Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

Progettazione e sviluppo di un Flow Meter a basso consumo energetico per impianti di irrigazione intelligenti

Relatore: Chiar.mo Prof. Marco di Felice

Presentata da: Simone Bertolini

Correlatori:
Tullio Salmon Cinotti
Simone Sindaco

Sessione
III
Anno Accademico
2018/2019

Sommario

1. Introduzione

L'acqua è un elemento vitale che consente di garantire la sicurezza alimentare all'intera popolazione del globo terreste. Questo perché è un elemento basilare per l'agricoltura.

Tra le attività economiche esistenti, l'agricoltura è infatti quella a più alto consumo idrico: su scala mondiale essa utilizza infatti circa il 70% delle risorse disponibili [1].

Se si analizzano unicamente i paesi in via di sviluppo la percentuale di efficienza idrica spesso sia attesta sotto al 50%. Questa quantità di acqua, già insufficiente a soddisfare a pieno i fabbisogni delle colture, tenderà sempre più a contrarsi nel tempo a causa della crescente competizione con gli usi civili ed industriali e dei cambiamenti climatici in atto.

Considerando che circa il 50% dell'acqua utilizzata in agricoltura è persa principalmente a causa di perdite nei sistemi di distribuzione e della scarsa efficienza d'uso [2], la riduzione degli attuali consumi idrici è una necessità impellente ed è la via più razionale da percorrere, anche ai fini della salvaguardia dell'ambiente.

Ci si può chiedere se gli attuali fabbisogni irrigui siano comprimibili senza contrarre le produzioni unitarie. In linea di massima, la risposta è affermativa, e le riduzioni sono possibili adottando tecniche ed accorgimenti, integrati tra loro, miranti ad elevare i valori di una o più delle singole componenti della lunga serie dell'efficienza d'uso totale dell'acqua, le quali riguardano aspetti sia di tipo idraulico sia agronomico.

Considerando, per esempio, la tecnica più comune che viene utilizzata per l'irrigazione, ovvero l'irrigazione superficiale, si può notare che questa spreca un'alta percentuale di acqua bagnando le aree in cui nessuna pianta ne trae beneficio.

L'irrigazione localizzata potrebbe quindi attenuare questo problema utilizzando l'acqua in modo più efficiente ed efficace, evitando così l'irrigazione superflua e soprattutto fornendo alle piante la giusta

quantità di acqua in base ad un algoritmo che tiene in considerazione tutti i vari fattori ambientali e le necessità della piantagione. Un grande aiuto può essere pertanto fornito dalla tecnologia. Questa infatti consentirebbe di rilevare il livello di acqua necessario alla piantagione, per farlo fluire dove e quando necessario.

1.1 SWAMP

Smart Water Management Platform (SWAMP) [3] è un progetto che affronta i succitati problemi, utilizzando Internet of Things (IoT), analisi dei dati, dispositivi autonomi e altre tecnologie correlate.
I principali obiettivi che SWAMP si è prefissato sono i seguenti:
1) Ridurre gli sprechi di acqua sviluppando applicazioni software intelligenti basate su dispositivi IoT;
2) Automatizzare piattaforme avanzate che consentano la lettura semplificata di dati relativi alle aree di irrigazione;
3) Integrare sensori eterogenei e avanzati di ogni tipo: fissi (sensori di temperatura, umidità …), mobili (controlli di flusso d'acqua) e volanti (droni), in grado di fornire maggior precisione nell'approvvigionamento idrico per l'irrigazione;
4) Proporre, testare e validare nuovi modelli di business per l'utilizzo dell'IoT in contesti di gestione intelligente dell'acqua;
5) Sviluppare componenti tecnologici al fine di fornire, allo stesso tempo, un alto di livello di scalabilità e di modularità applicabili a diversi contesti e luoghi.
Il progetto SWAMP mira a sviluppare un sistema di irrigazione intelligente completo ed in grado di garantire un'alta precisione ed affidabilità sui dati acquisiti. L'idea principale è quella di ottimizzazione l'irrigazione distribuendo l'acqua sulla base di un'analisi olistica che raccolga informazioni da tutti gli aspetti del sistema, tra cui il ciclo naturale dell'acqua stessa e le conoscenze cumulate relative alla coltivazione di particolari piante.

Questo sistema consegue notevoli risparmi, in quanto consente di prevenire una grande quantità di perdite con un alto livello di precisione, garantendo in questo modo una migliore disponibilità di acqua in situazioni in cui l'approvvigionamento idrico è limitato.

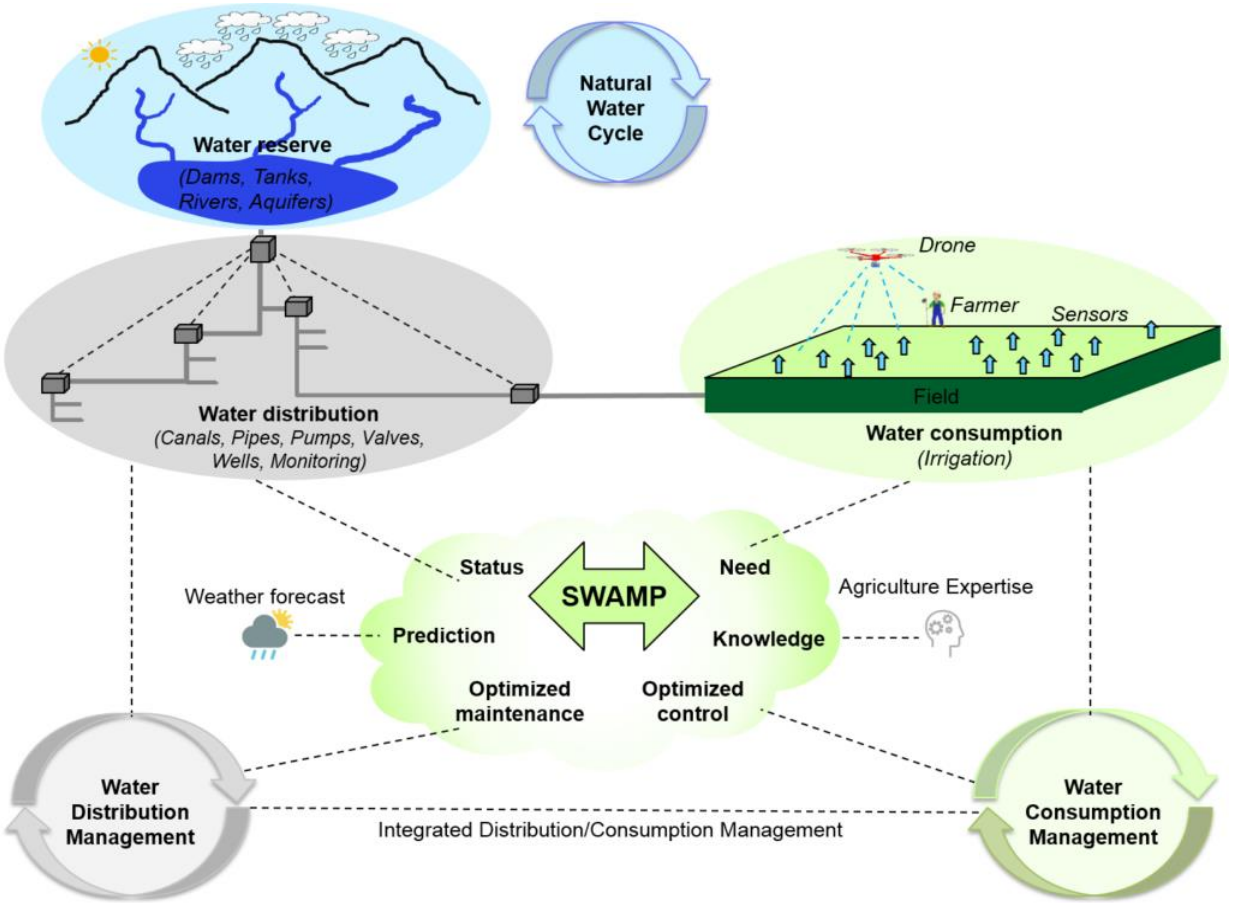


Figura 1.1.1 – Workflow progetto SWAMP [3]

Le riserve idriche derivano normalmente dal naturale ciclo dell'acqua, come per esempio fiumi oppure laghi. In Figura 1.1.1 è possibile notare come l'acqua venga trasportata dalle varie riserve all'utilizzatore finale mediante una rete di canali o tubature costantemente monitorati attraverso l'utilizzo di sensori. L'obiettivo di SWAMP è proprio quello di fornire una soluzione implementativa per questo sistema. Le modalità di distribuzione dell'acqua sono svariate e principalmente dipendono dalla regione o paese di applicazione.

È tuttavia necessario che le risorse idriche vengano sempre gestite da un'autorità centrale, in modo da poter garantire l'esatta quantità di acqua in base alle necessità del proprio appezzamento di terra.
SWAMP sarà implementato in quattro installazioni pilota che verranno realizzate in Italia, nella zona di Reggio Emilia, in Spagna e Brasile.
Il pool di aziende che hanno preso parte a questo progetto è composto da cinque partner brasiliani e cinque europei, tra i quali l'Alma Mater con i Dipartimenti di Scienze e Tecnologie Agro-alimentari, di Ingegneria Civile, Chimica, Ambientale e dei Materiali e l'Advanced Research Center on Electronic Systems (ARCES).
La suddivisione dei ruoli all'interno di questo progetto prevede che il Dipartimento di Scienze e Tecnologie Agro-alimentari si occupi di sviluppare modelli matematici per la stima del fabbisogno della pianta; che il Dipartimento di Ingegneria Civile, Chimica, Ambientale e dei Materiali sviluppi un modello di ottimizzazione per la gestione del sistema di distribuzione acqua irrigua del Consorzio di bonifica dell'Emilia centrale; che infine L' Unibo Advanced Research Center on Electronic System, coordinandosi con gli altri due dipartimenti, abbia il compito di sviluppare sensori, droni, applicazioni, la piattaforma IoT e algoritmi di volo autonomo.

1.2 Area di operazione e obiettivi del progetto di tesi

Riassumendo il capitolo precedente, SWAMP può essere suddiviso su due rami di progetto. Il primo riguarda il miglioramento del sistema di distribuzione idrico, ovvero tutto l'impianto di apertura e chiusura dei canali di distribuzione dell'acqua. Il secondo, invece, prevede il miglioramento della gestione vera e propria della risorsa idrica, riducendo gli sprechi sul singolo campo conoscendo il fabbisogno del singolo terreno.
Il progetto illustrato nel presente elaborato, si colloca all'interno del secondo ramo, e punta a risolvere un grosso problema legato al conteggio delle quantità d'acqua per irrigare i campi effettuato in bonifica.

In pratica, attualmente, le quantità di acqua a disposizione di ogni singolo appezzamento vengono calcolate facendo riferimento al consiglio irriguo elaborato con Irrinet [4] o altri sistemi di stima dei fabbisogni irrigui e/o alle dotazioni agronomiche medie per ettaro, i quali tengono conto:

- Della coltura, e quindi delle diverse esigenze;
- Del metodo irriguo, che comporta una diversa efficienza nell'uso aziendale dell'acqua e incide, in relazione al sistema irriguo consortile, sul rendimento della rete consortile stessa;
- Del periodo di adacquamento, in relazione alla fase fenologica delle piante;
- Della pedologia del suolo, per cui, in relazione al tipo di suolo, può essere necessario utilizzare maggiori o minori dotazioni idriche.

Per fornire la quantità d'acqua stimata da questi sistemi, l'addetto stima a sua volta un tempo di irrigazione approssimato che tiene conto della portata della conduttura; ma tale metodo fornisce un conteggio impreciso e superficiale della quantità di acqua da rilasciare: spesso la quantità di acqua rilasciata supera infatti anche del 20% la quantità inizialmente stimata [5].

È quindi necessario andare a ridurre questi sprechi raffinando il conteggio, e ciò sarebbe possibile mediante l'introduzione di un misuratore di flusso, in grado di conteggiare l'esatta quantità con una risoluzione il più piccola possibile, eliminando la maggior parte dell'errore umano. I dati relativi al conteggio dovranno poi essere inviati alla piattaforma di monitoraggio. In aggiunta a tutto questo, vi è da tenere in considerazione un problema di non lieve conto: il consumo energetico. Infatti, questo misuratore di flusso dovrà essere in grado di "sopravvivere" un'intera stagione di irrigazione venendo alimentato unicamente attraverso una serie di tre batterie stilo (AA).

Ad aggravare la situazione, interviene il problema delle condizioni ambientali in cui il nodo sensore dovrà operare. L'umidità e il diretto contatto con l'acqua infatti, oltre a mettere a rischio il corretto funzionamento del sensore, porta ad una riduzione della durata delle batterie, comportando la necessità di dotare il nodo sensore di un isolamento dall'ambiente esterno, eliminando ogni possibilità di entrare a contatto con esso.

Infine, come ultimo requisito, vi è la necessità di trasmettere via radiofrequenza a distanze dell'ordine di qualche centinaio di metri; tale distanza è considerata in ragione delle dimensioni che possono avere i campi agricoli, ma ciò incide negativamente sulle condizioni di risparmio energetico richieste.

Suddetto dispositivo è stato interamente progettato in questa tesi. Dal punto di vista elettronico sono stati prodotti gli schematici ed il circuito stampato (PCB). Lato software ci si è occupati dello sviluppo del firmware del microcontrollore, in modo tale che il dispositivo fosse in grado di registrare il flusso d'acqua ed inviarne i relativi dati. Per concludere, sul fronte meccanico, si è realizzato un prototipo pienamente funzionante ed idoneo all'ambiente di operazione (impermeabile), ovvero che consentisse di effettuare, nel breve periodo, test su campo realistici in grado di confermare l'affidabilità del prodotto.

D'ora in poi questo misuratore di flusso verrà nominato "Flow Meter".

2. Stato dell'arte

2.1 IoT, definizione ed applicazioni

Oggigiorno la domanda di sviluppo di applicazioni direttamente connesse ad Internet è molto alta.

Per questo motivo le tecnologie IoT stanno largamente prendendo piede su scala mondiale, dando così vita ad applicativi sempre più complessi e con funzionalità sempre più ampie. Quando si fa riferimento al mondo dell'IoT, si vuole fondamentalmente fare riferimento ad una rete in cui la maggior parte degli oggetti fisici sono connessi a Internet tramite dispositivi di rete o router scambiandosi in continuazione dati.

Una rete di dispositivi IoT consente agli oggetti di essere controllati da remoto attraverso la complessa infrastruttura di rete sulla quale sono appoggiati. Questa tecnica ha anche una funzione di controllo autonomo tramite la quale qualsiasi dispositivo può eseguire operazioni in base ai dati acquisiti dall'ambiente che lo circonda senza alcuna interazione umana.

Per "Cose" nella definizione di IoT (Internet of Things) si intende la combinazione di hardware, software, dati e servizi in riferimento ad una vasta gamma di dispositivi embedded che integrano al loro interno una parte di acquisizione (sensori), una componente di elaborazione (CPU, MCU) ed infine un modulo di trasmissione dati (WiFi, Bluetooth, Ethernet etc...). La versatilità dell'IoT è diventata molto popolare negli ultimi anni grazie a molteplici vantaggi che l'uso di questa tecnologia fornisce. Secondo i dati riportati dal Mckinsey Global Institute, entro il 2025 le attività dell'IoT raggiungeranno i 11.1 trilioni di ricavi [6].

Le principali applicazioni che incorporano dispositivi IoT operano sulle seguenti aree:

1) Domotica: l'ambiente domotico è l'esempio, forse più di tutti, direttamente riconducibile al mondo IoT. Si pensi solamente alle smart plug, ovvero alle prese di corrente il cui stato di accensione e spegnimento può essere controllato da remoto tramite un applicativo per smartphone; o si pensi ancora

al termostato Nest di Google, che apprende automaticamente le routine delle persone e regola di conseguenza la temperatura sulla base delle loro attività;

2) Gestione energetica e delle risorse: l'infrastruttura di misurazione avanzata delle singole risorse energetiche e non, è l'esempio principale in questo settore. In [7] viene descritta una piattaforma interamente dedicata a questi scopi;

3) Assistenza medica e sanitaria: il monitoraggio remoto della salute e il sistema di notifica di emergenza sono esempi di IOT in campo medico. In tale ambito si annovera l'Health Monitor, il cui utilizzo è principalmente quello di assistere il paziente che non può raggiungere il medico, consentendo a quest'ultimo di ottenere i dati vitali del paziente come ECG, frequenza cardiaca, frequenza respiratoria, temperatura della pelle, postura del corpo, rilevamento di cadute e letture di attività da remoto. Ne è un esempio HEMAN [8], un sistema di assistenza sanitaria elettronica basato su IoT per la telemedicina.

4) Trasporto: ne è un esempio il sistema di pagamento automatico del pedaggio autostradale basato sull'identificazione di un codice RFID posizionato sul veicolo, come descritto in [9].

5) Smart Cities: ci sono vari grandi progetti in corso nel mondo. Songdo (Corea del Sud) è la prima città nel suo genere ad essere completamente cablata per supportare qualunque tipologia di oggetto smart. Tutto in questa città è pianificato per essere interconnesso e trasformato in un flusso di dati monitorato da una serie di computer senza alcuna interazione umana. Altro esempio molto celebre è la costruzione, da parte della società francese Sigfox, di una rete dati wireless narrow band nell'area della baia di San Francisco iniziata nel 2014; oppure in America, da Waterways a New York City è presente un sofisticato impianto di monitoraggio delle navi che consente di mettere le stesse in costante comunicazione ed ottenere informazioni circa gli eventi su tutta la tratta 24/7. Queste sono quindi alcune delle grandi applicazioni che l'IoT è in grado di svolgere. Nel prossimo futuro è previsto che un gran numero di dispositivi verranno connessi a Internet e formeranno una sofisticata struttura di interconnessione, interazione e monitoraggio in tutto il mondo.

2.2 Tecnologie LP-WAN

Nel momento in cui si vuole creare una infrastruttura composta da così tanti dispositivi che devono essere interconnessi tra loro, è obbligatorio chiedersi quale tipo di collegamento sia il più adattato. In realtà non vi è una soluzione definitiva a questa domanda, ma solamente un gran numero di proposte.

Tendenzialmente il concetto alla base è il seguente: se il dispositivo da connettere è fisso, ovvero non verrà mai spostato, ed è necessario che esso trasmetta i propri dati in tempo reale, e che questi non vengano persi, allora sarà opportuno adottare una soluzione cablata, ove possibile. Ciò non significa tuttavia che tale soluzione sia sempre la più consona alla situazione. Vi possono infatti essere luoghi in cui cablare una linea di comunicazione porterebbe, oltre che a problematiche infrastrutturali, anche un incremento dei costi esponenziale. Per queste motivazioni l'alternativa alla soluzione cablata risulta essere, come di solito avviene, la trasmissione a radiofrequenza. In particolare, nel mondo dell'IoT, la trasmissione di informazioni deve poter fornire una soluzione efficace ai seguenti problemi:

1) Interconnettere ed identificare milioni di dispostivi contemporaneamente, supportando meccanismi di connessione e disconnessione rapidi.

   Pensiamo a tutti i dispositivi che potrebbero essere connessi ad una rete in una città: ad esempio telecamere, controllo di accessi stradali e non, dispositivi di monitoraggio qualità dell'aria, dispositivi di tracciamento dei mezzi pubblici etc..

   Immaginiamo una città delle dimensioni di Milano, non è difficile pensare che il numero di dispositivi IoT interconnessi possa facilmente superare la soglia delle 100 mila unità.

   Identificare e mettere in comunicazione 100 mila dispositivi non è per niente facile, soprattutto se si vuole optare per costi ridotti dell'infrastruttura.

2) La copertura massima supportabile.

   Non è insolito infatti che un dispositivo debba raggiungerne un altro posizionato a notevole distanza. Un esempio chiave è il campo agricolo completamente monitorato da sensori

posizionati ai vertici del poligono. Il gateway di raccolta dati è difficilmente posizionato esattamente al centro del campo, mentre è invece più frequente che venga installato all'interno della struttura principale della azienda agricola. In questo caso i sensori dovranno essere in grado di trasmettere i dati di monitoraggio del suolo (per esempio) al gateway, percorrendo quindi lunghe distanze, anche nell'ordine di diversi chilometri.

3) Il consumo energetico. Chiaramente un dispositivo fisso piantato nel terreno, magari a diversi metri sotto il livello del suolo, può risultare scomodo da estrarre, per esempio per cambiare la batteria. Di conseguenza minore è il consumo energetico di questo, minori saranno i costi di manutenzione da affrontare per mantenerlo funzionante nel tempo.
4) Comunicazione tra molteplici dispositivi ad un costo ridotto. Chiaramente implementare una infrastruttura ad hoc garantirebbe ai singoli dispositivi una maggiore efficienza, sia dal punto di vista della velocità ed affidabilità della connessione, sia dal punto di vista energetico. Non sempre, però, il rapporto che lega questa efficienza al costo dell'infrastruttura necessaria per raggiungere i migliori risultati è lineare.

   Spesso quindi è necessario raggiungere un compromesso, che miri a raggiungere i migliori risultati riducendo al minimo i costi infrastrutturali ed hardware dei singoli dispositivi.

L'unione di questi quattro punti ha portato alla creazione di una categoria di trasmissione a radiofrequenza denominata LP-WAN (low-power wide-area network)[10]. Come si deduce dalla definizione stessa, essa possiede esattamente tutte le proprietà precedentemente descritte, ovvero:

- Lungo raggio: il raggio operativo della tecnologia LPWAN varia da pochi chilometri nelle aree urbane a oltre 10 km in contesti rurali. Può anche consentire un'efficace comunicazione dei dati in luoghi interni e sotterranei precedentemente non praticabili;
- Bassa potenza: ottimizzati per il consumo energetico, i ricetrasmettitori LPWAN possono funzionare fino a 20 anni con batterie piccole ed economiche;

- Basso costo: i protocolli semplificati e leggeri di LPWAN riducono la complessità nella progettazione dell'hardware e riducono i costi dei dispositivi. La sua lunga portata combinata con una topologia a stella riduce i costosi requisiti di infrastruttura e l'uso di bande senza licenza riduce i costi di rete.

I dispositivi LPWAN tendono inoltre ad essere limitati nell'utilizzo della larghezza di banda, limitando la quantità di tempo in cui un dispositivo può trasmettere su determinate frequenze (duty cycle di solito espressi come percentuale di tempo all'ora che il dispositivo è autorizzato a trasmettere).

Mentre normalmente tutte le reti con dispositivi limitati devono bilanciare il consumo di energia e/o la durata della batteria con i costi e la larghezza di banda utilizzata, le LPWAN danno priorità alla potenza di trasmissione ed al basso costo, accettando, però, severi vincoli di larghezza di banda e di duty cycle.

Questo consente dunque di supportare collegamenti radio chilometrici, come richiesto in alcune applicazioni descritte in precedenza.

Nel corso degli anni sono stati sviluppati diversi standard che descrivono soluzioni congrue con la categoria LP-WAN, che hanno subito mostrato un enorme potenziale e creato un grande interesse industriale. Stiamo parlando delle principali tecnologie attualmente presenti sul mercato: LoRa[11], Wi-SUN[12], SIGFOX[13] and NBIoT[14].

Ognuna di esse ha i suoi pro e contro, per quanto riguarda sicurezza, copertura, prestazioni in condizioni di copertura non in linea d'aria, topologia di rete, modello di business, complessità di implementazione ed operazione, velocità dei dati per i costi di up e downlink e molti altri aspetti. Tra queste tecnologie, le più promettenti e che possono essere utilizzate nell'ISM (industriale, scientifica e medica) e in altre bande di frequenza ad uso gratuito, sono LoRa e Wi-SUN.

LoRa

LoRa è principalmente conosciuto per essere sufficientemente soddisfacente su tutti i fronti, ovvero consente di comunicare a distanze elevate, specialmente in condizioni di vista tra un dispositivo e l'altro.

Il lungo raggio di comunicazione viene raggiunto per mezzo del guadagno di elaborazione fornito dallo Chirp Spread Spectrum (CSS), che utilizza una larghezza di banda maggiore del minimo necessario per trasmettere il segnale. Uno svantaggio invece è la limitazione della velocità di trasmissione dei dati, sempre inferiore a 40 kbps quando si utilizza CSS; tuttavia, è possibile attivare LoRA in modalità GFSK in modo da ottenere una velocità massima di 50 kbps.

WI-SUN

La WI-SUN Alliance promuove l'implementazione di reti interoperabili di Smart Utility wireless mediante:

1) L'adozione di standard industriali aperti, come definiti dalle organizzazioni di sviluppo di standard internazionali e regionali;

2) La fornitura di input al processo di standardizzazione;

3) I programmi di certificazione di conformità e di interoperabilità.

Il livello fisico adottato dal profilo WI-SUN FAN (Field Area Network) è IEEE 802.15.4g, nient'altro che una versione modificata dello standard IEEE 802.15.4 che si rivolge alle esigenze di comunicazione speciali tra reti di Smart Metering Utility (SUN).

I SUN svolgono un ruolo chiave nel contesto delle reti intelligenti: consentono a più applicazioni di operare su risorse di rete condivise, supportando comunicazioni bidirezionali tra i dispositivi di misurazione e controllo di un sistema di utilità, spesso fornendo una grande area di copertura.

Tradizionalmente IEEE 802.15.4 fu proposto appositamente per reti wireless personali (WPAN) a bassa velocità (LR), che avessero lo scopo di trasmettere informazioni su distanze relativamente brevi con un'infrastruttura semplice fornendo una velocità dati limitata a 250 kbps.

IEEE 802.15.4g, invece, è progettato per raggiungere velocità di trasmissione dati da 2,4 kbps a 800 kbps e funzionamento in diverse bande di frequenza, da 169 MHz a 2,4 GHz. Attualmente, il profilo FAN di WI-SUN adotta solo la modulazione GFSK, ma è previsto l'uso di altre modulazioni.

Sigfox

Questa tecnologia impiega il phase shifting keying differenziale binario (DBPSK) e il Gaussian frequency shift keying (GFSK), che consente la comunicazione utilizzando la banda radio ISM industriale, scientifica e medica sulle frequenze 868 MHz in Europa e 902 MHz negli Stati Uniti.

