

ELABORATO DI LAUREA IN
INGEGNERIA DELL'AUTOMAZIONE

Wireless gesture recognition sensors based on MQTT protocol

Relatore
Chiar.ma Prof.ssa
Annalisa Liccardo

Candidato
Antonio Maiello
N39/1500

Correlatore
Ing. Federico Gargiulo

Anno Accademico 2023/2024

Indice

CAPITOLO 1: Contesto interazione uomo macchina	3
1.2: Struttura della tesi.....	6
Capitolo 2: Gesture recognition	7
2.1: Gestii umani.....	7
2.2 Industria 5.0 e il riconoscimento gestuale	8
2.3 Sensore IR Gesture	9
2.4: ADP2140.....	10
Capitolo 3: Protocollo MQTT	12
3.1: Internet of Things (IoT)	12
3.2: Protocollo MQTT.....	12
3.3: Wifi Esp Click	15
3.3.1 AT Command.....	17
CAPITOLO 4: Architettura Nucleo-WiFi Esp Click-MQTT-CubeMonitor	19
4.1 Nucleo STM32F401RE	19
4.2 Arduino Uno click shield	20
4.3 Hardware.....	20
4.4 STM32Cube	22
4.4.1 STM32CubeIDE.....	22
4.4.2 STM32CubeMonitor	22
Capitolo 5: implementazione software	24
5.1: Codici WiFi Esp Click.....	25
5.2: Codici Sensore IR.....	30
5.3: Visualizzazione dei risultati	32
CAPITOLO 6: Conclusioni e sviluppi futuri	34
Bibliografia	35

CAPITOLO 1: Contesto interazione uomo macchina

L'epoca in cui viviamo è caratterizzata da delle trasformazioni radicali dell'industria, note come rivoluzioni industriali. Il passaggio dall'industria 4.0, dove si è enfatizzata la connettività e l'automazione grazie all'Internet of Things (IoT), a quella 5.0 che ha come obiettivo quello di creare un equilibrio tra macchine e operatori umani, promuovendo una collaborazione simbiotica che valorizzi capacità uniche degli esseri umani e le potenzialità delle tecnologie avanzate.

La presente tesi esplora le dinamiche che definiscono questa nuova era industriale, esaminando come l'interazione tra l'uomo e la macchina si sta evolvendo verso una partnership più equilibrata, dove l'intelligenza artificiale, la robotica collaborativa e altre tecnologie emergenti collaborano con l'uomo per creare ambienti di lavoro più personalizzati, flessibili e sostenibili.

Nell'industria 4.0 abbiamo una trasformazione significativa del panorama industriale, causata dall'integrazione di tecnologie digitali, che riguarda anche una trasformazione culturale e organizzativa delle aziende, personalizzando così i processi produttivi per adattarsi alle esigenze dei clienti. Fino a questo momento, si è sempre visto il lavoro di macchine e umani come interscambiabili: possono svolgere le stesse funzioni ma con prestazioni e tempi diversi, facendo nascere così la convinzione che in una total smart factory non ci fosse bisogno dell'intervento umano.

Con l'avvento dell'industria 5.0 questo concetto cambia, infatti essa punta a creare un processo di produzione inclusivo, intelligente e sostenibile che promuova la creatività e le competenze umane attraverso l'uso di automazione avanzata e intelligenza artificiale. La robotica collaborativa, o "cobotica", rappresenta una tecnologia chiave con l'obiettivo di migliorare la destrezza umana facendo dei robot delle estensioni delle capacità umane e, in definitiva, dei membri del team.

I robot hanno svolto un ruolo sempre più importante nelle rivoluzioni industriali, dall'automazione iniziale nell'Industria 3.0 ai sistemi interconnessi che permettono team collaborativi uomo-robot nell'Industria 4.0. Mentre l'Industria 4.0 ha introdotto la collaborazione uomo-robot sulle linee di produzione, l'Industria 5.0 mira a portare questo concetto oltre, concentrandosi su processi incentrati sull'uomo che promuovano resilienza, sostenibilità e una più stretta simbiosi tra lavoratori umani e sistemi robotici.

Mentre le fasi precedenti dell'evoluzione industriale si concentravano pesantemente sull'uso dei robot per massimizzare la produttività e la precisione, l'Industria 5.0 ha un duplice obiettivo: la convivenza vantaggiosa e la prosperità condivisa. Questo non implica solo il favorire

il lavoro di squadra tra esseri umani e macchine, ma anche dare priorità alla sicurezza dei lavoratori, allo sviluppo delle competenze, al potenziale creativo e al benessere complessivo, oltre ai guadagni di efficienza. Gli obiettivi fondamentali dell'Industria 5.0 includono la condivisione fluida delle informazioni tra uomini e robot, sistemi di produzione adattativi che possano rapidamente reagire ai cambiamenti o ai guasti, e l'utilizzo di AI e automazione per aumentare le capacità umane piuttosto che sostituire i posti di lavoro. [1]

L'Industria 5.0 rappresenta una nuova era industriale che si distingue per il suo approccio centrato sull'uomo, sostenibile e resiliente. Questi tre pilastri sono fondamentali per comprendere la visione e gli obiettivi di questa rivoluzione industriale.

1. Incentrato sull'Uomo

Il pilastro incentrato sull'uomo si basa sull'idea che le tecnologie avanzate e i processi produttivi debbano servire a migliorare la qualità della vita e il benessere dei lavoratori. Questo principio implica un cambiamento di paradigma rispetto alle precedenti rivoluzioni industriali, dove l'accento era posto sull'automazione e l'efficienza a scapito spesso dei bisogni umani. I principali aspetti di questo pilastro sono:

- **Benessere dei Lavoratori:** L'Industria 5.0 si impegna a creare ambienti di lavoro che promuovano la salute fisica e mentale dei dipendenti. Questo include la riduzione dei rischi lavorativi, l'ergonomia delle postazioni di lavoro e il supporto per il benessere psicologico.
- **Collaborazione Uomo-Macchina:** Invece di vedere le macchine come sostituti degli esseri umani, l'Industria 5.0 punta a una collaborazione simbiotica. I robot collaborativi, o "cobot", lavorano a fianco degli esseri umani, ampliando le loro capacità e migliorando la produttività senza eliminare posti di lavoro.
- **Autonomia e Dignità:** Le tecnologie devono rispettare e promuovere i diritti fondamentali dei lavoratori, inclusi la privacy, l'autonomia e la dignità. Questo significa sviluppare strumenti che supportino e non controllino eccessivamente i dipendenti, valorizzando il loro contributo unico.

2. Sostenibilità

La sostenibilità è al centro della visione dell'Industria 5.0, riconoscendo che la produzione industriale deve essere ecologicamente responsabile e rispettare i limiti planetari. Questo pilastro comprende diverse strategie e pratiche volte a minimizzare l'impatto ambientale:

- **Efficienza delle Risorse:** L'ottimizzazione dell'uso delle risorse naturali è essenziale. Questo implica l'adozione di tecnologie che riducono gli sprechi, aumentano l'efficienza energetica e promuovono il riciclo e il riuso dei materiali.
- **Energia Pulita:** Integrando fonti di energia rinnovabile come l'energia solare, eolica e idroelettrica nei processi produttivi,

l'Industria 5.0 mira a ridurre le emissioni di carbonio e l'impatto ambientale complessivo.

- Produzione a Basso Impatto: L'uso di materiali sostenibili e tecniche di produzione ecocompatibili è incoraggiato per ridurre l'inquinamento e conservare le risorse naturali. Questo include anche lo sviluppo di prodotti progettati per essere più facilmente riciclabili o biodegradabili.

3. Resilienza

Il pilastro della resilienza si concentra sulla capacità dei sistemi produttivi di adattarsi rapidamente e resistere a shock esterni come crisi economiche, pandemie o disastri naturali. La resilienza è cruciale per garantire la continuità e l'affidabilità delle operazioni industriali:

- Adattabilità: I sistemi di produzione devono essere flessibili e riconfigurabili per rispondere a nuove esigenze o situazioni di emergenza. Questo può includere la modularità delle linee di produzione e l'uso di tecnologie che permettono una rapida riconversione delle capacità produttive.
- Continuità Operativa: Le strategie di gestione del rischio e la pianificazione della continuità operativa sono fondamentali. Questo implica l'implementazione di soluzioni tecnologiche che assicurino la robustezza delle infrastrutture e la capacità di recupero rapido in caso di interruzioni.
- Innovazione e Miglioramento Continuo: La resilienza si basa anche su una cultura di innovazione continua e miglioramento. Le aziende devono essere pronte a adottare nuove tecnologie e approcci per mantenere la competitività e rispondere efficacemente ai cambiamenti del mercato.

Questo approccio contrasta con il principio incentrato sulla macchina o sulla piena automazione delle passate rivoluzioni industriali, dove la motivazione principale è quella di raggiungere la produzione di massa, sottovalutando quindi i costi planetari e umani. Il principio incentrato sull'uomo mira a rispettare il ruolo, i talenti e i diritti degli esseri umani ponendo il loro benessere generale allo stesso livello di importanza dell'ottimizzazione dei processi industriali. Questo principio propone l'introduzione di tecnologie e strumenti in grado di potenziare e promuovere i talenti e la diversità dei lavoratori industriali. I sistemi sviluppati con queste tecnologie devono anche salvaguardare i diritti umani fondamentali (ad esempio, autonomia, dignità e privacy), creare ambienti di lavoro inclusivi, dare priorità alla salute mentale e fisica umana e migliorare l'efficienza, la sicurezza e la soddisfazione del lavoro.

Il principio sostenibile si concentra sulla creazione di processi di produzione in grado di rispettare i confini planetari attraverso il riutilizzo e il riciclaggio delle risorse naturali, nonché la riduzione dei rifiuti industriali [2].

Le HMI, o interfacce uomo-macchina, sono sistemi di interazione che permettono agli operatori umani di comunicare e controllare le macchine e i processi automatizzati. Nell'era dell'Industria 5.0, le HMI giocano un ruolo cruciale nel realizzare una cooperazione efficiente e intuitiva tra umani e sistemi robotici avanzati. Le HMI sono progettate per essere user-friendly, flessibili e adattative, migliorando così l'efficienza, la sicurezza e il benessere degli operatori.

Le HMI giocano un ruolo cruciale facilitando tale interazione, consentendo agli operatori di monitorare e controllare i processi industriali in modo intuitivo ed efficiente, hanno avuto un notevole sviluppo, passando da semplici pulsanti a interfacce grafiche avanzate e, recentemente, a interazioni basate su comandi vocali e gestuali. L'HMI, nell'industria 5.0, diventa ancora più sviluppata sfruttando tecnologie come la realtà aumentata, la realtà virtuale e i sistemi di riconoscimento gestuale per migliorare l'esperienza dell'utente e aumentare l'efficienza operativa.

