**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

## Architetture di controllo per sistemi embedded con sistema operativo

Come vengono i programmi con un sistema operativo che ci consente di utilizzare dei thread, come riscriveremo tutto quello studiato fino ad oggi ?

Superloop architettura di controllo che ha un loop che viene eseguito di continuo

Tutto ciò che veniva modellato come task nelle architetture senza sistema operativo, viene modellato come thread, quindi a questo livello diventano delle entità che devono eseguire un compito e possono essere implementate come una macchina sincrona o asincrona o super loop o event loop (architettura pervasiva e generica)

Agente è l'entità che esegue il task

Nell'esempio del prof ho un semplice Blinking Led dove emula un' accensione e spegnimento di un led nello nostro sistema, il professore utilizza un super loop controller che incapsula i metodi principali per emulare quelli utilizzati da Arduino, rispettivamente:

- Setup()  Dove abbiamo le dichiarazioni delle variabili e costruttore.
- Loop() Dove abbiamo la logica del programma che vogliamo eseguire

Una volta inseriti questi due metodi astratti e protetti, li va a richiamare dentro i programmi veri e propri del blinker o del button led.
Per il momento nella parte di setup richiamata non inserisce nulla, e nella parte che si chiamava run sostituisce alcune cose tra cui la firma del metodo richiamandolo loop come dichiarato nel super loop controller.

Il prof ha creato un package con tutti i dispositivi utilizzati per gli esempi, e dentro ogni interfaccia ha inserito delle funzioni base per ognuna, esempio nel button ha inserito la funzione booleana isPressed().

Oltre alle interfacce ha inserito anche delle classi che emulano l'accensione o lo spegnimento di un dispositivo come per esempio il led.

Ci sono anche delle classi che non sono emulate, ma sono delle classi che utilizzano delle librerie che possono controllare e pilotano tutti i dispositivi e non sono emulazioni in standard output (seriale eclipse o visual studio code).

Come implemento una macchina a stati finiti con queste nuove cose aggiunte come i thread ?

A differenza delle funzioni utilizzate in Arduino per simulare un attesa come per esempio delay che sono funzioni che generano del busy waiting andando a utilizzare la CPU, in sistemi con thread andiamo a usare delle funzioni bloccanti come waitfor che rilasciano la risorsa in oggetto concedendone l'utilizzo ad altre richieste di altri thread.

*Nicholas Ciarafoni*

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

In una singola applicazione io  posso avere più thread e utilizzare la CPU per mettere in attesa una funzione di un thread sarebbe uno spreco, quindi posso decidere di fermare per il tempo richiesto dalla funzione del thread 1 l'uso della CPU, e permettere al thread 2 di utilizzarla nel lasso di tempo in cui thread 1 con priorità ha finito il tempo di attesa o delay, senza così creare del busy waiting, e ottimizzare di fatto l'utilizzo delle risorse importanti come per esempio la CPU di questo caso.

**Come è fatta una macchia stati finiti sincrona ?**

SyncSFMController, è un semplice modo di implementare dei task in un modo che richiami le macchine a stati finiti sincrone dove abbiamo un discorso di periodo, ogni tot di tempo il thread procede richiamando un tick.

Comportamento periodico che a seconda del tempo deciso in un delta, abbiamo un richiamo al tick.

Esempio del button led , ho due stati per la mia macchina a stati finiti, ed a ogni colpo di clock della mia macchina a stati finiti, vado a controllare in che stato sono, se lo stato è acceso lo spengo, se lo stato è spento lo accendo.

Sostanzialmente non devo farmi lo scheduler, ma richiamo il tick con un periodo che dipende dal periodo che mi ha specificato l'utente.
Nell'esempio specifico vado a richiamare tick,con un tempo di attesa che tiene conto del fatto che la computazione dell'ultimo tick ha richiesto del tempo, quindi nel periodo si fa una sleep di un certo numero di millisecondi dipende dal tempo che tick vecchio ha impiegato per essere eseguito.

Overrun se io vedo che è trascorso più tempo rispetto al periodo che ho impostato, non solo che aspetto 0 millisecondi, anzi sono in ritardo, quindi sono in una situazione d' eccezione.

**Come funziona un architettura EventLoop ?**
Viene contenuto in un unico flusso di controllo, che non fa altro di rimanere in attesa nella sua coda di eventi, e interagisce con il mondo esterno solamente con gli eventi, e può percepire solamente gli stessi.
Aspetta di vedere il prossimo evento, si blocca con blocking queue, struttura a coda, concorrentemente, che blocca una risorsa.
Non appena si sblocca fa una chiamata a funzione.
Non è ancora un sistema asincrono.

**Observable**
classe base nel pattern observer, che è una sorgente di eventi, dove mi registro come osservatore di eventi e posso notificare un evento, per ogni osservatore io notifico ogni evento in esame.

*Nicholas Ciarafoni*

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

Tutte le volte che viene premuto un pulsante, dal frame, viene cambiato lo stato dalla GUI che simula il pulsante, e poi viene notificato l'evento del button.
La notifica cosa fa ? Non fa altro che inserire questo evento nella coda degli eventi, poi chiama proccess event, per capire cosa deve fare con quel'evento specifico.

