BPS - Bat Preservation Society

Deliverable F: Prototype 1 and Customer Feedback

Team C03-14

Engineering Design GNG1103

Team Members:

Jett Nguyen (300418721)

Kian O'Reilly (300438056)

Jake Easthouse (300404970)

Syamalan Krishnakumar (300434069)

Tawfiq Muhmmad (300371225)

Faculty of Engineering

November 3rd, 2024

Abstract

In the previous deliverable, a final design was presented to our client to gather feedback. This feedback was used to generate our first prototype, Prototype I. This prototype will then be presented to the client and peers to gather more feedback, so additional modifications can be applied to our design to further meet the expectations for the project.

Table of Contents

List of Figures

List of Tables

1. Introduction

The purpose of this deliverable is to generate our first prototype based on our final conceptual design. This prototype consists of basic development of each subsystem that can be represented through cardboard designs and programming in Arduino IDE and python v3.14. Feedback that was given to us from the previous client meeting and from our peers will be discussed as well as an updated project schedule and prototype II test plan.

2. Prototype I

2.1. Bat Box

For the bat box component of the first prototype, we constructed a small-scale bat box made of scrap cardboard and tape. This to test how much time it would roughly take to construct the bat box and see if we needed to simplify the design so make its assembly quicker. The first prototype took only 30 minutes to complete proving that the bat box design is quite easy to assemble. There are problems with this test as cutting and taping cardboard together is faster than sawing and screwing together MDF, but it is not unrealistic to say it would only take four times the amount time to do so.

Figure 1. Prototype I Bat Box

We also learned how structurally stable the bat box design is. Despite Prototype I being made of a very unstable material such as cardboard, the bat box was quite sturdy. This shows that our design won't collapse on itself when constructed out of MDF.

2.2. Load Cell Sensor

After doing additional research on our load cell sensor, we have altered our detailed design drawings to make it easier to understand how our sensor functions. This load cell sensor measures the amount of bend the sensor experiences when weight is applied to the landing pad. Elevation blocks are placed on opposite ends to create gaps between the sensor and the landing panels, allowing for the sensor to experience maximum bending.

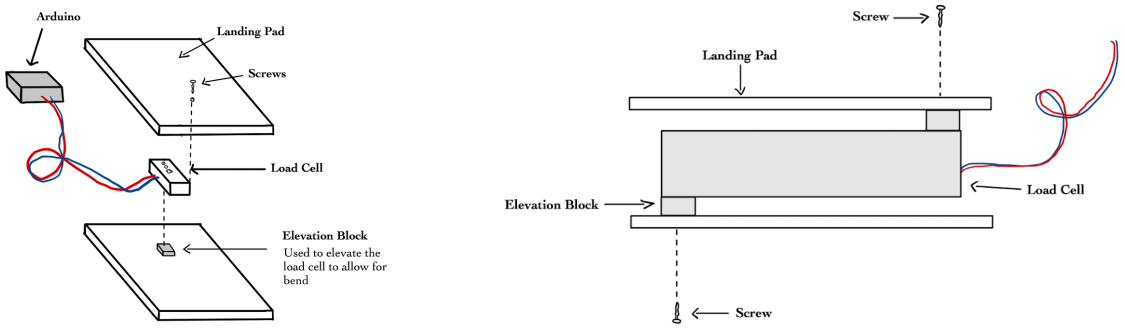

Figure 2. Updated Detailed Sensor Design

When searching for a code to be able to program it onto Arduino IDE, we found several YouTube tutorials which leading us to GitHub links with step-by-step tutorials on programming this code. Using this as inspiration, we were able to program a code for the load cell sensor on Arduino IDE. For our code, the HX711 load cell amplifier library is used to read the data from the load cell and applies a calibration factor to convert the reading into kilograms and displays the weight on an OLED screen. The code initializes the HX711 in OLED in setup(), then in loop(), it continuously measures the weight, corrects for negative values, and updates both the Serial Monitor and OLED display with the current weight.

