

Università degli Studi della Campania

Luigi Vanvitelli - Dipartimento di Ingegneria

Laurea Triennale in Ingegneria Elettronica e Informatica

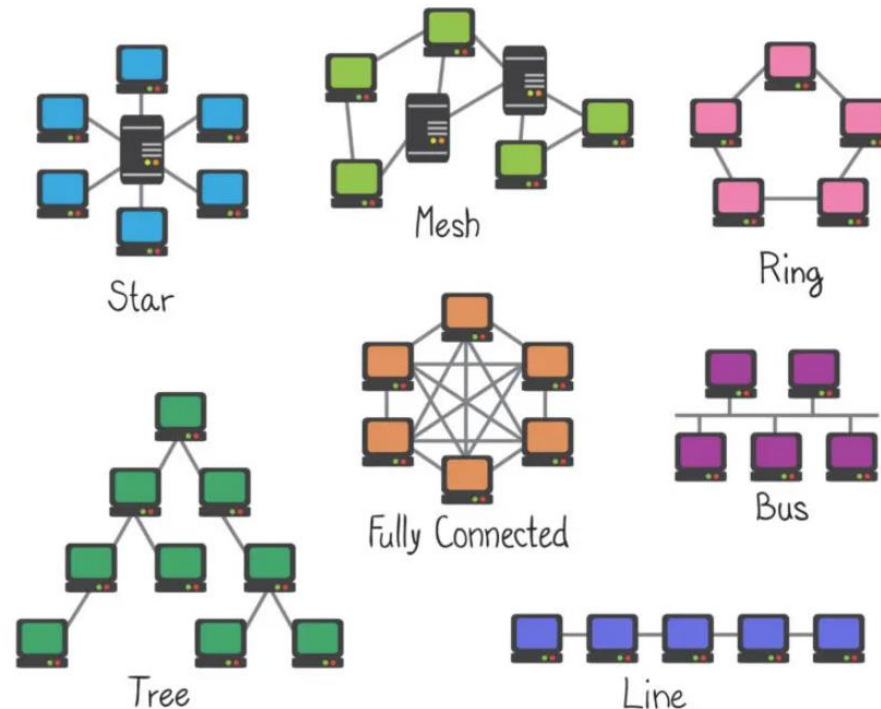
Laboratorio di Sviluppo di Applicazioni per IoT
a.a. 2023-2024

Topologie di Rete

Docente: Carlo Mazzocca
e-mail: carlo.mazzocca@unibo.it

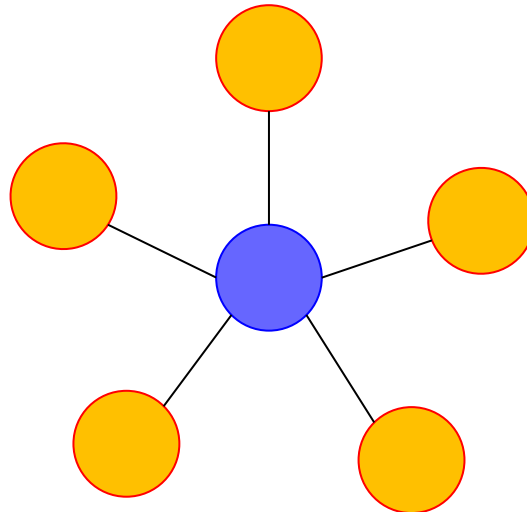
Tipi di Topologie

- Le topologie di rete definiscono il modo in cui sono collegati e comunicano tra di loro
- La scelta dipende dalle esigenze specifiche della rete, inclusi i requisiti di prestazioni, affidabilità e costi



Stella

- Tutti i dispositivi hanno un collegamento diretto al nodo centrale della rete, chiamato hub o switch
- Tutte le comunicazioni passano per il nodo centrale che è l'unico che può scambiare messaggi con i nodi remoti