Quando ho un sistema complesso posso dividerlo in vari thread ognuno dei quali ha un event loop a parte e ognuno di questi può comunicare con altri task.
Mandando un messaggio o accodando dei messaggi.
Paradigma ad attori, sistemi che lavorano e scambiano dei messaggi e reagiscono a questi messaggi ricevuti.

Quando si lavora con event loop, negli handler che reagiscono come eventi non mi posso bloccare perché se mi blocco li, bloccherei il flusso di controllo che reagisce agli eventi, non posso bloccarmi e tutto è asincrono.

*Nicholas Ciarafoni*

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

## Internet of Things

Viene attribuito, alla fine del 1999, in relazione ai progetti alle tecnologie delle RFID del MIT, il loro obiettivo era di introdurre delle tecnologie che permettono di digitalizzare il mondo fisico, automatizzare ad esempio un sistema di raccolta dati in tempo reale, date da sensori impostati sulla vita reale.
RFID sono etichette elettroniche che possono essere scritte in remoto, senza la necessità di avere una batteria, quindi lavorando in passivo, ha sostituito il sistema a barre.
Esempio, il cartellino da timbrare o badge elettronico.

Questi oggetti vengono inseriti in rete e vengono automatizzati dei processi che evitano la comunicazione manuale, senza l'intromissione dell'uomo.
Oggi tutti noi possiamo comprare questi dispositivi, moduli, oggetti fisici, tramite i vari negozi online, e tramite la rete possiamo gestire delle politiche di controllo da remoto, vedi le case in domotica.

Il modo di rivedere il concetto di prodotto e di servizio, questo cambiamento viene chiamato industria 4.0, una specie di nuova rivoluzione industriale.
Il numero dei dispositivi che sono in rete e interagiscono in modo semiautonomo sono molti di più del numero delle persone presenti sulla terra.
Processo di trasformazione digitale ad ampio raggio.

Things:
- Sistemi embedded
- Sensori
- Attuatori

Come comunicano ?
- Protocolli di comunicazione basati su IP, più specifici
  - **MQTT** protocollo di riferimento TCP.
  - XMPP.
  - UPnP.
  - **CoAP** protocollo di riferimento UDP.
    Minimizzano l'overead per comunicare.

Oggetto fisico + Controllore/attuatore/sensore + Parte che collega alla rete.

Non tutti i dispositivi serve che abbiano un collegamento diretto alla rete, ma basta un modulo dei X moduli presenti in questo sistema a renderlo un sistema IOT.

Comunicando con un dispositivo che ha la funzione di gateway, tramite onde Wi-Fi a basso consumo o Bluetooth, possiamo impostare tutto un sistema che internamente comunica in

modo diretto tra i vari moduli e ogni modulo indirettamente comunica con il gateway e l'esterno del sistema.

Ruolo degli smartphone?
Sono anche loro dei sistemi IOT, dato che al loro interno contengono molti sensori, tra cui luminosità distanza cellulare orecchio, Face ID, giroscopio, GPS, etc.

## RFID

Sono stati un punto di partenza importante, però RFID devono essere inseriti all'interno di cose per renderle molto più smart e intelligenti.
nei dispositivi passivi, abbiamo alcune caratteristiche:

- Distanza 2 metri.
- Capacità di memorizzazione all'interno dei kilobyte.
- Evoluzione in NFC, Near-Field Communication.
    - Distanza minore dei 10 centimetri.
    - Velocità di trasmissione pari a 424KBit/s.
    - Interazione tra 2 lettori

Evoluzione di sistemi IOT:

- Compro una caldaia.
- La caldaia diventa programmabile, e mi fornisce la possibilità di cambiare determinate funzionalità tra calore, tramite Bluetooth e telefono, all'interno del progetto ho un computer che mi consente di apportare modifiche, etc.
- Aggiunta di un servizio di rete, che da remoto mi consente di accedere e modificare o monitorare il prodotto, esempio se il prodotto è accessibile in rete, da remoto posso monitorare la caldaia, e controllando i dati in automatico, ho una garanzia a vita, dato che riesco a dirti in anticipo se la tua caldaia può essere in uno stato o potrebbe vedendo i dati incorrere in problemi o guasti, che riesco ad anticipare e si parla di predictive maintenance. Dal punto di vista business, abbiamo un grande servizio e più introiti per le aziende che danno questi servizi.

Evoluzione del quinto stadio:

- Il passo precedente, alla domotica, posso aggiungere un livello molto più ampio, avendo in esempio una casa completamente in domotica, situata in un quartiere di una determinata città posso aumentando il livello di dettaglio, i miei dispositivi possono interagire e fruire un sistema di sicurezza, o possono interagire tra più di una città, quindi questo è un livello di sistema di sistemi, livello di governance, interoperabilità, di scalabilità con tanti dispositivi collegati ad una singola rete.

Industria 4.0

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

IOT è la chiave per questa nuova rivoluzione industriale, tramite le nostre things, abbiamo la capacità di raccogliere i dati continuamente mediante una sensoristica opportuna.

**Enterprise IOT**
Termine generale, con cui ci facciamo un'idea di come IOT viene utilizzato a livello interprise per contrapporlo a quello consumer.
**Industrial IOT**
Fa riferimento all'enteprise IoT utilizzato al mondo manifatturiero dentro le fabbriche, c'è una necessità di un salto che porti da un'industria distante dal livello digitale IT, ad un industria più vicina che integri a pieno questo mondo, industrial IOT le racchiude tutte, facendovi riferimento.

