UNIVERSITEIT TWENTE

SMART ENVIRONMENTS PROJECT

DOCUMENTATION REPORT

Group 1

BEE-EATERS

Members

Matt Hassing s3052060
Badr Boubric s3036952
Carlijn le Clercq s3003280
Yana Volders s2933713
Isaac Sánchez s2871424
Ozan Kurtulus s2872080
Euripides Christofides s2727021

UNIVERSITEIT TWENTE.

Table of Contents

Chapter 0: Introduction

Our team is known as the BeeEaters, this species of bird is known for its colourful appearance. This is similar to our team, all of our team members have different individual strengths. Which, when combined, forms a beautiful collaboration, as beautiful as the bee-eater itself. Our team used this range of diverse strengths to develop a solution to the high cost and substantial allocation of research personnel, on the research of avian species. Current technology, such as radar, have a high initial and maintenance cost **[27][28]**. Hardware costs aside, radar transmitters require a considerably big amount of power for their standard operation. Necessary power ranges from about 5kW to 60kW. **[29]** This makes radar technology even more expensive and results in a substantial increase in the regular allocation of research personnel. Our motivation for addressing this problem is to facilitate more extensive and impactful avian research. By developing a cost-effective and efficient approach, we aim to remove barriers for future progress. Our goal is to make a meaningful contribution to the understanding and conservation of avian species.

The solution we propose is the Bioacoustic Intelligent Recording Device or B.I.R.D. for short. This solution will consist of a microcontroller that's running our custom machine-learning algorithm to detect bird calls from the Grey Heron (Ardea cinerea). Once detecting this bird call, the B.I.R.D. sends a message over LTE, which is a form of telecommunications, to our local server containing readings from all of our chosen electronic sensors. Since our primary goal is to tackle the problem of high setup and maintenance costs when researching avian species, it is important to note that any sensors can be hooked up to our microcontroller, and we are just using these sensors as an example of what's capable using a system such as the one we have designed. The system we have designed is made with expandability in mind, such that when needed we can produce multiple B.I.R.D. units and distribute them over a large area for large-scale sensing. Due to the artificial time constraint within our project, it is not within our limits to test the scalability of the B.I.R.D.

Despite the limitations of our project, we have been able to focus on one specific avian species to demonstrate the sensing capabilities of our B.I.R.D., the Grey Heron (Ardea cinerea). We've chosen the Grey Heron (Ardea cinerea) as it is a common bird in Enschede, with a distinctive bird call. This species provides the perfect opportunity to demonstrate the use and effectiveness of the B.I.R.D. in a real-world scenario. Additionally, the global population of the Grey Heron is slowly decreasing, making it all the more important to monitor the population, and protect and preserve its habitats. This will contribute to the long-term survival of this species.

This report serves to document all of our progress and the results of our efforts in developing a cost-effective solution to address the challenges of avian research using monitoring technology. Our proposed solution is the Bioacoustic Intelligent Recording Device (B.I.R.D). It utilises a custom machine-learning algorithm to detect bird calls from the Grey Heron. The data received from environmental sensors is sent to our server, and data can be accessed on our website.

Chapter 1: Literature Review

Drones count wildlife more accurately and precisely than humans
Using drones to monitor wildlife populations which are undergoing alarming declines, by targeting colonies to detect population fluctuations with high accuracy. This is achieved by collecting data through semi-automated detection and counting of wildlife using computer vision techniques.**[1]**

Wildlife monitoring, modelling, and fugacity. Indicators of chemical contamination
Observing and monitoring wildlife populations and their health in surroundings where chemical contaminations have peaked. Mostly focused on the concentrations of toxic chemicals in lakes and how wildlife is affected by this. For example, chemical residues are found in birds' eggs and migrate through food chains.**[2]**

Internet of Things for wildlife monitoring
location tracking, habitat environment observation, and behaviour recognition. By using sensors which communicate with each other, data is collected which is then visualised on a platform. Real-time communications between sensors happen through cellular(GSM, LTE) and capillary(RFID), depending on the distance between the sensors. Some of the sensors used are GPS, accelerometer, motion, temperature, and humidity sensors.**[3]**

A Wireless Sensor Network Air Pollution Monitoring System
This paper explains the usage of wireless sensors for air pollution in Mauritius. With the island's industrial activity expanding quickly, air pollution is posing a serious threat to the population's health. To address this, the Wireless Sensor Network Air Pollution Monitoring System (WAPMS) was developed.**[4]**

Camera trap research in Africa: A systematic review to show trends in wildlife monitoring and its value as a research tool
This paper talks about how camera traps contribute to wild monitoring in Africa. The author states that with camera traps, it is possible to explore rare species at lower costs. The solutions that could be used to reduce risks to biodiversity are also explained in the paper. **[5]**

Using drones to improve wildlife monitoring in a changing climate
This paper talks about the usage of drones in wild monitoring for climate change. According to the author, drones can easily gather crucial information to spot changes in vital factors like species abundance, distribution, and condition. The usage of drones is intended to assess mitigation plans for biodiversity loss and to enhance ecosystem management.**[6]**

UAV and IA wildlife monitoring (image recognition)
This paper suggests using UAVs (Unmanned Aerial Vehicles), AI and thermal image detection as it states that existing ground-based monitoring systems are expensive, inefficient, and inaccurate. Results were surprisingly good and showed the correct functioning of every component, but the paper concluded that detecting a large number of species would be difficult while also having limitations such as UAV regulations.**[7]**

Automated vs traditional monitoring techniques for marbled murrelets (seabird) using acoustic sensors
This paper states that autonomous sensors have a great advantage over traditional monitoring techniques, such as cost, efficiency, and data correlation. As the main experiment, autonomous acoustic sensors were implemented and showed that data was obtained 10 times faster, and had a near-perfect correlation between different days, all for the same price as the traditional sensing methods.**[8]**

Cloud Connected Smart Birdhouse for Environmental Parameter Monitoring
This paper states that smart technology is common in our daily lives and homes. However, we are not the only ones on this planet, so we could use this technology to stop the rapid decrease of birds. That is why the researchers want to build smart, cloud-connected birdhouses. This can be used to monitor and help endangered bird species. **[9]**

A prickly problem: developing a volunteer-friendly tool for monitoring populations of a terrestrial urban mammal, the West European hedgehog (Erinaceus europaeus)
This paper states that hedgehogs are in decline, but it is hard to help them since it is hard to monitor them. Monitoring animals usually requires the active participation of residents. One possible solution would be the 'footprint-tunnels'. These attract hedgehogs with food, mark the hedgehog's feet with ink, and can monitor their presence. **[10]**

Designing wildlife-vehicle conflict observation systems to inform ecology and transportation studies
This paper analysed web systems about data collection, management, visualisation and sharing regarding wildlife-vehicle collisions and how to improve these systems. Monitoring wildlife-vehicle collisions are important, as these can significantly decline the population of a certain species or in the worst case can lead to extinction. **[11]**

Biodiversity conservation and the extinction of experience
This paper states that people are disconnected from nature, mainly because half of the world's population lives in urban areas. Thus, they are not motivated to prevent biodiversity loss. The author suggests providing opportunities for people to interact with nature in highly populated places. It might even be beneficial for humans' mental health. **[12]**

Evolution and sustainability of a wildlife monitoring sensor network
This paper describes the findings from a one-year deployment of an automated wildlife monitoring system. This system is used to analyse European badgers also called Meles meles. It describes a different insight into the badger's behaviour and the potential of their monitoring system. They made continuous improvements to the prototype. They learned that it usually costs more to maintain a monitoring system, to carefully look at software and hardware interaction, to make sure that the prototype can easily be deployed and that the expertise of domain scientists is needed to optimise the system. **[13]**

Biodiversity conservation technologies in fishery
This paper discusses various ways to minimise the impact of fishing operations on wildlife in the ocean. For example, selective and eco-friendly fishing gear. This is called bycatch reduction technology. Different nets can be used for the different behaviour of different fish, to reduce the catching of non-target resources. By making these technologies eco-friendly there will be less pollution in the ocean. **[14]**

Region of Interest and Redundancy Problem in Migratory Birds WildLife Surveillance.
This paper talks about how climate change is impacting wetlands or humid zones. These zones are important to many species of birds. Because of this rare species are threatened with extinction. They need to closely monitor these species of birds to ensure they won't go extinct. Modern ways of monitoring birds are expensive and have a high impact on a network because of a constant video and audio stream. The paper explores the possibility of using image processing techniques to reduce both the cost and its impact on a network. **[15]**

