Smart Safe

Dennis Sorokin

I. Executive Summary

In order to mitigate package theft, the overall goal of our project is to build a receptacle for which packages can be safely held upon delivery. We aim to do this by creating a smart box capable of remotely notifying a recipient when a package is received. For safe access, a fingerprint scanner is used to unlock the safe for the recipient, and a keypad on the front of the box provides an alternate means of accessing the box. People with the code will be able to securely deliver items while the owner is away.

II. Motivation

For this term, we made a smart safe. It's similar to the Amazon drop boxes, but with a bit more functionality. It is able to lock after receiving information from an ultrasonic sensor. It has an IR receiver and a remote circuit to enter a code and open the box again. In addition, it can notify the owner when his/her package is delivered. In 2017, 11 million homeowners have had at least one package stolen. Also, 53 percent of homeowners are worried about leaving their packages outside their home because it might get stolen. 74 percent of packages are stolen during the day when homeowners are at work or outside their house. Victims claimed that the average value of stolen packages is between 50 and 100 dollars. This problem will cost sellers or carrier companies millions of

dollars every year. Sellers or carrier companies usually spend around 200 dollars to replace each stolen package.

Stealing packages is convenient for thieves during Christmas since Christmas has the highest shopping rate in the year. 70 percent of all homeowners expect to receive packages over the holiday season. As a result, FedEx and UPS deliver more than 30 million packages a day between Black Friday and Christmas. Therefore, we selected the smart safe project because it is important to protect those packages and save money for sellers, buyers, and carriers.

The smart safe project includes features that allows it to lock automatically whenever the delivery man puts a package in it. Moreover, it sends the owner a notification when the package is inside. We thought about adding a camera to take a picture if someone is trying to open it without the passkey or approve from the owner. The multiple parts of this project gave clear objectives for each of our teammates. While the diversity of this project was a great learning experience for our team.

In a nutshell, our project has very practical applications because it's targeting a real life issue that is facing a lot of house owners, a lot of deliver and selling companies. Our Final demo wasn't exactly what we want to deliver to the class because it was small box barely can fit a wallet, but at least we captured the big picture within the Budget.

III. Project Requirements

The Smart Safe that we are going to design and build shall have two main objectives the first is to keep the objects safe, secondly, it should alert the owner that a package has arrived. To keep the objects safe it will shall have several different security features the first is a keypad so only the owner can take the package out and another feature is an alert system that will send an email to the owner that their product has arrived and they can then take appropriate actions to either set a timer that will keep the box lockdown for that time or do some other action. Another feature that could be implemented is a camera to take pictures of any person attempting to take the package. To alert the owner that their package has arrived there will be a detection sensor tell the owner that their product has arrived.

Specifications

- Size 25" x 15" x 19"
- Material durable plastic or plexiglass
- Wiring and electronics need to enclosed and protected from the elements and possibly thieves trying to unlock the box
- Ideal color either brown, black or grey a dull color that blends in and might even be mistaken as a house decoration or a plant stand and not a box for packages but for this project it will most likely

be clear.

Image of Proposed box safe for packages

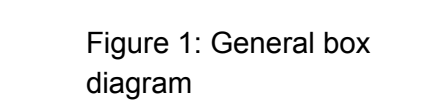

Figure 1: General box diagram

Functional decomposition

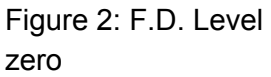

Figure 2: F.D. Level zero

Figure 3: F.D. Level one

For the first portion, this project shall be successful if the box will lock and can only open with the correct code. The main point of this is to prevent theft and not have prices for the products and have online stores charging more because when a product gets stolen the buyer can simply inform the store that they did not receive their package and the company usually up paying for the stolen item and this makes them increase the prices over time and we end up paying more for it.

For the second portion, this project shall be successful if we can communicate to the owner that their package has arrived either by using a weight sensor or an ultrasonic sensor that will detect when something is placed in front of it. Either sensor will then send an email to the owner and they can decide what to do from there.