Le moderne HMI permettono una maggiore personalizzazione delle interfacce, adattandosi alle specifiche esigenze dell'operatore e del processo produttivo. Ciò non solo migliora la user experience, ma ottimizza anche l'efficienza complessiva dei sistemi industriali.

Il riconoscimento gestuale, il fulcro del nostro studio, rappresenta una delle tecnologie più avanzate e innovative nel campo delle HMI, permettendo un'interazione naturale e senza contatto tra l'operatore e il sistema.

Dopo questa introduzione, che ha fornito uno sguardo alle opportunità e implicazioni di questa nuova industria, ci si propone di analizzare le tecnologie abilitanti con particolare attenzione al riconoscimento gestuale, descrivendo il ruolo cruciale di quest'ultimo nell'ottimizzazione delle interazioni uomo macchina e il collegamento di esso con un broker esterno per la visualizzazione dei risultati.

1.2: Struttura della tesi

Dopo questa introduzione, il secondo capitolo si concentra sul riconoscimento gestuale e il sensore utilizzato, facendone una panoramica sul funzionamento. Il terzo capitolo descrive il protocollo utilizzato per la comunicazione e connessione tra il microcontrollore, che processa ed elabora le informazioni rilevate dal sensore, e il modulo che le trasmette al broker per renderle accessibili agli utenti in remoto. Il quarto capitolo indica gli strumenti impiegati per condurre gli studi, fornendo con dettaglio gli aspetti di tali strumenti.

Nel quinto capitolo, invece, ci si concentra sull'aspetto software del progetto, andando a spiegare l'ambiente di sviluppo e i codici realizzati.

Nel sesto capitolo traiamo delle conclusioni sullo studio intrapreso e sugli sviluppi futuri.

Capitolo 2: Gesture recognition

2.1: Gestì umani

L'interazione tra uomo e computer è una disciplina volta alla valutazione, progettazione e realizzazione di sistemi informatici interattivi che instaurano una comunicazione tra una o più persone, con una o più macchine computazionali. Questo campo di ricerca, si occupa di interpretare i compiti impartiti da esseri umani alle macchine, enfatizzando anche la struttura comunicativa che sussiste tra loro. Di conseguenza, è quindi necessario creare interfacce che siano le più naturali possibili dal punto di vista dell'uomo.

Dal lato macchina sono rilevanti tecniche di computer grafica, sistemi operativi, linguaggi di programmazione e ambienti di sviluppo. Dal lato umano entrano in gioco la teoria della comunicazione, le discipline del design industriale, della grafica, della linguistica, delle scienze sociali e della psicologia cognitiva. In questa evoluzione le interfacce si sono continuamente spostate da un punto di vista centrato attorno al computer ad un punto di vista centrato sull' uomo.

L'utilizzo di comandi gestuali per interagire con i sistemi rende l'approccio, da parte dell'utente, più naturale. Sistemi che riescono a catturare i movimenti di particolari arti del corpo sono espressivamente più efficaci. Da un punto di vista cognitivo, l'utente diventa, in prima persona, il controller di input, non dipendendo così da dispositivi intermedi.

I gesti fanno parte del linguaggio del corpo e veicolano una comunicazione non verbale, partendo dalle specifiche di dispositivo, si devono prendere in considerazione solamente dei set di gesti interpretabili dalla specifica macchina. Ovviamente questi gesti devono appartenere a specifiche convenzioni etniche-sociali.

Ogni sequenza di gesti è rappresentata da posizioni iniziali, che hanno lo scopo di far partire un comando, seguito poi da una sequenza di posizioni assunte dall'arto sino ad arrivare ad una posizione finale che determina la fine della sequenza di comandi.

Il riconoscimento di gesti (gesture recognition) si pone l'obiettivo di interpretare e discriminare diverse tipologie di gesti umani attraverso algoritmi. Ci sono innumerevoli gesti che, presi singolarmente o combinati tra loro, possono essere generati da qualsiasi parte del corpo. Interpretare questi movimenti è di fondamentale importanza nel progettare un'interfaccia efficace ed efficiente di interazione uomo-macchina, creando così una comunicazione più naturale, senza l'ausilio di particolare hardware d'acquisizione.

La gesture recognition trova applicazione in diversi settori, tra cui l'industria, la robotica, il campo biomedico, automotive, ma anche nell'ambito dei videogiochi.

Fornendo una panoramica dei diversi campi applicativi si rende ancora più evidente il potenziale che questa tecnologia ha in un contesto così evoluto tecnologicamente [3].

2.2 Industria 5.0 e il riconoscimento gestuale

Il riconoscimento gestuale è una componente cruciale delle moderne interfacce uomo-macchina nell'Industria 5.0. Esso migliora significativamente l'interazione tra operatori e macchine, rendendo i processi industriali più efficienti, sicuri e intuitivi.

Grazie a questa tecnologia, le macchine possono trasformare in un linguaggio a loro comprensibile il movimento delle mani o del corpo e quindi essa è estremamente utile per migliorare l'interazione tra gli utenti e i dispositivi migliorando i processi industriali.

Essendo una tecnologia versatile, il riconoscimento gestuale trova applicazione in molti campi come l'automotive, in cui i conducenti possono utilizzare gesti per controllare il sistema senza distogliere lo sguardo dal veicolo, oppure l'automazione industriale in cui gli operatori sono in grado di controllare macchine utensili, robot e altre apparecchiature industriali attraverso i gesti.

La classificazione delle tecnologie di riconoscimento gestuale può essere suddivisa in due principali categorie: dispositivi contact-based e dispositivi vision-based. Ciascuna di queste categorie utilizza approcci diversi per rilevare e interpretare i movimenti delle mani e del corpo, offrendo differenti vantaggi e sfide.

- I dispositivi contact-based richiedono un contatto fisico diretto tra l'utente e il dispositivo per rilevare i gesti, sono spesso dotati di sensori integrati che possono misurare il movimento, la posizione e la pressione delle dita e delle mani come ad esempio: touch screen, guanti sensorizzati;
- I dispositivi vision-based utilizzano telecamere e algoritmi di elaborazione delle immagini per rilevare e interpretare i gesti senza necessità di contatto fisico, questi sistemi possono catturare i movimenti delle mani e del corpo a distanza.

Il sensore utilizzato nel seguente studio è vision-based e sfrutta la tecnologia a infrarossi per rilevare e interpretare i movimenti delle mani. Il sensore IR è dotato di uno o più diodi emettitori di infrarossi (LED IR) che emettono raggi IR nell'area di rilevamento del sensore, essi sono invisibili all'occhio umano ma possono essere rilevati da fotodiodi o fototransistor di cui è composto il sensore.

Quando un oggetto, come una mano, entra nell'area di rilevamento del sensore, i raggi infrarossi emessi vengono riflessi dall'oggetto. La quantità e l'angolo della luce riflessa dipendono dalla posizione, dalla distanza e dal movimento dell'oggetto.

L'elaborazione del segnale consente al sensore di identificare specifici movimenti delle mani, come spostamenti verso l'alto, verso il basso, verso sinistra, verso destra, avvicinamenti e allontanamenti. Una volta

che il sensore ha identificato un gesto, invia i dati interpretati al microcontrollore per eseguire l'azione corrispondente. Sono impiegati nel campo della gesture recognition perché sono in grado di rilevare ostacoli o movimenti in tempo reale, analogamente ai sensi visivi umani.

2.3 Sensore IR Gesture

L'IR Gesture 3 Click è un modulo compatto progettato per riconoscere e interpretare i gesti della mano utilizzando la tecnologia a infrarossi, esso è una compact add-on board di MikroElektronika realizzata con la tecnologia delle click boards, che sono delle soluzioni *plug and play* progettate per rivoluzionare il modo in cui gli utenti aggiungono nuove funzionalità alle schede di sviluppo, promettendo al programmatore un elevato risparmio di tempo, poiché può concentrarsi solo sull'idea di sviluppo senza preoccuparsi della configurazione hardware[4].



Figura 2.1: Sensore IR Gesture

IR Gesture 3 Click è basato su un circuito integrato ADPD1080 di Analog Devices, che svolge il ruolo di front-end analogico (AFE) multifunzione. Oltre al front-end analogico integrato, il chip dispone di un ADC a 14 bit, vari LED e un nucleo di temporizzazione che consente la capacità di reiezione della luce ambientale senza filtri ottici a fotodiodo.

IR Gesture 3 Click consente il riconoscimento dei gesti in due dimensioni, con un filtro ottico integrato e un netto taglio della luce visibile. Elimina la necessità di lenti esterne e preserva la gamma dinamica del sensore quando posizionato sotto la luce solare o l'illuminazione interna. Non richiede un allineamento preciso perché il suo sensore mantiene una risposta lineare entro il campo visivo angolare di $\pm 35^\circ$. Questa Click board™ è la soluzione perfetta per lo sviluppo di varie applicazioni di rilevamento dei gesti, dai gesti di scorrimento della mano (sinistra, destra, su, giù, onda), clic in aria e gesti rapidi al rilevamento di prossimità multizona [5].

Per la traduzione dei gesti in comandi sono necessarie tre funzioni:

- L'abilità di riconoscere l'inizio e la fine del gesto;
- La traccia del movimento durante il gesto;

-L'identificazione del gesto in base al movimento della mano dall'inizio alla fine.

In sostanza, il sensore rileva le variazioni nel campo infrarosso riflesso dagli oggetti, interpretando queste variazioni come gesti specifici. Il software o il microcontrollore a cui è collegato possono poi interpretare questi segnali per eseguire azioni specifiche in base alla programmazione definita.

Il dispositivo comprende diversi moduli integrati on-board per il rilevamento e l'interpretazione dei gesti. Questi includono:

- ADPD1080: un front-end fotometrico sviluppato da Analog Devices;
- ADPD2140: un sensore di angolo della luce infrarossa sviluppato da Analog Devices;
- SFH4249: un emettitore ad infrarossi ad alta potenza sviluppato da ams OSRAM.

L'integrazione di questi moduli consente al sensore IR Gesture 3 Click di catturare, analizzare e interpretare i gesti umani nell'ambiente circostante, facilitando l'interazione con dispositivi o sistemi compatibili.

