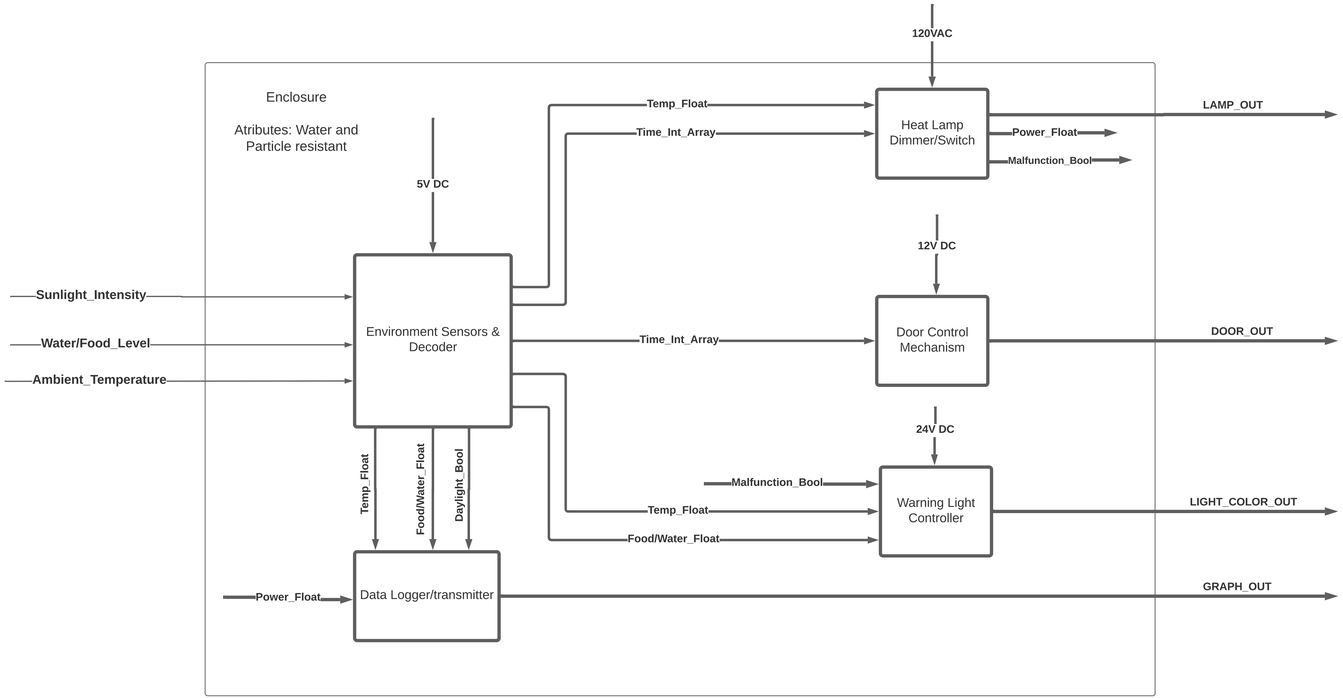
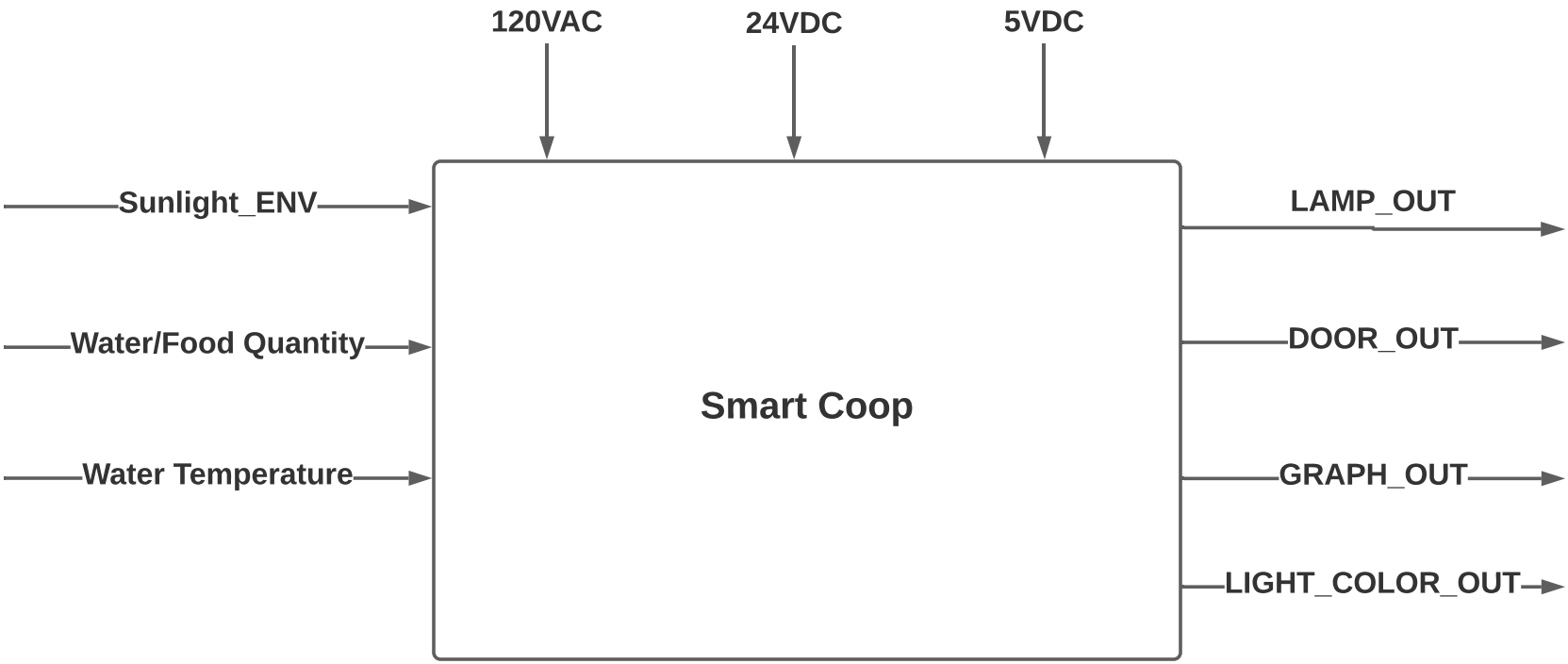
Interface Definitions

