Internet Of Things – A.A. 2021/2022
Domande, risposte ed esercizi proposti

Esame del 21/01/2021

Che cosa è un beacon in una rete 802.11? Chi lo trasmette?

La rete 802.11 per funzionare ha bisogno di essere rilevata dai dispositivi che hanno intenzione di collegarsi ad essa, e per fare ciò è necessario utilizzare i beacon. I beacon sono pacchetti segnalatori trasmessi in broadcast su tutta la rete a tempi schedulati (schedulati perché si evita una trasmissione sovrapposta ad un'altra); questi broadcast, quindi, vengono raggiungi sia dai dispositivi già connessi alla rete sia a quei dispositivi che hanno intenzione di collegarsi ad essa: un dispositivo per connettersi alla rete 802.11 ha necessariamente bisogno di informazioni, in particolare della SSID che rappresenta il nome della rete dove vogliamo collegarci.

Descrivere le tre classi di funzionamento dei dispositivi LoRaWAN

I dispositivi LoRaWAN supportano tre tipologie di classi di funzionamento:

- **Classe A: i dispositivi comunicano tra di loro tramite comunicazione bidirezionale, in cui abbiamo un uplink per il dispositivo finale e due finestre per il downlink.**
- **Classe B: come la classe A, ma i dispositivi di classe B aprono finestre di ricezione extra in orari schedulati (in modo da evitare collegamenti duplicati). Necessita di un gateway per effettuare le sincronizzazioni.**
- **Classe C: come la classe B, ma le finestre di ricezione sono quasi sempre aperte fino a quando non avviene una trasmissione di informazioni.**

Spiegare sinteticamente il funzionamento del protocollo MQTT

MQTT sta per Message Queuing Telemetry Transport ed un protocollo di trasmissione dati che sfrutta il protocollo TCP/IP, basandosi sul concetto di Publish/Subscribe. In pratica, il mittente invia un messaggio di un determinato argomento e il destinatario risponde a questi messaggi con una conferma (publish – subscribe); per effettuare questa operazione, è necessario utilizzare i broker, ossia un dispositivo che fa da tramite tra il mittente e il destinatario. I broker MQTT hanno il compito di gestire i messaggi da parte del mittente e del destinatario.

Esercizio Arduino:

Si implementi uno sketch Arduino che realizzi un semaforo con chiamata pedonale. Il semaforo deve permettere al Led rosso di rimanere acceso fintanto che un "pedone" non avrà premuto il pulsante di "chiamata pedonale". A questo punto il pulsante deve fornire un segnale digitale alla scheda Arduino e questa dovrà fare accendere il Led verde dopo cinque secondi (tempo di attesa per permettere al semaforo "lato macchine" di diventare rosso). Il Led verde dovrà rimanere acceso per 10 secondi per poi passare il testimone al Led giallo (rimarrà acceso per 3 secondi), per poi ritornare nuovamente al Led rosso acceso. A titolo esemplificativo si riporta uno schema di collegamento del sistema dove i led rappresentano le luci del semaforo. Il pulsante è collegato al pin 2, il led verde è collegato al pin 3, il led giallo è collegato al pin 4, il led rosso è collegato al pin 5.

```
#define led_rosso_pedone 5
#define led_giallo_pedone 4
#define led_verde_pedone 3
#define button 2

void setup(){
  pinMode(led_rosso_pedone, OUTPUT);
```

```
  pinMode(led_giallo_pedone, OUTPUT);
  pinMode(led_verde_pedone, OUTPUT);
  pinMode(button, INPUT);
}

void loop(){
  digitalWrite(led_rosso_pedone, HIGH);
  digitalWrite(led_giallo_pedone, LOW);
  digitalWrite(led_verde_pedone, LOW);
  stato = digitalRead(button); //preme il bottone e dopo 5 secondi diventa verde
  if(stato==1){
    delay(5000)
    digitalWrite(led_rosso_pedone, LOW);
    digitalWrite(led_verde_pedone, HIGH);
    delay(10000); //rimane acceso per 10 sec
    digitalWrite(led_verde_pedone, LOW);
    digitalWrite(led_giallo_pedone, HIGH); //dopo accende il led giallo per 3 sec
    delay(3000);
    digitalWrite(led_giallo_pedone, LOW);
    digitalWrite(led_rosso_pedone, HIGH);
  }
}
```

Esame del 14/01/2022

Spiegare il fenomeno dei terminali nascosti nelle reti 802.11 ed eventuali soluzioni per mitigare il problema.

