Progetto di Industrial informatics

Implementation of a home automation system for controlling lights using IEC 62541.

Trovato Simone Sebastiano

Sommario

[1. INTRODUZIONE 2](#_Toc21173823)

[2. Parte 1: Server OPC e Controllo 2](#_Toc21173824)

[2.1 Controllo 2](#_Toc21173825)

[2.2 Server OPC 3](#_Toc21173826)

[3. Parte 2: Applicazione Android 4](#_Toc21173827)

[3.1 MainActivity 4](#_Toc21173828)

[3.2 SessionActivity 4](#_Toc21173829)

[3.3 BrowsingActivity 4](#_Toc21173830)

[3.4 SubscriptionActivity e SingleSubscriptionActivity 4](#_Toc21173831)

[3.5 MonitoringItemActivity 5](#_Toc21173832)

[3.6 MonitoringActivity 5](#_Toc21173833)

[3.7 Considerazioni 5](#_Toc21173834)

[4. Parte 3: Prototipo 6](#_Toc21173835)

# INTRODUZIONE

Il progetto realizzato può essere diviso in tre parti.

Il primo obiettivo è stato quello di implementare un sistema di controllo, principalmente per le luci di una casa, e di un server OPC-UA che permetta di interagire col sistema. Questa parte è stata realizzata utilizzando un raspberry come server OPC-UA e come controllore principale, il tutto è stato programmato con python.

Secondariamente è stato realizzato un client Android OPC che permettesse di interagire col server OPC. Si è scelto di progettare un client generico che non facesse riferimento a necessità proprie del sistema, in pratica il client può essere usato per interagire con qualsiasi server OPC.

Infine è stato realizzato il modellino di un appartamento, in legno, che permettesse di testare tutte le funzionalità create.

# Parte 1: Server OPC e Controllo

La prima cosa che è stata fatta è stata analizzare il problema, scegliere cosa andava controllato e infine scegliere le librerie e l’hardware da utilizzare.

Si è deciso di concentrare il controllo sulle luci di un appartamento, costruendo anche un sistema di sensori che permetta il calcolo delle persone che entrano ed escono da una stanza; inoltre è stato aggiunto un sensore di temperatura, in previsione di una possibile aggiunta di comando per accendere e spegnere il condizionare (per questo progetto il sensore di temperatura resta comunque un oggetto per la sola lettura).

Per quanto riguarda l’hardware sono stati usati dei led bianchi e gialli per le luci, il sensore di temperatura è un LM35, dei bottoni per spegnere e accendere le luci manualmente, dei diodi led in trasmissione e in ricezione per i contatori.

Il controllore principale è un Rapsberry Pi 3 Model B, a cui va aggiunto un Arduino, collegato tramite seriale, per leggere il sensore di temperatura (il Rapsberry non ha un convertitore analogico digitale integrato) e una board di alimentazione per non superare la soglia di erogazione di corrente del Rapsberry .

Il raspberry si integra perfettamente con python che è il linguaggio scelto, le librerie principali utilizzate sono <https://github.com/FreeOpcUa/python-opcua> per la realizzazione della parte server, la libreria RPi.GPIO per leggere e scrivere con i GPIO del Rapsberry e PySerial per la comunicazione tra Rapsberry e Arduino.

## Controllo

È stato costruito un package python, che oltre al main contiene moduli di configurazione e i moduli per i due processi controllo e server OPC.

Nel modulo di configurazione “gpio.py” vengono costruiti dei dizionari appositi, uno per i led, uno per i bottoni, uno per i diodi led Rx grazie ai quali il nome di una stanza viene associata a un gpio specifico; in questo file di configurazione viene inizializzata anche la seriale.

Il modulo “controllo.py” definisce il controllo vero e proprio. Fondamentalmente è una classe che contiene un metodo ‘run’ per avviare il processo di controllo e delle callback per gestire degli eventi.

Le callback servono per gestire i bottoni, le luci infatti si possono accendere e spegnere anche manualmente, e per gestire i contatori di persone.

