

Comptage d'humains et d'animaux en milieu naturel

Présentation mini-projet M2M 7 avril 2020



Image: Premium Camera Nest Box System®

Binôme: WAMPACH Sylvestre, DRIOWYA Abdelghafour.

Comptage d'humains et d'animaux en milieu naturel

M2M 2019-2020

Binôme:

WAMPACH Sylvestre
DRIOWYA Abdelghafour

Bonjour,

Dans cette présentation de notre mini-projet, nous allons vous présenter notre travail réalisé dans le cadre du cours de M2M.

Le projet consiste à utiliser une mini carte électronique équipé d'une caméra pour réaliser un comptage des événements détecté devant la caméra.

L'idée est de pouvoir intégrer l'ensemble comme sur l'image (**Premium Caméra Nest Box System**). Dans un boîtier fixé par exemple sur un arbre.

Applications IoT: Comptage d'humains et d'animaux en milieu naturel

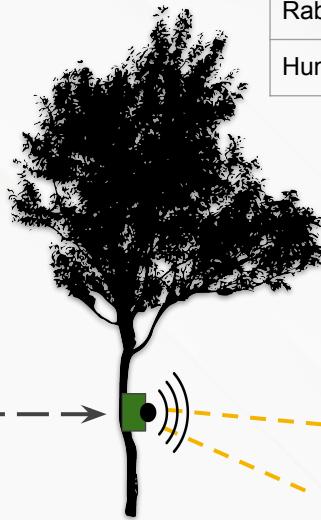


Le comptage d'animaux et de personnes en milieu naturel peut être important pour des employés de parcs nationaux, gardes de classe, zoologistes (etc.) qui souhaitent surveiller les activités dans des secteurs spécifiques.

Surveillance de l'environnement, faune et flore. Et permettre d'améliorer la protection des zones surveillées.

Le Maix bit est une carte électronique qui peut être équipée d'une caméra et d'un algorithme de reconnaissance d'images afin de compter le nombre de personnes et d'animaux se déplaçant dans une zone.

Applications IoT: Comptage d'humains et d'animaux en milieu naturel



Entity	Accuracy	Time
Rabbit	0.87	2020-01-27 04:16:22
Humain	0.95	2020-04-05 11:16:47

Équipé d'une caméra et d'un algorithme de reconnaissance d'images, l'équipement enregistre dans une base de données les événements. Il peut ensuite transmettre les données à l'aide d'un réseau de type LoRaWAN.

Dans un scénario d'usage, où un nombre important de caméras sont réparties sur une grande surface géographique, cela peut permettre de surveiller des régions entières.

Les données récupérées peuvent ainsi servir à des organisations pour faire des études statistiques grâce aux données collectées par ces objets connectés.

Composants logiciels / matériels utilisés



MaixPy IDE



Équipement Sipeed MaiX Bit



Grafana plugin: Grafana worldmap panel



Capture d'écran d'un jeu de données sur Grafana

MaixPy

Pour réaliser ce projet nous avons utilisé la carte électronique Sipeed Maix Bit. Cette carte est destinée à exécuter des programmes utilisant l'intelligence artificielle.

Cette carte par ces petites dimensions, un processeur performant (jusqu'à 800 MHz) et une basse consommation permet des applications dans le monde des objets connectés.

Maix bit contient un KPU : un processeur supplémentaire avec le support de convolution.

Maix offre des solutions rapides pour déployer des projets facilement grâce à une collection de bibliothèques. Un IDE permet de facilement visualiser le flux vidéo de la carte électronique et de téléverser le programme sur la carte.

Lors de la réalisation de ce projet nous avons utilisé tous les accessoires présents dans le kit.

Ainsi nous avons utilisé l'écran tactile LCD pour afficher le programme de reconnaissance d'images (voir vidéo dans README du git).

Les images sont capturées à l'aide d'une caméra de 3 Mégapixel.

Et l'ensemble est alimenté par un câble USB relié à un ordinateur.

Dans un projet idéal (utilisation en production) l'écran ne serait pas utile car l'équipement n'est pas destiné à être accessible facilement par un opérateur.

En effet cet appareil est destiné à surveiller une zone en étant accroché sur un support (arbre/ poteaux) et lorsque l'algorithme détecte une entité par la caméra (personne / Animal) grâce à la reconnaissance d'image, il pourra stocker l'information dans une base de données textuelle.

De plus l'écran consomme énormément de batterie ce qui n'est pas idéal pour objet connecté installé en milieu naturel.

Ici dans notre projet nous ne disposons pas d'antenne de communication mais nous pouvons transmettre des données à l'aide du port série USB.

Ces données transmises par l'appareil connecté peuvent être par la suite visualisé à l'aide d'un logiciel comme grafana.

Pour simplifier la visualisation de données sur grafana nous avons développé en python un script de génération d'un jeu de données simulant les données transmises par des boîtiers de surveillances.

