolato che una rete basata su CAN-BUS a *1 Mbit/s*, con un'utilizzazione media del bus del *50%*, una lunghezza media dei messaggi di *80 bit* e un tempo di lavorazione di *8* ore al giorno per *365* giorni l'anno, avrà un errore non rilevato ogni *1000* anni. Praticamente la rete non è soggetta ad errori per tutta la durata della sua vita. Questo è il maggior punto di forza di questo bus.

- **Confinamento degli errori:** ciascun nodo è in grado di rilevare il proprio malfunzionamento e di autoescludersi dal bus se questo è permanente. Questo è uno dei meccanismi che consentono alla tecnologia CAN di mantenere la rigidità delle temporizzazioni, impedendo che un solo nodo metta in crisi l'intero sistema.

- **Maturità dello standard:** la larga diffusione del protocollo CAN in questi venti anni ha determinato un'ampia disponibilità di chip ricetrasmettitori, di microcontrollori che integrano porte CAN, di tools di sviluppo, oltre che una sensibile diminuzione del costo di questi sistemi.

- **Velocità di trasmissione e lunghezza del bus:**

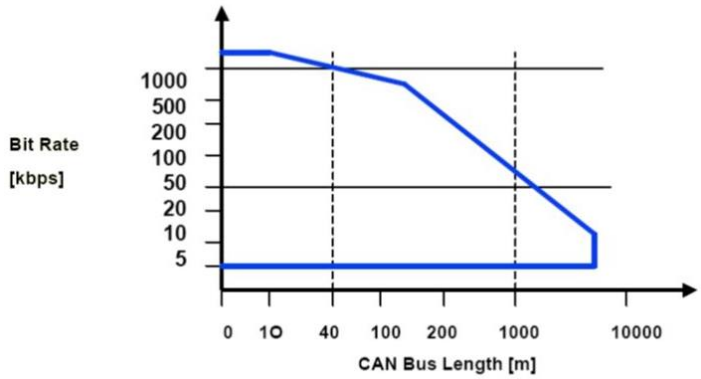

Figura 1 - Bit Rate CAN-BUS

La lunghezza massima del bus dipende dalla velocità usata per la trasmissione (e vice-versa). Come si può vedere dal grafico è stato calcolato che il CAN-BUS può raggiungere *1 Mbit/s* per reti lunghe meno di *40m*. Per velocità inferiori si possono raggiungere distanze maggiori es: per velocità di *50kbit/s* si possono raggiungere distanze di *1000m*.

- **Prototipazione e costi:** Tipicamente un nodo intelligente ha un suo processore con una sua memoria, esso è in grado di gestire ed elaborare autonomamente i dati. I singoli microcontrollori sono provvisti di ram, eeprom, convertitore A/D e di un interfacciamento con i dispositivi esterni. Oggi, questa tecnologia è economicamente alla portata di tutti. Ad esempio un microcontrollore "*MCP2515-I/SO*" su cui si basa la comunicazione CAN-BUS ha un costo che è inferiore ai *2€* [20].

- **Criticità:** La trasmissione dei pacchetti contenenti le informazioni avviene in broadcast ovvero che ogni dispositivo riceve il segnale e sta a lui capire se quelle determinate informazioni sono destinate a lui o meno. Il CAN è un protocollo a basso livello e non supporta intrinsecamente alcuna funzionalità di protezione e non esiste alcuna crittografia nelle implementazioni standard. Questo vuol dire che è possibile per chiunque riesca a collegarsi fisicamente alla rete di intercettare i pacchetti informativi, replicarli e modificarli a proprio piacimento. La protezione delle informazioni deve essere effettuata ad un più alto livello ad esempio tramite la crittografia. Per questo motivo si può pensare di utilizzare i microcontrollori *ATMEGA* per proteggere il sistema da intromissioni [21] [22].

1.3 Comunicazione Vehicle-to-Everything (V2X)

L'acronimo ***V2X***(***Vehicle-to-Everything***) e le sue possibili soluzioni stanno ad indicare lo scambio di informazioni tra veicoli e tra infrastrutture di rete del veicolo. Gli obbiettivi che si sono prefissati nella creazione di questo tipo di comunicazione sono i seguenti:

- Migliorare la sicurezza stradale cercando di ridurre al minimo il rischio di incidenti evitabili.
- Aumentare l'efficienza del flusso dei veicoli ovvero ridurre il traffico suggerendo percorsi alternativi liberi.
- Ridurre l'impatto ambientale dei mezzi suggerendo ad esempio la velocità da sostenere in base alle condizioni del traffico, della strada percorsa e della presenza o meno di semafori.
- Fornire informazioni utili come ad esempio informazioni meteo o relative a lavori stradali e servizi ausiliari ai viaggiatori come il pagamento automatico dei pedaggi o dei parcheggi.

La V2X è un sistema di comunicazione veicolare che comprende molti altri specifici tipi di comunicazione come:

- *V2I Vehicle-to-Infrastructure*
- *V2V Vehicle-to-Vehicle*
- *V2P Vehicle-to-Pedestrian*
- *V2N Vehicle-to-Network*

La figura sottostante riassume in modo completo tutte queste tipologie di comunicazione supponendo che queste siano bidirezionali:

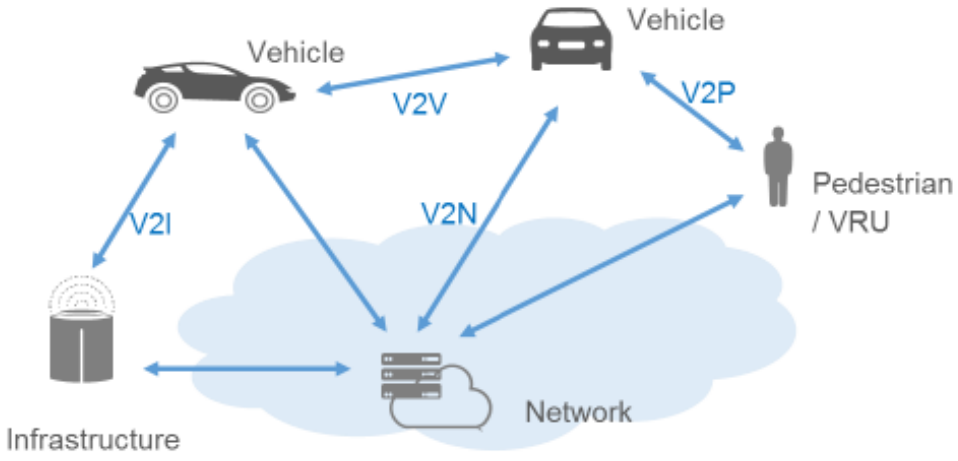

Figura 2 - V2X Comunication

La trasmissione V2V e quella V2P si basa principalmente sulla capacità di trasmettere in modalità broadcast tra veicoli e veicoli (V2V) e tra veicoli e utenti stradali vulnerabili (V2P) come pedoni e ciclisti. Questo tipo di comunicazione è riservata al dispaccio di informazioni come la posizione, la velocità e la direzione dei soggetti interessati al fine di evitare incidenti. Per trasmissione V2I si intende la comunicazione tra veicoli e infrastrutture sia per ricevere informazioni su traffico, lavori in corso o incidenti stradali, ma anche per inoltrare messaggi ricevuti dai veicoli e svolgere quindi la funzione di un ripetitore. Per V2N si intende la comunicazione tra veicoli ed un server dedicato.

Sulla base delle ricerche effettuate dal Dipartimento dei trasporti Statunitensi (*USDOT*) si è dedotto che con un sistema V2X perfettamente implementato si potrebbero evitare l'ottanta percento degli incidenti in cui vengono coinvolti più di un veicolo. Risultati alla mano il Dipartimento si aspetta che i produttori di automobili dotino i propri veicoli del sistema di comunicazione a corto raggio (dedicated short range communication *DSRC*) su tutti i modelli che verranno prodotti dalla fine del 2019 in avanti anche se questa si prospetta un'ardua sfida.

Fortunatamente i sistemi DSRC non sono l'unica soluzione, infatti a seconda di quale V2X si sta parlando ci possono essere più sistemi di comunicazione adottabili:

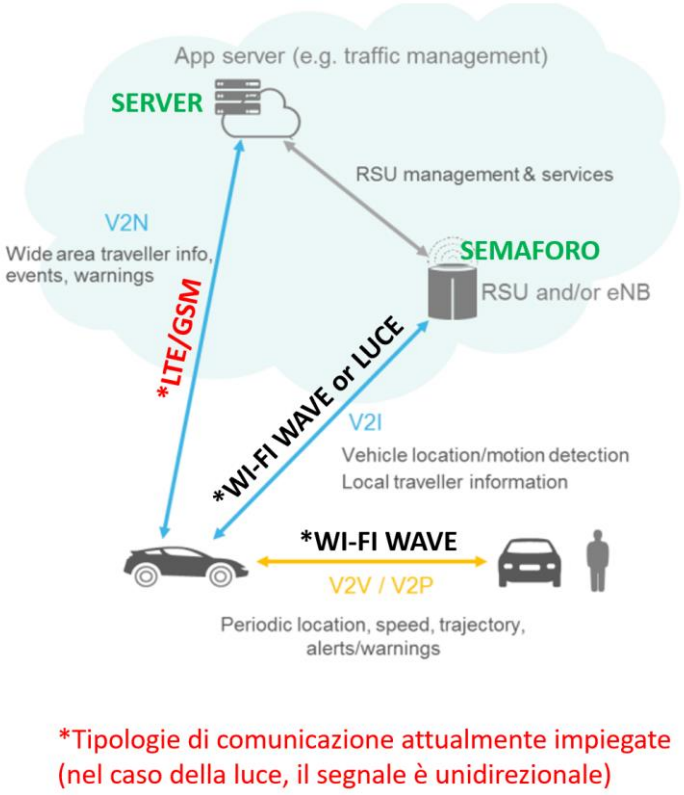

Figura 3 - V2X tipologie e connessioni

La figura mostra i diversi mezzi di comunicazione impiegati a seconda della tipologia di V2X presa in considerazione.

Riguardo alle comunicazioni tra veicoli e veicoli, veicoli ed utenti stradali si utilizzano protocolli di comunicazioni analoghi a quelli delle reti wireless domestiche ed industriali le quali utilizzano lo spettro delle frequenze dei *2,4 GHz* (suddiviso in *14* canali con ampiezza *20MHz* o *40MHz*) o dei *5 GHz* (suddiviso in *30* canali con ampiezza da *20 MHz* fino ad un massimo di *160 MHz*).

Le comunicazioni tra veicoli ed infrastrutture possono avvenire come per quelle precedenti utilizzando protocolli *Wi-Fi* oppure mediante l'utilizzo di segnali luminosi, ma in questo caso la direzione del segnale è univoca.

Infine per le comunicazioni tra veicoli e rete si utilizzano protocolli come *LTE* (*4G*) o la quinta generazione degli standard di telefonia mobile (*5G*) attualmente in fase di sperimentazione.

1.3.1 Wireless Access in Vehicular Enviroments (Wave) Standard IEEE 802.11p

Le comunicazioni wireless del tipo V2V e V2I impongono dei nuovi requisiti rispetto ai normali standard utilizzati in ambito domestico o industriale (IEEE 802.11) difatti le applicazioni di sicurezza delle autovetture non possono tollerare il ritardo dovuto alla stabilizzazione del collegamento prima di essere abilitati a parlare con gli altri veicoli incrociati per strada. Allo stesso modo lo scambio di informazioni con le infrastrutture deve essere il più veloce possibile perché il tempo di permanenza del veicolo nell'area è molto limitato. Per questi motivi lo standard IEEE 802.11 non è stato giudicato adatto a questo tipo di situazioni, di conseguenza è stato creato un nuovo standard: ***IEEE 802.11p*** che non è altro che un aggiustamento del protocollo già presente, ma dedicato al Wireless Access in Vehicular Environment (WAVE).

Lo spettro delle frequenze utilizzato è quello dei *5,9GHz* con un range di *75MHz* (da 5,85GHz a 5,925GHz) ed è suddiviso in sette canali da *10MHz*.

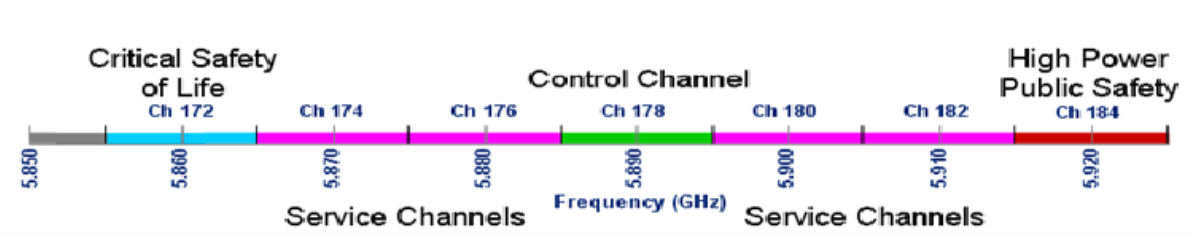

Figura 4 - Spettro DSRC

La banda di guardia è di *5MHz* che non viene utilizzata per prevenire i disturbi. Il canale *178*, denominato Canale di Controllo (*CCH*, Control Channel) compreso tra le frequenze *5,885GHz* e *5,895GHz* è riservato esclusivamente alle comunicazioni di sicurezza il quale diffonde gli allarmi e i beacon. Il suddetto canale è l'unico condiviso tra tutti i dispositivi Wave e realizza un punto di incontro tra i nodi. I canali *174*, *176*, *180* e *182* sono i Canali si Servizio (*SCH*, Service Channel) utilizzati dalle normali applicazioni e comunicazioni. I canali agli estremi dello spettro sono riservati ad usi speciali come la prevenzione di incidenti (*172*) e alla pubblica sicurezza (*184*).

Le differenze più grandi tra i due protocolli sono la frequenza utilizzata: *5GHz* per la IEEE 802.11a e 5,9GHz per la IEEE 802.11p e la durata di ogni singolo *OFDM* (Orthogonal Frequency-Division Multiplexing) che è raddoppiata da *4μs* a *8μs*. Le costanti temporali sono raddoppiate in modo da dimezzare la banda occupata a *10MHz* e raddoppiare il tempo di guardia tra singoli OFDM.

L'utilizzo di questo tipo di tecnologie richiede un hardware dedicato all'interno dei veicoli e nel caso di comunicazione V2I si può utilizzare per collegare ad internet i diversi veicoli e quindi far svolgere al semaforo la funzione di un router [23-29].

1.3.2 Protocollo Vehicle-to-Infrastructure (V2I)

Nel campo automobilistico alcuni produttori hanno lanciato o stanno testando nuovi sistemi di comunicazione tra veicolo ed infrastrutture. Prima fra tutte Audi, la quale ha immesso sul mercato una serie di veicoli in grado di comunicare con i semafori (per ora di Las Vegas) in modo da ricevere informazioni utili sul tempo di durata dei diversi stati semaforici. La tecnologia denominata Audi Traffic Light Information (*Audi TLI*) si basa sull'utilizzo di una connessione ad internet per ottenere queste informazioni ed è disponibile per alcuni modelli prodotti a partire da giugno 2017; la vettura deve avere a disposizione il modulo LTE/4G ed il display Audi Virtual cockpit [30][31].

Figura 5 - Audi TLI

Altri brand mondiali nella produzione di automobili come Ford ed il centro tecnico europeo della Tata Motors (detentore dei marchi Jaguar e Land Rover) stanno collaborando per mettere a punto nuove tecnologie che permettano di ricevere informazioni dai semafori e che possano comunicare fra loro. L'obbiettivo comune è quello di informare il guidatore di possibili ingorghi, di suggerirgli la via più rapida per raggiungere la destinazione oppure di indicargli la velocità da mantenere per sfruttare la cosiddetta "onda verde" [32-34].

Il colosso automobilistico asiatico Hyundai Motor Group (che ingloba i brand Hyundai e Kia) punta non solo a creare sistemi avanzati di guida, ma anche di intrattenimento. La loro idea di auto del futuro non è solo quella che è in grado di guidare autonomamente, ma che fornisca un servizio di assistenza e manutenzione remoto in modo da prevenire i disagi con una preventiva comunicazione dello stato di usura dei componenti, un network di analisi del traffico aggiornato in tempo reale ed infine di dare all'auto la possibilità di dialogare con tutto ciò che la circonda in modo da ottenere informazioni per la sicurezza ed il comfort [35].

General Motors tramite il marchio Cadillac ha introdotto sul mercato la prima automobile dotata di tecnologia V2V e V2I testandone le capacità con la collaborazione delle Agenzie stradali del Michigan. Il test condotto ha dimostrato la capacità dell'auto di ricevere informazioni dai sistemi di monitoraggio del traffico e di informare il pilota che alla velocità sostenuta in quel momento avrebbe incrociato il semaforo rosso. La tecnologia utilizzata da Cadillac si basa sull'utilizzo dei dati gps del veicolo e della DSRC la quale è in grado di gestire fino a mille messaggi al secondo fino a trecento metri di distanza [36] [37].

Il produttore Tesla non si occupa dell'utilizzo dei protocolli V2I e V2V, ma basa le sue ricerche sulla creazione della guida autonoma da parte dei veicoli utilizzando videocamere e sensori ad ultrasuoni [38].

1.3.3 Comunicazione tramite segnali luminosi led

Date le caratteristiche intrinseche superiori dei diodi led rispetto alle normali lampade ad incandescenza, il loro utilizzo è in ascesa in tutti i campi in cui è previsto l'uso di segnali luminosi, anche in quello semaforico. Infatti i led garantiscono prestazioni superiori rispetto alle lampadine classiche dato che presentano:

- basso consumo energetico
- elevata affidabilità
- lunga durata di vita
- alta visibilità luminosa
- alta tolleranza all'umidità
- scarsa produzione di calore
- la possibilità di essere commutati a frequenze molto elevate

Date tutte queste premesse si può affermare che sono i dispositivi ideali per essere usati nei sistemi di comunicazione; di conseguenza le lanterne semaforiche led possono essere trasformate in trasmettitori e siccome la luce segue un percorso lineare è possibile inviare informazioni differenti per ogni direzione di marcia.

I led sono usati come emettitori mentre come ricevitori si hanno due possibilità:

- la prima consiste nell'utilizzare un fotodiodo, ma in questo caso non è possibile adottare la comunicazione in parallelo di più informazioni perché questo dispositivo non è in grado di rilevare e separare la luce da due o più trasmettitori. Le tre lampade vengono quindi modulate nello stesso modo.
- la seconda consiste nell'utilizzare una telecamera ottica ad alta velocità che permette di riconoscere la sorgente che emette il segnale e quindi garantisce la comunicazione in parallelo di più informazioni [39-42].

1.4 Protocollo di comunicazione Bluetooth e Low Energy (BLE)

Il Bluetooth è uno standard tecnologico per la trasmissione dei dati su brevi distanze tramite una rete personale senza fili o WPAN (Wireless Personal Area Network).

Questa tecnologia è stata creata e sviluppata nel 1994 dalla *Ericsson*; nel 1998 viene formalizzata dalla Bluetooth Special Interest Group (*SIG*), società costituita dalla Sony Ericsson, IBM, Toshiba, Nokia e altre società minori. La IEEE ha creato lo standard IEEE 802.15.1, che descrive le specifiche di questa tecnologia, ma oggi lo sviluppo, il controllo e l'utilizzo del marchio è gestito dalla SIG. Qualunque produttore che vuole utilizzare questo marchio deve rispettare gli standard imposti da questa società.

Questa tecnologia opera nello spettro delle frequenze dei *2,45GHz* con un range di *80MHz* (da *2,402GHz* a *2,480GHz*, tenendo conto delle bande di guardia all'inizio ed alla fine della banda) suddivisa in *79* canali da *1MHz* ciascuno; questo range di frequenze è libero da ogni licenza di utilizzo ed è denominato ISM (industrial Scientific Medical). Il Bluetooth utilizza una tecnica trasmissiva di tipo FHSS, in cui la frequenza non è fissata, ma varia passando da un canale all'altro ad intervalli regolari in modo pseudocasuale. Il data-rate lordo può variare da *1Mbps* fino a *3Mbps* a seconda della modalità in cui si trasferiscono i dati. Il raggio di trasmissione è di circa una decina di metri, una distanza limitata dall'esigenza di contenere la potenza di trasmissione e, di conseguenza, il consumo di energia. Le altre caratteristiche importanti di questa tecnologia sono l'ingombro ridotto, la sicurezza delle connessioni ed il basso costo dei componenti.

Con il passare degli anni lo standard Bluetooth si è evoluto, migliorando le caratteristiche e le capacità di questa tecnologia; oggi la versione più evoluta è la *5.0* che rispetto alla versione precedente (*4.2*) presenta un range di trasmissione quadruplicato fino a raggiungere un'estensione massima di *200m*, la velocità in trasmissione in modalità basso consumo è stata raddoppiata fino a *4Mbps* e la capacità di trasmissione è aumentata di ben otto volte.

Riguardo alle modalità di funzionamento degli ultimi standard, possiamo dividerle in due tipologie:

Basic Rate/Enhanced Data Rate (BR/EDR): modalità classica che permette il collegamento wireless continuo ed utilizza una tipologia di rete point-to-point (*P2P*). Questa configurazione è ottimizzata per la riproduzione di file audio in streaming tramite cuffie, speaker e sistemi audio implementati nell'automobili compatibili con la tecnologia Bluetooth.

Low Energy: modalità a basso consumo energetico che utilizza diversi profili di funzionamento:

- Point-to-point (P2P): permette le comunicazioni one-to-one (1:1) con i dispositivi. Ideale per i trasferimenti di dati e per il collegamento di dispositivi realizzati per il fitness, il monitoraggio della salute e gli accessori Pc.

- Broadcast: permette le comunicazioni one-to-many (1:m). Ottimizza la condivisione di informazioni localizzare ed è ideale per l'utilizzo dei beacons.

- Mesh: permette le comunicazioni many-to-many (m:m). Questa modalità crea reti su larga scala ed è ideale per la domotica, per l'utilizzo di sensori wireless e di dispositivi di tracciamento.

Il Bluetooth Low Energy (*BLE*), commercializzato come Bluetooth Smart, è stato sviluppato da Nokia nel 2006 con il nome di Wibree, ma solo nel 2010, con l'adozione della specifica 4.0, è entrato a far parte degli standard Bluetooth. Ad oggi, la versione più moderna di questa tecnologia è la 5.0 presentata ufficialmente il 16 giugno del 2016.

Le caratteristiche più importanti del BLE sono le seguenti:

- Le ridotte dimensioni dei dispositivi
- Rapido setup: tempo di discovery di pochi ms
- Basso consumo della batteria
- Portata superiore al Bluetooth classico

Dal confronto tra la modalità "Classica" e quella Low Energy (Smart) otteniamo i seguenti risultati [43-47]:

Technical specification	**Classic Bluetooth**	**Bluetooth Smart**
Distance/range	100 m	>100 m
Over the air data rate	1–3 Mbit/s	125 kbit/s – 2 Mbit/s
Application throughput	0.7–2.1 Mbit/s	0.27 Mbit/s
Active slaves	7	Not defined;
Latency (from a non-connected state)	Typically 100 ms	6 ms
Minimum total time to send data (det. battery life)	100 ms	3 ms
Power consumption	1 W	0.01 – 0.50 W
Peak current consumption	<30 mA	<15 mA

Figura 6 - Tabella confronto Bluetooth Classic e BLE

Cap.2: Dispositivo per la comunicazione real-time delle transizione di fase

2.1 Generalità

In questo capitolo è descritta la realizzazione del prototipo di un dispositivo elettronico da installare facilmente nelle centraline semaforiche, in grado analizzare in real-time, tramite varie misure, lo stato in cui si trova la lanterna semaforica, elaborare e comunicare le informazioni digitali agli utenti interessati.

Un esempio applicativo di questa tecnologia è quello di fornire ausilio ad una persona non vedente o ipovedente, in modo da favorire la loro autonomia nell'attraversamento pedonale, fornendo loro uno strumento, che gli permetta di svolgere questa azione in sicurezza. Un secondo impego possibile è quello di monitoraggio dei cicli semaforici, in modo da raccogliere più dati possibili sulla durata di ogni stato semaforico, così da poter consigliare agli automobilisti la strada più veloce da percorrere o l'andatura da tenere per limitare il consumo di carburante ed il tempo di attesa in coda al semaforo. Un'altra possibile applicazione è quella di agevolare la manutenzione dell'impianto: nel caso in cui una lampada fosse guasta o l'impianto fosse fuori servizio a causa di un malfunzionamento, si potrebbe comunicare a chi di dovere l'anomalia rilevata in tempo reale e migliorare di conseguenza i tempi di attesa per le eventuali riparazioni.

Le basi su cui si fonda tutto il lavoro descritto in questo elaborato sono le seguenti:

- Realizzare un prototipo galvanicamente isolato rispetto all'impianto semaforico.
- Contenere il più possibile i costi di realizzazione.

Il primo punto è fondamentale per ottenere i permessi necessari per l'istallazione del dispositivo sugli impianti semaforici, infatti questo significa realizzare un dispositivo fisicamente separato con l'altro circuito e che non influisca in alcun modo sul normale funzionamento della centralina, anche in caso di guasto del prototipo stesso.

Riguardo al secondo punto, per poter essere competitivi su questo tipo di mercato bisogna utilizzare componenti che a parità di prestazioni hanno costo minore, senza però rinunciare all'affidabilità ed alla sicurezza.

Per poter conciliare questi due punti è stata condotta una ricerca su quali siano i dispositivi presenti sul mercato che soddisfino entrambe le richieste.

Nella prima parte di questo capitolo si andranno ad analizzare i metodi teorici di misura di tensione e corrente ed i vari componenti utilizzati per eseguire tali tipi di operazioni. Dopo di che si tratterà l'analisi dell'impianto semaforico, la realizzazione del circuito di simulazione ed infine la realizzazione del primo prototipo, le relative evoluzioni e le prove eseguite su un vero incrocio funzionante.

2.2 Elementi di misure elettriche

La conoscenza di ogni fenomeno si basa sulla misura delle grandezze che lo caratterizzano.

Il termine misura indica, usualmente, sia il procedimento di misurazione che trasforma la grandezza da misurare o misurando in un'altra percepibile, sia il risultato di essa, che fornisce il valore del misurando.

Il risultato della misurazione, ossia la misura, si esprime con un numero dato dal rapporto tra il misurando e un'altra grandezza ad esso omogenea assunta come unità di riferimento. Si valuta poi la qualità della misura determinando un intervallo di incertezza in cui si garantisce il valore del misurando (precisione, o incertezza).

Le modalità di misurazione si distinguono in dirette e indirette, in cui la prima si ottiene dal confronto, eseguito con strumenti e metodi appropriati, del misurando con un'altra grandezza della stessa specie assunta come campione di riferimento o unità di misura. Una misura indiretta invece fornisce il misurando tramite la misura di altre grandezze anche non omogenee con il misurando stesso, ma legate ad esso da una relazione matematica.

Uno strumento di misura è un apparecchio singolo o un sistema di più apparecchi idoneo a riconoscere il misurando (entrata o ingresso) dalla sua immagine (uscita). La corrispondenza misurando immagine stabilita dallo strumento si effettua attraverso successive elaborazioni di un segnale che deve essere funzione iniettiva del misurando. Il numero e la natura delle elaborazioni del segnale di misura dipendono dal misurando e dallo strumento di misura. Il funzionamento di uno strumento di misura si può descrivere con una catena di elementi funzionali, entrata-uscita, che ricevano e trasmettano grandezze sia omogenee che non omogenee.

Gli elementi funzionali che compongono questa catena sono:

- Sensore o sonda
- Convertitore
- Comparatore
- Amplificatore
- Trasmettitore
- Visualizzatore (display)
- Registratore (memoria)
- Elaboratore numerico (microprocessore)

I quali caratterizzano lo schema funzionale di uno strumento di misura:

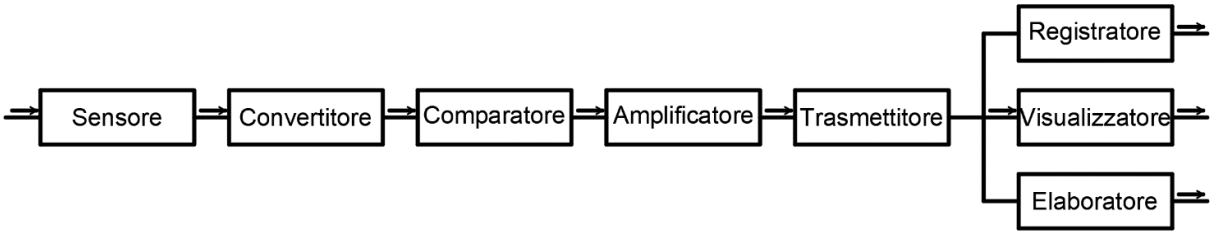

Figura 7 - Schema funzionale di uno strumento di misura

Lo schema è utile nello studio dello strumento, per l'esame del suo funzionamento e per l'analisi degli errori introdotti dai suoi componenti.

In base alla natura del segnale di misura gli strumenti si distinguono in analogici e numerici. Negli strumenti analogici il segnale di misura è una funzione continua del misurando, mentre negli strumenti numerici o digitali il segnale è una funzione discreta del misurando, cioè assume solo un numero finito di differenti valori in un determinato campo.

I digitali rispetto agli strumenti analogici hanno i seguenti vantaggi:

- Eliminazione dell'errore di lettura
- Letture più veloci e quindi maggiore capacità di misura
- Precisione più elevata
- Possibilità di interfaccia diretta degli strumenti con elaboratori numerici.

Sono possibili altre classificazioni degli strumenti di misura:

- in base alla struttura possono essere elettromeccanici o elettronici
- in base al modo di funzionamento possono essere strumenti a deviazione, oscillografici, selettivi, integratori, derivatori, balistici e strumenti a campionamento.

Anche i metodi di misura si possono raggruppare con vari criteri. Una classificazione distingue due grandi categorie di metodi di misura: a indicazione e di confronto.

La scelta del metodo di misura dipende da vari fattori tra cui: la natura del misurando, il tipo di misura se statica o dinamica, la precisazione desiderata, il tempo richiesto per la misura, il costo.

L'esame dello schema in *figura 7* fornisce alcuni criteri di base per la costruzione di uno strumento di misura. L'inserimento dello strumento nel misurando ne provoca sempre una modifica, perciò non si può conoscere sperimentalmente il valore vero del misurando (principio indeterminazione di Heisenberg). In pratica si deve rendere minimo il consumo dello strumento, ossia la potenza che lo strumento preleva dal misurando per attivare il segnale di misura. Il segnale di misura nel passaggio da uno stadio all'altro dello strumento deve trasferirsi con la massima informazione, ciò può richiedere che il segnale si trasmetta con la massima tensione o la massima potenza. Inoltre il segnale di misura deve essere immune da altre grandezze di influenza di origine interna o esterna, disturbi. Questi devono quindi essere ridotti o neutralizzati con adeguate tecniche costruttive e circuitali, fino a contenere il loro effetto nella fascia di errore accettata per lo strumento.

La misura di qualunque grandezza presenta inevitabilmente un errore. **Errore assoluto *ΔX*** è la grandezza dimensionata data dalla differenza algebrica tra il valore misurato *X* ed il valore esatto o effettivo X_0:

$$\Delta X = X - X_0$$

Il valore X_0 è anche chiamato valore vero, però il valore vero non è determinabile con l'esperienza, quindi nella formula precedente X_0 rappresenta un valore di riferimento ottenibile da una misurazione la più perfetta possibile.

Errore relativo ε è il rapporto adimensionale tra l'errore assoluto *ΔX* e il valore X_0:

$$\varepsilon = \frac{X - X_0}{X_0}$$

Questo errore si può esprimere anche il percentuale:

$$\varepsilon = \frac{\Delta X}{X_0} \cdot 100$$

A parte gli errori grossolani come inserzioni errate o strumenti non tarati, gli errori si classificano in **sistematici** ed **accidentali**, ove i primi sono quelli che si mantengono costanti in modulo e segno quando si ripeta la misurazione nelle medesime condizioni. Mentre gli accidentali sono quelli che variano in modulo e segno quando si ripeta la misurazione nelle medesime condizioni.

La bontà della misura si caratterizza globalmente con la **precisone** o **incertezza** data dall'errore, assoluto o relativo, ad esso associato.

2.3 Metodi di misura

2.3.1 Trasformatori di misura

I trasformatori di misura si impiegano negli impianti elettrici a frequenza industriale per variare la portata degli strumenti di misura e per alimentare apparecchiature di controllo, di manovra e di interruzione.

Sono di due tipi: trasformatori di corrente o amperometrici (TA) e trasformatori di tensione o voltmetrici (TV); i primi lavorano connessi in serie sulla linea, lavorano a corrente primaria impressa, invece i TV, connessi in parallelo sulla linea, lavorano a tensione imposta come mostrato nello schema che segue:

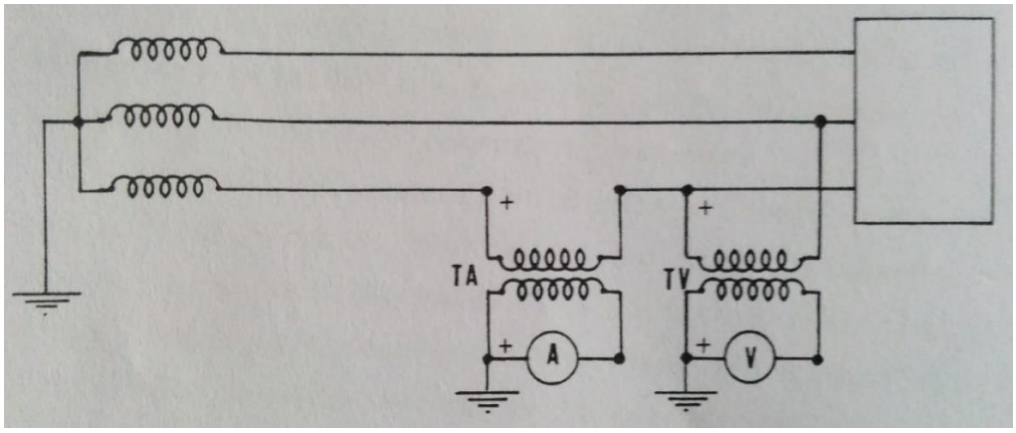

Figura 8 - Inserzione dei TA e TV

I trasformatori di misura negli impianti a tensione superiore al migliaio di volt si usano, in primo luogo, per motivi di sicurezza del personale e delle apparecchiature. L'avvolgimento secondario a cui sono connessi gli strumenti può essere messi a terra in modo da evitare che il potenziale elettrostatico da esso assunto raggiunga valori elevati. L'uso dei trasformatori di misura presenta altri vantaggi:

- Evita difficoltà di costruire: sia shunt indipendenti dalla frequenza, sia partitori resistivi non reattivi e indipendenti dalle resistenze di dispersione.
- Permette una normalizzazione delle portate degli strumenti di misura (es. i secondari dei TA sono per correnti di 5A, mentre quelli dei TV sono per tensioni di 100V).
- Riduce il consumo degli strumenti.
- Rende agevole tramite cavetti a bassa tensione l'installazione degli strumenti, anche più di uno, sul secondario di un unico trasformatore di misura in posti distanti decine di metri dal punto di misura.
- Permette misure anche con precisione elevata.

2.3.1.1 Circuito equivalente e tipologie di funzionamento

Il circuito equivalente di un trasformatore monofase a frequenza industriale è il seguente:

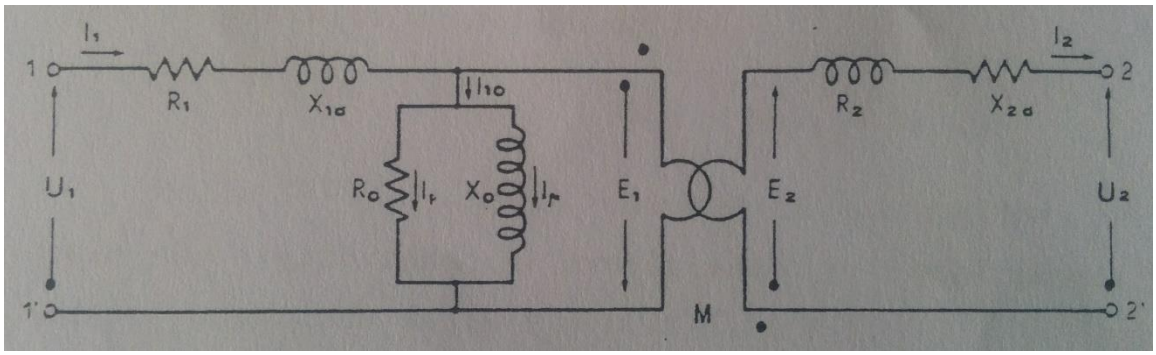

Figura 9 - Circuito equivalente di un trasformatore reale

Il significato degli elementi rappresentati è riportato qui di seguito:

- M trasformatore ideale, cioè mutua induttanza con coefficiente di accoppiamento unitario tra i due circuiti elettrici di resistività nulla, concatenati tramite un circuito magnetico di permeabilità infinita senza perdite. Per tale trasformatore valgono le relazioni:

  tensioni: $U_1 = a \cdot U_2$

  correnti: $I_1 = \frac{I_2}{a}$

  potenze: $P_1 = P_2$

  Ove $a = {}^{N_1}/_{N_2}$ è il rapporto tra le N_1 spire del primario e le N_2 spire del secondario.

- R_1 e R_2 rispettivamente resistenza del primario e del secondario
- X_{1d} e X_{2d} rispettivamente reattanza di dispersione del primario e del secondario
- R_0 resistenza equivalente alle perdite non nulle del circuito magnetico
- X_0 reattanza di magnetizzazione

Si distinguono due modi di funzionamento del trasformatore dipendenti dalla sua connessione sulla linea di alimentazione a frequenza costante.

Se il **trasformatore è alimentato a tensione costante**, caso degli ordinari trasformatori di tensione TV connessi in parallelo sulla line a di alimentazione, la corrente di eccitazione I_{10} coincide praticamente con la corrente primaria a vuoto.

Se il **trasformatore è alimentato a corrente primaria imposta** ed è chiuso su un'impedenza costante, caso dei trasformatori di corrente TA connessi in serie sulla linea di alimentazione, per ogni valore della corrente primaria I_1, la corrente di eccitazione I_{10} assume valori che provocano nell'avvolgimento primario e secondario corrispondenti valori di tensione.

2.3.1.2 Trasformatori di corrente TA

I trasformatori di corrente o amperometrici (TA) si impiegano per misure di correnti basse o elevate su linee ad alta tensione, oppure in misure di correnti elevate su linee a bassa tensione. Un TA è connesso con il primario in serie sulla linea percorsa dalla corrente che si vuole misurare o controllare. In primo luogo la corrente secondaria I_2 è praticamente indipendente dall'impedenza su cui è chiuso il secondario ed è determinata dalla corrente primaria I_1, che a sua volta dipende dall'impedenza del carico sulla linea di alimentazione. Per una data corrente primaria I_1, se l'impedenza secondaria del TA aumenta (diminuisce) deve corrispondentemente aumentare (diminuire) la tensione secondaria. L'impedenza su cui è chiuso il secondario deve essere molto bassa per far funzionare il secondario del TA in condizioni prossime al corto circuito.

Per costruire un TA che abbia bassi errori si devono utilizzare molto rame e molto ferro di buona qualità, quindi con un costo dei materiali molto alto.

Si costruiscono **TA a barra passante** impiegati normalmente in misure di corrente sufficientemente intense (centinaia di ampere). In questi trasformatori il primario è come se fosse costituito da una sola spira che attraversa l'anello magnetico che porta l'avvolgimento secondario:

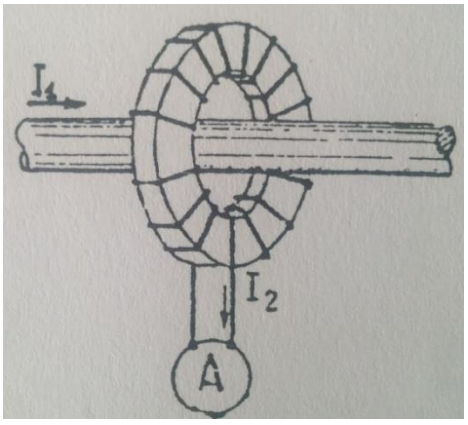

Figura 10 - TA a barra passante

Sullo stesso principio si realizzano TA a pinza o a tenaglia per racchiudere il conduttore percorso dalla corrente da misurare. Ciò rende possibile la misura di una corrente in un circuito senza richiedere la sua interruzione per l'inserimento dello strumento. Nel caso di tensioni elevate l'isolamento mutuo tra gli avvolgimenti deve essere elevato (TA in olio).

Come già detto il TA deve essere chiuso su un'impedenza secondaria molto bassa, perciò si deve evitare di aprire il secondario, altrimenti il circuito magnetico raggiungerebbe la saturazione, aumentando le perdite nel ferro e se l'apertura fosse continua si potrebbe avere un riscaldamento eccessivo che può portare al deterioramento dell'isolante tra gli avvolgimenti. Tra i morsetti del secondario aperto si ha poi una sovratensione che può essere pericolosa per le persone e può provocare scariche tra le spire dell'avvolgimento. Inoltre nel primario si introduce una caduta di tensione nella linea molto più elevata di quella normale.

Riguardo alle protezioni non è buona norma inserire fusibili di corrente sul secondario del TA e se il TA è inserito sulla linea e non è utilizzato è bene chiudere in corto circuito il secondario. Per protezione si può inserire tra i morsetti una valvola di tensione che interviene se la tensione supera una data soglia e chiude in corto circuito il secondario.

2.3.1.3 Trasformatori di tensione TV

I trasformatori di tensione o voltmetrici Tv si impiegano per alimentare da una linea ad alta tensione voltmetri, bobine voltmetriche di wattmetri etc. Un TV è connesso in parallelo alla linea, perciò il funzionamento è simile a quello dei normali trasformatori di potenza. In particolare se la tensione di alimentazione è costante, il flusso nel nucleo magnetico è circa costante.

Per costruire un TV che abbia bassi errori si devono usare elevata sezione in rame e massimo coefficiente di accoppiamento tra primario e secondario. Per limitare la caduta di tensione sul secondario è necessario limitare la corrente I_2, perciò l'impedenza del carico non deve scendere al di sotto di un valore limite. La condizione ideale si ha con un'impedenza del carico tendente all'infinito, cioè con il secondario aperto.

Come per il TA, anche per il TV il problema della precisione è solo economico di costo del trasformatore.

Normalmente un TV per la misura della tensione di linea presenta al primario due isolatori passanti proporzionati per sostenere la tensione primaria, questo perché un problema preminente in questo tipo di trasformatori è quello dell'isolamento per tensioni elevate (fino a centinaia di kV). Questi problemi si possono attenuare con l'impiego di Tv connessi a valle di un **riduttore capacitivo** che, inoltre permette di risolvere in parte i problemi che si hanno con le tensioni a fronte ripido.

2.3.2 Partitori, sonde di tensione e di corrente

I **partitori o divisori di tensione** si impiegano per aumentare la portata di tensione di strumenti di misura elettromeccanici ed elettronici. Il tipo più semplice di partitore di tensione è costituito da resistori connessi in serie allo strumento come da schema:

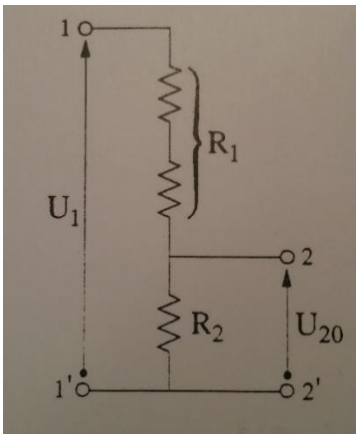

Figura 11 - Partitore di tensione resistivo

La tensione di uscita risulta:

$$U_{20} = \frac{R_2}{R_1 + R_2} \cdot U_1$$

Per ottenere una maggiore risoluzione nella partizione della tensione, si collegano in cascata fra loro più partitori resistivi.

