|  |
| --- |
| BIOQUAI |
| Smart Gate Overview |
| Detailing the design principles, construction, and operation of the Smart Gate |

|  |
| --- |
| Thomas Scott  01/02/2023 |

**Table of Contents**

[1 About this Document 2](#_Toc126053061)

[1.1 Author Contact Information 2](#_Toc126053062)

[1.2 Engineering Process 2](#_Toc126053063)

[1.3 Supporting Documentation 2](#_Toc126053064)

[2 Problem Definition and Scoping 3](#_Toc126053065)

[2.1 Problem Definition 3](#_Toc126053068)

[2.2 Project Scope 3](#_Toc126053069)

[2.3 Stakeholder Analysis 3](#_Toc126053070)

[2.4 System Requirements 3](#_Toc126053071)

[2.5 Assumptions and Constraints 4](#_Toc126053072)

[2.6 Design Criteria 4](#_Toc126053073)

[3 Initial Prototyping 5](#_Toc126053074)

[3.1 First Prototype 5](#_Toc126053076)

[3.2 Second Prototype 6](#_Toc126053077)

[4 Development and Manufacture 7](#_Toc126053078)

[4.1 Manufacturing 7](#_Toc126053080)

[4.2 Material Cost Breakdown 8](#_Toc126053081)

[4.3 Software Development 9](#_Toc126053082)

[5 Prototype Operation 10](#_Toc126053083)

[5.1 For Demonstration 10](#_Toc126053085)

[5.2 For Maintenance 11](#_Toc126053091)

[6 Further Work 12](#_Toc126053093)

[6.1 Upgrading the Existing Prototype 12](#_Toc126053095)

[6.2 Future Prototypes 12](#_Toc126053096)

1. About this Document
   1. Author Contact Information

Table 1: Author contact information

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Organisation** | **Phone** | **Email** |
| Thomas Scott | Australian National University | 0408 326 878 | thomas.scott@anu.edu.au |

* 1. Engineering Process

This project was planned and executed as a systems engineering project following the Australian National University’s system engineering model, and as such the structure and contents of this document are reflective of the approach taken.

* 1. Supporting Documentation

**Document Title:** ProjectDiary

**Location:** 12thLevel Science & Tech SharePoint > Documents > Projects > SmartGate

**Description:** Journal kept by Thomas Scott as part of his internship

**Document Title:** Nano Datasheet.pdf

**Location:** 12thLevel Science & Tech SharePoint > Documents > Projects > SmartGate > Smart Gate Diagrams and Datasheets

**Description:** Datasheet for the NVIDIA Jetson Nano

**Document Title:** Nano User Guide.pdf

**Location:** 12thLevel Science & Tech SharePoint > Documents > Projects > SmartGate > Smart Gate Diagrams and Datasheets

**Description:** User guide for the NVIDIA Jetson Nano

**Document Title:** Nano Pin map.pdf

**Location:** 12thLevel Science & Tech SharePoint > Documents > Projects > SmartGate > Smart Gate Diagrams and Datasheets

**Description:** Pin map for the NVIDIA Jetson Nano

**Document Title:** Smart Gate Fusion360 Files

**Location:** 12thLevel Science & Tech SharePoint > Documents > Projects > SmartGate

**Description:** All CAD files relevant for the Smart Gate (stabilizing feet, acrylic plate, Jetson Nano mounting plate)

1. Problem Definition and Scoping
3. 1. Problem Definition

Australia is one of the most biodiverse nations in the world, however this biodiversity is under constant threat. New South Wales is home to numerous small mammal species, such as bettongs and quolls, that are bordering on extinction due to the introduced predators such as foxes and feral cats. To combat this, animal sanctuaries are being established in New South Wales, however, the fencing technology and approach is still relatively primitive. The current strategy is to create an impenetrable barrier enclosing an area inside which the protected animals are able to live, free of predators. While this strategy may be effective at conserving endangered species, it can have adverse effects on the surrounding region. For example, without access to the protected species the predators will adapt to a new prey, potentially creating new endangered species and furthering the problem.

We are aiming to create a product that revolutionizes the purpose and operation of animal sanctuary fencing, transforming them from a barrier into a filter, allowing the protected and neutral species to traverse the region freely (or with some control as determined by the user), whilst still creating a safe haven from the predatory species.

* 1. Project Scope

To develop a functional prototype as a proof of concept for a Smart Gate; a device which can be installed into existing and future animal sanctuary fences leveraging computer vision to detect the species of nearby animals and operate a gate that controls access into and out of the sanctuary.

