Bird Box
Final Report
Team 18: Aaron Bergen, Chris Manna, Will Volpe
EE459: Spring 2023

Table of Contents

Problem

Birdwatching is a popular pastime for many people, as it allows them to connect with nature and observe local wildlife. However, as urbanization continues to expand, birds are facing increasing challenges when it comes to finding suitable food sources and nesting sites. Bird feeders are an effective way to support local bird populations and provide people with an opportunity to observe them up close.

Despite the benefits of bird feeders, there are several challenges associated with using and maintaining them. Firstly, traditional bird feeders can be difficult to keep full and maintain, which can discourage people from participating in this hobby. Additionally, squirrels and other animals may be attracted to bird feeders, causing damage and stealing food meant for birds.

Moreover, many people lead busy lives and may not be able to observe birds during the day when they are most active. This can be frustrating for birdwatchers, who may miss out on opportunities to observe different species or unique behaviors.

The Bird Box smart bird feeder aims to address these challenges by providing a solution that is both easy to maintain and accessible to bird enthusiasts.

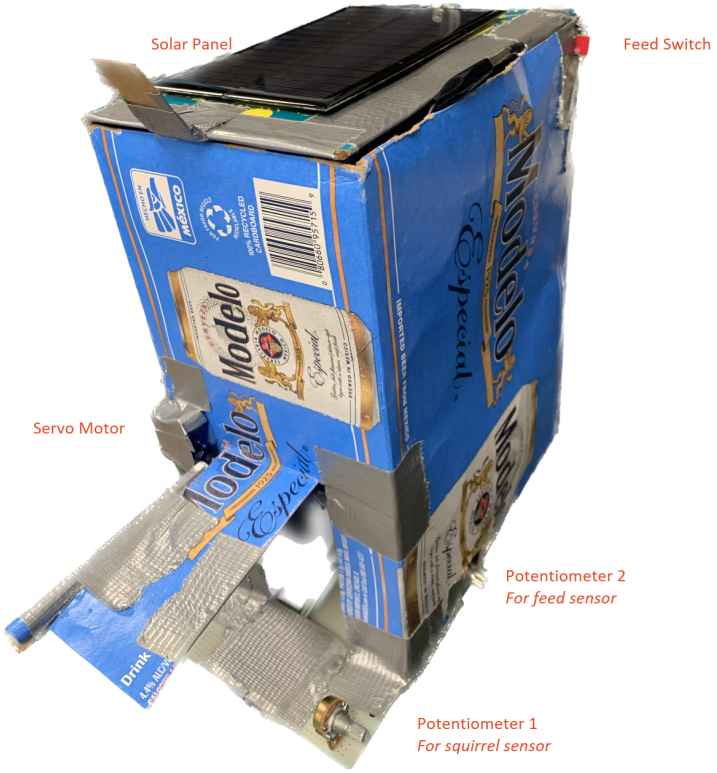

Figure 1: Exterior view of BirdBox. Interior components (ultrasonic sensor, camera, ext, not labeled)

Overview of Project

To solve the previously mentioned problems of normal birdfeeders, we wanted to create a product that could act intelligently using onboard sensors and processors as well as provide a clean user interface to monitor the status of the birdfeeder. Below is a block diagram of our circuit and how the connections to the various sensors were made.