Essa utilizza un segnale ad ampio raggio che passa liberamente attraverso oggetti solidi, chiamato "Ultra Narrowband" necessitando di poca energia.

La rete Sigfox si basa su una topologia a stella a un hop. Il segnale può anche essere usato per coprire facilmente ampie aree e per raggiungere oggetti sotterranei.

Narrowband Internet of Things (NB-IoT)

Questo è uno standard di tecnologia radio LPWAN sviluppato da 3GPP per connettere un'ampia gamma di dispositivi tramite rete cellulare e fornire loro determinati servizi.

NB-IoT si concentra in particolare a fornire una copertura indoor, a basso costo, a basso consumo energetico, data un'alta densità di nodi connessi. NB-IoT utilizza un sottoinsieme dello standard LTE, ma limita la larghezza di banda a una singola banda stretta di 200 kHz. Utilizza la modulazione Orthogonal Frequency-Division Multiplexing (OFDM) per la comunicazione in downlink e Single-carrier frequency-division multiple access (SC-FDMA) per le comunicazioni in uplink.

2.3 IoT ed agricoltura di precisione

Tornando ora sui binari dell'argomento principale di questo documento, l'IoT sta tutt'ora rivoluzionando notevolmente anche il mondo dell'agricoltura, apportando migliorie ed innovazioni tecnologiche finalizzate al raggiungimento di risultati ottimali sia sul fronte della produttività sia su quello del risparmio economico e di risorse.

L'Internet of Things (IoT) sta aiutando a cambiare il modo in cui gli agricoltori lavorano attraverso l'agricoltura di precisione, un concetto di gestione dell'azienda agricola che utilizza sensori, dati e comunicazioni di rete per adattare il sistema agricolo alle condizioni specifiche di ciascun campo.

Il risultato è un sistema più efficiente che promuove la crescita sostenibile riducendo i costi.

Affinché l'agricoltura di precisione sia efficace, vari fattori dovrebbero essere accuratamente considerati. Alcuni di questi sono lo stato appropriato dei macchinari utilizzati, il comportamento delle colture, la luce, la temperatura, l'umidità, la composizione del suolo e il tempo.

È necessaria un'applicazione efficace dei molteplici fattori per influenzare l'effettivo rendimento ottenuto da un particolare pezzo di terra: infatti, l'efficienza in agricoltura dipende dall'accuratezza delle decisioni prese, sulla base di informazioni ottenute anche utilizzando satelliti, droni, GPS, API e sensori.

Con l'IoT, è possibile raccogliere dati in tempo reale allo scopo di aiutare gli agricoltori a prendere decisioni intelligenti e ad hoc, prendendo in considerazione le problematiche osservate o le opportunità fornite dai singoli terreni agricoli.

I dati statistici ottenuti da qualsiasi terreno agricolo sono importanti per effettuare analisi storiche del lotto di terra. Un simile approccio può essere utile per formulare previsioni sui vari tipi di coltura e sul livello di resa da aspettarsi, in modo tale da migliorare la pianificazione. Pertanto, l'agricoltura intelligente richiede un'efficace interconnessione di oggetti come telecamere e sensori per l'estrazione di dati locali che possono essere utilizzati nei Big Data, nonché strumenti di analisi il cui scopo è archiviare enormi quantità di dati, che, elaborando e trasformando le informazioni, apportano un miglioramento alla produttività agricola. Tutto questo processo, se svolto con accuratezza, può consentire di integrare tutti i servizi di ogni singolo componente IoT, formando così un coordinamento strutturale ben organizzato.

2.4 IoT-based Water metering

Uno degli aspetti di cui si occupa l'agricoltura di precisione è il controllo ed il conteggio dell'acqua utilizzata durante una irrigazione, al fine di evitare sprechi ed eccessi nell'utilizzo di una delle risorse agricole più importanti che, in determinati periodi, può arrivare a scarseggiare.

Negli ultimi anni sono di conseguenza nate diverse soluzioni che cercano di affrontare questo problema effettuando misurazioni delle quantità di acqua utilizzate durante le singole irrigazioni. Conoscendo la tipologia di terreno e le sue proprietà, è possibile infatti stimare l'esatta quantità di acqua necessaria per eseguire un'irrigazione perfetta evitando sprechi. Ciò che differenzia i sistemi di questo tipo già esistenti, è la tecnologia utilizzata per la trasmissione dei dati al centro di archiviazione.

In [15] viene proposto un contatore implementato con una scheda Arduino, ove si prevede che ogni nodo sia connesso ad un router attraverso un cavo Ethernet. Ciò rende però necessaria una cablatura dell'infrastruttura di comunicazione; l'autore prevede, ad ogni modo, che questa soluzione possa essere ampliata utilizzando shield WiFi oppure GSM/3G/4G.

Chiaramente questa soluzione non rispetta al 100% la politica Low Power richiesta per ambienti outdoor, ma si concentra a fornire un servizio che massimizza l'affidabilità sulla trasmissione dati.

Diversa è invece la soluzione proposta in [16], ove vengono messi al primo posto i principi di Low Power e Long Range. Proprio grazie all'adozione di una tecnologia di trasmissione LP-WAN, in questo caso NB-IOT, l'autore ha potuto implementare un sistema funzionante su grandi appezzamenti terrieri che fosse controllabile da remoto attraverso il protocollo di comunicazione CoAP basato sul concetto di risorsa. Vi sono poi altre soluzioni come la [17], che optano per trasmissione a corto raggio, come ZigBee, per garantire una maggiore durata della batteria, perdendo però la possibilità di coprire grandi aree con un solo dispositivo. Probabilmente la costruzione dell'infrastruttura necessaria per trasmettere il segnale lungo un campo agricolo necessiterebbe di costi maggiori di quelli affrontati per implementare un dispositivo con al suo interno un modulo di trasmissione LP-WAN.

3. Progettazione e Sviluppo

3.1 Componenti utilizzati

Lo sviluppo del Flow Meter ha necessitato di diversi componenti per soddisfare i requisiti che il progetto imponeva. In particolare, ogni scelta è stata volta a mantenere un equilibrio tra:

1) Conteggio dei litri d'acqua che attraversano la conduttura con errore pari alla sensibilità dello strumento di acquisizione;
2) Impermeabilità del circuito;
3) Trasmissione real-time a lungo raggio dei dati acquisiti ed immagazzinamento su piattaforma web;
4) Basso consumo energetico che consenta al dispositivo di affrontare almeno una stagione intera di irrigazione.

Per superare questi scogli implementativi, sono state scelte tecnologie che fossero da un lato pienamente conformi a quanto richiesto e dall'altro facilmente reperibili ed il cui costo fosse affrontabile per un eventuale commercializzazione.

Il primo componente preso in esame è stato il sistema contenente il sensore di conteggio del flusso d'acqua (Figura 3.1.1).

Figura 3.1.1 – Sistema di acquisizione flusso d'acqua

Tale sistema ci è stato fornito dal Dipartimento di Idraulica dell'Università di Bologna, così strutturato: dall'esterno si presenta come una normale conduttura di lunghezza 50 cm e di diametro 25 cm perfettamente allacciabile ai tubi di canalizzazione dell'acqua utilizzati nell'irrigazione, con dimensioni tali da consentire una portata oraria d'acqua pari a 160 m^3/h.

All'interno della conduttura è presente un mulinello che, muovendosi, incrementa il contatore analogico posizionato nella parte superiore. Il contatore analogico è composto da tre indicatori (due a lancette, e uno a cifre rotanti), collegati tra loro attraverso un sistema di ingranaggi a riduzione. I due indicatori a lancette hanno rispettivamente come moltiplicatori 0.01 m^3 e 0.1 m^3 (valori da 0 a 9) mentre il restante indica i m^3 transitati all'interno della conduttura.

Il dato registrato in questi contatori analogici è sempre cumulativo e risulta così molto complesso reimpostare il contatore: per fare ciò, sarebbe necessario ruotare la pala interna in senso opposto al flusso dell'acqua fino a che tutto il sistema analogico di conteggio non torni sullo zero.

Il sistema meccanico non è però completamente isolato dall'acqua e questo è infatti il motivo per cui non può essere utilizzato da una scheda elettronica. Al fine di consentire la lettura del flusso d'acqua passante da parte dell'apparato elettronico, è stato inserito a fianco del contatore analogico con moltiplicatore 0.01 m^3 un Reed Switch completamente impermeabilizzato con silicone e cavo di plastica IP68 (Figura 3.1.2)

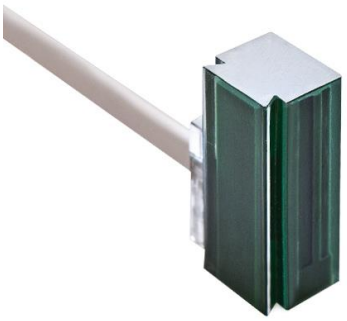


Figura 3.1.2 – Reed Switch impermeabilizzato

Un Reed Switch è un interruttore a lamina (normalmente aperto) che si chiude in presenza di un campo magnetico.

Nella forma più semplice è costituito da due lamine, realizzate con materiale ferromagnetico (una lega di ferro-nichel), parzialmente sovrapposte e separate tra loro di qualche decimo di millimetro. Sulle lamine contrapposte sono riportati dei contatti (generalmente in oro diffuso).

Le lamine vengono sigillate all'interno di un piccolo contenitore di vetro riempito di gas inerte (azoto o argon). Le estremità delle lamine (opposte ai contatti) fuoriescono dal contenitore e costituiscono i terminali del contatto. In presenza di un campo magnetico le lamine diventano sede di flusso magnetico e sulle estremità si formano poli di segno opposto che tendono ad attrarsi. Se il campo magnetico è sufficientemente forte (100-200 amperspire), la forza d'attrazione vince la resistenza a flessione, e le lamine, attraendosi, chiuderanno il contatto.

Nel caso di specie è proprio il contatore a lancetta con moltiplicatore 0.01 m^3 che, quando indica valori compresi tra 4-6, avvicina un piccolo magnete al Reed Switch forzando la chiusura del circuito: in questo modo la conduttura genera un'onda quadra ad ogni giro della lancetta relativa ai 0.01 m^3.

Considerando la portata massima del tubo di 160 m^3/h, ed il fatto che venga generato un impulso ogni 0.1 m^3 (100 litri) di acqua, la frequenza massima che il contatore può produrre è

$$Fs = \frac{160\frac{m^3}{h}}{0.1m^3 * 3600s} = 0.44\ Hz$$

Il duty cycle del segnale è invece

$$DCs = \frac{6-4}{10} = 20\%$$

Questo garantisce che in condizioni di portata massima il circuito rimanga chiuso per un quinto del tempo. Il Reed Switch per poter essere letto necessita di un collegamento diretto con un'unità di elaborazione dati, ossia un microcontrollore. Ricordiamo che vi sono ancora due requisiti da soddisfare: basso consumo energetico e trasmissione dati a lungo raggio.

La soluzione ambivalente è l'introduzione della board B-L072Z-LRWAN1.

Alla base di questa scheda vi è l'utilizzo del modulo CMWX1ZZABZ-091 [18] prodotto da Murata.

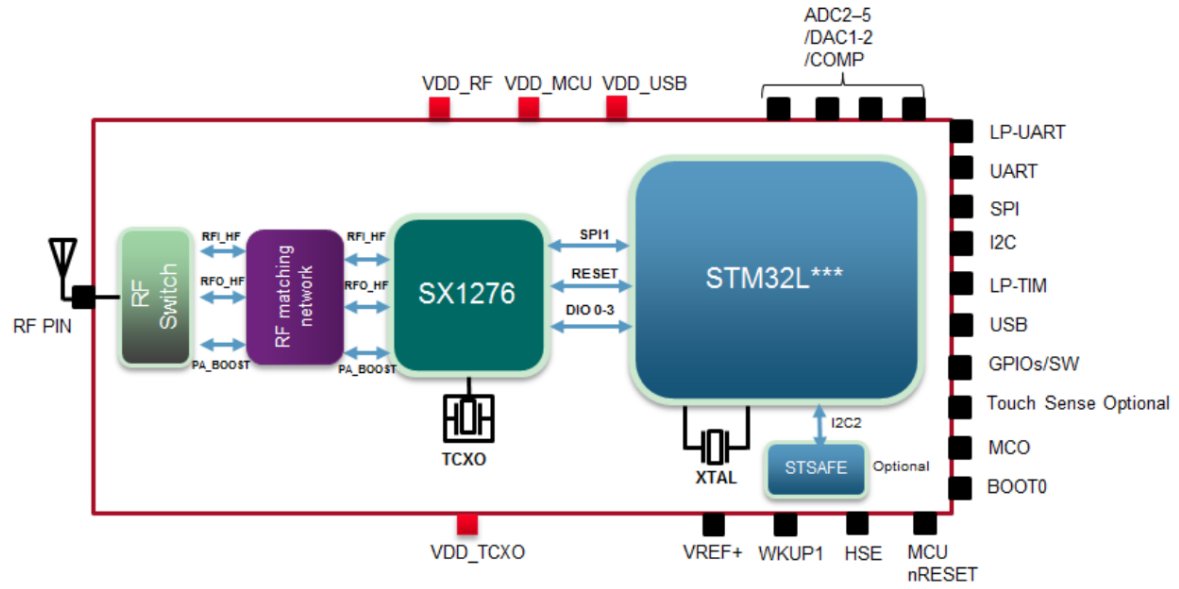


Figura 3.1.3 – Schema modulo CMWX1ZZABZ-091

Come è possibile notare dalla Figura 3.1.3 il modulo è composto da 4 componenti.

Da una parte abbiamo il microcontrollore STM32L072CZ appartenente alla serie a basso consumo energetico L0, dall'altra il Transceiver SX1276, la RF matching network e lo switch RF, i quali identificano la componente di trasmissione dati a radio frequenza.

La tecnologia sulla quale si basa questa trasmissione è denominata LoRa ed opera nell'intervallo di frequenze 868-930 MHz. La normale potenza di trasmissione del modulo è +14 dBm ma, utilizzando un'amplificazione passiva per mezzo di una antenna, è possibile incrementarne il guadagno a +20dBm per applicazioni che richiedono un lungo raggio di trasmissione o per applicazioni indoor soggette a forti attenuazioni dovute a riflessioni ed alla presenza di pareti.

Per quanto riguarda invece la sola MCU, questa possiede una architettura RM Cortex-M0+ a 32 bit con un clock a 32 MHz, 192 KB di memoria flash and 20 KB di RAM.

In aggiunta, un STSAFE Secure Element opzionale può essere incorporato alla MCU in modo da aumentare il livello di sicurezza all'interno della rete.

3.2 Comunicazione RF: LoRa e LoRaWAN

Per la trasmissione in radio frequenza, come già anticipato in precedenza, viene utilizzato LoRa.

LoRa è l'abbreviazione di long range, ed è una tecnica di modulazione a spettro diffuso derivata dalla tecnologia chirp spread spectrum (CSS).

In particolare, CSS è una tecnica a spettro diffuso che usa impulsi di chirp modulati in frequenza lineare a banda larga per codificare informazioni.

Un chirp è un segnale sinusoidale in cui la frequenza aumenta o diminuisce con l'avanzare del tempo, spesso seguendo un'espressione polinomiale che mette in relazione tempo e frequenza.

In questo modo si trasmette un segnale di base su una banda più ampia ottenendo un aumento della resistenza al rumore.

Una caratteristica della modulazione CSS è l'utilizzo dello Spreading Factor (SF) che, in base alla Bandwidth (BW), ovvero la larghezza di banda, permette di regolare il Bit Rate (Rb), ovvero la velocità in bit, e quindi la durata della trasmissione e di conseguenza anche i consumi di energia tramite la seguente formula:

$$R_b = SF * \frac{1}{\left[\frac{2^{SF}}{Bw}\right]} \; bits/sec$$

Durante la comunicazione le frequenze utilizzate ed il Data-Rate vengono modificate in base alle esigenze e alla distanza utilizzando il meccanismo dell'Adaptive Data Rate (ADR) che permette di aumentare l'efficienza energetica.

Inoltre, il canale selezionato viene cambiato in maniera casuale ad ogni nuova comunicazione, in modo da rendere la rete meno soggetta alle interferenze e, se il nodo è servito da più gateway, riduce la potenza di trasmissione in modo che il segnale sia ricevuto da meno gateway.

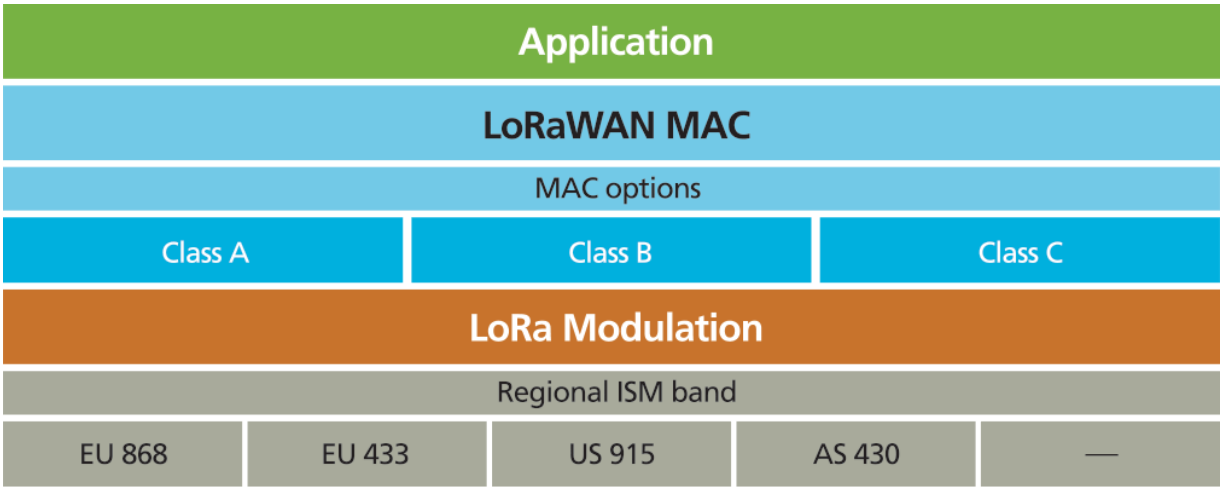


Figura 3.2.1 – Stack di una applicazione che utilizza LoRa

Intendendo LoRa come metodo di modulazione (livello fisico), è necessario completarlo introducendo un livello MAC, in grado di garantire un accesso al mezzo in modo sincrono tra i diversi dispositivi all'interno della rete e di fornire loro un indirizzo univoco. Suddetto compito è appunto assegnato a LoRaWAN (Figura 3.2.1).

L'architettura di una rete LoRaWAN è strutturata su una topologia a stella in cui ciascun nodo finale comunica con più gateway. I Gateway, a loro volta, incapsulano il frame ricevuto in un pacchetto TCP/IP inviandolo successivamente ai server di accumulo ed analisi dei dati Online. La tecnologia LoRa prevede una comunicazione di tipo bidirezionale, ma la trasmissione da nodo a gateway o messaggio di Uplink è quella maggiormente utilizzata.

Risulta invece meno frequente quella da gateway a nodo o messaggio di Downlink, questo perché tendenzialmente lo scopo dei nodi è quello di raccogliere dati per poi mandarli al Network Server e quindi all'Application Server. Gli elementi principali della rete sono quindi: i nodi, i gateway, il Network Server e l'Application Server.

I nodi hanno la funzione di produrre i dati acquisendoli attraverso sensori, per poi successivamente inviare messaggi in Uplink a tutti i gateway nel loro raggio di trasmissione in modalità broadcast.

I gateway hanno la funzione di interfaccia radio-TCP/IP, ricevendo i messaggi dai nodi ed inoltrandoli al Network Server aggiungendo informazioni riguardanti la qualità della comunicazione attraverso una connessione IP instradata su Ethernet, Wi-Fi o 3G.

Il Network Server si occupa della gestione dei messaggi di Uplink duplicati e della selezione del miglior gateway da utilizzare nel caso debba essere inviato un messaggio di Downlink al nodo; inoltre il Network Server gestisce anche la velocità di trasmissione dei nodi tramite il meccanismo dell'ADR (Adaptive Data Rate) per massimizzare la capacità della rete analizzando il rapporto segnale-rumore (SNR) ed il numero di gateway che hanno ricevuto ciascun messaggio di Uplink.

Di seguito uno schema dell'architettura di rete LoRaWAN (Figura 3.2.2).

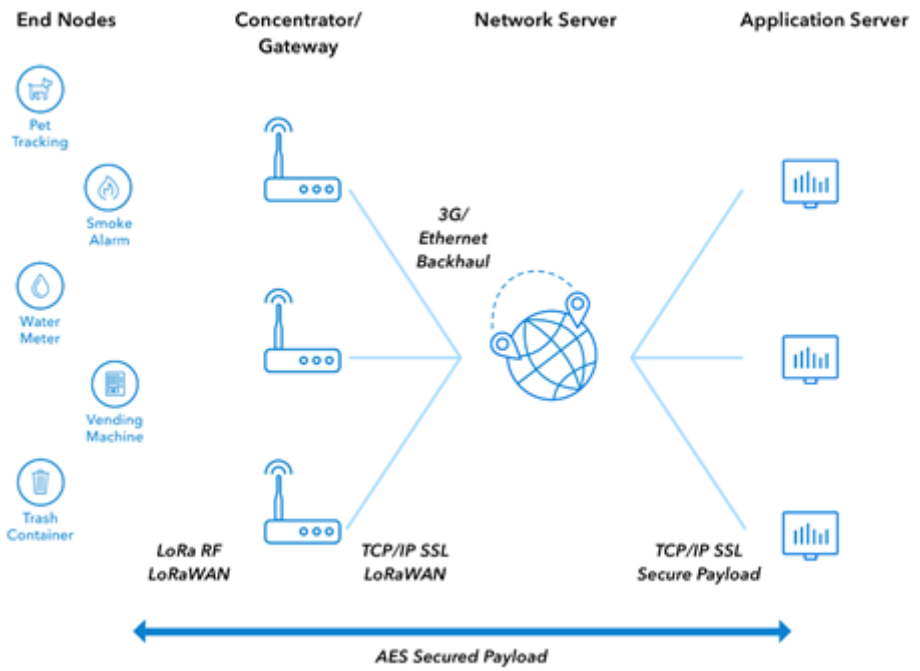


Figura 3.2.2 – Esempio di rete LoRaWAN

Originariamente LoRaWAN è stato progettato basandosi sul protocollo ALOHA puro, e successivamente, per poter soddisfare necessità di diversi scenari applicativi, LoRaWAN è stato rivisitato in modo da garantire maggiore elasticità nella trasmissione.

Le specifiche di LoRaWAN definiscono tre tipologie di nodi: nodi di classe A, di classe B e di classe C. Tutti i dispositivi compatibili LoRaWAN devono implementare la classe A, mentre le classi B e C rappresentano delle estensioni.

Nodi di classe A

La classe A appresenta la configurazione principale, normalmente utilizzata dai nodi.

In classe A i nodi supportano la comunicazione bidirezionale con il gateway, ma questa viene sempre inizializzata dai nodi in modo asincrono.

I messaggi in Uplink, dal nodo al gateway, possono essere inviati in qualsiasi momento.

A seguito di un messaggio di Uplink il nodo apre due finestre di ricezione.

Il Network server può rispondere con un ACK, se richiesto dal nodo, tramite un gateway con un messaggio in Downlink in una delle due finestre.

Solitamente la prima finestra è aperta sullo stesso canale utilizzato nella trasmissione in Uplink, mentre la seconda finestra viene aperta su un canale differente, accordato in precedenza con il Network Server, per migliorare la resistenza alle interferenze. Questa classe è la più efficiente dal punto di vista energetico ed è maggiormente utilizzata da nodi in cui è stato implementato un sistema di inattività finalizzato al risparmio energetico ed in cui le comunicazioni in Uplink sono le più frequenti, come ad esempio nei sensori.

Nodi di classe B

La classe B estende le funzionalità della classe A, implementando finestre di ricezione programmate per i messaggi di Downlink inviati dai Network Server tramite i gateway. Tramite l'utilizzo di segnali temporizzati trasmessi dal gateway, i nodi aprono periodicamente delle finestre di ricezione consentendo quindi la ricezione di messaggi in Downlink indipendentemente dal fatto che siano stati precedentemente inviati messaggi di Uplink.

Questa funzionalità richiede però che i nodi vengano sincronizzati con il Network Server attraverso un meccanismo che sfrutta dei pacchetti beacon trasmessi dai gateway ogni 128 secondi.

All'interno del pacchetto beacon è contenuta l'indicazione di un tempo specifico in cui programmare l'apertura della finestra di ricezione, chiamata ping slot. Questa classe è utilizzata da quei nodi che hanno la necessità di ricevere dei comandi da remoto, come ad esempio gli attuatori.

Nodi in classe C

I nodi in classe C estendono le funzionalità della classe A implementando una finestra di ricezione sempre aperta, a meno che il nodo non stia trasmettendo. Questo permette una comunicazione con bassa latenza, ma va ad aumentare molto il consumo di energia rispetto alla classe A.

Spesso questa classe è utilizzata da nodi che in cui la ricezione di comandi deve rispettare vincoli temporali. L'operatività, di questa classe, si traduce, infatti, in un elevato consumo energetico che rende solitamente necessario alimentare i nodi tramite rete elettrica.

Identificazione

Affinché un nodo possa essere connesso ed identificato all'interno della rete, è necessario che sia provvisto dei seguenti dati.

- L'indirizzo identificativo del device all'interno della rete è denominato DevAddr (Device Address) e si compone di 32 bit, di cui i sette più significativi identificano la rete mentre i rimanenti sono assegnati in arbitrariamente dal Network Server;
- Un identificativo di applicazione denominato AppEUI (Application Identifier) e composto da 64 bit. Questo è l'identificativo esclusivo dedicato al proprietario dell'applicazione a cui è assegnato il dispositivo
- Una chiave di sessione di rete denominata NwkSKEY (Network Session Key). Questa chiave AES-128 bit è specifica per il nodo ed è utilizzata dal Network Server e dal device per generare il MIC (Message Integrity Code) al fine di controllare l'integrità del messaggio e per criptare e decriptare il FRMPayload (Frame Payload).
- Una chiave di sessione per l'applicazione denominata AppSKey (Application Session Key), questa chiave AES-128 bit è specifica per il dispositivo ed è utilizzata dall'Application Server e dal nodo per criptare e decriptare il payload dei messaggi specifici di un'applicazione in modo da creare un canale di comunicazione sicuro end-to-end.

