Brumariu Cosmin-Nicusor

Optimizarea metodei de binarizare Otsu folosind extensii DSP pe Raspberry Pi Pico 2 W

Conducător științific: Șef lucr. dr. ing. KERTÉSZ Csaba Zoltán

**Abstract**

This paper presents the implementation and optimization of the Otsu image binarization method on the Raspberry Pi Pico 2 W platform. The Otsu algorithm automatically determines the optimal threshold for separating background and foreground based on inter-class variance. Optimization was achieved using the CMSIS-DSP library, which leverages the DSP extensions of the ARM Cortex-M33 core. The optimized version was benchmarked against a non-optimized implementation, evaluating execution time and image quality metrics such as PSNR and SSIM. Results demonstrate a ~39× speed increase without compromising output quality, showcasing the potential of low-cost microcontrollers for real-time image processing applications.

# 1. Rezumat

Această lucrare propune implementarea și optimizarea metodei de binarizare Otsu pe platforma embedded Raspberry Pi Pico 2 W. Algoritmul determină automat pragul optim de separare a fundalului și obiectelor într-o imagine, pe baza varianței inter-clasă. Optimizarea a fost realizată prin utilizarea funcțiilor din biblioteca CMSIS-DSP(Cortex Microcontroller Software Interface Standard – Digital Signal Processing), care valorifică extensiile de procesare semnal ale nucleului ARM Cortex-M33. Performanța metodei este comparată cu o implementare neoptimizată, analizând timpul de execuție și calitatea binarizării prin metrici PSNR și SSIM. Proiectul evidențiază potențialul microcontrolerelor low-cost pentru procesare eficientă de imagini în timp real.

# 2. Introducere

Binarizarea unei imagini este un pas esențial în procesarea imaginilor, folosit pentru extragerea contururilor sau a trăsăturilor importante dintr-un cadru vizual. Metoda Otsu este una dintre cele mai cunoscute tehnici automate de alegere a unui prag optim. Aceasta a fost introdusă de Nobuyuki Otsu în 1979 [1] și este utilizată pe scară largă în prelucrarea imaginilor. În acest proiect, algoritmul a fost implementat pe Raspberry Pi Pico 2 W, un microcontroler cu două nuclee ARM Cortex-M33 și extensii DSP. Proiectul compară două implementări, una clasică și una optimizată prin CMSIS-DSP, și analizează avantajele oferite de procesarea pe instrucțiuni specializate.

# 3. Arhitectură hardware și software

Raspberry Pi Pico 2 W este echipat cu microcontrolerul RP2350, care conține două nuclee ARM Cortex-M33 tactate la 150 MHz și suportă extensii DSP și FPU [3]. Proiectul a fost implementat în limbaj C++, folosind toolchain-ul standard arm-none-eabi-gcc și biblioteca CMSIS-DSP dezvoltată de Arm Ltd. [2]. Codul implementează generarea unei imagini artificiale, calculul histogramei intensităților, aplicarea algoritmului Otsu (optimizat și neoptimizat), și evaluarea PSNR/SSIM.

# 4. Algoritmul Otsu și optimizarea sa

Metoda Otsu determină pragul de binarizare optim prin maximizarea varianței între două clase: fundal și obiect [1]. Algoritmul pornește de la histograma unei imagini în tonuri de gri (valori între 0 și 255), apoi testează toate valorile posibile de prag. Pentru fiecare valoare, imaginea este împărțită în două clase: fundal (pixeli ≤ prag) și obiect (pixeli > prag). Pentru fiecare împărțire se calculează media și proporția fiecărei clase, iar separabilitatea este măsurată prin varianța inter-clasă.

*σ²\_b(t) = ω₀(t) \* ω₁(t) \* [μ₀(t) - μ₁(t)]²*

- ω₀(t), ω₁(t): proporțiile pixelilor în clasele fundal și obiect  
- μ₀(t), μ₁(t): media intensităților în cele două clase  
Se caută valoarea t care maximizează această varianță, separând optim cele două regiuni din imagine.

Pentru evaluarea calității binarizării au fost utilizate două metrici standard: PSNR și SSIM. PSNR (Peak Signal-to-Noise Ratio) exprimă cât de apropiată este imaginea binarizată de original, în timp ce SSIM (Structural Similarity Index) evaluează similaritatea structurală dintre cele două imagini.

*PSNR = 10 \* log₁₀ (MAX² / MSE)*

- MAX este valoarea maximă posibilă (255 pentru imagini pe 8 biți)  
- MSE este media pătratelor diferențelor dintre pixelii originali și binarizați.