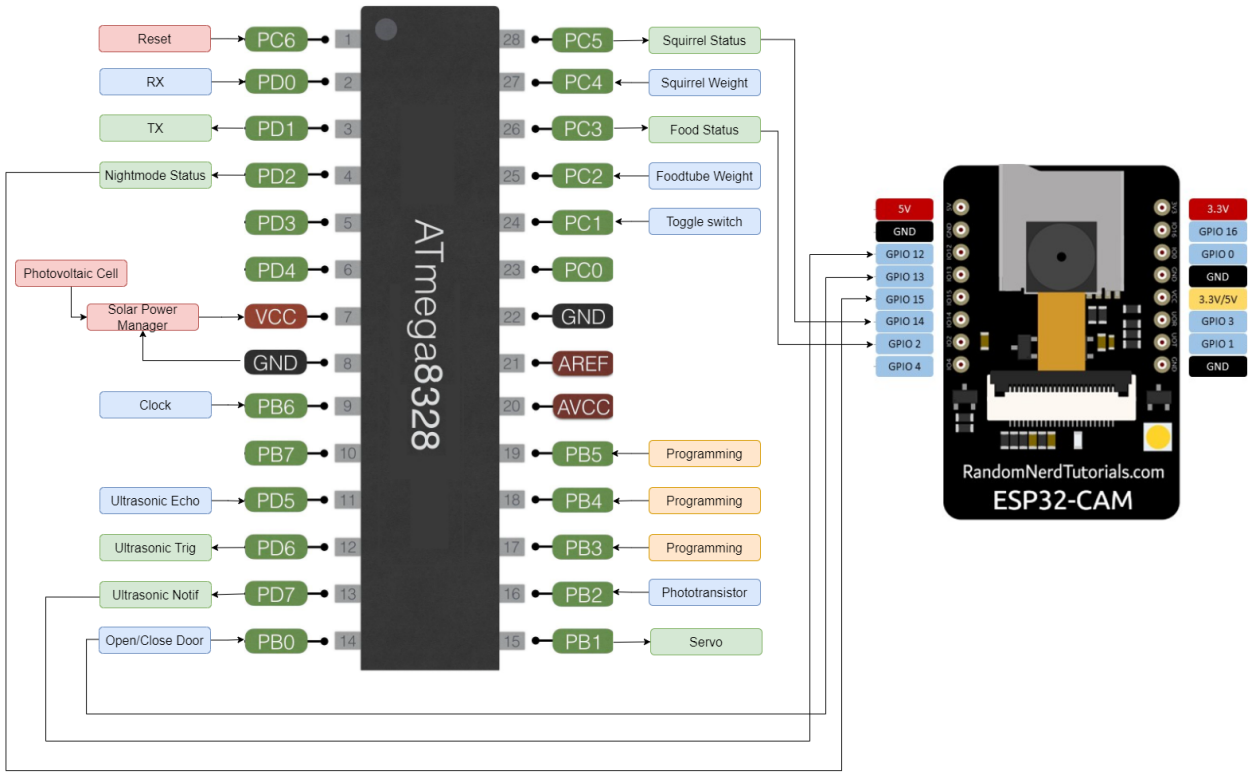

Figure 2

Our bird feeder has 6 main inputs on the Atmega328p microcontroller (figure 1 - left): a squirrel weight sensor, feed weight sensor, toggle (ON/OFF) switch, phototransistor, door control input, and ultrasonic sensor. Weight sensor 1 detects if a squirrel is perched on the feeder. Weight sensor 2 determines if the feeder is full or empty. The ultrasonic sensor detects if a bird is perched. The photoelectric sensor detects if it's nighttime to shut off all other components on the feeder. The feed switch turns off sensors so that the user can refill the bird feeder. The door input opens and closes the door.

The bird feeder also has 3 main outputs on the Atmega328p: a servo motor, a DC motor, and an LED. The Servo motor closes the door to the feeder at night. The DC motor turns on when there is a squirrel on the feeder. The LED light turns green when there is feed in the bird feeder.

We also utilize an ESP32-CAM WiFi module (figure 1 - right). This module records birds, hosts a web server that controls the birdfeeder and displays the live feed, and sends and

receives inputs/outputs to the Atmega328p microcontroller. The ESP32-CAM and Atmega328p communicate via 5 GPIOs, 4 of which are outputs on the Atmega and inputs on the ESP32-CAM, and 1 that is an input on the Atmega328 and output on the ESP32-CAM. We will go into further detail below as to how all of the inputs and outputs work and how the Atmega328p and ESP32-CAM interact with each other as well as the web server.

Sensors

Squirrel Sensor

The squirrel repellent device incorporates a variable resistor pressure sensor that is configured as a voltage divider circuit. We could have polled the voltage readings and interpreted them the same way we did for the photoelectric sensor, but at the time, we were having problems with the general functionality of our board. Instead of waiting for the board to be fixed, we decided to simultaneously implement the squirrel sensors purely through hardware. We designed an op-amp circuit configured as a comparator with positive feedback to create a hysteresis-type behavior that would stabilize to a high signal once the threshold of resistance (i.e. pressure) was surpassed, helping to eliminate any unwanted oscillations of the signal. In the circuit, there is a potentiometer on the negative input of the op-amp such that the threshold pressure sensitivity value that results in a high output can be adjusted manually to the user's desire. Below is the schematic of the circuit. FSR stands for force-sensitive resistor, which is the weight sensor and R1 is the potentiometer.