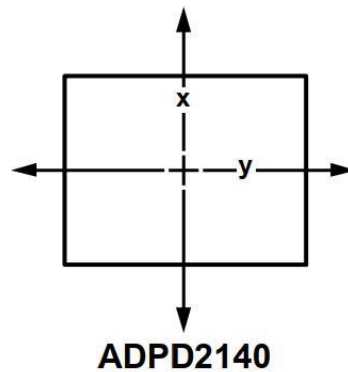


Figura 2.2: Angoli positivi nelle direzioni x e y

2.4: ADP2140

ADP2140 è un sensore ottico che misura l'angolo di incidenza della luce infrarossa. Le risposte per gli angoli calcolati da ADP2140 sono lineari fino a $\pm 5^\circ$ entro un campo visivo angolare di $\pm 35^\circ$. ADP2140 può essere utilizzato con una sorgente di luce infrarossa come un diodo ad emissione di luce (LED) per rilevare i movimenti della mano dell'utente per il riconoscimento dei gesti. È costituito da array di fotodiodi di silicio di tipo p, intrinseco e di tipo n (PIN), che forniscono una misurazione lineare dell'angolo di luce infrarossa incidente. In particolare, il dispositivo include quattro canali separati, ciascuno corrispondente a un fotodiodo [6]. ADPD2140 consente la misurazione dell'angolo di luce su 2 assi, sia nella direzione x che in quella y. Per calcolare gli angoli y rispetto al sensore, nelle direzioni x e y, bisogna utilizzare i quattro canali dei fotodiodi (x_L , x_R , y_T e y_B) e le seguenti equazioni:

$$x = (x_L - x_R) / (x_L + x_R) \quad (1)$$

$$y = (y_T - y_B) / (y_T + y_B) \quad (2)$$

Le quantità risultanti (x e y) sono rapporti relativi agli angoli attraverso un termine costante, M (0.00631 Ratio/°). Gli angoli vengono calcolati nella direzione orizzontale e verticale dividendo x e y per il termine costante, tenendo conto che gli angoli positivi nelle direzioni x e y sono quelli illustrati in figura 2.2.

ADPD2140 si impiega in combinazione con un LED o un emettitore laser e consente di misurare l'angolo della luce senza richiedere l'uso di una lente esterna. Un filtro ottico visibile integrato su ADPD2140 blocca la luce indesiderata proveniente dall'ambiente, come la luce solare o l'illuminazione interna, fornendo un'efficace reiezione dei segnali di luce visibile indesiderati.

La bassa capacità di giunzione e la bassa corrente di buio di ADPD2140 consentono un'integrazione ottimale con ADPD1080. Questa soluzione offre una completa reiezione della luce ambientale, un funzionamento a basso consumo e una conversione analogico-digitale dei segnali analogici.

Con la connessione in figura 2.3, la soluzione ADPD2140 e ADPD1080 può operare utilizzando impulsi LED sincroni per rilevare l'angolo di luce riflessa dagli oggetti o può essere utilizzata in modalità misurazione per fornire una misura dell'angolo di incidenza di una sorgente luminosa

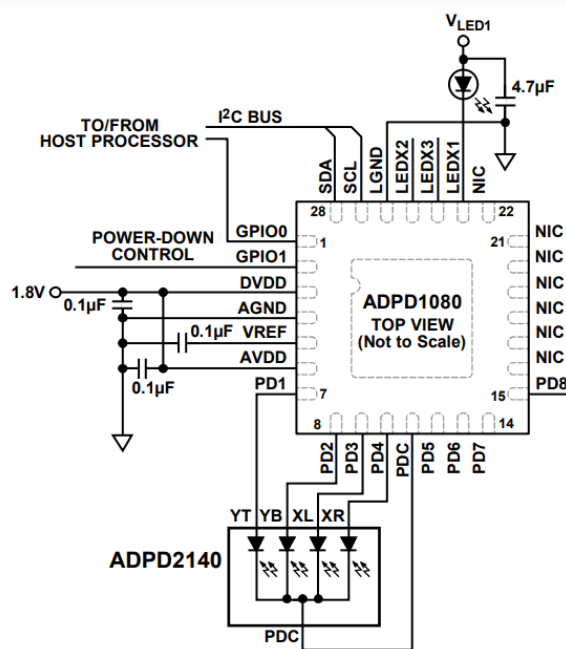


Figura 2.3: Tipico diagramma di connessione per ADPD2140 e ADPD1080

Capitolo 3: Protocollo MQTT

3.1: Internet of Things (IoT)

L'IoT rappresenta una delle innovazioni tecnologiche più significative dell'era digitale. Esso riguarda la connessione di oggetti fisici, noti come "cose", a internet, permettendo loro di raccogliere, inviare e ricevere dati.

L'IoT riguarda, soprattutto, oggetti quotidiani di cui siamo circondati. Ad esempio, quelli all'interno delle case, sul luogo di lavoro, nelle città e che ci accompagnano nella vita di tutti i giorni. L'Internet of Things nasce proprio con questo scopo: ovvero, dall'idea di portare nel mondo digitale gli oggetti della nostra esperienza quotidiana, creando un ecosistema interconnesso che migliora l'efficienza, l'automazione e la qualità della vita.

Esso si basa su una serie di pilastri fondamentali:

- 1) Dispositivi e sensori: qualsiasi dispositivo fisico in grado di raccogliere dati dall'ambiente e trasmetterli. Essi sono essenziali per la raccolta di dati accurati e in tempo reale, che alimentano le applicazioni IoT;
- 2) Connettività: riguarda i mezzi e i protocolli attraverso cui i dispositivi IoT tra loro e altri sistemi, una connessione stabile e affidabile è cruciale per il funzionamento continuo dei dispositivi e per la trasmissione dei dati raccolti;
- 3) Elaborazione dei dati e infrastruttura cloud: consentono di trasformare i dati grezzi in informazioni utili per poi memorizzarle in cloud per garantire scalabilità, flessibilità e accesso remoto ai dati e alle applicazioni IoT.
- 4) Interfacce utenti e applicazioni: permettono agli utenti finali di interagire con i sistemi IoT con la possibilità di monitorare, controllare e gestire i dispositivi in modo intuitivo.

Comprendere e ottimizzare ciascuno di questi pilastri è essenziale per realizzare appieno il potenziale trasformativo dell'Internet delle Cose, migliorando l'efficienza operativa, la qualità della vita e creando nuove opportunità di business.

3.2: Protocollo MQTT

Un componente fondamentale per il funzionamento efficiente dei sistemi IoT è il protocollo di comunicazione utilizzato per trasmettere i dati. Uno dei protocolli più diffusi in questo contesto è il MQTT (Message Queuing Telemetry Transport).

"MQTT è un protocollo di trasporto di messaggistica di pubblicazione/sottoscrizione Client Server. È leggero, aperto, semplice e progettato in modo da essere facile da implementare. Queste caratteristiche lo rendono ideale per l'uso in molte situazioni, inclusi ambienti vincolati come la comunicazione in contesti Machine to

Machine (M2M) e Internet of Things (IoT) in cui è richiesto un ingombro ridotto di codice e/o la larghezza di banda della rete è limitata. "

Citazione dalla specifica ufficiale [MQTT 3.1.1](#)

Quando MQTT viene descritto come "leggero", significa che è stato progettato per essere semplice, efficiente e non dispendioso in termini di risorse. MQTT è stato creato con l'obiettivo di inviare piccole quantità di dati su reti inaffidabili con larghezza di banda e connettività limitate. Rispetto ad altri protocolli, MQTT ha un ingombro ridotto del codice, un basso sovraccarico e un basso consumo energetico. Grazie al minimo sovraccarico dei pacchetti, MQTT eccelle nel trasferimento di dati via cavo rispetto a protocolli come HTTP. Ciò lo rende ideale anche per l'uso in dispositivi con potenza di elaborazione, memoria e durata della batteria limitate, come sensori e altri dispositivi IoT.

MQTT utilizza un formato di messaggio binario per la comunicazione tra client e broker. Ciò è in contrasto con altri protocolli che utilizzano formati basati su testo, come HTTP o SMTP.

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

Figura 3.1: formato messaggio MQTT

Il formato binario utilizzato da MQTT è progettato per ridurre la dimensione dei messaggi e aumentare l'efficienza della comunicazione. Utilizzando un formato binario, il protocollo può ridurre al minimo la quantità di dati da trasmettere e ridurre la potenza di elaborazione richiesta per interpretare i messaggi. Ciò rende MQTT particolarmente adatto per l'uso in ambienti con larghezza di banda ridotta o a basso consumo, come i dispositivi IoT con risorse limitate. Viene utilizzato anche nei sistemi aziendali, dove è necessaria la comunicazione dei dati in tempo reale.

Un altro aspetto importante del protocollo è che MQTT è estremamente semplice da implementare sul lato client. La facilità d'uso è stata una preoccupazione chiave nello sviluppo di MQTT e questo lo rende perfetto per dispositivi vincolati con risorse limitate.

La trasmissione dati TCP/IP, sul quale il protocollo è formato, coinvolge due componenti principali: il broker e il client. Il broker è il cuore di un sistema MQTT. Si tratta di un server che riceve tutti i messaggi pubblicati dai client e li distribuisce ai client iscritti ai rispettivi topic. Il broker gestisce la comunicazione tra i client, assicurando che i messaggi siano consegnati correttamente e in modo efficiente, può memorizzare i messaggi pubblicati con determinati livelli di QoS (Qualità di servizio).

Ecco la ripartizione di ciascun livello:

- QoS 0: questo livello fornisce la consegna "al massimo una volta", in cui i messaggi vengono inviati senza conferma e potrebbero andare persi. Questo è il livello più basso di QoS e viene generalmente

utilizzato in situazioni in cui la perdita del messaggio è accettabile o in cui il messaggio non è critico. Ad esempio, QoS 0 potrebbe essere appropriato per l'invio di dati del sensore in cui la perdita occasionale di dati non avrebbe un impatto significativo sui risultati complessivi.

- QoS 1: questo livello prevede la consegna "almeno una volta", in cui i messaggi vengono confermati e rispediti se necessario. Con QoS 1 l'editore invia il messaggio al broker e attende conferma prima di procedere. Se il broker non risponde entro un tempo prestabilito, l'editore invia nuovamente il messaggio. Questo livello di QoS viene in genere utilizzato in situazioni in cui la perdita dei messaggi è inaccettabile, ma la duplicazione dei messaggi è tollerabile. Ad esempio, QoS 1 potrebbe essere appropriato per inviare messaggi di comando ai dispositivi, dove un comando mancato potrebbe avere gravi conseguenze, ma i comandi duplicati no.
- QoS 2: questo livello fornisce la consegna "exactly once", in cui i messaggi vengono confermati e inviati nuovamente finché non vengono ricevuti esattamente una volta dall'abbonato. QoS 2 è il livello più alto di QoS e viene generalmente utilizzato in situazioni in cui la perdita o la duplicazione dei messaggi è completamente inaccettabile. Con QoS 2, l'editore e il broker si impegnano in un processo di conferma in due fasi, in cui il broker memorizza il messaggio finché non viene ricevuto e riconosciuto dall'abbonato. Questo livello di QoS viene generalmente utilizzato per messaggi critici come transazioni finanziarie o avvisi di emergenza.

