Introduction

An increasing demand of data-driven ecosystem and agricultural production system monitoring has been paralleled by a rapid advancement of hardware and software. Recent devices are expensive, non-intelligent, and consequently have prevented mass deployment required for large scale eco-system monitoring. Originally out of the Sustainable Agricultural Systems & Engineering Laboratory of Westlake University, and now at our spin-off company ecoNect, we have been developing an AI-based ecosystem monitoring system that is designed to overcome all shortcomings of current methods.

Our first product, the ecoEye, comprises of a low-power machine vision module, specifically engineered modules including an energy harvesting power management system for independent operation during day and night, and a purpose-driven, rugged, and waterproof case. The embedded nature of the ecoEye allows not only to automatically monitor ecosystems, but also to trigger a response to a detection event. Our detection scope and identification accuracy are much higher than those of conventional surveillance cameras based on passive infrared sensor (PIR) detection.

The ecoEye is currently field-tested in a wide range of applications. For example, the conservation NGO TNC is monitoring biodiversity in protected areas and in their ecological restoration sites. Agricultural researchers use the devices to monitor pollinators and agricultural pests to minimize pesticide use and promote sustainable agriculture. Our educational partners use them to raise environmental awareness among their students and train them for research activities through project-based-learning projects. The device is also used internationally and in different climatic conditions in China, Indonesia, and Brazil as an agricultural assistant to optimize good agricultural practices by monitoring pollinators and crop pests' activities.

In the future, the ecoEye and its subsequent multi-sensor versions to also monitor sound, climatic and soil parameters may become a useful tool to implement environmental alternatives to GDP such as the Gross Ecological Product (GEP; REF).

For these and similar applications, we are currently developing the networked Internet of Things version of the device, which is fully powered by solar energy. We are also advancing the remote data access capabilities and intelligent analytics capabilities.

EcoEye User Manual

Hardware

To access the battery compartment and interior of the ecoEye (Fig.1), the latch mechanism must be opened by pulling the latch up. The camera has a battery compartment for 3 in parallel connected rechargeable Lithium-ion 18650 cells with 3.7V nominal voltage. Batteries must have the same capacity (mAh or mWh rating) and charging state (voltage across poles) when inserted together. The camera can also function with just one or two batteries although the total capacity will be reduced. Theoretically, a combined 10 050 mAh battery resulting from inserting three 3350 mAh batteries should be able to run the camera in continuous operation for 24 hours. It should be noted that using the WLAN shield, infrared LEDs, or any other module will increase power consumption and decrease autonomy. The three individual spaces allow each battery to be held firmly inside the case. For a stronger mounting, the horizontal bar can be screwed across the centre of the batteries.