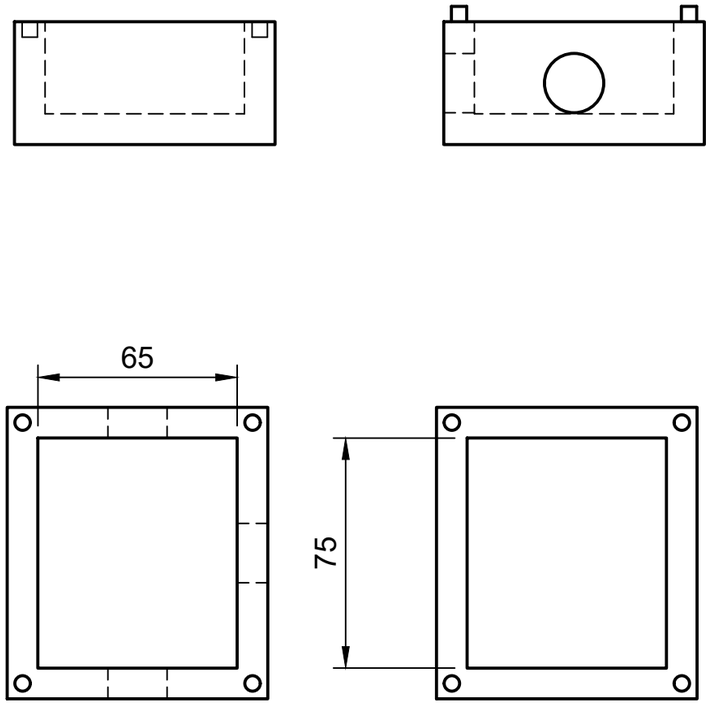
Interface Name	Property 1	Property 2
Time_Int_Array	An array, 0th index: hours 1st index: minutes 2nd index: seconds	Accuracy within 30 seconds of actual time
Temp_Float	Unit Celsius in floating point	Accuracy within 2C of actual temperature
Food/Water_Float	Weight in Grams for food, mL for water (Floating Point)	Number is proportional to amount of resources (higher number = more resources)
5VDC	5V DC within 0.25V	Minimum current rating of 200mA
24VDC	24VDC within 0.5V	Minimum current rating of 2.5A
120VAC	110-130Vrms	Minimum current rating of 3A
Power_Float	Floating Number, in Watt-Hours	Considers load from the heat lamp
Light Level	Boolean 1=Daylight, 0=No daylight	Daylight defined as light level >= 40 Lux
12VDC	12VDC within .25V	Minimum current rating of 1A
LAMP_OUT	Provides 0-120VAC to heat lamp to visbily dim the bulb	Minimum current rating of 2A
DOOR_OUT	Door opens and closes at preset time	Door must not harm chickens while operating
LIGHT_COLOR_OUT	Set of colors in light tower to be illuminated	Correspoding LED colors that will be illuminated: Red : Temperature <= 2C Yellow: Food/Water < 200g/100mL Green: No warnings
GRAPH_OUT	Set of graphs and indicators that display the past 7 days of data	Data includes: Resource consumption (food/water), temperature, power usage, and daylight presence
Enclosure Atributes (PCB)	Water-resistant such that no waters through main seams during a stained soaking for over 10 seconds	Particle resistant such that no material enters through main seams for in a extremly dirty/dusty contact envorment for over 10 seconds
Malfunction_Bool	Boolean value: indicates if heat lamp is malfunctioning	Should have value True when heat lamp is not correctly operating