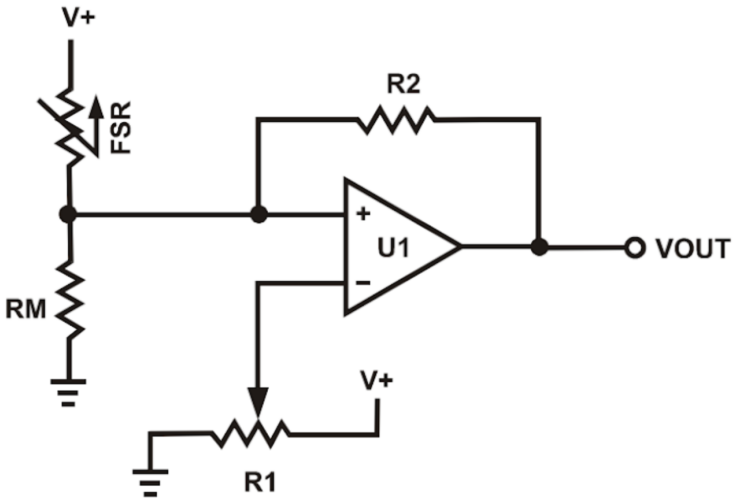

Figure 3

On a high output, the squirrel sensor would activate a DC motor that would repel any squirrel (since squirrels weigh more than birds, the threshold can be set such that only a squirrel

can trigger the motor). Once we got the board working, we polled the output voltage and used the ADC capabilities of the ATmega board so that a certain analog value above a set threshold would become a digital high, activating the flash on the camera board to further help repel a squirrel. We soldered the circuitry to a separate PCB board since the cable management of our main board was already getting messy.

Additionally, the output of the squirrel sensor controlled the output of Pin PC5 on the atmega328p. Output pin PC5 on the Atmega328p is connected to input pin GPIO 14 of the ESP32-CAM. As an additional squirrel deterrent, we configured this connection to turn on the flash LED (it was surprisingly very bright). Therefore, when GPIO 14 was high (i.e. a squirrel is present) the flash would turn on.

Feed Sensor

Since the feed status also incorporated a weight sensor, we decided to use the same schematic as above, but instead replace the DC motor at the output line with a green LED. This would produce the effect that when there is enough food in the food chamber, the LED goes off to indicate sufficient food status. Again, similar to how we polled the voltage values on the output terminal and sent it to the ESP wifi module, except we flipped the logic, a low output voltage would toggle the feed status to empty. Since the LM358 op-amp component has two op-amps inside the chip, we made all of the connections on the same breakout board that we used for the squirrel sensor.

Similar to how the Atmega328p and ESP32-CAM microcontroller interacted in the previous section, the feed sensor incorporates the same communication. Atmega pin PC3 is configured as an output controlled by the feed sensor. This output is connected to input pin GPIO 2 on the ESP32-Cam. As such, when GPIO 2 is low (i.e., food empty), the ESP32-CAM will notify the web server to update the food status to “EMPTY”.

Ultrasonic Sensor

The ultrasonic sensor works by turning our designated trigger pin high for 10 microseconds. We then get a response signal from the ultrasonic echo pin and can measure the time it takes between when the pulse of the trigger pin was sent and when the response signal was received. When the echo is received, an interrupt is triggered. When the distance calculated by the timing delay is greater than the desired threshold, we set output pin PD7 high on the atmega328p. Please see the code below for detailed implementation details.

```
198  //interrupt for Ultrasonic sensor
199  ISR(PCINT2_vect){
200    if (bit_is_set(PIND,PD6)) //if rising edge
201    {
202      TCNT0 = 0; //set timer to 0
203      PORTD |= (1 << PD0);
204    }
205    else
206    {
207      uint8_t pulse_time = TCNT0;
208      uint8_t oldSREG = SREG;
209      cli(); //disable interrupt
210      if(debug_flag){
211        char buffer[100];
212        itoa(pulse_time, buffer, 10);
213        serial_send_string(buffer); //send ultrasonic distance
214        serial_out('\n');
215        debug_flag = 0;
216      }
217      //if ultrasonic is below distance output the camera
218      if(pulse_time <= ULTRASONIC_DISTANCE_THRESHOLD){
219        PORTD |= (1 << PD7);
220      }
221      else{
222        PORTD &= ~(1 << PD7);
223      }
224      SREG = oldSREG; //enable interrupt
225    }
226  }
```

The final result is that when a bird visits our bird feeder and is in the range of < 4 inches, it triggers the pin PD7 to go high. Output pin PD7 on the atmega328p is connected to input pin GPIO 12 on the ESP32-CAM. In our ESP32-CAM code, we continuously poll the input level of GPIO 12. When the pin is high, we upload our live camera stream to the webserver to view the birds. When the pin is low, we cease stream uploading. As such, when the ultrasonic sensor detects an object (most likely a bird) within 4 inches of the bird feeder, the live camera stream turns on so that the user can see birds feeding on the web server.

