

Relazione Progetto Iot

Anna Sacchet
153005

sacchet.anna@spes.uniud.it

Paola Sagliocca
151861

sagliocca.paola@spes.uniud.it

Riccardo Del Chin
151930

delchin.riccardo@spes.uniud.it

Federico De Bona
152254

debona.federico@spes.uniud.it

January 27, 2025

Contents

1 Introduzione	2
2 Obiettivi	3
3 Componentistica fisica essenziale e realizzazione del circuito	3
3.1 WeMos D1 R2	3
3.2 Fototransistor	3
3.3 Il circuito	3
4 Teoria fondamentale	4
4.1 Protocollo MQTT	4
4.2 Paho project	4
4.3 Eclipse Mosquitto	4
4.4 MQTT over TLS	5
5 Architettura MQTT	5
5.1 Client MQTT Publisher	6
5.2 Paho Client MQTT Subscriber	7
5.3 Principi di veridicità e autenticità	8
6 Creazione dell'Artwork in Processing	8
6.1 Processing	8
6.2 Come funziona Processing	9
6.3 Processing e Arduino	9
6.4 Il nostro progetto su Processing	10
7 Tokenizzazione come Non-Fungible Token e pubblicazione nella galleria digitale Async	12
7.1 Blockchain e NFT	12
7.2 Async marketplace	13
8 Conclusioni	15
8.1 Apporto componenti del gruppo	16

1 Introduzione

Con l'avvento del Web 3.0 e la sempre maggiore popolarità della criptovalute, ha suscitato sempre più scalpore l'utilizzo di NFT, ovvero Non-Fungible Token, in grado di subentrare in gran parte delle attività umane odierne, tra cui l'arte. Questi artefatti possono essere assimilati alle criptovalute, se non per la loro mancanza di valore intrinseco, dal momento che possono essere visti essenzialmente come prodotti non fungibili, ovvero senza chiaro utilizzo se non collezionismo o altri scopi pensati appositamente. Questo nuovo mondo ha permesso di integrare concetti come arte, videogiochi e collezionismo con la tecnologia, facendo in modo che la blockchain, la cui natura prevede come concetto principale la decentralizzazione, potesse regolare questi mondi tradizionali, facendo in modo che potessero essere accessibili da qualsiasi parte del mondo con qualche click. Il progetto che viene presentato prevede un idea di arte innovativa, non più generata dall'artista con la tavolozza, ma grazie all'uso di un computer, per poi renderla pubblica grazie all'uso del Web 3.0 e della blockchain. All'interno di questo progetto si vuole andare a sperimentare e far interagire il mondo razionale con quello irrazionale, permettendo a valori di luminosità in questo caso, di sviluppare vere e proprie opere d'arte, rendendo l'arte un concetto matematico e del tutto unico. Inoltre sarà possibile riprodurre queste opere in qualsiasi momento data la loro produzione a partire da valori numerici, dando importanza così alla riproducibilità dell'arte e importanza al modo in cui sono state create le artwork dal sistema realizzato.

Il sistema che è stato progettato fa uso di varie tecnologie, partendo da un circuito elettrico in cui è presente WeMos D1R2, utilizzato prevalentemente per ricevere dati in questo caso da un sensore presente sulla breadboard, nonché la base di partenza per la creazione dell'artwork. Oltre all'uso di WeMos D1R2 con l'IDE Arduino, è stato creato un programma grazie all'uso di Processing IDE per la creazione dell'immagine a partire dai dati ottenuti dalla rilevazione, sfruttando il protocollo di comunicazione MQTTs. Infine, l'immagine generata è stata tokenizzata, rendendola a tutti gli effetti un Non-Fungible Token, pubblicata e presumibilmente venduta presso un NFT marketplace fissato.

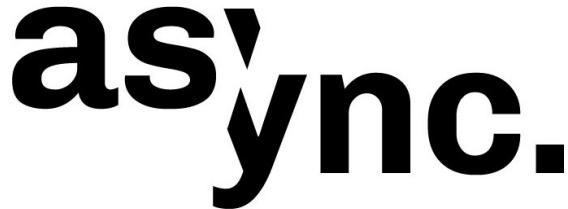


Figure 1: Async, uno dei tanti NFT marketplace presenti per la compra-vendita di NFT online.

L'interazione tra Internet of Things (IoT) e crypto art sta rivoluzionando il modo in cui le opere d'arte digitali vengono create, condivise e scambiate. La crypto art è strettamente legata alla tecnologia blockchain, che garantisce una vendita sicura delle opere digitali e la prova della loro autenticità e valore commerciale. L'IoT, invece, si riferisce alla vasta gamma di oggetti fisici dotati di sensori e software che consentono loro di interagire raccogliendo e scambiando dati tramite una rete. Quando l'IoT e la crypto art si incontrano, emergono nuove possibilità per gli artisti e gli appassionati d'arte. Ad esempio la creazione di opere d'arte interattive, la possibilità di tracciare la storia e la proprietà di un'opera d'arte digitale nel tempo, facilitare lo scambio e la compravendita di opere d'arte digitali oppure personalizzare delle opere d'arte in base alle preferenze dell'utente o alle condizioni ambientali. Tuttavia, l'interazione tra IoT e crypto art solleva anche alcune preoccupazioni in termini di sicurezza e privacy. Ad esempio, la sicurezza dei dispositivi IoT e dei dati trasmessi tra di essi è fondamentale per proteggere le opere d'arte e le informazioni sensibili degli utenti. Inoltre, la raccolta e l'utilizzo di dati personali e ambientali possono sollevare questioni etiche e di privacy. E' importante quindi lavorare per sviluppare soluzioni che garantiscano un equilibrio tra innovazione e protezione dei dati sensibili degli utenti.

2 Obiettivi

La realizzazione di questo progetto vuole andare a perseguire determinati obiettivi, che vengono descritti di seguito:

1. La realizzazione di un prodotto finito grazie all'utilizzo di un circuito elettrico che, sfruttando WeMos D1R2 e vari sensori (in questo caso luminosità, ma integrabile anche con temperatura, umidità ecc.) permette di realizzare delle opere d'arte, ognuna diversa dall'altra, in maniera tale da automatizzare la creazione di dipinti per vari scopi, tra cui la vendita attraverso la sua Tokenizzazione e pubblicazione su un NFT Marketplace.
2. Un altro obiettivo importante è la progettazione e creazione di un sistema Publish-Subscribe grazie al quale è possibile garantire la comunicazione e uso di un programma di generazione di immagini sfruttando il linguaggio di programmazione Java con Processing IDE, operanti tramite un server centrale e un protocollo di messaggistica per la comunicazione tra dispositivi prevalentemente di IoT (Internet of Things) chiamato MQTT (Message Queuing Telemetry Transport).
3. Oltre alla comunicazione dichiarata al punto precedente, si vuole andare ad integrare il protocollo che sfrutta una comunicazione su linea non sicura con vari strati di sicurezza, tra cui autenticazione con credenziali e comunicazione tramite SSL (Secure Socket Layer), in maniera tale da avere un prodotto finito che può essere utilizzato senza che terze parti non autorizzate riescano ad interagire con il sistema, manipolando l'algoritmo di creazione delle opere d'arte.
4. Realizzazione di un NFT, sfruttando le principali tecniche di Tokenizzazione disponibili sul Web 3.0, partendo dall'immagine generata grazie al sistema progettato, fino ad arrivare alla sua pubblicazione su un NFT marketplace interagendo con il mondo degli NFT e la blockchain utilizzata.
5. Fornire all'utente finale la possibilità, grazie alla creazione di un database MySQL, di aver a disposizione i dati di creazione dell'immagine da esso scelta. Questo va a garantire la tracciabilità dei dati utilizzati.