Stack di riferimento **Enterprise IOT**
***Precision [qualcosa]*** dato che le decisioni vengono prese a seconda dei dati che ho a disposizione e delle predizioni.

1. Do → Azione per la quale avendo tutti questi dati agisco sul sistema.
2. Learn → Tecniche con le quali posso fare delle predizioni, machine learning.
3. Collect → Dove metto i miei dati ? Basi di dati.
4. Connect → Rete.
5. Things → Sensori.

**Architettura di un sistema IOT**

Things viene considerato come livello fisico, per poi salire ai livelli di comunicazione di rete, e arrivando al livello di applicazione, ogni livello si suddivide in ulteriori livelli con un gradi di dettaglio maggiore.

**Sistemi scada**
Sistemi con cui si controllano i dispositivi da remoto, sistemi di controllo industriale.
Controllano da remoto dei PLC, cioè dei Programmable logic Controller, che sono una rete di sensori a cui p collegato uno o più PLC, supervisionano la raccolta di informazioni fornendo istruzioni sia alle RTU che ai sensori, è il PLC a indicare alla rete sensoriale quali sono gli intervalli di tempo nei quali effettuare le misurazioni e controllare i valori dei macchinari.
Sono programmabili in un linguaggio particolare e dedicato della famiglia dei IEC 61131-3:

- SFC
- LD

*Nicholas Ciarafoni*

- etc…

Questi sistemi scada, mettono a disposizione tutto il necessario alle persone per il controllo, quindi contengono un reparto d'interfaccia che permette l'operatività all'utente di lavorare e interrogare il sistema.

**OPC** connette il mondo dell'informatica a quello dell'automazione
è l'acronimo per le piattaforme di comunicazione aperte, si parla del livello software che mette in grado di interagire la sorgente dati industriale attraverso i propri protocolli per l'interrogazione e altre funzioni, tramite l'uso di interfacce di alto livello e sistemi esterni.

**EDGE**
Permette di portate fuori istruzioni o dati, fuori dal confine della industria stessa.

**CLOUD e IOT**
Vuol dire la possibilità di sfruttare tramite la rete x cose, cioè qualsiasi cosa si può sfruttare tramite la rete: possibilità di memorizzazione dei dati senza la necessità di avere in casa un cluster di server, uso il cloud, oppure, utilizzo il cloud per acquistare la potenza computazionale richiesta per un determinato progetto o servizio, questa potenza la mettono a disposizione in vendita x as a service.
Sopra ad un livello fisico vengono create delle macchine virtuali che sfruttano la virtualizzazione, questi 100 computer presi in affitto, vengono inseriti in una macchina fisica che tramite il livello di virtualizzazione ti consente di utilizzarne 100.
Computazione, rete, memorizzazione → Sono i principali servizi messi a disposizione.

Il vantaggio di questi tipi di servizi è quello di pagare solamente per l'effettivo utilizzo, molto utile per abbattere i costi iniziali di avviamento di una nuova azienda.

**IAAS** Gestione autonoma, ognuno si deve gestire la propria macchina
**PAAS**
Qui il lavoro è meno oneroso, e molte delle funzioni che avrei dovuto gestire in modo autonomo non vengono visualizzate e sono eseguite in modo automatico tramite l'uso delle API per interfacciarsi al sistema, senza dover mettere mano ai livelli inferiori, più dettagliati.

**SAAS**
Tutte le applicazioni cloud che un utente usa senza la necessità d'installazione, lo utilizzo as a service come ad esempio google documenti.Grazie alla rete e i suoi protocolli riesco ad utilizzarle in modo fluido.

Per smart city e IOT il prof parla di un sistema operativo messo in un livello a se, come middleware, niente a che vedere con linux, mac, windows, che consente la gestione di una città a seconda dei dati che può elaborare, per garantire ad esempio al sicurezza del quartiere.
Tutto questo con un occhio al futuro, di guida autonoma, potrebbe essere molto interessante, anche per esempio uno smart hospital, che fornisce in modo asincrono molti

dati e istruzioni su ogni paziente, sulle operazioni, su come raggiungere un determinato ambulatorio o la capienza della struttura come sotto il regime pandemico del covid-19, etc…

**Sicurezza & Privacy**
I dati che vengono raccolti, di chi sono? come vengono utilizzati?
Discorso molto ampio, articolato e importante!
Sono aspetti critici, e nel corso si sicurezza e reti della magistrale vengono approfonditi in modo maggiore, avendo un grande livello d'importanza.
Se io ho comprato un dispositivo, che raccoglie i dati per cui viene utilizzato o in modo autonomo, questi dati che raccoglie chi ha la proprietà di questi dati ?

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

## Fine modulo 4.1

Arriviamo ad IOT a valle di tutto quello che si è visto fino ad oggi nelle lezioni, introducendo cosa s'intende per IoT, con due macro sviluppi di IOT lato consumer e l'altro enterprise con industria 4.0, ambiente manifatturiero e fabbriche.