1. First requirement: To have a locking system by keypad, the main objective of this project design is to keep the packages safe so that is the number one priority. 2. Second requirement: To have an alert system that will tell the user that a package has arrived so they can then take appropriate action. 3. Third requirement: make it blend in with the environment by painting it a darker color and making it inconspicuous looking. (may not be practical for this class) 4. Fourth requirement: make the cables and other electronic devices protected from the elements such as making sure it can withstand high temperature and freezing temperature as well as from anyone cutting the wires or tampering with it. 5. Fifth requirement: Made out of a durable material and preferably cut with the laser cutter for optimal precision. 6. Sixth requirement: Keep it under a certain 40-dollar budget 7. Seventh requirement: Neat appearance 8. Eighth requirement: Noise device when the package was added and the lock was closed securely. 9. Ninth constraint: time, has to be done in 8 weeks so need to plan and set up trello with all the tasks and details.

IV. Project planning

For this project, we used Trello to organize and track our tasks. Continuing on from the last term we had already completed a Gantt chart with most of our tasks already determined. At the beginning of this term, we compiled all of our tasks into the backlog and used the standard scum way of breaking down the tasks into what we would be working on for each sprint respectively. Since this was an
iterative process; we had several items that either didn't work out due to time constraints or were changed in order to better suit our needs. These items included changing from an IR type sensor to an ultrasonic sensor, changing our lock because the mechanism that we used to open and close the lid on our prototype wasn't compatible with the final box, and being able to successfully implement features such as the fingerprint scanner.

Outside of Trello, we tried to set aside time at least once a week to discuss where each member was with their assigned tasks and form a lookahead agenda. This way we could see if there were any items that would need to be rolled over to the next sprint or if more collaboration was needed. Due to our busy schedules, there were many times when we couldn't meet in person during class. In order to work around this, we used Google Hangout to post updates, communicate any changes, and notify the scrum master. In conjunction with Google Hangout, we also communicated on our phone's via group text, which allowed us to share information informally with each other.

Overall, we were able to follow the rundown of our Gantt chart pretty closely, but as we progressed, we found that many of our projected tasks could be broken down into smaller subtasks. This was particularly true for items that involved the prototyping and final construction. In the end, despite a few minor setbacks, we were able to meet all of our project requirements and have our project completed by the assigned deadline. Our only suggestions for improvement would be to breakdown smaller subtasks earlier in the project.

Sprint Goals

Our first sprint involved each member researching one of the four core components and acquiring hardware. During the second sprint, we began defining our code and building prototype circuits. The third sprint consisted of developing code and combining each of our circuit's. Our final sprint focused on assembling the final product and debugging code.

Areas of Specialization

Although we all maintained an active role in each aspect of this project, **Initial Gantt chart**

Final Gantt chart

V. Prototype Description

The way we decided to implement our solution is to use a fingerprint scanner along with a keypad this would ensure extra security that would prevent theft. We also had a notification system that would alert the owner that a package has arrived and a servo lock that kept the box lid secure. We had to combine the code from all the components together into a single file and make sure we used correct pins for each element.

The prototype used a metal geared servo instead of plastic geared one that came with our elegoo kits because it needed more power to open and close the lid as well as the plastic geared one was simply not reliable enough for this application.

The fingerprint scanner was made by adafruit and could store up to 162 fingerprints on its onboard flash memory. It also has a working temperature range of -20C to +50C that was ideal for our application as the user could have the lockbox in different climates.

We also decided to go with the IR receiver and remote that came in our kits to be used as the keypad. The IR receiver has a very small distance where it can actually connect with the signal from the remote so the way we went around that was by placing the remote right next to the receiver and that was it was almost like a keypad.

For the alarm system we used a buzzer connected to the arduino uno board. We has to be careful to use the correct pins for buzzer because the tone function that makes the sound only uses the analog signals on the board. We also made it play different sound and tones depending on the setting you want to have.
Figure 4: Arduino connected to Buzzer Figure 5: esp32 connection to sensor

Figure 6: prototype one Figure 7: Smart Safe Prototype

VI. Prototype Testing

To test our prototype each of circuits was tested separately to ensure that they were operating within standards prior to testing the project as a whole. This allowed us to identify problems arising from code or wiring efficiently without needing to isolate other components or perform re-work to undo construction.