Date queste informazioni, occorre però considerare l'esistenza di due tipologie di attivazione del nodo. La prima è la Over-The-Air Activation (OTAA), in cui i nodi devono effettuare una procedura di join prima di poter scambiare dati con il Network Server. La procedura di join deve essere eseguita nuovamente ogniqualvolta le informazioni riguardo la sessione vengano perse. Qui la chiave AppKey viene utilizzata per generare la chiave di sessione di rete NwkSKEY e la chiave di sessione per l'applicazione AppSKey.

La procedura di join consiste in un messaggio MAC di join request tramite il quale un nodo invia anche il proprio DevEUI e l'AppEUI. Come risposta al messaggio MAC di join request, il Network Server spedisce al nodo un messaggio MAC di join accept inviando il DevAddr del nodo e un valore casuale chiamato AppNonce utilizzato dal nodo per ricavare l'AppSKey e la NwkSKEY.

La seconda è la modalità Activation By Personalization (ABP).

Questa volta i nodi non devono eseguire una procedura di join inviando messaggi MAC request-join accept, come descritto precedentemente, perché l'indirizzo identificativo del device DevAddr, ossia la chiave di sessione di rete NwkSKey e la chiave di sessione per l'applicazione AppSKey, sono memorizzate direttamente nel nodo. In altre parole, in questa modalità il nodo possiede già le informazioni necessarie per associarsi ad una rete LoRaWAN all'interno del proprio firmware.

3.3 Analisi e verifiche pre-progettazione

Prima di potersi addentrare nelle fasi di progettazione, vi è uno step preliminare e necessario da esaminare: è necessario avere il pieno controllo e di conseguenza conoscere alla perfezione il comportamento di ogni singola componente che si dovrà utilizzare.

Una scelta inderogabile nella progettazione del Flow Meter è stata l'utilizzo della conduttura contenente il reed switch fornitoci dal Dipartimento di Idraulica dell'Università di Bologna. Inizialmente non ci è stata fornita nessuna documentazione tecnica dettagliata circa il suo funzionamento, è stato dunque nostro compito effettuare delle verifiche che andassero a definire in modo corretto, ma soprattutto completo, la relazione tra il segnale generato dal reed switch ed il flusso d'acqua transitante all'interno della conduttura.

Tutto portava a pensare che l'elica a mulino interna contenesse in una delle sue pale un magnete in grado di annullare la forza tra lue due lamine del reed switch chiudendo il circuito.

Per poter verificare questa cosa è stato costruito un circuito come segue.

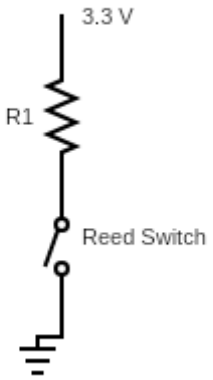


Figura 3.3.1 – Circuito dio lettura stato del reed switch

In sostanza abbiamo collegato il reed switch attraverso una resistenza di pull-up ad una alimentazione 3.3V. Il valore della resistenza è stato scelto in modo da non bruciare il cavo, limitando dunque la corrente a qualche decina di mA.

Utilizzando l'oscilloscopio, abbiamo posto una sonda di lettura della tensione tra il reed switch e massa. In questo modo, alimentando il circuito a 3.3V, leggevamo sul display dell'oscilloscopio 3.3 quando il circuito era aperto, mentre 0.0V quando questo si chiudeva. Così facendo, è stato possibile riconoscere che il reed switch non interagiva direttamente con l'elica ma con i contatori analogici.

Così, in seguito a svariate misurazioni, è stato possibile identificare il segnale generato dal circuito in Figura 3.3.1 con un'onda quadra il cui fronte di discesa corrispondeva alla lancetta in posizione 4 del contatore analogico rappresentante i 0.01 m^3, mentre il fronte di discesa corrispondeva alla lancetta in posizione 6.

Il duty cycle di quest'onda è quindi approssimativamente del 20%.

Spostandoci ora sul fronte dei consumi energetici relativi all'utilizzo di un microcontrollore, occorreva non sottovalutate alcune problematiche appresso indicate.

Infatti, per quanto un microcontrollore possa essere a basso consumo energetico, nel momento in cui questo deve far funzionare le periferiche, esse richiedono un'alimentazione, e se ci si aggiunge un minimo di elaborazione dei dati e trasmissione dell'elaborato, si arriva facilmente ad ottenere un consumo medio di corrente nell'ordine dei mA.

Questo è chiaramente un dato fuori scala per il nostro sistema, in quando siamo intorno a tre ordini di grandezza sopra al consumo medio richiesto dai requisiti.

È stato dunque necessario studiare a fondo la serie L dei microcontrollori ST, in modo da capire se ci fosse qualche meccanismo che venisse incontro alle nostre esigenze di ricercare l'ottimizzazione dell'utilizzo del microcontrollore nell'affrontare i seguenti task:

- Riconoscimento condizione di inizio irrigazione in un tempo dell'ordine di qualche secondo;
- Conteggio della quantità di acqua transitata all'interno della conduttura con un errore massimo pari all'unità di misura minima utilizzata per il conteggio stesso;
- Riconoscimento condizione di fine irrigazione;
- Trasmissione dei dati salvati ed elaborati.

Analizzando il datasheet del prodotto [19] è stato poi possibile intuire come l'impegno della ST Microelectronics sul fronte del Low Power sia quello di garantire le migliori prestazioni sul mercato nel rapporto qualità-prezzo.

La serie L0 è infatti il risultato della combinazione tra un'architettura Arm Cortex-M0+ ed alcune funzionalità convertite in una modalità ultra-low-power. Ciò fa sì che STM32L0 sia perfetto per applicazioni che operano utilizzando un'alimentazione a batteria (come nel nostro caso) e/o energy harvesting. Per fornire la migliore efficienza energetica, STM32L0 offre un ridimensionamento dinamico della tensione, un oscillatore di clock a bassissima potenza e un breve tempo di wake-up.

Le nuove periferiche autonome (tra cui USART, I²C, Controller touch sense) riducono il carico del core portando a un minor numero di riattivazioni della CPU e contribuendo a ridurre i tempi di elaborazione e il consumo di energia.

Per ridurre il costo base totale delle applicazioni, STM32L0 integra comparatori a basso consumo, EEPROM incorporata e un'interfaccia USB priva di cristalli.

Altre funzionalità come ADC a 16 bit (a sovra-campionamento hardware), DAC a doppio canale, Timer ottimizzati, crittografia hardware e periferiche di comunicazione in grado di operare in modalità a bassissima potenza, offrono un giusto compromesso tra integrazione delle funzionalità, prestazioni e consumo energetico ridotto. STM32L0 supporta 3 modalità di risparmio energetico: Sleep, Stop e Standby.

Low power mode	Consumption	CPU	Flash / EEPROM	RAM	DMA & Peripherals	Clock	LCD	RTC
Sleep	41 µA/MHz (Range 1)	No	ON	ON	Active	Any	Available	
	36 µA / MHz (Range 2)							
	35 µA/MHz (Range 3)							
Low power run	8.55 µA (Flash OFF, 32 kHz)	Yes	ON or OFF	ON	Active	MSI	Available	
Low power sleep	4.65 µA (peripherals OFF)	No	OFF	ON	Active	MSI	Available	
Stop with RTC	0.82 µA (1.8 V)	No	OFF	ON	Frozen	LSE, LSI	OFF	ON
	1.0 µA (3 V)							
Stop	415 nA	No	OFF	ON	Frozen	-	OFF	OFF
Standby with RTC	655 nA (3 V)	OFF	OFF	OFF	OFF	LSE	OFF	ON
	845 nA (1.8 V)							
Standby	290 nA	OFF	OFF	OFF	OFF	-	OFF	OFF

Figura 3.3.2 – STM32L0 modalità di funzionamento

Come è mostrato in Figura 3.3.2, ogni modalità di riduzione del consumo energetico disabilita determinate funzionalità al microcontrollore.

La scelta di quale funzionalità utilizzare è stata presa sulla base delle necessità formatosi in fase di progettazione. Nel nostro caso, però, il requisito più stringente dal punto di vista implementativo è stata la possibilità di contare il flusso d'acqua transitante all'interno del tubo. Per affrontare questo problema è stato necessario trovare una soluzione che si comportasse in modo simile alla scrittura in DMA nei registri del microcontrollore. La scelta sulla quale siamo ricaduti, che si è poi dimostrata essere quella più corretta ed allo stesso più completa, includeva l'utilizzo dei timer.

Tutti i microcontrollori moderni sono integrati con moduli timer-contatori, e generalmente vengono utilizzati per generare basi dei tempi, contare gli impulsi, misurare i periodi di tempo delle forme d'onda, generare segnali di modulazione della larghezza di impulso (PWM) e azionare dispositivi esterni e temporizzare eventi speciali.

I timer vengono gestiti in modo diverso da ogni azienda produttrice di silicio; tuttavia, diversamente dalla maggior parte dei microcontrollori a 8 bit che possiedono due o tre timer con funzionalità limitate, i timer degli STM32 sono molto elaborati e complessi.

La serie L possiede una versione semplificata di questa periferica, in grado di fornire alcune delle sue funzionalità al prezzo di un bassissimo consumo di corrente, denominata Low Power Timer (LPTIM). L'utilizzo di questo modulo è stato la chiave che ci ha consentito di ottenere risultati incredibili sul piano energetico. Il LPTIM consente infatti al sistema di eseguire attività semplici mentre il consumo di energia è ridotto al minimo. Esso è dotato di un registro di ricarica automatica a 16 bit per impostare il periodo e di un registro di confronto a 16 bit per impostare il duty-cycle per una forma d'onda PWM generata come segnale di uscita sul timer. LPTIM può essere utilizzato per il timing e per la generazione di segnali in uscita, mentre il microcontrollore è in modalità a basso consumo, e presenta inoltre un'ampia varietà di sorgenti di clock che gli consentono di rimanere attivo nella maggior parte delle modalità di alimentazione, ad eccezione delle modalità Standby e Shutdown.

Uno schema di clock flessibile consente all'LPTIM di essere utilizzato come contatore di impulsi utilizzando una sorgente di clock esterna. Una delle più importanti funzionalità di LPTIM è la possibilità di riattivare il sistema da modalità a risparmio energetico e di realizzare una "funzione di timeout" con un consumo di energia estremamente basso.

Esiste, inoltre, un contatore di impulsi, che consente di monitorare ed eventualmente azzerare il conteggio una volta raggiunto il valore del registro comparatore.

LPTIM, come tutti i timer, supporta due modalità di conteggio: sincrono (a polling) e asincrono (interrupt). Quando è configurato in modalità di conteggio asincrono, LPTIM continua a funzionare anche qualora non sia attiva alcuna sorgente di clock interna. LPTIM rimane, inoltre, attiva sia in modalità di Sleep che in modalità di Stop ed è in grado di riattivare il controllore da queste modalità.

Al contrario, in modalità Standby o Shutdown, il timer viene spento e deve essere completamente re-inizializzato, qualora l'MCU esca da una di queste modalità. Nei capitoli successivi vedremo meglio come questa periferica è stata sfruttata per risolvere il problema del conteggio a basso consumo energetico.

Un'altra componente la cui conoscenza è risultata fondamentale per portare a termine la parte di progettazione è il transceiver SX1276.

Il microcontrollore, come è mostrato in Figura 3.1.3, invia i comandi al transceiver attraverso la sua SPI1.

SPI è un bus standard di comunicazione seriale che consente la trasmissione di dati tra un dispositivo master ed uno slave. All'interno di questo bus vi possono essere presenti un master ed N slave.

Il master genera il clock della trasmissione e seleziona a quale slave inviare i dati fornendo alimentazione al suo pin di Chip Select.

I dati vengono inviati sui due pin MISO (master input slave output) e MOSI (master output slave input)

Ogni dispositivo slave ha quindi un proprio pin di CS direttamente connesso al master e condivide con tutti gli altri le line MISO, MOSI ed il segnale di clock.

Il pin di reset, per definizione, consente al microcontrollore di riavviare il trasceiver; spesso il riavvio può essere utilizzato in caso di errore sul bus condiviso per ripristinare la comunicazione.

Altri pin condivisi tra i due dispositivi sono i DIO (digital input-output), la cui funzione riguarda principalmente il debug ed appunto per questo non sono stati utilizzati.

Un pin di controllo molto importante è invece il PA12 del microcontrollore, il quale consente di controllare l'alimentazione sul clock del SX1276. Come è possibile notare dalla Figura 3.1.3, il clock del trasceiver viene generato da un quarzo posizionato esternamente (TCX0).

Per poter sfruttare questa potenzialità è necessario però collegare il PA12 con la VDD del quarzo.

Se invece la VDD del quarzo venisse collegata direttamente alla VDD del micro, questa funzionalità si andrebbe a perdere, lasciando sempre acceso il transceiver, con conseguente esponenziale aumento dei consumi energetici. Nella board B-L072Z-LRWAN1 questa procedura viene resa facilmente configurabile attraverso il jumper 9. La possibilità di controllare a pieno il trasceiver garantisce al microcontrollore maggiore liberà di esecuzione della propria macchina a stati senza dover subire limitazioni comportamentali a causa di questo vincolo energetico.

3.4 Progettazione Hardware

Prefazione

In questa sezione verranno descritte tutte le fasi legate alla progettazione hardware del dispositivo.

Il motivo per cui la presentazione di questa fase viene anticipata rispetto a quella relativa alla progettazione software, è il volere semplificare la comprensione di quest'ultima, anche se durante l'avanzamento dei lavori sono state pressoché contemporanee, dando però sempre maggiore priorità alle limitazioni hardware. Il software, infatti, offre maggiore flessibilità e minor costo in caso di errore di progettazione, ricoprendo quindi un ruolo fondamentale nella risoluzione dei problemi progettuali che l'hardware non riesce a risolvere (o la cui risoluzione presenta un costo economico fuori budget).

Per poter progettare al meglio l'hardware è stato necessario utilizzare molti strumenti presenti solo in un laboratorio elettronico, e perciò sono stati sfruttati i laboratori ARCES.

Oltre alla normale strumentazione di saldatura (quali saldatori, stagno, pompetta succhia stagno, trecciola di rame etc...), sono stati inoltre impiegati strumenti di precisione come microscopi, alimentatori da laboratorio, multimetri, microscopi, analizzatori di corrente ed analizzatori di segnale digitale.

A seguire sono elencate le caratteristiche degli strumenti utilizzati.

Nome	Specifiche
Alimentatore Agilent E3631A	0-6V, 5A (DC) ±25V, 1° (AC)
Oscilloscopio Agilent Infiniium 54832D	Frequenza massima: 1GHz/4GSa/s Sistema Operativo: Window XP Numero canali: 4
Multimetro Fluke 187	Sensibilità strumento Tensione: mV Corrente: uA Temperatura: °C Resistenza: Ω
Power Monitoring ST	Tempo massimo di acquisizione: 60 s Frequenza di campionamento: 12802 Hz Canali:4

Per la parte di disegno del circuito stampato è stato utilizzato KiCad [20].

KiCad è un software multipiattaforma open source Electronic Design Automation (EDA) per il disegno di schemi elettrici e circuiti stampati (PCB), il quale possiede un ambiente di sviluppo integrato (IDE) con editor di schematici, generazione della distinta base, sbroglio circuitale del PCB e visualizzatore di file Gerber.

Circuito di alimentazione

Un obiettivo fino ad ora non molto descritto, ma anch'esso linea guida fondamentale di questo progetto, è il contenimento del costo.

Infatti, per rientrare nel budget inizialmente prefissato sono state fatte scelte a livello componentistico che andassero a minimizzare il costo della parte digitale di acquisizione.

Una di queste scelte ha riguardato l'opzione della batteria.

Proprio per ridurre i costi al minimo, si è scelto di non includere nel prodotto finito alcuna batteria, ma di alimentare il circuito utilizzando 3 pile alcaline formato AA da 1.5V ciascuna, che dovranno essere inserire dall'utilizzatore una volta acquistato il prodotto.

Ciò che viene in cambio fornito, però, è la garanzia che queste dureranno per diverse stagioni di irrigazione, e che quindi la loro usura arrivi prima della loro scarica.

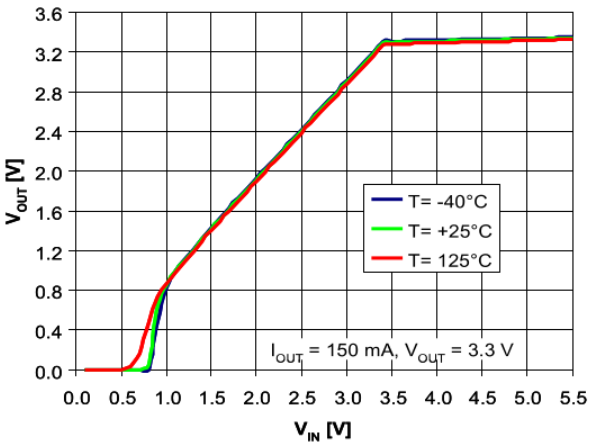
Se fosse per esempio stata inclusa una batteria Li-Po, essa avrebbe da un lato comportato notevoli vantaggi, come per esempio il fatto di poter essere ricaricata facilmente, di avere una maggiore resistenza alle condizioni di umidità e, di conseguenza, di essere soggetta a un processo di invecchiamento fosse più lento; ma d'altro canto, essa avrebbe aumentato il costo dell'oggetto di circa il 15%.

Non è stato quindi ritenuto conveniente includerla all'interno del progetto, nonostante offrisse una maggiore durabilità sia a livello di carica sia dal punto di vista fisico del deterioramento.

Come ben sappiamo però, una tensione di 4.5V risulta non adatta ad un microcontrollore. Questa deve essere ridotta fino a 3.3V, e a tal fine è necessario includere un regolatore di tensione.

La scelta è ricaduta nel STLQ015 [21], un regolatore di tensione a bassissimo consumo energetico.

Le caratteristiche principali di questo componente sono le seguenti:

ID Prodotto	STLQ015
Produttore	ST Mircoelectronics
Tensione di ingresso	1.5-5.5V
Corrente massima	150mA
Tensione in uscita	3.3
Tensione di dropout	

Come è possibile notare dalla descrizione soprastante, esso soddisfa pienamente i requisiti imposti dal nostro sistema:

- La tensione in ingresso applicata dalle pile è 4.5 V e STLQ015 supporta fino a 5.5 V;
- Il PCB, come vedremo successivamente, richiede una corrente di picco di al massimo una decina di mA e STLQ015 può arrivare a generare una corrente di al massimo 150 mA;
- La tensione in uscita è esattamente quella richiesta dal microcontrollore (3.3V);
- Analizzando il grafico della tensione di dropout, si può notare che è sufficiente fornire in ingresso una tensione 3.4 V per ottenere in uscite una tensione di 3.3V, quindi il delta delle pile

utilizzabile è di 1.1V (4.5 – 3.4). Questo consente di sfruttare al meglio la tensione di alimentazione. STLQ015 è infatti considerato in regolatore a bassa tensione di dropout.

Progettazione Filtro Passa Basso

Come è possibile immaginare, quando vengono utilizzati switch analogici, come nel nostro caso il reed switch, non si può evitare di considerare il rimbalzo del segnale dovuto alla chiusura meccanica del circuito. Questo problema si verifica troppo rapidamente per la percezione umana. Quando un interruttore viene attivato, i contatti devono infatti spostarsi fisicamente da una posizione all'altra, ed esso. Collocandosi nella loro nuova posizione, rimbalzano meccanicamente, facendo aprire e chiudere il circuito sottostante più volte. Per venire a capo di questo problema vi sono due possibili soluzioni:

- La prima è quella maggiormente utilizzata, e consiste nel risolvere il problema via firmware.
  Si può dunque sviluppare un algoritmo che campioni il segnale in ingresso per un lasso di tempo definito con una determinata frequenza e restituisca valore alto o basso in base alla percentuale di campioni alti rispetto a quella di campioni bassi.
- La seconda invece prevede che i disturbi nel segnale vengano risolti via hardware.

La strada più comune consiste nell'implementazione di un filtro passa basso in grado di eliminare i rimbalzi. Normalmente questo viene costruito utilizzando una resistenza ed un condensatore, i quali consentono di regolare la frequenza di taglio a -3db del filtro. Come si può immaginare, è possibile eliminare i disturbi in questo modo solo se la frequenza del disturbo è maggiore della frequenza del segnale. In caso contrario, se si utilizzasse comunque questo metodo si andrebbe a cancellare parte del segnale stesso oltre che il disturbo.

Prima di andare a definire quale, limitatamente al nostro caso, si sia rivelata essere la soluzione vincente tra le due, ed i motivi di questa scelta, è molto importante analizzare l'entità del disturbo.

Nel nostro caso abbiamo avuto modo effettuare diverse misurazioni in laboratorio tramite l'oscilloscopio.

Il disturbo da noi identificato è mostrato in Figura 3.1.

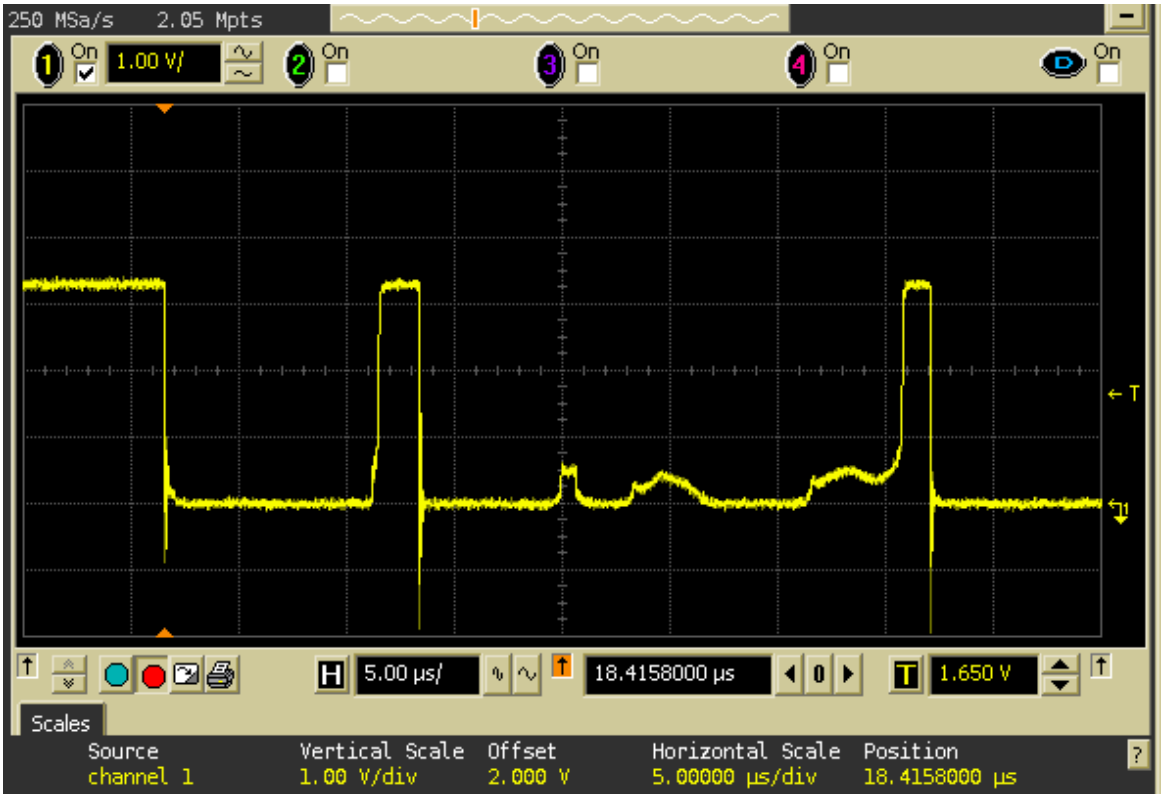


Figura 3.4.1a - Disturbi sulla chiusura del reed switch

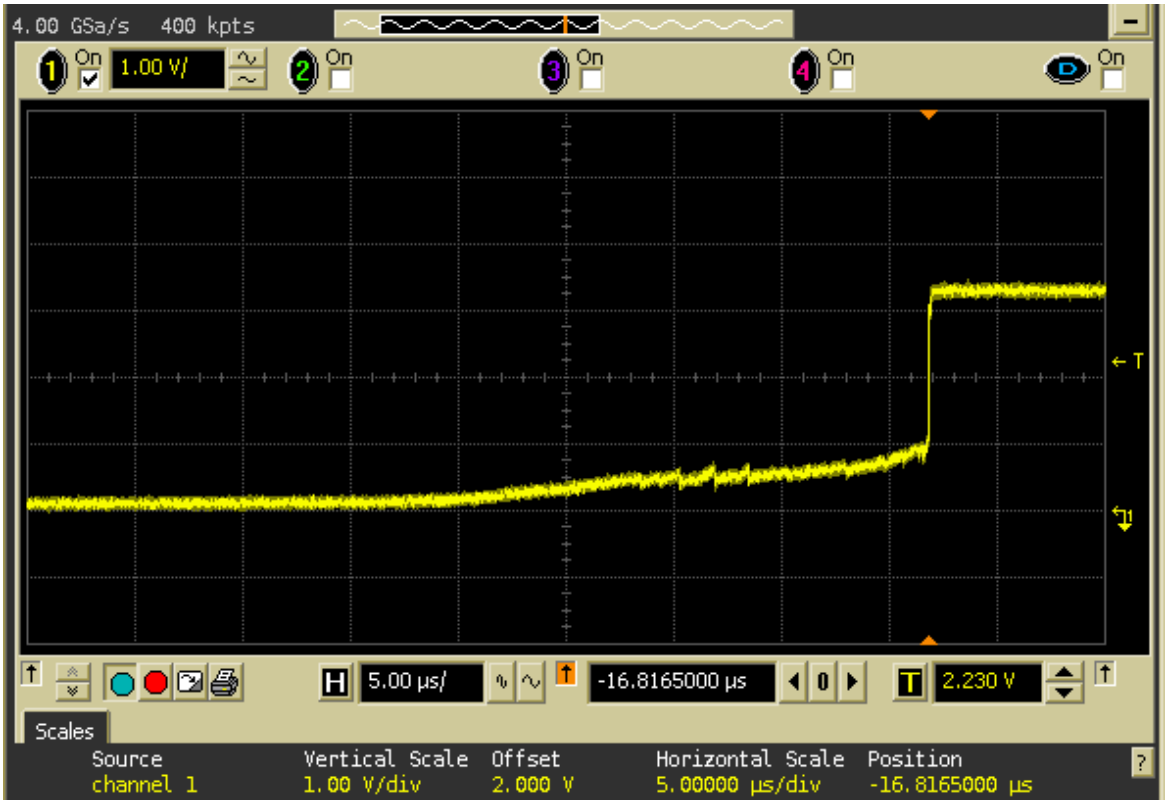


Figura 3.4.1b – Disturbi sull'apertura del reed switch

Come è possibile notare dalla Figura 3.4.1 il disturbo è visibile solo sul fronte di discesa del segnale, ovvero quando l'interruttore si chiude (Figura 3.4.1a), mentre per quanto riguarda il fronte di salita (apertura del circuito) non vi sono particolari incongruenze che porterebbero ad una lettura errata.