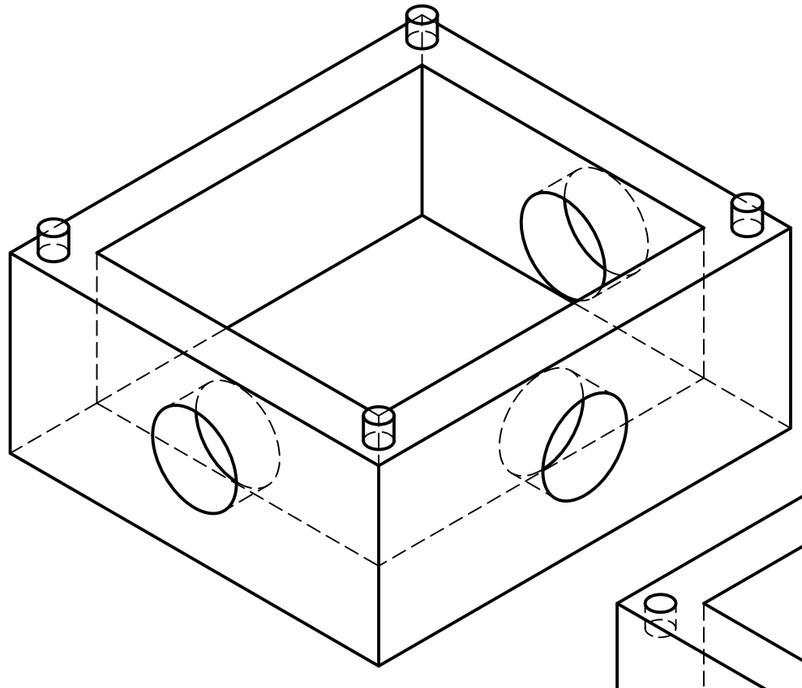


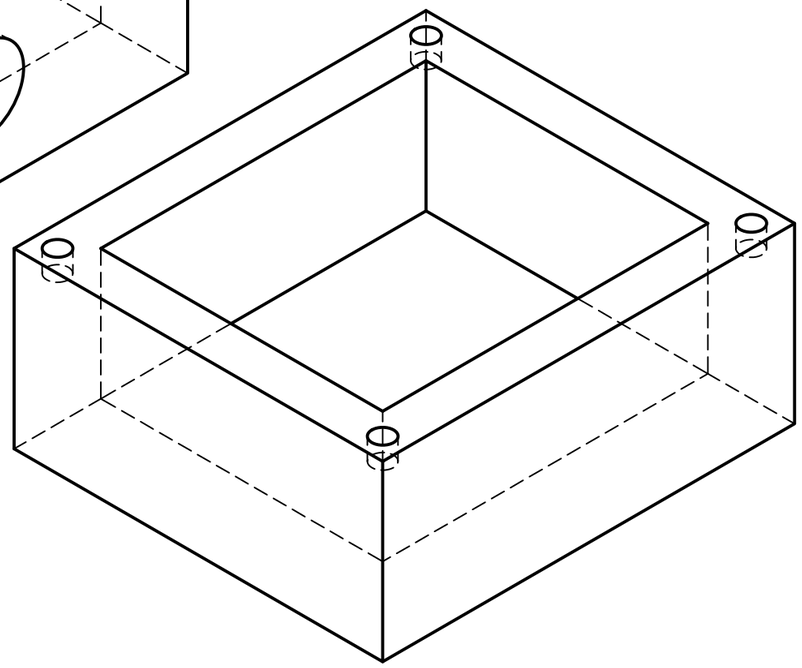
Block Owners

Mark Huynh	Michael Crockett	Nicholas McBee
Data Logger/transmitter	Enclosure	Environment Sensors & Decoder
Warning Light Controller	Door Control/Motor Mechanism	Heat Lamp Dimmer/Switch

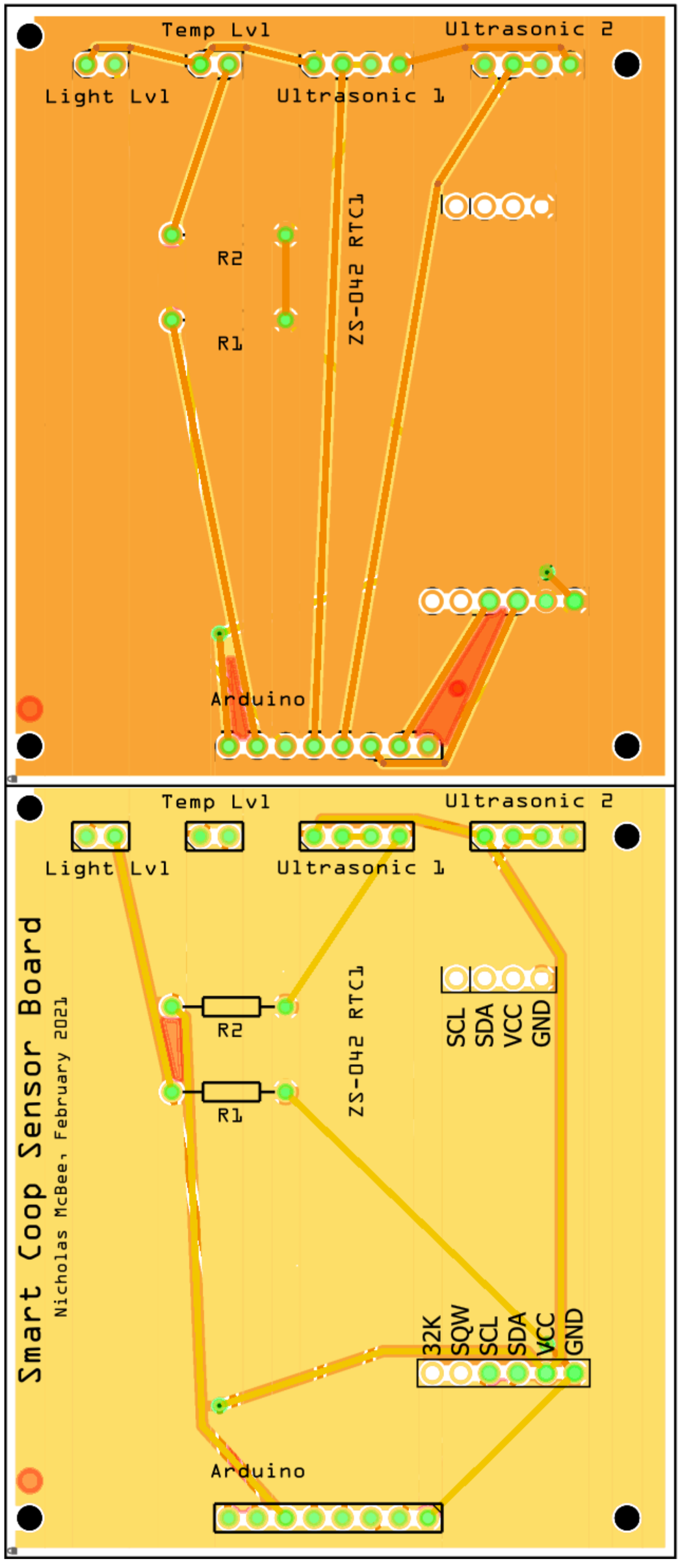

120VAC
24VDC
5VDC
Sunlight_ENV
Water/Food Quantity
Water Temperature
Smart Coop
LAMP_OUT
DOOR_OUT
GRAPH_OUT
LIGHT_COLOR_OUT