For this project, we will not be displaying the measured weight on the screen. However, the change in measured weights will be used to track entries. For each time the measured weight experiences a change from 0, an entry will be tracked as this is considered to trigger the sensor. For our next prototype, we will modify our code to account for this.

sketch_nov3a.ino

```
#include " HX711.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define DOUT  3
#define CLK  2
HX711  scale(DOUT, CLK);

float weight;
float calibration_factor = 419640; // for me this value works just perfect 419640
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");
  scale.set_scale();
  scale.tare(); //Reset the scale to 0
  long zero_factor = scale.read_average(); //Get a baseline reading
  Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
  Serial.println(zero_factor);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
}
void loop() {
  scale.set_scale(calibration_factor); //Adjust to this calibration factor
  Serial.print("Reading: ");
  weight = scale.get_units(5);
    if(weight<0)
  {
    weight=0.00;
    }
  //Serial.print(scale.get_units(), 2);
 // Serial.print(" lbs"); //Change this to kg and re-adjust the calibration factor if you follow SI units like a sane person
  Serial.print("Kilogram:");
  Serial.print( weight);
  Serial.print(" Kg");
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

   display.clearDisplay();
  display.setTextSize(3);
  display.setCursor(0,0); // column row
  display.print("Weight:");
  display.setTextSize(3);
  display.setCursor(0,35);
  display.print(weight);
  display.setTextSize(3);
  display.setCursor(75,35);
  display.print("Kg");
 display.display();
 delay(100);
}
```

Figure 33. Load Cell Sensor Code

2.3. Wi-Fi Module

The Wi-Fi module for our system is intended to act as a module that will transmit our data from the box, into the cloud. This requires a Wi-Fi connection which is why we chose the ESP32 which has both Wi-Fi and Bluetooth capabilities. The module will take the data collected from the system and send it to an excel file in the cloud, where it will be stored and accessed within the data application.

We began to search for code inspiration within a site called Github, where we found multiple examples of the different uses of the ESP32. We realized that this component was also a microcontroller, but we continued to search for more code related to this product. We then went through YouTube we found a perfect walk-through of some very similar code.

Using the information that we gathered, we programmed a code in the Arduino IDE shown below. We first built a function that connects the microcontroller to the Wi-Fi and only if it is connected, can the rest of the code run. The next part was to send the inputted data. However, with no way to test the code, we're unsure that this part of the code will run properly.

sketch_oct31a.ino

```
#include <wifi.h>
#include <HTTPClient.h>

//This introduces what the wifi SSID and password are
const char* ssid = "Our_SSID";
const char* password = "Our_PASSWORD";

//This one introduces where the data will be sent
const char* excelDataSheet = "Our_Data_Sheet";

void setup() {
  Serial.begin(115200);

  //Connect to the wifi, enter the ssid and password
  Wifi.begin(ssid, password)
  Serial.print("Connecting to the Wifi");
  while (Wifi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
Serial.print("\nConnected to Wifi");
}

void loop() {
  //Check wifi connection status
  if (Wifi.status == WL_CONNECTED){

    HTTPClient http;

    //Say data type and URL
    http.begin(excelDataSheet);
    http.addHeader("Content-Type","the URL");

    //preparing the data to send
    int sensorValue = analogRead(34); //input of a data point, coming from sensor
```

sketch_oct31a.ino

```
  //Check wifi connection status
  if (Wifi.status == WL_CONNECTED){

    HTTPClient http;

    //Say data type and URL
    http.begin(excelDataSheet);
    http.addHeader("Content-Type","the URL");

    //preparing the data to send
    int sensorValue = analogRead(34); //input of a data point, coming from sensor
    String httpRequestData = "value=" + String(sensorValue);

    //Send HTTP POST request
    int httpResponseCode = http.POST(httpRequestData);

    //printing response from the sheet
    if (httpResponseCode > 0){
      String response = http.getString();
      Serial.print("HTTP Response Code: ");
      Serial.println(httpResponseCode);
      Serial.print("Response from server: ");
      Serial.println(response);
    } else {
      Serial.print("Error code: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  } else {
    Serial.println("WiFi Disconnected");
  }


}
```

Figure 44. WiFi Module Code

2.4. Camera

The camera component of this project involves a camera module that takes a picture every time the sensor is triggered. It then needs to upload the picture to a cloud storage system. Then it must be accessible from our application so the user can review the results to determine if it was a false flag or multiple bats.

