**TEXT READER – PROIECT ADIV**

**2020-2021**

**Nume**: Mateescu Viorel-Cristian

**Grupa**: 322 CB

**Coordonator**: Surpateanu Adrian

**Obiectiv:**

Dezvoltarea acestui proiect la nivel mondial a adus nenumarate beneficii in societate, mai ales pentru persoanele care au diverse probleme in vorbire, dar si in dezvoltarea Artificial Intelligence.  
Prefer o prezentare proprie in vederea opiniei mele personale viz-a-vi de acest subiect. Ideea acestui proiect mi-a venit in momentul in care acum ceva timp un inginer IT, sau mai degraba un Developer, specializat in domeniul de Inteligenta Artificiala (A.I), s-a folosit de cel mai usor limbaj de programare Python, pentru a crea un program de recunoastere a limbajului semnelor mainilor, avand la indemana doar o camera web. In momentul de fata, nu este foarte bine dezvoltat acest subiect, ci doar folosit in diverse domenii industriale, neaccesibile publicului larg. Cel mai bun exemplu sunt masinile Tesla, care se folosesc de un pilot automat bazat pe inteligenta artificiala si automatizata (invata si se adapteaza/corecteaza in timp real). (1)

1. https://www.youtube.com/watch?v=\_1MHGUC\_BzQ&t=1s

Ar fi fost foarte folositor o aplicatie pe telefon care sa fie bazata pe camera web si recunoasterea literelor gestionate doar prin maini. La televizor, sunt angajate persoane (rar) care s-a exprime cuvinte prin semne, oferind pentru persoanele cu dizabilitati o intelegere a subiectului difuzat.  
In prezent sunt aplicatii care fac acest lucru, dar doar pentru text printat sau tiparit, ci nu pentru un limbaj mai avansat / greu, precum cel al semnelor “real time”.

Acest proiect de “intelegere” este extrem de util si uneori poate fi singura metoda viabila de traducere al unui limbaj de comunicare. Fie ca suntem intr-o tara straina si vedem un panou, chiar si cu o aplicatie de traducere precum “Google Translate”, nu putem sa intelegem in totalitate semnificatia respectiva, daca nu vorbim limba respectiva. Prezenta unui “Text Reader” schimba total modul de lucru. Nu trebuie tastat manual sau s-a cunoastem limba din tara respectiva. Acesta poate analiza limba tinta, oferind identificarea instantanee a celor scrise.

**Descrierea starii actuale in domeniul lucrarii:**

Recunoasterea Optica a Caracterelor, reprezinta o translatare mecanica sau electronica a imaginilor cu scris de mana, tiparit sau printat care de obicei este scanat prin intermediul unei camere video, intr-un text editabil. Acesta este bazat pe recunoasterea optica a caracterelor (folosind scanere si algoritmi in limbaje de programare precum C++, Python, LabView, etc).

La inceput acestea erau bazate pe invatarea manuala a caracterelor (exact cum am dezvoltat in acest proiect, prin exemple ale fiecarui caracter) pentru a indentifica un font specific. Primele concepte au aparut in anii 1970, iar pe parcurs pana in ziua de astazi s-a ajuns la programe “dezvoltate” care au un grad mare de acuratete, prin identificare aproximativa a tuturor literelor scrise de tipar sau de mana.