There are four primary considerations that underpin all design decisions for this system. They are:

1. Performance – The ability of the device to identify the species accurately and quickly, then display the appropriate output
2. Power – The power requirements of the system and necessary infrastructure
3. Portability – The
4. Cost – The material, manufacture, and operating costs of the device
   1. Stakeholder Analysis
   2. System Requirements

From the problem definition and project scope (Sections 2.1 and 2.2), the following list of system requirements were extracted:

Table 2: System requirements

| **ID** | **System Requirement** | **Considerations** |
| --- | --- | --- |
| R1 | The system is able to detect and categorise native Australian animals. | 1 |
| R2 | The system is able to allow animal movement through a barrier. | 1 |
| R3 | The system is able to prevent animal movement through a barrier. | 1 |
| R4 | The system is able to trigger an output conditional on the input. | 1, 2 |
| R5 | The system is able to send collected data to a centralized hub. | 1, 3 |
| R6 | The system is able to receive updates remotely. | 1, 3 |
| R7 | The system is able to be controlled remotely. | 1, 3 |
| R8 | The system is operable independent of human interference. | 1, 2, 3 |
| R9 | The system has a low power consumption. | 2 |
| R10 | The system is operable in all weather conditions typical of Australia. | 3 |
| R11 | The system is able to be integrated with existing infrastructure. | 3, 4 |
| R12 | The system is able to be installed quickly. | 3, 4 |
| R13 | The system is inexpensive to manufacture and assemble. | 4 |

* 1. Assumptions and Constraints
  2. Design Criteria

The

Table 3: Design criteria

| ID | Design Criteria | Units | Min | Ideal | Max | System Requirements |
| --- | --- | --- | --- | --- | --- | --- |
| DC1 | Power consumption | W |  |  |  | R1 – R9 |
| DC2 | Detection confidence | % |  |  |  | R1, R4, R8, R9 |
| DC3 | Upfront cost | $ |  |  |  | R11, R13 |
| DC4 | Operating cost | $ /year |  |  |  | R5 – R8, R11 |
| DC5 | Mass | kg |  |  |  | R12, R13 |
| DC6 | Communication capabilities | - |  |  |  | R5 – R7, R9, R11 |
| DC7 | Communication range | km |  |  |  | R5 – R7, R9, R11 |
| DC8 |  |  |  |  |  |  |