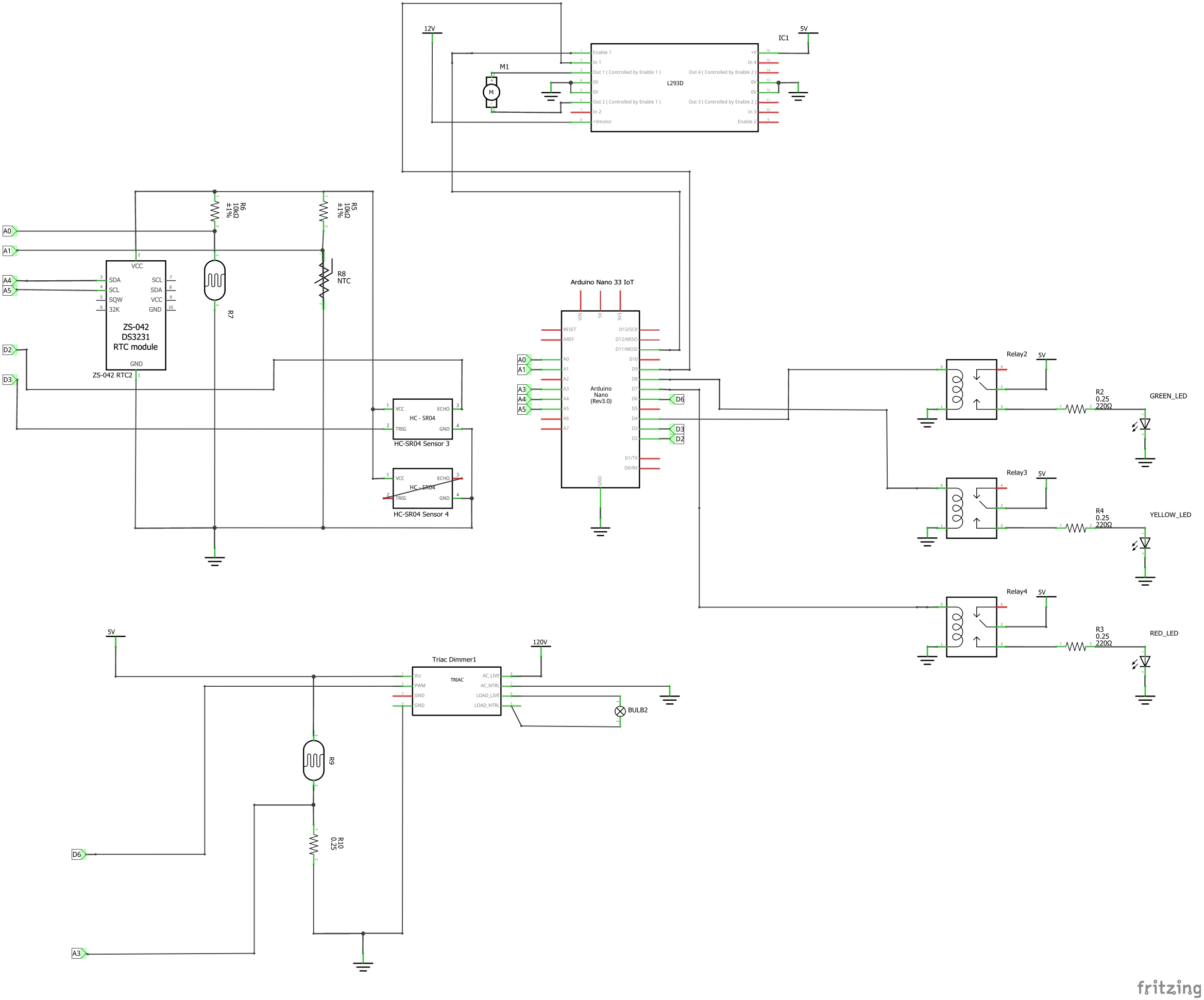






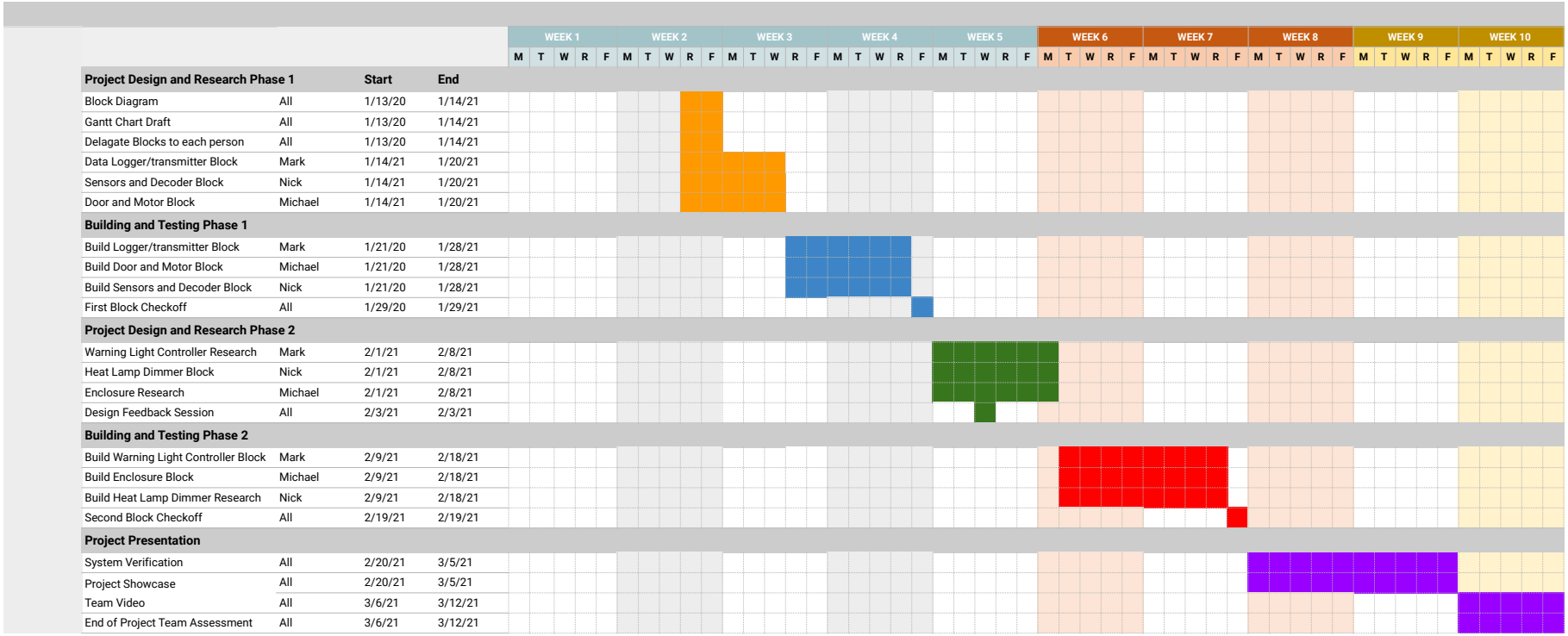
Dept.	Technical reference	Created by Michael Crockett 2/15/2021	Approved by
		Document type	Document status
		Title PCB Enclosure	DWG No.
			Rev. / Date of issue 2/15/21 / Sheet 1/1


