Internship
Report

Design of a remote device to autonomously detect rats in Guatemala sugar can fields.

Submitted by

EDOUARD Thibault
Autonomous robotics student
ENSTA Bretagne

Under the guidance of

Cameron Baker
CEO of Envico Technologies

Envico Technologies
5 De Havilland Way, Mount Maunganui, New Zealand

Second year internship, May 2024 - August 2024

Nomenclature

A	Ampère
AI	Artificial Intelligence
BW	Bandwidth
CEO	Chief Executive Officer
COO	Chief Operating Officer
CSS	Chirp Spread Spectrum
DB	Database
dBm	Decibel Milliwatts
EEPROM	Electrically Erasable Programmable Read-Only Memory
GDP	Gross Domestic Product
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit
IO	Input/Output
IoT	Internet of Things
IR	Infrared
LoRa	Long Range
MCU	Microcontroller Unit
NTP	Network Time Protocol
NZD	New Zealand Dollar

P Power

PCB Printed Circuit Board

RX Receiver

SCL Serial Clock

SDA Serial Data

SF Spreading Factor

SPIFFS Serial Peripheral Interface Flash File System

TX Transmitter

UART Universal Asynchronous Receiver/Transmitter

UAV Unmanned Aerial Vehicle

USB Universal Serial Bus

USD United States Dollar

V Volt

W Watt

Abstract

This report presents the development of an automated rodent detection system incorporating a multi-sensor approach to accurately identify rats while minimizing false positives from other animals. The system integrates two Infrared (IR) Sensor Modules, a temperature sensor, and an ESP32-CAM for visual verification. The IR Sensor Module, comprising an IR emitter and receiver, serves as the primary detection mechanism. To enhance accuracy, two IR modules are used in tandem to confirm the presence of an object, while the MLX90614 temperature sensor distinguishes between warm-blooded rats and cold-blooded reptiles, such as snakes. The data from these sensors are processed by an Arduino Nano, which sends a signal to the ESP32-CAM. The ESP32-CAM captures and stores images for manual or AI-based verification during the trial phase. The choice of these sensors is justified by their affordability and sufficient accuracy for initial testing. Additionally, a Real-Time Clock (DS3231) is employed to timestamp detections, supporting integration with a database for comprehensive data management. This multi-faceted approach aims to create a robust detection system capable of effectively distinguishing rats from other animals, with considerations for cost and accuracy. Furthermore, a graphical user interface (GUI) was designed using Qt Designer with Python, allowing for user-friendly interaction and visualization of detection data. The final system leverages these components to achieve robust rodent detection, communication, and data management, aligning with the project's goals.

Résumé

Ce rapport présente le développement d'un système automatisé de détection de rongeurs, intégrant une approche multi-capteurs pour identifier avec précision les rats tout en minimisant les faux positifs provenant d'autres animaux. Le système intègre deux modules de capteurs infrarouges (IR), un capteur de température et un ESP32-CAM pour la vérification visuelle. Le module de capteur IR, comprenant un émetteur et un récepteur IR, sert de mécanisme principal de détection. Pour améliorer la précision, deux modules IR sont utilisés en tandem pour confirmer la présence d'un objet, tandis que le capteur de température MLX90614 distingue les rats à sang chaud des reptiles à sang froid, comme les serpents. Les données de ces capteurs sont traitées par un Arduino Nano, qui envoie un signal à l'ESP32-CAM. L'ESP32-CAM capture et stocke des images pour une vérification manuelle ou basée sur l'IA durant la phase d'essai. Le choix de ces capteurs est justifié par leur coût abordable et leur précision suffisante pour les tests initiaux. De plus, une horloge temps réel (DS3231) est employée pour horodater les détections, soutenant l'intégration avec une base de données pour une gestion complète des données. Cette approche multifacette vise à créer un système de détection robuste capable de distinguer efficacement les rats des autres animaux, tout en prenant en compte le coût et la précision. De plus, une interface graphique (GUI) a été conçue à l'aide de Qt Designer avec Python, permettant une interaction conviviale et une visualisation des données de détection. Le système final exploite ces composants pour atteindre une détection de rongeurs, une communication et une gestion des données, en accord avec les objectifs du projet.

Contents

Chapter 1

Introduction

The sugarcan is a critical issue for Guatemala. In 2018, 10.7% of Guatemala's GPD was due to agriculture. The same year, 1/3 of the country's exports are agricultural [1]. In 2020/2021 Guatemala was the 9^{th} sugarcane producer in the world and the 5^{th} largest exporter [2]. Rats are a considerable burden for farmers, when a field is infested with a large amount of rodents, the latter are able to make huge damages to the crop. In the 1999 - 2000 harvest seasons, it has been estimated that in Australia rats have destroyed approximately 825,000 tonnes of sugar-cane worth 50 million USD [3]. In 2003, in Pakistan, it has been estimated that the field's performance reduce by at least $7,5 \pm 1,5\%$ when infested by rodents [4]. Unfortunately for them, farmers in Guatemala also have to cope with the rodents. The specie known to damage the crops in latin America is the Sigmodon hispidus also known as cotton rats. The specie is very prolific, several litters may be produced annually, ranging from 2 to 15 young per litter. One rat only need to be 2 or 3 months old to breed. If nothing is done the growing of the rats population is exponential and so are the damage to the sugar can [5]. My internship had one goal, designing a device from scratch which is able to detect rats autonomously in a rural area. Currently, in the fields what are used are basic traps to capture rats which require humans to check whether or not there is a rat. If a threshold of captured rats is reached it means there are too many rats in that specific field and baits needs to be spread, currently the spreading of toxins baits is made by humans, by hand. The traps are checked only once every two weeks, that leaves plenty of time to the rats to damage the fields. During my internship I had to start thinking an autonomous device able to detect rats 24/7, it requires me to think about the sensors, the softwares, the future design of such a device. How to use it in an environment where both cellular connection and WiFi are unavailable, but still we need to communicate with the device. Such a device would be able

to have sereval advantages, first it would be able to work 24/7 so it would be way more reactive than the current method which requires humans. Second the goal is to combine the device with a drone which would take off instantly when a defined threshold of rats detected is reached to spread baits, one drone is able to spread baits in a large area, for exemple the current drones of the company are able to spread baits over one hectare every 7 minutes. It allows them to cut the costs but also to take most of the humans out of the equation which in the case of toxins is desirable. My report deals with the solutions I have thought would be a good idea but may not be the better one and the solutions I finally think are the better but are they really usable in the context of a sugarcan field ?

Figure 1.1: Cotton rats aka sugar can nightmare

The brief of my internship given by my internship supervisor is : Automated pest control - for landscape scale control of rodents in agricultural settings. The brief is to integrate a near live ground based monitoring and and control tool that provides an animal density count to the end user via an interactive user interface. The monitoring tool will interact with a automated drone bait delivery system which will deploy aerial baits in the desired location before returning to the recharging station to await the next deployment. The intended scale is 250,000ha The initial trial will compose 100ha The ground based monitoring device data will be interpreted manually until a digital link is developed. The drone baiting will be operated manually until a digital command system is developed. The scale will then be field tested and presented to a potential end user.
This report is for both the school and the company, I've tried my best to

ensure that the needs of both parties are met.

Chapter 2

Work Done

2.1 Sensors

2.1.1 Conditions to choose

One of the main goal I was given by the company was to choose the good sensors for the project. Choosing the sensors was subject to some conditions. Firstly, the price, the budget for the project is not unlimited and the device I have to design is only a part of it, I need to be really careful about the price of each sensor since the final goal is to have several device per hectare for a limited budget so if one cost a lot it may not fit the budget. Secondly, the size, our goal is to have a device that does not interfere with farmer's work plus does not afraid the rats since the goal is to detect them if the device's too big we may scare the rats and we won't detect any of them. Thirdly, the accuracy, having cheap sensors is a thing but if we can't trust them does it really make sense ? Indeed the goal at the end is to reduce the work of humans as much as possible, if we can't trust the sensors we will need humans to be sure they are accurate which is contrary to what we try to achieve in this project. Fourthly, what about the availability of the sensors ? Indeed at the age of the Internet of Things (IoT), some sensors become gold, some PCB such as some Raspberry Pi are really becoming rare. With time the company aims to multiply the numbers of these device across thousands of hectares, if we can't produce them we'll face a problem. These criteria have evolved in the course of my project, we'll see about that in the next subsection.

2.1.2 My choices (Node)

The following subsection are the sensors I decided to choose for the node. Indeed my device has several part, we have the nodes and the gateways. The nodes are supposed to be in the fields and to send a signal when a rat is detected, one node is linked to one gateway. The gateways are supposed to be outside the fields, ideally with an internet connection. One gateway can be connected to plenty of nodes, we just need an ID for the node to be identified.

Thermal Camera

When I first arrived and when my internship supervisor explained me the project I thought about a thermal camera. Indeed Envico already used thermal camera for others projects such as aerial mapping. The camera they use for this cost 6.000 USD, it has a very good resolution but cost way too much for my device. I started to look for my own camera which would cost way less, I started to find something interesting with the Flir Lepton module. The one I was looking for was the FLIR Lepton 3, combined with the DroneThermal v4 module it has a cost of approximately 600 NZD ($\approx$ 340€). Of course the resolution of this camera is not as good as the 6000 USD camera but would have been enough for the project. Finally after one meeting with my tutor supervisor and some colleagues we agreed that a thermal camera was not the right sensor to use, firstly the price is too high, if we want to multiply the device we can't have one sensor which cost 600 NZD on its own. Secondly, a thermal camera is certainly not the best sensor. In the sugarcan field there should be too many things in front of the camera and it would reduce it's efficiency a lot. Also, why I first thought of a thermal camera was because I wanted to combine it with a trained AI algorithm able to detect rats, the problem of such an algorithm is the power it requires, we would not have been able to implement such an algorithm on a low power MCU, that was not suitable for the final hardware architecture I thought of. Finally, the fact the project was set in Guatemala also played a role. Indeed, in the field there is no WiFi which is not surprising but there is no cellular connection too, so we need to rely on another way to communicate, for this I decided to use LoRa technology (LoRa stands for Long Range). We will see the specificity of this technology latter, but because we use it, it is another argument why using a thermal camera is not the best idea. Knowing all of this, investing in a thermal camera would not have been that relevant for the device.