A cost-effective protocol for monitoring birds using autonomous recording units: a case study with a night-time singing passerine.
This article aims to talk about a cost-effective method to monitor the night-time singing of Dupont's Lark. They want to achieve this by using autonomous recording units (ARUs). They found that the number of ARUs needed for a good result wildly differed depending on how dense the bird population in that area is. Furthermore, they also found that the recording time could be as little as an hour to achieve a reliable result. This way they can figure out how big the population is and how they are distributed in the area. This way they can assess the conservation status of species. **[16]**

Towards a New Opportunistic IoT Network Architecture for Wildlife Monitoring System
This article talks about a new Internet of Things architecture to monitor wildlife. They figured that processing data consumes less energy than sending raw data to a server. So the wildlife monitoring system (WMS) processes the data locally and then sends the much smaller processed data to a server. This way they can have the WMS out on a field for months or even years. **[17]**

Normalised Difference Vegetation Vigour Index: A New Remote Sensing Approach to Biodiversity Monitoring in Oil Polluted Regions
This paper talks about how current biodiversity methods are limited in their geographical coverage. The paper talks about a new way to remote-sense the species diversity and for monitoring the impact of oil pollution and environmental pressure on biodiversity at the regional scale. It integrates satellite remote sensing and field data to develop a set of spectral metrics for biodiversity monitoring. This new method is a superior remote sensing index for monitoring biodiversity indicators in oil-polluted areas than the previously used monitoring method. **[18]**

Persistent negative effects of pesticides on biodiversity and biological control potential on European farmland

A paper about the “Persistent negative effects of pesticides biodiversity and biological control potential on European farmland.” They concluded that even though some farms used pesticides classified as “Harmless”. They didn’t see the positive effects they expected to see on bird populations, because of the widespread use of pesticides across Europe and the size of their feeding areas. **[19]**

Importance of insects in environmental impact assessment

This paper discusses the importance of insects and their diversity to the functioning of an ecosystem, and how useful they can be to monitor the relative health of an ecosystem. Which their main point is that just looking at the levels of different chemical compounds can’t be enough to understand the underlying problem of the ecosystem.**[20]**

Chapter 2: Identification of General Problems and Challenges

1. Chemical contamination in wildlife causes a decline in the reproductive capacities of several species, which causes population decline. **[2]**
2. Air pollution monitoring on a national scale can be ineffective as it is not cost-effective, making it difficult to monitor and address the issue of air pollution. **[4]**
3. The current monitoring system for hedgehogs is reliant on human volunteers, making it insufficient and ineffective for a long period. Which in turn makes it difficult to track population trends and identify potential threats against hedgehogs.**[10]**
4. Collisions between automotive vehicles and wildlife are a serious problem which can cause damage to both the wildlife and vehicles. **[11]**
5. Declining biodiversity in ecosystems can have a negative impact on the ecosystem and all the species that depend on it. **[12]**
6. Ocean pollution can have a big impact on local water species, affecting their health, survival and reproduction. **[14]**
7. Optimising wildlife monitoring systems' data transfer protocol can help to improve the accuracy and efficiency of data collection and analysis, which allows for better understanding and conservation of wildlife. **[17]**
8. Pesticides can have a large-scale impact on birds, causing population declines. **[19]**

Chapter 3: Identification of Relevant Problems

1. Nitrogen pollution in freshwater ditches near farms, due to the overuse of artificial fertilisers, can have negative effects on the water quality and the amount of sunlight that can reach the bottom. **[21] [24]**
2. An estimated ten million animals are killed on Dutch roads every year. **[22]**
   Mostly focusing on squirrels trying to cross the highway as a result of habitat fragmentation. **[25]**
3. Many toads die from the annual toad migration. They are difficult to see for motorists because they mainly migrate at night when it is dark. Current solutions rely on human volunteers or are very expensive. **[23][26]**
4. Monitoring specific bird species using current technologies takes time and is resource intensive, which makes analysing their specific behaviour in correlation to their environment on a big scale not feasible nor affordable. **[9][15][16]**
5. Water currents in small bodies of water are difficult to measure over long periods of time which makes it difficult to monitor aquatic environments for aquatic wildlife. We want to find a way to make this easier and more affordable. **[14]**

Chapter 4: Problem Selection and Motivation

Our team has selected a critical problem in the field of avian research, specifically about the monitoring of specific bird species. At present, the primary method for large-scale bird monitoring, radar technology, is expensive. The initial costs of building the technology and the maintenance costs of keeping the technology operating are substantial. **[27][28]** Additionally, radar technology uses a lot of power to operate. The power necessary to operate this technology is around 5kW to 60 kW. **[29]** This makes the technology even more expensive, and results in a substantial increase in the regular allocation of research personnel.

Our motivation for addressing this problem is that we hope to facilitate more extensive and impactful avian research. By developing a cost-effective and efficient alternative for radar technology, we aim to remove barriers for future progress. Our goal is to make a meaningful contribution to the understanding and conservation of avian species.

Chapter 5: Potential Solutions

SOLUTIONS

Thermal camera
One solution to monitor birds is to use a high-sensitivity thermal camera.
Use a high-sensitivity thermal camera which can detect a bird's presence and take a picture with a camera when the thermal camera detects a lot of heat radiating from a small point.

Infrared sensor
An infrared sensor senses if there is movement. Then, a camera captures a picture or a video to monitor a bird's presence in its natural habitat.

Ultrasonic sensors on water
With the use of ultrasonic sensors (to measure the water surface) and wind measuring tools, we can monitor if birds are disturbing the water surface. We can also find out what species of bird this is by measuring the size and shape and by utilising a local neural network to process this data.

Record singing
The singing of different bird species is recorded with microphones. With these recordings, we can monitor their presence in different environments. When a bird is present, different sensors inside a mobile box can be used to collect data of the environment.

Weight Sensor
A weight sensor could be placed in the natural habitat of a bird species. By placing food to attract the birds we could detect the presence of a bird by sensing a change in weight.

Chapter 6: Solution Selection

The solution we decided on was to use a microphone to record the audio of the environment and process this audio in real time to detect if our selected bird is singing or not. For this solution to work we opted to create an alternative sensing unit, this unit consists of a microcontroller that's running a custom machine learning algorithm to detect bird calls from a specific species of bird. Once detecting this bird call, our sensing unit sends a message over LTE, which is a form of telecommunications, to our local server containing readings from all of our chosen electronic sensors. We decided to choose a Pressure, Light, Humidity, Temperature and Air quality sensor for our demo. Since our primary goal is to tackle the problem of high setup and maintenance cost when doing research on avian species, it is important to note that any sensors can be hooked up to our microcontroller, and we are just using these sensors as an example of what's capable using a system such as the one we have designed. We call this solution the Bioacoustic Intelligent Recording Device, or B.I.R.D. for short.
A point worth mentioning is, by using an IoT network, the possibility to use several autonomous modules exists, converting our model into a large-scale sensing system. For our model, we decided to study the behaviour of a specific bird species, the Grey Heron (*Ardea cinerea*) so we can test our solution in a real-world scenario. By using our solution, only retraining of our machine learning model is needed to fit this solution to a different type of bird.

The reason this solution was selected over the other proposed solutions is because of several reasons. By using a microphone we could focus our solution on one specific bird species, because there is a noticeable difference between bird calls of two different species. We are easily able to detect which species of bird is singing. This is not as, or is not, noticeable for a difference in weight or the size of waves between two different bird species. Furthermore, other solutions, like a weight sensor, heavily relied on the chosen species to be present in the exact right place to collect data, whereas using microphones broadens the test range.

To summarise, the selected solution is the most suitable for our use case. The solution is both affordable to design, implement and modular so it's able to be customised for multiple use cases. This makes our solution, the B.I.R.D. the most suitable solution for our specified problem.

Task division

Setting up audio pre-processing for the microcontroller (Matt)

- Writing the code for the microcontroller to handle the real-time preprocessing of audio data to make the audio more comprehensible for the machine learning algorithm to classify.
- Writing the code for preprocessing of the training data, the same way the audio will be preprocessed in the field.

Classifying and finding relevant sets of bird call sounds for machine learning (Carlijn)

- Finding and classifying a data set of bird calls of our chosen species of bird, the Grey Heron (*Ardea cinerea*).
- Finding and classifying a data set of environmental noise, and bird calls of other species.