Temp Lvl
Ultrasonic 2
Light Lvl
Ultrasonic 1
R2
R1
ZS-042 RTC1
Arduino
Smart Coop Sensor Board
Nicholas McBee, February 2021
SCL
SDA
VCC
GND
32K
SQW
SCL
SDA
VCC
GND


12V
5V
IC1
M1
Enable 1
In 1
Out 1 (Controlled by Enable 1)
0V
0V
Out 2 (Controlled by Enable 1)
In 2
+Vmotor
L293D
+V
In 4
Out 4 (Controlled by Enable 2)
0V
0V
Out 3 (Controlled by Enable 2)
In 3
Enable 2
R6
10KΩ
±1%
R5
10KΩ
±1%
R8
NTC
R7
A0
A1
A4
A5
D2
D3
VCC
SDA
SCL
SQW
32K
SCL
SDA
VCC
GND
ZS-042
DS3231
RTC module
GND
ZS-042 RTC2
HC - SR04
VCC
TRIG
ECHO
GND
HC-SR04 Sensor 3
HC-SR04 Sensor 4
Arduino Nano 33 IoT
Arduino
Nano
(Rev3.0)
D6
D3
D2
Relay2
Relay3
Relay4
5V
R2
0.25
220Ω
GREEN_LED
R4
0.25
220Ω
YELLOW_LED
R3
0.25
220Ω
RED_LED
120V
Triac Dimmer1
TRIAC
Vcc
PWM
GND
GND
AC_LIVE
AC_NTRL
LOAD_LIVE
LOAD_NTRL
BULB2
R9
R10
0.25
D6
A3

CHICKEN COOP GANTT CHART

PROJECT TITLE	Smart Chicken Coop
PROJECT MANAGERS	Mark Huynh, Nick McBee , Michael Crockett



Project Design and Research Phase 1		**Start**	**End**
Block Diagram	All	1/13/20	1/14/21
Gantt Chart Draft	All	1/13/20	1/14/21
Delagate Blocks to each person	All	1/13/20	1/14/21
Data Logger/transmitter Block	Mark	1/14/21	1/20/21
Sensors and Decoder Block	Nick	1/14/21	1/20/21
Door and Motor Block	Michael	1/14/21	1/20/21
Building and Testing Phase 1			
Build Logger/transmitter Block	Mark	1/21/20	1/28/21
Build Door and Motor Block	Michael	1/21/20	1/28/21
Build Sensors and Decoder Block	Nick	1/21/20	1/28/21
First Block Checkoff	All	1/29/20	1/29/21
Project Design and Research Phase 2			
Warning Light Controller Research	Mark	2/1/21	2/8/21
Heat Lamp Dimmer Block	Nick	2/1/21	2/8/21
Enclosure Research	Michael	2/1/21	2/8/21
Design Feedback Session	All	2/3/21	2/3/21
Building and Testing Phase 2			
Build Warning Light Controller Block	Mark	2/9/21	2/18/21
Build Enclosure Block	Michael	2/9/21	2/18/21
Build Heat Lamp Dimmer Research	Nick	2/9/21	2/18/21
Second Block Checkoff	All	2/19/21	2/19/21
Project Presentation			
System Verification	All	2/20/21	3/5/21
Project Showcase	All	2/20/21	3/5/21
Team Video	All	3/6/21	3/12/21
End of Project Team Assessment	All	3/6/21	3/12/21

