Omniseeker’s Processing - Informe

Grup: G1-6
Álvaro Díaz Laureano, 1639484
Jan Planas Batllori, 1636492
Pol Pugibet Martinez, 1630568
Marc Serra Ortega, 1551550

Introducció

Aquest és un informe on explicarem a profunditat el nostre projecte en diferents aspectes com el plantejament i el desenvolupament de la idea. Es tracta d’explicar de forma concisa el funcionament de l’aplicació.

1 Plantejament

En aquest apartat exposarem el "que s'ha fet" fent referencia a la idea principal del projecte i el propòsit d'aquesta.

La idea del nostre projecte es basa en crear un servidor que ens permeti controlar i monitorar un, dos o més robots en temps real. És la unió del nostre projecte de robòtica, on creem un eixam de robots omnidireccionals que disposen d'una càmera que grava tot el que veuen, a més d'un altre mòdul de control que permet el moviment. Les càmeres ja disposen d'un codi Arduino que és capaç de mostrar en temps real el que estan veient els robots, però no funciona a manera de servidor ni permet veure més d'un vídeo a la vegada. El propòsit, doncs, és fer que més d'un robot es pugui connectar a un servidor i, també, que més d'una persona (client) pugui veure la retransmissió d'aquests robots i controlar el seu mòdul de moviment mitjançant una aplicació web. Així doncs, etiquetarem el servidor com bidireccional.

Una vegada connectats els robots, la imatge que envien aquests serà processada pel servidor en cas que es detectin objectes i es modificarà la imatge per tal de dibuixar el "bounding box" de l'objecte i que sigui molt més visual pel client. La direcció de les dades en aquest cas és dels robots al client. Per altra banda, com ja hem mencionat, el client també podrà enviar ordres des de la pàgina web cap als robots tant per botons com per veu, per fer que es moguin cap a una direcció o una altra.

Finalment, un cop realitzat tot el trajecte que es vulgui fer, hi ha l'opció de crear un informe on es mostri tot el que el robot ha observat amb molta més especificitat.

2 Desenvolupament

Un cop clara la idea principal del projecte, explicarem el “com s’ha fet” pas a pas.

2.1 Mòduls ESP32

Abans de parlar del servidor, mencionarem la part relacionada amb els dispositius robòtics. L’únic que s’ha de tenir en compte en aquests dispositius és que siguin capaços tant d’enviar dades com de rebre-les i tractar-les. Cada robot conté dos controladors principals: la càmera i el mòdul controlador de moviment. La càmera únicament enviarà video en forma de dades cap al servidor. El mòdul controlador haurà de poder rebre dades i transformar-les a ordres.

Per dur a terme aquestes funcionalitats s’ha hagut de programar els controladors, que són específicament ESP32, en Arduino. No entrarem molt en detall, perquè no correspon amb l’assignatura, però si mencionarem quins aspectes s’ha de tenir en compte per tal que els mòduls siguin capaços de comunicar-se amb el servidor sense cap problema i a temps real. Bàsicament, són controladors connectats via WiFi a la xarxa, que és el que permet enviar i rebre dades. No obstant, aquest factor no és el que permet la comunicació amb el servidor, sino la declaració d’una instància “WebsocketsClient” de la llibreria “ArduinoWebsockets”. Aquesta llibreria permet que el dispositiu sigui capaç de connectar-se al servidor especificant camps com el port al que s'haurà de connectar i la IP externa del servidor. Una vegada fet això i confirmada la seva connexió, la qual es pot veure amb el “Serial Monitor” d’Arduino IDE, el dispositiu ja és capaç d’enviar dades mitjançant buffers i rebre-les de la mateixa forma. En el cas de la càmera, únicament la connexió i la configuració d’aquesta son necessàries per poder veure el vídeo des del servidor. El controlador de moviment, a part de la connexió, també haura de poder controlar les dades entrants que s’envien des del servidor, amb ajuda de “CallBacks”. Qualsevol missatge que es rebi

serà tractat tenint en compte si és una ordre de moviment o d'altre tipus i del valor que contingui.