I client MQTT sono i dispositivi o le applicazioni che si connettono al broker per inviare (publish) o ricevere (subscribe) messaggi. Un client può svolgere entrambi i ruoli: pubblicare messaggi su un topic e iscriversi per ricevere messaggi su uno o più topic.

Uno degli elementi chiave che rende il protocollo MQTT efficace e flessibile per le applicazioni IoT è il suo sistema di comunicazione basato su topic, essi sono fondamentali nel modello di comunicazione publish-subscribe utilizzato da MQTT.

Un topic è una stringa gerarchica che rappresenta un canale di comunicazione utilizzati dal client per pubblicare messaggi e per sottoscrivere a messaggi di interesse.

I topic in MQTT sono gerarchici, il che significa che possono essere strutturati in livelli per rappresentare categorie o sottocategorie. Ad esempio, home/livingroom/lights potrebbe rappresentare le luci del soggiorno di una casa, mentre home/bedroom/temperature potrebbe rappresentare la temperatura della camera da letto.

La comunicazione broker-client utilizza meccanismi come username/password, certificati SSL/TLS o autenticazione basata su token che sono la base della sicurezza di tale protocollo che

ovviamente risulta influenzata da una buona configurazione e implementazione [7].

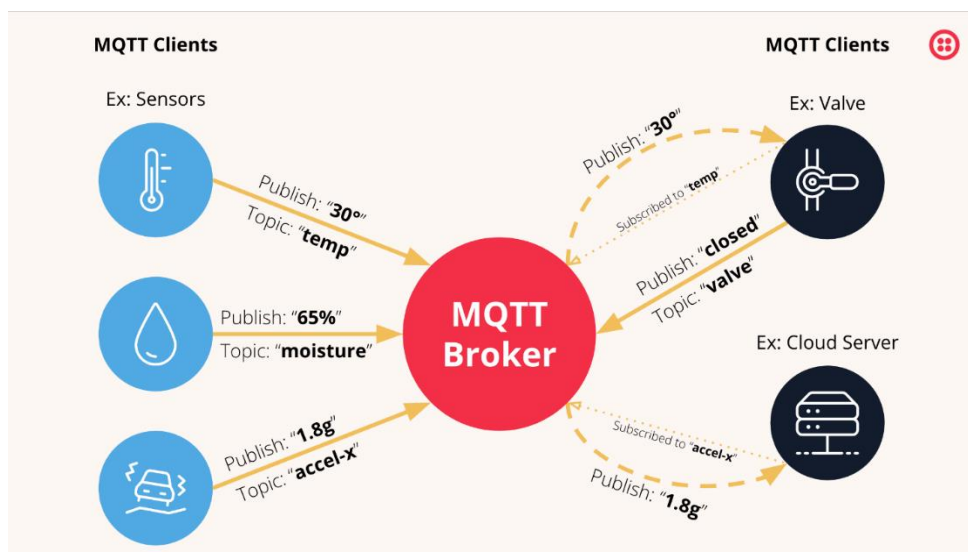


Figura 3.2: Esempio di rete Broker-Clients MQTT

3.3: Wifi Esp Click

Per la comunicazione con il broker e lo scambio di informazioni sul movimento elaborate dal microcontrollore, nel nostro studio è stato utilizzato il modulo WiFi ESP Click.



Figura 3.3: WiFi Esp Clik

WiFi ESP Click si basa su ESP-WROOM-02, un modulo WiFi completamente integrato di Espressif. Trasporta il noto cavallo di battaglia ESP8266EX, una soluzione SoC altamente integrata che soddisfa le continue richieste di utilizzo efficiente dell'energia, design compatto e prestazioni affidabili nel settore. Il modulo include anche 2 MB di flash SPI per memorizzare un programma utente.

Grazie alle funzionalità di rete WiFi complete e autonome, può funzionare come applicazione autonoma (il modulo WROOM stesso) o come slave di un host MCU, che è l'intenzione principale della click board nel suo complesso. Pertanto, questa scheda a clic viene applicata a qualsiasi progetto di microcontrollore come adattatore WiFi tramite l'interfaccia UART (linee RX, TX su presa pin mikroBUS). Il WiFi ESP Click viene fornito con 5 GPIO esposti del modulo, che fanno parte dell'interfaccia HSPI/GPIO del modulo. Il GPIO0 viene utilizzato per accedere alla modalità di download UART dell'ESP8266EX accorciandolo con GND proprio accanto ad esso. In questo modo puoi aggiornare il firmware del modulo o caricarne uno personalizzato.

Questa Click board™ può essere utilizzata solo con un livello di tensione logica di 3,3 V. La scheda deve eseguire un'appropriata conversione del livello di tensione logica prima di utilizzare MCU con livelli logici diversi. Tuttavia, la Click board™ è dotata di una libreria contenente funzioni e un codice di esempio che può essere utilizzato, come riferimento, per ulteriori sviluppi [8].

Per la comunicazione tra WiFi Esp Click e nucleo viene utilizzato il protocollo USART (Universal Synchronous/Asynchronous Receiver Transmitter) che nel microcontrollore STM32F401RE offre una potente capacità di comunicazione seriale sia sincrona che asincrona. La modalità asincrona che è stata utilizzata, comunemente nota come modalità UART, supporta la comunicazione seriale con velocità configurabili fino a diversi Mbps. È ideale per comunicazioni a breve distanza e senza un clock di riferimento condiviso.

Il protocollo generalmente è composto da un bit di start, otto bit dati, un bit di parità e qualche volta può anche contare uno o due bit di stop. La comunicazione avviene tramite due canali dedicati, uno di trasmissione ed uno di ricezione, ovviamente intrecciati tra i due dispositivi: il canale di trasmissione di un dispositivo è connesso al canale di ricezione dell'altro e viceversa. Il canale di trasmissione normalmente si trova al livello logico alto, quando un dispositivo è pronto per trasmettere abbassa il livello sulla linea di trasmissione mandando così il primo bit, ovvero il bit di start. Successivamente il dispositivo invia i dati partendo dal meno significativo, per poi concludere con i bit di parità o di stop. Il bit di parità viene utilizzato per il controllo finale del pacchetto ricevuto e si basa sul numero di 1 presenti nel pacchetto, i quali all'interno del pacchetto devono sempre essere in numero pari, così se ad esempio se tra i bit di dati ci sono cinque 1 il bit di parità sarà un altro 1. Per quanto riguarda invece i bit di stop, generalmente si possono avere tre opzioni, ovvero 1, 2 bit di stop, durante l'invio del bit di stop il trasmettitore riporta la linea nello stato di riposo. In una comunicazione seriale un bit viene definito da un tempo, ovvero stabilita la velocità della comunicazione ad esempio 115200 baud (bit al secondo o bps) un singolo bit durerà $1/115200$ secondi, circa 8.68 us,

pertanto, dal momento in cui inizia la comunicazione (nel caso asincrono) ogni 8,68 us si avrà un bit diverso nel pacchetto, in questo caso campionando ogni 8,68 us il ricevitore sarà in grado di distinguere un bit da un altro. Le caratteristiche peculiari della comunicazione spesso vengono riassunte tramite una sigla, che indica la velocità, la presenza di bit di parità e di stop, così ad esempio una sigla come 19200n1, indica una comunicazione con una velocità di 19200 baud, nessun bit di parità ed un bit di stop.

3.3.1 AT Command

Gli ESP, come l'ESP Wroom-02 su cui è basato il modulo WiFi Esp Click, sono una serie di microcontrollori Wi-Fi prodotti da Espressif Systems, utilizzati ampiamente per progetti di IoT (Internet of Things).

Per la configurazione e gestione del modulo utilizziamo il firmware AT (Attention), un tipo di firmware molto comune utilizzato con i moduli ESP. Il firmware AT è progettato per semplificare l'interfacciamento dei moduli ESP con microcontrollori esterni, come Arduino o Raspberry Pi. Invece di programmare direttamente l'ESP con il codice C/C++, si comunica con esso attraverso comandi AT: un set di comandi standardizzati per svolgere varie funzioni come connettersi a una rete WiFi, creare un punto di accesso, inviare e ricevere dati tramite TCP/IP, e altro ancora.

Il vantaggio più grande di questa opzione è che non dobbiamo avere familiarità con alcun linguaggio o framework specifico per usare il modulo. Possiamo semplicemente inviargli una serie di comandi per raggiungere il nostro obiettivo. Lo svantaggio è che abbiamo bisogno di un microcontrollore aggiuntivo coinvolto o di un adattatore USB-Seriale per inviare i comandi necessari.

Un comando AT è tipicamente composto da una sequenza di caratteri ASCII, inizia con "AT" seguito da un comando specifico e, opzionalmente, da parametri aggiuntivi. Ad esempio, "AT+CWJAP="SSID","password"" è un comando per connettersi a una rete Wi-Fi.

Ogni comando è terminato con un carattere speciale, solitamente "\r\n" (carriage return e newline), che indica al dispositivo che è stato completato un comando e che è pronto a eseguirlo.

Dopo aver ricevuto un comando AT valido, il modulo invia una risposta per confermare l'accettazione del comando. Solitamente questa risposta è "OK", che indica che il comando è stato ricevuto e sarà eseguito mentre altre risposte possono fornire informazioni dettagliate sullo stato corrente del modulo.

Gli AT Command utilizzati sono i seguenti:

- AT: Test di funzionamento base. Solitamente, risponde con "OK" se il dispositivo è operativo;
- AT+RST: Riavvia il dispositivo;
- AT+CWMODE viene utilizzato per configurare la modalità di funzionamento Wi-Fi del modulo, nel nostro caso è 1, cioè in Station Mode e il modulo si connette a un router WiFi;
- AT+CWJAP: Connettiti a una rete Wi-Fi;
- AT+CIPMUX abilita o disabilita il multiplexing delle connessioni TCP/IP sul modulo;
- AT+CIPSTART viene utilizzato per avviare una connessione TCP/IP verso un host remoto specificato che in questo caso è il mio broker mqtt.

Quindi l'obiettivo dell'uso degli AT Command è quello di connettere il modulo WiFi Esp Click ad un broker MQTT e mandare/ricevere messaggi mediante la sottoscrizione ad un topic.

Nel seguente studio i messaggi visualizzati dai client MQTT, grazie all'uso di STM32CubeMonitor, si riferiscono al riconoscimento dei vari gesti, riconosciuti mediante l'ausilio del sensore IR gesture, eseguiti da un braccio robotico.