I parametri parassiti più temibili sono le capacità parassite tra gli elementi del partitore e fra gli elementi a massa. Per limitarne l'influenza si ricorre a **partitori di tensione compensati** (usando delle capacità note). Questi partitori vengono sono impiegati sia come attenuatori all'ingresso di strumenti elettronici, sia come **sonde di tensione**.

I **partitori o divisori di corrente** si impiegano per aumentare la portata in corrente di strumenti di misura elettromeccanici ed elettronici. Il più semplice partitore di corrente è costituto da uno shunt o derivatore, connesso in parallelo allo strumento di misura. Questi strumenti impiegano anche altri tipi di shunt, come lo **shunt universale** mostrato nella figura che segue:

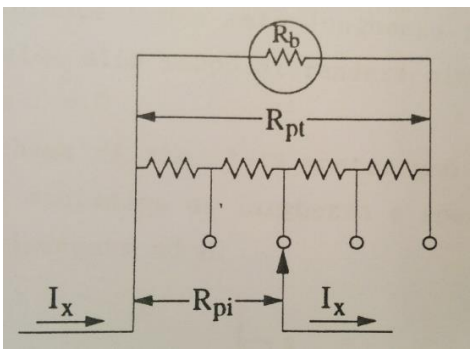

Figura 12 - Shunt universale

Dallo schema risulta che, per ciascun valore di R_{pi}, vi è una proporzionalità tra la corrente misurata I_x e la corrente I che percorre la bobina:

$$I_x = \frac{R_b + R_{pt}}{R_{pi}} \cdot I = m \cdot I$$

Con m potere **moltiplicativo dello shunt** universale inversamente proporzionale a R_{pi}.

Gli shunt precedenti si usano in corrente continua, in corrente alternata a frequenze industriali normalmente si impiegano trasformatori di corrente TA. Quando in corrente alternata o in regime variabile è richiesto l'uso di uno shunt, questo deve presentare induttanza e capacità parassite minime per ottenere una tensione in uscita indipendente dalla frequenza.

Per le analisi di forme d'onda di corrente effettuate con strumenti analogici e digitali, si impiegano adatte pinze amperometriche dette **sonde di corrente**. Queste sono costituite da un piccolo trasformatore di corrente con nucleo magnetico apribile, in modo da concatenare il conduttore percorso dalla corrente che si vuole analizzare.

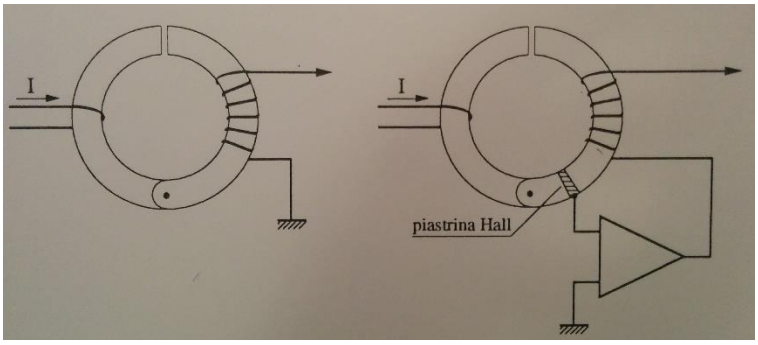

Figura 13 - Sonde di corrente

Lo schema a sinistra mostra una sonda di corrente descritta in precedenza, con materiali magnetici speciali si arriva a sensibilità di 1mV/10mA con banda passante da 100Hz a 20MHz.

Per estendere la banda passante verso le basse frequenze fino a quella nulla si utilizzano sonde di corrente che sfruttano l'effetto Hall. Lo schema a destra mostra lo stesso schema a sinistra, ma in cui è stata incorporata una **piastrina ad effetto Hall** che fornisce in uscita una tensione proporzionale all'induzione B, a sua volta proporzionale alla intensità della corrente primaria in esame. Pertanto a una componente unidirezionale o a bassa frequenza della corrente primaria, la piastrina fa corrispondere una tensione proporzionale che, amplificata e connessa in serie con il secondario, provoca in uscita dalla sonda una tensione avente la stessa forma d'onda della corrente primaria.

2.3.3 Trasformatori di isolamento e accoppiatore opto-elettronico

Un **trasformatore di isolamento** è un dispositivo di accoppiamento in cui il flusso magnetico concatenato con gli avvolgimenti, spesso schermati tra loro, trasmette l'informazione dal segnale d'ingresso all'avvolgimento secondario in uscita, che è metallicamente "isolato" dall'avvolgimento primario. Questi trasformatori presentano: buona linearità, piccola capacità C di accoppiamento tra gli avvolgimenti (inferiore a *0,1pF*), frequenze di lavoro da una decina di hertz a una decina di kilohertz.

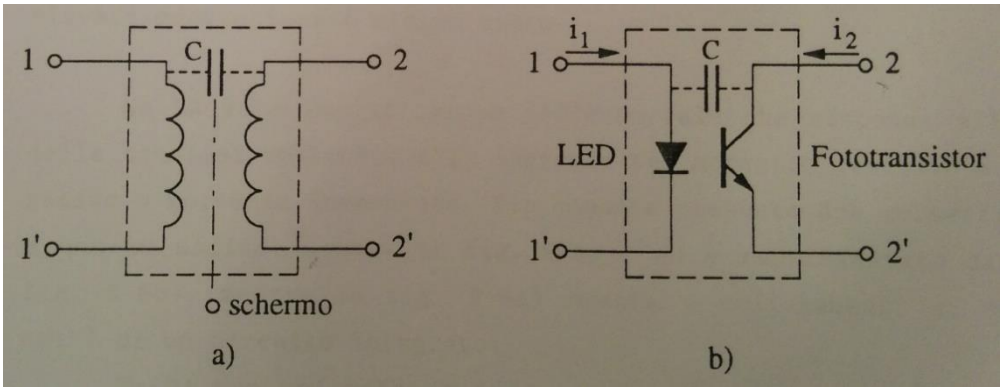

Figura 14 - a) trasformatore di isolamento; b) accoppiatore opto-elettronico

In un **accoppiatore opto-elettronico** un diodo emettitore di luce LED ("Light Emetting Diode") è posto in prossimità di un fototransistore in una singola struttura. La luce emessa dal diodo, provocata dalla corrente i_1, colpisce il fototransistor che determina la corrente i_2. L'informazione contenuta nella corrente i_1 è quindi trasferita al fototransistor (elemento fotorilevatore) senza nessun collegamento elettrico diretto. Tra led e fototransistor si hanno capacità di circa *1pF*; rispetto ai trasformatori di isolamento si ha il vantaggio di funzionamento in un campo di frequenze da 0 a una decina di kilohertz [48].

2.4 Analisi di un impianto semaforico di base

La prima operazione svolta nella fase di progettazione è quella di analizzare le grandezze elettriche che caratterizzano un impianto semaforico reale. L'azienda torinese IREN S.p.a ci ha gentilmente concesso due tipologie di lanterne che attualmente vengono utilizzate negli impianti semaforici di ultima generazione. Le caratteristiche principali sono le seguenti:

- le tre lampade di ciascuna lanterna utilizzano la tecnologia led per l'illuminazione
- la potenza assorbita da ogni lampada è di circa *10W* (precisamente *10W* per quella rossa e *8W* per quella gialla e quella verde)
- la tensione di alimentazione è differente, *230V* per la prima e *42V* per seconda
- la frequenza a cui vengono alimentate è di *50Hz*

Ogni lampada al suo interno presenta:

- un circuito di conversione AC/DC che trasforma la tensione alternata in continua e la adatta ai valori compatibili con il funzionamento dei led
- un sistema di filtraggio delle armoniche che elimina quelle dannose create dal convertitore in modo da evitare che esse si propaghino nella rete pubblica

Per avere un'idea dell'ordine di grandezza della corrente assorbita da ogni lampada di entrambe le tipologie di lanterne si può utilizzare la **terza legge di Ohm**:

$$P = V \cdot I$$

Nel caso di lanterna alimentata con tensione di *230V*:

$$I = \frac{P}{V} = \frac{10\ [W]}{230\ [V]} = 43 \cdot 10^{-3}[A] = 43[mA]$$

Nel caso invece di tensione di *42V* la corrente assorbita è:

$$I = \frac{P}{V} = \frac{10\ [W]}{42\ [V]} = 238 \cdot 10^{-3}[A] = 238[mA]$$

Ovviamente questi calcoli sono approssimativi, ovvero servono soltanto per capire quali sono le grandezze in questione e quale sia il caso peggiore da analizzare; risulterebbero veri, infatti, se la tensione e la corrente fossero perfettamente sinusoidali senza la presenza di armoniche di alcun tipo, in quel caso si sarebbe calcolato solamente il valore del modulo della corrente senza tener conto dello sfasamento.

Dai risultati ottenuti, visto la notevole differenza di corrente, il prototipo che verrà realizzato sarà configurato per misurare ed elaborare le correnti del caso peggiore ovvero per la lampada con tensione di alimentazione di *230V*. Si è fatta questa scelta perché se si riesce a realizzare un prototipo che riesce a gestire la situazione peggiore, quasi certamente, con le opportune modifiche funzionerà anche per il caso più semplice.

Per avere un'idea più precisa dell'ampiezza e della forma d'onda delle correnti in gioco si è utilizzato un oscilloscopio digitale posto in serie all'alimentazione della lampada, che ha mostrato il seguente risultato:

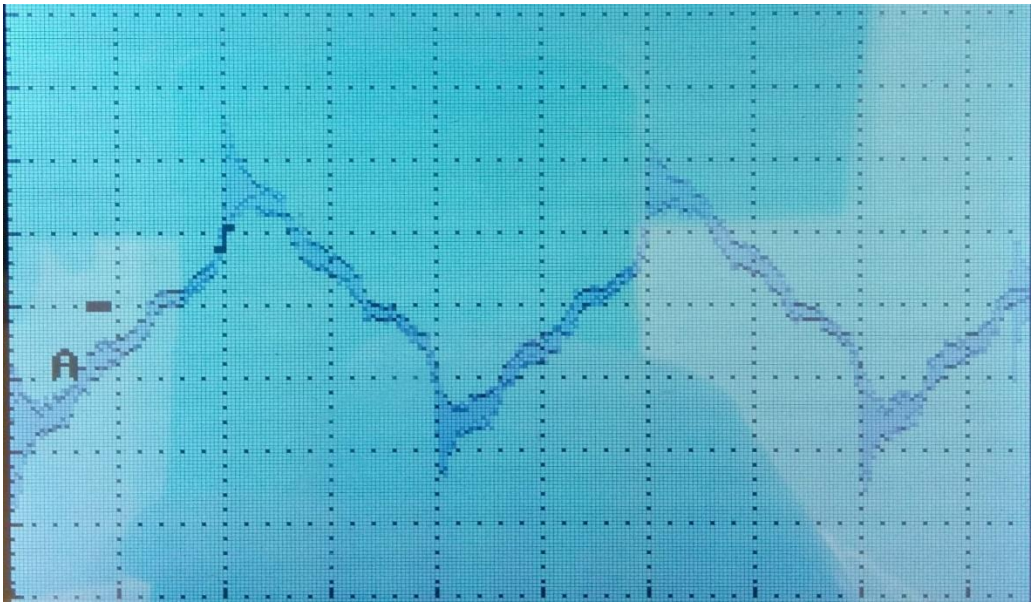

Figura 15 - Andamento della corrente di alimentazione della lampada 230V

Come si può vedere dall'immagine l'andamento della corrente è di tipo alternato, ma non sinusoidale per via della presenza del convertitore che introduce molte armoniche che i filtri non riescono ad eliminare. Il periodo completo ha una durata di *20ms* (frequenza di *50Hz*) e ogni semiperiodo è caratterizzato da un picco di corrente positivo o negativo preceduto da un'ascesa o discesa molto ripida, tipo quella di un impulso, e da una discesa o ascesa meno ripida e più costante.

A questa prima fase in laboratorio è seguita un'indagine su come sono strutturate le cabine in cui sono poste le centraline che controllano gli impianti semaforici. L'obbiettivo di questa ricerca è la comprensione di quali sono gli elementi installati all'interno, quali di essi possano essere utilizzati e gli spazi funzionali a disposizione del prototipo da costruire.

Grazie alla collaborazione con *Iren S.p.A*., il 12 maggio 2017 un tecnico addetto alla manutenzione ci ha mostrato l'interno della cabina che gestisce l'incrocio tra Corso Stati Uniti e Corso Duca degli Abruzzi a Torino.

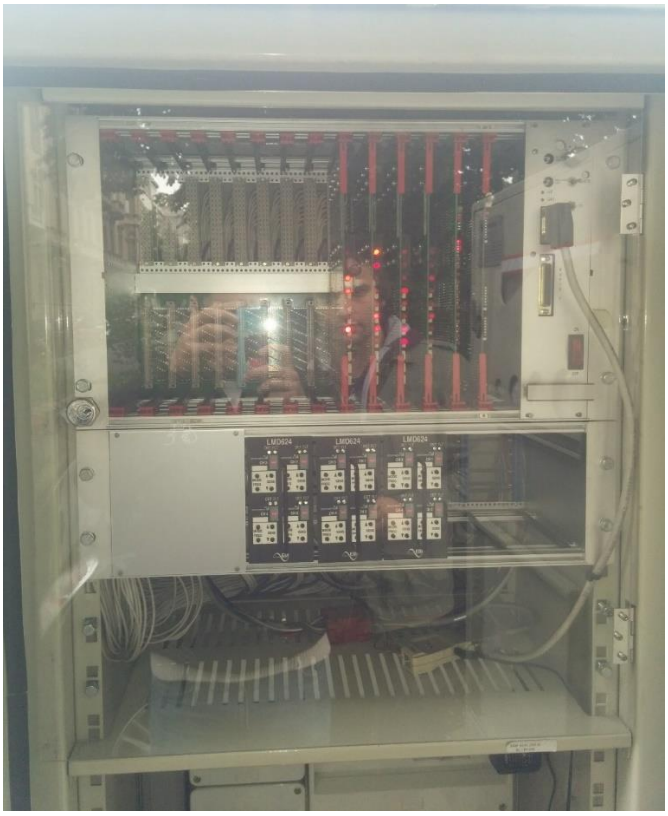

Figura 16 - Centralina Corso Stati Uniti-Corso Duca degli Abruzzi (1)

Figura 17 - Centralina Corso Stati Uniti-Corso Duca degli Abruzzi (2)

Figura 18 - Centralina Corso Stati Uniti-Corso Duca degli Abruzzi (3)

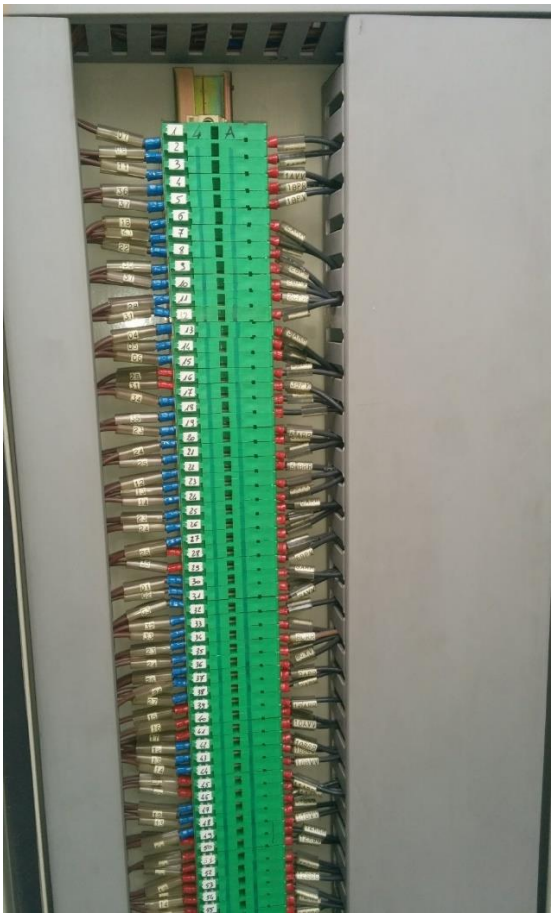

Figura 19 - Centralina Corso Stati Uniti-Corso Duca degli Abruzzi (4)

Dalle immagini precedenti si possono individuare cinque spazi funzionali in cui suddividere la cabina:

- il primo contiene il PLC che gestisce tutte le lanterne dell'incrocio ed è protetto da un vetro in modo da evitare che persone o oggetti di qualsiasi natura vengano a contatto con i componenti

- il secondo contiene i sistemi di sicurezza, i sistemi ausiliari e la morsettiera per il collegamento all'impianto di terra
- nel terzo si può trovare la morsettiera in cui sono collegati i cavi in uscita dal PLC che comandano i vari gruppi di lampade

- nel quarto è installata la morsettiera a cui in ingresso arrivano i cavi dei gruppi provenienti dai morsetti del punto precedente e dalla quale si diramano i cavi di ogni singola lampada

- del quinto spazio non vi è nessuna immagine, ma in esso c'è il pulsante che permette di mettere fuori servizio l'impianto da parte delle forze dell'ordine.

Lo spazio concessoci per l'installazione del dispositivo per i test da effettuare su incroci attivi è l'intero piano posto al di sotto del PLC; per l'alimentazione ci è stato concesso l'utilizzo della presa di servizio installata all'interno della cabina. Le protezioni elettriche da sovraccarico o cortocircuito sono quelle utilizzate dalla struttura stessa, quindi non è necessario dotare il prototipo di protezioni aggiuntive. Le prove verranno eseguite sulle tre lampade di una singola lanterna che gestisce il passaggio pedonale di uno dei controviali.

Da queste informazioni e dall'analisi della cabina, si deduce che, nella prima fase di sperimentazione, al prototipo non viene richiesto alcuna specifica riguardo alle dimensioni, al peso o alle modalità di alimentazione.

2.5 Realizzazione del primo prototipo

Dopo l'analisi delle grandezze elettriche e della cabina del sistema semaforico il passo successivo è stato la costruzione del prototipo di misura. Questa operazione si può dividere in due parti:

- la prima riguarda la scelta e l'assemblaggio dei componenti da utilizzare per la costruzione dell'hardware del prototipo da installare sul campo
- la seconda parte riguarda la programmazione del processore che dovrà leggere le grandezze, elaborarle ed inviarle agli utenti.

Iniziamo a vedere le scelte ed i componenti utilizzati per l'hardware.

2.5.1 Componenti e assemblaggio dell'hardware

Il prototipo fisico è costituito da tre diversi circuiti:

- Circuito di misura
- Circuito di elaborazione
- Circuito di invio dei dati

Per quanto riguarda il primo dei tre, l'isolamento galvanico, come anticipato all'inizio del *Capitolo 2*, è essenziale per poter installare il dispositivo nelle centraline semaforiche. Dalla teoria di misure elettriche si è dedotto che i dispositivi migliori per realizzarlo sono i trasformatori di misura di corrente o trasformatori amperometrici (TA); essi inoltre permettono di non interrompere il circuito dell'impianto già esistente, in quanto dotati di un foro in cui far passare il cavo di alimentazione della lampada. Dalla ricerca svolta il TA che si presta meglio al nostro scopo è il seguente:

Figura 20 - Trasformatore amperometrico (TA)

Prodotto dalla *Talema Electronic*, la taglia del trasformatore è *5A*, il rapporto di trasformazione è *1000:1*, la tensione di isolamento è *4kV* e la resistenza interna è di *42*Ω. (Per le specifiche complete guardare il datasheet allegato in appendice).

Sul mercato sono presenti trasformatori amperometrici e sensori ad Effetto Hall le cui taglie si avvicinano molto di più alla corrente da misurare (circa *40mA*), ma il costo di tali componenti è così elevato che permette a quello scelto di essere un ottimo compromesso tra qualità e costo del prodotto. Inoltre nel caso in cui allo stesso cavo di alimentazione fossero collegate più lampade in parallelo allora la taglia maggiore del trasformatore permette di continuare ad usare lo stesso circuito di misura senza dover cambiare i componenti.

Data la presenza di molte armoniche di corrente non è possibile eseguire una misura di corrente adeguata utilizzando solamente un sensore di corrente, un elaboratore ed al contempo mantenere bassi i costi dei componenti. Le possibili soluzioni a questo problema sono di utilizzare componenti con prestazioni migliori, ma dal costo elevato che riescano a campionare con sufficiente precisone il segnale oppure eliminare le armoniche dalla corrente dal circuito di misura. Siccome la prima possibilità non è attuabile perché viola i principi su cui si basa questo progetto, la scelta è ricaduta sulla seconda soluzione.

Per migliorare la lettura della grandezza e ridurre l'incidenza degli errori sul valore della corrente si è deciso di effettuare più avvolgimenti a primario. Questa operazione è fisicamente possibile in quanto il foro in cui deve passare il cavo di alimentazione è di dimensioni adeguate per compiere tale operazione, in altri TA presi in considerazione questo non era possibile per le scarse dimensioni del foro. In totale si sono ottenuti otto avvolgimenti a primario, di conseguenza la corrente ottenuta sul secondario è otto volte maggiore rispetto al caso di singolo avvolgimento.

Oltre al trasformatore, il circuito di misura è composto da un raddrizzatore **ponte a diodi** costituito da quattro **diodi zener** da *0,5W* e tensione di breakdown di *3,9V*. La scelta è ricaduta su questo tipo di diodi per avere una bassa caduta di tensione sul componente e quindi un consumo energetico ridotto. Il ponte a diodi viene utilizzato per invertire i valori di corrente negativi ed ottenere un andamento con soli valori positivi o al più nulli, in modo da facilitare la lettura della grandezza senza dover tenere conto del segno.

Il terzo componente del circuito è un **filtro passa-basso** costruito con una resistenza variabile da *0* a *5k*Ω e da una capacità di *100nF*. Questo è utilizzato per filtrare le armoniche ad alta frequenza e migliorare la qualità della corrente. Le formule utilizzate per la scelta dei componenti del filtro sono:

$$f_t = \frac{1}{2 \cdot \pi \cdot \tau}$$

Dove τ è:

$$\tau = \frac{1}{R \cdot C}$$

La costante di tempo (τ) è un'equazione con due incognite (*R* e *C*), quindi le possibili combinazioni delle due grandezze sono teoricamente infinite a parità di frequenza di taglio. Essendo un circuito di misura, la potenza assorbita deve essere trascurabile rispetto a quella di alimentazione; questo significa utilizzare valori di resistenza elevati per limitare la corrente, ma non abbassarli troppo in modo da poter rendere fattibile la lettura. Se invece si utilizzano

capacità con taglie troppo elevate si ottiene un segnale quasi continuo, ma il condensatore impiegherebbe troppo a scaricarsi quando viene disalimentato creando quindi degli errori sulla lettura. Dopo vari tentativi verificati tramite oscilloscopio si sono scelti i valori della resistenza e della capacità rispettivamente di *3kΩ* e di *100nF*. Dalle formule sopra riportate si ottiene una frequenza di taglio pari a *530Hz*.

Infine in parallelo al condensatore viene collegata una resistenza da *100KΩ* che permette al condensatore di scaricarsi ed ai capi della quale viene letta la tensione dell'elaboratore.

Lo schema riepilogativo del circuito di misura riferito ad una singola fase:

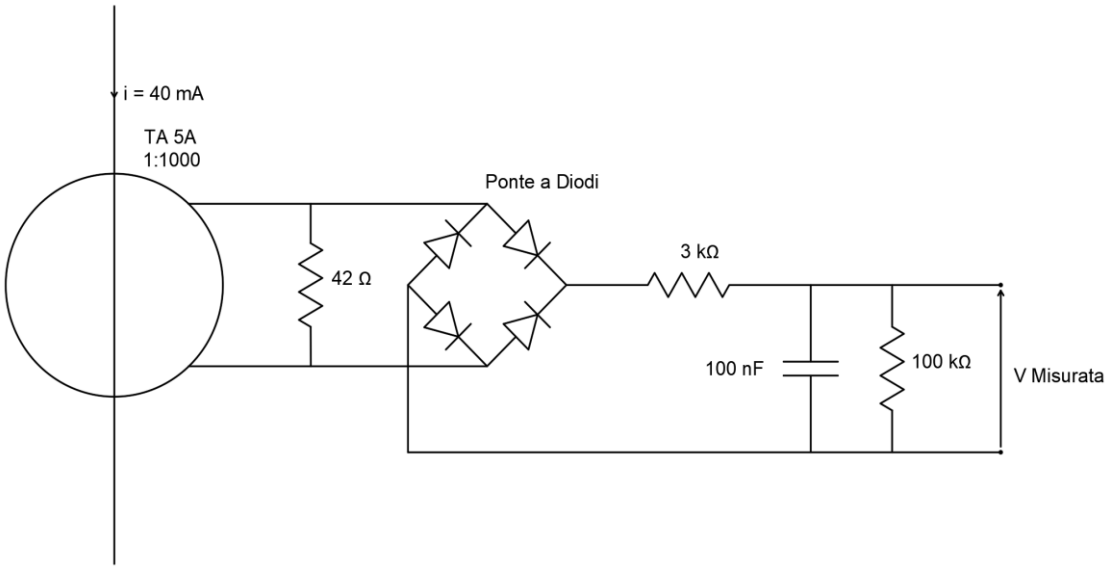

Figura 21 - Schema del circuito di misura della corrente primo prototipo

L'andamento della tensione ai capi della resistenza, ottenuto dall'oscilloscopio è il seguente:

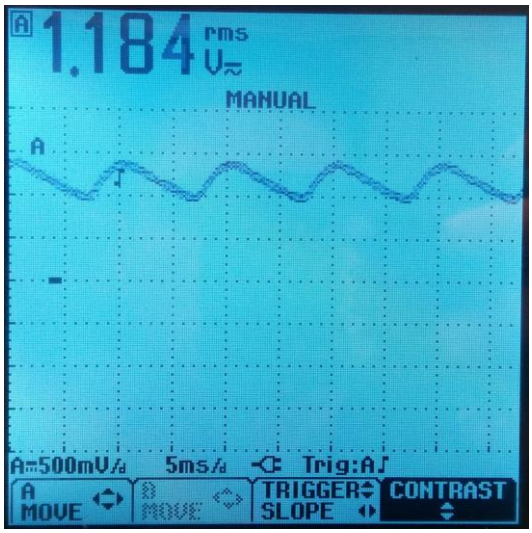

Figura 22 - Andamento della tensione sul circuito di misura della corrente

Si ottiene un segnale con ancora un po' di disturbo residuo, ma comunque accettabile per questi scopi.

I componenti scelti sono stati prima montati su breadboard (*fig.23*) per poter essere testati facilmente ed eventualmente sostituiti in caso di malfunzionamento o scelte differenti.

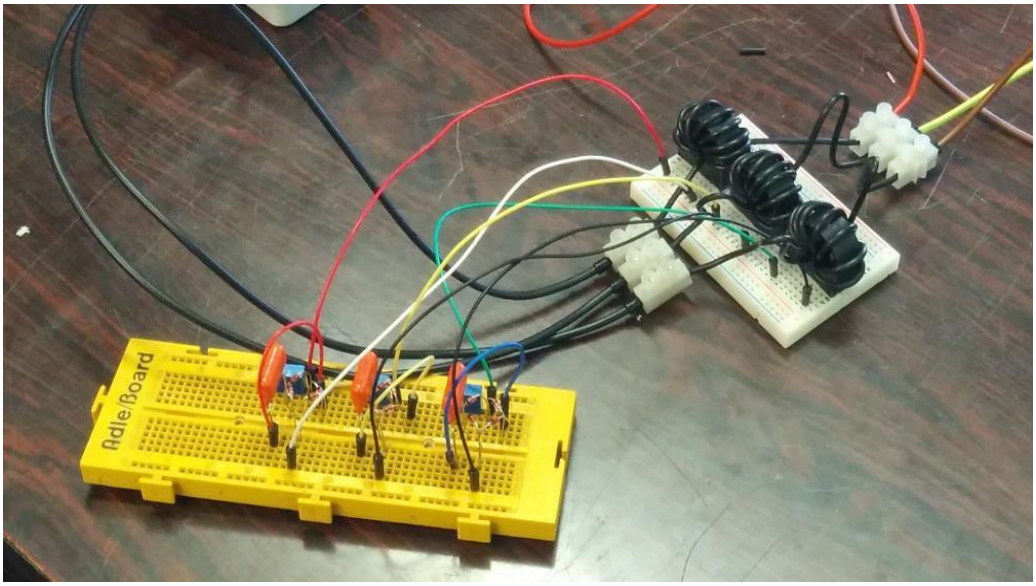

Figura 23 - Circuito di misura della corrente su breadboard

Infine il tutto è stato saldato su una scheda millefori:

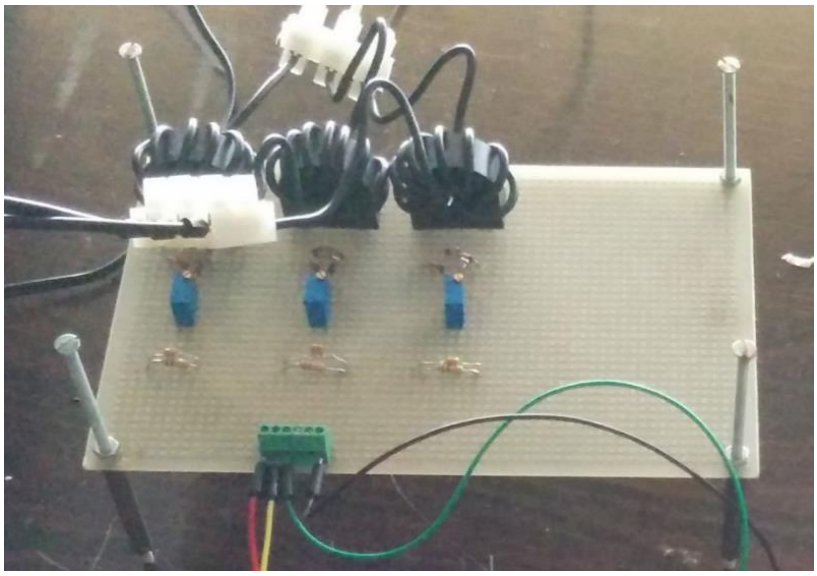

Figura 24 - Circuito di misura della corrente completo

Lo schema è stato replicato per ognuna delle tre lampade presenti nella lanterna in modo da poter individuare quale delle tre è accesa.

La parte di elaborazione viene realizzata utilizzando **Arduino**: una piattaforma hardware composta da una serie di schede elettroniche dotate di microcontrollore. Questa scheda è stata ideata e sviluppata ad Ivrea come strumento per la prototipazione rapida e per scopi hobbistici, didattici e professionali. La facilità di programmazione, l'alto numero di forum, di tutorial presenti online e la quasi infinita quantità di gadget ed applicazioni presenti sul mercato rendono Arduino la scelta migliore per la costruzione di questo tipo di prototipo. Sono state valutate molte schede elettroniche simili a questa (es. Raspberry pi), che permettono prestazioni superiori e personalizzazioni ad hoc da parte del produttore, ma la programmazione richiede una conoscenza del linguaggio tecnico superiore alla nostra ed i prezzi non sono minimamente paragonabili con quelli della scheda scelta.

Tra le molte schede prodotte da questa azienda, si è scelta **Arduino Due**, in quanto ha un alto numero di ingressi analogici (*12*), una potenza di calcolo superiore, nonché una maggiore memoria ram e flash rispetto alla sorella minore "Uno".

Di seguito sono riportate le specifiche delle due schede per un confronto visivo.

ARDUINO UNO	
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

ARDUINO DUE	
Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

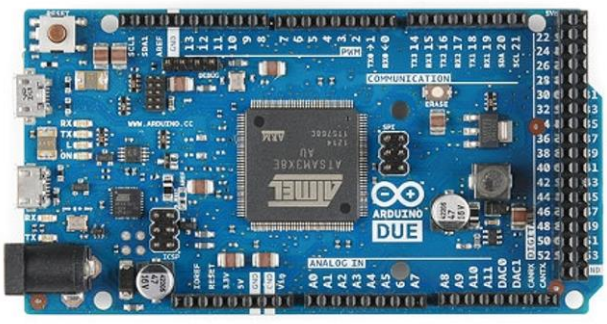

Figura 25 - Confronto specifiche Arduino Uno e Due

Come da schema in *fig.21*, a capo di ognuna delle tre resistenze da *100kΩ* viene collegato un ingresso analogico della scheda, ad esempio per la lampada rossa è stato scelto l'ingresso *A0*, per la gialla l'ingresso *A1* e per il verde l'ingresso *A2*. Gli altri capi delle resistenze sono stati ponticellati tra loro e collegati al pin GRD (ground) per dare il potenziale di riferimento al dispositivo.

La terza parte del circuito, ovvero quella che gestisce la comunicazione dei dati, comprende sempre Arduino Due, che oltre a misurare le grandezze e ad elaborarle, ha anche il compito di inviare i risultati finali in una rete locale a cui gli utenti si collegheranno e ad un server che si occuperà dell'immagazzinamento. Queste operazioni sono rese possibili collegando la scheda Arduino tramite una **shield Ethernet** (*fig.25*) compatibile ad un **router Wi-Fi** con accesso alla rete internet.

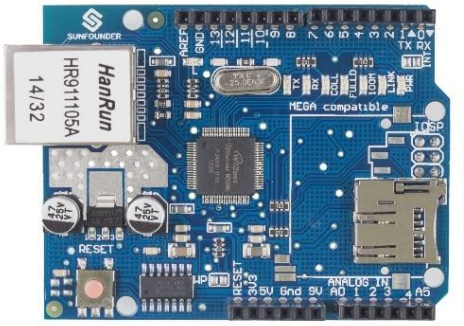

Figura 26 - Ethernet Shield

Come router è stato scelto il *TP-Link 8790* (*fig.27*) le cui caratteristiche più importanti sono:

- 4 porte LAN/Ethernet
- 1 porta per il collegamento del cavo telefonico, utile nel caso in cui il quadro semaforico venga dotato di una connessione Internet tradizionale
- 1 porta USB
- 2 antenne per creare una rete Wi-Fi che possono essere facilmente sostituite

La scelta è ricaduta su questo dispositivo per i seguenti motivi:

- basso costo di acquisto
- facilità di programmazione
- possibilità di estendere il raggio di azione della rete sostituendo semplicemente le antenne Wi-Fi con altre più potenti
- la presenza dello slot usb che permette di collegare una chiavetta gsm compatibile con la quale è possibile accedere ad internet senza l'installazione di cavi telefonici o fibra ottica nella centralina semaforica

Figura 27 - Router Tp-Link 8790

Una possibile alternava che è stata valutata è quella di utilizzare **Arduino Yun**, la quale è dotata di *12* ingressi analogici e, a differenza della Due, possiede un modulo *Wi-Fi* incorporato che permette di creare un collegamento diretto con un altro dispositivo abilitato allo scambio di dati in modalità Wi-Fi Direct. Per la comunicazione con il server remoto si utilizzerebbe una *shield Gsm* compatibile con la scheda.

Questa soluzione è stata esclusa per i seguenti motivi:

- la *Yun* ha in dotazione un processore inferiore rispetto alla Arduino Due, il che determinerebbe prestazioni inferiori
- la programmazione e la gestione delle operazioni sarebbe risultata molto più complicata
- la comunicazione mediante modalità *Wi-Fi direct* permette lo scambio di informazioni con un solo dispositivo alla volta, mentre il collegamento ad un router permette la gestione di più comunicazioni contemporanee
- i costi stimati sono superiori rispetto alla configurazione precedente

Un'ulteriore alternativa a quella precedente consiste nel collegare *Arduino Due* ad una *shield Wi-Fi* senza utilizzare il router, ma anche questa soluzione è stata subito esclusa perché questa tipologia di scheda non permette il collegamento diretto tra dispositivi, ma solo quello a router o modem.

Infine il prototipo è stato dotato di una striscia led per la comunicazione del tempo disponibile rimasto ai pedoni per terminare l'attraversamento.

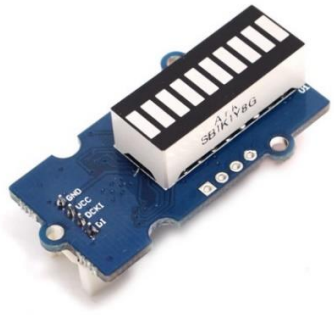

Figura 28 - Grove Led bar

Il prototipo assemblato in laboratorio è il seguente:

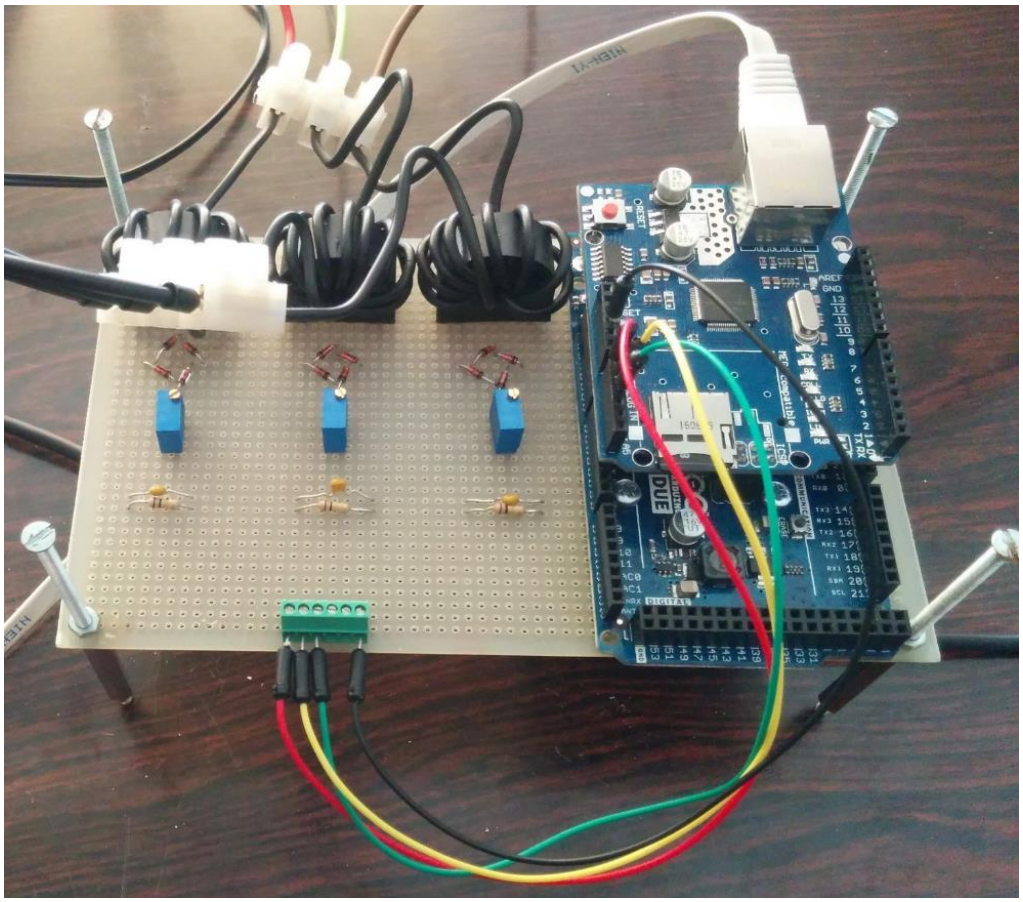

Figura 29 - Primo prototipo figura 1

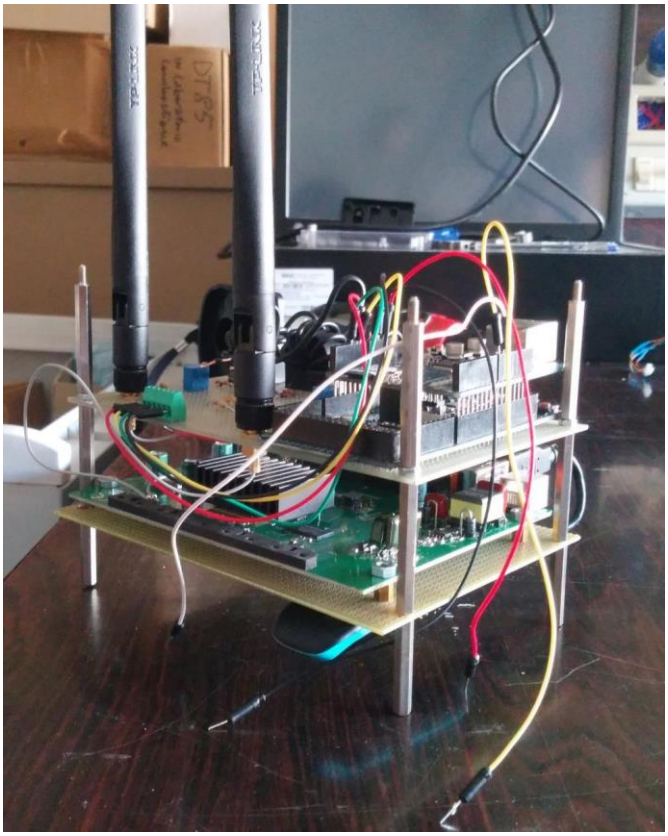

Figura 30 - Primo prototipo figura 2

Il quale è stato inscatolato per poter eseguire le prove in sicurezza:

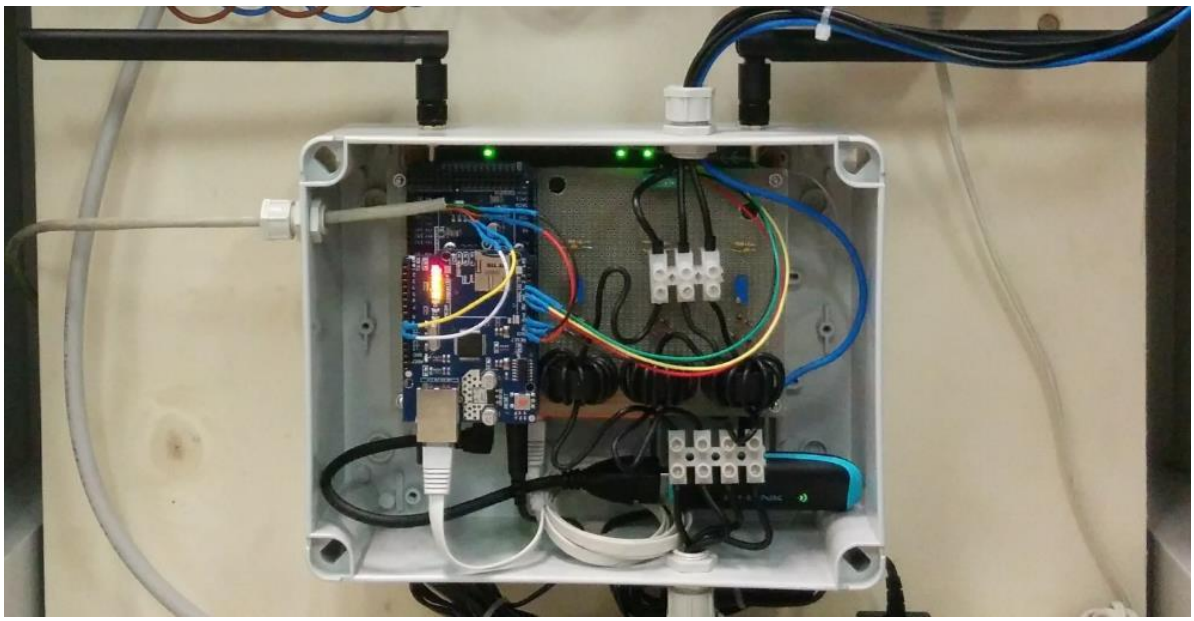

Figura 31 - Primo prototipo inscatolato

2.5.2 Scrittura del software

Per la programmazione della scheda è necessario utilizzare l'ambiente di sviluppo integrato (IDE) di Arduino, un'applicazione multipiattaforma in Java, derivata dall'IDE creato per il linguaggio di programmazione Processing e per il processo Wiring.

