Specifiche Software

for

Progetto Judge

Version 1.0 approved

Prepared by

Giovanni D’Ambrosio (A13002622),

Luca De Gennaro (A13002597),

Alessandro Fedele (A13002644)

Università degli Studi della Campania Luigi Vanvitelli

02/06/2024

Indice

1. Introduzione

Progetto Judge nasce da un problema: tenere organizzata la propria collezione di carte è un processo faticoso che porta via troppo tempo, soprattutto per collezioni voluminose.
Da qui nasce l'idea di un sistema di scannerizzazione e salvataggio rapido e automatico delle proprie carte, comodamente gestito da una base dati. Il progetto è utilizzabile sia da utenti esperti, che da chi non ha mai avuto esperienze informatiche avanzate.
Il servizio offre un sistema di scan efficiente, che attraverso l'utilizzo di un modello OCR per il riconoscimento testuale, analizza la carta da gioco inserita nell'apposito ingresso carta e salva la copia appena valutata in modo totalmente trasparente all'utilizzatore.
Con un semplice display LCD, il sistema comunica all'utente le informazioni sul proprio stato, incluso quando inserire o rimuovere la carta per la scansione, mentre 2 LED comunicano all'utilizzatore il risultato della scansione.
L'interfaccia è semplice, industriale ed efficace, e la collezione salvata è controllabile comodamente da PC accedendo all'interfaccia Web messa a disposizione dal sistema.
In breve, con Progetto Judge, abbiamo voluto semplificare uno degli aspetti più stancanti e noiosi di quello che è l'affascinante mondo delle carte collezionabili.

2. Componentistica

2.1. ESP CAM 0V2640

Al centro del progetto, splende il modulo ESP CAM 0V2640 per l’analisi delle carte. Il compito della fotocamera è quello di scattare in maniera nitida e precisa la foto della carta che dovrà essere analizzata dal modello OCR di riconoscimento testuale.
Con una risoluzione di 1600x1200px il dispositivo offre una qualità più che sufficiente per il modello di riconoscimento, ed è in grado, grazie ad un led RGB, di scattare sempre in condizioni di illuminazione perfetta per garantire il massimo della qualità di scannerizzazione.

2.2. HC-SR04 ULTRASONIC DISTANCE SENSOR

Ad aiutare la fotocamera a stampare la carta nel momento opportuno, è stato utilizzato un sensore di distanza basato su rimbalzo di onde a ultrasuoni per rilevare quando è stato inserito un oggetto all’interno della camera d’ombra della fotocamera.
Il funzionamento è semplice: un pin trigger attiva l’invio del segnale a ultrasuoni, un pin echo attende la ricezione del segnale in ritorno. Quando la distanza percepita cambia, significa che l’onda, che prima rimbalzava sul muro a circa 6 cm di distanza, rimbalza ora sulla carta. Quando la differenza della distanza captata è maggiore di una costante MIN_DELTA, viene avviato lo scatto della fotocamera.
Il sensore viene quindi messo in attesa fin quando il sistema non è nuovamente pronto per lo scan di una nuova carta.

2.3. DISPLAY LCD 16X2 (I2C)

Per offrire un’esperienza interattiva, è stato utilizzato un display LCD che comunica con l’utilizzatore lo stato del sistema. Dalla fase di setup a quella di utilizzo, il display

è il componente principale per il coinvolgimento dell’utilizzatore durante il funzionamento del sistema.

2.4. COMPONENTI DI COMUNICAZIONE

Insieme al display LCD, 2 LED di comunicazione e un buzzer sono stati inseriti per comunicare l’esito della scansione. Al termine dell’analisi, in caso di carta riconosciuta, un LED blu comunicherà all’utilizzatore il successo dell’operazione. In caso contrario, sarà acceso un LED rosso e un buzzer per richiedere di riprovare la scansione della carta.

