



Università degli Studi di Salerno
Dipartimento di Informatica

Tesi di Laurea di I livello in
Informatica

Sistema di autenticazione hardware e software per l'IoT con co-processore crittografico ATECC508A

Relatore

Prof. Christiancarmine Esposito

Candidato

Carmine Citro

Anno Accademico 2021/2022

Abstract

L'Internet of Things (IoT) è un settore in costante espansione che offre numerose opportunità in diversi campi.

Tuttavia, il rapido sviluppo dell'IoT ha anche creato nuovi rischi di sicurezza, aumentando l'esposizione di reti e dispositivi a possibili attacchi da parte di cybercriminali.

Questa tesi si focalizzerà sull'IoT Security e presenterà un progetto hardware volto a migliorare l'autenticazione dei dispositivi IoT.

Il progetto utilizza due SparkFun Artemis RedBoard, uno configurato come master e l'altro come slave, entrambi collegati a processori crittografici ATECC508A.

Il master è collegato ad un sensore BME280 per raccogliere i dati ambientali, mentre lo slave è dotato di uno schermo SparkFun Micro OLED Breakout per visualizzare i dati.

La connessione tra master e slave avviene attraverso Bluetooth.

In caso di verifica positiva dei dati da parte dello slave, le informazioni saranno visualizzate sullo schermo. Questo progetto dimostra come una solida autenticazione hardware possa migliorare la sicurezza dei dispositivi IoT e proteggere le reti da possibili minacce esterne.

Indice

Introduzione	1
1 La Tecnologia IoT	2
1.1 Definizione IoT	2
1.2 Applicazioni IoT	4
1.3 Architettura, elementi e protocolli IoT	5
1.3.1 Perception Layer	5
1.3.2 Network Layer	6
1.3.3 Application layer	10
1.4 La Sicurezza nell'ambito IoT	11
1.5 Attacchi alla sicurezza IoT	12
1.6 Minacce alla sicurezza IoT	13
1.6.1 Perception layer threats	13
1.6.2 Network layer threats	14
1.6.3 Application layer threats	16
1.6.4 Requisiti di sicurezza IoT	17
1.6.5 La sicurezza dei dati	17
1.6.6 Sicurezza delle Comunicazioni	18
1.6.7 Sicurezza di dispositivi	19
1.6.8 Conclusioni	20
2 Dispositivi e strumenti di sviluppo utilizzati	21
2.1 Introduzione della Soluzione IoT Security	21
2.2 SparkFun RedBoard Artemis	21
2.3 SparkFun Cryptographic ATECC508A	23
2.4 SparkFun Environmental Combo BME280	24
2.5 SparkFun Micro OLED Breakout	26
2.6 BLE Cortex-M4F	28
2.7 Camblaggio utilizzato	28
2.8 Tecnologie utilizzate	30
2.9 Architettura sistema	31

3	Implementazione software della soluzione	32
3.1	Prelevamento, firma, e invio dei dati	32
3.1.1	Dichiarazioni variabili	33
3.1.2	Inizializzazione componenti hardware	34
3.1.3	Connessione	34
3.1.4	Firma e invio dati ambientali	36
3.1.5	Invio firma	37
3.1.6	Ulteriori funzioni utilizzate	38
3.2	Ricezione, verifica e visualizzazione dati	39
3.2.1	Inizializzazione varibili	39
3.2.2	Connessione destinatario	40
3.2.3	Ricezione della firma e dati ambientali	41
3.2.4	Verifica e visualizzazione	42
3.2.5	Esempio di verifica effettuata	44
3.2.6	Esempio di verifica fallita	45
4	Conclusioni e sviluppi futuri	46
	Bibliografia	46
	Elenco delle figure	53

Introduzione

Negli ultimi anni, l'Internet of Things (IoT) è diventato uno dei temi più caldi in ambito tecnologico, suscitando grande interesse e aspettative in diversi settori industriali.

L'IoT si riferisce alla connessione di oggetti e dispositivi, permettendo loro di scambiarsi informazioni tra loro e con l'utente, senza bisogno dell'intervento umano.

Grazie all'IoT, oggetti come lampade, frigoriferi, telecamere di sicurezza, sensori e dispositivi di monitoraggio possono comunicare tra loro e con gli utenti, offrendo una vasta gamma di applicazioni e servizi.

L'espansione dell'IoT ha generato nuove opportunità e benefici, come la riduzione dei costi e la creazione di soluzioni di automazione intelligenti, ma ha anche portato alla creazione di nuovi rischi e vulnerabilità.

In particolare, l'IoT Security è diventato un tema sempre più critico, poiché la connessione dei dispositivi crea nuovi punti di accesso per gli attacchi informatici, mettendo a rischio la privacy e la sicurezza dei dati.

La presente tesi si concentrerà sull'IoT e sulla sua sicurezza, con l'obiettivo di esplorare le sfide legate alla sicurezza dell'IoT e di identificare le misure di sicurezza necessarie per mitigare i rischi.

L'obiettivo principale della tesi sarà quello di fornire una panoramica esauriente della sicurezza dell'IoT, evidenziando le principali minacce e le contromisure possibili per proteggere la privacy e la sicurezza dei dati.

L'elaborato è organizzato come di seguito: nel capitolo 1 verrà introdotta la tecnologia IoT e IoT Security la sua evoluzione e tutte le sue caratteristiche. Nel capitolo 2 verranno descritte tutte le tecnologie e dispositivi utilizzati, le loro caratteristiche e funzioni infine verrà mostrata l'architettura finale del sistema proposto.

Nel capitolo 3 verrà mostrata tutta l'implementazione software utilizzata per programmare lo schema di autenticazione. Verrà illustrata la logica completa che si trova alla base del software.

Infine nel capitolo 4 verranno illustrate le conclusioni e introdotte alcune migliorie che si possono apportare al progetto in sviluppi futuri.

Capitolo 1

La Tecnologia IoT

1.1 Definizione IoT

Una rete in espansione di oggetti fisici regolarmente utilizzati connessi a Internet è nota come "Internet of Things" (IoT).

Consente la conversione di gadget abilitati a Internet in un ecosistema in rete in cui i dati digitali sono disponibili costantemente e ovunque. I dispo-

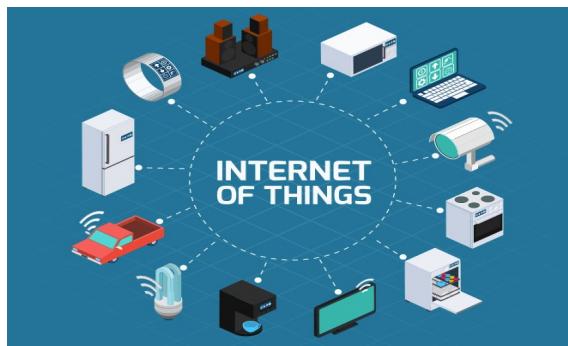


Figura 1.1: Esempio di dispositivi IoT.

sitivi IoT sono oggetti fisici, che vanno dalle piccole ad arrivare alle grandi macchine, che si connettono senza sforzo tra loro tramite Internet, senza la necessità di un coinvolgimento umano [1].

Cisco stima che attualmente ci siano 50 miliardi di dispositivi collegati al Web [2].

I dispositivi IoT sono dotati di sensori che consentono loro di percepire l'ambiente circostante in modo intelligente,

e attuatori per svolgere autonomamente le attività [3].

Questi dispositivi hanno naturalmente risorse limitate, hanno una scarsa quantità di memoria e deboli capacità di elaborazione e calcolo.

CAPITOLO 1. *La Tecnologia IoT*

Un esempio di dispositivi IoT possiamo visualizzarlo nella figura 1.1.

Diverse tecnologie abilitanti come le reti di sensori wireless (WSN),

l'identificazione a radiofrequenza (RFID) e il cloud computing si evolvono come componenti essenziali per l'emergere del paradigma IoT [4].

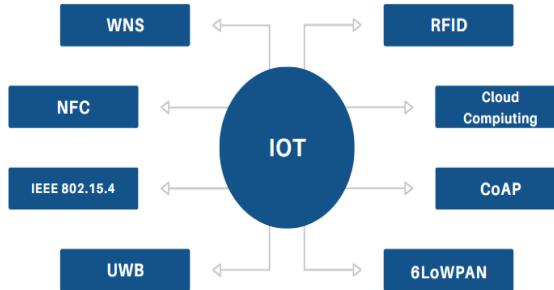


Figura 1.2: Esempio di Tecnologie IoT abilitanti.

Le principali tecnologie abilitanti sono:

Il Wireless Sensor Network (WSN) comprende un grande numero di sensori fisici dislocati nell'ambiente utilizzato per controllare le condizioni ambientali [5].

Per identificare e tracciare gli oggetti IoT viene utilizzata l'identificazione a radiofrequenza(RFID), essa ci permette lo scambio di dati tramite segnali radio a breve distanza.

Il Cloud Computing offre all'infrastruttura IoT risorse di archiviazione e potenza di elaborazione in modo illimitato [6].

Il Protocollo designato per i dispositivi con vincoli di risorse si trova nel livello di applicazione ed è il Constrained Application Protocol(CoAp).

IPv6 Low power Wireless Personal Area Network (6LoWPAN) combina IPv6 e LoWPAN e consente la trasmissione di pacchetti IPv6 su reti IEEE 802.15.4 [7].

Il 6LoWPAN è adatto all'IoT e presenta diversi vantaggi. Ultra WideBand (UWB) è una tecnologia utilizzabile per una vasta scelta di applicazioni IoT a causa del suo basso consumo energetico e la maggiore precisione e sicurezza.

IEEE 802.15.4 è un protocollo per il livello fisico e MAC(Medium Access Control) nelle Wireless Personal Area Network (WPAN) fornisce la connessione ai dispositivi presenti nell'area personale a basso consumo energetico [8].

Near Field Communication(NFC) è una tecnologia che funziona a raggio corto e può essere utilizzata in molti schemi IoT come i pagamenti e l'autenticazione.

CAPITOLO 1. *La Tecnologia IoT*

Le operazioni di accesso alla rete e scambio dei dati tramite NFC sono molto rapide.

La figura 1.2 mostra le principali tecnologie abilitanti.

1.2 Applicazioni IoT

L'IoT fornisce un gran numero di applicazioni per migliorare la vita quotidiana delle persone e attività.

La Smart House è composta da un vasto insieme di dispositivi intelligenti(ad esempio, smart lock, baby monitor, rilevatore di incendio) installati all'interno dell'abitazione e la comunicazione tra quest'ultimi avviene tramite canali wireless. Per accedere in remoto a questi dispositivi domestici, bisogna utilizzare un gateway domestico. G

Grazie allo Smart healthcare possiamo raccogliere trasmettere e archiviare informazioni fisiologiche dei pazienti.

Ad esempio, la frequenza cardiaca del paziente può essere raccolta dal medico tramite degli appositi sensori e trasmessi al server dell'ospedale per scopi di diagnosi e monitoraggio.

Il Trasporto intelligente è composto da un gran numero di veicoli intelligenti che possono comunicare tra di loro (veicolo-veicolo), con la stazione esterna(da veicolo- infrastruttura) e ai pedoni (da veicolo a pedone) su reti wireless.

Grazie a un veicolo intelligente possiamo rilevare lo stato del traffico corrente, gestire la velocità e scambiare questi dati per fornire servizi efficienti e guida sicura.

Per prevenire perdite finanziarie nell settore dell agricoltura si è introdotta **la Smart agriculture** che consente il controllo remoto di temperatura, umidità, irrigazione del suolo, condizioni di umidità e microclima adatto per favorire un'elevata produzione/qualità. Per monitorare i comportamenti del bestiame e le condizioni di salute si possono collegare i sensori direttamente sugli animali.

La Smart industry nota anche come industrial IoT (IIoT), per automatizzare il processo di produzione con un intervento umano insignificante utilizza la tecnologia machine-to-machine.

L'IIoT mira a controllare il processo di produzione, i dati e le problematiche, in modo da fornire prodotti finali efficienti e affidabili.

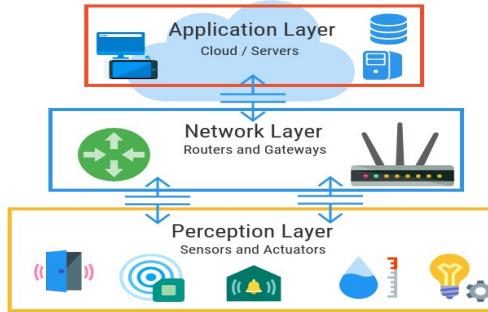


Figura 1.3: Esempio architettura IoT a tre livelli.

La Smart retail consente il tracciamento dei prodotti nei magazzini o durante i viaggi. Si può collegare un sensore a un articolo al dettaglio per monitorare lo stato del prodotto. Vari sistemi intelligenti di acquisto sono stati sviluppati per i clienti, in modo da attirare la clientela.

La Smart grid è un'applicazione tipica dell'IoT che ci permette di misurare, monitorare e di gestire il consumo dell'elettricità, e riduce i problemi e eventuali guasti delle reti elettriche.

1.3 Architettura, elementi e protocolli IoT

L'architettura dell'IoT non è standardizzata, la tipica architettura dell'IoT ha tre strati: percezione, rete e applicazione [9] come mostrato nella Figura 1.3.

1.3.1 Perception Layer

Il livello di percezione comprende una vasta scelta di dispositivi IoT, e sono responsabili del dialogo tra i vari dispositivi e della raccolta dei dati IoT. La raccolta dei dati avviene tramite dei dispositivi intelligenti come tag e sensori di identificazione a radiofrequenza(RFID).

1.3.1.1 Sensori Wireless

Un ruolo essenziale nell'IoT viene svolto dai Sensori Wireless che ci forniscono servizi di rilevamento e comunicazione. [10]. Un'aggregazione di sensori wireless (WSN) è composta da un grande numero di sensori intelligenti dislocati in ambienti remoti, utilizzati per rilevare dati come temperatura, umidità, vibrazioni ecc. I dati che vengono rilevati vengono trasmessi attraverso un multi-hop a un gateway. Come illustrato nella figura 1.4.