Questo ambiente è stato creato per iniziare alla programmazione gli utenti principianti, che siano a digiuno di come si sviluppa un software.

Per permettere la stesura del codice sorgente, l'ambiente di sviluppo include un editor di testo dotato di alcune funzioni come la colorazione della sintassi, il controllo delle parentesi e l'inserimento automatico degli spazi ad inizio riga. Il programma è inoltre in grado di compilare e caricare su Arduino il codice scritto dall'utente dopo averne verificato la validità.

L'ambiente di sviluppo è caratterizzato dalla presenza di molte librerie precompilate che rendono molto più semplice implementare via software operazioni di input e output. I programmi sono scritti in un linguaggio che deriva da C/C++, ma all'utente è chiesto di definire solamente due funzioni:

- **void setup ()**: funzione richiamata una sola volta all'inizio del programma che viene utilizzata per i settaggi iniziali
- **void loop ()**: funzione invocata ripetutamente, la cui esecuzione si interrompe resettando la scheda o disalimentandola

Esistono altri ambienti di sviluppo altrettanto validi per la programmazione della scheda Arduino, ma si è scelto di utilizzare quello ufficiale per evitare possibili problemi di compatibilità che avrebbero complicato e conseguentemente rallentato la stesura del codice.

2.5.2.1 Codice di programmazione primo prototipo

Per la stesura completa del programma di Arduino si rinvia agli allegati; di seguito invece sono riportati il flow chart principale e quelli delle varie funzioni in cui è stato suddiviso. Per ogni diagramma di flusso è fornita una breve descrizione delle operazioni svolte.

Per semplicità di ordine, nel programma sono state create delle funzioni che vengono richiamate durante l'esecuzione del codice. In testa sono descritte brevemente le operazioni che svolge, seguite dalle librerie ed i pin che Arduino utilizza per effettuare le letture degli ingressi analogici. Successivamente sono elencate tutte le variabili con vicino un breve commento che ne specifica l'uso. Una volta definito l'indirizzo ip dal quale si ricevono le richieste di informazioni, si entra nel **setup**. Questa parte del programma viene eseguita una sola volta, così come le varie funzioni vengono a meno della presenza di cicli for o while. Nel setup, per prima cosa, è definito il numero di monitor seriale che Arduino utilizza nel caso in cui sia prevista la scrittura a terminale, in questo caso è stato scelto il 9600. La seconda operazione effettuata è la connessione all'indirizzo ip *192.168.1.177* specificato in precedenza. Segue un ciclo while in cui vengono richiamate le funzioni *Misura()*, *Tempisticastock()* e *Cambiostato()*. Ogni funzione verrà spiegata al termine di questa breve introduzione. Il ciclo while viene eseguito finché la variabile *conteggiostock* non assume un valore pari a *5*.

Conclusa questa sequenza di operazioni a terminale apparirà la scritta *“Fine Stock”*. Completata la fase di setup inizia la fase di **loop**, dove tutte le operazioni presenti all’interno di questo ciclo vengono ripetute all’infinito a meno che venga a mancare l’alimentazione o il dispositivo venga resettato o riprogrammato. A differenza del ciclo di setup, nel loop vengono richiamate le funzioni: *Tempistica()*, *Confronto()*, *Reset()* e *Led()*. Il diagramma di flusso in *fig. 32* riassume brevemente tutte queste operazioni.

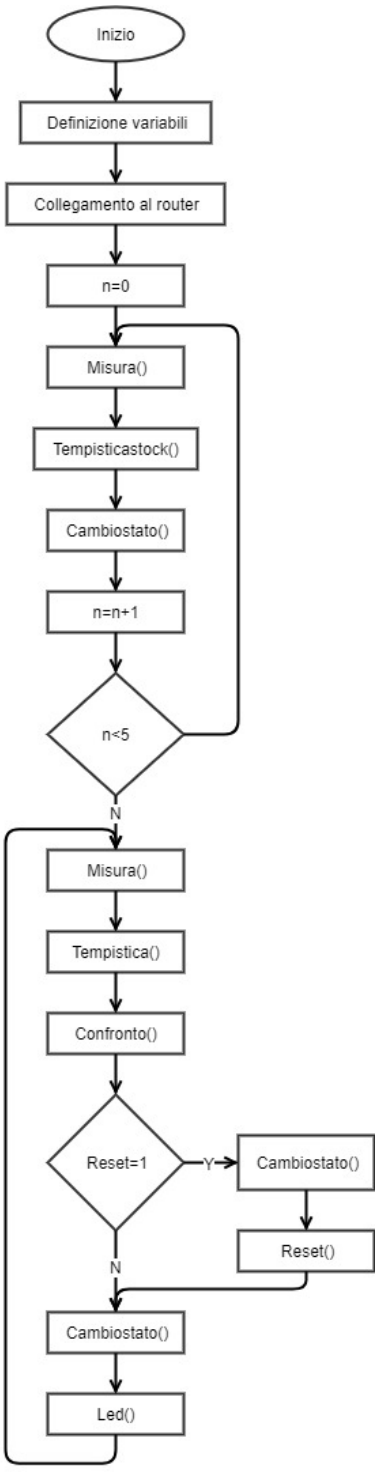

Figura 32 - Diagramma di flusso primo prototipo

Vediamo ora le funzioni richiamate durante l'esecuzione del codice.

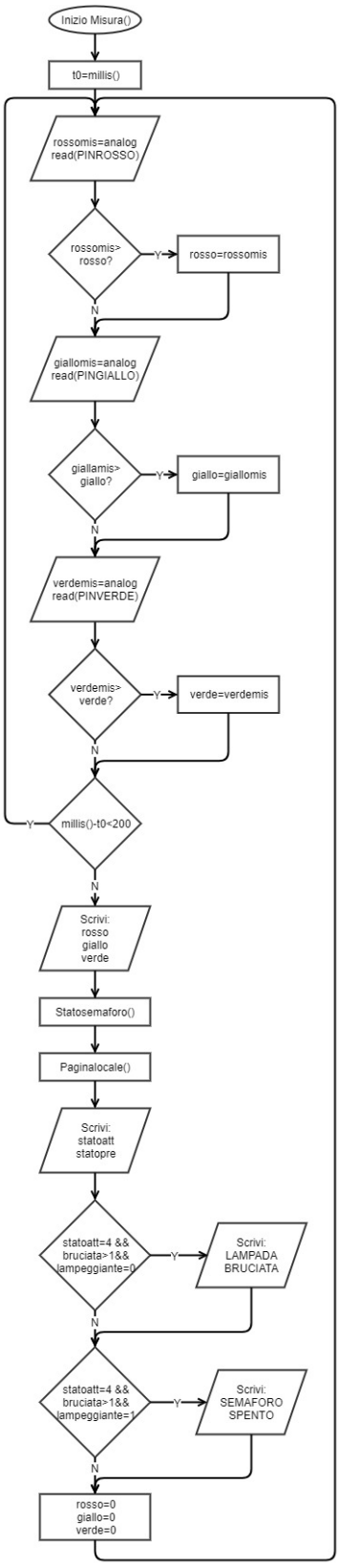

Figura 33 - Diagramma di flusso Misura()

Per ottenere un riferimento temporale, viene utilizzato il comando *millis(),* grazie al quale viene presa nota dell'istante in cui la funzione ***Misura()*** (*fig.33*) è richiamata. A questa prima operazione segue un ciclo while in cui per *200ms* Arduino legge i valori delle tre tensioni provenienti dal circuito di misura collegato agli ingressi analogici. Per ogni lettura, viene eseguito un confronto con il valore più grande ricavato in precedenza, nel caso la grandezza letta sia maggiore, Arduino va a sovrascrivere il nuovo valore altrimenti tiene quello già in memoria. Questo stratagemma viene usato per avere la certezza di ricavare sempre il valore maggiore su quell'intervallo di tempo ed evitare errori dovuti ai disturbi. Nel caso in cui le grandezze fossero perfettamente continue allora basterebbe una lettura puntuale. Terminati i *200ms,* in uscita si ottengono tre grandezze comprese in una scala tra *0* e *1023*, dalle quali possiamo ricavare quale lampada è accesa in quel lasso di tempo. In seguito vengono richiamate due altre funzioni: *Statosemaforo()* e *PaginaLocale()*. Infine viene stampato a monitor lo stato del semaforo.

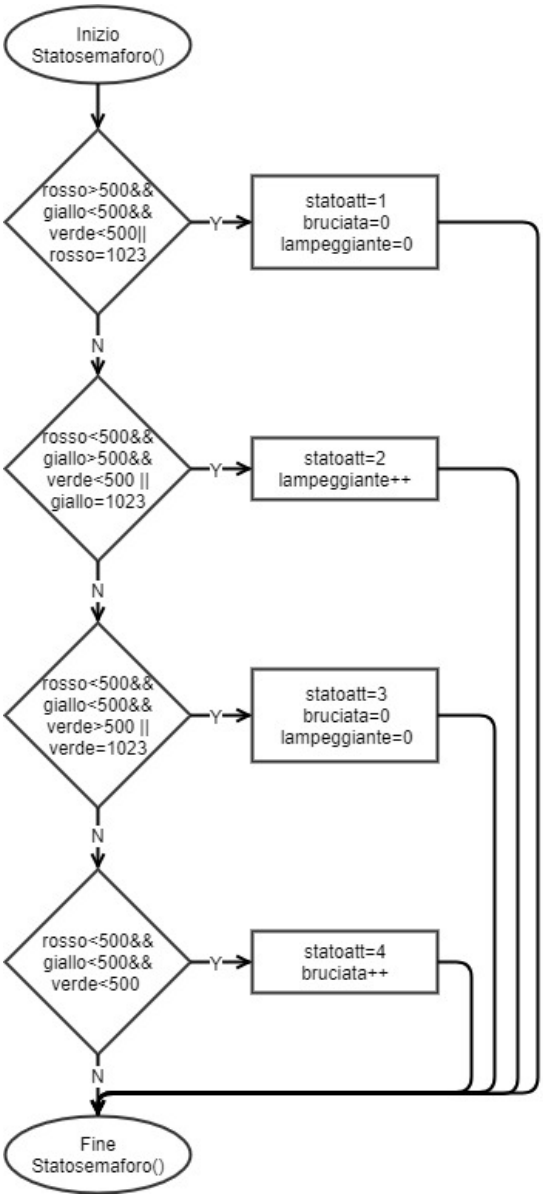

Figura 34 - Diagramma di flusso Statosemaforo()

Queste poche righe di codice servono a determinare lo stato del semaforo, cioè ad associare alla lampada accesa un numero che viene utilizzato dall'applicazione installata sul dispositivo mobile. La variabile stato può assumere quattro valori:

- Luce accesa **rossa**: stato = **1**
- Luce accesa **gialla**: stato = **2**
- Luce accesa **verde**: stato = **3**
- Semaforo **giallo lampeggiante** o **tutte e tre le spente** per un guasto: stato = **4**

Su quest'ultimo stato (*4*) si deve precisare che si tratta solamente di una bozza iniziale del codice al quale seguiranno sviluppi nei prototipi successivi.

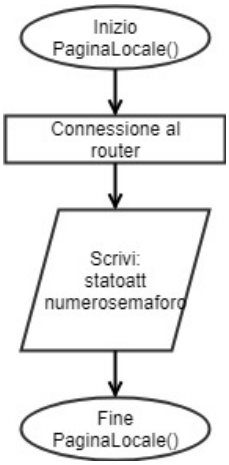

Figura 35 - Diagramma di flusso PaginaLocale()

La funzione ***PaginaLocale()*** permette la comunicazione del prototipo con altri dispositivi elettronici. Ogni volta che un computer, un dispositivo mobile tramite browser o applicazione, connesso allo stesso router del prototipo si collega alla pagina *http://192.168.1.177*, Arduino invia due numeri: **lo stato del semaforo** e il **numero del semaforo**. Dal primo si ricava la lampada accesa in quell'istante, dal secondo il nome dell'incrocio in ci si trova. Ovviamente ad ogni impianto semaforico verrà associato un numero univoco.

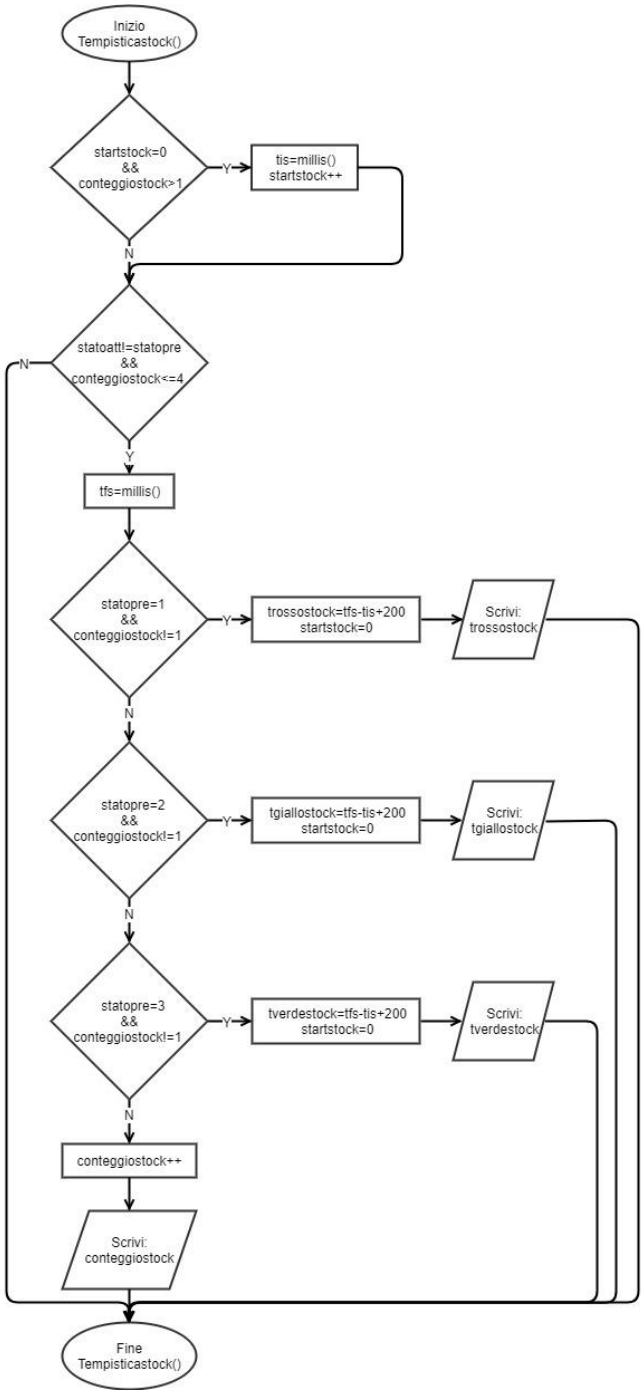

Figura 36 - Diagramma di flusso Tempisticastock()

Tempisticastock() è utilizzata per calcolare i tempi di durata di ogni stato al primo avvio del dispositivo, in modo da salvarli ed utilizzarli per confronti futuri. Dal momento che la lettura degli stati del semaforo avviene in modo passivo, questo perché non si ha accesso alla centralina semaforica, ma solo ai cavi di alimentazione, non si può sapere in quale istante il dispositivo viene messo in funzione. Il programma legge lo stato del semaforo (definito nella funzione *Statosemaforo()*) ed aspetta che cambi per iniziare a calcolare i tempi. In questo modo, con quattro cambi di stato si riescono ad ottenere delle tempistiche esatte. Ai tempi rilevati vengono aggiunti *200ms* dovuti alle incertezze della misurazione.

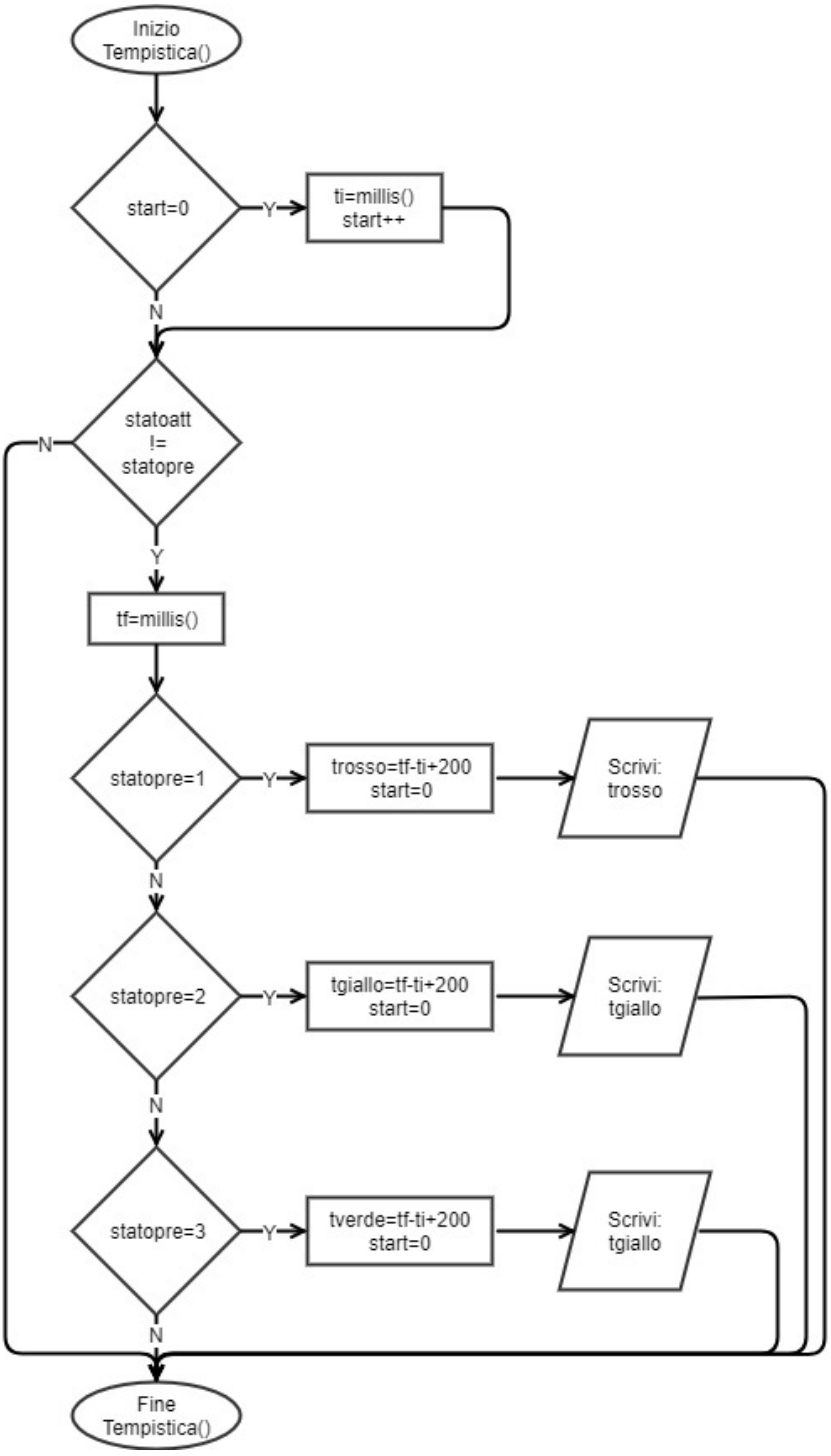

Figura 37 - Diagramma di flusso Tempistica()

La funzione ***Tempistica()*** esegue le stesse operazioni descritte in *Tempisticastock()* con la differenza che non deve aspettare il cambio di stato per iniziare a calcolare i tempi. Gli intervalli vengono salvati in variabili per i confronti con quelli ottenuti con *Tempisticastock()*, in questo modo si può sapere se il ciclo semaforico è cambiato rispetto al precedente.

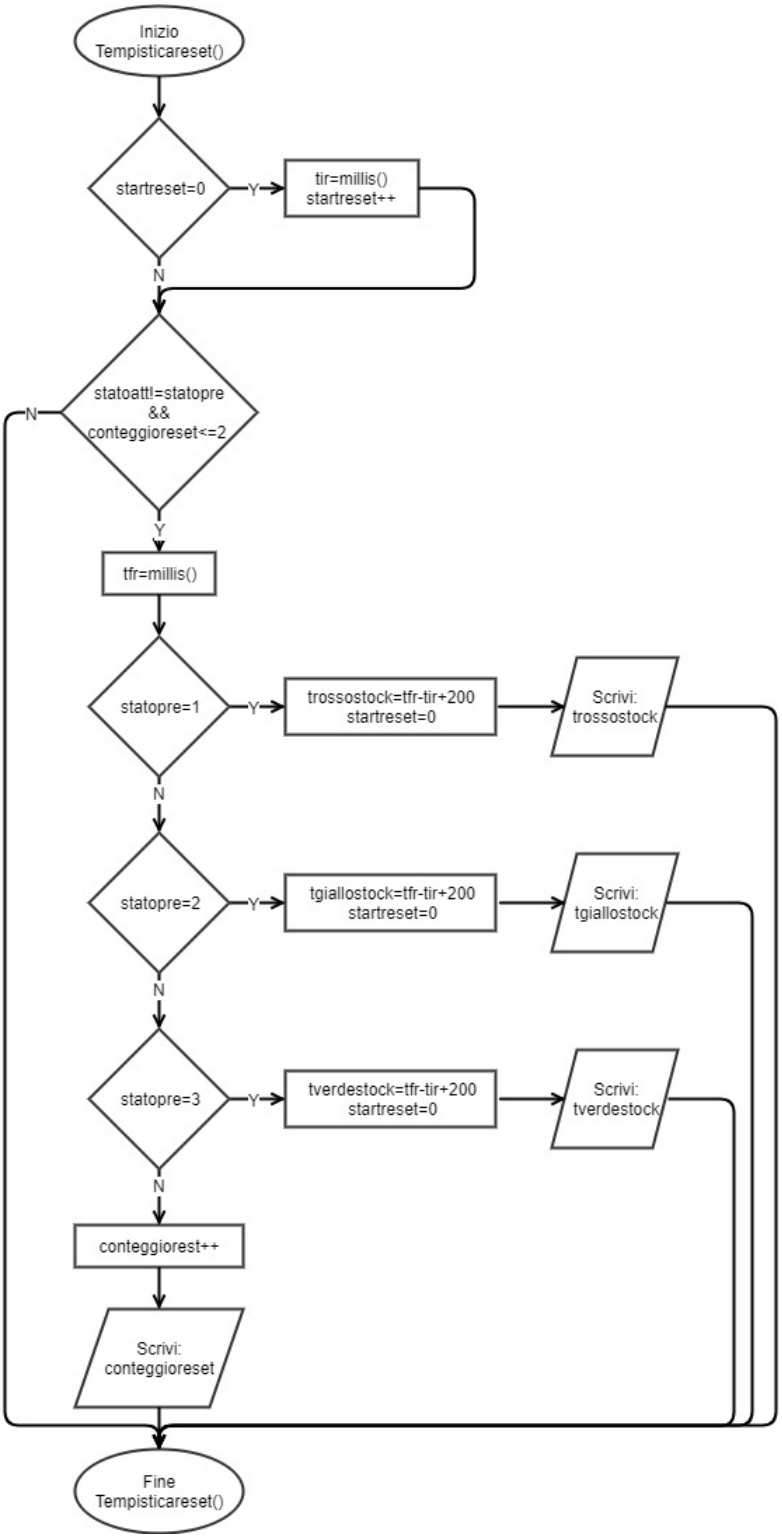

Figura 38 - Diagramma di flusso Tempisticareset()

Tempisticareset() usa gli stessi principi di *Tempistica()* e *Tempisticastock()* in quanto determina ad ogni cambio di stato il tempo di durata della lampada precedente e lo salva nella variabile opportuna che viene utilizzata per i confronti. La differenza con le precedenti sta nel fatto che invece di dover determinare tutti e tre i tempi, ne calcola solamente due.

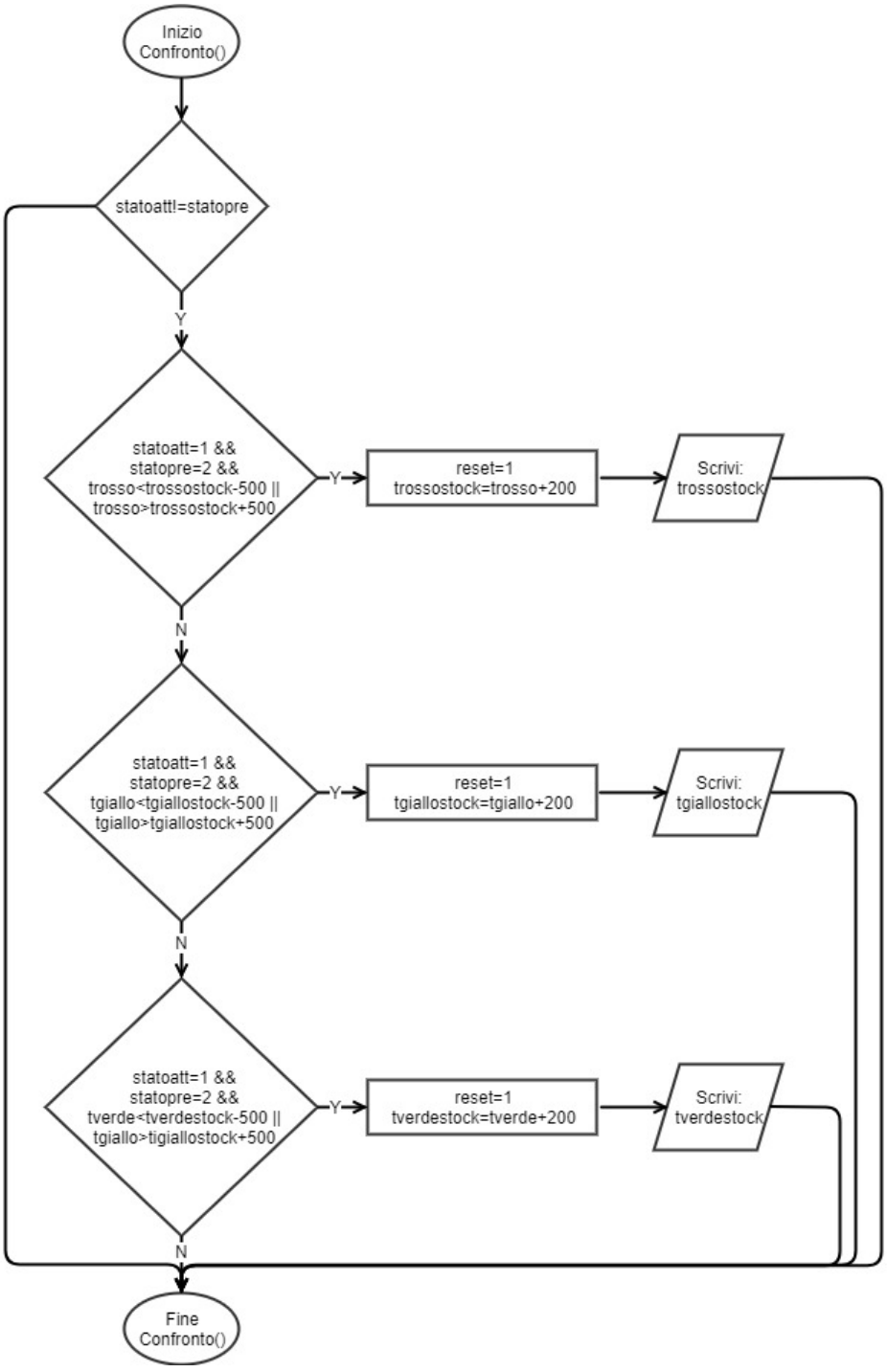

Figura 39 - Diagramma di flusso Confronto()

Confronto() esegue il raffronto tra i tempi ottenuti dalla funzione *Tempisticastock()* e quelli ottenuti da *Tempistica()*. Se questi ultimi non rientrano in un range di mezzo secondo rispetto ai primi, significa che il ciclo semaforico è stato alterato. La variabile *reset* viene uguagliata a *1* ed il nuovo tempo viene immagazzinato nella relativa variabile per i confronti. Nel caso in cui il tempo rientri in questo intervallo allora la variabile reset rimane nulla.

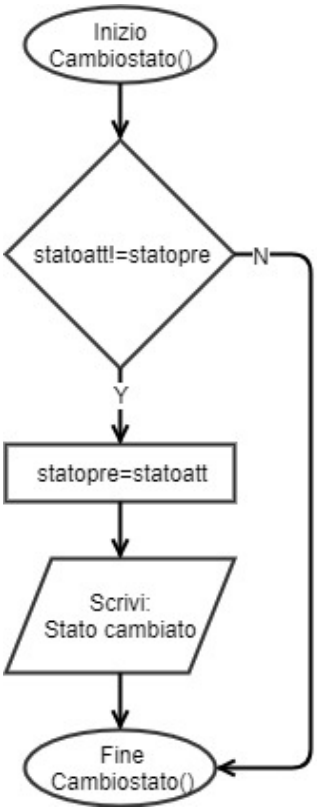

Figura 40 - Diagramma di flusso Cambiostato()

La funzione ***Cambiostato()*** serve ad aggiornare la variabile che tiene in memoria la condizione precedente (*statopre*) ad ogni passaggio di stato, inoltre a terminale viene notificato l'evento.

La funzione ***Reset()*** (*fig.41*) viene richiamata solamente quando la variabile *reset* è di valore unitario. Lo scopo è quello di aggiornare i tempi di ogni lampada quando il ciclo semaforico viene modificato. Al suo interno è presente un ciclo *while* da cui è possibile uscire solo quando la variabile *conteggioreset* è uguale a due, ovvero il ciclo termina quando si sono determinati i tempi delle due lampade rimanenti. Questo è possibile perché in *Confronto()* si è già determinato il nuovo tempo della prima lampada. Questa soluzione permette di semplificare il programma e renderlo più celere. I nuovi tempi vengono salvati nelle variabili che poi saranno utilizzate per i confronti successivi. Vengono richiamate le funzioni: *Misura()*, *Tempisticareset()*, *Cambiostato()* e *Led()*. Una volta terminato viene stampato su monitor la scritta: *"Fine Reset"*.

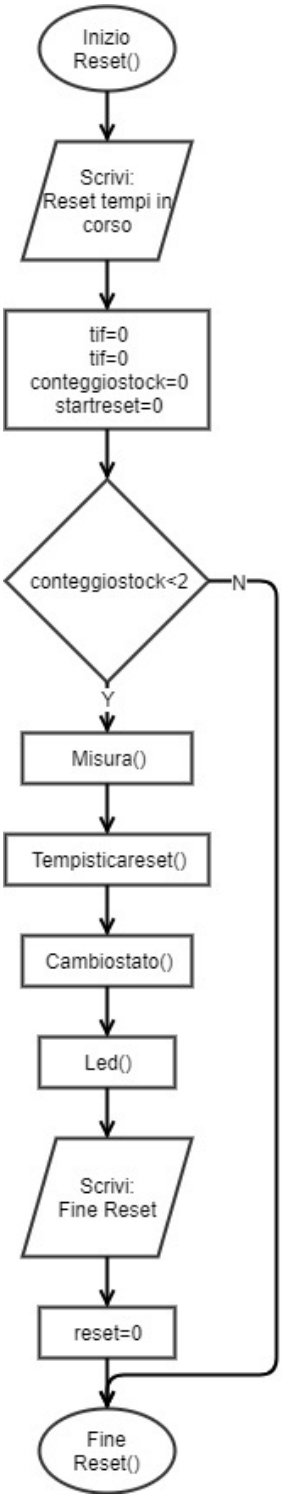

Figura 41 - Diagramma di flusso Reset()

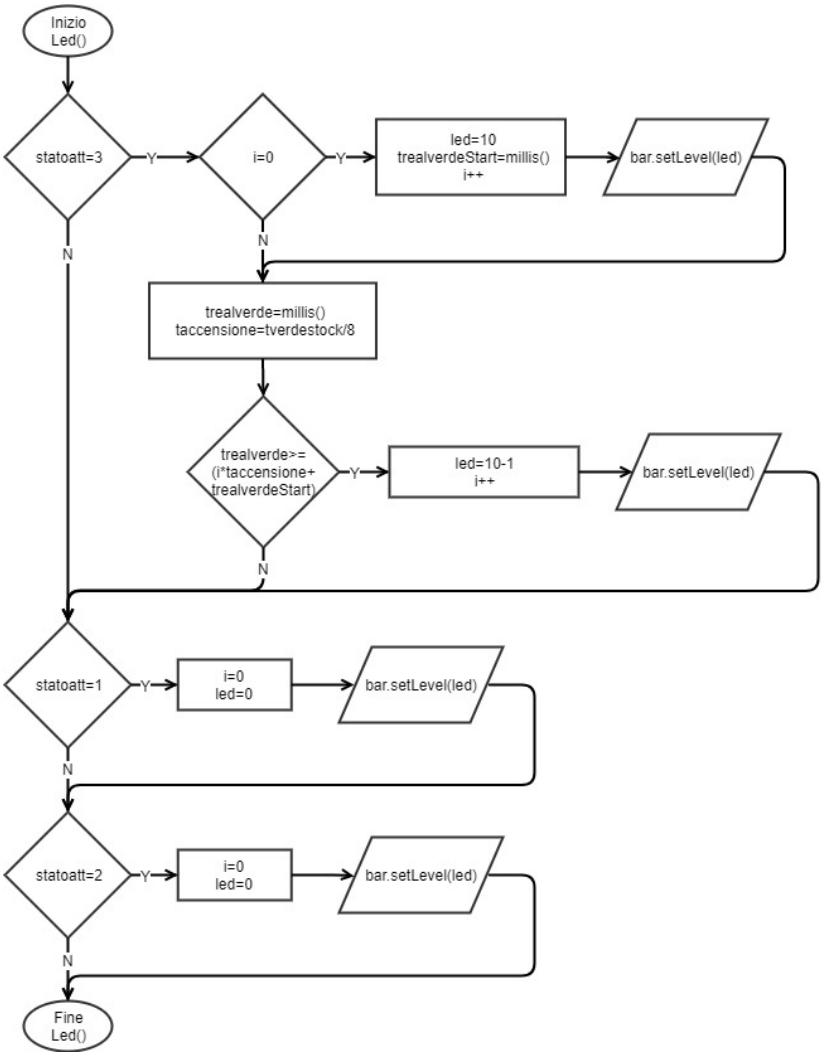

Figura 42 - Diagrammadi flusso Led()

Quest'ultima funzione viene utilizzata per realizzare il conto alla rovescia tramite striscia led. Una volta che la lampada verde è accesa (*stato=3*) vengono accesi tutti i led verdi. Il programma legge il tempo di durata della luce verde (precedentemente immagazzinato durante il setup) e lo si divide per il numero di led a disposizione: in questo caso *8*. Il risultato ottenuto indica l'intervallo di tempo dopo il quale un led va spento; in questo modo si può avere un'idea di quanto tempo si ha ancora a disposizione per attraversare. Quando è accesa la lampada gialla o quella rossa allora la barra viene spenta automaticamente.

In caso di variazione dei tempi di durata del ciclo, sorgono un paio di problemi:

- nel caso in cui il tempo di durata del verde venga accorciato, a fine ciclo (verde) verranno spenti più led contemporaneamente

- nel caso in cui il tempo di durata del verde venga allungato, per qualche istanti tutti i led rimangono spenti anche se la lampada (verde) è ancora accesa

Questi problemi sono dati dal fatto che si utilizzano tempistiche ottenute con metodi passivi e non prendendo i tempi direttamente dalla centralina del semaforo. Quando il ciclo viene cambiato non si può sapere per quanto tempo la lampada verde rimarrà accesa, infatti il programma utilizza come riferimento il tempo ottenuto dal ciclo precedente. Ovviamente il codice è in grado di capire quando avvengono modifiche temporali, quindi dal ciclo successivo la striscia led tornerà a funzionare correttamente.

Infine vorrei ricordare che questa striscia serve solo come indicazione del tempo utile per poter attraversare, e che anche se ci fossero dei piccoli errori nell'indicazione, dell'ordine di qualche secondo) rimane ancora il tempo della lampada gialla per completare l'attraversamento.

2.6 Test del primo prototipo

Il prototipo descritto nel *paragrafo 2.5* e l'applicazione per smartphone Android riportata nel C*apitolo 3* sono stati testati su una riproduzione di un attraversamento pedonale costruita in laboratorio. Gli scopi per cui è stata eseguita questa emulazione sono i seguenti:

- verificare il funzionamento del prototipo sia hardware che software
- verificare il funzionamento dell'applicazione
- riscontrare e correggere errori di qualsiasi natura al fine di migliorare l'intero prodotto

In laboratorio è stato allestito un "banco di prova" dove poter effettuare queste simulazioni:

Figura 43 - Test primo prototipo in laboratorio

Figura 44 - Quadro semaforico laboratorio

Come si vede dall'immagine precedente, per poter effettuare queste prove è stato necessario costruire una centralina che simuli quella presente negli impianti semaforici esistenti.

2.6.1 Realizzazione della centralina di emulazione

Per realizzare una prova il più fedele possibile alla realtà, si è deciso di assemblare una centralina che emuli quella semaforica che regola giornalmente un passaggio pedonale. Si è preso come riferimento l'attraversamento di Corso Duca degli Abruzzi posto davanti al Politecnico di Torino il quale è composto semplicemente da due lanterne semaforiche e appunto da una centralina di comando.

Il principio di funzionamento è il seguente: un circuito elettronico programmato accenderà in sequenza e con una certa durata ciascuna delle tre lampade presenti nella lanterna. La sequenza di accensione è la seguente: prima la lampada rossa, seguita da quella verde ed infine quella gialla. Ognuna di esse verrà accesa singolarmente, infatti non è presente la

contemporanea accensione di due o più lampade in quanto il nuovo codice della strada non prevede più questa casistica (in passato era possibile che il giallo ed il verde fossero accese contemporaneamente).

La nostra centralina semaforica è stata programmata per poter scegliere tre modalità di funzionamento:

1. Semaforo spento con giallo lampeggiante
2. Semaforo acceso con ciclo “normale”
3. Semaforo acceso con ciclo “ridotto”, cioè un ciclo “normale”, ma le tre luci verranno accese per una durata inferiore.

Il ciclo “normale” prevede che la luce rossa rimanga accesa per 30 secondi, seguita dalla verde per 20 secondi ed infine dalla gialla per 10 secondi. Dopodiché il ciclo riprenderà dall’inizio.

Il ciclo “ridotto” serve per simulare quei semafori che variano la durata dei tempi nel caso in cui ci sia un autobus nei pressi dell’incrocio. La durata di ogni lampada è la seguente: 27 secondi per la rossa, 18 secondi per la verde ed infine 9 secondi per la gialla. Anche in questo caso dopo il giallo il ciclo ricomincerà nuovamente.

Quando il semaforo è spento la luce gialla lampeggerà ininterrottamente.

La scelta tra i cicli deve poter essere effettuata in modo semplice ed immediato, con la possibilità di cambiarla in ogni momento senza dover modificare i collegamenti, disalimentare il semaforo o resettare la centralina. Per i motivi sopra elencati si è deciso di adottare la seguente soluzione: collegare un potenziometro alla centralina e programmarla in modo tale che, a seconda del valore impostato dal dispositivo elettrico permetta di scegliere uno dei tre cicli.

2.6.1.1 Componenti e assemblaggio dell’hardware

Per la realizzazione pratica della centralina di comando sono stati scelti i seguenti componenti:

- Arduino Uno
- Grove Base Shield compatibile con Arduino Uno
- Relay Shield compatibile con per Arduino Uno
- Potenziometro da 0 a 5MΩ

Di seguito verrà analizzato ogni componente della centralina:

Arduino Uno: la scelta è ricaduta su questo tipo di scheda perché è semplice da programmare, sul mercato sono presenti molti "gadget" che permettono di estendere la sua funzionalità a moltissimi scopi ed ha un basso costo d'acquisto, il che rende Arduino la miglior scelta per questo tipo operazione. Nello specifico si è scelta la Uno perché le funzioni che andrà ad eseguire non richiedono elevata potenza di calcolo ed inoltre si è cercato di minimizzare quanto più possibile il costo della realizzazione di questo emulatore (la centralina semaforica).

Figura 45 - Arduino Uno

Relay Shield: una scheda applicabile ad Arduino Uno tramite i pin di collegamento. Presenta 4 relè comandabili in modo indipendente, ognuno di essi è collegato a tre morsetti di connessione: il comune (COM), il normalmente aperto (NO) e il normalmente chiuso (NC). Con questo tipo di scheda è possibile comandare fino a otto lampade. In questo caso si utilizzeranno solamente tre relè, per ognuno di essi si usufruirà del morsetto comune più quello normalmente aperto, in questo modo nel caso venisse a mancare l'alimentazione ad Arduino, la lanterna verrà disalimentata evitando di fornire segnali errati durante le simulazioni.

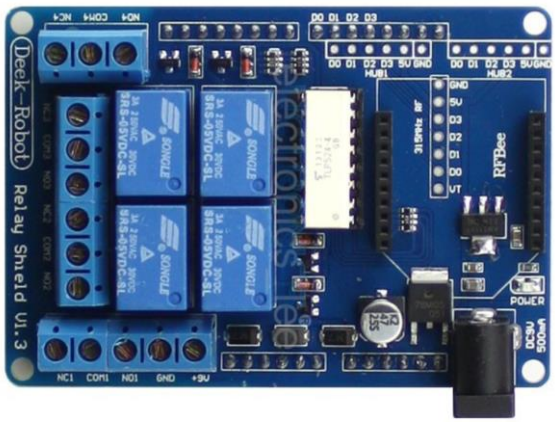

Figura 46 - Relay Shield

Grove Base shield: una scheda applicabile ad Arduino Uno tramite il collegamento attraverso i pin che viene utilizzata per connettere il potenziometro ad Arduino.

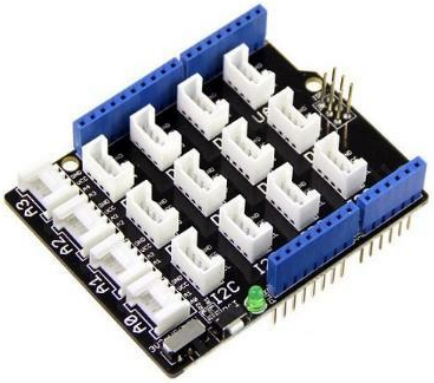

Figura 47 - Groove Base shield

Potenziometro: dispositivo elettrico equivalente ad un partitore di tensione resistivo variabile, dotato di tre pin di collegamento, dai quali è possibile ottenere una resistenza variabile da *0* a *5000000*Ω (*5M*Ω). Nella centralina viene utilizzato per selezionare i tre metodi di funzionamento del semaforo.

Figura 48 - Potenziometro 0-5MΩ

Di seguito (*fig.49*) è riportata la foto della centralina terminata, la componentistica elettronica è stata installata in una scatola di derivazione per proteggerla da eventi esterni e per poter essere utilizzata in modo sicuro.