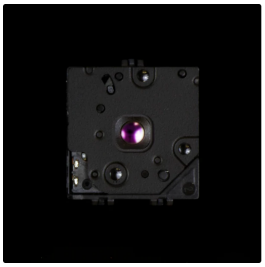

Figure 2.1: FLIR Lepton 3

LoRa Module

Before starting, I'd like to say a word about LoRa, Cocorico it's first been developed by two french in Grenoble before being acquired by Semtech, a Californian based company. Let's get back to the report. After realizing I couldn't rely on WiFi nor cellular communication (3G / 4G...). I started to look for another way to communicate. I started to read about LoRa technology and find it very interesting for the project. I found one paper which really helped me [6], this paper "A Remote Monitoring System for Rodent Infestation Based on LoRaWAN" published in 2023 in the magazine sensors is using the LoRa technology to do exactly what I'm trying to, detecting rodents and communicate the results using LoRa module. The strenght of this technology are numerous. The price, the maximum lenght coverage and the energy consumption. To communicate with LoRa you need one module which plays the transmitter role and one other which plays the receiver role. The communication is via radio waves and we have the choice of 3 different ranges of frequency which depends on the continent. For EU one can use 433MHz or 868MHz, for US 915MHz and for Asia 430MHz. When we need to find what band to use, we can refer to this website. Even if it is not an official one it gives us a good idea of what we can use or not. For Guatemala nothing is written, we should be able to use any of the frequencies quoted earlier. How is LoRa most likely to be used ? Usually there is one LoRa module per node and each node is refering to one gateway. But one gateway can cover an area of hundreds of kilometers. Just one LoRa module can transmit datas over few kilometers in dense area and even more than ten kilometers in a rural area which will be the case for the sugarcan field. So we will be able to cover hundreds of hectares with just one gateway. It raises the issue of multipath and fading, LoRa is based on chirp spread spectrum (CSS) which is highly resistant to multipath and fading [7]. One other key advantage of LoRa module is energy consumption, it is one of the way to communicate which uses the less energy. The only drawback of this

technology is the Data Rate, indeed when WiFi is able to transmit 5 Gigabit per second, LoRa can only transmit 250 Kilobit per second. It means WiFi is 20 000 times stronger than LoRa technology to transmit datas [6]. This is something I must have in mind at all time when designing the device, I won't be able to transmit large datas, this is also one of the reason why I decided not to use a thermal camera, transmitting a video would have been impossible using LoRa module. After knowing this, I had to choose which LoRa Module to use.

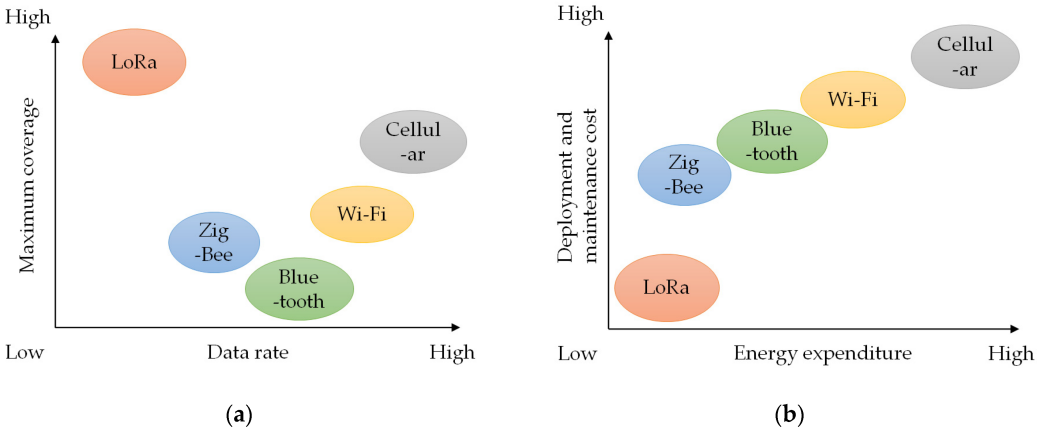

Figure 2.2: Comparison of various communication technologies in terms of: (a) maximum coverage, data rate; (b) energy expenditure and operationnal cost [6].

In the paper [6], they use the LoRa module SX1278. I didn't want to choose the same blindly, I wanted to know why they chose this particular one. All the coming information are from the LoRa module SX1276/77/78/79 datasheet [8] . I immediatly focused on SX1276, SX1278 and SX1279, because of their max sensitivity. Indeed it is said that max sensitivity equals -148 dBm, when the other module are -130 dBm. For this characteristics the lower the better, indeed a low max sensitivity will able the module to detect weak signals. It means it will allow us to increase the range between the gateway and the node or to reduce the strength of our signals to consume less energy. I had to make a choice, they had the same maximal TX Power of +20 dBm (TX Power is the capacity to transmit datas), the same RX current of 11mA, the same link budget of 168 dBm. The link budget is an accounting of all the gains and losses in a transmission system. It can be expressed as

$$P_{RX} = P_{TX} + G_{SYSTEM} - L_{SYSTEM} - L_{SYSTEM} - L_{CHANNEL} - M(dBm)$$

$$\begin{aligned} \text{where} \quad P_{RX} &= \text{received power}, \\ P_{TX} &= \text{transmitted power}, \\ G_{SYSTEM} &= \text{system gain (directionnal antennas..)}, \\ L_{SYSTEM} &= \text{system loss (feed-lines, antennas ..)}, \\ L_{CHANNEL} &= \text{channel loss}, \\ M &= \text{fading margin}, [7]. \end{aligned}$$

Among all communication system, LoRa is one with the better link budget. The data rate of theses modules is also the same we can compute it with this formula :

$$R_b = SF * \frac{BW}{2^{SF}}$$

$$\begin{aligned} \text{where} \quad R_b &= \text{Bit Rate (bit/sec)}, \\ SF &= \text{Spreading Factor}, \\ BW &= \text{BandWidth}, [6]. \end{aligned}$$

The only notable difference between theses 3 modules is their frequency range, the SX1276 ranges from 137MHz to 1020MHz, the SX1278 from 137 MHz to 525 MHz and the SX1279 is optimal for regions which requires the use of the 915MHz band. Because I had to make a choice I chose to use SX1278 such as in the paper cited earlier, when I checked for the prices it was also the cheaper (only a few NZD difference). But it was an error because according to the website I linked earlier the frequency plan I should use for my test in New Zealand is 915 - 928 MHz so I should work with SX1276 or SX1279.

Figure 2.3: LoRa Module SX1278

The price for three of these module on AliExpress was 24 NZD ($\approx 13,5$€), one for each sensor node and one for the gateway I will make for my trials.

Infrared (IR) Sensor Module

As I said before, I won't use a thermal camera to detect rats. I needed so to find another way, some others sensors able to detect rats. My idea was to use infrared (IR) Sensor Module. They are very easy to connect to an arduino, cheap, small and available in enormous quantities.

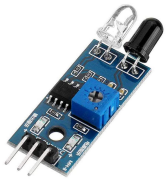

Figure 2.4: Infrared (IR) Sensor Module

On this sensor module, we have two leds, the black is an IR Receiver and the transparent an IR Emitter. 3 pins one for Vcc, one for Gnd and one dealing with the MCU to say whether or not there is something in front of the emitter. My idea is to have two of these sensor module which when they are both triggered is the first step to the detection of rats. One of the problem is, what about others animals ? We can adjust the distance the module is able to detect but what if another animal is close enough to be detected, what if a snake is close enough ? To answer this question we need a combination of sensors, and this module is only the first one.
The price for 10 of these sensor module was 7 NZD ($\approx$ 4€) on AliExpress.

Temperature sensor

In order to deal with the previous question I wanted to combine the Infrared Sensor Module with a temperature sensor. Indeed rats are a hot blood temperature animals whereas snake are a cold blood animals. By sending the signal when both IR sensor module are triggered plus the temperature of the body is in the range defined as the average temperature of rats body $\pm$ a few degrees. Will it be enough ? Surely not, but still, it is a step forward in the design of the hardware. My idea is that by combining all the sensors with a coherent structure design we will be able to detect almost only rats and not others animals. What temperature sensor did I choose ?
I decided to work with the MLX90614, this sensor is a infrared thermometer for non-contact temperature measurements. We can know the animal temperature without the latter touching the sensor. After choosing the sensor I

had to choose how to interface it with my MCU. I had two choices, combined the sensor myself to a SMBus as described in the sensor datasheet [9], or buying a thermometer module using this sensor. I decided to buy a module, the price difference was not significant and a part of the job was already done.

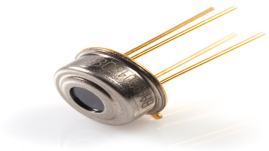

Figure 2.5: MLX90614

Figure 2.6: GY-206 : Thermometer module using MLX90614

The sensor is easy to interface with an arduino, indeed there exist a library which has been coded by Adafruit Industries which allows us to receive and process the datas coming from the sensor.

```
#include <Adafruit_MLX90614.h>
```

The module is also very easy to interface in the hardware because it's using I2C communication with SCL and SDA pins, SDA stands for Serial Data it is the wire used by slaves and masters to send and receive data, whereas SCL stands for Serial Clock, this wire carries the clock signal so that both the sensor and the MCU are using the same timeframe.

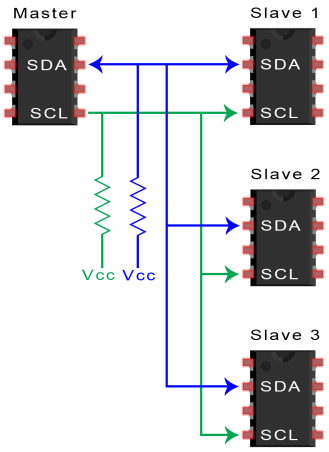

Figure 2.7: I2C protocol example with one master and three slaves.

So far, the advantages of the module are that is is inexpensive, small and available in large quantities. But what about its accuracy ? To check the accuracy of the module we have to check the accuracy of the sensor, MLX90614. Datas about the difference between the value transmitted by the sensor and one industry standard measuring instruments, reproduced from article Surayitno et al.(2019) [10] in the next table.

