**Real-time alerts from AI-enabled camera traps using the Iridium satellite network: a case-study in Gabon, Central Africa**

**Supplementary Material: System components and design**

Contents

[**1.** **Hardware stack** 2](#_Toc90405653)

[**2.** **Software stack** 6](#_Toc90405654)

[**3.** **Supplementary figures to main text** 9](#_Toc90405655)

[**4.** **Supplementary tables to main text** 11](#_Toc90405656)

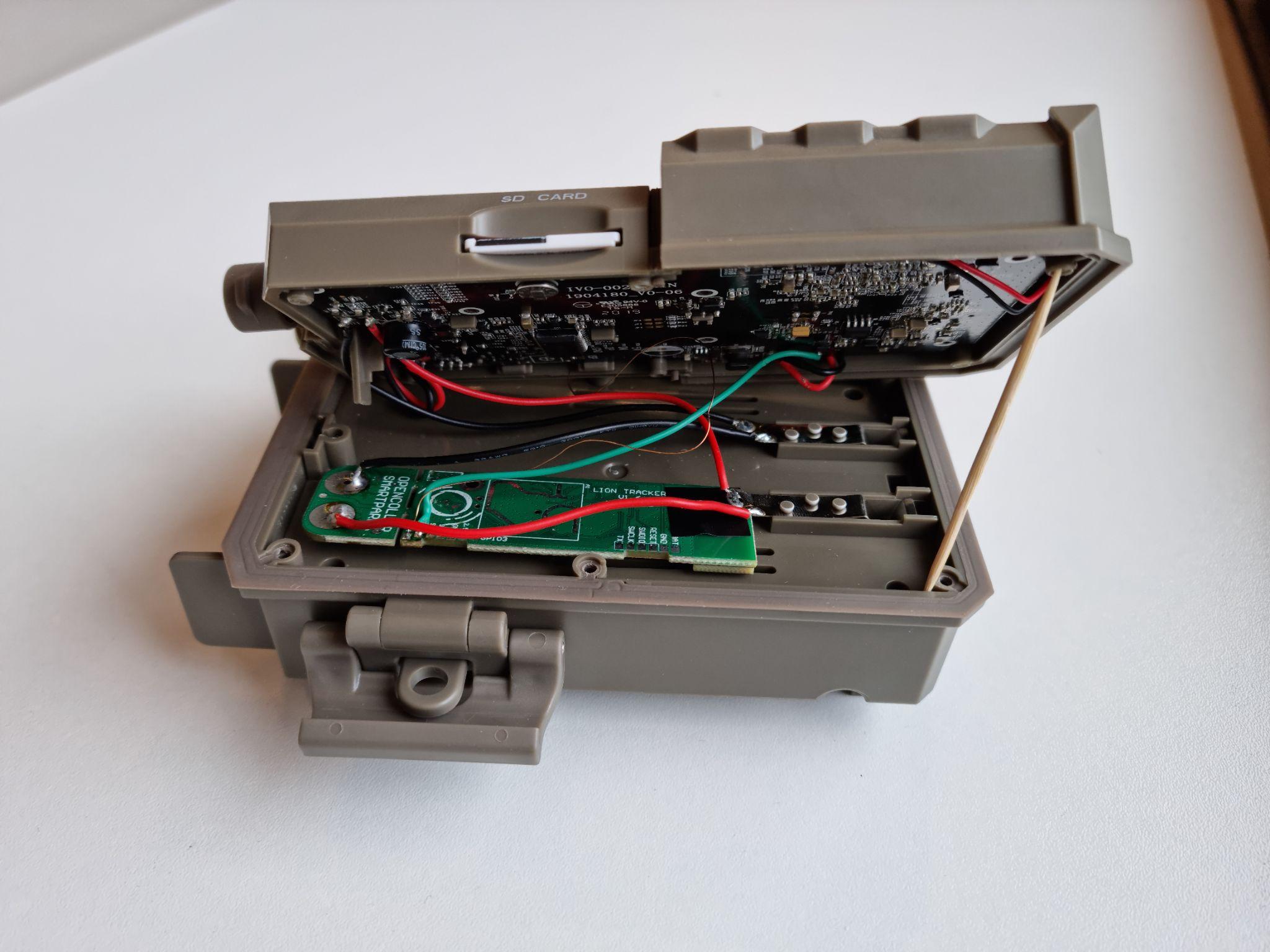
[**5.** **References** 14](#_Toc90405657)