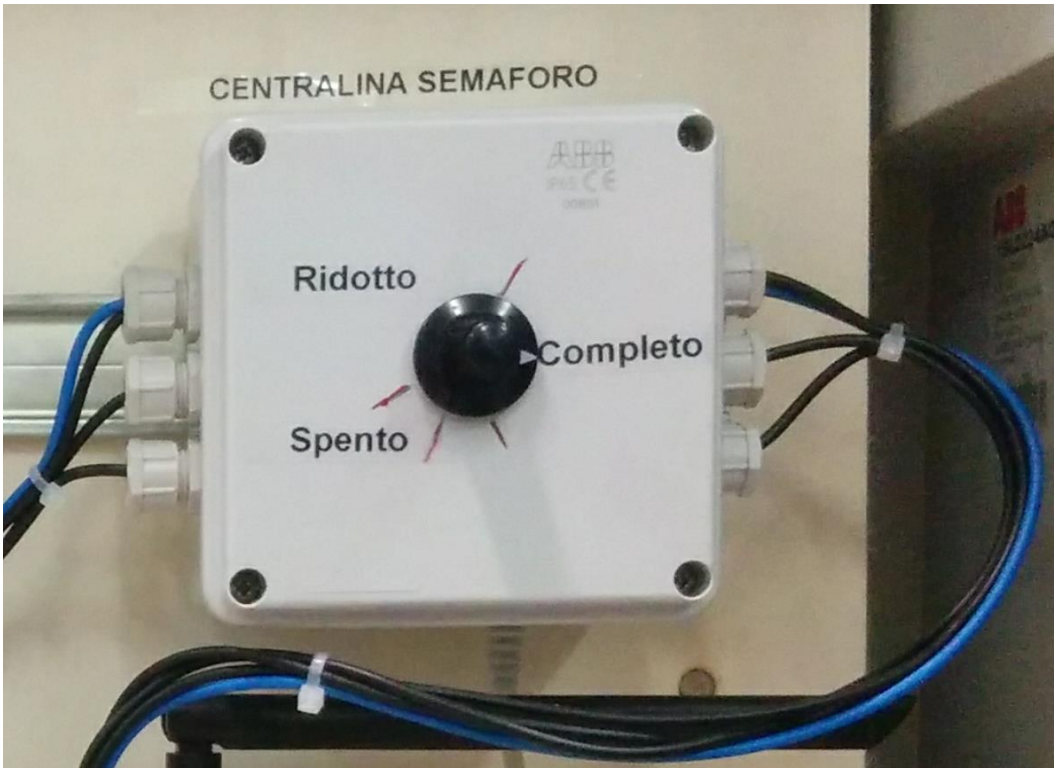

Figura 49 - Centralina semaforica completa

Dall'immagine si notano i quattro cavi di alimentazione (tre neri e uno blu) delle tre lampade e il cavo bianco per l'alimentazione di Arduino e dei Relè meccanici. Per facilitare l'accesso al potenziometro da parte degli utenti, questo è stato piazzato sul coperchio della scatola, il quale è stato suddiviso in tre zone a seconda del tipo di funzionamento desiderato.

2.6.1.2 Scrittura del software

Per il codice completo di Arduino si rinvia agli allegati; di seguito è riportato il flow chart delle operazioni eseguite dal compilatore per l'emulazione di un ciclo semaforico:

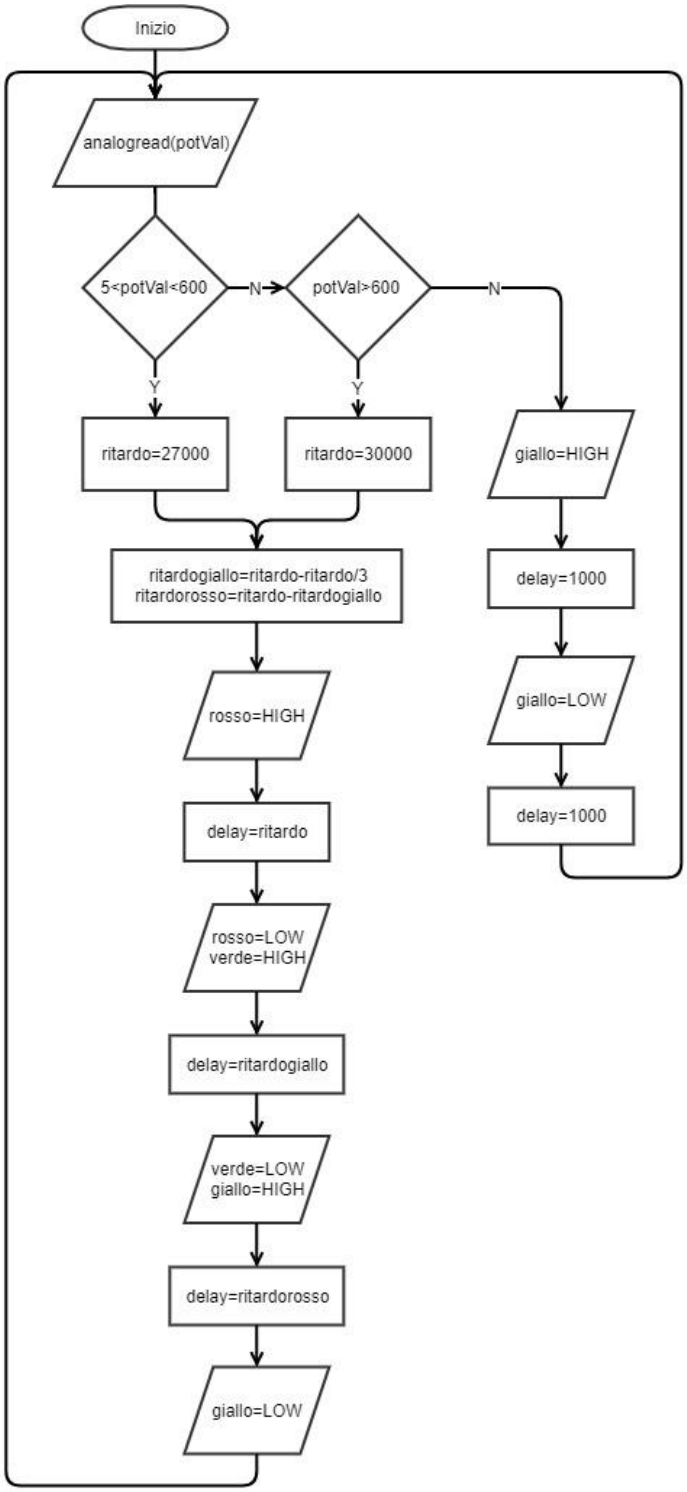

Figura 50 - Diagramma di flusso centralina semaforica

Nella prima parte del programma vengono definite le variabili da utilizzare durante l'esecuzione del codice ed il vettore che racchiude le uscite di Arduino a cui vengono collegati i tre relè. I pin utilizzati sono il 4, il 5 ed il 6 ed occupano la prima, la seconda e la terza posizione del vettore; al numero 4 viene collegato il relè della lampada rossa, al 5 quello della lampada gialla e al 6 quello della verde.

Nel *setup* si vanno a definire quali sono gli output (i tre pin del vettore) utilizzando un ciclo for e l'unico input, ovvero il valore impostato dal potenziometro collegato all'ingresso analogico *A0*.

In entrata del *loop*, Arduino legge il valore imposto dal potenziometro in un range che va da 0 a 1023, in cui zero significa cortocircuito (*0*Ω), mentre 1023 significa che la resistenza variabile è impostata su *5M*Ω. Se la lettura risulta superiore a 5, ma minore di 600, allora il tempo di durata della lampada rossa viene impostato a 27 secondi; negli altri casi la durata viene aumentata a 30 secondi. Con due semplici operazioni si ricavano anche i tempi delle altre due lampade, che, per nostra scelta, dipendono dalla durata della lampada rossa. Infine se il potenziometro è posizionato per fornire un valore compreso tra 0 e 4 allora si attiva la modalità giallo lampeggiante.

Queste operazioni si ripetono fino a quando la centralina non viene disalimentata.

2.6.2 Test in laboratorio ed incontro con associazione non vedenti

I test eseguiti in laboratorio hanno dato risultati soddisfacenti, infatti questo primo prototipo è in grado di:

- comunicare con precisione lo stato della lanterna semaforica all'utente dotato di smartphone su cui è installata l'applicazione descritta nel *Capitolo 3*. Le richieste di aggiornamento avvengono in modo automatico ogni mezzo secondo
- rilevare il cambiamento di ciclo del sistema semaforico, calcolare ed immagazzinare i nuovi tempi
- eseguire il countdown tramite striscia led del tempo di accensione della lampada verde
- riconoscere se il semaforo è spento o se una lampada è bruciata, anche se, quest'ultimo punto, è ancora da migliorare

L'applicazione in fase di test non ha dato grossi problemi; è possibile che dopo qualche ciclo il dispositivo smetta di vibrare e comunichi le informazioni solo tramite messaggi vocali.

Dati i risultati più che buoni delle prove, si è deciso di far testare il prototipo ad un esponente de **Unione Italiana Ciechi e Ipovedenti** per ottenere pareri e consigli da chi dovrà utilizzarlo quotidianamente.

L'incontro, avvenuto nel laboratorio di elettronica del Politecnico di Torino, dipartimento DENERG, è stato molto fruttuoso sotto molti punti di vista: l'idea su cui si basa questo progetto è apparsa fin da subito innovativa e soprattutto di grande utilità a chi di questo strumento si dovrà servire. Durante le prove sono sorte delle problematiche e nuovi spunti per la

realizzazione di un secondo prototipo che punta a migliorare quello precedentemente realizzato.

Per prima cosa si sono riscontrati dei problemi con l'utilizzo dell'applicazione: il funzionamento è risultato incompatibile con i programmi di aiuto (es. *Google Talkback*) normalmente usati da questo tipo di utenti per l'utilizzo dello smartphone. L'interfaccia sullo schermo è troppo complessa e le troppe immagini presenti mandano in confusione l'utente.

Al contrario sono stati apprezzati:

- il funzionamento in background dell'applicazione
- il fatto che comunichi le informazioni grazie a frasi brevi e chiare
- il cambio di vibrazione dal passaggio da luce verde a luce rossa
- la comunicazione di dove ci si trova quando il dispositivo si collega al Wi-Fi

Dopo i test sono stati forniti i seguenti suggerimenti:

- dare all'applicazione un titolo di facile intuizione
- configurare l'applicazione in modo da essere compatibile con i programmi a disposizione dei non vedenti
- snellire l'interfaccia inserendo poche icone e scritte di grosse dimensione per gli utenti ipovedenti che non usano i programmi di assistenza all'utilizzo dello smartphone
- Comunicare all'utente il prossimo incrocio a seconda della direzione intrapresa
- Automatizzare il più possibile l'applicazione

Un ulteriore problema individuato durante i test in laboratorio è l'utilizzo del protocollo *Wi-Fi* come mezzo di comunicazione tra smartphone e prototipo. La rete wireless locale usata per i test era disconnessa da internet, quindi se lo smartphone è impostato sull'opzione di rete intelligente, anche se collegato al router, il dispositivo ignora le informazioni ricevute dal prototipo ed utilizza la rete dati per connettersi ad internet.

In caso di segnale debole il dispositivo può decidere di scollegarsi dalla rete locale e passare all'utilizzo della rete dati con risultato identico al caso precedente. Nell'ipotesi in cui questa opzione non sia abilitata, l'utente può usare correttamente l'applicazione e riceve le informazioni desiderate dal prototipo, ma in questo caso lo smartphone rimane senza accesso ad internet per tutto il tempo in cui rimane agganciato alla rete *Wi-Fi*. Questo tipo di situazione non è consigliata perché le applicazioni di messaggistica (*WhatsApp*, *Telegram* etc…) e di navigazione (*Maps*, *Tom-Tom* etc…), o più semplicemente tutti quei programmi che necessitano dell'accesso alla rete, non possono funzionare correttamente.

2.7 Realizzazione del secondo prototipo

Per i motivi esplicati precedentemente (*Capitolo 2.6.2*) sono state adottate le seguenti soluzioni:

- abbandonare il protocollo *Wi-Fi* per un altro tipo di comunicazione che permetta di risolvere i problemi legati all'esclusione dalla rete internet quando si ricevono le informazioni dal prototipo
- aumentare l'affidabilità del sistema e conseguentemente diminuire la possibilità di comunicare all'utente informazioni errate dovuti ad errori di campionamento utilizzando non solo il circuito di misura della corrente, ma anche un circuito di misura della tensione
- ridurre le dimensioni del circuito e dei componenti utilizzati
- snellire e semplificare il codice di programmazione di Arduino ed evitare di effettuare due diverse letture analogiche dai due circuiti di misura

I punti elencati in precedenza sono diventati le linee guida per la realizzazione di un nuovo prototipo che punta a migliorare il precedente.

Riguardo al primo punto, il nuovo dispositivo utilizza ancora il protocollo *Wi-Fi* in quanto si è deciso di sviluppare e verificare la solidità del circuito di misura anziché quello di comunicazione.

Per rispettare l'obbiettivo prefissato nell'ultimo punto, si è deciso di non effettuare due diverse letture analogiche sui due circuiti di misura, quindi è stato realizzato un comparatore analogico che fornisce un output digitale.

Alimentato direttamente da Arduino (*5V*), il comparatore, permette di usare la lettura di segnali digitali invece di quella analogica. Si ottiene quindi un valore uguale a uno quando sono presenti sia tensione che corrente sulla lampada (lampada accesa) ed un valore nullo quando una delle due od entrambe vengono a mancare (lampada guasta o non alimentata).

2.7.1 Componenti e assemblaggio dell'hardware

La lanterna scelta è la stessa utilizzata per i test del prototipo precedente ovvero con tre lampade led alimentate con alimentazione con tensione di 230V.

Per effettuare la misura di tensione senza perturbare il circuito di comando del semaforo, si deve necessariamente utilizzare un optoisolatore. Come descritto nel *Capitolo 2.3.3*, l'optoisolatore permette di realizzare un accoppiamento tra due circuiti di segnale mantenendo l'isolamento galvanico. Il componente è realizzato abbinando otticamente un led con un elemento fotosensibile.

Commercialmente sono presenti sul mercato diversi modelli con caratteristiche differenti tra loro, per la realizzazione del secondo prototipo è stato scelto il modello *4n35* prodotto dalla *Vishay*.

Di seguito sono riportati lo schema circuitale e l'immagine del componente:

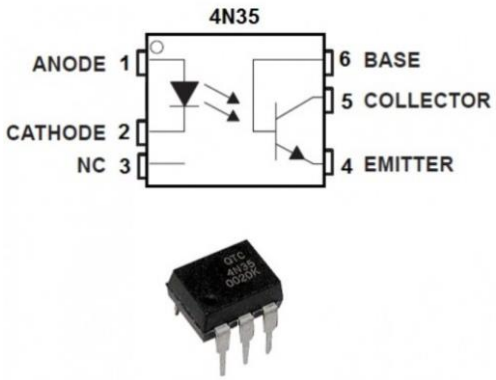

Figura 51 - Optoisolatore

Come si può vedere dallo schema, l'integrato ha sei piedini numerati: i numeri *1* (*Anodo*) e *2* (*catodo*) devono essere collegati al circuito di potenza mentre i piedini *4* (*Emettitore*) e *5* (Collettore) vanno collegati al circuito di misura che viene interfacciato con Arduino. Di seguito sono riportati i parametri più importanti dell'optoisolatore presi dal datasheet (*fig.52-53*):

ABSOLUTE MAXIMUM RATINGS (1)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
INPUT				
Reverse voltage		V_R	6	V
Forward current		I_F	50	mA
Surge current	t ≤ 10 µs	I_{FSM}	1	A
Power dissipation		P_{diss}	70	mW
OUTPUT				
Collector emitter breakdown voltage		V_{CEO}	70	V
Emitter base breakdown voltage		V_{EBO}	7	V
Collector current		I_C	50	mA
	t ≤ 1 ms	I_C	100	mA
Power dissipation		P_{diss}	70	mW
COUPLER				
Isolation test voltage		V_{ISO}	5000	V_{RMS}
Creepage			≥ 7	mm
Clearance			≥ 7	mm
Isolation thickness between emitter and detector			≥ 0.4	mm

Figura 52 - Datasheet optoisolatore (1)

ELECTRICAL CHARACTERISTICS (1)							
PARAMETER	TEST CONDITION	PART	SYMBOL	MIN.	TYP.	MAX.	UNIT
INPUT							
Junction capacitance	V_R = 0 V, f = 1 MHz		C_j		50		pF
Forward voltage (2)	I_F = 10 mA		V_F		1.3	1.5	V
	I_F = 10 mA, T_{amb} = - 55 °C		V_F	0.9	1.3	1.7	V
Reverse current (2)	V_R = 6 V		I_R		0.1	10	µA
Capacitance	V_R = 0 V, f = 1 MHz		C_O		25		pF
OUTPUT							
Collector emitter breakdown voltage(2)	I_C = 1 mA	4N35	BV_{CEO}	30			V
		4N36	BV_{CEO}	30			V
		4N37	BV_{CEO}	30			V
Emitter collector breakdown voltage(2)	I_E = 100 µA		BV_{ECO}	7			V
OUTPUT							
Collector base breakdown voltage (2)	I_C = 100 µA, I_B = 1 µA	4N35	BV_{CBO}	70			V
		4N36	BV_{CBO}	70			V
		4N37	BV_{CBO}	70			V
Collector emitter leakage current (2)	V_{CE} = 10 V, I_F = 0	4N35	I_{CEO}		5	50	nA
		4N36	I_{CEO}		5	50	nA
	V_{CE} = 10 V, I_F = 0	4N37	I_{CEO}		5	50	nA
	V_{CE} = 30 V, I_F = 0, T_{amb} = 100 °C	4N35	I_{CEO}			500	µA
		4N36	I_{CEO}			500	µA
		4N37	I_{CEO}			500	µA
Collector emitter capacitance	V_{CE} = 0		C_{CE}		6		pF
COUPLER							
Resistance, input output (2)	V_{IO} = 500 V		R_{IO}	10^{11}			Ω
Capacitance, input output	f = 1 MHz		C_{IO}		0.6		pF

Figura 53 - Datasheet optoisolatore (2)

Per le specifiche complete guardare datasheet allegato in appendice.

Da queste tabelle si ricavano i parametri più importanti che, se rispettati, garantiscono un corretto funzionamento dell'integrato:

- ai capi dei morsetti *1* e *2* si deve applicare una tensione di *1,3V* (tensione di soglia) per poter far circolare all'interno del fotodiodo una corrente pari a *10mA*
- la massima corrente che può circolare nel circuito primario è di *50mA*
- la massima tensione inversa che l'integrato può sopportare è di *6V*

Dai parametri ottenuti, per un utilizzo corretto dell'optoisolatore si devono prendere i seguenti accorgimenti:

1. adattare la tensione di alimentazione della lampada (*230V*) a quella nominale dell'integrato (*1,3V*)
2. il primario è costituito da un fotodiodo, il quale emette radiazioni luminose solamente quando la tensione impressa ai morsetti è positiva e superiore alla tensione di soglia. Di conseguenza si deve raddrizzare e filtrare la sinusoide in modo da renderla simile ad una tensione continua ed evitare il funzionamento a luce pulsata
3. limitare la corrente che percorre il primario evitando che di superare i *50mA* per non compromettere irrimediabilmente il componente

Per soddisfare il primo punto sono due le soluzioni possibili:

- usare una resistenza di shunt
- usare un trasformatore a doppio isolamento

Nel primo caso si realizza un circuito dissipativo ed il valore della resistenza deve essere variato a seconda del numero di lampade collegate a quel determinato gruppo; nel secondo caso si collega in parallelo alla lampada un trasformatore a doppio isolamento con rapporto di trasformazione tale da abbassare la tensione da *230V* a *6V*. Questo metodo, anche se più dispendioso dal punto di vista dei componenti, permette di aggiungere un'ulteriore isolamento al circuito di misura e di ridurre i costi energetici.

Per la realizzazione del secondo prototipo è stata scelta quest'ultima soluzione. Durante la costruzione e le relative prove di collaudo in laboratorio, è stata misurata una corrente al primario del trasformatore troppo elevata rispetto a quella assorbita dal secondario (che varia da *10mA* a *50mA*), questo eccesso è dovuto alla corrente di magnetizzazione che in questo caso è confrontabile con quella del carico. Il problema si risolve con un opportuno rifasamento del trasformatore tramite un banco di condensatori.

Tra i vari trasformatori di tensione presenti sul mercato si è scelto uno commercializzato da *RS Pro* le cui caratteristiche sono riportate nel datasheet presente in appendice.

Le caratteristiche più importanti estratte dalla documentazione sono:

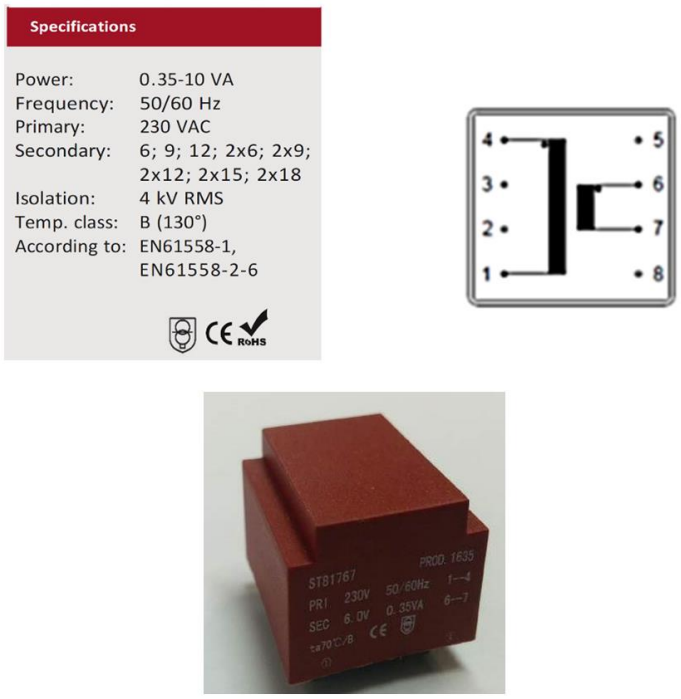

Figura 54 - Trasformatore circuito di misura della tensione

RS Stock number	Power VA	Specification	Primary [V]	Secondary voltage [V]	Secondary current [mA]	No-load voltage ±10% [V]
1213819	0.35	PCB mount transformer 0.35VA 1x6 o/p	230	6	58.3	9.7

Power	Dimensions			Pins		Weight
[VA]	L1 [mm]	W [mm]	H [mm]	R [mm]	Spaces	[kg]
0.35	22	23	15	15	2x4	0.03

Figura 55 - Datasheet trasformatore circuito di misure della tensione

Il primario del trasformatore è collegato in parallelo ai cavi che alimentano la lampada e la tensione su di esso ha il valore efficace di *230V* con andamento sinusoidale e frequenza di *50Hz*. Al secondario la tensione ha sempre andamento sinusoidale, la frequenza è sempre *50Hz* con valore efficace pari a *6V* (*9,7V* a vuoto).

La tensione ai capi dell'optoisolatore deve superare quella di soglia (*1,3V*) ed essere il più possibile continua per evitare il funzionamento intermittente del transistor posto sul secondario.

Da questi dati si deduce che non è possibile collegare direttamente i due componenti, pena la distruzione dell'optoisolatore. Tra i due dispositivi si deve interporre una resistenza sulla quale creare una caduta di tensione e che limiti la corrente del circuito di misura. Questa semplice accortezza evita sia il breakdown del diodo per un eccesso di tensione, sia la distruzione del componete dovuto al superamento dei limiti termici per un eccesso di corrente.

Come anticipato in precedenza, il fotodiodo emette radiazione luminosa solamente se viene alimentato con tensione positiva superiore ad una certa soglia, di conseguenza se l'andamento è sinusoidale a *50 Hz* il secondario dell'optoisolatore diventa un cortocircuito per circa *10ms* (una semionda) ed un circuito aperto per lo stesso periodo di tempo. Questo tipo di funzionamento non è adatto per il funzionamento del prototipo. La soluzione è raddrizzare la semionda negativa utilizzando un ponte a diodi. A differenza del primo prototipo, in cui sono stati utilizzati quattro diodi zener, in questo viene usato un circuito integrato in modo da ottimizzare ulteriormente gli spazi ed evitare errori di saldatura durante il montaggio.

L’integrato scelto è il *2W08G 416C* prodotto dalla *Vishay*. Di seguito vengono riportati l’immagine, lo schema del dispositivo e le caratteristiche più importanti estratte dal datasheet:

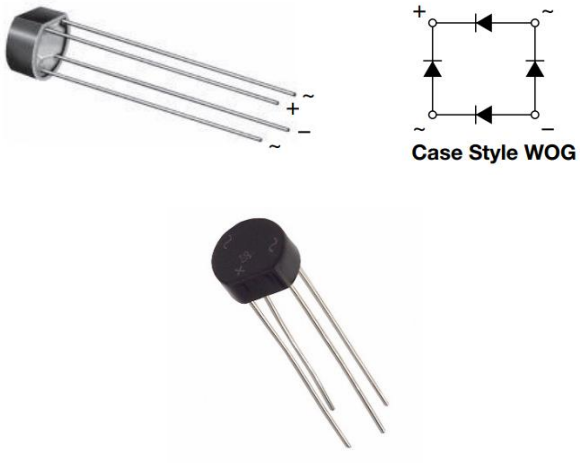

Figura 56 - Ponte a diodi integrato

PRIMARY CHARACTERISTICS	
Package	WOG
$I_{F(AV)}$	2.0 A
V_{RRM}	50 V, 100 V, 200 V, 400 V, 600 V, 800 V, 1000 V
I_{FSM}	60 A
I_R	5 µA
V_F at I_F = 2.0 A	1.1 V
T_J max.	150 °C
Diode variations	Quad

Figura 57 - Datasheet ponte a diodi (1)

MAXIMUM RATINGS (T_A = 25 °C unless otherwise noted)									
PARAMETER	**SYMBOL**	**2W005G**	**2W01G**	**2W02G**	**2W04G**	**2W06G**	**2W08G**	**2W10G**	**UNIT**
Maximum repetitive peak reverse voltage	V_{RRM}	50	100	200	400	600	800	1000	V
Maximum RMS voltage	V_{RMS}	35	70	140	280	420	560	700	V
Maximum DC blocking voltage	V_{DC}	50	100	200	400	600	800	1000	V
Maximum average forward rectified current at 0.375" (9.5 mm) lead length at (fig. 1)	$I_{F(AV)}$	2.0							A
Peak forward surge current single half sine-wave superimposed on rated load	I_{FSM}	60							A
Rating for fusing (t < 8.3 ms)	I^2t	15							A^2s
Operating junction and storage temperature range	T_J, T_{STG}	- 55 to + 150							°C

Figura 58 - Datasheet ponte a diodi (2)

Le caratteristiche più importanti estratte dalle tabelle precedenti sono:

- la massima tensione inversa di picco: *800V*
- la massima corrente media che può sopportare: *2A*

Il datashhet completo del ponte a diodi è allegato in appendice.

Questi parametri permetto di avere un margine di funzionamento molto più ampio rispetto alle grandezze che vengono utilizzate nel normale funzionamento.

L'utilizzo del ponte a diodi risolve in parte il problema, ma ci sono ancora degli istanti in cui la tensione sull'optoisolatore scende al di sotto del valore di soglia, facendo aprire il transistor al secondario, il che comporta una valutazione errata da parte di Arduino. Il problema è facilmente risolvibile: si pone un filtro passa basso tra il ponte a diodi e l'optoisolatore; inoltre si pone una resistenza in parallelo al condensatore in modo da permettergli di scaricarsi più velocemente quando non viene più alimentato dal trasformatore.

Dopo vari tentativi sono stati scelti i seguenti valori per i tre componenti del filtro:

- la resistenza in serie: *R=470*Ω (per limitare le sovracorrenti)
- il condensatore: *C=150µF*
- la resistenza in parallelo: *R=2200*Ω

Di seguito è riportato lo schema utilizzato per la realizzazione del circuito di misura della tensione:

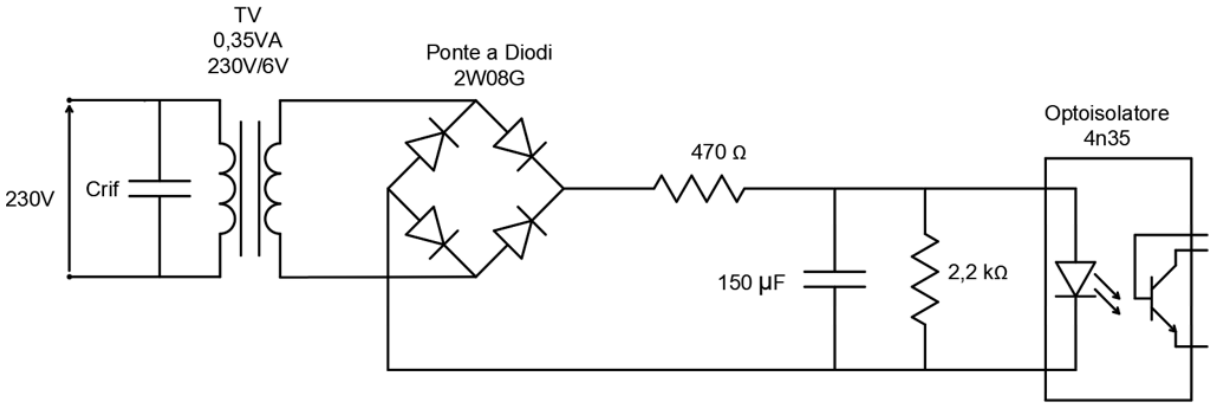

Figura 59 - Schema del circuito di misura della tensione secondo prototipo

Lo schema di misura di corrente utilizzato nel prototipo precedente ha subito delle modifiche:

- le resistenze e la capacità sono stati sostituiti con altri più adatti a questa nuova tipologia di circuito
- il ponte a diodi è stato sostituito con l'integrato utilizzato nel circuito di misura della tensione
- in parallelo al condensatore ed alla resistenza è stato aggiunto un mosfet in modo da poter utilizzare la lettura digitale di Arduino invece che quella analogica

Tra le alternative a questo tipo di componente si è valutato anche l'utilizzo di un *transistor*. Quest'ultimo, per chiudere il circuito tra *Collettore* ed *Emettitore*, necessita di una corrente da iniettare nella base del dispositivo. Nel mosfet invece per chiudere il circuito tra *Drain* e *Source* si deve solamente imporre una tensione ai capi dei morsetti di *Gate* e *Source*. Questo significa che si può utilizzare il trasformatore amperometrico del prototipo precedente e siccome la corrente assorbita per la presenza del mosfet è minima, allora la potenza necessaria per il funzionamento del circuito di misura è di ordini di grandezza inferiore rispetto a quella assorbita dalla singola lampada.

I componenti elettronici come ad esempio gli IGBT non sono stati presi in considerazione perché necessitano di una tensione di tensione di comando superiore dispetto ai mosfet.

Il mosfet scelto è lo *ZVN2106A*:

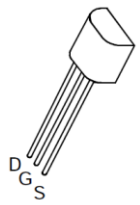

Figura 60 - Mosfet ZVN2106A

Dal datasheet completo allegato in appendice si sono estratte le seguenti caratteristiche:

PARAMETER	SYMBOL	VALUE	UNIT
Drain-Source Voltage	V_{DS}	60	V
Continuous Drain Current at T_{amb}=25°C	I_D	450	mA
Pulsed Drain Current	I_{DM}	8	A
Gate Source Voltage	V_{GS}	± 20	V
Power Dissipation at T_{amb}=25°C	P_{tot}	700	mW
Operating and Storage Temperature Range	T_j:T_{stg}	-55 to +150	°C

Figura 61 - Datasheet mosfet (1)

ELECTRICAL CHARACTERISTICS (at T_{amb} = 25°C unless otherwise stated).

PARAMETER	SYMBOL	MIN.	MAX.	UNIT	CONDITIONS.
Drain-Source Breakdown Voltage	BV_{DSS}	60		V	I_D=1mA, V_{GS}=0V
Gate-Source Threshold Voltage	$V_{GS(th)}$	0.8	2.4	V	ID=1mA, V_{DS}= V_{GS}
Gate-Body Leakage	I_{GSS}		20	nA	V_{GS}=± 20V, V_{DS}=0V
Zero Gate Voltage Drain Current	I_{DSS}		500 100	nA µA	V_{DS}=60 V, V_{GS}=0 V_{DS}=48 V, V_{GS}=0V, T=125°C(2)
On-State Drain Current(1)	$I_{D(on)}$	2		A	V_{DS}=18V, V_{GS}=10V
Static Drain-Source On-State Resistance (1)	$R_{DS(on)}$		2	Ω	V_{GS}=10V,I_D=1A
Forward Transconductance (1)(2)	g_{fs}	300		mS	V_{DS}=18V,I_D=1A
Input Capacitance (2)	C_{iss}		75	pF	V_{DS}=18 V, V_{GS}=0V, f=1MHz
Common Source Output Capacitance (2)	C_{oss}		45	pF	
Reverse Transfer Capacitance (2)	C_{rss}		20	pF	

Figura 62 - Datasheet mosfet (2)

I parametri più importanti sono:

- la tensione di Breakdown tra i morsetti D e S: V_{DS}=*60V*
- la corrente di leakage del Gate: I_G=*20nA*
- la tensione di soglia massima e minima tra i morsetti G e S: V_{GSmin}=*0,8V* e V_{GSmax}=*2,4V*

Da secondo punto si ha la conferma che la corrente assorbita dal mosfet durante il funzionamento è talmente piccola da poter essere considerata trascurabile.

Dai primi test effettuati in laboratorio su questo nuovo circuito sono sorti dei problemi sull'innesco del mosfet, infatti la tensione tra Gate e Source non era abbastanza elevata e continua per consentire il funzionamento corretto del dispositivo. Questo problema creava delle persistenti aperture e chiusure del mosfet che davano ad Arduino dei segnali errati. Per garantire un corretto funzionamento del componente sono stati aumentati gli avvolgimenti al primario del TA passando da otto a quindici. La soluzione adottata ha portato ad un aumento della tensione ai capi del condensatore utilizzato per il filtraggio a cui è collegato il mosfet risolvendo il problema delle aperture-chiusure incontrollate.

Come anticipato in precedenza le variazioni ai componenti utilizzati per la realizzazione pratica del circuito sono:

- ponte a diodi integrato *2W08G 416C* prodotto dalla *Vishay* (lo stesso utilizzato per il circuito di misura della tensione)
- una resistenza variabile settata al valore di *500*Ω che limita la sovracorrente dovuta all'alimentazione del condensatore scarico
- una resistenza da *18k*Ω per scaricare il condensatore
- un condensatore da *1µF*

Lo schema del circuito di misura della corrente diventa:

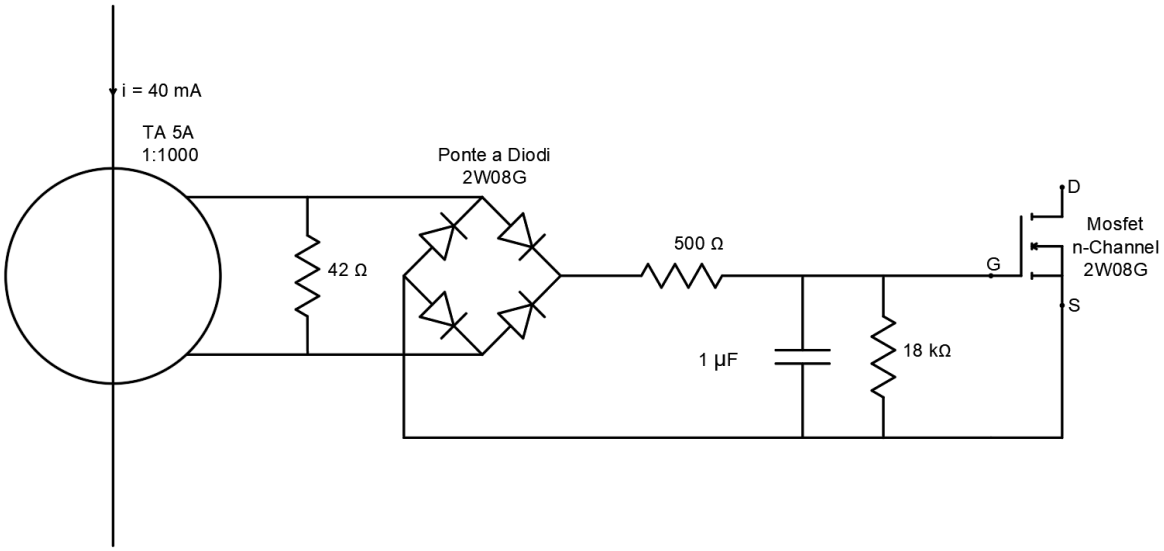

Figura 63 - Schema circuito di misura della corrente secondo prototipo

Per snellire, semplificare e velocizzare il codice di Arduino, si è optato per passare dalla lettura analogica a quella digitale dei valori ottenuti dai due circuiti di misura. Arduino attraverso il comando *"digitalRead()"* legge il valore di tensione di uno specifico PIN e identifica se questo si trova nello stato *HIGH* o nello stato *LOW*.

Questi due stati vengono definiti come segue:

1. il PIN si trova nello stato *HIGH* se la tensione rispetto al riferimento (*GROUND*) è superiore a *3,0V* se è alimentato con il PIN a *5V* della scheda stessa oppure se è superiore a *2,0V* nel caso di alimentazione dal PIN a *3,3V* sempre della scheda

2. il PIN si trova nello stato *LOW* se la tensione rispetto al riferimento (*GROUND*) è inferiore a *1,5V* se è alimentato con il PIN a *5V* della scheda stessa oppure se è inferiore a *1,0V* nel caso di alimentazione dal PIN a *3,3V* sempre della scheda

Con questa soluzione si evita di andare a leggere analogicamente i valori della tensione su un certo periodo per ottenere il più grande ed essere sicuri che lo stato della lampada sia quello e non un errore dovuto all'oscillazione della grandezza elettrica. Ai capi del mosfet e dell'optoisolatore si impone la tensione di *5,0V* del pin presente su Arduino Due.

Un ulteriore sviluppo del prototipo è la realizzazione di un comparatore che eviti di effettuare due letture sequenziali separate sui circuiti di misura della tensione e della corrente.

Lo schema finale del circuito di misura totale è il seguente:

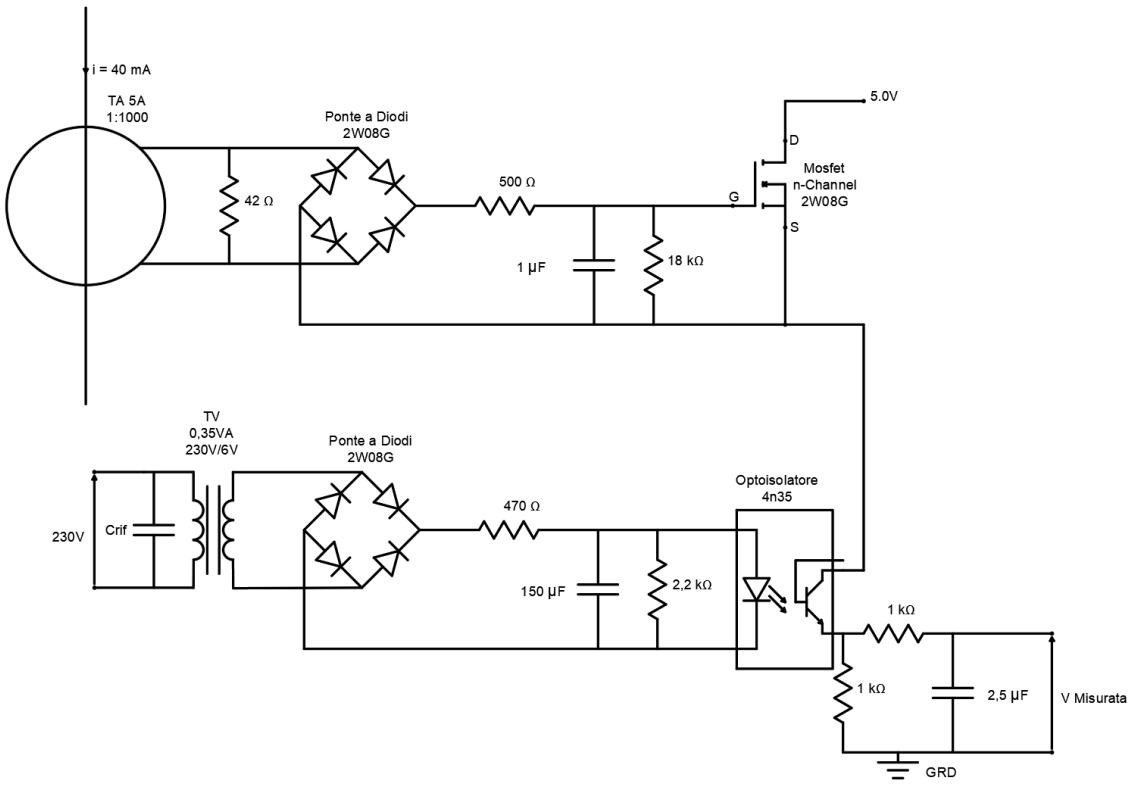

Figura 64 - Schema di misura finale secondo prototipo

Per la realizzazione si sono collegati in serie il mosfet ed il secondario dell'optoisolatore; il *Drain* del mosfet è collegato al Pin di Arduino, il quale fornisce la tensione continua (*5,0V*) mentre sul Pin numero *4* dell'optoisolatore, che corrisponde all'*Emettitore* del transistor, sono collegate due resistenze da *1k*Ω, un condensatore da *2,5µ*F ed il PIN con il riferimento della tensione (*GROUND*). La lettura della tensione ai capi di questo condensatore ci permette di capire se la lampada a cui sono collegati i due circuiti si trova nello stato *HIGH* o nello stato *LOW*.

Per come è stato impostato il comparatore, Arduino rileva lo stato *HIGH* solo se sulla lampada sono presenti contemporaneamente sia tensione che corrente; nel caso mancasse una delle due grandezze (es. quando la lampada è bruciata non c'è corrente, mentre la tensione è presente) o entrambe allora Arduino rileverà lo stato *LOW*.

Di seguito è riportata una foto della parte circuitale del prototipo:

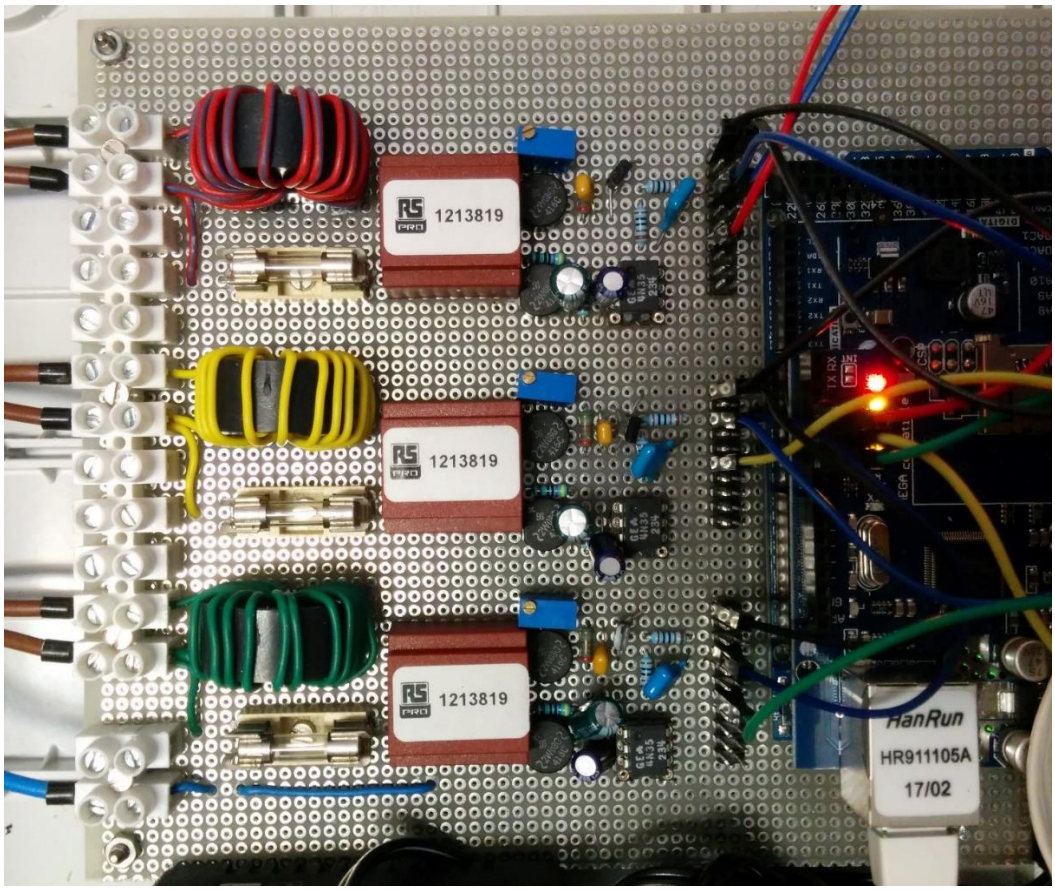

Figura 65 - Circuito di misura e comparazione del secondo prototipo

Come si può vedere dalla foto i tre circuiti di misura e di comparazione sono stati saldati su una scheda millefori come nel primo prototipo, scheda sulla quale sono stati posti su uno stesso piano Arduino due con l'ethernet shield ed il router *Wi-Fi* per la comunicazione con i dispositivi mobili. È stata aggiunta una morsettiera per facilitare il collegare dei tre cavi di alimentazione delle tre lampade ed il neutro. Infine, come ulteriore precauzione sono presenti tre fusibili da *20mA* per proteggere il circuito di misura della tensione da eventuali corto circuiti e sovracorrenti.