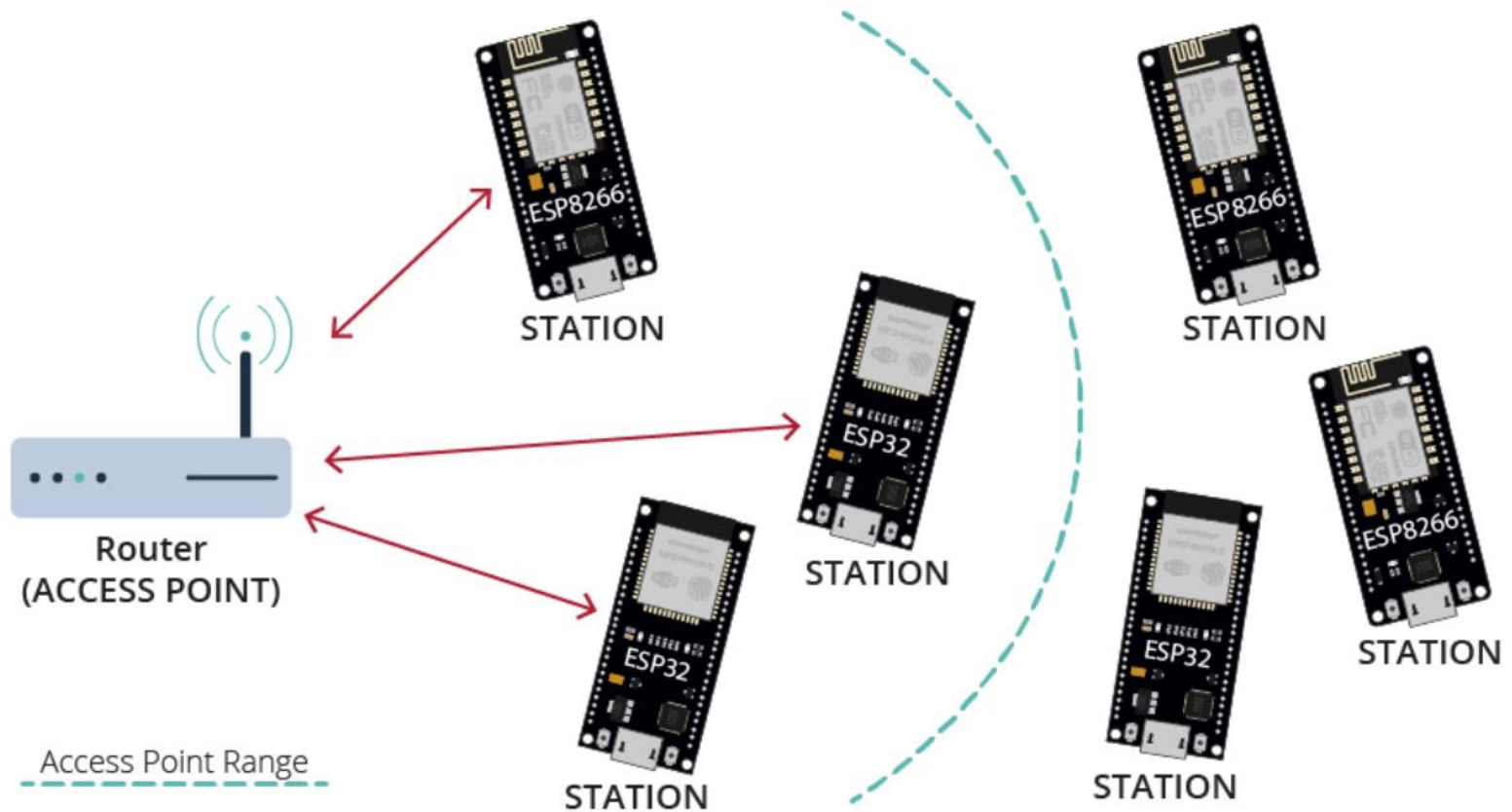
Vantaggi

- **Isolamento:** i guasti sono localizzati e non compromettono il funzionamento del resto della rete
- **Manutenzione:** facile individuare e risolvere i problemi
- **Alta velocità:** larghezza di banda è dedicata a ciascun dispositivo collegato
- **Facilità di espansione:** aggiungere o rimuovere un dispositivo non richiede modifiche alla struttura della rete