**Livello Collect**
Sfrutto il cloud per collezionare i dati in rete, attraverso i protocolli web e protocolli pensati al lato IoT, portandoli fuori dalla fabbrica, o dal contesto in cui sono i dispositivi PLC etc, portandoli sul cloud posso lavorare sui dati ed eseguire tutte quelle elaborazioni utili al miglioramento, nell'ottica per esempio di manutenzione perpetua e preventiva.

**Interoperabilità**
Esistono molti protocolli a basso livello,che consentono la comunicazione tra dispositivi diversi che normalmente non riescono a comunicare tra di loro.
**Web of things, o WoT**
Architettura REST, aperta, scalabile e dinamica attraverso le tecnologie web, l'idea è che ogni things che vuole comunicare con le altre deve disporre di una API web, e sotto la thing potrebbe essere stata costruita con tecnologie diverse, che potrebbero includere già altre tipologie di comunicazione wireless.
Ogni dispositivo si espone alla connessione e comunicazione verso altri dispositivi tramite un livello d'interfacciamento basato sui protocolli web come HTTP.
Tutti gli utenti esterni possono così utilizzare il dispositivo senza conoscere cosa c'è sotto.

Livelli WOT:

1. Networked things → Non un livello reale indica i dispositivi. QR/Arduino/Wi-Fi/etc
2. Access → Layer 1 HTML/JSON/MQTT/HTTP/URI/URL/REST-API/WebSocket
3. Find → Layer 2 JSON-LD/Link HEADER/mDNS/SemanticWeb/RDFa/WebThingMod
4. Share → Layer 3 SocialNetworks/ApiTokens/TLS/PKI/DTLs/Authentication/Encryption
5. Compose → Layer 4 Systems/IFTTT/automated/WebApplication/Integration/UIGeneration

**Mobile Computing**
Possiamo vedere il nostro smartphone come un sistema embedded o una thing che al suo interno è composta da più componenti che comunicando verso le altre riescono a fornire un servizio a un consumatore, quindi riusciamo a sviluppare dei sistemi software in esecuzione su dispositivi mobile come telefoni e tablet, abbiamo delle cose particolari a cui fare attenzione come l'interazione con l'utente del dispositivo, l'utilizzo della batteria che non deve essere troppo altro per non incorrere in problemi d'utilizzo, connessione alla rete intermittente in posti non coperti per intero dalle celle delle connessioni a banda 4G e 5G.

Dispositivi raggiungibili a corto raggio, che comunicano con il Bluetooth, come SmartWatch etc.

*Nicholas Ciarafoni*

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

**Comunicazione ESP32**
Clonato la cartella seiot2021 su una cartella, per poi ricercare il test-esp-basic e modificarlo inserendo dei delay a seconda del valore preso dal potenziometro, con degli if a cascata per un primo momento, avrei potuto utilizzare uno switch per il valore e tutti i case di seguito.

Nell'impianto il prof evita di mettere un resistore essendo che esp genera un'intensità pari a 3.3v che non crea dei problemi con il nostro sistema, in caso contrario avrei bruciato il led.

**Android studio** *10/12/2020*

Kotlin linguaggio di programmazione che viene tradotto in java.
Scarica tutto il necessario per programmare un applicazione test, usando come retro compatibilità la versione marshmellow, e andando in alto a destra sulla SDK platform devi scaricarti la versione inerente a quella scelta come compatibilità minima per appunto marshmellow, andando a vedere l'estensione dei pacchetti su show package details, devi includere nell'istallazione Android SDK, sources for Android, e un immagine di sistema come google api intelx86 Atom system image.
andando su SDK tools invece vai ad aggiungere google play service, google usb driver.

Definire un'interfaccia utente, utilizzo la mia main activity.
La convenzione in Android è activity_main, activity_ etc.

Sulla sinistra si trova una cartella chiamata **res**, andando in **layout** abbiamo un file XML che ci consente la programmazione del design della nostra applicazione.

**Domanda esame:**
*Come è stata organizzata l'activity ? Che scelte avete fatto ? Reattività? Organizzazione dell'applicazione e architettura utilizzata, conta più il lato progettuale che quello di design.*

**Stringhe dinamiche**
Molte delle stringhe della mia applicazione potrebbero dare errori, in un ottica globale, è utile utilizzare delle macro, che a seconda della località cambiano il proprio valore, infatti dentro il progetto troviamo una cartella chiamata res, al suo interno abbiamo values, e in strings, possiamo aprire l'editor e nel mappamondo aggiungere le lingue che vogliamo utilizzare per la traduzione di tutte le parti interne alla mia applicazione in modo dinamico.

**Shared Preferences**
potresti avere la necessità di volere memorizzare alcune configurazioni, come l'id per connettervi al dispositivo esterno, tipo Arduino, senza dovere ogni volta specificarlo.

*Nicholas Ciarafoni*

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

**Task asincrono**
compito che vogliamo mettere in esecuzione. tipo il download di un'immagine, un task è un qualcosa che termina anche se ha una sua durata in cui continua ad esistere.
consente di mettere in esecuzione doinbackground e onpostexecute.

Una viene eseguita in un thread secondario, l'altra viene eseguita nel thread main.
Metodo meno generico, ma molto specifico per mettere in pratica tutti quei task che hanno una durata abbastanza importante.
Async task esegue in background il proprio lavoro e poi il risultato viene dato nella onPostExecute al termine, passando da un thread secondario a quello principale.

