ECE 492/592: Flying Cows/Camera Trap (A14) Final Report

Due: May 2, 2023

Team Members: Andy Cheung, Connie Zhang, Cassidy Petrykowski, Eshan Parikh, Ilena Johnson, Jacky Hong, & Michael Mathew

Abstract

Many unique animals can only be found in remote areas, far from human interference. Therefore, having a completely autonomous camera trap that can take pictures of animals without the presence of humans can provide efficient and safe means for animal documentation. In order for the camera trap system to be completely autonomous, the camera trap needs to be powered with a reliable source for daytime and nighttime use. The camera trap also needs to be durable under all sorts of weather conditions. As part of our solution, a drone would be used to communicate with the camera trap to retrieve the photos, so that the camera trap's data could still be collected from extremely remote locations. Our team used a drone and an embedded system built with an ESP32-CAM and an IR sensor to capture images and transfer the data. Our drone successfully found and received images from the camera trap, and our camera trap was weatherproof, lasted for four days without human interference

Introduction

Camera traps are often used to study animal behavior in remote locations, for example in the canopy of the Amazon rainforest. This project addresses the need for a low power, minimal maintenance mechanism to take data in remote environments for study and research. Our team's objective was to design a camera trap with WiFi capability to reduce the need for humans to retrieve the image. Our camera trap also needed to be able to survive without loss of power for long periods of time while still being able to work at night. To achieve this, we used a solar panel to recharge a battery pack that powers the camera trap.

A long-term solution to track animals and/or the general movement in a remote environment that is not easily accessible to humans can be achieved using a camera trap built on an embedded system and a drone. The camera trap wakes up when the motion sensor is triggered and snaps a photo, and then goes back to sleep. The camera trap also wakes up when the drone pings it, and sends the image(s) data to the drone. The drone automatically goes to the predetermined GPS location of the camera trap and pings the camera trap to see if it has taken any images. The drone then retrieves images from the camera trap if there are any, and automatically returns back to base using RTL mode.

Our solution provides an answer to the problem, as the camera trap is extremely low power, self perpetuating during stretches of poor weather and nighttime while also taking photos upon motion detection while being waterproof and within a reasonable scale. The drone is able to communicate with the camera trap and receive the photos stored on the SD card for the camera trap, which wakes up every 10 seconds to seek the drone. Possible applications of our solution involve remote forest or mountain locations in a variety of climates, given the weather proofing, ability to run on low power for long periods of time and size of the project.

Related Work

Camera traps are commonly used by wildlife researchers and hunters alike, and there are many camera traps commercially available. One such product is currently being developed by Arribada Initiative [1] for use in the Amazon rainforest. Arribada Initiative's camera trap is a network of cameras placed in trees in the Amazon rainforest canopy, running on solar power and communicating over Wi-Fi. Although we didn't reference any of their design documents, Arribada's work shows that our project is commercially viable.

Another project we initially looked at was a DIY/citizen science type work-in-progress design from Hackaday [2]. The important takeaway from their discussion was that a low-power design capable of surviving in the field for any extended period of time required a microcontroller, not a Raspberry Pi. Although our power

requirements are different – the Hackaday project does not include solar power – this was an important piece of evidence to support our decision to use a Raspberry Pi.

Finally, the most notable related work is the motion detector with photo capture project at Random Nerd Tutorials by Sara and Rui Santos [3]. This project heavily influenced our choice of microcontroller, and provided us with starter code and circuitry for motion detection. The idea to use the Arduino framework to reduce code development time proved to be extremely helpful.

Our Approach

After reading the report left to us from the previous camera trap group, the team started the entire project from scratch and in an effort to divide and conquer, split the work between two teams, one focused on the embedded camera trap system (referred to henceforth as "the camera trap," and one focused on the drone. Both teams worked on their own hardware and software and divided tasks amongst subteam members, but there was cross collaboration, especially on the communications between the drone team and the camera trap team.