Svantaggi

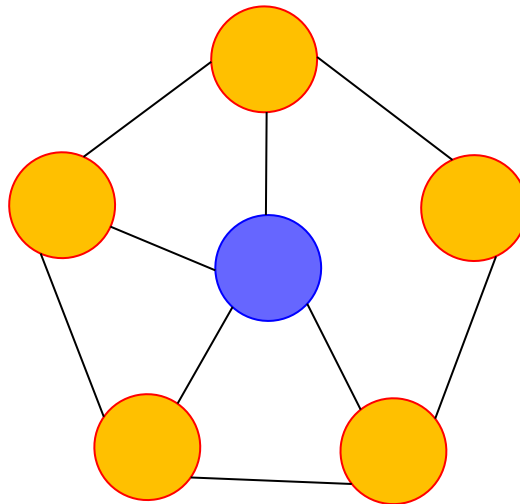
- **Singolo punto di fallimento:** la rete è totalmente dipendente dal corretto funzionamento del nodo centrale
- **Scalabilità:** le capacità del nodo centrale può diventare il collo di bottiglia della rete
- **Distanza:** la distanza tra il nodo centrale e i dispositivi può essere limitata specialmente quando si utilizzano collegamenti fisici

ESP Stella



Mesh

- Consentono la trasmissione dei dati da un nodo ad un altro, che si trova nel suo raggio di trasmissione radio
- Per inviare un messaggio ad un nodo che si trova fuori dal suo raggio di comunicazione, bisogna utilizzare nodi intermediari



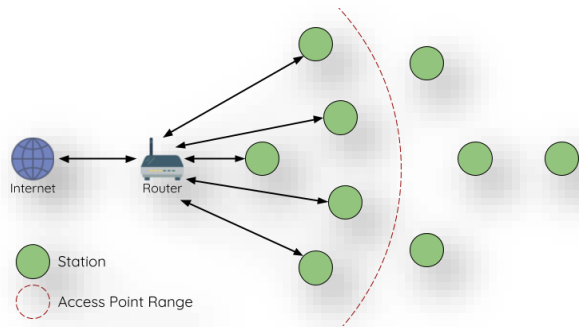
Vantaggi

- **Affidabilità:** la ridondanza dei collegamenti garantisce che la rete continui a funzionare anche in caso di guasti. La rete non è soggetta a singoli punti di fallimento
- **Tolleranza ai guasti:** i guasti di un singolo nodo non influenzano la comunicazione tra gli altri dispositivi
- **Facilità di espansione:** aggiungere o rimuovere un dispositivo non richiede modifiche alla struttura della rete

Svantaggi

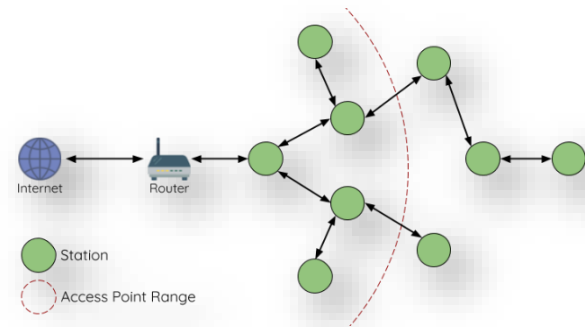
- **Costo elevato:** richiede un numero elevato di connessioni e dispositivi, rendendola più costosa da implementare
- **Complessità:** la gestione e il monitoraggio può essere complesso per via del gran numero di connessioni di gestire
- **Consumo energetico:** a causa della necessità di mantenere attive molte connessioni, una rete mesh può consumare più energia rispetto ad altre topologie