2.2 Interfície Pública

Continuant amb la idea de l'apartat anterior, abans d'explicar finalment el servidor, mencionarem els aspectes de la interfície de l'usuari, la part més visual del projecte.

Hem realitzat una aplicació web mitjançant eines com HTML, CSS i JavaScript, que ens ha permès fer una interfície simple i entenedora per tothom. A més, com la idea és que qualsevol es pugui connectar amb els robots, s'ha fet que la pàgina web sigui responsiu, aconseguint així que des del mòbil es pugui veure el mateix que des de l'ordinador.

La visualització és possible mitjançant una instància "WebSocket" on s'especifica la IP del servidor i el port al qual es connectarà. Per exemple, en el cas de fer-ho localment seria "ws://192.168.X.X:8888". Si fos remotament, el que canvia és la IP de la màquina virtual i el port a 65080 per servidors de Google. Un cop declarada aquesta instància que indica a quin servidor ens estem connectant, ja es pot rebre i enviar dades mitjançant "sends".

En aquesta pàgina es troben les següents seccions: menú d'inici, instruccions i pàgina de control. El menú d'inici i les instruccions no comporten cap funcionalitat a l'aplicació. Únicament permet a l'usuari tenir una millor experiència en la navegació de la web i ajuda a entendre la funcionalitat d'aquesta. La pàgina de control, per altra banda, conté la secció de les càmeres dels robots i la del control del robot, amb botons i micròfon (Figura 1). La secció de càmeres conté el vídeo retransmès de cadascun dels robots, el qual ha passat prèviament pel servidor per al seu processament, que es comenta més endavant. A més, a sota de cadascuna de les imatges, es troba un botó de selecció el qual en activar-se, al robot al qual pertany aquella imatge se li enviarà totes les ordres que indiqui l'usuari mitjançant els botons de moviment o amb la veu. Per tant, és possible tant enviar únicament les dades a un sol robot com enviar-les a tots els robots per igual.

Figura 1. Pàgina de Control

L'enviament de dades és bastant similar en tots els casos. En qualsevol botó s'utilitza un "Event Listener" que permet l'enviament de dades en el mateix instant en el qual es fa clic al botó. Les dades trameses son en format "JSON", on s'especifica el port de la connexió, l'operació que es vol realitzar, el dispositiu al qual es vol enviar aquestes dades (tot i que no sempre és necessari) i el missatge. En el cas dels botons de selecció, les dades van enviades al dispositiu al que correspon el botó i s'envia l'estat d'aquest (si està actiu o no). Les fletxes de moviment, a diferència d'abans, no especifiquen un receptor en concret, s'envien al servidor per poder crear la comanda i, més endavant, enviar el missatge corresponent als dispositius que hagin estat actius després de prémer el botó de "select". Per tant, les fletxes envien com a missatge l'ordre de la que es tracta i el valor d'aquesta.

Per últim, la funcionalitat del micròfon varia lleugerament quant a com s'envien les dades. Primer la pàgina demana permisos a l'usuari per tal de poder utilitzar els recursos media del dispositiu. Una vegada té els permisos, el micròfon ja té funcionalitat. És similar a la d'un botó: prem una vegada per començar a gravar i una segona per parar la gravació. L'àudio recollit amb ajuda d'instàncies com "MediaRecorder" es guarda a una instància de "Blob" i s'envia finalment al servidor.

La pàgina també compta amb altres funcionalitats més secundàries, com poden ser les següents:

- Missatge d'error a la secció de càmeres quan no es detecta cap dispositiu connectat (Figura 2).
- Si hi ha un dispositiu connectat, però la seva càmera no està disponible, es mostrarà una imatge de "càmera no disponible" (Figura 1).