3 Componentistica fisica essenziale e realizzazione del circuito

3.1 WeMos D1 R2

Il WeMos-D1R2 è un'unità di elaborazione basata sull'ESP8266-12 con funzionalità Wi-Fi integrate e un layout simile ad Arduino-UNO. Ciò significa che la scheda funziona, nella maggior parte dei casi, come un Arduino UNO, con il vantaggio aggiunto del Wi-Fi integrato. Molti shield, sensori e dispositivi di output progettati per la piattaforma Arduino possono essere utilizzati con il WeMos-D1R2. Per programmare il WeMos-D1R2, abbiamo utilizzato l'IDE di Arduino. Dopo aver installato il supporto per la scheda, è possibile utilizzare le funzioni e le librerie di Arduino per scrivere sketch e caricarli direttamente sull'ESP8266.

3.2 Fototransistor

Un fototransistor è un dispositivo semiconduttore a due o tre terminali che converte l'energia luminosa in corrente o tensione elettrica. Funziona in modo simile a un fotodiode, ma ha un fattore di amplificazione, infatti amplifica la corrente tra i suoi terminali utilizzando l'intensità della luce risultando in questo modo più sensibile di un fotodiode. Il fototransistor da noi utilizzato prende in input un range di valori che va da 0 a 1024. Il principio del progetto presuppone che un qualsiasi sensore possa essere integrato e utilizzato affinché venga aggiunta una nuova componente creativa e artistica al progetto, un esempio può essere l'integrazione di un sensore DHT11 di temperatura e umidità, per fare in modo di rendere la crypto-art più complessa e ricca di significato con ulteriori dati percepiti dai sensori.

3.3 Il circuito

Per la realizzazione di questo progetto è stato progettato un semplice circuito utilizzando alcune componentistiche. E' presente un fototransistor, una resistenza da 10K, collegati al WeMos D1R2 con alimentazione 3.3V.

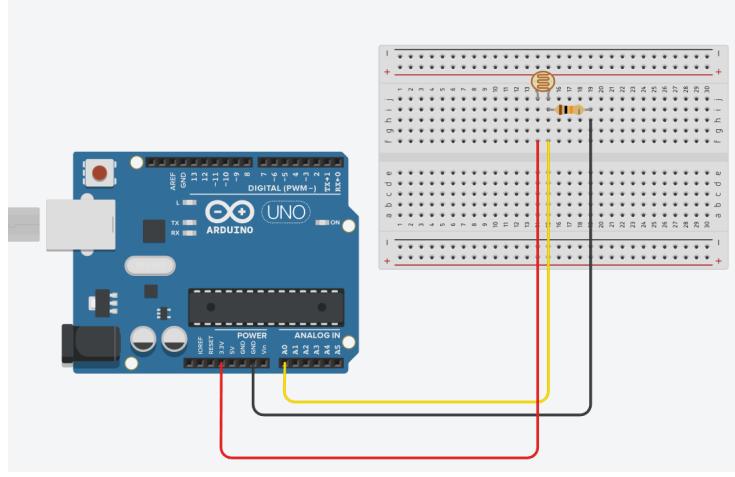


Figure 2: Schema del circuito elettronico realizzato.

4 Teoria fondamentale

4.1 Protocollo MQTT

MQTT (Message Queuing Telemetry Transport) è un protocollo di messaggistica basato sul modello publish/subscribe che lavora sopra il protocollo TCP/IP. Esso è progettato per essere leggero e adatto alla comunicazione tra dispositivi con risorse limitate, come dispositivi IoT. MQTT definisce due tipi di entità di rete: un broker di messaggi e un certo numero di client. Un broker MQTT è un server che riceve tutti i messaggi dai client e poi instrada i messaggi ai client di destinazione appropriati. I client MQTT sono dispositivi che eseguono una libreria MQTT e si connettono a un broker MQTT su una rete. Le informazioni sono organizzate in una gerarchia di argomenti (topics). Quando un publisher ha un nuovo elemento di dati da distribuire, invia un messaggio di controllo con i dati al broker connesso. Il broker poi distribuisce le informazioni a tutti i client che si sono sottoscritti a quell'argomento. I topic in MQTT sono uno dei principali punti di progettazione del protocollo. I topic non sono pre-registrati e possono essere creati al volo dai publisher.

4.2 Paho project

Il progetto Paho per MQTT Java è una libreria open-source che supporta l'implementazione del protocollo MQTT in applicazioni Java e quindi perfetto nel caso dell'utilizzo di Processing come in questo progetto. Questa libreria offre una semplice API per pubblicare messaggi e iscriversi a vari topic secondo il modello MQTT supportando sia il livello di QoS zero, che quello di livello uno, permettendo quindi di garantire una consegna affidabile dei messaggi in casi di rete instabile. Dal momento che in questo progetto si vuole permettere lo scambio di messaggi mediante l'uso di TLS, questa libreria offre numerosi funzioni anche per questo scopo, consentendo di adattare l'utilizzo del protocollo in maniera facile e specifico all'esigenza dell'applicazione. Essenzialmente questa libreria è stata utilizzata per permettere al programma, scritto in Java con Processing, che si occupa della creazione delle immagini, di poter iscriversi per esempio al topic della luminosità e ricevere i dati relativi alla luminosità percepita dal sensore collegato ad WeMos D1R2 come punto di partenza per la creazione delle immagini.