An alternative approach to detecting birds that we considered was to just use the weight sensor and configure it such that there are two thresholds rather than one: the first for detecting if there is any pressure on the sensor (i.e., a bird is present), and the other for if there is a large weight presence on the sensor (a squirrel). This would mitigate the additional overhead of the ultrasonic sensor and would work well in theory as birds typically weigh 200-250 grams whereas squirrels typically weigh over 400 grams.

We decided against this alternative approach because the fidelity of the weight sensor was not enough at the lower end to sense just a little bit of weight. The weight sensor operated better at heavier weights, so we only kept it for detecting squirrel presence. The ultrasonic sensor was a good compromise because it could be used in conjunction with the squirrel weight sensor. If the weight sensor is tripped, the ultrasonic sensor will also be high, but we will not turn on the camera since we are not interested in recording squirrels.

Door Control

To prevent squirrels and other pests from feeding at night, and to improve the aesthetics of our design, we included a door to access the feed. The door is controlled by a servo motor which operates via PWM. The angle of the servo was determined by a duty cycle of a signal pin.

The signal could be varied by using the PWM timer1 functionality of the ATmega board. Please see the code below.

```
void servo_set(uint16_t deg,uint16_t max_deg){

        float set = (float)deg / (float)max_deg;

        set = (((float)SEVRO_MAX-(float)SEVRO_MIN)*set) + (float)SEVRO_MIN;

        uint16_t piont = (uint16_t)set;

        update_pwm(piont);

}
```

The door is operated under two circumstances.

1. If night mode is activated, the door will automatically close
2. If the door switch is toggled on the web server, the door will open or close. The way this works is that an output pin on the ESP32-CAM (GPIO) is connected to an input pin on the atmega328p. The atmega328p is continuously polling the status of this pin and will open/close the door based on if the status changes.

Night Status

Night status is controlled by a photoresistor at the top of the bird feeder. It is set up as a voltage divider circuit such that once the lighting is low enough, the ATmega board will interpret the polled analog value as a digital low. This will disable normal “day functionality” on the birdfeeder, as well as close the door to prevent any night critters from accessing the food.

The photoresistor also controlled the output level of pin PD2 on the atmega328p. Pin PD2 on the atmega328p is connected to input pin GPIO 15 on the ESP32-CAM. As such, when GPIO 15 is low (i.e., nighttime), the ESP32-CAM will notify the web server to update the night status to “ON”. Please see the atmega328p code to the right.

```
void check_servo(){
  if( (PINB & (1<<PINB2))){ //if sunlight
    //make night mode pin 0
    PORTD &= ~(1<<PD2);
    if( !(PINB & (1<< PINB0))){
      servo_set(OPEN_DOOR,180);
      state = DAY_DOOR_OPEN;
    }
    else{
      servo_set(CLOSE_DOOR,180);
      state = DAY_DOOR_CLOSED;
    }
  }
  else{ //else night time close door
        //make night mode pin high
        PORTD |= (1<<PD2);
        servo_set(CLOSE_DOOR,180);
        state = NIGHTMODE;
      }
}
```

Feed Switch

We also included a breakout switch to the PCB board with the force weight sensors. The user is intended to flip this switch when they would like to open the feeder to either add more bird feed or to mess around with the electronics. When the switch is toggled, it cuts power to the force resistor breakout board. We also polled the input voltage to the board, and when there is no power being supplied to the board, this disables all of the other sensors on the bird feeder and prevents updates to the web server.

Camera & Web Server

Below is a block diagram of the ESP32-CAM microcontroller and webserver.

