Osservazioni:

Limiti imposti da mediapipe:

1. **Framerate limitato**: comprensibile contando che siamo su dispositivi mobile, meno elaborazioni facciamo meno batteria viene consumata]
2. **Dati imprecisi quando non tutte le dita vengono mostrate**: se si mostra una gesture che non espone tutte le dita bene in vista(pollice in su) e si muove la mano cambiando punto di vista, le coordinate rilevate dalla libreria non rimangono coerenti
3. **Applicazione basata su immagini ben illuminate**: se nello stream video manca di luminosità l’accuratezza del rilevamento cala drasticamente (problema noto nel mondo di computer vision)

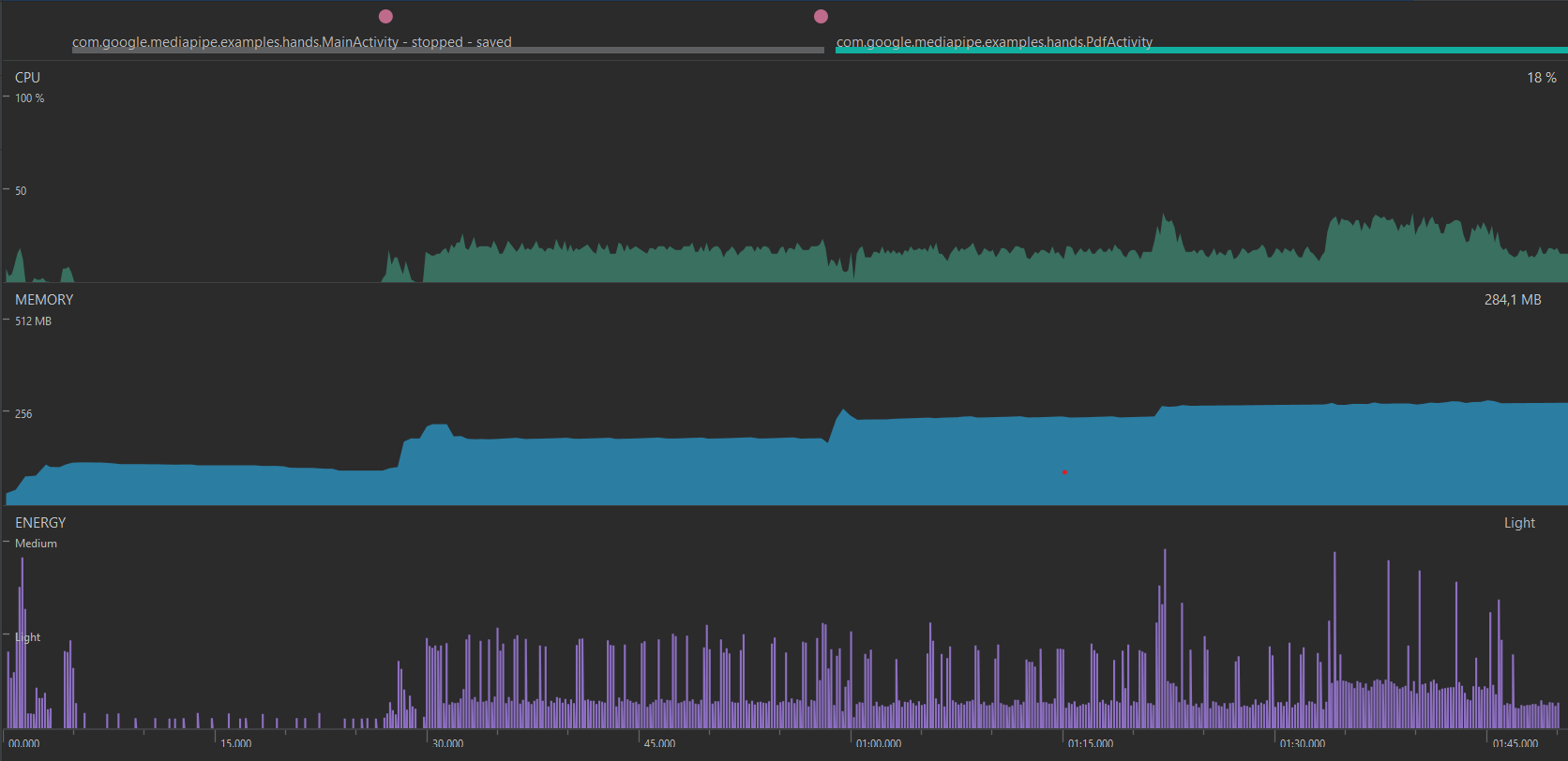
Limiti della nostra soluzione:

1. **Consumo batteria relativamente elevato:** per una app mobile porre attenzione ai consumi è fondamentale, la nostra applicazione consuma un 2% ogni 5 minuti circa su una batteria 4000mAh. Questo consumo è comprensibile considerando il lavoro che svolge, un fattore critico è rappresentato dall’impossibilità di avere momenti di stand-by in quanto il tracking delle mani lavora anche quando non vi sono mani da riconoscere, portando i momenti morti(dove non accade nulla ma l’applicazione è attiva) ad una fonte di spreco risorse.
2. **Lentezza di gesture:** essendo il framerate limitato bisogna effettuare le gesture con calma per stare dietro alle elaborazioni
3. **Gesture non eccessivamente elastiche:** L’applicazione utilizza le proporzioni usano come riferimento la lunghezza fisica del dito medio fino ad arrivare al centro del polso, tutte le altre misure rilevate sono comparate con quella. Dal punto di vista teorico dovrebbe funzionare su tutte le mani, ma, dovendo discriminare le gesture in modo abbastanza rigido per evitare che gesture differenti si confondano tra di loro, per effettuare correttamente i segni in alcuni casi le dita vanno orientate in un modo specifico (crab gesture= il pollice deve essere particolarmente inclinato (vincolo posto per non confondere tale gesture con il pinch))

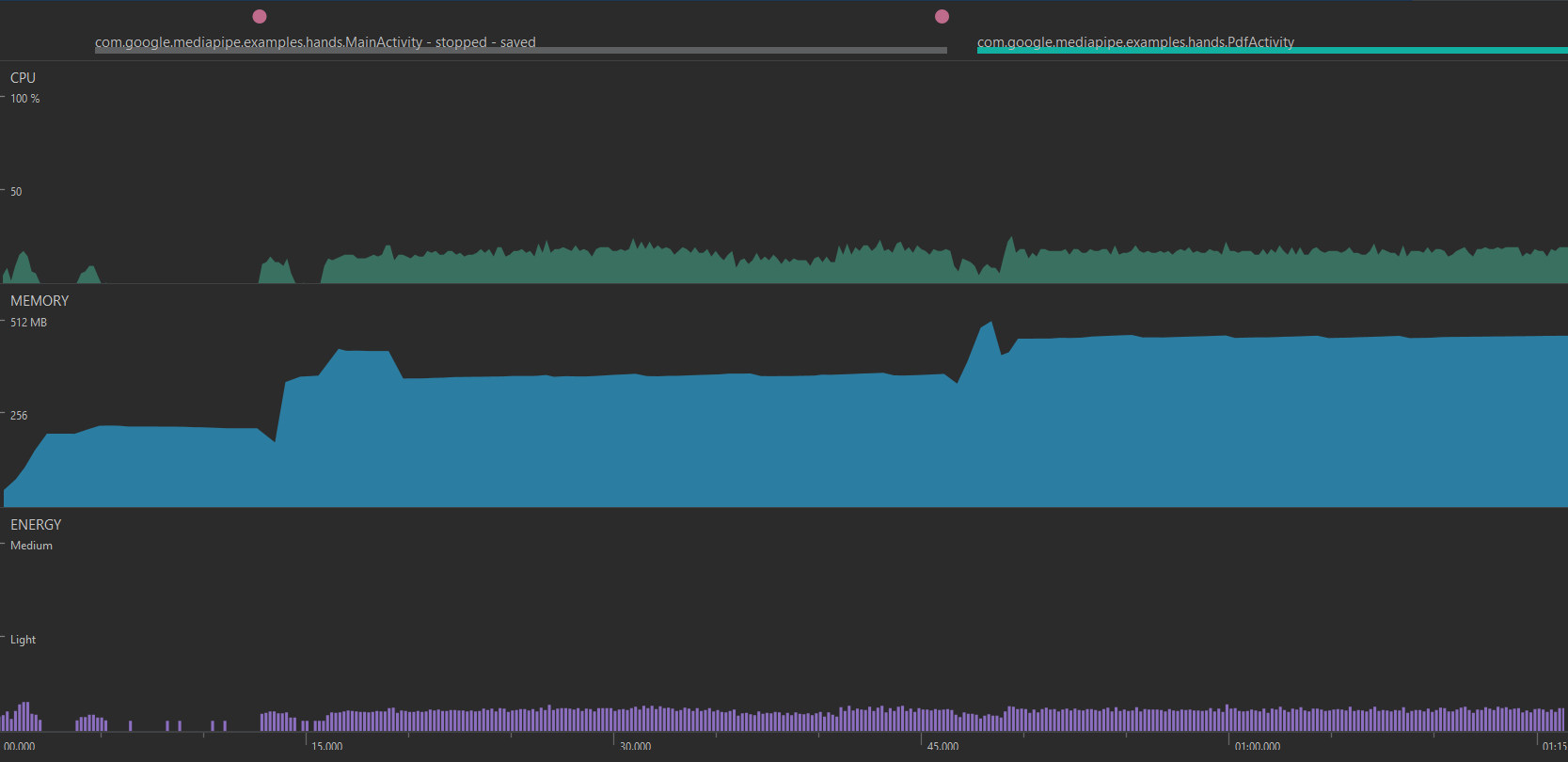
Dati raccolti da Android profiler su dispositivo con seguenti specifiche:

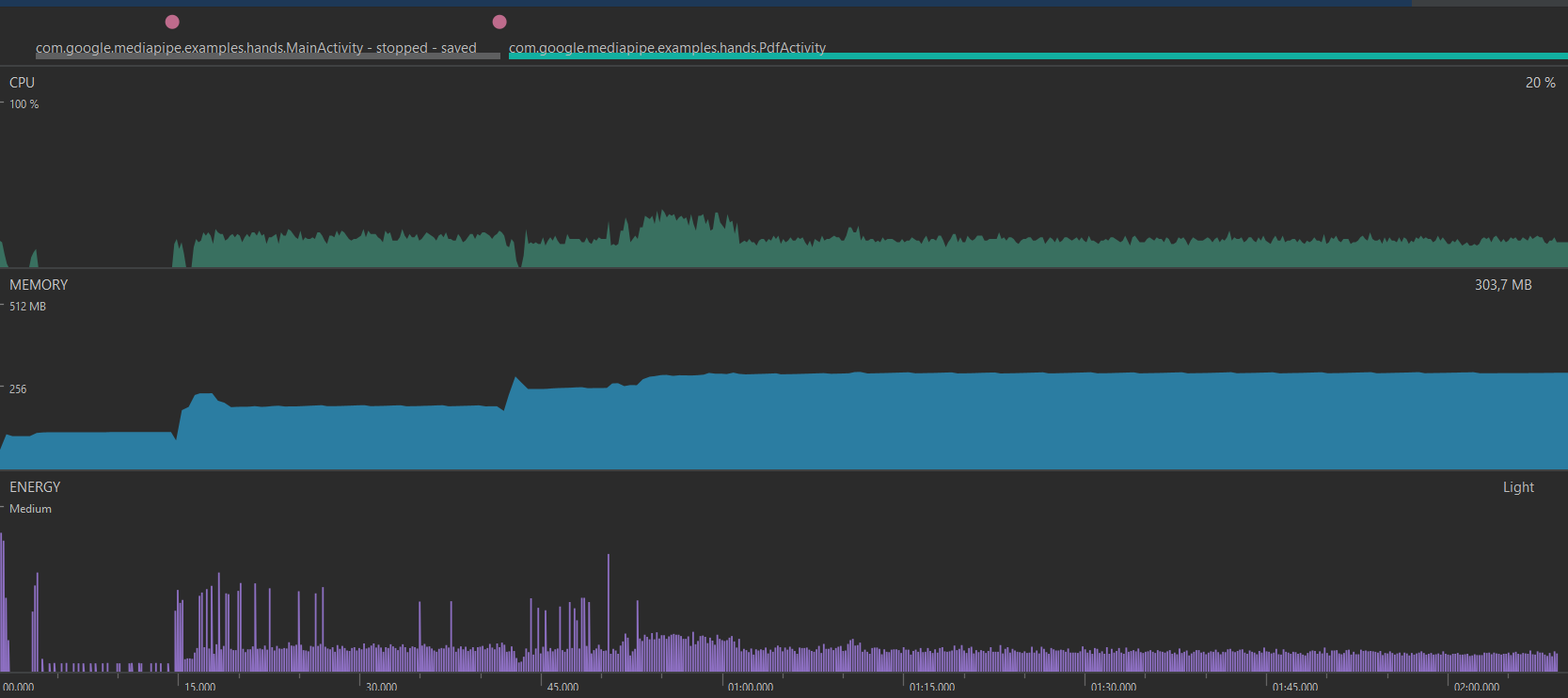
* **GPU**: ARM Mali-G72 MP3 (850MHz)
* **CPU**: Octa-core, 2 processori: 4x 2.3GHz ARM Cortex-A73 (Quad-core), 4x 1.7GHz ARM Cortex-A53 (Quad-core)
* **RAM**: 6GB