Since we lack experience coding with Arduinos, we searched online for similar projects to learn from. Below are several screenshots of multiple parts of the code. The program so far can take a picture from the module and display it on the computer screen. It has several tabs in the Arduino IDE often with over two hundred lines of code.

From this we learned a lot more time will need to be invested in learning about the Arduino IDE and spent on coding. We do not have an Arduino or a camera module yet, so we are unable to test this code. We'll need to get as much code written as soon as possible to have time to test and alter the code.

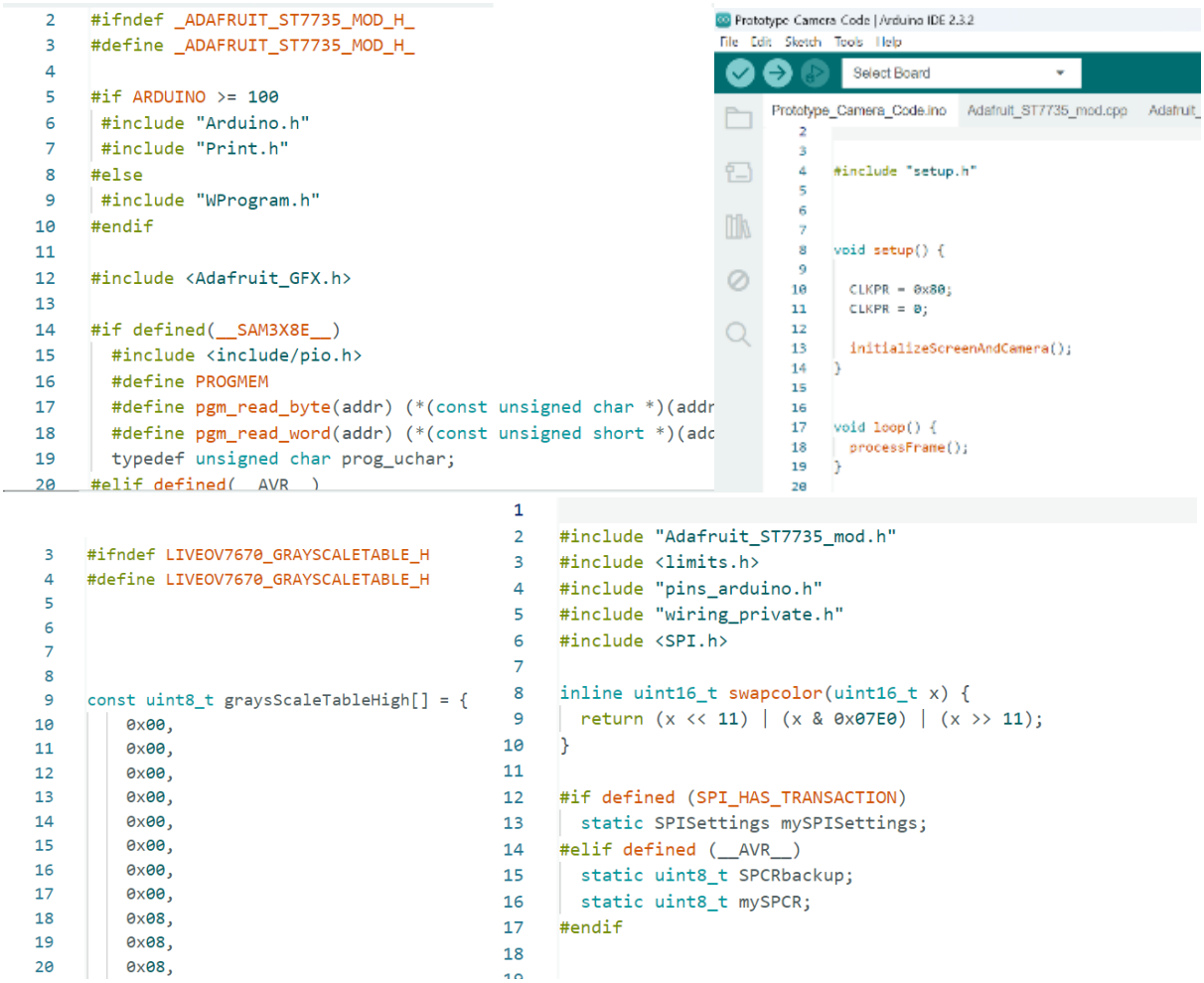

Figure 55. Camera Code

2.5. Data Application

For Prototype I, we began the development of our data application. To develop the graphical user interface (GUI), we used the tkinter library in the python v3.14 software. As we outlined in our final design, we decided that our application would have multiple interactive windows such as a main, login, data display, and data review/sort. In this first prototype, we have developed these windows that follow our application flowchart outlined in deliverable E.

For our next prototype, we will further develop our GUI and have our application access and display data from excel spreadsheets. We will also fix the formatting and have our application look more aesthetically pleasing.

Figure 66. Prototype I Data Application

3. Feedback

The first piece of feedback was from our client. Our client said that he had some concerns about the amount of storage that would be used up by taking video of every entry into the bat house. Based on the feed back we were provided here we decided to switch from using the camera to record footage of each entry that we could instead use it too take pictures of the entries, taking up significantly less data in the process.

For our first prototype some of the feedback we received is that while the weight sensor would work well for tracking only the entrances into the bat box it could have problems with external factors such as the wind. If the wind can exert enough force on the bat box to cause the sensor to trigger, then we could get multiple false entries in our data set. While this wouldn't be a problem for false numbers because of the camera system it would force the client to sort through all the images to verify entries which would be time consuming and expensive due to the hours required. However, according to Sahota (2022), the wind speeds are fastest during winter and lowest at summer (para. 3) and since our bat box primarily operates during summer it shouldn't be an issue.