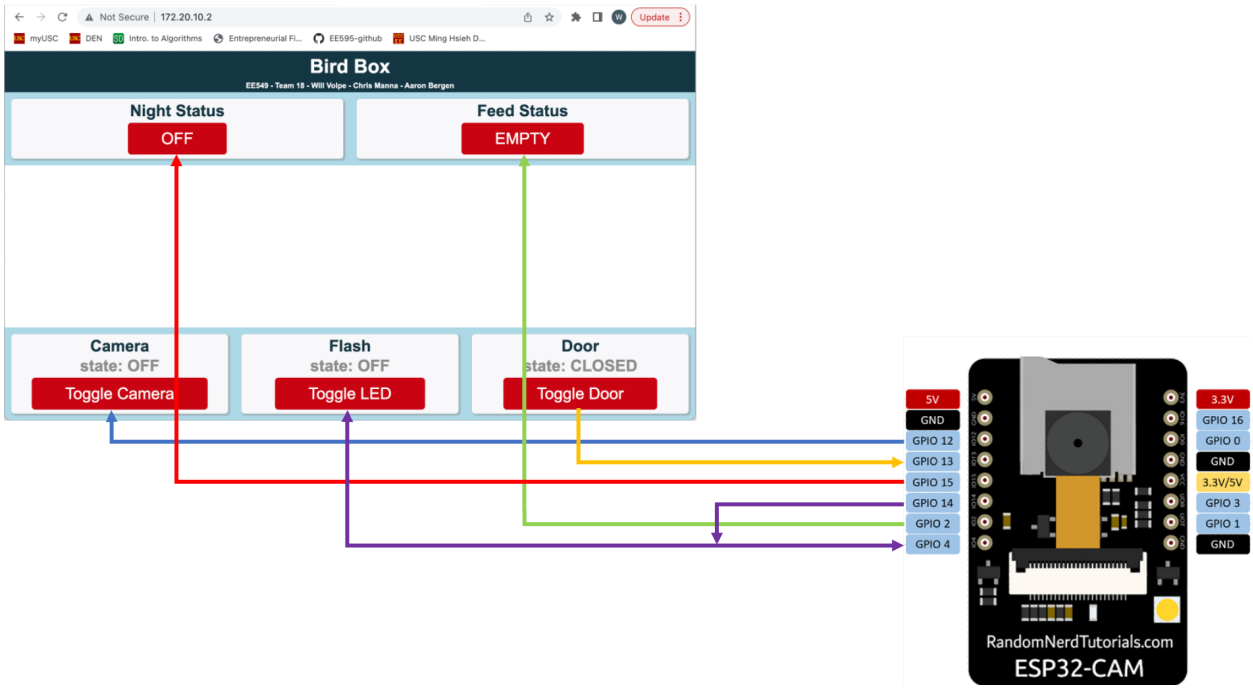

Figure 4

The ESP32-CAM is a development board that supports WiFi via the ESP32 microcontroller and has an OV2640 camera attached to it. The board hosts a web server that both controls and tracks the status of certain sensors attached to the Atmega328p. In the previous section, we explained how the ESP32-CAM and atmega328p communicate via GPIO pins, going into the specifics of each pin's functionality. In this section, we will describe how the ESP32-CAM works in tandem with the webserver.

The ESP32-CAM is built on top of an ESP32 core, allowing us to communicate via WiFi. For our purposes, we used this board to host a web server that allowed us to have a nice GUI to control our bird feeder and view the live stream. The front end of the web server is written in HTML and CSS and javascript, while the backend is written in Arduino (which is basically just C). The frontend controls the web server interface while the backend controls the variables and pins on the ESP32-CAM. The board first needs to establish a connection to a WiFi router. We then established an asynchronous web server using one of ESP's libraries. We used 5 variables to track the state of the night, feed, live stream (camera), LED flash, and door. These state variables are updated in two ways (one or both depending on the variable):

1) The ESP32 code constantly polls the status of GPIO 12, 15, 14, and 2 (does not poll 13 as the door button is an output control) every second. If any one of these

pins changes from high to low or vice versa, the state of the corresponding variable will change. This causes a notification message to be sent to the web server with the specific change to be made.

a) For example, if the photoelectric detects that it is dark out now, GPIO 15 will change from high to low. This will cause the state of the night variable to change and will send a message containing the number '6' to the web server. On receiving a message, the web server runs a javascript function with case statements to update the HTML and CSS that corresponds to the message received. In this case, receiving a '6' causes the HTML for night status to change from 'OFF' to 'ON'. See the code below.