Dati raccolti:









Sviluppi futuri & applicazioni in altri ambiti:

[spunti di chat gpt:

* Integrazione con l'Internet of Things (IoT): le applicazioni di riconoscimento gesti potrebbero essere utilizzate per controllare altri dispositivi intelligenti nella casa, come luci, termostati e sistemi di sicurezza.
* Realtà aumentata: le applicazioni di riconoscimento gesti potrebbero essere utilizzate per interagire con oggetti virtuali nell'ambiente reale.
* Comandi vocali: la combinazione di riconoscimento gesti e comandi vocali potrebbe offrire una maggiore flessibilità e precisione nell'interazione con i dispositivi mobili.
* Gaming: le applicazioni di riconoscimento gesti potrebbero essere utilizzate per creare giochi più immersivi e coinvolgenti.
* Riconoscimento delle espressioni facciali: le applicazioni di riconoscimento gesti potrebbero essere integrate con il riconoscimento delle espressioni facciali per fornire un'interazione più naturale e intuitiva con i dispositivi mobili.

]

La possibilità di poter interagire con un dispositivo tramite una videocamera introduce un metodo alternativo di interfacciamento con la macchina alla tastiera. Nel corso degli anni i computer si sono evoluti sempre di più mentre il metodo per interagire con essi è rimasto sempre lo stesso: la tastiera. Solo nell’ultimo decennio, grazie ad una sufficiente maturazione della tecnologia, si è cominciato concretamente a pensare a mezzi differenti per comunicare con i dispositivi. D’altronde siamo di fronte ad una situazione dove due macchine computazionali potenti(cervello e computer) che nella loro collaborazione sono ostacolate dall’interfacciamento Input/Output che gli si pone in mezzo, per questo soluzioni che rendono più immediato e naturale il trasferimento dati saranno probabilmente l’argomento di discussione principale degli anni a seguire. Il nostro team si è occupato di analizzare un “proof-of-concept” di una applicazione PdfReader gestita non dall’interfacciamento proposto dello schermo touch ma dalla videocamera frontale che rileva specifiche gesture. Svolgendo i test finali siamo rimasti soddisfatti del risultato, l’unico tallone d’Achille è il fatto di avere un riconoscimento gesti in alcuni casi abbastanza rigido, dovuto proprio ai dati non molto precisi che mediapipe pone quando non tutte le dita vengono mostrate, per evitare che gesture differenti si sovrappongano. Una idea di risoluzione sarebbe porre due videocamere che riprendono due angolazioni diverse e poi elaborate insieme per ricostruire la posizione delle dita. Questa proposta però non avrebbe molti punti a favore considerando che nei dispositivi già sul mercato non sarebbe compatibile oltre al fatto di svolgere il doppio delle elaborazioni, ma per prodotti non mobile ancora da inserire sul mercato come una lavagna LIM che adotta questa tecnica non sarebbe una idea da scartare a priori.

In questo progetto ci si è limitati a gestire un pdf, ma una volta che la logica di gesture detection è stata messa a punto e anche il consumo migliorato ecco che cambiare scenario diviene immediato, è sufficiente collegare le gesture a delle azioni, esempio potrebbe essere la simulazione del cursore tramite gesture, quindi navigare nell’interfaccia proposta dallo schermo. Combinando poi il riconoscimento gesture con altri tipi di riconoscimenti(face, body …) potrebbe nascere qualcosa di interessante. Rimanendo però nel mondo mobile potendo consumare poca batteria e di conseguenza non avendo una grande potenza computazionale a disposizione, [?sarebbe valido proporre una soluzione dove l’applicazione non elabora direttamente i dati ma delega un server remoto, non dovrebbero esserci problemi di latenza dovuti alla connessione?] il framerate limitato non permette di riconoscere accuratamente gesture dinamiche veloci, ciò nonostante si riescono a realizzare applicazioni con interazioni più naturali rispetto ai classici pulsanti.