Camera Trap

The camera trap team was responsible for building an embedded system capable of functioning for a long period of time (four days without sun) that could take images upon motion activation and then transfer those images to a drone for collection. The camera trap team approached our problem with a focus on low power, as we realized after reading the previous team's report that this was a major concern and one that had to be addressed from the start rather than factored in after designing other elements of the system. We set up a power system using a DGDGGGD device, the Wanderer, which directed the power from the solar panel to the battery and the power from the battery to a 5V2A output which we used to power the circuit. The circuit was built using an ESP32-CAM and a PIR sensor (for motion detection; the circuit layout is shown in figure 1, with the actual physically assembled circuit shown in figure 2.

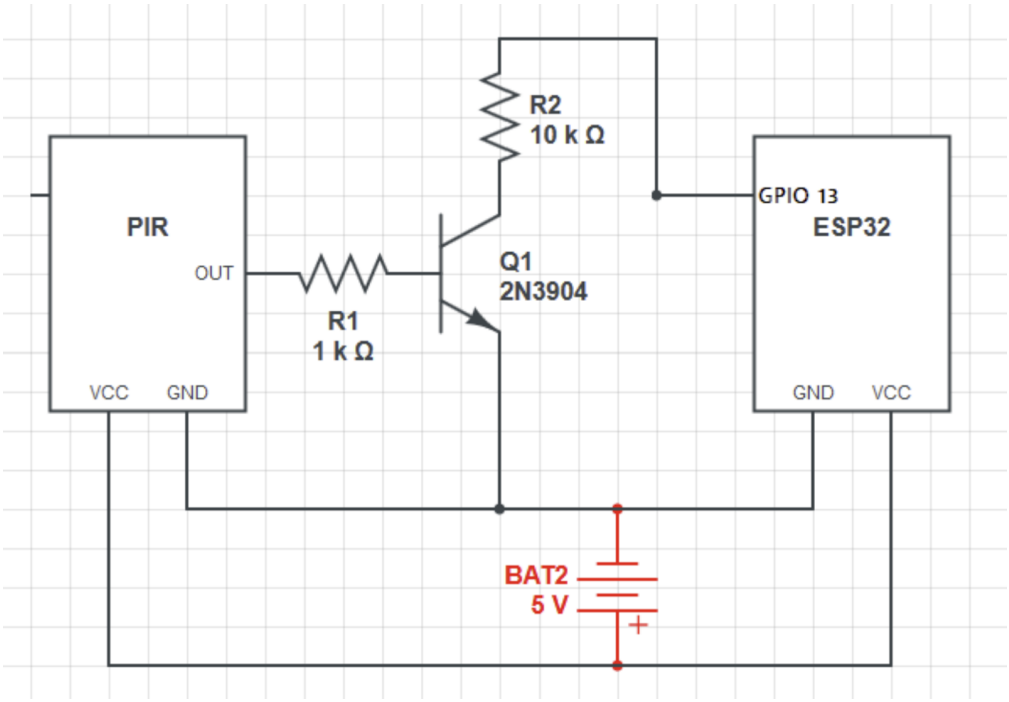

Figure 1 - Circuit diagram [3]