b) This functionality is the same for the feed status (which is controlled by a feed weight sensor), camera (which is controlled by the ultrasonic sensor), and flash (which is controlled by the squirrel weight sensor).

```
function onMessage(event) {
  switch(event.data)
  {
    case '0': document.getElementById("cam_state").innerHTML = "OFF"; document.getElementById('cam_button').style.backgroundColor = "#c90411"; document.getEle
    case '1': document.getElementById("cam_state").innerHTML = "ON  "; document.getElementById('cam_button').style.backgroundColor = "#04b50a"; document.
    case '2': document.getElementById("led_state").innerHTML = "OFF"; document.getElementById('led_button').style.backgroundColor = "#c90411"; break;
    case '3': document.getElementById("led_state").innerHTML = "ON  "; document.getElementById('led_button').style.backgroundColor = "#04b50a"; break;
    case '4': document.getElementById("door_state").innerHTML = "OPEN  "; document.getElementById('door_button').style.backgroundColor = "#04b50a"; break
    case '5': document.getElementById("door_state").innerHTML = "CLOSED"; document.getElementById('door_button').style.backgroundColor = "#c90411"; break;
    case '6': document.getElementById("night_button").innerHTML = "OFF"; document.getElementById('night_button').style.backgroundColor = "#c90411"; break;
    case '7': document.getElementById("night_button").innerHTML = "ON  "; document.getElementById('night_button').style.backgroundColor = "#04b50a"; brea
    case '8': document.getElementById("fill_button").innerHTML = "EMPTY"; document.getElementById('fill_button').style.backgroundColor = "#c90411"; break;
    case '9': document.getElementById("fill_button").innerHTML = "FULL"; document.getElementById('fill_button').style.backgroundColor = "#04b50a"; break;
  }
}
```

2) The variables are also updated in-house (without intervention from the Atmega328p).

a) For example, if the door button is pressed, the web server will run a javascript function that sends a message via a toggle_door() function that updates the state variable for the door in the backend of the code. Since the state variable for the door changed, GPIO 13 will change as a result. This will then update the pin on the Atmega328p that controls the servo motor to open/close the door.

b) This functionality is the same for the camera and flash which can be independently controlled via the web server (Note* GPIO 4 is connected to the on-board flash).

Power Source

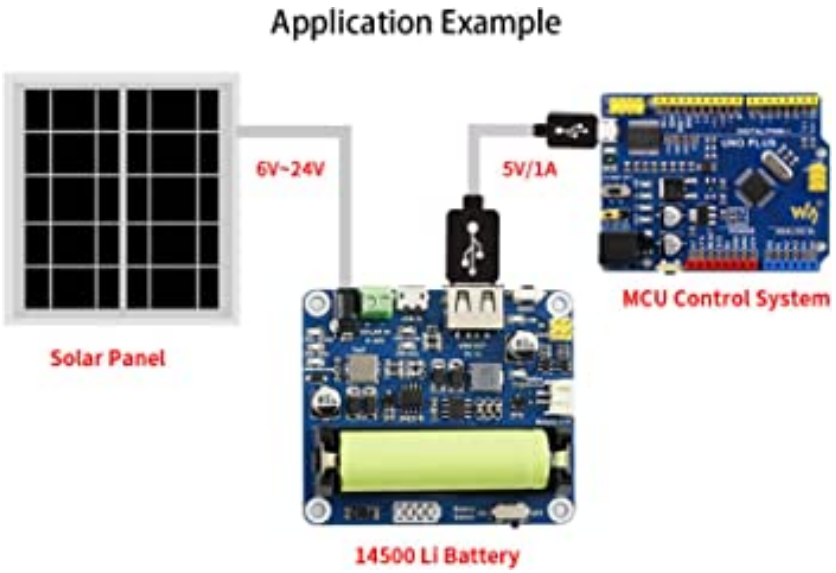

Figure 5

We used a 5v 30mA mini solar panel on top of our bird feeder to charge the battery of our feeder, along with a 5v~24v solar power management board. Our initial lithium-ion battery had too small of an energy capacity, with only 110 mA hours, so we upgraded to a new battery that contained 2500 mA hours. This was sufficient to reliably generate the required 200 mA maximum current that our system needed to run properly without discharging the battery too quickly.

We did not conduct any battery life tests on our module, however, during testing we would leave it on for stretches of up to an hour without depleting the battery. Additionally, we did not exactly test how long it would take to fully recharge the battery with the solar panel, but we approximate around 2-3 hours. Ultimately, we are pleased that our bird feeder is fully able to be powered exclusively through solar energy and does not require a wall outlet to work. In the future, if we had more time, it would have been in our best interest to see if our solar power system could remain functional throughout the entire day, with the solar panel recharging the battery at roughly the same rate (hopefully a little less) than the electronics within the feeder are consuming. We could then play around with different sized solar panels to reach this desired power balance.