4.3 Eclipse Mosquitto

Per la realizzazione del broker MQTT è stato necessario installare Eclipse Mosquitto, un broker MQTT open-source ampiamente utilizzato che permette di fruire del protocollo di messaggistica propriamente utilizzato su dispositivi di Internet of Things e altre applicazioni di comunicazione di questo tipo. Per fare in maniera tale di garantire la sicurezza e la proprietà del sistema sviluppato, il broker è stato interamente implementato su una macchina proprietaria Oracle grazie all'uso del sistema operativo Linux Ubuntu. Per l'installazione del broker sono stati necessari alcuni passaggi seguiti però da un'attenta configurazione del broker, in modo tale che prevedesse essenzialmente la comunicazione su

linea TLS e l'autenticazione dei client mediante un file predefinito in cui custodire tutte le credenziali; il fatto che la macchina fosse di proprietà, ha permesso di custodire in maniera sicura e senza condividere a terzi i file delle credenziali e fare in modo che solamente gli autorizzati potessero accedere ai file sensibili del broker MQTT. Una volta correttamente configurato sono stati utilizzati i comandi mosquitto_sub e mosquitto_pub per, rispettivamente, testare il funzionamento della sottoscrizione ad un topic e la pubblicazione di un messaggio all'interno di un topic; non appena verificato il corretto funzionamento è stato possibile proseguire con la comunicazione vera e propria tra i client previsti.

4.4 MQTT over TLS

MQTTS (MQTT over TLS) è una variante del protocollo MQTT che introduce una sicurezza maggiore mediante il protocollo TLS (Transport Layer Security). Essa estende MQTT con una crittografia end-to-end e un'autenticazione per garantire la privacy, l'integrità e l'autenticità dei dati trasmessi; dal momento che si tratta di comunicazione tra dispositivi di IoT, è sempre necessario mettere al primo posto la sicurezza per evitare spiacevoli inconvenienti. Dal momento che il protocollo MQTT funziona tramite un broker MQTT, lo stesso vale per il protocollo MQTTS, con la differenza che la comunicazione avviene mediante il protocollo TLS per creare una connessione sicura tra client e broker. Questo previene attacchi come Man-In-The-Middle prevenendo l'intercettazione dei dati da terze parti e garantendo la privatezza dei dati scambiati durante la comunicazione; inoltre, tramite questo protocollo l'autenticazione che fornisce permette al client di verificare l'identità del broker, assicurandosi che sia realmente il broker corretto e non una terza parte che vuole minacciare il sistema. Questo meccanismo viene utilizzato prevalentemente nei casi in cui i dati scambiati sono realmente sensibili, soprattutto in ambito di domotica e Internet of Things, come in questo caso in cui non si vuole fare in modo che le opere generate vengano sniffate da altre persone o broker non autorizzati. Il meccanismo di sicurezza che è stato realizzato grazie all'implementazione di MQTTS è stato quello di garantire l'autenticità del server a cui i dispositivi client comunicano; infatti, durante l'handshake TLS il server risponde client hello presentando il suo certificato, che sarà confermato da una certificate authority, in maniera tale da verificare l'affidabilità di tale certificato; nel nostro caso i certificati sono stati generati appositamente per il test, utilizzando CertBot ovvero uno strumento open-source in grado di generare certificati di una certificate authority denominata Let's Encrypt, che fornisce certificati TLS a moltissimi siti in maniera gratuita per scopi di questo tipo. Una volta creato il certificato per il broker, è necessario che ogni client, ogni volta che stabilisce una connessione, abbia a disposizione tale certificato firmato dalla certificate authority per confrontarlo con quello presentato dal broker MQTTS: nel caso in cui non fosse lo stesso, ci potrebbe essere una terza parte che vuole manomettere il sistema, in questo caso il nostro applicativo sarebbe protetto. Nell'altro senso, è stata implementato un altro tipo di meccanismo di sicurezza, questa volta relativo ai client che effettuano le connessioni al broker e riguarda l'autenticazione. Infatti, è possibile che ci siano dei client che sono risaliti al certificato del broker e che possono intromettersi nel sistema; per evitare questo tipo di intrusione è stata introdotta l'autenticazione mediante username e password, in maniera tale da permettere ai soli client autorizzati di accedere e connettersi al broker per il corretto funzionamento del sistema.

5 Architettura MQTT

Come specificato in precedenza, il protocollo MQTTS realizzato in questo progetto, prevede la presenza di due client ed un broker MQTT in cui il client Publisher rappresentato dal circuito con Arduino provvede a pubblicare ogni quanto di tempo le informazioni relative alla luminosità all'interno di un apposito topic tramite il broker MQTT. Una volta che questo client riesce a inviare correttamente i messaggi inerenti alla luminosità rilevata sul fotoresistore, c'è un altro client che si occupa esclusivamente di iscriversi al topic della luminosità tramite il broker MQTT, ricevere i dati inviati dall'altro client e usarli per creare le varie opere digitali mediante l'uso di Processing e di un programma realizzato interamente in Java, sfruttando numerose librerie. Viene di seguito descritto brevemente il funzionamento sia del Client MQTT Publisher sviluppato interamente in Arduino e il Client MQTT Subscriber tramite il progetto Paho e grazie all'uso di Java e Processing.

5.1 Client MQTT Publisher

Per la realizzazione del client MQTT Publisher è stato utilizzato l'IDE di Arduino che sfrutta il linguaggio di programmazione Wiring (derivato da C++) per la compilazione e il caricamento del programma in memoria della scheda. Il funzionamento ad alto livello del codice è semplice e riguarda in primo luogo il setup della rete WiFi, a cui Arduino si connette tramite il modulo ESP8266, successivamente la connessione al broker MQTTs con il precedente caricamento del certificato del broker firmato dall'apposita Certificate Authority e l'autenticazione con username e password; infine si verifica il vero e proprio invio dei messaggi sul corrispondente topic, solamente una volta che la connessione al broker è avvenuta con successo. Per il setup del WiFi è stato necessario l'utilizzo della libreria ESP8266WiFi, che ha permesso di stabilire la connessione alla rete WiFi utilizzata mediante il componente ESP8266, fornendo credenziali di accesso e tutto l'occorrente per stabilire la connessione in maniera corretta. Una volta impostata correttamente la connessione è stato possibile procedere con il tentativo di connessione al broker MQTTs. Una fase fondamentale all'interno della funzione di setup, in cui vengono impostate tutte le variabili che vengono utilizzate dentro la funzione di loop, è quella di impostare il certificato firmato dalla Certificate Authority tramite l'oggetto che poi servirà per collegarsi al broker tramite TLS; dopo la procedura svolta per la creazione dei certificati per il test di questo servizio, è stato caricato nel programma tramite la variabile ca_cert ed è stato impostato mediante la funzione setTrustAnchors dell'oggetto utilizzato per la connessione MQTTs. Sempre nella fase di setup è stato impostato il server a cui collegarsi, tramite la porta 8883 prevista per le connessioni over TLS e una funzione di callback, nel caso di ricevuta di messaggi dal broker MQTTs una volta collegato. Infine, la funzione setup_wifi() permette di configurare tutte le impostazioni necessarie affinché si possa verificare il collegamento al WiFi utilizzato per connettersi ad Internet.

```
void setup() {
    pinMode(BUILTIN_LED, OUTPUT);
    Serial.begin(115200);
    BearSSL::X509List *serverTrustedCA = new BearSSL::X509List(ca_cert);
    espClient.setTrustAnchors(serverTrustedCA);
    setup_wifi();
    setClock();
    client.setServer(mqtt_server, 8883);
    client.setCallback(callback);
}
```

Figure 3: La funzione di setup() descritta.

