AI & SUSTAINABILITY
IN VET EDUCATION

AI-VET Mooc Project Submission Template

Funded by
the European Union

This project has been funded with support from the European Commission.
This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

1. Project Title:

"Visual translator of sign language"

2. Project Category:

- AI for Sustainable Development
- Connected Systems and Smart Data (AI + IoT)
- AI for Industry 4.0
- **Social Impact and Inclusive AI**
- Computer Vision and Image Recognition
- Predictive Analytics and Decision Support
- Intelligent Automation and Control Systems
- Smart Environments and Monitoring
- Other:_____

3. Project Description:

This system, based on machine learning technologies, enables automatic recognition and translation of sign language used by deaf and mute individuals through the analysis of video signals from a camera. It does not require the use of specialized equipment, such as gloves or additional sensors, as is the case with currently developed sign language translation technologies, making it practical, accessible, and easy to implement.

4. Keywords:

translator, sign (gesture), visual, camera, deaf-mute

5. Target Audience:

- **Both Educators and Students**

6. Project Objectives:

The goal of this project is to create a modern technological solution to improve communication between deaf-mute individuals and others who do not know sign language, thereby enhancing their social life. Through this project, the process of developing and implementing a visual sign language translator is learned, with a focus

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

on the technical and theoretical aspects that contributed to its realization. The development of this project also involves learning the implementation of machine learning methods.

The aim of this project is to acquire knowledge of working with Raspberry Pi, programming in Python, and integrating libraries.

Through this exercise, the following is learned:

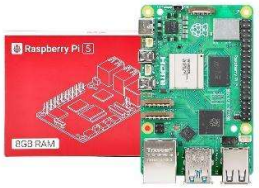
1. Installing an operating system for Raspberry Pi
2. Hardware connection of components
3. Understanding the integration of hardware and software (Python, libraries)
4. Demonstrating hardware-software interaction using Python

7. Project Timeline:

- Start Date: 23.3.2026
- End Date: 1.4.2026

8. Tools and Technologies Used:

1. During the development of this model for visual sign language translation, the following components from the AI-VET kit were used:

	component	picture	function
1.	Raspberry Pi 5 – 8GB		It is used as the central control unit (“brain”) of the system. After receiving the results from the camera, it processes them and transmits them to the display for presentation.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

2.	Raspberry camera module 2		It represents the camera module of the system. It captures the gesture displayed in front of it and, through the embedded code, recognizes it and sends the result for further processing.
3.	FPC кабел за камера 200mm		Used to connect the camera module to the main board, enabling the transmission of data and signals between them.
4.	micro HDMI to HDMI кабел 1m		Used to connect the system to a display, transmitting both video and audio signals.
5.	27W USB Напојување од тип C за Raspberry Pi 5		The power cable is used to supply electrical power to the system, enabling it to operate.
6.	Micro SDHC картица 64GB		They are used for serial connection and power supply of all the listed components, enabling their mutual communication and synchronized operation.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

2. AI-VET – Assembly Manual with Step-by-Step Instructions

3. Python code and libraries

4. Collecting photographic materials as a dataset for machine learning – it was necessary to provide a sufficient number of images for each class. The collected images are uploaded to the platform and manually labeled, precisely indicating to the model what each individual gesture represents.

9. Step-by-Step Implementation Plan:

The first phase of the development process for a functional prototype of the visual sign language translator involved research, analyzing existing technological solutions and methods in the fields of machine learning and computer vision, with the aim of setting clear goals and selecting appropriate tools and approaches.

The second phase focused on detailed planning of the system's schematics and architecture, precisely defining its components and their interconnections, followed by the development of code and implementation of algorithms for image processing and gesture recognition.

The third phase involved collecting images of gestures to train the machine learning model, using the Edge Impulse platform with object recognition functionality, through which the system was trained to classify gestures.

The fourth and final phase was assembling all components into a single system, followed by testing and practical verification of functionality to ensure accuracy and stability in operation.

Step-by-step process:

1. **Plan/idea:** Developing a visual sign language translator
2. **Research technological solutions and select appropriate components**

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

3. **Prepare hardware components:** gather all necessary materials – Raspberry Pi 5, power cables, camera, and then proceed with connections
4. **Assemble Raspberry Pi 5 and its enclosure with a temperature-controlled fan**
5. **Install Raspbian OS on the SD card, configure network access via VNC or SSH, and update system packages**
6. **Collect photographic materials as a dataset for machine learning**

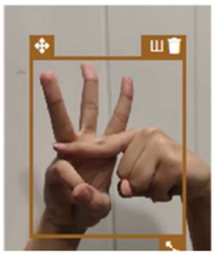


Figure 1: Letter Sh (Ш)