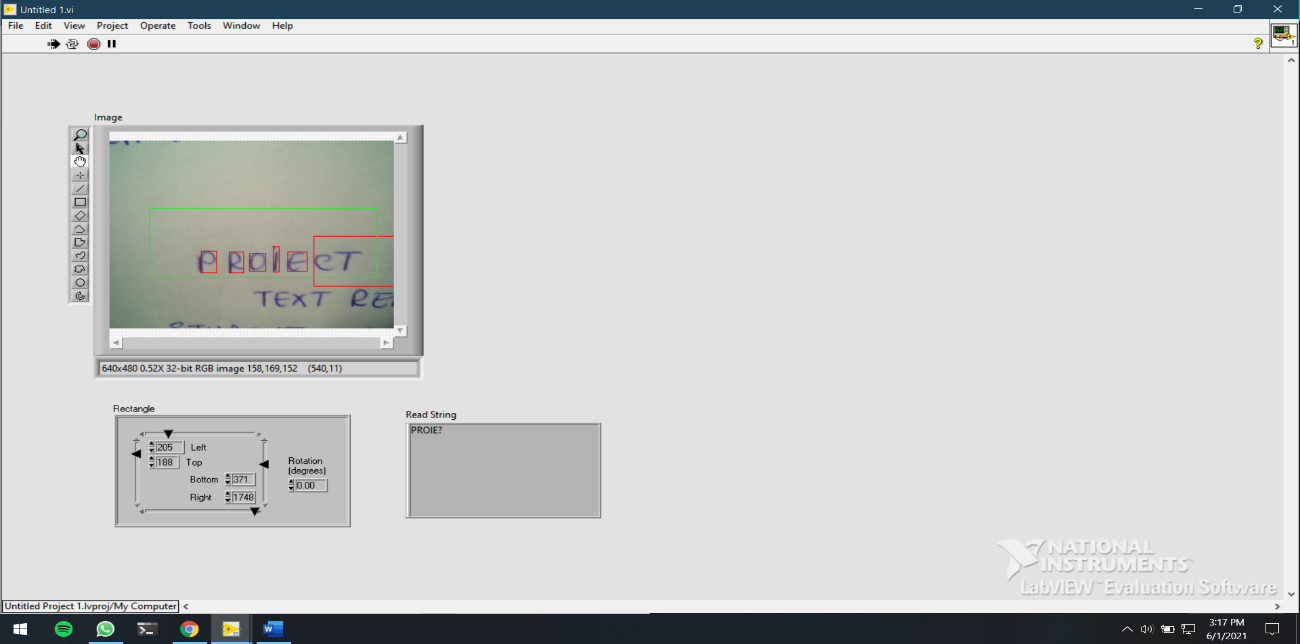
(\*) Mentiune: proiectul include litere scrise de mana, ci nu litere printate caligrafic. La finalul documentului voi atasa cateva poze si cu text printat pentru a verifica functionalitatea proiectului, dar mai ales acuratetea sa.

**Principalele concepte si tehnologii aflate la baza dezvoltarii proiectului:**

La baza proiectului se afla simularea unui cititor de text in programul LabView (o prescurtare de la “Laboratory Virtual Instrumentation Engineering Workbench”), fiind o platforma si un mediu de dezvoltare si proiectarea a unui limbaj de programare vizual.

In realizarea proiectului m-am folosit de versiunea 2020 Standard Edition, in care au fost incluse toate pachetele necesare dezvoltarii si prelucrarii proiectului, mai putin doua, si anume: “**Vision and Motion**” (pentru utilizarea celor 3 utilitare in prelucrarea imaginii webcam: “**Vision** **Utilities**”, “**Image** **Processing**”, “**Machine** **Vision**”), si “**Vision Development Module**” (pentru utilizarea mai multor blockuri de tip “instructiuni”, folosite pentru generarea unei interfete grafice). (1)

1. *Block “Image Display”*



**Contributii personale:**

1. **Deschiderea Camerei WebCam:** Pentru inceput trebuie creata o sesiune de tip “camera”. Pentru a face legatura intre camera web si spatiul de lucru in LabView, se conecteaza camera prin USB, unde trebuie specificat doi parametrii, si anume: **numele camerei** si **modul de control al camerei**. Toate sesiunile camerelor in **NI-IMAQdx** au ca referinta un nume specific (cam0 pentru 1 singura camera conectata prin USB, cam1 pentru a doua camera prin USB, cam3 a treia camera USB, etc). In proiect fiind folosita doar o singura camera, voi lucra doar cu denumirea de cam0. Din punct de vedere al **modului de control**, acesta are doua optiuni: **controller** (care controleaza camera si primeste date video) si **listener** (care primeste doar date video);
2. Se aloca un buffer unic de imagine “**Image Name**” (cam0) care initial este gol, unde “**Image Type**” specifica tipul de imagine care este creat (in cazul meu, nu am folosit un tip de culoare specific precum: Graysacale (U8), sau RGB(U32) ci le-am lasat default pentru ca **IMAQdx** sa ajusteze automat valorile in functie de camera mea). “**New Image**” contine un punct de referinta catre “**Image Name**” (care initial este o memorie goala). Cand se face legatura prin fir intre “New Image” si “Image In” a unei achizitii de imagine VI, se aloca cantitatea corecta de memorie. ~~In urma procesarii imaginii, am folosit un “~~**~~Border Size~~**~~”, unde acesta reprezinta latime in pixeli printr-un chenar in jurul imaginii afisate.~~ Am folosit o structura de programare grafica similara instructiunii de ciclare “while”. In fereastra “**Diagram Block**”, am utilizat o bucla “while” pentru a repeta anumite portiuni de cod si a le executa intr-o anumite ordine folosind **tuneluri** (date care intra si ies din **subdiagrama**).
3. **Bucla WHILE** – executa codurile din interiorul ei pana la indeplinirea unei anumite conditii. Bucla prezinta doua terminale, dintre care doar una este folosita:

**Terminal de conditionare** (partea dreapta a buclei) care primeste o anumita valoare logica (**true** sau **false**). Valoarea de conditionare se stabileste din meniul buclei, iar ce-a folosita de mine este optiunea de “**Stop if True**” (pentru a putea reda neintrerupt imaginea vizuala din camera web, iar aceasta bucla se executa pana la apasarea butonului de **STOP.**

1. **Citirea unui fisier OCR:** Pentru recunoasterea literelor dintr-un text predefinit, trebuie creat un asa numit “train font”, care are ca rol vizualizarea textului prin intermediul camerei web, iar mai apoi fiecare litera (index) trebuie tradusa/recunoscuta. (1)

**Vision Assistant:** Este un instrument intermediar care testeaza procesarea imaginilor dintr-un proiect (aplicatie). In cazul meu, am folosit optiunea de “**Processing Function Windows/Setup Window**” reprezentand un algoritm de procesare a unei imagini, fiind puse la dispozitie diverse instrumente de prelucrare, dintre care am ales functia de **“Processing Function: Indentification” -> OCR/OCV** (functie care citeste caracterele dintr-o imagine, pe care le compara cu cele predefinite, iar mai apoi afiseaza intr-un block textul vazut prin imaginea video / webcam).

Sunt legate fire intre blocuri pentru a putea realiza interactiunea camerei web cu functia din LabView (Vision Assistant) precum: Image In – Image Out, Error In – Error Out, etc

1. **Crearea** **unei** **sesiuni** **de** **citire OCR**:

*Vision and Motion -> Machine Vision -> OCR*

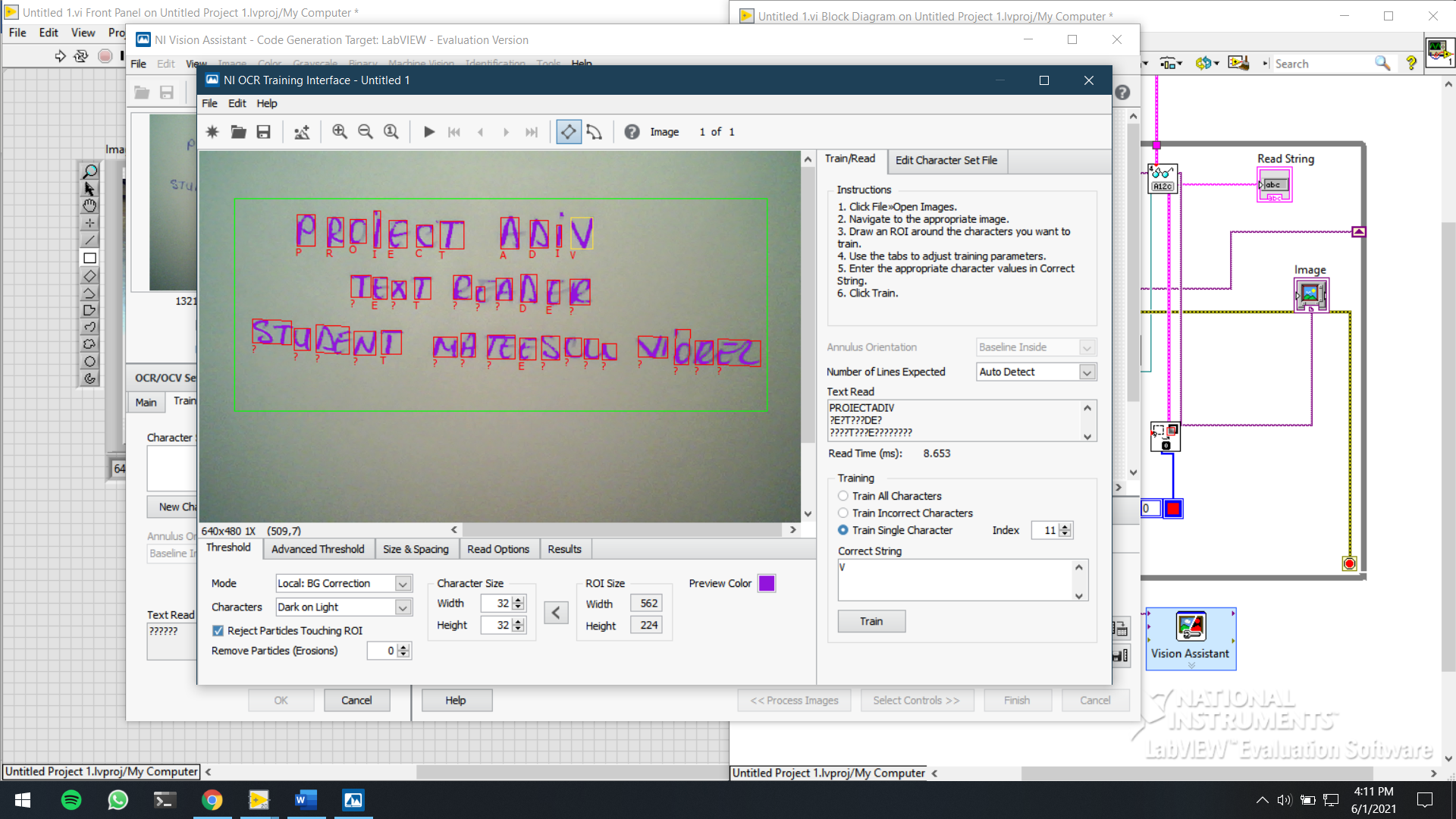
**IMAQ OCR Create Session**, creeaza o sesiune OCR si returneaza o valoare refnum (nu am inteles exact la ce se refera) asociat sesiunii curente. (2)