Training our machine learning model (Isaac)

- Compiling a machine learning model.
- Feeding the found dataset into a machine learning model to accurately train our machine learning algorithm.

Deploying our tiny machine learning model (Isaac)

- Turning the model into a tiny machine learning model (TinyML).
- Writing the code so the TinyML model can run on the microcontroller.

Setting up other relevant inputs (environmental) sensors and power usage (Euripides)

- Writing the code for the microcontroller to read the current sensor data from the pressure, light, humidity, temperature and air quality sensor, and take the average of these sensors over 5 minutes.
- These sensors will provide a detailed reading of the environmental situation at the exact moment a Grey Heron call is detected.

Setting up communications with our main web server from the microcontroller (Badr)

- Writing code to make the microcontroller interface with a LTE capable sim card header.
- Send the data from the sensors to a server when the machine learning algorithm detects a bird call.

Designing the Web Interface (Yana)

- Receiving and formatting the data received from the microcontroller.
- Designing the website and programming it so users can see our data in a nice way.

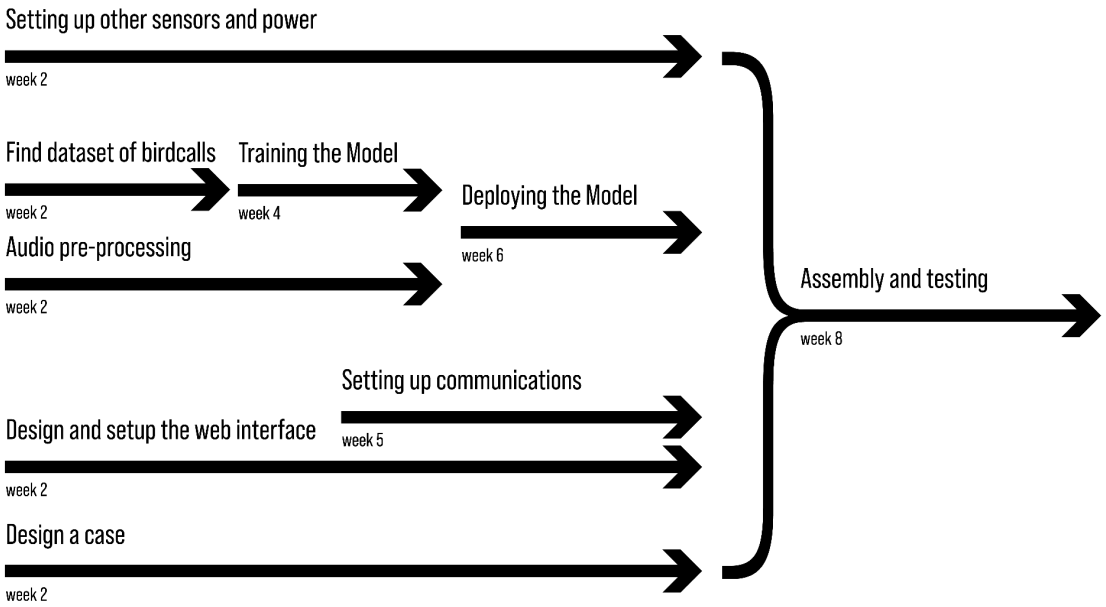
Designing a nature proof casing (Ozan)

- Design and create a casing that is waterproof and wear resistant.
- Making sure our sensors don't read wrong data because of a lack of airflow inside our casing.
- Making sure that our microphone can pick up sound.

Assembly, testing and documentation (Everyone)

- Assembling our final B.I.R.D.
- Testing our final B.I.R.D.
- Comparing our results.
- Ensuring that our final B.I.R.D. works as it's supposed to and gets accurate readings.
- When finished with individual tasks, documentation of the project.

Roadmap



Chapter 7: Methodology

The B.I.R.D. continuously gathers audio data from its surroundings through the use of a microphone, which is subsequently analysed by a custom machine learning algorithm for the specific bird call of the Grey Heron. When this call is detected, the B.I.R.D. activates various sensors and transmits the collected data from these sensors to a local server and is visualised on our own website. This system can be classified as a smart environment due to its ability to gather, process and utilise data to adapt and respond to the environment.

Equipment

Components

For our final B.I.R.D. we use an ESP32 as our microcontroller, the heart of operations. The two cores that are present are perfect for the concurrent tasks of preprocessing audio data and running our machine learning model. Besides that, its communication capabilities such as Wi-Fi and Bluetooth are a great fit for deploying multiple B.I.R.D.'s and making them communicate with each other. Note that this is not what we have got planned for the current project, but it is a great way to leave room for future expansion. The ESP32 is also relatively cheap for its capabilities. The BME280 is an outstandingly performing multisensor with a lot of bio-sensing capabilities such as humidity, temperature, and air quality. These sensors are used to demonstrate our systems capabilities. The MAX9814 is a microphone amplifier with automatic gain adjustment which is useful for normalising our input data, which in return gives our machine learning model a better chance at accurately identifying our audio data. For the detection of the amount of light we use a light dependent resistor which can show us how much light there was at the moment of sensing. Furthermore, we include a battery pack, which is powerful enough to run our whole project for days without having to recharge the battery. Finally, we have the casing which is made to be weatherproof and made to not hinder our sensors' sensing capabilities.

Component list

- ESP32 microcontroller
- SIM7020E NB-IoT HAT (LTE capable SIM card header)
- Humidity & Temperature BME280 & Air quality sensor CCS811 (SparkFun Environmental Combo Breakout - CCS811/BME280)
- Electret Microphone Amplifier MAX9814 with Auto Gain Control for our microphone
- Light Dependant Resistor (LDR)
- 2x PN2222 transistor
- Battery pack
- Solar cell for charging the battery
- Breadboard/PCB
- A casing for the sensors

Data collection

The microcontroller will start sending data from the environmental sensors whenever the machine learning algorithm detects a bird call. This will work by detecting sounds from a microphone, pre-processing that audio to make it easier for our TinyML to detect the bird calls. The data we will be collecting from the environment is the air quality using the CCS811. The humidity, pressure, and temperature using a BME280. The light intensity from a LDR, and finally the time from the built-in ESP32 clock. Even without the presence of a grey heron, that data about that day will be sent to an LTE capable SIM card header, so it can be sent to a server.
The air quality sensors detect the concentration of volatile organic compounds in percentages and an estimated concentration of carbon dioxide calculated from the known total VOC concentration. Calibration for these sensors are not needed, because they are all factory calibrated.

Method

Machine Learning model

The audio classification model consists of several different tasks.
First of all, a data set of not only bird calls, but also environmental noise needs to be created. Which results in the ability for the machine learning model to decide whether a recorded noise is the correct species.
For training our model several steps are required, firstly we must extract the individual data points from our training data. This is done by a script written in a modern programming language such as python. After each individual data point is collected we run these data points through a fast fourier transform algorithm which is imported from the same c library to ensure that our training data matches our real world collected data. After this step is completed, and we've collected all the samples and their corresponding FFT transforms, we start feeding our model the two datasets labelled “Heron” and not “Heron” This way our trained model can be used to determine whether a call corresponds to our chosen species or not. There are also other general improvements to the gathered input signal which can be done such as a high pass filter against wind, or physical protection against the wind.

Data use/analysis

Once our machine learning model detects a bird call from the Grey Heron (*Ardea cinerea*), all audio gathering gets stopped and we start reading all sensors. After the data is gathered the data will be communicated towards our sim-module to be sent towards the database. The active part of the B.I.R.D. is the microphone and machine learning model, while the rest of the sensors are off until a call from the Grey Heron (*Ardea cinerea*) is detected.

If no samples are collected within a day, meaning there was no presence of a Grey Heron (*Ardea cinerea*) call, information about the environment will still be sent at the end of that day. This is important for two reasons, first of all it’s used to detect if the device is still active, secondly it’s used to see what the conditions were like on a day that no Grey Heron (*Ardea cinerea*) call was detected.

Validation

Validating

We will be validating our project by using a loudspeaker to emulate a Grey Heron (*Ardea cinere*a) bird call noises. Our machine learning algorithm should then detect a Grey Heron (*Ardea cinerea*) and start sensing the environment. We will be doing this both inside and outside, this way we can check if reverb and feedback caused by objects near the emitter of the sound may cause any discrepancies. Our priority lies with minimising the chance of detecting a bird call when there is none (false positive), over a missed detection (false negative). This means that we prioritise that the tinyML does not recognize a random sound as a Grey Heron call.