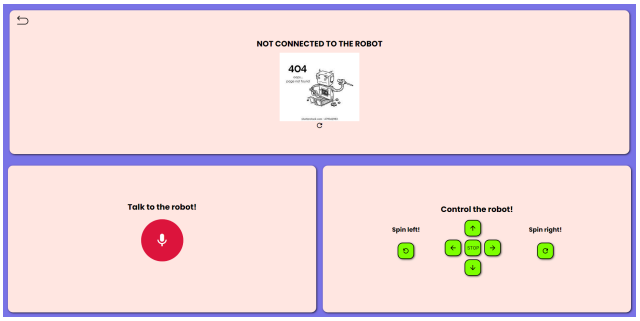

Figura 2. Missatge d'error

2.3. Servidor

Una vegada explicat com, tant els dispositius robòtics com els usuaris poden enviar i rebre dades, hem d'explicar com funciona el servidor perquè aquesta bidireccionalitat sigui possible.

El servidor que s'ha creat des de zero es basa en la utilització de tecnologies com NodeJS o serveis de Google Cloud per poder fer sostenible i escalable l'aplicació.

El nostre servidor s'allotja en una màquina virtual creada a partir del servei Compute Engine de GC. Aquesta conté les característiques necessàries per ser tan òptima com sigui possible quant a rendiment. Hem triat el tipus de màquina "n2-standard-8" per termes de tarifa i pressupost. Una màquina molt més potent suposa més diners per mantenir amb vida el servidor. I, realment, el requisit més necessari és la memòria, on hem triat 32 GB de RAM perquè així pugui suportar totes les peticions i respostes.

Perquè la màquina virtual es pugui comunicar amb l'exterior hem hagut d'utilitzar el servei de Xarxa de VPC, per tal d'obrir tants ports "websockets" com sigui necessari. Això es fa mitjançant la creació de regles Firewall on es creen dues regles, per entrada i sortida, i s'especifica, sobretot, l'interval IP que és "0.0.0.0/0" i el protocol i port, que és "tcp:65080-65090". Com podem veure, hem utilitzat únicament deu ports, ja que per al nostre projecte no en necessitem més. En cas de tenir molts més dispositius que s'ha de connectar, s'haurà d'incrementar aquesta quantitat de ports únicament.

El port que s'obre pel protocol HTTP i permet que la pàgina web pugui escoltar és el 80 pel servidor remot. En cas de ser localment és el 8888. Aquest port (80) ja està especificat a les regles Firewall per defecte al servei de VPC de Google, el qual és d'entrada.

Els ports que utilitza cada dispositiu s'especifiquen a un fitxer ".json" anomenat "sensors.json" on es concreta cadascun dels dispositius que es connecten al servidor i els seus atributs. Aquest fitxer ens serà d'utilitat en funcionalitats com la selecció d'un robot. Trobarem parelles de robots, ja que cada robot està format per un mòdul de control i un altre de càmera. Aquestes parelles estan especificades pel seu atribut "number". Més endavant veurem com es modifica l'atribut "selected" o com es crea l'atribut "image".

Un cop mencionat això, ja podem explicar la infraestructura del servidor amb NodeJS. El primer que es fa és importar totes les llibreries i APIs necessàries pel nostre servidor. Entre les més destacables, podem mencionar "ws" per la utilització de "Websockets" i las APIs de Google Cloud Speech, Vision i el servei de Storage.

Al ser un flux de dades complexe, l'explicació del funcionament serà per parts per poder acabar d'entendre correctament cadascuna de les funcionalitats. El codi que s'executa per llançar el servidor es divideix en dues parts fonamentals: el tractament de dades rebudes per part de l'usuari i el tractament de dades rebudes per part dels dispositius ESP32.

Les dades trameses des de l'usuari, de les quals hem explicat a l'apartat anterior com funciona el seu enviament, es realitza de la següent forma: Primer de tot, s'ha de crear una instància "WebSocket" on el port que s'utilitza és el 65080, pertanyent a la pàgina web. Aquesta instància "WebSocket" estarà a l'espera de qualsevol missatge que s'hagi enviat des del client. Dins d'aquesta part de codi es tenen en compte diferents aspectes, com el que hi hagi molts clients connectats a la pàgina web (mitjançant una llista de connexions), i el tractament de dades en cas que sigui àudio o una simple ordre enviada en format JSON. Les ordres enviades com a àudio a través d'un buffer es converteixen a "base64" per poder ser utilitzades per l'API de Google Cloud. Es configura el "request" i es crida a l'API Speech To Text per rebre la transcripció. No obstant, per facilitat, ens interessa que l'ordre sigui codificada com un enter, i per això s'utilitza una funció que ens retorna l'ordre en un número, per així poder tractar-la millor. Aquest

número s'utilitzarà per instanciar una variable global, anomenada "comando" que contindrà una cadena de text com "move=1", on el dispositiu al qual va adreçat aquesta ordre distingeix la clau i el valor separats per un "=". La interpretació d'aquesta ordre no entra dins d'aquest apartat.