Il disturbo presenta diverse irregolarità, alcune ben definite (il segnale raggiunge il livello di tensione della VDD), ed altre che invece potrebbero non risultare determinanti nella compromissione del segnale in quanto il loro valore di tensione è intermedio tra VDD e 0V. Sono però state effettuate diverse verifiche per confermare il problema e ciò che si è notato è il fatto che queste anomalie non sia mai uguali, ma anzi varino ogni volta che si effettua la procedura. Quello mostrato in Figura 3.4.1 rappresenta una sola lettura del disturbo.

Detto ciò, diversamente da come si potrebbe pensare, abbiamo optato per la costruzione di un filtro anti-rimbalzo in hardware piuttosto che adottare una soluzione software. Le motivazioni di questa scelta sono principalmente due. Da una parte, l'utilizzo de il LPTIM come periferica di acquisizione perderebbe ogni significato, in quanto proprio le potenzialità di quest'ultimo sono il fatto che il microcontrollore possa rimanere in modalità di stop e risvegliarsi solo nel momento in cui vengono generati interrupt.

Dall'altra, appunto, l'utilizzo di una acquisizione software, richiedendo che il microcontrollore sia completamente attivo, provocherebbe un aumento esponenziali dei consumi energetici.

La progettazione del filtro passa-basso in hardware prevede che prima venga definita la frequenza del segnale disturbante.

Se consideriamo che la misura effettuata in Figura 3.4.1a ha una risoluzione di 5 us per divisione, possiamo identificare il disturbo come un'onda quadra con frequenza nell'intervallo 100-1000 KHz .

In un filtro passa-basso, la frequenza di taglio rappresenta il livello di soglia oltre al quale ogni segnale di frequenza maggiore viene annullato. Per questo motivo, è importante riuscire a tagliare fuori dal range di frequenza valido (0-Frequenza di taglio) ogni componente del disturbo.

Per fare ciò è necessario selezionare nella misura effettuata il picco con minor frequenza ipotizzando un duty cycle del 50%: questo è il segnale che deve rappresentare la soglia sulla quale andare a calcolare la frequenza di taglio.

Nel nostro caso, il picco con durata maggiore è il primo ed è circa la metà (2.5 us) dell'intera divisione (5 us). Possiamo dunque tenere come riferimento un disturbo con una frequenza di circa 200 KHz.

Considerando che l'intensità di segnali con frequenza equivalente alla frequenza di taglio del filtro viene applicato un guadagno di -3db, ovvero un dimezzamento, questo potrebbe risultare non sufficiente, in quanto se abbiamo il valore 1 del segnale rappresentato da un valore di tensione pari a 3.3V e lo dimezziamo, otteniamo una tensione circa pari a 1.7V.

1.7 V è però anche il livello di soglia del microcontrollore per definire un 1 o uno 0 dato un segnale digitale, quindi questo potrebbe creare ulteriori problemi a livello di lettura.

È quindi importante considerare un filtro costruito sulla frequenza di taglio letta precedentemente calcolata, solo come base di partenza dalla quale poi continuarne a diminuire il valore fino a che i picchi di disturbo sul segnale non arrivano quasi ad annullarsi. Nel nostro caso vi è un limite inferiore di frequenza di taglio utilizzabile, rappresentato dalla frequenza del segnale originale generato dal reed switch che, come abbiamo calcolato all'inizio, non può superare i 0.45 Hz dovuti alla portata massima della conduttura. Il circuito che funzionalmente rappresenta un filtro passa basso è il seguente (Figura 3.4.2).

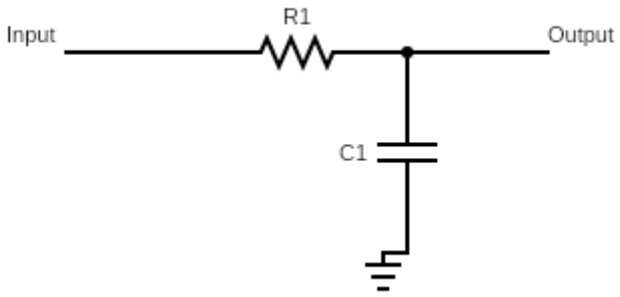


Figura 3.4.2 – RC LowPass Filter

I componenti su cui lavorare sono la resistenza R1 ed il condensatore C1. Per calcolare la frequenza di taglio del filtro dati R1 e C1 è necessario utilizzare la seguente formula.

$$fc = \frac{1}{2\pi RC}$$

Per poter ottenere il risultato desiderato abbiamo deciso di fissare la capacità del condensatore a 100nF e di variare, fino a compimento del test, solo la resistenza.

I seguenti grafici hanno tutti la stessa scala ovvero 50us per divisione.

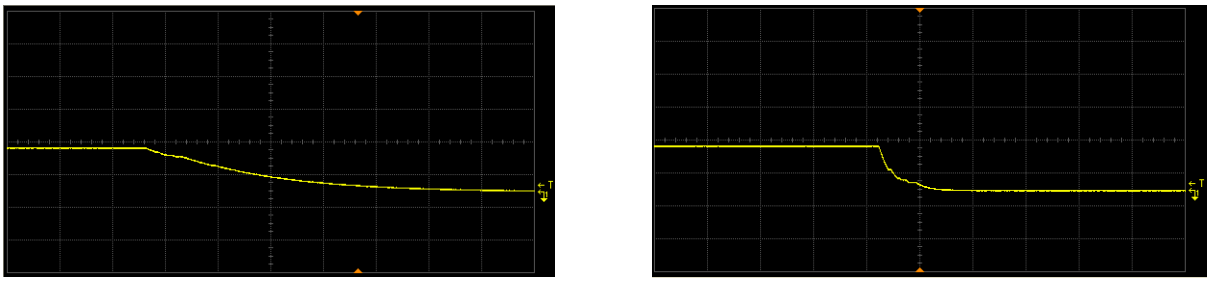


Figura 3.4.3a – Risultato del filtraggio utilizzando una resistenza da 1 KΩ (a sinistra) ed una resistenza da 100Ω (a destra).

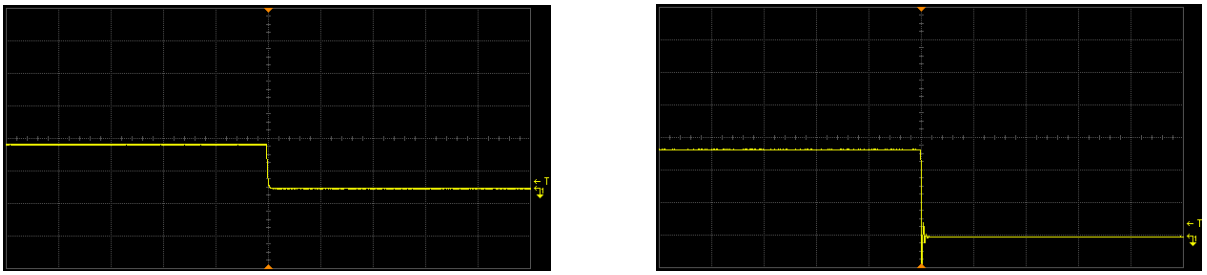


Figura 3.4.3b – Risultato del filtraggio utilizzando una resistenza da 10 Ω (a sinistra) ed una resistenza da 1Ω (a destra).

Come si può notare dalla Figura 3.4.3a, utilizzando una resistenza da 1 KΩ si otterrebbe un filtro sì in grado di filtrare tutti i disturbi sul reed switch, ma allo stesso tempo sovradimensionato in quanto la frequenza dei disturbi è di gran lunga superiore alla frequenza di taglio del filtro.

Analizzando invece il grafico del filtro con la resistenza da 1 Ω (Figura 3.4.3b), si nota come i disturbi non vengano eliminati, ma solamente attenuati.

Per avere maggiori conferme osserviamo gli stessi grafici della Figura 3.4.3b ma ingranditi in modo da scorgere tutti i particolari circa gli effetti del filtraggio (Figura 3.4.4).

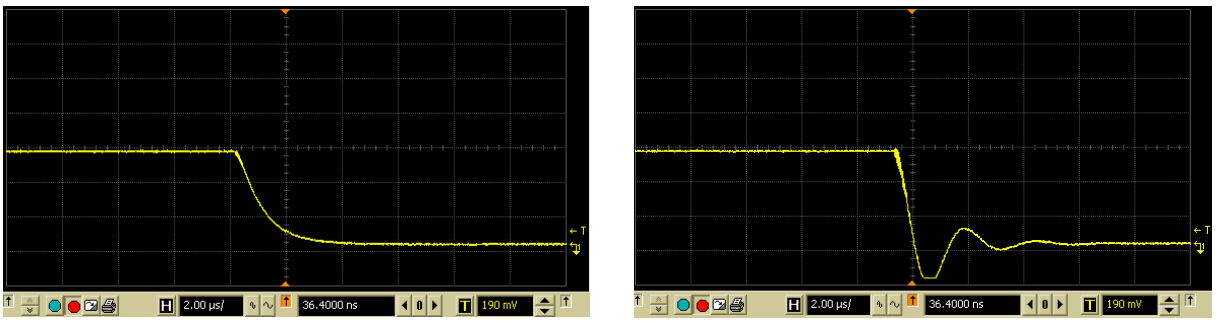


Figura 3.4.4 - Figura 3.4.3b ingrandita

Si può chiaramente constatare come la resistenza da 1 Ω non sia sufficiente ad eliminare tutti i disturbi e di consegua il limite inferiore di resistenza per la progettazione di questo filtro deve essere imposto a 10 Ω.

I due valori di resistenza consigliati in base a quelle che sono state le nostre acquisizioni, sono quindi 10 Ω e 100 Ω. Per motivi di sicurezza ed affidabilità si è optato per la resistenza a 100 Ω, questo in modo di garantire un filtraggio leggermente sovradimensionato ma in grado di eliminare qualunque tipologia di disturbo. È stata fatta questa scelta in in quanto non si conosceva l'esatta variabilità del disturbo. Non è infatti detto che un disturbo si presenti sempre con la stessa frequenza di segnale.

Sviluppo Filtro Passa Basso per LPTIM

LPTIM necessita, come abbiamo visto in precedenza, di un'onda quadra praticamente perfetta in ingresso altrimenti il conteggio del numero di impulsi ricevuti rischia di risultare errato. Come descritto nella sezione precedente, l'utilizzo di un filtro passa-basso sull'ingresso del LPTIM è in grado di risolvere questo problema.

Bisogna però considerare il fatto che reed switch della conduttura ha bisogno di un circuito in cui è presente una resistenza di pull-up per poter generare un segnale ogni volta che il magnete si avvicina ad esso. A questo punto possiamo quindi unire i due circuiti ed il risultato finale è il seguente.

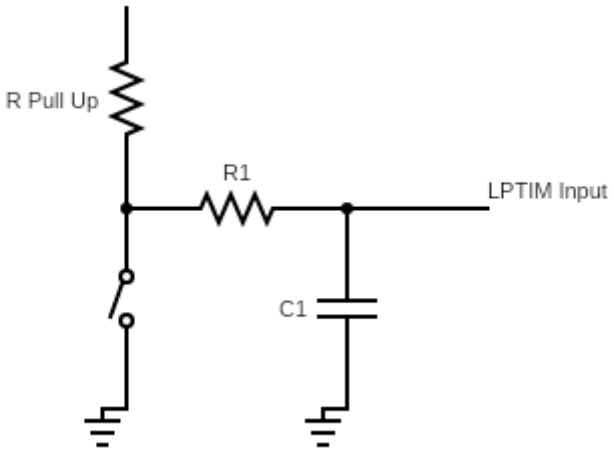


Figura 3.4.3 - RC LowPass Filter nel LPTIM

L'unione del circuito del reed switch con quello del filtro, in realtà, va ad alterare il funzionamento del filtro stesso.

In particolare, la resistenza di pull up si va a sommare a R1 nella fase di carica del condensatore C1.

Questo modifica il comportamento del filtro scorrelando il tempo del fronte di discesa del segnale con il fronte di salita.

Infatti, mentre il tempo del fronte di discesa (tempo di scarica del condensatore) rimane invariato, poichè la corrente uscente da C1 attraversa R1 per poi terminare a ground, il tempo di carica di C1, quindi la durata sul fronte di salita, aumenterà proporzionalmente in base alla resistenza di pull up.

Questo effetto dovuto all'apertura del reed switch è influenzato dalla somma delle resistenze R1 ed R pull up che in questo scenario risulterebbero in serie.

In Figura 3.4 viene mostrata la differenza del fronte di salità di un segnale in uscita da filtro passa-basso con (Figura 3.4.3 b) e senza (Figura 3.4.3 a) resistenza di pull-up.

Contando il numero di divisioni che impiega il segnale a tornare alto in Figura 3.4.3 a e b e moltiplicandonli per i secondi per divisione della misurazione, questa differenza è netta.

Descrizione	Figura	# Divisioni per risalire	Tempo per divisione	Totale tempo
No pull up	3.4 a	5	10 us	50 us
Pull up	3.4 b	4	50 us	200us

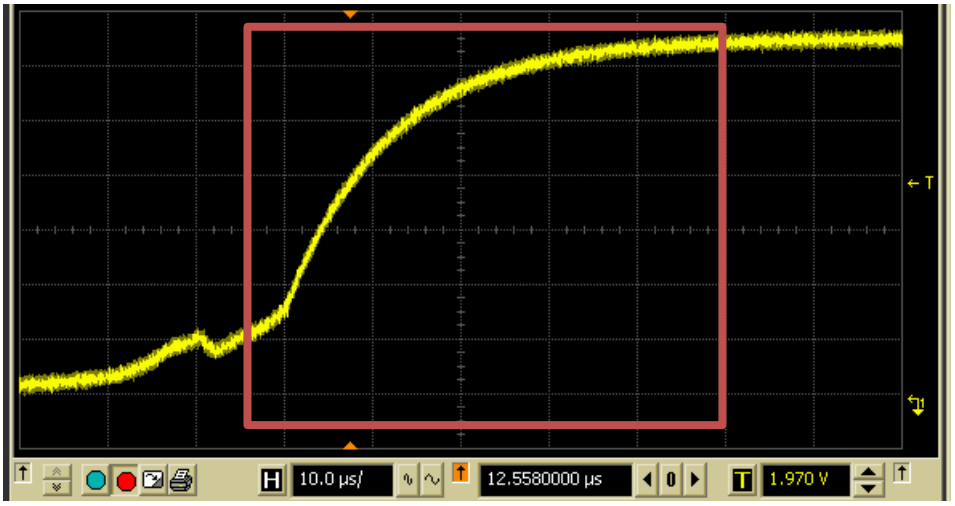


Figura 3.4.3a – Apertura del reed switch con filtro senza resistenza di pull up

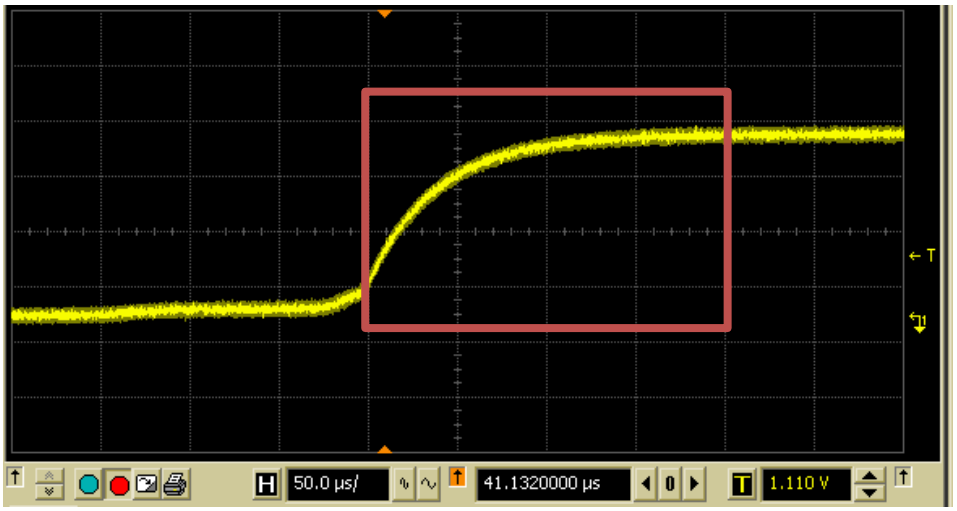


Figura 3.4.3b– Apertura del reed switch con filtro e resistenza di pull up

In questo test la resistenza di pull up non rispecchiava le dimensioni reali ed infatti il tempo del caso senza resistenza di pull up è di sole quattro volte superiore al tempo senza.

Detto ciò, nel nostro caso, fortunatamente, questa situazione non ci preoccupa in quanto il LPTIM consente di effettuare il conteggio sia sul fronte di salita che sul fronte di discesa, sia nel caso si utilizzi un meccanismo di lettura del conteggio a polling (sincrono), sia a interrupt (asincrono). Nella modalità asincrona non è pero possibile conteggiare sia i fronti di salita che quelli di discesa contemporaneamente, ma solo gli uni o gli altri. Noi abbiamo scelto, proprio per scavalcare questo problema di rallentamento del fronte di salita, il conteggio sui fronti di discesa del segnale.

Vi è però un altro problema di maggior severità da affrontare. L'LPTIM, essendo a tutti gli effetti una periferica, necessiterebbe di un clock. Il clock può essere assegnato internamente, esternamente, oppure, come nel nostro caso, può essere utilizzata l'onda quadra generata dal reed switch. Questa metodologia di utilizzo ha però una controindicazione: per potersi inizializzare sono necessari cinque impulsi ogni volta che viene resettato o reimpostato (come descritto nel datasheet del microcontrollore).

Sono state quindi proposte due soluzioni progettuali. La prima consiste nel non effettuare nessuna modifica, ma di considerare all'interno del conteggio i cinque impulsi mancati; ma ciò comporterebbe che, per irrigazioni inferiori ai 500 litri, non venga effettuata nessuna segnalazione via radiofrequenza.

Quindi, per evitare precludersi eventi di questa entità, si è scelto di adottare altra soluzione, ovvero l'inserimento nel circuito di un multiplexer (MUX) in grado di selezionare il canale di input de il LPTIM. Così facendo, ad ogni riattivazione di suddetta periferica, il microcontrollore può generare una sequenza di esattamente cinque impulsi mettendosi così in condizione di contare dal successivo.

In pratica, il MUX consente di selezionare tra due linee digitali quale collegare al canale di uscita, che in questo caso sarebbe il pin di ingresso dell'LPTIM.

Il microcontrollore non deve fare altro che reimpostare l'LPTIM, generare i cinque impulsi in un tempo notevolmente rapido in modo da non entrare in conflitto con il segnale proveniente dal reed switch

sull'ingresso uno del MUX, ed infine attivare ingresso due dello stesso sul quale è collegato il circuito descritto in Figura 3.4.2.

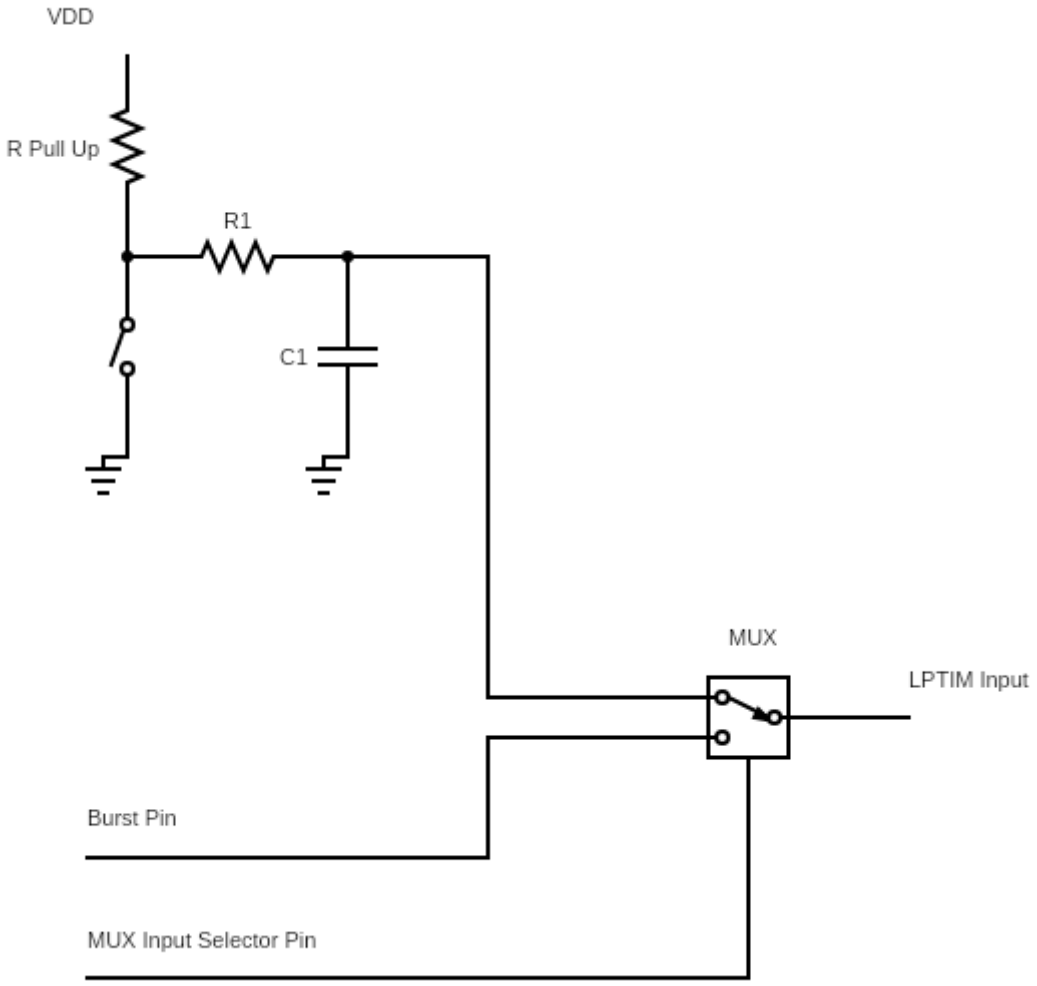


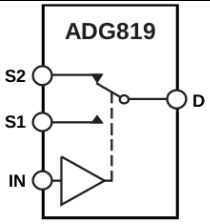
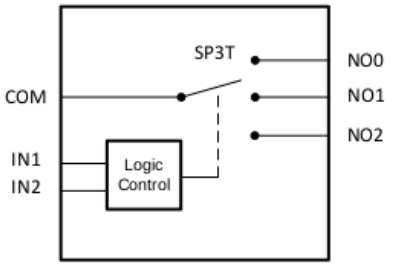
Figura 3.4.4 – Circuito Filtro LP con MUX

Non è stato inserito alcun filtro sul pin di generazione dei cinque impulsi (Burst Pin) in quanto, essendo essi generati digitalmente, sono privi di disturbi. La selezione di quale dei due ingressi collegare all'ingresso del LPTIM viene effettuata attraverso un pin del MUX (MUX Input Selector pin) collegato direttamente al microcontrollore.

Lo schema teorico in Figura 3.4.4 deve essere poi adattato al tipo di multiplexer che viene utilizzato nel sistema.

Nel nostro caso, abbiamo acquistato due multiplexer diversi, in modo da poter scegliere quello le cui caratteristiche fossero maggiormente conformi alle nostre esigenze.

Il primo è un MUX 2:1 ADG819 (Analog Devices)[22], il secondo invece è un MUX 3:1 TS5A3359 (Texas Instruments)[23].

Produttore	Analog Devices	Texas Instruments
Codice	ADG819	TS5A3359
Schema		
R interna a 25°	0.5 Ω	1 Ω
Tempo di accensione	35-45 ns	16-30.5 ns
Tempo di spegnimento	10-16 ns	6-11.5 ns
Frequenza a -3 db	17 MHz	73 MHz
Corrente di alimentazione	1 uA	20 nA
Tensione di alimentazione	1.8-5.5 V	1.65-5.5 V
Corrente massima supportata	200 mA	100 mA
Costo unitario	2.7$	1 $

Come si può intuire dalla colorazione della tabella soprastante, è stato scelto il componente della Texas in quanto vincente su tutti i fronti se consideriamo che una corrente sempre inferiore a 50 mA.

I punti che convincono in via definitiva ad utilizzare questo componente sono la corrente di alimentazione (20 nA) e principalmente il costo unitario (1$).

Inoltre, vi è un ulteriore dettaglio non menzionato nella tabella, ovvero il fatto che ad ogni operazione di switch tra le linee S1-S2 nell'ADG819 si verificasse un buco nel segnale, come se per un istante nessuna delle due linee fosse connessa all'uscita D. Questo è stato possibile verificarlo assegnando sia a S1 che a S2 un segnale alto (3.3 V) e successivamente effettuare un cambio di linea tramite la variazione di tensione sul pin di selezione del MUX.