Nelle reti 802.11 può esserci la probabilità che due o più stazioni non riescono a contattare un access point in quanto risulta essere troppo lontano rispetto al range massimale. In questo caso, la stazione può aver rilevato delle possibili ritrasmissioni o trasmissioni simultanee su uno stesso canale di comunicazione, causando una collisione. Per evitare questo problema si utilizza la trasmissione TCP/IP handshake a quattro vie e utilizza due variabili:

- **RCS: sta per "request to send" e serve per verificare se un canale di comunicazione è libero**
- **CTS: sta per "clear to send" e serve come risposta all'autorizzazione effettuata durante la fase del RCS**

Durante la fase di RCS e CTS abbiamo un timer stabilito direttamente da loro stessi e, fino allo scadere del tempo, i restanti dispositivi non saranno in grado di inviare pacchetti in quanto la rete risulta essere occupata (pur rilevandola come libera)

Spiegare a cosa serve e come funziona il modulo ADR per le reti LoRaWAN

Nelle reti LoRaWAN sono presenti i moduli ADR, che hanno il compito di captare la quantità di disturbo e la potenza dei pacchetti che provengono dalle singole stazioni (il valore ottenuto verrà assegnato al SF, tra la stazione e il gateway). Il nostro obiettivo è di avere un valore di SF nella media, in quanto:

- **Se il valore è molto basso, esso risulta essere molto veloce ed efficiente ma è più soggetto a perdita di segnale e disturbi;**
- **Se il valore è molto alto, esso risulta essere molto letto e poco efficiente ma è più protettivo riguardo la perdita di segnale e disturbi.**

Spiegare come funzionano, e quali sono le differenze, tra una multiplazione a divisione di tempo TDMA (Time Division Multiple Access), e una multiplazione a divisione di frequenza FDMA (Frequency Division Multiple Access).

TDMA e FDMA sono tecniche di multiplazioni diverse, necessarie per l'accesso multiplo di un sistema di comunicazione. Il TDMA divide lo spettro radio in intervalli di tempo e in ogni settore disponibile un solo utente può trasmettere o ricevere; invece, la FDMA consente di divere la larghezza di banda in frequenze stabilite: in quella determinata frequenza solo un utente può lavorarci, e nessun altro.

Spiegare come funziona un convertitore analogico digitale (ADC).

La conversione analogico digitale si basa su tre fondamentali fasi:

- **Campionamento: ci consente di trasformare un onda di grandezza analogica in una grandezza che varia in specifici istanti di tempo, chiamato periodo di campionamento. Più sono i periodi di campionamento, maggiore sarà la qualità ottenuta delle informazioni ma peggiore sarà la durata dell'esecuzione.**
- **Quantizzazione: ci consente di limitare il numero di valori che vogliamo conservare per la futura grandezza. Questa operazione può introdurre degli errori in quanto sfrutta l'approssimazione.**
- **Codifica: ci consente di prendere i valori ottenuti (suddivisi in intervalli) nella fase di quantizzazione e di convertirli in segnale digitale (nella maggior parte dei casi viene utilizzato il codice binario).**

Esercizio Arduino:

Si implementi uno sketch Arduino che realizzi l'interfaccia di funzionamento di una macchinetta del caffè. A tal fine si consideri la board Arduino e i relativi collegamenti presenti in figura. Sono previsti due pulsanti per il controllo dello zucchero e un pulsante per l'avvio della preparazione. Il livello dello zucchero è visualizzato all'utente tramite la barra led di sinistra (tutti led spenti significa assenza di zucchero, tutti i led accesi significa massima dose di zucchero, si considerino 3 livelli intermedi). La pressione dei due pulsanti (up/down) riduce/aumenta la quantità di zucchero. Il livello di completamento della preparazione viene visualizzato all'utente attraverso la barra led di destra. Premendo sul pulsante di destra l'utente avvia la preparazione, il tempo di preparazione è pari a quattro secondi. L'utente a contezza della preparazione tramite accensione progressiva della barra led di destra (ad ogni secondo si accenderà un led in più fino al completamento). Durante la fase di preparazione del caffè il controllo della quantità di zucchero è disabilitato.

```
#define button_add 13
#define button_remove 12
#define button_start 7
#define led1_sugar 11
#define led2_sugar 10
#define led3_sugar 9
#define led4_sugar 8
#define led1_coffee 6
#define led2_coffee 5
#define led3_coffee 4
#define led4_coffee 3

void setup(){
  pinMode(button_add, INPUT);
  pinMode(button_remove, INPUT);
```

```
    pinMode(button_start, INPUT);
    pinMode(led1_sugar, OUTPUT)
    pinMode(led2_sugar, OUTPUT)
    pinMode(led3_sugar, OUTPUT)
    pinMode(led4_sugar, OUTPUT)
    pinMode(led1_coffee, OUTPUT)
    pinMode(led2_coffee, OUTPUT)
    pinMode(led3_coffee, OUTPUT)
    pinMode(led4_coffee, OUTPUT)
}

void loop(){
    digitalWrite(led1_sugar, LOW);
    digitalWrite(led2_sugar, LOW);
    digitalWrite(led3_sugar, LOW);
    digitalWrite(led4_sugar, LOW);
    digitalWrite(led1_coffee, LOW);
    digitalWrite(led2_coffee, LOW);
    digitalWrite(led3_coffee, LOW);
    digitalWrite(led4_coffee, LOW);
    flag = false;
    add_sugar = digitalRead(button_add);
    remove_sugar = digitalRead(button_remove);
    start_coffee = digitalRead(button_start);

    if(!flag){
        if(add_sugar==1 && led1_sugar == LOW)
            digitalWrite(led1_sugar, HIGH);
        else if(add_sugar==1 && led2_sugar == LOW)
            digitalWrite(led2_sugar, HIGH);
        else if(add_sugar==1 && led3_sugar == LOW)
            digitalWrite(led3_sugar, HIGH);
        else if(add_sugar==1 && led4_sugar == LOW)
            digitalWrite(led4_sugar, HIGH);

        if(remove_sugar==1 && led1_sugar == HIGH)
            digitalWrite(led1_sugar, LOW);
        else if(remove_sugar==1 && led2_sugar == HIGH)
            digitalWrite(led2_sugar, LOW);
        else if(remove_sugar==1 && led3_sugar == HIGH)
            digitalWrite(led3_sugar, LOW);
        else if(remove_sugar==1 && led4_sugar == HIGH)
            digitalWrite(led4_sugar, LOW);

        if(start_coffee==1){
            digitalWrite(led1_coffee, HIGH);
            delay(1000);
            digitalWrite(led2_coffee, HIGH);
```

```
        delay(1000);
        digitalWrite(led3_coffee, HIGH);
        delay(1000);
        digitalWrite(led4_coffee, HIGH);
        delay(1000);
        flag = true;
      }
    }
}
```

Esame del 24/01/2022

Descrivere il principio di funzionamento delle modulazioni digitali ed entrare nel dettaglio di una di loro (a scelta).

Una modulazione digitale è un metodo che consiste di prendere in esame una porzione di un segnale analogico e di ricavarne delle informazioni, convertendole successivamente in codice binario. La modulazione avviene ad esempio nelle tv, nei dispositivi mobili, nei satelliti e nei ponti radio.