Operation Manual

Initialization

The operation of our bird feeder is relatively simple. To turn it on, you simply open the bird feeder. On the underside of the lid, where the battery is fastened, you should also be able to see the solar power management module. This module has an on/off switch at the top of the board. Flipping this switch will turn on power to the rest of the board.

Make sure that food is placed inside the feeder to begin with or else the status of the birdfeeder will always read empty.

Make sure that the bird feeder is placed somewhere outside with plenty of sunlight so that the solar panel can properly recharge the bird feeder and so that the photoelectric sensor can accurately detect the time of day.

Setting up the Web Server

In order for the webserver to run correctly, the ESP32-CAM needs to be connected to a WiFi router. At the beginning of the birdbox.ino file, make sure to update the SSID and password variables to your in-home WiFi SSID and password. Then upload the code to the ESP32-CAM

Then, just power on the bird feeder and the WiFi should connect within a few seconds. If connected, the red LED onboard the ESP32-cam (next to the reset button) will be constant; if it is still trying to connect to WiFi it will be flashing.

Finally, you need a computer or phone connected to the same WiFi network. Then, simply open a web browser and type in the IP address of the ESP32-CAM and you will be able to control the bird feeder.

- Note: there are a couple of ways to figure out the IP address of the ESP32-CAM.
    1. (harder) Use an app such as ‘Fing’ to scan your WiFi network and look for the ESP32 module
    2. (easier) After changing the WiFi SSID and password, upload the code with a serial monitor connected between your board and computer. The code will print out the IP address of the board on the serial monitor.

Refilling Bird Seed

To refill the bird seed, simply flip the feed switch at the top of the birdfeeder. This will disable all of the other sensors on the feeder and stop sending updates to the web server while you are refilling the feeder. Next, open the lid and find the feeding tube located on the inside of the feeder. Fill this tube to the top with birdseed and then close the lid of the box. Lastly, flip the feed switch again to turn on normal functionality to the board.

State Diagram

Below is a detailed state machine diagram to better help illustrate how different sensor inputs result in different operational states.