À l'aide de plugin est des graphiques nous pouvons visualiser plus facilement les données et même créer des alertes. Cet outil (grafana) permettrait de surveiller une flotte de plusieurs objets assez facilement.

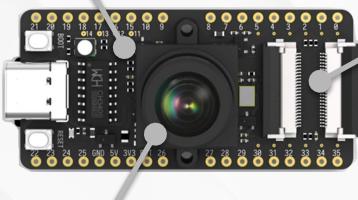
Voir vidéo dans le readme:

https://github.com/Th3CracKed/M2M_Image_Recognition/tree/master/Grafana

les métriques (langages de programmation, sloc, performance ...)

Processeur:

RISC-V Dual Core 64bit, 400MHz-800MHz



Caméra:

3 Mégapixel / 19fps

Ram:
8MB



MySQL



Micropython



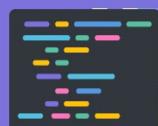
Reconnaissance d'images:

20 classes d'images reconnu



Bibliothèque:

30 lignes de code dans 2.8 MB



Nous avons utilisé le modèle tiny-yolov2 développé en Micropython.

Ce modèle peut reconnaître 20 classes: ['aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor']

<https://blog.sipeed.com/p/677.html>

Concernant les données visualisées dans grafana nous avons choisi du MySQL.

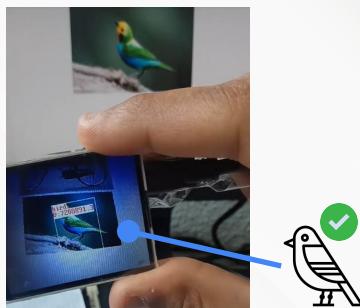
Il est important de noter que le format JSON est probablement plus adapté lorsqu'il s'agit de transférer de l'information par un réseau de type LoRaWan (moins lourd).

Dans ce projet nous n'avons pas réussi à tester le transfert de données de la carte vers le PC. Nous sommes parvenu à afficher les logs mais pas à enregistrer par exemple à l'aide d'un socket directement les données dans une base de données.

Les architectures implémentées

Modèle utilisé: [tiny-yolov2](#)

- Redirection du flux vidéo sur l'écran LCD
- Envoi d'information textuel (Port USB).



Reconnaissance d'image détectant un chat et un oiseau avec le Sipeed Maix bit et tiny-yolov2.



Pour monter en compétence sur l'utilisation de Maix Bit, nous avons choisi d'implémenter d'une façon incrémentale la solution envisagé. Pour cela nous avons d'abord expérimenté des petits programmes exemples (blink LED etc.), pour vérifier si nous arrivions à transférer et exécuter le programme sur la carte. Suite aux tests nous avons poursuivi sur l'utilisation de la caméra qui est le périphérique principal de ce projet (nécessaire pour la reconnaissance des images). Nous avons réussi à faire afficher le flux vidéo capté par la caméra sur l'écran LCD. Ensuite nous avons utilisé un modèle de détection de visage qui ne requiert pas de changement de firmware préinstallé. Finalement après de nombreuses recherches et tests nous avons réussi à utiliser un modèle pré-entraîné qui est basé sur le système de détection YOLO (you only look once) et qui permet l'évaluation l'image en temps réel du flux vidéo capté par la caméra et d'afficher le résultat sur l'écran LCD.

Phase de flashage :

Suite à la taille de mémoire limitée de la ram 8mo, nous avons utilisé un firmware minimal qui possède uniquement des fonctions nécessaires pour notre programme.

Le modèle est placé dans l'adresse 0x500000 de la mémoire.

Phase d'upload du programme:

Maix bit expose une api pour pouvoir utiliser un processeur KPU.

Elle permet de charger le modèle du 0x500000 de la mémoire, et retourne un 'kpu network object'.

```
kpuNetworkObject = kpu.load(leModel)
```

Ce kpu network object permet d'initialiser yolo2 network

```
kpu.init_yolo2(kpuNetworkObject , ...parameters)
```

Ensuite grâce à une boucle infinie :

- Le programme prend à chaque fois une image avec la caméra.
- Puis utilise le Yolo2 network pour évaluer l'image.
- Affiche les informations adéquates en cas de détection d'un objet.
- Ou un affichage simple de l'image si non.

```
While(true) {  
    img = sensor.snapshot() // prendre une photo  
    code = kpu.run_yolo2(task, img) // evaluer cette photo  
    // Affichage du résultat sur l'écran  
}
```

Remarque

Puisque nous n'avons pas de mémoire flash (carte SD), nous étions limité par une taille mémoire interne maximale de 5.5 MB pour le modèle de données.

les métriques (langages de programmation, sloc, performance ...)



GPS
Latitude
Longitude



Précision
de
l'identification



Date
&
Heure



Niveau
de
batterie



Statistiques
des
passages

Métriques:

Nous pouvons collecter les données suivantes pour chaque équipement.