Part Name	Quantity	Price	Link
Data Transmitter Block			
Arduino Nano 33 IoT	1	$18	https://store.arduino.cc/usa/nano-33-iot
Door Motor Block			
Motor	1	$8	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P1550609964
Jumper Wires	7	Owned	https://www.adafruit.com/product/1953?gclid=Cj0KCQiA3smABhCjARIsAKtrg6LOu07vFTYT5zJEPkEIJSlgGoH29DIM0WkGRrqbbIxVI6VfYKeuKVYaAseYEALw_wcB
Door (Cardboard)	1	Owned	N/A
Doorframe (Cardboard)	1	Owned	N/A
L293 Transistor	1	Owned	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P274
Sensor Module Block			
Thermistor	1	$1	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P348
RTC Module	1	$2.20	https://www.amazon.com/HiLetgo-AT24C32-Arduino-Without-Battery/dp/B00LX3V7F0/
Ultrasonic sensor	2	6	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P392
Photoresistor	1	2	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P349
1/4W or 1/8W 10k THT resistor	2	0.1	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P359
PCB	1	$4	
Warning Lights Block			
Through Hole LEDs	1	$7.25	https://www.amazon.com/eBoot-Pieces-Emitting-Diodes-Assorted/dp/B06XPV4CSH/ref=sr_1_7?dchild=1&keywords=through+hole+led&qid=1613719315&sr=8-7
2 Channel Relays	2	$3	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P1424917095
220 Ohm Resistors	4	$1	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P370
Enclosure			
3D Printed Model	1	$0	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/userPrintsSubmit.php
Lamp Dimmer Block			
Photoresistor	1	$2	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P349
AC Triac dimmer board	1	$24	https://www.amazon.com/gp/product/B06Y1DT1WP/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1
1/4W or 1/8W 10k THT resistor	2	0.1	https://eecs.oregonstate.edu/education/tekbotSuite/tekbot/pages/publicInventoryPart.php?stocknumber=P359
	Total Price	$86.85	

```
/*  merged_coop
 *  Complete code used for system verification
 */

#include "thingProperties.h"
#define LAMP_MALFUNCTION_TRESH 1000 //Comparison value for ADC brightness measurement
#define LAMP_ON_AM 6                // Time that lamp will turn on in morning
#define LAMP_OFF_AM 9               // Time lamp will turn off in morning
#define LAMP_ON_PM 18               // Time lamp will turn on at night
#define LAMP_OFF_PM 21              // Time lamp will turn off at night
#define LAMP_WATTAGE 4              //Wattage rating of lamp bulb

#include "RTClib.h"
RTC_DS3231 rtc;
#include "NewPing.h"

const int dimmerPin = 6;       //Pin for PWM control of dimmer board
const int lampPhotoPin = A3;  //Photoresistor used for heat lamp

const int RED_PIN = 7;  // the Arduino pin, which connects to the IN pin of relay for the RED LED
const int YELLOW_PIN = 8;  // the Arduino pin, which connects to the IN pin of relay for the YELLOW LED
const int GREEN_PIN = 4;  // the Arduino pin, which connects to the IN pin of relay for the GREEN LED
const int ALARM_PIN = A5;  // the Arduino pin, which connects to the IN pin of relay for the BLUE LED

byte brightness_setting = 0; //Brightness setting for lamp
bool lamp_malfunction = 0;    //Indicates malfunction when high
//int currTime[3];              //Time array, holds dummy values for simulation
float power_consumption = 0;    //Power consumed by lamp in Watt-

#define INPUT1   9  // Motor driver input 1
#define INPUT2   10 // Motor driver input 2
#define ENABLE  11  // Motor driver enable

#define Low 75 // determined as the optimal speed after testing
int Speed;

const byte photosensorPin = A0; // Environment light sensor pin

const byte thermistorPin = A1;
//Steinhart-Hart constants for temperature calculations
const float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;
const int R3 = 10000; //Value of resistor in series with thermistor

const byte food_data_pin = 2;
const byte water_data_pin = 3;
const float foodDensity = 0.7; //grams per mL
const float containerLength = 20;
const float containerRadius = 4.0;
const float waterContLength = 16.0;
const float waterContRadius = 4.25;
const float waterDensity = 1.0;
const int samples = 5; //Number of ultrasonic readings to take per call

bool isOpen = false;    // Current door state

NewPing food_sensor(food_data_pin, food_data_pin, 20);
NewPing water_sensor(water_data_pin, water_data_pin, 16);

float temp;
int currTime[3];
bool lightLevel;

float foodLevel, waterLevel;

void setup() {

  //Setup for transmitter module
  Serial.begin(57600);
  delay(1500); // Search for serial monitor
  //initProperties(); //Setup for transmitter
Serial.println("Starting code");
    // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
    setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  //Setup for sensors module
  pinMode(thermistorPin, INPUT);

  #ifndef ESP8266
    while (!Serial); // wait for serial port to connect. Needed for native USB
  #endif

  if (! rtc.begin()) {
```

```
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }

  //Pin setup for door controller
  pinMode(ENABLE, OUTPUT);
  pinMode(INPUT1, OUTPUT);
  pinMode(INPUT2, OUTPUT);

  //Pin setup for warning lights
  pinMode(RED_PIN, OUTPUT);
  pinMode(YELLOW_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(ALARM_PIN, OUTPUT);
  Serial.println("Leaving setup");

  Speed = Low; // Set door motor speed
}

void loop() {

  //Update values and print to monitor
  getTime();
  temp = getTemp();
  lightLevel = getIfDaylight();
  foodLevel = getFoodLevel();
  waterLevel = getWaterLevel();
  printData();

  //Update cloud variables/graphs
  //ArduinoCloud.update();
  temperature = getTemp();
  waterLevel = getWaterLevel();
  foodLevel = getFoodLevel();
  powerConsumption = get_power_consumption();
  lightLevel = getIfDaylight();

  lamp_controller(); //Handles heat lamp dimming and malfunction checking
  print_lamp_data();
  getTime();
  printData(); //Print sensor values to serial monitor for testing

  doorController(); //Handles door opening and closing
  changeLights(waterLevel, foodLevel, temp, lamp_malfunction); //Handles warning light outputs
  delay(1000);

}

/* Func: print_lamp_data
 * Test bench function used to print the current test values
 * for time and heat lamp status
 */
void print_lamp_data() {
Serial.print("Power Consumption: ");
Serial.println(power_consumption);
if(lamp_malfunction)
  Serial.println("Lamp malfunction detected");
else
  Serial.println("Lamp OK");
  Serial.println("");
}

/* Func: lamp_controller
 * Determines if the heat lamp should be turned on based
 * on current time and how bright it should be
 */
void lamp_controller(){
  if(brightness_setting == 255) //Checks lamp when fully turned on
    lamp_malfunction = check_lamp();
  if(((currTime[0] >= LAMP_ON_AM) && (currTime[0] < LAMP_OFF_AM))|| temp <= 2) { //Turns on lamp in early morning or in freezing temp
    brightness_setting = 255;
  }
  else if((currTime[0] >= LAMP_ON_PM) && (currTime[0] < LAMP_OFF_PM)) { //Turns on lamp in evening
    brightness_setting = 255;
  }
  else if(currTime[0] == LAMP_OFF_PM) { //Dim lamp until off at nighttime
    dim_lamp();
  }
  else {
    brightness_setting = 0;   //Turn off lamp at all other times
  }
  analogWrite(dimmerPin, brightness_setting); //Set signal for brightness
```

```
}

/* Function: get_power_consumption
 * Calculates the power consumption of the heat lamp for the current day.
 */
float get_power_consumption() {
  float lamp_time = (LAMP_OFF_AM - LAMP_ON_AM) + (LAMP_OFF_PM - LAMP_ON_PM); //Time lamp is turned on
  float power = LAMP_WATTAGE*lamp_time; //Compute watt-hour usage
  return power;
}

/* Func: dim_lamp
 * Lowers the lamp brightness with respect to number of seconds elapsed.
 * Should only be called when it is time for the lamp to be dimmed and turned off.
 */
void dim_lamp() {
  if(brightness_setting <= 5) //Turns off lamp when dimming is complete
    brightness_setting = 0;
  else {
    brightness_setting = 255 - 10*currTime[2]; //Reduces brightness setting by about 4% each second
  }
}

/* Func: check_lamp
 * Reads the analog voltage from the lamp photocell divider to determine if
 * the lamp is functioning properly. This should only be called when the lamp is set at maximum brightness
 * to remain accurate.
 */
bool check_lamp() {
  int lamp_brightness = analogRead(lampPhotoPin);
  if(lamp_brightness < LAMP_MALFUNCTION_TRESH) //Check if ADC reading is at corresponding light threshold
    return 1;
  return 0;
}

void Forward(void){ //Starts the motor in a counterclockwise rotation in order to open the door
  analogWrite(ENABLE, Speed);
  digitalWrite(INPUT1, HIGH);
  digitalWrite(INPUT2, LOW);
}

void Backward(void){ //Starts the motor in a counterclockwise rotation in order to close the door
  analogWrite(ENABLE, Speed);
  digitalWrite(INPUT1, LOW);
  digitalWrite(INPUT2, HIGH);
}


void stopped(void){//Funtion stops motor by setting both inputs to high
  digitalWrite(INPUT1, HIGH);
  digitalWrite(INPUT2, HIGH);
}

void opening(void) {
  Forward(); //opens door
  delay(300);
  stopped();

}

void closing(void) {
  Backward(); //closes door
  delay(300);
  stopped();

}

void doorController() {
    Speed=Low; // Testing showed that 75 was about as slow as it could be set while still maintaining reliable operation.
    if((currTime[0] == 19)&& (currTime[1] == 50)) {
      opening();
      delay(300);
    }
    if((currTime[0] == 5)&& (currTime[1] == 0)) {
    closing();
    }
}

/*Function: getFoodLevel
 * Determines the depth of the food in the container from
 * distance sensor and known constants.
 */
float getFoodLevel() {
  float h1 = get_food_distance(); //Distance of food from sensor
```

```
  float h2 = containerLength - h1; //Depth of food in container
  if(h2 < 0)
    h2 = 0;
  float volume = PI*containerRadius*containerRadius*h2; //Food volume in cm^3
  float mass = foodDensity*volume;

  return mass;
}

//Water verison of getFoodLevel
float getWaterLevel() {
    float h1 = get_water_distance(); //Distance of food from sensor
  float h2 = waterContLength - h1; //Depth of food in container
  if(h2 < 0)
    h2 = 0;
  float volume = PI*waterContRadius*waterContRadius*h2; //Food volume in cm^3
  float mass = waterDensity*volume;

  return mass;
}

/*Function: getFoodDistance
 * Calculates the distance (in cm) of the food level from
 * the food ultrasonic sensor
 */
 float get_food_distance() {
  float duration, cm;
  duration = food_sensor.ping_median(samples);
  cm = (duration / 2) * 0.0343;
  return cm;
 }

//Water version of get_food_distance
 float get_water_distance() {
  float duration, cm;
  duration = water_sensor.ping_median(samples);
  cm = (duration / 2) * 0.0343;
  return cm;
}

/*Function: getIfDaylight
 * Reads analog value from photoresistor voltage divider
 * and provides a light level approximation.
 */
 bool getIfDaylight() {
  int adcVal = analogRead(photosensorPin);
  float photoVo = 5*(float)adcVal/(1023); //Output voltage of voltage divider
  if(photoVo >= 4.17) { //Check if light level is at least 40 Lux
    return true;
  }
  return false;
 }

/*Function: getTemp
 *Reads analog value from thermistor voltage
 *divier and calculates the temperature in Celsius.
 */
float getTemp() {
  int adcVal = analogRead(thermistorPin);
  float rTherm = R3*(1023.0/(float)adcVal - 1.0); //Current thermistor resistance
  float logRtherm = log(rTherm);
  //Temperature in Kelvin from Steinhart equation:
  float tempK = (1.0/(c1 + c2*logRtherm + c3*logRtherm*logRtherm*logRtherm));
  return (tempK - 273.15); //Return temp converted to Celsius

 }

/*Function: printData
 *Prints the sensor values to the
 *serial monitor for testing and debugging.
 */
 void printData() {
  Serial.print("Time: ");
  Serial.print(currTime[0]);
  Serial.print(":");
  Serial.print(currTime[1]);
  Serial.print(":");
  Serial.println(currTime[2]);

  Serial.print("Temperature (C): ");
  Serial.println(temp);

  if(lightLevel)
    Serial.println("Daylight brightness detected");
```

```
    else
      Serial.println("Light level below daylight levels");

    Serial.print("Food Level: ");
    Serial.println(foodLevel);
    Serial.print("Water Level: "  );
    Serial.println(waterLevel);

    Serial.println(" ");
   }

  /*Function: getTime
   *Collects current time info from
   *the RTC module and updates the
   *current time array.
   */
  void getTime() {
    DateTime now = rtc.now();

    currTime[0] = now.hour();
    currTime[1] = now.minute();
    currTime[2] = now.second();

  }

  void onFooChange() {
    // These functions are intentionally left blank. They are what enable IoT cloud to cause changes based off of variable values
  }


  void onTemperatureChange() {
    // Do something
  }


  void onFoodLevelChange() {
    // Do something
  }


  void onWaterLevelChange() {
    // Do something
  }


  void onLightLevelChange() {
    // Do something
  }


  void onPowerConsumptionChange() {
    // Do something
  }

  // Function that will take each of the sensor values and determine corresponding LEDs to turn on
  void changeLights(float waterLvl, float foodLvl, float temperature, bool lampMalfunction) {
    int yellow_light = waterLvl < 200 || foodLvl < 200; //Resources threshold
    int red_light = temperature <= 3; // Temperature threshold
    int green_light = !(yellow_light || red_light); // If no other lights are on, green should be on


  //   Write all of the determined values to corresponding digital pins
    digitalWrite(RED_PIN, red_light);
    digitalWrite(YELLOW_PIN, yellow_light);
    digitalWrite(GREEN_PIN, green_light);
  }

  // Function to return lights back to green status
  void returnNormalLights() {
    changeLights(100, 100, 100, 100);
    delay(1000);
  }
```