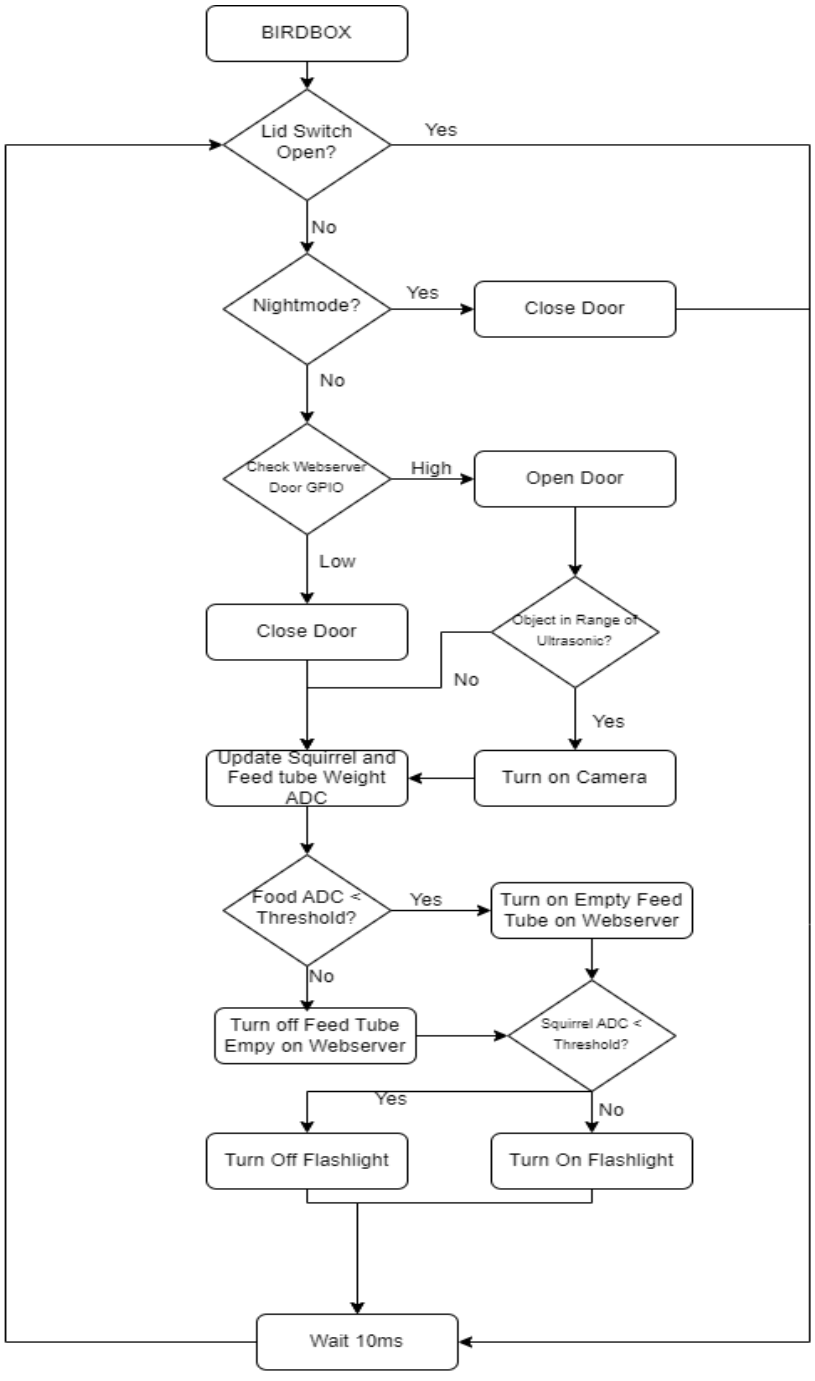

Figure 6

Electrical Engineering Standards

Lithium Ion Safety Standards

Lithium-ion safety standards are established to make sure that batteries are safe while handling and testing. It is imperative that these standards are referenced when evaluating equipment containing lithium-ion based batteries.

Performance standards: IEC 61960

Safety standards: IEC 62133-2:2017, IEC 62281:2019

Testing standards: IEC 61959:2004

Lithium-ion battery standards should be followed for several reasons. Firstly, they help streamline business practices and provide a level playing field for businesses to develop products and services. Secondly, they ensure that lithium-ion batteries are safe for consumers and the environment. Standards help ensure that these batteries are safe to use and operate reliably.

Standards for Environmental Assessment of Electronic Products

IEEE 1680-2009: This standard provides guidance and implementation procedures for the environmental assessments of electronic products. This standard aims to promote environmentally sustainable designs and manufacturing practices. The performance criteria for IEEE 1680-2009 is evaluated based on energy efficiency, product lifetime extension, and material selection. It also establishes a set of requirements for manufacturers to meet in order to declare their product as environmentally sustainable.

IEC 61724: This standard outlines the terminology, equipment, and methods to performance monitoring and analysis of photovoltaic systems. It defines classes of photovoltaic performance monitoring systems and provides guidance for monitoring system choices including sensor installation and accuracy for mounting photovoltaic systems. It also serves as the basis for other standards that rely on proper PV assessment.

ISO 50001: This is the standard for specifying requirements for an energy management system. It provides the framework for managing energy performance, including energy efficiency system, energy use, and consumption. This standard is mainly for helping to reduce greenhouse gas emissions and decreasing energy costs. This standard applies to any size organization (even fitting for our small handmade product).

Signature Sheet

	Aaron	Will	Chris
System Design	33.3	33.3	33.3
Component Selection	33.3	33.3	33.3
Hardware Design	70	15	15
Software Design	10	45	45
Documentation	40	30	30
Oral Presentation	10	45	45
Written Presentation	40	40	20

Aaron Bergen _____
Will Volpe _____
Chris Manna _____

Appendix

Part Name	Supplier Product Name	Cost
Weight Sensor 1	Adafruit 166	7.00
Weight Sensor 2	Adafruit 1075	5.95
Ultrasonic Sensor	Adafruit 3942	3.95
Photoelectric Sensor	Adafruit 2831	0.95
DC Motor	Amazon B00GN6EHQI	10.00
Servo Motor	Sparkfun ROB-09065	9.95
Photovoltaic Cell	Amazon B07BMMHMSJ	15.99
LED	Sparkfun COM-00105	2.25
Op Amp	LM358M	0.75
Lithium Ion Battery	Sparkfun PRT-13112	9.99
Solar Panel Management	Amazon B088QFMCMV	10.99
Wifi Module	ESP32-CAM	12.99
Total Cost		**90.76**

Appendix 1: project cost breakdown.