IOT Final Project Report
ECE:5550 Internet of Things
Samuel Nicklaus
Cole Arduser
Sam Loecke
Luke Farmer
Professor Tyler Bell
04/17/2023
Source code: HERE

Project Name: Extra Easy Eggs

Project Background

The motivation of this project comes from Sam having his own chicken house and wanting to automate the egg tracking and know when eggs are laid. The goal is to provide an easy way for a chicken farmer to monitor their egg production and be able to know if there are eggs ready to be picked up. We wanted to be able to show the user the egg production compared to different conditions in the coop (temperature, humidity, and light exposure). We also wanted to inform the user when eggs are laid in order to eliminate unnecessary trips to the chicken house.

Overview of System Diagram

Our system diagram can be broken up into three distinct sections, the client, the server, and the hardware. For the hardware side of things, some sensors such as cameras and LED light strips are connected physically to the pi for configuration. Other sensors such as light and temperature data are transmitted via bluetooth from an arduino. That data is then processed and sent from the Pi via socket.io to our server.

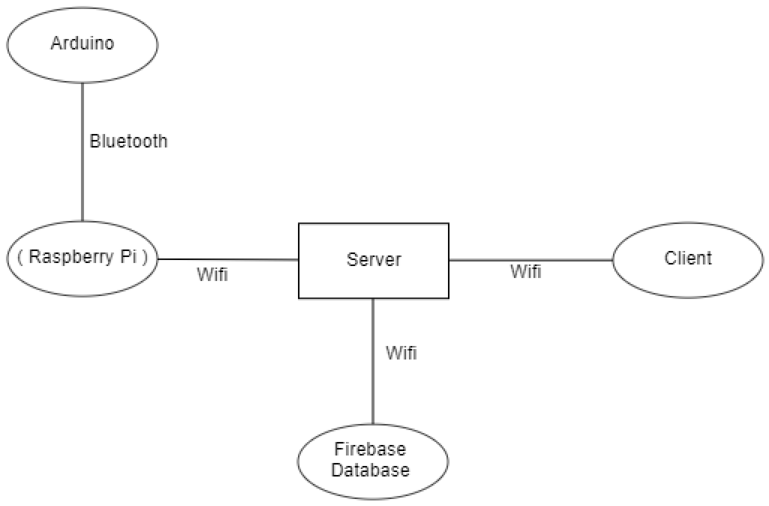

Figure 1: Client server relationship diagram

Our server is the communication point for our hardware and our client to the firebase database. The server receives API requests via wifi and after verification will send the desired data back to the device.

Sensors

In this project we are keeping track of the coop's temperature, humidity, and light levels. Also we are taking pictures to run our machine learning model to tell what is currently in each nesting box. To do this, we used the temperature, humidity, and ambient light sensors on the arduino nano. For the images we used two usb webcams connected to our raspberry Pi. All the sensor's data gets to the Pi where it then sends it all at once to the server. The Arduino collects values for all three characteristics temperature, humidity, and light every 10 seconds and then sends that data to the Pi. In order to make our communications more efficient, the arduino holds onto the last sent values for each characteristic and only sends the data to the Pi if the newly collected data is significantly different than the last sent value. The Pi uses the two webcams to take a picture of each box every 10 seconds as well. Once all the data is collected then the Pi can send that data to the server.

Wireless Communication

In this project, we used a variety of different wireless communication techniques. Starting with the Arduino Nano, the Nano collects its data and uses bluetooth to send it to the Pi. To do this, an ESS service is used for both temperature and humidity values since they have standard bluetooth configurations. The Nano also sends the light data to the Pi using the UART service since light values do not have a bluetooth standard.

The Raspberry Pi then receives the data from the Nano, performs all of its logic, and sends the data along with the images to the Server using client side node.js socket.io. The server is hosted on glitch and receives the data from the pi acting as the socket.io server. The server then has the data it needs to do its analytics and logic. In order to run the machine learning model, the server uses an http api call where it sends the image to the model. The server also has to communicate with the database and the web app.