1. Initial Prototyping
2. 1. First Prototype

Graphical user interface

Description automatically generated with low confidence

Figure 1: Smart Gate concept sketch

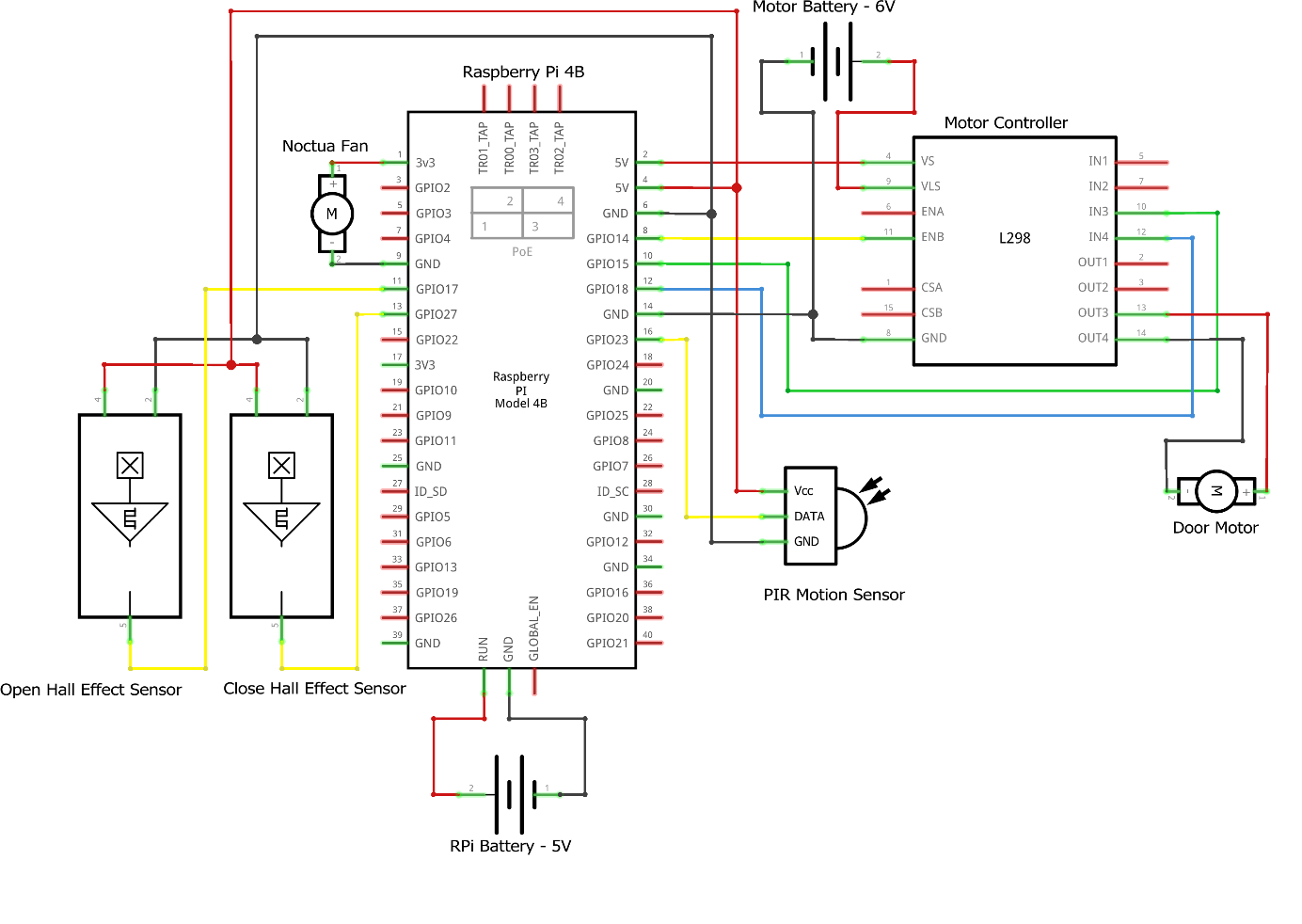


Figure 2: Smart Gate electrical wiring schematic

The first prototype was successful in meeting the essential desired functionally of the system and was demonstrated to both Australia’s Chief Scientist Dr Kathy Foley as well as Prof. Adrian Manning from the ANU.

* 1. Second Prototype

Whilst the first prototype was successful in demonstrating the essential functionality of the system, it was clear that the Raspberry Pi was barely keeping up with the workload and was generating a significant amount of heat despite the heatsinks and fan operating at full speed. Additionally, we were forced to run the CPU variant of PyTorch as the Raspberry Pi does not feature any dedicated GPU cores, which significantly bottlenecked the computer vision algorithm.

It had always been intended that the system would deviate from the Raspberry Pi at some point, but it became apparent that changing the Raspberry Pi was an urgent activity. After conducting some research, it was determined that the NVIDIA Jetson Nano was the ideal replacement, as it featured CUDA 10.2 support with dedicated GPU cores for accelerated AI performance while still maintaining the 15W operating power of the Raspberry Pi and superior thermal performance at a similar price.

However, the Jetson Nano is substantially larger than the Raspberry Pi and necessitated a larger control box. It was then determined that the new larger control box should house the componentry from the previous two and should be mounted lower to the ground to improve the stability of the device. Those changed resulted in the following concept sketch.