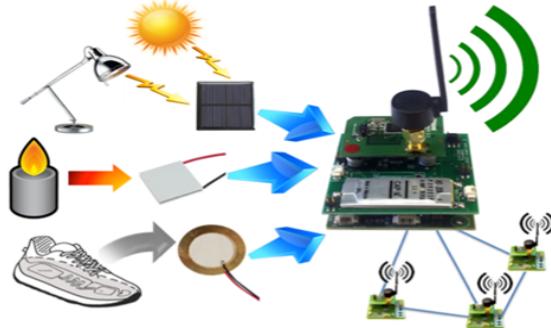


Figura 1.4: Esempio architettura WSN.

1.3.1.2 Radio frequency identification (RFID)

Una tecnologia molto importante per l'IoT è l'RFID perché ha delle caratteristiche molto importanti per l'utilizzo di un dispositivo IoT, e sono l'identificazione la tracciabilità e il monitoraggio degli oggetti [11]. Un sistema RFID è composto da un trasponder di segnale radio (tag) che memorizza un'identità univoca dell'oggetto e un lettore di tag che identifica l'oggetto attraverso onde radio. Il lettore dei tag trasmette il numero di identificazione a un computer per tracciarlo e monitorare l'oggetto, come nella figura 1.5.

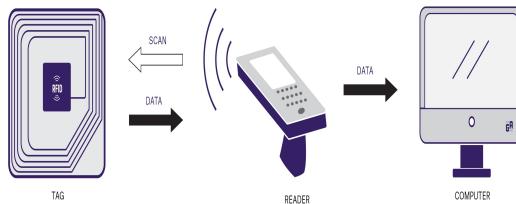


Figura 1.5: Esempio Sistema RFID.

1.3.2 Network Layer

Il livello di rete elabora i dati forniti dal livello di percezione, e invia dati al livello di applicazione. Questo strato è il più importante di un architetture IoT perchè integra diverse tecnologie di comunicazione che abilitano la connettività dei dispositivi IoT. Le tecnologie più utilizzate includono ZigBee, Bluetooth low energy (BLE), IPV6, reti locali a bassa poteza e (6LoWPAN) e reti geografiche a lungo raggio(LoRaWAN).

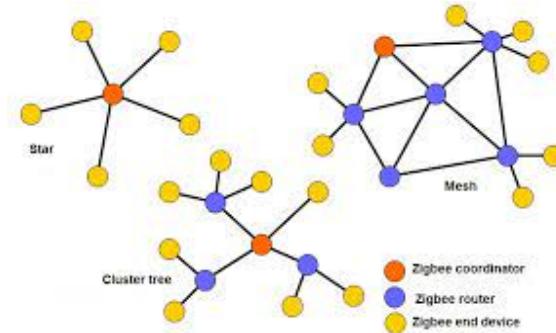


Figura 1.6: Esempio di topologia ZigBee.

1.3.2.1 ZigBee

Per le comunicazioni a corto raggio è molto comune la tecnologia wireless ZigBee [12]. Può essere impiegata in case intelligenti e per l'assistenza Sanitaria intelligente. Lo stack protocollare ZigBee è composto da: livelli fisici (PHY) e di controllo di accesso medio (MAC) basati sullo standard IEEE 802.15.4 [13], poi da un livello di rete(NWK) e infine da un livello applicazione(APP). Una rete ZigBee può avere una topologia a stella, ad albero, oppure a maglia, e ogni rete ha un nodo radice che coordina le operazioni interne alla rete e mantiene i dispositivi della rete in modo sicuro.

Nella rete a stella, i dispositivi finali vengono collegati direttamente al nodo che coordina la rete, invece nelle reti a maglia e ad albero i router intermedi vengono utilizzati per estendere la rete.

Come mostrato nella figura 1.6. Il livello NWK fornisce il routing dei dati utilizzando cluster-tree e algoritmi modificati ad hoc on-demand distance vector(AODV).

Un dispositivo ZigBee può comunicare solo con un altro dispositivo ZigBee, questo ci fa capire che ha un'interoperabilità limitata.

1.3.2.2 Bluetooth Low Energy (BLE)

BLE è una tecnologia a corto raggio, rispetto al classico Bluetooth, BLE è a basso consumo di energia [14] quest'ultimo è ampiamente utilizzato nei sistemi veicolari IoT.

BLE è composto da uno stack protocollare a sua volta composto dal livello PHY, livello MAC, controllo del collegamento logico e protocollo di adattamento (L2CAP) e protocollo di attributo(ATT). Il BLE adotta una topologia a stella e i dispositivi che la compongono sono suddivisi in due categorie master e slave come mostrato nella figura 1.7.

Ogni nodo slave è collegato a un singolo nodo master.

Il nodo master è responsabile di due operazioni, la prima è l'avvio della comunicazione e la seconda operazione è fornire la tabella di programmazione in base al protocollo **time division multiple access(TDMA)**.

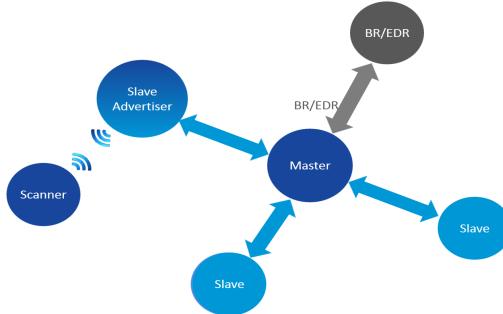


Figura 1.7: Esempio di topologia BLE.

1.3.2.3 6LoWPAN

6LoWPAN combina l'ultima versione del protocollo Internet (IPv6) e la rete personale senza fili a bassa potenza(LoWPAN) [15].

Abilita i dispositivi IoT con limitata capacità di trasmissione dati attraverso canali wireless, utilizzando IPv6, esso è adatto a dispositivi con risorse limitate perchè riduce i costi di trasmissione e supporta la mobilità.

Gli utilizzi principali della tecnologia 6LoWPAN sono la Smart Home, la Smart Agriculture e l'industria IoT.

Rispetto alla topologia ZIgBee, un dispositivo 6LoWPAN può comunicare con un altro dispositivo 6LoWPAN oppure con un dispositivo IEEE 802.15.4. Può anche comunicare con un'altra rete basata su IP come il WI-FI, come nella figura 1.8.

La specificazione di 6LoWPAN definisce uno stack protocollare costituito dai livelli PHY e MAC basato sullo standard IEEE 802.15.4., il livello NWK, il livello di trasporto e il livello di applicazione [16].

Per instradare i pacchetti all'interno della rete 6LoWPAN viene utilizzato il protocollo di instradamento per i dispositivi a bassa potenza e reti con perdite (RPL) [17].

RPL supporta le comunicazioni point-to-point e le comunicazioni point-to-multipoint. Si basa sul grafico aciclico (DAG).

A partire dal DAG, RPL crea un albero DAG(direct acyclic graph) che è un albero orientato alla destinazione che contiene un percorso dal nodo foglia alla radice.

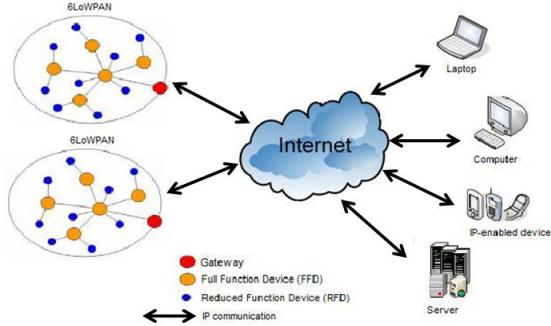


Figura 1.8: Esempio di Architettura 6LoWPAN.

1.3.2.4 LoRaWAN

LoRaWAN è un protocollo di comunicazione a lungo raggio, progettato per connessioni a bassa potenza e per applicazioni IoT scalabili [18].

Come illustrato nella figura 1.9.

La rete LoRaWAN è costituita da dispositivi finali ed è composta da un gateway e un singolo server in una topologia a stella. I Dispositivi finali possono comunicare con uno o più gateway utilizzando lo schema ALOHA tramite link one-hop.

I gateway sono collegati al server di rete tramite protocollo internet.

Le comunicazioni sono bidirezionali e avviate dal dispositivo finale.

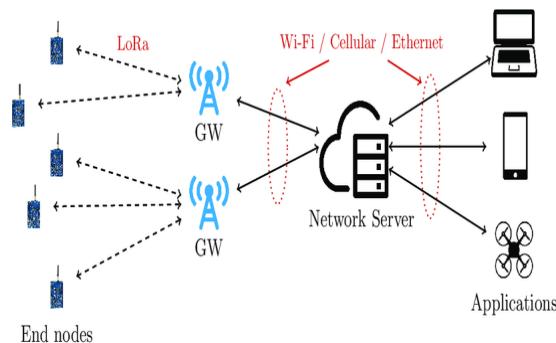


Figura 1.9: Esempio di Architettura LoRaWAN.

1.3.3 Application layer

Il livello di applicazione riceve i dati dal livello di rete e fornisce i servizi richiesti agli utenti IoT.

Supporta una grande varietà di applicazione come, Smart retail, Smart grid e molte altre applicazioni.

I protocolli applicativi più utilizzati sono: il protocollo di applicazione vincolato (CoAP) e il trasporto di telemetria dell'accoppiamento dei messaggi (MQTT).

1.3.3.1 CoAP

Poiché i dispositivi IoT sono limitati in termini di risorse, si è adottata una strategia diversa, invece di utilizzare il protocollo HTTP che non è adatto per dispositivi a bassa potenza a causa della sua complessità è stato introdotto

il protocollo CoAP, che include le funzionalità di HTML come mostrato nella figura 1.10.

CoAP è un protocollo di messaggistica basato sull'architettura REST (Representational State Transfer).

Questo protocollo restituisce quattro tipologie di risposta e sono: confermabile, non confermabile, riconoscimento e ripristino.

Il protocollo CoAP fornisce funzionalità in più rispetto al protocollo HTTP, come ad esempio la notifica push (ovvero il server invia una notifica al dispositivo) e il rilevamento delle risorse (ovvero il server può memorizzare l'elenco dei dispositivi).

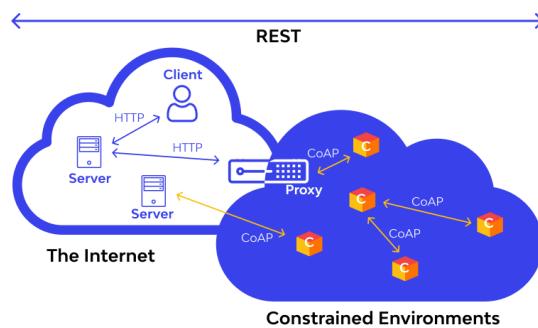


Figura 1.10: Esempio di Architettura CoAP.

1.3.3.2 MQTT

MQTT è un protocollo di messaggistica leggero che fornisce la connettività di rete a gli utenti con applicazioni.

Si basa su un architettura di publish/subscribe, in cui il sistema è costituito da tre componenti principali: publishers, subscribers e broker, come presentato nella figura 1.11. Nel contesto dell'IoT i publishers sono dispositivi incorporati che inviano dati al broker, invece i subscribers sono i server delle applicazioni.

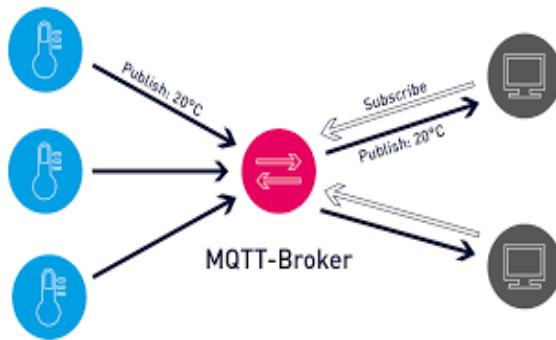


Figura 1.11: Esempio di Architettura MQTT.

1.4 La Sicurezza nell'ambito IoT

L'IoT ha attirato un'attenzione particolare poiché negli ultimi anni ha portato dei cambiamenti rivoluzionari alla vita umana.

l'IoT consente lo scambio di informazioni in una vasta gamma di applicazioni come ad esempio: smart buildings, smart health, smart transport e così via. Nella prima parte di questo capitolo abbiamo introdotto la definizione di rete IoT e presentato le principali tecnologie abilitati dell'IoT che motivano l'emergere dell'IoT.

Inoltre, abbiamo esaminato diverse applicazioni fornite dal paradigma IoT e discusso dei protocolli integrati nell'architettura IoT a tre livelli.

In questa seconda parte di capitolo, ci concentreremo sulle vulnerabilità della sicurezza e sui requisiti dell'IoT.

Presenteremo diversi attacchi alla sicurezza che minacciano gli ambienti IoT. Forniremo alcune tecniche per la sicurezza delle comunicazioni e spiegheremo il concetto di autenticazione.

1.5 Attacchi alla sicurezza IoT

L'IoT si sta evolvendo molto veloce e anche gli attacchi alla sicurezza stanno avanzando, bisogna includere attentamente i requisiti di sicurezza nei sistemi IoT in modo da analizzare le vulnerabilità e gli attacchi IoT.

L'IoT è soggetto a vari tipi di attacchi poiché combina diverse tecnologie esistenti come WSN e RFID.

Pertanto L'IoT eredita le falte di sicurezza di ciascuna tecnologia.

La figura 1.12 fornisce diversi possibili attacchi che possono essere fatti tramite reti IoT.

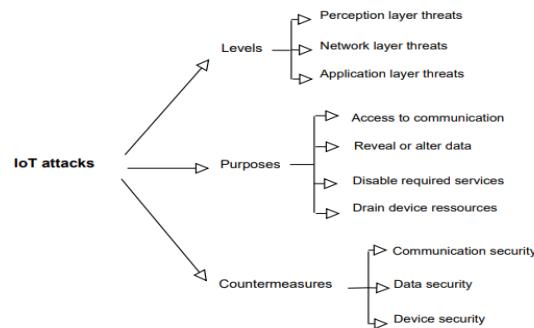


Figura 1.12: Esempio di tassonomia degli attacchi alla sicurezza IoT.

1.6 Minacce alla sicurezza IoT

In base agli attacchi alla sicurezza studiati, formiamo una tassonomia degli attacchi IoT sulla base di livelli, scopi e contromisure, come dimostrato nella figura 1.12.