Figura 3.4.5 – Acquisizione burst con ADG 819

Analizzando la Figura 3.4.5, la parte evidenziata in rosso rappresenta S1 mentre quella in verde S2.

La parte di segnale evidenziata in blu corrisponde a quella mancanza di tensione relativa alla fase di switching precedentemente citata.

Con questa caratteristica il componente diventata praticamente inutilizzabile o comunque complicato da gestire.

L'obiettivo, nel momento in cui si progetta sia la parte Hardware che quella Software, è cercare di non complicare inutilmente la fase di progettazione complementare a meno che non vi sia un particolare ragione, per esempio la mancanza di una soluzione alternativa al problema.

Queste sono le principali motivazioni che ci hanno portato ad utilizzare come multiplexer, sebbene sovradimensionato (3 canali quando ne erano sufficienti 2), il TS5A3359.

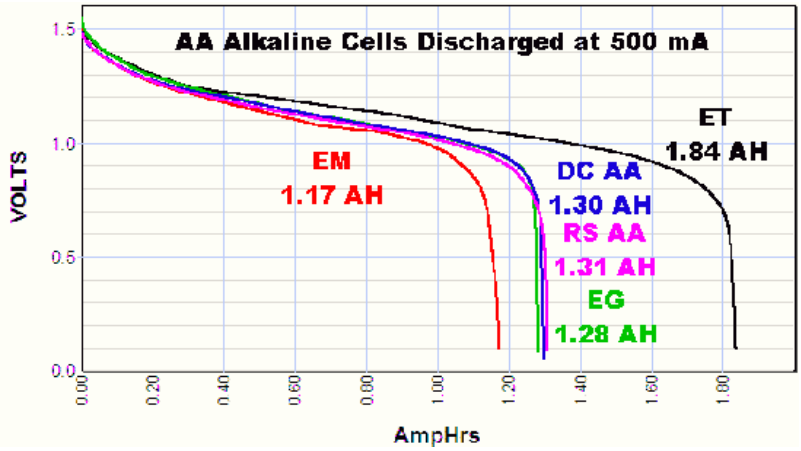


Figura 3.4.6 – Curva di scarica tipica delle batterie stilo AA [24]

Le curve di scarica delle batterie sono spesso composte da tre periodi come viene mostrato in figura 3.4.6. Una fase inziale di scarica che segue un andamento quadratico, una fase centrale linearmente approssimabile ed infine una caduta polinomiale che si attiva nel momento in cui viene raggiunto un certo livello di tensione di soglia. Rappresentare questo andamento e fornirlo all'utilizzatore del dispositivo è spesso complicato ed il valore aggiunto è pressoché nullo.

Si tende quindi ad approssimare le prime due fasi con un andamento lineare, partendo dal valore della batteria al 100% fino al raggiungimento della tensione minima necessaria per alimentare il nostro circuito che deve corrispondere alla tensione di drop out del regolatore di tensione utilizzato.

Per poter leggere il valore di tensione ai capi della batteria è necessario utilizzare un convertitore analogico-digitale (ADC). L'ADC è un circuito elettronico in grado di convertire un segnale analogico con andamento continuo (ad es. una tensione) in una serie di valori discreti. Questa funzione è presente solo su alcuni GPIO del microcontrollore. Per poter utilizzare l'ADC del microcontrollore, però, è necessario tenere in considerazione il fatto che i GPIO possono supportare al massimo una tensione di 3.3V. Questo ci impedisce dunque di connettere direttamente la batteria all'ADC in quanto la tensione ai capi della batteria è di circa 4.5V (3x1.5V). Per poter leggere questo valore di tensione è necessario abbassare il valore di tensione da applicare sull'ADC utilizzando un partitore.

Un partitore consente di abbassare il valore della tensione solo nel momento in cui gli si garantisce un passaggio di corrente. Lasciare sempre abilitato un partitore, porta ad un aumento notevole dei consumi energetici direttamente indotto dal fluire della corrente.

Per risolvere questo problema si è dovuto far fronte ad una soluzione che includesse la possibilità di attivare e disattivare a proprio piacimento il partitore. Si è dunque scelto di adottare un multiplexer ed utilizzarlo come interruttore. Basandosi sul precedente confronto tra i MUX e consci del fatto che il costo di più unità dello stesso componente comporterebbero una diminuzione del suo prezzo, è stato anche qui inserito il TS5A3359. Il circuito risultante è dunque il seguente.

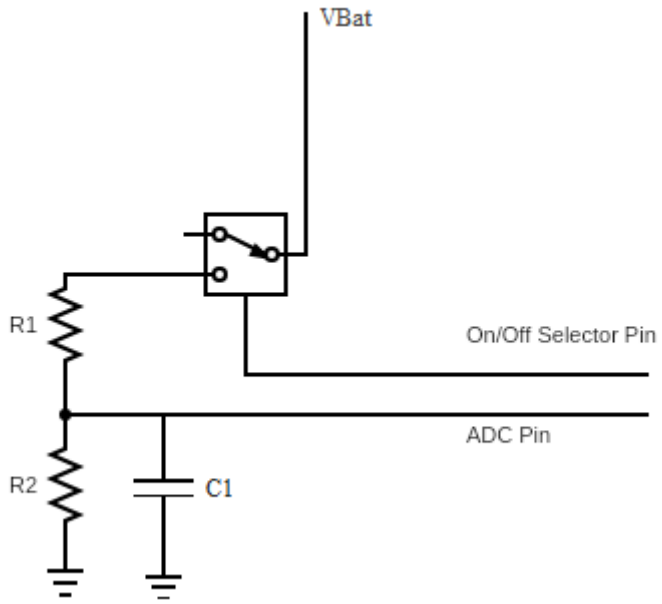


Figura 3.4.7 – Circuito di lettura della tensione sulla batteria

Come è possibile vedere in Figura 3.4.7, il pin COM del MUX viene utilizzato per alimentare il partitore quando il pin di selezione lo abilita, altrimenti rimane flottante. La lettura dell'ADC viene effettuata tra le due resistenze di partizione in modo da non avere una tensione in ingresso superiore ai 3.3V.

Infatti, se impostiamo i valori di R1 ed R2 equivalenti, otteniamo esattamente la metà della tensione di alimentazione sull'ADC, ovvero 2.3V quando la batteria è al 100%. Chiaramente dimezzando la tensione che l'ADC deve leggere, otteniamo una conseguente dimezzamento della precisione di lettura.

Di norma è consigliabile adottare R1 ed R2 tali per cui 4.5V vengono scalati fino a 3.3V esatti, per sfruttare la massima dinamica dell'ADC. Si è scelto di impostare ad entrambe lo stesso valore, in quanto non risultava essere di nostro interesse la precisione con la quale andare a leggere la tensione ai capi della batteria. Risulta inoltre importante leggere i valori di resistenza prima che R1 ed R2 vengano installate nel circuito, questo perché, in caso contrario, si andrebbero a sommare in parallelo alle altre resistenze del circuito complicandone la lettura.

Il loro rapporto tra R1 ed R2 identifica il fattore principale per il calcolo della percentuale di batteria residua. Ciò che invece è importante tenere in considerazione è l'ordine di grandezza di queste resistenze, perché scegliendo resistenze troppo piccole si andrebbe inutilmente a incrementare il costo energetico del circuito per il lasso di tempo in cui questo rimane collegato. Un ordine di grandezza accettabile è infatti sulla scala dei 10^5 o 10^6 Ω.

Analizzando ora, nel dettaglio, il funzionamento del circuito in Figura 3.4.6, nel momento in cui il microcontrollore attiva il MUX utilizzando l'On/Off Selector Pin, la corrente inizia a transitare attraverso il partitore caricando il condensatore C1. C1 ha la funzione di fornire un boost di corrente alla capacità interna dell'ADC in modo che questa possa caricarsi più velocemente durante la fase di lettura. Se C1 venisse eliminata dal circuito, la capacità dell'ADC non avrebbe abbastanza corrente per potersi caricare in tempo al fine di leggere il valore della tensione ai capi di R2. Questo porterebbe ad una lettura solo parziale della percentuale di batteria restante. Il condensatore C1 deve essere quindi dimensionato in modo da soddisfare i requisiti di carica dell'ADC.

Date le innumerevoli variabili in gioco, la soluzione più comune in questi casi è procedere per tentativi, quindi partendo da un valore molto basso di C1 incrementarlo e successivamente effettuare una lettura con l'ADC.

Nel momento in cui la lettura raggiunge il 98-99% del valore di tensione effettivamente presente su R1 si può considerare il condensatore correttamente dimensionato. All'interno del sistema in analisi è stata utilizzata una capacità di boost pari a 100 nF in quanto sufficiente per una lettura con precisione del 99.5%.

Correzioni alle alimentazioni dei componenti

I dispositivi attivi di un sistema elettronico (transistor, circuiti integrati etc…) sono collegati ai loro alimentatori attraverso conduttori con resistenza e induttanza finite.

Se la corrente assorbita da un dispositivo attivo cambia, anche la tensione dell'alimentazione del dispositivo scende di conseguenza. Se più dispositivi attivi condividono un percorso comune all'alimentazione, i cambiamenti nella corrente assorbita da un elemento possono produrre variazioni di tensione sufficientemente grandi da influire sul funzionamento di altri.

Per sopperire a questa problematica sia al regolatore che ai due MUX abbiamo inserito un condensatore di disaccoppiamento da 100 nF (consigliato dai datasheet dei componenti), in grado di fornire un percorso di bypass per le correnti transitorie. Così facendo vengono risolti tutti i problemi di instabilità a livello di tensione sui componenti. Per quanto riguarda il regolatore di tensione, vi è un altro aspetto da considerare: i disturbi generati dal componente sulla tensione di uscita.

Anche qui, come viene descritto all'interno del suo datasheet, è stato introdotto un condensatore tra VOut e GND creando così un filtro unicamente capacitivo in grado di eliminare i disturbi in alta frequenza.

3.5 Progettazione Software

Prefazione

L'implementazione di interi sistemi utilizzando soltanto la parte di progettazione in hardware è possibile, ma allo stesso tempo costosissima e soprattutto poco scalabile e flessibile.

I motivi per cui al giorno d'oggi un oggettistica hardware include sempre al suo interno un'unità di calcolo programmabile sono svariati.

Riassumendoli, il software permette di ridimensionare l'hardware al minimo necessario, fornendogli la possibilità di essere utilizzato per scopi anche molto diversi tra loro.

La flessibilità che il software è in grado di fornire all'hardware è praticamente illimitata, questo ha consentito all'avanzamento tecnologico di ridurre notevolmente i costi dei prodotti nel corso degli anni.

La diminuzione del costo del prodotto non è solamente frutto della diminuzione dei componenti al suo interno, ma anche del fatto che un prodotto possa essere rilasciato sul mercato più velocemente ed in caso di errori di progettazione, questi possano essere corretti mediante un semplice aggiornamento Firmware automaticamente proposto all'utente attraverso un sofisticato sistema remoto che sfrutta, per esempio, la connettività Internet. Nel nostro caso le componenti principali Software sono due: il Firmware del Flow Meter e l'interfaccia web che raccoglie i dati inviati dal sensore. In questo documento verrà presa in analisi la parte riguardante il Firmware del sensore, mentre per quanto riguarda la componente di accumulazione dati ed interfaccia utente, questa sarà solamente accennata perché non facente parte di questo progetto ma del suo macro-progetto SWAMP.

Progettazione Flow Meter

La progettazione del Firmware del Flow Meter è stata processata su tre fasi principali:

1) Progettazione macchina a stati finiti e definizione delle periferiche utilizzate;
2) Costruzione della struttura del progetto con STM32CubeIDE [25] ed inclusione librerie;
3) Implementazione;

Nella prima fase, l'obiettivo è stato quello di raccogliere tutti i requisiti di cui il sistema avrebbe avuto bisogno, confrontarli ed infine descriverne la procedura che garantisce il loro soddisfacimento in accordo con gli obietti inizialmente descritti di questo progetto. Questa procedura è stata progettata sotto forma di macchina a stati finti. Lo strumento utilizzato per esprimerla è Dia, un software multipiattaforma open source che consente di disegnare diverse tipologie di schemi a blocchi.

Per poter comprendere al meglio il funzionamento del Flow Meter e come questo vada a migliorare l'efficienza intesa come consumi idrici, è necessario riprendere in analisi la situazione evidenziata nella fase introduttiva di questo documento che descrive come attualmente l'addetto al rilascio idrico decide quanta acqua fornire ai campi durante ogni sessione di irrigazione.

Mentre prima quest'ultimo aveva, oltre che il compito di aprire la valvola di irrigazione, l'obbligo di conteggiare il tempo che trascorso e moltiplicandolo per la portata di acqua della conduttura ottenere il numero di litri utilizzati, ora dovrà solamente preoccuparsi dell'apertura e della chiusura della valvola idraulica.

Il Flow Meter possiede due stati principali e due modalità di esecuzione. Parlando di stati identifichiamo quello di IDLE, o di attesa, e quello di FLOODING o di irrigazione. Come è possibile scorgere nella rappresentazione della macchina a stati in Figura 3.4.1, il nodo, non appena avviato, entra in una fase di inizializzazione all'interno della quale il suo stato viene impostato ad IDLE.

Successivamente viene richiesto l'intervento della periferica LPTIM, qui infatti quest'ultimo viene inizializzato ed attivato in modalità Interrupt con un valore di confronto pari a tre.

Questo significa che non appena il registro contatore conterrà un valore superiore al registro comparatore verrà generato un interrupt che dovrà poi essere gestito dal microcontrollore. I tre impulsi iniziali servono al dispositivo per identificare l'inizio della fase di irrigazione evitando che la pala interna alla conduttura possa attivare erroneamente il conteggio dell'acqua durante il trasporto, il montaggio e/o la rimozione del sensore. Questo valore rappresenta dunque una soglia da valicare prima di iniziare il conteggio.

Proseguendo con l'analisi della schedulazione di istruzioni iniziali, vengono avviati due timer T1 e T3 il cui funzionamento verrà descritto in seguito.

Con timer intendiamo un processo software basato sull'utilizzo della periferica RTC, in grado di attendere silenziosamente per un lasso di tempo impostato dall'utilizzatore ed al termine di esso eseguire una funzione di callback (routine di interrupt).

Infine, per concludere la fase di boot del dispositivo, questo entra in modalità di STOP. All'inizio abbiamo parlato di due modalità, la prima è quella tradizionale di RUN, dove tutti gli apparati del microcontrollore rimango attivi e pronti per essere utilizzati.

La seconda è la modalità di STOP. Qui, invece, il microcontrollore rimane spento lasciando attiva la RAM ed alcune periferiche tra cui LPTIM ed RTC, l'effetto ottenuto è simile ad una standby finalizzato al risparmio energetico. Come ben sappiano l'obiettivo di questo dispositivo è di contare, attraverso un generatore di impulsi, la quantità di acqua transitata all'interno di una conduttura, considerando che ad ogni impulso corrispondono 100 litri d'acqua.

Il pin di lettura dell'impulso deve essere dunque collegato al LPTIM in modo che questo possa incrementare il proprio registro contatore di volta in volta. Nel momento in cui viene registrato il quarto impulso, viene generato l'interrupt invocando la propria callback di gestione.

La caratteristica principale della modalità di STOP è il fatto che ad ogni input ricevuto, il microcontrollore venga svegliato e entri nuovamente in modalità di RUN. La funzione di callback di questo meccanismo, attivandosi, esegue le seguenti operazioni:

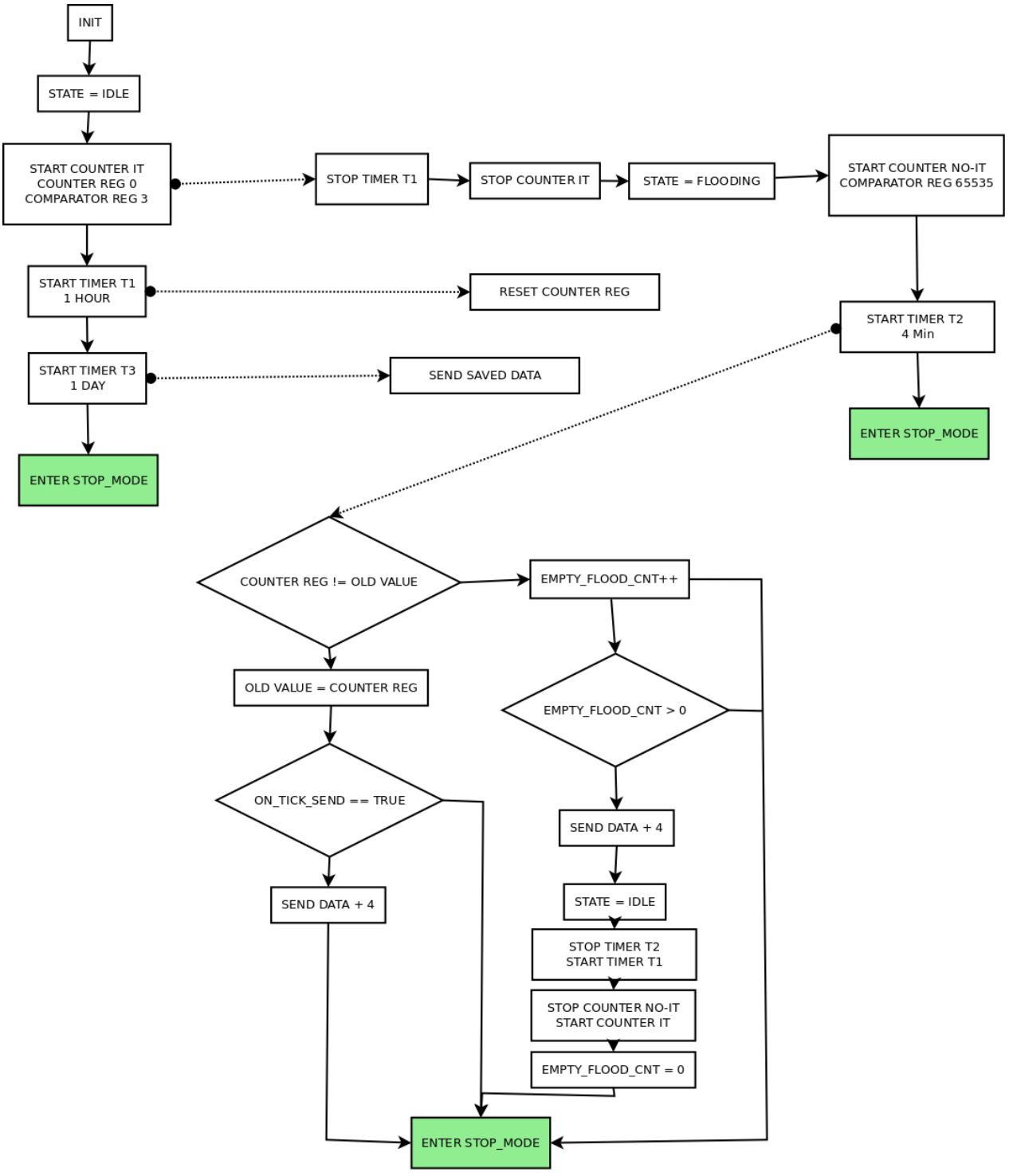


Figura 3.5.1 – Macchina a stati Flow Meter

1) Imposta come stato corrente quello di FLOODING in quanto la soglia dei tre impulsi è stata valicata ed è possibile avviare il conteggio.

2) Blocca il timer T1. T1 è il timer la cui callback è finalizzata al reset del registro contatore prima che il dispositivo raggiunga lo stato di FLOODING. Ovvero reimposta il valore in modo da tornare ad alzare il livello di soglia da superare per passare in stato di FLOODING.
3) Avvia l'LPTIM in modalità No-Interrupt, questo vuol dire che d'ora in poi non verranno più generati interrupt ma sarà compito del software effettuare operazioni in polling per la lettura del valore dal registro contatore.
4) Avvia un timer T2 la cui funzione è quella di svegliare il microcontrollore, il quale potrà poi andare a leggere i dati del conteggio, processarli ed inviarli.
5) Torna in modalità di STOP.

Una volta scattato il timer T2 il microcontrollore dovrà risvegliarsi.

Gli scenari possibili a questo punto sono principalmente due.

1) Sta ancora scorrendo acqua all'interno della conduttura
2) Non vi è più un flusso d'acqua

L'unico modo per distinguere queste due condizioni è salvare il valore relativo al conteggio registrato durante l'ultimo risveglio del microcontrollore e confrontarlo con il valore attuale. Se i due valori corrispondono, la condizione di fine irrigazione è verificata, altrimenti potrebbe essere ancora presente un flusso d'acqua nella conduttura. Questo è infatti il primo controllo che viene effettuato all'interno della callback del timer T2.

Successivamente si prosegue poi, nello scenario 1, salvando il nuovo valore registrato e nel caso inviando il dato aggiungendo però i quattro impulsi relativi livello di soglia. Se effettuare o no l'invio del dato durante la fase intermedia è stata lasciata come possibilità eventualmente attivabile abilitando un flag nelle direttive di preprocessing.

Nello scenario 2 invece, si invia inizialmente il dato maggiorato sempre dei quattro impulsi iniziali e successivamente si ripristina lo stato di IDLE facendo ripartire l'LPTIM in modalità IT, bloccando il

timer T2 (quello che ha scatenato l'evento in analisi) ed avviando il timer T1, ovvero quello di reset del livello di soglia. Infine, in entrambi casi, la modalità viene reimpostata a quella di STOP.

Seguendo il procedimento indicato dalla macchina a stati mostrata in Figura 3.5.1, si riesce quindi a tenere in considerazione tutti le possibili situazioni verosimili ed affrontarle con un approccio basato prettamente sul risparmio energetico.

Non è, però, stato ancora menzionato lo scopo del timer T3. T3 serve per inviare una volta al giorno i dati di conteggio attualmente salvati in memoria. Ciò che viene salvato in memoria non è nient'altro che un valore numerico sempre progressivo che indica il numero di impulsi letti dal Flow Meter.

Il fatto che questo numero non sia relativo alla sola sessione di irrigazione, ma all'intera vita del dispositivo, garantisce che il dato trasmesso via radiofrequenza sia idempotente, ovvero in caso di perdita di pacchetti o di ritrasmissioni dello stesso non venga generato alcun errore sul processamento del dato. Infatti, nel caso in cui vi fossero operazioni che mettono in relazione il valore del dato precedente con il dato corrente (esempio una somma tra i due), la trasmissione multipla o la mancata trasmissione di un pacchetto comporterebbero questo tipo di errore.

Garantendo una trasmissione giornaliera di un dato sempre progressivo, il peggiore scenario che può essere riscontrato è la mancata trasmissione di un dato per un tempo che può risultare infinito (in caso di assenza di copertura radio), ma nel momento in cui questo trasferimento avverrà, il dato sarà esattamente quello atteso. Prima di poter procedere con la stesura della macchina a stati sul Firmware del dispositivo, è necessario produrre un elenco di tutti i componenti del microcontrollore che dovranno essere impiegati nello sviluppo.

Questi dovranno poi essere inseriti all'interno di un generatore di codice sorgente che produrrà le librerie per l'interazione con l'hardware, delle quali parleremo più tardi, da utilizzare nello sviluppo del Firmware. Nel nostro caso sono stati utilizzati:

- LPTIM: per il conteggio degli impulsi
- RTC per l'utilizzo dei timer in grado di svegliare il microcontrollore generando un interrupt

- Memoria FLASH per il salvataggio del dato di conteggio
- SPI per la comunicazione con il transceiver LoRa ed il conseguente invio dei dati
- ADC per la lettura della batteria rimanente

Nella fase 2 vediamo l'introduzione dell'ambiente di sviluppo fornito dalla casa produttrice della scheda BL072ZLRWAN1 STMicroelectrinics inclusa in questo progetto. Questo software è denominato STM32CubeIDE.

STM32CubeIDE è una piattaforma di sviluppo C / C ++ avanzata in grado di effettuare la configurazione delle periferiche, auto generazione del codice, compilazione e funzionalità di debug per microcontrollori e microprocessori STM32. Si basa sul framework ECLIPSE / CDT e sulla toolchain GCC per lo sviluppo e GDB per il debug. Consente l'integrazione delle centinaia di plugin esistenti che completano le funzionalità dell'IDE ECLIPSE. STM32CubeIDE integra tutte le funzionalità STM32CubeMX.

STM32CubeMX è uno strumento che consente, dopo aver selezionato un MCU o MPU STM32, di creare il progetto e generare il codice di inizializzazione e di utilizzo delle sue periferiche.

In qualsiasi momento durante lo sviluppo, l'utente può tornare ad utilizzare questo strumento e modificare le configurazioni precedentemente impostate relative alle periferiche o al middleware e rigenerare il codice di inizializzazione senza alcun impatto sul codice utente.

STM32CubeIDE include analizzatori di build e stack che forniscono all'utente informazioni utili sullo stato del progetto e sui requisiti di memoria. STM32CubeIDE include, inoltre, tutte le principali funzionalità di debug standard e avanzate, tra cui visualizzazioni di registri della memoria, memorie e registri periferici, nonché controllo variabile live, interfaccia Serial Wire Viewer o analizzatore di guasti.

Per la creazione del progetto è stato utilizzato lo strumento fornito STM32CubeIDE, una volta selezionato il modello del microcontrollore utilizzato (STM32L072CZ) ed inserite le periferiche utilizzate con i relativi GPIO associati, è stato possibile strutturare il progetto e generare le librerie.

Quest'ultime sono denominate Hardware Abstraction Layer e forniscono una serie di API (interfacce di programmazione dell'applicazione) in grado di far interagire il livello applicativo (applicazione, librerie e stack) con l'hardware. Sono inoltre costruite direttamente attorno a un'architettura generica e consente ai layer incorporati, come il layer middleware, di implementare le loro funzioni senza sapere in dettaglio come utilizzare l'MCU. Questa struttura migliora la riusabilità del codice della libreria e garantisce una facile portabilità ad altri dispositivi.