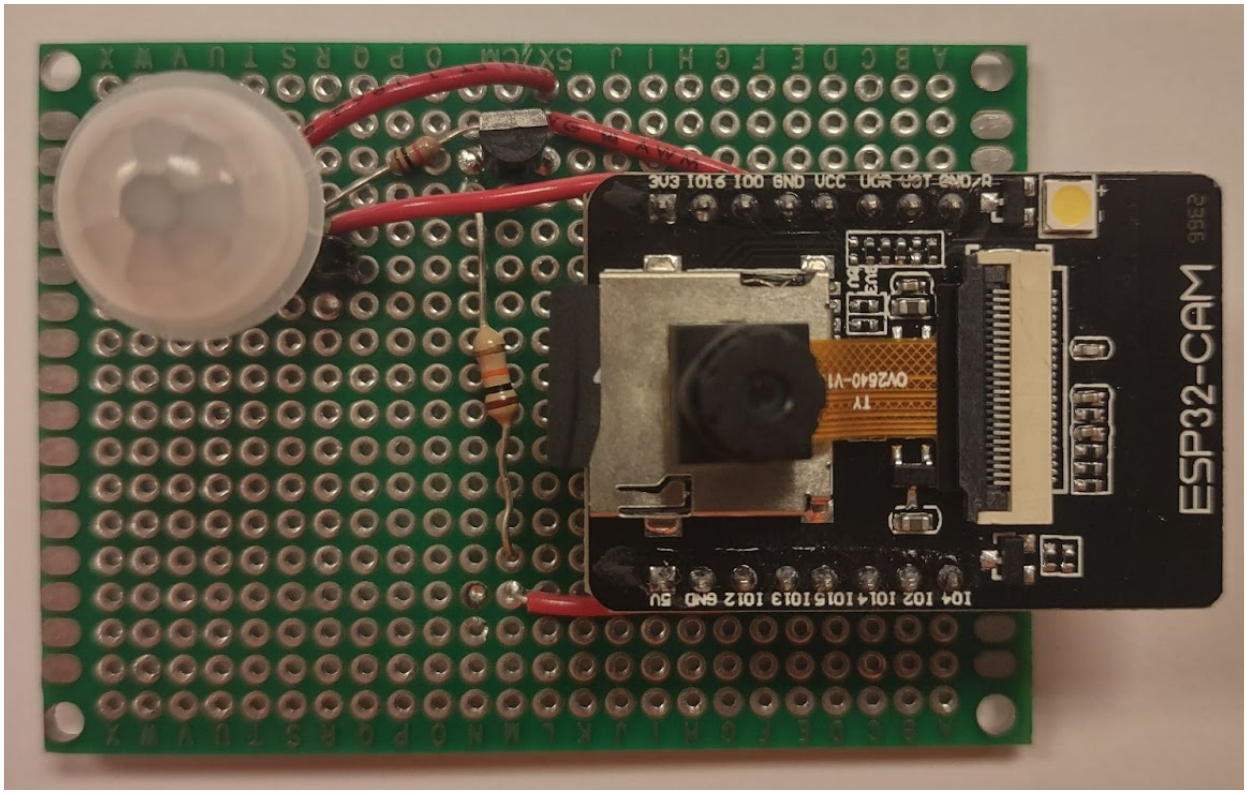

Figure 2 - Protoboarded motion detection and camera circuit.

One of the major design considerations was the choice of on-board computer for the camera trap. The previous design used a Raspberry Pi Zero W. A Raspberry Pi based design provides significant advantages: Raspberry Pis are well-supported and well-documented; there are cameras that integrate effortlessly with the Pi; running Linux and developing in Python reduces the overhead of writing low-level embedded code, allowing the team's resources to be allocated to solving more high-level problems. However, any Raspberry Pi, even the Zero and Pico, draws significantly more current than a microcontroller even in deep-sleep mode. The ideal compromise between low power and ease of use proved to be the ESP32-CAM programmed using the Arduino framework. The ESP32-CAM is an ESP32 microcontroller, which is known for its low power consumption, with an added camera and SD card interface. Programming it with the Arduino framework allowed us to use prebuilt libraries, reducing software complexity on our end while still allowing us access to deep-sleep modes and interrupts.

On the software side, the camera trap team had two main objectives beyond requiring as little power as possible. The first was to wake up upon being activated by the IR sensor, and the second was to wake up and transfer images to the drone. The two functionalities were programmed separately and then integrated into the setup look in Arduino. In order to achieve the lowest power consumption, the ESP32-CAM actually stays in deep sleep, meaning that each time it wakes up it reruns the setup and based on the reason for the wakeup, a different function is run, either communicating with the drone or taking a photo. Motion detection from the IR sensor activates a GPIO interrupt whereas, for drone communication the ESP32-CAM actually wakes up every ten seconds and seeks the drone before going back to deep sleep. If the ESP32 connects with the drone, it transfers the images before returning to sleep.

Drone

The drone team was responsible for creating a drone payload and writing code that was capable of flying to the GPS location of the camera trap and attempting to make a connection with the camera trap. If the connection was

successful, the drone would accept the transmission of images from the camera trap and then return to the base; if the connection was unsuccessful, the drone would attempt to connect for a total of 60 seconds before using the RTL function to return to the launch site. The drone team approached our problem by building a straightforward payload using a Raspberry Pi and a power step down (from ~27-30V to 5V) that connected to the orange cube and the power supply.