In questa sezione ci concentreremo sulle vulnerabilità di sicurezza dell'IoT a tre livelli.

livelli(levels) essi esaminano i problemi di sicurezza dell'IoT a tre livelli. Strato di percezione le minacce affrontate sono gli attacchi alla sicurezza all'interno dei principali elementi dell'IoT come WSN e RFID. Le minacce a livello di rete analizzano le vulnerabilità dei suddetti protocolli di comunicazione, invece le minacce a livello di applicazione includono attacchi relativi al software IoT e dispositivi degli utenti finali.

Gli scopi(Purposes) valutano gli impatti che hanno alcuni attacchi sulla sicurezza dei sistemi IoT.

Gli scopi principali degli attacchi IoT sono i seguenti:

- **Accesso alla comunicazione.**
- **Rivelare o alterare i dati.**
- **Disabilitare i servizi richiesti.**
- **Svuotare le risorse del dispositivo.**

Le Contromisure(Countermeasures) consistono nei requisiti di sicurezza per limitare la finalità degli attacchi IoT. Questa classe include la sicurezza delle comunicazioni, la sicurezza dei dati e sicurezza del dispositivo.

Le comunicazioni IoT possono essere protette fornendo autenticazione, controllando gli accessi e non ripudio.

Per proteggere i dati, devono essere considerati dei requisiti di sicurezza come ad esempio: la riservatezza, la provacy e l'integrità. Un'altro importante requisito è la fiducia e la disponibilità dei dispositivi IoT necessari in diversi ambienti.

1.6.1 Perception layer threats

Le risorse limitate e la natura eterogenea dei dispositivi IoT li rendono vulnerabili a vari attacchi alla sicurezza.

I WSN sono generalmente distribuiti in ambienti difficili e non presidiati e quindi sono soggetti a diversi attacchi.

I comuni attacchi alla sicurezza delle WSN sono: **sinkhole, blackhole, wormhole, sybil, denial of service(Dos), node capture e node injection attack** [19].

Analogamente al WSN, le reti RFID sono suscettibili a diversi tipi di attacchi inclusi attacchi **spoofing, clonazione e sniffing**.

L'IoT eredita le minacce alla sicurezza di WSN e RFID perché essi sono elementi cardini delle reti IoT.

1.6.2 Network layer threats

Il protocollo ZigBee implementa meccanismi di sicurezza inclusa la crittografia avanzata standard con **codice di autenticazione del messaggio con concatenamento di blocchi cifrati(AES-CCM)** e **codice di integrità del messaggio (MIC)** per fornire riservatezza, autenticazione e integrità. La sicurezza ZigBee di base su tre chiavi: una chiave di collegamento (per le comunicazioni unicast), una chiave di rete (per le comunicazioni broadcast) e una chiave master (per chiave di collegamento a reti e generazioni di chiavi). La chiave principale è installata nel dispositivo durante il processo di fabbricazione.

La chiave di collegamento può essere generata utilizzando i metodi di trasporto o di creazione della chiave, mentre la chiave di rete può essere acquisita con il metodo di trasporto delle chiavi.

Poiché la chiave master è conservata all'interno del dispositivo, un individuo malintenzionato potrebbe estrarla dalla memoria in seguito a un attacco di acquisizione del nodo.

Dopo aver avuto successo nell'attacco di acquisizione del nodo, un altro tipo di attacco descritto in [20] mira a esaurire l'energia dei nodi ZigBee.

Gli autori del documento [21] hanno condotto una valutazione sulla vulnerabilità della rete ZigBee all'attacco sinkhole.

In [22], gli autori hanno dimostrato che tre sistemi di illuminazione intelligente basati su ZigBee non sono sicuri nei confronti di diversi tipi di attacchi, tra cui attacchi **di negazione del servizio (DoS), estrazione della chiave di rete e iniezione di codice**.

Il protocollo BLE offre sicurezza e autenticazione mediante l'utilizzo dell'algoritmo AES-CCM a 128 bit, simile a ZigBee.

chiave simmetrica viene generata attraverso una procedura di accoppiamento. Innanzitutto, i dispositivi IoT scambiano le informazioni necessarie per l'autenticazione. Successivamente, generano e scambiano chiavi temporanee utilizzando un metodo di accoppiamento. Infine, i dispositivi possono scambiare e memorizzare chiavi comuni da utilizzare in future comunicazioni.

Tuttavia, i metodi di accoppiamento presentano vari problemi di sicurezza, come **l'intercettazione, gli attacchi man-in-the-middle (MTM) e brute force**, come evidenziato in [23] e [24]. Recentemente, è stata introdotta una nuova procedura di accoppiamento basata sulla **curva ellittica di Diffie-Hellman (ECDH)**. Tuttavia, gli autori in [25] e [26], Hanno

dimostrato che presenta problemi simili in [27], gli autori hanno descritto altri tipi di attacchi, come la **divulgazione di dati e l'attacco DoS**, che possono essere eseguiti su un sistema BLE.

Questi attacchi possono essere eseguiti anche su un sistema di chiusura intelligente basato su BLE.

I dispositivi a basso consumo energetico possono collegarsi a Internet utilizzando il protocollo 6LoWPAN e gli indirizzi IPv6.

La compressione dell'intestazione IPv6 e la frammentazione dei pacchetti sono utilizzate per ridurre l'overhead di trasmissione.

Tuttavia, questo protocollo non fornisce riservatezza, autenticazione o integrità dei dati.

Un attaccante potrebbe inserire frammenti di dati falsi nell'intestazione di un pacchetto legittimo, il quale sarebbe utilizzato dal nodo ricevente per ricostruire il pacchetto originale.

Ciò potrebbe causare la creazione di un pacchetto corrotto, con conseguente occupazione dello spazio di buffer del nodo ricevente e l'impossibilità di ricevere altri frammenti [28].

Questo tipo di attacco potrebbe essere utilizzato in modo ripetitivo per causare un attacco DoS [29].

RPL prevede tre opzioni di sicurezza: non protetta, preinstallata e autenticata nell'intestazione del pacchetto.

La modalità non protetta viene utilizzata quando la sicurezza è garantita dal livello MAC. Nella modalità preinstallata, vengono utilizzate chiavi preinstallate per l'accesso alla rete RPL.

La modalità autenticata non è completamente descritta dalle specifiche di RPL.

In assenza di sicurezza a qualsiasi livello, un attaccante può eseguire diversi tipi di attacchi contro le reti RPL, come ad esempio sinkhole, blackhole, flooding, sybil e DoS, come presentati in [29] e [30].

La sicurezza delle comunicazioni in 6LoWPAN si basa sulla protezione a livello MAC o APP. La crittografia e l'integrità dei dati a livello MAC sono garantite mediante AES-CCM e MIC.

Tuttavia, le specifiche dell'IEEE 802.15.4 non definiscono la procedura di gestione delle chiavi. Invece, il protocollo LoRaWAN utilizza **l'algoritmo AES a 128 bit** e il MIC per garantire la riservatezza e l'integrità dei dati. Quando un dispositivo IoT si unisce alla rete LoRaWAN, il server di rete invia al dispositivo finale due chiavi di sessione: la chiave di sessione di rete e quella dell'applicazione. Queste chiavi vengono utilizzate per la crittografia/decrittografia dei dati e per il MIC.

La principale vulnerabilità del protocollo LoRaWAN è dovuta alla gestione delle chiavi, in quanto sono memorizzate sui dispositivi finali e possono essere compromesse tramite **attacchi side-channel**. Inoltre, l'utilizzo delle

stesse chiavi di sessione da parte di più end-device per proteggere le comunicazioni multicast può rendere possibile per un attaccante accedere alle chiavi di un nodo e quindi compromettere le comunicazioni di altri dispositivi [31]. In [32], è stato dimostrato che la rete LoRaWAN è vulnerabile anche ad attacchi DoS e MTM.

1.6.3 Application layer threats

CoAP è un protocollo di livello applicativo che consente ai dispositivi con risorse limitate di interagire in modo RESTful. Poiché CoAP si basa su UDP come protocollo di trasporto, è stato proposto l'utilizzo del Datagram Transport Layer Security (DTLS) per garantire la riservatezza, l'autenticazione e l'integrità dei dati nel protocollo CoAP [33]. Tuttavia, le restrizioni di DTLS possono essere considerate come potenziali minacce per la sicurezza del protocollo CoAP [34]. Per proteggere il trasferimento dei dati con il protocollo MQTT, è stato introdotto il **Secure Socket Layer (SSL)**.

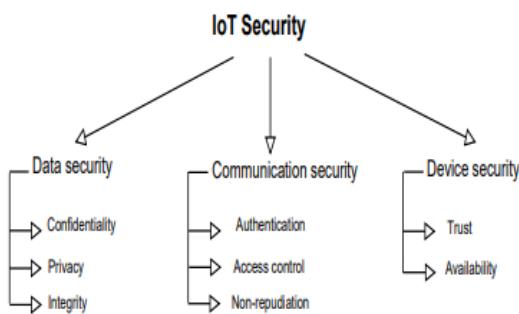


Figura 1.13: Esempio di Tassonomia dei requisiti di sicurezza dell'IoT.

Il protocollo SSL utilizza una tecnica **crittografica asimmetrica** per proteggere i dati durante la loro trasmissione.

Tuttavia, rimane ancora vulnerabile ad attacchi Man-in-the-Middle (MTM) [35]. Per migliorare la sicurezza nella trasmissione dei dati, è stata proposta un'estensione di MQTT chiamata Secure MQTT (SMQTT) [36]. In SMQTT, i publisher e i subscriber si registrano presso il broker e ottengono una chiave segreta per la crittografia e la decrittografia dei dati. Tuttavia, la generazione della chiave e gli algoritmi di crittografia non sono standardizzati. Nell'ambito dell'IoT, le vulnerabilità del software e dei dispositivi degli utenti possono essere sfruttate dagli aggressori, che possono impersonare o manipolare gli utenti legittimi per accedere al sistema IoT e imiettare software dannoso. L'assenza di autenticazione dell'utente ha portato a diversi attacchi IoT, come ad esempio Bashlite e Mirai [37].

1.6.4 Requisiti di sicurezza IoT

In base alla finalità dei possibili attacchi contro gli oggetti connessi all'IoT, possiamo suddividere i criteri di sicurezza IoT in tre categorie: protezione dei dati, protezione della comunicazione e protezione dei dispositivi, come mostrato nell'immagine 1.13.

In particolare, per garantire la riservatezza delle informazioni riservate, è necessario proteggere adeguatamente i dati raccolti dai dispositivi IoT.

In aggiunta, è fondamentale preservare l'integrità e la sicurezza delle comunicazioni tra i dispositivi per evitare eventuali tentativi di intercettazione da parte di malintenzionati.

In taluni casi, gli oggetti connessi tra loro nell'ambito dell'IoT possono cooperare per fornire servizi avanzati all'utente.

Ecco perché è di primaria importanza proteggere adeguatamente questi dispositivi.

1.6.5 La sicurezza dei dati

I dispositivi IoT sorvegliano l'ambiente fisico e trasmettono i dati raccolti tramite canali wireless.

Tuttavia, questi dati trasmessi sono vulnerabili a varie minacce alla sicurezza, come l'ascolto abusivo e la manipolazione.

Per garantire la protezione dei dati nel contesto IoT, è essenziale mantenere la loro riservatezza, privacy e integrità.

La riservatezza dei dati rappresenta il processo di mascherare le informazioni private agli oggetti IoT non autorizzati [38] [39]. Secondo [40], la riservatezza dei dati rappresenta un'importante questione che richiede molta attenzione.

In quanto i dispositivi IoT dispongono di risorse limitate [41], i meccanismi

di crittografia standard non possono essere direttamente implementati nel sistema IoT.

Per garantire la protezione e la riservatezza dei dati, gli autori di [42] hanno proposto l'utilizzo di algoritmi crittografici leggeri.

Allo stesso modo, gli autori di [43] hanno sottolineato che l'impiego di progetti basati sulla privacy può incrementare i livelli di confidenzialità.

La tutela della privacy comprende l'anonimizzazione delle informazioni personali e la possibilità di controllare l'utilizzo di tali informazioni [44].

È necessario valutare la privacy dei dati durante la raccolta, la trasmissione e l'archiviazione. Esistono diverse soluzioni pratiche per gestire la privacy dei dati, tra cui l'anonimizzazione, l'uso di **generatori di numeri pseudocasuali, cifrari a blocchi e cifrari a flusso** [45].

L'integrità dei dati è fondamentale per garantire che i dati trasmessi non siano stati alterati o modificati. Ciò implica il mantenimento della coerenza, dell'accuratezza e dell'affidabilità dei dati.

Tuttavia, molti algoritmi crittografici di hash, come **MD5 e SHA1**, non possono essere implementati a causa delle limitazioni di risorse dei dispositivi IoT [46].

Per ovviare a questo problema, sono state proposte diverse funzioni di hash leggere, come quelle descritte in [47] e [48]. In caso di rilevamento di dati manomessi, i meccanismi di correzione degli errori come i controlli di ridondanza ciclica (CRC) e le funzioni di checksum possono essere utilizzati per risolvere il problema [49].

1.6.6 Sicurezza delle Comunicazioni

È importante avere un processo di autenticazione prima di qualsiasi comunicazione tra dispositivi IoT, in modo da garantire l'accesso solo ai dispositivi autorizzati ai sistemi o alle informazioni.

Questo processo di **autenticazione** aiuta anche a evitare il ripudio della comunicazione. Il processo di validazione dell'identità tramite login e altre informazioni **come password, PIN e certificati digitali è noto come autenticazione** [45].

L'autenticazione è necessaria per garantire la sicurezza delle comunicazioni nel sistema IoT tra due o più parti. L'autenticazione garantisce che solo gli utenti autorizzati possano accedere ai dispositivi IoT e che le comunicazioni non possano essere ripudiate. Quando un nuovo dispositivo si collega alla rete, deve autenticarsi prima di poter scambiare dati.

L'autenticazione può essere verificata utilizzando algoritmi **crittografici leggeri, primitive fisiche o identificazione biometrica** [50] [51].

Il controllo degli accessi è una funzione di sicurezza che verifica l'autorizzazione di utenti e sistemi per accedere ad altri sistemi e risorse [52].

Ci sono diversi tipi di algoritmi di controllo degli accessi, tra cui quelli basati su ruoli, organizzazioni, capacità, attributi e fiducia [45].