Quindi, la modulazione rappresenta l'insieme delle tecniche di trasmissione dove vengono manipolati due segnali:

- **Modulante, segnale elettrico che contiene le informazioni dei dati;**
- **Portante, segnale elettrico sviluppato ad alta frequenza.**

Le modulazioni possono essere di due tipologie:

- **Analogiche;**
- **Digitale.**

Una modulazione digitale abbastanza utilizzata ai giorni nostri è la modulazione FSK, che consiste nel variare la frequenza della portante con il ritmo del segnale modulante.

Riassumere il meccanismo di backoff esponenziale per le reti IEEE 802.11

Nelle reti WiFi esiste la possibilità che possano avvenire collisioni durante la trasmissione di pacchetti da due o più stazioni diverse; per evitare ciò, si utilizza un timer chiamato backoff. Il backoff ha il compito di fermare per un tempo fissato delle stazioni prima ancora di poter trasmettere. Se avessimo una perdita di pacchetto, la finestra del backoff verrebbe calcolato come calcolo esponenziale (2^m), dove m corrisponde al numero di pacchetti persi (se perderemo un pacchetto, la finestra raddoppierà causando un attesa indefinita da parte della stazione).

Descrivere le tre classi di funzionamento dei dispositivi LoRaWAN

I dispositivi LoRaWAN supportano tre tipologie di classi di funzionamento:

- **Classe A: i dispositivi comunicano tra di loro tramite comunicazione bidirezionale, in cui abbiamo un uplink per il dispositivo finale e due finestre per il downlink.**
- **Classe B: come la classe A, ma i dispositivi di classe B aprono finestre di ricezione extra in orari schedulati (in modo da evitare collegamenti duplicati). Necessita di un gateway per effettuare le sincronizzazioni.**
- **Classe C: come la classe B, ma le finestre di ricezione sono quasi sempre aperte fino a quando non avviene una trasmissione di informazioni.**

Spiegare sinteticamente il funzionamento del protocollo MQTT

MQTT sta per Message Queuing Telemetry Transport ed un protocollo di trasmissione dati che sfrutta il protocollo TCP/IP, basandosi sul concetto di Publish/Subscribe. In pratica, il mittente invia un messaggio di un determinato argomento e il destinatario risponde a questi messaggi

con una conferma (publish – subscribe); per effettuare questa operazione, è necessario utilizzare i broker, ossia un dispositivo che fa da tramite tra il mittente e il destinatario. I broker MQTT hanno il compito di gestire i messaggi da parte del mittente e del destinatario.

Esercizio Arduino:

Si implementi uno sketch Arduino che realizzi l'interfaccia di funzionamento di un termostato per il riscaldamento. A tal fine si consideri la board Arduino e i relativi collegamenti presenti in figura. Sono previsti due pulsanti per il settaggio della temperatura desiderata e un sensore TMP36 per il rilevamento della temperatura collegato al pin A0. Il valore di temperatura impostato è visualizzato all'utente tramite la barra led (ad ogni led corrisponde uno specifico valore di temperatura, da sinistra verso destra, è possibile settare i seguenti valori di temperatura, 14°, 16°, 18°, 20°). La pressione dei due pulsanti (up/down) riduce/aumenta la temperatura (solo tra 14° e 20°, e sempre a step di 2). Quando il valore di temperatura rilevato va al di sotto di quello impostato allora il sistema accende il led presente nella breadboard di destra, che altrimenti deve rimanere spento. Il TMP36 permette di acquisire temperature comprese nell'intervallo tra -40°C e +125°C restituendo in uscita valori di tensione lineari tra circa 0.1Vdc e 1.7Vdc. Una variazione di un grado produce una variazione della tensione di uscita pari a 10mV; alla temperatura di 0°C il sensore eroga una tensione di 500mV. Una volta convertito il valore letto sul piedino (0-1024 intero), in millivolt, si usi la seguente formula per la conversione in temperatura: temperatureC = (milliVolts - 500) / 10

```
#define button_up 12
#define button_down 13
#define led_14 11
#define led_16 10
#define led__18 9
#define led_20 8
#define temp A0
#define led_t 6

void setup(){
   pinMode(button_up, INPUT);
   pinMode(led_14, OUTPUT);
   pinMode(led_16, OUTPUT);
   pinMode(led_18, OUTPUT);
   pinMode(led_20, OUTPUT);
   pinMode(led_t, OUTPUT);
   pinMode(button_down, INPUT);
   pinMode(temp, INPUT);
}

void loop(){
   digitalWrite(led_14, LOW);
   digitalWrite(led_16, LOW);
   digitalWrite(led_18, LOW);
   digitalWrite(led_20, LOW);
   digitalWrite(led_t, LOW);

   temp_up = digitalRead(button_up);
```

```
  temp_down = digitalRead(button_down);

  if(temp_up==1 && led_14==LOW){
    digitalWrite(led_14, HIGH);
  }
  else if(temp_up==1 && led_16==LOW){
    digitalWrite(led_16, HIGH);
  }
  else if(temp_up==1 && led_18==LOW){
    digitalWrite(led_18, HIGH);
  }
  else if(temp_up==1 && led_20==LOW){
    digitalWrite(led_20, HIGH);
  }

  if(temp_down==1 && led_14==HIGH){
    digitalWrite(led_14, LOW);
  }
  else if(temp_down==1 && led_16==HIGH){
    digitalWrite(led_16, LOW);
  }
  else if(temp_down==1 && led_18==HIGH){
    digitalWrite(led_18, LOW);
  }
  else if(temp_down==1 && led_20==HIGH){
    digitalWrite(led_20, LOW);
  }
  temperatura = analogRead(temp);
  float millVolt = temperatura * (5000/1024); //trovare la temperatura espressa in °C
  temperaturaC = (millVolt - 500) /10;

  if(temperaturaC<14.0)
    digitalWrite(led_t, HIGH);
  else
    digitalWrite(led_t, LOW);
}
```

Esame del 24/01/2022

Descrivere il principio di funzionamento di un convertitore analogico digitale per board Arduino

La board Arduino, oltre alle porte PWM, sono presenti anche delle porte di Input per la lettura analogica (A0 – A5), ciascuna con risoluzione di 10 bit. Per convertire un valore analogico ad un valore digitale viene sfrutta la formula 2^10-1, che corrisponde a 1023 (ossia 5V). Per calcolare la tensione in ingresso basta fare la seguente proporzione:

V_ingresso:Valore_ADC=5:1024

Per estrapolare il valore della porta analogica utilizziamo la funzione analogRead e, tramite la proporzione precedente, possiamo ottenere il valore digitale.

Si ricordi che una conversione analogico digitale sfrutta tre metodi:

- **Campionamento;**
- **Quantizzazione;**
- **Codifica.**

Nella tecnologia WiFi IEEE 802.11, quali sono le principali differenze tra le due modalità di funzionamento, Infrastructure BSS e Independent BSS?

BSS è una topologia di rete che consente ai dispositivi wireless di comunicare con un mezzo di comunicazione comune, come gli access point. Questa topologia prende il nome di Infrastructure BSS. BSS, oltre al caso precedente, ha la possibilità di poter essere una topologia di rete dove i dispositivi comunicano tra di loro tramite collegamento diretto (ad-hoc): in questo caso prende il nome di Independent BSS. Si predilige la Infrastructure BSS se dovessimo connettere e comunicare con dispositivi che non hanno la possibilità di poter collegare direttamente al modem (vedasi smartphone); viceversa, si predilige la Independent BSS quando vogliamo comunicare tra due o più stazioni che sono collegati direttamente tra di loro. Dato che l'Infrastructure BSS sfrutta la tecnologia wireless, è possibile che la trasmissione di pacchetti subisca una latenza (delay), mentre nella Independent BSS la trasmissione di pacchetti è diretta (senza latenza).

Descrivere il sistema di cifratura dei pacchetti utilizzati nella rete LoRaWAN.