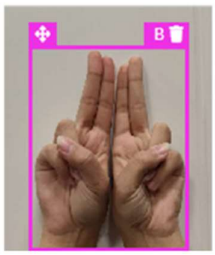


Figure 2: Letter V (В)

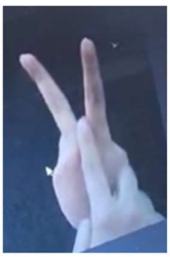


Figure 3: Letter Dz (Ц)

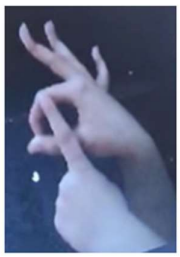


Figure 4: Letter F (Ф)

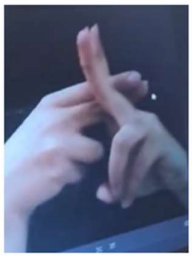
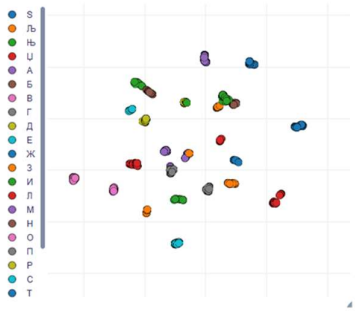


Figure 5: Letter Ch (Ч)

After collecting and labeling the data, the most important phase follows – training the machine learning model. During training, a coordinate (spatial) system is formed in which each class, that is, each gesture, is represented by a distinct color. The more the samples of the same class are clustered together and clearly separated from the others, the greater the model's ability to correctly recognize gestures and avoid confusing them with one another.





After successfully completing the training, the system allows testing of the model and displaying the accuracy percentage. This metric represents the model's performance, that is, how many times out of the total number of tests (for example, 150 or another

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

specified number) the machine produced a correct result. In this way, a clear picture of the efficiency and reliability of the developed system is obtained.

7. Installing the commands required for implementation

sudo apt update

curl -sL https://deb.nodesource.com/setup_20.x | sudo bash -

sudo apt install -y gcc g++ make build-essential nodejs sox gstreamer1.0-tools gstreamer1.0-plugins-good gstreamer1.0-plugins-base gstreamer1.0-plugins-base-apps

sudo npm install edge-impulse-linux -g --unsafe-perm

sudo npm install -g npm@10.8.1

sudo apt update

sudo apt install -y docker.io

sudo systemctl enable docker

sudo systemctl start docker

sudo docker run -it --rm --privileged --network=host -v /dev/:/dev/ --env UDEV=1 --device /dev:/dev --entrypoint /bin/bash ubuntu:20.04

apt-get update

apt-get install wget -y

wget https://deb.nodesource.com/setup_20.x

bash setup_20.x

apt install -y gcc g++ make build-essential nodejs sox gstreamer1.0-tools gstreamer1.0-plugins-good gstreamer1.0-plugins-base gstreamer1.0-plugins-base-apps vim v4l-utils usbutils udev

apt-get install npm –y

edge-impulse-linux

edge-impulse-linux-runner

8. Coding the program

9. Testing

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



10- Code

```python
#!/usr/bin/env python

import device_patches       # Device specific patches for Jetson Nano (needs to be before importing cv2)

try:
    import cv2
except ImportError:
    print('Missing OpenCV, install via `pip3 install "opencv-python>=4.5.1.48,<5"`')
    exit(1)
import os
import sys, getopt
import signal
import time
from edge_impulse_linux.image import ImageImpulseRunner

runner = None
# if you don't want to see a camera preview, set this to False
show_camera = True
if (sys.platform == 'linux' and not os.environ.get('DISPLAY')):
    show_camera = False

def now():
    return round(time.time() * 1000)

def get_webcams():
    port_ids = []
    for port in range(5):
        print("Looking for a camera in port %s:" %port)
        camera = cv2.VideoCapture(port)
        if camera.isOpened():
            ret = camera.read()[0]
            if ret:
                backendName =camera.getBackendName()
                w = camera.get(3)
                h = camera.get(4)
                print("Camera %s (%s x %s) found in port %s " %(backendName,h,w, port))
                port_ids.append(port)
            camera.release()
    return port_ids