Nel contesto dell'IoT, il non ripudio è un elemento importante della sicurezza della rete. Si riferisce alla capacità di garantire che un nodo IoT non possa negare l'invio di un messaggio e che il destinatario non possa negare di averlo ricevuto [38].

Questo è particolarmente importante in ambito commerciale, ad esempio per i contratti digitali, poiché garantisce che le comunicazioni siano autentiche e valide. Per ottenere il non ripudio si può utilizzare la **crittografia a chiave pubblica (PKC)** [53].

1.6.7 Sicurezza di dispositivi

Per garantire un ambiente critico sicuro, è fondamentale assicurare la collaborazione e la fiducia tra i nodi che interagiscono. Inoltre, c'è una crescente richiesta di dispositivi IoT disponibili sul mercato.

La fiducia è un aspetto fondamentale per gli utenti dell'IoT, come evidenziato nel [54]. La gestione della fiducia consiste nel prendere decisioni sulla comunicazione con entità sconosciute, come indicato nel [55].

Per garantire un sistema IoT sicuro, è necessario interagire esclusivamente con dispositivi IoT fidati, in modo da evitare azioni indesiderate da parte di nodi maligni.

Secondo [56], le strategie di gestione della fiducia si possono classificare in due categorie principali: fiducia deterministica e non deterministica. La fiducia deterministica si avvale di meccanismi basati su regole e certificati, mentre la fiducia non deterministica fa affidamento su metodi che utilizzano la reputazione, le previsioni e le reti sociali.

I meccanismi basati sulle regole si basano su un insieme di regole per valutare la fiducia.

Invece, gli approcci basati sui certificati utilizzano **le chiavi pubbliche o private e le firme digitali** per determinare la fiducia.

I sistemi che si basano su esperienze passate possono definire la fiducia nell'IoT. Se non esiste alcuna informazione preliminare, i metodi basati sulla previsione possono essere utilizzati. Invece, i sistemi basati sulla reputazione si affidano alla reputazione globale delle entità, mentre quelli basati sulle reti sociali considerano la reputazione sociale delle entità [56].

La disponibilità dei dispositivi è un fattore cruciale nei sistemi IoT, poiché sono utilizzati in settori vitali come l'economia, l'industria, la sanità, ecc. [57].

Secondo [58], la disponibilità della rete IoT deve essere garantita sia a livello hardware che software. La disponibilità hardware dell'applicazione IoT si riferisce alla presenza di tutti i dispositivi in ogni momento, mentre la disponibilità del software si riferisce alla capacità di fornire servizi ovunque e in qualsiasi momento. I dispositivi IoT sono soggetti a vari attacchi, come

CAPITOLO 1. *La Tecnologia IoT*

DoS e DDoS, che possono impedire la fornitura dei servizi o compromettere la disponibilità della rete [59]. Gli autori sottolineano l'importanza del rilevamento e del recupero degli attacchi DoS/DDoS negli ambienti IoT.

1.6.8 Conclusioni

La tecnologia IoT si riferisce alla successiva evoluzione di Internet. Presto, un grande numero di dispositivi sarà connesso alla rete. Questi dispositivi IoT possono scambiare informazioni delicate che potrebbero essere compromesse. Per questo motivo, la sicurezza dell'IoT rappresenta una delle maggiori preoccupazioni. In questo capitolo, abbiamo esaminato le vulnerabilità e le minacce alla sicurezza delle reti IoT e presentato una classificazione degli attacchi IoT. Abbiamo anche suggerito una classificazione dei requisiti di sicurezza in base agli obiettivi degli attacchi, al fine di garantire un sistema IoT sicuro. Nel prossimo capitolo, invece, parleremo delle tecnologie utilizzate per sviluppare una soluzione che riguarda la sicurezza nell'ambito IoT, basata sull'autenticazione.

Capitolo 2

Dispositivi e strumenti di sviluppo utilizzati

2.1 Introduzione della Soluzione IoT Security

La soluzione di sicurezza nell'ambito dell'IoT proposta è uno schema di autenticazione, che ci permette di prelevare dati tramite un sensore **SparkFun Environmental Combo Breakout - CCS811/BME280** [60] e firmarli tramite **Cryptographic Co-Processor ATECC508A** [61], e inviarli tramite **SparkFun RedBoard Artemis** [62] che ha un ble integrato.

In ricezione ci sarà un altro **SparkFun RedBoard Artemis** che acquisirà i dati e li verificherà tramite una funzione, messa a disposizione dalla libreria del **Cryptographic Co-Processor ATECC508A**, se questa funzione restituirà un booleano di valore true allora visulizzerà i dati provenienti dal sensore **SparkFun Environmental Combo Breakout - CCS811/BME280** su uno **SparkFun Micro OLED Breakout** [63].

In questo capitolo andremo a spiegare passo passo tutte le componenti e le tecnologie utilizzate per completare la soluzione introdotta in precedenza.

2.2 SparkFun RedBoard Artemis

Nella soluzione proposta abbiamo utilizzato due dispositivi del genere, uno funge da mittente e un altro da destinatario, nel mittente viene caricato il programma di prelevamento e firma dei dati e nel destinatario quello di verifica dei dati e visulizzazione.

La comunicazione tra questi ultimi avviene tramite la scheda ble integrata. La scheda RedBoard Artemis [62] è molto simile ad Arduino, ma presenta alcune caratteristiche aggiuntive, come il **Bluetooth Low Energy** integrato [23], 1 MB di memoria flash e un consumo di energia inferiore a 1 mA.

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

Inoltre, è in grado di eseguire modelli TensorFlow.

La scheda utilizza il modulo Artemis di SparkFun, che è molto potente, e lo presenta in un formato Uno facile da usare.

La programmazione dell'Artemis è stata resa più familiare grazie alla ri-scrittura del core Arduino, rendendo la programmazione semplice come `Serial.begin(9600)`.

Il tempo per il primo lampeggio della scheda è inferiore a cinque minuti.

La scheda RedBoard Artemis è dotata di un condizionamento dell'alimentazione migliorato e di un collegamento USB-seriale, entrambi migliorati nel corso degli anni sulla linea di prodotti RedBoard.

La scheda utilizza un moderno connettore USB-C per semplificare la programmazione e un connettore Qwiic per semplificare l'utilizzo di I2C.

La scheda è completamente compatibile con il core Arduino di SparkFun e può essere facilmente programmata utilizzando **l'IDE Arduino**.

Inoltre, è dotata di un connettore JTAG per gli utenti che preferiscono utilizzare strumenti professionali per la programmazione. La scheda presenta anche un microfono MEMS digitale integrato per i comandi vocali sempre attivi con TensorFlow e l'apprendimento automatico, e un jumper per la misurazione del consumo di corrente per i test a basso consumo.

La scheda dispone di 1 MB di memoria flash e 384 KB di RAM per i progetti. Il modulo Artemis funziona a 48 MHz con una modalità turbo a 96 MHz e supporta anche il Bluetooth.

Nella figura 2.1 viene visualizzato il dispositivo in questione.

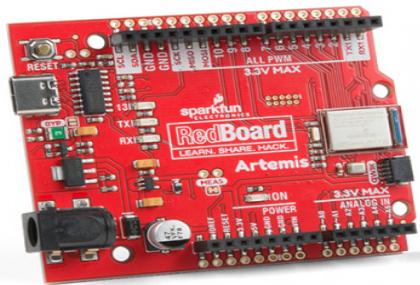


Figura 2.1: Dispositivo SparkFun RedBoard Artemis.

2.3 SparkFun Cryptographic ATECC508A

Nella soluzione proposta, abbiamo utilizzato due ATECC508A entrambi collegati a due SparkFun RedBoard Artemis, uno ci permette di firmare i dati che provengono dal sensore SparkFun Environmental Combo Breakout CCS811/BME280, e l'altro ci permette di verificare questi dati.

La scheda SparkFun ATECC508A per il cripto-processore consente di aggiungere una solida autenticazione sicura a un dispositivo IoT, sistema embedded o dispositivo periferico in modo semplice e veloce.

Dispone di due porte Qwiic plug-and-play che rendono facile il collegamento al resto del sistema senza la necessità di saldature.

In alternativa, è possibile utilizzare un pin a passo 0,1" per collegarlo a una breadboard.

Il chip ATECC508A supporta molti processi crittografici, tra cui:

- La creazione e l'archiviazione sicura di coppie di chiavi asimmetriche univoche basate su **crittografia a curva ellittica (FIPS186-3)** [64].
- **La creazione e la verifica di firme digitali di 64 byte (da 32 byte di dati di messaggio)** [65].
- La creazione di una chiave segreta condivisa su un canale pubblico tramite **l'algoritmo di Diffie-Hellman a curva ellittica** [66].
- Un protocollo di sfida-risposta basato su hash standard utilizzando un algoritmo **SHA-256** [67].

Inoltre, dispone di un generatore di numeri casuali FIPS di alta qualità incorporato nel chip e di una matrice EEPROM da 10 Kb per archiviare chiavi, certificati, dati, registri di consumo e configurazioni di sicurezza.

L'accesso alle sezioni di memoria può essere limitato e la configurazione può essere bloccata per impedire modifiche.

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

Ogni ATECC508A è dotato di un numero di serie univoco di 72 bit garantito e dispone di diverse funzionalità di sicurezza per impedire attacchi fisici al dispositivo stesso o attacchi logici ai dati trasmessi tra il dispositivo. È possibile visualizzare il dispositivo in questione nella figura 2.2.



Figura 2.2: Dispositivo SparkFun Cryptographic ATECC508A.

Prima di essere bloccato in modo permanente, il chip può essere configurato una sola volta.

Tuttavia, questa scheda non ha la capacità di eseguire operazioni di crittografia e decrittografia dei dati. Invece, è in grado di eseguire diversi processi di autenticazione crittografica, come la creazione di chiavi private sicure, l'archiviazione di chiavi sicure e la creazione e verifica di firme digitali.

Per programmare questo dispositivo è stata utilizzata la libreria SparkFun Cryptographic consultabile tramite il link GitHub: [68].

2.4 SparkFun Environmental Combo BME280

Nello schema di autenticazione che abbiamo proposto, questo sensore viene collegato allo Sparkfun Artemis RedBoard mittente per prelevare i dati ambientali, successivamente questi dati verranno firmati e inviati al destinatario.

L'environmental Combo utilizza i circuiti integrati CCS811 e BME280 per rilevare la qualità dell'aria in modo preciso.

Questo dispositivo fornisce informazioni su vari fattori ambientali, come la pressione barometrica, l'umidità, la temperatura e i livelli di CO₂ equiva-

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

lente (eCO₂) e composti organici volatili totali (TVOC).

Tutte le comunicazioni avvengono attraverso il sistema Qwiic tramite I2C, ma ci sono anche pin spaziati di 0,1" disponibili per l'utilizzo su breadboard. Il CCS811 è un sensore diffuso che fornisce letture di eCO₂ in parti per milione (PPM) e di TVOC in parti per miliardo (PPB).

Il sensore è anche in grado di calibrare le letture per l'umidità e la temperatura corrente, e con il BME280, che fornisce informazioni sull'umidità, la temperatura e la pressione barometrica, i sensori possono lavorare insieme per fornire letture più precise.

L'indirizzo I2C del breakout è 0x77/0x5B, selezionabile tramite un ponticello a 0x76/0x5A.

Se si desidera utilizzare più sensori su un unico bus, sarà necessario un multiplexer/Mux.

È possibile visualizzare il dispositivo in questione nella figura 2.3.

Il sistema SparkFun Qwiic Connect è un'ecosistema di sensori, attuatori, schermature e cavi I2C che semplificano la prototipazione e riducono gli errori.

Tutte le schede Qwiic utilizzano un comune connettore JST a 4 pin con passo di 1 mm, consentendo di ridurre lo spazio richiesto sulla scheda e di evitare errori di connessione.

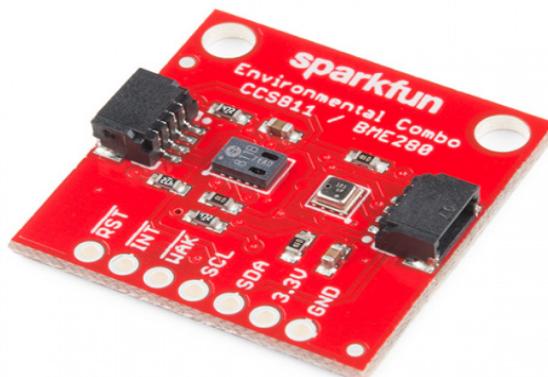


Figura 2.3: Sensore SparkFun Environmental Combo BME280.

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

Per programmare questo sensore è stata utilizzata la libreria sparkfun Qwiic BME280 CCS811 Combo consultabile tramite il link GitHub: [69].

2.5 SparkFun Micro OLED Breakout

Dopo la verifica dei dati ambientali tramite lo SparkFun Cryptographic ATECC508A, li abbiamo fatti visualizzare su un SparkFun Micro OLED Breakout.

È possibile visualizzare il dispositivo in questione nella figura 2.4.

Lo SparkFun Qwiic Micro OLED Breakout è una versione abilitata per Qwiic.

Questo breakout è utile per visualizzare informazioni diagnostiche senza ricorrere a un'uscita seriale, il tutto con l'uso del sistema Qwiic.

Questa versione del micro breakout OLED ha esattamente le stesse dimensioni dello schermo non Qwiic, con uno schermo di 64 pixel di larghezza e 48 pixel di altezza e una dimensione di 0,66".

Ma è stato anche dotato di due connettori Qwiic, che lo rendono ideale per le operazioni I2C.

Sono presenti due fori di montaggio e un comodo supporto per il cavo Qwiic incorporato in una linguetta staccabile sulla scheda, che può essere facilmente rimossa grazie a un bordo con scanalatura a V.

E presente anche un ponticello di pull-up I2C è un ponticello ADDR sul retro della scheda, in modo che se si dispone di pull-up I2C personalizzati o se è necessario modificare l'indirizzo I2C della scheda, si puo procedere senza problemi.

l'indirizzo I2C del Micro OLED è 0x3D ed è selezionabile con un ponticello fino a 0x3C.

Per comunicare con più sensori Micro OLED su un unico bus è necessario un multiplexer/Mux.

Se è necessario utilizzare più di un sensore Micro OLED, si consiglia di utilizzare il Qwiic Mux Breakout.

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

Lo SparkFun Qwiic Micro OLED Breakout è una versione abilitata per Qwiic.