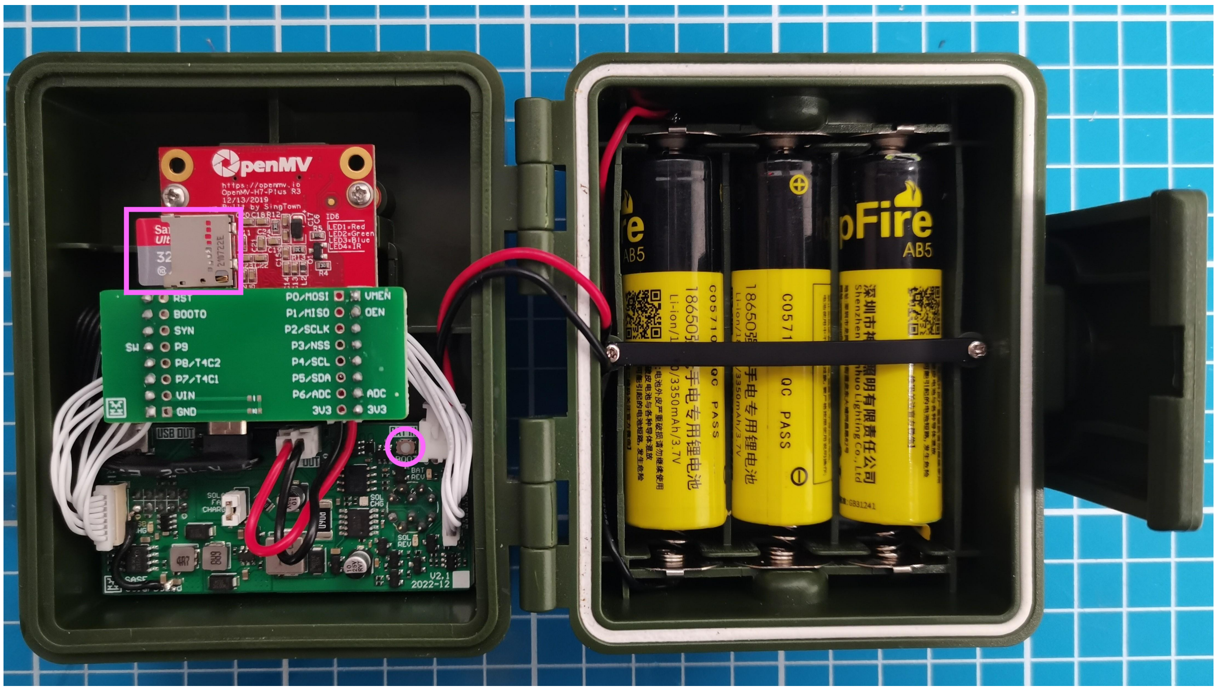


Figure 1. Inside view of the ecoEye camera

When installing the batteries for the first time, in case the camera does not turn ON (power–on of the main system is indicated by the illumination of the LED on the front side) when pressing the power button on the front side of the camera, the tiny push button labelled 'BOOT' (circled in pink; Fig. 1) on the (green) power management board (named 'Solarboard') must be pressed once to reset the battery protection circuit. Verify that the connections inside the case correspond to the picture above. To enable picture saving and higher computation speed, a micro–SD card should be installed in the dedicated holder (pink rectangle; Fig. 1) on the (red) openMV main board. SD cards with a minimum speed of Class 10 or UHS–1, and a minimum storage capacity of 32GB are recommended for best efficiency.

Waterproofness is assured by compressing the rubber joints along the openings on the case. This includes the door, which is tightly closed by the latch mechanism (green arrow Fig.2; and can be locked shut with a UNC1/4–20 screw), the two connectors closed off by the lid caps (blue arrows Fig. 2), the rubber boot covering the power push button (blue button Fig. 2) and the rubber ring on the lens (orange arrow Fig 2). This ring should be positioned on the end of the thread so that it will be pressed against the case surface when screwing in a new lens and focusing. Pay attention that the ring is properly positioned after turning the lens in and out of its mount during the image focusing process.

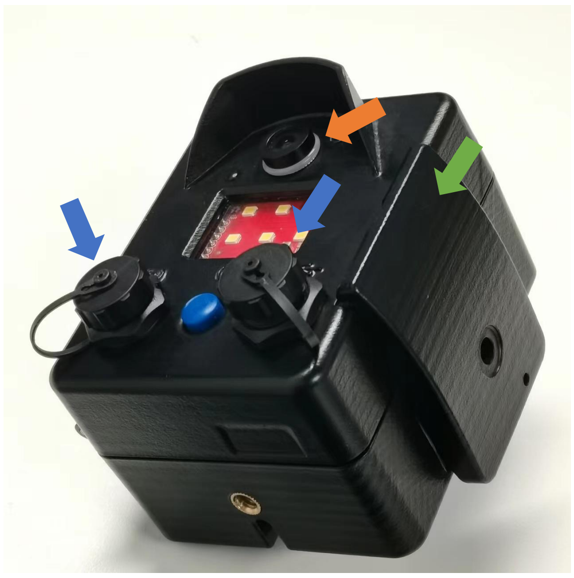


Figure 2. Waterproofing mechanisms.

There are several different types of lenses, marked with their focal length and sometimes also the image resolution and magnification ratio. The higher the focal length (defined by the "mm" unit), the narrower the angle of view. Some lenses have a reddish glass filter at their back (Fig. 3 right), which is an infrared cut-off filter that blocks infrared light. This is required for capturing accurate colour (RGB565) images but makes the lenses not usable for night monitoring (grayscale). Other lenses do not have the reddish glass filter at their back, meaning that infrared light can pass and be detected by the image sensor (Fig. 3 left). These lenses are useful for night monitoring with grayscale images where the target is illuminated with infrared (invisible to our eyes) light using LEDs. The filter can be removed destructively to obtain an "infrared" lens.