Name & Age	**Average body temperature (°C)**		**Deviation**
	MLX90614	**Thermometer**	
Fariz (18 years)	33.97	34.04	0.07
Rizal (22 years)	32.98	33.00	0.02
Dani (21 years)	33.61	33.74	0.13
Suparmi (46 years)	33.65	33.74	0.09
Sumarli (49 years)	34.53	34.68	0.15

Table 2.1: Testing the accuracy of body temperature sensors (MLX90614) against temperature measuring devices (Infrared Thermometer (IR60)) - Reproduced of "Measurement device for detecting oxygen saturation in blood, heart rate, and temperature of human body", par Suprayitno, E. A., Marlianto, M. R., & Mauliana, M. I. (2019), Journal of Physics: Conference Series, 1402, 033110.

Thanks to theses datas, we can compute that the percentage of accuracy is between 99.5% and 99.9%. It is coherent from what we can find in the sensor datasheet. All these arguments are the reason why I decided to use

this sensor and not another one. The price for two of these sensor in June 2024 was 17 NZD (≈ 9.5€).

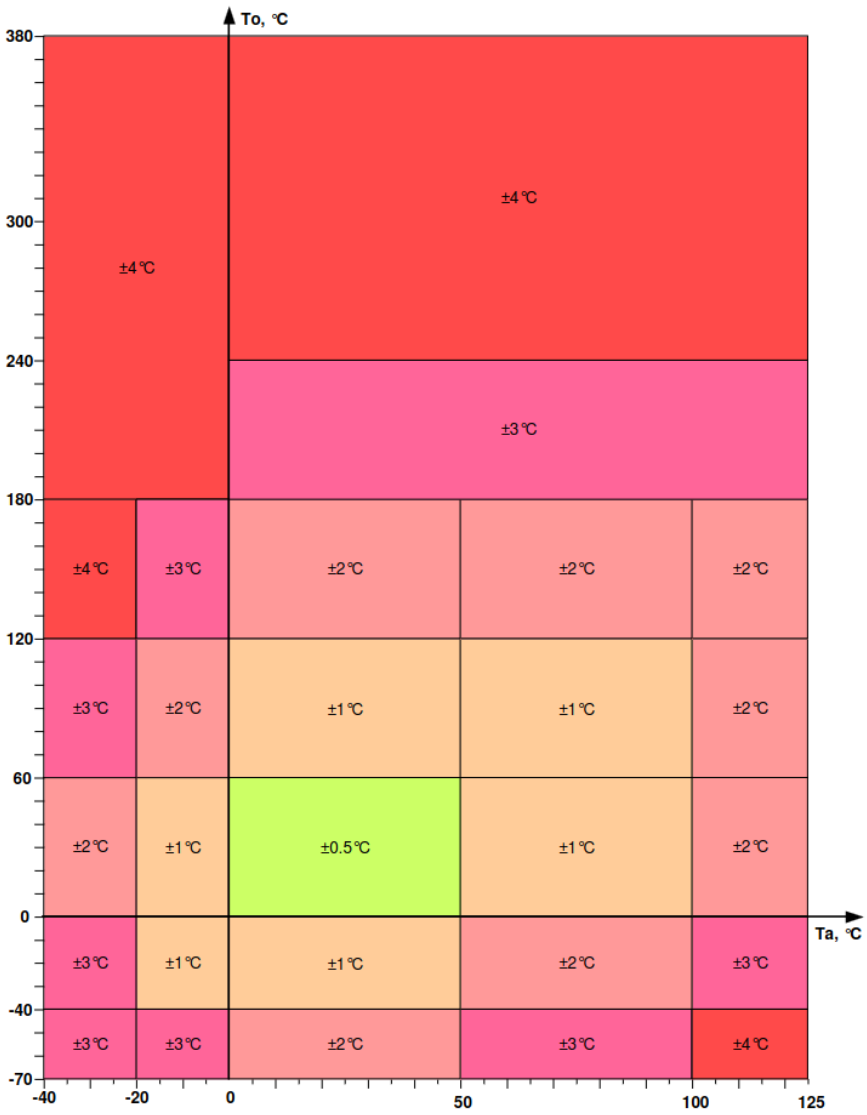

Figure 2.8: Accuracy of MLX90614 where Ta is Ambient temperature and To is Object temperature, image from [9].

ESP32 - CAM

Why using a camera when I said earlier I didn't want to use one. Firstly it is because of the nature of the camera. This time it's no more a thermal one. It allows us to reduce the price of the sensor and to increase the resolution

of the picture simultaneously. Secondly, my idea is to use this camera only for the trial stage. Indeed as we saw before the only way to communicate is with LoRa and transmitting photos may need a high bit rate we don't have. Furthermore if we have 100 nodes connected to the same gateway, the gateway will never be able to handle that much information. But for trial stage we still have human in the process, my idea is to use this camera to be sure we detect rats. How ? When both IR module are triggered and the temperature transmitted by the thermometer module is in range of the rats body temperature, we take one photo, store it to a SD card, later one can retrieve the SD card and check if one can see rats on each photo. Normally there should be only rats on the photo since we want the ESP32 - CAM to take one photo when all sensors are triggered simultaneously. We can check all the photos by hand, or with a trained AI algorithm able to detect rats. Why do we use a ESP32 - CAM ? Just like other sensors the price, one ESP32 - CAM is cheap that means we don't have an insane camera resolution but still we can have with the OV2640 camera a max resolution of 1600x1200 pixels (2MP). That's more than enough for our purposes.

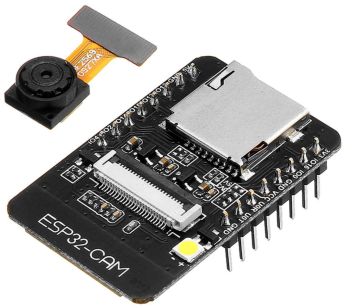

Figure 2.9: ESP-32 Cam board and the OV2640.

This board has another advantage, it has its own microcontroller unit (MCU) so it can process some code on its own, at some point it helped me debugging my code. The gray layer on the figure 3.8 is where we insert the SD card. This board can also communicate via UART protocol.

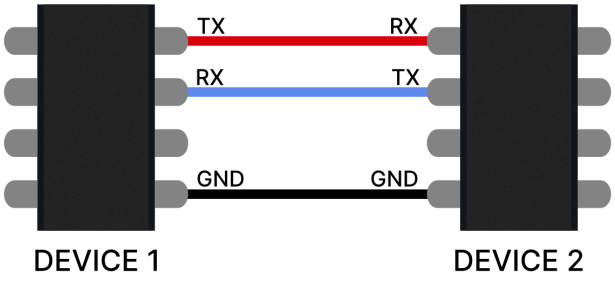

Figure 2.10: UART protocol

Via this protocol the camera receives the instructions to take a photo. I first struggled a lot using this camera, first with the code but also with the OV2640. It is a very fragile little camera I may have unfortunately killed the first I purchased with my different trials. At some point I was not able to interface my code with the camera anymore. I thought it was because of the code I had written (since it was a modification of an existing GitHub code) but it was not, when I tried with another ESP-32 board and OV2640 my code was working. It was a good lesson, I learnt I should always buy at least two of the same components because when one (a trainee doing some tests for example) accidentally destroyed one we have to wait for the next one to be shipped. To upload code on the ESP32 - CAM board it requires a little module (ESP32 - CAM MB) which connects to the computer via micro-USB. There exists other solutions but this one the simplest because the module, the board and the OV2640 are often sold together.

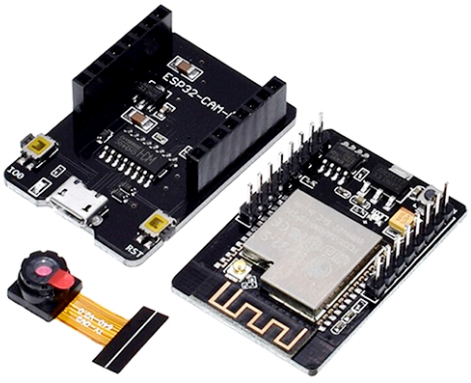

Figure 2.11: The avengers united, from left to rigth ESP32-CAM-MB, ESP32-CAM and OV2640.

For 4 ESP32-CAM board with the MB module and the OV2640 I paid a

total of 90 NZD (≈ 50€).

Real-Time Clock

At first I had not planned to have a Real-Time Clock (RTC), why adding this ? It will just increase the costs, I thought. I changed my mind after working on the DataBase (we'll deal with it a bit later in this report). To get a sense of time I decided that I will store the date with each detection of rats. I thought there was a way to do with the MCU I chose (next subsection) but the only way to do (at least the one I found) is to add a RTC. Then I started to look for the one I wanted. I chose the RTC DS3231.

Figure 2.12: Real-Time Clock DS3231

Once again let's wonder why this one ? Just like all others sensors this is a cheap module, small and available in large quantity. Plus the protocol to communicate is a protocol we already know, indeed it communicates via I2C (see Section 2.1.2). This clock has the advantage to have a backup battery, if our main power supply shuts off for some reason, the backup battery will keep the RTC turn on and we won't need to set the date of the RTC again (what we need to do the first time we use the RTC with the device). Let's take a concrete example, it is 05-12-2025 during 500ms the power supply stops providing power. All sensors stop working during this 500ms but then they start again, if we don't have the backup battery the RTC date will be the 1-1-1970 when it will restart. If we have this backup battery, it will be 05-12-2025 + 500ms when restarting. When looking for RTC I also found lot of different notices praising the precision of the clock. This is the main reasons why I chose this RTC.
The price for one RTC was 1.65NZD ($\approx 0,9$€) on AliExpress on June 2024.

Microcontroller Unit (MCU)

So far we have, our way to communicate LoRa, 2 Infrared sensor module, one thermometer module, one ESP32-CAM and one Real-Time Clock. As I said we have one MCU on the ESP32-CAM but we can't rely on this one for the project, there is not enough pins to connect everything and it's not powerful enough since it has not been designed for that. To choose the right MCU I needed to first think about how many pins I require. After I calculated this, my goal was to find the best MCU, what is the best MCU in this context ? The best MCU is the one which has enough pins, fits with the characteristics I listed for all others sensors (small, cheap, availability...). The one I decided to use is Arduino Nano. Indeed I managed to connect every sensors to it, the LoRa module, the two IR sensor module, the thermometer module, the ESP32-CAM and the DS3231. In order to have a better representation of what it should looks with all the wires connected I designed a schematic using KiCad v8 (during the year we used a software named Altium for PCB design but unfortunately it's unavailable on Linux so I decided to use KiCad, it is free with a big community it makes it easy to learn and use).