Plan

Basic plan

For our basic plan we want to have a functioning B.I.R.D. box that can detect bird calls and sense its environment. The B.I.R.D. should also be able to send the data to a server with a LTE network. The basic plan is to not detect the behaviour of the bird but only their presence.

Ambitious plan

For our ambitious plan we would like to have multiple B.I.R.D.'s working in a grid based setup with one (or multiple) "Main" B.I.R.D. that sends the data to the server. This way not every B.I.R.D. needs to be equipped with a LTE capable sim card header, this will lower the cost. By using multiple boxes we could also see how the bird species moves through an area, giving us insight into the behaviour and movement of the bird.

Chapter 8: Validation

To validate the performance of the B.I.R.D., experiments in various environments were conducted. The call of the Grey Heron (*Ardea cinerea*) was replicated by playing the call on a speaker, and the B.I.R.D.`s microphone and machine learning algorithm were used to detect and recognize the call. However, during this experiment we encountered some difficulties.

During our first testing round our machine learning model was far from perfect, we had a fitness score of around 70%, which means our model was underperforming. After tweaking some of the training variables and marginal improvements, this version of the model was deemed unfit for deployment. So we started retraining the model, the new training showed marginal improvements. After careful consideration we started deployment and were faced with the same disappointing results, our model evaluated all of the Grey Heron calls wrong.

After these results, we re-evaluated our training data and realised that it consisted mostly of silence. After this realisation we hypothesised that we fitted our model to recognize silence, not Grey Heron calls. To confirm this hypothesis we deployed the physical unit to a silent place, where it indeed indicated that there were Grey Herons present while it was completely silent.

Thus, the decision was made to trim all our positively labelled datasets to only contain definite and discrete bird calls with approximately 100 to 200 milliseconds of silence at the start and end of each call. After training our model with this labelled dataset fitness scores of around 90% were achieved, in the end even an astonishing 97.5%. These results were quite positive to say the least. Our final model has a percentage of 12% for false positives. Which, after working on the algorithm we got down to 0.2%. However, note that an improvised casing utilising a container was used to protect the electronics during these experiments due to limitations of time in the making of the casing. This could have affected the validation experiment.

VERSION	CREATED	USER	DESCRIPTION	TRAINING ACCURA...	TEST ACCURACY	NO. OF SAMPLES	PUBLIC	
7	Today, 18:56:31	Badr	Epochs and training speed tweaks. Reduced overfitting and increased accuracy.	97%	97%	29m 25s	-	⋮
6	Today, 18:26:48	Badr	Reimported dataset with trimmed silence.	96%	96%	29m 25s	-	⋮
5	Today, 15:43:51	Badr	Reduction of window length	81%	78%	47m 25s	-	⋮
4	Today, 03:32:47	Badr	More epochs and training speed tweaks. Attempt at reduction of overfitting	81%	78%	47m 25s	-	⋮
3	Today, 02:58:35	Badr	First tested model with acceptable accuracy. Tweaks in epochs and training speed.	81%	71%	47m 25s	-	⋮
2	Today, 02:02:29	Badr	Reduce epochs, accelerate learning speed	70%	70%	57m 28s	-	⋮
1	Today, 01:43:20	Badr	V1	70%	70%	57m 28s	-	⋮

Figure 1: *Accuracy of different versions of the tinyML*

A video was made during the validation experiment. This was uploaded to YouTube for viewing purposes. (https://youtu.be/6nD04wWEBpo)

During the deployment of our model into an ESP we experienced some major difficulties, for instance our model had the wrong size and we had the wrong microphone for the provided scripts. But after careful revisions of our script and considering the size of the RAM of the ESP. A solution was found by filling in a buffer that perfectly fills the whole RAM of the ESP with our model, processing the audio data and outsourcing the actual collection of environmental data to another microprocessor. For this microprocessor another ESP-32 was chosen to be a good fit. This ESP was responsible for not only data collection but also communication, since our plan to use cellular communication ended up being unfeasible due issues with the cellular provider that was chosen, despite our valiant effort, could not be solved within the allotted time window.

The validation experiments were conducted both indoors, inside the DesignLab, where the ability of the model to filter out crowd noise was tested successfully**,** and in open air, which returned positive results as well. These tests were done to test the system's ability to detect the call in different settings and to ensure that specific sound reflections or feedback caused by nearby objects would not affect the systems' accuracy. Due to time constraints we were unable to do extensive non-automated testing and provide meaningful real world results, but our online testing showed the following numerical results.

Firstly, we've got the results of testing on non-training data. This shows that when using data from the training dataset we have no false positives, row and column (0,1). And 19.8% of our Grey Heron samples are detected as false negatives. Do note that this is data the model has been trained on.

Figure 2: *Accuracy of the Machine Learning Model*

Secondly, represented below are the resulting metrics of simulating ten minutes of real world examples created by concatenating different noises from our training set. In addition to real world environmental noises. This was to test how well our model is fitted training data of the simulated real world background noises. This rigorous training showed us that our model has a false positive rate of 0.2% as well as a false negative rate of 23% on our simulated real world data. The following image describes the relationship between the simulated real world audio and the models estimations. Green means correctly evaluated by the model, blue means false positive and red means false negative.

Figure 3: *Accuracy of the Machine Learning Model*

Overall, the validation experiments have provided a thorough evaluation of the B.I.R.D. system, even despite our lack of real world testing results. Our model has demonstrated that it could accurately detect the Grey Heron calls in different environments. The main focus of our validation experiment was to minimise false positives as much as possible. We think this was accomplished by only having a chance of 0% to 0.2% of false positives. The B.I.R.D.'s adaptability to different environments can now be seen as a strength. However, further improvements may be necessary to optimise the performance of the system in the future, such as fine-tuning the code and upgrading the microphone for broader frequency response.

Chapter 9: Results and Conclusion

The B.I.R.D. is designed to detect the specific bird call of the Grey Heron. The microcontroller (ESP32) runs a machine learning model to distinguish the Grey Heron from other audio taken by the microphone. The model is trained by two different audio datasets, a dataset of the Grey Heron call **[30]**, and different environmental audio samples. These datasets were added to help minimise the chance of false positives. These audio samples were for example other bird calls from different species or sounds from a busy street. **[31,32]** The model compares the audio input from the microphone to the audio datasets to identify the Grey Heron call. The code for the machine learning algorithm is included in appendix A and B.

In the end, two ESPs were used. The microphone is connected to the first ESP, and is always powered on. The second ESP is activated once the microphone hears a bird sound and determines that it is a Grey Heron. The latter will activate the other sensors and then transmit the data gathered at that time to the server. Our website obtains the data it needs to display from this server.

All the sensors included in the B.I.R.D. function properly. The microphone is continuously powered on and listens to the environment, while the ESP32 analyses the sounds. Once the sound has been declared a Grey heron call, the microphone is powered off, and the sensing ESP is notified, the sensing ESP then powers the environmental sensor and LDR in order to take environmental readings. After which the sensors are turned off again and the readings are sent to the server. Through the use of transistors, specifically 1 PN2222 transistor, we are able to control the power supply to the environmental sensor and the LDR.