Per altra banda, les ordres com les especificades a l'apartat anterior, que són enviades en format JSON, són rebudes i parsejades per facilitar la seva lectura. Pel cas dels botons de select, l'ordre rebuda conté un missatge com "selected=0/1". El valor, que és 0 o 1, s'utilitza per redefinir el valor de l'atribut "selected" del dispositiu que hagi estat seleccionat. Si l'ordre ve dels botons de moviment, s'aprofita que ja venen amb una Id numèrica, per la qual cosa no cal codificar-les, i es redefineix la variable global "comando" amb el valor corresponent. Aquesta variable canvia constantment, sempre que es rebi una ordre, tot i que una vegada s'ha enviat es consumeix.

Fins aquí el tractament de dades provinents de l'usuari. Ara passarem a explicar el funcionament de la segona part fonamental, la qual rep dades provinents dels dispositius ESP32 i envia les actualitzacions als clients.

Com ja hem mencionat, aquesta part té relació amb els dispositius robòtics. Per poder tractar cadascun dels dispositius per separat, es crea una instància "WebSocket" amb el port especificat en el fitxer "sensors.json" per cadascun. D'aquesta forma, tots estaran a l'espera de qualsevol event o missatge. Les dades que s'envien dels robots, que únicament són imatges, són tractades per tal de fer funcionar l'API de Google Cloud Vision, per detectar objectes. La imatge enviada com a buffer de dades es converteix a "base64" per poder ser utilitzada per l'API. Una vegada es crida a l'API, la qual retorna únicament els objectes detectats amb un 0.65 de confidència, es guarden a una variable anomenada "objects". Si aquesta variable no és buida, es dibuixa el "Bounding Box" de l'objecte detectat amb llibreries com "Jimp". La imatge s'actualitza pel dispositiu en concret al seu atribut "image". En cas de que no s'hagi trobat cap objecte, no es fa cap procés i la variable "image" és queda amb la imatge original, en "base64".

En aquest apartat també es duu a terme l'enviament de la variable "comando" als dispositius que estan connectats. Això es durà a terme únicament si la variable "comando" no és nul·la i si el dispositiu que està escoltant té l'atribut "selected" a 1, que vol dir que les ordres estan habilitades per aquest. Una vegada enviada la comanda, es posa el valor de la variable a "null" per rebre la següent instanciació.

Finalment, s'envien els atributs actualitzats dels dispositius al client, fent possible així la visualització dels vídeos dels robots correctament, per exemple.

2.4. Informe de Deteccions

Una altra funcionalitat del servidor és la de crear un informe després d'haver acabat el recorregut dels robots, on es podrà seleccionar exactament el robot a analitzar.

Per poder realitzar aquesta funció, hem hagut d'implementar una part de codi que envia frames a un "bucket" de Google Cloud. A cada robot li correspon una subcarpeta dins del "bucket" on s'emmagatzemen aquests frames. Per poder enviar els frames del servidor al Bucket del projecte s'ha utilitzat el servei de Storage. Els robots enviaran, mitjançant una funció on es realitza la connexió amb el bucket, 1 frame de cada 10. Tenint en compte que la latència és baixa i tenim una velocitat mitjana d'uns 30 fps per segon, guardar una sola imatge de cada deu no significa una pèrdua de dades elevada, sobretot si considerem que els robots no es mouran ràpidament i no es saltaran objectes.