Questo breakout è utile per visualizzare informazioni diagnostiche senza ricorrere a un'uscita seriale, il tutto con l'uso del sistema Qwiic.

Per programmare questo dispositivo è stata utilizzata la libreria SparkFun Micro OLED library consultabile tramite il link GitHub: [\[70\]](#).

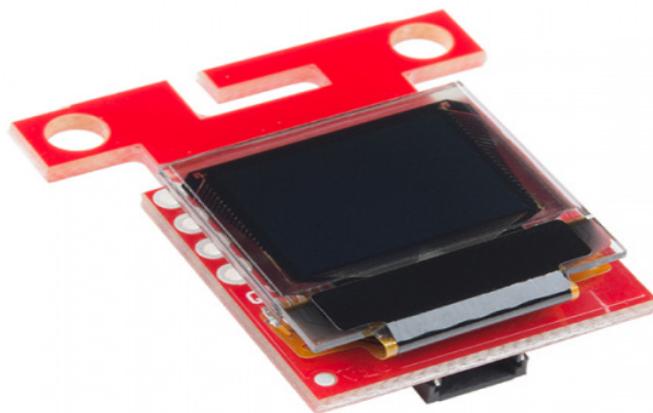


Figura 2.4: Dispositivo SparkFun Micro OLED Breakout.

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

2.6 BLE Cortex-M4F

Nella soluzione proposta per trasmettere i dati dal mittente al destinatario è stato utilizzato il bluetooth low energy che nel nostro caso era integrato nello SparkFun RedBoard Artemis.

È possibile visualizzare il dispositivo in questione nella figura 2.5. Il modulo Artemis di SparkFun è un Cortex-M4F con BLE 5.0 che funziona fino a 96 MHz e con una potenza minima di 6uA per MHz meno di 5m. Il modulo Artemis misura 10x15 mm e contiene tutti i circuiti di supporto necessari per utilizzare il processore Ambiq. Per programmare la parte che riguarda la comunicazione tra mittente e destinatario stata utilizzata la libreria ArduinoBLE.h consultabile tramite il link GitHub: [71].



Figura 2.5: SparkFun Artemis Module BLE Cortex-M4F.

2.7 Camblaggio utilizzato

Per alimentare i due SparkFun RedBoard Artemis abbiamo utilizzato due USB 2.0 Type-C Cable - 1 Meter. Questo cavo è perfetto per l'uso di Raspberry Pi 4 e altre schede USB Type-C. I connettori e i gruppi di cavi USB

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

Type-C offrono un maggiore risparmio di PCB e consentono l'accoppiamento ad alta frequenza in applicazioni di dati, consumer e altre applicazioni di I/O.

Questo cavo utilizza il fattore di forma Type-C ma collega solo i pin per la specifica USB 2.0.

Supporta una velocità di segnalazione massima di 480 Mbit/s (60 MB/s) e una corrente massima di 1,5A (con trasferimento dati) e fino a 5A se utilizzato solo per l'alimentazione/carica.

Questo cavo a 4 fili è lungo 50 mm ed è dotato di connettori JST SH femmina a 4 pin su entrambe le estremità. ha un passo di 1 mm.

Questo cavo può essere utilizzato con le schede Qwiic o STEMMA QT, per collegare facilmente sensori e driver da una scheda all'altra. I cavi sono realizzati in modo simmetrico, quindi puo essere inserito in entrambi i lati. I cavi vengono mostrati nella figura 2.6.



Figura 2.6: Cablaggio utilizzato.

CAPITOLO 2. DISPOSITIVI E STRUMENTI DI SVILUPPO UTILIZZATI

2.8 Tecnologie utilizzate

La stesura del codice e il caricamento sulle schede SparkFun RedBoard Artemis avviene tramite l'IDE di Arduino. Un **IDE** è un ambiente di sviluppo integrato (**Integrated Development Environment**), ovvero un software utile alla realizzazione di codice. In particolare, l'IDE di Arduino permette la stesura del codice e successivo caricamento sulla memoria della scheda. Nella figura 2.7 viene mostrata l'icona dell'Ide di arduino.

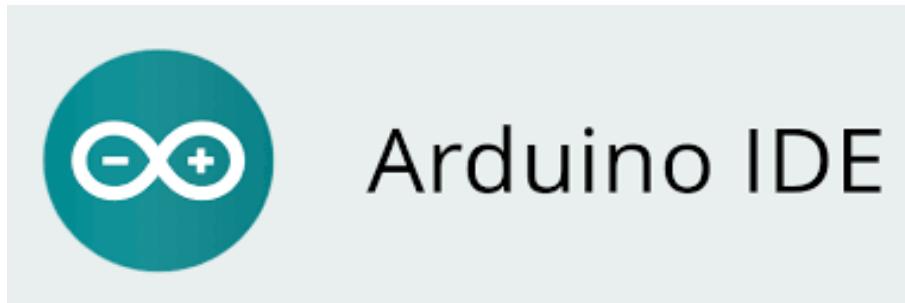


Figura 2.7: Arduino IDE.

Per attivare la visibilità bluetooth e inizializzare la connessione a entrambe le schede SparkFun RedBoard Artemis è stata utilizzata l'applicazione **nRF Connect** per cellulare. Questa è un'applicazione multipiattaforma facile da usare per i test di connettività Bluetooth Low Energy. Supporta il rilevamento automatico dei kit di sviluppo collegati e i caricamenti del firmware, oltre a supportare le funzionalità di sicurezza Bluetooth Low Energy. È possibile cercare dispositivi Bluetooth Low Energy che pubblicizzano e scoprire i loro servizi, mantenere la connessione e i parametri di connessione, accoppiare i dispositivi e modificare la configurazione del server per il tuo dispositivo locale. Nella figura 2.8 viene mostrata l'icona dell'applicazione.



Figura 2.8: Applicazione nRF Connect .

2.9 Architettura sistema

Questa è l'architettura finale del sistema di autenticazione IoT proposto.

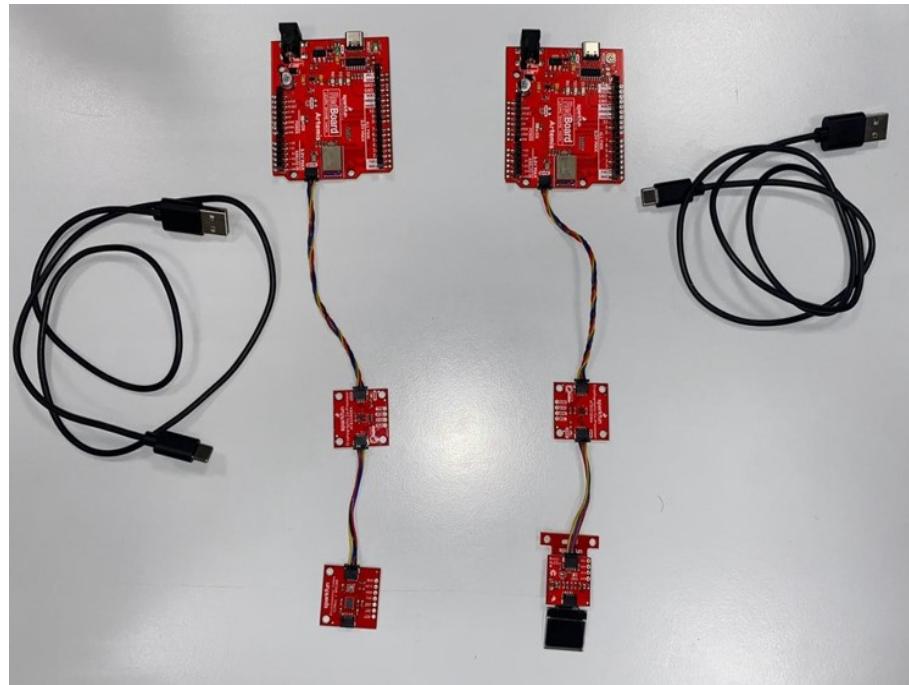


Figura 2.9: Architettura finale del sistema.

Capitolo 3

Implementazione software della soluzione

In questo capitolo parleremo della parte di implementazione software che riguarda il nostro schema di autenticazione IoT.

È stato utilizzato il linguaggio Wiring cioè un linguaggio derivato dal C++. Questo progetto è composto da due programmi. Il primo programma viene denominato Alice(Master), mentre il secondo Bob(Slave).

3.1 Prelevamento, firma, e invio dei dati

In questa sezione andremo a spiegare il codice di Alice.

Alice (master) ha il compito di prelevare i dati tramite il sensore Spark-Fun Environmental Combo Breakout - CCS811/BME280 (Qwiic), generare la chiave pubblica e firmare i dati del sensore tramite Cryptographic Co-Processor ATECC508A e inserirli in tre array diversi.

Successivamente avviare la connessione bluetooth e aspettare che bob si colleghi per l'invio di quest'ultimi.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

3.1.1 Dichiarazioni variabili

Nelle prime quattro righe dello snippet di codice visualizzabile in quest’immagine 3.1, sono presenti tutte le librerie che abbiamo utilizzato per realizzare questo progetto, ognuna di esse e per programmare un componente. Le librerie sono consultabili tramite questi link, **SparkFun Cryptographic ATECC508A**: [68], **SparkFun Environmental Combo BME280**: [69], **ArduinoBLE.h**: [71].

Nelle righe 6 e 7 vengono dichiarate le costanti per effettuare la connessione **bluetooth**.

Nella riga 8 viene dichiarata l’istanza del sensore da cui preleveremo i dati. Nella riga 9 viene dichiarato un **array Environment** in cui andremo a salvare i dati ambientali che provengono dal sensore **BME280**. nella riga 10 invece viene dichiarata l’istanza dello SparkFun Cryptographic ATECC508A, per firmare i dati e generare la **chiave pubblica** successivamente.



```
1 #include <SparkFun_ATECCX08a_Arduino_Library.h>
2 #include "SparkFunBME280.h"
3 #include <ArduinoBLE.h>
4 #include <Wire.h>
5 //dichiarazione costanti connessione bluetooth
6 const char* deviceServiceUuid = "19b10000-e8f2-537e-4f6c-d104768a1214";
7 const char* deviceServiceCharacteristicUuid = "19b10001-e8f2-537e-4f6c-d104768a1214";
8 BME280 mySensor; //Variabile sensore
9 uint8_t Environment[32]; //Array dati Proveniente dal sensore
10 ATECCX08A atecc; //Variabile
```

Figura 3.1: Dichiarazioni e inizializzazioni .

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

3.1.2 Inizializzazione componenti hardware

In Questa sezione di codice 3.2, ci sono le varie inizializzazioni dei compinetti hardware ovvero il sensore bme280 e il microprocessore ATECC508A, ovviamente prima di controllare se i due dispositivi sono collegati correttamente, si setta il **baudrate** a 115000 cioè la velocità in cui vengono trasmessi i byte.

Se le due funzioni di begin restituisco un valore booleano diverso da true allora c'è qualche problema di collegamento dei dispositivi.



```
1 void setup()
2 {
3     Wire.begin();
4     Serial.begin(115000);
5     if (mySensor.beginI2C() == false)
6     {
7         Serial.println("il Sensore BME280 non risponde si prega di controllare il cablaggio");
8         Serial.println("");
9     }
10    Wire.begin();
11    Serial.begin(115000);
12    if(atecc.begin() == true)
13    {
14        Serial.println("Connessione I2C corretta.");
15        Serial.println("");
16    }
17 }
```

Figura 3.2: Inizializzazione componenti hardware.

3.1.3 Connessione

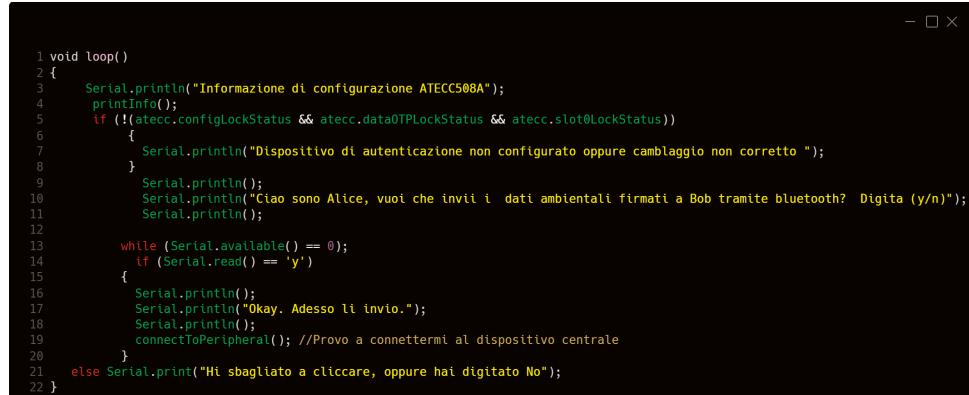
Nella prima parte di questa sezione di codice 3.3 controlliamo se il microprocessore ATECC508A sia configurato correttamente controllando le tre instance **atecc.configLockStatus**, **atecc.dataOTPLockStatus**, **atecc.slot0LockStatus** se quest'ultime restituiscono un valore booleano uguale a true allora la configurazione (che viene fatta in precedenza lanciando un specifico programma di configurazione messo a disposizione dalla casa produttrice del dispositivo) è corretta altrimenti o la configurazione non è stata effettuata oppure è stato configurato in modo incorretto. Oppure ci sono problemi di collegamento.

Nella riga 4 viene chiamata la funzione **printInfo()** nel quale si controllano e si stampano delle infomrazioni che riguardano il microprocessore ATECC508A, inoltre è presente anche la chiamata alla funzione che genera la **chiave pubblica**.3.8

Nella riga 17 chiediamo a l'utente se vuole iniziare il collegamento, con Bob. L'utente se vuole iniziare la comunicazione deve digitare "y" sul terminale. Dopo la digitazione dell'utente verrà chiamata la funzione **connectToPe-**

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

ripheral(), in cui andremo a generare la chiave pubblica e prelevare i dati ambientali tramite una funzione apposita, e firmarli.



```
1 void loop()
2 {
3     Serial.println("Informazione di configurazione ATECC508A");
4     printInfo();
5     if (!(atecc.configLockStatus && atecc.dataOTPLockStatus && atecc.slot0LockStatus))
6     {
7         Serial.println("Dispositivo di autenticazione non configurato oppure camblaggio non corretto ");
8     }
9     Serial.println();
10    Serial.println("Ciao sono Alice, vuoi che invii i dati ambientali firmati a Bob tramite bluetooth? Digita (y/n)");
11    Serial.println();
12
13    while (Serial.available() == 0);
14    if (Serial.read() == 'y')
15    {
16        Serial.println();
17        Serial.println("Okay. Adesso li invio.");
18        Serial.println();
19        connectToPeripheral(); //Provo a connettermi al dispositivo centrale
20    }
21    else Serial.print("Hi sbagliato a cliccare, oppure hai digitato No");
22 }
```

Figura 3.3: Connessione.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

In questa sezione di codice 3.4 siamo riusciti a rilevare bob e facciamo visualizzare sul terminale tutte le sue caratteristiche, ovvero il suo MAC Address e il suo codice uuid.