1. **Hardware stack**

*Camera*

We used a BushnellTM Core 24MP Low Glow 119936C camera trap for development but similar modifications can be made to other models and brands. The camera was set to take single images (2304 x 1296 pixels, 72 dpi) at 10s intervals with sensitivity set to auto, and the flash was set to low power mode. Normally, the BushnellTM immediately cuts power to the SD card once it has finished writing an image or images. This does not allow sufficient time for images to be transmitted from the WiFi SD card to the smart bridge using the WiFi network. To address this, the custom microcontroller keeps the WiFI SD card powered on until the images have been transmitted to the smart bridge. The WiFi SD card is secured permanently into the camera (to prevent the power connection being damaged, Figure S1), but the images are also stored on a removable micro-SD for later download if required.

A picture containing text, electronics

Description automatically generated

**Figure S1.** Modified BushnellTM camera trap showing the LoRA relay and printed circuit board, the WiFi SD card and power supply (removed and installed).

*Smart-bridge*

The Smart-bridge (Figure 2) contains a custom printed circuit board (PCB) with a LoRa STM32L0 ultra-low-power microcontroller, RockBLOCK satellite modem and connections for a Raspberry Pi 4 Compute module. The hardware is stored in a weatherproof NANUK NANO 330 case (L188 x W130 x H65 mm). By default, the Raspberry Pi is turned off and thus the system consumes minimal power (less than 50 microampère; see *Power* later). When the smart bridge receives a LoRa message from a nearby camera it turns on the Raspberry Pi, which then downloads and classifies the images from the BushnellTM using artificial intelligence. After sending the results over the satellite network (see later), the system powers down.

*A picture containing electronics

Description automatically generated*

**Figure S2.** Smart Bridge PCB

*Raspberry Pi*

The Raspberry Pi 4 compute module is integrated onto the Smart-bridge PCB with a pair of 100-pin mezzanine connectors. Raspberry Pis provide an excellent platform for development purposes and have been used widely in ecology (Jolles, 2021; Sethi et al., 2018; Sturley & Matalonga, 2020). Furthermore, although the Raspberry Pi 4 is power inefficient relative to other similar boards on the market (e.g. Arduino based systems), the Pi 4 can run artificial intelligence models built on relatively large architectures. Our approach of only briefly powering the Pi when needed allowed us to harness its computational power in an energy-optimised way.

The Raspberry Pi 4 Compute module runs Raspbian lite and Python 3 scripts together with the Tensorflow Lite runtime to fetch the images and run the artificial intelligence model. A SQLLite database is used to track image status (download status, transmission status etc).

*Satellite modem*

There are many satellite networks available for civilian use. We chose the Iridium satellite network because it has near global coverage, is relatively inexpensive, and has widely available hardware including miniaturised, low-power modems. The Iridium network is also well known in the ecology community where it is regularly used for animal tracking using GPS collars. We used the RockBLOCK 9603 modem from Rock Seven to connect to the Iridium network.

*Environmental data*

The smart bridge PCB is equipped with a temperature, humidity and barometric pressure sensor. Since these are mounted directly on the PCB they are not currently suitable for external environmental monitoring (other than barometric pressure) but they are useful for evaluating if the smart bridge is intact. For example, the smart bridge housing is completely sealed once closed and contains silica gel. In a humid environment such as a tropical forest, the humidity should drop once the bridge is installed and closed. A future rise in humidity could be used as an indicator of a possible hardware problem. We do not present data from these sensors or discuss them further.

*Power*

For the smart bridge we used six NCR18650PF rechargeable batteries totalling 16,500 mAh power and a 6-volt 6 watt solar panel for charging. Initial testing in the Netherlands showed an active smart bridge, processing and transmitting approximately 17 images per day (see Results in main text), could be powered indefinitely by a solar panel without intervention.

For the BushnellTM camera trap, we used six Energizer© Ultimate LithiumTM AA batteries (non-rechargeable). Normally the BushnellTM has a battery life of approximately one year using these batteries. The addition of the microcontroller and the WiFi SD card draws additional power, however, which will reduce deployment times. During testing in the Netherlands the camera achieved three months of battery life when activated up to 17 times per day on average (see Results in main text). We expect field deployment times to be longer than this since the camera is likely to be triggered less frequently when correctly installed and parameterized.

