# Information about the EMLI mini project and the individual presentation portfolio

*Version v2024-04-16v3, Kjeld Jensen kjen@sdu.dk*

## Introduction

The course description presents the exam in EMLI as an “individual presentation portfolio submitted in the end of the semester” which will be graded according to the 7-point grading scale. This document contains further information about the EMLI mini project and the individual presentation portfolio.

## Course flow

The course is taught as a laboratory course. The first 9 modules follow a scheme of introductions to the theory followed by laboratory exercises performed in class in the course teams. No reports are submitted during those modules.

The modules 10-12 are reserved for an embedded linux mini project. At the 10th module a project description for the project is provided and introduced, and the students will start working on the project. During the modules 10-12 the teacher will be available in class for questions and support

## Project teams

The students will conduct the project and prepare the team contributions to the individual presentation portfolio in the existing course teams.

## Individual presentation portfolio submission deadline

The individual presentation portfolio must be submitted via SDU digital exam by June 3'rd, 2024.

The content of the individual presentation portfolio is described later in this document, however it is important that all students must individually submit both the *team document*, the *individual document* and the *team video*.

The team document must be submitted as the main submission, the individual document and team video must be submitted as appendixes.

## Exam assessment

The assessment is described in the document: emli_2024_portfolio_assessment_guide.pdf

## Project background

SDU is leading a reserach project called *WildDrone - Research in Autonomous Drones for Nature Conservation.* The project focuses has two wildlife reserves, the Wadden Sea National Park in Denmark and the Ol Pejeta Conservancy in Kenya.

One of the ideas that has come out of this project, especially for our work in Kenya, is to develop a next generation wildlife camera. In this project you will build the embedded linux setup for such a wildlife camera.


You are welcome to read more about the project at this website,
however this is not directly relevant to the project: https://wilddrone.eu

## Project assignment

In the EMLI 2024 project you will build the embedded linux setup for a next generation *wildlife camera system*:

*A camera that takes pictures based on different triggers; that does not need connection to the internet but instead offloads the pictures to a drone that periodically fly across the wildlife park and circles for a short while at each camera; and performs AI based automatic annotation of the pictures.*

The project assignment is to...

*in the team:*

I. Design and develop a *wildlife camera system* that fullfills all or as many as possible of the functional requirements and nonfunctional requirements.

II. Prepare the team document, the team video and the team git repository according to the description of the individual presentation portfolio.

*individually:*

III. Prepare the individual document according to the description of the individual presentation portfolio.

IV. Submit the individual presentation portfolio describing the project in accordance with the listed requirements of the individual presentation portfolio.

## Extra components

The EMLI kit contains all components required to complete the project with the exception of a *Raspberry Pi Camera Module 3* which will be handed out at module 10.

Please be careful with the camera and please also take good care of the box and bag from the camera.

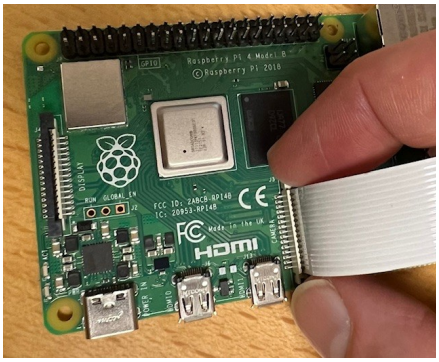
Please also be very careful when installing and removing the camera. The connector is very delicate and breaks easily. If you have any concerns about this or get stuck in the process, please approach one of the teachers.

When the camera has been installed please be careful when transporting the Raspberry Pi. Alwas use a plastic bag with nothing else inside.

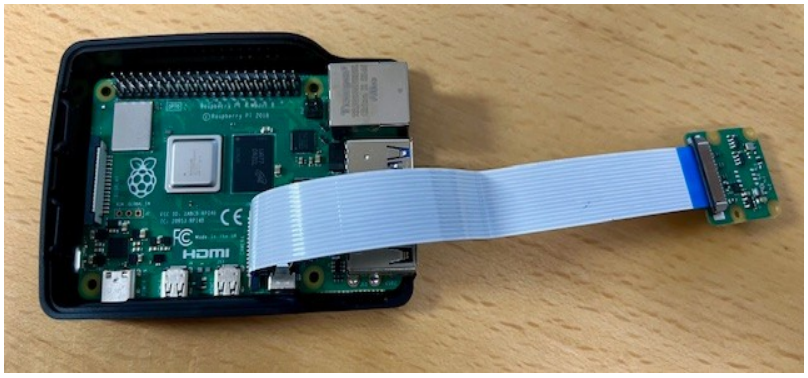
Installation instructions (please follow carefully step by step)