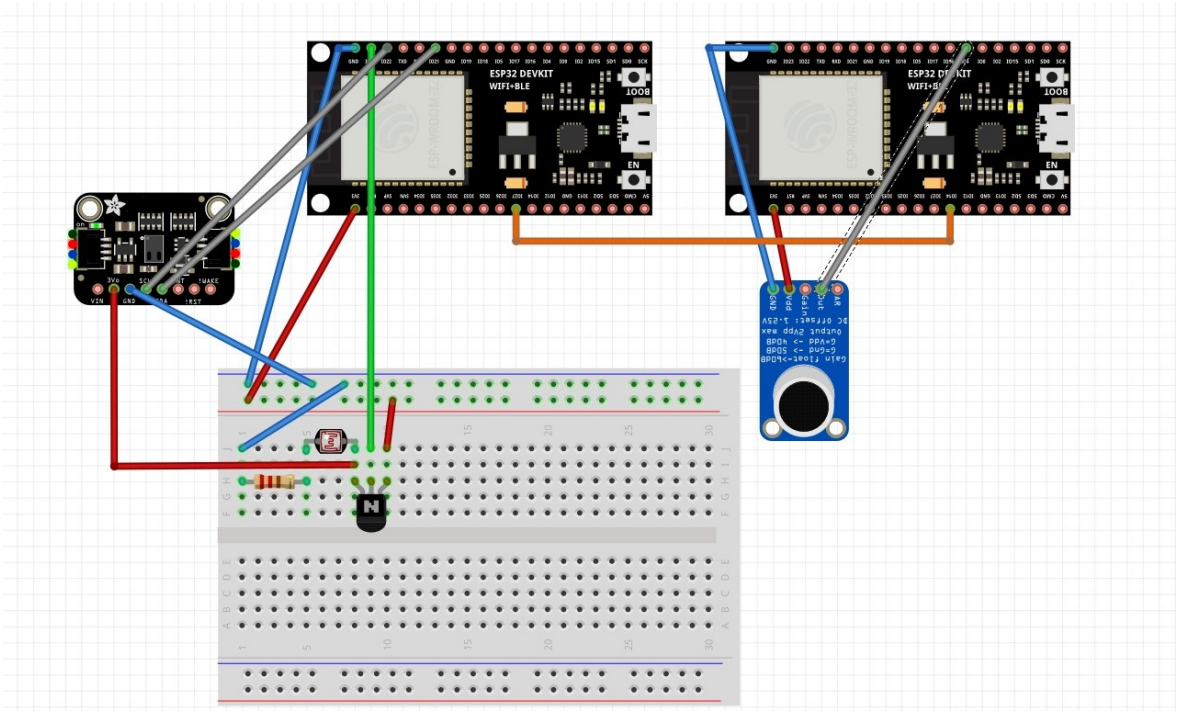


Figure 4: *Schematic of the electronics used in the B.I.R.D.*

The collected data from the B.I.R.D. can be viewed on our website (https://bird.yanavolders.com/). It features an interactive map displaying the locations of each B.I.R.D. with pins that showcase the latest reading of each device. Additionally, it provides a table of all gathered data, which can be sorted by ascending or descending order for greater clarity. This website is updated in real-time. Moreover, the website also has a small introduction of our project. Lastly, the programming of the website and database were designed for a network expansion in the future. This means that there is space for future additions for more tables and B.I.R.D.'s. Although this part of our project (as with many other parts of our project) are not represented as thoroughly as we had hoped in this report it is most definitely worth it to visit this website on a desktop browser, since it's a great representation of the physical attributes of the project.

For the sensors to function optimally, a protective casing with proper ventilation and waterproofing is required. The casing was created using Autodesk Fusion 360, a high-level design software. The casing consists of two primary bodies **[figure 5]**. The upper body **[figures 6,7]** was primarily designed to make the casing waterproof. The top portion of the body shields the microphone from water, while a small hole just beneath it allows for enough airflow. There is a small hole just beneath the top part, this way the microphone can pick up sounds without being obstructed by the casing. It also allows for enough air flow for the air quality sensor. In the event of rain, the top part will direct water to flow from top to bottom, thereby protecting the internal components, such as the microphone. All the other electronics are placed in the lower body of the casing **[figure 8]**. There are two cuts on the exterior of the casing that connect to the upper body, as was previously indicated. Due to the fact that not only the microphone but also the other sensors require airflow, there is a slight airflow when it is connected that passes through the lower body. At the bottom, there is a little block to support the casing. An interactive 3D model of the casing is shown on the homepage of the website (https://bird.yanavolders.com/).

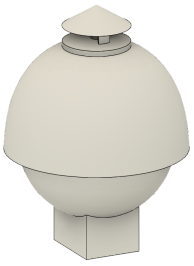


Figure 5: *Casing as a whole*

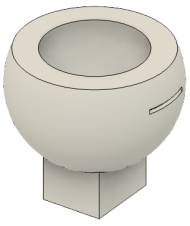


Figure 6: Lower body of the casing

Figure 7: *Upper body of the casing*

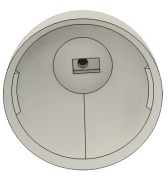


Figure 8: Inside of upper body

Firstly, the LTE capable SIM card header malfunctioned, it was not able to connect to the provider. After trying all the troubleshooting steps and trying to manually bind to an operator. The problem still kept persisting. Due to this problem and the time limitations the decision was made to omit the SIM card header and use the ESP32 on-board wifi in combination with a mobile hotspot to upload the data to the database.

The B.I.R.D. had an overall cost of approximately 100 euro per device, with the environmental sensor costing 40 euros, the microphone costing 10 euros, and shipping costs for both sensors at 12 euros. One individual ESP32 costs around 12 euro's, so for two ESP32's it would be 24 euros. The shipping cost was found to be a significant contributor to the overall cost. Like previously mentioned the B.I.R.D. did not include the SIM card header as originally planned. However, the SIM card header would be useful for implementing multiple B.I.R.D.'s into a grid system. This would allow for data to be transferred to the database without the need for the ESP32 to transfer the data using the in-built wifi. If the SIM card header is included in the grid system, it would not have to be included on every B.I.R.D. This would reduce the costs where multiple B.I.R.D.`s are utilised. In comparison to existing systems, our B.I.R.D.'s initial and maintenance costs are relatively low.

In conclusion, our team was able to develop a cost-effective solution that met the goal of our project. While one B.I.R.D. might be expensive, but when used in a grid system of multiple B.I.R.D.'s the production costs can be reduced. For future improvement, the B.I.R.D. can be redesigned with mass-production in a factory in mind. This would help reduce the production costs. For the validation experiment environmental sensors were used to demonstrate the B.I.R.D.'s capability. But these sensors can be replaced by a camera or different environmental sensors needed by the user. This would make our B.I.R.D. adaptable to the specific needs of the user.

Bibliography

[1]	Hodgson, J. C., Mott, R., Baylis, S. M., Pham, T. T., Wotherspoon, S., Kilpatrick, A. D., Raja Segaran, R., Reid, I., Terauds, A., & Koh, L. P. (2018b). Drones count wildlife more accurately and precisely than humans. Methods in Ecology and Evolution, 9(5), 1160–1167. https://doi.org/10.1111/2041-210x.12974
[2]	Clark, T., Clark, K., Paterson, S., Mackay, D., & Norstrom, R. J. (1988). Wildlife monitoring, modelling, and fugacity. Indicators of chemical contamination. Environmental Science &Amp; Technology, 22(2), 120–127. https://doi.org/10.1021/es00167a001
[3]	Internet of Things for wildlife monitoring. (2015, November 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/7961581
[4]	Khedo, K. K. (2010, May 11). A Wireless Sensor Network Air Pollution Monitoring System. arXiv.org. https://arxiv.org/abs/1005.1737
[5]	Ehlers Smith, D. A., Ehlers Smith, Y., & Downs, C. T. (2022). Camera trap research in Africa: A systematic review to show trends in wildlife monitoring and its value as a research tool. Global Ecology and Conservation, 40, e02326. https://doi.org/10.1016/j.gecco.2022.e02326
[6]	Koh, L. P. (2021, January 22). Adelaide Research & Scholarship: Using drones to improve wildlife monitoring in a changing climate. https://digital.library.adelaide.edu.au/dspace/handle/2440/129637
[7]	Gonzalez, L., Montes, G., Puig, E., Johnson, S., Mengersen, K., & Gaston, K. (2016). Unmanned Aerial Vehicles (UAVs) and Artificial Intelligence Revolutionising Wildlife Monitoring and Conservation. Sensors, 16(1), 97. https://doi.org/10.3390/s16010097
[8]	Borker, A. L., Halbert, P., Mckown, M. W., Tershy, B. R., & Croll, D. A. (2015). A comparison of automated and traditional monitoring techniques for marbled murrelets using passive acoustic sensors. Wildlife Society Bulletin, 39(4), 813–818. https://doi.org/10.1002/wsb.608
[9]	Raychev, J., Hristov, G., Kinaneva, D., Zahariev, P., & Kyostebekov, E. (2019). Cloud Connected Smart Birdhouse for Environmental Parameter Monitoring. 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). https://doi.org/10.23919/mipro.2019.8757080
[10]	Williams, B. (2018, August 27). A prickly problem: developing a volunteer-friendly tool for monitoring populations of a terrestrial urban mammal, the West European hedgehog (Erinaceus europaeus). SpringerLink. https://link.springer.com/article/10.1007/s11252-018-0795-1?error=cookies_not_supported&code=257c1717-5ea0-4981-bb96-d55dac697099
[11]	Shilling, F., Collinson, W., Bil, M., Vercayie, D., Heigl, F., Perkins, S. E., & MacDougall, S. (2020). Designing wildlife-vehicle conflict observation systems to inform ecology and transportation studies. Biological Conservation, 251, 108797. https://doi.org/10.1016/j.biocon.2020.108797
[12]	Miller, J. R. (2005). Biodiversity conservation and the extinction of experience. Trends in Ecology & Evolution, 20(8), 430–434. https://doi.org/10.1016/j.tree.2005.05.013