Nelle reti LoRaWAN, via la loro infrastruttura, è necessario che i pacchetti che devono essere trasmessi devono essere cifrati. Per poter cifrare i dati, è necessario avere alcune informazioni:

- **DevEUI: corrisponde al codice identificativo univoco del dispositivo che stiamo andando a utilizzare per mandare i pacchetti;**
- **AppEUI: corrisponde al codice identificativo univoco del server dell'applicazione;**
- **AppKey: corrisponde alla chiave crittografata, che si interpone tra il mittente e il destinatario.**

Ogni dispositivo, prima di poter inviare o ricevere messaggi, devono necessariamente attivare uno dei due metodi seguenti:

- **OTAA: metodo più sicuro e consigliato perché i dispositivi eseguono una procedura di unione con la rete, ove viene assegnato un indirizzo IP dinamico e una chiave di sicurezza (viene memorizzata nel dispositivo)**
- **ABP: richiede hardcoding dell'indirizzo del dispositivo e della chiave di sicurezza. Presenta come svantaggio che non può cambiare provider di rete senza poter cambiare manualmente le chiavi.**

La cifratura dei pacchetti avviene tramite l'algoritmo hash, che consiste di estrapolare le informazioni del pacchetto e di modificarle in modo da poterle decifrare una volta arrivata a destinazione: se i valori tra mittente e destinatario coincidono, allora il pacchetto è arrivato senza modifiche o alterazioni. Durante la cifratura, viene assegnata una firma digitale da parte del mittente (in modo tale che, durante la verifica, i dati non vengano manomessi).

Spiegare le principali differenze tra un microcontrollore e un microprocessore.

Il microprocessore è un'unita di controllo di un microcontrollore, e ha il compito di svolgere operazioni a livello logico (ALU, ad esempio, svolge calcoli matematici). Il microprocessore, a differenza del microcontrollore, è costituito da un'unica unità di elaborazione dati, mentre il microcontrollore rappresenta quel sistema che contiene svariati componenti (CPU, memoria, I/O).

Esercizio Arduino:

Si implementi uno sketch Arduino che realizzi un sistema di segnalazione un basato su barra led (stile SuperCar). Il sistema è composto da cinque led, i led devono accendersi, uno alla volta, da sinistra verso destra e viceversa. Il funzionamento deve essere a ciclo continuo. Il tempo in cui ciascun led rimane acceso è settato dall'utente attraverso l'utilizzo di un potenziometro analogico.

Il tempo di accensione deve essere configurabile nell'intervallo 0,1 - 1 secondo (0,1sec è il minimo, 1 sec è il massimo). A tal fine, si consideri la board Arduino e i relativi collegamenti presenti in figura. Il potenziometro analogico è collegato al pin A0 per il rilevamento del valore di settaggio.

```
#define led1 11
#define led2 10
#define led3 9
#define led4 8
#define led5 7
#define led6 6
#define potenziometro A0

void setup(){
   pinMode(led1, OUTPUT);
   pinMode(led2, OUTPUT);
   pinMode(led3, OUTPUT);
   pinMode(led4, OUTPUT);
   pinMode(led5, OUTPUT);
   pinMode(led6, OUTPUT);
   pinMode(potenziometro, INPUT);
}

void loop(){
   digitalWrite(led1, LOW);
   digitalWrite(led2, LOW);
   digitalWrite(led3, LOW);
   digitalWrite(led4, LOW);
   digitalWrite(led5, LOW);
   digitalWrite(led6, LOW);

   signal = analogRead(potenziometro);

   if(signal>255){
      delay(1000);
      digitalWrite(led3, HIGH);
      digitalWrite(led4, HIGH);
      delay(1000);
      digitalWrite(led3, LOW);
      digitalWrite(led4, LOW);
      digitalWrite(led2, HIGH);
      digitalWrite(led5, HIGH);
      delay(1000);
      digitalWrite(led2, LOW);
      digitalWrite(led5, LOW);
      digitalWrite(led1, HIGH);
      digitalWrite(led6, HIGH);
      delay(1000);
      digitalWrite(led1, LOW);
```

```
    digitalWrite(led6, LOW);
  }
  else if(signal<=0){
    delay(100);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    delay(100);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    digitalWrite(led2, HIGH);
    digitalWrite(led5, HIGH);
    delay(100);
    digitalWrite(led2, LOW);
    digitalWrite(led5, LOW);
    digitalWrite(led1, HIGH);
    digitalWrite(led6, HIGH);
    delay(100);
    digitalWrite(led1, LOW);
    digitalWrite(led6, LOW);
  }
  else{
    valore = map(signal, 1, 1022, 200, 900);
    delay(valore);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    delay(valore);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    digitalWrite(led2, HIGH);
    digitalWrite(led5, HIGH);
    delay(valore);
    digitalWrite(led2, LOW);
    digitalWrite(led5, LOW);
    digitalWrite(led1, HIGH);
    digitalWrite(led6, HIGH);
    delay(valore);
    digitalWrite(led1, LOW);
    digitalWrite(led6, LOW);
  }
}
```