Reti Mesh



Architettura di rete Wi-Fi tradizionale

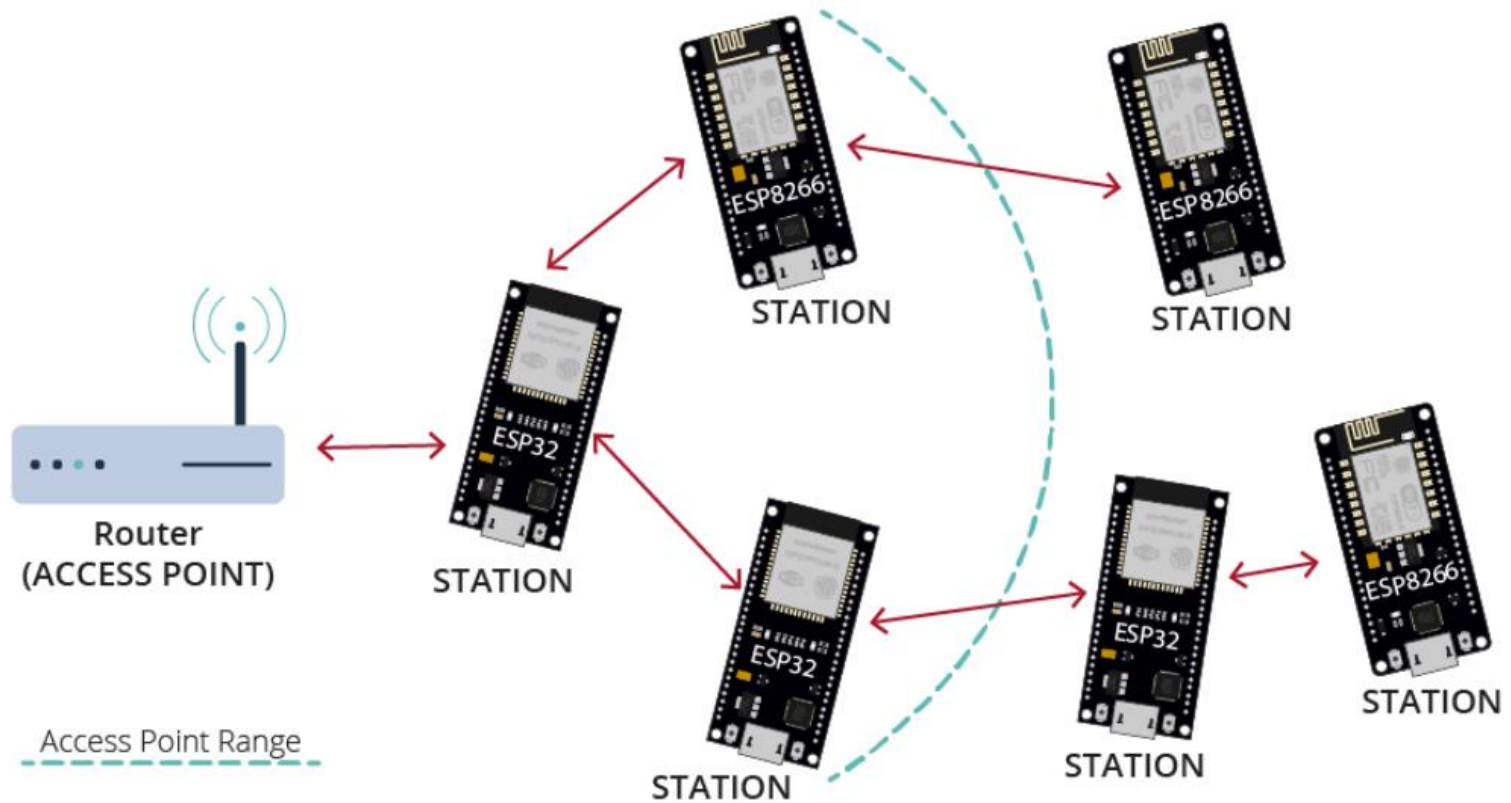
- In un'architettura di rete Wi-Fi tradizionale, un singolo nodo (punto di accesso) è collegato a tutti gli altri nodi (stazioni)
- Ogni nodo può comunicare tra loro utilizzando il punto di accesso; tuttavia, è limitato alla copertura Wi-Fi del punto di accesso
- Ogni stazione deve essere nel raggio del punto di accesso per connettersi direttamente