- Données de position GPS du boîtier (stocké en BDD lors de l'installation du boîtier sur la zone par un équipement externe) Le point GPS est lié à un identifiant unique.
- Le pourcentage de précision (fiabilité) de la reconnaissance d'image par l'algorithme.
- La date et l'heure de chaque événement. Et également le type du passage (Personne / animal) par boîtier.
- Le niveau de batterie

Aujourd'hui le niveau de batterie est simulé car nous n'avons utilisé uniquement le port USB pour l'alimentation.

Voir Grafana + script python de génération des données pour plus de détails sur la visualisation des données sur une interface.

Nous pensons qu'avec ce type de donnée, qu'il est suffisant et ainsi possible d'obtenir des résultats satisfaisant en matière de statistiques et surveillance d'une zone.

Les problèmes rencontrés et les solutions élaborées



- Trouver de la documentation relative aux erreurs rencontrées.
- Trouver des ressources en ligne



- Version du firmware
- Utilisation de MicroPython / C
- Connecter la carte sur nos ordinateurs.
- Téléverser le programme sur la carte depuis l'IDE

Lors de la réalisation de ce projet nous avons rencontré plusieurs difficultés.

La documentation en ligne n'est pas très grande. Ainsi il nous a été difficile de déboguer les problèmes que nous rencontrions avec cette carte.

Par exemple pour installer le modèle de reconnaissance d'image il faut flasher un firmware spécifique sur la carte. Il ne nous a pas été évident de comprendre que le problème provenait du firmware.

Nous avons pu détruire le fonctionnement de la carte et des fonctions à l'aide de vidéos et des projets existants sur les différentes plateformes en ligne.

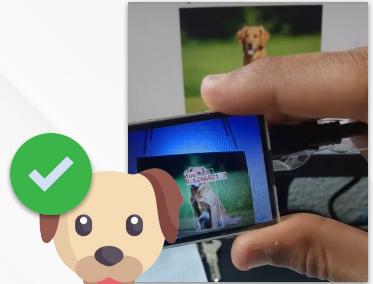
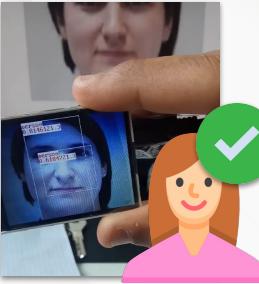
L'IDE par défaut MaixPy IDE s'est avéré être une source de problèmes lors de nos tests. En effet nous avons découvert pour transférer du code depuis l'IDE vers la carte MaixBit, pouvait entraîner des problèmes d'exécution sur la carte. La solution a été de ne pas utiliser l'IDE mais d'utiliser la console. Le système d'exploitation à aussi était un point de blocage. Nous sommes passé de Windows à linux en utilisant la distribution Ubuntu.

Transfert des données collectées:

Nous sommes parvenu à afficher les logs des données de la carte dans une console sur l'ordinateur à l'aide du port USB, mais nous ne sommes pas parvenu à enregistrer (par exemple à l'aide d'un socket) directement les données de la carte vers une base de données. Cependant nous avons

supposé qu'il était probablement plus simple d'utiliser un module supplémentaire sur la carte pour accomplir la fonction d'envoi des données. En effet dans un scénario d'usage, le câble USB n'est pas une solution envisageable pour transférer des données sur de longues distances. Nous n'avons donc pas approfondi ce point dans ce projet.

Conclusion



Photos de l'algorithme en fonctionnement sur notre projet

Video démo reconnaissance d'images: https://drive.google.com/file/d/1MpDJM1vwmq-47Gfrhz_TaW3qFaYXqid/view

Video démo grafana: https://drive.google.com/file/d/1to3Lmrpa5p6n_IdvlMCuhmxqRK6jarPy/view

Pour conclure,

Nous sommes avec ce projet de comptage d'humains et d'animaux en milieu naturel parvenus à utiliser un réseau neuronal existant, permettant d'identifier une vingtaine d'images/objet différentes.

Malgré certaines complications rencontrées pour utiliser cet équipement, nous avons un prototype fonctionnel permettant d'identifier les problématiques d'un tel appareil de comptage (Pour plus d'informations voient les fiches analyse de Cycle de Vie et Privacy & Security).

Par l'absence en notre possession de par exemple un modem LoRaWAN, nous n'avons pas pu mettre en avant dans un cas concret, les contraintes et les bonnes solutions pour transmettre les données vers une base de données.

Le kit que nous avons ne serait pas adapté pour une utilisation dans un usage réel, en effet la caméra n'est pas assez performante / nette pour détecter des objets au loin.

Comme nous ne sommes pas parvenu à transmettre les données nous avons utilisé notre script python pour générer des données et les visualiser à l'aide du logiciel grafana.

Globalement le projet a été une découverte positive du matériel du type IOT.

Plus d'informations disponibles dans les READMES du dépôt git. Et les fiches analyse de Cycle de Vie et Privacy & Security.