Dopo aver visualizzato le caratteristiche del dispositivo verra chiamata la funzione **controlPeripheral()** che ci permetterà di generare i dati che ci servono e inviarli alla destinazione (Bob). Nel caso in cui non riusciamo a rilevare il dispositivo, allora verrà visualizzato un messaggio di errore sul terminale.

```
1 if (peripheral)
2 {
3     Serial.println("* Ho rilevato BOB");
4     Serial.print("*Questo è il suo MAC ADDRESS: ");
5     Serial.println(peripheral.address());
6     Serial.println();
7     Serial.print(" Codice UUID: ");
8     Serial.println(peripheral.advertisedServiceUuid());
9     Serial.println();
10    BLE.stopScan();
11    controlPeripheral(peripheral);
12 } else
13     Serial.println("Error Dispositivo(BOB) non Trovato");
14 }
```

Figura 3.4: Dispositivo slave(Bob) rilevato .

3.1.4 Firma e invio dati ambientali

Nella prima parte di questo snippet 3.5 di codice richiamiamo la funzione **EnvironmentalData()**, mostrata nella figura 3.7 che ci permette di prelevare i dati ambientali dall’istanza mysensor e inserirli nell’array Environment tramite l’istanza mysensor.

Successivamente alla riga 4 firmiamo i dati ambientali con la funzione **atecc.createSignature(Environment)** come parametro gli passiamo l’array Environment in cui sono presenti i dati ambientali che abbiamo prelevato tramite la funzione **EnvironmentalData()**. Nella riga 8 tramite l’istanza di connessione bluetooth inviamo i dati ambientali a Bob con la funzione **WriteValue**. le righe successive sono state utilizzate per il debugging.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE



```
1 while (peripheral.connected())
2 {
3     EnvironmentalData(); //Chiamo la funzione per Prendere i dati dal sensore BME280
4     atecc.createSignature(Environment); //firma i dati
5     for(int i=0;i<=3;i++)
6     {
7         Blesend.writeValue(Environment[i]); //invio dati ambientali
8         Serial.print(i); //debug
9         Serial.print(" "); //debug
10        Serial.print("DATI"); //debug
11        Serial.print(" "); //debug
12        Serial.println(Environment[i]); //debug
13    }
14 }
```

Figura 3.5: Firma e invio dati ambientali.

3.1.5 Invio firma

Nella riga 3 di questo snippet di codice 3.6 andiamo ad inviare la firma dei dati al destinatario (Bob) tramite l'istanza di **bluetooth Blesend** con la funzione **writedata** che prende come parametro **atecc.siganture()** cioè la firma dei dati ambientali, prelevati in precedenza tramite la funzione **EnvironmentalData()**.

Nella riga 11 andiamo a disconnettere i dispositivi appena viene inviato l'ultimo byte di dati, in modo da non trasmettere dati non significativi che non appartengono alla firma o ai dati ambientali, se vengono inviati dati non coerenti a quelli originali la verifica che dovrà eseguire il destinatario (Bob) fallirà. Tutte le altre righe di codice sono state utilizzate per il debugging.



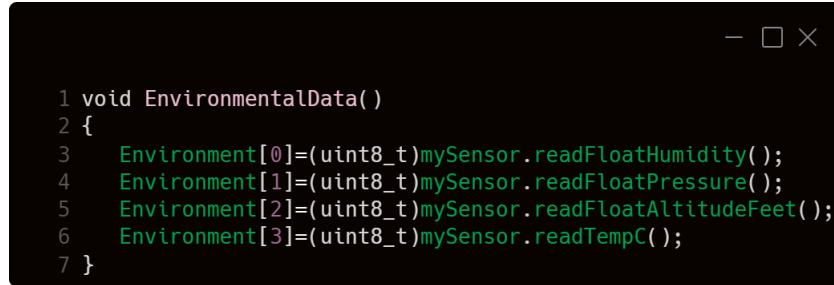
```
1 for (int j = 0; j < sizeof(atecc.signature) ; j++)
2 {
3     Blesend.writeValue(atecc.signature[j]); //invio firma
4     Serial.print(j); //debug
5     Serial.print(" "); //debug
6     Serial.print("FIRMA"); //debug
7     Serial.print(" "); //debug
8     Serial.println(atecc.signature[j]); //debug
9     if (j==64)
10     {
11         peripheral.disconnect(); //disconnettiamo appena invia dati e
12                         //firma in modo da non mandare
13                         //dati che comprometterebbero la firma
14         Serial.println("Avvio e Verifica avvenuti Correttamente,Disconnessione in Corso");
15         Serial.println();
16     }
17 }
18 break;
19 }Serial.println("Disconnessione");
```

Figura 3.6: Firma.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

3.1.6 Ulteriori funzioni utilizzate

Nella figura 3.7 è mostrata la funzione **EnvironmentalData()** che ci permette di prelevare i dati ambientali come **umidità, pressione, altitudine e temperatura**. I dati vengono prelevati tramite l'istanza mysensor e castati in **uint8t**, per permettere la trasmissione bluetooth e poi vengono inseriti nell'array Environment.



```
1 void EnvironmentalData()
2 {
3     Environment[0]=(uint8_t)mySensor.readFloatHumidity();
4     Environment[1]=(uint8_t)mySensor.readFloatPressure();
5     Environment[2]=(uint8_t)mySensor.readFloatAltitudeFeet();
6     Environment[3]=(uint8_t)mySensor.readTempC();
7 }
```

Figura 3.7: Funzione EnvironmentalData().

Nella figura 3.8 è mostrata la funzione che ci permette di generare e stampare sul terminale la chiave pubblica. Dopo aver generato e stampato la chiave pubblica dobbiamo copiare e incollarla nel codice del destinatario(Bob) perchè la chiave pubblica è un paramentro fondamentale per la funzione **atecc.verifySignature** che prende in input i dati ambientali la firma e la chiave pubblica che ci permette di verificare i dati.



```
1 void printAlicesPublicKey()
2 {
3     Serial.println("**Copia/incolla la seguente chiave pubblica (di alice) nella parte iniziale di (bob).**");
4     Serial.println("Bob ne ha bisogno per verificare la sua firma");
5     Serial.println();
6     Serial.println("uint8_t AlicesPublicKey[64] = {");
7     for (int i = 0; i < sizeof(atecc.publicKey64Bytes) ; i++)
8     {
9         Serial.print("0x");
10        if ((atecc.publicKey64Bytes[i] >> 4) == 0) Serial.print("0");
11        Serial.print(atecc.publicKey64Bytes[i], HEX);
12        if (i != 63) Serial.print(", ");
13        if ((63 - i) % 16 == 0) Serial.println();
14    }
15    Serial.println("};");
16    Serial.println();
17 }
```

Figura 3.8: Generazione chiave pubblica

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

Nella figura 3.9 vengono mostrate delle informazioni sullo stato di configurazione del microprocessore ATECC508A. Nella parte finale del codice viene controllato se la chiave pubblica è stata generata correttamente.



```
1 void printInfo()
2 {
3     atecc.readConfigZone(false);
4     Serial.println();
5     Serial.print("Serial Number: \t");
6     for (int i = 0 ; i < 9 ; i++)
7     {
8         if ((atecc.serialNumber[i] >> 4) == 0) Serial.print("0");
9         Serial.print(atecc.serialNumber[i], HEX);
10    }
11   Serial.println();
12   Serial.print("Rev Number: \t");
13   for (int i = 0 ; i < 4 ; i++)
14   {
15       if ((atecc.revisionNumber[i] >> 4) == 0) Serial.print("0");
16       Serial.print(atecc.revisionNumber[i], HEX);
17   }
18   Serial.println();
19   Serial.print("Config Zone: \t");
20   if (atecc.configLockStatus) Serial.println("Locked");
21   else Serial.println("NOT Locked");
22   Serial.print("Data/OTP Zone: \t");
23   if (atecc.dataOTPLockStatus) Serial.println("Locked");
24   else Serial.println("NOT Locked");
25   Serial.print("Data Slot 0: \t");
26   if (atecc.slot0LockStatus) Serial.println("Locked");
27   else Serial.println("NOT Locked");
28   Serial.println();
29   if (atecc.configLockStatus && atecc.dataOTPLockStatus && atecc.slot0LockStatus)
30   {
31       if (atecc.generatePublicKey(0, false) == false)
32       {
33           Serial.println("Errore la chiave pubblica di Alice non è stata generata correttamente");
34           Serial.println();
35       }
36       printAlicesPublicKey();
37   }
38 }
```

Figura 3.9: informazioni

3.2 Ricezione, verifica e visualizzazione dati

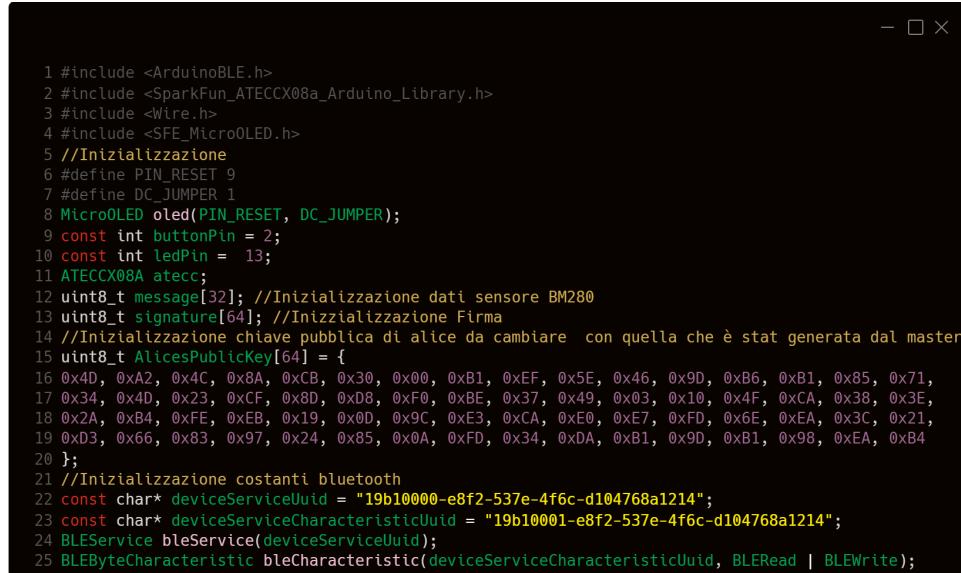
In questa sezione di capitolo parleremo del mittente della nostra architettura ovvero Bob(slave) esso si collega alla connessione bluetooth avviata da Alice, riceve i dati inviati da Alice, li verifica tramite una funzione, se sono corretti e non sono stati manomessi, allora li visualizza a schermo, altrimenti li eliminerà e visualizzera un messaggio di errore.

3.2.1 Inizializzazione variabili

Nella figura 3.10 è illustrato lo snippet di codice che riguarda tutte le inizializzazioni fatte nel programma del destinatario(Bob). Nelle prime 4 righe di codice sono state incluse le librerie utilizzate per programmare tutte le componenti e sono consultabili tramite questi link: **SparkFun Cryptographic ATECC508A:** [68], **SparkFun Environmental Combo BME280:** [69],

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

ArduinoBLE.h: [71] e **SFE microOLED.h** [70]. Nelle parte successiva abbiamo inizializzato lo schermo oled e abbiamo settato la larghezza e la lunghezza per visualizzare il messaggio finale sullo schermo. Nella riga 11 creiamo l’istanza per il microprocessore ATECC508A. Nelle righe 12 e 13 sono dichiarati gli array, in cui andranno i dati ambientali e la firma. Nella riga 15 viene incollata la chiave pubblica generata da dal mittente nel nostro caso Alice, che ci permetterà di verificare i dati. Nella parte finale dello snippet di codice viene inizializzato e settato il bluetooth.



```
1 #include <ArduinoBLE.h>
2 #include <SparkFun_ATECCX08a_Arduino_Library.h>
3 #include <Wire.h>
4 #include <SFE_MicroOLED.h>
5 //Inizializzazione
6 #define PIN_RESET 9
7 #define DC_JUMPER 1
8 MicroOLED oled(PIN_RESET, DC_JUMPER);
9 const int buttonPin = 2;
10 const int ledPin = 13;
11 ATECCX08A atecc;
12 uint8_t message[32]; //Inizializzazione dati sensore BM280
13 uint8_t signature[64]; //Inizializzazione Firma
14 //Inizializzazione chiave pubblica di alice da cambiare con quella che è stata generata dal master
15 uint8_t AlicesPublicKey[64] = {
16 0x4D, 0xA2, 0x4C, 0x8A, 0xCB, 0x30, 0x00, 0xB1, 0xEF, 0x5E, 0x46, 0x9D, 0xB6, 0xB1, 0x85, 0x71,
17 0x34, 0x4D, 0x23, 0xCF, 0x8D, 0xD8, 0xF0, 0xBE, 0x37, 0x49, 0x03, 0x10, 0x4F, 0xCA, 0x38, 0x3E,
18 0x2A, 0xB4, 0xFE, 0xEB, 0x19, 0x0D, 0x9C, 0xE3, 0xCA, 0xE0, 0xE7, 0xFD, 0x6E, 0xEA, 0x3C, 0x21,
19 0xD3, 0x66, 0x83, 0x97, 0x24, 0x85, 0x0A, 0xFD, 0x34, 0xDA, 0xB1, 0x9D, 0xB1, 0x98, 0xEA, 0xB4
20 };
21 //Inizializzazione costanti bluetooth
22 const char* deviceServiceUuid = "19b10000-e8f2-537e-4f6c-d104768a1214";
23 const char* deviceServiceCharacteristicUuid = "19b10001-e8f2-537e-4f6c-d104768a1214";
24 BLEService bleService(deviceServiceUuid);
25 BLEByteCharacteristic bleCharacteristic(deviceServiceCharacteristicUuid, BLERead | BLEWrite);
```