Another piece of feedback we got was there might be a problem with battery power consumption since running the sensor constantly while being connected to Wi-Fi could drain the battery power too much for the one-month maintenance period that's expected. One solution that was provided by our group was to instead run a cable to the sensor and an outlet nearby. This wouldn't create any more restrictions on our product as it already needs to be placed somewhere with an internet connection

4. Updated Project Schedule

Table 1. Updated Project Schedule

Item	Description	Teammates	Date Due	Duration
Client Meeting 3	Team will meet with the client to discuss first prototype	Whole team	November 5	5 minutes
Camera Sensor Prototype 2	Code for the camera sensor will be completed to be able to take pictures when the sensor is triggered.	Jake	November 7	4 days
Bat Box Prototype 2	A new bat box made of MDF will be constructed in the makerspace.	Tawfiq	November 7	2 hours

Wi-Fi Module prototype 2	The code for the Wi-Fi module will be reworked to send the inputted data (the pictures and the timestamps) to a cloud storage system.	Kian	November 7	4days
Data Application Prototype 2	The data application U.I. will be updated and the live camera feature will be removed.	Jett	November 7	4 days
Load Cell Sensor 2	The code for the load cell sensor will be improved upon to remove any errors.	Syamalan	November 7	3 days
Prototype Testing	Tests will be performed on each aspect of the design to determine which aspects works and which need to change.	Whole Team	November 9	2 days
Prototyping Test Plan 2	The team will create a new testing plan for	Whole Team	November 10	>1 Hour
Bat Box Prototype 3	Based on the testing, construct a new prototype or modify the existing one to meet requirements of test.	Tawfiq	November 16	>3 hours
Data Application Prototype 3	Based on the testing, construct a new prototype or modify the existing one to meet requirements of test.	Jett	November 16	6 days
Load Cell Sensor 3	Based on the testing, construct a new prototype or modify the existing one to meet requirements of test.	Syamalan	November 16	6 days
Wi-Fi Module prototype 3	Based on the testing, construct a new prototype or modify the existing one to meet requirements of test.	Kian	November 16	6 days
Camera Module Prototype 3	Based on the testing, construct a new prototype or modify the existing one to meet requirements of test.	Jake	November 16	6 days