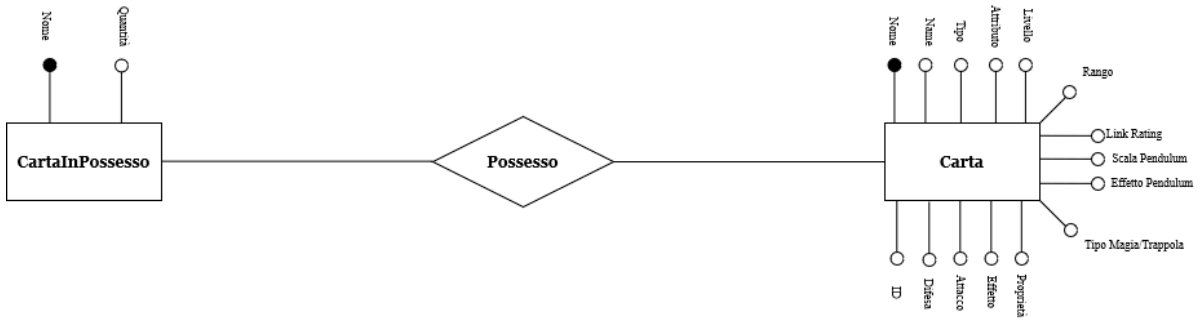
2.5. PULSANTE DI RESET

Per qualsiasi evenienza o necessità, è stato installato un pulsante esterno di reset per ripristinare il sistema alle condizioni di accensione.

3. PROGETTAZIONE BASE DATI

3.1. Progettazione Concettuale

Per la rappresentazione dei dati nel database, è stato realizzato uno schema concettuale basato su modello Entity-Relationship come segue:

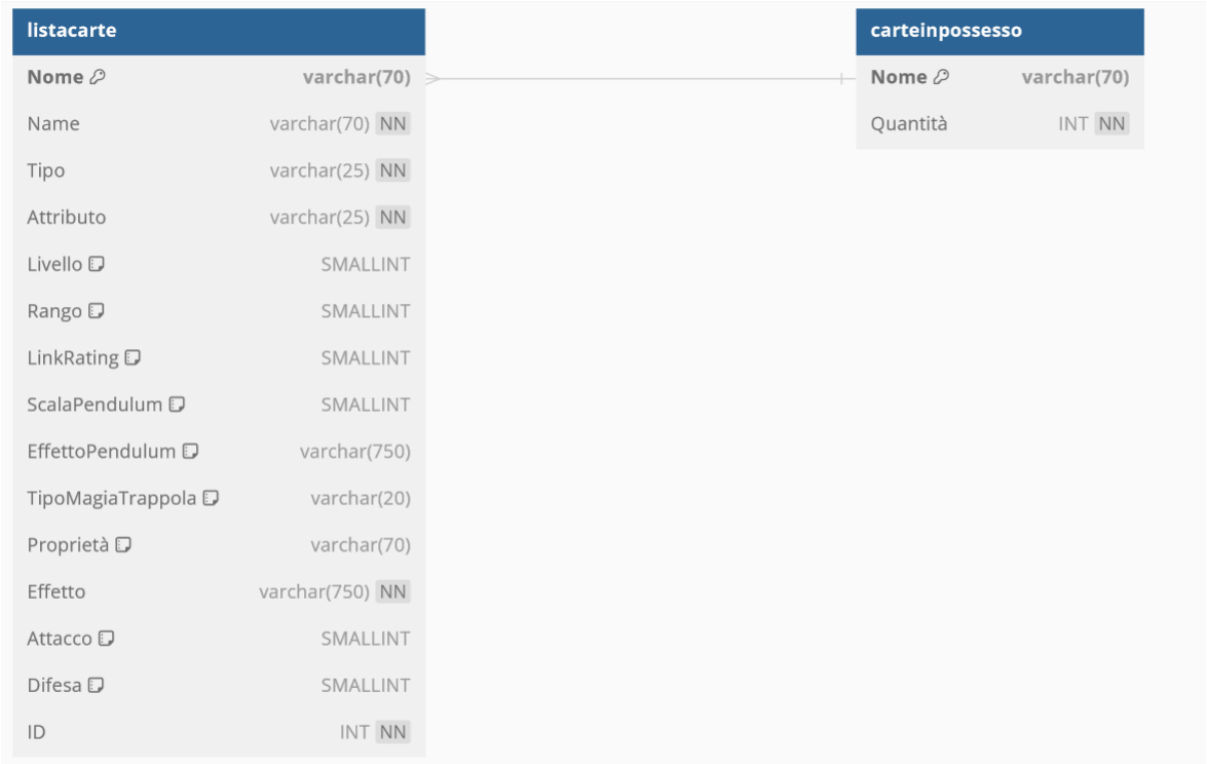


Dove l'entità *Carta* rappresenta l'insieme delle carte da gioco con tutte le caratteristiche associate, mentre l'entità *CartaInPossesso* rappresenta l'insieme delle carte in possesso nella propria collezione.
Lo schema non richiede ristrutturazione.