Figura 3.10: Inizializzazioni variabili

3.2.2 Connessione destinatario

Nella riga 2 della figura 3.11, viene controllato se la scheda bluetooth è stata avviata in modo corretto. Nella parte successiva viene avviata la connessione tra Master(Alice) e Slave(Bob) è successivamente lo slave si mette in ascolto sul canale di connessione, aspettando che il Master (Alice) invii i dati.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE



```
1 BLE.begin(); //inizializzazione bluetooth
2 if (!BLE.begin())
3 {
4     Serial.println("Avvio del modulo Bluetooth non riuscito!");
5 }
6     BLE.setLocalName("Artemis Sparkfun Read Board (Dispositivo di periferia)");
7     Serial.println();
8     BLE.setAdvertisedService(bleService);
9     bleService.addCharacteristic(bleCharacteristic);
10    BLE.addService(bleService);
11    BLE.advertise();
12    Serial.println("Artemis Sparkfun Read Board (Dispositivo di periferia)");
13    Serial.println();
14    printInfo();
15 if (!(atecc.configLockStatus && atecc.dataOTPLockStatus && atecc.slot0LockStatus))
16 {
17     Serial.print("Dispositivo di autenticazione non configurato si prega di configurarlo");
18     Serial.println();
19 }
20 Serial.println("Ciao sono Bob, sto ascoltando i messaggi in arrivo da Alice tramite bluetooth");
21 Serial.println();
22
```

Figura 3.11: Connessione destinatario(Bob)

3.2.3 Ricezione della firma e dati ambientali

Nella figura 3.12 viene illustata la sezione di codice del destinatario (Bob) in cui riceviamo i dati ambientali e la firma degli stessi da parte del mittente cioè Alice. Nella riga 5 tramite la funzione **blecharacteristic.written()** andiamo a controllare se ci sono dati sul canale trasmittivo. Nella riga 9 con la funzione **blecharacteristic.value()** preleviamo i dati ambientali e li inseriamo nell'array message, dopo aver inserito tutti e quattro i dati ambientali facciamo una serie di scritture a video dei dati per controllare se la ricezione e l'inserimento sono corretti. Successivamente nella riga 19 sempre con la funzione **blecharacteristic.value()** preleviamo i restanti dati ovvero la firma dei dati e li inseriamo nell'array signature, le restanti scritture a video sono state utilizzate per il debugging.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE



```
1 int i =0;
2         int bytes=0;
3         while (central.connected())
4         {
5             while (bleCharacteristic.written())
6             {
7                 if (i<4)
8                 {
9                     message[i]= bleCharacteristic.value(); //ricezione
10                    Serial.print("DATI "); //Debug
11                    Serial.print(" "); //Debug
12                    Serial.println(message[i]); //Debug
13                }else if ( i < 68)
14                {
15                    Serial.print(i); //Debug
16                    Serial.print(" "); //Debug
17                    Serial.print("FIRMA"); //Debug
18                    Serial.print(" "); //Debug
19                    signature[bytes] = bleCharacteristic.value(); //ricezione
20                    Serial.println(signature[bytes]); //Debug
21                    bytes++;
22                }

```

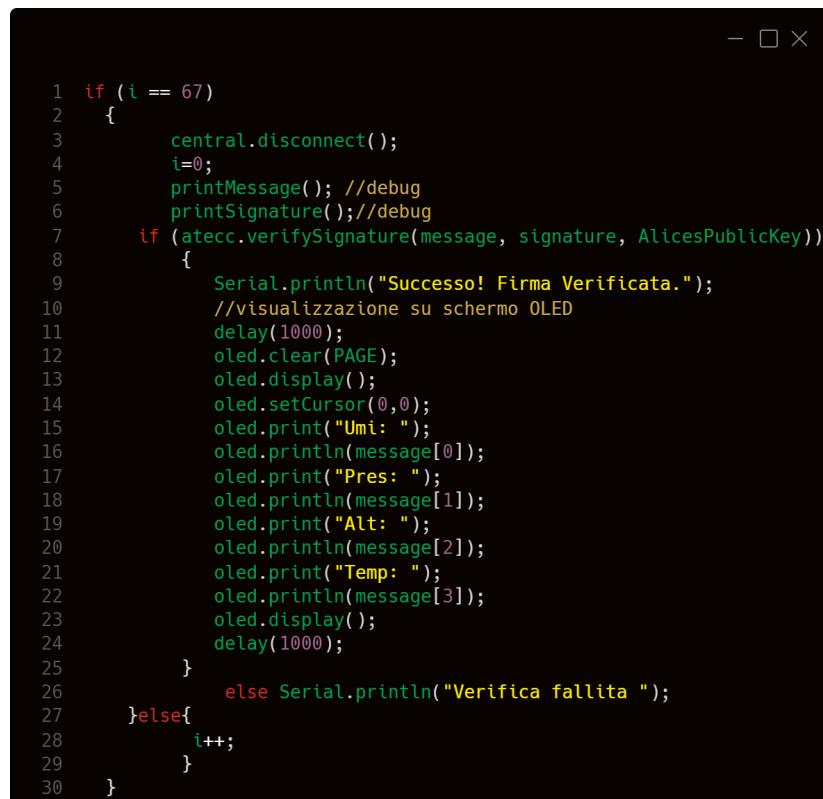
Figura 3.12: Ricezione di dati ambientali e firma.

3.2.4 Verifica e visualizzazione

Nella figura 3.13 è consultabile la sezione di codice in cui viene fatta la verifica dei dati e la visualizzazione di quest'ultimi. La verifica viene effettuata tramite la funzione `atecc.verifySignature()`, questa funzione prende in input tre parametri fondamentali e sono: **dati ambientali, la firma dei dati e la chiave pubblica**.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

Se questa funzione restituisce un booleano di valore true, allora la verifica è andata a buon fine e verrà visualizzato sul terminale un messaggio di verifica effettuata; altrimenti verrà visualizzato un messaggio di errore sul terminale; inoltre nella sezione successiva se la verifica ha restituito un valore booleano uguale a true allora verranno visualizzati i dati ambientali sullo schermo Micro OLED. Le righe di codice che riguardano la visualizzazione sullo schermo sono dalla riga 15 a 22. Il messaggio di errore che verrà visualizzato nel caso in cui la funzione `atecc.verifySignature()` restituirà false e presente nella riga 26.



```
1  if (i == 67)
2  {
3      central.disconnect();
4      i=0;
5      printMessage(); //debug
6      printSignature(); //debug
7      if (atecc.verifySignature(message, signature, AlicesPublicKey))
8      {
9          Serial.println("Successo! Firma Verificata.");
10         //visualizzazione su schermo OLED
11         delay(1000);
12         oled.clear(PAGE);
13         oled.display();
14         oled.setCursor(0,0);
15         oled.print("Umi: ");
16         oled.println(message[0]);
17         oled.print("Pres: ");
18         oled.println(message[1]);
19         oled.print("Alt: ");
20         oled.println(message[2]);
21         oled.print("Temp: ");
22         oled.println(message[3]);
23         oled.display();
24         delay(1000);
25     }
26     else Serial.println("Verifica fallita ");
27 }else{
28     i++;
29 }
30 }
```

Figura 3.13: Verifica dei dati e visualizzazione.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

3.2.5 Esempio di verifica effettuata

Nella figura 3.14 andremo a visualizzare la verificà effettuata con successo. In questo caso la chiava pubblica è comune sia a Bob che a Alice, cioè abbiamo copiato la chiave pubblica generata da alice nel codice di Bob in modo corretto. Nella figura 3.15 è possibile visualizzare anche i dati trasmessi messi sullo schermo dopo la verifica della firma e chiave pubblica.

```
uint8_t message[32] = {  
    0x3F, 0x1A, 0xCA, 0x12, 0x00,  
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00  
};  
  
uint8_t signature[64] = {  
    0x16, 0x4B, 0x49, 0xD6, 0x95, 0xE9, 0xAF, 0xFE, 0x11, 0x00, 0x5B, 0xD7, 0xE3, 0xBF, 0x36, 0x33,  
    0xCC, 0x41, 0x9C, 0x6D, 0x13, 0x7F, 0x5F, 0x88, 0xED, 0x8B, 0x75, 0x6D, 0x92, 0x45, 0x15, 0x0F,  
    0x83, 0x98, 0x68, 0xE0, 0x5B, 0x83, 0x4B, 0x7E, 0x96, 0x80, 0xCC, 0x42, 0xFF, 0x6A, 0x88, 0xE5,  
    0xBF, 0x91, 0xFF, 0x28, 0xD9, 0x58, 0x30, 0xD8, 0x71, 0x42, 0xC9, 0x52, 0x88, 0xC5, 0x53  
};  
  
Successo! Firma Verificata.
```

Figura 3.14: Verifica effettuata con successo.

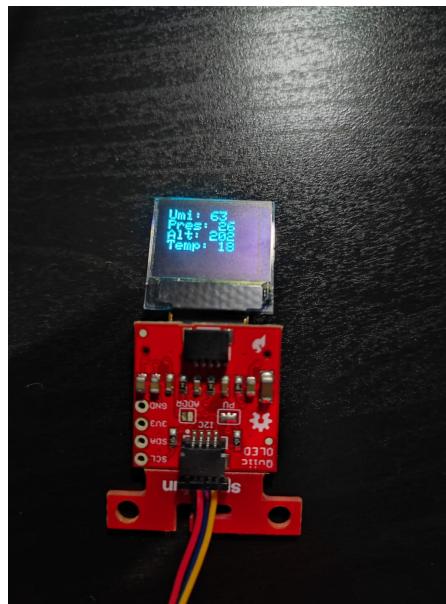


Figura 3.15: Dati ambientali visualizzazione su schermo.

CAPITOLO 3. IMPLEMENTAZIONE SOFTWARE DELLA SOLUZIONE

3.2.6 Esempio di verifica fallita

Nella figura 3.16 è possibile visualizzare un'esecuzione di verifica fallita. Per non effettuare una verificare dei dati corretta abbiamo copiato una chiave pubblica nel codice di Bob non coerente con quella generata da alice. Nella figura 3.17 è possibile visualizzare la chiave pubblica errata inserita nella sezione in alto del codice di Bob. Invece nella figura 3.18 è possibile visualizzare la chiave corretta generata da Alice che doveva aessere incollata nella sezione in alto del codice di Bob.

```
uint8_t message[32] = {  
    0x42, 0x4B, 0xBB, 0x12, 0x00,  
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
};  
  
uint8_t signature[64] = {  
    0xE2, 0xF1, 0x70, 0xE0, 0x9A, 0xCC, 0x98, 0x8E, 0xC2, 0x8B, 0x80, 0x20, 0xBD, 0x66, 0xBF,  
    0xEF, 0x4A, 0xCD, 0xDD, 0x86, 0xD7, 0x8E, 0x90, 0x82, 0x2A, 0x20, 0xBD, 0xEA, 0x39, 0x58,  
    0xC0, 0x8B, 0xC0, 0x60, 0x4A, 0xA9, 0xEB, 0x91, 0x2F, 0xF6, 0x60, 0xD9, 0xC0, 0x55, 0xB7,  
    0x67, 0x4B, 0x21, 0xF3, 0x2D, 0xA0, 0x79, 0x1A, 0x92, 0x44, 0xDB, 0xF5, 0xE5, 0x57,  
};  
  
Verifica fallita
```

Figura 3.16: Verifica fallita.

```
18 //Inizializzazione chiave pubblica di alice da cambiare con quella che è stata generata dal master  
19 uint8_t AlicesPublicKey[64] = {  
20     0x00,  
21     0x00,  
22     0x00,  
23     0x00,  
24 };  
...
```

Figura 3.17: Chiave pubblica errata.

```
uint8_t AlicesPublicKey[64] = {  
    0x4D, 0xA2, 0x4C, 0x8A, 0xCB, 0x30, 0x00, 0xB1, 0xEF, 0x5E, 0x46, 0x9D, 0xB6, 0xB1, 0x85,  
    0x34, 0x4D, 0x23, 0xCF, 0x8D, 0x8B, 0xF0, 0xBE, 0x37, 0x49, 0x03, 0x10, 0x4F, 0xCA, 0x38,  
    0x2A, 0xB4, 0xFE, 0xEB, 0x19, 0x0D, 0x9C, 0xE3, 0xCA, 0xE0, 0xE7, 0xFD, 0x6E, 0xEA, 0x3C,  
    0xD3, 0x66, 0x83, 0x97, 0x24, 0x85, 0xA0, 0xFD, 0x34, 0xDA, 0xB1, 0x9D, 0xB1, 0x98, 0xEA,  
};
```

Figura 3.18: Chiave pubblica corretta.

Capitolo 4

Conclusioni e sviluppi futuri

In conclusione possiamo dire che lo scopo di questo progetto è stato raggiunto con successo. Possibili sviluppi futuri possono essere diversi; potremmo aggiungere una funzionalità di monitoraggio remoto che consente agli utenti di controllare i dati del sensore BME280 da un dispositivo diverso da quello del dispaly micro oled, ad esempio introdurre un interfaccia utente sul web o su un'applicazione mobile. Un'altra miglioria che potremmo applicare è inserire un sistema di allarme per avvisare gli utenti in caso di valori anomali provenienti dal sensore bme280, tramire delle notifiche push oppure con email in tempo reale. Potremmo implementare le funzionalità di salvataggio dei dati del sensore BME280 in un database o in una scheda di memoria per l'archiviazione a lungo termine e l'analisi successiva. Questo potrebbe includere anche la creazione di grafici o report per la visualizzazione dei dati storici. Infine Potremmo utilizzare tecniche di **machine learning** per analizzare i dati del sensore BME280 e rilevare eventuali modelli o anomalie. Questo potrebbe includere la creazione di modelli di previsione o la rilevazione di eventuali deviazioni dai valori di riferimento.