On the software side the drone script was built on the same structure as our original homework script, SquareOff.py and is contained in the `drone_mission.py` script. The drone script begins by setting up several constants and variables, including the target altitude for takeoff, a threshold for breaking from the takeoff loop, a maximum distance from waypoints, a socket server port, a buffer size for receiving images, and a server timeout.

The drone code then creates a list of GPS coordinates (`coordinates_list`) for the drone to visit. Each coordinate is specified by a latitude, longitude, and altitude. The script connects to the drone using a specified `connection_string` and sets the target altitude for takeoff (`TARGET_ALTITUDE`) to 20 meters. Once the drone is armed and in guided mode, it takes off to the target altitude using the `vehicle.simple_takeoff()` method. The drone then waits until it reaches 95% of the target altitude before proceeding to the next waypoint in the `coordinates_list`. For each waypoint, the drone moves to the location using `vehicle.simple_goto()` and hovers at that location until it receives images from a client via a socket server that is started at the drone's current location. Once a client connects, the server receives image data in chunks of size `BUFFER_SIZE` until a timeout or no data is received. The received data is then split into segments delimited by the bytes `b"BEGIN"` and `b"DONE"`, and the image data between `b"BEGIN"` and `b"DONE"` is extracted and saved as a file in the `images_received` directory. After the drone has received images at each waypoint, it returns to the launch site using `vehicle.mode = VehicleMode("RTL")`. The script waits for the drone to land before closing the vehicle object and exiting.

The drone script was tested first on the payload in conjunction with the camera trap, then on v-drones, and finally on the actual drone itself.

Experimental Results

Contextualized as an experiment, our drone project's main objectives were to test:

- Ability to withstand inclement weather
- Duration that the camera trap can function autonomously
- Types of motion activation that work with camera trap
- Ability to take and send photos from camera trap to drone
    - Photo quality
    - Camera delay
    - Time for photo transmission

Some of the features of the camera trap discovered in testing are shown in table 1.

Feature	Status	Notes
Ability to withstand implement weather	Withstood a rainstorm	Is able to get power from solar panel even in the rain.
Autonomous functionality (able to power itself and maintain ability to take photos)	Lasted alone 4 nights, 5 days	Was fully operational, very likely could last for months at a time barring extreme periods with no sun.
Types of motion activation	(i) Direct motion overhead (person, clouds) (ii) Horizontal motion	Took shots of cloud movement
Photo quality	2 Megapixels	Specifications of OV2640 camera
Time to take photos	Took 4 photos in ten seconds during instructor demonstration, so approximately 2.5 seconds per image taken	Every time an image is taken, the camera trap must wake up from deep sleep, call upon the SD card and activate the camera

Table 1 - Table showing the features of the camera trap identified during testing.

The time for photo transfer is shown in figure 3, and taken from three of our recorded field tests. The data used is the total time for image transfer set against the number of images.