3.2. Progettazione Logica

3.2.1 Schema Relazionale

A partire dallo schema concettuale è stato realizzato lo schema logico che segue:



Da cui è stato elaborato il seguente script SQL per la costruzione delle tabelle:

```
CREATE TABLE listacarte(
    Nome varchar(70) PRIMARY KEY,
     Name varchar(70) NOT NULL,
```

```
     Tipo varchar(25) NOT NULL,
     Attributo varchar(25) NOT NULL,
     Livello SMALLINT DEFAULT NULL,
     Rango SMALLINT DEFAULT NULL,
     LinkRating SMALLINT DEFAULT NULL,
     ScalaPendulum SMALLINT DEFAULT NULL,
     EffettoPendulum varchar(750) DEFAULT NULL,
     TipoMagiaTrappola varchar(20) DEFAULT NULL,
     Proprietà varchar(70) DEFAULT NULL,
     Effetto varchar(750) NOT NULL,
     Attacco SMALLINT DEFAULT NULL,
     Difesa SMALLINT DEFAULT NULL,
     ID INT NOT NULL
)

CREATE TABLE carteinpossesso(
     Nome varchar(70) PRIMARY KEY,
     Quantità INT NOT NULL,
     FOREIGN KEY (Nome) REFERENCES listacarte(Nome)
)

—-------------------------------------------------------------
```