2.7.2 Scrittura del software

Come nel codice del primo prototipo, di cui questo è l'evoluzione, sono state create diverse funzioni, editate alla fine del programma principale, le quali possono essere richiamate più volte senza essere riscritte. Con questa semplice accortezza si riesce a creare uno programma più snello e semplice da analizzare da parte del lettore.

2.7.2.1 Codice di programmazione secondo prototipo

Per la stesura completa del programma di Arduino si rinvia agli allegati; di seguito sono riportati il flow chart principale e quelli delle varie funzioni in cui è stato suddiviso che hanno subito modifiche rispetto a quelle del prototipo precedente. Per ogni diagramma di flusso è fornita una breve descrizione delle differenze con quello del prototipo precedente.

Nel programma principale (*fig.66*) sono poche le differenze rispetto a quello precedente:

- vengono definiti i tre pin (*4,5,6*) collegati ai tre comparatori da usare come input per la lettura digitale
- non viene più richiamata la funzione ***Led()*** in quanto non è più previsto l'utilizzo dei led come countdown visivo per i pedoni

Il codice richiama le funzioni: ***Misura()***, ***Statosemaforo()***, ***PaginaLocale()***, ***Tempistica()***, ***Tempisticastock()***, ***Tempisticareset()***, ***Confronto()***, ***Cambiostato()*** e ***Reset()***.

Alcune di esse, come descritto in seguito, sono state modificate rispetto al programma precedente per adattarle alla lettura digitale e migliorare il funzionamento software del prototipo.

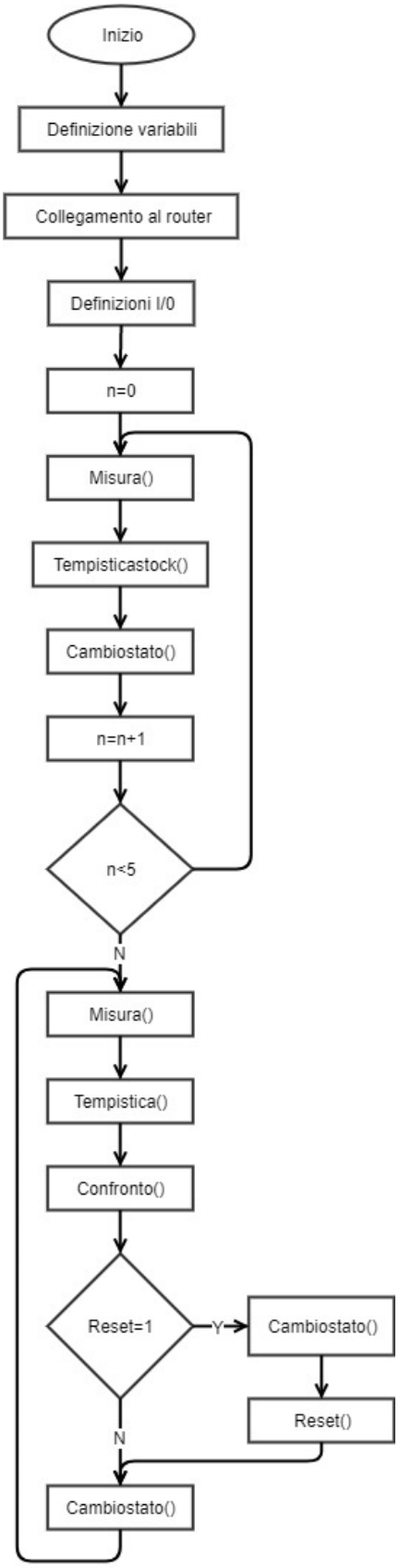

Figura 66 - Diagramma di flusso secondo prototipo

Vediamo ora le funzioni richiamate durante l'esecuzione del codice.

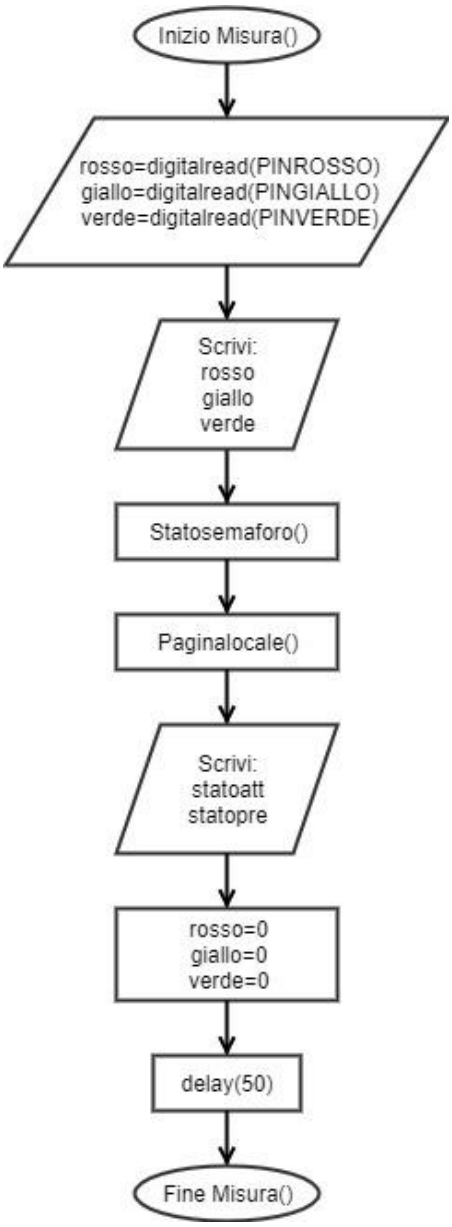

Figura 67 - Diagramma di flusso Misura()

La funzione ***Misura()*** richiamata in questo codice, grazie alla presenza del comparatore analogico, non deve più effettuare la lettura analogica su un periodo di campionamento di*200ms* per cercare il massimo delle grandezze, ma esegue quella digitale sui pin *4*, *5* e *6* su cui sono collegati i tre circuiti di comparazione. Un'altra differenza consiste che in questa parte di codice non vengono più inviate sul monitor seriale informazioni sullo stato della lanterna se questa è spenta o lampeggiante perché implementate in altre funzioni. Infine è stato aggiunto un ritardo di *50ms* per non far lavorare il processore a frequenze troppo elevate, poiché questo causerebbe il surriscaldamento della scheda ed un consumo energetico troppo elevato.

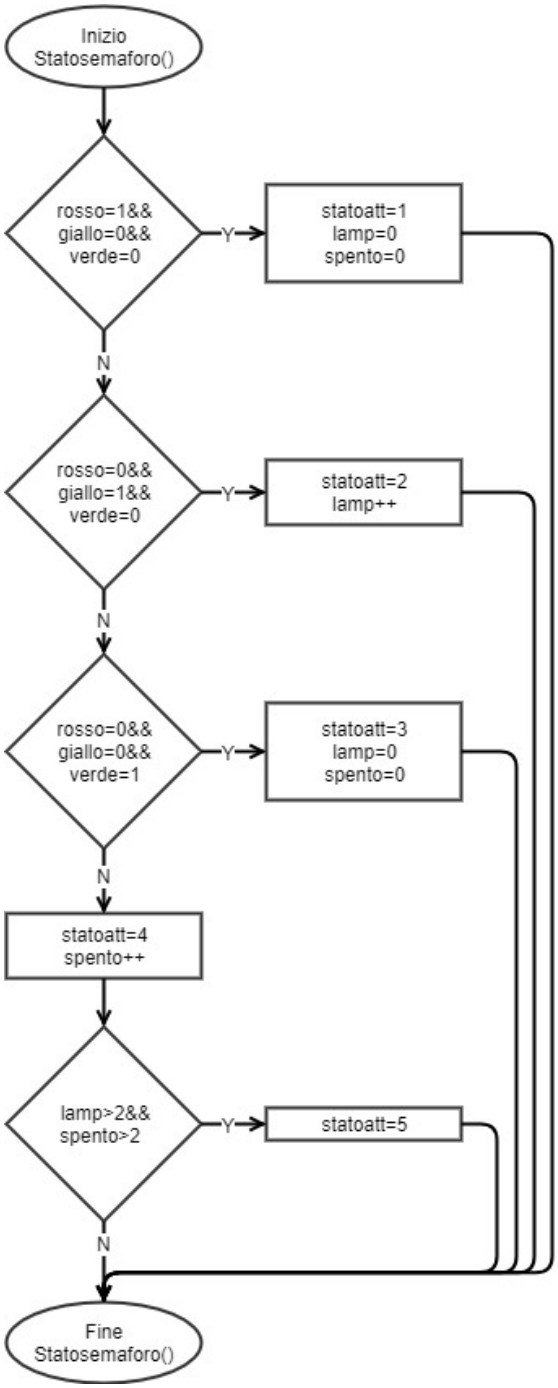

Figura 68 - Diagramma di flusso Statosemaforo()

In ***Statosemaforo()*** la differenza sostanziale con il prototipo precedente è la possibilità da parte del programma di riconoscere non solo quale luce è accesa in un determinato istante e quindi associare lo stato corretto (*1* se rosso, *2* se giallo e *3* se è verde), ma riesce a individuare correttamente quando una delle tre lampade è guasta (*stato 4*) o se il semaforo è giallo lampeggiante (*stato 5*). Nel codice precedente lo *stato 5* non era presente e il riconoscimento dello *stato 4* da parte di Arduino presentava diverse difficoltà ora risolte.

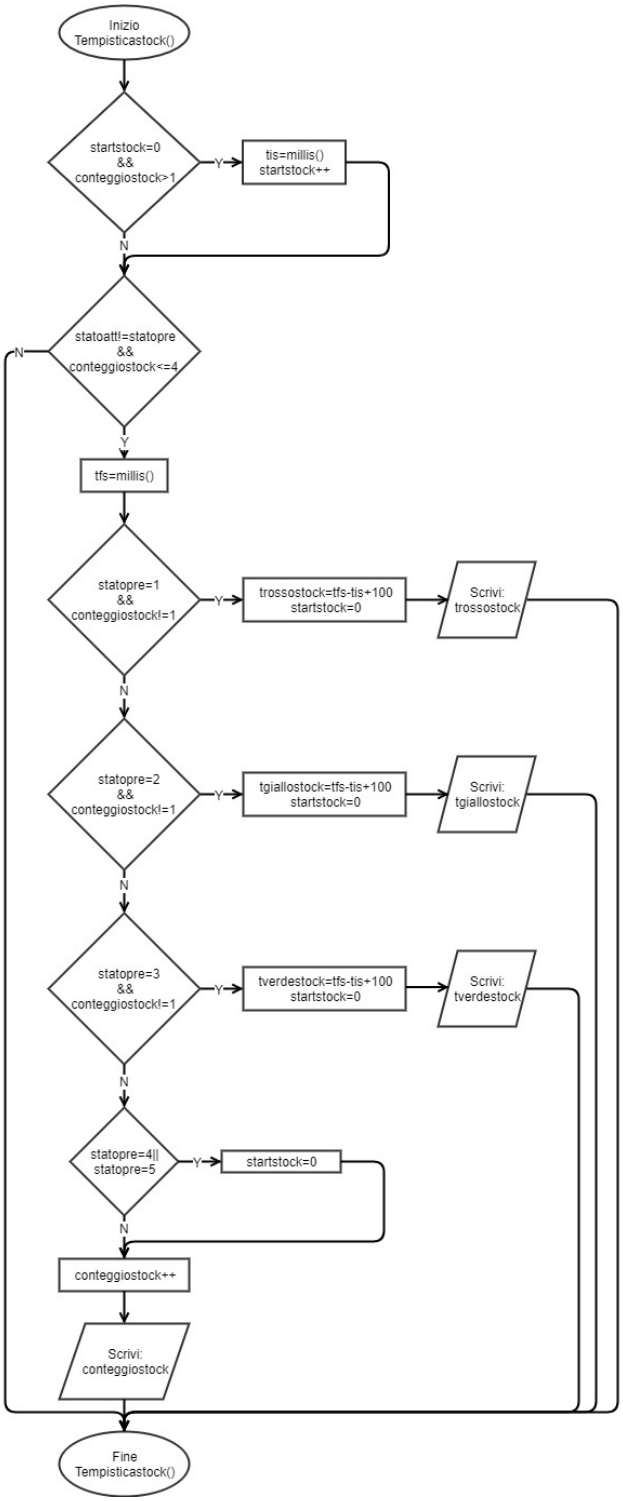

Figura 69 - Diagramma di flusso Tempisticastock()

In ***Tempisticastock()*** se lo stato del semaforo corrisponde a *1*, *2* o *3* allora il funzionamento di questa parte di codice è identico a quello del prototipo precedente in cui vengono determinati i tempi di durata di ogni stato. La differenza con la funzione omonima del primo prototipo sta nel fatto che se una delle lampade è bruciata (*stato 4)* o il semaforo è giallo lampeggiante (*stato 5*) allora il calcolo dei tempi viene bloccato.

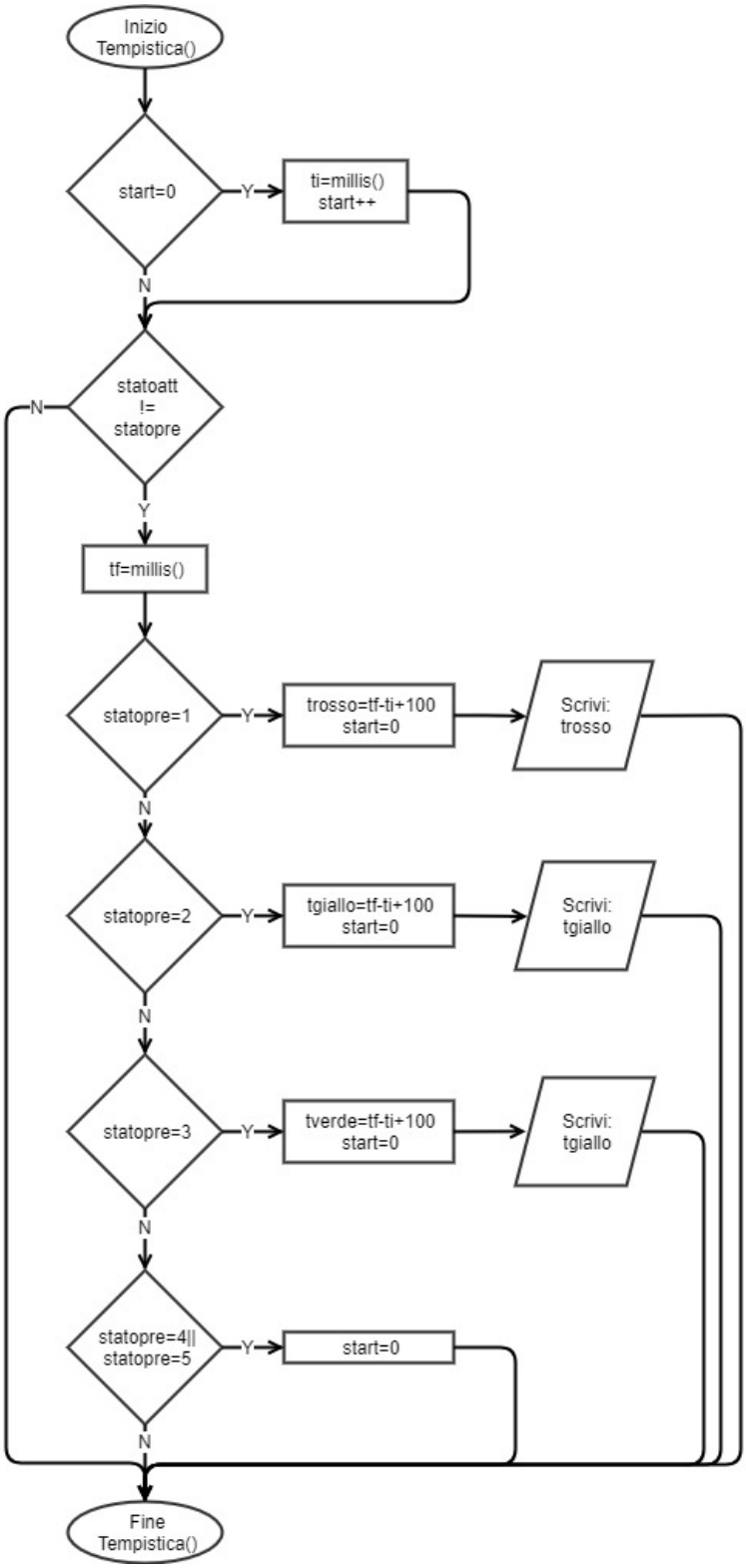

Figura 70 - Diagramma di flusso Tempistica()

Anche in questo caso, la funzione ***Tempistica()***, come per ***Tempisticastock()***, ha lo stesso funzionamento descritto per il prototipo precedente, tranne nel caso in cui lo stato della lanterna sia il *4* o il *5*: in quei due casi il calcolo dei tempi viene bloccato.

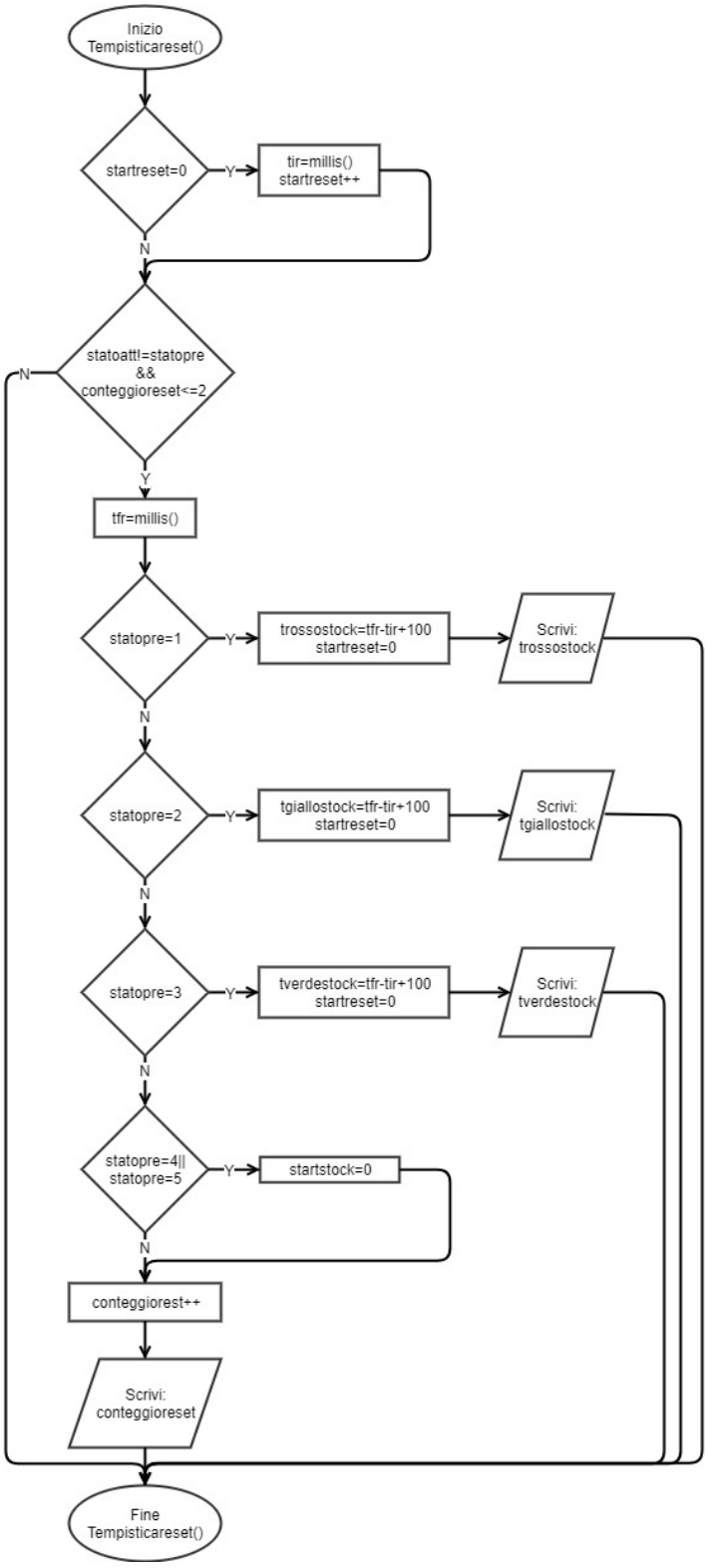

Figura 71 - Diagramma di flusso Tempisticareset()

Come nei due casi precedenti, anche in ***Tempisticareset()*** il funzionamento, quando lo stato è *1*, *2* o *3*, è identico a quello del primo prototipo, mentre viene inibito il calcolo dei tempi se lo stato corrisponde a *4* o *5*.

2.8 Test del secondo prototipo

Come per il precedente, anche il secondo prototipo è stato testato in laboratorio con il circuito di emulazione di un attraversamento pedonale; le diverse prove eseguite avevano lo scopo di verificare che il dispositivo funzionasse correttamente in modo da poterlo installare in un vero quadro semaforico per dei test sul campo.

I risultati ottenuti in laboratorio sono più che soddisfacenti, quindi in accordo con *Iren S.p.A*, società che gestisce gli impianti semaforici a Torino, si è deciso di provare questo nuovo prototipo su una lanterna di una vera centralina semaforica in data 27 luglio 2017.

L'incrocio scelto per questi test è quello di *Via Vittorio Emanuele* con *Via dell'Arsenale* a *Torino*; la lanterna concessa per le prove è quella che regola l'attraversamento pedonale del controviale lato *Piazza Carlo Felice*.

Per poter essere installato nel quadro semaforico il prototipo ha subito le seguenti modifiche:

- è stato inscatolato per evitare qualunque tipo di contatto diretto con le parti in tensione e che detriti o la polvere possano danneggiarlo compromettendo il funzionamento
- all'interno della scatola si è aggiunta una multipresa a cui sono collegati gli alimentatori del router Wi-Fi e di Arduino
- è stata creata una prolunga per l'antenna del router Wi-Fi

Il risultato finale è quello che segue:

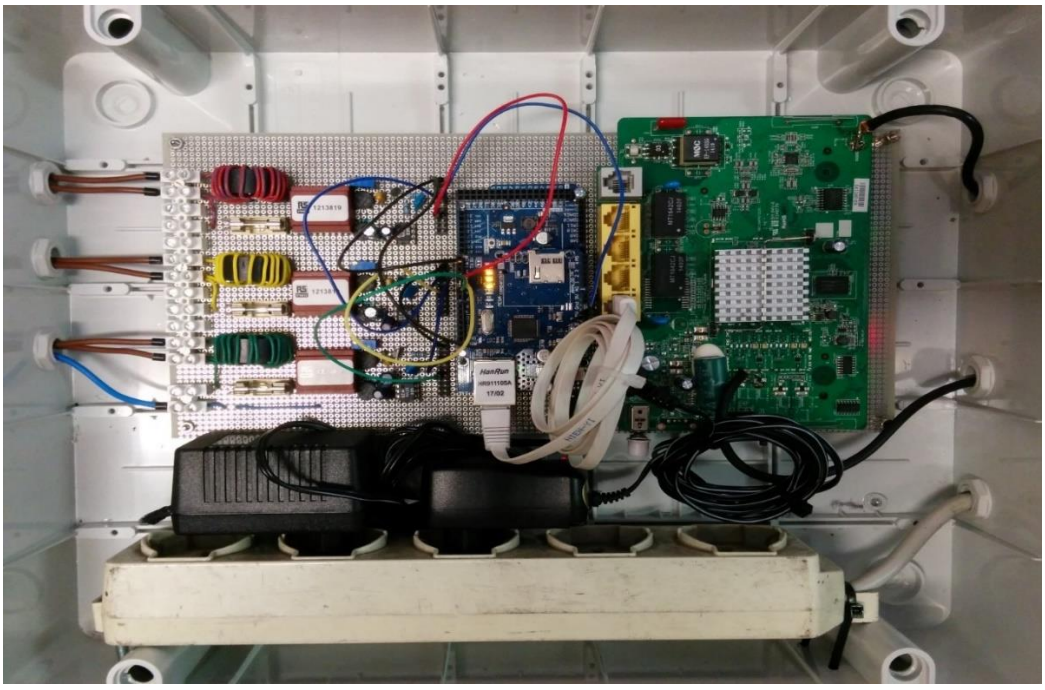

Figura 72 - Prototipo per test su strada

Per l’alimentazione del dispositivo si è usufruito della presa di servizio presente nella centralina.

L’installazione è stata eseguita da parte degli operai dell’*Iren* con la supervisione dei membri del dipartimento *DENERG* del *Politecnico di Torino* che si sono occupati della progettazione e lo sviluppo di questo progetto. Ad oggi, ottobre 2017, questo prototipo è ancora installato all’interno del quadro ed è perfettamente funzionante.

Durante l’installazione, la prolunga dell’antenna è stata fatta passare vicino alle griglie di areazione della cappelliera del quadro, in modo da aumentare la ricezione da parte dei dispositivi posti nelle zone più remote dell’incrocio; non è stato possibile, infatti, apporre un’antenna esterna perché sarebbe stata soggetta ad atti vandalici da parte di soggetti sconosciuti.

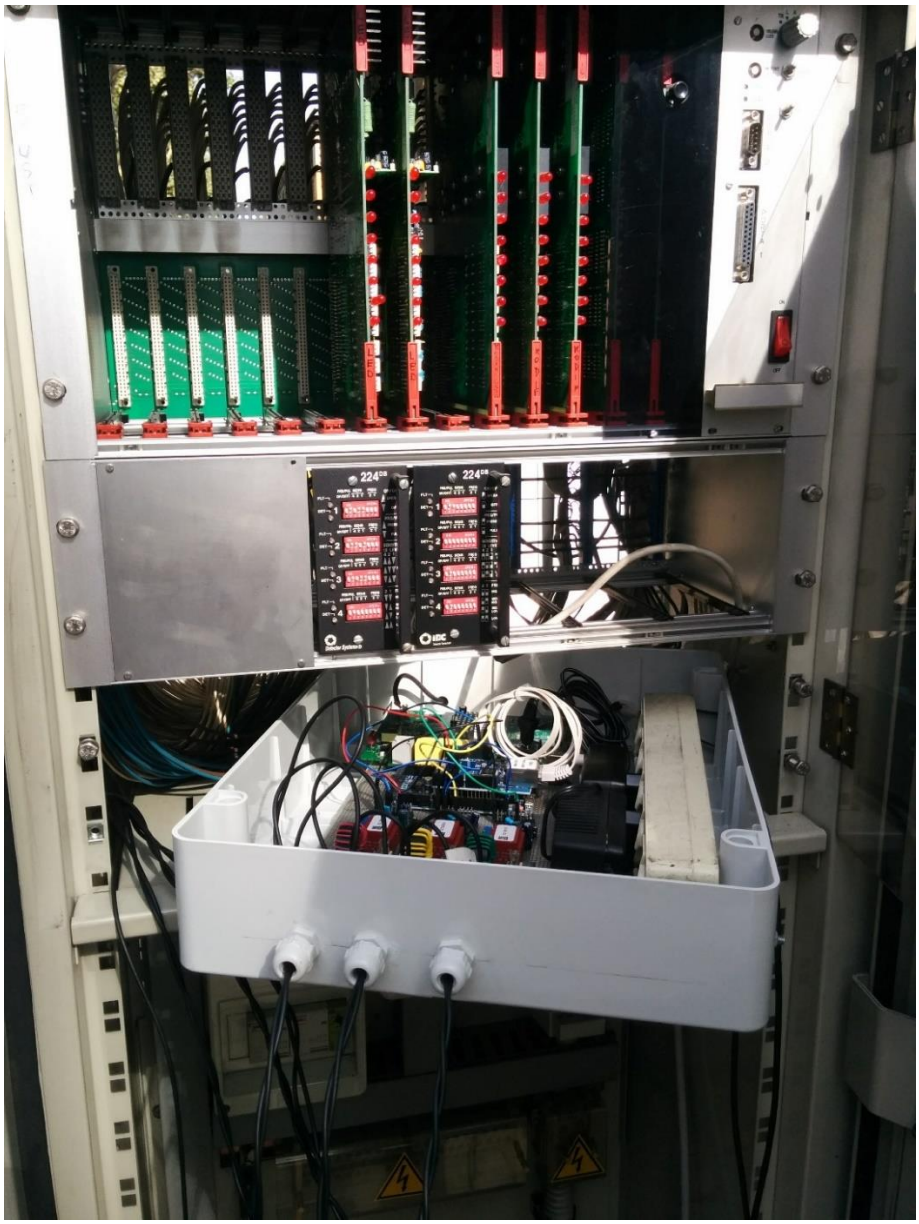

Figura 73 - Installazione del secondo prototipo

L’esperienza è servita per effettuate varie prove in cui si sono verificati i seguenti punti:

- il normale funzionamento della lanterna semaforica dopo l’installazione del prototipo
- la semplicità dell’installazione e che richiedesse poco tempo all’operatore
- il corretto funzionamento del prototipo dopo l’installazione
- l’estensione del segnale *Wi-Fi* adeguata alla grandezza dell’incrocio
- i ritardi tra la lettura del segnale, l’elaborazione, l’invio e la ricezione del messaggio fossero adeguati a questo tipo di funzionamento

L’esponente dell’**Unione Italiana Ciechi e Ipovedenti,** già presente alle prove del primo prototipo, ha testato di persona anche questo nuovo dispositivo rimanendo soddisfatto dell’operato, fornendo nuovi pareri e spunti ed ha invitato caldamente a trovare una soluzione alle problematiche sorte già con il primo prototipo che non sono state ancora risolte.

2.8.1 Problematiche del secondo prototipo

Durante le varie prove condotte all’incrocio di *Corso Vittorio Emanuele - Via dell’Arsenale*, è sorto un nuovo tipo di problema che in laboratorio non era stato preso in considerazione: la schermatura del quadro in cui è installata la centralina semaforica. Il quadro in questione è costruito con materiale metallico ed è bloccato a terra per non creare pericolo ai pedoni che passano a fianco quando attraversano l’incrocio. Il collegamento con il terreno rende la struttura una vera e propria gabbia di Faraday: i segnali inviati dal router tramite *Wi-Fi* non riescono a propagarsi al di fuori del quadro perché incontrano la schermatura che ne impedisce il passaggio all’ambiente circostante. Finché la porta del quadro è rimasta aperta, l’intensità e la zona coperta dal segnale erano da considerarsi sufficienti per questo tipo di incrocio, ma una volta chiusa, il livello del segnale captato dai dispositivi è crollato a zero in molte parti dell’incrocio tranne che dal lato in cui è stata fatta passare l’antenna dalle griglie di ventilazione.

Purtroppo questo risultato è da considerarsi insufficiente con l’utilizzo preposto per questo dispositivo, infatti non potendo installare nessun tipo di antenna esterna rende questo tipo di comunicazione inutilizzabile.

Bisogna ricordare che, oltre a questo nuovo problema, sono ancora presenti le vecchie problematiche sorte già con il primo prototipo:

- l’utilizzo della comunicazione *Wi-Fi* che, oltre ad essere molto energivora, impedisce la ricezione delle notifiche perché disconnette lo smartphone dalla rete internet
- l’applicazione per Android con un’interfaccia troppo complicata per essere utilizzata da un non vedente

Le soluzioni a queste problematiche vengono riportate nel *Capitolo 4* di questo elaborato.

Cap. 3: Applicazione Android per non vedenti e ipovedenti

3.1 Generalità

La programmazione delle applicazioni per smartphone e tablet è, di solito, un processo molto laborioso, in cui la maggior parte delle ore viene impiegata nella scrittura del codice che ne definirà sia il funzionamento che l'interfaccia con l'utente. Per evitare tutto ciò e diminuire di molto il tempo impiegato per raggiungere l'obbiettivo, lo strumento più adatto per procedere in questa fase è l'utilizzo del sito ***MIT App Inventor*** (http://appinventor.mit.edu/explore/).

In questo capitolo è descritta la realizzazione ed il funzionamento dell'applicazione per smartphone Android che consente la comunicazione delle informazioni ricevute dal primo e dal secondo prototipo. Entrambi i dispositivi sono descritti nel *Capitolo 2* di questa tesi (*2.5* e *2.7*).

3.2 MIT App Inventor

Creato dall'Università del Massachusetts questo sito consente di realizzare applicazioni per Android in modo molto più semplice e intuitivo rispetto alla programmazione standard; inoltre permette di testare in qualsiasi momento il lavoro svolto con i seguenti metodi:

- utilizzando la loro applicazione scaricabile da Google play Store
- tramite un emulatore sul computer di Android
- compilando il file ***.apk***. Il suddetto che viene scaricato sul pc o direttamente sul dispositivo tramite *QR Code*, per poi venire installato sul proprio smartphone o tablet (Android). Per utilizzare questa modalità bisogna impostare il dispositivo in modo da permettere l'installazione di file non proveniente da Google Play Store (Impostazioni>Sicurezza>Amministrazione dispositivo> Origini sconosciute: Consenti l'installazione di app da origini sconosciute)

Per aiutare i neofiti alla programmazione delle applicazioni sono presenti vari tutorial sia sul sito stesso sia nei vari forum ed ogni funzione che si può utilizzare è spiegata nel minimo dettaglio, il che rende l'apprendimento molto più veloce e chiaro.

Per collegarsi al sito di MIT App Inventor si può utilizzare un qualsiasi browser, dopo di che si deve effettuare la registrazione obbligatoria inserendo username e password di un account Google. Completate queste operazioni si potrà accedere in qualunque momento alla propria area riservata dove si può trovare la lista di tutti i progetti personali (*fig.74*).

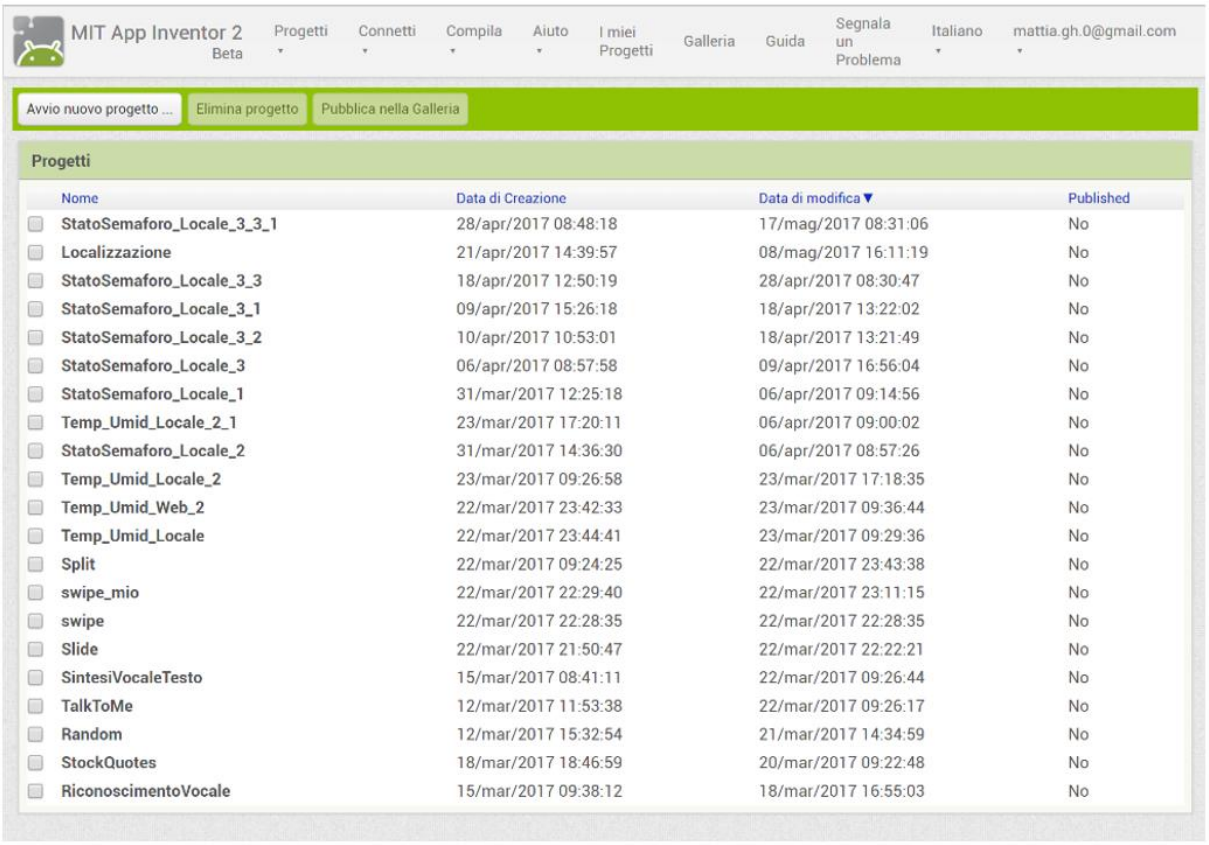

Figura 74 - MIT App Inventor schermata progetti

Cliccando sul pulsante *"Avvio nuovo progetto…"* viene chiesto all'utente di inserire un titolo e, dopo aver completato questo semplice passaggio, si può iniziare con la vera e propria programmazione dell'applicazione.

La schermata di progettazione (*fig.75*) si può suddividere in tre parti differenti:

- a sinistra sono presenti diversi menù a tendina da cui scegliere i componenti (visibili e non) che vengono utilizzati per la realizzazione dell'app
- al centro vi è posta un'anteprima della schermata del dispositivo (smartphone e tablet) su cui verrà caricata l'applicazione in modo da poterne ottimizzare il design
- a destra sono presenti l'elenco dei componenti utilizzati e le varie proprietà

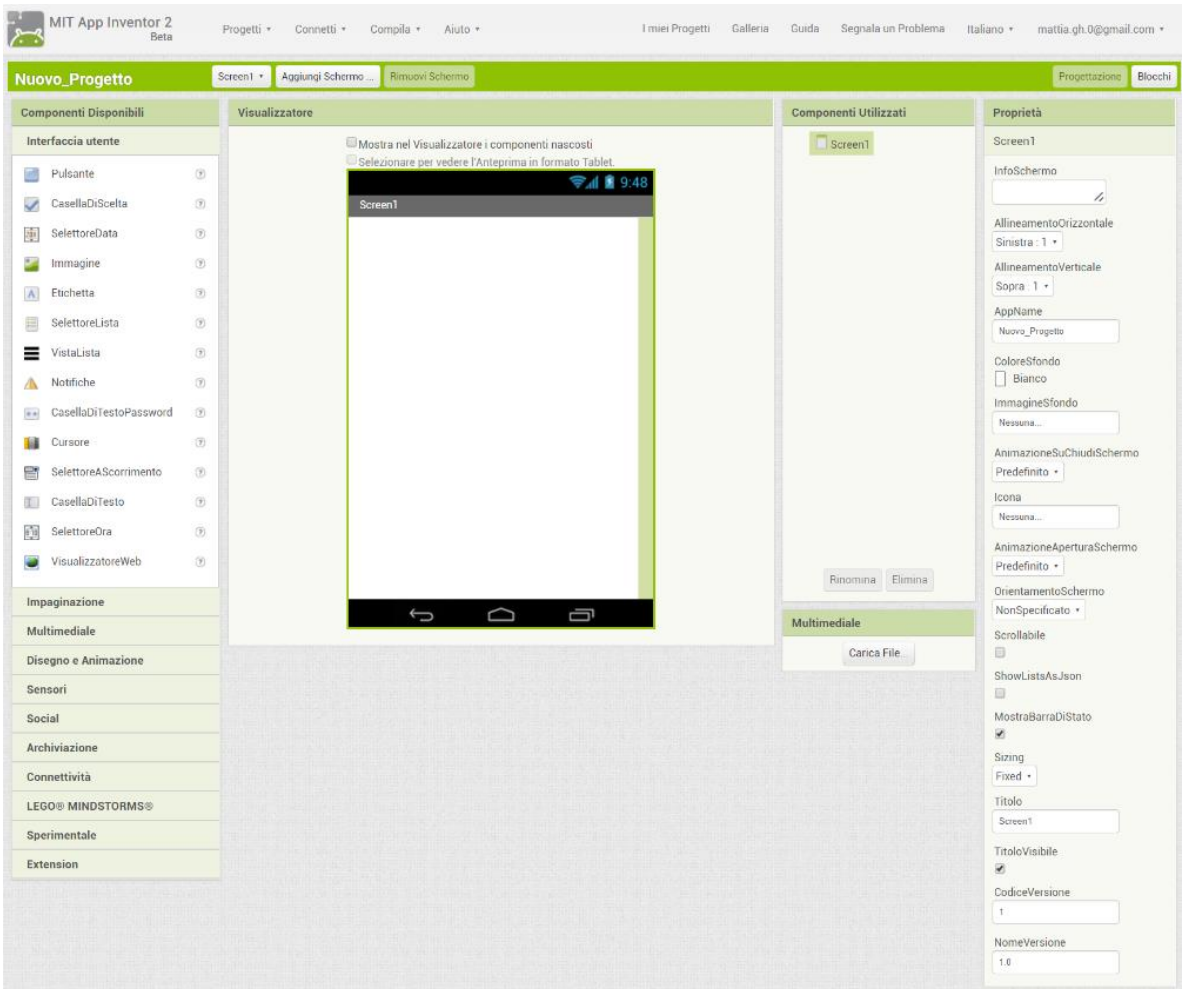

Figura 75 - MIT App Inventor Schermata nuovo progetto

Oltre a quella riportata in *figura 75* è presente una seconda schermata (*fig. 76*), denominata *"blocchi"*: questa è il cuore dell'applicazione, in cui mediante l'utilizzo di veri e propri blocchi logici ne viene definito e impostato il funzionamento. Uno dei vantaggi di utilizzare MIT App Inventor è la presenza di questa schermata, che permette anche a chi è poco competente nella programmazione Android di poter realizzare un'applicazione semplicemente incastrando tra di loro blocchi a cui corrispondono differenti righe di codice.

A sinistra dello schermo è presente la lista con tutti i componenti scelti nella schermata di progettazione. Cliccando su qualunque di essi si apre un menù a tendina che contiene tutti i corrispettivi blocchi. Per poterne utilizzare uno, lo si deve selezionare con il puntatore e trascinare sulla parte bianca dello schermo dedicata alla compilazione del codice.

I blocchi possono essere incastrati tra loro come un puzzle, possono essere raggruppati in funzioni le quali possono essere richiamate più volte senza dover ripetere gli stessi blocchi, in questo modo si ottiene un codice più snello ed ordinato. Le operazioni all'interno delle funzioni vengono soddisfatte in modo consecutivo, una alla volta, dall'alto verso il basso.