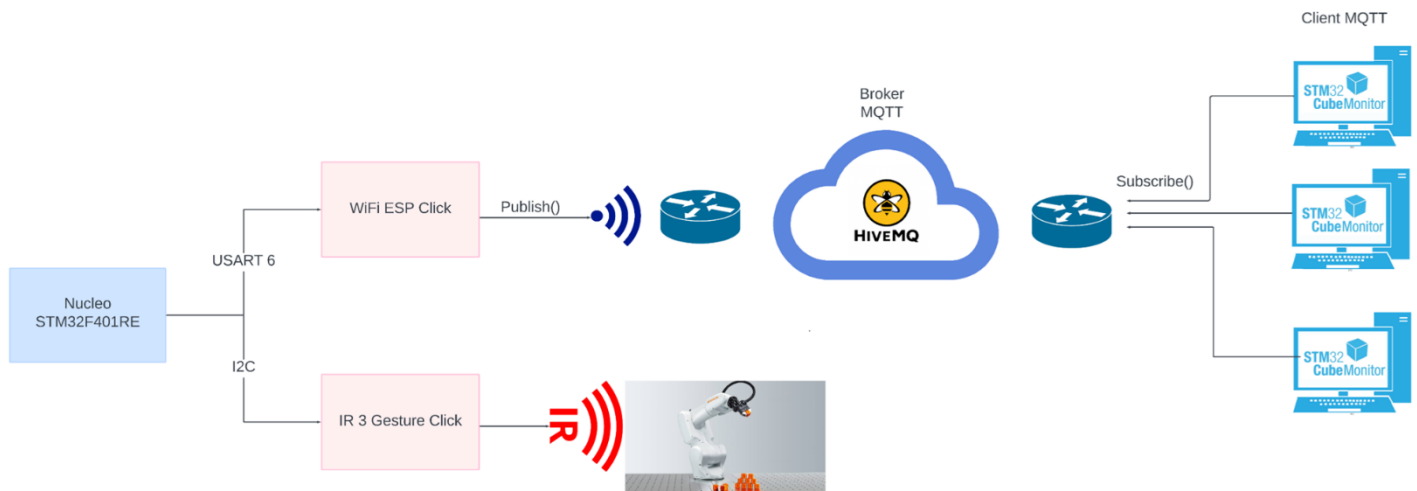


Figura 3.4: Schema a blocchi del progetto realizzato

CAPITOLO 4: Architettura Nucleo-WiFi Esp Click-MQTT-CubeMonitor

4.1 Nucleo STM32F401RE

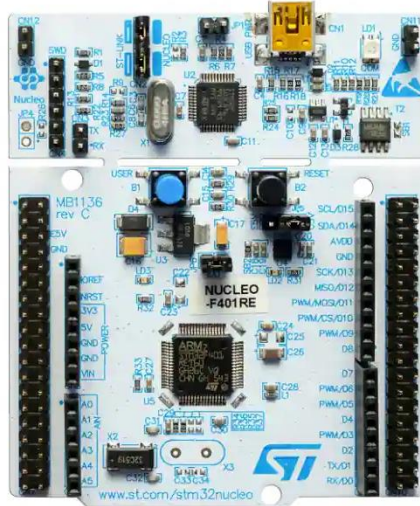


Figura 4.1: Nucleo STM32F401RE

La scheda STM32 Nucleo-64 offre agli utenti un modo conveniente e flessibile per provare nuovi concetti e costruire prototipi scegliendo tra le varie combinazioni di prestazioni e caratteristiche di consumo energetico fornite dal microcontrollore STM32. Per le schede compatibili, l'SMPS interno o esterno riduce significativamente il consumo energetico in modalità Run.

Il supporto di connettività ARDUINO® Uno V3 e gli header morpho ST consentono la facile espansione delle funzionalità della piattaforma di sviluppo aperta STM32 Nucleo con un'ampia scelta di scudi specializzati.

La scheda STM32 Nucleo-64 non richiede alcuna sonda separata in quanto integra il debugger/programmatore ST-LINK.

La scheda STM32 Nucleo-64 viene fornita con le librerie software gratuite complete STM32 e gli esempi disponibili con il pacchetto MCU STM32Cube.

Caratteristiche comuni:

- Microcontrollore STM32 in un package LQFP64 o LQFP48
- 1 LED utente condiviso con ARDUINO®
- 1 pulsante utente e 1 pulsante reset
- Oscillatore a cristallo 32.768 kHz
- Connettori della scheda:
 - Connettore di espansione ARDUINO® Uno V3
 - Intestazioni pin di estensione morpho ST per l'accesso completo a tutti gli I/O STM32
- Opzioni flessibili di alimentazione: ST-LINK USB V_{BUS} o fonti esterne

- Librerie ed esempi software gratuiti completi disponibili con il pacchetto MCU STM32Cube
- Supporto di un'ampia scelta di ambienti di sviluppo integrati (IDE), tra cui IAR Embedded Workbench®, MDK-ARM e STM32CubeIDE [9].

4.2 Arduino Uno click shield

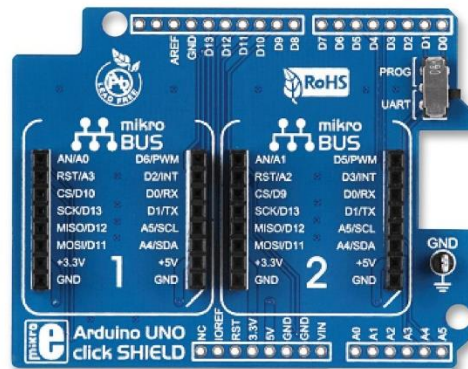


Figura 4.2: Arduino Uno Click Shield

L'Arduino Uno Click Shield è un'espansione per le schede Arduino Uno e altre che permette di collegare facilmente una vasta gamma di moduli chiamati "click boards" sviluppati da MikroElektronika. Questi moduli sono progettati per aggiungere rapidamente funzionalità specifiche al tuo progetto, come sensori, attuatori, moduli di comunicazione, display e tutti i componenti compatibili a seconda dei progetti da realizzare.

4.3 Hardware

Dopo aver descritto tutti i componenti utilizzati in questo studio è necessario analizzare come questi si interfacciano tra loro per instaurare la comunicazione. Per permettere al modulo di interfacciarsi con il microprocessore è risultato conveniente utilizzare una Click Shield Arduino Uno, per facilitare il collegamento.

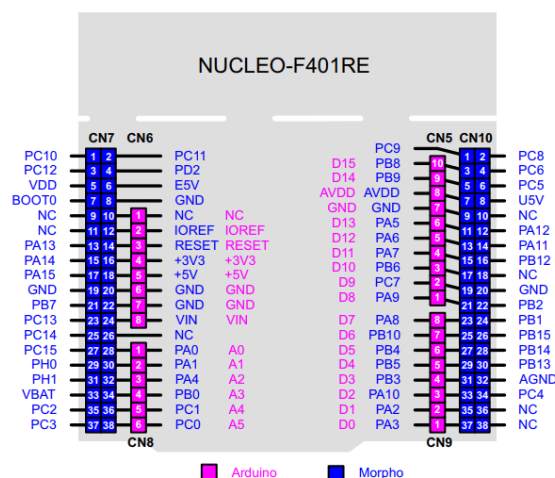


Figura 4.3: STM32 Nucleo-F401RE Pinout

Nella figura 4.3 è possibile distinguere i pin del nucleo stm32f401re e la shield Arduino ai quali sono collegati. Per la comunicazione modulo-

nucleo si è utilizzato il canale USART 6 che corrisponde ai pin PC6 e PC7 del nucleo. Oltre ai pin per la comunicazione, abbiamo anche i pin per l'alimentazione del modulo che corrispondono ai pin di enable e reset, essi corrispondono rispettivamente ai pin PB6 e PB0 del nucleo che sono stati impostati come gpio output in fase di configurazione. Per quanto riguarda il sensore IR Gesture, la comunicazione con il modulo è realizzata attraverso l'I2C (Inter-Integrated Circuit). L'I2C è un protocollo di comunicazione master/slave, in cui i dispositivi slave rispondono solo quando interrogati dal master. Poiché ci possono essere più slave sullo stesso bus, ciascuno viene identificato in modo univoco tramite un indirizzo slave specifico. Essendo un protocollo seriale, l'I2C trasferisce i dati bit per bit lungo un'unica linea (SDA) in pacchetti di 8 bit. Questi pacchetti contengono diversi componenti essenziali:

- Condizione di Start: La linea SDA passa da alta a bassa prima che la linea SCL faccia lo stesso.
- Condizione di Stop: La linea SDA passa da bassa ad alta prima che la linea SCL faccia lo stesso.
- Frame di Indirizzo: Una sequenza di 7 o 10 bit che identifica univocamente lo slave.
- Bit di Lettura/Scrittura (R/W): Indica se l'operazione richiesta è di lettura o scrittura.
- Bit di Acknowledge/Not Acknowledge (ACK/NACK): Indica se l'operazione è stata completata con successo (ACK) o meno (NACK).

Essendo un protocollo sincrono, l'I2C utilizza la linea SCL per controllare il trasferimento dei dati, sincronizzando il master e gli slave. I pin SDA e SCL sono collegati rispettivamente ai pin PB9 e PB8 del nucleo.

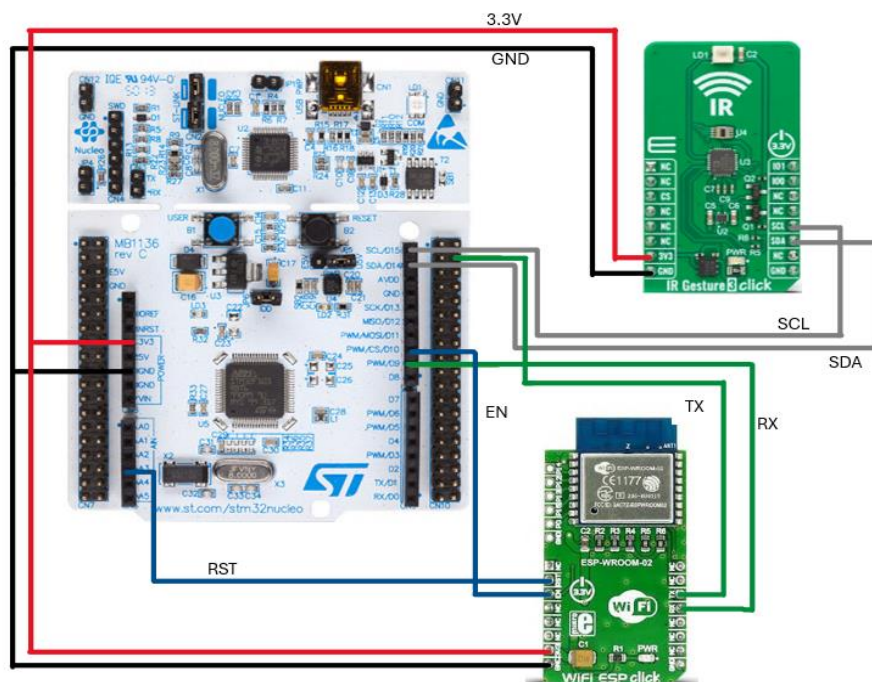


Figura 4.4: Collegamenti Nucleo-WiFi Esp Click-IR3 Gesture

4.4 STM32Cube

STM32Cube è un ecosistema software di STMicroelectronics per i microcontrollori e microprocessori STM32. Consente di migliorare la produttività dei progettisti andando a ridurre il tempo e i costi di sviluppo, includendo sia tool software, che si occupano dello sviluppo del progetto, sia librerie software. Nello specifico, in questo lavoro, verranno utilizzati STM32CubeIDE e STM32CubeMonitor [10].