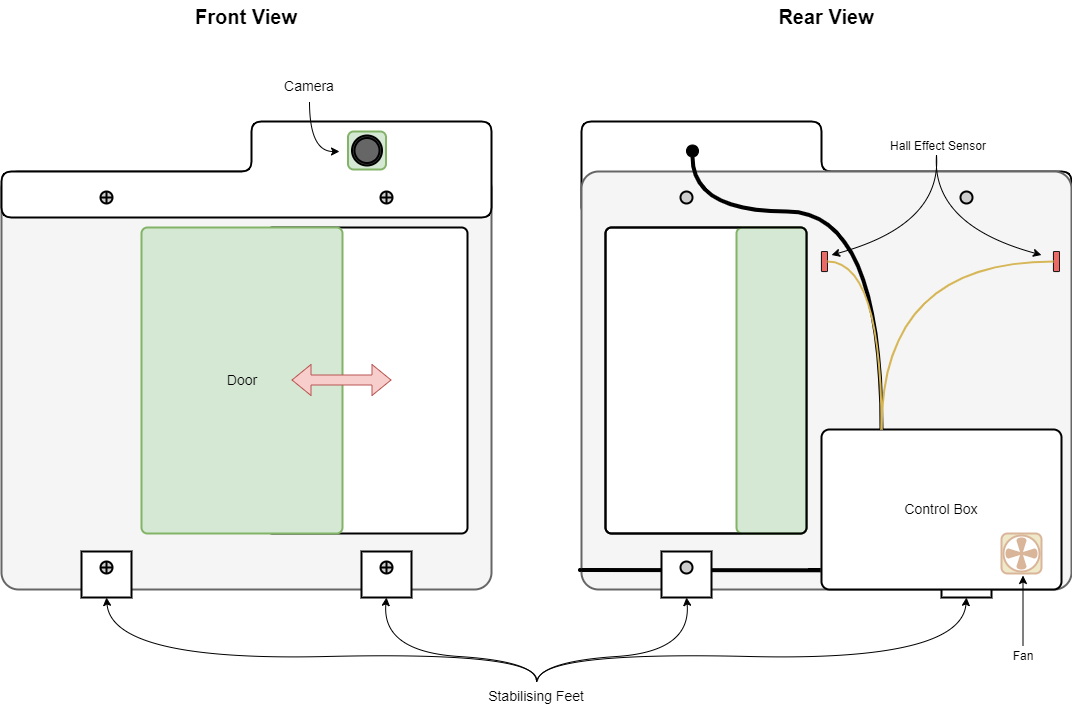


Figure 3: Smart Gate V2 concept sketch

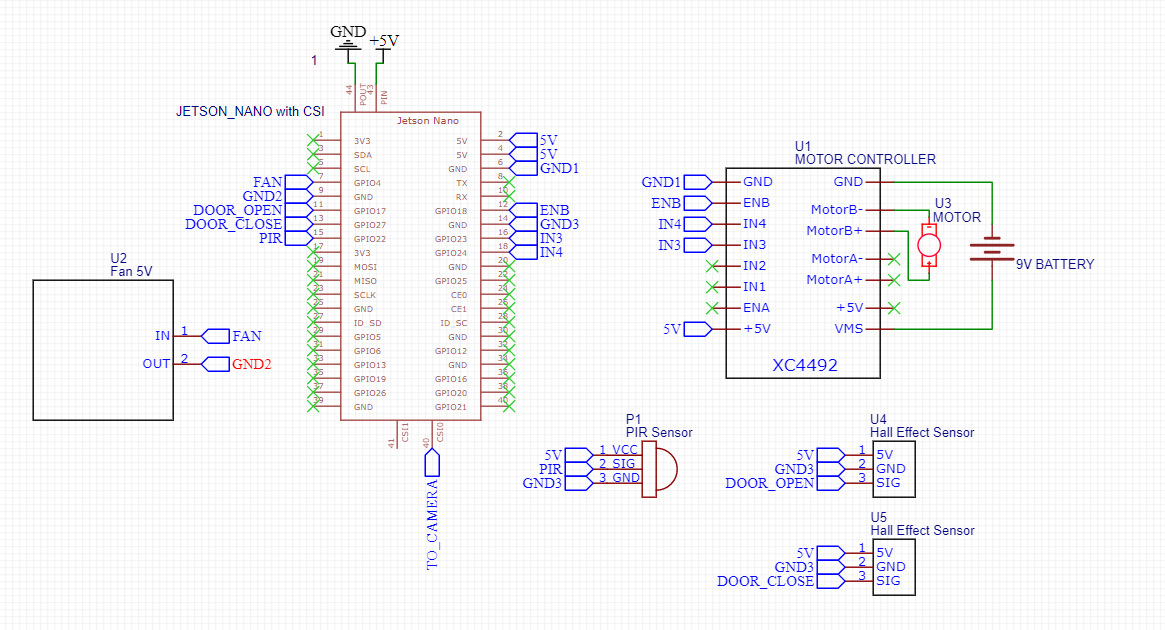


Figure 4: Smart Gate V2 wiring diagram

**Note**: This diagram does not include the jumper which must be applied to the Autostart and GND pins on the rear of the Jetson Nano and the connection of the two red power switch cables to the PWR\_BTN and GND pins on the rear of the Jetson Nano.

The redesigned control box also allowed for some other improvements. Previously, the battery was housed underneath the Raspberry Pi, which made removing the battery for charging a rather strenuous task as all of the wires needed to be removed from the Raspberry Pi. With a larger box, the battery is now able to be mounted to the lid of the box, easily within reach once the lid is removed.