Per contare le persone è stato costruito una sorta di encoder utilizzando 2 diodi led Tx e 2 diodi led Rx, le coppie sono messe negli stipiti delle porte, un coppia Rx-Tx prima della stanza e una nella stanza. Se tra tx e rx non ci sono ostacoli sul pin del raspberry si legge 0, nel momento che qualcosa si interpone tra il tx e rx sul pin del raspberry viene letto 1, le callback quindi stanno in ascolto di un fronte di salita, se i fronti di salita si attivano nell’ordine esterno-interno vuol dire che qualcuno è entrato quindi il contatore si incrementa, e se era 0 si accende la luce, se si attiva nell’ordine interno-esterno il contatore si decrementa e se diventa zero la luce si spegne (il contatore non può scendere sotto zero, attualmente viene mostrato un messaggio d’errore sulla console).

Per gestire la lettura della temperatura, viene quindi eseguito un ciclo che ogni secondo controlla se ci sono dati sulla seriale e in caso positivo legge e setta la variabile globale che fa riferimento alla temperatura (è condivisa col processo del Server OPC grazie al modulo “Manager” del package multiprocessing di python).

## Server OPC

Per la realizzazione del server opc viene utilizzato il package reperibile su GitHub a <https://github.com/FreeOpcUa/python-opcua>. Anche per il server è stata realizzata una classe che contiene un metodo ‘run’ richiamato per creare il processo.

La classe “ServerOPC” contiene un metodo per inizializzare il server, anche se la libreria permette di dichiarare il namespace e i nodi tramite xml è stato deciso di lasciare gli elementi tutti nel codice. Per prima cosa viene settato l’endpoint, vengono caricati i certificati e settate le security policy; un bug la cui origine non è stata trovata impedisce attualmente al client android di connettersi al server usando una security policy.

Viene quindi settato il namaspace e vengono aggiunti i nodi per leggere lo stato delle luci e del sensore di temperatura e i contatori di persone, quindi, i nodi per l’esecuzione dei metodi, viene definito il parametro “inarg” e i metodi sono due uno per accendere e uno per spegnere le luci.

Il metodo ‘run’ è un ciclo che ogni secondo si occupa di settare il valore di temperatura e dei contatori, leggendoli dal dizionario condiviso col processo di controllo, e inoltre legge il valore dei GPIO delle luci.

# Parte 2: Applicazione Android

Per la realizzazione dell’app Android non sono stati utilizzati framework particolari, ma ci si è basati direttamente sullo Stack opc-ua java ufficiale.

In un primo momento sono stati individuati gli elementi principali di una connessione OPC-UA, un client per comunicare con un server ha bisogno di creare una Sessione, quindi, se vuole può creare una Subscription e monitore dei valori. La progettazione del client Android è partita proprio con la creazione di questi 3 elementi.

In particolare, è la Sessione a mantenere una lista delle Subscription attive, e quest’ultime hanno una lista dei loro Monitored items; i Monitored items invece hanno una lista che costituisce il buffer nel quale vengono conservate le Notification quando viene notificato il cambiamento del valore monitorato.

È stata quindi realizzata una classe ManagerOPC per gestire diverse Sessioni.

A questo punto sono state realizzate a uno a uno le activity e i thread per eseguire le operazioni opc.

## MainActivity

La Main activity è l’activity principale, consiste in un campo di input in cui inserire l’indirizzo IP del server e un bottone per avviare il discovery. Premuto il bottone viene avviato un thread che si connette al server esegue l’operazione di discovery, quindi, gli endpoint trovati vengono mostrati e all’utente non resta che selezionare quello desiderato.

## SessionActivity

Selezionato un endpoint e confermata la connessione si passa alla SessionActivity.

Da questa pagina si può creare una Subscription e avviare il browsing, con la navbar posta in basso ci si può spostare nell’activity Subscription che permette di vedere tutte le sottoscrizioni o in quello di monitoring che permette di avere in un un’unica schermata tutti i Monitored item.

Il pulsate crea Subscription mostra un Dialog che permette di inserire i parametri della Subscription.

## BrowsingActivity

La browsing activity sfrutta il sistema di fragment di Android per mostrare all’utente il contenuto del server e dei suoi nodi.