3.2.2 Riempimento tabelle

Per il riempimento delle tabelle con le informazioni correnti delle carte rilasciate sul mercato, è stato elaborato uno script SQL per caricare le informazioni da un file CSV da noi realizzato sulla base di un dataset in formato JSON reperito online:

```
LOAD DATA INFILE 'percorso/file/csv'
INTO TABLE listacarte
CHARACTER SET utf8
FIELDS TERMINATED BY '$'
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

3.3 Progettazione fisica

Per la gestione database è stato utilizzato MySQL come DBMS. Per l’ottimizzazione delle query è stato sufficiente utilizzare gli indici forniti in automatico da MySQL sulle chiavi primarie delle tabelle.

4. TRASMISSIONE DATI

4.1 Origine

La generazione dei dati avviene a partire dal microcontrollore, grazie al modulo *ESP CAM 0V2640* (vedi Componentistica 2.1). L'immagine scattata in formato jpeg viene elaborata dalla scheda, che si mette in contatto con il server inviando i dati attraverso protocollo HTTP (POST).

4.2 Analisi

Una volta ricevuti i dati, il server raccoglie l'immagine dal flusso di comunicazione TCP e la inoltra ad uno script Python che la analizza con un modello OCR per il riconoscimento del testo. Il sistema utilizza l'API Google Cloud Vision sviluppata da Google per ottenere l'esito della richiesta, che viene restituita al server.

4.3 Interrogazione

Una volta recuperata la risposta, il server invia una query al database per verificare l'esistenza della carta scansionata e, in caso positivo, una seconda query per il salvataggio in raccolta della copia appena scansionata.

4.4 Richiesta Esito

Per ricevere l'esito dell'elaborazione, il microcontrollore si rimetterà in contatto con il server attraverso HTTP (GET) al termine dell'elaborazione, attendendo in risposta informazioni sul testo letto.

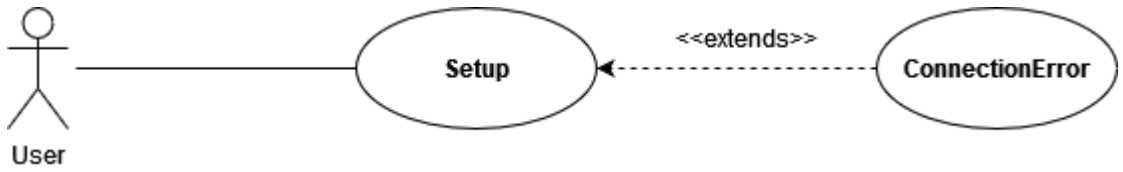
4.5 Visualizzazione Esito

Ricevuto ed elaborato l’esito, questo sarà riportato all’utilizzatore attraverso i *Componenti di comunicazione* (vedi Componentistica 2.4).
Al termine, il microcontrollore avviserà l’utilizzatore di essere pronto per la prossima scansione.

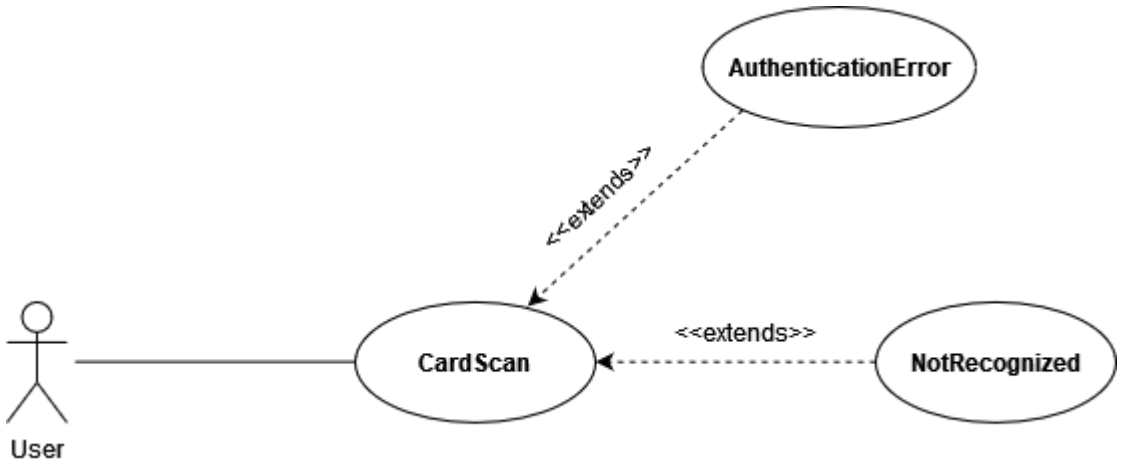
5. CASI D’USO

5.1 Diagrammi UML

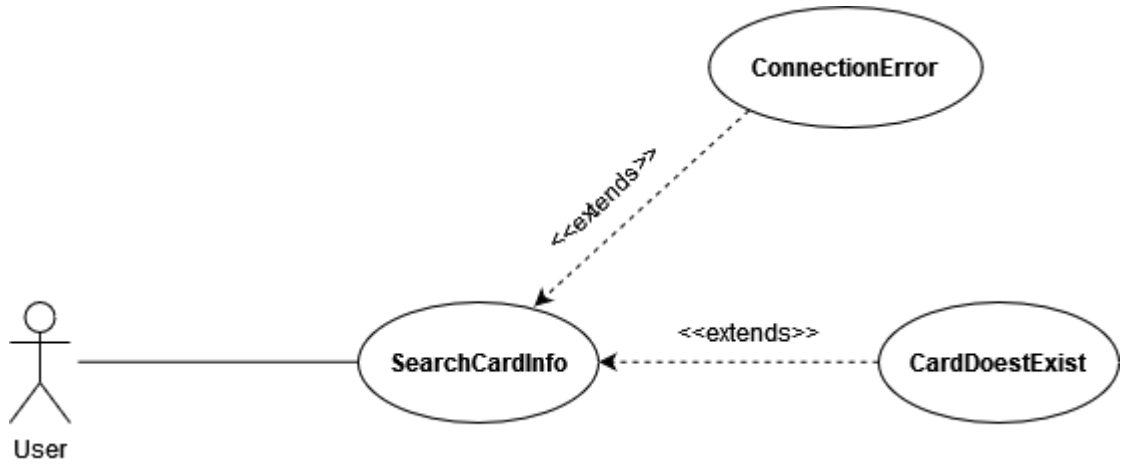
Per le funzionalità di sistema, sono stati costruiti i seguenti diagrammi UML dei casi d’uso:



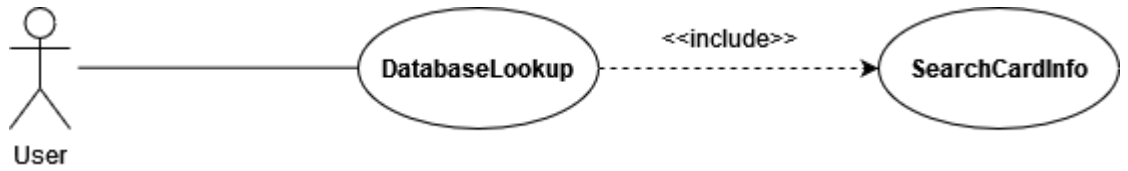
Use Case 0001: Setup



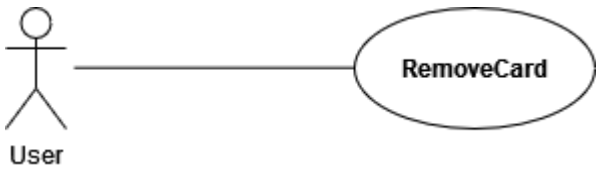
Use Case 0002: CardScan



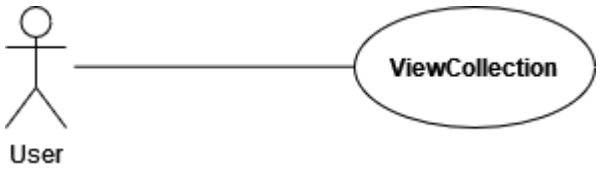
Use Case 0003: SearchCardInfo



Use Case 0004: DatabaseLookup



Use Case 0005: RemoveCard



Use Case 0006: ViewCollection

5.2 Scenari dei casi d’uso

Dai diagrammi dei casi d’uso, sono stati realizzati i seguenti scenari dei casi d’uso:

5.2.1 Setup

Use Case ID:	#0001
Use Case Name:	Setup
Actors:	User
Description:	L’utente inizializza il sistema
Preconditions:	
Postconditions:	Il sistema rende disponibile l’utilizzo delle funzionalità.
Normal Flow:	1. L’utente effettua il boot up della board. 2. La board avvia la connessione ad internet per interagire con la rete locale. 3. La board comunica all’utente le informazioni relative alla connessione.
Alternative Flow:	
Exceptions:	La board non riesce a completare la connessione: 1. La board mostra all’utente un messaggio di errore.
Includes:	

Scenario 1: Setup

5.2.2 CardScan

Use Case ID:	#0002
Use Case Name:	CardScan
Actors:	User
Description:	L'utente richiede lo scan di una carta da gioco
Preconditions:	
Postconditions:	Il sistema salva le informazioni sulla carta scannerizzata.