A few minor alterations were made during the manufacture process. Firstly, as the rear reinforcing acrylic sheet was going to be recut to help support the new heavier control box, it was decided to mount the camera and PIR sensor to the new reinforcing plate. This should improve the rigidity of the system and reduced the complexity of manufacture. Secondly, the fan was moved from the back side of the smart gate to the side, to better channel the fan’s airflow through the heatsink fins of the Jetson Nano.

1. Development and Manufacture

The manufacture and software development were done in tandem for the creation of the Smart Gate as many of the decisions for the physical design were dictated by the software requirements and vice versa.

* 1. Manufacturing

All manufacturing activities were completed at the ANU Makerspace during regular opening hours due to the availability and proximity of various tools and ease of access for me as an ANU student. There were 5 main manufacturing activities that were required for this project:

1. Laser cutting (reinforcing plate)
2. 3D printing (feet)
3. Soldering (wires, power switch)
4. Drilling (mounting holes, airflow holes)
5. General assembly (epoxy, screws)

All laser cutting was performed on a Trotec Speedy 360 with 4.5mm clear acrylic from Plastic Creations in Fyshwick. I chose 4.5mm acrylic due to the additional rigidity it could provide as the reinforcing plate. 3D printing was done using a Prusa i3 MK3S+ with Prusament 1.75mm PLA filament. CAD files for all 3D printing and laser cutting can be found on the Science & Technology SharePoint (see section 1.3).

* 1. Material Cost Breakdown

Table 4: Material cost breakdown

| **Item** | **Description** | **Performance Criteria** | **Cost** |
| --- | --- | --- | --- |
| NVIDIA Jetson Nano Developer Kit | System computer to house and run the object recognition model, control I/O |  | $432.56 |
| Omlet automatic chicken coop door | Door to act as system output |  | $279.99 |
| Raspberry Pi HQ Camera Board | Camera module to capture images and video for analysis |  | $85.00 |
| 6mm wide-angle lens | Lens for camera module |  | $42.00 |
| Camera cable (610mm) | Cable for connecting camera to Raspberry Pi |  | $5.95 |
| USB battery pack | Battery to supply system without requiring mains power |  | $59.95 |
| 32 GB micro SDHC card | To house operating system for Jetson Nano |  | $22.95 |
| SPST Toggle Switch | For turning on Jetson Nano |  | $7.95 |
| Passive IR Sensor | For detecting movement to trigger camera |  | $9.95 |
| Control box | Housing for computer + battery |  | $42.95 |
| Jumper leads | For connecting electrical components |  | $14.95 |
| Heat shrink | For insulating electrical connections |  | $19.95 |
| WH3040 Red wire | For connecting electrical components |  | $1.40 |
| WH3041 Black wire | For connecting electrical components |  | $1.40 |
| Acrylic sheet | For mounting electrical components and providing stability |  | $50.00 |
| 9V Battery clip | To connect 9V battery to motor controller |  | $2.90 |
| USB 3.0 type A plug | To connect to battery |  | $7.90 |
| Fused cigarette lighter plug to 2.1mm DC plug | To connect to Jetson Nano |  | $8.95 |
| Noctua 5V Fan | For cooling the Raspberry Pi |  | $59.35 |
| M2.5 Standoff kit | For attaching various components |  | $42.65 |
| Velcro strips | For mounting components within control box |  | $7.90 |
| LN298 motor controller board | For controlling the door motor |  | $14.95 |
| Hall effect sensors | For detecting when the door is open/closed |  | $9.90 |
| Epoxy Glue | For attaching the hall effect sensors and magnets to the door |  | $8.40 |
| Ferrite Magnets | For signaling when the door is open/closed |  | $4.95 |

Total: $1244.80

* 1. Software Development

The NVIDIA Jetson Nano is built to run with Ubuntu and officially supports up to Ubuntu 18.04. NVIDIA provides a complete image file called the NVIDIA JetPack SDK that includes many preinstalled packages (such as CUDA 10.2) which are useful for AI applications. The Jetpack SDK is available here:

<https://developer.nvidia.com/embedded/jetpack-sdk-461>

Note: Jetpack 5.0 and later do not support the Jetson Nano. I chose to use Jetpack 4.6.

I have also uploaded the Jetpack SDK image that I used to the Science & Tech SharePoint. The image file was flashed to the micro-SD card before insertion into the Jetson Nano.