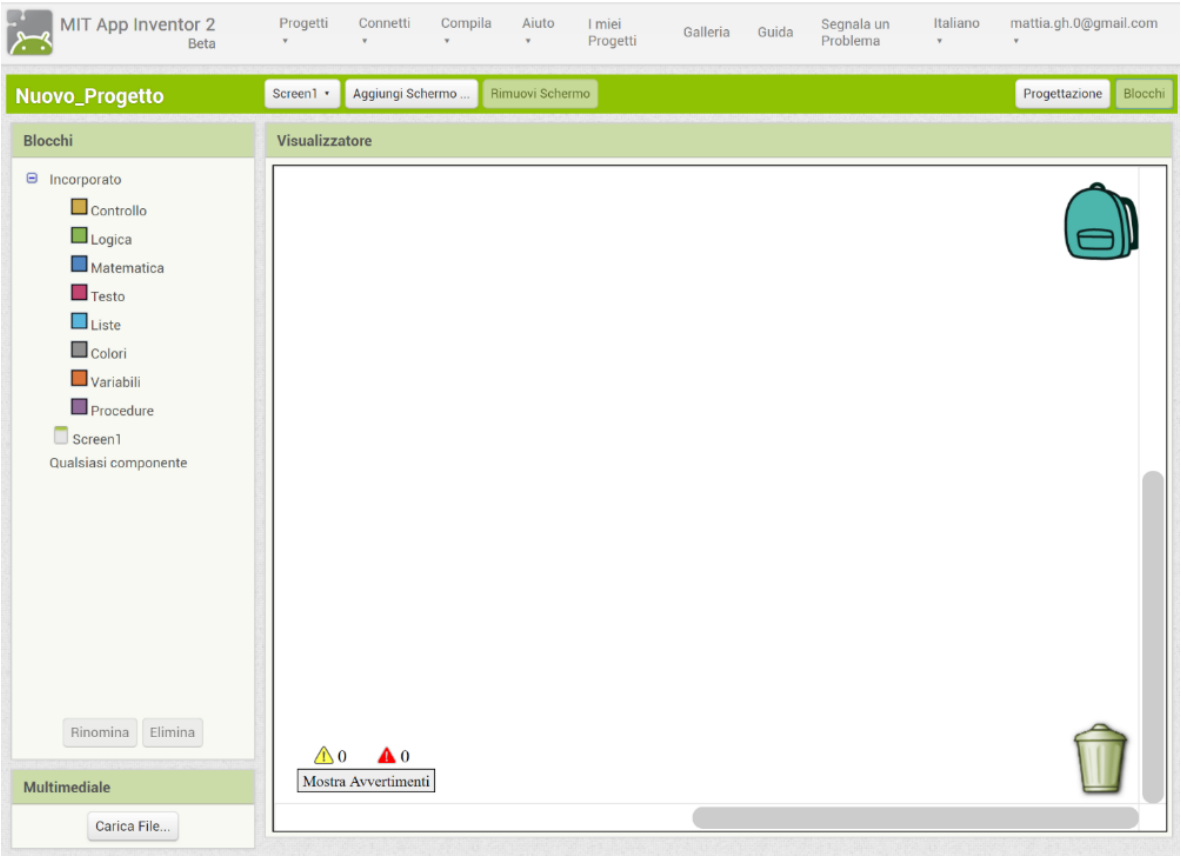

Figura 76 - MIT App Inventor schermata blocchi (vuota)

3.3 Applicazione per il primo prototipo

In fase preliminare, per creare un'applicazione che potesse essere utile a questo progetto si sono poste le seguenti domande:

1. Quali sono le informazioni che l'utente deve ricevere?

La risposta è ovvia, essendo un'applicazione pensata per un non vedente che vuole sapere se può attraversare una strada o meno, questa dovrà fornire lo stato del semaforo, ovvero se in quel momento la lampada accesa è quella rossa, quella gialla o quella verde.

2. Come fare ad ottenere questa informazione da Arduino?

Come descritto nei capitoli precedenti (*Cap. 2.5.2 e 2.7.2*) ad ogni stato (quindi ad ogni colore) viene associato un numero:

- Uno per il rosso
- Due per il giallo fisso
- Tre per il verde
- Quattro per il semaforo spento (in fase di sviluppo nel primo prototipo)
- Cinque per il semaforo lampeggiante (solo nel secondo prototipo)

È stata scelta questa soluzione perché semplice ed è univoca. Arduino associa in ogni stato un numero, l'applicazione legge questo numero ed in uscita trasmette lo stato del semaforo.

3. Come connettere l'applicazione al prototipo?

Per connettere l'applicazione ad Arduino la scelta è ricaduta sull'utilizzo di un router Wi-Fi, in questo modo è più facile gestire le connessioni con più dispositivi rispetto all'utilizzo di una connessione peer to peer (la quale probabilmente permette la connessione tra Arduino ed un solo altro dispositivo alla volta) ottenuta utilizzando *Arduino Yun*. L'applicazione per poter funzionare deve utilizzare il modulo Wi-Fi del dispositivo su cui è installata, il quale deve essere attivato manualmente dall'utente, che inoltre deve connetterlo alla rete locale del prototipo. Arduino utilizza una pagina locale, ad esempio: h*ttp://192.168.1.177*, con la quale gestisce le richieste da parte dell'applicazione e fornisce come risposta lo stato del semaforo.

4. Come interfacciarsi con un'utente non vedente?

Siccome la quasi totalità degli utenti che utilizzerà questa applicazione non può avvalersi delle facoltà visive, per poter interagire con essi, l'applicazione deve utilizzare il sintetizzatore vocale presente in ogni smartphone e tablet. Per inviare la richiesta ad Arduino, dopo aver aperto l'applicazione ed essersi connesso alla rete del semaforo, basta semplicemente premere lo schermo. Una volta ottenuto lo stato l'applicazione comunica vocalmente con diverse frasi preimpostate nel codice.

Gli sviluppi successivi dell'applicazione hanno aggiunto nuove funzionalità rispetto alla versione base in cui si comunicava solamente lo stato su richiesta dell'utente.

La *figura 77* mostra come si presenta la schermata iniziale della versione 3.3.1 dell'app.

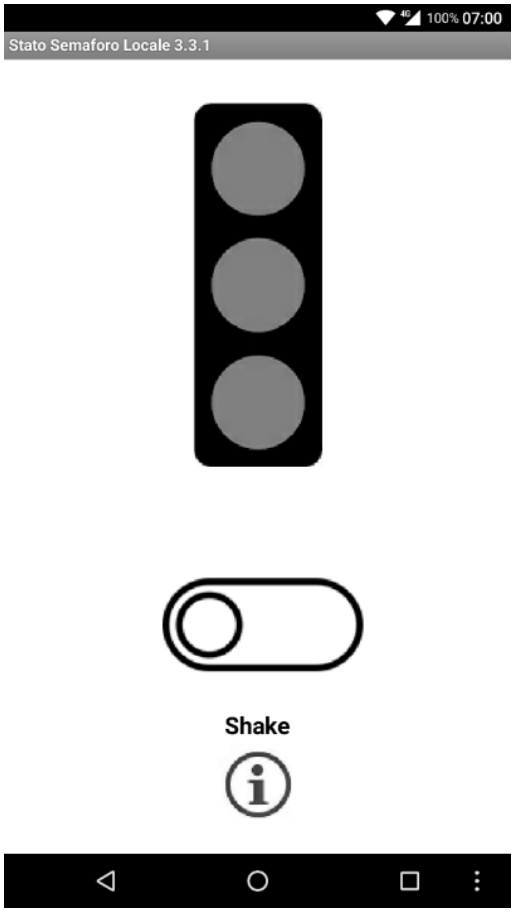

Figura 77 - Schermata applicazione: Stato Semaforo Locale 3.3.1

Si potrebbe pensare che l'interfaccia grafica sia superflua dato che l'applicazione verrà utilizzata da una persona affetta da disabilità visive, ma si è comunque creata per renderla più accattivante anche per le persone normodotate; inoltre questa può essere una buona base di partenza per lo sviluppo successivo di un'app dedicata agli automobilisti.

Le nuove funzionalità, rispetto a quella base, sono le seguenti:

- una volta connesso alla rete locale ed avviata l'applicazione, verrà comunicato a quale incrocio è installato il dispositivo di comunicazione

- lo schermo è stato suddiviso in tre aree:

    1. l'area in alto coincidente con la figura stilizzata del semaforo (*fig.79*): se premuta, il dispositivo emetterà una piccola vibrazione ed invierà la richiesta di stato ad Arduino. Una volta ottenuta la risposta accenderà una delle tre lampade a seconda dello stato in modo da avere sul proprio dispositivo una riproduzione in tempo reale del semaforo dell'attraversamento

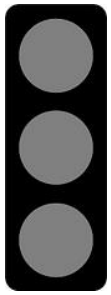

Figura 78 - Icona semaforo spento

2. l'area centrale è dedicata ad uno switch che avrà il compito di impostare la modalità con cui inviare la prima richiesta di stato:

Figura 79 - Switch SX

Per inviare la prima richiesta di stato si deve premere l'area in alto dello schermo coincidente con la figura del semaforo.

Figura 80 - Switch DX

Per inviare la richiesta di stato basta agitare una volta il dispositivo (*shake*).
Come impostazione di default si è scelta la modalità *"Shake"*, per passare alla modalità in cui si deve premere la figura del semaforo basta semplicemente scorrere il dito sullo switch da sinistra verso destra; il dispositivo emette una debole vibrazione, viene riprodotta vocalmente la parola *"Premere",* la quale compare sullo schermo sotto l'icona dello switch. Viceversa per tornare alla modalità *"Shake"* basta scorrere da destra verso sinistra sull'area dello schermo corrispondente allo switch, cosicché viene emessa una debole vibrazione, viene riprodotta vocalmente la parola *"Shake"* la quale compare sullo schermo sotto l'icona dello switch.

3. l'area in basso è stata dotata di un pulsante (*fig.81*) che fornisce informazioni su attività, locali, luoghi da visitare o punti di interesse posti nelle vicinanze dell'attraversamento pedonale. Premuta la suddetta area, il dispositivo vibra brevemente ed il sintetizzatore vocale riproduce una frase precaricata sull'applicazione che vara a seconda della rete locale semaforica a cui si è collegati.

Figura 81 - Pulsante informazioni

4. dopo aver effettuato la prima richiesta di stato, le successive vengono effettuate automaticamente ogni mezzo secondo fino a quando l'applicazione non viene terminata. In questo modo i cambiamenti di stato vengono notificati in automatico all'utente senza che esso debba ripetere manualmente la richiesta.

5. è stata aggiunta una vibrazione personalizzata a seconda dello stato del semaforo: lenta nel caso in cui si possa attraversare, una più veloce quando la lampada accesa è quella gialla ed infine nessuna vibrazione nel caso in cui non si possa attraversare.

3.3.1 Blocchi MIT App Inventor

In questo paragrafo vengono riportati tutti i blocchi utilizzati per programmare l'applicazione con le caratteristiche descritte in precedenza; di ognuno vengono spiegate le loro funzioni e l'ordine logico con cui vengono utilizzati.

La schermata di programmazione nella totalità si presenta come segue:

Figura 82 - MIT App Inventor schermata blocchi applicazione S.S.L.3.3.1

Dall'immagine precedente si può notare che, utilizzando questo programma invece che scrivere righe di codice, si ottiene una struttura ordinata e semplice, in cui è facile raggruppare in diverse zone i blocchi della stessa natura in modo da semplificare la verifica visiva da parte del programmatore.

3.3.1.1 Variabili globali

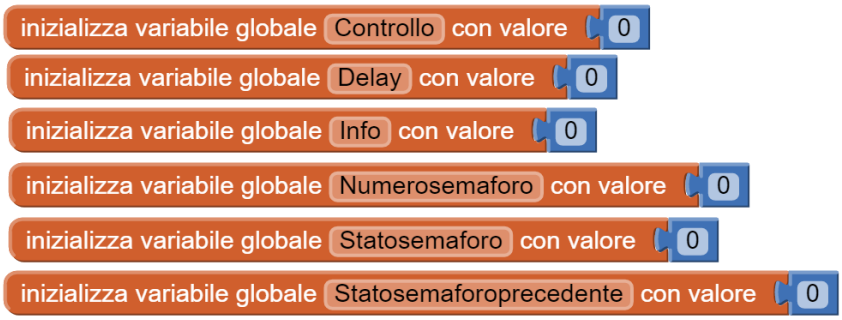

Figura 83 - MIT App Inventor blocchi variabili globali

La prima operazione svolta dall'applicazione quando viene avviata è di porre uguale a zero tutte le variabili globali.

Queste variabili devono essere dichiarate all'avvio per poter essere utilizzate durante il funzionamento ed ognuna svolte un diverso compito:

- CONTROLLO: utilizzata per evitare che vengano effettuate troppe richieste di stato alla pagina *http://192.168.1.177* prima che Arduino riesca a rispondere
- DELAY: utilizzata per creare un ritardo nelle richieste automatiche da parte dell'applicazione verso Arduino
- INFO: utilizzata per evitare che vengano effettuate troppe richieste di informazioni alla pagina *http://192.168.1.177* prima che Arduino riesca a rispondere
- NUMEROSEMAFORO: indica il numero associato a quel semaforo
- STATOSEMAFORO: indica lo stato attuale del semaforo, può assumere i valori *1*, *2* o *3* a seconda di quale lampada sia accesa, 4 se il semaforo lampeggia o 5 se il semaforo è spento (questi due valori vengono utilizzati correttamente dal secondo prototipo in poi)
- STATOSEMAFOROPRECEDENTE: indica lo stato precedente del semaforo

3.3.1.2 Azioni automatiche

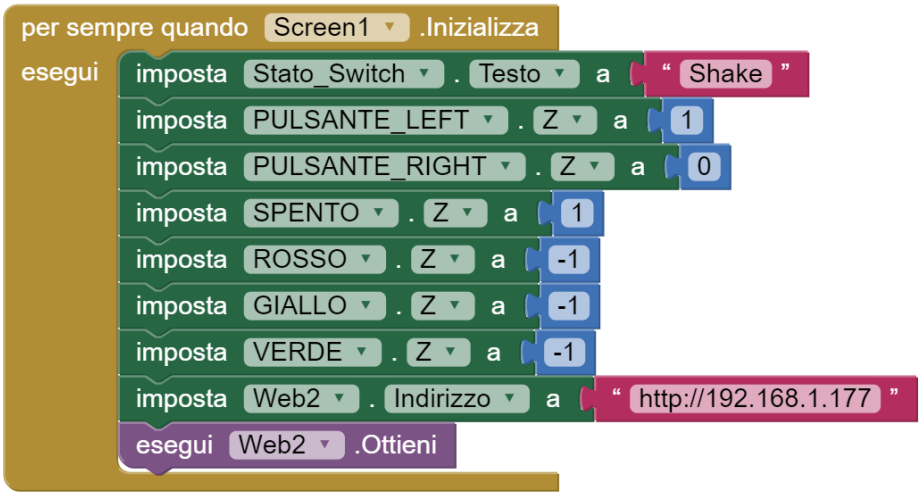

Figura 84 - MIT App Inventor blocco inizializzazione schermo

Quando si avvia l'applicazione, in automatico, vengono eseguite le seguenti operazioni:

- sullo schermo, l'etichetta *"Stato_Switch"* viene impostata con la scritta *"Shake"* per indicare che per inviare la richiesta di informazione sullo stato del semaforo si deve agitare il telefono
- lo Switch viene posizionato sulla sinistra
- la figura del semaforo mostra tutte e tre le luci spente
- al componente *"Web2"* viene impostato l'indirizzo *http://192.168.1.177*
- viene eseguita la richiesta di collegamento all'indirizzo impostato nel punto precedente

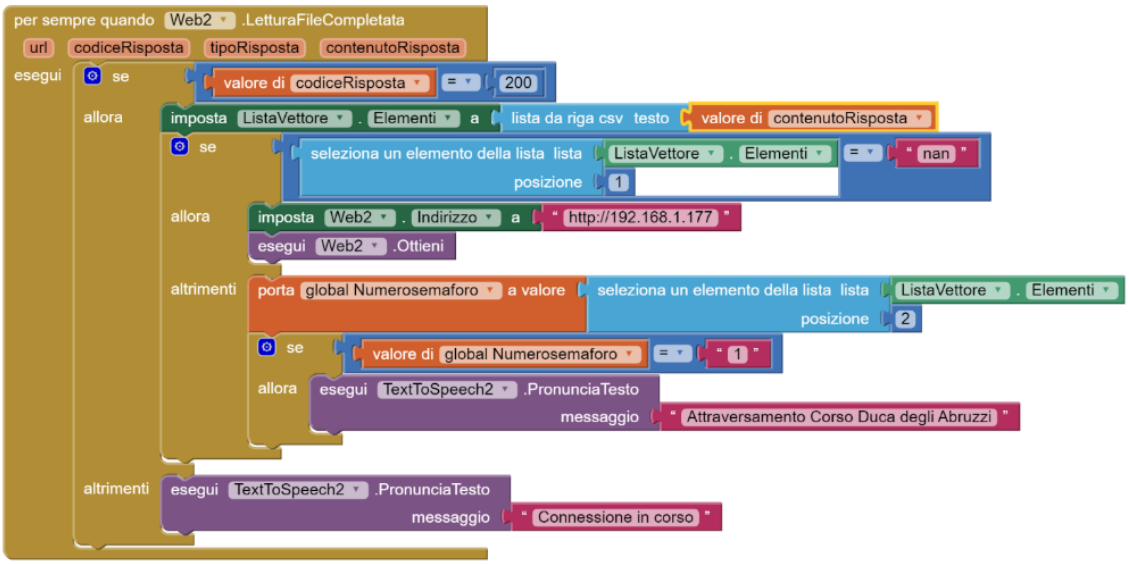

Figura 85 - MIT App Inventor blocco Web2

Dopo aver effettuato la richiesta di collegamento, il blocco *"Web2"* esegue le seguenti operazioni:

- se il codice di risposta ottenuto dall'indirizzo *http://192.168.1.177* è uguale a *200* (valore imposto dal protocollo di comunicazione web) il componente *"Listavettore"* viene "riempito" degli elementi contenuti nella riga a quell'indirizzo. In parole più semplici: all'indirizzo *http://192.168.1.177* Arduino invia due valori divisi da una virgola, il primo è lo stato del semaforo, il secondo è il numero del semaforo; se il codice di risposta è uguale a *200*, tramite questa operazione poniamo questi due valori in un elenco. Si possono utilizzare anche più di due valori, basta che questi siano separati da una virgola.

- Dopo di che viene effettuata un'altra scelta:

    - se il primo elemento dell'elenco *"ListaVettore"* è uguale a *"nan"* (not a number) significa che Arduino non ha inviato nessun valore, ma comunque riesce a comunicare con l'applicazione. In questo caso viene di nuovo inviata una richiesta all'indirizzo impostato in *"Web2"*.

    - se invece l'elemento non è uguale a *"nan"* la variabile *"Numerosemaforo"* viene posta uguale al secondo elemento dell'elenco *"ListaVettore"* e nel caso in cui questo sia uguale a uno allora verrà pronunciata la frase: *"Attraversamento Corso Duca degli Abruzzi"*.

- Nel caso in cui il codice di risposta è differente da 200 allora il dispositivo dovrà pronunciare la frase: *"Connessione in corso"*.

Lo scopo di questo blocco è quello di indicare all'utente a quale attraversamento pedonale si trova in quel momento.

3.3.1.3 Interazione con l'utente

Di seguito vengono illustrati i blocchi utilizzati dall'utente per interagire con l'applicazione di ottenere le informazioni desiderate:

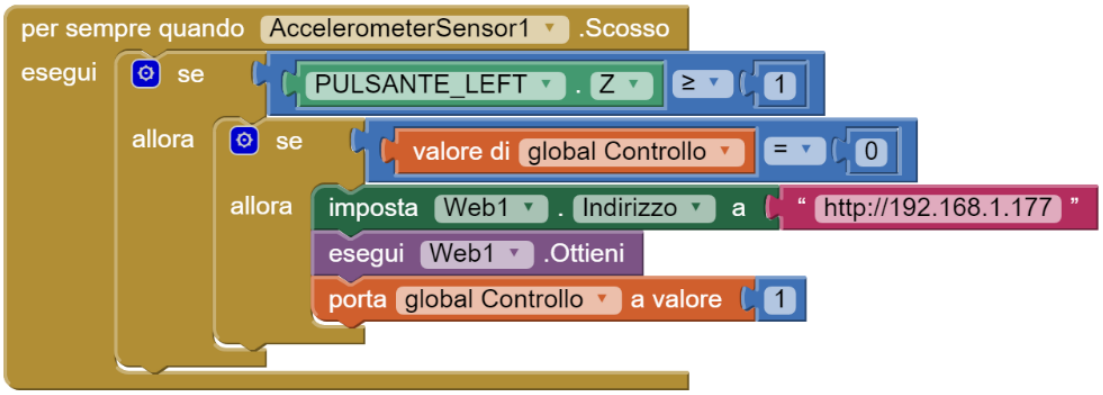

Figura 86 - MIT App Inventor blocco Accelerometro

Questo blocco utilizza il sensore di accelerazione presente in ogni dispositivo per eseguire la prima richiesta di stato ad Arduino. Per poter essere effettuata devono verificarsi contemporaneamente i seguenti punti:

- il dispositivo deve venir scosso con una certa intensità (l'intensità viene tarata in modo tale da evitare richieste accidentali)
- il pulsante di switch deve essere posizionato a sinistra, quindi il dispositivo deve essere in modalità *"Shake"*
- la variabile globale *"Controllo"* deve essere uguale a zero, cioè non devono essere già state effettuate altre richieste

Se queste discriminanti si verificano, allora il componente *"Web1"* viene impostato con l'indirizzo *http://192.168.1.177* e la richiesta di collegamento viene effettuata. Infine la variabile globale di controllo viene posta uguale ad uno.

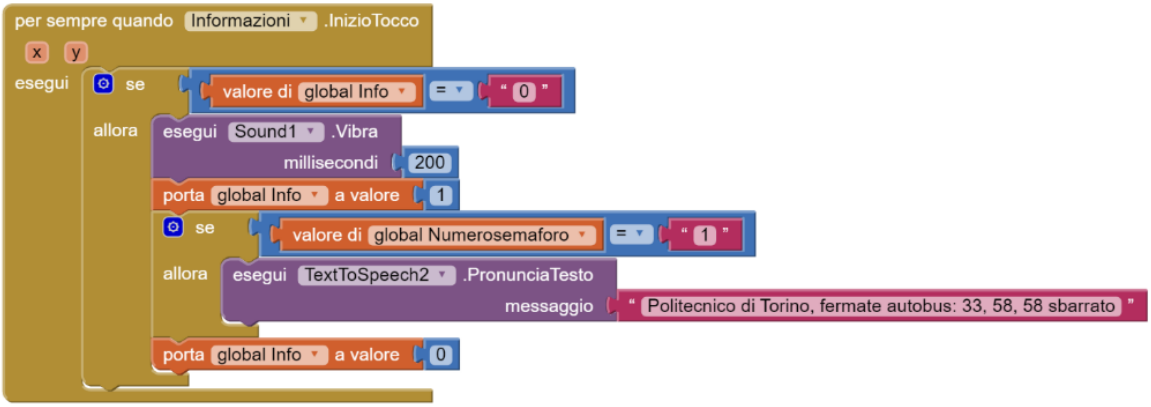

Figura 87 - MIT App Inventor blocco Informazioni

Il blocco in questione riguarda la richiesta di informazioni su attività commerciali, musei, fermate autobus, luoghi di interesse e qualsiasi cosa presente nelle vicinanze dell'incrocio o dell'attraversamento pedonale possa essere utile all'utente. Quando l'icona del pulsante viene premuta e la variabile globale *"Info"* è uguale a zero, vengono eseguite le seguenti operazioni:

- il dispositivo emette una lieve vibrazione per indicare all'utente che è stato premuta l'icona
- la variabile globale "*Info*" viene posta uguale ad uno per evitare che vengano effettuate più richieste consecutive accidentali
- viene letta la variabile *"Numerosemaforo"* ed a seconda di quale semaforo corrisponde verrà pronunciato un messaggio differente. In questo caso si è preso come riferimento l'attraversamento davanti al Politecnico di Torino a cui è stato assegnato il numero uno, quindi il messaggio predefinito è: *"Politecnico di Torino, fermate autobus: 33, 58, 58 sbarrato"*.
- infine viene riportata uguale a zero la variabile globale *"Info"*

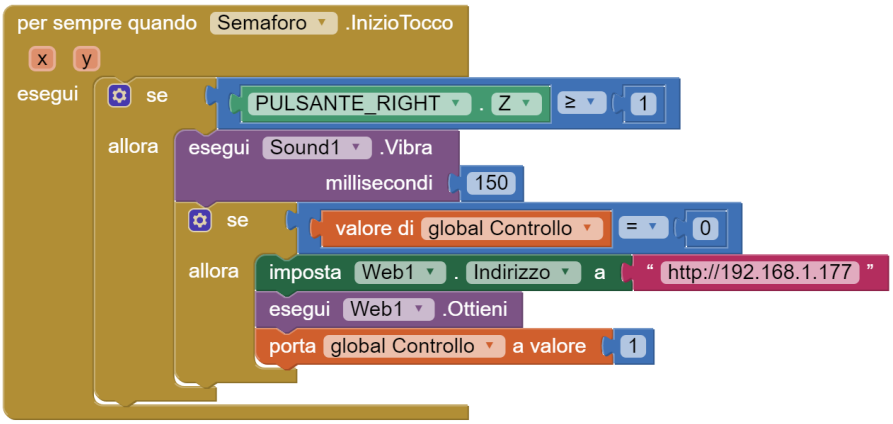

Figura 88 - MIT App Inventor blocco Semaforo

Il blocco sopra riportato descrive il metodo alternativo allo “shake” del dispositivo per effettuare la richiesta di stato del semaforo. Questa può essere effettuata se:

- viene toccata la figura in alto sullo schermo raffigurante il semaforo
- il dispositivo è in modalità *“Premere”,* cioè lo switch deve essere sulla destra

Dopo di che il dispositivo emette una breve vibrazione per dare conferma all’utente che la figura è stata premuta. Se la variabile globale *“Controllo”* è uguale a zero, cioè non sono state effettuate altre richieste di stato, allora il componente *“Web1”* viene impostato con l’indirizzo *http://192.168.1.177* e viene effettuata la richiesta di collegamento. Infine la variabile *“Controllo”* viene di nuovo posta uguale a zero.

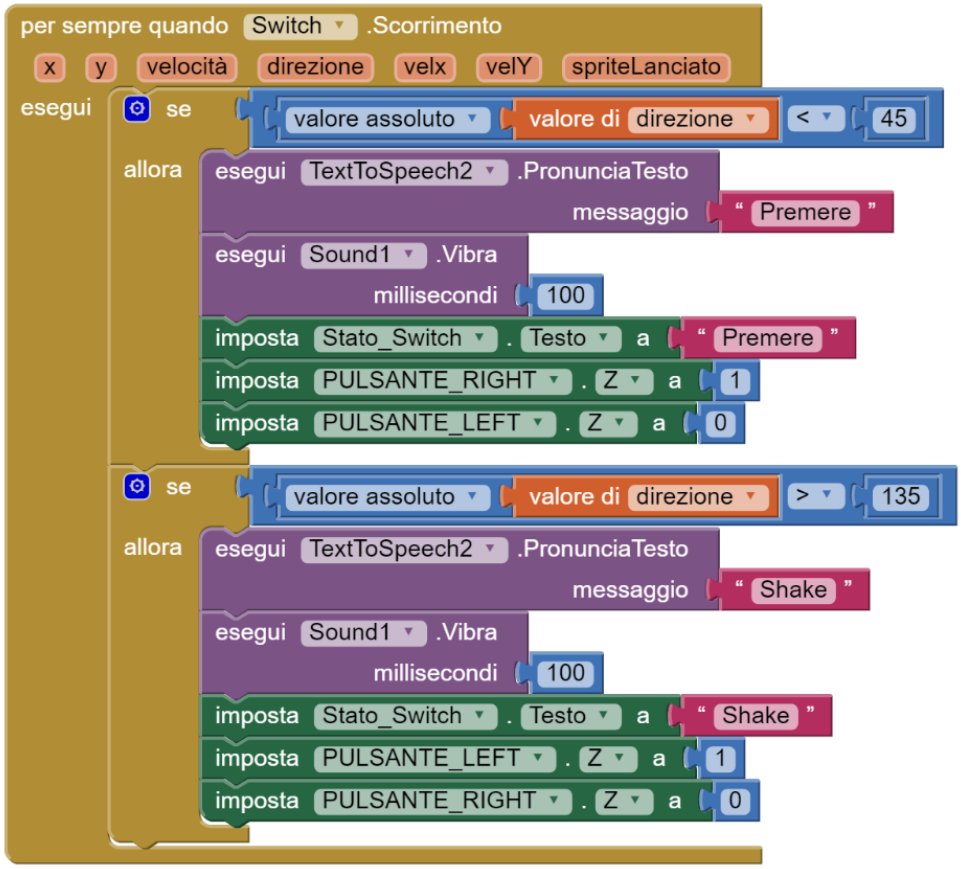

Figura 89 - MIT App Inventor blocco Switch

Quest'ultimo blocco riguarda l'interfaccia dello switch e le sue funzionalità. Quando la parte dello schermo dedicata allo switch viene premuta da sinistra verso destra vengono eseguite le seguenti azioni:

- viene pronunciata dal dispositivo la parola *"Premere"*
- viene emessa una breve vibrazione per indicare all'utente che si sta utilizzando lo switch
- compare sullo schermo la parola *"Premere"*
- l'icona dello switch cambia colore passando dal bianco al nero e la pallina bianca si sposta da sinistra a destra

In questo modo l'applicazione stabilisce che la richiesta di stato del semaforo possa avvenire solo quando viene premuta la figura del semaforo e non quando il dispositivo viene scosso.

Quando la parte dello schermo dedicata allo switch viene premuta da destra verso sinistra vengono eseguite le seguenti azioni:

- viene pronunciata dal dispositivo la parola *"Shake"*
- viene emessa una breve vibrazione per indicare all'utente che si sta utilizzando lo switch
- compare sullo schermo la parola *"Shake"*
- l'icona dello switch cambia colore passando dal nero al bianco e la pallina bianca si sposta da destra a sinistra

In questo modo l'applicazione stabilisce che la richiesta di stato del semaforo possa avvenire solo quando il dispositivo viene scosso e non quando viene premuta la figura del semaforo.

3.3.1.4 Richiesta di stato

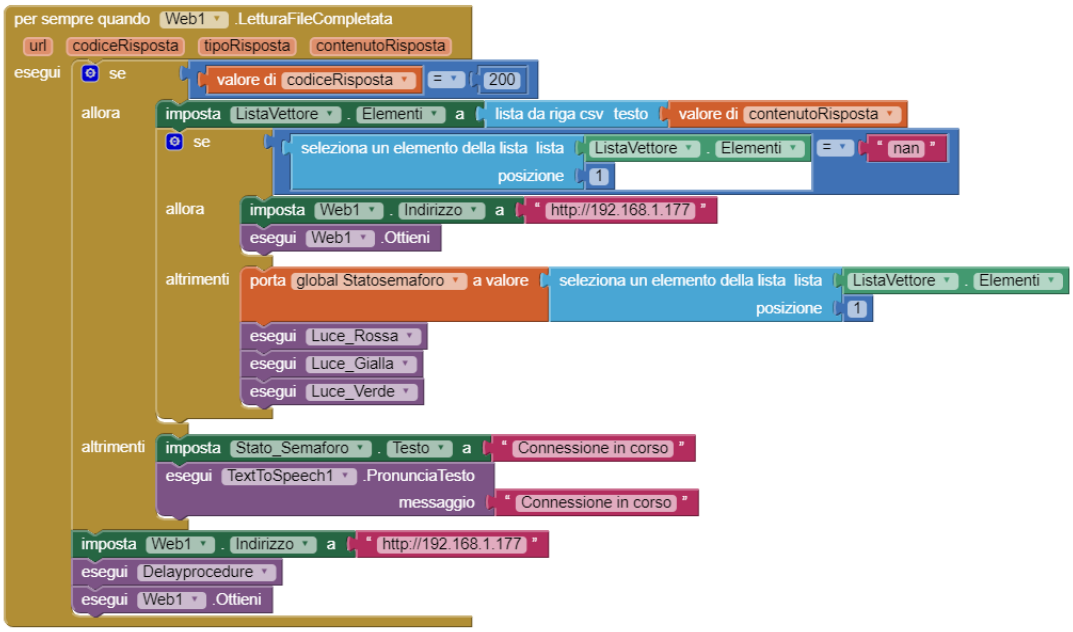

Figura 90 - MIT App Inventor blocco Web1

Il blocco "*Web1*" ha lo scopo di comunicare all'utente lo stato della lanterna a cui è collegato in quel momento.
Per poter realizzare questa operazione sono necessari i seguenti processi logici:

- se il codice di risposta ottenuto dall'indirizzo *http://192.168.1.177* è uguale a 200 (valore dato dal protocollo di comunicazione web) il componente *"Listavettore"* viene "riempito" degli elementi contenuti nella riga a quell'indirizzo. In parole più semplici: all'indirizzo *http://192.168.1.177* Arduino invia due valori divisi da una virgola, il primo è lo stato del semaforo, il secondo è il numero del semaforo; se il codice di risposta è uguale a *200*, tramite questa operazione poniamo questi due valori in un elenco. Si possono utilizzare anche più di due valori, basta che questi siano separati da una virgola.

- Dopo di che viene effettuata un'altra scelta:

    - se il primo elemento dell'elenco *"ListaVettore"* è uguale a *"nan"* significa che Arduino non ha inviato nessun valore, ma comunque riesce a comunicare con l'applicazione. In questo caso viene di nuovo inviata una richiesta all'indirizzo impostato in *"Web1"*

    - Se invece l'elemento non è uguale a *"nan"* la variabile *"Statosemaforo"* viene posta uguale al primo elemento dell'elenco *"ListaVettore"*. Dopo di che vengono eseguite in sequenza le procedure denominate: *"Luce_Rossa"*, *"Luce_Gialla"* e "*Luce_Verde"*; queste verranno spiegate nel dettaglio in seguito

- Nel caso in cui il codice di risposta è differente da *200* allora il dispositivo dovrà pronunciare la frase: *"Connessione in corso"*

Infine si reimposta nuovamente l'indirizzo del componente *"Web1"* (*http://192.168.1.177*), viene richiamata la procedura *"Delayprocedure"* e si effettua la richiesta all'indirizzo specificato in precedenza.

3.3.1.5 Procedure richiamate

Figura 91 - MIT App Inventor blocco Delayprocedure

Il blocco in questione permette di creare un ritardo temporale tra le richieste di stato automatiche. Quando la procedura viene richiamata, per prima cosa la variabile globale *"Delay"* viene impostata con l'ora del sistema, questo è possibile utilizzando il componente nascosto *"Clock"* il quale fornisce l'ora del telefono, un timer e permette i calcoli sul tempo. Dopo di che viene aggiunto un intervallo di *500ms* al valore della variabile *Delay*. Si è scelto questo lasso di tempo perché si ritiene accettabile mezzo secondo di ritardo tra due richieste; soprattutto per evitare che questo tipo di operazione vada ad inficiare il corretto funzionamento di Arduino. Infine si confronta l'ora del sistema rispetto a questo nuovo valore (*Delay+500ms*): finché l'ora del dispositivo non sarà maggiore l'applicazione dovrà attendere senza fare nulla.

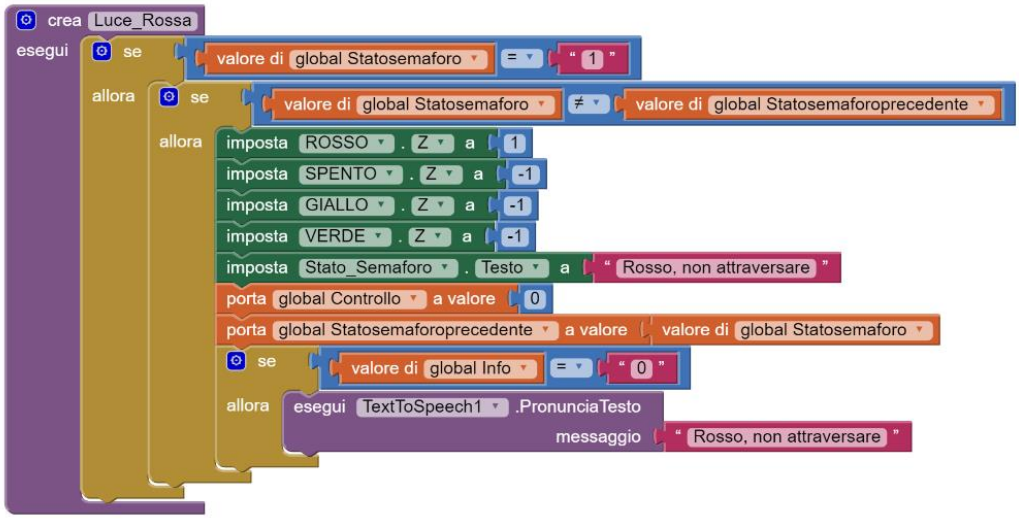

Figura 92 - MIT App Inventor blocco Luce_Rossa

Questa procedura viene richiamata tutte le volte che viene effettuata una richiesta di stato da parte dell'applicazione. Per prima cosa si verifica che la variabile globale *"Statosemaforo"* sia uguale ad **uno** e che sia differente rispetto alla variabile *"Statosemaforoprecedente"*; se queste due fasi danno esito negativo il programma passera alle fasi successive del blocco *"Web1"*. Se l'esito è positivo, significa che in quel momento il semaforo ha la lampada rossa accesa e o è avvenuto il cambio di stato o che è stata eseguita la prima richiesta; in questo caso le operazioni svolte dal programma sono le seguenti:

- la lanterna semaforica stilizzata sullo schermo viene impostata con la luce rossa accesa
- viene visualizzata la scritta *"Rosso, non attraversare"*
- la variabile globale *"Controllo"* viene impostata a zero
- la variabile globale *"Statosemaforoprecedente"* viene posta uguale alla variabile *"Statosemaforo"*, in questo modo non verranno ripetute queste operazioni se non è la prima richiesta o non c'è un cambio di stato
- se la variabile globale *"Info"* è uguale a zero, cioè non sono state effettuate richieste di informazioni sulle attività o sui luoghi negli intorni del semaforo (blocco *"Web2"*), allora il dispositivo riprodurrà vocalmente il testo: *"Rosso, non attraversare"*

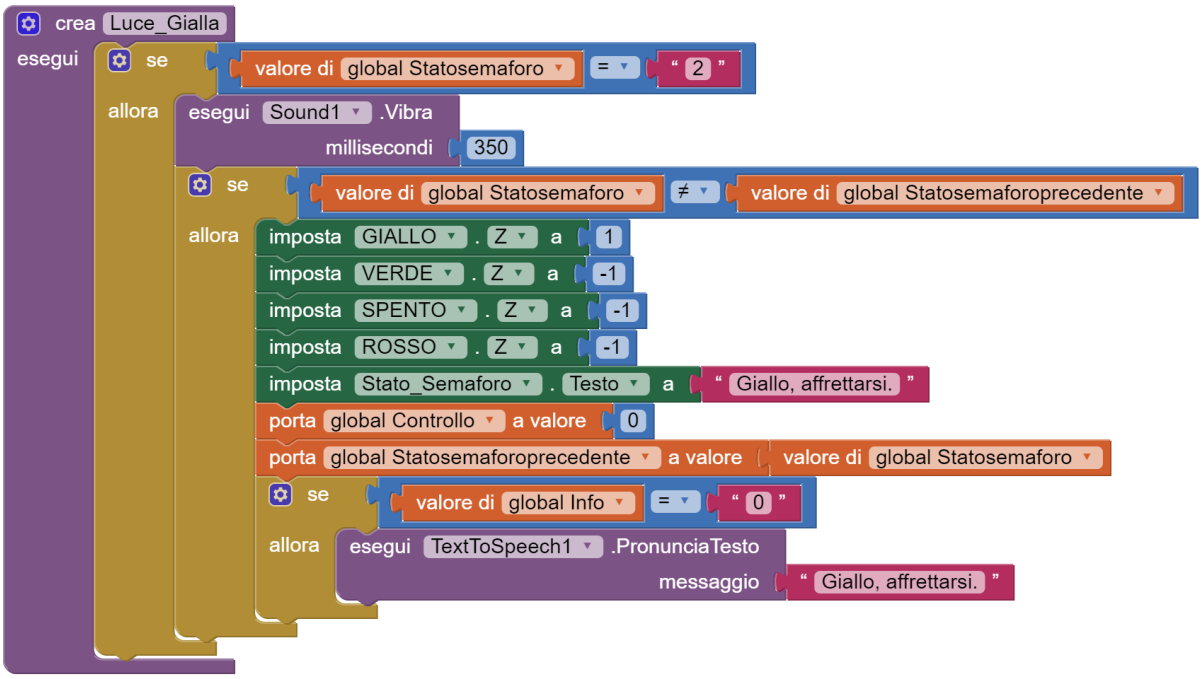

Figura 93 - MIT App Inventor blocco Luce_Gialla

Questa viene richiamata ogni volta che è eseguita una richiesta di stato da parte dell'applicazione. Esegue le stesse operazioni della procedura *"Luce_Rossa"*, ma presenta le seguenti differenze:

- la variabile globale *"Statosemaforo"* deve essere uguale a **due**
- nel caso in cui il punto prima sia corretto emette una vibrazione della durata di *350ms*
- la lanterna semaforica stilizzata viene impostata con la luce gialla accesa
- il testo visualizzato sullo schermo è: *"Giallo, affrettarsi"*
- nel caso in cui la variabile "Info" sia uguale a zero verrà riprodotta vocalmente la frase: "*Giallo, affrettarsi*"

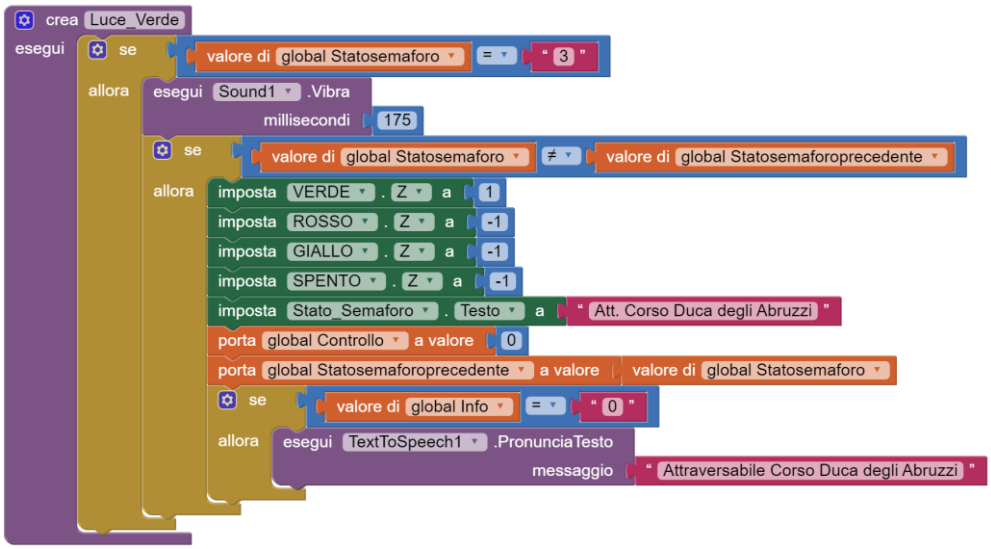

Figura 94 - MIT App Inventor blocco Luce_Verde

Questa viene richiamata ogni volta che è eseguita una richiesta di stato da parte dell'applicazione. Esegue le stesse operazioni delle procedure *"Luce_Rossa"* e *"Luce_Gialla"*, ma presenta le seguenti differenze:

- la variabile globale *"Statosemaforo"* deve essere uguale a tre
- nel caso in cui il punto prima sia vero emette una vibrazione della durata di 175ms
- la lanterna semaforica stilizzata viene impostata con la luce verde accesa
- il testo visualizzato sullo schermo è: *"Att. Corso Duca degli Abruzzi"*
- nel caso in cui la variabile "Info" sia uguale a zero verrà riprodotta vocalmente la frase: *"Attraversabile Corso Duca degli Abruzzi"*

3.4 Applicazione di geolocalizzazione

Un ulteriore sviluppo del progetto prevede di ricavare le coordinate geografiche per localizzare l'utente e poter sfruttare questi dati per diversi fattori:

- calcolare la distanza dall'incrocio o dall'attraversamento pedonale
- indicare all'utente in quale via, corso o piazza si trova
- determinare la direzione che l'utente sta seguendo ed indicargli quale è l'incrocio successivo sul suo percorso

Ovviamente questi dati possono essere utilizzati sia per l'applicazione dedicata ai non vedenti che per quella dedicata agli automobilisti. Come prima prova si è realizzata un'applicazione molto semplice con le seguenti caratteristiche:

- pulsante per effettuare la richiesta delle coordinate
- informazioni sulla latitudine
- informazioni sulla longitudine
- informazioni sull'altitudine
- informazioni sull'indirizzo più vicino rispetto a quelle coordinate, sia in modo scritto che vocalmente

L'applicazione è stata realizzata dunque utilizzando *MIT App Inventor*, ma presenta molti limiti dovuti al fatto che essendo un programma per neofiti, permette di realizzare solo app semplici. Se si volesse migliorare la qualità dell'applicazione si dovrebbero utilizzare risorse più adatte, ma più complicate e di non facile comprensione a chi non è nel settore. I limiti che questa applicazione sono i seguenti:

- difficoltà nel reperire le coordinate del dispositivo se questo non è in movimento
- lentezza nel trovare le coordinate

Come riferimento per un confronto, si sono prese in considerazioni altre applicazioni che permettono di localizzare i dispositivi su cui sono installate ed a differenza della nostra non presentano questi problemi. Ovviamente la precisione con cui si ottengono le coordinate dipende dal tipo di sensore installato sul dispositivo, più sarà preciso e quindi costoso e più si ridurranno gli errori.