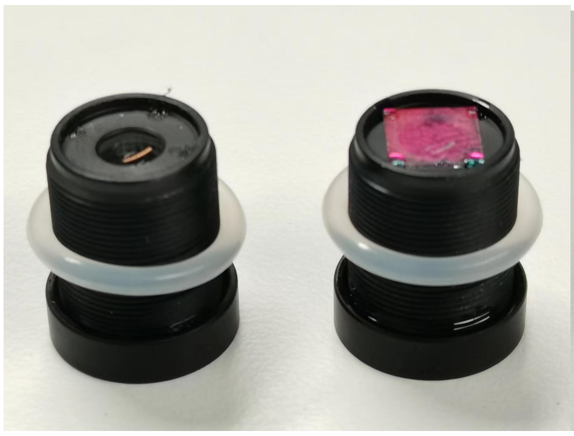


Figure 3. Lenses without and with infrared filter.

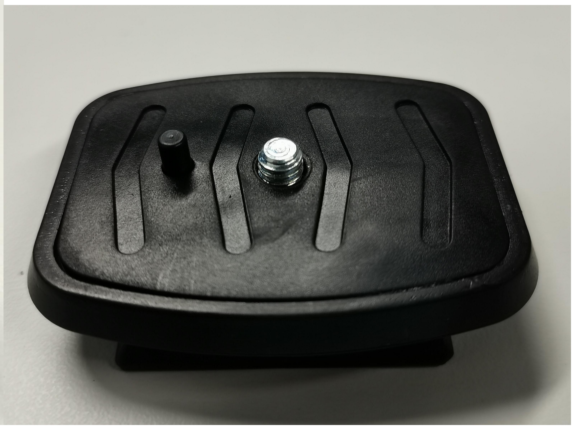


Figure 4. Tripod holder

To fix the camera on a standard tripod or to attach accessories to it, the outer faces of the case contain threaded inserts UNC1/4-20. The slot in front of the threaded insert on the top and bottom faces can be used to insert the little sliding plastic stick that can be found on certain tripod base plates (Fig. 4).

There are two connectors and a blue power push-button on the front face of the camera (Fig. 5). The left multi-pin connector can be used to add a solar panel, different types of sensors, and relays to control self-powered devices. The right USB type-C connector can be used to charge the batteries (the recommended way of charging) and to configure the main script through the openMV software. In the centre, the push button is used to turn the camera ON. A short press will then send a signal to the microcontroller, that will safely power OFF through the software once all current tasks are completed. To force power OFF the system, in case of a software or hardware error, the button must be pressed for about 3 seconds when powered by the batteries or for about 10 seconds when solar or USB power is connected. Additional multi-press button functions can be customized.