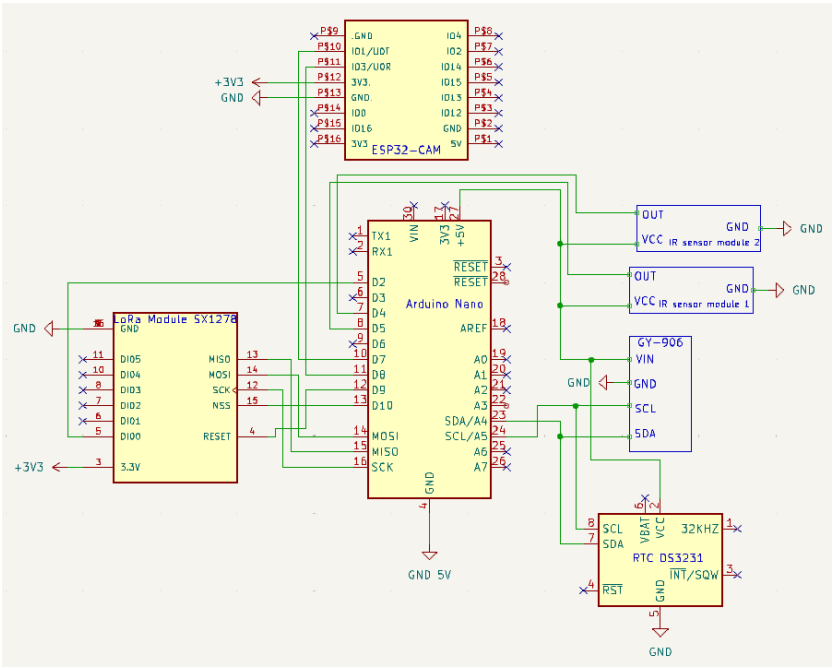

Figure 2.13: Schematic of my hardware.

This schematic allows us to check whether or not an Arduino Nano can fit for the hardware. Did it successfully pass the test of the pins ? Absolutely.

If an Arduino Nano fits for the job, of course others MCU of Arduino bigger such as Mega or Uno would also fit, but they would require more energy to work and cost more. One Arduino that is very interesting for the IoT is the Arduino Nano ESP32, why ? It has a chip allowing the communication by both WiFi and Blue-tooth. But as said before, we can't use WiFi for this project. Blue-tooth could have been interesting but the coverage distance is too short to be used here. For the same reasons, a commonly used MCU is useless here, it is the ESP32. It does the same job as an Arduino but can access WiFi and Blue-tooth.

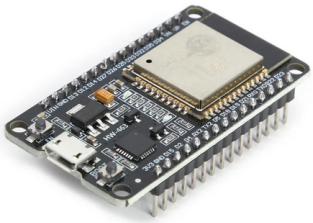

Figure 2.14: ESP32 board

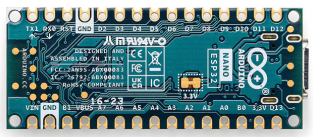

Figure 2.15: Nano ESP32 board

For 3 Arduino Nano I paid a total of 14NZD (≈8€) on AliExpress in June 2024.

Power consumption

Please note that for all next calculations the power consumption of the RTC has not been taken into account. Indeed according to its datasheet the maximum active supply current at 5V is 300μA = 0,3mA. [11]

I think it's time for me to sum up all the sensors we have for the moment. To do so, we will have a look at the next tables which contains all the sensors, associated price and power consumption (there are two tables instead of one, for display reasons).

	MLX90614	**IR Module (x2)**	**Arduino Nano**
Mode	Active	Active	Active
Power consumption (mA)	2	20	19
Voltage (V)	5	5	5
Power (mW)	10	100	95
Price (NZD per unit)	8.5	0.78	14

Table 2.2: Power consumption and price for chosen sensor part 1.

	LoRa Module SX1278			**ESP32-Cam**	
Mode	TX	RX	Standby	Flash-On	Flash-Off
Power consumption (mA)	93	12.15	1.6	310	180
Voltage (V)	3.3			3.3	
Power (mW)	307	41	6	1550	900
Price (NZD per unit)	8			22.5	

Table 2.3: Power consumption and price for chosen sensor part 2.

The figures for power consumption and voltage comes from the datasheet product for each sensors except for the IR Module where there is no datasheet, the figures come from several website indicating 20mA is the max input power. Every power (mW) figure has been over-estimating, for example power should be 5.28 for LoRa in standby mode I decided to write 6, this way it gives us a margin to choose a good power supply. Also when there is just one mode per sensor (active) the power consumption is at the maximal capacity of the sensor / MCU, it allows us to have an other margin.

It's now time to compute the required power asked by the device in the worst case. The worst case is when the device is transmitting datas, so TX mode for the LoRa Module and the flash is on.

$$P_{tot} = \sum_i P_i$$

With P_{tot} the total power required and P_i the power required by each sensor / MCU.
In the worst case we find that P_{tot} equals 2.17 W, again it is an overestimated figure (2.162 W is the exact power). Using the same method to find the price for one hardware of the device on June 2024 buying most of the compounds on AliExpress expect the ESP32-Cam which have been bought on Amazon. The total price (for the hardware) is 56,3 NZD ($\approx$ 31€). We have to keep in mind that for the ESP32-Cam is only for the trial stage, a priori we won't need it in the field so we will be able to lower the costs.

Power supply

Now that we have the energy consumption we need to figure out how we will find this energy. In the paper "A Remote Monitoring System for Rodent Infestation Based on LoRaWAN" from Shin-Chi Lai et al. [6], they use two 3.7 V 18,650 batteries in series, that gives them a 7.4 V power supply, thus they need two integrated circuit (IC) to convert the voltage to 5V and then to 3.3V. Having additional IC is not the problem here since they are really cheap (for example the one they use cost less than 1 NZD for each). But the batteries cost a lot, it costs 25 USD out of the 43 USD for the device, that means 58 % of the price is for the batteries. I'm not saying we won't use batteries for our device rather I prefer asking why the way they use them will be inefficient for our device ? First their device is not designed to work 24/7, which is the case for our device. Their device is designed to last 15 days before changing the batteries. This is something we want to avoid, changing the batteries means two things : human in the process and additional costs. Two things we don't want, we don't want to require humans every two weeks just to check on the device (this is currently what's happening in Guatemala with basic traps, farmers are checking each trap every two weeks) and we don't want to replace batteries every two weeks we want to cut the costs not increasing them, 2 batteries every two weeks during one year means 25 * 26 = 650 USD / year/ device. Even if we have batteries which cost less it would still be too much money just for one device. But we can use one of Guatemala's strengths, the daily sunshine. Indeed according to this website DonnéesMondiales.com, even during the worst month of the year the daily sunshine is about 11 hours, why not using solar panel to provide the energy during daylight and store it for for use at night ? Since our device is a low power consuming device it won't require a big solar panel, nor a big battery to store the energy. Furthermore the computed power of the device is made in the worst case, when we transmit datas and the flash is on, but they are rare event, only when we detect a rats most of the time the LoRa module is on standby and the flash is off which means the total power $P_{tot} = 1.2W$ the working power is almost divived by two, we can even go further, we won't have any cameras on the device when it is not in the trial stage, this means we will remove the biggest energy consumer, when we don't detect rats we will only need $P_{tot} = 0.31$ W. That's a lot of figures, let's sum it up in a table.

	Trials stage		Final device	
Mode	Rats detected	Standby	Rats detected	Standby
Power (W)	2.17	1.2	0.62	0.31

Table 2.4: Total energy consumption of the device in the different modes.

This is very important to store enough energy for the whole night, rats are more likely to be detected during the night. The next paragraph is just a supposition, it needs to be verified by experimentation.

Let say we detect 20 rats a day (it really is a big assumption it could be way less or more). When we detect a rat, we take a photo. The flash is on for one second, during that same second the LoRa module communicates that the device detected a rat. In total, 20 seconds during the day we are not in standby mode. This means that during 99.97 % of one full day we are in standby mode. The total average power becomes :

$$P_{tot}^{trial} = 99.97\% * 1.2 + 0.03\% * 2.17 = 1206mW$$

$$P_{tot}^{final} = 99.97\% * 0.31 + 0.03\% * 0.62 = 311,7mW$$

To conclude we need to be able to provide during the trial stage at least 1206 mW per hour to the device (remember we took some margins earlier) and 311,7 mW for the final device. Shouldn't be too hard to find solar panel able to provide this power for a reasonable price. After a very quick research on AliExpress I've found several solar panel ranging from 0.15W to 2W, 0.15W is not enough but 2W is almost 1W more than what we require, the prices are ranging from 1.5NZD to 12NZD ($\approx$ 0.8 to 6.5 €)

2.1.3 My choices (Gateway)

The next subsection deal with the sensors I chose the gateway.

LoRa Module

We need another LoRa Module to receive signal when a rat is detected by the nodes. I will not deal again with the specifications (see Section 2.1.2 LoRa Module). I choose the same as before the SX1278 module (once again knowing we may change it for the test for the SX1276). Congratulation, we have all our sensors.

MCU

For the gateway I didn't choose to work with arduino nano. It's not because it has a small number of pins, we've seen it is enough for a node which has 5x more sensors. I changed of MCU because I would like my gateway to be connected to internet and ESP32 (be careful not to confuse with ESP32 - CAM) is known to be a great MCU able to do so. It's better to use it only for the gateways because its main advantage over the arduino nano is the fact it can connect to internet, but in the field it's useless however this advantage has a cost, one ESP32 cost 7.5NZD ($\approx$ 4.1 €) on june 2024, 3 NZD more than arduino nano. That's not a lot but imagine if we have hundreds of nodes, costs add up and can make the project unprofitable.

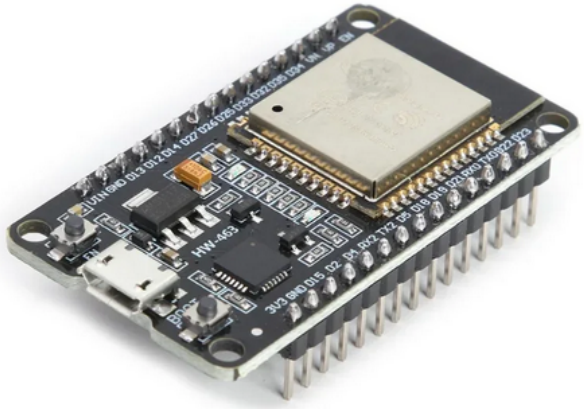

Figure 2.16: ESP32

This is all for the gateway, we don't need much. Just something able to receive the datas (LoRa Module) and to process it (ESP32). We don't even need a RTC, if we need to have any date or time we can make a request to a NTP server (NTP stands for Network Time Protocol) to get it. Or even easier, we can install the ESP32Time library by fbiego and there exists everything we need to access to the ESP32 internal RTC, almost too easy. Once again I designed a schematic using KiCad.