**Tecniche adottate per ciascuna gesture:**

Ci siamo occupati di riconoscere due tipologie di gesture: statiche e dinamiche. Gli strumenti utilizzati sono stati i **vettori** per rilevare gli spostamenti e le **distanze dei vari punti rispetto al centro del polso**, che identifica la nostra origine. La scelta di adottare le distanze dal polso rimane valida perché le dita si possono muovere solo di 180°, gli unici punti su cui porre attenzione sono quelli del pollice che spesso assumono la stessa distanza dal polso con gesture diverse:

Immagine che contiene persona

Descrizione generata automaticamente

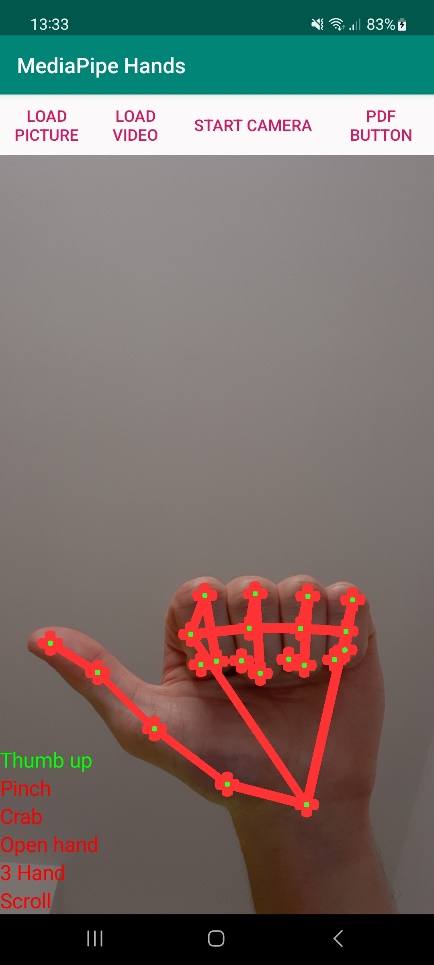




Quindi la distanza punto-centro polso può essere utilizzata come discriminante, questione diversa sarebbe stata se le dita fossero state in grado di muoversi liberamente a 360°. Infine, ritornando alla problematica del pollice, in alcuni casi è stato necessario verificare se un dito preso in analisi fosse dritto o piegato.

Piccola anticipazione necessaria ma che verrà trattata meglio nel prossimo capitolo: per le distanze sono state adottate le proporzioni prendendo come punto di riferimento la lunghezza definita dal polso, lungo tutta la falange, fino ad arrivare alla punta del dito medio. In questa maniera non si lavora su lunghezze assolute e si permette all’applicazione di riconoscere gesti di persone diverse. Inoltre questa lunghezza è stata suddivisa in 70 livelli, numero non a caso ma scelto empiricamente in base alla precisione/affidabilità dei dati offerti da MediaPipe: un numero di livelli maggiore significava quantizzare le distanze rilevate dalla libreria conservando molta della loro precisione; non possiamo affidarci troppo ai dati offerti, ma considerare un certo margine di errore, andando ad introdurlo nella fase di quantizzazione**(COLLEGAMENTO AL CAPITOLO DEI LIVELLI DI RIK)**. In seguito vediamo le tecniche adottate per ciascuna gesture, raggruppate per tipologia:

**Gesture statiche:**

**Thumb Up:** per riconoscere un “pollice in su” abbiamo intato rilevato la distanza delle sole estremità delle dita dal polso, identificando le seguenti lunghezze da rispettare con un margine di errore di 8 livelli:

(HandPoints.THUMB\_TIP, 52)

(HandPoints.INDEX\_TIP, 35)

(HandPoints.MIDDLE\_TIP, 29)

(HandPoints.RING\_TIP, 27)

(HandPoints.PINKY\_TIP, 29)

Il pollice richiede un ulteriore controllo per il motivo decritto nei paragrafi precedenti, viene verificato che sia in posizione dritta.