Date	In	Out	Focus	Hours		**Nick**
1/10	*2:00 PM*	*3:00 PM*	*Requirements*	*1*		Time Spent:
1/13	*2:00 PM*	*4:00 PM*	*Block diagram*	*2*		73
1/14	*10:00 AM*	*12:00 PM*	*Block diagram*	*2*		
1/15	*3:00 PM*	*5:00 PM*	*Sensors research*	*2*		
1/16	*9:30 AM*	*3:00 PM*	*Sensor schematic*	*5.5*		
1/19	*9:00 AM*	*1:00 PM*	*Sensor code*	*4*		
1/21	*11:00 AM*	*2:00 PM*	*Sensor testing*	*3*		
1/23	*9:30 AM*	*3:00 PM*	*Sensor docs*	*5.5*		
1/26	*10:00 AM*	*4:00 PM*	*Sensor testing*	*6*		
1/29	*3:00 PM*	*6:00 PM*	*Sensor docs*	*3*		
2/2	*4:00 PM*	*5:30 PM*	*Feedback docs*	*1.5*		
2/9	*9:00 AM*	*1:00 PM*	*Dimmer research*	*4*		
2/11	*3:00 PM*	*6:00 PM*	*Dimmer design*	*3*		
2/13	*10:00 AM*	*2:00 PM*	*Dimmer schematic*	*4*		
2/16	*11:00 AM*	*2:00 PM*	*PCB design*	*3*		
2/18	3:00 PM	6:00 PM	PCB design	3		
2/21	10:00 AM	4:00 PM	Sensor testing	6		
2/23	10:00 AM	12:00 PM	Dimmer testing	2		
2/25	11:00 AM	1:00 PM	Dimmer video	2		
2/27	10:00 AM	12:00 PM	Sensor fixing	2		
2/28	11:30 AM	8:00 PM	System integration	8.5		