Architettura di rete mesh

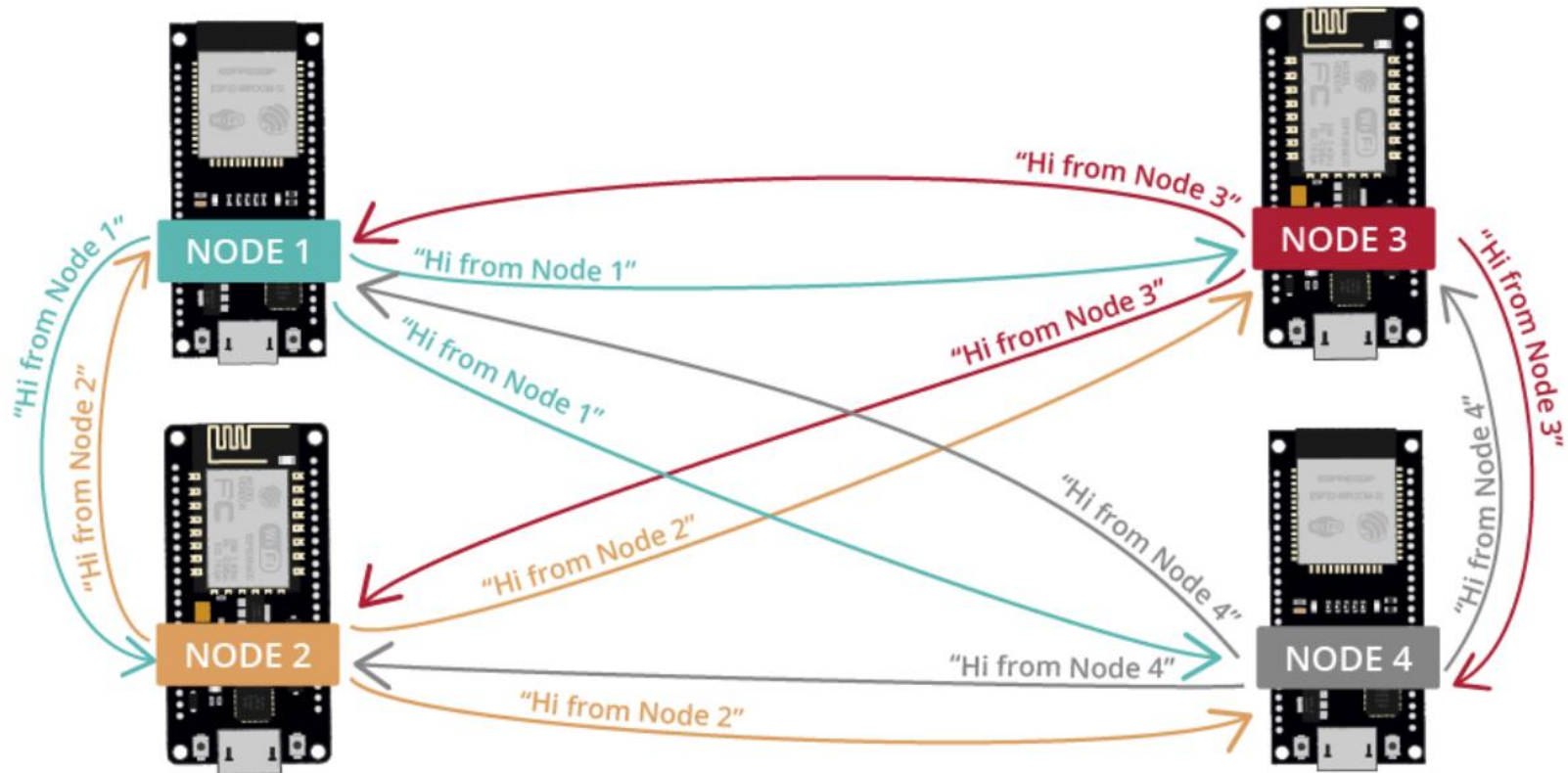
- I nodi non hanno bisogno di connettersi ad un nodo centrale (nessuna gerarchia)
- I nodi sono responsabili della trasmissione a vicenda delle trasmissioni; ciò consente a più dispositivi di diffondersi su una vasta area fisica
- I nodi possono auto-organizzarsi e parlare dinamicamente tra loro; se un nodo viene rimosso dalla rete, è in grado di auto-organizzarsi per assicurarsi che i pacchetti raggiungano la destinazione