Esame del 17/02/2022

Qual è il principio di funzionamento alla base della conversione digitale analogica tramite PWM e come si definisce il duty cycle?

Per effettuare la conversione digitale analogico abbiamo necessariamente bisogno di utilizzare il PWM (Pulse Width Modulation). Ciascun pin GPIO porta in uscita una tensione approssimata (se volessimo una tensione più accurata, bisognerebbe utilizzare componenti elettronici come il MOSFET). I pin digitali possono emettere 3V o 5V ma, durante la fase di OUTPUT, possiamo

ottenere lo stesso effetto ma con voltaggio differente. Questo dipende anche dal duty cycle, che sarebbe la percentuale di tempo durante un periodo.

Che cosa è un beacon in una rete WiFi IEEE 802.11? Chi lo trasmette nella modalità infrastruttura e ad-hoc? Quale è l'impatto sulla procedura di scansione attiva e passiva?

La rete 802.11 per funzionare ha bisogno di essere rilevata dai dispositivi che hanno intenzione di collegarsi ad essa, e per fare ciò è necessario utilizzare i beacon. I beacon sono pacchetti segnalatori trasmessi in broadcast su tutta la rete a tempi schedulati (schedulati perché si evita una trasmissione sovrapposta ad un'altra); questi broadcast, quindi, vengono raggiungi sia dai dispositivi già connessi alla rete sia a quei dispositivi che hanno intenzione di collegarsi ad essa: un dispositivo per connettersi alla rete 802.11 ha necessariamente bisogno di informazioni, in particolare della SSID che rappresenta il nome della rete dove vogliamo collegarci.

Il passive scanning è un metodo di rilevamento delle vulnerabilità, che si basa sullo sniffing dei pacchetti che vengono inviati durante la trasmissione. Per un intruso, il vantaggio è di non lasciare tracce che potrebbero avvisare utenti e amministratori; invece, per un amministrato il vantaggio è che non rischia di causare comportamenti indesiderati sul computer di destinazione.

L'active scanning è una tecnica più costosa e complessa rispetto al passive, e comporta effetti di vasta portata sui tempi delle attività e sull'affidabilità del sistema.

Presentare l'architettura e il funzionamento del protocollo MQTT. Elencare i diversi attori che compongono il sistema e il loro comportamento. Soffermarsi sui concetti di Topic, Keep-alive Time, Last Will & Testament (LWT) e Wildcards.

MQTT sta per Message Queuing Telemetry Transport ed un protocollo di trasmissione dati che sfrutta il protocollo TCP/IP, basandosi sul concetto di Publish/Subscribe. In pratica, il mittente invia un messaggio di un determinato argomento e il destinatario risponde a questi messaggi con una conferma (publish – subscribe); per effettuare questa operazione, è necessario utilizzare i broker, ossia un dispositivo che fa da tramite tra il mittente e il destinatario. I broker MQTT hanno il compito di gestire i messaggi da parte del mittente e del destinatario.