*SSIM(x, y) =*

- μₓ, μᵧ: mediile imaginilor  
- σₓ², σᵧ²: varianțele  
- σₓᵧ: covarianța între imagini  
- C₁, C₂: constante pentru stabilitate numerică

Tabel 1 – Medii obținute pe 10 de imagini cu zgomot variabil.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Versiune | Timp mediu execuție (µs) | PSNR mediu (dB) | SSIM mediu | Raport viteză (x) | Prag mediu Otsu |
| Optimizată | 642.78 | 18.89 | 0.796 | 38.86 | |  | | --- | | 62.65 |  |  | | --- | |  | |
| Neoptimizată | 24975.49 | 18.89 | 0.796 | 1.0 | |  | | --- | | 62.65 |  |  | | --- | |  | |

Rezultatele arată că varianta optimizată este în medie de aproximativ 38.9 ori mai rapidă decât cea neoptimizată, menținând aceleași niveluri de calitate (PSNR și SSIM).

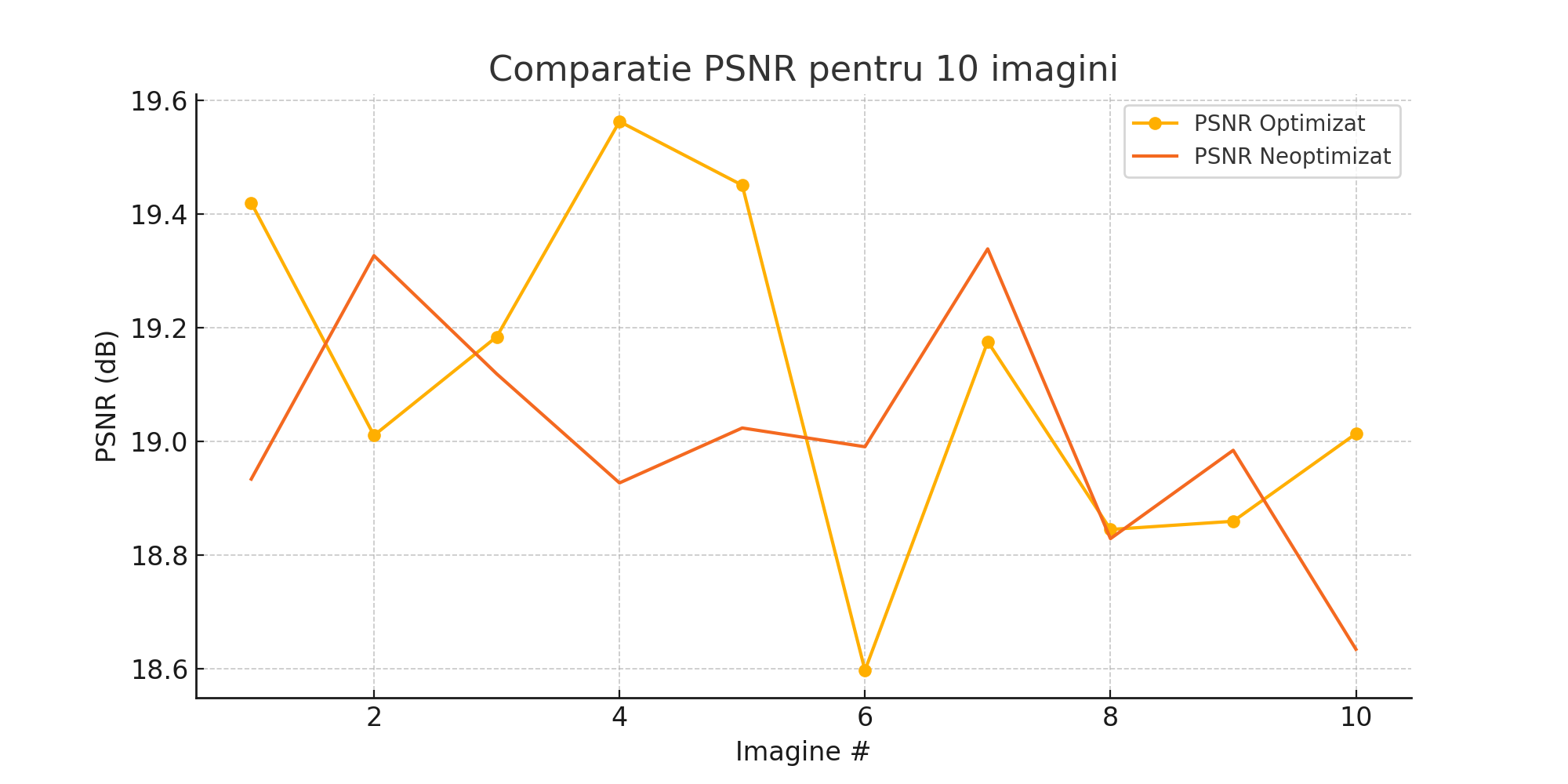


Figura 3 – PSNR pentru 10 imagini testate

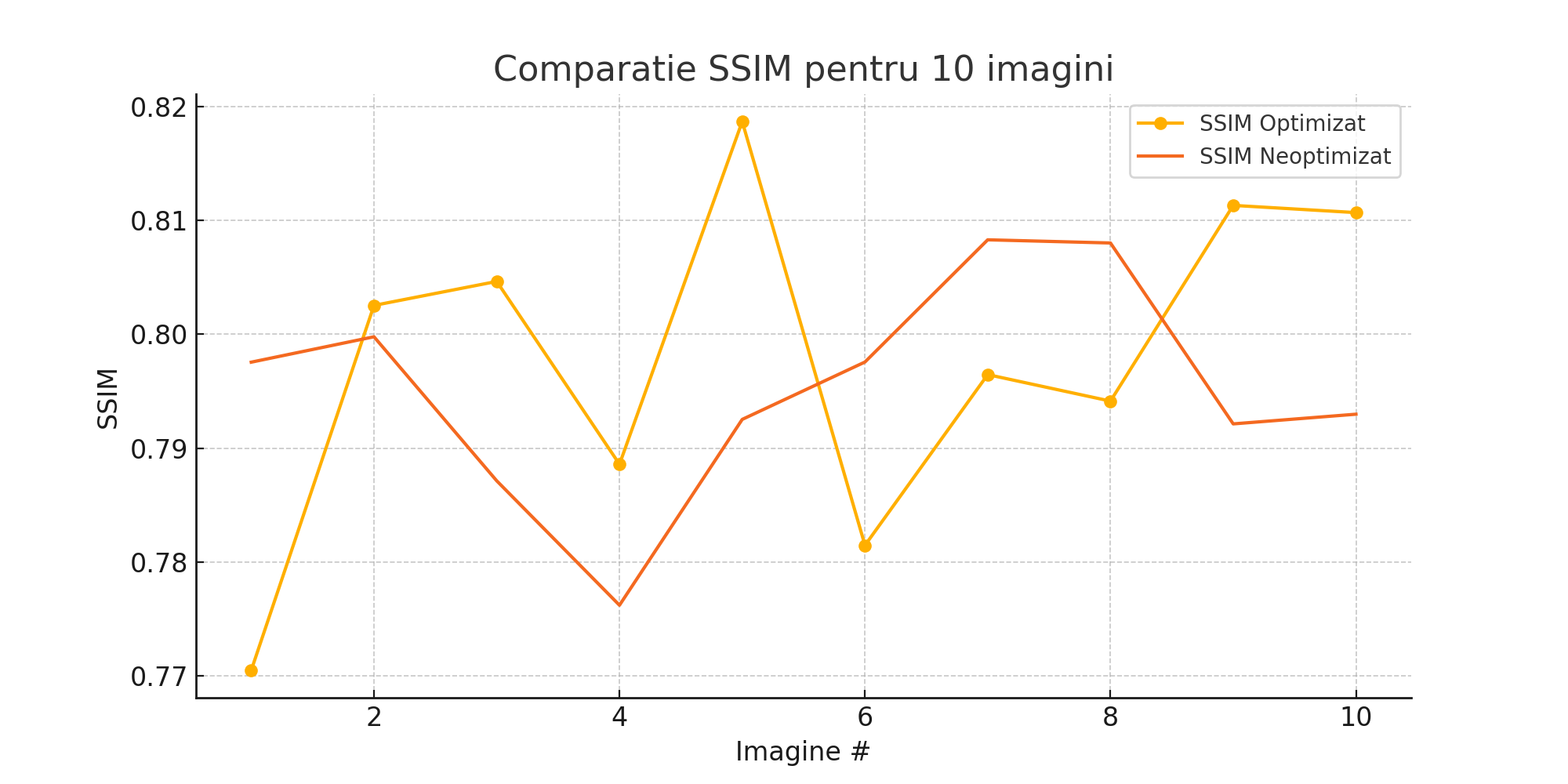


Figura 4 – SSIM pentru 10 imagini testate

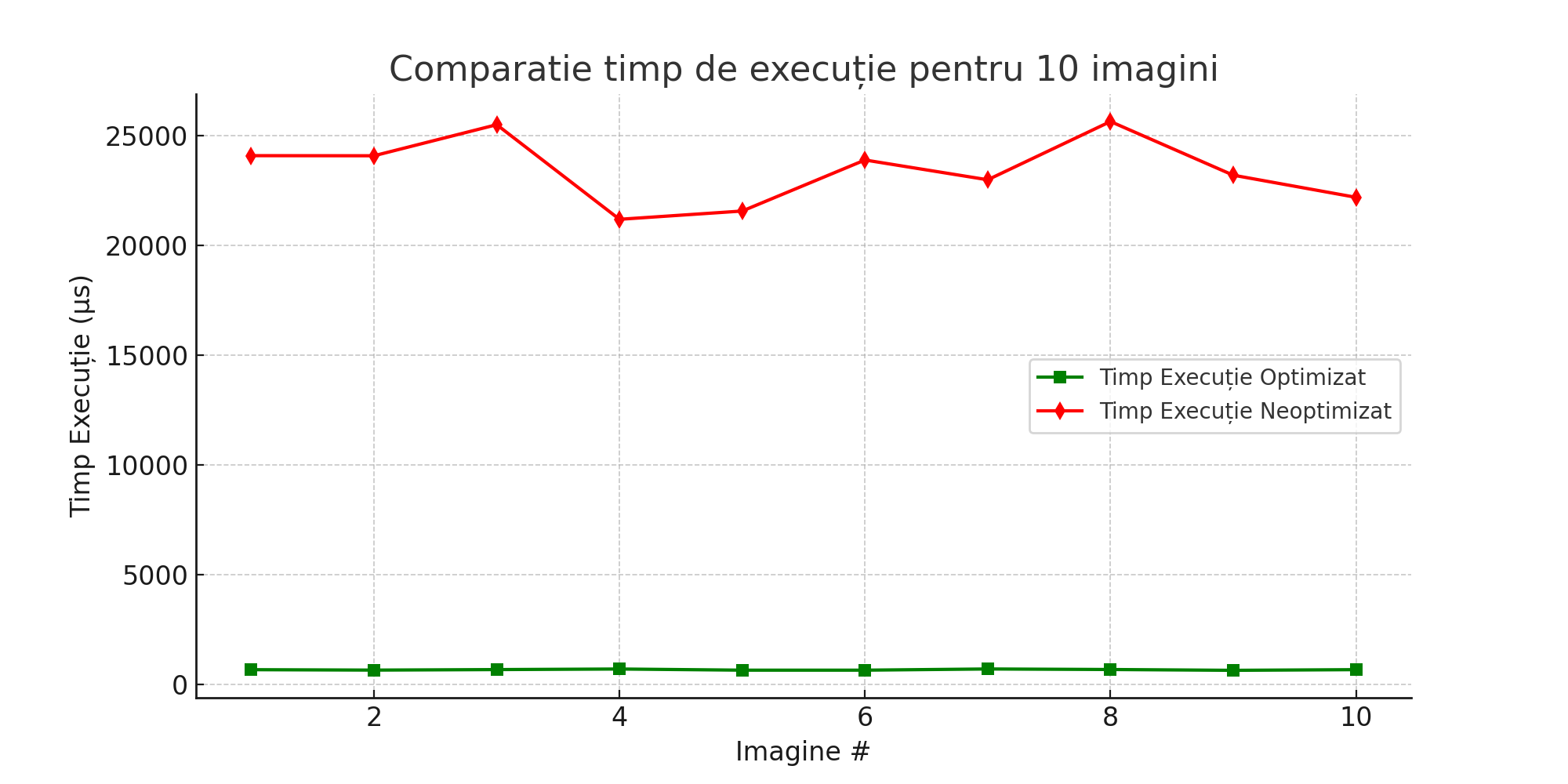


Figura 5 – Timp de execuție optimizat vs neoptimizat

# 6. Concluzii și perspective

Metoda Otsu poate fi eficient optimizată pe microcontrolere low-cost utilizând CMSIS-DSP. Acest tip de procesare este aplicabil în recunoaștere vizuală, sisteme embedded de control sau aplicații IoT. Pentru viitor, se propune extinderea proiectului prin utilizarea datelor reale din senzor, integrarea cu cameră sau rularea pe două nuclee.

# 7. Bibliografie

[1] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” IEEE Transactions on Systems, Man, and Cybernetics, 1979.

[2] Arm Ltd., “CMSIS-DSP Software Library Documentation,” [Online]. Available: https://developer.arm.com

[3] Raspberry Pi Ltd., “Raspberry Pi Pico 2 W Datasheet,” [Online]. Available: https://datasheets.raspberrypi.com/pico2w/pico2w-datasheet.pdf

[4] S. W. Smith, The Scientist and Engineer’s Guide to Digital Signal Processing. California Technical Publishing, 1997. [Online]. Available: https://www.dspguide.com

[5] ARM, “Cortex-M33 Technical Reference Manual,” [Online]. Available: https://developer.arm.com/documentation/100235