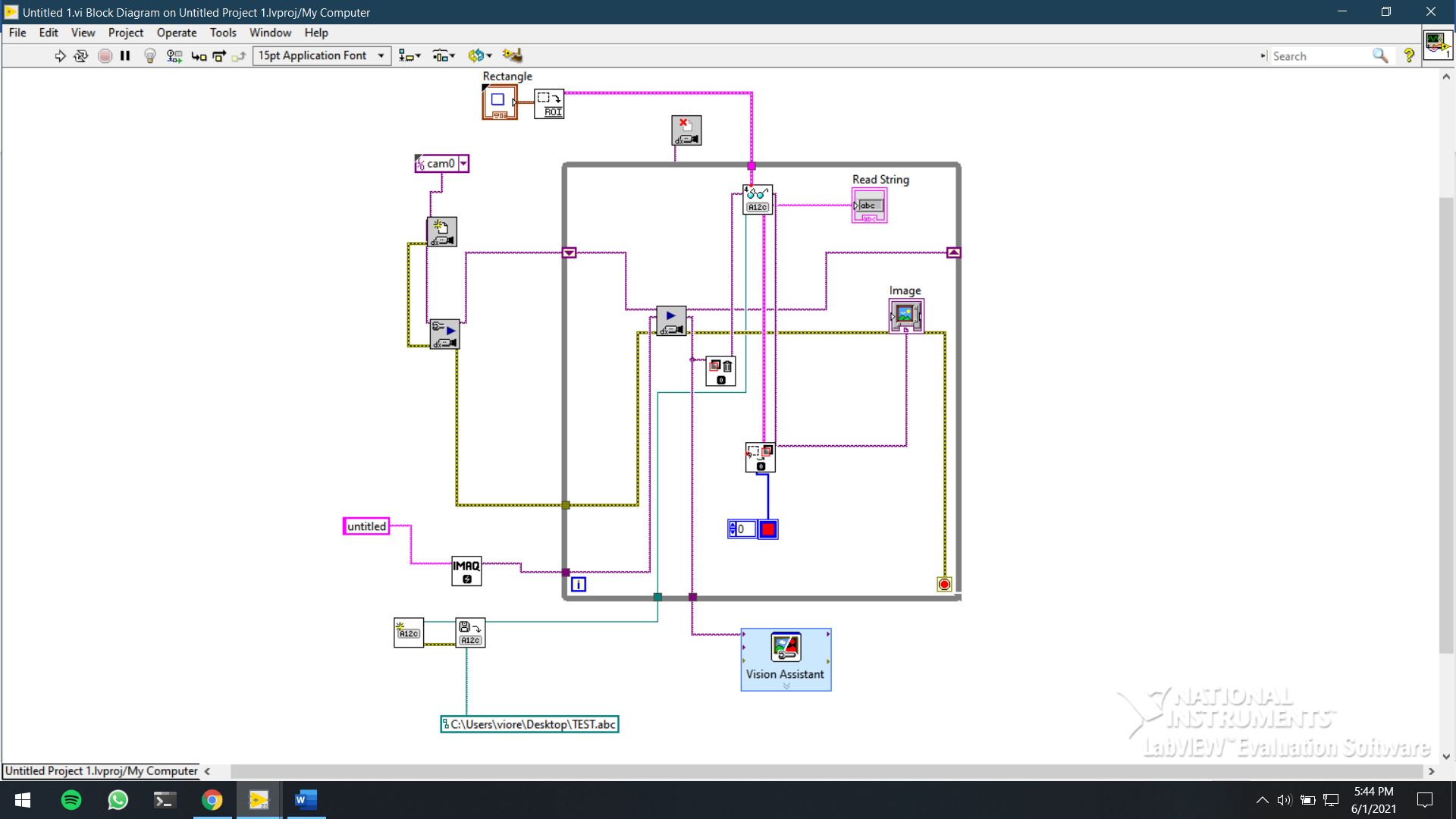
**IMAQ OCR Read Character Set**, folosit impreuna cu (2) pentru a traduce caracterele din sesiunea fisierului specificat ( C:\\Users\viore\Desktop\TEST.abc). Functioneaza pe principiul de legatura intre sursa si destinatie, unde sursa este o imagine simpla de la camera in format OCR si destinatia fiind un Read Text.