ESP Mesh



Comunicazione Broadcast

- **Tutorial:** <https://randomnerdtutorials.com/esp-mesh-esp32-esp8266-painlessmesh/>

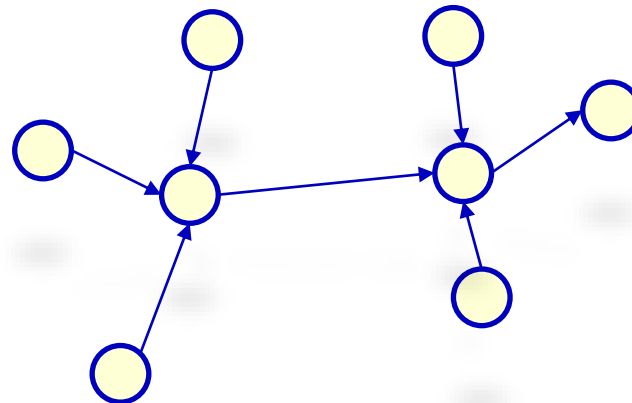


Libreria `painlessMesh`

- **Istallazione:** Tools > Manage > Libraries > `painlessMesh`
- `painlessMesh` è una libreria che consente di creare una semplice rete mesh utilizzando `esp8266` e `esp32`
- Non richiede nodi centrali o router. Ogni sistema di 1 o più nodi si auto-organizza in una mesh
- I messaggi sono trasmessi come JSON
- Per questioni di prestazione, non utilizza le librerie WiFi Arduino

Protocollo

- Ogni nodo funge da **punto di accesso** per altri nodi a cui connettersi e come client per connettersi a un punto di accesso di un altro nodo
- Tutti i nodi conoscono immediatamente la **topologia completa**, la riconfigurazione è automatica
- Qualsiasi nodo può essere disconnesso o integrato in qualsiasi momento



Protocollo

- Per i singoli messaggi viene specificato un nodo di destinazione e origine, cercando un percorso diretto verso il nodo
- Quando non è possibile il messaggio è inoltrato alla connessione successiva, che viene cercata nello stesso modo.