1. Remove the MicroSD card from the Rasberry Pi
2. Remove the Raspberry Pi from the enclosure. Start by lifting up the side where the double row of GPIO pins is. Please remember to only touch the PCB edges, the corners and the large metal ethernet- and USB connectors.
3. Gently lift up the grey part of the camera connector using a finger nail at both ends, one end at a time. The connector is very delicate and breaks easily.
4. Insert the ribbon cable into the slot. The blue side must face the four large USB-A connectors.
5. Gently press down both ends of the grey part of the connector, one end at a time.
6. Insert the Raspberry Pi into the enclosure
7. Insert the MicroSD card into the Raspberry Pi
8. Put on the enclosure lid while gently routing the ribbon cable out above the USB connectors (see image). then fasten the camera with some tape.

Installation step 2



Installation step 5

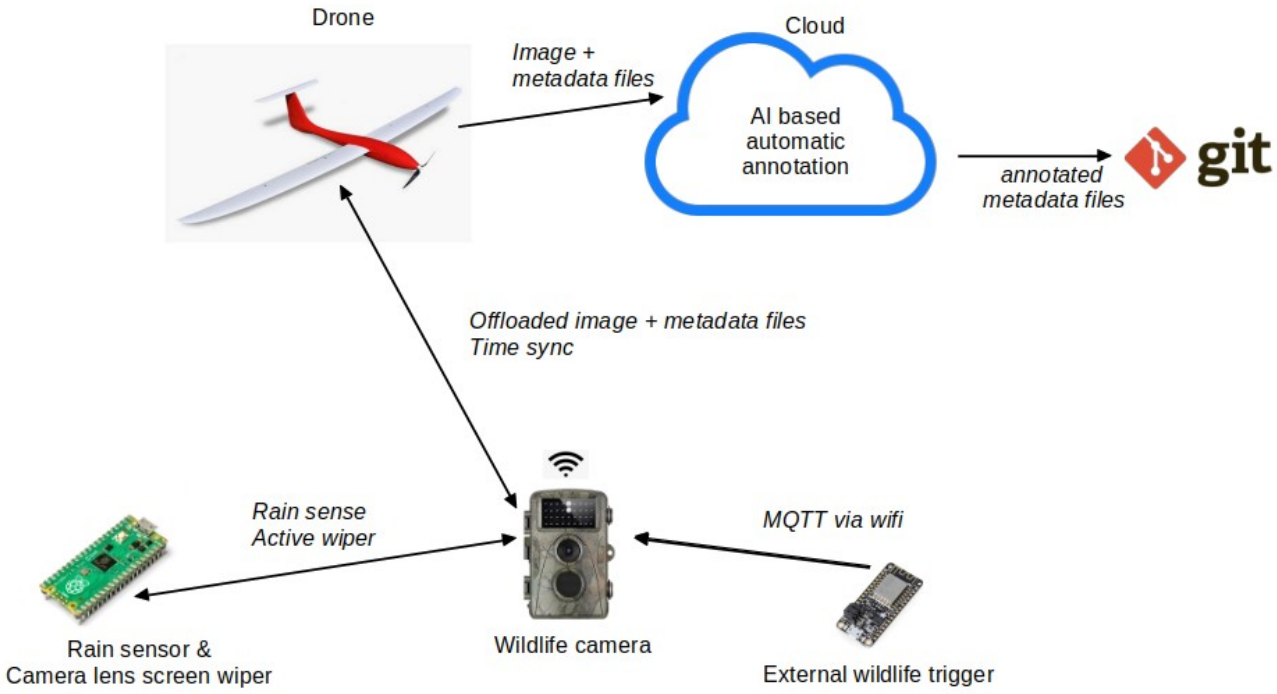


Installation step 7

Installation step 8

Installation step 8

## System description




## Wildlife camera description

The wildlife camera linux system is your Raspberry Pi with a Raspberry Pi Camera Module 3.

Please follow the installation instructions under Extra components. You can then test the camera using the command:

```
$ rpicam-still -t 0.01 -o test.jpg
```

To perform motion detection you may use the script `motion_detect.py` available on itslearning.