Un cop tenim tots els frames corresponents al recorregut que s'ha fet, haurem de crear un vídeo i utilitzar l'API de Video Intelligence de Google Cloud per analitzar tots els objectes possibles. Per crear-lo hem realitzat un script en Python que accedeix al "bucket" i transforma tots els frames ordenats en un vídeo. Un cop el tenim, en un altre script, també en Python, fem una crida a la API de Video Intelligence i especifiquem el "path" d'aquest. El script s'encarregarà de deixar l'informe el més visual possible per la seva màxima comprensió.

Aquest informe el podem considerar com una cosa addicional al nostre projecte que ens dona una idea de com de bé hem analitzat l'entorn durant el recorregut del robot.

Per acabar, el diagrama general del funcionament del servidor és el següent (Figura 3):

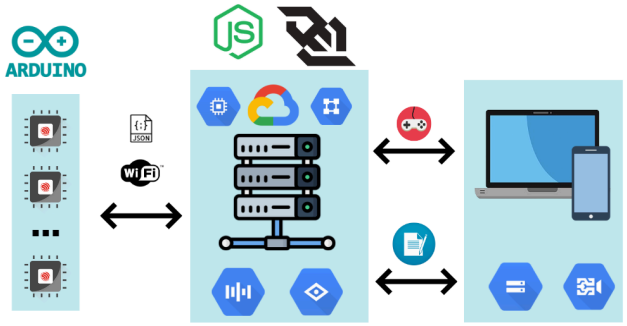

Figura 3. Diagrama de funcionament de l'aplicació

3 Problemes i Solucions

Com tot projecte, ens hem trobat amb diversos problemes, encara que, majoritàriament, tots tenen solució. A continuació mencionarem els dos problemes més destacables del nostre projecte.

El primer a mencionar tracta sobre la memòria del Heap de JavaScript. Aquest problema es dona quan fem una gran quantitat de peticions al servidor. El que fem així és escurçar la vida del servidor. Si comencem a fer moltes peticions, arribarà un punt en què la memòria destinada als programes en JavaScript no suportin més la càrrega i deixin de funcionar. Una solució fàcil, però temporal, és la d'expandir la quantitat de memòria destinada. Generalment, s'utilitzen com a màxim 4GB, però el que hem fet nosaltres és augmentar-la fins a 16GB. Com ja hem dit, aquesta és una solució temporal, ja que arribarà un punt on es torni a arribar al límit i el servidor deixi de funcionar. Per tant, una opció més viable i que no ha estat implementada per falta de coneixement, seria alliberar el màxim de memòria possible en tot moment.

Un altre problema que hem tingut està relacionat amb els permisos per la utilització del micròfon. Quan es tracta de servidors remots, on s'utilitza un protocol HTTP, obtenim certs problemes per la utilització de dispositius de l'usuari. El convenient és utilitzar un protocol HTTPS, el qual és molt més segur. Localment no tenim aquest problema perquè ens trobem dins de la nostra xarxa. En canvi, per poder fer ús de recursos "media" com el micròfon del dispositiu de l'usuari, no és tan trivial. Entre totes les solucions possibles, on la recomanable és utilitzar eines com Apache pel certificat SSL, nosaltres hem utilitzat una solució molt més ràpida però manual. Si als flags de Google Chrome ens dirigim a l'apartat de "Insecure origins treated as secure", fiquem la URL de la nostra pàgina web i donem a activar, ja se'ns permet fer ús dels recursos "media" del dispositiu i ja tindrem disponible la funcionalitat d'ordres per veu.

Finalment, cal comentar la dualitat remot-local, tot i que no el podem considerar un problema, però si un aspecte al qual adaptar-nos segons es vulgui fer d'una forma o una altra. Durant tot l'informe hem mencionat quines són les coses que canvien segons es vulgui fer des d'una màquina virtual amb Compute Engine o si es vol fer localment. Ja hem anat comentant que l'únic que varia són els ports i les adreces IP. Cadascun ha de considerar de quina forma li resulta més assequible llençar el servidor i quina és la més adequada per les seves necessitats. Segons això, podem considerar doncs que el nostre servidor està totalment adaptat a les necessitats de l'usuari que el vulgui utilitzar.