```

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
def sigint_handler(sig, frame):
    print('Interrupted')
    if (runner):
        runner.stop()
    sys.exit(0)

signal.signal(signal.SIGINT, sigint_handler)

def help():
    print('python classify.py <path_to_model.eim> <Camera port ID, only required when more than 1 camera is present>')

def main(argv):
    try:
        opts, args = getopt.getopt(argv, "h", ["--help"])
    except getopt.GetoptError:
        help()
        sys.exit(2)

    for opt, arg in opts:
        if opt in ('-h', '--help'):
            help()
            sys.exit()

    if len(args) == 0:
        help()
        sys.exit(2)

    model = args[0]

    dir_path = os.path.dirname(os.path.realpath(__file__))
    modelfile = os.path.join(dir_path, model)

    print('MODEL: ' + modelfile)

    with ImageImpulseRunner(modelfile) as runner:
        try:
            model_info = runner.init()
            # model_info = runner.init(debug=True, timeout=60) # to get debug print out and set longer timeout
            print('Loaded runner for "' + model_info['project']['owner'] + ' / ' + model_info['project']['name'] + '"')
            labels = model_info['model_parameters']['labels']
            if len(args)>= 2:
                videoCaptureDeviceId = int(args[1])
            else:
                port_ids = get_webcams()
                if len(port_ids) == 0:
                    raise Exception('Cannot find any webcams')
                if len(args)<= 1 and len(port_ids)> 1:
                    raise Exception("Multiple cameras found. Add the camera port ID as a second argument to use to this script")
                videoCaptureDeviceId = int(port_ids[0])

            camera = cv2.VideoCapture(videoCaptureDeviceId)
            ret = camera.read()[0]
            if ret:
                backendName = camera.getBackendName()
                w = camera.get(3)
                h = camera.get(4)
                print("Camera %s (%s x %s) in port %s selected." %(backendName,h,w, videoCaptureDeviceId))
                camera.release()
            else:
                raise Exception("Couldn't initialize selected camera.")

            next_frame = 0 # limit to ~10 fps here

            for res, img in runner.classifier(videoCaptureDeviceId):
                if (next_frame > now()):
                    time.sleep((next_frame - now()) / 1000)

                # print('classification runner response', res)

```

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
            if "classification" in res["result"].keys():
                print('Result (%d ms.) ' % (res['timing']['dsp'] + res['timing']['classification']), end='')
                for label in labels:
                    score = res['result']['classification'][label]
                    print('%s: %.2f\t' % (label, score), end='')
                print('', flush=True)
            elif "bounding_boxes" in res["result"].keys():
                print('Found %d bounding boxes (%d ms.)' % (len(res["result"]["bounding_boxes"]), res['timing']['dsp'] + res['timing']['classification']))
                for bb in res["result"]["bounding_boxes"]:
                    print('\t%s (%.2f): x=%d y=%d w=%d h=%d' % (bb['label'], bb['value'], bb['x'], bb['y'], bb['width'], bb['height']))
                    img = cv2.rectangle(img, (bb['x'], bb['y']), (bb['x'] + bb['width'], bb['y'] + bb['height']), (255, 0, 0), 1)
            elif "freeform" in res['result'].keys():
                print('Result (%d ms.)' % (res['timing']['dsp'] + res['timing']['classification']))
                for i in range(0, len(res['result']['freeform'])):
                    print(f'    Freeform output {i}:', ", ".join(f"{x:.4f}" for x in res['result']['freeform'][i]))

            if (show_camera):
                cv2.imshow('edgeimpulse', cv2.cvtColor(img, cv2.COLOR_RGB2BGR))
                if cv2.waitKey(1) == ord('q'):
                    break

            next_frame = now() + 100
        finally:
            if (runner):
                runner.stop()

if __name__ == "__main__":
    main(sys.argv[1:])

```

11. Evaluation Criteria:

- **Student Feedback:** Feedback will be collected through surveys to evaluate how well students understood sign language recognition, AI processing, and the translation process. It will also assess how useful and engaging they found the project.
- **Project Success Metrics:** Success will be measured by the system's ability to correctly recognize sign language gestures and display the corresponding visual output or text. Accuracy, speed of recognition, and reliability during testing will be key indicators.
- **Skills Assessment:** Students will be assessed on their knowledge of AI programming, computer vision basics, and system integration. Emphasis will be placed on data handling, model accuracy, and implementation of a functional translation system.

12-Credits

Student Vasilisa Kurzakova - Worked independently from idea, research, assembly, installation, hardware solution, connectivity and software programming.

Professor Gordana Naumovska - Provided guidance and support throughout the project.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

13- Video presentation link (max. 5 minutes)

https://drive.google.com/file/d/1SQzJSp773lVl_t9fMmEftGkGcFyOxRt7/view?usp=drive_link

14- Declarations

I declare that I have read and agree with the following statements:

1- The project is original and developed by the listed team members.

2- All technologies and sources are properly cited.

3- The project was developed using components from the AI-VET kit.

4- We authorize the dissemination of the project for educational and promotional purposes (non-exclusive rights).

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.