Ultrasonic Sensor and Notification system: This circuit was responsible for identifying whether or not the box was empty or full. Once an item is placed inside, the expectation is that in under 10 min. a notification will be sent to recipients' email with a message saying "Package Received". Once our circuit was wired in place, we were able to test the functionality by placing an item in the box near the sensor. If the sensor was registering properly, we would receive a high value on Adafruit IO, using a gauge we had created to monitor this process. Once we saw that a high value was received, we continued to verify that it would return back to low once the item was moved away from the sensor. The high value would trigger a protocol established using IFTTT to send an email. Then it was just a matter of waiting to verify that our protocol had forwarded the email to the recipient.

VII. Technical Summary

In today's growing market of online shopping, there are countless boxes getting delivered to

your house. Our smart safe will help you protect those packages until you can retrieve them. We think that it will be able to save money for both the consumer and business owner, as well as deter would-be thieves. With a wide range of possible functionality, the smart safe will be able to effectively keep your packages locked up and alert you when they have been delivered. All of these factors are what make the smart safe a great project and hopefully an even better product.

VIII. Lessons learned

This term, timing was challenging for us. Each of us had a busy schedule that did not allow us to meet as much as we planned. The time issue did not let us test our project and make sure everything is going okay. In our presentation, not everything worked well as we expected. Moreover, Our Plastic servo in kit was very weak and small, so we had to buy more expensive one to make it lock the box and open it when we tell it to do that. The IR receiver and remote did not work efficiently with our project. I think next time we should use a keypad instead of a remote.

During prototyping we had an issue when we switched to the second box. The dimensions of the second box was a little off from the acceptable range it requires a lot of time and effort to make the second box open and close for us, so we decide to rotate our servo and make it latch the box instead of opening and closing the box. Other issues happened

On the Final demo date, we had some issues with our project. The IR receiver was still having issues receiving and sending data. This was causing we speculate that it was probably due in part to the nature of the remote system and the continuous looping code in the Arduino Uno. We had tested the box many times and the problem was very intermittent, and thus hard to pin down a direct solution. But we believe replacing the IR receiver and remote with a keypad would solve it.

IX. References

https://www.thepackageguard.com/wp-content/uploads/2017/11/Package-Guard_Press-Data - Sheet_Industry-Delivery-9.3.pdf

https://www.hackster.io/jrance/super-mario-theme-song-w-piezo-buzzer-and-arduino-1cc2e4

http://www.adafruit.com/products/751

X. Appendix

- HUZZAH32 esp32 featherboard (For notification system)
- Hitec 32645S HS-645MG High Torque 2BB Metal Gear Servo
- Arduino uno Board
- Buzzer for alarm
- Adafruit Fingerprint scanner
- Ultrasonic sensor
- Box for prototype