[13]	Dyo, V., Yousef, K., Ellwood, S. A., Macdonald, D. W., Markham, A., Mascolo, C., Pásztor, B., Scellato, S., Trigoni, N., & Wohlers, R. (2010). Evolution and sustainability of a wildlife monitoring sensor network. Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems - SenSys '10. https://doi.org/10.1145/1869983.1869997
[14]	Boopendranath, M. R. (2012, January). Biodiversity conservation technologies in fisheries. Researchgate. https://www.researchgate.net/publication/239172037_BIODIVERSITY_CONSERVATION_TECHNOLOGIES_IN_FISHERIES
[15]	Region of Interest and Redundancy Problem in Migratory Birds WildLife Surveillance. (2022, September 17). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/9931576
[16]	A cost-effective protocol for monitoring birds using autonomous recording units: a case study with a night-time singing passerine. (n.d.). Taylor & Francis. https://www.tandfonline.com/doi/full/10.1080/00063657.2018.1511682
[17]	Towards a New Opportunistic IoT Network Architecture for Wildlife Monitoring System. (2018, February 1). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/8328721
[18]	Onyia, N. N. (n.d.). Normalised Difference Vegetation Vigour Index: A New Remote Sensing Approach to Biodiversity Monitoring in Oil Polluted Regions. MDPI. https://www.mdpi.com/2072-4292/10/6/897
[19]	Geiger, F., Bengtsson, J., Berendse, F., Weisser, W. W., Emmerson, M., Morales, M. B., Ceryngier, P., Liira, J., Tscharntke, T., Winqvist, C., Eggers, S., Bommarco, R., Pärt, T., Bretagnolle, V., Plantegenest, M., Clement, L. W., Dennis, C., Palmer, C., Oñate, J. J., . . . Inchausti, P. (2010). Persistent negative effects of pesticides on biodiversity and biological control potential on European farmland. Basic and Applied Ecology, 11(2), 97–105. https://doi.org/10.1016/j.baae.2009.12.001
[20]	Rosenberg, D. M. (1986, November 1). Importance of insects in environmental impact assessment. SpringerLink. https://link.springer.com/article/10.1007/BF01867730?error=cookies_not_supported&code=aa6afd8f-2b13-48c3-8a47-9f1416a79216
[21]	Ministerie van Economische Zaken en Klimaat. (2022, 5 april). Maatregelen mestgebruik. Mest | Rijksoverheid.nl. https://www.rijksoverheid.nl/onderwerpen/mest/maatregelen-mestgebruik
[22]	Robin. (2016, 12 februari). More animals killed on the roads is "good news" for wildlife. DutchNews.nl. https://www.dutchnews.nl/news/2016/02/84971-2/
[23]	De paddentrek komt weer op gang en daarom is de hulp van vrijwilligers hard nodig. (2021, 5 december). RTV Utrecht. https://www.rtvutrecht.nl/nieuws/2139422/de-paddentrek-komt-weer-op-gang-en-daarom-is-de-hulp-van-vrijwilligers-hard-nodig
[24]	The Issue. (2022, August 11). US EPA. https://www.epa.gov/nutrientpollution/issue
[25]	Verboom, B. (1990, July 1). Effects of habitat fragmentation on the red squirrel, Sciurus vulgaris L. SpringerLink. https://link.springer.com/article/10.1007/BF00132859?error=cookies_not_supported&code=7cdde65d-9b7f-4791-8ec7-48bb874f64fe

[26]	De paddentrek komt weer op gang en daarom is de hulp van vrijwilligers hard nodig. (2021, December 5). RTV Utrecht. https://www.rtvutrecht.nl/nieuws/2139422/de-paddentrek-komt-weer-op-gang-en-daarom-is-de-hulp-van-vrijwilligers-hard-nodig
[27]	Duer, S. (2010). Expert Knowledge Base to Support Maintenance of a Radar System. Defence Science Journal, 60(5), 531–540. https://core.ac.uk/download/pdf/333719768.pdf
[28]	Ehasz, R. F., Cunningham III, W. A. & Bell, J. E. (n.d.). COST BENEFIT ANALYSIS OF AVIAN RADAR SYSTEMS. http://wdsinet.org/Annual_Meetings/2013_Proceedings/papers/paper94.pdf
[29]	Gauthreaux, S. A. & Schmidt, P. M. (2013). Radar technology to Monitor Hazardous Birds at Airports. Wildlife in Airport Environments, Chapter 13. https://books.google.nl/books?hl=en&lr=&id=wekEAQAAQBAJ&oi=fnd&pg=PA141&dq=radar+wildlife+monitoring+risk&ots=mJSoP1Yt6R&sig=sxWQYQjgjxsXKVD1eyxbAOdZiyw&redir_esc=y#v=onepage&q=radar%20wildlife%20monitoring%20risk&f=false
[30]	Xeno-Canto. (n.d.). Grey Heron Calls [Audio Samples; Xeno-Canto]. https://xeno-canto.org/set/8260 *Certain audio samples were modified to enhance their suitability for the use in the machine learning algorithm.*
[31]	Gloaguen, J.-R., Lagrange, M., Can, A., & Petiot, J.-F. (2017). Isolated Urban Sound Database [Audio Samples; Zenodo]. https://doi.org/10.5281/zenodo.1213793 *Certain audio samples were removed from the dataset as they were deemed irrelevant for the purpose of training the machine learning algorithm.*
[32]	Gloaguen, J.-R., Can, A., Lagrange, M., & Petiot, J.-F. (2018). Realistic urban sound mixture dataset [Audio Samples; Zenodo]. https://doi.org/10.5281/zenodo.1184443 *Certain audio samples were removed from the dataset as they were deemed irrelevant for the purpose of training the machine learning algorithm.*