## Rain sensor & Camera lens screen wiper description

The camera is connected via USB to the Raspberry Pi which acts as rain sensor (simulated by press of the onboard BOOTSEL botton) and camera lens screen wiper (simulated by a servo performing a wiping movement).

Connect the servo to the following Raspberry Pico pins

Servo red wire to pin VBUS
Servo brown wire to pin GND
Servo orange wire to pin 15

At itslearning you will find an Arduino Sketch for the Raspberry Pico which you must use without edits.

Please make sure that you use the boards file Raspberry Pi Pico/2040 from Earlephilpower:

https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json

Please also make sure that you use the ArduinoJSON library from Bernoit Blanken.

If not, you may get errors when compiling the Arduino sketch.

The Arduino sketch will continuously output a JSON object of the format:

```
{'wiper_angle': 0, 'rain_detect': 0}
```

and it accepts a JSON object as input if formatted as:

```
{'wiper_angle': 180}
```

where the angle is between 0 and 180.

If you send an angle outside 0-180, you will receive:

```
{'serial': 'angle_error'}
```

If you send erroneous JSON data, you will receive:

```
{'serial': 'json_error'}
```

## External wildlife trigger descriptions

The external wildlife trigger is the ESP8266 connected to the wildlife camera WiFi access point. The trigger mechanism is a gound wire and a digital input wire that simulate an animal walking above a pressure plate causing a short circuit of the wires (you may use an external button).

Modify the Arduino sketch `esp8266_count_mqtt.ino` from module 7 (included in the project materials on itslearning) to send data immediately when a short circuit (button press) is detected.

## Drone description

The drone is simulated by you linux desktop environment on your laptop.

During drone flight your laptop should be disconnected from the internet and a "drone flight" script should use the laptop WiFi to search for nearby cameras by means of the WiFi SSID.

When a camera WiFi SSID is found, the laptop should offload any new photos of from the wildlife camera.

The drone flight is terminated by manually stopping the drone flight script and connecting the WiFi to internet.

## Cloud description

When the laptop is connected to internet we will for practical reasons consider the laptop to be the cloud i.e. there is no need to move the files from the drone to the cloud.

Instead you will do the AI based automatic annotation directly on your laptop (offline) and upload the updated metadata files to a directory in you git repository.

To achieve AI based annotation it is recommended to use https://ollama.com but it is ok to work with other AI based solutions such as YOLO etc.

Feel free to choose between the following commands depending on your laptop computing power. The result may not be impressive for the 7b model but no worry, the quality of the annotation text is irrelevant for the exericse. The focus is on implementing the functionality.

```
$ ollama run llava:7b "describe /home/emli/test.jpg"
$ ollama run llava:13b "describe /home/emli/test.jpg"
$ ollama run llava:34b "describe /home/emli/test.jpg"
```

## Functional requirements

The functional requirements for the wildlife camera are:

a) The wildlife camera must run Raspberry Pi OS

b) The wildlife camera must have a shell script `take_photo.sh` that when executed, takes a photo using the Raspberry Pi camera and saves it to disk. The filename must be formatted as "`143557_045.jpg`" for local summer time 14:35:57 and 045 ms stored in a folder named by the current date, formatted as "`2024-04-16`".

c) When `take_photo.sh` is executed, a JSON metadata file with the same name as the photo but with the extension "`json`" must be saved in the same directory as the photo (also known as a sidecar file). The JSON file must contain the keys in this example:

```
{
      "File Name": "143567_045.jpg",
      "Create Date": "2024-04-16 14:35:57.045+02:00",
      "Create Seconds Epoch": 1713270957.045,
      "Trigger": "Time",
      "Subject Distance": 0.5574136009 m,
      "Exposure Time": "1/33",
      "ISO": 200
}
```

The key Trigger defines the cause of the capture which may have one of the following values:
`Time/Motion/External`

The values for the keys "Subject Distance", "Exposure Time" and "ISO" are available via the photos EXIF data which can for instance be read with the linux command `exiftool`

d) The wildlife camera must save a photo every 5 minutes with the key Trigger set to `Time`

e) The wildlife camera must take a photo approximately each second. If a motion is detected between any two consequitive photos, the latest photo must be saved with the key Trigger set to `Motion`

f) If an animal is detected by the external wildlife trigger (ESP8266), this information must immediately be received via MQTT. Based on this a photo must be taken with the JSON key Trigger set to External

g) If rain is detected by the Raspberry Pico (the BOOTSEL button is pressed) this info must be sent via MQTT to another script that then via MQTT requests a wipe of the lens screen (in a sequence of 0-180-0 degrees)

h) Via the wildlife camera access point a simple local website must provide access to all saved images and associated JSON metadata.