Successivamente una volta stabilita la connessione al broker MQTTs parte la fase all'interno della funzione di loop all'interno della quale, ogni due secondi, avviene una misurazione del valore di luminosità tramite il circuito specificato nell'apposita sezione e, una volta inserito nell'apposito buffer per la trasmissione del messaggio con il protocollo MQTTs viene inviato al broker, in maniera tale che successivamente, coloro iscritti al topic predefinito, possano ricevere il dati inviati e utilizzarli.

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    unsigned long now = millis();
    if (now - lastMsg > 2000) {
        lastMsg = now;
        ++value;
        char buffer[10];
        itoa(analogRead(A0), buffer, 10);

        sprintf(msg, MSG_BUFFER_SIZE, buffer, value);
        Serial.print("Publish message Lum: ");
        Serial.println(msg);
        client.publish(topiclum, msg);
    }
}
```

Figure 4: La funzione loop() descritta.

5.2 Paho Client MQTT Subscriber

Un client subscriber è un componente chiave per ricezione di messaggi MQTT su uno o più argomenti (topic) specifici. È stato implementato quindi un Paho Client subscriber MQTT in Processing, una piattaforma di sviluppo per creare applicazioni visive e interattive, in linguaggio Java. Il client MQTT si connette al broker sulla porta 8883 e si sottoscrive ai nostri topic d'interesse. Inoltre, viene monitorata la connessione alla rete WiFi. Quando il Client riceve un messaggio, il metodo messageArrived() viene richiamato e il messaggio viene elaborato.

È importante sottolineare il modo in cui vengono utilizzate le librerie all'interno del Paho Client: ce ne sono alcune relative all'introduzione della sicurezza per gestire il certificato del server firmato dalla Certificate Authority e l'autenticazione durante la connessione mediante l'username e la password; l'altra principale libreria è quella del paho client rilasciata da Eclipse per la connessione al broker MQTT e successivamente per la ricezione dei messaggi dal topic della luminosità.

Per prima cosa nel metodo di setup avviene la connessione al broker MQTTs, essendo un programma su una macchina già connessa ad internet non c'è la gestione della connessione al WiFi; la funzione dell'oggetto MqttClient chiamata connect prevede di passare un parametro come argomento riguardante le opzioni di connessione che in questo caso include l'oggetto socketFactory che si occupa di creare la socket over TLS per la comunicazione che verrà spiegato successivamente, le credenziali per l'autenticazione con username e password e la funzione da eseguire nel momento in cui c'è un messaggio disponibile sul topic che si occupa di leggerlo e salvarlo nella variabile relativa alla luminosità.

```
try {
    client = new MqttClient(broker, clientId, persistence);

    MqttConnectOptions connOpts = new MqttConnectOptions();

    SSLSocketFactory socketFactory = getSocketFactory();

    connOpts.setSocketFactory(socketFactory);

    connOpts.setCleanSession(true);

    connOpts.setUserName(mqttUser);
    connOpts.setPassword(mqttPassword.toCharArray());

    client.setCallback(new SimpleMqttCallback());
    client.connect(connOpts);

    println("MQTT Connected");
    client.subscribe("153005/lum");
} catch (MqttException e) {
    e.printStackTrace();
    print("CONNESSIONE AL BROKER FALLITA!!!");
}
```

Figure 5: funzione di setup() riguardante la connessione al broker MQTTs.

La creazione del socket over TLS avviene mediante un metodo factory che permette di partire dal certificato radice in formato PEM della Certificate Authority che certifica il broker MQTTs, che è stato creato appositamente mediante CertBot caricandolo mediante le varie librerie previste da Java per la manipolazione ed elaborazione di certificati e chiavi; una volta fatto ciò viene creato un KeyStore vuoto, ovvero una repository dove salvare e mantenere informazioni riservate di questo tipo, necessario affinché si possa creare una socket SSL che utilizzi tale certificato una volta effettuato il tentativo di connessione. Infatti, appena dopo viene creato un TrustManager che utilizza il KeyStore con il certificato inserito appositamente: esso permette di confrontare le informazioni presenti nel KeyStore con quelle del server, in maniera tale da consentire la connessione nel caso in cui esse coincidano. Il passo finale della funzione è restituire il contesto SSL realizzato per essere passato come parametro alle opzioni di connessione al broker MQTTs; in questo modo viene realizzata una connessione TLS con il broker, garantendo autenticazione del server e, tramite le apposite opzioni, anche l'autenticazione da parte del client tramite credenziali. Di seguito viene presentato il codice della funzione, che tramite qualche commento riesce ad essere maggiormente esplicativo.

```

private static SSLSocketFactory getSocketFactory() {
    try {
        // Carica il certificato radice dal formato PEM
        CertificateFactory cf = CertificateFactory.getInstance("X.509");
        InputStream caInput = new ByteArrayInputStream(rootCACert.getBytes());
        X509Certificate ca = (X509Certificate) cf.generateCertificate(caInput);

        // Crea un KeyStore vuoto e carica il certificato radice (CA) in esso
        KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
        keyStore.load(null, null);
        keyStore.setCertificateEntry("ca", ca);

        // Crea un TrustManagerFactory che utilizzerà il KeyStore con il certificato radice
        TrustManagerFactory tmf = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        tmf.init(keyStore);