Navigando fino a un nodo, ad esempio Root->Object->Sensori->Temperatura, se è un nodo di tipo variable sarà possibile eseguire la Read cliccato su quest’ultimo, si aprirà infatti un un dialog che mostra il valore letto e che permette anche di eseguire una Write.

Write che per questa applicazione non ha nessuno valore per Temperatura e luci, infatti, questi valori vengono letti costantemente dai sensori, andrebbe evitato anche di modificare i counter per non introdurre bug; la funzione di Write è stata inserita solo per mantenere quando più possibile l’app adatta a interagire con qualsiasi server OPC.

Una menzione a parte meritano i **Metodi**, attualmente infatti sono raggiungibili solo tramite il browsing e non sono stati aggiunti bottoni diretti, questo sempre per mantenere l’app generalizzata.

Raggiuto il metodo ed entrando sarà possibile cliccare sugli InputArguments, in questo caso inserito l’input e premuto call verrà applicato il metodo. Per il sistema realizzato i metodi sono due uno per accendere e uno per spegnere le luci, l’input può essere uno tra ‘salotto’, ‘camera’, ‘bagno’, ‘ingresso’ in base alla stanza con cui vogliamo interagire.

## SubscriptionActivity e SingleSubscriptionActivity

Utilizzando la navbar il basso e cliccando Subscription verrà mostrata la SubscriptionActivity che mostra tutte le Subscription con le loro informazioni per la sessione corrente.

Cliccando su una di queste Subscription si accede alla SingleSubscriptionActivity relativa, qui vi sarà la possibilità di creare Monitored item per questa Subscription o interagire con quelli già creati.

## MonitoringItemActivity

Dalla session activity cliccando su uno dei Monitored item si arriva a questa activity che mostra le ultime 5 letture in tempo reale.

## MonitoringActivity

Tornati Indietro fino a ritrovare la navbar si può accedere cliccando su sensori alla MonitoringActivity, questa mostra all’utente tutte le Subscription con l’ultima lettura di ogni Monitored item registrato e se il thread di aggiornamento è attivo aggiorna le letture in tempo reale.

## Considerazioni

L’app non è esente da bug, sia grafici legati alle view sia funzionali. Uno dei bug è che la sessione e quindi il monitoring non funziona in background, probabilmente usare framework come Prosys avrebbe semplificato la realizzazione dell’app e avrebbe ridotto i bug.