4.4.1 STM32CubeIDE

STM32CubeIDE è una piattaforma di sviluppo C/C++ avanzata con funzionalità di configurazione delle periferiche, generazione di codice, compilazione di codice e debug per microcontrollori e microprocessori STM32. Si basa sul framework Eclipse® /CDT™ e sulla toolchain GCC per lo sviluppo e GDB per il debug. Permette l'integrazione delle centinaia di plugin esistenti che completano le funzionalità dell'IDE Eclipse®.

STM32CubeIDE integra le funzionalità di configurazione STM32 e di creazione di progetti di STM32CubeMX per offrire un'esperienza di strumento all-in-one e risparmiare tempo di installazione e sviluppo. Dopo la selezione di un MCU o MPU STM32 vuoto, o di un microcontrollore o microprocessore preconfigurato dalla selezione di una scheda o dalla selezione di un esempio, viene creato il progetto e generato il codice di inizializzazione. In qualsiasi momento durante lo sviluppo l'utente può ritornare all'inizializzazione e configurazione delle periferiche o del middleware e rigenerare il codice di inizializzazione senza alcun impatto sul codice utente.

STM32CubeIDE include analizzatori di build e stack che forniscono all'utente informazioni utili sullo stato del progetto e sui requisiti di memoria.

STM32CubeIDE include anche funzionalità di debug standard e avanzate, tra cui visualizzazioni dei registri del core della CPU, delle memorie e dei registri periferici, nonché controllo delle variabili in tempo reale, interfaccia Serial Wire Viewer o analizzatore di errori [11].

4.4.2 STM32CubeMonitor

La famiglia di strumenti STM32CubeMonitor aiuta a ottimizzare e diagnosticare le applicazioni STM32 in fase di esecuzione leggendo e visualizzando le relative variabili in tempo reale. Oltre alle versioni specializzate (alimentazione, RF, USB-PD), il versatile STM32CubeMonitor fornisce un editor grafico basato sul flusso per creare in modo semplice

dashboard personalizzate e aggiungere rapidamente widget come indicatori, grafici a barre e grafici. Grazie al monitoraggio non intrusivo, STM32CubeMonitor preserva il comportamento in tempo reale delle applicazioni e integra perfettamente gli strumenti di debug tradizionali per eseguire la profilazione delle applicazioni.

Con il monitoraggio remoto e il supporto nativo per display multiformato, STM32CubeMonitor consente agli utenti di monitorare le applicazioni su una rete, testare più dispositivi contemporaneamente ed eseguire la visualizzazione su vari dispositivi host come PC, tablet o telefoni cellulari. Inoltre, con il supporto diretto della comunità aperta Node-RED [®], STM32CubeMonitor consente una scelta illimitata di estensioni per affrontare un'ampia varietà di tipi di applicazioni [12].

Capitolo 5: implementazione software

Terminata la configurazione hardware si passa a quella software su STM32CubeIDE.

Il primo step è quello della scelta della scheda da utilizzare, per poi configurare i pin e i parametri.

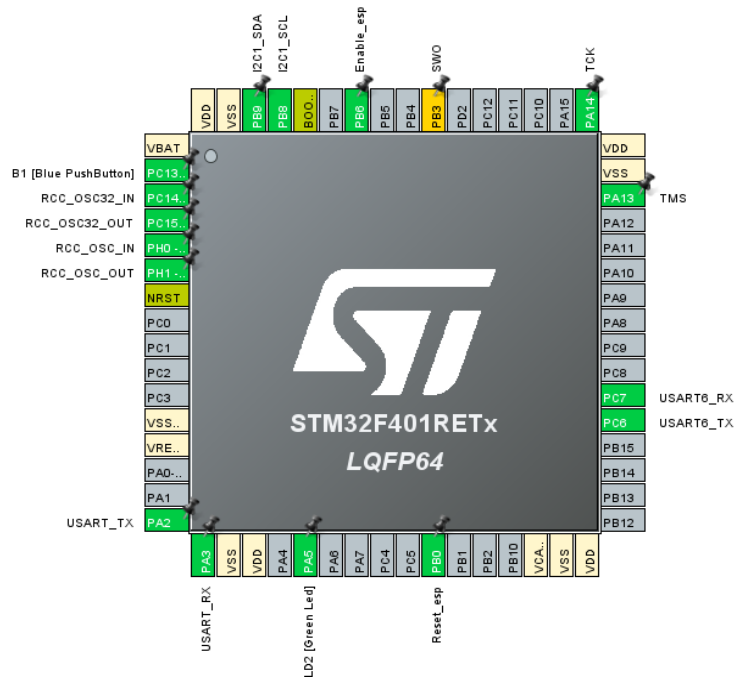


Figura 5.1: Configurazione Nucleo STM32F401RE

Come già accennato nel capitolo precedente, per il collegamento hardware sono stati utilizzati i pin PC6 e PC7 che fanno riferimento ai canali TX e RX di usart6. Per quanto riguarda i pin di Enable e Reset del modulo sono stati collegati ai pin PB6 e PB0. Come si può notare dalla figura 5.1 è attiva anche Usart2 che è stata utilizzata come canale di comunicazione con un terminale per visualizzare messaggi di risposta e di debug dal modulo wifi esp click. I parametri utilizzati per Usart6 sono i seguenti:

Mode

Mode Asynchronous

Configuration

Reset Configuration

DMA Settings

GPIO Settings

Parameter Settings

User Constants

NVIC Settings

Configure the below parameters :

Basic Parameters

Baud Rate

115200 Bits/s

Word Length

8 Bits (including Parity)

Parity

None

Stop Bits

1

Advanced Parameters

Data Direction

Receive and Transmit

Over Sampling

16 Samples

Gli stessi sono utilizzati da usart2, l'unica differenza è che per l'usart6 è

abilitata la modalità interrupt per la gestione della risposta come vedremo in seguito.

Cliccando su "Device Configuration Tool Code Generator" verrà automaticamente generato il codice del main.c, che sarà il punto di partenza di questo lavoro.

Nel main.c vengono automaticamente inizializzate le periferiche e configurato il clock secondo le scelte effettuate nella creazione del progetto, ed è possibile andare a scrivere il codice utente in una specifica posizione, delimitata da due commenti (user code begin e user code end), in modo tale che questo resti invariato anche se c'è la necessità di rigenerare il codice in seguito a modifiche di progetto.

5.1: Codici WiFi Esp Click

I file principali di cui è composto il codice sono il main.c e i file di configurazione del sensore.

Nel main.c troviamo le funzioni per l'inizializzazione e il funzionamento del modulo wifi esp click che sono:

- Wifi_Init: funzione che serve per attivare il modulo e resettarlo andando ad attivare il pin enable e di reset.

```
void WiFi_Init()
{
    HAL_GPIO_WritePin(GPIOB, Enable_esp_Pin, GPIO_PIN_SET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(GPIOB, Reset_esp_Pin, GPIO_PIN_RESET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(GPIOB, Reset_esp_Pin, GPIO_PIN_SET);
    HAL_Delay(1500);
}
```

- Blocco AT Command: come detto precedentemente gli at command sono utilizzati per la configurazione e il controllo.

In questo caso per la verifica del successo dell'operazione richiesta viene trasmesso un messaggio di debug al terminale collegato tramite uart2.

Per l'invio dei comandi al modulo è stata utilizzata la funzione send_at_command basata su HAL_UART_Transmit, una funzione della libreria HAL (Hardware Abstraction Layer) per i microcontrollori STM32, sviluppata da STMicroelectronics. Questa funzione è utilizzata per trasmettere dati via UART.

```
void send_at_command(const char *command) {
    strcpy((char *)tx_buffer, command);
    HAL_UART_Transmit(&huart6, tx_buffer, strlen(command), HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart6, (uint8_t *)"\r\n", strlen("\r\n"), HAL_MAX_DELAY);
}
```

I parametri di HAL_UART_Transmit sono quattro:

1. &huart6 è un puntatore che contiene la configurazione dell'UART, inclusi i registri e altre informazioni di stato;
2. tx_buffer è un puntatore al buffer di dati che devono essere trasmessi. I dati vengono letti da questo buffer e inviati attraverso l'UART;
3. attraverso il comando strlen della libreria string.h calcolo la dimensione del comando;
4. HAL_MAX_DELAY: questo parametro specifica il timeout massimo per la trasmissione dei dati in millisecondi. Se la trasmissione non è completata entro questo periodo di tempo, la funzione ritorna con un errore.

La funzione ritorna un valore di tipo HAL_StatusTypeDef che può segnalare il successo o meno della trasmissione.

Ogni AT Command ha un tipo di risposta precisa che viene utilizzata per verificare se il comando viene eseguito correttamente o meno:

```
do{
    response_received = 0;
    HAL_UART_Transmit(&huart6, (uint8_t *) "AT+CWJAP=\"", strlen("AT+CWJAP=\""), HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart6, (uint8_t *)SSID, strlen((char *)SSID), HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart6, (uint8_t *) "\",\"", 3, HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart6, (uint8_t *)password, strlen((char *)password), HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart6, (uint8_t *) "\"\r\n\"", 3, HAL_MAX_DELAY);
    HAL_Delay(8000);
    if (strstr((char*)rx_buffer, "WIFI GOT IP\r\n\r\nOK\r\n") != NULL) {
        HAL_UART_Transmit(&huart2, (uint8_t *) "Connesso al router\r\n", strlen("Connesso al router\r\n"), HAL_MAX_DELAY);
        response_received = 1; } // Imposta il flag della risposta attesa
}while(response_received != 1);
```

Per la verifica si è utilizzato un do-while con un if che verifica se il messaggio da parte del modulo corrisponde a quella desiderata, se è così, va avanti ed esegue il prossimo comando.