Una volta risolti questi errori si potranno fondere le due applicazioni in una e sfruttare al meglio le qualità di una e dell'altra in modo da fornire a chi la utilizza tutte le informazioni necessarie per compiere in sicurezza l'attraversamento della strada.

L’interfaccia per ora è molto minimale:

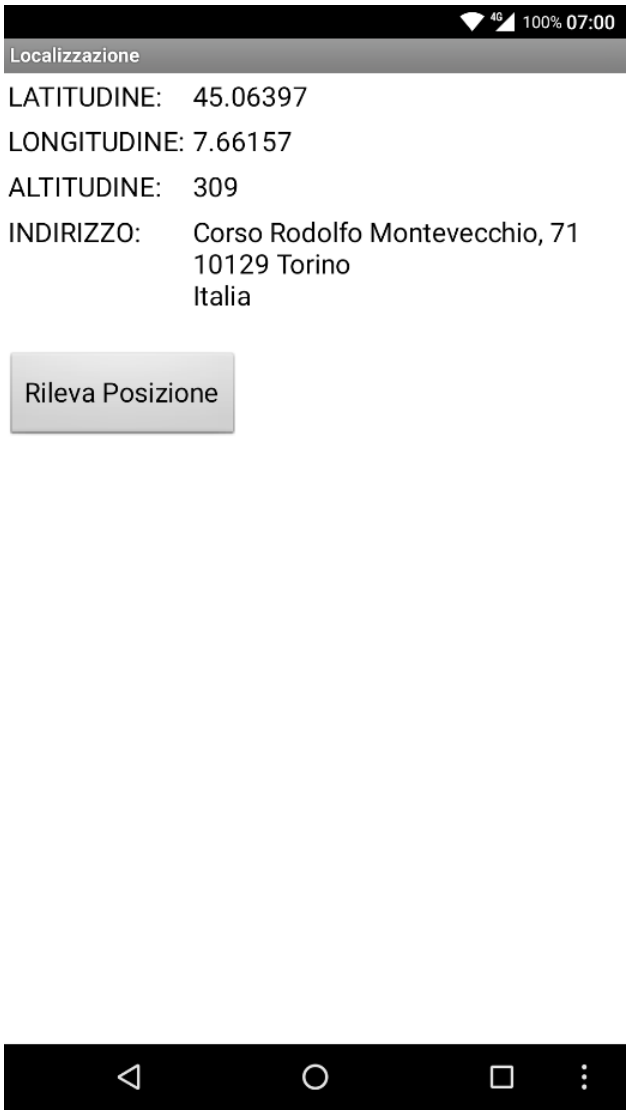

Figura 95 - Schermata applicazione: Localizzazione

Si può notare che la schermata presenta solamente un pulsante con cui si può effettuare la richiesta e poche righe dove vengono riportate le coordinate della latitudine, longitudine, altitudine e l’indirizzo più vicino.

3.4.1 Blocchi MIT App Inventor

Di seguito vengono riportati i blocchi utilizzati per la programmazione dell'applicazione con le caratteristiche descritte in precedenza.

3.4.1.1 Interazione con l'utente

Figura 96 - MIT App Inventor blocco PulsanteRilevaposizione

Quando viene premuto il pulsante si abilita l'utilizzo da parte dell'applicazione del sensore di localizzazione e sullo schermo compare la scritta: *"Attendere che la posizione venga determinata…"*.

3.4.1.2 Azioni automatiche

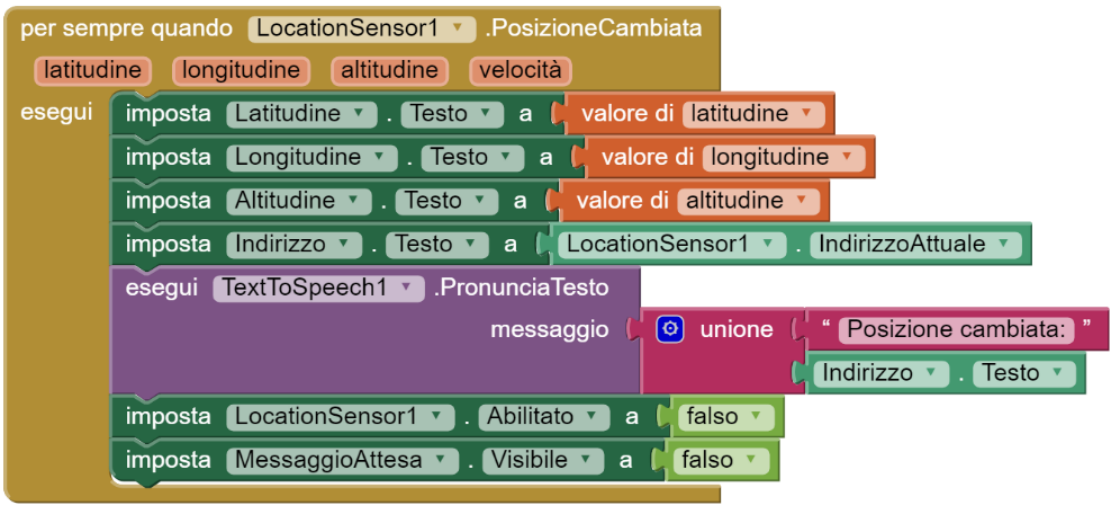

Figura 97 - MIT App Inventor blocco LocationSensor1

Quando l'applicazione rileva un cambio di posizione sullo schermo compaiono le coordinate ottenute dal sensore *GPS*; il dispositivo riprodurrà vocalmente la frase: *"Posizione cambiata:"* ed andrà a completarla pronunciando l'indirizzo più vicino alla posizione in cui si trova. Infine il messaggio di attesa viene nascosto e viene disabilitato l'utilizzo del sensore *GPS* da parte dell'applicazione.

3.5 Applicazione per il secondo prototipo

Assieme allo sviluppo del secondo prototipo, sono state apportate delle piccole modifiche anche all'applicazione Android che fornisce all'utente le informazioni ottenute da Arduino.

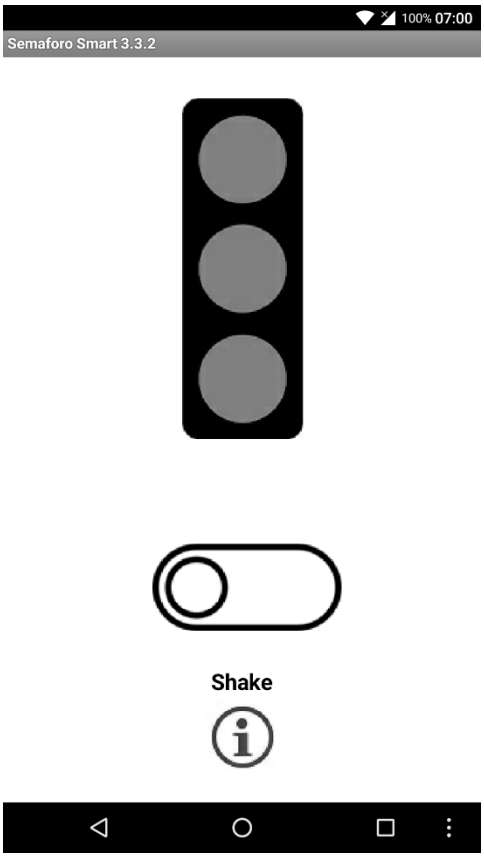

Figura 98 - Schermata applicazione: Semaforo Smart

Come si può vedere nell'immagine precedente, la schermata di interfaccia con l'utente è rimasta quasi totalmente inalterata rispetto alla prima applicazione, l'unica differenza è il nome in alto a sinistra: "*Semaforo Smart 3.3.2*" e non più "*Stato Semaforo Locale 3.3.1*".

Da un consiglio dell'esponente dell'associazione non vedenti, dopo i primi test in laboratorio, il nome dell'applicazione è stato modificato per semplificarne la ricerca nel menù del dispositivo su cui è installata. Il nome della prima app è: "*S.S.L.3.3.1*" acronimo di "Stato Semaforo Locale 3.3.1" dove 3.3.1 è la versione dell'applicazione; la seconda in vecce è stata rinominata semplicemente "Semaforo Smart".

Le altre novità riguardano i blocchi con cui è stata programmata l'applicazione.

3.5.1 Blocchi MIT App Inventor

In questo paragrafo vengono analizzate le modifiche eseguite sui blocchi della nuova applicazione.

La schermata di programmazione nella totalità si presenta come segue:

Figura 99 - MIT App Inventor schermata blocchi applicazione Semaforo Smart

Rispetto alla versione precedente sono stati aggiunte le funzioni "***Spento***" e "***Giallo_lamp***", mentre i blocchi "***Web1***" e "***Web2***" sono stati modificati.

3.5.1.1 Azioni automatiche

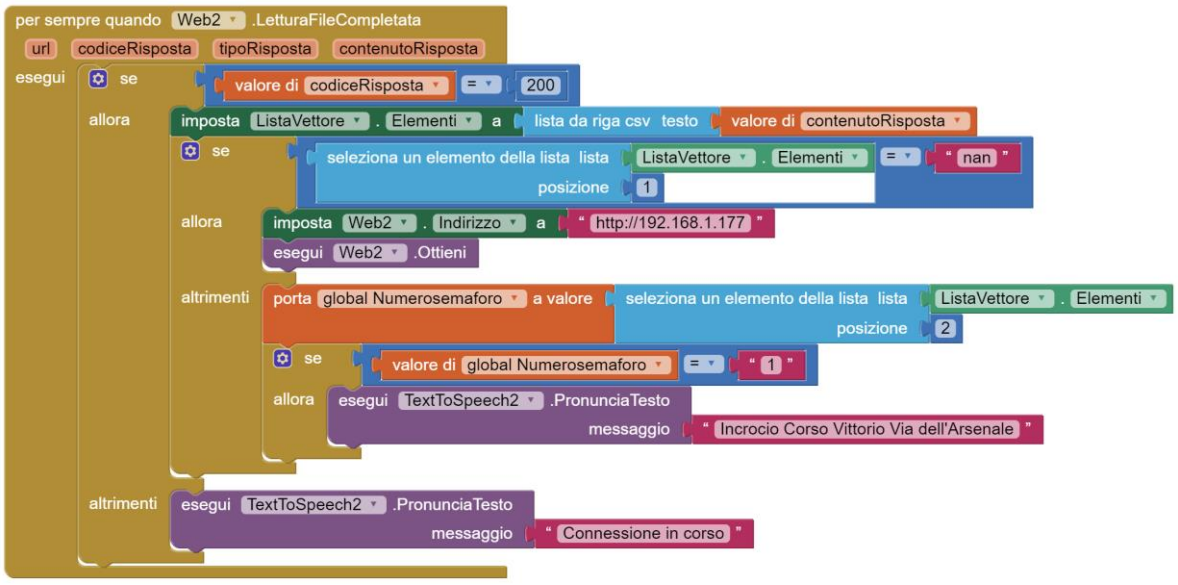

Figura 100 - MIT App Inventor blocco Web2 Semaforo Smart

Il blocco "*Web2*" svolge le stesse funzioni di quello omonimo descritto nel *Capitolo 3.2.1.2* con la differenza che invece di pronunciare *"Attraversamento Corso Duca degli Abruzzi"* riprodurrà la frase: "Incrocio Corso Vittorio Via dell'Arsenale".

3.5.1.2 Richiesta di stato

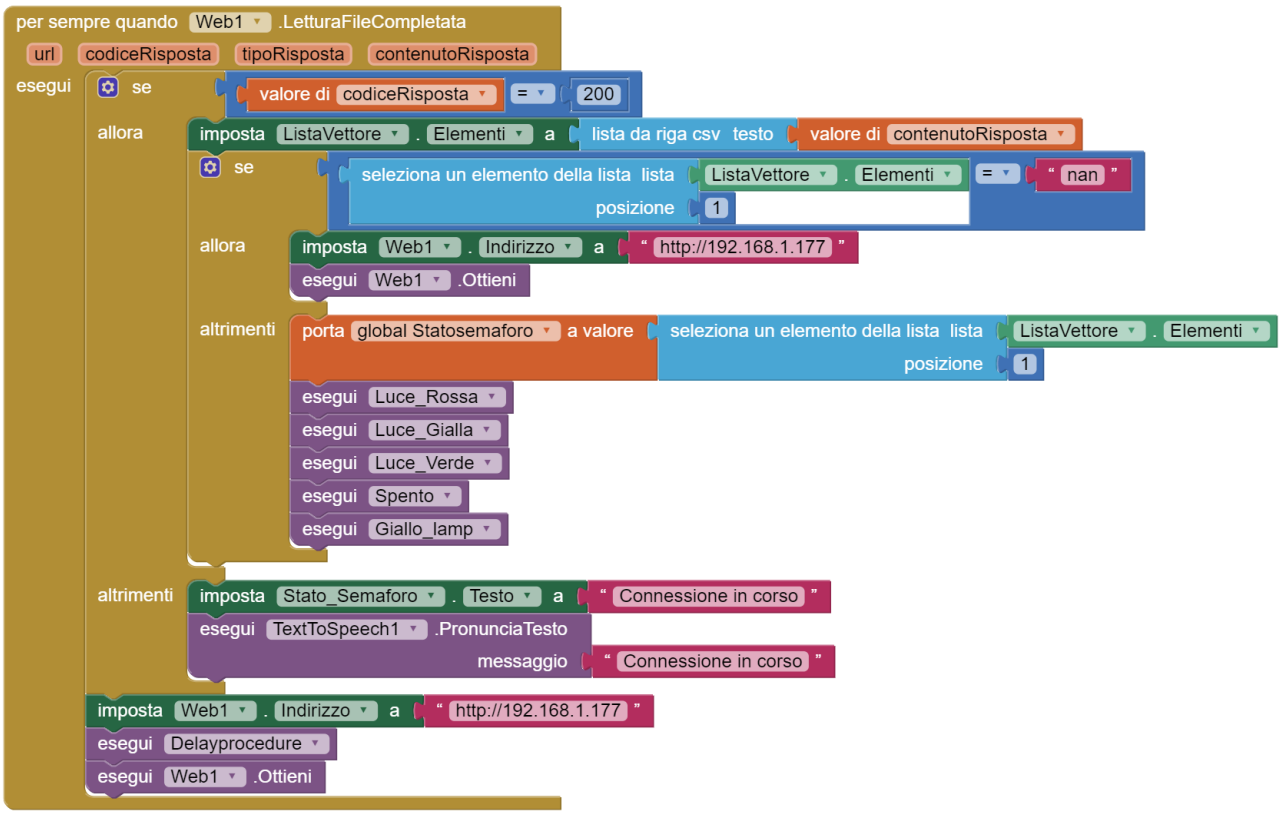

Figura 101 - MIT App Inventor blocco Web1 Semaforo Smart

Anche il blocco "*Web2*" svolge le stesse funzioni di quello omonimo presente nell'applicazione precedente, con la differenza che sono state aggiunte le procedure "*Spento*" e "*Giallo_lamp*" prima non presenti. Queste due vengono richiamate dopo quelle relative alle luci: "*Luce_Rossa*", "*LuceGialla*" e "*Luce_Verde*".

3.5.1.3 Procedure richiamate

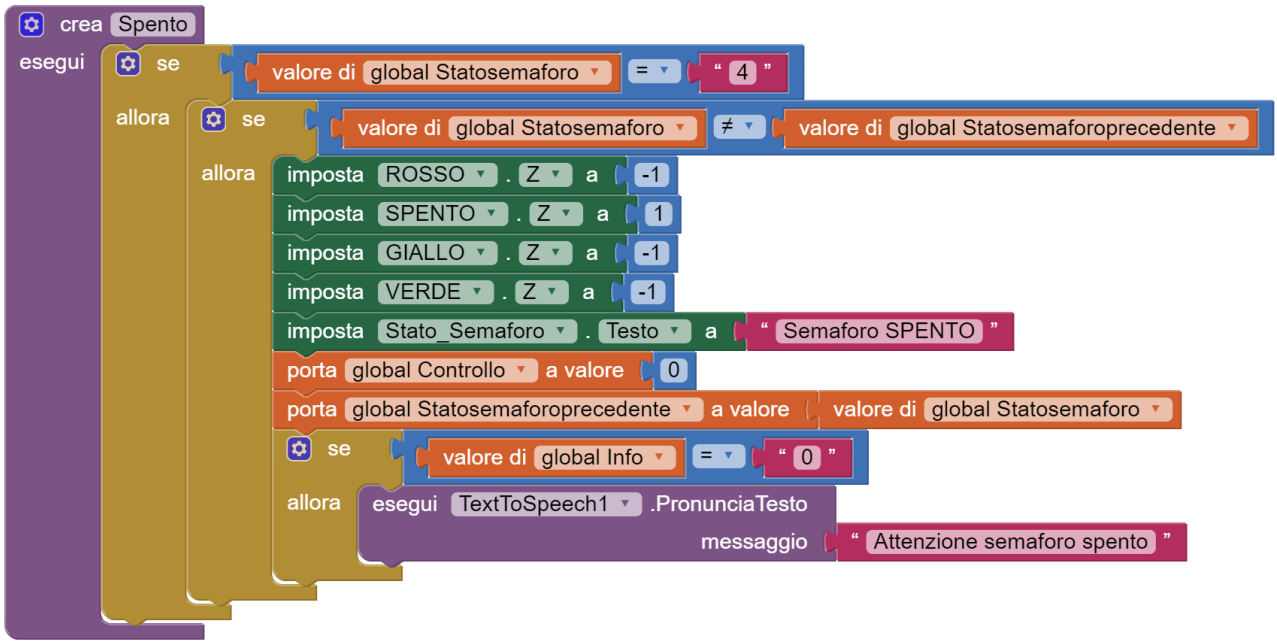

Figura 102 - MIT App Inventor blocco Spento

Questa procedura viene richiamata tutte le volte che viene effettuata una richiesta di stato da parte dell'applicazione. Per prima cosa si verifica che la variabile globale *"Statosemaforo"* sia uguale ad **quattro** e che sia differente rispetto alla variabile *"Statosemaforoprecedente"*; se queste due fasi danno esito negativo il programma passerà alle fasi successive del blocco *"Web1"*. Se l'esito è positivo, significa che in quel momento il semaforo ha tutte le lampade spente e o è avvenuto il cambio di stato o che è stata eseguita la prima richiesta; in questo caso le operazioni svolte dal programma sono le seguenti:

- la lanterna semaforica stilizzata sullo schermo viene impostata con le luci spente
- viene visualizzata la scritta *"Semaforo SPENTO"*
- la variabile globale *"Controllo"* viene impostata a zero
- la variabile globale *"Statosemaforoprecedente"* viene posta uguale alla variabile *"Statosemaforo"* in questo modo non verranno ripetute queste operazioni se non è la prima richiesta o non c'è un cambio di stato
- se la variabile globale *"Info"* è uguale a zero, cioè non sono state effettuate richieste di informazioni sulle attività o sui luoghi negli intorni del semaforo (blocco *"Web2"*) allora il dispositivo riprodurrà vocalmente il testo: *"Attenzione semaforo spento"*

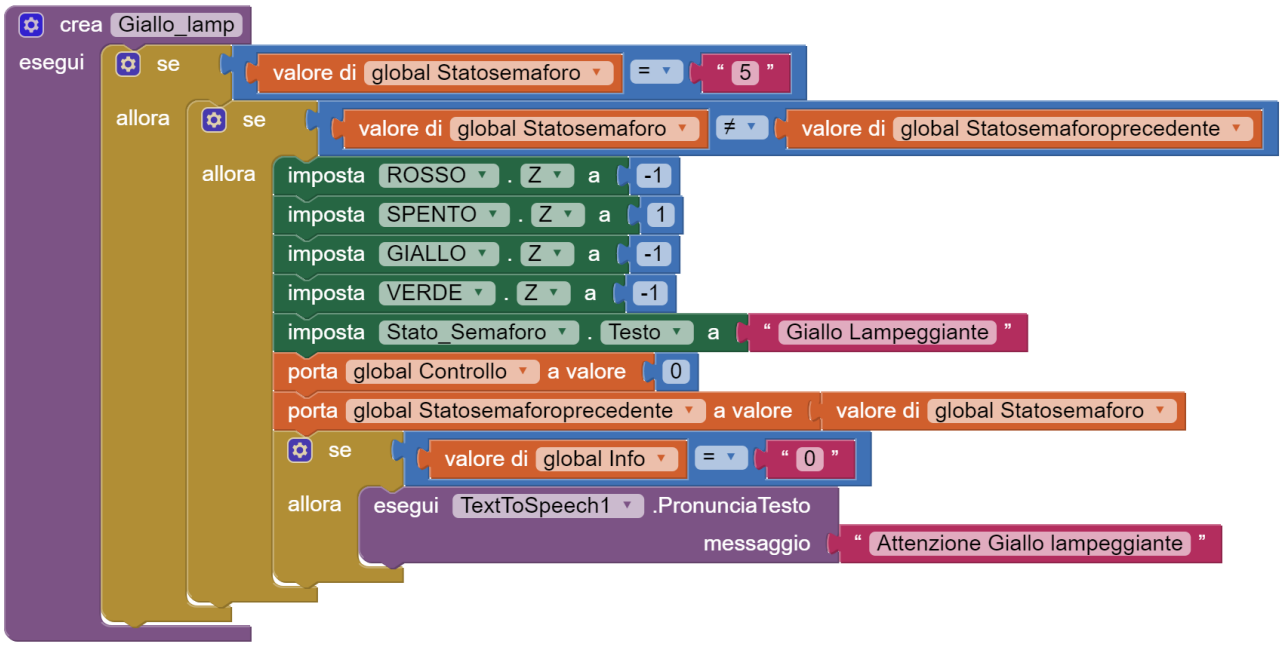

Figura 103 - MIT App Inventor blocco Giallo_lamp

Questa viene richiamata ogni volta che è eseguita una richiesta di stato da parte dell'applicazione. Esegue le stesse operazioni della procedura *"Spento"*, ma presenta le seguenti differenze:

- la variabile globale *"Statosemaforo"* deve essere uguale a **cinque**
- il testo visualizzato sullo schermo è: *"Giallo Lampeggiante"*
- nel caso in cui la variabile "Info" sia uguale a zero verrà riprodotta vocalmente la frase: "Attenzione *Giallo lampeggiante*"

Con quest'ultimo blocco si conclude il confronto tra le applicazioni: "***S.S.L.3.3.1***" e "***Semaforo Smart***".

Cap.4: Applicazioni future

Gli sviluppi successi al secondo prototipo con relativa applicazione verranno trattati più nel dettaglio in elaborati successivi, di seguito sono riportati solo degli accenni su come si è sviluppato e come si pensa di sviluppare in futuro il progetto.

4.1 Generalità

Per i motivi elencati precedentemente (*Capitolo 2.8.1*), il nuovo dispositivo non verrà installato all'interno della centralina, ma bensì all'interno della lanterna semaforica di cui si vuole ricevere il segnale e dovrà avere le seguenti caratteristiche:

- ridotte dimensioni
- autoalimentazione dalle tre lampade della lanterna
- utilizzo della tecnologia Bluetooth Low Energy
- applicazione per smartphone Android compatibile con questa nuova tecnologia e di facile utilizzo

4.2 Componenti e assemblaggio dell'hardware

Alla parte circuitale di misura sono state apportate le seguenti modifiche:

- i tre trasformatori con taglia *0,35VA* sono stati sostituiti da altrettanti che presentano una potenza nominale di *7,5VA* e due secondari la cui tensione nominale è *6V*. La scelta di utilizzare questo tipo di trasformatori è dovuta alla necessità di realizzare l'autoalimentazione, infatti il primo secondario viene collegato al circuito di misura, mentre il secondo andrà ad alimentare Arduino
- un ponte a diodi (come quelli utilizzati nei circuiti di misura) raddrizza la tensione di alimentazione fornita da uno dei due secondari perché Arduino deve essere alimentato con una tensione continua
- per evitare che durante i passaggi da uno stato all'altro del semaforo venga a mancare l'alimentazione ad Arduino, i tre circuiti in uscita dal ponte a diodi vengono collegati ad un condensatore elettrolitico con capacità pari a *10000µF*. In questo modo oltre a realizzare l'autonomia necessaria, la tensione viene filtrata passando da una doppia semionda positiva ad una tensione pressoché continua
- sono stati aggiunti tre led colorati (rosso, giallo e verde) che vengono accesi da Arduino a seconda dello stato in cui si trova il semaforo

- è stato aggiunto un pulsante per effettuare più facilmente le operazioni di reset di Arduino
- è stato aggiunto un interruttore che apre il circuito di alimentazione di Arduino permettendo così di spegnere il dispositivo senza scollegare l'alimentazione dell'intero apparato

- per la parte di elaborazione è stata scelta la scheda ***Arduino Pro Mini*** che grazie alle ridotte dimensioni, all'elevato numero di pin digitali ed analogici messi a disposizione, il tutto correlato ad un discreto processore permette di svolgere adeguatamente le operazioni assegnate

- per quanto riguarda la comunicazione si è scelto di abbandonare la connessione *Wi-Fi* e di utilizzare la ***scheda Bluetooth 4.0 HM10 BLE*** compatibile con Arduino Pro Mini

Il prototipo completo costruito in laboratorio è il seguente:

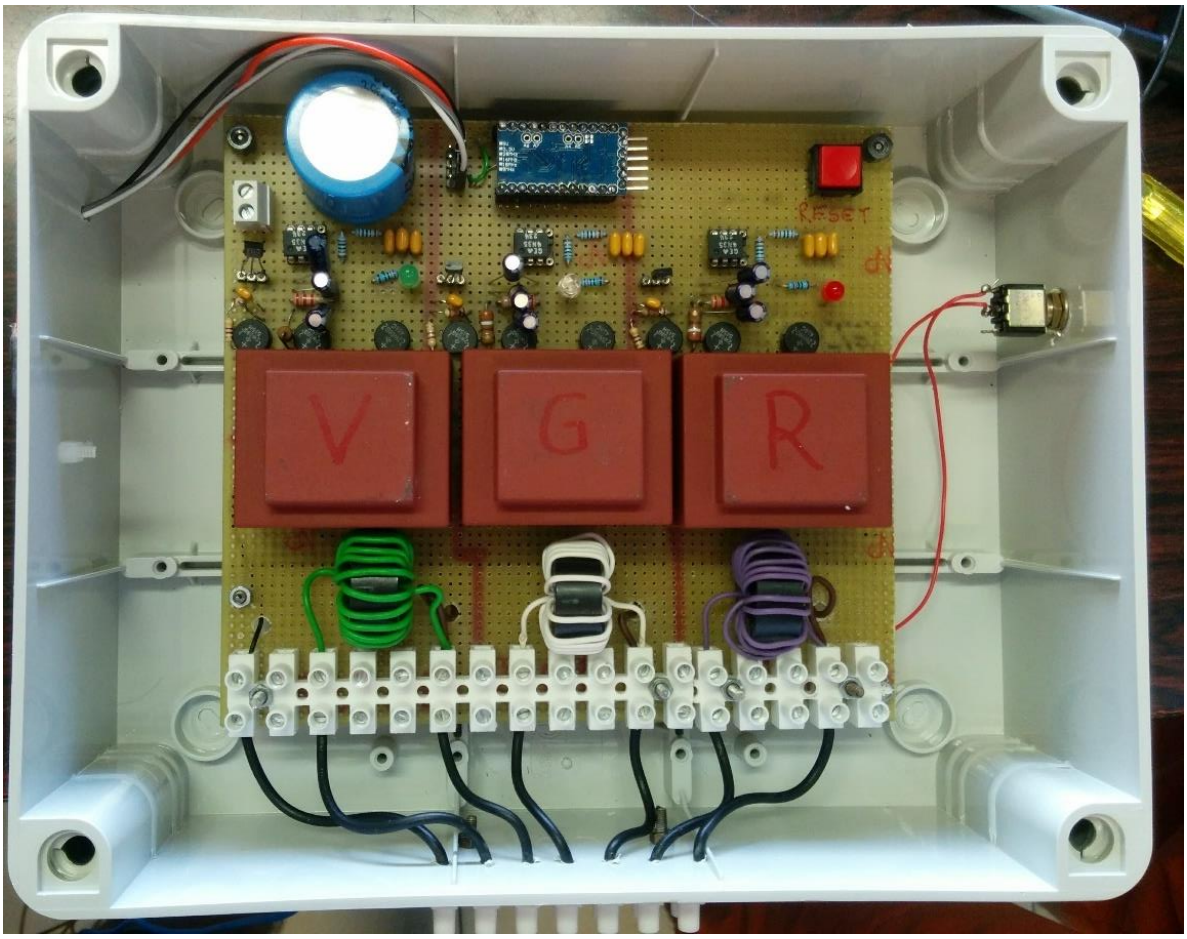

Figura 104 - Prototipo BLE

4.3 Scrittura del software

Per la stesura completa del codice di Arduino si rinvia agli allegati; di seguito è riportato il flow chart ed una breve descrizione del programma.

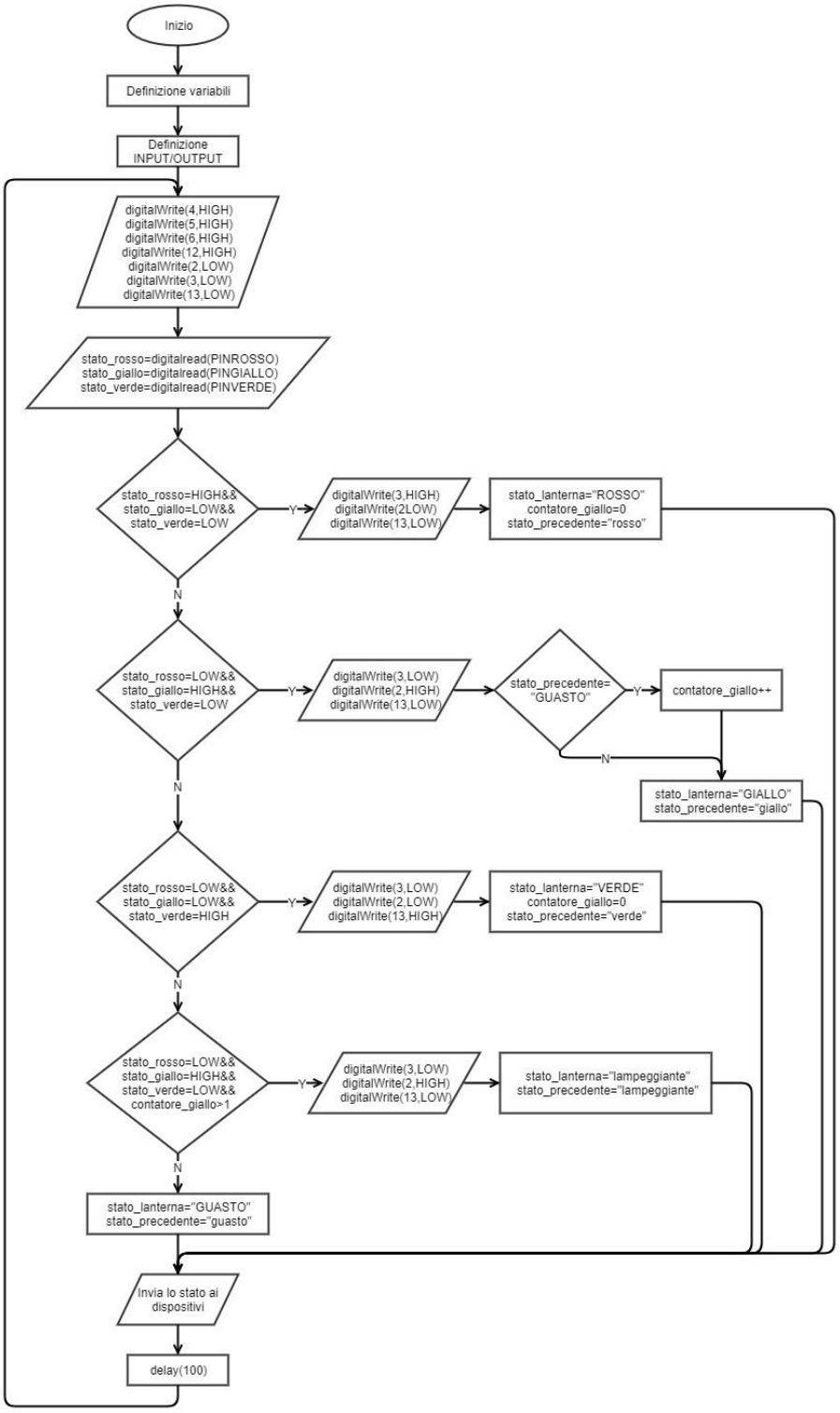

Figura 105 - Diagramma di flusso prototipo BLE

Nella prima parte del codice vengono definiti la libreria ed i pin di comunicazione *RX* e *TX* (rispettivamente i PIN *10* e *11*) utilizzati da Arduino per poter usufruire del modulo *BLE* per l'invio dei dati agli smartphone Android; infine vengono dichiarate le variabili utilizzate nel programma. Nel *setup()* vengono impostati i vari pin di *OUTPUT:*

- il *2*, il *3* ed il *13* sono collegati ai rispettivi circuiti di alimentazione dei led
- il *12* al modulo BLE
- il *4*, il *5* ed il *6* ai circuiti di campionamento dei segnali digitali dai vari comparatori

Nel *loop()* Arduino legge i valori digitali ottenuti dai tre comparatori, determina in quale stato si trova la lanterna ed accende il corrispettivo led sul prototipo. Infine invia tramite il modulo *BLE* l'informazione dello stato e la localizzazione del semaforo agli smartphone muniti dell'applicazione. Tutte le operazioni del *loop()* vengono ripetute ogni 100ms.

4.4 Applicazione Android per il prototipo BLE

Per questo terzo prototipo è necessaria la creazione di una applicazione dedicata, in quanto cambiando mezzo di comunicazione (*Bluetooth Low Energy*), non è più possibile utilizzare la precedente che basa la ricezione dei dati tramite protocollo *Wi-Fi*.

Come per le precedenti, per la realizzazione è stato utilizzato *MIT App Inventor*. Per poter utilizzare il Bluetooth Low Energy è necessario aggiungere un pacchetto accessorio non presente tra quelli standard; il file .aix che contiene questa estensione si può scaricare al seguente indirizzo: *http://appinventor.mit.edu/extensions/* .

Le funzionalità principali dell'app rimangono invariate:

- deve saper riconoscere il messaggio inviato dal prototipo
- comunicarlo sia visivamente che acusticamente all'utente che sta utilizzando il dispositivo su cui è installata l'applicazione

È doveroso premettere che questa applicazione funziona, ma essendo la prima realizzata con questo tipo di connessione, presenta limiti e problemi che verranno risolti in sviluppi successivi a questa tesi.

L'utente dopo aver acceso manualmente il *Bluetooth*, per poter utilizzare l'app, deve avviarla dal menù del proprio dispositivo. Una volta eseguita questa operazione la schermata che compare è la seguente:

Figura 106 - Schermata applicazione: BLE Test

Come si può vedere dalla figura sono presenti quattro pulsanti, l'immagine della lanterna semaforica ed una scritta che riporta sullo schermo lo stato della lanterna in modo che l'utente ipovedente possa avere una conferma visiva di quanto comunicato vocalmente. Partendo dall'alto verso il basso i pulsanti sono:

- ***"Scan"***: permette di eseguire la scansione dei dispositivi Bluetooth circostanti; completata questa operazione comparirà sullo schermo una scritta che indica che la scansione è stata eseguita e che sono stati trovati dei dispositivi, se presenti

- ***"Choose BLE Device"***: dopo aver premuto questo pulsante sullo schermo compare un elenco in cui sono riportati tutti i dispositivi trovati durante la scansione; da questo elenco è possibile scegliere l'apparecchiatura a cui connettersi

- ***"Connect"***: viene effettuata la connessione alla sorgente selezionata; eseguita questa operazione, automaticamente verrà comunicato all'utente lo stato dell'attraversamento pedonale, queste informazioni vengono inviate ogni *100ms*

- ***"Disconnect"**:* permette la disconnessione dal dispositivo *BLE*

4.4.1 Blocchi MIT App Inventor

Di seguito vengono riportati tutti i blocchi utilizzati per programmare l'applicazione per sistemi Android. Vediamo ora come si presenta la schermata di programmazione nella totalità:

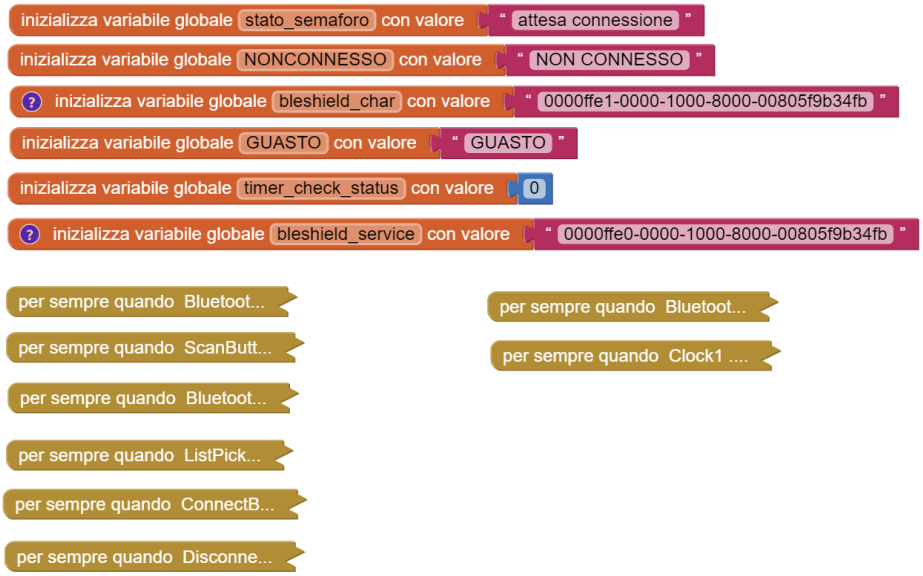

Figura 107 - MIT App Inventor schermata blocchi applicazione BLE Test

Come per le applicazioni precedenti i blocchi posso essere raccolti in gruppi che ne differenziano la natura gli uni dagli altri.

4.4.1.1 Variabili globali

Figura 108 - MIT App Inventor blocchi variabili globali BLE Test

La prima azione compiuta dall'applicazione al lancio è quella di impostare le variabili ai valori sovrastanti.

4.4.1.2 Interazione con l'utente

Figura 109 - MIT App Inventor blocco ScanButton

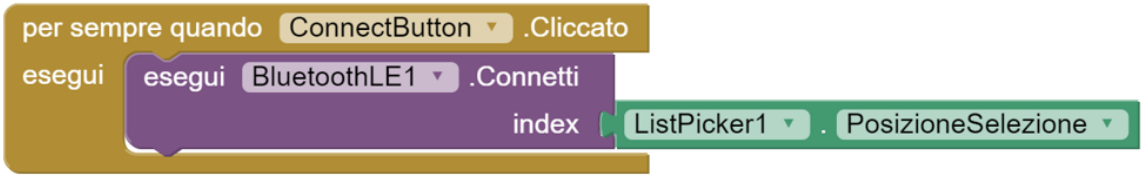

Figura 110 - MIT App Inventor blocco ConnectButton

Figura 111 - MIT App Inventor blocco Disconnect

Figura 112 - MIT App Inventor blocco ListPicker1

I blocchi riportati in questo paragrafo vengono utilizzati per creare l'interazione con l'utente. Il primo (*fig.109*) esegue la scansione dei dispositivi Bluetooth e fa comparire sullo schermo la scritta "*Scanning*". Il secondo (*fig.110*) avvia la connessione con il prototipo di misura. Il terzo (*fig.112*) viene utilizzato per effettuare la disconnessione dal sistema a cui si è connessi. L'ultimo blocco (*fig.112*) permette la selezione del dispositivo Bluetooth a cui connettersi.

4.4.1.3 Azioni automatiche

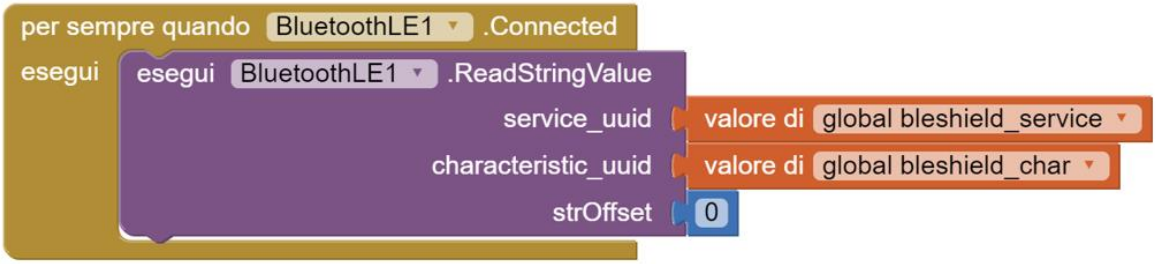

Figura 113 - MIT App Inventor blocco BluetoothLE1 .Connected

Figura 114 - MIT App Inventor blocco BluetoothLE1 .DeviceFound

Figura 115 - MIT App Inventor blocco Clock1

I blocchi presenti in questo paragrafo sono quelli di sistema che permetto il funzionamento dell'applicazione con il modulo Bluetooth Low Energy. Il primo (*fig.113*) realizza la connessione con il prototipo. Il secondo (*fig.114*), se vengono individuati dei dispositivi nelle vicinanze, fa comparire sullo schermo la scritta: "*Found devices, Please choose BLE Shield from list*" e creare una lista di tutti quelli trovati durare la scansione. Infine il terzo ed ultimo blocco (*fig.115*) controlla che ogni secondo lo smartphone o il tablet su cui è installata l'applicazione riceva informazioni dal prototipo. Se questa condizione non si verifica allora avverte l'utente con la scritta "*inattivo*".