Appendix A

The code for the Machine Learning Model ESP

```
/* CODE BASED ON EDGE IMPULSE'S DISTRIBUTED EXAMPLE, CAME WITH
 * THE FOLLOWING COPYRIGHT NOTICE:
 *
 * Edge Impulse ingestion SDK
 * Copyright (c) 2022 EdgeImpulse Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */

/* Includes
---------------------------------------------------------------- */
#include <bird_inferencing.h>

float features[24000];

#define micPin 4
#define AISuccesPin 14
#define SAMPLES 24000        //must be 2^N more samples bigger RAM
usage...
#define SAMPLING_FREQ 32000  // Hz, must be 40000 or less due to ADC
conversion time. Determines maximum frequency that can be analysed by the
FFT Fmax=sampleF/2.

unsigned int samplingPeriodInMUSeconds;  // calculates the sampling
period
unsigned long timeElapsedSince;          // keeps track of the amount of
time since previous itteration of sampling has ran
int count;                               // used to keep track of the
amount of itterations the audio has been sampled like an oversized for
loop
```

```
/**
 * @brief      Copy raw feature data in out_ptr
 *             Function called by inference library
 *
 * @param[in]  offset   The offset
 * @param[in]  length   The length
 * @param      out_ptr  The out pointer
 *
 * @return     0
 */
int raw_feature_get_data(size_t offset, size_t length, float *out_ptr) {
  memcpy(out_ptr, features + offset, length * sizeof(float));
  return 0;
}


/**
 * @brief      Arduino setup function
 */
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(AISuccesPin, OUTPUT);
  samplingPeriodInMUSeconds = round(1000000 * (1.0 / SAMPLING_FREQ));
  digitalWrite(AISuccesPin, LOW);

  count = 0;

  // comment out the below line to cancel the wait for USB connection
(needed for native USB)
  while (!Serial)
    ;
  Serial.println("Edge Impulse Inferencing Demo");
}

/**
 * @brief      Arduino main function
 */
void loop() {
  float floatedmic;
  //big sort of forloop to keep our samopling consistant
  if (count < SAMPLES) {
    // Sample the audio pin
    // a wait function to keep the sample rate consistent with the
internal clock
```

```
    while ((micros() - timeElapsedSince) < samplingPeriodInMUSeconds) {
      // don't do anything and wait for next sample
      Serial.println("ik doe niks");
    }
    features[count] = (((float)analogRead(micPin)) / 4048);
  } else {
    count = 0;

    ei_printf("Edge Impulse standalone inferencing (Arduino)\n");

    if (sizeof(features) / sizeof(float) !=
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE) {
      ei_printf("The size of your 'features' array is not correct.
Expected %lu items, but had %lu\n",
                EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, sizeof(features) /
sizeof(float));
      delay(1000);
      return;
    }

    ei_impulse_result_t result = { 0 };

    // the features are stored into flash, and we don't want to load
everything into RAM
    signal_t features_signal;
    features_signal.total_length = sizeof(features) /
sizeof(features[0]);
    features_signal.get_data = &raw_feature_get_data;

    // invoke the impulse
    EI_IMPULSE_ERROR res = run_classifier(&features_signal, &result,
false /* debug */);
    if (res != EI_IMPULSE_OK) {
      ei_printf("ERR: Failed to run classifier (%d)\n", res);
      return;
    }

    // print inference return code
    ei_printf("run_classifier returned: %d\r\n", res);
    print_inference_result(result);
    delay(1000);
  }
  count++;
}
void print_inference_result(ei_impulse_result_t result) {
```

```
  // Print how long it took to perform inference
  ei_printf("Timing: DSP %d ms, inference %d ms, anomaly %d ms\r\n",
            result.timing.dsp,
            result.timing.classification,
            result.timing.anomaly);

  // Print the prediction results (object detection)
#if EI_CLASSIFIER_OBJECT_DETECTION == 1
  ei_printf("Object detection bounding boxes:\r\n");
  for (uint32_t i = 0; i < result.bounding_boxes_count; i++) {
    ei_impulse_result_bounding_box_t bb = result.bounding_boxes[i];
    if (bb.value == 0) {
      continue;
    }
    ei_printf("  %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\r\n",
              bb.label,
              bb.value,
              bb.x,
              bb.y,
              bb.width,
              bb.height);
  }

  // Print the prediction results (classification)
#else
  ei_printf("Predictions:\r\n");
  for (uint16_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT; i++) {
    ei_printf("  %s: ", ei_classifier_inferencing_categories[i]);
    ei_printf("%.5f\r\n", result.classification[i].value);
  }
  if(result.classification[1].value>0.75{
    Serial.println("HARON");
    digitalWrite(AISuccesPin, HIGH);
  }else{
    digitalWrite(AISuccesPin, LOW);
  }
#endif

  // Print anomaly result (if it exists)
#if EI_CLASSIFIER_HAS_ANOMALY == 1
  ei_printf("Anomaly prediction: %.3f\r\n", result.anomaly);
#endif
}
```