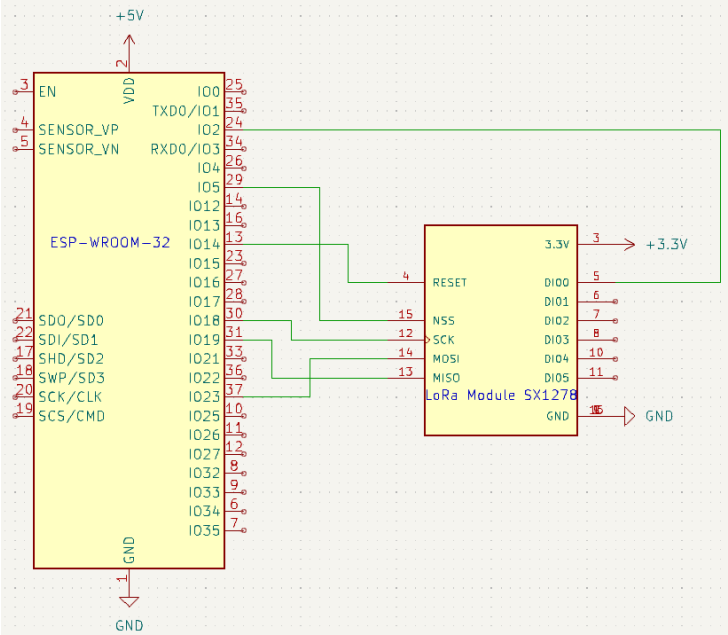

Figure 2.17: Schematic of gateways hardware

To be fully honest I may have confused between the IO numbers and the numbers in red. It's very easy to find reliable schematic on internet. For example I used this website.

2.2 Software

After choosing the best hardware for the device, now is the time to think about the software. To be totally honest I didn't first choose the hardware then the software, I was thinking about both at the same time. In my opinion it's pointless to think of a good hardware if we can't interface with any code and to think of a good code if we won't be able to have to better hardware to run that code.

The software part is divided in three distinct part. We have everything related to the sensors this is mostly arduino, coded in C. Then we have all the code related to the DataBase for this I used SQLAlchemy this is a python module which allows us to translate our code into SQL. I chose to work with this module instead of using SQL directly because I needed to implement some logic between my sensors, the GUI (Graphical User Interface) and the DB (DataBase), in my opinion it was the simplest way to do it. Finally we have the GUI, which is coded using python and the framework Qt Designer.

2.2.1 DataBase

In this subsection I will describe how my DataBase (DB) is working. We have 4 elements, the nodes, the farms, the gateways and the alerts. Each object is identified by its ID which is its own primary key. It can be anything a number, a name (John's farm, Gateway 2, Alert 50 ...). When a rat is detected, a signal is sent by the node to the gateway, with the date and the ID of the node. This way we can know for each node how many rats we detected at what time. Which is our goal, indeed the second part of the project is to use a drone when a threshold of rats is detected, if for example we define this threshold for 15 for John's Farm, just using a query about how many rats have been detected by all the nodes related to John's farm we could know whether or not this threshold has been reached. In my DB, the nodes are not directly connected to the gateway, they are connected to the farms and the farms are connected to the gateway. They are all connected by foreign key, to access how many rats have been detected in a specific farm at a specific date we can join all bases like in the following SQL query :

```
SELECT alert.id AS alert_id, alert.date AS alert_date,
alert.node_id AS alert_node_id
FROM alert JOIN node ON node.id = alert.node_id JOIN farm
ON farm.id = node.farm_id
WHERE alert.date = 2000-12-05 AND farm.id = John's Farm
```

Counting the number of rats detected since a specific day, taking inventory of all the detections and sorting them by farm is the main goal of this DB.

Let's sum up how it works. We have in the reality one farm. For this farm, according to its distance to the different gateways we decide that we will connect all the nodes of this farm to one gateway and not to the others. Thus all installed nodes in this farm will send the alert to this specific gateway and not to the other. For each farm we can define a threshold of rats, for the moment it is randomly set since it's a number that must results from a discussion with the customer.

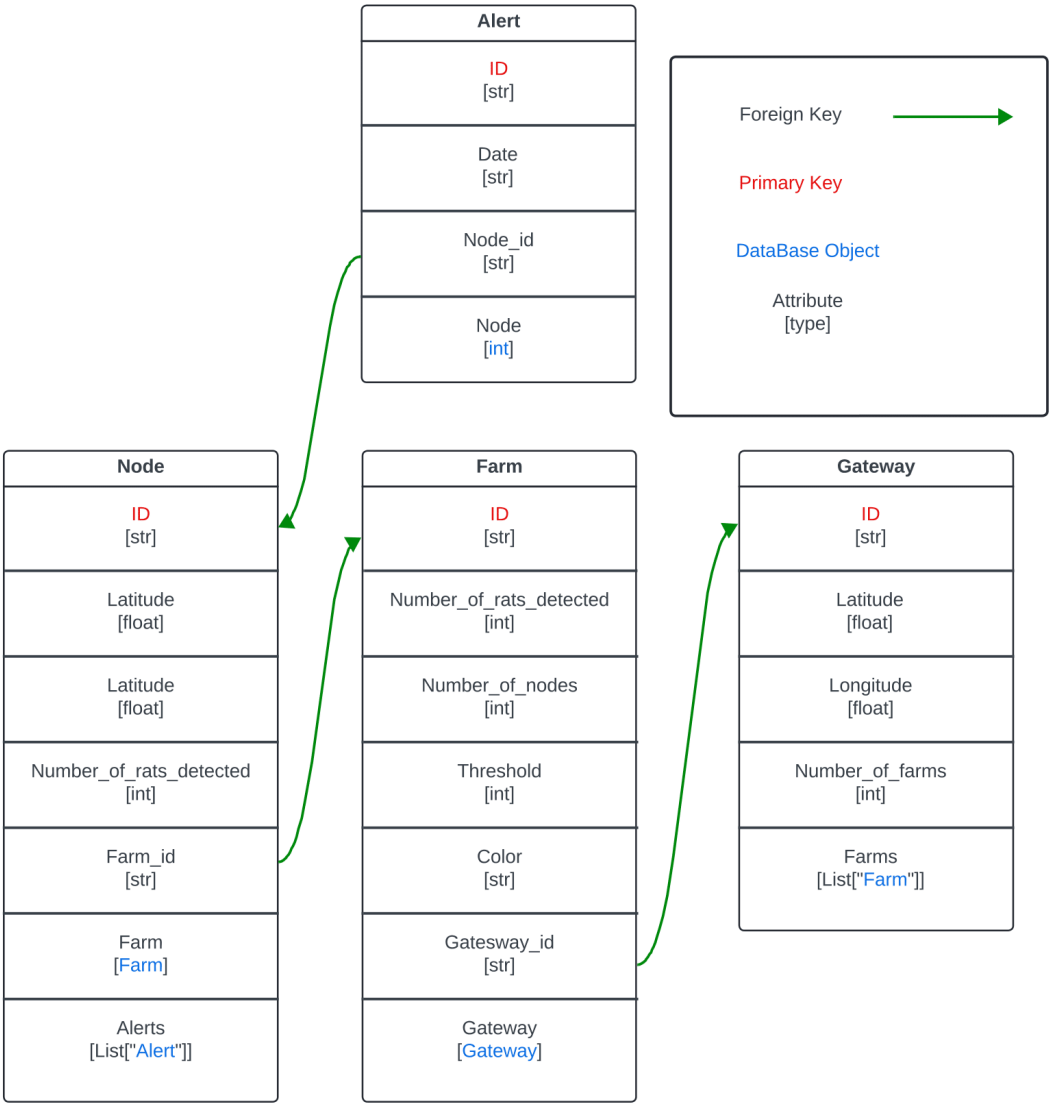

Figure 2.18: DataBase

2.2.2 Graphical User Interface

In the brief I was given, the goal was to provide to the customer an interactive user interface for him to monitor what's happening in the field. For the first stage in Guatemala, we will have only one person dealing with the drone. Indeed when a threshold will be reached in one farm, the person will go with the drone to spread the baits. This explains some of my choices in the GUI. To show you my interface I prefered to make a video to explain the most we can do with the GUI . This is not intented to be the final GUI but the idea was to show what we can do with the project and how I think it could work. You can find the video by clicking here. You won't find the code in the appendice since it's composed of a bit less than 2 000 lines of codes splited in 14 scripts.

2.2.3 Arduino and Python

Let's have a quick look at what it looks.

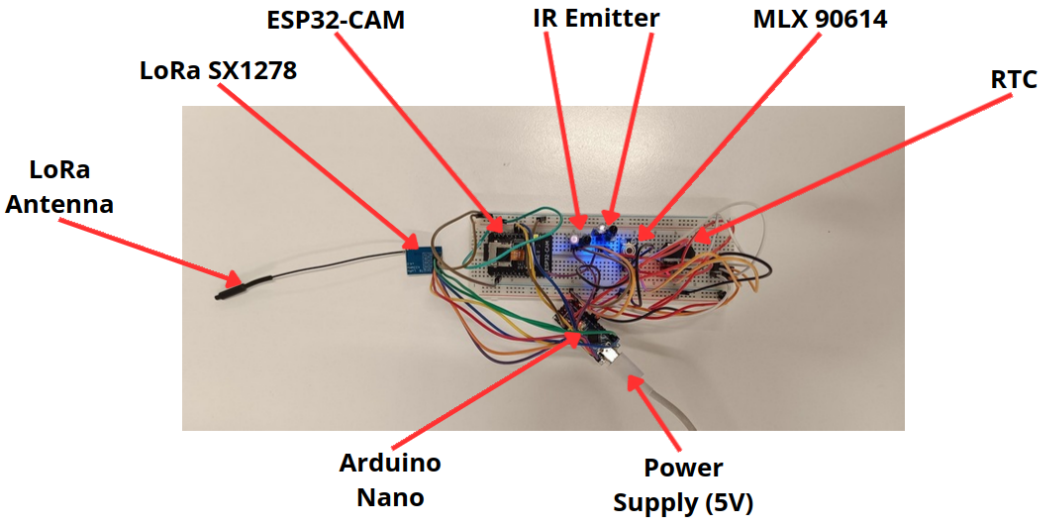

Figure 2.19: Node hardware

Here you have the hardware of one node, with all sensors. I advise using an other power supply for the ESP32-CAM since it requires lot of power and Arduino Nano might have troubles providing that evergy when the flash is on + LoRa is transmitting datas. Warning, LoRa module 3.3V pin has to be connected to the Arduino Nano otherwise it will never be initialized (lost couple of hours trouble shooting that alone problem).