Date	In	Out	Hours		**Michael**
1/10	*2:00 PM*	*3:00 PM*	*1:00:00*		Time Spent:
1/13	*2:00 PM*	*4:00 PM*	*2:00:00*		61:00:00
1/14	*1:00 PM*	*4:00 PM*	*3:00:00*		
1/16	*12:00 PM*	*6:00 PM*	*6:00:00*		
1/18	*8:00 PM*	*11:00 PM*	*3:00:00*		
1/20	*3:00 PM*	*4:00 PM*	*1:00:00*		
1/23	*10:00 AM*	*5:00 PM*	*7:00:00*		
1/25	*7:00 PM*	*11:00 PM*	*4:00:00*		
1/28	*3:00 PM*	*4:00 PM*	*1:00:00*		
1/29	*1:00 PM*	*1:30 PM*	*0:30:00*		
2/1	*7:00 PM*	*10:00 PM*	*3:00:00*		
2/4	*2:00 PM*	*6:00 PM*	*4:00:00*		
2/6	*11:00 AM*	*2:00 PM*	*3:00:00*		
2/11	*11:00 AM*	*1:00 PM*	*2:00:00*		
2/18	*1:00 PM*	*5:00 PM*	*4:00:00*		
2/19	*11:30 AM*	*12:00 PM*	*0:30:00*		
2/25	*1:00 PM*	*3:00 PM*	*2:00:00*		
2/28	*11:00 AM*	*7:00 PM*	*8:00:00*		
3/2	*2:00 PM*	*8:00 PM*	*6:00:00*		

Date	In	Out	Hours		**Mark**
1/11	6:00 PM	10:00 PM	4:00:00		Time Spent:
1/12	1:00 PM	3:00 PM	2:00:00		67:00:00
1/15	2:00 PM	5:00 PM	3:00:00		
1/17	5:00 PM	10:00 PM	5:00:00		
1/20	12:00 PM	3:00 PM	3:00:00		
1/25	3:00 PM	4:00 PM	1:00:00		
1/27	6:00 PM	11:00 PM	5:00:00		
1/29	5:00 PM	7:00 PM	2:00:00		
1/30	2:00 PM	5:00 PM	3:00:00		
2/3	5:00 PM	8:00 PM	3:00:00		
2/5	2:00 PM	6:00 PM	4:00:00		
2/7	11:00 AM	1:00 PM	2:00:00		
2/11	11:00 AM	1:00 PM	2:00:00		
2/18	1:00 PM	5:00 PM	4:00:00		
2/19	11:00 AM	4:00 PM	5:00:00		
2/21	11:00 AM	3:00 PM	4:00:00		
2/28	11:00 AM	8:00 PM	9:00:00		
3/2	7:00 PM	9:00 PM	2:00:00		
3/3	11:00 AM	3:00 PM	4:00:00		

Team Member	Time Spent
Nick	73:00:00
Michael	61:00:00
Mark	67:00:00