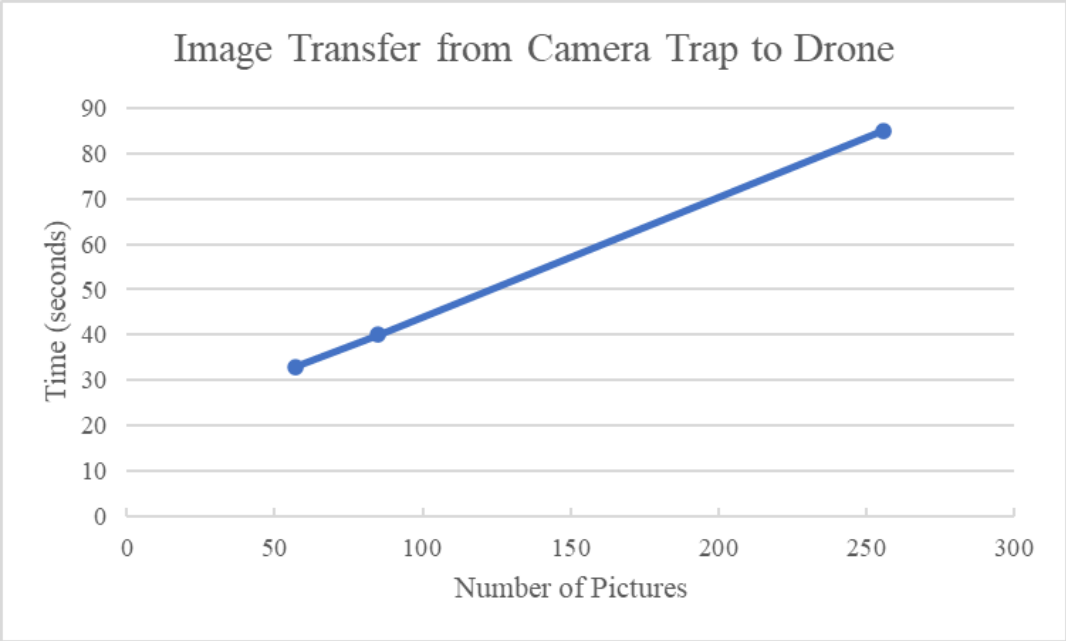

Figure 3 - Graph showing the time to transfer images from the camera trap to the drone.

Analysis

When we approached this project we had three key objectives:

1. Ability to withstand inclement weather and last for long periods of time without power.
2. Take photos upon motion activation.
3. Transfer photos from camera trap to drone.

As shown in the second and third rows of Table 1, the camera trap was able to achieve a low power state that endured through stormy weather and isolation. The quality of the photos taken is shown in row 5 of Table 1, and the different types of motion activation that were shown to successfully activate the IR sensor and camera circuit as shown in row 4 of Table 1, thus satisfying the second objective. The third objective was successful and the rate at which images are taken is discussed in row 6 of Table 1, while the rate at which images are transferred from the camera trap to the drone is shown in figure 3.

Reflecting on the three main objectives, the camera trap/drone approach was certainly successful and the time in which photos are taken is conducive for capturing animals passing by. If an animal passes by, at least one photo will be taken, and if an animal pauses by the trap for a five second duration, two photos will be taken. The camera trap also transfers photos to the drone at a rate of approximately 2.88 seconds per photo, extracted from the data in figure 3.

Conclusion

Through our efforts in this project, we have developed a safer, more efficient way to document and record animal sightings. This was done by attaching a payload, which consisted of a raspberry pi and a powerboard to a drone which opened up a socket communication and received images autonomously. The goal that was achieved with the drone was to fly to the location of the camera and hover low enough to be able to get a proper connection. This would allow us not to have to endanger ourselves and receive images from the camera trap.The camera trap itself had a few goals it had to accomplish on its own. The first goal we desired for the end result was to have an extremely low-power system. We achieved this goal by making decisions to use a significantly lower-power MCU, the ESP32-CAM, and utilize solar power. This was done by using a solar panel which would recharge a battery that in turn, supplied voltage to the MCU. Along with achieving low power consumption, another goal we had was to ensure the camera trap enclosure was water-proof, ensuring that the camera trap could remain outside even given inclement weather conditions. This was achieved by the box used to house the camera trap as it is already weatherproof with a rubber lining. Based on the results, our end result was a camera trap capable of transferring, on average, an image per 3 seconds to the drone itself.

References:

[1] Dangerfield, A. “Building a better camera trap.” *Arribada Initiative,* 11-Mar-2019. https://arribada.org/2019/03/11/building-a-better-camera-trap/ . [Accessed: 02-May-2023].

[2] Ramanauskaite, E. “Camera Traps for Citizen Science.” *Hackaday*, n.d. https://hackaday.io/project/11088-camera-traps-for-citizen-science. [Accessed: 02-May-2023].

[3] Santos, R., & Santos, S. “ESP32-CAM PIR motion detector with photo capture,” *Random Nerd Tutorials*, 30-Jul-2020. [Online]. Available: https://randomnerdtutorials.com/esp32-cam-pir-motion-detector-photo-capture/. [Accessed: 02-May-2023].