MQTT si basa sulle seguenti caratteristiche:

- **Topic: corrisponde ad un argomento (generalmente una stringa con codifica UTF-8) dove viene mandato al destinatario per svolgere determinati compiti. Il topic può essere diviso in uno o più livelli, ciascuno separato da un slash.**
- **Keep-alive Time: rappresenta il tempo massimo di comunicazione del MQTT client (generalmente è di 60 secondi)**
- **Last Will & Testament: si tratta della gestione dello stato dei componenti a fronte disconnessioni improvvise e non gestite. Serve per capire se un componente riesce a connettersi ad Internet o per capire se lo stato del dispositivo è Online o Offline**
- **Wildcard: quando un cliente sottoscrive un topic, va a cercare effettivamente la posizione del topic per poi poterlo eseguire. Una wildcard può essere di due tipologie:**
    - **Singolo livello: cerco un livello specifico /myhome/groundfloor/+/temperature**
    - **Multilivello: cerco un livello generico /myhome/groundfloor/#**

Spiegare a cosa serve e come funziona il modulo ADR-NET (Adaptive Data Rate network side) per le reti LoRaWAN. Quali sono i due parametri (radio) del dispositivo LoRaWAN che vengono settati attraverso i comandi dell'ADR?

Nelle reti LoRaWAN sono presenti i moduli ADR, che hanno il compito di captare la quantità di disturbo e la potenza dei pacchetti che provengono dalle singole stazioni (il valore ottenuto verrà assegnato al SF, tra la stazione e il gateway). Il nostro obiettivo è di avere un valore di SF nella media, in quanto:

- **Se il valore è molto basso, esso risulta essere molto veloce ed efficiente ma è più soggetto a perdita di segnale e disturbi;**
- **Se il valore è molto alto, esso risulta essere molto letto e poco efficiente ma è più protettivo riguardo la perdita di segnale e disturbi.**

I due parametri che vengono settati dal dispositivo LoRaWAN durante l'utilizzo dei comandi ADR sono SF e TP:

- **SF rappresenta il tempo in cui viene trasmesso un "simbolo"; questo viene fatto con una sorta di modulazione in frequenza in cui utilizziamo portanti diverse per SF diversi.**
- **TP rappresenta la potenza di trasmissione dei pacchetti, espresso in dB.**

Esercizio Arduino:

Si implementi uno sketch Arduino che realizzi un sistema di segnalazione del livello di un liquido contenuto in un bicchiere. A tal fine, si consideri la board Arduino e i relativi collegamenti presenti in figura. Il sistema è composto da tre led, e un Water Level Sensor. Il sensore riporta sul pin S (collegato al pin A0 dell'Arduino) una valore proporzionale al livello del liquido. Il sensore è stato calibrato rispetto ai due livelli di minimo e massimo, e riporta in corrispondenza le seguenti letture: lowerThreshold = 420 upperThreshold = 520 (questi sono i valori in uscita alla funziona analogRead). I tre led devono segnalare se il liquido si trova sotto il minimo (<lowerThreshold), led verde, tra il minimo e il massimo (>lowerThreshold and < upperThreshold), led giallo, sopra il massimo (>upperThreshold), led rosso. Soltanto un led per volta deve rimanere acceso.

```
#define led1 2
#define led2 3
#define led3 4
#define sensore A0 //fra 420 e 520

void setup(){
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(sensore, INPUT);
}

void loop(){
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);

  int stato = analogRead(sensore);

  if(stato<420 && led2==HIGH && led3==HIGH){
    digitalWrite(led1, HIGH);
```

```
      digitalWrite(led2, LOW);
      digitalWrite(led3, LOW);
    }
    else if(stato>520 && led1==HIGH && led3==HIGH){
      digitalWrite(led1, LOW);
      digitalWrite(led2, HIGH);
      digitalWrite(led3, LOW);
    }
    else if(stato>421 && stato<519 && led1==HIGH && led2==HIGH){
      digitalWrite(led1, LOW);
      digitalWrite(led2, LOW);
      digitalWrite(led3, HIGH);
    }
}
```