**Tre Hand:** per il “tre” la strategia è stata analoga a quella del “pollice in su”, adottando i seguenti livelli:

(HandPoints.THUMB\_TIP, 48)

(HandPoints.INDEX\_TIP, 65)

(HandPoints.MIDDLE\_TIP, 67)

(HandPoints.RING\_TIP, 39)

(HandPoints.PINKY\_TIP, 32)

**Four Hand:** per il “quattro” vale la stessa cosa, con livelli:

(HandPoints.THUMB\_TIP, 42)

(HandPoints.INDEX\_TIP, 63)

(HandPoints.MIDDLE\_TIP, 67)

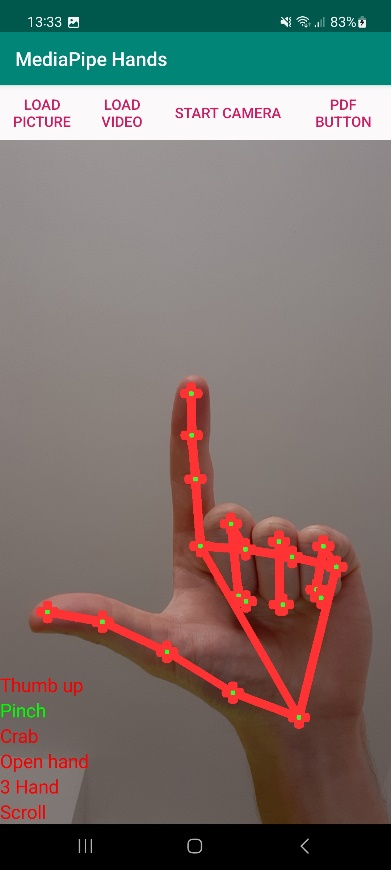
(HandPoints.RING\_TIP, 62)

(HandPoints.PINKY\_TIP, 54)

**Gesture dinamiche:**

Immagine che contiene testo

Descrizione generata automaticamente**Crab Hand:** qui diversamente da quanto detto prima, ci si occupa di verificare che le punta del pollice ed indice si tocchino e che il pollice sia piegato. Oltre a rilevare la gesture era necessario sapere di quanto si spostava la punta del pollice mentre il riconoscimento del gesto era valido: essendo i valori offerti da mediapipe già in forma di coordinate normalizzate rispetto allo schermo è stato sufficiente andarsi a costruire un vettore con partenza il primo punto rilevato appena la gesture viene riconosciuta e come punto di destinazione l’ultimo punto rilevato.

**Pinch:** si verifica che le distanze punta del medio, anulare e mignolo rispetto al polso siano rispettate con i livelli:



(HandPoints.MIDDLE\_TIP, 29)

(HandPoints.RING\_TIP, 27)

(HandPoints.PINKY\_TIP, 29)

Successivamente si rileva la distanza tra la punta dell’indice e del pollice.