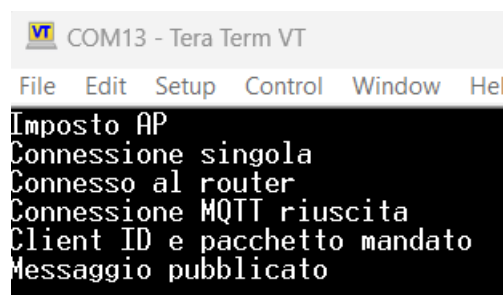


Figura 5.2: Esempio visualizzazione messaggi di debug

Il controllo del flusso è gestito tramite modalità interrupt, una tecnica utilizzata per gestire la comunicazione seriale in modo efficiente e

reattivo, specialmente in contesti dove il tempo di risposta è critico o la quantità di dati trasmessi è elevata.

Un'interruzione è un meccanismo attraverso il quale un microcontrollore può interrompere il flusso normale di esecuzione del programma per gestire eventi o dati in arrivo: quando si riceve un dato sulla porta seriale (UART), ad esempio, può essere generata un'interruzione per segnalare al microcontrollore di occuparsi immediatamente del dato ricevuto, anziché attendere attivamente. Per attivare tale modalità è stata utilizzata la funzione `HAL_UART_Receive_IT`, è parte della libreria HAL, questa funzione viene utilizzata per ricevere dati tramite UART modalità interrupt, il che significa che la ricezione dei dati è gestita tramite interrupt e non in modalità sincrona o bloccante.

```
HAL_UART_Receive_IT(&huart6, rx_buffer + rx_index, 1);
```

Tale funzione è formata da tre parametri: il primo già definito precedentemente, il secondo è un puntatore al buffer in cui verranno memorizzati i dati ricevuti, in questo caso `rx_buffer` indica il punto del buffer in cui il dato sarà ricevuto e `rx_index` indica l'indice corrente nel buffer; quindi, la loro somma indica la posizione esatta in cui memorizzare il prossimo byte ricevuto.

Infine, il terzo, che specifica il numero di byte da ricevere.

- `HAL_UART_RxCpltCallback`: gestisce la logica post-ricezione, come l'aggiornamento dell'indice del buffer e la riattivazione dell'interrupt per ricevere il prossimo byte.

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {  
    if (huart->Instance == USART6) {  
        if (rx_index < RX_BUFFER_SIZE - 1) {  
  
rx_index++; }else {  
        rx_index = 0; // Gestione overflow  
    }  
    HAL_UART_Receive_IT(&huart6, rx_buffer + rx_index, 1); // Ripristina l'interrupt di ricezione  
}}
```

Quando arriva un byte di dati sull'UART (USART6), l'interrupt associato viene generato, il controllo passa automaticamente alla funzione `HAL_UART_RxCpltCallback` che verifica l'UART di provenienza e gestisce l'indice del buffer di ricezione. Se c'è spazio nel buffer, il byte ricevuto viene memorizzato nel buffer e l'indice `rx_index` viene incrementato, se invece il buffer è pieno, viene gestito l'overflow rinizializzando l'indice `rx_index`. Infine, viene richiesto il prossimo byte di dati tramite `HAL_UART_Receive_IT`, ripristinando così l'interrupt di ricezione per continuare a ricevere dati.

Questo ciclo si ripete ogni volta che un nuovo byte di dati viene ricevuto sull'UART, garantendo una gestione efficiente e reattiva della ricezione dei dati tramite interrupt.

- MQTT_Connect: è la funzione per la connessione del nostro modulo ad un protocollo MQTT utilizzando gli AT Command.

```
void MQTT_Connect(const char* client_id) {
    char cmd[256];
    int client_id_len = strlen(client_id);
    int packet_len = 12 + client_id_len; // Lunghezza del pacchetto MQTT CONNECT
    int cmd_length;

    // Calcola la lunghezza del comando AT+CIPSEND
    cmd_length = sprintf(cmd, "AT+CIPSEND=%d\r\n", packet_len + 2);
    HAL_UART_Transmit(&huart6, (uint8_t*)cmd, cmd_length, HAL_MAX_DELAY);

    // Attendi per un po' (200 ms)
    HAL_Delay(200);

    // Costruisci il pacchetto MQTT CONNECT
    uint8_t connect_packet[256];
    int index = 0;
    connect_packet[index++] = 0x10; // Tipo di messaggio: CONNECT
    connect_packet[index++] = packet_len; // Lunghezza del rimanente

    // Variabile header
    connect_packet[index++] = 0x00; connect_packet[index++] = 0x04; // Lunghezza del protocollo
    connect_packet[index++] = 'M'; connect_packet[index++] = 'Q';
    connect_packet[index++] = 'T'; connect_packet[index++] = 'T'; // Nome del protocollo
    connect_packet[index++] = 0x04; // Livello del protocollo
    connect_packet[index++] = 0x02; // Connessione flag: Clean session
    connect_packet[index++] = 0x00; connect_packet[index++] = 0x3C; // Keep-alive: 60 secondi

    // Payload
    connect_packet[index++] = 0x00; connect_packet[index++] = client_id_len; // Lunghezza del client ID
    memcpy(&connect_packet[index], client_id, client_id_len); // Client ID
    index += client_id_len;

    // Invia il pacchetto CONNECT
    HAL_UART_Transmit(&huart6, connect_packet, index, HAL_MAX_DELAY);

    // Attendi per un po' (2000 ms) per completare la connessione
    HAL_Delay(2000);
}
```

Tale funzione realizza il pacchetto MQTT CONNECT, fondamentale per stabilire una connessione tra client e broker. Questo pacchetto contiene le informazioni necessarie ad autenticare il client e definire i parametri di connessione come, ad esempio, il flag Clean session che determina se il broker deve conservare l'informazione o meno e il keep alive, intervallo di tempo entro cui il client deve inviare messaggi al broker.

- MQTT_Publish: è progettata per pubblicare un messaggio su un topic specificato

```
void MQTT_Publish(const char* topic, const char* message) {
    char cmd[256];
    int topic_len = strlen(topic);
    int message_len = strlen(message);
    int msg_len = topic_len + message_len + 2;
    int cmd_length;

    // Calcola e invia il comando AT+CIPSEND
    cmd_length = sprintf(cmd, "AT+CIPSEND=%d\r\n", msg_len + 2);
    HAL_UART_Transmit(&huart6, (uint8_t*)cmd, cmd_length, HAL_MAX_DELAY);

    // Attendi per un po' (200 ms)
    HAL_Delay(200);

    // Costruisci il pacchetto MQTT PUBLISH
    uint8_t publish_packet[256];
    int index = 0;
    publish_packet[index++] = 0x30; // Tipo di messaggio: PUBLISH
    publish_packet[index++] = msg_len; // Lunghezza del rimanente

    // Variabile header
    publish_packet[index++] = 0x00; publish_packet[index++] = topic_len;
    memcpy(&publish_packet[index], topic, topic_len); // Topic
    index += topic_len;

    // Payload
    memcpy(&publish_packet[index], message, message_len); // Messaggio
    index += message_len;

    // Invia il pacchetto PUBLISH
    HAL_UART_Transmit(&huart6, publish_packet, index, HAL_MAX_DELAY);

    // Attendi per un po' (1000 ms) per completare la pubblicazione
    HAL_Delay(1000);
}
```

Così come MQTT_Connect, tale funzione realizza un pacchetto di tipo publish dove calcola la lunghezza del messaggio e del topic di riferimento.

5.2: Codici Sensore IR

Per quanto riguarda la configurazione e gestione del sensore gestuale codice è costituito due file principali: `ir_gesture3.c` e il `main.c`.

Il file `ir_gesture3.c` è formato da:

- `irgesture3_default_cfg`: si occupa di inizializzare e configurare il dispositivo IR Gesture3 per il suo funzionamento normale, gestendo anche eventuali errori che possono verificarsi durante il processo di inizializzazione.
- `irgesture3_write_register` e `irgesture3_read_register`: richiamano rispettivamente `HAL_I2C_Mem_Write` e `HAL_I2C_Mem_Read`, due funzioni della libreria HAL di ST che permettono la scrittura e la lettura dei registri;
- `irgesture3_get_gesture`: consente di riconoscere la tipologia di gesto che è stata effettuata e viene eseguita solo se non ci sono stati errori durante l'acquisizione delle coordinate del gesto. Questa parte del codice è responsabile del calcolo del gesto basato sulle coordinate iniziali e finali del gesto rilevato dal sensore. Le operazioni effettuate sono:
 1. Se non ci sono stati errori durante l'acquisizione delle coordinate del gesto, il codice procede al calcolo del gesto;
 2. Calcolo delle coordinate normalizzate `end_x`-`end_y` utilizzando le differenze tra le coordinate delle regioni di interesse del sensore (`pd_data.x_left`, `pd_data.x_right`, `pd_data.y_top`, `pd_data.y_bottom`). Queste coordinate vengono normalizzate dividendo la differenza tra le regioni superiori e inferiori per la somma di entrambe. Questo processo è fatto per entrambi gli assi `x` e `y`;
 3. Calcolo della pendenza della linea tra le coordinate iniziali e finali (slope) utilizzando la formula: $(start_y - end_y) / (start_x - end_x)$. In caso di divisione per zero, viene aggiunto un valore molto piccolo (`10-6`), proprio per evitare l'errore di divisione per zero;
 4. Calcolo della distanza euclidea (distance) tra le coordinate iniziali e finali utilizzando la formula della distanza euclidea:
 $\sqrt{(start_x - end_x)^2 + (start_y - end_y)^2}$;
 5. In base alla pendenza e alla distanza calcolati, viene determinato il tipo di gesto;
 6. Se la distanza è inferiore alla soglia del gesto (`IRGESTURE3_DISTANCE_THRESHOLD`), viene considerato un "click";
 7. Se la pendenza è maggiore di 1 e la coordinata `y` iniziale è maggiore della coordinata `y` finale, viene considerato uno "swipe-right";
 8. Se la pendenza è maggiore di 1 e la coordinata `y` iniziale è minore o uguale alla coordinata `y` finale, viene considerato uno "swipe-left";
 9. Se la pendenza è minore di 1 e la coordinata `x` iniziale è maggiore della coordinata `x` finale, viene considerato uno "swipe-down";

10. Se la pendenza è minore di 1 e la coordinata x iniziale è minore o uguale alla coordinata x finale, viene considerato uno "swipe-up".
11. Il gesto rilevato è memorizzato nella variabile puntata da gesture. Infine, viene restituito il flag di errore.

- Application_task: funzione per la gestione del movimento del braccio robotico che invia a tutti i client iscritti allo stesso topic un messaggio relativo al tipo di movimento rilevato.