Code For Arduino Uno

```
#include
<Servo.h>

#include
<IRremote.h>

#include
<Adafruit_Fingerprint.h>

// For UNO and others without hardware serial, we must use software
serial...

// pin #2 is IN from sensor (GREEN
wire)

// pin #3 is OUT from arduino (WHITE wire)

// comment these two lines if using hardware
serial

SoftwareSerial mySerial(2,
3);

Adafruit_Fingerprint finger =
Adafruit_Fingerprint(&mySerial);

byte LedPin = 8; // indicate when unlocked
```

```
unsigned long Value1 = 0xFFC23D; // where XXXXXXXX is on our your remote's values // RA //code
0367

unsigned long Value2 = 0xFF22DD; // where XXXXXXXX is another button on your remote //
LA

unsigned long Value3 = 0xFF6897; // where XXXXXXXX is another button on your remote //
0

unsigned long Value4 = 0xFF52AD; // where XXXXXXXX is another button on your remote //
9

unsigned long Value5 = 0xFF7A85; // where XXXXXXXX is another button on your remote //
3

unsigned long Value6 = 0xFF42BD; // where XXXXXXXX is another button on your remote //
7

unsigned long Value7 = 0xFF5AA5; // where XXXXXXXX is another button on your remote //
5
unsigned long Value8 = 0xFF30CF; // number
1

unsigned long Value9 = 0xFF18E7; // number
2

bool flag;

bool flag2;

bool flag3;

bool flag4;

bool flag5;

bool flag6;

int RECV_PIN =
7;
```

```
byte piezoPin =
4;

IRrecv
irrecv(RECV_PIN);

decode_results
results;

Servo servo1;

// the setup routine runs once when you press
reset:

void setup()
{

Serial.begin(9600)
;

irrecv.enableIRIn(); // Start the
receiver

// initialize the digital pin as an
output.

servo1.attach(12); // attach servo to digital pin
12

servo1.write(0); // set it up to zero at
first

bool
flag=false;

bool
flag2=false;

bool
flag3=false;
```

```
bool
flag4=false;
bool
flag5=false;

bool
flag6=false;

pinMode(piezoPin,OUTPUT)
;

  pinMode(LED_BUILTIN,
  OUTPUT);

  Serial.begin(9600)
  ;

  while (!Serial); // For Yun/Leo/Micro/Zero/...

  delay(100);

  Serial.println("\n\nAdafruit finger detect
  test");

  // set the data rate for the sensor serial
  port

  finger.begin(57600);

  if (finger.verifyPassword())
  {

    Serial.println("Found fingerprint
    sensor!");

  } else
  {

    Serial.println("Did not find fingerprint sensor
    :(");
```

```
   while (1) { delay(1); }

  }

  finger.getTemplateCount()
  ;

  Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println("
  templates");

  Serial.println("Waiting for valid
  finger...");


} // the loop routine runs over and over again

forever:

void loop()
{

if (irrecv.decode(&results))
{

Serial.println(results.value,
HEX);
Serial.println(flag)
;

Serial.println(flag2)
;

Serial.println(flag3)
;

Serial.println(flag4)
;

Serial.println(flag5)
;

Serial.println(flag6)
```

```
;

irrecv.resume(); // Receive the next
value}

if(results.value ==
Value3){

flag=true;
}

if(results.value ==
Value5){

flag2=true;}

if(results.value ==
Value6){

flag3=true;}

if(results.value ==
Value7){

flag4=true;}

if(results.value ==
Value8){

  flag6=true;}

if(flag5 == false && flag6 ==
true){

 getFingerprintIDez();

  delay(100); //don't ned to run this at full speed.

  if(flag5 == true)
  {

    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level) }
```

```
} if(flag == true && flag2 == true && flag3 == true && flag4 ==

true) {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)


} if((results.value == Value1) && ((flag == true && flag2 ==true && flag3 == true && flag4 ==true) ||

(flag5 == true))) {

servo1.write(30)
;

   //soundFX(3000.0,30+200*(1+sin(millis()/5000))); // wonky sonic
   screwdriver

   //soundFX(3000.0,30); // sonic
   screwdriver

   soundFX(100.0,30.0); // ray gun

   //soundFX(3.0,30.0); // star
   trek

   //soundFX(1.0,30.0); // star trek
   high


} if ((results.value == Value2) && ((flag== true && flag2 == true && flag3 == true && flag4 ==true)

|| (flag5 == true))){

servo1.write(155)
;

   //soundFX(3000.0,30+200*(1+sin(millis()/5000))); // wonky sonic
   screwdriver

   //soundFX(3000.0,30); // sonic
   screwdriver

     soundFX(100.0,30.0); // ray gun
```

```
  //soundFX(3.0,30.0); // star
  trek

  //soundFX(1.0,30.0); // star trek
  high


} else if (results.value ==

Value4){

flag = false;

flag2=false;

flag3=false;

flag4=false;
flag5=false;

flag6=false;

digitalWrite(LED_BUILTIN, LOW); // turn the LED on (low is the voltage level)


} } void soundFX(float amplitude,float

period){

 int
 uDelay=2+amplitude+amplitude*sin(millis()/period);

 for(int
 i=0;i<5;i++){

  digitalWrite(piezoPin,HIGH)
  ;

  delayMicroseconds(uDelay)
  ;

  digitalWrite(piezoPin,LOW);
```

```
  delayMicroseconds(uDelay);
  }

} uint8_t

getFingerprintID() {

  uint8_t p =
  finger.getImage();

  switch (p) {

    case
    FINGERPRINT_OK:

      Serial.println("Image
      taken");

      break;

    case
    FINGERPRINT_NOFINGER:

      Serial.println("No finger
      detected");

      return
      p;

    case
    FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication
      error");

      return
      p;

    case
    FINGERPRINT_IMAGEFAIL:

      Serial.println("Imaging
```

```
    error");

    return
    p;

  default
  :

    Serial.println("Unknown
    error");

    return
    p;

}

// OK success!

p =
finger.image2Tz();

switch (p) {

  case
  FINGERPRINT_OK:

    Serial.println("Image
    converted");

    break;

  case
  FINGERPRINT_IMAGEMESS:

    Serial.println("Image too
    messy");

    return
    p;

  case
  FINGERPRINT_PACKETRECIEVEERR:
```

```
    Serial.println("Communication
    error");

    return
    p;

  case
  FINGERPRINT_FEATUREFAIL:

    Serial.println("Could not find fingerprint
    features");

    return
    p;
  case
  FINGERPRINT_INVALIDIMAGE:

    Serial.println("Could not find fingerprint
    features");

    return
    p;

  default
  :

    Serial.println("Unknown
    error");

    return
    p;

}

// OK
converted!

p =
finger.fingerFastSearch();

if (p == FINGERPRINT_OK)
```

```
{

  Serial.println("Found a print
  match!");

} else if (p == FINGERPRINT_PACKETRECIEVEERR)
{

  Serial.println("Communication
  error");

  return p;

} else if (p == FINGERPRINT_NOTFOUND)
{

  Serial.println("Did not find a
  match");

  return p;

} else
{

  Serial.println("Unknown
  error");

  return p;

}

// found a
match!

Serial.print("Found ID #");
Serial.print(finger.fingerID);

Serial.print(" with confidence of ");
Serial.println(finger.confidence);
return
finger.fingerID;
```

```
} // returns -1 if failed, otherwise returns ID

#

int getFingerprintIDez()
{

  uint8_t p =
  finger.getImage();

  if (p != FINGERPRINT_OK) return -1;

  p =
  finger.image2Tz();

  if (p != FINGERPRINT_OK) return -1;

  p =
  finger.fingerFastSearch();

  if (p != FINGERPRINT_OK) return -1;

  // found a
  match!

  Serial.print("Found ID #");
  Serial.print(finger.fingerID);

  Serial.print(" with confidence of ");
  Serial.println(finger.confidence);

  flag5=true;

  //digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)

  //delay(1000); // wait for a second

  //digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW

   return
   finger.fingerID;
```

