# **Trail cam**

**1. Arduino Nano 33 BLE Sense**

* An excellent choice for low-power AI inference.

**2. Arducam Mini Module Camera Shield with OV2640**

Compatible with Arduino Nano 33 BLE Sense, great for lightweight image processing tasks. (**Pins Required:** The Arducam Mini communicates with the Arduino via SPI and I2C interfaces. Make sure to connect the following:

SPI Pins: MISO, MOSI, SCK, and CS.

I2C Pins: SDA and SCL.

Power: Connect VCC and GND.)

**3. IR LED Lights**

* Provide sufficient illumination for night vision.
* Consider testing different power levels and wavelengths for optimal camera sensitivity.

**4. PIR Motion Sensor**

* Detects movement to trigger the camera and minimize power consumption.
* Ensure that it is properly aligned and shielded to avoid false positives.

**5. Battery**

* Select a battery pack or power source with sufficient capacity.
* Rechargeable batteries or solar power setups are great for field deployment.

**6. Weatherproof Enclosure**

* Protect your electronics from the elements.
* Make sure the enclosure allows IR light to pass through easily and is properly vented.

**7. AI Model & Logic**

* Design and train your AI model to differentiate the Tsushima yamaneko from other local wildlife.
* Use a framework like TensorFlow Lite for Microcontrollers or Edge Impulse to create and deploy an optimized model.
* Write the code to handle PIR-triggered image capture and AI classification.

**8. Storage**

* 64GB SD card

**Additional Considerations**

* **Network:** If data needs to be collected remotely, a wireless communication module could help.
* **Power Efficiency:** Implement sleep modes or power-saving measures to conserve energy.
* **Tensorflow lite for TinyML**

# **TinyML model**

**Steps to Train Your TinyML Model**

1. **Collect and Prepare Data:**
   * **Data Collection:** Gather images of the Tsushima yamaneko and other local wildlife that you want your model to distinguish.
   * **Preprocessing:** Resize images to a smaller resolution (e.g., 96x96 or 128x128) to match the camera and minimize the model size.
   * **Labeling:** Organize and label your dataset into categories.
2. **Model Design and Training:**
   * **Framework:** Use TensorFlow Lite for Microcontrollers or Edge Impulse to design and train a lightweight model.
   * **Architecture:** Consider using a pre-trained model like MobileNetV2 or train a custom convolutional neural network (CNN).
   * **Training Process:** Train the model on your computer using GPU acceleration if available.
3. **Model Optimization:**
   * **Quantization:** Reduce model size and optimize for inference on microcontrollers by using post-training quantization.
   * **Pruning:** Consider model pruning techniques to reduce the size further.
4. **Export for Deployment:**
   * **TFLite Model:** Convert your trained model into TensorFlow Lite format.
   * **Validation:** Test the model on your PC using the test dataset to ensure accuracy and efficiency.
5. **Setup Software Development Environment:**
   * **Platform:** Install Arduino IDE or a similar development environment
   * **Libraries:** Download and set up relevant machine learning and camera libraries.
6. **Prepare Deployment Code:**
   * Create a skeleton code base that handles loading the model, motion detection, image capture, and decision logic.

**Testing**

* Once the hardware arrives, you can directly deploy the pre-trained model onto the microcontroller.
* Test and refine the system using real-world images captured by the camera.