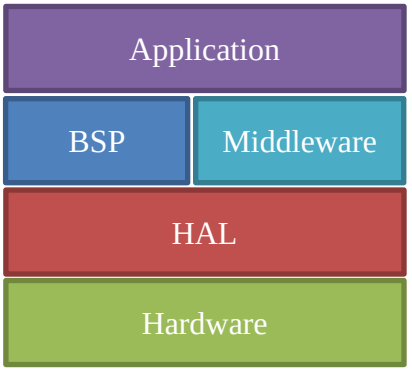


Figura 3.5.2 – Stack del Firmware di un dispositivo embedded

In Figura 3.5.2 viene mostrato lo Stack di un'applicazione per MCU ST.

Partendo dal basso troviamo il livello Hardware, ovvero l'insieme di tutte le periferiche utilizzabili dal microcontrollore come per esempio I/O paralleli, I/O seriali, I/O analogici, PWM, Timer etc…

Per interagire con questo livello è necessaria la scrittura/lettura diretta nei registri del MCU al fine di poter inizializzare correttamente, in base alla funzionalità richieste dai livelli superiori, le periferiche di interesse.

Salendo troviamo appunto le HAL, il cui compito è quello di astrarre l'utilizzo diretto delle periferiche Hardware, fornendo un insieme di API semplificate composte da funzioni che non sono nient'altro che una sequenza di scritture e letture nei relativi registri associati alle periferiche stesse. Queste vengono pienamente sfruttate dal livello applicativo ed il pre-applicativo.

Con livello pre-applicativo si intende quello strato intermedio che utilizza le funzioni della libreria HAL per gestire i vari componenti connessi alla MCU, come per esempio EEPROM esterne, display, led ect..
Fanno parte di questo livello il BSP ed il Middleware.
Nei sistemi embedded, il Board Support Package (BSP) rappresenta lo strato di software contenente i driver specifici dell'hardware ed altre routine che consentono a un particolare sistema operativo RTOS (o livello applicativo in questo caso) di funzionare su di un determinato hardware personalizzato. I BSP sono in genere personalizzabili, consentendo all'utente di specificare quali driver e routine devono essere inclusi nella build in base alla selezione di opzioni hardware e software.
Dall'altra parte abbiamo il middleware, sostanzialmente un insieme di driver per la gestione di periferiche più complesse che hanno uno stack protocollare più ampio come USB e TCP/IP. In alcuni casi anche i sistemi operativi real time vengono inclusi nello strato di middleware, ma tendenzialmente si preferisce aggiungere un ulteriore livello tra quello attuale e quello applicativo in cui collocarlo. Infine, in cima allo stack, troviamo il livello applicativo che comprende tutta la logica procedurale del dispositivo.
In questo progetto sono stati inseriti all'interno del BSP i processi di gestione del multiplexer, del generatore di impulsi per l'attivazione de il LPTIM e la lettura del valore di tensione sull'ADC.
Per quanto riguarda i multiplexer, essendo entrambi utilizzati come interruttori o comunque selettori di canale, le funzioni principali che il BSP di questi oggetti fornisce riguardano appunto l'inizializzazione dello stesso, quindi dei pin che esso utilizza e la funzione di selezione del canale da utilizzare.
Il generatore di impulsi consente attraverso un GPIO di generare un'onda quadra, accuratamente scelta di frequenza notevolmente maggiore per non interferire con il segnale generato dal reed switch.
La frequenza utilizzata è di 1 KHz con un duty cycle del 50%.
Vedremo poi nel livello applicativo come queste componenti vengono coordinate.
Analizzando ora lo strato middleware di questo progetto, troviamo tutta la gestione della comunicazione a radio frequenza LoRa.

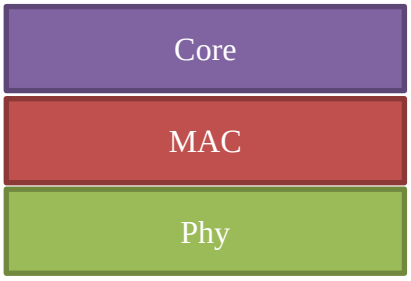


Figura 3.5.3 – Stack LoRa

La libreria di comunicazione via LoRa presenta uno stack applicativo come quello mostrato in Figura 3.5.3. Partendo dal basso vi è il livello fisico. Il livello fisico definisce i mezzi per la trasmissione dei bit anziché pacchetti di dati logici. Il flusso di bit viene qui convertito in un segnale fisico, trasmettendolo poi su un mezzo di trasmissione.

Il layer fisico fornisce normalmente un'interfaccia elettrica, meccanica e procedurale al mezzo di trasmissione. Qui vengono specificate le forme e le proprietà dei connettori elettrici, le frequenze su cui trasmettere, il codice di linea da utilizzare e parametri simili di basso livello. In questa libreria il layer fisico è composto da funzioni che mettono in comunicazione il livello MAC con il transceiver della trasmissione radio. Per far questo il livello fisico si basa sulle utilities HAL sfruttando il driver SPI. Questo rende possibile pilotare il transceiver grazie ad un set di comandi specifici denominati Hayes commands (o AT). Questi comandi furono inventati da Dannis Hayes nel 1981 e consentivano di controllare un determinato modello di modem. Da allora furono utilizzati in larga scalda su diversi componenti hardware relativi alle telecomunicazioni.

Oggi la stragrande maggioranza dei modem dial-up utilizza il set di comandi Hayes in numerose varianti. Il set di comandi è costituito da una serie di brevi stringhe di testo che possono essere combinate per produrre comandi per operazioni come la composizione, il riaggancio e la modifica dei parametri della connessione. Inizialmente queste istruzioni coprivano solo le operazioni supportate dai primi modem a 300 bit / s.

Con il susseguirsi degli eventi che hanno poi portato ad una innovazione tecnologica riguardante soprattutto la velocità di trasmissione, sono stati richiesti nuovi comandi per controllare funzionalità aggiuntive relative alle nuove velocità. È quindi stato necessario personalizzare questo standard in modo che ogni fornitore potesse coprire tutte le proprie esigenze, mantenendo però la struttura e la sintassi dei comandi di base. Non sarà parte di questo documento l'analisi dei comandi AT qui utilizzati, ma ci si limiterà alla descrizione partendo dall'astrazione del livello MAC e delle funzioni che questa libreria espone al livello applicativo.

Proseguendo sullo stack troviamo il livello MAC (Medium Access control).

Il livello MAC fa già parte dello standard LoRaWAN introdotto da Samtech.

Riallacciandoci alla descrizione presente nella parte introduttiva di questo documento, le funzioni principali implementate all'interno del livello MAC di questa libreria sono:

- La configurazione del dispositivo in modo da selezionare il range di frequenze da utilizzare per la trasmissione e ricezione dei dati in base allo standard della regione in cui il dispositivo stesso si trova. Per quando riguarda l'Europa la trasmissione LoRa deve avvenire nel range di frequenze 867-869 MHz.
- In questo intervallo di frequenze vengono definiti 10 canali di cui 8 con un data rate variabile da 250 bps a 5.5 kbps, 1 canale con velocità limitata a 11 kbps ed infine l'ultimo modulato in FSK con una velocità che può raggiungere anche i 50 kbps. Un secondo compito del livello MAC riguarda appunto l'assegnazione del canale sul quale il dispositivo dovrà comunicare.
- Un ulteriore compito è invece la gestione vera e proprio dell'accesso al canale che, come visto inizialmente, opera sulla basa del protocollo Aloha modificato per supportare le tre classi (A, B, C).

  Questo chiaramente include tutta la gestione dei messaggi di uplink, ma anche quelli di donwlink.

L'utilizzo di quest'ultima tipologica di messaggi potrebbe consentire eventualmente un controllo remoto oppure un aggiornamento del Firmware del dispositivo.

La filosofia, che in generale LoRa utilizza, è quella di non sprecare il tempo di trasmissione, ma di prestare sempre attenzione allo spreading factor utilizzato e massimizzare la velocità di trasmissione in quanto ciò porta a una maggiore durata della batteria e a un minore utilizzo del gateway. È dunque chiaramente importante riuscire a raggiungere sempre il gateway in qualunque condizione, ma senza sprecare risorse quali carica della batteria e tempo di occupazione del canale.

Sopra al livello MAC è posizionato lo strato identificato come Core, ovvero un set di funzioni che possono direttamente essere invocate al livello applicativo. Il loro scopo principale è la gestione degli indirizzi e l'autenticazione presso il gateway.

All'interno della libreria, per quanto riguarda gli indirizzi, si fa riferimento al file denominato Commissioning.h. Al suo interno vengono definiti staticamente il DevEUI (unico), DevAddr (non unico) e l'AppEUI (unico).

Per quanto riguarda invece le direttive utilizzate dal programma del microcontrollore sono principalmente le seguenti:

- void LORA_Init (*LoRaMainCallback_t** callbacks, *LoRaParam_t** LoRaParam);

Consente di inizializzare con i valori contenuti all'interno della struttura dati LoRaParam la libreria LoRaWAN, resettando la macchina a stati interna.

```
typedef struct sLoRaParam
{
        bool AdrEnable;
        int8_t TxDatarate;
        bool EnablePublicNetwork;
} LoRaParam_t;
```

AdrEnable indica l'abilitazione dell'Adaptive Datarate. Nel caso questo parametro venga impostato a false, risulta necessario specificare all'interno di TxDatarate la velocità di invio che si ha intenzione di utilizzare. Infine, vi è il flag che specifica se la rete alla quale si sta accedendo è pubblica o privata. Questi parametri sono stati rispettivamente impostati a False, Default, True all'interno del FlowMeter. Oltre alle impostazioni vi è però un ulteriore parametro da passare a questa funzione, ovvero l'insieme di tutte le callback relative ad eventi interni della macchina a stati LoRa. Questa è la struttura dati LoRaMainCallback.

```
typedef struct sLoRaMainCallback
{
/*!
 * @brief Process Rx Data received from Lora network
 * @param [IN] AppData structure
 */
        (1) void ( *LORA_RxData ) ( lora_AppData_t *AppData);

/*!
 * @brief callback indicating EndNode has just join
 * @param [IN] None
 */
        (2) void ( *LORA_HasJoined)( void );
/*!
 * @brief Confirms the class change
 * @param [IN] AppData is a buffer to process
 * @param [IN] port is a Application port on which Appdata will be sent
 * @param [IN] length is the number of received bytes
 */
        (3) void ( *LORA_ConfirmClass) ( DeviceClass_t Class );
/*!
 * @brief callback indicating an uplink transmission is needed to allow a pending downlink transmission
 * @param [IN] None
 */
        (4) void ( *LORA_TxNeeded) ( void);
```

```
/*!
 *\brief    Will be called each time a Radio IRQ is handled by the MAC layer.
 *\warning  Runs in a IRQ context. Should only change variables state.
 */
        (5) void ( *MacProcessNotify )( void );
} LoRaMainCallback_t;
```

La (1) rappresenta la callback invocata nel momento in cui un frame viene ricevuto.

La (2) identifica il momento in cui il nodo è riuscito ad autenticarsi presso un gateway.

La (3) conferma il cambiamento da una classe MAC ad un'altra.

La (4) chiede al nodo di effettuare una trasmissione in modo da consentire al gateway di generare successivamente un invio in downlink.

La (5) è una callback di gestione degli eventi relativi al processo di invio e ricezione a livello MAC.

In questo progetto, considerando il fatto che non vi sono trasmissioni in downlink e la classe MAC non viene mai cambiata, le uniche callback che sono state gestite sono la (2) e la (5).

La (2) perché chiaramente ci notifica che il dispositivo è pronto per inviare dati mentre la (5) perché in seguito ad una Send attesta l'avvenuto accodamento del pacchetto alla coda di trasmissione.

I casi in cui un pacchetto può non essere accodato sono per esempio la mancanza di copertura del gateway oppure il tempo trascorso tra un invio ed il successivo inferiore al livello di soglia per nodo (calcolato in base allo spreading factor utilizzato).

- void LORA_Join(*void*);

Effettua l'autenticazione presso il gateway. Vi sono per due modalità di Join.

La prima è Over-the-Air Activation (OTAA), qui i dispositivi eseguono una procedura di join con la rete, durante la quale viene assegnato un DevAddr dinamico e le chiavi di sicurezza vengono negoziate tra server e dispositivo stesso.

Questa è chiaramente la formula più comoda e duttile di accesso alla rete.

Vi è però una seconda modalità, Activation by Personalization (ABP) questa è quella adottata anche in questo progetto e costringe lo sviluppatore a definire tutti gli indirizzi staticamente, nel nostro caso facciamo riferimento al file commissioning.h. Utilizzando l'ABP l'operazione di join diventa fittizia e la callback viene chiamata immediatamente.

- bool LORA_send(*lora_AppData_t** AppData, *LoraConfirm_t* IsTxConfirmed);

L'ultima funzione dello strato Core utilizzata è la Send.

Questa necessita del frame di invio incapsulato all'interno della struttura dati lora_AppData_t e di un flag che specifichi se deve essere richiesto l'Acknoledge oppure no.

```
typedef struct
{
 /*point to the LoRa App data buffer*/
 uint8_t* Buff;
 /*LoRa App data buffer size*/
 uint8_t BuffSize;
 /*Port on which the LoRa App is data is sent/ received*/
 uint8_t Port;
} lora_AppData_t;
```

All'interno della struttura dati troviamo il buffer da inviare, la dimensione del buffer ed infine la porta dalla quale il pacchetto deve essere spedito. Il valore di ritorno di questa funzione identifica il risultato dei controlli sul frame da inviare in corrispondenza con lo stato della macchina a stati del livello MAC.

Se vi è stato qualche problema il valore ritornato è False.

L'ultimo livello del firmware del dispositivo è quello applicativo. Qui viene descritta ed implementata considerando tutti i minimi particolari la macchina a stati presente in Figura 3.5.1. Ora non torneremo a descrivere il funzionamento di questa, ma ne analizzeremo l'implementazione delle singole parti più importanti. Partiamo innanzitutto dai timer logici, quindi, sempre facendo riferimento allo schema della macchina a stati, T1 T2 e T3. La libreria utilizzata in questo caso è la TimerServer, la quale si basa sull'utilizzo dell RTC per la gestione del tempo e la generazione dell'interrupt. La struttura dati da compilare al fine di poter gestire il timer è la TimerEvent.

```
typedef struct TimerEvent_s
{
    uint32_t Timestamp;
    uint32_t ReloadValue;
    bool IsStarted;
    bool IsNext2Expire;
    void ( *Callback )( void* context );
    void *Context;
    struct TimerEvent_s *Next;
} TimerEvent_t;
```

Timestamp: Valore di millisecondi rimanenti alla generazione dell'evento.

ReloadValue: Valore da assegnare a Timestamp quando il timer viene riavviato.

IsStarted: Indica se il timer è avviato.

IsNext2Expire: Indica se è il prossimo timer a generare l'evento espirazione.

Callback: Puntatore alla callback da invocare quando l'evento viene generato.

Context: Puntatore all'oggetto che l'utente può utilizzare per scopi personali e che si ritroverà poi nella callback.

Next: Puntatore al timer successivo (viene utilizzata una gestione a lista).

La compilazione di questi campi non deve essere però effettuata manualmente, ma è necessario utilizzare le funzioni di libreria relative che effettuano questa operazione di inizializzazione della struttura.

- void TimerInit(*TimerEvent_t* * obj, *void* (*callback)(*void* *context));

Questa funzione crea un timer la cui callback di espirazione viene passata come puntatore ed invocata dalla libreria nel momento in cui il periodo impostato termina.

Per configurare questo tempo è necessario invocare la seguente funzione.

- void TimerSetValue(*TimerEvent_t* * obj, *uint32_t* value);

Il valore passato indica i millisecondi che anticipano la generazione dell'evento e la conseguente invocazione della calback.

Infine, per avviare il conteggio da parte del timer è necessario eseguire la relativa funzione di start.

- void TimerStart(*TimerEvent_t* * obj);

Considerando il fatto che tutti i timer riguardano o la lettura del valore di conteggio del FlowMeter o l'invio del dato, questi sono stati utilizzati all'interno del codice di gestione del FlowMeter (FlowMeterManager).

Nel codice del main viene inizializzato tutto il processo del conteggio invocando la funzione InitFlowCounter del FlowMeterManager. Questa imposta lo stato corrente del dispositivo in IDLE

(nessuna irrigazione avviata), inizializza i timer T1 (Reset del valore di soglia ogni ora) e T3 (Invio del dato giornaliero), avvia il conteggio de il LPTIM in modalità con interrupt impostando il valore di comparazione a tre ed infine genera i cinque impulsi di burst per l'attivazione de il LPTIM utilizzando le routine del BSP.

Come sappiamo, sarà però solo il quarto impulso a generare l'evento di interrupt de il LPTIM. Questo perché la callback che viene sovrascritta è:

void HAL_LPTIM_AutoReloadMatchCallback(*LPTIM_HandleTypeDef** hlptim)

Così facendo l'evento che realmente scatenerà l'interrupt, non sarà il raggiungimento del registro comparatore da parte di quello contatore, ma l'azzeramento di quest'ultimo che avverrà alla lettura di un impulso successivo al raggiungimento dell'uguaglianza dei valori dei due registri.

Qui viene poi attivata la fase di irrigazione impostando lo stato del dispositivo a FLOODING. Così facendo viene bloccato T1, avviato T2 (timer di lettura ed invio dati durante la fase di irrigazione) e configurato il LPTIM in modalità NO-IT (polling). Le funzioni relative al LPTIM sono prese direttamente dalle librerie HAL. Anche qui è necessario generare il burst dei cinque impulsi in modo da consentire al LPTIM di attivarsi.

Il cambio di stato viene applicato attraverso la funzione SetNextState() invocata all'interno del codice Main, questo perché ogni volta che termina la gestione di un interrupt (per esempio quello di un timer), il program counter del MCU punta alla istruzione successiva della chiamata che imposta la modalità di esecuzione in Stop.

LPM_EnterLowPower() consente di entrare in modalità di stop, impostando come istruzione successiva quella seguente. Questa istruzione viene utilizzata all'interno del ciclo while(1) del main in modo che una volta terminato il cambio di stato l'MCU possa tornare in STOP e ridurre al minimo il consumo energetico.

Un altro importante controllo che è stato inserito nella funzione Main, è la verifica della necessità di inviare un messaggio. I vari interrupt, per inviare un messaggio, avanzano semplicemente una richiesta di invio impostando un flag a 1 o a 2 per segnalare al Main che vi sono messaggi pendenti ed è necessario effettuare una trasmissione Lora appena possibile. Il valore 1 rappresenta un invio con Ack, mentre 2 viene utilizzato per gli invii Uncornfirmed.

I punti della macchina a stati in cui è previsto l'invio di un messaggio al server sono tre. Un primo messaggio viene spedito nel momento esatto in cui si entra in fase di irrigazione, ovvero alla lettura del quarto impulso sulla linea di ingresso de il LPTIM. Una seconda sequenza di messaggi viene spedita ogni volta che scatta il timer T2, ogni 4 minuti. Questa sequenza termina solo nel momento in cui la lettura corrente del contatore su il LPTIM ritorna lo stesso valore della lettura precedente.

In questo caso viene identificata la condizione di STOP ed è proprio qui che si verifica l'ultimo dei tre casi di invio del messaggio. La funzione che viene utilizzata per inviare il messaggio è la seguente.

static void Send(*LoraConfirm_t* msgtype)

Il parametro msgtype identifica la tipologia di invio, con o senza richiesta di Acknowlegde.

Per garantire un risultato completo in grado di rappresentare l'intera fase di irrigazione, il primo ed il terzo tipo di messaggio sono inviati utilizzando la richiesta dell'Ack, mentre la tipologia due è Unconfirmed.

La struttura di pacchetto inviato dal FlowMeter è suddivisa in tre livelli come è possibile notare in Figura 3.5.4. Il primo è quello fisico, qui troviamo una parte di preambolo più header aggiunta dal transceiver radio, un payload ed infine un CRC di controllo di tutto il pacchetto. Il payload rappresenta il livello MAC, composto a sua volta da un header, un payload e un Message Integrity Code (codice di controllo).

Infine, vi è l'ultimo livello, ovvero quello del frame. Anche qui abbiamo un frame header contenente l'indirizzo del dispositivo mittente ed altri parametri relativi alle caratteristiche del dispositivo, la porta sorgente ed il frame payload. Nel nostro caso il frame payload è stato strutturato in quattro sezioni:

- Session status (1 byte), questo valore descrive lo stato attuale del FlowMeter in base alla seguente tabella.

Valore	Descrizione
0	Nessuna irrigazione in corse
1	Irrigazione appena iniziata
2	Irrigazione in corso
3	Irrigazione appena terminata

- Session counter (8 bytes), numero progressimo che conta le sessioni di irrigazione attivate durate tutta la vita del FlowMeter.
- Impulse counter (8 bytes), numero progressivo di impulsi registrati sull'ingresso de il LPTIM durante l'intera vita del FlowMeter.
- Battery percentage (1 byte), percentuale di batteria rimanente. Questa viene calcolata leggendo l'ADC prima di inviare il pacchetto.

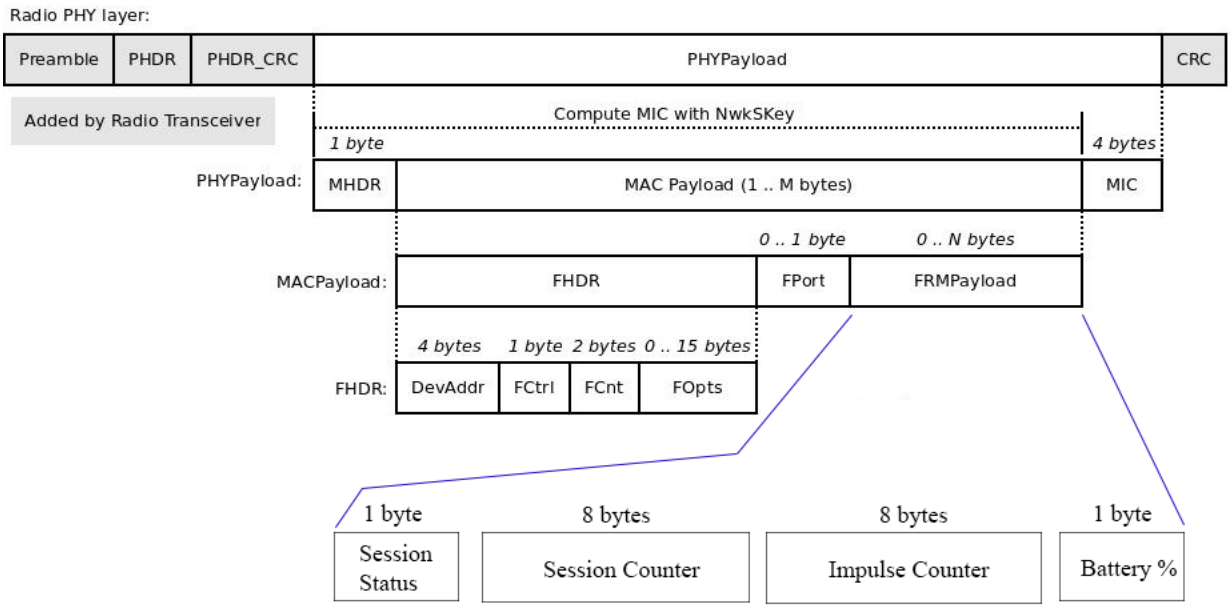


Figura 3.5.4 – LoRaWAN struttura del pacchetto

Al termine di ogni invio di pacchetto, il Flow Meter prevede il salvataggio dei dati nella memoria Flash interna del microcontrollore. I dati vengono salvati lì in quanto, per definizione, questo tipo di memoria presenta come caratteristica principale proprio la persistenza dei dati. Questo significa che nel momento in cui il microcontrollore si spegne i dati al suo interno non vengo persi, ma rimangono sempre disponibili per letture successive. Andando più nello specifico, i dati che ci interessa salvare nella memoria Flash sono l'Impulse Counter (contatore del numero di impulsi letti sul reed switch) e il Session Counter (numero di sessioni di irrigazioni effettuate).

Per poter leggere e scrivere in Flash è prima necessario comprenderne la struttura ed il corretto utilizzo. Facendo riferimento al microcontrollore che è stato utilizzato in questo progetto, la memoria Flash è divisa in settori ognuno composto di 32 pagine. Ogni pagina ha una dimensione di 32 words (32 bit) quindi 128 bytes. Il totale dell'ammontare di memoria Flash in questo MCU è 192 Kbytes.

Mentre la lettura può essere effettuata senza limiti di dimensione minima, quindi anche byte per byte, la scrittura e la cancellazione hanno delle limitazioni ben precise. La lettura consiste in un normale utilizzo

dei puntatori con indirizzi di memoria relativi alla memoria Flash specificando la dimensione del dato che si vuole leggere, a seguire le definizioni di lettura di questo progetto.

#define ImpulseCounter (*(*uint64_t**)FLASH_IMPULSE_COUNTER_ADDR)

#define SessionCounter (*(*uint64_t**)(FLASH_SESSION_COUNTER_ADDR))

Questi due contatori sono stati posizionati nella medesima pagina di Flash, uno dopo l'altro. In questo si è potuto dimensionare al minimo lo spazio utilizzato per la definizione di questi due valori. Nel nostro caso questo non sarebbe stato un problema, ma è sempre buona norma non sprecare spazio in memoria in quanto in aggiornamenti software futuri potrebbe essere richiesta una quantità di bytes contigui che non sempre può risultare disponibile, ad esempio un'area di swap per l'aggiornamento firmware. La scrittura e la cancellazione, invece, utilizzano funzioni definite nelle HAL, in particolare

HAL_StatusTypeDef HAL_FLASH_Program (*uint32_t* TypeProgram, *uint32_t* Address, *uint32_t* Data)

HAL_StatusTypeDef HAL_FLASHEx_Erase (*FLASH_EraseInitTypeDef* *pEraseInit, *uint32_t* *PageError)

Dove la prima consente di scrivere in un determinato indirizzo della Flash una word, mentre la seconda cancella una o più pagine in base a cosa viene indicato all'interno della struttura dati FLASH_EraseInitTypeDef.

Questo comporta dunque che ogni qualvolta si voglia salvare un nuovo valore di uno dei due contatori, sia in realtà necessario cancellare l'intera pagina e riscriverla completamente trovandosi entrambi nella stessa pagina. La limitazione della Flash è dunque proprio questa, la difficolta di sovrascrittura.