*Optimising alerts and minimizing data transmission costs*

The Iridium satellite network supports short burst data and a maximum of 340 bytes per transmission. Satellite data is relatively expensive so we optimised the alerts to maximise the amount of information transmitted per message. The timestamp was reduced to 4 bytes by sending the number of elapsed seconds since January 1st 2010. The image label from the artificial intelligence model (e.g. elephant) was mapped to a 1 byte number and later converted back to a text label on the web backend. All other data, like AI prediction ‘confidence’ for the top-scoring species label (softmax algorithm probabilities), temperature and smart bridge voltage are mapped to 1 byte numbers. This allowed us to send up to 55 image classification results in a single satellite message.

1. **Software stack**

*Artificial intelligence model*

Our aim was to provide reliable alerts of species detections without requiring images to be transmitted to the end-user over a wireless network. Since our focus was on forest elephants during the pilot, we initially tested the model from (Whytock et al., 2021), which classifies 26 central African forest mammal and bird species, including forest elephants. However, the model was built using a relatively large convolutional neural network (CNN) architecture (ResNet50) and is 100 MB in size. This model took over 20 seconds to classify a single image using the Raspberry Pi 4 compute module, which drew a substantial amount of power and made the model unsuitable for our purposes.

To find a suitable alternative architecture to ResNet50, we compared inference times among a suite of 16 pre-trained computer vision models using their Fast.ai (Howard & Gugger, 2020) implementations (see Figure S3 for results). We did not evaluate classification accuracy using these models but only inference times. Then, we trained a Tensorflow Lite model (using Google Cloud’s AutoML service) and a Fast.ai model (SqueezeNet 1.1, the second-fastest from our tests) using a dataset of 105,000 images (a subset from Whytock et al. (2021)) with three, almost equally distributed classes (elephant, human and other). For these two models, we compared model precision and accuracy using a smaller, held out subset of 14,642 images, with almost equal distribution among the classes. We found that the TensorFlow Lite model provided the shortest inference time (~100 ms vs ~1200 ms for SqueezeNet) and precision and accuracy was similar between the two architectures (Table S1). Therefore, the Tensorflow Lite model trained using AutoML was chosen for deployment during the pilot.

**Chart, bar chart

Description automatically generated**

**Figure S3.** Mean inference time in seconds (*n* = 8 images 224 x 224 pixels) for 16 pre-trained computer vision CNN architectures run on the Raspberry Pi 4 compute module using PyTorch.

**Table S1.** Comparison between model accuracy for the Fast.ai SqueezeNet model and the TensorFlow Lite model trained using three classes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Class** | **Number correct** | **Percentage correct** |
| SqueezeNet | Human | 4529 / 5000 | 90.58% |
| TensorFlow Lite | Human | 4631 / 5000 | 92.62% |
| SqueezeNet | Elephant\_African | 4516 / 5000 | 90.32% |
| Tensorflow Lite | Elephant\_African | 4507 / 5000 | 90.14% |
| SqueezeNet | Other | 4358 / 5000 | 87.16% |
| Tensorflow Lite | Other | 4586 / 5000 | 91.72% |

*Back-end*

An important element of receiving real-time alerts from camera traps is a centralised platform that can be used to receive, interpret and display the incoming data. Following our philosophy of using existing technology, we integrated the system with the EarthRanger platform (www.earthranger.com). Incoming data is first stored on our own Django-based back end. Once an alert is received the raw data is stored in a SQL database. A task-queue based system is then used to send the data to integrated platforms (e.g. EarthRanger or others). As well as offering a web-platform and mapping capabilities for displaying alerts, EarthRanger can also be configured to send messages in real-time using WhatsAppTM, short message service (SMS), e-mail and other methods.