```
void application_task ( void )
{
    uint8_t gesture = 0;
    if ( IRGESTURE3_OK == irgesture3_get_gesture ( &irgesture3, &gesture ) )
    { switch ( gesture ) {
        case IRGESTURE3_GESTURE_CLICK:
        {
            gesture_maiello=IRGESTURE3_GESTURE_CLICK;
            MQTT_Publish(MQTT_TOPIC,"Click");
            break;
        }
        case IRGESTURE3_GESTURE_SWIPE_UP:
        {
            gesture_maiello=IRGESTURE3_GESTURE_SWIPE_UP;
            MQTT_Publish(MQTT_TOPIC,"Up");
            break;
        }
        case IRGESTURE3_GESTURE_SWIPE_DOWN:
        {
            gesture_maiello=IRGESTURE3_GESTURE_SWIPE_DOWN;
            MQTT_Publish(MQTT_TOPIC,"Down");
            break;
        }
        case IRGESTURE3_GESTURE_SWIPE_LEFT:
        {
            gesture_maiello=IRGESTURE3_GESTURE_SWIPE_LEFT;
            MQTT_Publish(MQTT_TOPIC,"Left");
            break;
        }
        case IRGESTURE3_GESTURE_SWIPE_RIGHT:
        {
            gesture_maiello=IRGESTURE3_GESTURE_SWIPE_RIGHT;
            MQTT_Publish(MQTT_TOPIC,"Right");
            break;
        }
        default:
        {
            break;
        }
    } } }
```

La funzione dichiara una variabile gesture di tipo uint8_t e richiama la funzione irgesture3_get_gesture per ottenere il gesto rilevato dal sensore.

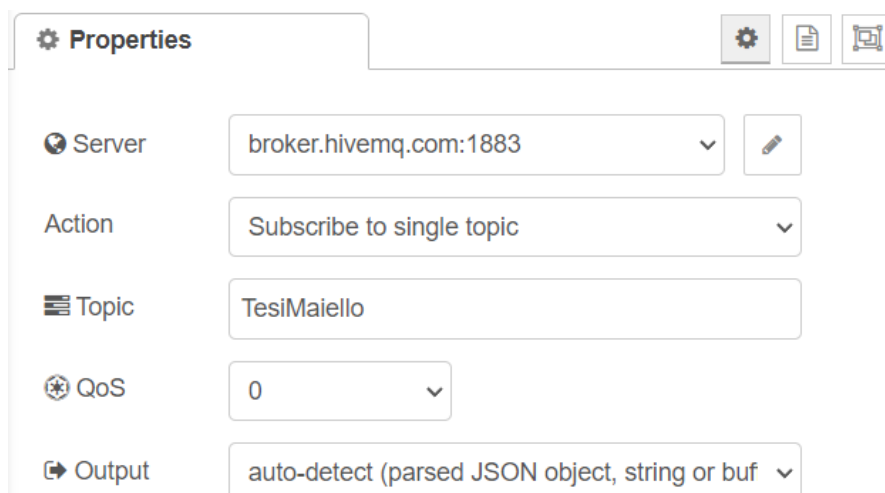
Se il gesto è ottenuto correttamente, viene eseguito uno switch sulla variabile gesture per determinare quale gesto è stato rilevato. Ogni

caso all'interno dello switch imposta una variabile globale `gesture_maiello` con il valore corrispondente al gesto rilevato e richiama la funzione `MQTT_Publish` per pubblicare a tutti i client l'informazione relativa al tipo di movimento rilevato dal sensore. Se il gesto non corrisponde a nessuno dei casi previsti, non viene eseguita alcuna azione.

5.3: Visualizzazione dei risultati

Con l'aiuto di STM32CubeMonitor è possibile visualizzare nell'area debug l'informazione relativa al movimento.

Il blocco `mqtt_in` è stato usato con i seguenti parametri:



The screenshot shows the 'Properties' window for a block named 'mqtt_in'. It contains the following settings:

- Server:** `broker.hivemq.com:1883`
- Action:** `Subscribe to single topic`
- Topic:** `TesiMaiello`
- QoS:** `0`
- Output:** `auto-detect (parsed JSON object, string or buf`

- Il primo si riferisce all'indirizzo del broker utilizzando per la ricezione e trasmissione dati, in questo caso è stato utilizzato HiveMQ;
- Il secondo specifica il tipo di azione da eseguire che in questo caso corrisponde alla sottoscrizione ad un determinato topic;
- Il terzo indica il Topic utilizzato;
- Il quarto indica che la qualità del messaggio ricevuto che in questo caso è 0;
- Il quinto invece indica il tipo di uscita.

Oltre alla visualizzazione del messaggio in tempo reale, nella dashboard è stato aggiunto un blocco "Time&Gesture" che mi permette di salvare gli swape effettuati dal braccio robotico all'interno di un file.

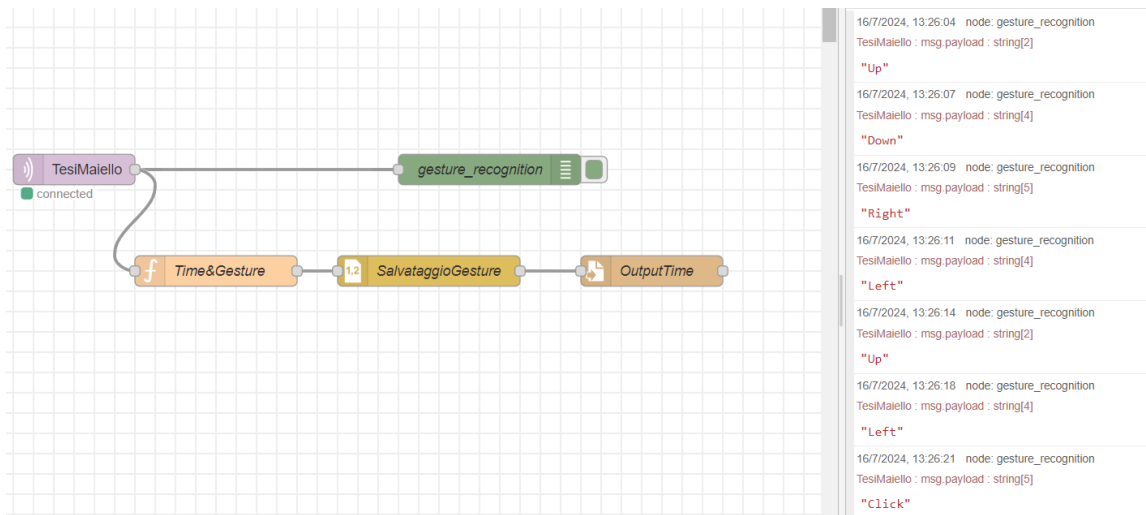


Figura 5.3: Dashboard STM32CubeMonitor

Per la verifica del corretto funzionamento e della percentuale di messaggi trasmessi sono state effettuate svariate prove, raccolte nella seguente tabella:

Validazione	
N campione	1
Dimensione Campione	30 swipe
Tasso di correttezza	100%

Come si può evincere, dopo aver effettuato 30 swipe per tipo di gesto (up, down, right, left, click) il tasso di correttezza e quindi la percentuale di messaggi ricevuti dai client è stato del 100%.

CAPITOLO 6: Conclusioni e sviluppi futuri

In questa tesi, abbiamo esplorato come il riconoscimento gestuale sia diventato un campo di ricerca sempre più promettente, evidenziando l'importanza di dispositivi di rilevamento dei gesti per migliorare l'interazione tra l'uomo e la macchina. In particolare, abbiamo analizzato il potenziale del sensore IR Gesture 3 Click nei vari campi applicativi, con l'obiettivo di ottimizzare l'esperienza utente attraverso il rilevamento accurato dei gesti.

Abbiamo condotto una serie di esperimenti e test per valutare l'efficacia del sensore IR Gesture 3 Click nel rilevare i gesti specificati in diverse condizioni ambientali, come variazioni di luce ambiente. I risultati hanno confermato che il sensore è in grado di rilevare i gesti con elevata precisione e affidabilità, dimostrando una robustezza significativa anche in condizioni di illuminazione variabili.

Un aspetto cruciale di questo studio è stato l'integrazione del modulo WiFi ESP Click per la trasmissione delle informazioni rilevate dal sensore. Una volta che il sensore IR Gesture 3 Click ha rilevato un gesto, i dati relativi al movimento sono trasmessi tramite il modulo WiFi a un broker MQTT. Questo processo avviene in tempo reale, permettendo una condivisione immediata e affidabile delle informazioni tra diversi dispositivi e sistemi.

Il protocollo MQTT (Message Queuing Telemetry Transport) è stato scelto per la sua leggerezza ed efficienza nella gestione delle comunicazioni machine-to-machine (M2M). Utilizzando MQTT, i dati sui gesti possono essere pubblicati su specifici "topic" e sottoscritti da vari client, consentendo un'integrazione flessibile e scalabile in diverse applicazioni.

Gli sviluppi futuri nell'uso di MQTT e nella condivisione delle informazioni promettono di ampliare significativamente le capacità e l'applicabilità del protocollo, rendendolo una componente ancora più vitale nell'ecosistema IoT. Con l'integrazione di tecnologie avanzate come AI, edge computing e standardizzazione, MQTT continuerà a essere una scelta preferita per la trasmissione di dati leggeri e affidabili, aprendo la strada a nuove innovazioni e miglioramenti nelle applicazioni future.

In conclusione, la gesture recognition offre un'interfaccia utente intuitiva e innovativa che potrebbe rivoluzionare l'interazione umana con la tecnologia e l'ambiente circostante. Integrando questa tecnologia con il protocollo MQTT, si aprono nuove possibilità per una comunicazione efficiente e affidabile tra dispositivi, migliorando ulteriormente l'esperienza utente. Con il continuo avanzamento delle tecnologie e l'espansione delle applicazioni, questa integrazione ha il potenziale di migliorare significativamente la nostra quotidianità, rendendo le interazioni con i dispositivi più fluide e immediate, e aprendo la strada a nuove innovazioni e miglioramenti nella qualità della vita.

Bibliografia

- [1] Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review.
- [2] <https://www.sciencedirect.com/science/article/abs/pii/S0278612522000577>
- [3] <https://core.ac.uk/download/pdf/11807623.pdf>
- [4] <https://www.mikroe.com/1000-click-boards>
- [5] <https://www.mikroe.com/ir-gesture-3-click>
- [6] Analog Devices. ADPD2140: Infrared Light Angle Sensor (Rev. 0) 2018
- [7] <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>
- [8] <https://www.mikroe.com/wifi-esp-click>
- [9] <https://www.st.com/en/evaluation-tools/nucleo-f401re.html>
- [10] <https://www.st.com/en/ecosystems/stm32cube.html>
- [11] <https://www.st.com/en/development-tools/stm32cubeide.html>
- [12] <https://www.st.com/en/development-tools/stm32cubemonitor.html>