Questa operazione, come già citato in precedenza, viene effettuata però solo al termine della sessione di irrigazione. La motivazione è che in realtà questa tipologia di memoria prevede un numero di cancellazioni della singola pagina limitato, all'esaurimento la pagina rischia di corrompersi da un momento all'altro e nel caso questo accada non vi è più la possibilità di scriverci. Essendo la nostra scrittura effettuata ad indirizzi statici, questa usura comporta l'interruzione del funzionamento del dispositivo hardware. L'ordine di grandezza del numero massimo di scritture in flash sulla singola pagina si aggira però sulle centinaia di migliaia.

Un'ultima parte software di rilevanza in questo progetto è la lettura della batteria rimanente. Per poter fornire un valore veritiero e coerente che stimi la percentuale di carica rimanente nelle batterie, è necessario effettuare una lettura di tensione tramite l'ADC. Come già visto nella sezione dedicata all'implementazione hardware, il circuito di lettura dell'ADC è provvisto di un MUX in gradi di attivare o disattivare l'alimentazione sul partitore. Il partitore è fondamentale per consentire di leggere sull'ADC dei valori compresi tra 0 e 3.3 V. Nel nostro caso il partitore è impostato al 50%, ovvero le due resistenze sono dello stesso valore e di conseguenza utilizzando per esempio una batteria da 4.5V avremo un massimo di lettura a 2.3V. Presupponendo di avere il regolatore di tensione che necessita di almeno 3.4V per poterne fornire 3.3V al circuito, il margine di utilizzo della batteria risulta essere 1.1V (4.5-3.4). Quindi 4.5V equivarrà al 100% di carica mentre 3.4V allo 0%.

Purtroppo, la curva di scarica della batteria non è mai lineare, anzi risulta spesso imprevedibile e spesso non tutti i produttori di batterie la forniscono. La soluzione più semplice per ovviare a questo problema è chiaramente l'approccio lineare, ovviamente preso in considerazione anche in questo progetto per il semplice fatto che, come vedremo successivamente, la carica della batteria risulta essere un'informazione non rilevante dato il consumo energetico del dispositivo.

La funzione che abilita l'alimentazione del circuito sul MUX, legge i valori dell'ADC e calcola la percentuale di batteria rimanente è la seguente.

```
int HW_BAT_Read(void);
```

Andando nel dettaglio di questa funzione, una volta abilitato il partitore è necessario attendere un lasso di tempo tale da consentire al condensatore in parallelo al partitore di caricarsi completamente in modo poi da essere in grado di eseguire una lettura sull'ADC corretta. Quest'ultima viene effettuata con le seguenti linee di codice

1) SupplyVoltage = HW_AdcReadChannel(ADC_CHANNEL_VREFINT);
2) BatteryVoltage = HW_AdcReadChannel(ADC_CHANNEL_4);
3) SupplyVoltage = ((*uint32_t*)VDDA_VREFINT_CAL * (*VREFINT_CAL)) / (*uint32_t*)SupplyVoltage;
4) BatteryVoltage = (BatteryVoltage/4096)*SupplyVoltage;

Con la 1. e la 2. vengono rispettivamente letti i valori di tensione di alimentazione e tensione sul partitore. Le 3. Invece consente di ottenere il valore in Volt esatto della tensione di alimentazione utilizzando i valori salvati dal produttore in determinate aree di memoria. VREFINT_CAL è l'indirizzo d memoria in cui è posizione il valore di calibrazione dell'ADC, mentre VDDA_VREFINT_CAL è l'effettivo valore di uscita VREFINT convertito dall'ADC.
Supponendo di utilizzare, come nel nostro caso, un ADC a 12 bit con un valore massimo di 4096, è possibile, attraverso una proporzione, ottenere il valore di tensione effettiva sul partitore usando la formula descritta dalla linea di codice 4.

Per conoscere a quanto questa tensione corrisponda in percentuale di batteria rimanente, bisogna rapportarlo in base a quella che è il valore di divisione del partitore. Noi abbiamo utilizzato due resistenze equivalenti, di conseguenza basterà moltiplicare il valore ottenuto per 2 e si otterrà la tensione corrente della batteria. A questo punto basterà sottrarci il valore minimo di tensione richiesta dal regolatore e successivamente rapportarla con 1.1 V per ottenere la percentuale di batteria rimanente linearmente approssimata.

3.6. Progettazione Meccanica

All'interno di questa sezione verrà brevemente descritta la parte di progettazione e creazione del prototipo utilizzato per le verifiche e misurazioni di convalidazione del progetto.

Prima di tutto è necessario identificare bene il contesto all'interno del quale il Flow Meter verrà impiegato, descrivendo al meglio le condizioni ambientali e territoriali alle quali verrà sottoposto.

Figura 3.6.1 – FlowMeter in campo

Come è possibile notare in Figura 3.5.1, questo dispositivo verrà posizionato in un sistema di irrigazione all'aperto quindi oltre a dover essere perfettamente resistente all'acqua dovrà possedere un'alta resistenza al deterioramento dovuto agli agenti atmosferici. Analizziamo ora la conduttura contente il contatore meccanico ed il reed switch. Questa, attraverso i bulloni di fissaggio presenti sulle giunture ed alle guarnizioni interne, garantisce una perfetta condizione di isolamento evitando così sprechi idrici sul flusso di canalizzazione. Il cavo elettrico che trasporta il segnale generato dal reed switch è un tradizionale cavo bipolare da esterni resistente all'umidità. Non ci sono però state fornite informazioni circa i tempi di usura e deterioramento. Rimane quindi da progettare il rivestimento esterno della scheda riguardante la parte di analisi e trasmissione dei dati. È necessario che essa sia sì impermeabile e resistente agli urti in modo da non danneggiare il circuito interno in caso di umidità e/o urti, ma che allo stesso tempo risulti di facile accesso per poter sostituire le batterie ed accendere, spegnere o riavviare il dispositivo in caso di malfunzionamento. Inoltre, questo contenitore avrà la necessità di essere montato presumibilmente attorno a condotti cilindrici e possedere una modalità di fissaggio/smontaggio semplice ed alla portata di tutti. Per ovviare a problemi di costi rispettando le criticità sopra citate, si è optato per l'utilizzo di una scatola di derivazione IP56 abbinata a un pressacavo impermeabile contenente una guarnizione in gomma per l'isolamento. La scatola è stata poi livellata utilizzando uno strato in plastica. Questo serve principalmente per poterci fissare i distanziali per sollevare il PCB. Il foro di incastro del pressacavo è stato eseguito in modo da risultare perfettamente allineato con il punto di ingresso del jack relativo al cavo proveniente dal reed switch.

Per agevolare la sostituzione delle batterie ed il reset del dispositivo si è optato per una scatola di derivazione con la possibilità di svitare solamente due viti su quattro per aprire il coperchio. Così facendo il coperchio rimarrà sospeso formando un angolo di 90° con il basamento della scatola.

Questo contenitore è sviluppato in PVC e garantisce quindi una maggiore longevità in quanto non arrugginisce a differenza di uno in ferro.

4. Validazione/Valutazione sperimentale

4.1 Verifiche e acquisizione dati post-sviluppo

La pressione che spinge a prototipare le caratteristiche o gli elementi di un design può far perdere di vista il prodotto finale risultante. Concentrarsi su un prototipo porta esattamente a ottenere solo questo: un prototipo, e forse uno dei molti che si andranno a realizzare man mano che il design verrà perfezionato. È dunque necessario guardare avanti, oltre la fase di prototipazione, pensare in termini di prodotto finale e delle caratteristiche prestazionali richieste.

L'obiettivo di tale approccio è ottenere prototipi che vadano ben oltre la mera rappresentazione fisica di un prodotto parzialmente funzionante, ma che preparino e facilitino, invece, il cammino verso la produzione finale, oltre che fornire preziose informazioni sulla performance e la conformità del prodotto.

Per compiere questo risulta fondamentale verificare che i prototipi siano in grado di soddisfare tutti i requisiti con test di affidabilità, accuratezza e conformità agli standard industriali, la verifica del ciclo di vita del prodotto e la dimostrazione che un prodotto soddisfi tutte le aspettative degli utenti. In tutti i prodotti tecnologici vi sono due facce appartenenti alla stessa medaglia, entrambe da sottoporre a test e verifiche pre-produzione: quella software e quella hardware.

Una delle maggiori disparità tra test software e hardware è che i test software possono essere copiati e riutilizzati, mentre i processi utilizzati per i test hardware devono essere sviluppati ad hoc. Come è ben noto, il software può essere facilmente modificato e migliorato attraverso l'introduzione di nuove versioni e aggiornamenti, mentre l'hardware ha costi di modifica più elevati e non sempre può essere sottoposto a refactoring dopo la produzione. La soluzione maggiormente adottata al giorno d'oggi è la metodologia di test agile. Ciò significa che i team possono scrivere, assegnare, pianificare e automatizzare i test secondo

necessità. Ciò garantisce un costante aggiornamento della parte software e aiuta a mantenere alti gli standard di qualità nel tempo.

Tuttavia, i test hardware possono adattarsi ad anche solo un piccolo errore. Quando si pensa a come viene realizzato un dispositivo, è facile tornare all'immagine di una catena di montaggio in cui tutte le parti vengono sempre assemblate allo stesso modo. Qualsiasi deviazione in questo può causare anomalie una volta che il prodotto è uscito dalla catena produttiva. Proprio per questo i test e le verifiche su hardware devono sempre essere legati alla revisione di quest'ultimo. Ogni nuova revisione comporterà una variazione nella procedura di test.Tornando al progetto descritto in questo documento, possiamo identificare i test hardware e software da validare come segue. La parte di verifiche hardware di questo progetto coincide esattamente con le verifiche effettuate nella fase di progettazione. Nello specifico, riguardano la costruzione del filtro passa basso da applicare sulla lettura dello stato del reed switch e la scelta di quali componenti utilizzare per quanto riguarda sia i multiplexer che il regolatore. Essendo questo argomento già stato trattato nel capitolo relativo alla progettazione e sviluppo non verrà qui ripreso. Ciò invece su cui ci si concentrerà, è di come siano state prodotte le verifiche di funzionamento della componente software/firmware del MCU.

4.2 Verifiche software

La principale verifica software alla base del completamento del Flow Meter riguarda la precisione nel conteggio. In pratica è stato necessario verificare che il numero di impulsi generati dalla chiusura del reed switch fosse esattamene equivalente al conteggio effettuato dal MCU sul il LPTIM.

Per poter mettere in moto la pala presente all'interno della conduttura e di conseguenza generare il segnale, si è pensato di utilizzare un compressore in modo da poter simulare un flusso d'acqua costante.

Il flusso d'acqua simulato doveva emulare una portata almeno equivalente a quella massima descritta nelle specifiche della conduttura, che rapportata alle rotazioni dei contatori meccanici avrebbe generato un segnale ad una frequenza di 0.44 Hz. Di conseguenza la frequenza del segnale generato dalla rotazione della pala utilizzando il compressore doveva superare questo valore. Per sicurezza si è scelto di riprodurre un segnale 4 volte più veloce del massimo richiesto dalle specifiche, quindi la frequenza risultante era compresa tra i 1.5 e i 2 Hz. A inizio simulazione i contatori meccanici venivano posizionati sullo zero, in modo da poter calcolare l'esatto numero di impulsi generati al termine del tempo prefissato.

Per esempio, se alla fine della simulazione i tre contatori visualizzavano rispettivamente i valori 10 (quello dei m^3), 5 (quello dei 0.1 m^3) e 2 (quello dei 0.01 m^3) il numero di impulsi totali avrebbe dovuto essere 10*5= 50, in quanto un impulso viene generato ad ogni rotazione completa del contatore dei 0.01 m^3.

La verifica di correttezza del valore della frequenza del segnale generato veniva monitorata facendo compiere due giri completi del contatore meccanico dei 0.01 m^3 in un secondo. Questo è stato spanno metricamente possibile osservando, durante il rilascio dell'aria compressa, sia il contatore che un timer.

Tempo di durata del test	Impulsi generati	% errori nel conteggio	Distanza dal gateway	% messaggi persi
5 minuti	~550	0	50-60 mt	0
10 minuti	~1000	0	50-60 mt	0

Tabella relativa ai parametri utilizzati nei test

Le misure sono state effettuate considerando due lassi temporali rispettivamente di 5 e 10 minuti. In questo tempo la generazione del flusso d'aria attraverso il compressore non veniva mai interrotta.

Grazie alla potenza del compressore si sono potuti verificare conteggi fino all'ordine del migliaio di impulsi. La lettura degli impulsi registrati dal FlowMeter veniva direttamente fatta sulla piattaforma web di ARCES relativa alla visualizzazione dei dati, questo in modo che ci consentisse anche di verificarne la connettività. Il gateway LoRa era posizionato all'interno di un edificio posizionato a 50-60 mt in linea d'aria rispetto al punto di prova che si trovava invece all'aperto.

I test effettuati sono durati circa una giornata ed hanno evidenziato inizialmente alcuni errori per quanto riguarda l'endianess di codifica e decodifica degli UInt64 relativi a Session Counter ed Impulse Counter. Una volta corretti gli errori di parsing dei dati, si è potuto accertare che il conteggio avveniva in modo impeccabile, ovvero senza mai perdere nemmeno un impulso. Per stressare ulteriormente il sistema, si è provato a generare impulsi a frequenza variabile da 0.1 Hz a 2-3 Hz.

Non riscontrando nessuna anomalia durante tutta la fase di test, si è deciso di considerare il risultato soddisfacente.

4.3 Caratterizzazione energetica

Una sezione vitale di questo progetto risulta essere l'analisi dei consumi energetici della scheda una volta completato il prototipo a livello di funzionalità. Per le misurazioni è stata utilizzata la power monitoring insieme al software che ne consente la lettura dei dati. Questa consente di campionare la corrente richiesta da un circuito durante un lasso temporale di al massimo 60 secondi.

La frequenza di campionamento utilizzata è di 12802 Hz, ma viene ridotta nel caso venga analizzato più di un canale contemporaneamente (5974 Hz per due ingressi, 4266 Hz per tre etc…).

Nel nostro caso è stato sufficiente utilizzare un solo input di lettura per volta, questo ha inoltre aiutato a sfruttare la massima risoluzione di campionamento.

Date le limitazioni temporali della power monitoring, si è deciso di registrare tutte le possibili fasi di azione del FlowMeter, in particolare:

- Fase di Boot: istante iniziale, primi secondi dopo l'accensione dell'alimentazione al dispositivo (inizializzazione periferiche).
- Fase di avvio irrigazione: ricezione dei 4 impulsi sull'LPTIM, invio dei dati, generazione dei 5 impulsi per far ripartire il LPTIM in modalità NO-IT e attivazione timer dei 4 minuti.
- Fase intermedia: termine dei 4 minuti impostati che implicano l'invio del dato ed il conseguente ripristino del timer.
- Fase conclusiva: riconoscimento della fine del flusso d'acqua all'interno della conduttura, salvataggio in flash dei dati ed invio.

I valori relativi a corrente e tensione sono stati registrati istante per istante sia utilizzando lo spreading factor 7 (ovvero quello che aumenta la velocità di trasmissione diminuendone la portata) che il 12 (ovvero quello che aumenta la distanza di copertura rallentando la velocità di trasmissione). Analizzammo ora i grafici a confronto: in blu è rappresentata la corrente (mA) mentre in rosso la tensione (V). L'asse delle ascisse rappresenta la scala dei tempi in secondi.

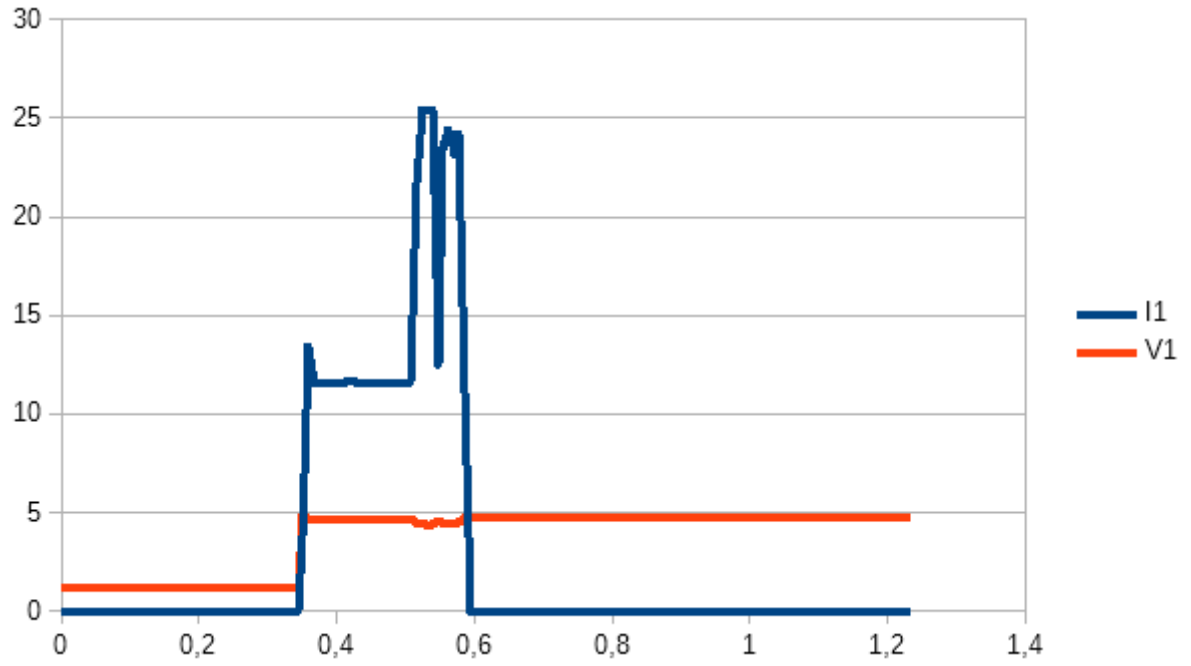


Figura 4.3.1 - Fase di boot

L'inizio della vita del Flow Meter è rappresentato dalla fase di boot. Come ci si accorge osservando la Figura 4.3.1, all'incirca sul valore dell'ascissa 0.3 si nota un rialzamento del valore della tensione a 5V, quello è il momento vero e proprio in cui è stata alimentata la scheda. Successivamente abbiamo un enorme consumo di corrente per circa 0.3 secondi, questa parte rappresenta l'inizializzazione di tutte le periferiche. Fatto questo il microcontrollore passa alla modalità di STOP, azzerando praticamente il livello di corrente richiesta. Questa fase è ovviamente comune sia a SF7 che SF12 in quanto non prevede invii di messaggi LoRa.

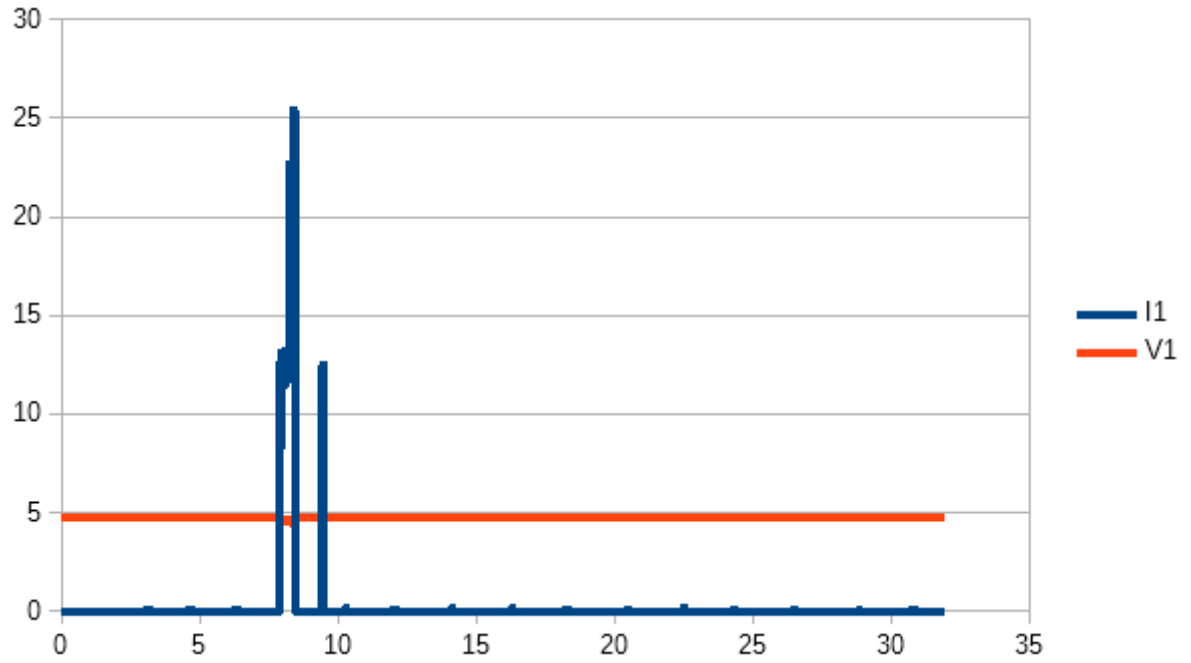


Figura 4.3.2a – Fase di inizio irrigazione SF7

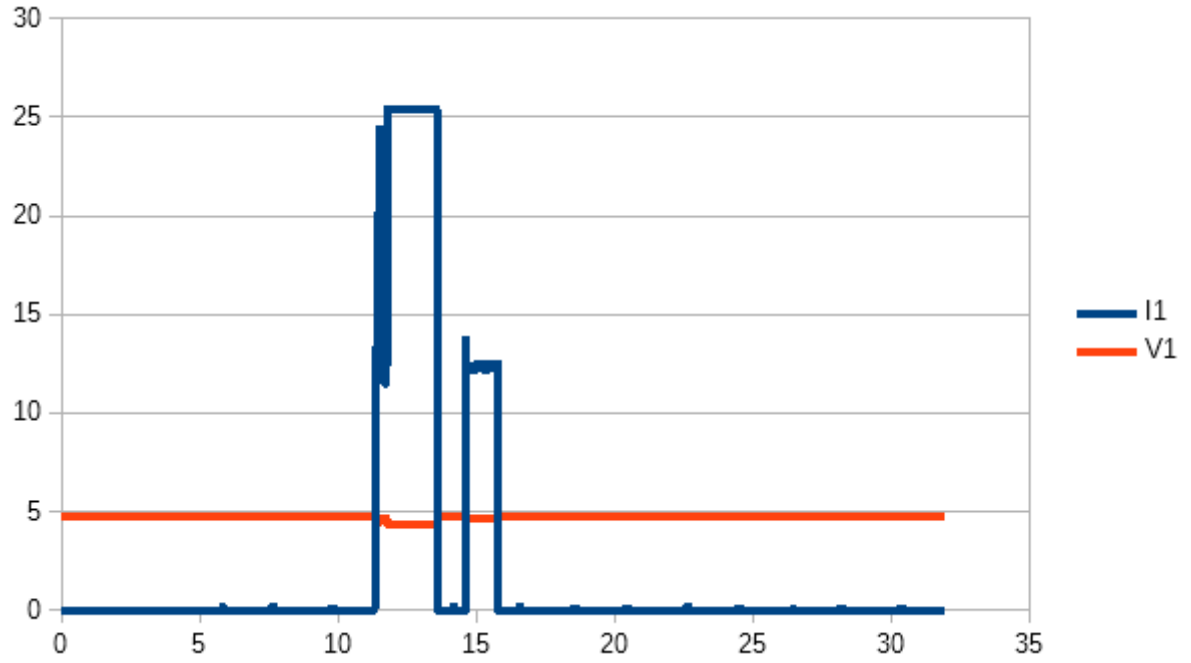


Figura 4.3.2b – Fase di inizio irrigazione SF12

A partire dalla fase di inizio irrigazione per tutta la durata del flusso d'acqua, si noteranno all'interno dei grafici dei piccoli picchi di corrente, questi rappresentano la chiusura del reed switch ed il conseguente

passaggio di corrente dalla resistenza di pull-up a ground. Analizzando in modo dettagliato le figure 4.3.2 a/b, è possibile notare una analogia. Entrambi i grafici sono composti da due aree di consumo: la prima rappresenta la trasmissione LoRa delle informazioni, mentre la seconda è la derivante risposta da parte del gateway in quanto il messaggio inviato nella fase di inizio richiede l'acknowledge. Ciò che invece differisce tra ii due grafici è la durata dell'atto di trasmissione e ricezione. Questo chiaramente va a pesare notevolmente sui consumi energetici.