} **Code for Ultrasonic Sensor/ Notification System:**

```
#include "config.h" bool EMAIL = false; //
set up the 'digital' feed AdafruitIO_Feed *digital =
io.feed("digital");


// defines pins
numbers const int
trigPin = 27; const int
echoPin = 32; //
defines variables
long
duration; int
distance;


void
setup() {
  // set input pinMode(EMAIL, INPUT); pinMode(trigPin,
  OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input


      // start the serial
  connection
  Serial.begin(115200);


  // connect to io.adafruit.com
  Serial.print("Connecting to Adafruit
  IO"); io.connect();


  // wait for a connection while(io.status()
  < AIO_CONNECTED) {
  Serial.print("."); delay(500); }
  // we are connected
  Serial.println();
  Serial.println(io.statusText())
  ;

//pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
```

```
//pinMode(echoPin, INPUT); // Sets the echoPin as an Input
//Serial.begin(115200); // Starts the serial communication }

void loop() {
      io.run(); // Clears the trigPin digitalWrite(trigPin, LOW); // Sets the
 trigPin on HIGH state for 2 second delayMicroseconds(2000000);
 digitalWrite(trigPin, HIGH); delayMicroseconds(10); digitalWrite(trigPin,
 LOW); // Reads the echoPin, returns the sound wave travel time in
 microseconds duration = pulseIn(echoPin, HIGH); // Calculating the
 distance distance= duration*0.034/2; // Prints the distance on the Serial
 Monitor
    Serial.print("Distance:
 ");
 Serial.println(distance); if
 (distance < 10)
 {
 bool EMAIL = true; distance = 0; // Sets distance at a
 stable value // save the current state to the 'digital' feed
 on adafruit io Serial.print("sending button -> ");
 Serial.println(EMAIL); digital->save(EMAIL); //
 delayMicroseconds(20000000);

 } if (distance >=
 10) {
 bool EMAIL = false;
 digital->save(EMAIL);
 }


}
```