*al click del bottone, creo il mio task asincrono, poi chiamo l'esecuzione che scarica un'immagine.*

Tutti i task asincroni devono essere creato nel thread principale,su ogni istanza di task asincrono si può chiamare una sola volta il metodo execute().
Tutti i task asincroni eseguiti all'interno della stessa applicazione fanno riferimento allo stesso worker thread, una doinbackground è bloccante, allora per X tempo fino al termine dell'esecuzione, non possiamo lanciare un'altra execute, finchè non abbiamo terminato il task precedente.
**Utilizzando invece executeOnExecutor() al posto di execute() possiamo gestire compiti in parallelo.**
Utilizzando il componente **service**, quando ho un compito che non ha un tempo definito. (non utile al progetto d'esame)

Solo il thread dedicato al layout può lavorare su questo settore, quindi potrebbero esserci situazioni di corse critiche nella gestione di multithread.

## Tecnologie e protocolli per la comunicazione sistemi embedded

*Modulo 4.2*

Parliamo di sistemi wired e wireless, stiamo lavorando nello stack dei protocolli internet, il livello di rete.
Possiamo collegare il dispositivo in due modi in modo diretto, che in modo indiretto. indirettamente utilizza un dispositivo intermedio che si colloca in un ambiente con più dispositivi IoT, e viene utilizzato un protocollo a corto raggio che consuma poco, che consente di collegarsi direttamente con IoT.
Possiamo considerare di utilizzare due moduli che sfruttino la rete cellulare per la comunicazione di messaggi tra dispositivi.
Più è ampio il raggio di azione della rete, più consumo ho sulla mia batteria a livello energetico, sulle reti 4/5G.
Da evitare il più possibile o da utilizzare nel minor numero di casi, preferendo altre tecnologie di comunicazione al posto di questa.

*Nicholas Ciarafoni*

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

L'architettura ISO/OSI, garantisce l'interoperabilità tra dispositivi grazie a questi strati o livelli.

Spiegazione del **modem**, componente che converte la sequenza di bit in segnali elettrici, che attraverso un canale di telecomunicazione, arrivando alla destinazione tradotti di nuovo in sequenze di bit.

Short range:
- Bluetooth → consuma poco decine di metri di raggio d'azione.

Medium range
- Wi-Fi

Long range
- 4/5G → grandissimo raggio d'azione, e permette di avere delle buone performance, bassa latenza e riduzione dei consumi energetici.

NFC
- raggio cortissimo di qualche centimetro o meno del cm.

17/12/2020
https://web.microsoftstream.com/video/73d5f6dc-7066-49ac-9518-feaa0dd60fd2?list=user&userId=8bf5e0a6-9e33-4a87-9fa5-b9332620bedc

**Modulo 4.2**

Quando il Bluetooth lampeggia rosso, significa che non è ancora connesso a niente, quando diventa invece fisso, allora è presente una connessione ad un altro dispositivo.

Dopo svariate prove sono riuscito a far funzionare il modulo Bluetooth che abbiamo in dotazione, il modulo H05 che lavora sia in master che slave mode.
per prima cosa copiando lo schema del professore sono andato ad eseguire alcune modifiche, per esempio la coppia di pin GPIO che utilizzo e TX=6 e RX=7, collegate rispettivamente all'inverso per consentire la comunicazione seriale.
Successivamente ho cambiato la velocità di comunicazione del canale seriale da 9600baud come settata dal prof, a 38400 baud che è la velocità di default del modello H05 che abbiamo in dotazione.
Nel pannello della porta seriale dobbiamo selezionare come riga non il NewLine ma "Both NL & CR" e la velocità a 38400 baud.
Ora i comandi dovrebbero funzionare, ma seguendo cosa il mio seriale mi mostra a schermo, trovo che mi restituisce inserendo il comando AT un errore(0), stessa cosa con tutti gli altri comandi che ho

La funzione Msg del file bt_pinpong funziona con le stringhe senza l'utilizzo del terminatore newline, quindi possiamo omettere l'inserimento.
Ulteriori modifiche possono venire fuori andando a modificare il service ed inserendo

*Nicholas Ciarafoni*

all'interno la possibilità di riconoscere sequenze numeriche al posto di stringhe in chiaro, che sicuramente possono andare ad occupare minor memoria, ricordando che per ogni carattere della parola la relativa  codifica a 8 bit ha un peso maggiore rispetto a quella creata con un numero rappresentato da 8 bit.

Per la comunicazione tra applicazione Android e modulo H05, occorre apportare delle modifiche importanti all'interno dell'applicativo in Android studio:
Per prima cosa andiamo a controllare che stiamo utilizzando la funzione adeguata con il corretto valore di UUID

```
final UUID uuid = BluetoothUtils.getEmbeddedDeviceDefaultUuid();
```

Questa funzione è messa a disposizione per i dispositivi embedded di default.

```
public static final String BT_DEVICE_ACTING_AS_SERVER_NAME =
"ciccio";
```

Nella classe C dove abbiamo le nostre costanti, andiamo a modificare la costante che serve per il collegamento al Bluetooth, occorre inserire il nome del dispositivo che utilizziamo come "server" dato che risponde ad un messaggio che viene inviato da client, per esempio il nostro dispositivo Android.