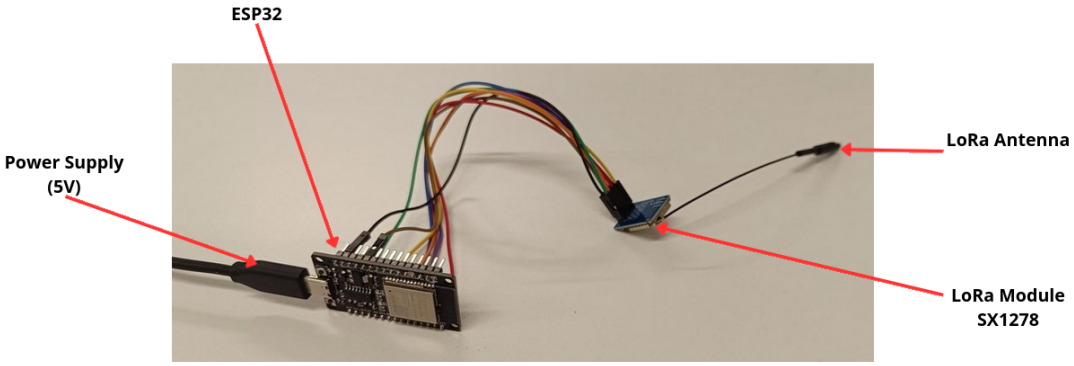

Figure 2.20: Gateway Hardware

And here is the gateways hardware, very simple as we've seen before. Let's talk about how it works and how I interfaced it with code. Let's begin with the node. You can find the full code in the first appendice. First we need to include each library linked with the sensors. We use the library SoftwareSerial to create a communication between the ESP32-CAM and the Arduino. Indeed Arduino Nano has only one Serial which is occupied by the USB Cable when connected to the computer, this Serial is the standard Arduino Serial. Since the idea for the trial is to have an ESP32-CAM I need to communicate to the cam when to take a photo. This is what my CamSerial is made for, by using this we create a virtual serial which sends command to the ESP32-CAM and allows us to trigger the camera. You can find the code which is on the ESP32-CAM MCU in the second appendice. The code provides from the github repository of Emre Şahin, I just modify what I needed for the ESP32-CAM to take a photo when it receives a specific command from the arduino. It raises a question, when does the ESP32-CAM take a photo ? This is the next condition.

```
    if (ir1_status == 0 && ir2_status == 0 &&
       inRange(temp)){..}
```

This basically means, if IR Emitter 1 AND IR Emitter 2 are triggered AND the temperature perceived by MLX90614 is in the defined range (for my test I chose the range to be [15,45]°C because it was easier, but we can of course defined the range for it to fit the rats body temperature range) then we send a command thanks to our CamSerial.

```
    CamSerial.println("TAKE_PHOTO");
```

When received by the ESP32-CAM MCU, flash is turned on and the camera takes a photo which is stored in a SD Card. During the same time,

under the same conditions, a message is sent to the gateway thanks to the LoRa Module.

```
LoRa . beginPacket () ;
LoRa . print ( " Node 1 ; " ) ;
LoRa . print ( date + " ; " ) ;
LoRa . print ( counter ) ;
LoRa . endPacket () ;
```

All signals need to be standardized, this way they will be easier to read. I decided to write it like that, nodeID + date + counter, the counter is the number of rats detected by this node when the signal is sent. To get the date we use the RTC library and the following function :

```
String getDate(){..}
```

It returns the date in the format "YYYY-MM-DD-HH-MM-SS", we basically just need "YYYY-MM-DD" for the device, but for my trials it was important for me to also have "HH-MM-SS" to know when we received the signal since they are sent the same day. The code works perfectly, it took me a long time to figure out how to make all sensors work together but it allows me to learn a lot, whether it is on the sensor side / hardware , Arduino, interfacing everything together... The Flowchart sums up the code.

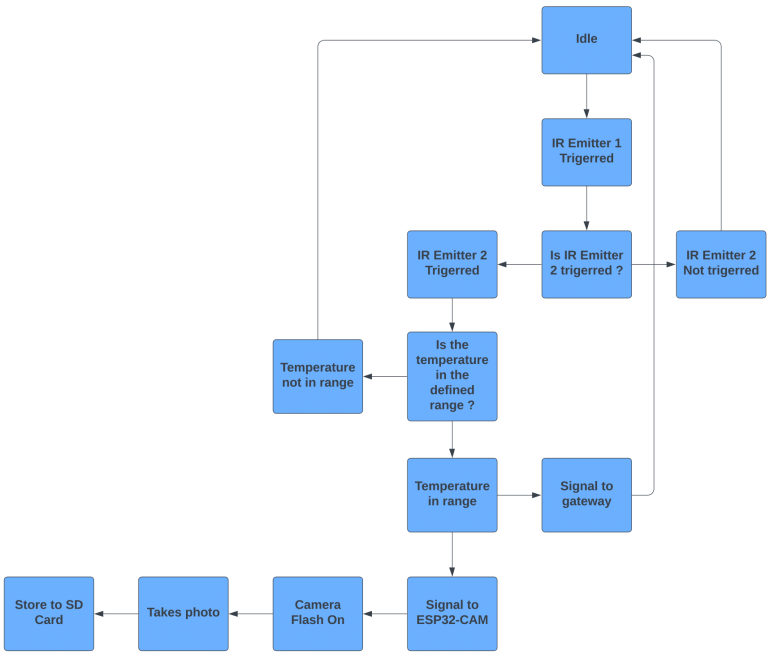

Figure 2.21: Flowchart of nodes.

In the third appendice you'll find the code related to the gateway module. Once again we need to first initialize everything, Serial, LoRa module ... I had to find a way to store the detections without having access to WiFi since it's not 100 % sure we will have access to WiFi at all moment (that would be ideal..). To do so I had to find a way to access the memory of ESP32. It's not as easy as accessing the memory of your computer, it's not designed for it. Developpers have develloped few ways to do so, we can access the EEPROM (Electrically Erasable Programmable Read-Only Memory), use the preferences library designed especially for ESP32 or use SPIFFS (SPI Flash File System). EEPROM doesn't allow us to store large amount of datas and have a limit of 100 000 cycles lifespan, which means after writing 100 000 times over the same cell she will die and this memory won't be available anymore. 100 000 cycles seems a lot but it is very easy to reach if we have hundreds of nodes connected to the same gateway. I chose preferences over SPIFFS, preferences stores datas in a system of {key : value}, that is exactly what we need. Whereas SPIFFS is made to store huge amout of datas in structured files. We don't need such a tree structure, as we say in French it would have been like using a hammer to kill a fly, we just need one file which store the value of the different detection, nodeID, date and counter. Preferences allows it, there are few implemented functions just what we need to write all kind of data type (int, float, char ...) and functions to read theses datas. The next line helps us synchronizing the LoRa Module, indeed by specifying a particular entry the LoRa module will only listen to the LoRa transceiver with the same entry. By specifying it in each node related to the gateway, we are sure the good gateway will listen to the nodes we want the gateway to listen.

```
LoRa.setSyncWord(0xA5);
```

There are two main functions :

```
void storeRatDetection(String date, int count){..};
void printRatDetection(){};
```

The first function allows us to store each rat detection in the ESP32 memory once the gateway receives a detection signal. When looking at this code you'll see the nodeID is specified and not sending by the node itself, we can change it very easily and add it as an other argument to the storeRatDetection function. Anyway it works, I did it this way since I did all my trials with only one entire node module (at some point I tried to connect 2 LoRa transmitter to the same LoRa receiver, it's been working perfectly, the LoRa receiver was receiving each message and the information contained in these messages). Now that we know how to store datas in ESP32 memory we need

to be able to retrieve this data. As I said before the preference library is pleasant to use because the implemented function are easy to use and they go straight to the point. For exemple if we want to both store an integer and retrieve an integer from the memory we can use the two following functions.

```
preferences.putInt();
preferences.getInt();
```

But theses functions are running on the ESP32, and we don't have our Database on the ESP32. This is our goal, retrieve the data from the ESP32 memory and put it in the Database. If we have access to WiFi , we can use PHP to send data to an online server where we have our DB, this solution is preferrable if we are sure we will have a good access to internet. In the case we don't, I tried to retrieve the data just by connecting the ESP32 to my computer. By using two python module I managed to do it.

```
import pyserial
```

```
import sqlalchemy
```

The first one pyserial allows us to interface between arduino and python. You'll find in the appentice the code I used to access the data, read it, write it to a text file and then using sqlalchemy to add the detection to my Database. With this code, we do not need to access WiFi, we can run everything locally. Once we have transfered all the data to the Database we need to make sure that we overwrite the ESP32 memory. Indeed if we don't, the next time we will transfer the data we will have duplicates of the same alert. To do so, the preference library has a function.

```
preferences.clear();
```

When we call this function, all the pairs {key : value} are deleted but the namespace is not (the namespace is the "file" where we store our data). We can easily imagine adding a pushbutton to our gateway which triggers the function when pushing.

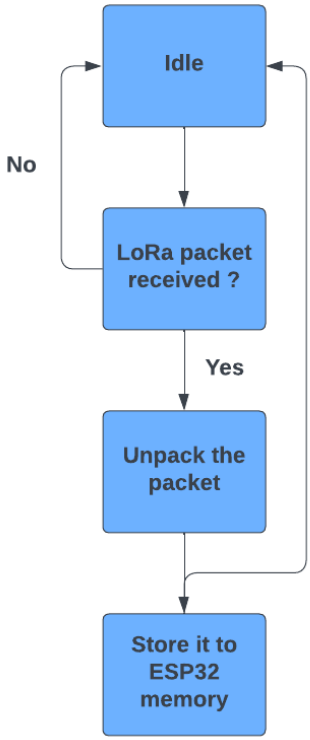

Figure 2.22: Flowchart of the gateway.