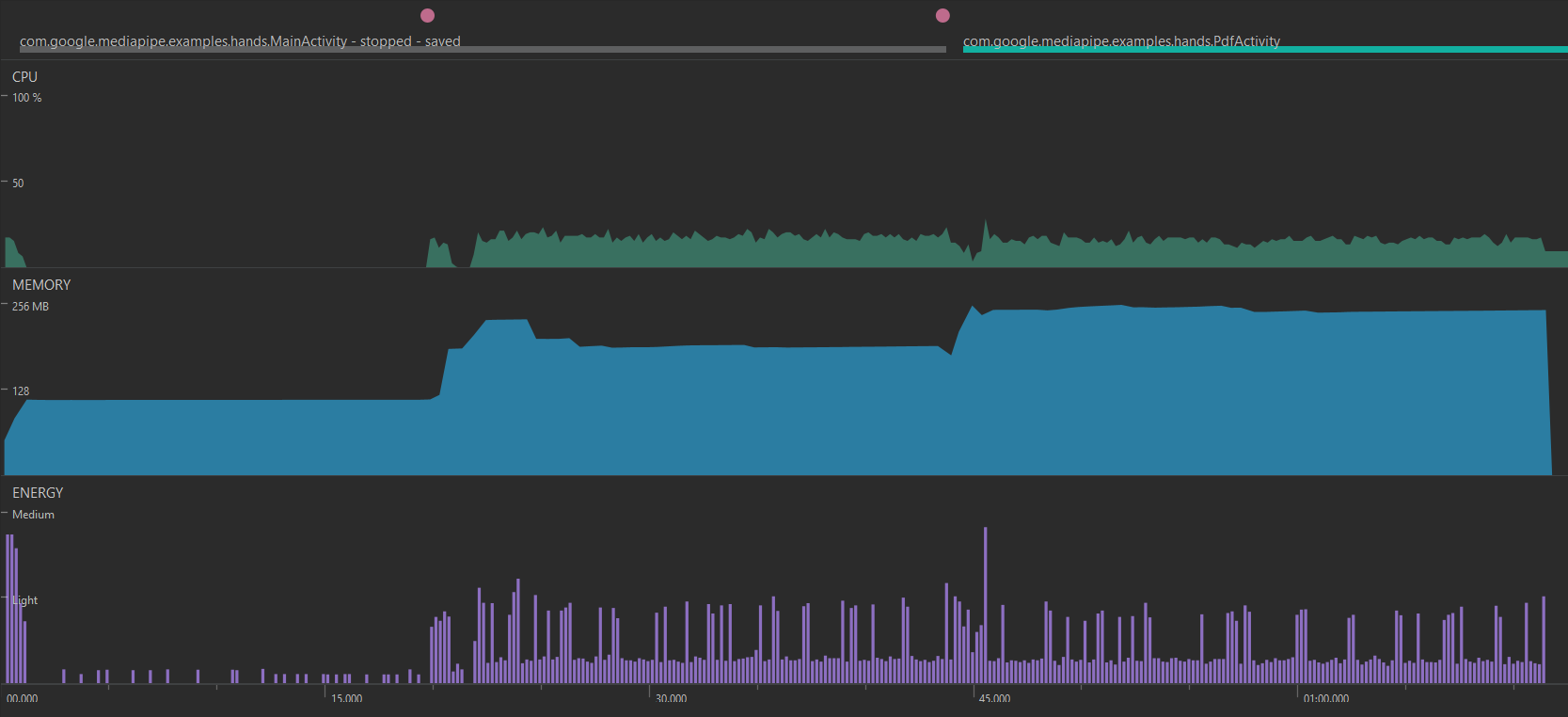




**Scroll:** si controlla che le due dita indice e medio siano alzati, mentre il pollice non deve trovarsi in posizione retta. Per tutto il tempo che la gesture viene riconosciuta ci si impegna a tenere traccia dei movimenti cercando di rilevare quando le dita si spostano superando una velocità e distanza soglia.

PERFORMANCE:

L’ambiente di sviluppo Android Studio offre un comodissimo pannello per verificare il consumo relativo delle risorse utilizzate dalla propria applicazione. La seguente immagina riassume i nostri valori rilevati:



Il profiling è suddiviso in tre sezioni che corrispondono alle tre pagine visualizzabili, in ordine: pagina home, pagina start camera e pagina PdfReader. Nel complesso non abbiamo misurazioni stravolgenti, la CPU viene utilizzata mediamente il 17% del suo potenziale quando il tracking delle mani è attivato (start camera e PdfReader), mentre alla home page dove non succede nulla giustamente il consumo è nullo tranne che al suo primo caricamento. La memoria viene utilizzata principalmente nella gestione dei Pdf con picchi massimi di 256MB. Mentre sotto il punto di vista dei consumi Android Studio pone nell’asse delle ordinate tre livelli: Light, Medium e High; al di là dei cambi di pagina che vengono etichettati come Medium, il consumo generale viene considerato Light. Siccome questo dato non ci ha orientato più di tanto per capire se tale consumo possa essere considerato tollerabile, come accennato nel paragrafo precedente, abbiamo provato molto empiricamente a verificare di quanto scendeva il livello di batteria durante l’utilizzo dell’app realizzando un calo di un 2% ogni 4 minuti con una batteria 4000mAh. Facendo lo stesso con la nota app Instagram il consumo è risultato differente, con un uso di un 2% ogni 7 minuti, quindi messa a confronto la nostra applicazione consuma il doppio. Ipotizzando uno scenario realistico dove consideriamo il 100% di batteria di un utente intenzionato a studiare con la nostra applicazione(con le specifiche di dispositivo citate prima) e dedicando un 80% alla nostra applicazione si ottiene una durata massima di 2 ore e mezza. Esito non malvagio ma pensare che dopo 2 ore e mezza di lettura il dispositivo diviene inutilizzabile non è molto invitante, i ruoli però si potrebbero invertire se cambia il contesto dove si utilizza il riconoscimento gesture.