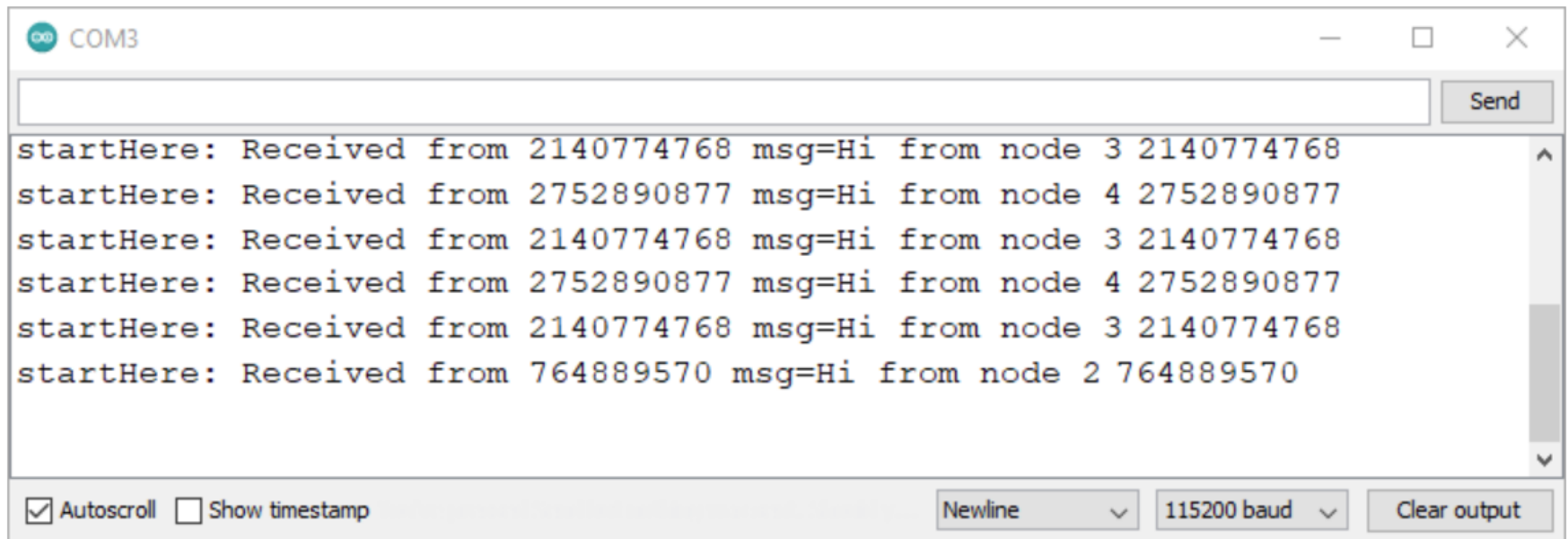
Comunicazione Broadcast - Sketch

```
1  #include "painlessMesh.h"
2
3  #define MESH_PREFIX    "whateverYouLike"
4  #define MESH_PASSWORD  "somethingSneaky"
5  #define MESH_PORT      5555
6
7  Scheduler userScheduler; // to control your personal task
8  painlessMesh mesh;
9
10 // User stub
11 void sendMessage() ; // Prototype so PlatformIO doesn't complain
12
13 Task taskSendMessage( TASK_SECOND * 1 , TASK_FOREVER, &sendMessage );
14
15 void sendMessage() {
16     String msg = "Hi from node1";
17     msg += mesh.getNodeId();
18     mesh.sendBroadcast( msg );
19     taskSendMessage.setInterval( random( TASK_SECOND * 1, TASK_SECOND * 5 ));
20 }
21
22 // Needed for painless library
23 void receivedCallback( uint32_t from, String &msg ) {
24     Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
25 }
26
27 void newConnectionCallback(uint32_t nodeId) {
28     Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
29 }
30
31 void changedConnectionCallback() {
32     Serial.printf("Changed connections\n");
33 }
34
35 void nodeTimeAdjustedCallback(int32_t offset) {
36     Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(),offset);
37 }
```


Comunicazione Broadcast - Sketch

```
39 void setup() {
40     Serial.begin(115200);
41
42     //mesh.setDebugMsgTypes( ERROR | MESH_STATUS | CONNECTION | SYNC | COMMUNICATION | GENERAL | MSG_TYPES | REMOTE ); // all types on
43     mesh.setDebugMsgTypes( ERROR | STARTUP ); // set before init() so that you can see startup messages
44
45     mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
46     mesh.onReceive(&receivedCallback);
47     mesh.onNewConnection(&newConnectionCallback);
48     mesh.onChangedConnections(&changedConnectionCallback);
49     mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
50
51     userScheduler.addTask( taskSendMessage );
52     taskSendMessage.enable();
53 }
54
55 void loop() {
56     // it will run the user scheduler as well
57     mesh.update();
58 }
```

Comunicazione Broadcast - Output



The screenshot shows a serial communication window titled "COM3". It features a text input field at the top with a "Send" button. The main area displays a log of received messages. The messages are as follows:

```
startHere: Received from 2140774768 msg=Hi from node 3 2140774768
startHere: Received from 2752890877 msg=Hi from node 4 2752890877
startHere: Received from 2140774768 msg=Hi from node 3 2140774768
startHere: Received from 2752890877 msg=Hi from node 4 2752890877
startHere: Received from 2140774768 msg=Hi from node 3 2140774768
startHere: Received from 764889570 msg=Hi from node 2 764889570
```

At the bottom, there are controls for "Autoscroll" (checked), "Show timestamp" (unchecked), a "Newline" dropdown menu, a "115200 baud" dropdown menu, and a "Clear output" button.