If you have a quick look at the code you'll see a variable which name is RSSI, RSSI stands for Received Signal Strenght Indicator. The unit of RSSI is dBm, basically let say the higher the better.

RSSI	Signal Strength
> -70 dBm	Excellent
-70 dBm to -85 dBm	Good
-86 dBm to -100 dBm	Fair
< -100 dBm	Poor
-110 dBm	No signal

Figure 2.23: RSSI order of magnitude.

For example when both of LoRa are 1 meter away, the RSSI ranges from -25 to -35 dBm which is an excellent signal strenght if we consider dBm to

be a logarithmic scale.

We now have everything related to the nodes and the gateways, from the hardware to the software.

2.3 Fieldays

2.3.1 What are Fieldays ?

Fieldays are the equivalent for "Salon de l'agriculture". It is a place where during 4 days between June 12 and 15 in 2024, companies become exhibitors, they show what they achieve and what kind of goods and services they sell or are about to bring to market. It is not only for farmers, for example Envico is not directly a company dealing with agriculture but the company projects can perfectly fit for use in an agricultural applications. At first I wasn't supposed to go but since one my colleague Joseph couldn't make it, Andrew the chief operator officer (COO) of the company asked for my help.

2.3.2 What was my job ?

At Fieldays we had two stands, I was exhibiting with Andrew at the Seedpod stand in the forestry hub. What are seedpods ? Envico is a company using UAVs for environmental purposes, as said before they can use UAVs for aerial mapping, but not only. One of the projects in progress is seedpods. Seedpods are little balls of dirts with seeds of New Zealand native trees for most of the balls. It is not only dirt, it is in reality a mixture of fertilizer, nutrients, dirt and seeds. The goal of the company is to spread them using UAVs, indeed using drones are a game-changer. One drone can spread seedpods over an area of one hectare each seven minutes, impressive isn't it? My job at Fieldays was to describe and to advertise the company projects but above all seedpods.
That was a pleasant experience, it allows me to practice my English a lot. I really noticed a shift before and after Fieldays, being surrounded all day long by people speaking English and being an exhibitor obliged me to speak a lot more than in everyday life. I met a lot of people from different companies, when we had time we were pleased to explain to each others our different projects that was very interesting.

Figure 2.24: Our stand in the forestry hub.

Being an exhibitor at Fieldays has been an amazing experience for me, it was very different from the other things I had to do during my internship. It brought me out of my comfort zone because I had to have the best communication with sometimes a stammering English to describe the best as possible the company projects. I really did my best in a role which was not easy at first.

2.3.3 Bonus

I would like to thank Cameron, my intern supervisor and also CEO of the company for the amount of food we ate this week, I will remember this Korean fried chicken for a long time. I will also have a very good memory of the AirBnb where I discovered a passion for making fire after a long day of work. Last but not least, thank you for the Greenstone Necklace.

Figure 2.25: Toki, Greenstone Necklace. Part of the Maori culture.

Chapter 3

What's coming next ?

3.1 More sensors ?

I've thought about it too late, but it could have been wise to add a load cell. Indeed if we calibrate it to send a signal when it perceives a weight which in the range of the rat body weight it could add an additionnal check. The transmitter signal condition would become :

```
if (ir1_status == 0 && ir2_status == 0 &&
    inRangeTemp(temp) && inRangeWeight(weight){..}
```

At the exact moment I'm writing this report I don't have any other sensor I find relevant for the device.

3.2 LoRa settings

Do you remember this equation ?

$$R_b = SF * \frac{BW}{2^{SF}}$$

I did not during my work change any of this settings, the spreading factor (SF, defines the number of bits that can be encoded by a symbol: LoRa Documentation) can be very easily changed, indeed the LoRa Arduino library has a function which allows us to change this setting :

```
LoRa.setSpreadingFactor(7);
```

I did not change any of this because I didn't not need since my two LoRa modules where less than 1meter away for my trials. But it could be interesting to do so, for maximum SF is SF12, the higher the SF is the lower the number of bits we can encode will be, but since the information sent by

the node only are the date, the counter we really don't need to encode much. Plus, the lower we encode the less power we consume.

We also can work on different BandWidth (BW), indeed having a tight BandWidth let say 125kHz allows us to send data over longer distance by increasing the sensibility of the receiver. It allows the receiver to perceive lower signal, whose power would have been challenged by the distance. A higher BW allows the transmitter to send larger amount of datas, a BW between 250 / 500 kHz will be able to send more datas than if we work with 125kHz, of course increasing the BW will also increase the power consumption.

```
LoRa.setSignalBandwidth(125E3);
```

The TX power can also be changed using the next funtion :

```
LoRa.setTxPower(17);
```

We can choose a value between 2 and 20 dBm. Once again, a stronger TX Power will allow to cover longer distance but will cost a higher energy consumption.

Finally we have one more lever to increase the distance between the nodes and the gateway which is the antenna. The antenna I was using for my node and gateway was the one provided by the supplier when buying the module. I've done very few research about some others antennas which I honestly didn't understand much. But some antennas are more directional than others, some are made to transmit data in a very precise range of frequency. It could also help us having a longer distance between the node and the gateway.

3.3 Field trials

The next step of the project would be to try all my ideas in real conditions. I've done lot of trials in the office of the company, but it was in perfect conditions, good light, both LoRa transmitter and receiver 1 meter away, no electricity shutdown... It requires us to design a protection for all the sensors and the wires, something that doesn't scare the rats. Some of my colleagues have worked on other company projects involving animal behavior, particularly that of rats, their work could be very interesting for my device to find the perfect shape for the rat to be willing to get detected.

3.4 AI Algorithm

As I said for the trials I would like to take a photo every time we think to detect a rat, I'm not too afraid that the sensor will only triggers when we detect something and random photos won't be taken. But I'm afraid of others animals like birds or snakes... I would like to take a photo every time we think we detect a rat, having the percentage of success with different settings in our algorithm would allow us to have the best software to detect the rats. In order to have this success percentage we could use a trained AI algorithm for the rats detection. One could once a week, retrieve the photos on the SD Card and use the algorithm on them. Such an algorithm only makes sense if we have hundreds of photos to process, if we have a few dozens photos it may be uninteresting.

Chapter 4

Conclusion

The goal I was given was to find the best sensors for the project knowing all conditions, a rural area, sugarcan field ... Thanks to my resarch and some papers I decided to choose to use a node sensor module which communicates to a gateway using LoRa communication. Thanks to my knowledge and numerous resources found both on Internet and on academic papers I managed to make both the node and the gateway to communicate. I also designed a data base to store and process all the data we receive from the nodes. In addition, because I was eager to match the brief I was given when communicating with Cameron by mail (that you can find in the introduction), I also designed a prototype of graphical user interface which use the data from the data base whose data directly comes from the nodes. This internship allows me to make huge progress in Python, Arduino, the choose of the good sensors according to the restrictive conditions but also in the progress of a project when starting from scratch.

Chapter 5

Appendices

.1 Node Code

```
#include <LoRa.h>
#include <SPI.h>
#include <DS3232RTC.h>
#include "Adafruit_MLX90614.h"
#include <SoftwareSerial.h>

int counter = 0 ;

Adafruit_MLX90614 mlx = Adafruit_MLX90614();
DS3232RTC rtc;
SoftwareSerial CamSerial(7,8);

int ir1 = 4;
int ir2 = 5;
float temp;

void setup(){
  Serial.begin(115200);
  CamSerial.begin(115200);
  while(!Serial);
  Serial.println("LoRa Sender");

  while(!LoRa.begin(433E6)){
    Serial.println("LoRa Failed");
    delay(500);
  }
  if (!mlx.begin()) {
    Serial.println("Erreur de communication avec le
       MLX90614.");
    while (1);
  }
```

```
  pinMode(ir1,INPUT);
  pinMode(ir2,INPUT);

  Serial.println("MLX Ok!");
  LoRa.setSyncWord(0xA5);
  Serial.println("LoRa OK!");
}

void loop(){
  int ir1_status = digitalRead(ir1);
  int ir2_status = digitalRead(ir2);

  temp = mlx.readObjectTempC();

  if (ir1_status == 0 && ir2_status == 0 &&
     inRange(temp)){
    String date = getDate();
    CamSerial.println("TAKE_PHOTO");
    counter ++;

    Serial.print("Sending signal to gateway :");
    Serial.print(temp);
    Serial.print(" ; ");
    Serial.println(date);

    LoRa.beginPacket();
    LoRa.print("Node 1 ; ");
    LoRa.print(date + " ; ");
    LoRa.print(counter);
    LoRa.endPacket();
  };
  delay(200);
}

bool inRange(float temp){
  return 15 < temp < 45;
}

String getDate(){
  int Year = year();
  int Month = month();
  int Day = day();
  int Hour = hour();
  int Minute = minute();
  int Second = second();

  String yearStr = String(Year);
```

```
    String monthStr = (Month < 10) ? "0" + String(Month) :
        String(Month);
    String dayStr = (Day < 10) ? "0" + String(Day) :
        String(Day);
    String minuteStr = String(Minute);
    String secondStr = String(Second);
    String hourStr = String(Hour);

    return yearStr + "-" + monthStr + "-" + dayStr + "-" +
        hourStr + "-" + minuteStr +"-"+secondStr;
}
```