Bibliografia

- [1] Peng Zhang Zheng Yan and Athanasios V Vasilakos. A survey on trust management for internet of things. *journal of network and computer applications.* 42:120–134, 2014.
- [2] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. 1(2011):1–11, 2011.
- [3] Sean Peasley Irfan Saif and Arun Perinkolam. Safeguarding the internet of things: Being secure, vigilant, and resilient in the connected age. 17, 2015..
- [4] Jianying Zhou Rodrigo Roman and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. 7(10):2266–2279, 2013.
- [5] Slaven Marusic Jayavardhana Gubbi, Rajkumar Buyya and Marimuthu Palaniswami. A vision, architectural elements, and future directions. *future generation computer systems.,* 29(7):1645–1660, 2013.
- [6] Valerio Persico Alessio Botta, Walter De Donato and Antonio Pesce. Integration of cloud computing and internet of things: a survey. 56:684–700, 2016.
- [7] Mehdi Mohammadi Mohammed Aledhari Ala Al-Fuqaha, Mohsen Guizani and Moussa Ayyash. A survey on enabling technologies, protocols, and applications. 17(4):2347–2376, 2015.
- [8] Mehdi Mohammadi Mohammed Aledhari Ala Al-Fuqaha, Mohsen Guizani and Moussa Ayyash. A survey on enabling technologies, protocols, and applications. 17(4):2347–2376, 2015.
- [9] Nan Zhang Xinyu Yang Hanlin Zhang Jie Lin, Wei Yu and Wei Zhao.. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. 4(5):1125–1142, 2017.

-
- [10] Mustafa Kocakulak and Ismail Butun. An overview of wireless sensor networks towards internet of things. in 2017 ieee 7th annual computing and communication workshop and conference (ccwc). :, 2017.
 - [11] Taihua Fan Xiaolin Jia, Quanyuan Feng and Quanshui Lei. Rfid technology and its applications in internet of things (iot). :In 2012 2nd international conference on consumer electronics, communications and networks (CECNet), pages 1282– 1285. IEEE, 2012.
 - [12] ZgBee Standards Organization. Zigbee specification. zigbee document 053474r13. :2006.
 - [13] IEEE Standard. Ieee working group et al. wireless medium access control and physical layer specifications for low-rate wireless personal area networks.. :802(4):2003, 2003.
 - [14] Specification of the Bluetooth System. Sig bluetooth. bluetooth core specification version 4.0. :1:7, 2010.
 - [15] Kai Zhao and Lina Ge. A survey on the internet of things security. : 2013.
 - [16] T Geoff Mulligan. The 6lowpan architecture. in proceedings of the 4th workshop on embedded networked sensors. :pages 78–82, 2007.
 - [17] Anders Brandt Jonathan W Hui Richard Kelsey Philip Levis Kris Pister Rene Struik Jean-Philippe Vasseur Roger K Alexander Tim Winter, Pascal Thubert. Ipv6 routing protocol for low-power and lossy networks. : 6550:1–157, 2012.
 - [18] Lorawan 1.1 specification. Lora alliance. : 2017..
 - [19] Saad Harous Abdelhak Bentaleb Yasmine Harbi, Zibouda Aliouat and Allaoua Refoufi. A review of security in internet of things. :Wireless Personal Communications, 108(1):325–344, 2019.
 - [20] Yu Cheng Zequ Yang Yang Zhou :Xianghui Cao, Devu Manikantan Shilla and Jiming Chen. Ghost in zigbee:. : Energy depletion attack on zigbee-based wireless networks. : IEEE Internet of Things Journal, 3(5):816–829, 2016.
 - [21] Salvatore DAntonio Leonid Levy Luigi Coppolino, Valerio DAlessandro and Luigi Romano. :my smart home is under attack. : In 2015 IEEE 18th International Conference on Computational Science and Engineering, pages 145–151. IEEE, 2015.

-
- [22] Zinaida Benenson Christian Müller Philipp Morgner, Stephan Mattejat and Frederik Armknecht. :insecure to the touch: attacking zigbee 3.0 via touchlink commissioning. : In Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pages 230–240, 2017.
 - [23] Mike Ryan. : Bluetooth: With low energy comes low security. : In 7th USENIX Workshop on Offensive Technologies (WOOT 13), 2013. .
 - [24] Andrew Y Lindell. :attacks on the pairing protocol of bluetooth v2. 1. : Black Hat USA, Las Vegas, Nevada, 2008. .
 - [25] Wondimu K Zegeye. :exploiting bluetooth low energy pairing vulnerability in telemedicine. : International Foundation for Telemetering, 2015 .
 - [26] Tomas Rosa. : Bypassing passkey authentication in bluetooth low energy. : IACR Cryptol. ePrint Arch., 2013:309, 2013 .
 - [27] Hao Yang Mengmei Ye, Nan Jiang and Qiben Yan. : Security analysis of internet of-things: A case study of august smart lock. : In 2017 IEEE conference on computer communications workshops (INFOCOM WKSHPS), pages 499–504. IEEE .
 - [28] Hanno Wirtz Martin Henze Hossein Shafagh René Hummen, Jens Hiller and Klaus Wehrle. : 6lowpan fragmentation attacks and mitigation mechanisms. : In Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, pages 55–66, 2013 .
 - [29] A Khannous A Rghiout and M Bouhorma. : Denial-of-service attacks on 6lowpan rpl networks: Issues and practical solutions. : Journal of Advanced Computer Science Technology, 3(2):143–153, 2014. .
 - [30] Remi Badonnel Anthea Mayzaud and Isabelle Chrisment. :a taxonomy of attacks in rpl-based internet of things. : International Journal of Network Security, 18(3):459–473, 2016.
 - [31] Robert Miller. : Lora security: Building a secure lora solution. : MWR Labs Whitepaper, 2016.
 - [32] Christian Doerr Xueying Yang, Evgenios Karapatzakis and Fernando Kuipers. : Security vulnerabilities in lorawan. : . In 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI), pages 129–140. IEEE, 2018.

-
- [33] Edmundo Monteiro Jorge Granjal and Jorge Sá Silva. : Security for the internet of things: a survey of existing protocols and open research issue. : IEEE Communications Surveys & Tutorials, 17(3):1294–1312, 2015.
 - [34] Reem Abdul Rahman and Babar Shah. :security analysis of iot protocols: A focus in coap. :In 2016 3rd MEC international conference on big data and smart city (ICBDSC), pages 1–7. IEEE, 2016.
 - [35] MN Saroja Cynthia, H Parveen Sultana and J Senthil. : Security protocols for iot. :In Ubiquitous computing and computing security of IoT, pages 1–28. Springer, 2019.
 - [36] VL Shivraj Meena Singh, MA Rajan and P Balamuralidhar. : Secure mqtt for internet of things (iot). :In 2015 Fifth International Conference on Communication Systems and Network Technologies, pages 746–751. IEEE, 2015.
 - [37] Osvaldo Fonseca Elverton Fazzion Cristine Hoepers Klaus Steding-Jessen Marcelo HPC Chaves Italo Cunha-Dorgival Guedes Artur Marzano, David Alexander and Wagner Meira. : The evolution of bashlite and mirai iot botnets. :In 2018 IEEE Symposium on Computers and Communications (ISCC), pages 00813–00818. IEEE, 2018.
 - [38] Maziar Fotouhi Md Mahmud Hossain and Ragib Hasan. : Towards an analysis of security issues, challenges, and open problems in the internet of things. :In Services (SERVICES), 2015 IEEE World Congress on, pages 21–28. IEEE, 2015.
 - [39] Ioannis Papapanagiotou Diego M Mendez and Baijian Yang. : Internet of things: Survey on security and privacy. :Internet of things: Survey on security and privacy. arXiv preprint arXiv:1707.01879, 2017.
 - [40] Francesco De Pellegrini Daniele Miorandi, Sabrina Sicari and Imrich Chlamtac. :internet of things: Vision, applications and research challenges. :Internet of things: Vision, applications and research challenges. Ad hoc networks, 10(7):1497–1516, 2012.
 - [41] Mohammad MR Chowdhury Sarfraz Alam and Josef Noll. :interoperability of security-enabled internet of things. Wireless Personal Communications, 61(3):567–586, 2011.
 - [42] Neeli Prasad Jaydip Sen Sachin Babar, Antonietta Stango and Ramjee Prasad. :proposed embedded security framework for internet of

-
- things (iot). In Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on, pages 1–5. IEEE, 2011.
- [43] Rolf H Weber. : Internet of things: Privacy issues revisited. Computer Law Security Review, 31(5):618–627, 2015.
 - [44] Muthucumaru Maheswaran Sridipta Misra and Salman Hashmi. : Security challenges and approaches in internet of things. Security challenges and approaches in internet of things. Springer, 2017.
 - [45] Yacine Challal Arbia Riahi Sfar, Enrico Natalizio and Zied Chtourou. : A roadmap for security challenges in the internet of things. SDigital Communications and Networks, 2017.
 - [46] Antonio Iera Luigi Atzori and Giacomo Morabito. : The internet of things:. A survey. Computer networks, 54(15):2787–2805, 2010.
 - [47] Gregor Leander Deniz Toz Kerem Varıcı Andrey Bogdanov, Miroslav Knežević and Ingrid Verbauwhede. :sponge: A lightweight hash function:. In InternationalWorkshop on Cryptographic Hardware and Embedded Systems, pages 312–325. Springer, 2011.
 - [48] Willi Meier Jean-Philippe Aumasson, Luca Henzen and María Naya-Plasencia. :quark: A lightweight hash. Journal of cryptology, 26(2):313–339, 2013.
 - [49] Chrysostomos Chrysostomou Ioannis Andrea and George Hadjichristofi. :internet of things: Security vulnerabilities and challenges. In Computers and Communication (ISCC), 2015 IEEE Symposium on, pages 180–187. IEEE, 2015.
 - [50] Mohammad Reza Sohizadeh Abyaneh. : Security analysis of lightweight schemes for rfid systems. Ph.D. thesis, The University of Bergen, Norway, 2012.
 - [51] Rachel Greenstadt and Jacob Beal. : Cognitive security for personal devices. In Proceedings of the 1st ACM workshop on Workshop on AISec, pages 27–30. ACM, 2008.
 - [52] Yang Xiao Jing Liu and CL Philip Chen. :internet of things' authentication and access control. International Journal of Security and Networks, 7(4):228–241, 2012.
 - [53] Haider al Khateeb Edewede Oriwoh and Marc Conrad.. :responsibility and nonrepudiation in resource-constrained internet of things scenarios.

International Conference on Computing and Technology Innovation (CTI 2015), 2016.

- [54] Louis Coetze and Johan Eksteen. : The internet of things-promise for the future? an introduction. In IST-Africa Conference Proceedings, 2011, pages 1–9. IEEE, 2011.
- [55] Jerry den Hartog Sandro Etalle and Stephen Marsh. : Trust and punishment. In Proceedings of the 1st international conference on Autonomic computing and communication systems, page 5. ICST (Institute for Computer Sciences, SocialInformatics and Telecommunications Engineering), 2007.
- [56] Yacine Challal Arbia Riahi Sfar, Enrico Natalizio and Zied Chtourou. :a roadmap for security challenges in the internet of things. Digital Communications and Networks, 2017.
- [57] Maurizio A Spirito Prabhakaran Kasinathan, Claudio Pastrone and Mark Vinkovits. : Denial-of-service detection in 6lowpan based internet of things. In Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on, pages 600–607. IEEE, 2013.
- [58] Mehdi Mohammadi Mohammed Aledhari Ala Al-Fuqaha, Mohsen Guizani and Moussa Ayyash. : Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys Tutorials, 17(4):2347–2376, 2015.
- [59] Caifeng Zou Hui Suo, Jiafu Wan and Jianqi Liu. : Security in the internet of things: a review. In Computer Science and Electronics Engineering (ICCSEE), 2012 international conference on, volume 3, pages 648–651. IEEE, 2012.
- [60] SparkFun. : Sparkfun environmental combo breakout - ccs811/bme280 (qwiic). URL: <https://www.sparkfun.com/products/14348>.
- [61] SparkFun. : Sparkfun cryptographic co-processor breakout - atecc508a (qwiic). URL: <https://www.sparkfun.com/products/15573>.
- [62] SparkFun. : Sparkfun redboard artemis. URL: <https://www.sparkfun.com/products/15444>.
- [63] SparkFun. : Sparkfun micro oled breakout (qwiic). URL: <https://www.sparkfun.com/products/14532>.
- [64] David W. Kravitz. :digital signature algorithm (dsa). URL: https://it.wikipedia.org/wiki/Digital_Signature_Algorithm.

-
- [65] Adi Shamir e Leonard Adleman Ronald Rivest. :firma digitale. URL: https://it.wikipedia.org/wiki/Firma_digitale.
 - [66] NIST. :elliptic-curve diffie–hellman (ecdh). URL: https://en.wikipedia.org/wiki/Elliptic-curve_Diffie-Hellman.
 - [67] National Security Agency (NSA) e NIST. :secure hash algorithm. URL: https://it.wikipedia.org/wiki/Secure_Hash_Algorithm.
 - [68] SparkFun. :sparkfun cryptographic. URL:https://github.com/sparkfun/SparkFun_AECCX08a_Arduino_Library.
 - [69] SparkFun. :sparkfun qwiic bme280 ccs811 combo. URL:https://github.com/sparkfun/Qwiic_BME280_CCS811_Combo.
 - [70] SparkFun. :sparkfun micro oled library. URL:https://github.com/sparkfun/SparkFun_Micro_OLED_Arduino_Library.
 - [71] Arduino. :arduinoble.h. URL:<https://github.com/arduino-libraries/ArduinoBLE>.

Elenco delle figure

1.1	.	2
1.2	.	3
1.3	.	5
1.4	.	6
1.5	.	6
1.6	.	7
1.7	.	8
1.8	.	9
1.9	.	9
1.10	.	10
1.11	.	11
1.12	.	12
1.13	.	16
2.1	.	22
2.2	.	24
2.3	.	25
2.4	.	27
2.5	.	28
2.6	.	29
2.7	.	30
2.8	.	30
2.9	.	31
3.1	.	33
3.2	.	34
3.3	.	35
3.4	.	36
3.5	.	37
3.6	.	37
3.7	.	38
3.8	.	38

3.9	.	39
3.10	.	40
3.11	.	41
3.12	.	42
3.13	.	43
3.14	.	44
3.15	.	44
3.16	.	45
3.17	.	45
3.18	.	45

Dediche e ringraziamenti