Dato che con il comando di configurazione AT+NAME="ciccio", ho assegnato come nome ciccio al mio dispositivo, questa costante va impostata allo stesso modo con la stessa stringa.

Questo serve per la comunicazione in modo fisico, e reale tra Arduino e Android.

Differenze di connessione tra dispositivo reale, e dispositivo emulato da Android studio:

- Utilizzo di BTClient per quello reale.
- Utilizzo di BTClientForEmulator per quello emulato.
    - Usa un task differente, preso dalla libreria Bluetooth (btlib) *ConnectedToEmulatedBluetoothServerTask che estende da ConnectionTask. Al posto di connettersi al Bluetooth utilizza una Socket per connettersi attraverso via seriale.* *ecco dove non riuscivi a collegarti nel 2018, mancava questo passo.*
    - Emulatore → Vostro PC dove hai in esecuzione l'emulatore → Viene in esecuzione un programma:  **Java che consente via seriale di comunicare via seriale con Arduino.**
      Test-PingPong-emo cartella  Lab 4_3.
        - Definisce un meccanismo per il terminatore di stringa **$\n**. l'idea è quella che se i messaggi terminano con un \n, allora si assume che quei messaggi arrivino da quel canale Bluetooth emulato, allora le risposte vengono inviate all'emulatore Android, viceversa se non

vengono identificati i simboli **$** allora i messaggi vengono inviati via seriale tra Arduino e qualunque cosa sia collegata in via seriale.

*NOTA: Nella versione emulata, eclipse lanciando i sorgenti dati in esempio dal prof, inizialmente mi restituisce che la porta dove è collegato Arduino è occupata, in seguito, invece il metodo OpenPort() va in errore, dicendo che la porta è chiusa.*
*No riesco quindi a lanciare il programma emulato da Java e tanto meno la connessione con Android studio dall'applicazione bt_pingPong_emu.*

## Modulo Lab Android

Grande spiegazione dei sensori, differenziandoli tra quelli software e hardware, a volte alcuni sensori sono emulati tramite la parte software, e aumentano il livello di astrazione dei dati che si vogliono ricevere.
Device di più alta qualità hanno un maggior numero di sensori disponibili.
Abbiamo un **SesorManager** che permette di creare l'istanza di un oggetto che rappresenta il servizio associato ad un sensore specifico, fornendo quindi i metodi per l'accesso ai sensori e per la registrazione di listener.
**Sensor** permette di creare l'istanza specifica per ogni sensore.
**SensorEvent** rappresenta l'istanza per ciascun evento propagato da ciascun sensore, include i dati prodotti dal sensore, le informazioni correnti associare al sensore etc…
**SensorEventListener** è l'interfaccia che deve essere implementata da qualunque oggetto che debba essere progettato per ricevere informazioni dai sensori d'interesse.

**GetDefaultSensor**() verifica la disponibilità di ogni singolo sensore tramite SensorManager.

```
SensorManager sm ;
Sensor fumo;
fumo = sm.getDefaultSensor(Sensor.TYPE_FUMO);
```

Se vogliamo prendere ogni variazione del valore di un sensore occorre che utilizziamo la funzione onSensorChanged che prende un SensorEvent "evento" come parametro.

```
public void onSensorChanged(SensorEvent evento){

        float actualvalue = evento.values[0];
        Log.d(LOG_TAG, "Actual Value: " + actualvalue);

}
```

Va in esecuzione nel thread principale, essendo che viene gestita dal sistema operativo, possiamo farla tramite il flusso di controllo principale.

Appunti lezione
26/11/2020

Sarebbe buona norma, ottenere accesso ai sensori, registrandoli per ricevere le notifiche delle variazioni dei valori, e de registrarli quando non si ha più la necessità di utilizzarli.

Esempio:

Nella parte di ON RESUME, vado a registrare il listener di uno specifico sensore attraverso il manager dei sensori.
Nella parte di ON PAUSE eseguo la de registrazione del listener sul sensor manager.
**Posso definire quanto delay inserire, quando eseguo la registrazione, cioè inserisco la frequenza precisa per la quale il sensore mi invia i dati che riceve, è una cosa molto importante per il consumo di risorse.**
***NOTA:*** *Per esempio l'accelerometro che riprende tutte le micro variazioni di movimento del dispositivo, potrei ritrovarmi con un numero considerevole di dati che potrei facilmente interpretare male e/o non sarei in grado di salvarmeli per il poco spazio a disposizione.*

1)
```
protected void onResume(){
        super.onResume();

if( lightSensor != null ){
        sm.registereListener(lsListener,
        lightSensor,SensorManager.SENSOR_DELAY_NORMAL);
}
```
2)
```
protected void onPause(){
        super.onPause();

if( lightSensor != null ){
        sm.unregistereListener(lsListener);
}
```

Orientamento di default por trait per i telefoni e landscape per i tablet.
Sono i sistemi di riferimento standard per dare significato al valore ricevuto dal sensore dell'accelerometro, e altri sensori di movimento.

## NFC

Near field communication sono i sensori di prossimità, che consentono gli initiator e i tag di comunicare.