I then set up the Jetson Nano with the following user details:

**Username:** smartgate

**Password:** bioquai

With the Jetson Nano set up, I then began installing the necessary packages for the YOLOv5 package by following the tutorial found at this GitHub address:

<https://github.com/ultralytics/yolov5/issues/9627>

It should be noted that the Jetson Nano does not come with any wireless capabilities by default and requires a wired ethernet connection. This is likely a concern for future prototype iterations (see section 6.1). Once an internet connection is established, it then takes several hours for all of the necessary wheel files to be built and installed, so this tutorial was completed over a number of days. I am unsure if it would be possible to speed this process up, perhaps by creating our own image file with all of the necessary libraries installed that we could simply upload to the micro-SD card.

Additionally, I installed VS Studio Code as a better Python IDE.

To enable the GPIO pins on the Jetson Nano, I followed this tutorial:

<https://pypi.org/project/Jetson.GPIO/>

Then, when it came time to test the device, I received an error message saying that there were no cameras detected. I came to realise that this was because the Jetson Nano does not natively support the sensor in the Raspberry Pi HQ Camera and furthermore the Jetson uses a 1.8V reset signal, whilst the camera was designed for a 3.3V reset signal, which trapped the camera in an endless reset cycle. More information can be found here:

<https://www.hackster.io/SaadTiwana/embedded-diaries-how-to-use-rpi-hq-camera-with-jetson-e2063e>

The result of this discovery is that the Raspberry Pi HQ Camera is not suitable for out use without physical modification which could potentially damage the camera and is infeasible for mass-production. To rectify the problem, I have decided to switch to the **Raspberry Pi Camera V2**, which is natively supported by the Jetson Nano. This change will come at a cost of some image quality and customizability in terms of lens choice, but ultimately saves a lot of time and unnecessary risk. It is highly recommended that more research is done into finding a Jetson Nano compatible CSI-2 camera (see section 6.1).

Download the models from:

<https://github.com/carlosclaiton/marsupial>

As we are using jetpack 4.6 the most updated available python version is 3.6 then it requires to use

Torch 1.10.1 and torchvision 0.11.1

Then I have create a conda environment:

**>>> conda create -n smartgatepy36 python=3.6**

**>>> pip install pandas numpy**

**>>> pip install torch==1.10.1**

**>>> pip install torchvision==0.11.1**

Note that we also need to install the opencv-python to use torch, and here is the command I get it working:  
**>>> sudo apt-get install python3-opencv**

* 1. Known Issues

After configuring the Raspberry Pi Camera V2 and verifying its correct operation, it was time to incorporate the machine learning model. I began by using the steps from this guide:

<https://github.com/ultralytics/yolov5/issues/36>

However, I would receive a warning saying that YOLOv5 requires Python >= 3.7, whereas we only have Python 3.6.9 installed. It appears that this was changed sometime last year when Python 3.6 became no longer supported. Despite the warning, the program continued, but got stuck at the ‘Fusing Layers’ step, where the entire Jetson Nano froze. I suspect this is due to the entire memory being used up, which may not have been an issue for the Raspberry Pi because of its larger 8 GB memory capacity compared to the Jetson’s 4 GB. However, I have read online of people who are able to perform inference with only 2 GB of memory, so I know that it must be possible. Perhaps upgrading to Python 3.7 will fix the issue, but I have not had time to explore this as it would require the rebuilding of all dependencies (OpenCV, torch, torchvision, etc.) which takes several hours.

<https://github.com/ultralytics/yolov5/issues/7631>

<https://github.com/ultralytics/yolov5/commit/a19406b39dbc45db0bbae8d0b7da9d6281f9af1e>

The above links are to a forum discussion and related Git commit wherein the Python requirement was changed from 3.6.2 to 3.7.0, and it appears that someone has had success in simply ignoring this commit, but would necessitate the use of local models (which we would be wanting to do anyway so that we are not dependent on the internet), however I run into this issue when attempting to do so:

[Errno 2] No such file or directory: ‘~/yolov5/hubconf.py’

I have looked and the hubconf.py file appears to be in the correct location, so I suspect there is some deeper issue afoot. Maybe the model needs to be run at least once from the internet so that it becomes stored in cache or is downloaded properly? Unfortunately, I don’t think my Python or Linux troubleshooting skills are sufficient to diagnose this problem properly.

I found the following GitHub repo which appears to be in some way related to using YOLOv5 on the Jetson Nano, but I was unsure how to implement this.