        // Crea un contesto SSL personalizzato che utilizzerà il TrustManagerFactory
        SSLContext sslContext = SSLContext.getInstance("TLS");
        sslContext.init(null, tmf.getTrustManagers(), null);
        // Restituisce l'oggetto SSLSocketFactory
        return sslContext.getSocketFactory();
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

```

Figure 6: funzione getSocketFactory() che permette di realizzare il contesto SSL da utilizzare durante la connessione con il broker MQTTs dal client Paho.

Inoltre, è presente la connessione al database e l'inserimento dei dati relativi alla creazione dell'immagine mediante le librerie java.sql e il driver JDBC. In questo modo vengono salvati i dati di creazione abbinati all'identificativo dell'immagine in questione e il suo timestamp.

5.3 Principi di veridicità e autenticità

Altri principi fondamentali da rispettare per poter definire sicuro un sistema sono sicuramente:

- **Veridicità:** si riferisce alla correttezza, all'integrità e all'affidabilità delle informazioni o dei dati. Indica se le informazioni sono accurate, complete e non alterate. La veridicità implica che le informazioni non siano state manipolate, corrotte o compromesse in alcun modo. Nella sicurezza delle informazioni, garantire la veridicità significa adottare misure per proteggere i dati da modifiche non autorizzate o corruzione e per garantire che le informazioni siano affidabili.
- **Autenticità:** si riferisce alla prova dell'identità o dell'origine di un'entità, di un'informazione o di un'azione. Riguarda la conferma che qualcosa sia ciò che dichiara di essere, che un'entità sia effettivamente chi afferma di essere o che un'azione sia stata effettuata da un soggetto autorizzato. Nella sicurezza, l'autenticità è cruciale per garantire che le persone, i dispositivi, i dati o le transazioni siano autentiche e non falsificate o manipolate

Per soddisfare questi requisiti, la scelta è stata di implementare un database in cui vengono registrati i dati di composizione per ciascun NFT, cioè un identificatore unico universale (UUID), i vari livelli di luminosità registrati dal sensore e, infine, un timestamp per avere un'informazione temporale aggiuntiva. In pratica, ogni volta che viene creato un nuovo NFT, i dati di composizione vengono registrati nel database per fornire un punto di riferimento affidabile in modo da aiutare a verificare, da parte di un possibile acquirente, la veridicità delle informazioni associate a un NFT e a garantire che l'immagine o il contenuto conservato nel token siano autentici.

6 Creazione dell'Artwork in Processing

6.1 Processing

Processing è un ambiente di sviluppo integrato (IDE) open-source compatibile con i sistemi operativi Linux, macOS e Microsoft Windows, che combina un linguaggio di programmazione basato su Java con una grafica interattiva. Creato da Ben Fry e Casey Reas presso il MIT Media Lab nel 2001, Processing è stato sviluppato con l'obiettivo di consentire a artisti, designer e programmatore di creare

facilmente opere d'arte e applicazioni visive interattive. Essendo dotato di una vasta collezione di librerie che estendono le sue funzionalità di base, offrendo strumenti aggiuntivi per la grafica 3D, il suono, il networking e molto altro ancora, consente agli sviluppatori di creare applicazioni complesse e coinvolgenti, come ad esempio:

- Opere d'arte generative: artisti di tutto il mondo utilizzano Processing per creare opere d'arte uniche, basate su algoritmi e logica computazionale. Le opere d'arte generative si creano spesso attraverso la combinazione di forme, colori e pattern, generati in tempo reale dal codice.
- Visualizzazioni dati: Processing può essere utilizzato per creare visualizzazioni interattive dei dati, che consentono di esplorare e comprendere meglio le informazioni complesse. Queste visualizzazioni possono includere grafici, mappe, diagrammi e animazioni che rappresentano i dati in modo coinvolgente e accessibile.
- Installazioni interattive: molte opere di arte digitale o installazioni interattive utilizzano Processing per creare esperienze coinvolgenti. Attraverso l'uso di sensori, telecamere o altri dispositivi di input, queste installazioni permettono al pubblico di interagire con l'opera d'arte e di influenzarne il comportamento in tempo reale.
- Animazioni e filmati: Processing offre strumenti per creare animazioni e filmati. È possibile creare sequenze di immagini, applicare effetti visivi, creare effetti di transizione e combinare animazioni con suoni e interazioni.

6.2 Come funziona Processing

La sintassi di Processing, basata sul linguaggio di programmazione orientato agli oggetti Java, ma arricchita di funzionalità per la gestione semplificata di aspetti grafici e multimediali, permette di organizzare il codice in blocchi logici, rendendone più semplice la comprensione e la modifica.

In particolare le funzioni principali di Processing sono due: “setup()” e “draw()”.

Quando il programma viene avviato, infatti, la prima funzione che viene chiamata automaticamente è la funzione “setup()”, in cui vengono definite le proprietà iniziali dell'applicazione come dimensione e sfondo, oppure eseguite semplici operazioni come il caricamento di immagini etc...

A seguire vi è l'esecuzione della funzione “draw()”, in cui si andranno a definire gli elementi grafici desiderati, anche attraverso una vasta gamma di funzioni per il disegno grafico come line(), rect(), ellipse() e text() per creare disegni su un canvas grafico. Questa viene eseguita continuamente in un ciclo fino a che il programma non termina o c'è una chiamata noLoop() e può venire richiamata indirettamente tramite “redraw()” oppure “loop()”.

6.3 Processing e Arduino

Collegare Arduino a Processing apre molte potenzialità creative e pratiche. Ecco alcuni dei vantaggi e delle potenzialità di questa combinazione:

- Controllo di dispositivi esterni: Arduino è una scheda di prototipazione elettronica che può essere utilizzata per controllare una vasta gamma di dispositivi fisici come motori, sensori, LED e attuatori. Con la comunicazione tra Arduino e Processing, si può inviare comandi da Processing alla scheda Arduino per controllare questi dispositivi in tempo reale.
- Visualizzazione interattiva dei dati: Arduino può essere utilizzato per raccogliere dati da sensori e inviarli a Processing per elaborazione e visualizzazione. Ad esempio, si possono collegare sensori di temperatura, umidità o movimento ad Arduino e utilizzare Processing per creare grafici, visualizzazioni o animazioni interattive basate su questi dati.
- Interazione con il computer: Processing offre potenti strumenti per l'interazione con il computer, come la grafica 2D/3D, l'elaborazione dell'immagine e il riconoscimento del suono. Collegando Arduino a Processing, è possibile creare interfacce utente innovative e interattive che combinano l'input da sensori fisici con la potenza di elaborazione del computer.

- Creazione di installazioni artistiche e interattive: La combinazione di Arduino e Processing consente la realizzazione di installazioni artistiche e interattive complesse. Si può utilizzare Arduino per controllare gli aspetti fisici dell'installazione, come luci, suoni o movimenti, mentre Processing gestisce l'interazione con il pubblico e la generazione di contenuti visivi o sonori.

6.4 Il nostro progetto su Processing



In questa fase del progetto lo scopo ultimo era quello di mettere in contatto il mondo dell'Iot con l'ambiente di sviluppo Processing e dare, così, origine a una rappresentazione artistica di alcuni dati registrati in tempo reale tramite la scheda WeMos D1R2. Abbiamo dunque creato un'applicazione interattiva che visualizza graficamente un'immagine dinamica i cui parametri variano in base ai dati presi in input dal sensore di luminosità, collegato alla scheda, inizialmente passati al broker broker-mqttoiot.duckdns.org, e in ultimo estratti con Processing tramite un Paho Client MQTT subscriber. L'immagine che ne deriva è composta da un'unica sezione quadrata al cui interno vengono generati dei vettori chiamati "flow fields" ("campi di flusso"), i quali sulla base dei dati raccolti dal sensore variano:

- colore
- angolazione
- velocità

Per fare ciò abbiamo dapprima richiamato all'interno della funzione "setup()" un'altra funzione che, per ogni elemento dell'array soprannominato "points", crea un oggetto PVector, ovvero un vettore euclideo a due dimensioni, con coordinate x e y casuali generate dalla funzione "random", la quale restituisce un valore casuale di tipo float all'interno dell'intervallo specificato. In particolare, nel nostro caso, avendo impostato come intervallo della coordinata x: $(-\text{width}/2, \text{width}/2)$ e come intervallo della coordinata y: $(-\text{height}/2, \text{height}/2)$, i punti da cui si origineranno in seguito i vettori saranno quindi posizionati casualmente all'interno di un rettangolo centrato nella pagina.

Solo successivamente, nella funzione draw() siamo andati a lavorare sull'animazione di questi "flow fields" che determinerà la resa visiva finale del nostro digital ArtWork. Per prima cosa sono stati definiti dei range di valori di luminosità che andranno a decretare la colorazione dei vettori "flow

fields”, dopodichè con un ciclo for per ognuno di questi oggetti PVector, siamo andati a definirne l’angolazione con cui verranno indirizzati i vettori e successivamente la lunghezza totale, tutto questo in base al valore preso in input dal sensore di riferimento.

Più nello specifico:

Sono stati definiti dei range dei valori di luminosità entro i quali i flow fields assumeranno colori diversi. Per quanto riguarda i valori bassi di luminosità è stata scelta una tonalità di blu scuro che andrà a schiarirsi man mano che la luminosità aumenta, passando per le tonalità dell’azzurro e confluendo infine nel bianco, per i valori massimi di luminosità registrati.

Per impostare l’angolazione dei vettori abbiamo utilizzato la funzione `noise()`: interessante funzione sviluppata dal Professore Ken Perlin negli anni ’80, e che restituisce il valore del “rumore Perlin” in base alle coordinate specificate. Il “rumore Perlin” è un generatore di sequenze casuali che produce una successione di numeri più naturale e armonica rispetto a quella della funzione `random()` standard, e per cui vale la regola che minore è la differenza tra le coordinate, e più uniforme sarà la sequenza di rumore risultante. La funzione restituisce un valore pseudo-casuale compreso tra 0 e 1 che verrà poi moltiplicato per il valore di TAU (2π) in modo da convertirlo in radianti e, tramite la funzione `map()`, mappato in base al valore registrato dal sensore in un range da 0 a 3, con lo scopo di conferire armonia al disegno generato. Grazie all’utilizzo di questa funzione ogni volta che il valore di luminosità varierà, ovvero ogni volta che i vettori verranno rigenerati essi intraprenderanno angolazioni sempre diverse tra loro, rendendo ogni immagine unica.

Per fare un esempio: la funzione `map` utilizzata per mappare il valore di luminosità, e quindi l’angolazione dei vettori, sarà così definita: `map(lum, 0, 1023, 0, 3)`, in cui ”lum” è il dato in tempo reale proveniente dal sensore di luminosità, e (0, 1023) è l’intervallo di valori che il sensore è in grado di registrare e (0, 3) è il range entro cui verrà rappresentato il valore finale.

Successivamente, i vettori verranno allungati ”punto dopo punto” moltiplicando le loro coordinate per il coseno dell’angolo moltiplicato a sua volta per il valore registrato dal sensore mappato in un range da 0 a 5, creando in tal senso un effetto di movimento che ricorda le onde. In questo modo la velocità con cui essi crescono sarà influenzata dal valore del sensore. Un valore alto produrrà una crescita più veloce, mentre un valore più basso produrrà un allungamento più lento. Si può quindi affermare che la velocità con cui si allungano i vettori è direttamente proporzionale al valore registrato dal sensore di riferimento.