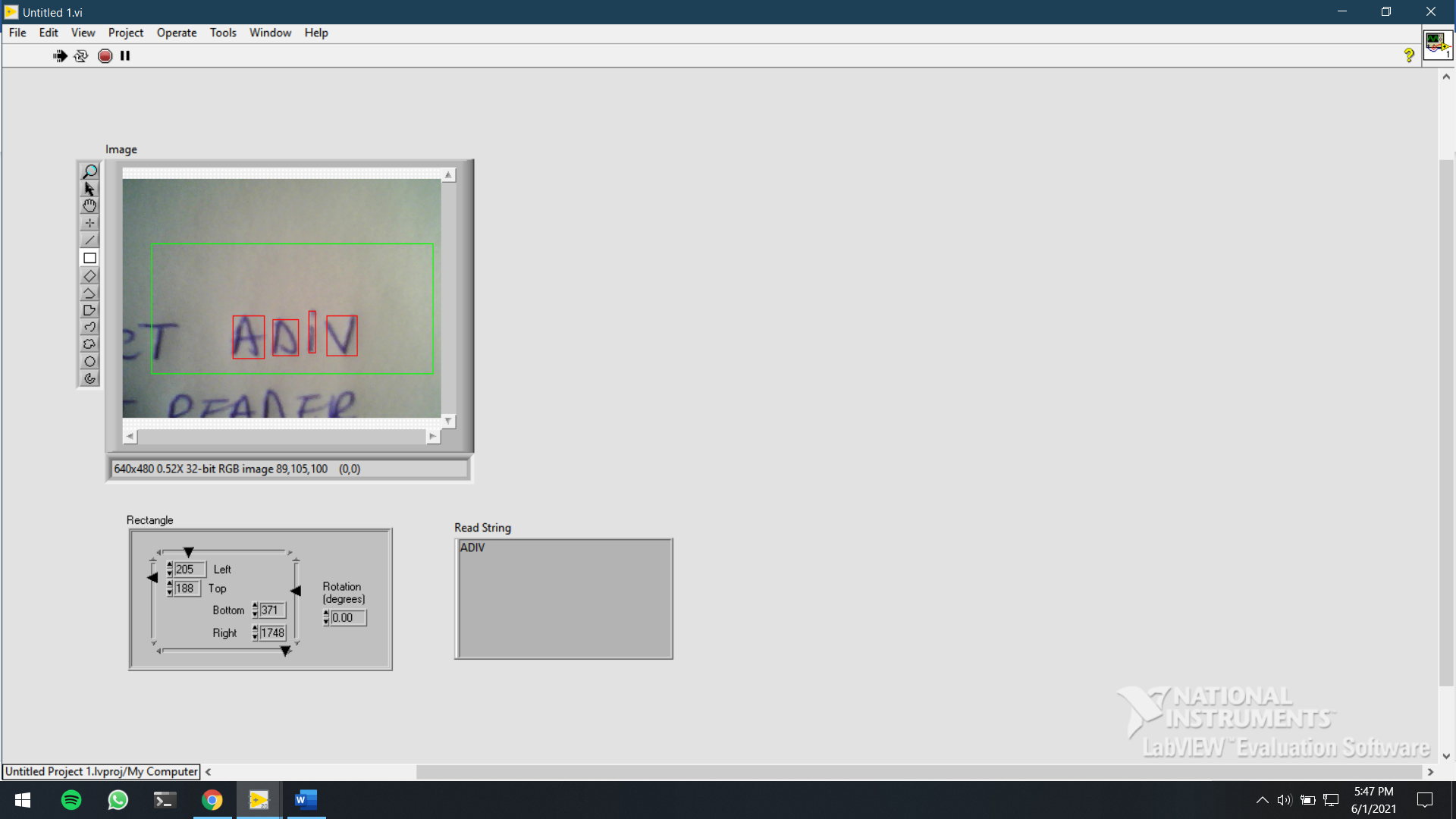
**IMAQ OCR Read Text**, citeste textul dintr-o imagine predefinita, unde algoritmul de procesare incerca sa identifice toate obiectele din imaginea video bazata pe “valorile” date/setate de mine. Incercand sa compare fiecare lucru vazut de camera web cu imaginea salvata. (1)