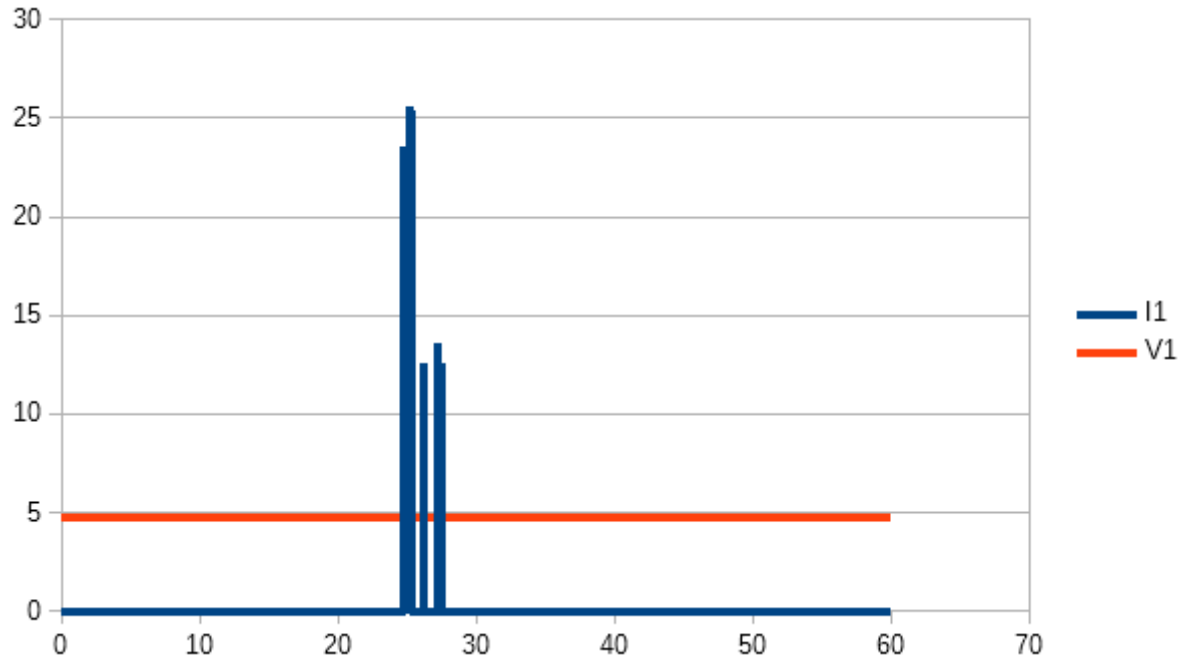


Figura 4.3.3a – Fase intermedia SF7

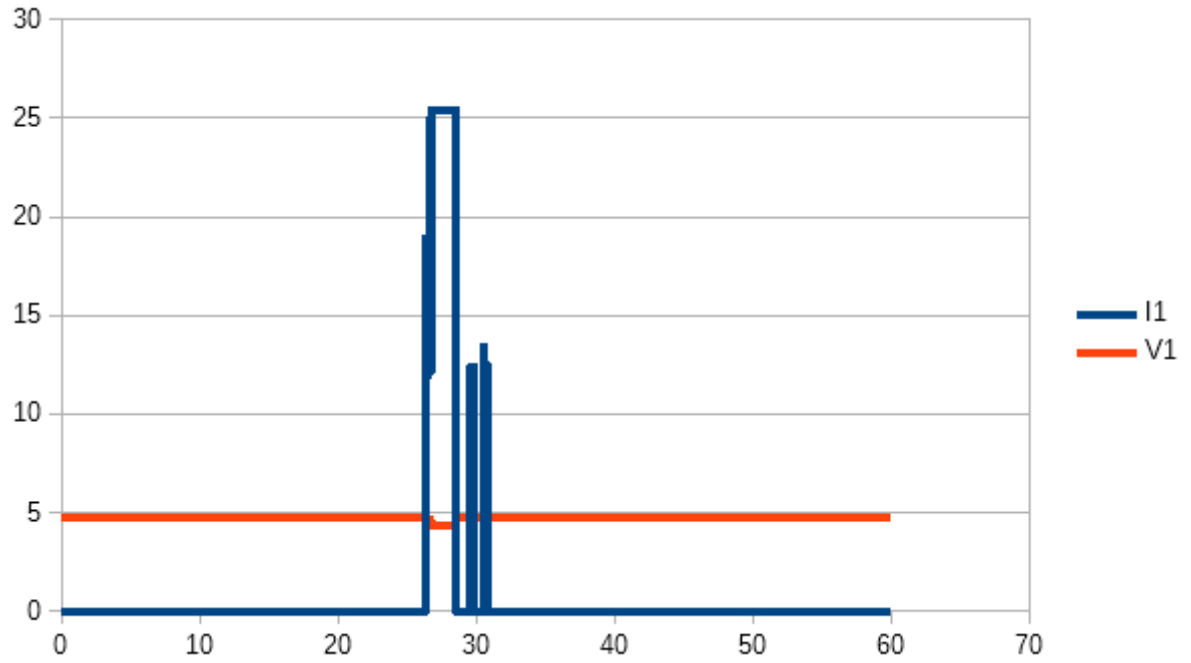


Figura 4.3.3b – Fase intermedia SF12

Nella fase intermedia, come in quella di inizio è presente un aumento dei consumi energetici dovuto alla trasmissione LoRa. Qui però a differenza della fase precedente, notiamo il picco di trasmissione seguito da due ulteriori picchi di corrente (figure 4.3.3a/b). In questo caso infatti il messaggio non richiede la conferma al gateway, ma vengono aperte due finestre per consentire eventuali messaggi in downlink come previsto dal protocollo. Ovviamente anche qui a trasmissione in SF12 ha una durata maggiore di quella in SF7.

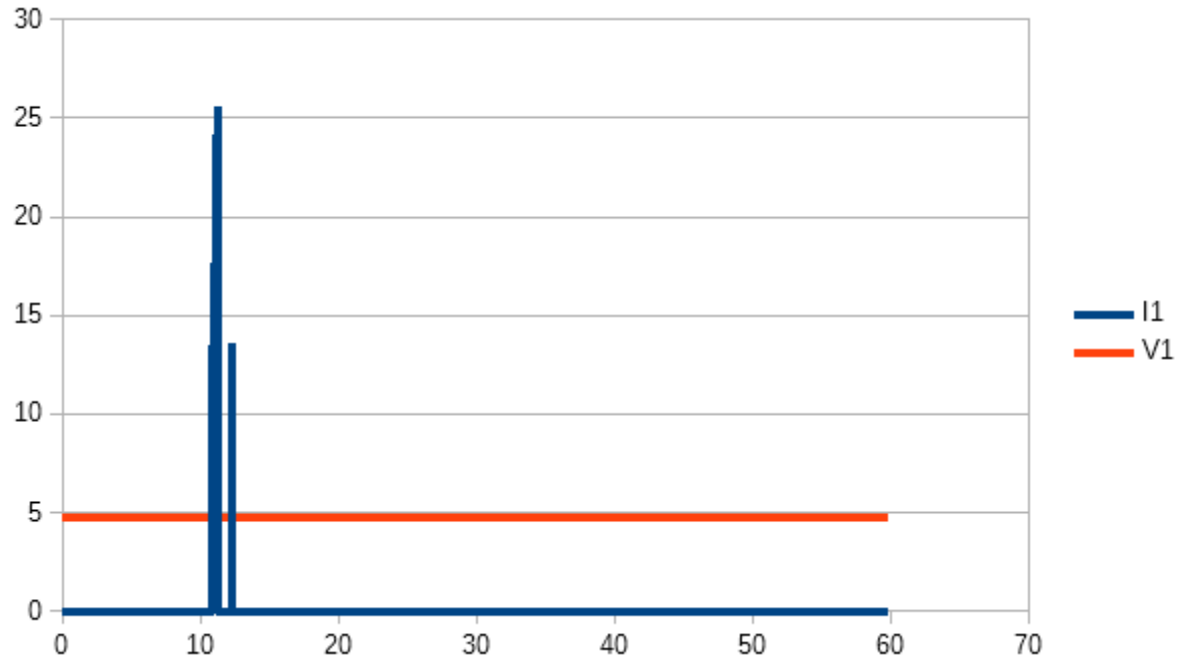


Figura 4.3.4a – Fase di terminazione SF7

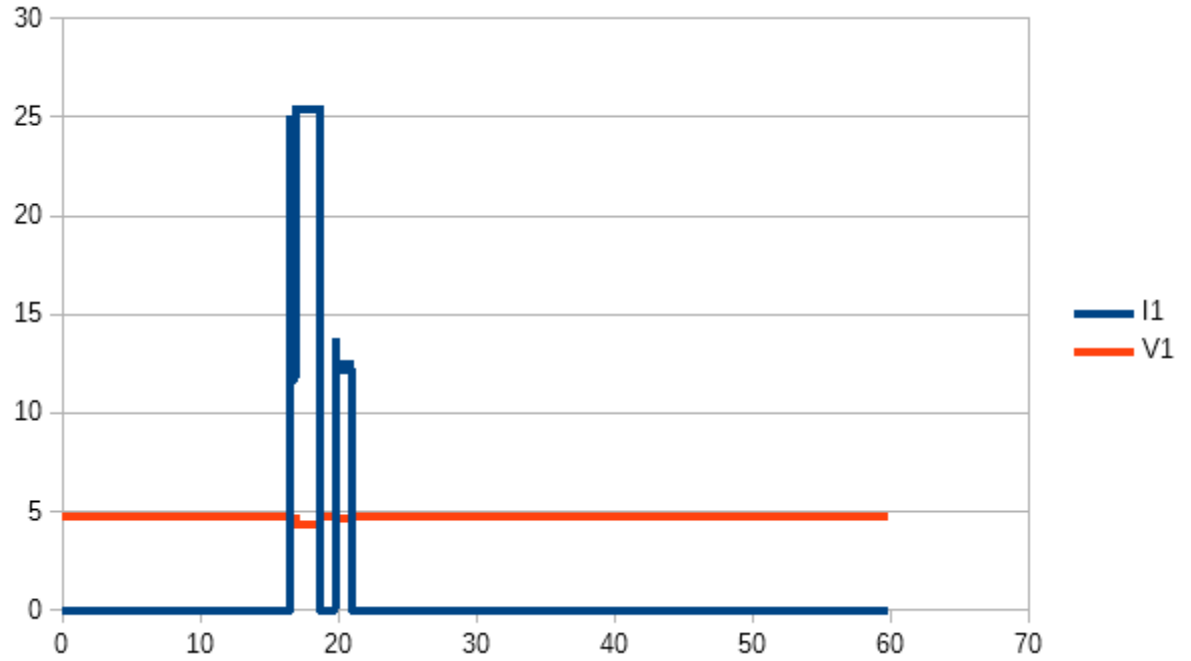


Figura 4.3.4b – Fase di terminazione SF12

Infine, troviamo la fase di terminazione, praticamente equivalente alla fase di start dove l'invio del messaggio avviene con conseguente richiesta di acknowledge.

Qui non sono presenti come è possibile notare dal grafico i picchi di corrente relativi alle chiusure del reed switch in quanto il flusso di acqua all'interno della conduttura è stato interrotto. Dai dati di questi grafici sono poi state ritagliate le varie fasi di invio dei messaggi LoRa (in campioni) eliminando i periodi di inattività del microcontrollore. In più è stato ritagliato 1 secondo di idle, ovvero in cui vi è il consumo minimo di corrente da parte della scheda, ed i 2.2 secondi relativi al singolo impulso. Con queste sei finestre di campionamento è stato possibile simulare una intera giornata di attività all'interno della quale vi era una sola sessione di irrigazione della durata di 2 ore. È stato scelto questo rapporto in quanto giudicato uno scenario medio verosimile dell'utilizzo del Flow Meter in base ai dati fornitoci dal consorzio di bonifica. Questa simulazione è stata ovviamente costruita sia per lo spreading factor 7 che per il 12. Riportiamo ora a seguire la tabella di simulazione costruita sia per i valori di corrente che per quelli di tensione.

TIME (S)	**I (MA)**	**V (V)**	**NUMERO RIPETIZIONI**	**DESCRIZIONE**
6.685902	0.00703	4.820293	3	Singolo Impulso * 3 volte
2.048664	4.091146	4.758043	1	Avvio flusso e invio dati
7198.488	0.00703	4.820293	3230	Singolo Impulso per 2h – 2.2 secondi (tempo del 4° impulso)
1.556008	5.360314	4.738778	1	Termine del flusso, salvataggio dati in flash ed invio (dopo 2h e circa 6 minuti in media)
83.05731	3.955212	4.761061	31	Ogni 4 minuti I dati venivano processati e salvati
79191.22	0.002939	4.822066		Consumo medio con dispositivo in perfetta standby

Tabella 4.3.1 – Simulazione 24 H, SF7

TIME (S)	I (MA)	V (V)	NUMERO RIPETIZIONI	DESCRIZIONE
6.685902	0.00703	4.820293	3	Singolo Impulso * 3 volte
4.678956	14.17648	4.593637	1	Avvio flusso e invio dati
7198.488	0.00703	4.820293	3230	Singolo Impulso per 2h
4.936728	13.44997	4.618062	1	Termine del flusso, salvataggio dati in flash ed invio (dopo 2h e circa 6 minuti in media)
142.384	12.33393	4.625282	31	Ogni 4 minuti I dati venivano processati e salvati
79185.21	0.002939	4.822066		Consumo medio con dispositivo in perfetta standby

Tabella 4.3.1 – Simulazione 24 H, SF12

Per comodità nella rappresentazione è stato deciso di calcolare prima i consumi medi durante l'irrigazione poi quelli relativi al tempo di inattività. La colonna "Time" rappresenta la durata in secondi del ritaglio di campionamento selezionato moltiplicato per il numero di ripetizioni che quel lasso temporale campionato viene riproposto durante tutto l'arco della giornata. La colonna "I" visualizza la corrente media in milliampere transitata durante lo stato specificato. La colonna "V" mostra il valore di tensione medio sull'alimentazione durante lo stato specificato. Infine, la colonna "Numero di Ripetizioni" specifica quante volte è stato considerato lo stato specificato.

Come ben sappiamo la sessione di irrigazione inizia nel momento in cui viene letto il quarto impulso sull'ingresso de il LPTIM, ma considerando il fatto che il 4 impulso corrisponde esattamente con il risveglio del microcontrollore (e di conseguenza appartiene alla fase di inizio irrigazione), questo non è stato considerato nella prima riga della tabella.

Proseguendo, nella seconda linea troviamo i valori medi dell'intervallo di inizio irrigazione ed invio dati. Chiaramente troviamo un aumento esponenziale del valore medio della corrente dovuto al risveglio del microcontrollore e del relativo invio del messaggio LoRa.

Successivamente è stato inserito il campionamento del singolo impulso. Questa volta il numero di ripetizione è però stato impostato a 3600 (secondi in un'ora) * 2 diviso 2.2 (tempo di durata dell'impulso).

Poi vi è la riga relativa alla terminazione della fase di irrigazione, questa prevede l'invio di un messaggio LoRa ed il salvataggio dei dati nella memoria flash.

In penultima posizione nella tabella, vi è il calcolo relativo ai consumi della fase intermedia ripetuta ogni 4 minuti. Di conseguenza questo è un consumo che deve essere aggiunto ogni 4 minuti al consumo degli impulsi. Viene effettuata la somma per semplificare i calcoli, in quanto il valore della corrente nella fase intermedia è talmente maggiore rispetto a quello dell'impulso da far risultare la somma dei due praticamente equivalente al valore della fase intermedia.

$$I_{Impulso} + I_{MiddlePhase} \approx I_{MiddlePhase}$$

Il numero di ripetizioni di questa azione è esattamente 120/4 + 1 (in quanto prima di poter terminare l'irrigazione è necessario non ricevere più impulsi per un intero intervallo di 4 minuti).

Concludendo abbiamo poi il riempimento del restante tempo della giornata utilizzando i valori dello stato di idle del dispositivo. Quello che è sorprendente è che in questo stato la corrente media transitante nel circuito è di 0.002939 mA, ovvero circa 3 uA. Ora per andare a calcolare quanto una serie di 3 Batterie Stilo comuni da 1200 mAh possa durare è necessario sommare tra loro i prodotti della colonna Time con i valori di corrente (colonna I). Una volta effettuata questa somma, calcoleremo la media di corrente transitante all'interno del circuito in una giornata dividendo il risultato ottenuto per il numero di secondi di una giornata 86400 (24h*60m*60s). I valori ottenuti sono stati di 0.007 mA per SF7 e 0.025 mA per SF12.

Questo vorrebbe dire che, se come detto in precedenza, usassimo una batteria da 1200 mAh, il FlowMeter potrebbe teoricamente sopravvivere senza mai sostituire le batterie per 18 anni e 9 mesi nel caso venisse utilizzato SF7 e per 5 anni e 5 mesi nel caso di SF12.

Per ottenere questi risultati è bastato dividere il mAh della batteria per le correnti medie della giornata. Così facendo si ottengono le ore di autonomia. È poi sufficiente dividere per le ore in un anno ottenendo in questo modo l'autonomia in anni del dispositivo.

$$\text{Autonomia in ore} = \text{Carica Batteria (mAh)} / I_{Circuito}(\text{mA})$$

Il risultato ottenuto è ovviamente straordinario in quanto, secondo questa stima, è molto più probabile che le batterie si deteriorino per il trascorrere del tempo che per il loro utilizzo.

Infatti, i fattori che influenzano secondariamente la scarica di una batteria sono in realtà molteplici, come per esempio umidità e temperatura. Proprio per questo fornire una simulazione realistica al 100% sarebbe stato impossibile principalmente per il fatto, che la scarica della batteria dovrebbe essere stimata anche tenendo in considerazione le condizioni ambientali alle quali il sistema viene sottoposto.

5. Conclusioni e Sviluppi Futuri

5.1 Conclusioni

I risultati ottenuti dalle simulazioni riguardanti l'utilizzo del Flow Meter hanno largamente soddisfatto quelle che erano le aspettative ed i requisiti di questo progetto, che andiamo qui a riproporre:

1) Possibilità di conteggiare i litri d'acqua di una intera sessione di irrigazione;
2) Garantire l'impermeabilità del circuito;
3) Trasmettere in tempo reale a lungo raggio e conservazione dei dati acquisiti;
4) Basso consumo energetico dell'impianto.

Le tecnologie chiave che hanno permesso di progettare un prodotto affidabile ed allo stesso tempo preciso sono state, sul fronte delle comunicazioni RF, l'utilizzo di LoRa e la sua implementazione a livello MAC LoRaWAN, mentre, lato processing, la sinergia prodotta dalla combinazione della modalità di STOP del microcontrollore con LPTIM.

In base ai risultati riscontrati dai test, la tecnologia LoRa ha dimostrato di essere efficiente dal punto di vista energetico ed in grado di garantire una ottima copertura a lungo raggio, oltre ad un basso costo a livello componentistico.

Escludendo il costo della conduttura comprensiva di reed switch ed il costo delle batterie, questo progetto ha un prezzo hardware intorno ai 50-70€, ulteriormente riducibile nel caso in cui la board LoRa, di fornitura ST Microelectronics, venisse riprogettata ed integrata nel PCB di acquisizione. A questo prezzo si è riusciti a fornire un prodotto stabile, con un alto livello di affidabilità ed allo stesso tempo longevo sul fronte energetico. Ricordiamo, infatti, che un pacco di tre batterie stilo potrebbe garantire l'alimentazione al Flow Meter per circa 18 anni in condizioni ottimali.

Questo progetto dimostra come anche un semplice contatore sia però complesso da realizzare in modo che possa funzionare in ogni condizione senza mai fallire. Infatti, come disse George Sand,

"*La semplicità è la cosa più difficile da ottenere a questo mondo; è l'estremo limite dell'esperienza e l'ultimo sforzo del genio*".

5.2 Sviluppi futuri

Diverse sono le strade di miglioramento che questo progetto può intraprendere, ma solo tre sono quelle che affronteremo in questa sezione.

Attualmente il Flow Meter effettua una scrittura in memoria flash ogni volta che la sessione di irrigazione termina. A lungo andare questo processo può generare rotture del settore di memoria utilizzato. La tipologia di problema di cui stiamo parlando è denomitata flash memory write endurance e corrisponde al numero di programmazioni/cancellazioni (cicli P/E) che può essere applicato a un blocco di memoria flash prima che il supporto di memorizzazione diventi inaffidabile.

Vi sono diversi algoritmi in grado di andare a risolvere questo problema che cercano, attraverso analisi continue della memoria, di identificare i settori corrotti e ripristinarli copiandoli magari in altre aree di memoria. Uno di questi è [26], che propone una soluzione in cui ogni pagina della memoria flash viene periodicamente letta ed ogni errore viene corretto utilizzando un Error Correction Code semplificato, trascrivendo i dati su una nuova pagina o sovrascrivendolo il dato corrotto, prima che la pagina accumuli più errori di quelli ECC corretti.

Al Flow Meter basterebbe però una soluzione molto più semplice, che gli consentirebbe di scrivere il dato in indirizzi flash sempre diversi ad ogni scrittura. In questo modo il numero di cicli P/E verrebbe distribuito su più settori prolungando la vita della memoria flash utilizzata.

Passando ora all'analisi della scalabilità di questo sistema, tutti i diversi parametri impostati staticamente potrebbero essere resi personalizzabili in base alle esigenze del singolo cliente. Per far fronte a ciò, si è previsto, ma non ancora implementato, la possibilità di ricevere comandi da remoto attraverso messaggi di downlink in grado di variare i valori di suddetti parametri.

Un esempio potrebbe essere il tempo della fase intermedia, attualmente impostato a 4 minuti, oppure il numero di impulsi soglia per entrare in stato di irrigazione etc...

Nulla toglie però che anche operazione più complesse come l'aggiornamento firmware possano essere supportate da questa tipologia di comunicazione. Fornendo la possibilità di personalizzare la maggior parte dei parametri di configurazione della macchina a stati del Flow Meter si potrebbero ridurre al minimo le personalizzazioni del firmware del microcontrollore.

Con un unico codice binario sarebbe dunque possibile soddisfare esigenze diverse derivanti dalle richieste dei clienti semplicemente inviando comandi di personalizzazione da remoto.

Infine, sarebbe necessario andare a ridurre ulteriormente i costi della parte elettronica, attualmente intorno ai 50-70€, includendo i componenti presenti nella board ST B-L072Z-LRWAN1 nel PCB da noi costruito, ovvero il modulo Murata CMWX1ZZABZ-091 e l'antenna per la trasmissione radio. Così facendo l'abbattimento dei costi hardware potrebbe raggiungere anche il 40%, consentendo di rilasciare sul mercato un prodotto competitivo.

La stampa del PCB avverrà in breve tempo, successivamente sarà necessario ricostruire il prototipo in modo da adattare la parte meccanica alle dimensioni del circuito stampato. A questo punto potranno essere effettuati test su campo riconducibili a situazioni reali, in modo da validare il prodotto e renderlo effettivamente commerciabile.

Bibliografia

[1] Bruxelles, R. (2019). L'agricoltura "beve" il 70% dell'acqua mondiale. Mentre 2 miliardi di persone sono a secco. [online] Agrifoodtoday.it. Available at: https://www.agrifoodtoday.it/ambiente-clima/barilla-acqua-consumi.html.

[2] Ufficio Stampa, L. (2019). Forum Acqua: tutela, zero sprechi e riuso. Forum Nazionale di Legambiente sulle filiere della sostenibilità della risorsa idrica. [online] Utilitalia.it. Available at: http://www.utilitalia.it/news/archivio?ecc67ffc-d4a4-40e9-8695-87ce37bd42cf.

[3] SWAMP Project, http://swamp-project.org/about/

[4] Irriframe-Irrinet, http://swamp-project.org/about/

[5] Bellini, G. (2014). UTILIZZO DELLA RISORSA IDRICA A FINI IRRIGUI IN AGRICOLTURA. 6th ed. [ebook] Roma: ISTAT, pp.8-13. Available at: https://www.istat.it/it/files/2014/11/Utilizzo_risorsa_idrica.pdf.

[6] Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J. and Aharon, D. (2015). THE INTERNET OF THINGS: MAPPING THE VALUE BEYOND THE HYPE. [ebook] Mckinsey Global institute, p.4. Available at: https://www.mckinsey.com/~/media/McKinsey/Industries/Technology%20Media%20and%20Telecommunications/High%20Tech/Our%20Insights/The%20Internet%20of%20Things%20The%20value%20of%20digitizing%20the%20physical%20world/Unlocking_the_potential_of_the_Internet_of_Things_Executive_summary.ashx.

[7] T. Ku, W. Park and H. Choi, "IoT energy management platform for microgrid," 2017 IEEE 7th International Conference on Power and Energy Systems (ICPES), Toronto, ON, 2017, pp. 106-110.

[8] C. Raj, C. Jain and W. Arif, "HEMAN: Health monitoring and nous: An IoT based e-health care system for remote telemedicine," 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, 2017, pp. 2115-2119.

[9] S. S. Al-Ghawi, S. A. Hussain, M. A. A. Rahbi and S. Z. Hussain, "Automatic toll e-ticketing system for transportation systems," 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC), Muscat, 2016, pp. 1-5.

[10] Internet Engineering Task Force, RFC-8376, LPWAN - Low-Power Wide Area Network, S. Farrell Ed., Trinity College Dublin, 2018.

[11] LoRa Alliance, https://lora-alliance.org/

[12] Wi-SUN, https://www.wi-sun.org/

[13] SIGFOX, https://www.sigfox.com/

[14] 3GPP, Standardization of NB-IOT, Release 13, https://www.3gpp.org/news-events/1785-nb_iot_complete, 2016

[15] M. Suresh, U. Muthukumar and J. Chandapillai, "A novel smart water-meter based on IoT and smartphone app for city distribution management," 2017 IEEE Region 10 Symposium (TENSYMP), Cochin, 2017, pp. 1-5.

[16] Z. Yan and H. Gang, "Design of Intelligent Water Metering System for Agricultural Water Based on NB-IOT," 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2019, pp. 1665-1669.

[17]L. Cao, J. Tian and Y. Liu, "Remote Real Time Automatic Meter Reading System Based on Wireless Sensor Networks," 2008 3rd International Conference on Innovative Computing Information and Control, Dalian, Liaoning, 2008, pp. 591-591.

[18] Datasheet Murata CMWX1ZZABZ-091, https://wireless.murata.com/pub/RFM/data/type_abz.pdf

[19] Reference Manual STM32L0x1 STMicroelectronics, https://www.st.com/content/ccc/resource/technical/document/reference_manual/21/bd/0f/bd/1c/88/40/f0/DM00108282.pdf/files/DM00108282.pdf/jcr:content/translations/en.DM00108282.pdf

[20] KiCad EDA, https://kicad-pcb.org/

[21] Datasheet STLQ015 STMicroelectronics, https://www.st.com/resource/en/datasheet/stlq015.pdf

[22] Datasheet ADG819 Analog Devices, https://www.analog.com/media/en/technical-documentation/data-sheets/ADG819.pdf

[23] Datasheet TS5A3359 Texas Instruments, https://www.ti.com/lit/ds/symlink/ts5a3359.pdf

[24] Discharge tests of AA Batteries, Alkaline and NiMH, Power Stream, July 2019, https://www.powerstream.com/AA-tests.htm

[25] STM32CubeIDE, https://www.st.com/en/development-tools/stm32cubeide.html

[26] Yu Cai, Gulay Yalcin, Onur Mutlu, Erich F. Haratsch, Adrian Cristal, Osman S. Unsal, Ken Mai, Flash Correct-and-Refresh Retention-Aware: Error Management for Increased Flash Memory Lifetime, Carnegie Mellon University, Barcelona Supercomputing Center, LSI Corporation, https://people.inf.ethz.ch/omutlu/pub/mutlu_iccd12_talk.pdf