La resa finale sarà che per i valori di luminosità alta i vettori saranno di colorazioni chiari tendenti al bianco, con velocità più alta e tendenzialmente con angolazioni più frastagliate, mentre per valori di luminosità bassi tenderanno a una tonalità di blu scura, velocità bassa e tendenzialmente una direzionalità meno curva.

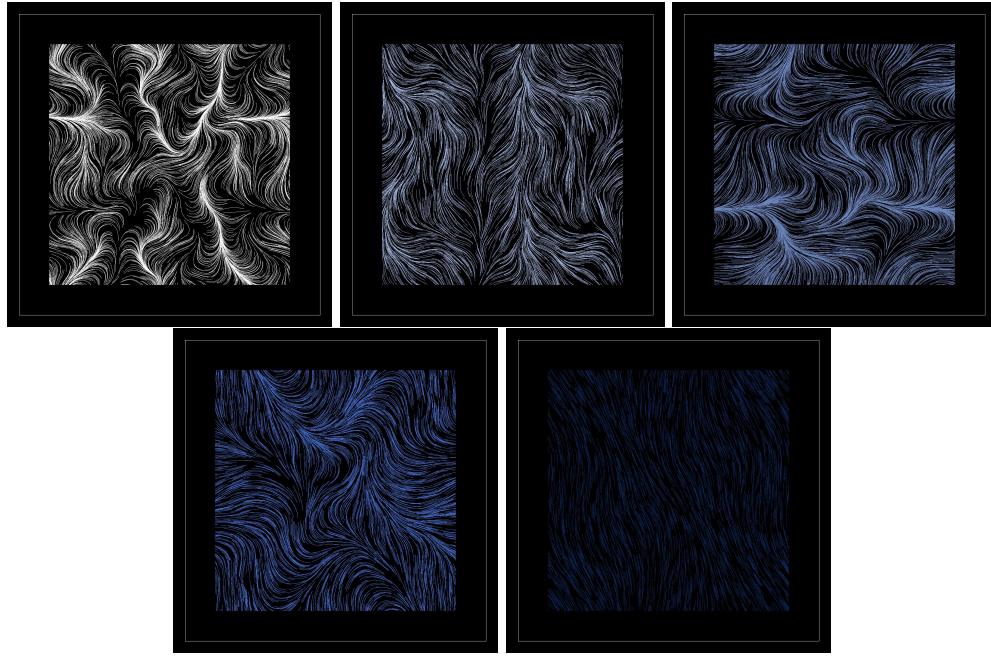


Figure 7: Immagini di partenza generate dal codice Processing, dal livello di luminosità più alto al più basso

7 Tokenizzazione come Non-Fungible Token e pubblicazione nella galleria digitale Async

7.1 Blockchain e NFT

La tecnologia blockchain consiste nel creare un registro distribuito che consente di registrare, verificare e conservare una serie di transazioni in modo sicuro, immutabile e trasparente. La particolarità di questa tecnologia è la mancanza di un'unità centrale e la presenza di una rete peer-to-peer di unità computazionali, chiamati nodi, che lavorano insieme per mantenere e validare il registro. Ogni nodo di questa rete contiene una copia dell'intero registro che viene aggiornato attraverso un processo chiamato consenso. Questo processo è distribuito su tutti i nodi della rete nel caso delle *blockchain permissionless* (*o pubbliche*) oppure affidato solo a dei nodi autorizzati nel caso delle *blockchain permissioned* (*o private*).

Le transazioni sono raggruppate in blocchi, ognuno contenente un insieme di dati e un riferimento al blocco precedente, formando così una catena di blocchi (da qui il nome *blockchain*), ed è proprio questo collegamento tra blocchi che garantisce l'integrità e la continuità del registro creando un vero e proprio storico organizzato cronologicamente. Una volta che un blocco viene aggiunto alla blockchain diventa immutabile, cioè le informazioni contenute in esso non possono essere modificate senza il consenso della maggioranza dei nodi della rete. Ciò fornisce un alto livello di sicurezza e trasparenza delle transazioni.

La blockchain ha diverse applicazioni tra cui:

- **Criptovalute:** originariamente questa tecnologia è stata sviluppata per supportare delle transazioni finanziarie senza la necessità di intermediari centralizzati come banche, che utilizzano come moneta di scambio delle criptovalute come Bitcoin o Ethereum.
- **Smart contract:** i contratti intelligenti sono protocolli informatici che facilitano, verificano o fanno rispettare l'esecuzione di un contratto. Ad esempio per contratti finanziari, gli smart contract possono definire le regole per un prestito peer-to-peer, specificando i termini del prestito, i tassi di interessi e gli obblighi di rimborso. Nel contesto degli NFT, gli smart contract vengono utilizzati per creare e registrare i token digitali unici e indivisibili che rappresentano un elemento digitale specifico.

- **Tracciabilità dei prodotti:** si riferisce alla capacità di seguire il percorso di un prodotto lungo l'intera catena di approvvigionamento, dalla sua origine fino al consumatore, garantendo la provenienza e la trasparenza delle merci. Questo può essere particolarmente utile in settori come l'industria farmaceutica, la logistica o l'agricoltura.
- **Voto elettronico:** il voto elettronico basato su tecnologia blockchain migliora sensibilmente la sicurezza e l'integrità del voto stesso, rendendo, ad esempio, più difficile la manipolazione dei risultati.
- **Videogiochi:** unendo il mondo delle blockchain al mondo videoludico si ottiene una combinazione unica: i giocatori possono possedere e scambiare in modo sicuro oggetti di gioco unici, come armi, personaggi abilità, garantendo la proprietà digitale e la scarsità degli oggetti stessi.

Per quanto riguarda gli NFT, acronimo di *Non-Fungible Token*, possono essere definiti come token digitali unici e indivisibili che rappresentano la proprietà di un elemento digitale specifico, come un'opera d'arte, un pezzo musicale, un video o altro tipo di contenuto digitale. A differenza delle criptovalute, che sono fungibili (cioè intercambiabili tra loro in modo equivalente), gli NFT non possono essere sostituiti o scambiati in modo diretto con altri token.

Questa tecnologia ha guadagnato popolarità perché consente ai creatori digitali di autenticare e monetizzare il loro lavoro. Infatti, il possesso di un'opera da parte di una persona è certificato direttamente dalla blockchain in modo inconfondibile. Inoltre, un aspetto interessante è che gli NFT possono consentire agli artisti di ricevere una percentuale delle future vendite quando le loro opere vengono rivendute sul mercato secondario. Gli NFT hanno aperto nuove opportunità nel mondo dell'arte digitale, del collezionismo e del commercio dei beni digitali, creando un mercato in cui i contenuti digitali possono essere oggetto di scambio e possesso unico.

7.2 Async marketplace

Uno tra i numerosi marketplace per gli NFT è Async, basato sulla blockchain di Ethereum. Questa piattaforma consente agli artisti di creare opere d'arte digitali interattive e componibili, introducendo un concetto innovativo chiamato *Layered Art*, in cui le opere d'arte possono essere composte da diversi livelli sovrapposti. Per esempio, è possibile creare un'immagine con un soggetto che non cambia e aggiungere diverse opzioni di sfondo e un acquirente al momento dell'acquisto otterrà un'opera con uno sfondo scelto casualmente basandosi su probabilità impostate dall'artista. In realtà è stato scelto questo NFT Marketplace in quanto gratuito, a differenza di altri in cui c'è la necessità di confermare identità o status di artista con vari portfolio, ma anche in maniera tale da lasciare uno spunto per sviluppi futuri di questo progetto, integrando l'opera esclusivamente visiva, per esempio con elementi sonori.