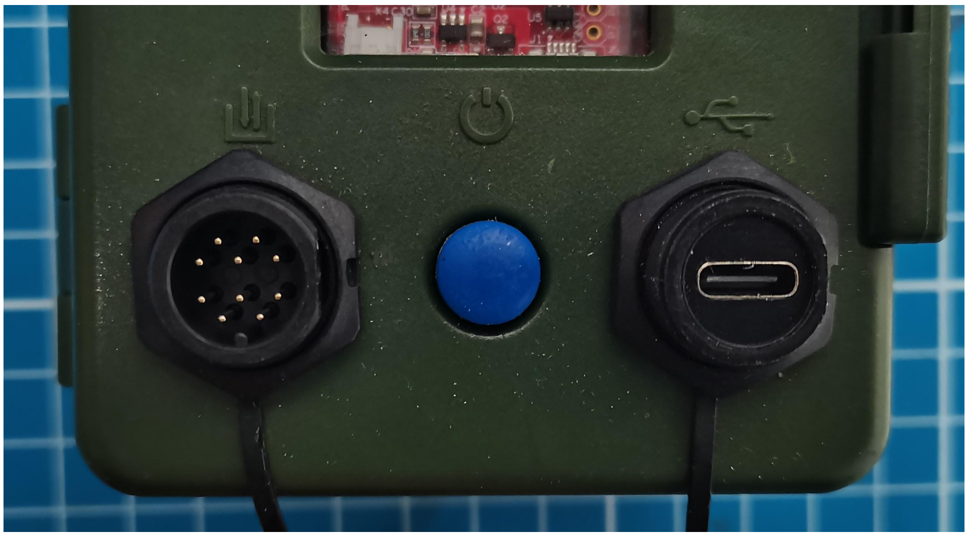
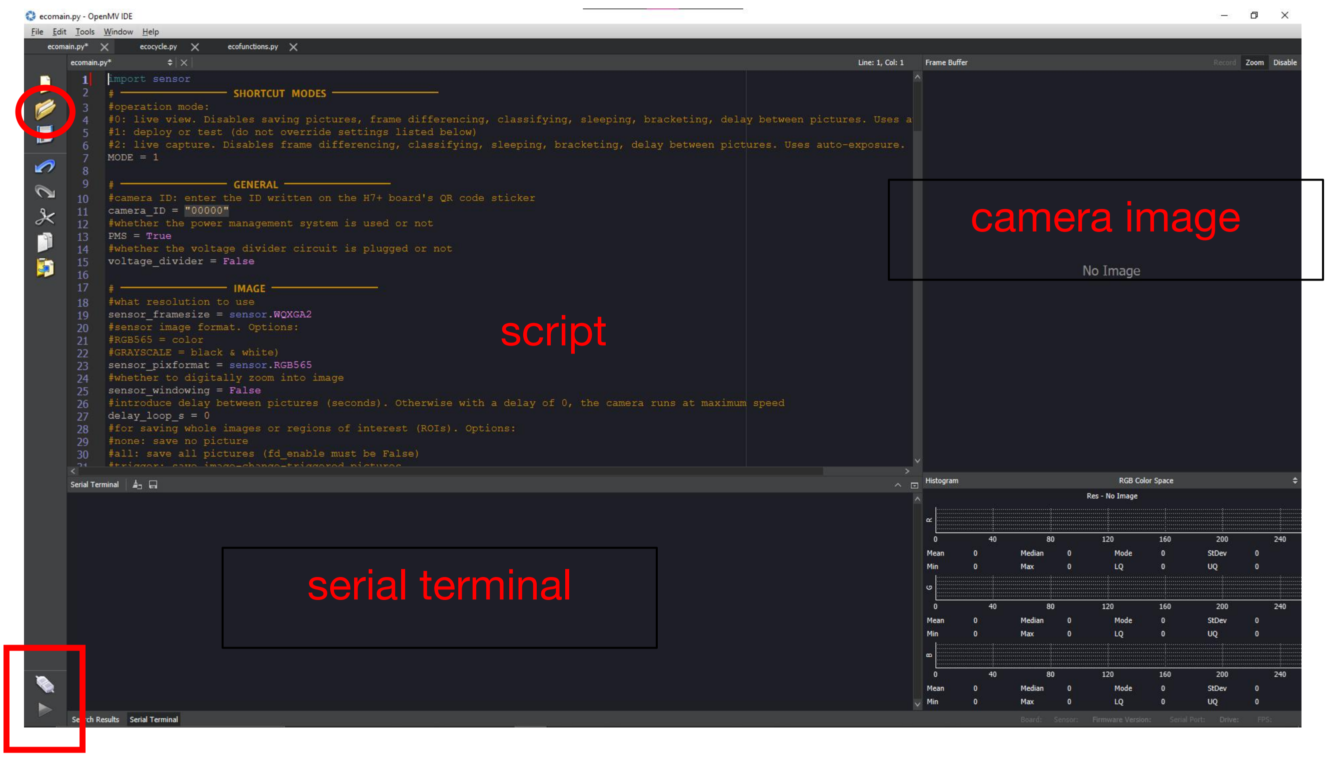


Figure 5. Front view onto connectors and power button.

Software

At first, the openMV IDE (Fig. 6) should be download here and installed on a computer to allow easy access and changes to the program script. After formatting the SD card, the files *ecofunctions.py, pcf8563.py*, ecocycle.*py* and *ecomain.py* should be copied to its root.



```
import sensor
# ——————————— SHORTCUT MODES ———————————
#operation mode:
#0: live view. Disables saving pictures, frame differencing, classifying, sleeping, bracketing, delay between pictures. Uses a
#1: deploy or test (do not override settings listed below)
#2: live capture. Disables frame differencing, classifying, sleeping, bracketing, delay between pictures. Uses auto-exposure.
MODE = 1

# ——————————— GENERAL ———————————
#camera ID: enter the ID written on the H7+ board's QR code sticker
camera_ID = "00000"
#whether the power management system is used or not
PMS = True
#whether the voltage divider circuit is plugged or not
voltage_divider = False

# ——————————— IMAGE ———————————
#what resolution to use
sensor_framesize = sensor.WQXGA2
#sensor image format. Options:
#RGB565 = color
#GRAYSCALE = black & white)
sensor_pixformat = sensor.RGB565
#whether to digitally zoom into image
sensor_windowing = False
#introduce delay between pictures (seconds). Otherwise with a delay of 0, the camera runs at maximum speed
delay_loop_s = 0
#for saving whole images or regions of interest (ROIs). Options:
#none: save no picture
#all: save all pictures (fd_enable must be False)
#trigger: save image-change-triggered pictures
```

Figure 6. OpenMV IDE

1.) The first step is to connect the camera to the computer using a USB type-C data cable. Turn ON the camera with the blue push button and click the cable button in the lower left corner (red rectangle Fig. 6) as soon as the USB icon appears on it (Fig. 7).