Normal Flow:	1. L'utente richiede lo scan della carta dal gioco alla board. 2. La board cattura un'immagine della carta da gioco. 3. La board invia la cattura al server per l'elaborazione. 4. Il server avvia il meccanismo di riconoscimento testuale per ottenere il nome della carta da gioco. 5. Il server salva le informazioni della carta da gioco e risponde alla board. 6. La board notifica l'utente dell'avvenuto scan.
Alternative Flow:	
Exceptions:	1. Il nome della carta da gioco non è riconosciuto tra quelle esistenti: 1. Il server risponde alla board con un messaggio di errore. 2. La board notifica l'utente del fallimento dello scan. 2. Il sistema di autenticazione per il riconoscimento testo fallisce: 1. Il server risponde alla board con un messaggio di errore. 2. La board notifica l'utente del fallimento del sistema di autenticazione.
Includes:	

Scenario 2: CardScan

5.2.3 SearchCardInfo

Use Case ID:	#0003
Use Case Name:	SearchCardInfo
Actors:	User
Description:	L'utente richiede informazioni su una carta da gioco
Preconditions:	
Postconditions:	Il sistema fornisce all'utente le informazioni sulla carta da gioco richiesta
Normal Flow:	1. L'utente inserisce il nome della carta da gioco. 2. Il sistema invia una query al database circa la carta richiesta dall'utente. 3. Il database risponde con le informazioni richieste. 4. Il sistema porta in visualizzazione i risultati della ricerca.
Alternative Flow:	
Exceptions:	Il nome della carta da gioco non è riconosciuto tra quelle esistenti: 1. Il sistema porta in visualizzazione un messaggio di assenza di risultati.
Includes:	

Scenario 3: SearchCardInfo

5.2.4 DatabaseLookup

Use Case ID:	#0004
Use Case Name:	DatabaseLookup
Actors:	User
Description:	L'utente richiede informazioni sul numero di copie di una carta da gioco da lui posseduta
Preconditions:	
Postconditions:	Il sistema fornisce all'utente le informazioni sul numero di copie della carta da gioco richiesta
Normal Flow:	1. L'utente inserisce il nome della carta da gioco. 2. Il sistema invia una query al database circa la carta richiesta dall'utente e verifica la validità della carta da gioco richiesta (Use Case #0003). 3. Il database risponde con le informazioni richieste. 4. Il sistema porta in visualizzazione la quantità della carta da gioco richiesta.
Alternative Flow:	
Exceptions:	Il nome della carta da gioco non è riconosciuto tra quelle esistenti: 1. Il sistema porta in visualizzazione un messaggio di errore.
Includes:	SearchCardInfo (Use Case #0003)

Scenario 4: DatabaseLookup

5.2.5 RemoveCard

Use Case ID:	#0005
Use Case Name:	RemoveCard
Actors:	User
Description:	L'utente rimuove dalla base dati una copia di una carta da lui in possesso
Preconditions:	
Postconditions:	Il sistema conferma la rimozione della carta
Normal Flow:	1. L'utente inserisce il nome della carta da gioco da rimuovere. 2. Il sistema invia una query al database per richiedere la rimozione di una copia. 3. Il database risponde con il risultato dell'operazione.
Alternative Flow:	
Exceptions:	
Includes:	

Scenario 5: RemoveCard

5.2.6 ViewCollection

Use Case ID:	#0006
Use Case Name:	ViewCollection
Actors:	User
Description:	L'utente visualizza le carte della sua collezione
Preconditions:	
Postconditions:	Il sistema porta in visualizzazione la collezione dell'utente
Normal Flow:	1. L'utente accede all'interfaccia di visualizzazione. 2. Il sistema invia una query al database per ottenere informazioni sulla collezione dell'utente. 3. Il database risponde con le informazioni richieste. 1. Il sistema porta in visualizzazione le informazioni all'utente.
Alternative Flow:	
Exceptions:	
Includes:	

Scenario 6: ViewCollection

6. FUNZIONI DI SISTEMA

Sulla base delle specifiche del sistema, sono state realizzate le seguenti funzioni principali per la gestione del flusso delle informazioni.

6.1 CAST_ECHO()