Initietor invia un segnale che al contatto con il Tag legge il valore presente nel tag e entrambi con NFC attivo possono scambiarsi dei dati.
Utilizzato in ApplePay etc…

Appunti lezione
26/11/2020

Come risultato abbiamo dei record che sono divisi in 4 dati:
- TNF 3 bit definisce il tipo di tipo di valore che andate ad inserire
- Tipo
- ID
- Payload

In caso di NFC occorre dare dei permessi specifici al dispositivo per utilizzarli, senza di questi il sensore, se presente non effettuerà nessun tipo di azione.

## Meccanismo sui listener per gli aggiornamenti sulla posizione

Creiamo un listener per la posizione, che sposta la posizione attuale alla nuova posizione, ogni volta che cambia lo stato.
Utilizzando il GPS, possiamo ricevere degli aggiornamenti con l'opportuno listener, abbiamo due 0, che fanno riferimento all'accuratezza e alla frequenza in cui vogliamo i nostri aggiornamenti sulla posizione.

```
LocationManager lm = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,0, listener);
```

Anche per accedere ai sensori di localizzazione è necessario specificare alcuni permessi nel manifest come ad esempio il **ACCESS_COARSE_LOCATION** e il **ACCESS_FINE_LOCATION.**

Da Android 6.0 in poi google ha iniziato a differenziare i permessi in:
- **Permessi normal**
  - Sono tutti quei permessi che non hanno a che fare con la privacy dell'utente.(NFC, Bluetooth, stato accesso rete)
- **Permessi dangerous**
  - Sono quelli che hanno a che fare con la privacy dell'utente. (foto, record audio,memoria esterna, GPS, etc.) se l'utente non da il permesso, dobbiamo richiedere all'utente il permesso specifico di quel sensore, e faccio apparire un pop up per l'approvazione.

Se non vengono date le opportune conferme e permessi, si incorre in problemi di sicurezza.

Dal Bluetooth però possiamo capire con chi ci siamo trovati in prossimità, come app immuni accedere al Bluetooth è accedere alla privacy dell'utente.

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

## Tecnologie radio

RFID e NFC, utilizzati in localizzazione indoor, protocollo Bluetooth BLE, questi bacon consentono di eseguire una localizzazione, cioè luoghi dove un certo tipo di dispositivo si possa trovare, come tessere, oggetti etc. vedi gli APPLETAG.

Tipologie di comunicazione:

- Diretta
- Indiretta

Quando facciamo comunicazione lato Arduino non usiamo delle primitive bloccanti, usiamo un approccio asincrono, non voglio delle recive bloccanti, tutto vine impostato, in modo che la comunicazione avvenga in modo asincrono, solamente nel momento in cui una risorsa come il canale di scambio dei messaggi, viene utilizzato, per esempio la nostra seriale.

In un sistema a thread sotto un sistema operativo, potrebbe essere semplice avere un modello sincrono dei recive, nelle reti potrebbe essere così.

Non dobbiamo creare dei problemi di performance, quando si programma a thread con sotto un sistema operativo, andrei ad utilizzare troppo la CPU per cose di poca utilità creando del busywaiting.
**NOTA:***Valutare una libreria JSON per i messaggi e i dati scambiati nei canali, potrebbe risultare più semplice e più dinamico l'uso di questi, considerando anche l'approccio web del progetto finale.*

Creare un event loop, con task, in modo asincrono è il goal principale.
Questo unico event loop gestisce gli eventi, e reagisce mandando in esecuzione degli handler di eventi in modo atomico, cioè per ogni evento un handler specifico.

*Nicholas Ciarafoni*

**Sistemi Embedded e Internet-of-Things**

Appunti lezione
26/11/2020

21/12/2020
https://web.microsoftstream.com/video/befcc26d-5448-41ec-99d7-ff37e72cf7af?list=user&userId=8bf5e0a6-9e33-4a87-9fa5-b9332620bedc

## RECAP FINALE

Domande alla discussione finale:

- Che cosa sono le macchine a stati finiti,
- Che sono vuol dire sincrono e asincrono
- Perché è importante il periodo,
- Come scelgo il periodo in una macchina a stati finiti,
- Problema dell'overrun
- Capire problemi di quando abbiamo più task da mandare in esecuzione.
- Architetture ad eventi, macchine a stati finiti asincrone, e gli schemi implementativi, come applicarle ad Arduino.
- Differenze tra micro controllori e a system on a  Chip.
  concettualmente simili, ma architetture diverse, che permettono di avere più memoria, più capacità di calcolo, più potenza di calcolo, maggiori frequenze, sistemi operativi real time.
- Differenze sistemi operativi real time e sistemi non real time.
- Come faccio ad architettare una macchina super loop o event loop? con sotto un sistema operativo o no ?
- Capire i collegamenti IOT tra le varie zone di interesse.

Colloquio, parte dai progetti consegnati, colloquio individuale, anche con stessa relazione e stessi sorgenti, quando si va alla relazione occorre conoscere tutte le parti del progetto come se fossero state eseguite in autonomia, non vengono fatte differenze se una parte l'ha gestita uno al posto dell'altro.

Sistemi distribuiti e programmazione concorrente e smart city, non sono necessarie da sapere per il colloquio, però essere infarinati non è male.

Soluzioni degli assignment già consegnati.