<https://github.com/fei4xu/yolov5-python3.6.9-jetson>

Furthermore, I have tried using the GPIO pins on the Jetson to control the door, but I have not been able to get them to do anything. I have installed the Jetson.GPIO library but it does not appear to work without additional permissions. I have looked online but haven’t found the exact solution yet, although this might be solvable with the use of Conda.

Finally, once these problems are solved and the program ‘liveObjectDetection.py’ is working properly, the Jetson will need to be set up so that the Python program runs on startup. I achieved this last time on the Raspberry Pi using crontab, and I assume this approach will apply to the Jetson as well, but I cannot be sure.

1. Prototype Operation
2. 1. For Demonstration

Demonstrating the functionality of the prototype is relatively simple and requires minimal technical knowledge regarding the inner workings of the device.

* + 1. Before the demonstration

1. Prepare the presentation space. The device displays the best performance when the targets are displayed approximately 1 metre from the camera in a room with minimal other objects that could influence detection.
2. Replace the 9V battery with a new one in the control box to ensure best gate motor performance.
3. Ensure the battery is connected to the Jetson Nano by removing the back panel. The battery should be attached to the back panel (via Velcro) and plugged into the Jetson Nano with the USB to barrel plug cable. Reinstall the panel once you are confident the battery is plugged in correctly.
4. Locate the switch on the back of the Smart Gate. Hold the switch forward for approximately one second. The fan should begin to spin up.
5. Remove the camera lens cap.
   * 1. Demonstrating the device:
6. The demonstration can begin with the door either open or closed, however the preference is to begin with an open door as the final product is expected to be open by default. The remainder of this guide will assume that the door begins in the open position.
7. Give background to the project and that this is strictly a proof-of-concept, meaning there will be significant changes made before being used in the field.
8. Explain the two input objects (a sign of a dog and a sign of a cat at the point of writing this).
9. While the door is in the open position, hold the dog sign in front of the camera. The door will not move (demonstrating that the door will stay open for the positive input).
10. Then remove the dog sign from frame and instead hold up the cat sign. The door will close (demonstrating that the door will react to the negative input).
11. Remove the cat sign from frame and instead hold up the dog sign. The door will begin to open (demonstrating the door will react to the positive input). Once the door is approximately halfway open, bring the cat sign back into frame (so that both signs are in frame). The door will stop opening and then begin to close again (demonstrating the door prioritizing the safety of the sanctuary).
    * 1. After the demonstration:
12. Replace the camera lens cap.
13. Turn off the Smart Gate by holding the power switch forward for approximately 10 seconds.
14. Remove the battery for charging (see section 5.2.1).


18. 2. For Maintenance
    3. 1. Charging the battery

Compared to the first prototype removing and charging the battery is a simple exercise.

1. Open the back of the large control box by unscrewing the 6 Phillips head screws.
2. Unplug the USB to barrel plug cable from the battery and remove the battery from the back panel (simply pull to remove as it is affixed with Velcro).
3. Plug the battery into a charging port with any micro-USB cable. The LEDs on the front of the battery should begin blinking to indicate that the unit is charging.
   * 1. Focusing the lens (RETIRED WITH CAMERA CHANGE)

Manual focusing the lens of the camera requires access to the camera output, which is not currently possible without removing the Raspberry Pi from the control box to gain access to the mini-HDMI port. However, there are two grub screws located visibly on the outside of the camera lens for the purpose of fixing the lens in place and through experimentation it was found that the configuration depicted in Figure 3 yields an optimal focus.



Figure 5: Camera grub screw positioning for focusing

* + 1. Programming the Jetson Nano

For a number of reasons, it may be necessary to reprogram the Jetson Nano. For example, you may want to change which objects act as the stimuli, incorporate a new output mechanism, or perform some testing. Unfortunately, there is no simple way to access the Jetson Nano without some disassembly in the prototype’s current form, however there is potential to make this process simpler in future iterations (see section 6.1).

The first method is similar to the process for charging the battery, in that it requires significant disassembly. Thus, this method is also quite slow when prototyping or testing as the gate must be fully disassembled and reassembled for every change.

1. Open the back of the large control box using a Phillips head screwdriver, being careful not to let the lid drop to minimise the risk of damaging the fan cables.
2. Disconnect all jumper wires connected to the Raspberry Pi and remove them from the control box through the side hole (one at a time to avoid damage).
3. Disconnect the camera ribbon cable by prying up on the black plastic clamp. Remove the cable from the control box through the side hole.
4. Remove the red mounting plate from the control box by carefully pulling on the cutouts.