Appendix B

The code for the Sensing ESP

```
#include <string.h>
#include <DFRobot_ENS160.h>
#include "DFRobot_BME280.h"
//inputs ----------------------------
#define LED 13
#define COMSPIN 27

//*****WHO?******
uint8_t BOXID = 1;
//*************

#define transistorPinSensors 23
#define SEA_LEVEL_PRESSURE 1038.0f
#define LDR_PIN 33

bool sensorSetupFinished = false;

//sensors adress-----------------------
DFRobot_ENS160_I2C ENS160(&Wire, / i2cAddr / 0x53);
typedef DFRobot_BME280_IIC BME;  // *** use abbreviations instead of full
names ***
BME bme(&Wire, 0x76);            // select TwoWire peripheral and set
sensor address

//ESP OS
//wifi-------------------------------------------
#include <WiFi.h>
#include <HTTPClient.h>
#include <WebServer.h>
#include <Preferences.h>

//esp name and credentials -----------------------
const char* name = "B.I.R.D.";  //your name
char systemName[20];            //device name

//variables for writing our credentials to memmory
Preferences preferences;  //setting up memmory management
String ssid, password;    //setting up Strings

//webserver--------------------------------------------------
WebServer server(80);              //specifying the webserver port
bool internetSetupActive = false;  //is the webserver already in
```

```
setupmode
String responseHeader;              //specifying a place where the users
response is stored


void setup() {
  Serial.begin(115200);
  //Defining the light dependant resistor as an input
  pinMode(LDR_PIN, INPUT);

  //Enviromental sensor
  pinMode(transistorPinSensors, OUTPUT);

  // the AI saw a thing input
  pinMode(COMSPIN, INPUT);

  delay(500);
  wifiManagerSetup();
}

void loop() {
  if (internetSetupActive) {
    server.handleClient();
  } else {
    if (digitalRead(COMSPIN) == HIGH) {
      readSensorData(true);
    }
  }
}

// show last sensor operate status
void printLastOperateStatus(BME::eStatus_t eStatus) {
  switch (eStatus) {
    case BME::eStatusOK: Serial.println("everything ok"); break;
    case BME::eStatusErr: Serial.println("unknow error"); break;
    case BME::eStatusErrDeviceNotDetected: Serial.println("device not
detected"); break;
    case BME::eStatusErrParameter: Serial.println("parameter error");
break;
    default: Serial.println("unknow status"); break;
  }
}

void setupEnviromentalSensor() {
```

```
  bme.reset();
  Serial.println("bme read data test");
  while (bme.begin() != BME::eStatusOK) {
    Serial.println("bme begin faild");
    printLastOperateStatus(bme.lastOperateStatus);
    delay(2000);
  }
  Serial.println("bme begin success");
  delay(100);

  // Initialising the sensor
  while (NO_ERR != ENS160.begin()) {
    Serial.println("Communication with device failed, please check
connection");
    delay(3000);
  }

  //Using the ENS160 in its standard mode
  ENS160.setPWRMode(ENS160_STANDARD_MODE);

  ENS160.setTempAndHum(/temperature=/bme.getTemperature(),
/humidity=/bme.getHumidity());
}

//If there is a bird we allow power from the esp32 to the sensors via the
transistor in order to get enviromental readings
void readSensorData(bool wasThereABird) {
  delay(1000);
  digitalWrite(transistorPinSensors, HIGH);   //Here we enable power
through the transistor to the sensors

  setupEnviromentalSensor();
  delay(120000);    //We have this 2 minute delay in order for the ENS160
to warm up and give us accurate readings
  //LDR readigs
  int ldrValue = analogRead(LDR_PIN);
  int mappedLDR = map(ldrValue, 0, 3000, 100, 1);   //Mapping the LDR
values to percentages
  if (mappedLDR < 1) {
    mappedLDR = 1;
  }
  //Enviromental sensor readings
  float temp = bme.getTemperature();
  uint32_t press = bme.getPressure();
  float alti = bme.calAltitude(SEA_LEVEL_PRESSURE, press);
```

```
  float humi = bme.getHumidity();
  /**
   * Get the air quality index
   * Return value: 1-Excellent, 2-Good, 3-Moderate, 4-Poor, 5-Unhealthy
   */
  uint8_t AQI = ENS160.getAQI();
  /**
   * Get TVOC concentration
   * Return value range: 0-65000, unit: ppb
   */
  uint16_t TVOC = ENS160.getTVOC();
  /**
   * Get CO2 equivalent concentration calculated according to the
detected data of VOCs and hydrogen (eCO2 - Equivalent CO2)
   * Return value range: 400-65000, unit: ppm
   * Five levels: Excellent(400 - 600), Good(600 - 800), Moderate(800 -
1000),
   *               Poor(1000 - 1500), Unhealthy(> 1500)
   */
  uint16_t ECO2 = ENS160.getECO2();

  Serial.println();
  Serial.println("======== start print ========");
  Serial.print("temperature (unit Celsius): ");
  Serial.println(temp);
  Serial.print("pressure (unit pa):         ");
  Serial.println(press);
  Serial.print("altitude (unit meter):      ");
  Serial.println(alti);
  Serial.print("humidity (unit percent):    ");
  Serial.println(humi);
  Serial.print("light (unit percent):       ");
  Serial.println(mappedLDR);
  Serial.println("=============================");
  Serial.print("Air quality index :         ");
  Serial.println(AQI);
  Serial.print("Concentration VOC :         ");
  Serial.print(TVOC);
  Serial.println(" ppb");
  Serial.print("CO2 concentration :         ");
  Serial.print(ECO2);
  Serial.println(" ppm");
  Serial.println("========  end print  ========");
  Serial.println(" ");
```

```
  digitalWrite(transistorPinSensors, LOW);
  delay(500);

  post("https://bird.yanavolders.com/esp_update.php", uint8_t(mappedLDR),
int8_t(AQI), double(temp), long(humi), press, TVOC, ECO2, wasThereABird,
BOXID);

  delay(1000);
}


void post(String url, uint8_t LDR, int8_t AirQuality, double Temperature,
long Humidity, uint32_t Pressure, uint16_t VOC, uint16_t CO2, bool
isDetected, int8_t boxID) {

  if (WiFi.status() == WL_CONNECTED) {  //Check WiFi connection status
    HTTPClient http;

    Serial.println("Commence communication");
    String postData = "addVariables=1";
    postData += "&LDR=" + String(LDR);
    postData += "&AirQuality=" + String(AirQuality);
    postData += "&Temperature=" + String(Temperature);
    postData += "&Humidity=" + String(Humidity);
    postData += "&Pressure=" + String(Pressure);
    postData += "&VOC=" + String(VOC);
    postData += "&CO2=" + String(CO2);
    postData += "&Detected=" + String(isDetected);
    postData += "&boxID=" + String(boxID);

    http.begin(url);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    int httpResponseCode = http.POST(postData);
    Serial.println(postData);
    String response = http.getString();
    Serial.println(httpResponseCode);
    Serial.println(response);

    http.end();
    delay(5000);
  }
}
```

```
void wifiManagerSetup() {
  //setting up our system
  strcpy(systemName, "ESP_OS_");
  strcat(systemName, name);

  delay(100);

  //bootup saying hello
  Serial.println();
  Serial.println("ESP_OS");
  Serial.print("Welcome, ");
  Serial.println(name);

  delay(10);

  //starting our trials to connect to a wifi network
  Serial.println("Attempting to connect to wifi.");
  preferences.begin("ESP_OS_", true);  //setting up writing to memmory

  if (preferences.getString("ssid", "NULL") != "NULL" &&
preferences.getString("password", "NULL") != "NULL") {
    //wifi pass and login has been found in memmory

    //retrieve them
    password = preferences.getString("password", "NULL");
    ssid = preferences.getString("ssid", "NULL");
    preferences.end();

    //start connection with wifi network
    Serial.println("Starting connection with " + ssid);

    wifiStartup();

  } else {
    //there hasn't been any wifi credentials, so booting up in AP mode...
    Serial.println("No valid string found, starting wifi setup.");
    preferences.end();
    wifiSetup();
  }
}

void wifiStartup() {
  WiFi.mode(WIFI_STA);
  Serial.print("Name: ");
  Serial.println(ssid.c_str());
```

```
  Serial.print("Password: ");
  Serial.println(password.c_str());
  WiFi.begin(ssid.c_str(), password.c_str());
  Serial.print("Connecting to WiFi ..");

  //if the wifi doesnt connect for to long, we go ahead and start up our
  //wifisetup wizard again
  int wifi_idle_counter = 0;
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    if (wifi_idle_counter >= 16) {
      Serial.println();
      Serial.println("returning to setup");
      wifiSetup();
      return;
    }
    delay(1000);
    wifi_idle_counter++;
  }
  Serial.println();

  //if it does connect, great!
  Serial.println("succesfully connected!");
  Serial.print("Ip Adress: ");
  Serial.println(WiFi.localIP());
}

void wifiSetup() {
  //stop the current wifi nonsense
  WiFi.softAPdisconnect();
  WiFi.disconnect();
  server.stop();

  //spin up a temporary webserver
  WiFi.mode(WIFI_AP);

  Serial.print("Join the network: ");
  Serial.println(systemName);
  WiFi.softAP(systemName);
  WiFi.softAPConfig(IPAddress(192, 168, 4, 1), IPAddress(192, 168, 4, 1),
IPAddress(255, 255, 255, 0));

  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
```

```
  internetSetupActive = true;
  server.begin();
  Serial.println("Server has started, /\\ visit tis ip  ");

  //setting up the methods for the server to handle,
  //if someone visits ipadres/submit handlesubmit is called
  server.on("/", handle_OnConnect);
  server.on("/submit", handle_submit);
  server.onNotFound(handle_NotFound);
}

//handles when a user request a connection
void handle_OnConnect() {
  Serial.println("Webclient has returned a connect request.");
  server.send(200, "text/html", SendHTML(0));
}

//handles when a requested url is not found
void handle_NotFound() {
  server.send(404, "text/plain", "The webpage was not found");
}

//this method gets called once someone presses on the
//submit button on our webpage, where they added in new credentials.
void handle_submit() {
  String wifiName_ = server.arg("WifiName");  //check what is returned as
our wifiname
  String password_ = server.arg("Password");  //check what is returned as
our password

  if (wifiName_ != "") {  //if the wifiname is not empty we run the code,
because an empty password can be true
    Serial.print("Specified wifi name: ");
    Serial.println(wifiName_);
    Serial.print("Specified wifi password: ");
    Serial.println(password_);

    //display the visitor of our webpage that their credentials have been
excepted
    server.send(200, "text/html", SendHTML(1));

    //saving these credentials to flash
    preferences.begin("ESP_OS_", false);
    preferences.putString("password", password_);
    preferences.putString("ssid", wifiName_);
```

```
    password = preferences.getString("password", "NULL");
    ssid = preferences.getString("ssid", "NULL");
    preferences.end();
    Serial.println("All safely stored!");

    //stop the current wifi and webserver nonsense
    WiFi.softAPdisconnect();
    WiFi.disconnect();
    server.stop();
    Serial.println("Starting connection with: " + password);

    internetSetupActive = false;
    delay(100);

    //and startup the wifi
    wifiStartup();

  } else {  //if the wifiname is not empty send the normal webpage again!
    server.send(200, "text/html", SendHTML(0));
  }
}

//html things!
String SendHTML(uint8_t caseNum) {
  //document header
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0, user-scalable=no\">\n";
  ptr += "<title>ESP_OS</title>\n";

  //styling
  ptr += "<style>";
  ptr +=
"h1,h3{color:#eee}.button,p{font-size:16px}html{font-family:Helvetica;dis
play:inline-block;margin:0
auto;text-align:center}body{margin-top:50px;background-color:#282828}h1{m
argin:50px auto
30px}h3{margin-bottom:50px}p{color:#e9e9e9;margin-bottom:10px}.button{dis
play:block;width:160px;background-color:#3498db;border:none;color:#fff;pa
dding:13px 30px;text-decoration:none;margin:20px auto
35px;cursor:pointer;border-radius:2px}.button-on,.button-on:active,input[
type=text]{background-color:#5f4bb6}.button-off{background-color:#3a325a}
.button-off:active{background-color:#3a2c50}input[type=text]{width:12rem;
padding:12px 20px;margin:8px
10px;font-size:14px;box-sizing:border-box;color:#e9e9e9;border:none;borde
```

```
r-radius:2px}";
  ptr += "</style>\n";
  ptr += "</head>\n";

  //html dependent on whats happening on our server
  switch (caseNum) {
    case 0:
      ptr += "<body>\n";
      ptr += "<h1>ESP_OS WifiWizard</h1>\n";
      ptr += "<h3>Welcome ";
      ptr += name;
      ptr += "</h3>\n";

      //input fields
      ptr += "<form action = \"/submit\" method = \"get\">";
      ptr += "<input type = \"text\" name = \"WifiName\" id =
\"WifiName\"></input>";
      ptr += "<input type = \"text\" name = \"Password\" id =
\"Password\"></input>";
      ptr += "<input type = \"submit\" class=\"button button-on\" value =
\"Submit\"></input>";
      ptr += "</form>";

      // ptr += "<a class=\"button button-on\"
href=\"/submit\">Submit</a>\n";

      break;

    case 1:
      ptr += "<body>\n";
      ptr += "<h1>ESP_OS WifiWizard</h1>\n";
      ptr += "<h3>Welcome ";
      ptr += name;
      ptr += "</h3>\n";
      ptr += "<p>Input has been accepted and the ESP will try to connect
shortly";
      ptr += "<p>\n";
      break;

    case 2:
      break;

    default:
      break;
  }
```

```
  //closing html tags...
  ptr += "</body>\n";
  ptr += "</html>\n";
    return ptr;
}
```