```
309  float cast_echo()
310  {
311    // Attivazione segnale
312    delay(100);
313    digitalWrite(TRIGGER_PIN, HIGH);
314    delayMicroseconds(10);
315    digitalWrite(TRIGGER_PIN, LOW);
316
317    //Ricezione segnale
318    long duration = pulseIn(ECHO_PIN, HIGH);
319    float distance = (duration * 0.0343) / 2;
320    return distance;
321  }
```

La funzione cast_echo è alla base del funzionamento del sensore a ultrasuoni. Settando ad alto il PIN associato al trigger, si invia il segnale che rimbalza sulla parete dello scanner. La funzione pulseIn rimane in attesa della ricezione del segnale, che permette di calcolare la distanza conoscendo la velocità del suono.

6.2 CAPTUREANDSENDPHOTO()

```
170  void captureAndSendPhoto()
171  {
172    camera_fb_t * fb = esp_camera_fb_get(); // Funzione che scatta la foto
173    // fb è la struttura che contiene la foto
```

```
191    http.begin(client, serverAddress);
192    http.addHeader("Content-Type", "image/jpeg");
193    int httpResponseCode = http.POST((uint8_t*)fb->buf, fb->len);
```

```
202      String response = http.getString();
203      Serial.println(response);
204      http.end();
205
206      esp_camera_fb_return(fb);
207
208      Serial.println("Waiting for GET");
209      delay(SERVER_TIME_RESPONSE); // Attende che il server elabora la richiesta
210      waitForServerResponse();
```

La funzione captureAndSendPhoto si occupa dell'acquisizione della foto attraverso la funzione di libreria esp_camera_fb_get (*esp_camera.h*), per poi inviare l'immagine al server attraverso HTTP (POST) per mezzo della funzione di libreria http.POST (HTTPClient.h).
Una volta ricevuto l'esito dell'invio, il microcontrollore, per fini di eventuale debugging, stampa su seriale le informazioni sulla risposta ricevuta.

6.3 UPLOAD.PHP

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_SERVER['CONTENT_TYPE']) && strpos($_SERVER['CONTENT_TYPE'], 'image/') !== false)
{
    try
    {
        $imageData = file_get_contents('php://input');
        if ($imageData !== false)
        {
            $uploadDirectory = 'images\\';

            $filename = uniqid('image_') . '.jpg';
            $filePath = $uploadDirectory . $filename;
            if (file_put_contents($filePath, $imageData) !== false)
            {
                $command1 = 'C:/xampp/htdocs/GooglePray/.venv/Scripts/activate.bat';
                $command2 = 'GooglePray\\main.py "' . $filePath . '"';
                $output = exec($command1 . ' && ' . $command2);

                $filename = "temp.txt";
                $file = fopen($filename, 'r');

                $string = "";

                if ($file)
                {
                    $string = fgets($file);
                    fclose($file);
                    scan($string);
                    unlink($filename);
                }
```

Lo script PHP target per la ricezione della foto, analizza la POST HTTP per recuperare l'immagine, che viene salvata localmente. Il percorso viene quindi utilizzato per richiamare lo script Python per il riconoscimento testuale. A seguito della terminazione dello script, il server apre quindi il file temporaneo in cui è presente il risultato dell'elaborazione per salvare la carta trovata sulla base dati.