Prototype Testing	Tests will be performed on each aspect of the design to determine which aspects works and which need to change.	Whole Team	November 22	6 days
Design Day	Team will prepare a presentation and present it to the client at design day	Whole Team	November 27	5 days
User Manual	Team will write a manual for potential users.	Whole Team	Dec 3	6 days

5. Prototype II Test Plan

Table 2. Prototype II Test Plan

Test ID	Objective	Test Description and Materials needed	Description of results to be recorded	Test duration
1	Validating Spec 5	We will be testing the codes that we built last week, using the components that we purchased. These codes are separated between the Wi-Fi module, Camera, and the data application.	Check if the code successfully compiles, and if an entrance is tracked.	<1 week
2	Validating Spec 4	Assemble a new small-scale prototype made of a material that is harder than cardboard.	Record the amount of time it takes to assemble	<2 hours

6. Updated Bill of Materials

Table 3. Updated BOM

Item/material	Description	Quantity	Size/Dimensions	Est. Cost per	Total est. Cost
Legend					
Green	We have the item				
Yellow	We might not the item				
Grey	We might not need				
Bat Box					
MDF	box panels and landing pad	2	12" x 24	$ 3.50	$ 7.00
Screws	Used to assemble the bat box	1	1" long	$ 3.97	$ 3.97
Sensor					
Arduino Uno R3	Microcontroller for our sensor	1		$ 9.00	$ 9.00
Load Sensor	Used to track bat entires by attaching to landing pad	1	3.0" x 0.5" x 0.5"	$ 13.59	$ 13.59
Power Source	Power source for our arduino	1		$ 2.50	$ 2.50
AA Batteries	Gives power to power source	1x4		$ 3.98	$ 3.98
Camera	Takes snapshot when load cell is triggered. Used for additional information	1		$ 9.99	$ 9.99
Wi-Fi (ESP32)	Used to send data to data application	1		$ 12.00	$ 12.00
Bread Board	Electric circuit for our device	1		$ 5.00	$ 5.00
Wires	Used to wire sensors	3		$ 1.00	$ 3.00
Software					
Python v3.14	Programming software used to create our data application	1	N/A	N/A	free
tkinter library	Python library used in data application code	1	N/A	N/A	free
Arduino IDE	used to create our sensor	1	N/A	N/A	free
DFRobot_HX711.h	Library used for our load cell	1	N/A	N/A	free
ArduImageCapture plugin	Library used for our camera	1	N/A	N/A	free
Ada_fruitOV7670	Library used for our camera	1	N/A	N/A	free
ESP32	Library used for WiFi module	1	N/A	N/A	free
LiveOV7670	Library used for our camera	1	N/A	N/A	free
TOTAL (with tax)		$79.13			

Link to BOM

7. Conclusion

In conclusion, Prototype I successfully captures the focus of the client. Feedback that was gained from the client and peers will be implemented into the development of our design to produce Prototype II. For the next Prototype, additional features will be added to our data application and the physical wiring of our sensor will begin to be developed.

8. References

Electronic Clinic. (2020, April 27). *How to calibrate HX711 Load Cell Weight Sensor with Arduino*

https://www.youtube.com/watch?v=AwSBbMUPjSc

ESPAsyncWebServer by me-no-dev(2021), *GitHub repository* https://github.com/me-no-dev/ESPAsyncWebServer

Electronic Clinic. (n.d.). HX711 Load Cell Arduino, HX711 Calibration, Weighing Scale, Strain Gauge. Electronic Clinic.

https://www.electroniclinic.com/hx711-load-cell-arduino-hx711-calibration-weighing-scale-strain-gauge/

Sahota M. (2022, October 28th). *Canadian Seasonal Wind Speeds and The Anorra.* Borrum Energy solutions
Canadian Seasonal Wind Speeds and The Anorra

Synthetic State, (2021, August 26th), *Setting up an ESP32 with Arduino IDE*
https://www.youtube.com/watch?v=CD8VJl27n94