Fenceless Grazing Tech Review - Ryan Alder

Danila Fedorin, Matthew Sessions, Ryan Alder

Abstract

One of the most important requirements before starting any large project is the separation of responsibilities between team members. In an effort to limit scope creep and ensure that my responsibilities are obvious to the rest of my team I have defined what I am accountable for in this project. My primary role is to write and implement software and the LoRa connection between the end-nodes (collars) and the gateway. I will be responsible for writing the high level logic software for each collar which will integrate with the drivers for the separate microcontrollers and modules. I will work with Danila to develop the server software which will act as the logic API which interacts with the user in the form of the Android application. Lastly, I will be responsible for implementing the LoRa protocol to connect each collar to the gateway(s). This will include managing connections per second and ensuring scalability for different sizes of fields and number of livestock.

CONTENTS

1 TEAM GOAL

Currently in order to maintain fields and prevent overgrazing, farmers need to constantly move sheep and cattle to different grazing areas. This requires either moving fence lines, or more likely having many different fenced in areas. This is a long and tedious process, especially for the modern day farmer with the technology currently available. This project will attempt to solve this issue by seriously decreasing the amount of time farmers spend moving their flocks to new grazing areas.

The Fenceless Grazing Collar System is an application of the LoRa and LoRaWAN internet of things technologies to animal farming. It will be implemented as a collar with which each individual farm animal will be equipped. The fenceless grazing system must be capable of precisely tracking high numbers of farm animals, controlling these animals using sound and electric shocks, and allowing end users (typically farmers) to monitor the animals and adjust their desired location. This project will be implemented via collars on each animal which communicates with a main server. Commands are sent to the server from an Android application, and data from the server is clearly displayed on the app. Ease of use, accuracy, and availability are the most important requirements for this project.

2 RESPONSIBILITIES

2.1 API Server

An API server is an absolutely necessary requirement for this project. LoRa - the communication protocol between the end nodes and the gateway is a proprietary system and as a result cannot directly interact with smartphones. Even if it could, limiting a farmer to only being able to change their fences while in proximity to the LoRa gateway is unnecessary and would just serve to be a large burden. In an effort to simplify the experience for the farmer, a Linux server will be running that will allow integration between the Android application and each collar. The main responsibilities of this server will be gather commands from the user and relay these commands to each end node. The server will need to be able to confirm the identity of the user, gather commands, send data to the user to display on the application, send updates to each end node over the LoRa protocol, gather and store data in a SQL database, and maintain a high level of availability. Our choices for the technology, operating system, and software running on the server will need to meet the above criteria in order to be deemed sufficient for this project. To reiterate, the criteria that needs to be met will include high level of availability (at least three 9s or better https://searchnetworking.techtarget.com/feature/The-Holy-Grail-of-five-nines-reliability), the ability to efficiently gather and store data, and communicate with the collars.

2.1.1 Server Technology

There are many options for the server itself including purchasing / building an actual server (Dell EMC PowerEdge or similar), a normal home computer acting as a server, or a smaller computer (Raspberry Pi) that is price efficient yet lacking in power and computational speed. For this project I recommend using a Raspberry Pi computer as these machines are extremely powerful for the price. https://www.raspberrypi.org/documentation/faqs/#pi-performance The server itself does not require extensive computation speed as communication will only be between the Android application and the LoRa end nodes, and a Raspberry Pi provides plenty of speed and memory for the SQL database while keeping costs down. Also, as this server will likely be installed in the home of the user, a low profile will only help by keeping electricity costs down while staying out of sight for the user. There is no reason to purchase anything more powerful than a Pi for this project due to the limited computational requirements. In comparison to other options, the Pi is simply one of the most supported small computers on the market. A Pi will provide reliability https://ieeexplore.ieee.org/document/7573242 and an extremely easy interface, which will decrease the amount of time us as developers need to put into this project to ensure reliability. Lastly, the Raspberry Pi already supports Linux and has its own distribution which also decreases the amount of work required. The Pi is the most reliable option on the market while staying well within budget in terms of cost and computational power.

2.1.2 Server Operating System

There are also many options for operating systems for the server including Windows Server and many flavors of Linux distributions (CentOS, Raspbian, Arch, etc.). In order to only maintain one server instead of a separate SQL server the main box will be running the Raspbian linux distribution. This distribution is ideal for any of the Raspberry Pi computers as it was specifically built for the hardware. While other distributions provide more support for large scale servers (CentOS), Raspbian is preferred for this specific application as it provides reliability while maintaining project requirements. The only two things that need to be running

on this server is the SQL database and the Python application which is automatically supported by Raspbian. https://raspberrytips.com/best-os-for-raspberry-pi/ https://www.techradar.com/news/best-raspberry-pi-distro

2.1.3 Software

The Raspberry Pi will be running an instance of Python that will host the HTTP server. Python was chosen over other languages such as NodeJS and Java for it's ease of use, our familiarity with the language, and the inherent support provided by Raspbian. Python, despite not being common in large scale HTTP server implementations, will be perfectly fine for our applications as the frequency of HTTP requests will be limited to just the one instance of the Android application. Using Python will speed up development time which will decrease costs and allow the developers to focus efforts on other parts of the project in order to ensure that everything is accomplished in time. Gunicorn will be the web server due to its compatibility with Python. https://docs.gunicorn.org/en/stable/ Gunicorn's support for Python is the primary reason why it is being chosen over nginx or Apache and will again provide reliability to the server.