6.4 DETECT_TEXT()

```python
def detect_text(path):
    from google.cloud import vision
    import io
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = r"Google Cloud Vision Key Here.json"
    client = vision.ImageAnnotatorClient()

    # Legge l'immagine salvata dal server
    with io.open(path, 'rb') as image_file:
        content = image_file.read()

    image = vision.Image(content=content)
    # Google API recognition
    response = client.text_detection(image=image)
    texts = response.text_annotations

    # Stampe di debugging
    print('Texts:')
    for text in texts:
        print('\n"{}"'.format(text.description))
        vertices = (['({},{})'.format(vertex.x, vertex.y)
                    for vertex in text.bounding_poly.vertices])
        print('bounds: {}'.format(','.join(vertices)))

    if response.error.message:
        raise Exception(
            '{}\nUlteriori informazioni sull\'errore: '
            'https://cloud.google.com/apis/design/errors'.format(response.error.message))
    return texts[0].description
```

La funzione detect_text effettua l'analisi dell'immagine attraverso API Google (google.cloud.vision). Il risultato viene elaborato in modo da restituire solo il testo riconosciuto dal modello, eliminando quindi informazioni aggiuntive come la posizione dei paragrafi in cui è stato rilevato il testo e la suddivisione in singole parole.

6.5 CLEAN_TEXT()

```python
def clean_text(text):
    # Elimina i seguenti caratteri dalla stringa
    while text.startswith("1") or text.startswith("7") or text.startswith("\n") or text.startswith("."):
        text = text[1:]
    lines = text.split('\n')
    # Cerca la prima riga di testo che soddisfa i seguenti requisiti
    for line in lines:
        while line.startswith("1") or text.startswith("7") or line.startswith("\n") or line.startswith("."):
            line = line[1:]
        line = line.replace('+', '')
        if len(line) > 3:
            return line
    return "Nessun Testo"
```

La funzione clean_text effettua una rifinitura del testo letto. A seguito di un processo di trial and error, la funzione è stata adattata alle caratteristiche della fotocamera utilizzata per migliorare il risultato elaborato.

6.6 SCAN()

```php
function scan($name)
{
    $filename = "match.txt";
    $file = fopen($filename, 'w');
    if(is_null($name))
    {
        $name = "";
    }
    $servername = "localhost";
    $username = "root";
    $password = "";
    $db_name = "carte";
    $foundMatch = "-1";
    $conn = mysqli_connect($servername,$username,$password,$db_name);
    if (!$conn)
    {
        die("Connessione fallita: " . mysqli_connect_error());
    }
    $query = "SELECT Nome FROM listacarte WHERE (Nome= ? or Name= ? )";
    $stmt = mysqli_prepare($conn, $query);
    mysqli_stmt_bind_param($stmt,"ss",$name, $name);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);
    $row = mysqli_fetch_assoc($result);
```

```php
    mysqli_close($conn);
    if ($row["Nome"] !== NULL)
    {
        $conn = mysqli_connect($servername,$username,$password,$db_name);
        $insertion_query = 'INSERT INTO carteinpossesso (Nome, Quantità) VALUES (?, "1") ON DUPLICATE KEY UPDATE Quantità = Quantità + 1';
        $stmt = mysqli_prepare($conn, $insertion_query);
        mysqli_stmt_bind_param($stmt,"s",$row["Nome"]);
        mysqli_stmt_execute($stmt);
        mysqli_close($conn);
        $foundMatch = "    Trovato:    ";
    }
```

Il compito della funzione di scan è quello di connettersi alla base dati per aggiornare le quantità della carta rilevata: se la carta è trovata tra quelle esistenti, viene inviata una query per inserire nella tabella *carteinpossesso* una nuova entry se questa non era presente in collezione, oppure per incrementare la quantità se invece questa era già presente.