To communicate with the database, the server uses http api calls to enter data into the database or to read the database. To communicate with the web app, it also uses http api calls to give the web app all the data it needs, and to receive settings changes from the web app.

Cloud-Based Storage

We used a Firestore Database to store all data for the chicken coop. We accessed the data by using Firebase Admin SDK with a service account. We made APIs that use the service account's credentials, which allowed us to perform basic CRUD operations on all tables. All API requests require an authorization header to make sure the requester is valid. To store the data we used four main tables.

The first was the box table which consisted of sub tables for each created box. Each of these tables stored data entries with date, humidity, light, number of eggs, and temperature data. The entries primary key was assigned to the date to make querying easy and make sure no primary keys would overlap. This table logged data for use on the web app's graphs as well as its current coop conditions.

The next table was the chicken table. Like the box data it was sorted and stored by the date. Each entry included a date, coop, enter, and exit date. This data was kept so we could keep track of box activity and whether a chicken is currently in the box.

The next table is the settings table which holds values for autoLight, brightness, tempMax, and tempMin. These are settings that the user can change on the web app to enable light and change the temperature thresholds.

The last table is the users table which holds the coop code for signing in, as well as a list of the valid users emails for notification sending.

Analytics

In this project we are keeping track of coop temperature, humidity, and light exposure. Our real time analytics include getting the light values from the Arduino and checking if it is within our time interval for the lights to be on, we turn on the lights. Also, as the user sets their temperature thresholds, we check if the temp is above or below and turn on the fan or heater as necessary. For the analytics that the user can request to see, we have graphs that range over one day, one week, or one month. Any or all of the coop characteristics (temp, humidity, light exposure) can be graphed over time as well as the number of eggs laid. These analytics help the user to see trends and optimize egg production and costs. To give the user these analytics we have to get all the data from our database within the time range the user requests. Then we have to keep track of how many eggs were laid in the time period as well as an average temperature, humidity, and light values. The server does all of these calculations by parsing the database and when finished sends only the necessary information to the user for display on the webapp.

Action

The action in our program is an email notification system where the user gets notified when an egg is detected. On log in, the user has to sign in using their gmail and that gets logged in our server for future use. Our server stores a list of every person that has previously logged in and sends out an email to everyone on egg detection. A time interval can be set to set the amount of time between emails so that they are not being sent constantly. The default is 2 hours. Additionally, users can directly interface with their coop through the webapp by being able to change light settings, temperature thresholds, and more.

Security

Security for the webapp starts at the sign in where the user logs in via google OAuth and that email token is verified in our server. Additionally, the user has to enter in the correct coop code that is a series of 6 letters/numbers that is used to associate a user with the correct coop. All api requests sent by the webapp have to contain the correct security header which prevents an unauthorized user from accessing the coop data.

Some security features for the socket.io communication between the Pi and the server include, having an authorization header for all data sent back and forth, a specific coop ID that is used to authenticate the Pi, and disconnecting a client if there is not a header included or the header is incorrect. These features allow the server to authenticate the sender (Pi) and make sure no other device is able to connect as a client. In the future if more coops are added, each new coop would just need a new Coop ID and the server would need to add that Coop ID to its allowed connections list.

The database has rules in place that only allow the Admin SDK to edit it. This places the security for the database in the hands of the API server because it is the only thing that has access to the service account credentials needed for Admin SDK. To secure the API server we require all API requests to have an authorization header with a code only known by valid requesters as well as the server.

Privacy and ethics

For this project, privacy and ethics are not a big issue. We are not saving any personal information of the users. We use google sign in for the user on the web app so we can use the built in google privacy with users passwords. We do take pictures in our project and anytime pictures are taken, they raise a privacy concern. For this, we made sure that the images are not saved anywhere after they are used to make a prediction. On the raspberry pi, only the most recent picture is saved. On the server side only the base64 encoded string is used and after being used is not stored anywhere.