Prendendo in considerazione questo caso sono state generate cinque immagini, ciascuna a partire da diversi livelli di luminosità.

Per ottenere delle opere ancora più dinamiche e uniche, le immagini sono state spezzate in tre parti con l'idea di mischiarle tra loro per ottenere così l'opera finale. Di seguito sono riportati i passaggi seguiti per ricreare questa composizione artistica.

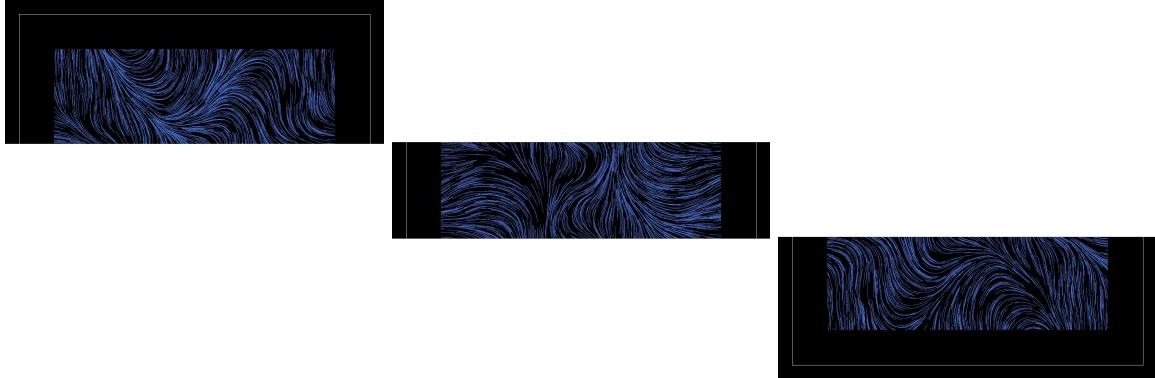


Figure 8: Esempio di divisione in tre parti di un'immagine originale

Una volta che tutto il materiale è pronto bisogna impostare su Async quello che viene chiamato *Art Blueprint* che permette di generare facilmente collezioni generative a partire dall'opera creata, permettendo ai collezionisti di coniare edizioni uniche da combinazioni casuali di livelli ed elementi. Per questo progetto sono stati utilizzati cinque *layer* (uno per immagine) ed ognuno di questi è composto da tre *stati* contenenti ciascuno una parte dell'immagine originale. Inoltre, per ogni *stato*, si può impostare una rarità che in questo caso è uniforme con somma pari al 100%, cioè verrà sempre scelto esattamente uno *stato* per ogni *layer*.

Layers (6)		Layer Title		
layer_5	Layers (3)	layer_1		
layer_4	Layers (3)	States		
layer_3	Layers (3)	State Title		Rarity
layer_2	Layers (3)	1_top	1_top.png	33.33%
layer_1	Layers (3)	1_middle	1_middle.png	33.33%
Background	Layers (2)	1_bottom	1_bottom.p...	33.33%

Figure 9: A sinistra la lista dei layer con il numero di stati per layer, a destra il layer 1 in dettaglio

Per via della struttura di questo *Blueprint* può succedere che vengano scelti, per esempio, sempre la parte superiore dell'immagine, lasciando vuote la parte centrale e inferiore. Di conseguenza è stato necessario aggiungere uno sfondo che è una combinazione statica di tre parti. Quindi per generare il pezzo artistico viene scelto uno stato per ogni layer e infine vengono uniti in un'unica immagine. In totale, quindi, sono possibili 468 combinazioni diverse ed ognuna sarà un pezzo unico. Di seguito sono riportati alcuni esempi di ciò che un cliente potrebbe ottenere al momento dell'acquisto:

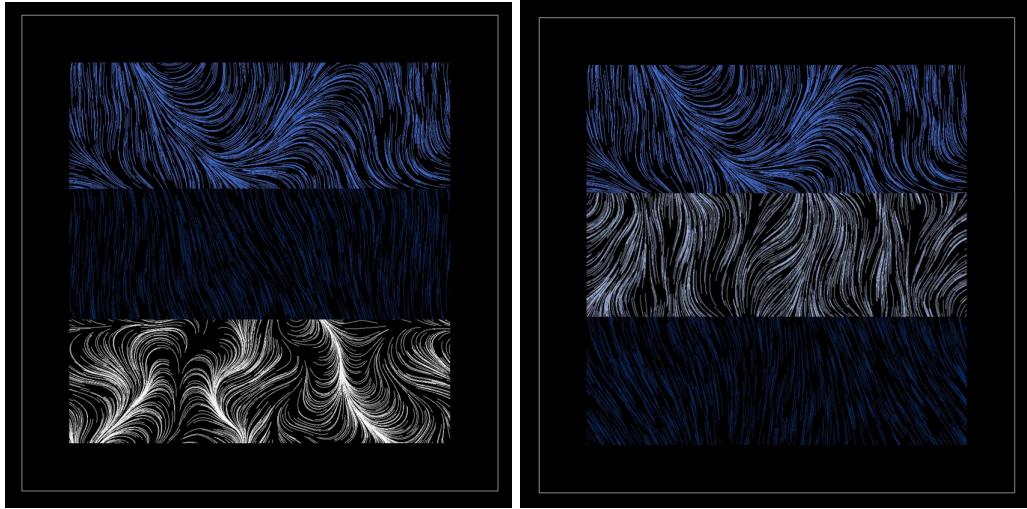


Figure 10: Esempi di opere finali

Questo *Blueprint* in particolare permette di avere delle edizioni leggendarie, cioè edizioni che vengono coniate con probabilità molto bassa, che non sono generate dinamicamente, ma statiche come immagini o, come in questo caso, video. Per questa sezione sono stati caricati due video che mostrano un'animazione dei vettori che sono stati descritti nella sezione 6.4. Le immagini sopra usate sono in realtà dei frame di questi video.

Se la preview non va è disponibile a questo [link](#)

A questo punto si può inviare il *Blueprint* che dovrà essere revisionato dal team di Async. Una volta approvato e pagata la tassa per creare il contratto(che al momento della stesura di questa relazione si aggira intorno ai 60\$ a causa dell'alta congestione della rete), verrà pubblicato sul marketplace e gli acquirenti potranno effettuare l'operazione di mint, cioè, creare effettivamente il token sulla blockchain di quest'opera e riceveranno un pezzo unico generato al momento.

8 Conclusioni

In conclusione, il progetto presentato ha l'obiettivo di esplorare l'intersezione tra il Web 3.0 e l'arte, utilizzando i Non-Fungible Token (NFT) come mezzo per trasformare l'arte tradizionale in opere digitali uniche. L'uso di NFT consente di garantire l'autenticità e di sfruttare un modo del tutto automatico per la creazione di arte digitale. Il progetto si basa sull'integrazione dell'Internet of Things (IoT) con

la Crypto Art, consentendo la creazione di opere d'arte interattive e personalizzabili attraverso l'uso di sensori e dispositivi fisici. L'utilizzo del microcontrollore WeMos D1R2, insieme a sensori come il fototransistor, permette di acquisire dati ambientali e utilizzarli per generare opere d'arte uniche. La comunicazione tra i dispositivi IoT e la creazione delle immagini artistiche sono gestite attraverso il protocollo MQTTs, garantendo una trasmissione affidabile dei messaggi.

La sicurezza e la privacy sono prese in considerazione nel progetto, con l'implementazione di strati di sicurezza come l'autenticazione con credenziali e la comunicazione tramite SSL (Secure Socket Layer). Ciò assicura che solo gli utenti autorizzati possano accedere al sistema e previene manipolazioni non autorizzate dell'algoritmo di creazione delle opere d'arte. Infine, le opere d'arte generate sono tokenizzate come NFT e pubblicate su un NFT marketplace, in questo caso Async Art, per consentire la compravendita e la collezione di opere d'arte digitali interamente realizzate tramite questo sistema in maniera automatica. In sintesi, il progetto rappresenta una fusione tra l'arte tradizionale, la tecnologia blockchain e l'IoT, aprendo nuove prospettive per la creazione e la fruizione delle opere d'arte digitali. L'utilizzo di NFT, insieme alla blockchain e all'IoT, offre un approccio innovativo e decentralizzato all'arte, consentendo agli artisti di esplorare nuove forme di espressione e di connettersi con un pubblico globale, facendo scoprire i propri pensieri e opere in maniera più facile e sofisticata.

8.1 Apporto componenti del gruppo

- Del Chin Riccardo e Sacchet Anna: Realizzazione Broker MQTTs in Oracle (Ubuntu), Client-Publisher, Client Paho Subscriber, realizzazione sicurezza TLS con relativi certificati CertBot, realizzazione circuito elettrico, test di funzionamento dell'architettura.
- Sagliocca Paola: Salvataggio dati nel database MySQL, creazione del Digital Artwork utilizzando Processing.
- De Bona Federico: Creazione dell'opera finale a partire dal Digital Artwork, utilizzo della piattaforma Async.art e tokenizzazione