The Raspberry Pi is then removed and free to be connected to a keyboard, mouse, and monitor for reprogramming. To reassemble, follow steps 1 through 5 in reverse, making sure to reconnect jumper wires as per Figure 2.

Alternatively, it is possible to gain access to the micro-HDMI and USB ports without disconnecting the Raspberry Pi from the surrounds, making it possible to test with the rest of the Smart Gate in real time. However, this method is quite precarious, risking damage to the Raspberry Pi and all electrical connections made to it.

1. Open the back of the large control box using a Phillips head screwdriver, being careful not to let the lid drop to minimise the risk of damaging the fan cables.
2. Support the lid of the control box whilst keeping all cables connected.
3. Unscrew the black nylon standoffs from the Raspberry Pi.
4. Tilt the Raspberry Pi forward such the relevant ports are exposed and usable, then make connections as needed.
5. Screw the standoffs back on to keep the Raspberry Pi in place.
6. Further Work

There are multiple avenues for further work on the Smart Gate as it begins to transition from a proof of concept to a field deployable prototype or even commercially viable product. There are numerous non-engineering tasks that will be required; however, I do not know the full extent of these tasks so I will only discuss further work relating to the development of the device.

1. 1. Upgrading the Existing Prototype

The current (second) prototype is able to demonstrate the essential functionality of the Smart Gate, but there is the potential to add a wide variety of additional functions for testing in a controlled environment. These could include:

1. A better camera – the second prototype utilizes the Raspberry Pi Camera V2 (after the realization that the Raspberry Pi HQ Camera was not supported by the Jetson Nano), which is a relatively low-cost and low-quality camera. It is recommended that a better (but still Jetson Nano-compatible) camera is purchased and integrated for improved computer vision performance.
2. Wireless communications – a critical function of the eventual final Smart Gate will be the ability to communicate with the gate wirelessly, removing the need for a ranger or similar person to go to the gate to check on it. To my knowledge the Jetson Nano does not come equipped with any wireless capabilities out of the box, but there are external devices that can be used for any kind of wireless communication protocol chosen. This could allow for the stimuli objects to be changed remotely or perhaps to display the camera feed to an external monitor for demonstration purposes.
3. Data capture – the prototype currently does not store any data that is gathered during operation. This could likely be solved relatively easily by writing to an external SD card or transmitting to a second computer wirelessly.
   1. Future Prototypes

While significant thought and energy has gone into the development of the two prototypes thus far, they were only ever intended as demonstration tools for attracting attention and investment, as well as proving the feasibility of a portable computer vision system. In order to create a field deployable device, there is a significant amount of further development required.

The most visibly obvious progression is to create a field deployable gate/door. The current door is an automatic chicken coop door by Omlet which utilizes a horizontal rack and pinion system. Future doors should consider the following.

1. The door will likely need a wider opening to encourage animals to travel through the door more. This may be an area for consultation with ecologists.
2. The door will need to be able to close in a significantly quicker manner than it can currently.
3. The door will need some method to prevent animals from breaking through the door (such as a locking mechanism, cementing beneath the door, etc.)
   1. Recommended Skillsets

With the previous suggestions in mind, there are a number of skillsets that I believe to be necessary to progress the development of the Smart Gate which I either did not possess or did not have time to put into effect. I would recommend looking for the following skillsets when bringing new people onto the project.

1. Linux experience – for this project I have had to teach myself Linux (Ubuntu) to program and control both the Raspberry Pi and Jetson Nano. However, as I have been working on all aspects of the project, I have only been able to invest enough time to get a surface level understanding of Linux, enough to be able to program the Smart Gate to perform the essential functionality. To progress the Smart Gate, I believe a deeper understanding of Linux will be required, particularly regarding
2. Mechanical design and manufacturing – the two prototypes I have developed are using relatively simple manufacturing techniques due to the performance goals of the device. My prototypes were never intended to be deployed in the field, so the device is not weatherproof, nor could it stop an animal from breaking through the door.
3. Wireless communications –
4. (Remote) power systems – the first two prototypes have both relied on a portable battery pack and standard Duracell batteries for power, which are obviously limited in their capacity and operating lifespan. For a field deployable device, there needs to be some kind of reliable power supply (solar, power along the fence, etc.). Someone with power systems experience would be ideal for this task.

**To do list:**

run liveObjectDetection.py on the boot

Get I/O pins working