Feasibility as Commercial Project

Backyard farming, especially regarding chickens, is on the rise, so there is a path to having our smart chicken coop becoming a viable product. In this section, we'll look at the feasibility of our smart chicken coop from an outsider's perspective.

At first glance, the appearance would stick out to an outsider. The design of our current coop is a prototype, so presentation was not at the forefront of our design. This shows with the cameras being different styles and our wires being tied together with electrical wires. On that front, we would make our design more appealing to the eye by making the cameras the same model and handling our wires in a better way. With a little effort put towards presentation, we feel we could have a product that performed the same or better while being more pleasing to a consumer's eye.

The other major critique would be that the scope is too small. Our current prototype is for two nesting boxes, but that wouldn't be very viable in the open market because it's too small of an application. The consumer would still have to build or buy the rest of their coop if we only offered nesting boxes. With this in mind, if we started to make this as a product, we would produce full coops. This would allow our customers to order full units. In addition, we could offer different levels of coop control and luxury.

If we worked on our presentation and increased our scope, we feel that this could be a competitive product in the backyard chicken farming market.

Issues Encountered and Lessons Learned

In this section, we'll discuss issues our group encountered and lessons learned from those issues. Throughout our project, we encountered a few issues that we had to overcome: connecting all parts, dropping the Pi, too many reads to the database, and calculating time data from database.

Our group was able to develop the code and hardware to get each of our individual pieces of the project working without much trouble. We ran into our main issues when we started to combine our parts to work together. For example, when we tried to connect our Pi to the server, we realized that the server needed lots of additional functionality added to it. During this process, we learned that sometimes API calls worked better for connections, but settled that a socket connection with the Pi and server was the best for our application. Connecting the machine learning model into the project was an issue too. Teachable machine offered many different ways to call the model, so we experimented with many different types, but we eventually settled on using the tensorflow.js model version. With this version, we added it into the functionality of our server hosted on glitch. We then found a useful library on github for using Teachable machine with javascript. Through this process, we learned the importance of making sure we were using the correct libraries when implementing our model. We connected the database and front end to our server using APIs. Through this, we got more knowledge on the difference between using APIs to connect things as opposed to using Socket connections. Connecting everything created many issues, but all of the issues were great opportunities to learn lessons.

Dropping the Pi was another issue we ran into. When the Pi was dropped, the microSD card that held all of the Pi code was ruined. This created the obvious problem of having to rewrite all of that code. The lesson learned from this was the importance of pushing our code to github and carrying the Pi with lots of caution.

The last point of difficulty was the database. The first problem was having too many reads. One day we tried to read from the database and we had 60,000 reads in a couple seconds. This surpassed the 50,000 reads per day limit, so this was obviously a major problem because firebase locked us out for a day. We figured out this was because each we were writing too much data to the database and then every line counted as a read from the database. To combat this, we wrote and read data to and from the database less frequently. This fixed the problem for our use case, but we would move to a SQL database to have more capabilities. This taught us more about the limitations of the free firebase database and the importance of knowing the limitations of the database we're using.

Group Member Contribution

Sam Nicklaus: Client Side GUI and Server conceptualization
Cole Arduser: Server/Database Creation
Sam Loecke: Arduino and Pi Code, built hardware, helped with Server
Luke Farmer: Arduino and Pi Code, developed Machine Vision

Conclusion(Everyone)

In summary, we set out to make a smart chicken coop because Sam Loecke has a chicken coop of his own and thought it would be great to have a way to track eggs laid without physically going to the coop. With this in mind, we set out to create a prototype that could inform a user with the number of eggs laid, temperature, humidity, and light exposure in the coop. On top of that, we wanted a way for the coop to have an option to be auto controlled and controlled manually from a web interface. While we did encounter some difficulties along the way, in the end, we were able to create a prototype that satisfied all of our goals of the initial scope. We look forward to taking our skills gained during this process into our future endeavors.