https://www.digitalocean.com/community/tutorials/a-comparison-of-web-servers-for-python-based-web-applications

2.2 Long Distance Communication - LoRa / LoRaWAN

Arguably the most important part for this project is ensuring that all collars can effectively communicate with the server over vast distances while allowing scalability. Each animal may be several kilometers away from the farm house, and our project needs to be able to support large farms with many different fields. As a result, the most important part of the communication protocol is distance, closely followed by required power. Each collar needs to be able to communicate with the gateway across vast distances, and maintain a reasonable battery life as requiring a change in batteries each week would be too great an annoyance, and would directly take away from our end goal. In order to be able to allow long distance communication while keeping battery usage low, the LoRa protocol was chosen. LoRa is the proprietary communication protocol found on the physical layer of the network stack, and stands for Long Range. LoRa was initially developed for Internet of Things (IoT) applications, which supports what we are trying to accomplish in this project.

2.2.1 LoRa

There are many options in terms of long distance communication. Of course, one could theoretically route wires across long distances. This, of course, will not be feasible in any way for this particular project. Communication between the collars has to be wireless. With that in mind, a couple of options were determined to be possible. One way of communication over vast distances is connecting each collar to an already existing cellular network using 4G or 5G by using Narrowband IoT (NB-IoT). Another option is to use Sigfox, one of the original setups for IOT devices. Sigfox would require connecting to their proprietary cellular tower, somewhat similar to NB-IoT. Our last, and best option is LoRa (Long Range). LoRa uses radio waves and a proprietary communication style called Chirp Spread Spectrum (CSS) to communicate long distances in short, small bursts. https://www.semtech.com/lora/what-is-lora All three of these options provide opportunities to transmit data long distances with limited bandwidth. As each collar only needs to transmit their current position and status, bandwidth is not a problem at all, and the less bandwidth the better as less electricity would be required to power the device. Sigfox was disregarded as the company itself seems to be failing, and is lacking in coverage across the country. Also, it does not give us the flexibility to implement the network ourselves and control our own communication. NB-IoT also prevents us from managing our own network, doesn't work well with roaming assets, and is expensive in terms of cost. LoRa, on the other hand, allows us to manage our own network, supports bidirectionality, works extremely well while the end-nodes are in motion (just like cattle will be), and have a longer battery life on average than the other two options https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox . The battery life is the most important part, as the idea is to limit the required interaction between the farmer and the livestock. While LoRa does not support high data rates like NB-IoT, this is a non-issue for us as the amount of data we will need to transmit each day is miniscule. After researching options with our requirements in mind, LoRa is clearly the best option for this project.

2.2.2 Implementation / Scalability / Frequency

Semtech owns the LoRa protocol, and as a result proprietary hardware will need to be purchased for every collar in the gateway. Each collar will host a LoRa transmitter/receiver which will be used to communicate both

with the gateway and the server. Setting up the communication between these devices, ensuring scalability, and providing a reliable service are my responsibilities for this project. Hardware will need to be sourced, integrated, and tested onto the collars. The number of gateways will need to be determined on a case-by-case basis for each farm. A study will need to be conducted into the optimal way to setup a LoRa network, with distance and battery life taken into consideration. Once the number of communications per day are determined, the distance required, and the battery life needed the number of gateways can be determined. Sourcing the right LoRa receivers and transmitters is not my responsibility, and will be covered by Matthew.

2.3 Collar Logic

As a result of using the LoRa protocol, and due to the large number of end nodes planned for each network, the number of times the collars communicate with the server will need to be minimized. As a result, the server cannot inform the collars when to do an action (play a noise when the animal is getting close to the fence, electrical stimulus afterwards, etc.), and instead can only tell the collars when the location of the fence updates. Each individual collar will need to provide its own logic based on the location of the animal from GPS and the most recent information about the fenceline. A program will need to be running on these collar constantly monitoring the location of the animal and providing the logic for communicating with the server. This program needs to be efficient, reliable, and fast as the actual computational power of the collars will be limited. The response of the program to the data being provided needs to be smooth in order to not confuse the cattle when approaching a fenceline. With these requirements in mind, the following language was determined to be the best choice.

2.3.1 Software

The ideal language for this application is C. The main reason for this is my extensive familiarity with the language (decreases programming time while keeping a high level of accuracy), alongside the fact that C is a compiled language. Running Python on the collars would require unnecessary RAM that may not be available depending on the actual hardware of the system. C is the language of choice for embedded systems https://www.allaboutcircuits.com/news/programming-languages-for-embedded-systems-101-background-and-resources/, and will easily be able to integrate with the drivers that Matthew is writing for the individual components. The program will be written alongside the work down by Matthew, and will provide all necessary logic on the collars. Alongside communicating with microcontrollers and modules, the program will provide logic that will track battery usage and turn things on and off as needed. The GPS module can be limited to gathering information once every couple of seconds to decrease battery time, as well as limiting the number of connections to the LoRa gateway. The code will be compiled and installed on each collar for deployment.