.2 ESP32-CAM Code

```
/*
   Author : Emre   ahin
   Date   : 1 Feb 20
   Email  : arxtechnologies@gmail.com
*/

#include "FS.h"        // SD Card
#include "SD_MMC.h"    // SD Card

#include "esp_camera.h"
#include "Arduino.h"
#include "soc/soc.h"           // Disable brownout problems
#include "soc/rtc_cntl_reg.h"  // Disable brownout problems
#include "driver/rtc_io.h"
#include <EEPROM.h>            // read and write from flash
    memory

// define the number of bytes you want to access
#define EEPROM_SIZE 4

// Pin definition for CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27

#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
```

```
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

uint8_t pictureNumber1 = 0;
uint8_t pictureNumber2 = 0;
uint8_t pictureNumber3 = 0;

void setup() {
  Serial.begin(115200);

  if (!SD_MMC.begin()) {
    Serial.println("Card Mount Failed");
    return;
  }
  uint8_t cardType = SD_MMC.cardType();

  if (cardType == CARD_NONE) {
    Serial.println("No SD_MMC card attached");
    return;
  }

  Serial.print("SD_MMC Card Type: ");
  if (cardType == CARD_MMC) {
    Serial.println("MMC");
  } else if (cardType == CARD_SD) {
    Serial.println("SDSC");
  } else if (cardType == CARD_SDHC) {
    Serial.println("SDHC");
  } else {
    Serial.println("UNKNOWN");
  }

  uint64_t cardSize = SD_MMC.cardSize() / (1024 * 1024);
  Serial.printf("SD_MMC Card Size: %lluMB\n", cardSize);

  delay(250);

  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable
     brownout detector
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
```

```
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG ;//PIXFORMAT_GRAYSCALE

  if (psramFound()) {
    config.frame_size = FRAMESIZE_VGA; // FRAMESIZE_ +
       QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 10;//0-63, lower means higher
       quality
    config.fb_count = 1;
  }

   // Init Camera
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  EEPROM.begin(EEPROM_SIZE);
  resetEEPROMMemory();      //Use this function when you
     need empty EEPROM
}

void takephoto(fs::FS &fs){
  String path = "/Photo - " + String(pictureNumber3) + "-" +
     String(pictureNumber2) + "-" + String(pictureNumber1) +
     ".jpg";
  Serial.printf("Writing file: %s\n",path.c_str());

  File file = fs.open(path.c_str(),FILE_WRITE);
  Serial.println("take photo..");
```

```
  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb){
    Serial.println("Camera capture failed");
    return;
  }

  if(!file){
    Serial.println("Failed to open file for writing");
    return;
  }
  file.write(fb->buf,fb->len);

  pictureNumber1++;
  if (pictureNumber1 > 254) {
    pictureNumber1 = 0;
    pictureNumber2 += 1;
    EEPROM.write(0, 0);
  }
  if (pictureNumber2 > 254) {
    pictureNumber2 = 0;
    pictureNumber3 += 1;
    EEPROM.write(1, 0);
  }
  if (pictureNumber3 > 6) {
    pictureNumber1 = 0;
    pictureNumber2 = 0;
    pictureNumber3 = 0;
    EEPROM.write(2, 0);
  }
  EEPROM.write(0, byte(pictureNumber1));
  EEPROM.write(1, byte(pictureNumber2));
  EEPROM.write(2, byte(pictureNumber3));
  EEPROM.commit();
  esp_camera_fb_return(fb);
}

void initBuff(char* buff){
  for (int i =0; i<240;i++){
    buff[i] = 0;
  }
}

void listDir(fs::FS &fs,const char * dirname,uint8_t levels){
  Serial.printf("Listing directory: %s\n",dirname);

  File root = fs.open(dirname);
  if (!root){
```

```
    Serial.println("Failed to open directory");
    return;
  }
  if (!root.isDirectory()){
    Serial.println("Not a directory");
    return;
  }

  File file = root.openNextFile();
  while(file){
    if (file.isDirectory()){
      Serial.print(" DIR : ");
      Serial.println(file.name());
      if (levels) {
        listDir(fs,file.name(),levels - 1);
      }
    } else {
      Serial.print(" FILE: ");
      Serial.print(file.name());
      Serial.print(" SIZE: ");
      Serial.println(file.size());
    }
    file = root.openNextFile();
  }
  file.close();
}

void resetEEPROMMemory(){
  for (int8_t i ; i < 4; i++){
    EEPROM.write(i,0);
  }
}

int i = 0;

void loop() {
  if (Serial.available()){
    Serial.println("Command received");
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "TAKE_PHOTO"){
      takephoto(SD_MMC);
    }
  }
}
```

.3 Gateway Code

```
#include <Preferences.h>
#include <SPI.h>
#include <LoRa.h>

Preferences preferences;
#define ss 5
#define rst 14
#define dio0 2

const char* node_id = "Node1";

void setup() {
  Serial.begin(115200);
  while(!Serial);
  LoRa.setPins(ss,rst,dio0);
  while(!LoRa.begin(433E6)){
    Serial.println("LoRa failed!");
    delay(500);
  }
  LoRa.setSyncWord(0xA5);
  Serial.println("LoRa Ok!");
  preferences.begin(node_id,false);
  printRatDetection();
}

void loop() {
  int packetSize = LoRa.parsePacket();
  if(packetSize){
    String incoming = "";
    while(LoRa.available()){
      incoming += (char)LoRa.read();
    }

    Serial.print("Received: ");
    Serial.println(incoming);

    int idIndex = incoming.indexOf(";");
    int dateIndex = incoming.indexOf(";", idIndex + 1);
    int countIndex = incoming.length();

    String nodeId = incoming.substring(0, idIndex);
    String date = incoming.substring(idIndex + 2, dateIndex);
    int count = incoming.substring(dateIndex + 2,
       countIndex).toInt();

    storeRatDetection(date, count);
  }
```

```
  delay(10);
}

void storeRatDetection(String date, int count){
  String entry = date + ";" + String(count);
  String allEntries = preferences.getString("entries", "");
  if (!allEntries.isEmpty()){
    allEntries += ",";
  }
  allEntries += entry;
  preferences.putString("entries", allEntries);
}

void printRatDetection() {
  String allEntries = preferences.getString("entries", "");
  Serial.println("Rat detections:");
  if (!allEntries.isEmpty()) {
    int start = 0;
    int end = allEntries.indexOf(',');

    while (end != -1) {
      String entry = allEntries.substring(start, end);
      int sepIndex = entry.indexOf(';');
      String date = entry.substring(0, sepIndex);
      int count = entry.substring(sepIndex + 1).toInt();
      Serial.print("Date: ");
      Serial.print(date);
      Serial.print(", Count: ");
      Serial.println(count);

      start = end + 1;
      end = allEntries.indexOf(',', start);
    }

    String entry = allEntries.substring(start);
    int sepIndex = entry.indexOf(';');
    String date = entry.substring(0, sepIndex);
    int count = entry.substring(sepIndex + 1).toInt();
    Serial.print("Date: ");
    Serial.print(date);
    Serial.print(", Count: ");
    Serial.println(count);
  } else {
    Serial.println("No detections recorded.");
  }
}
```

.4 Retrieve data from Arduino with python

.4.1 Interface with arduino

```
import serial

serial_port = '/dev/ttyUSB1' #My port, can be different
   according to your computer
baud_rate = 115200
write_to_file_path = "output.txt"

output_file = open(write_to_file_path, "w+")
ser = serial.Serial(serial_port, baud_rate)
while True:
    line = ser.readline()
    line = line.decode("utf-8")
    print(line)
    output_file.write(line)
```

.4.2 Add to the Database

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from tables import Base, Node, Farm, Gateway, Alert
```

[1]

```
DATABASE_URL = 'sqlite:///app_database.db'
engine = create_engine(DATABASE_URL)
Session = sessionmaker(bind=engine)
Base.metadata.create_all(engine)

path = "output.txt"
session = Session()
node_id = ('Node 2')
with open(path,'r') as f:
    for line in f.readlines():
        if line.startswith('Date'):
            line = line.rstrip('\n').split(' , ')
            date = line[0].split(': ')[1]
            date = date.split('-')[0:3]
            formatted_date = f"{date[0]}-{date[1]}-{date[2]}"
            count = line[1].split(': ')[1]
            print(f"Date : {formatted_date}, Count: {count}")
            new_alert =
                Alert(id=count,date=formatted_date,node_id=node_id)
```

[1]The tables module is my script to create the different tables of the Database.

```
session.add(new_alert)
session.commit()
```

Acknowledgment

I would like to first thank my parents, without them it would have been impossible for me to come to New Zealand to do my second year internship. They paid for my plane ticket, my visa and supported me monthly, without that financial support it would have been impossible for me to discover such an amazing country. Then, my thanks go to the entire team of Envico Technologies for the way they welcome me from the first day with Mike doing some groceries for me after my 50hours trip to everyday until the end of this internship. And I was very happy to go down the stairs to enter the office everyday. I would like to thank Cameron for everything, thank you for welcoming me, thank you for transforming a part of the office in accomodation to allow me (but not only) to stay here freely. Thank you again for the necklace and all this free food during my stay in Mount Maunganui. I will remember each person of the team and keep a very good memory of my internship. Currently the company tries to raise fundings, I really hope you'll manage to keep your culture, it makes the company a very pleasant place to work. I will end this part by thanking my teachers for their teaching which allows us to show the world what the student of ENSTA Bretagne are capable of.

Thank you !

References

[1] DG Trésor. *Situation économique et financière du Guatemala (Août 2019)*. 2019. URL: https://www.tresor.economie.gouv.fr/Articles/2019/08/19/situation-economique-et-financiere-du-guatemala-aout-2019.

[2] Karla Tay. "Guatemala:Sugar Annual". In: *USDA Foreign Agricultural Service* (2022).

[3] Michelle Smith et al. "Overcoming rat infestation in Australian sugarcane crops using an integrated approach that includes new rodenticide technology". In: *ACIAR MONOGRAPH SERIES* 96 (2003), pp. 238–241.

[4] Rajab Ali, S Fatima Mahdi, and M Farhanullah Khan. "Estimation of rodent damage on coconut plantations and sugarcane in Sindh". In: *Pak. J. Biol. Sci* 6.12 (2003), pp. 1051–1053.

[5] DE Williams and RM Corrigan. "The handbook: prevention and control of wildlife damage". In: *Nebraska: University of Nebraska* (1994).

[6] Shin-Chi Lai et al. "A Remote Monitoring System for Rodent Infestation Based on LoRaWAN". In: *Sensors* 23.9 (2023), p. 4185.

[7] Shilpa Devalal and A Karthikeyan. "LoRa technology-an overview". In: *2018 second international conference on electronics, communication and aerospace technology (ICECA)*. IEEE. 2018, pp. 284–290.

[8] SEMTECH. *SX 1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver*. Tech. rep. SEMTECH Corporation, Revised 7 - May 2020.

[9] Melexis. *MLX90614 family Datasheet Single and Dual Zone Infra Red Thermometer in TO-39*. Tech. rep. Melexis, 13 september 2019, pp. 42–44.

[10] E Suprayitno, M Marlianto, and M Mauliana. “Measurement device for detecting oxygen saturation in blood, heart rate, and temperature of human body”. In: *Journal of Physics: Conference Series* 1402 (Dec. 2019), p. 4. DOI: 10.1088/1742-6596/1402/3/033110.

[11] Maxim Integrated. *DS3231 Extremely Accurate I2C-Integrated RTC/TCX-O/Crystal*. Tech. rep. Maxim Integrated, 2015.