4.4.1.4 Richiesta di stato

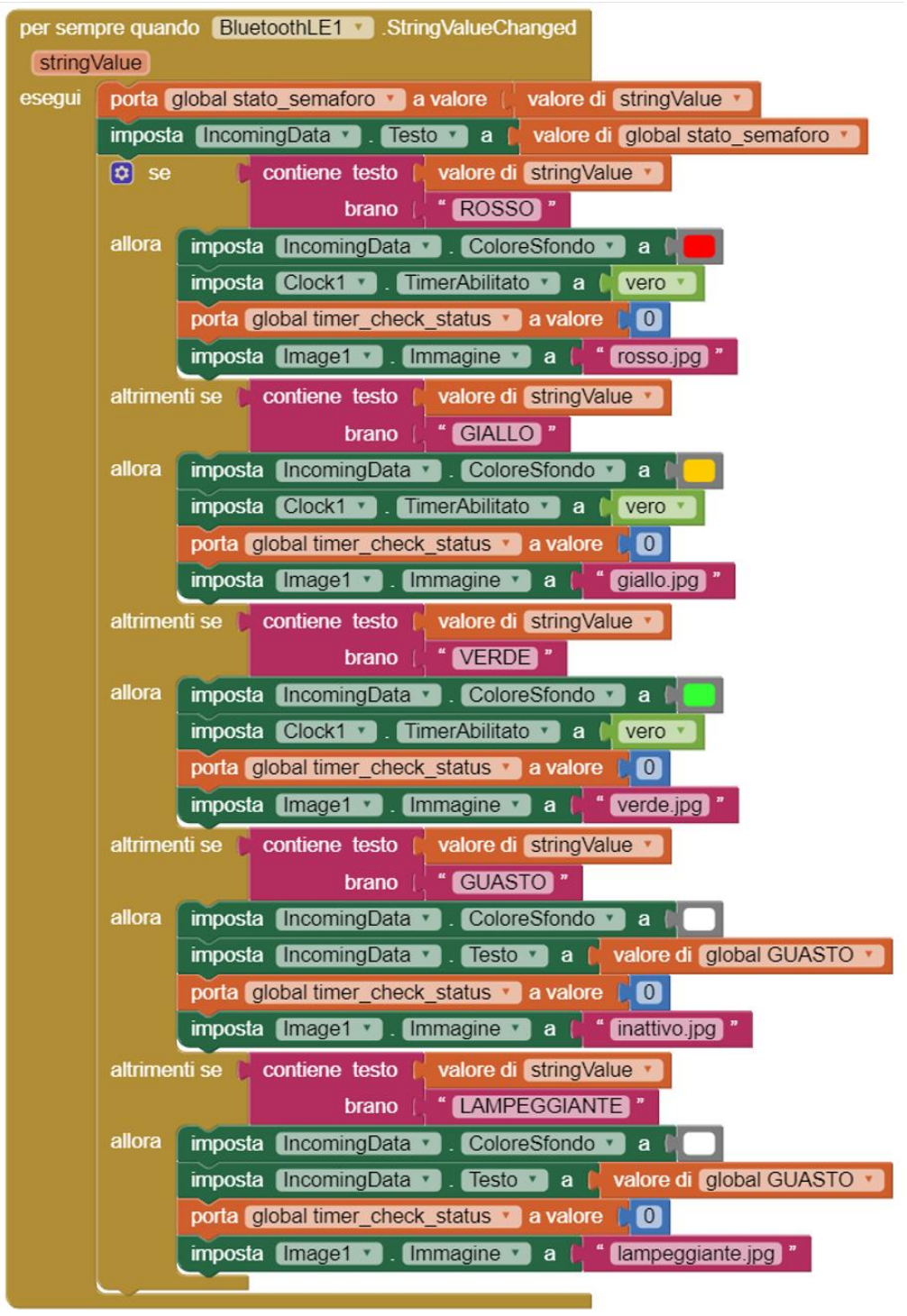

Figura 116 - MIT App Inventor blocco BluetoothLE1 StringValueChanged

Il blocco in questione analizza le informazioni ricevute dal prototipo e comunica all'utente lo stato della lanterna tramite scritte e segnali visivi colorati.

4.5 Test del prototipo BLE

Come per i precedenti prototipi, anche questo è stato testato in laboratorio ottenendo dei risultati più che soddisfacenti sia sul piano del funzionamento del circuito di misura ed elaborazione, sia sul metodo di comunicazione delle informazioni all'utente.

Anche in questo caso, grazie alla collaborazione con *IREN S.p.A.*, in data 9 agosto 2017, è stata realizzata una prova sulla centralina semaforica all'incrocio di *Via Vittorio Emanuele – Via dell'Arsenale* a Torino.

Figura 117 - Test prototipo BLE

A differenza dei primi due, questo prototipo non è stato collocato nel quadro della centralina, ma bensì sullo stesso palo ove è posizionata la lanterna da monitorare. I test eseguiti hanno dato risultati incoraggianti tenendo fede alle aspettative.

Conclusioni

L'analisi del mercato semaforico in Italia, svolta in questo elaborato, ha mostrato che sono presenti ampi spazi per soluzioni innovative a basso costo che permettono di migliorare i sistemi esistenti utilizzando le nuove tecnologie di comunicazione.

La realizzazione pratica del prototipo ha ampiamente dimostrato che è possibile aiutare le persone non vedenti o ipovedenti in un'azione quotidiana come quella di attraversare la strada, garantendo loro sicurezza e affidabilità che al giorno d'oggi ancora mancano nella maggior parte degli attraversamenti pedonali regolati da semafori. Questo tipo di aiuto può essere anche esteso alla presenza sempre più crescente degli smartphone walkers, ovvero coloro che camminano per le strade fissando il proprio telefono o tablet e che non si accorgono del mondo che li circonda.

L'impatto minimo del dispositivo nella centralina semaforica, sia come ingombro in termini di spazio, sia come interferenze con il circuito esistente, rendono questo progetto universale e di facile installazione sugli impianti tutt'ora esistenti.

L'applicazione creata per ricevere e comunicare le informazioni all'utente è installabile su qualunque apparecchio dotato di sistema operativo Android. Questa presenta ampi margini di miglioramento rispetto alla versione utilizzata durante la realizzazione di questa tesi; negli sviluppi successivi si cercherà di realizzare una versione compatibile anche con i prodotti Apple dotati di sistema operativo *iOS*.

Al lavoro svolto in questo elaborato si può aggiungere che i dati ricavati dal dispositivo potranno essere utilizzati anche per altri scopi, come fornire informazioni ai sistemi di guida autonoma e ai sistemi di navigazione già esistenti. In questo modo potranno essere utilizzati per ottimizzare i tempi di percorrenza, aumentare la fluidità del traffico cittadino e fornire indicazioni al guidatore sull'andatura migliore da tenere. Questo si traduce in un risparmio di carburante ed un abbattimento delle emissioni di inquinanti, i quali, al giorno d'oggi creano non pochi problemi ai grossi centri urbani come Torino, proprio dove questo progetto è stato sviluppato.

Allegati

- Datasheet trasformatore amperometrico *Talema AC1005*:
  http://www.talemaindia.com/AC%20QRG%20Jun-06.pdf
- Datasheet trasformatore voltmetrico *Rs Pro 1213820*:
  http://docs-europe.electrocomponents.com/webdocs/15c2/0900766b815c22e1.pdf
- Datasheet optoisolatore *Vishay 4n35*:
  https://www.vishay.com/docs/81181/4n35.pdf
- Datasheet ponte a diodi *Vishay 2W08G 416C*:
  https://www.vishay.com/docs/88528/2w005g.pdf
- Datasheet mosfet *ZVN2106A*:
  http://docs-europe.electrocomponents.com/webdocs/009e/0900766b8009ea4f.pdf
- Datasheet router *TP-Link 8790*:
  http://static.tp-link.com/resources/document/TD-W8970(UN_V3_Datasheet.pdf

Sketch di Arduino Uno per **centralina semaforica**:

```
/*Lo Sketch esegue le seguenti operazioni:
  -misura il valore della resistenza variabile del potenziometro in un range da 0 a 1023
  -a seconda del valore letto imposta i diversi ritardi di accensione delle tre lampade:
  se compreso tra 5 e 600 il ciclo è ridotto
  se superiore a 600 allora il ciclo sarà completo
  - nel caso in cui il valore del potenziometro sia inferiore di 5 allora imposterà lo stato di giallo lampeggiante
*/
//PIN 4 = ROSSO
//PIN 5 = VERDE
//PIN 6 = GIALLO
unsigned char relayPin[3] = {4, 5, 6};
const int potPin = A0;
int potVal;
float ritardo;
float ritardogiallo;
float ritardorosso;
void setup()
{
  int i;
  for (i = 0; i < 3; i++)
  {
    pinMode(relayPin[i], OUTPUT);
  }
  pinMode(potPin, INPUT);
}
void loop()
{
  potVal = analogRead(potPin);
  Serial.println(potVal);
  if (potVal >= 5 && potVal < 600) {
    ritardo = 27000; // ROSSO DURA 27 Secondi
  }
  else {
    ritardo = 30000; //ROSSO DURA 30 Secondi
  }
```

```
  ritardogiallo = ritardo - ritardo / (3); // VERDE DURA o 20 Secondi o 18 Secondi
  ritardorosso = (ritardo - ritardogiallo); // GIALLO DURA o 10 Secondi o 9 Secondi
  //SEMAFORO SPENTO
  if (potVal < 5) {
    //GIALLO
    digitalWrite(relayPin[2], HIGH);
    delay(1000);
    digitalWrite(relayPin[2], LOW);
    delay(1000);
  }
  else {
    //ROSSO
    digitalWrite(relayPin[0], HIGH);
    delay(ritardo);
    digitalWrite(relayPin[0], LOW);
    //VERDE
    digitalWrite(relayPin[1], HIGH);
    delay(ritardogiallo);
    digitalWrite(relayPin[1], LOW);
    //GIALLO
    digitalWrite(relayPin[2], HIGH);
    delay(ritardorosso);
    digitalWrite(relayPin[2], LOW);
  }
}
```

Sketch di Arduino Due per il **primo prototipo**:

```
/*Lo Sketch esegue le seguenti operazioni:

  -misura le correnti di ogni lampada semaforica e restituisce i valori ogni 200ms in un range da o a 1023

  -determina lo stato del semaforo 1 Rosso 2 Giallo 3 Verde 4 Spento o lampada bruciata (ancora da migliorare lo stato 4)

  -ad ogni richiesta della pagina http:\\192.168.1.177 invia lo stato ed il numero del semaforo

  -al primo ciclo calcola i tempi di ogni lampada e li stocca

  -durante i cicli successivi continua a calcolare i tempi di ogni lampada e li confronta con quelli stoccati

  -se i tempi concidono con quelli stoccati il ciclo continua senza modifiche, nel caso non coincidessero vengono calcolati e stoccati i nuovi tempi per i confornti futuri

  -quando il semaforo è verde (stato 3) viene realizzato il countdown con una striscia di 8 led */


#define PINROSSO A0          //definizione ingressi analogici del circuito di misura

#define PINGIALLO A1         //definizione ingressi analogici del circuito di misura

#define PINVERDE A2          //definizione ingressi analogici del circuito di misura

#include <Ethernet.h>        //libreria per invio tramite porta Ethernet su pagina http:\\192.168.1.177

#include <SPI.h>             //libreria per comunicazione seriale tra dispositivi

#include <Grove_LED_Bar.h>   //libreria per striscia Led

Grove_LED_Bar bar(9, 8, 0);  // Clock pin, Data pin, Orientation della striscia LED


int rosso = 0;               //Valore corrente lampada Rossa

int giallo = 0;              //Valore corrente lampada Gialla

int verde = 0;               //Valore corrente lampada Verde

//int cont=0;

int t0 = 0;                  //Tempo usato nella misura di corrente

int statoatt = 0;            //Stato attuale del Semaforo

int statopre = 0;            //Stato precendete del Semaforo

int ti = 0;                  //Tempo iniziale per il calcolo dei tempi di ogni lampada

int tf = 0;                  //Tempo finale per il calcolo dei tempi di ogni lampada

int tis = 0;                 //Tempo iniziale per il calcolo dei tempi di ogni lampada (Stock)

int tfs = 0;                 //Tempo finale per il calcolo dei tempi di ogni lampada (Stock)

int tir = 0;                 //Tempo iniziale per il calcolo dei tempi di ogni lampada (Reset)

int tfr = 0;                 //Tempo finale per il calcolo dei tempi di ogni lampada (Reset)

int trosso = 0;              //Tempo lampada Rossa

int tgiallo = 0;             //Tempo lampada Gialla
```

```
int tverde = 0;              //Tempo lampada Verde
int trossostock = 0;         //Tempo lampada Rossa (stock)
int tgiallostock = 0;        //Tempo lampada Gialla (stock)
int tverdestock = 0;         //Tempo lampada Verde (stock)
//int conteggio=0;           //Numero di misure
int conteggiostock = 0;      //Numero di cambi di stato durante il calcolo dei tempi (stock)
int conteggioreset = 0;      //Numero di cambi di stato durante il calcolo dei tempi (reset)
int start = 0;               //Impone ti=millis() quando è pari a zero
int startstock = 0;          //Impone tis=millis() quando è pari a zero (stock)
int startreset = 0;          //Impone tir=millis() quando è pari a zero (reset)
int lampeggiante = 0;        //Usata per determinare lo stato 4
int bruciata = 0;            //Usata per determinare lo stato 4
//int tempistica=0;
int reset = 0;               //Usata per determinare la sottofunzione Reset in caso di tempi non coincidenti
int i = 0;                   //Usata per il contdown con i led
int led = 10;                //Numero di led da accendere per il countdown
int trealeverde = 0;         //Tempo reale per spegnimento singolo led
int taccensione = 0;         //Tempo per cui ogni led deve rimanere acceso
int trealeverdeStart = 0;    //Tempo di partenza dello stato Verde
int numerosemaforo = 1;      //Numero identificativo Semaforo Duca degli Abruzzi

byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);
EthernetServer server(80);

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
```

```
  Serial.println(Ethernet.localIP());
  Serial.println("Stock");


  while (conteggiostock < 5) {
    Misura();             //Restituisce i valori di corrente delle lampade, lo stato del semaforo ed invia su richiesta lo
statoed il numero del semaforo alla pagina http:\\192.168.1.177
    Tempisticastock();    //Determina i tempi di durata delle lampade e li stocca per i confronti
    Cambiostato();        //Cambia lo stato del semaforo
  }
  Serial.println("Fine Stock");
}


void loop() {
  Misura();           //Restituisce i valori di corrente delle lampade, lo stato del semaforo ed invia su richiesta lo
statoed il numero del semaforo alla pagina http:\\192.168.1.177
  Tempistica();       //Determina i tempi di durata delle lampade
  Confronto();        //Confronta il tempo della lampada appena spenta con quello stoccato, nel caso siano differenti
impone reset=1
  if (reset == 1) {
    Cambiostato();    //Cambia lo stato del semaforo
    Reset();          //Ricalcola i tempi di ogni lampada e li stocca
  }
  Cambiostato();      //Cambia lo stato del semaforo
  Led();              //Esegue il countdown con i led nel caso in cui il semaforo è verde
}


void Misura() {
  t0 = millis();
  while ((millis() - t0) < 200) {
    int rossomis = analogRead(PINROSSO);
    if (rossomis > rosso) {
      rosso = rossomis;
    }
    int giallomis = analogRead(PINGIALLO);
    if (giallomis > giallo) {
      giallo = giallomis;
    }
```

```
    int verdemis = analogRead(PINVERDE);
    if (verdemis > verde) {
      verde = verdemis;
    }
    //cont++;
  }
  //int velo=cont/200;
  Serial.print("VALORI SEMAFORO rosso=");
  Serial.print(rosso);
  Serial.print(", giallo=");
  Serial.print(giallo);
  Serial.print(", verde=");
  Serial.print(verde);
  //    Serial.print(", n misure=");
  //    Serial.println(cont);
  //    Serial.print(", velocità=");
  //    Serial.println(velo);
  Statosemaforo();
  PaginaLocale();
  Serial.print(", Stato=");
  Serial.print(statoatt);
  Serial.print(", StatoPre=");
  Serial.println(statopre);
  if (statoatt == 4 && bruciata > 1 && lampeggiante == 0) {
    Serial.println("LAMPADA BRUCIATA");
  }
  if (statoatt == 4 && lampeggiante > 1 && bruciata > 1) {
    Serial.println("SEMAFORO SPENTO");
  }
  rosso = 0;
  giallo = 0;
  verde = 0;
  //cont=0;
}
```

```
void PaginaLocale() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    //Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be closed after completion of the response
          //client.println("Refresh: 3");  // refresh the page automatically every 5 sec
          client.println();
          // output the value of each analog input pin
          //            client.print(rosso);
          //            client.print(",");
          //            client.print(giallo);
          //            client.print(",");
          //            client.print(verde);
          //            client.print(",");
          client.print(statoatt);
          client.print(",");
          client.print(numerosemaforo);
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
```

```
        } else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
       }
    }
    // give the web browser time to receive the data
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}

void Statosemaforo() {
  if (rosso > 500 && giallo < 500 && verde < 500 || rosso == 1023) {
    statoatt = 1; //ROSSO
    bruciata = 0;
    lampeggiante = 0;
  }
  if (rosso < 500 && giallo > 500 && verde < 500 || giallo == 1023) {
    statoatt = 2; //GIALLO
    lampeggiante++;
  }
  if (rosso < 500 && giallo < 500 && verde > 500 || verde == 1023) {
    statoatt = 3; //VERDE
    lampeggiante = 0;
    bruciata = 0;
  }
  if (rosso < 500 && giallo < 500 && verde < 500) {
    statoatt = 4; //Semaforo spento o lampada bruciata
    bruciata++;
  }
}
```

```
void Tempistica() {
  if (start == 0) {
    ti = millis();
    start++;
  }
  if (statoatt != statopre) {
    tf = millis();
    if (statopre == 1) {
      trosso = tf - ti + 200;
      Serial.print("Tempo rosso=");
      Serial.print(trosso);
      Serial.println("ms");
      start = 0;
    }
    if (statopre == 2) {
      tgiallo = tf - ti + 200;
      Serial.print("Tempo Giallo=");
      Serial.print(tgiallo);
      Serial.println("ms");
      start = 0;
    }
    if (statopre == 3) {
      tverde = tf - ti + 200;
      Serial.print("Tempo Verde=");
      Serial.print(tverde);
      Serial.println("ms");
      start = 0;
    }
  }
}
```

```
void Tempisticastock() {
  if (startstock == 0 && conteggiostock > 1) {
    tis = millis();
    startstock++;
  }
  if (statoatt != statopre && conteggiostock <= 4) {
    tfs = millis();
    if (statopre == 1 && conteggiostock != 1) {
      trossostock = tfs - tis + 200;
      Serial.print("Tempo rosso stock=");
      Serial.print(trossostock);
      Serial.println("ms");
      startstock = 0;
    }
    if (statopre == 2 && conteggiostock != 1) {
      tgiallostock = tfs - tis + 200;
      Serial.print("Tempo Giallo stock=");
      Serial.print(tgiallostock);
      Serial.println("ms");
      startstock = 0;
    }
    if (statopre == 3 && conteggiostock != 1) {
      tverdestock = tfs - tis + 200;
      Serial.print("Tempo Verde stock=");
      Serial.print(tverdestock);
      Serial.println("ms");
      startstock = 0;
    }
    conteggiostock++;
    Serial.print("Conteggio stock=");
    Serial.println(conteggiostock);
  }
}
```

```
void Tempisticareset() {
  if (startreset == 0) {
    tir = millis();
    startreset++;
  }
  if (statoatt != statopre && conteggioreset <= 2) {
    tfr = millis();
    if (statopre == 1) {
      trossostock = tfr - tir + 200;
      Serial.print("Tempo rosso stock reset=");
      Serial.print(trossostock);
      Serial.println("ms");
      startreset = 0;
    }
    if (statopre == 2) {
      tgiallostock = tfr - tir + 200;
      Serial.print("Tempo Giallo stock reset=");
      Serial.print(tgiallostock);
      Serial.println("ms");
      startreset = 0;
    }
    if (statopre == 3) {
      tverdestock = tfr - tir + 200;
      Serial.print("Tempo Verde stock reset=");
      Serial.print(tverdestock);
      Serial.println("ms");
      startreset = 0;
    }
    conteggioreset++;
    Serial.print("Conteggio reset=");
    Serial.println(conteggioreset);
  }
}
```

```
void Cambiostato() {
  if (statoatt != statopre) {
    Serial.println("Stato cambiato");
    statopre = statoatt;
  }
}

void Confronto() {
  if (statoatt != statopre) {
    if (statoatt == 3 && statopre == 1 && (trosso < (trossostock - 500) || trosso > (trossostock + 500))) {
      Serial.println("Ciclo cambiato R");
      reset = 1;
      trossostock = trosso + 200;
      Serial.print("Tempo rosso stock reset=");
      Serial.print(trossostock);
      Serial.println("ms");
    }
    if (statoatt == 1 && statopre == 2 && (tgiallo < (tgiallostock - 500) || tgiallo > (tgiallostock + 500))) {
      Serial.println("Ciclo cambiato G");
      reset = 1;
      tgiallostock = tgiallo + 200;
      Serial.print("Tempo Giallo stock reset=");
      Serial.print(tgiallostock);
      Serial.println("ms");
    }
    if (statoatt == 2 && statopre == 3 && (tverde < (tverdestock - 500) || tverde > (tverdestock + 500))) {
      Serial.println("Ciclo cambiato V");
      reset = 1;
      tverdestock = tverde + 200;
      Serial.print("Tempo Verde stock reset=");
      Serial.print(tverdestock);
      Serial.println("ms");
    }
  }
}
```

```
void Reset() {
  Serial.println("Reset tempi stock in corso");
  //statopre=0;
  //statoatt=0;
  tir = 0;
  tfr = 0;
  conteggioreset = 0;
  startreset = 0;
  //trossostock=0;
  //tgiallostock=0;
  //tverdestock=0;
  while (conteggioreset < 2) {
    Misura(); //MI RESTITUISCE I VALORI DELLE VARIE LAMPADE E LO STATO 1 ROSSO 2 GIALLO 3 VERDE
4 LAMPADA BRUCIATA o  SEMAFORO SPENTO
    Tempisticareset(); //MI RESTITUISCE I TEMPI DELLE VARIE LAMPADE E LI STOCCA
    Cambiostato();//CAMBIA LO STATO DEL SEMAFORO
    Led();
  }
  Serial.println("Fine Reset");
  reset = 0;
}

void Led() {
  if (statoatt == 3) {
    if (i == 0) {
      led = 10;
      bar.setLevel(led);
      trealeverdeStart = millis();
      i++;
    }
    trealeverde = millis();
    taccensione = tverdestock / 8;
    if (trealeverde >= (i * taccensione + trealeverdeStart)) {
      led = 10 - i;
      bar.setLevel(led);
      i++;
```

```
    }
  }
  if (statoatt == 1) {
    i = 0;
    led = 0;
    bar.setLevel(led);
  }
  if (statoatt == 2) {
    i = 0;
    led = 0;
    bar.setLevel(led);
  }
}
```

Sketch di Arduino Due per il **secondo prototipo**:

```
#include <Ethernet.h>        //libreria per invio tramite porta Ethernet su pagina http:\\192.168.1.177
#include <SPI.h>             //libreria per comunicazione seriale tra dispositivi
int PINROSSO = 4;
int PINGIALLO = 5;
int PINVERDE = 6;
int rosso = 0;               //Valore corrente lampada Rossa
int giallo = 0;              //Valore corrente lampada Gialla
int verde = 0;               //Valore corrente lampada Verde
int t0 = 0;                  //Tempo usato nella misura di corrente
int statoatt = 0;            //Stato attuale del Semaforo
int statopre = 0;            //Stato precendete del Semaforo
int ti = 0;                  //Tempo iniziale per il calcolo dei tempi di ogni lampada
int tf = 0;                  //Tempo finale per il calcolo dei tempi di ogni lampada
int tis = 0;                 //Tempo iniziale per il calcolo dei tempi di ogni lampada (Stock)
int tfs = 0;                 //Tempo finale per il calcolo dei tempi di ogni lampada (Stock)
int tir = 0;                 //Tempo iniziale per il calcolo dei tempi di ogni lampada (Reset)
int tfr = 0;                 //Tempo finale per il calcolo dei tempi di ogni lampada (Reset)
int trosso = 0;              //Tempo lampada Rossa
int tgiallo = 0;             //Tempo lampada Gialla
int tverde = 0;              //Tempo lampada Verde
int trossostock = 0;         //Tempo lampada Rossa (stock)
int tgiallostock = 0;        //Tempo lampada Gialla (stock)
int tverdestock = 0;         //Tempo lampada Verde (stock)
int conteggiostock = 0;      //Numero di cambi di stato durante il calcolo dei tempi (stock)
int conteggioreset = 0;      //Numero di cambi di stato durante il calcolo dei tempi (reset)
int start = 0;               //Impone ti=millis() quando è pari a zero
int startstock = 0;          //Impone tis=millis() quando è pari a zero (stock)
int startreset = 0;          //Impone tir=millis() quando è pari a zero (reset)
int reset = 0;               //Usata per determinare la sottofunzione Reset in caso di tempi non coincidenti
int numerosemaforo = 1;      //Numero identificativo Semaforo Duca degli Abruzzi
int lamp = 0;
int spento = 0;
byte mac[] = {
```

```
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);
EthernetServer server(80);

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  // start the Ethernet connection and the server:
  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
  Serial.println("Stock");
  pinMode(PINROSSO, INPUT);
  pinMode(PINGIALLO, INPUT);
  pinMode(PINVERDE, INPUT);

while (conteggiostock < 5) {
    Misura();               //Restituisce i valori di corrente delle lampade, lo stato del semaforo ed invia su richiesta lo statoed il numero del semaforo alla pagina http:\\192.168.1.177
    Tempisticastock();    //Determina i tempi di durata delle lampade e li stocca per i confronti
    Cambiostato();        //Cambia lo stato del semaforo
  }
  Serial.println("Fine Stock");
}
```

```
void loop() {

  Misura();         //Restituisce i valori di corrente delle lampade, lo stato del semaforo ed invia su richiesta lo statoed il numero del semaforo alla pagina http:\\192.168.1.177

  Tempistica();      //Determina i tempi di durata delle lampade

  Confronto();       //Confronta il tempo della lampada appena spenta con quello stoccato, nel caso siano differenti impone reset=1

  if (reset == 1) {

    Cambiostato();   //Cambia lo stato del semaforo

    Reset();         //Ricalcola i tempi di ogni lampada e li stocca

  }

  Cambiostato();     //Cambia lo stato del semaforo

}


void Misura() {

  rosso = digitalRead(PINROSSO);

  giallo = digitalRead(PINGIALLO);

  verde = digitalRead(PINVERDE);

  Serial.print("VALORI SEMAFORO rosso=");

  Serial.print(rosso);

  Serial.print(", giallo=");

  Serial.print(giallo);

  Serial.print(", verde=");

  Serial.print(verde);

  Statosemaforo();

  PaginaLocale();

  Serial.print(", Stato=");

  Serial.print(statoatt);

  Serial.print(", StatoPre=");

  Serial.println(statopre);

  rosso = 0;

  giallo = 0;

  verde = 0;

  delay (50);

}
```

```
void PaginaLocale() {
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    //Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        // if you've gotten to the end of the line (received a newline
        // character) and the line is blank, the http request has ended,
        // so you can send a reply
        if (c == '\n' && currentLineIsBlank) {
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connection: close");  // the connection will be closed after completion of the response
          //client.println("Refresh: 3");  // refresh the page automatically every 5 sec
          client.println();
          // output the value of each analog input pin
         client.print(statoatt);
          client.print(",");
          client.print(numerosemaforo);
          break;
        }
        if (c == '\n') {
          // you're starting a new line
          currentLineIsBlank = true;
        } else if (c != '\r') {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }
```

```
    // give the web browser time to receive the data
    // close the connection:
    client.stop();
    Serial.println("client disconnected");
  }
}

void Statosemaforo() {
  if (rosso == 1 && giallo == 0 && verde == 0) {
    statoatt = 1; //ROSSO
    lamp = 0;
    spento = 0;
  }
  if (rosso == 0 && giallo == 1 && verde == 0) {
    statoatt = 2; //GIALLO
    lamp++;
  }
 if (rosso == 0 && giallo == 0 && verde == 1) {
    statoatt = 3; //VERDE
    lamp = 0;
    spento = 0;
  }
  if ((rosso == 0 && giallo == 0 && verde == 0) || (rosso == 0 && giallo == 1 && verde == 1) || (rosso == 1 && giallo
== 0 && verde == 1) || (rosso == 1 && giallo == 1 && verde == 0) || (rosso == 1 && giallo == 1 && verde == 1)) {
    statoatt = 4;
    spento++;
  }
  if (lamp > 2 && spento > 2) {
    statoatt = 5;
  }
}
```

```
void Tempistica() {
  if (start == 0) {
    ti = millis();
    start++;
  }
  if (statoatt != statopre) {
    tf = millis();
    if (statopre == 1) {
      trosso = tf - ti + 100;
      Serial.print("Tempo rosso=");
      Serial.print(trosso);
      Serial.println("ms");
      start = 0;
    }
    if (statopre == 2) {
      tgiallo = tf - ti + 100;
      Serial.print("Tempo Giallo=");
      Serial.print(tgiallo);
      Serial.println("ms");
      start = 0;
    }
    if (statopre == 3) {
      tverde = tf - ti + 100;
      Serial.print("Tempo Verde=");
      Serial.print(tverde);
      Serial.println("ms");
      start = 0;
    }
    if (statopre == 4 || statopre == 5) {
      start = 0;
    }
  }
}
```

```
void Tempisticastock() {
  if (startstock == 0 && conteggiostock > 1) {
    tis = millis();
    startstock++;
  }
  if (statoatt != statopre && conteggiostock <= 4) {
    tfs = millis();
    if (statopre == 1 && conteggiostock != 1) {
      trossostock = tfs - tis + 100;
      Serial.print("Tempo rosso stock=");
      Serial.print(trossostock);
      Serial.println("ms");
      startstock = 0;
    }
    if (statopre == 2 && conteggiostock != 1) {
      tgiallostock = tfs - tis + 100;
      Serial.print("Tempo Giallo stock=");
      Serial.print(tgiallostock);
      Serial.println("ms");
      startstock = 0;
    }
    if (statopre == 3 && conteggiostock != 1) {
      tverdestock = tfs - tis + 100;
      Serial.print("Tempo Verde stock=");
      Serial.print(tverdestock);
      Serial.println("ms");
      startstock = 0;
    }
    if (statopre == 4 || statopre == 5) {
      startstock = 0;
    }
    conteggiostock++;
    Serial.print("Conteggio stock=");
    Serial.println(conteggiostock);
  }
}
```

```
void Tempisticareset() {
  if (startreset == 0) {
    tir = millis();
    startreset++;
  }
  if (statoatt != statopre && conteggioreset <= 2) {
    tfr = millis();
    if (statopre == 1) {
      trossostock = tfr - tir + 100;
      Serial.print("Tempo rosso stock reset=");
      Serial.print(trossostock);
      Serial.println("ms");
      startreset = 0;
    }
    if (statopre == 2) {
      tgiallostock = tfr - tir + 100;
      Serial.print("Tempo Giallo stock reset=");
      Serial.print(tgiallostock);
      Serial.println("ms");
      startreset = 0;
    }
    if (statopre == 3) {
      tverdestock = tfr - tir + 100;
      Serial.print("Tempo Verde stock reset=");
      Serial.print(tverdestock);
      Serial.println("ms");
      startreset = 0;
    }
    if (statopre == 4 || statopre == 5) {
      startreset = 0;
    }
    conteggioreset++;
    Serial.print("Conteggio reset=");
    Serial.println(conteggioreset);
  }
}
```

```
void Cambiostato() {
  if (statoatt != statopre) {
    Serial.println("Stato cambiato");
    statopre = statoatt;
  }
}

void Confronto() {
  if (statoatt != statopre) {
    if (statoatt == 3 && statopre == 1 && (trosso < (trossostock - 500) || trosso > (trossostock + 500))) {
      Serial.println("Ciclo cambiato R");
      reset = 1;
      trossostock = trosso + 100;
      Serial.print("Tempo rosso stock reset=");
      Serial.print(trossostock);
      Serial.println("ms");
    }
    if (statoatt == 1 && statopre == 2 && (tgiallo < (tgiallostock - 500) || tgiallo > (tgiallostock + 500))) {
      Serial.println("Ciclo cambiato G");
      reset = 1;
      tgiallostock = tgiallo + 100;
      Serial.print("Tempo Giallo stock reset=");
      Serial.print(tgiallostock);
      Serial.println("ms");
    }
    if (statoatt == 2 && statopre == 3 && (tverde < (tverdestock - 500) || tverde > (tverdestock + 500))) {
      Serial.println("Ciclo cambiato V");
      reset = 1;
      tverdestock = tverde + 100;
      Serial.print("Tempo Verde stock reset=");
      Serial.print(tverdestock);
      Serial.println("ms");
    }
  }
}
```

```
void Reset() {
  Serial.println("Reset tempi stock in corso");
  tir = 0;
  tfr = 0;
  conteggioreset = 0;
  startreset = 0;
  while (conteggioreset < 2) {
    Misura(); //MI RESTITUISCE I VALORI DELLE VARIE LAMPADE E LO STATO 1 ROSSO 2 GIALLO 3 VERDE
4 LAMPADA BRUCIATA o  SEMAFORO SPENTO
    Tempisticareset(); //MI RESTITUISCE I TEMPI DELLE VARIE LAMPADE E LI STOCCA
    Cambiostato();//CAMBIA LO STATO DEL SEMAFORO
  }
  Serial.println("Fine Reset");
  reset = 0;
}
```

Sketch di Arduino Pro Mini per **prototipo BLE**:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX
// Schema per la connessione fra Arduino e modulo BLE
// Connect HM10      Arduino Uno
//     TXD          Pin 10  RXD          Pin 11
//variabili da dichiarare
int PINROSSO = 9;
int PINGIALLO = 8;
int PINVERDE = 7;
int stato_rosso = 0;
int stato_giallo = 0;
int stato_verde = 0;
int stato_precedente = " ";
int contatore_giallo = 0;
String stato_lanterna = "indefinito";
String messaggio_completo = "indefinito";
String indirizzo = " ";//Corso Vittorio Emanuele angolo Via Arsenale; ";
void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  pinMode(4, OUTPUT); //alimento il circuito giallo, rosso,verde con tensioni stabili
  digitalWrite(4, HIGH);
  pinMode(5, OUTPUT); //alimento il circuito giallo, rosso,verde con tensioni stabili
  digitalWrite(5, HIGH);
  pinMode(6, OUTPUT); //alimento il circuito giallo, rosso,verde con tensioni stabili
  digitalWrite(6, HIGH);
  pinMode(12, OUTPUT); //alimento la board hm-10  dall'arduino dal pin3 solo dopo che è tutto caricato e finito
  digitalWrite(12, HIGH);
  pinMode(2, OUTPUT); //alimento il circuito led giallo
  pinMode(3, OUTPUT); //alimento il circuito led rosso
  pinMode(13, OUTPUT); //alimento il circuito led verde
  digitalWrite(2, LOW); //i led devono essere spenti
  digitalWrite(3, LOW);
  digitalWrite(13, LOW);
}
```

```
void loop() {
  stato_rosso = digitalRead(PINROSSO);
  stato_giallo = digitalRead(PINGIALLO);
  stato_verde = digitalRead(PINVERDE);
  if     (stato_rosso == LOW && stato_giallo ==  LOW && stato_verde ==  HIGH)
  { stato_lanterna = "VERDE" ;
    digitalWrite(13, HIGH);
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    contatore_giallo = 0;
    stato_precedente = "verde";
  }
  else if (stato_rosso == LOW && stato_giallo == HIGH && stato_verde ==  LOW )
  { stato_lanterna = "GIALLO";
    digitalWrite(2, HIGH);
    digitalWrite(13, LOW);
    digitalWrite(3, LOW);
    if (stato_precedente = "GUASTO" ) {
      contatore_giallo++;
    }
    stato_precedente = "giallo";
  }
  else if (stato_rosso == HIGH  && stato_giallo ==  LOW && stato_verde == LOW )
  { stato_lanterna = "ROSSO";
    digitalWrite(3, HIGH);
    digitalWrite(13, LOW);
    digitalWrite(2, LOW);
    contatore_giallo = 0;
    stato_precedente = "rosso";
  }
  else if (stato_rosso == LOW && stato_giallo == HIGH && stato_verde ==  LOW && contatore_giallo > 1)
  { stato_lanterna = "lampeggiante";
    digitalWrite(2, HIGH);
    digitalWrite(13, LOW);
    digitalWrite(3, LOW);
```

```
    stato_precedente = "lampeggiante";
  }
  else
  { stato_lanterna = "GUASTO";
    digitalWrite(3, LOW);
    digitalWrite(13, LOW);
    digitalWrite(2, LOW);
    stato_precedente = "guasto";
  }
  messaggio_completo = indirizzo + stato_lanterna;
  mySerial.println(messaggio_completo);
  Serial.println(messaggio_completo);
  delay(100);
}
```

Indice delle figure

Bibliografia

[1] http://www.ibtimes.co.uk/history-traffic-lights-100th-anniversary-first-electric-traffic-system-1459680

[2] http://www.liberoquotidiano.it/news/11666320/Il-semaforo-compie-cento-anni-.html

[3] http://www.vivicasagiove.it/notizie/accadde-oggi-29-febbraio-1952-primo-semaforo-pedonale-a-ny/

[4] http://www.lasemaforica.com/it/prodotti/regolatori-semaforici/cartesio

[5] http://www.scae.net/prodotti-scae-scheda.asp?CAT=9&STCAT=3&ID=64

[6] http://www.lasemaforica.com/it/prodotti/lanterne-semaforiche

[7] http://www.inces.it/it/index.html?urlCorpo=home.html

[8] http://www.lasemaforica.com/it/prodotti/sistemi-gestione-traffico

[9] http://www.scae.net/index-it.asp

[10] http://www.emisrl.it/home-page-emi-srl-a-pietrasanta-impianti-e-forniture_1_ITA.html

[11] http://www.inces.it/it/index.html?urlCorpo=home.html

[12] http://www.lasemaforica.com/it/

[13] http://automazione-plus.it/wp-content/uploads/sites/3/2009/06/20060501041_11.pdf

[14] http://www.idea-automazioni.com/building-automation/lonworks.html?jjj=1507030736030

[15] http://www.sti.uniurb.it/romanell/Domotica_e_Edifici_Intelligenti/101223-Lez04b-LonWorks- DomoticaEdEdificiIntelligenti-Romanelli.pdf

[16] https://www.knx.org/knx-en/knx/association/what-is-knx/index.php

[17] https://www.knx.org/knx-en/index.php

[18] http://www.knx.it/

[19] http://scuola.itisurbino.it:81/materiale/konnexlezione.pdf

[20] http://it.farnell.com/microchip/mcp2515-i-so/can-controller-spi-1mbps-soic18/dp/1292239?CMP=KNC-GIT-GEN-KWL-RLSA-SA-NCATRIAL&mckv=mqrXUCSY_dc|pcrid|169847310409|&gclid=CjwKEAiAlZDFBRCKncm67qihiHwSJABtoNIg9rEOkf0-fGM1lZ5NIDUB2v1wyqbAsH68g4yu-OxxsBoCLMzw_wcB

[21] http://docenti.etec.polimi.it/IND32/Didattica/AzionamentixAutomazione/files/Dispensa_CAN.pdf

[22] http://www.dia.uniroma3.it/autom/Reti_e_Sistemi_Automazione/PDF/CAN% 20(Applicazioni% 20Auomobilistiche)%20Di%20Galanti%20Antonello.pdf

[23] D. Jiang, L. Delgrossi, *IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments*, VTC Spring 2008 - IEEE Vehicular Technology Conference, maggio 2008, doi:10.1109/VETECS.2008.458

[24] http://people.unica.it/michelenitti/files/2017/04/Tecniche-d%E2%80%99accesso-IEEE-802.11p-WAVE.pdf

[25] http://www.indjst.org/index.php/indjst/article/viewFile/34355/27974

[26] Alessandro Falaschi, *Vehicular Ad Hoc Networks: Scenari, Tecnologie, Sviluppo Applicazioni*, 2009

[27] G. Toscano, *Studio e analisi del protocollo 802.11p*, 2012

[28] 5G americans: V2X Cellular Solutions

[29] D.S. Miguel, A.R. Hirakawa, G. B. Castro, *Analysis of IEEE 802.11g Standard for Communication in a Traffic Lights Distributed Control System,* 2015 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, 2015, pp. 662-667. doi: 10.1109/CSCI.2015.123

[30] https://www.audiusa.com/newsroom/news/press-releases/2016/12/audi-launches-vehicle-to-infrastructure-tech-in-vegas

[31] https://www.sicurauto.it/sistemi-di-sicurezza/news/audi-lancia-la-connettivita-v2i-ora-lauto-parla-anche-ai-semafori.html

[32] https://www.sicurauto.it/sistemi-di-sicurezza/news/ford-e-jlr-cancellano-i-semafori-rossi-al-via-i-test-in-uk.html

[33] http://www.automotive-fleet.com/channel/safety-accident-management/news/story/2016/10/jaguar-land-rover-advancing-connected-car-tech.aspx

[34] http://sites.ieee.org/connected-vehicles/2016/10/21/jaguar-land-rover-ford-tata-motors-test-connected-autonomous-cars-uk/

[35] https://www.sicurauto.it/news/auto-connesse-hyundai-le-vuole-superintelligenti-ma-i-dati-pesano.html

[36] http://www.gm.com/mol/m-2017-may-0530-cadillac.html

[37] http://www.hdmotori.it/cadillac-continua-lo-sviluppo-dei-sistemi-v2i-e-v2v/

[38] https://www.tesla.com/it_IT/autopilot

[39] G. Russo, E. Baccaglini, L. Boulard, D. Brevi, R. Scopigno, *Video processing for V2V communications: a case study with traffic lights and plate recognition*, 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2015, Turin, Italy, doi:10.1109/RTSI.2015.7325064

[40] M. Wada, T. Yendo, T. Fujii, M. Tanimoto, *Road-to-Vehicle Communication Using LED Traffic Light*, IEEE Proceedings. Intelligent Vehicles Symposium, 2005, Las Vegas, NV, USA, doi**:** 10.1109/IVS.2005.1505169

[41] H. S. Liu, G. Pang, *Positioning Beacon System Using Digital Camera and LEDs*, IEEE Transactions on Vehicular Technology, 21 May 2003, doi: 10.1109/TVT.2003.808800

[42] M. Akanegawa, Y. Tanaka, M. Nakagawa, *Basic Study on Traffic Information System Using LED Traffic Lights*, IEEE Transactions on Intelligent Transportation Systems (Volume: 2, Issue: 4, Dec 2001), 2001, doi: 10.1109/6979.969365

[43] http://sisinflab.poliba.it/scioscia/resources/Bluetooth_intro.pdf

[44] https://www.bluetooth.com/what-is-bluetooth-technology/how-it-works

[45] https://www.digikey.it/it/articles/techzone/2014/aug/moving-forward-with-bluetooth-low-energy

[46] https://www.bluetooth.com/

[47] https://www.digikey.it/it/articles/techzone/2014/aug/moving-forward-with-bluetooth-low-energy

[48] Andrea Abate, M*isure elettriche ed elettroniche Volume 1*, Celid 1996

Tutti i siti sono stati consultati nell'ottobre del 2017.