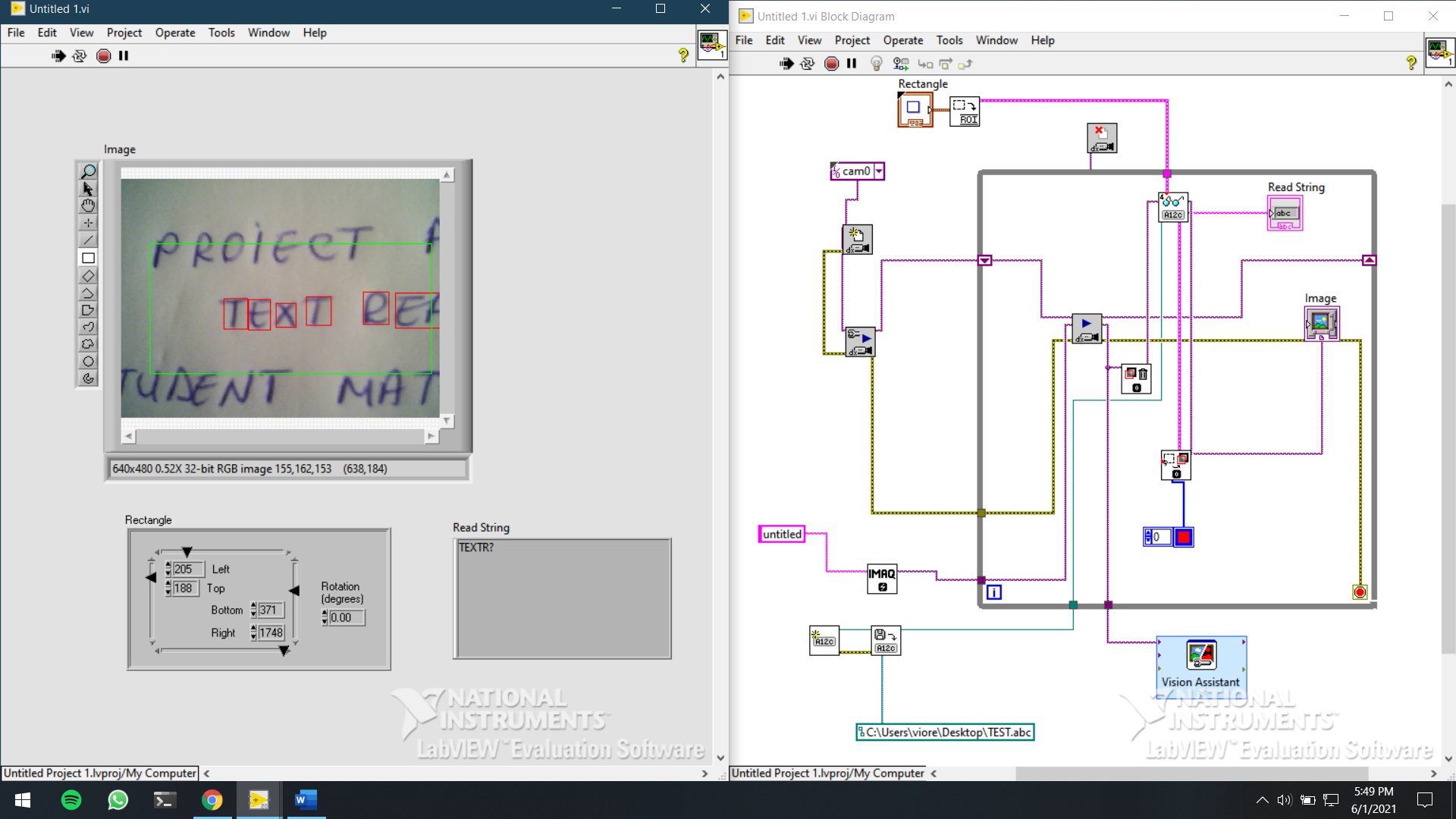
**IMAQ Overlay ROI VI** – suprapune o regiune (ROI) pe o imagine. Acesta se afla in **bucla while** si primeste doua intrari: **Colors**(**Green**) care este o serie de culori asociate cu fiecare dintre ROI-urile descrise de descriptorul ROI. “**Image** **Out**” care este o referinta de la o imagine destinatie. Daca “**Image** **DST**” este conectat, atunci **Image** **Dst** **Out** este acelasi ca si **Image** **Dst**. Altfel, **Image** **Dst** **Out** se rezuma la imaginea redata de **Image** **Src**. (2)

1. *Vision Assistant*









**Concluzii:**

Acest proiect nu incearca simularea unui adevarat cititor de litere. Din cate am cautat si observat pe internet, este foarte mult de munca in realizarea unui asemnea proiect, exact cum a-ti spus si dvs.  
Am incercat totusi s-a respect proiectul propus, si am creat un “Text Reader” bazat pe o imaginea care contine un text din litere (OCR), apoi fiecare litera tradusa manual (fiecare litera/index), dupa cu ajutorul unei functii din LabView sa incerce recunoasterea acelui text.

Timpul de lucru: 6-8 ore.

**Bibliografie:**

1. “NI Vision – NI Vision Assistant Tutorial” (<https://www.ni.com/pdf/manuals/372228m.pdf>)
2. “OCR – Recunoasterea optica a caracterelor” (<https://ro.wikipedia.org/wiki/Recunoa%C8%99terea_optic%C4%83_a_caracterelor>)
3. <https://curs.upb.ro/mod/resource/view.php?id=204208>
4. “documentation.HELP!” (<https://documentation.help/NI-IMAQdx/register_level_programming.html>)
5. Tutorial LabView – CAP.3.FUNCTII IN LABVIEW (<https://eprofu.ro/docs/labview/cap3.pdf>)