## PROGETTO 1

Approccio a stati, di un architettura super loop.

Ha creato uno switch iniziale con lo stato del gioco, ad ogni cambiamento di stato un case differente che andava a lanciare uno specifico stato o scenario di gioco.

Crea dentro un file config.h tutti i parametri di configurazione e di sistema, che poi vengono inclusi nelle specifiche parti d'interesse.
Ricordati nei **.h** ci vanno le interfacce e nel **.cpp** ci vanno le implementazioni.

*Nicholas Ciarafoni*

Appunti lezione
26/11/2020

- ***Game_Core*** contiene le procedure e le dinamiche di gioco.
- ***Led_board*** è il modulo che gestisce gli aspetti associati alla gestione dei vari dispositivi e alla board con cui il sistema interagisce con l'utente.
- ***User_consolle*** ho tutte le procedure e le parti di gestione dell'input.
- ***Game_wait_toStart()*** il gioco va in una modalità di attesa.
- ***Game_Over*** parte finale del gioco, parsing del led con architettura super loop, cioè quando si trova in waitToStart, attende che sia premuto un pulsante, va avanti a fare il parsing, parsing viene suddiviso in più parti dell'architettura super loop.

Creare delle procedure per astrarre (o simulare in modo più pratico) quello che viene fatto durante il gioco.

## Progetto 2

Parte su microcontrollore e parte java.

La prima parte è suddivisa in 4 task, abbiamo 2 oggetti o risorse condivise, oggetti passivi.

- **ExManTask** si occupa della gestione generale dell'esperimento, con l'idea che sia il task che si occupa del guidare il risparmio energetico e simula tutti il comportamento dell'esperimento.
  Parte da uno stato di ready, poi passa allo stato di sleep, poi non appena rileva le persone esce e una volta in ready e contestualmente alla pressione del tasto start, parte l'esecuzione dell'esperimento, in cui si traccia la posizione dell'oggetto.
- **Tracking task** task specifico che traccia continuamente la distanza.
  Lo uso perché ho voluto disaccoppiare le frequenze e i periodi con cui poteva funzionare la macchina e la gestione dell'esperimento generale.
  Al suo interno ha una macchina a stati finiti formata da 4 stati.
  Questo task ha un periodo base di 10 millisecondi.
  - **IDLE** quando non è partito l'esperimento è in questo stato.
    Permette la scelta di 4 frequenze diverse, e tra queste inserisce un delta differente, a seconda di questa frequenza, stabilisco i parametri che mi dicono ogni quanto devo andare a fare il campionamento.
  - **WORKING** quando l'esperimento è partito.
    - *READY*
    - *ONGOING*
    - *COMPLETED*
    - *ERROR*

Macchina stati finiti sincrona, con un periodo di 40 millisecondi, è stato scelto in base alla reattività dei pulsanti, evita e risolve il problema del bouncing.

- **UserConsole** serve per interazione input output con l'utente e invia i dati alla console o alla parte in esecuzione su PC, si usa la seriale asincrona.
- **TraskSpeed Task** specifico per indicare la sua velocità, ha un suo periodo di 100 millisecondi, per consentire il posizionamento nella posizione giusta. legge ogni volta

che arriva un tic del periodo e setta il tachimetro alla velocità che la funzione rileva. Posiziona il motore nella posizione giusta (usare per la pompa del terzo progetto)
- **Blinking task**, task specifico generato in caso d'errore, e l'oggetto si ferma anche durante l'esecuzione.

Per cambiare lo stato imposta una variabile **state**.

Parte su PC quindi lato java

Ha 3 componenti principali tra i quali:
- **MonitoringAgent** Si mette in attesa sul canale per ricevere i messaggi da Arduino che a seconda del prefisso che ha indica una precisa tipologia di msg , tra quelli di log e proprio dell'esperimento, anche i cambiamenti di stato e viene aggiornata la view con un pannello, con cui viene disegnato un qualcosa.
  Visualizza gli ultimi N elementi tracciati in questo buffer di 800 valori.
- **SmartExpView**
- **LogView**

## Wearable computing and Mixed reality

Tutta quella porzione di infrastruttura hardware and software mista che va sotto il cappello dei dispositivi indossabili, l'essere umano diventa parte delle things dell'IoT, e diventa a far parte del sistema per portare a un cambiamento fondamentale del nostro modo di vivere, lo smartphone è un entry level di questo sistema di dispositivi interconnessi.
Werable **computers** sono tutti i dispositivi che sono in grado di essere indossati e portati con se, deve essere un dispositivi pro attivo per l'utente.
Werable **computing** dispositivi in real time, con una potenza di calcolo personalizzata e utile per le cose di ogni giorno.
**Google Glass**, primi smart glass, che riportano l'idea di werable in voga, facendo prendere piede esempio oculus.
**Myo Armband**, dispositivo che riconosce gli impulsi elettrici dei muscoli che propagano e consentono di programmare applicazioni mobile interagibili con uno smartphone.
Alcuni non montano dei sistemi operativi, altri invece sono più sofisticati e montano un SO.

Grande discorso rappresentativo che sta alla base di tutti i dispositivi di realtà aumentata come visori, occhiali e altri simili:
**MAGIC LEAP ONE**
**Mirage Architecture.**