i) The wildlife camera must maintain a log functionality capturing all relevant events. The log file must be accessible via the simple local website.

j) The wildlife camera must become fully operational after a power failure without requiring user interaction (graceful shutdown can be assumed)

The functional requirements for the drone are:

k) When the drone is in the vicinity of the wildlife camera (i.e. the laptop can see the raspberry pi accesspoint SSID) the time of the wildlife camera must be synchronized with the time of the drone.
l) Also when the drone is in the vicinity of the wildlife camera, the drone must copy as many photos and associated metadata files not previously copied from the wildlife camera as possible. For each file succesfully copied the photo is retained on the wildlife camera. The copy must be registered on the wildlife camera by adding two keys to the metadata JSON file:

```
"Drone Copy": {"Drone ID": "WILDDRONE-001", "Seconds Epoch": 17132712340.458},
```

m) While connected to the wildlife camera, the drone must continuously log the WiFi link quality and signal level from /proc/net/wireless together with the seconds epoch. This should be logged to a simple sql database for subsequent use to optimize the flight path.

The functional requirements for the cloud are:

n) all photos must be annotated using ollama or a similar offline AI engine. The annotation text must be added to the metadata JSON file such as:

```
"Annotation": {"Source": "Ollama:7b", "Test": "The picture contains a lion cub playing...."},
```

o) The metadata JSON files for a batch of offloaded photos with the added annotations must be committed to a folder within the team's git repository. Please notice that this is only the JSON files, not the photos themselves.

## Nonfunctional requirements

a) The embedded system must be designed with the main tenets of the Unix philosopy in mind

b) Where appliable shell scipts should be used rather than other programming or scripting languages

c) Where applicable the embedded system must be scalable towards handling a large number of wildlife cameras (not near each other through).

d) Both the wildlife camera and and the drone (simulated by the linux desktop environment on a laptop computer) must to the extent possible be secured against malicious cyber attacks from both internet and the local wifi.

## Individual presentation portfolio

The individual presentation portfolio contains:

1) **A report describing the project.** Report pages must be A4 format, single column and have 2 cm margins. Any pages beyond the stated maximum number of pages will not be read. Add images, sketches etc. as applicable. Page numbering must be present but no other headers or footers on the pages. Font must be Times New Roman, Liberation Serif or similar at 12pt. The report consists of two individual pdf documents described in the following:
   a) **a team document (11 pdf pages)** following the structure:
      - Front page: clearly stating project group number, group member names and SDU email addresses, and link to the git repository. No other report content at this page. This front page is included in the page count i.e. there are 10 pages left for the actual document.
      - Solution approach: Describe the high level solution approach, the overall architecture and design, including advantages and drawbacks. Use sketches and diagrams.
      - Solution description: Describe solutions to key tasks and concepts in the project.
      - Tests and results: Describe how the embedded system was tested and present the results of those tests.
      - Conclusion: Short conclusion summarizing the achieved solution and results.
   b) **an individual document (4 pdf pages)** following this structure:
      - Front page: clearly stating student name, project group number and SDU email address. No other report content at this page. This front page is included in the page count i.e. there are 3 pages left for the actual document.
      - Discussion: Compare and discuss the achieved solution, results, demonstration to the functional requirements. Discuss to what extent the solutions achieve the nonfunctional requirements, and if not, why not.
      - Conclusion: Summarize and outline how the achieved solution and results can be improved in future work.
2) **A team video** presenting the outcome of the project:
   a) The video will be a combination of illustrations, recorded video of the physical system and any interactions, relevant screen recordings etc.
   b) The video length must be maximum 90 seconds.
   c) The video must be formatted as MP4 encoded using H.264/AAC with a maximum file size of 25 Mbytes. If not, the video may not be included in the assessment. Use of `ffmpeg` to convert to this format is recommended, the following parameters appears to work well for different sizes and formats:
      - `ffmpeg -i input.mov -c:v libx264 -vf scale=-1:1080 -crf 23 -maxrate 2M -bufsize 4M output.mp4`
3) **A link to a git repository** containing data according to the functional requirements as well as all relevant scripts, configuration files etc. created by the team for solving the project.

a) The git repository must provide a quick and intuitive access to the relevant material.