6.7 WAITFORSERVERRESPONSE()

```
250   void waitForServerResponse()
251   {
252     WiFiClient client;
253     HTTPClient http;
254
255     http.begin(client, serverResponseUrl);
256     int httpResponseCode = http.GET(); // Invio Get
```

Una volta inviata l'acquisizione, il microcontrollore invia una richiesta HTTP (GET) circa l'esito dell'elaborazione ad un opportuno modulo PHP, che legge da file il risultato della scansione e risponde di conseguenza.
Il seguito della funzione è stato omesso perché tratta operazioni di manipolazione della stringa ricevuta in risposta per adattarla alla comunicazione utente.

6.8 SCAN.PHP

```php
if($_SERVER["REQUEST_METHOD"] == "GET")
{
    $filename = "match.txt";
    $file = fopen($filename, 'r');

    $string = "";

    if ($file)
    {
        $string = fgets($file);
        fclose($file);
    }
    else
    {
        $string = "Errore Lettura File";
    }
    unlink($filename);
    echo $string;
```

Il file scan.php si occupa di gestire la richiesta GET del microcontrollore, analizzando il contenuto del file temporaneo *match.txt*, in cui è presente il messaggio da mostrare all'utente circa il risultato dell'operazione. La stringa viene quindi inviata in echo come risposta HTTP alla board.

6.9 RESEARCH.PHP

```
    <form action="" method="POST">
        <label for="cardSearch">Cerca la tua carta:</label>
        <input type="text" id="cardSearch" name="cardSearch">
        <input type="hidden" id="hiddeninfo" name="op" value="add">
        <br>
        <button type="submit">Cerca</button>
        <button type="submit" onclick="updateHiddenInput()">Rimuovi</button>

```

```
                $query = "SELECT * FROM listacarte WHERE (Name= ? or Nome= ?)";
                $stmt = mysqli_prepare($conn, $query);
                mysqli_stmt_bind_param($stmt,"ss",$_POST["cardSearch"], $_POST["cardSearch"]);
                mysqli_stmt_execute($stmt);
                $result = mysqli_stmt_get_result($stmt);
                $row = mysqli_fetch_assoc($result);
                if(isset($row)){
                    $query = "SELECT Quantità from carteinpossesso WHERE Nome= ?";
                    $stmt = mysqli_prepare($conn, $query);
                    mysqli_stmt_bind_param($stmt,"s",$row["Nome"]);
                    mysqli_stmt_execute($stmt);
                    $result = mysqli_stmt_get_result($stmt);
                    $quantity = mysqli_fetch_assoc($result);
                    if(!isset($quantity["Quantità"])) $quantity = 0;
                    else $quantity = $quantity["Quantità"];
```

```
    <script>
        function updateHiddenInput() {
            var hiddenInput = document.getElementById('hiddeninfo');
        hiddenInput.value = 'remove';
    }
</script>
```

Il file research.php offre un'interfaccia Web per l'interazione con la base dati, su cui è possibile cercare le informazioni di una qualsiasi carta da gioco rilasciata sul mercato oppure rimuovere una copia di una carta dalla propria collezione.
Ciò è ottenuto grazie ad una form HTML con doppio bottone di submit: *Cerca* effettua una ricerca all'interno della base dati, mentre *Rimuovi*, modificando il valore dell'input nascosto *hiddeninfo* attraverso la funzione JavaScript *UpdateHiddenInput*, comunica al PHP che la POST inviata ha come scopo la modifica delle quantità della carta.

7. LIBRERIE UTILIZZATE

7.1 ESP_CAMERA.H

Libreria offerta da Arduino per la gestione della fotocamera.

7.2 WIFI.H

Libreria per l'utilizzo del modulo WiFi per la connessione ad Internet.

7.3 HTTPCLIENT.H

Libreria di supporto per il protocollo HTTP per la gestione dell'invio delle richieste e della ricezione delle risposte.

7.4 LIQUIDCRYSTAL_I2C.H

Libreria di supporto per la stampa di caratteri sul display LCD.

7.5 WIRE.H

Libreria usata per il remapping dei PIN con funzionalità SCL e SDA per il display I2C, necessaria perché i PIN di default erano già in uso dal modulo della fotocamera.

7.6 GOOGLE.CLOUD.VISION

Libreria offerta da Google per l'utilizzo dell'API di riconoscimento del testo.

8. SVILUPPI FUTURI

Per le versioni successive di Progetto Judge, è in programma la realizzazione di una versione completa dell’interfacciamento Web con la base dati, insieme all’utilizzo di una fotocamera di qualità superiore per una scansione impeccabile delle carte da gioco.
Progetto Judge mira ad offrire la massima qualità ai giocatori, e lavorerà per continuare a farlo.