1. **Supplementary figures to main text**

Diagram

Description automatically generated

**Figure S3.** Alert times for each of *n* = 4 camera stations.

Graphical user interface, application, table, Excel

Description automatically generated

**Figure S4.** Confusion matrix for image-based classification

1. **Supplementary tables to main text**

**Table S2.** Images captured during each day of the field test for each location. NA indicates the system was deactivated.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Day (June 2021)** | **Forest East** | **Forest West** | **Cayette** | **Kazamabika** | **SEGC** |
| 13 | 12 | 2 | NA | NA | 50 |
| 14 | 18 | 4 | NA | 24 | 5 |
| 15 | 13 | 14 | NA | 1 | NA |
| 16 | 13 | 29 | 5 | 5 | NA |
| 17 | 8 | 4 | 4 | 32 | NA |
| 18 | 17 | 5 | 0 | 0 | NA |
| 19 | 1 | 8 | 3 | 0 | NA |
| 20 | 10 | 12 | 1 | 0 | NA |
| 21 | 7 | 9 | 11 | 32 | NA |
| 22 | 1 | 8 | 7 | 13 | NA |
| 23 | 16 | 30 | 2 | 1 | NA |
| 24 | 7 | 2 | 7 | 2 | NA |
| 25 | 12 | 7 | 7 | 27 | NA |
| 26 | 8 | 40 | 2 | 2 | 8 |
| 27 | 38 | 18 | 7 | 18 | 4 |
| 28 | 12 | 34 | 2 | 1 | 4 |
| 29 | 25 | 1 | 3 | 6 | 4 |
| 30 | 4 | 27 | 13 | 3 | 5 |

**Table S3.** Mean, minimum and maximum time difference between image creation time and alert time for four sites.

|  |  |  |  |
| --- | --- | --- | --- |
| **Site** | **Minutes** | **Min.** | **Max.** |
| Forest East | 6.9 | 2.37 | 863.8 |
| Forest West | 6.5 | 1.28 | 1299.2 |
| Kazamabika | 306.3 | 1.68 | 9473.9 |
| SEGC | < 1 | < 1 | 1277.1 |

**Table S4.** Model performance by class for *n* = 571 images and *n* = 142 events using vote counting.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test statistic** | **Elephant\_African** | **Human** | **Other** |
| *Images* |  |  |  |
| Pos Pred Value | 0.82 | 0.92 | 0.81 |
| Neg Pred Value | 0.89 | 0.95 | 0.91 |
| Precision | 0.82 | 0.92 | 0.81 |
| Recall | 0.86 | 0.80 | 0.82 |
| F1 | 0.84 | 0.86 | 0.82 |
| Prevalence | 0.44 | 0.22 | 0.34 |
| Detection Rate | 0.38 | 0.18 | 0.28 |
| Detection Prevalence | 0.46 | 0.20 | 0.34 |
| Balanced Accuracy | 0.86 | 0.89 | 0.86 |
|  |  |  |  |
| *Events* |  |  |  |
| Pos Pred Value | 0.62 | 0.72 | 0.94 |
| Neg Pred Value | 0.97 | 0.96 | 0.72 |
| Precision | 0.62 | 0.72 | 0.94 |
| Recall | 0.90 | 0.81 | 0.77 |
| F1 | 0.73 | 0.76 | 0.85 |
| Prevalence | 0.20 | 0.18 | 0.61 |
| Detection Rate | 0.18 | 0.15 | 0.47 |
| Detection Prevalence | 0.30 | 0.20 | 0.50 |
| Balanced Accuracy | 0.88 | 0.87 | 0.85 |

1. **References**

Howard, J., & Gugger, S. (2020). Fastai: A Layered API for Deep Learning. *Information*, *11*(2), 108. https://doi.org/10.3390/info11020108

Jolles, J. W. (2021). Broad-scale applications of the Raspberry Pi: A review and guide for biologists. *Methods in Ecology and Evolution*, *n/a*(n/a). https://doi.org/10.1111/2041-210X.13652

Sethi, S. S., Ewers, R. M., Jones, N. S., Orme, C. D. L., & Picinali, L. (2018). Robust, real-time and autonomous monitoring of ecosystems with an open, low-cost, networked device. *Methods in Ecology and Evolution*, *9*(12), 2383–2387. https://doi.org/10.1111/2041-210X.13089

Sturley, S., & Matalonga, S. (2020). PANDI: A Hybrid Open Source Edge-based System for Environmental and Real-Time Passive Acoustic Monitoring - Prototype Design and Development. *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, 1–6. https://doi.org/10.1109/IRASET48871.2020.9092006

Whytock, R. C., Świeżewski, J., Zwerts, J. A., Bara‐Słupski, T., Pambo, A. F. K., Rogala, M., Bahaa‐el‐din, L., Boekee, K., Brittain, S., Cardoso, A. W., Henschel, P., Lehmann, D., Momboua, B., Opepa, C. K., Orbell, C., Pitman, R. T., Robinson, H. S., & Abernethy, K. A. (2021). Robust ecological analysis of camera trap data labelled by a machine learning model. *Methods in Ecology and Evolution*, *12*(6). https://doi.org/10.1111/2041-210X.13576