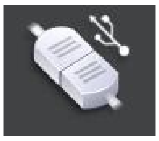


Figure 7. USB icon above connector

If it cannot connect, the camera script is executing a task or has entered a sleep mode. In this case, press the blue push button once to let the device power OFF itself (or force power OFF if an error occurred), remove the SD card from the openMV board, and delete the *main.py* file directly from the computer's SD card slot or a USB-SD card reader.

Note : Update the openMV board firmware if prompted by clicking 'OK'.

2.) To configure the scripts from the IDE, follow *File>Open* or the folder icon (red circle Fig. 6) and open the *ecomain.py* script located on the SD card. By adjusting the parameters in this script, the camera can be configured to the desired operation. To run a simple live view, mostly used for focusing and framing the image, change the 'MODE' parameter to 0. To execute the currently open script, hit the green play button (Fig. 8) in the lower left corner.

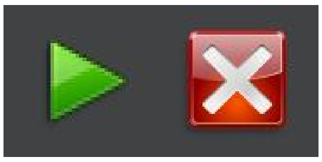


Figure 8.

The script should now be executing its tasks. Information about the process can be observed on the serial terminal (Fig. 6). While running the script, the green play button changes to a white on red 'X' (Fig. 8), which is used to stop the script.

Note : After removing and reinstalling the batteries from their compartment or if the low-battery protection cut-off has occurred, the time and date of the device needs to be reconfigured from the IDE, with a single run of the *rtc_set.py* script with current date and time. For this, the *pcf8563.py* library must be on the SD card.

By changing the 'MODE' back to 1, the script takes all the user-defined parameters into account. In this mode, the user must press the button within 2 seconds as soon as the blue LED lights up to confirm a script start. If the delay loop is zero (i.e., the camera will not go into deep sleep or power OFF mode), another run is recommended to verify the proper operations. If everything is working according to the user's specifications, the script can be stopped and the *ecomain.py* should be saved to the camera by clicking *Tools>Save open script to OpenMV Cam (as main.py)*. The script will automatically be renamed to *main.py* (note that the script must be named main.py to be recognized and run by the microcontroller). The camera can now be ejected by pushing the socket button (Fig. 9) and long pressing the blue switch button to force power OFF the system.

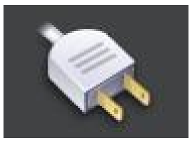


Figure 9.

3.) To run the configured operations of the camera, it must now be powered ON by pressing the blue switch button once. After confirming the script start with the second button press during the blue LED indication, the script will run with the operations configured previously. Similarly, to turn OFF the camera and stop the current run, the push button must be pressed once and when the script finishes ongoing operations, the LED shines cyan for 2 seconds during which a second button press will confirm the end of the script and the power OFF of the device.

Output data

After finishing a camera deployment, the SD card should contain a new folder named after the date and time of the deployment start and the camera ID. Inside this folder is a 'jpegs' folder containing all images taken and saved by the camera, and 3 CSV files with useful information about the images, the detections (if frame differencing or an image classification model is used) and the running status of the device.

Model Building

Users should choose images with the target at different angles and positions, to build a model. These images are then added to the EDGE IMPULSE project and labelled for the learning process. Using more training images (generally above 100) will result in a good model which is defined by the accuracy test. Once a satisfying accuracy is reached (>66% for embedded computer vision applications), the new model file can be generated along with the labels list and copied onto the root of the SD card. With this model, the ecoEye camera can perform real-time image or object classification. The result is the captured image with a label saved on the SD card along with other information about the detections in the corresponding CSV files.

4.) In the case of a code error, the serial terminal or message prompt (when running from the IDE) will display a message about the reason of the error. When the camera is not running from the IDE and an error occurs, the LED blinks red a few times and continues with the script. At this point, a text file named 'error.txt' is created on the root of the SD card, containing information about the error such as line number, time and reason. For debugging these errors, go visit the FAQ or contact the technical support.

Resources

OpenMV Cam Tutorials

Edge impulse object detection model using FOMO

Edge impulse image classification model