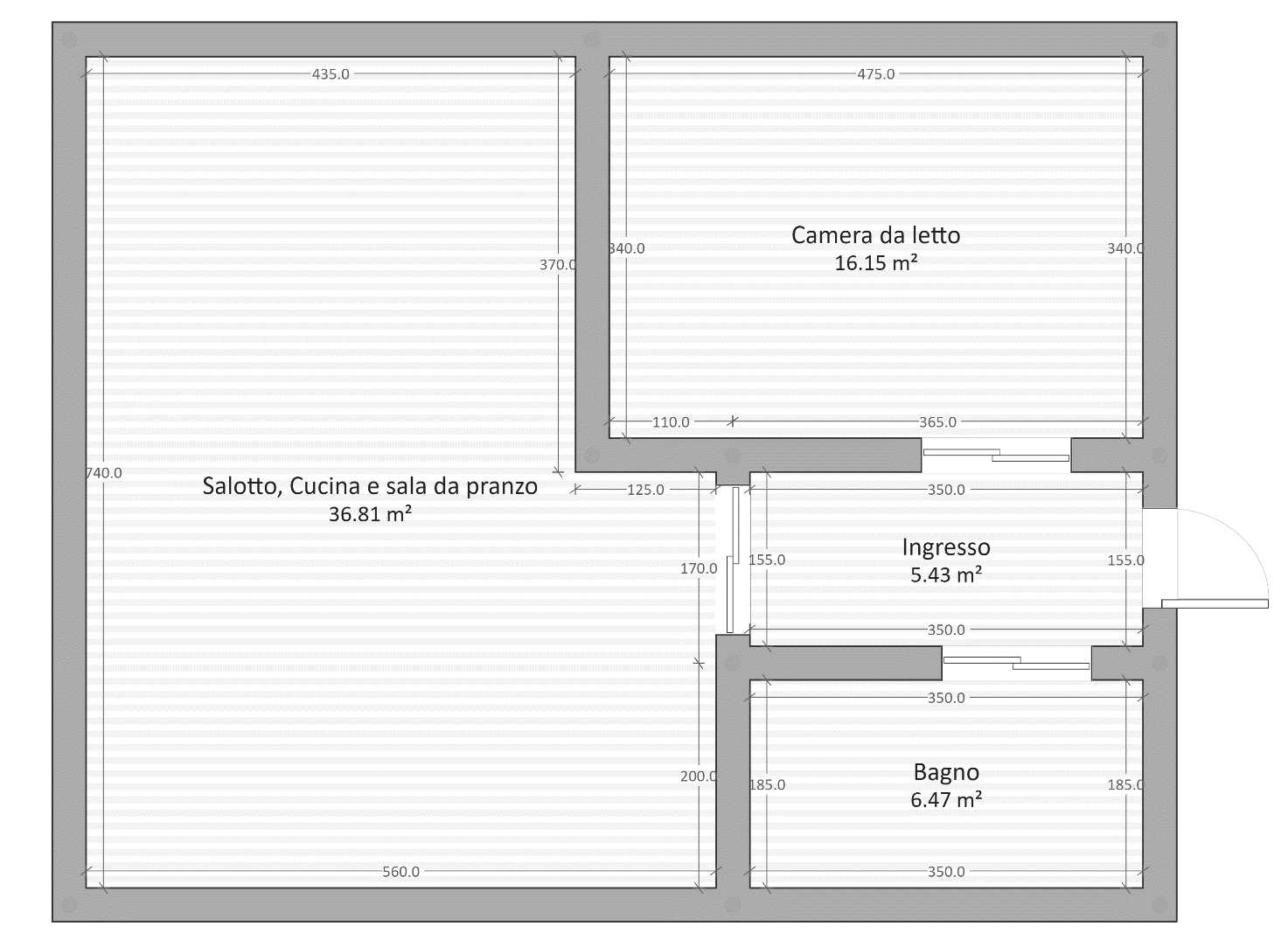
Un altro bug da segnalare è l’impossibilità di creare un canale sicuro tra client e server realizzato, nonostante i diversi test non si è trovata una soluzione. Con altri client è possibile aprire un secure channel verso il server realizzato, e con il client è possibile collegarsi ad altri server tramite secure channel; l’incompatibilità si presenta solo tra client e server realizzati (video su drive).

La scelta di creare un’app generalizzata adatta a qualsiasi server rende non immediato l’utilizzo con il sistema domotico realizzato, ad esempio, per accendere o spegnere le luci da app non esiste un bottone ma bisogna navigare col browsing fino al metodo che si vuole eseguire o per monitorare il sistema si devono aggiungere singolarmente tutti gli elementi dopo avere letto il loro namespace e il loro index. Per semplificare quest’ultimo passaggio ecco una lista di coppie namespace-node index per tutti gli elementi monitorabili:

* Sensore di temperatura 2-2
* Contatore persone bagno 2-3
* Contatore persone salotto 2-4
* Luce salotto 2-6
* Luce camera 2-7
* Luce bagno 2-8
* Luce Ingresso 2-9

# Parte 3: Prototipo

Per la realizzazione del prototipo si è deciso di realizzare una piantina realistica di un appartamento di 80 m2 in scala 1:20 utilizzando del compensato.



Dato che il progetto richiedeva l’utilizzo di una gran quantità di cavi il tutto è stato realizzato utilizzando due basi una per il modello dell’appartamento uno per posizionare l’hardware. Nella base inferiore sono posizionati quindi il Raspberry, l’Arduino, e le breadboards che sono state utilizzate per semplificare e in parte ordinare i cavi; nella base superiore è stato realizzato il modellino e i cavi sono passati attraverso dei buchi che collegano la base con la parte inferiore.

Sono inseriti:

* Salotto: 2 led, 1 bottone, 1 sensore di temperatura, un contatore di persone all’ingresso della stanza.
* Bagno: 1 led, 1 bottone, un contatore di persone all’ingresso della stanza.
* Ingresso: 1 led, 1 bottone.
* Camera: 1 led, 1 bottone.