

RASHID'S INTERNSHIP REPORT

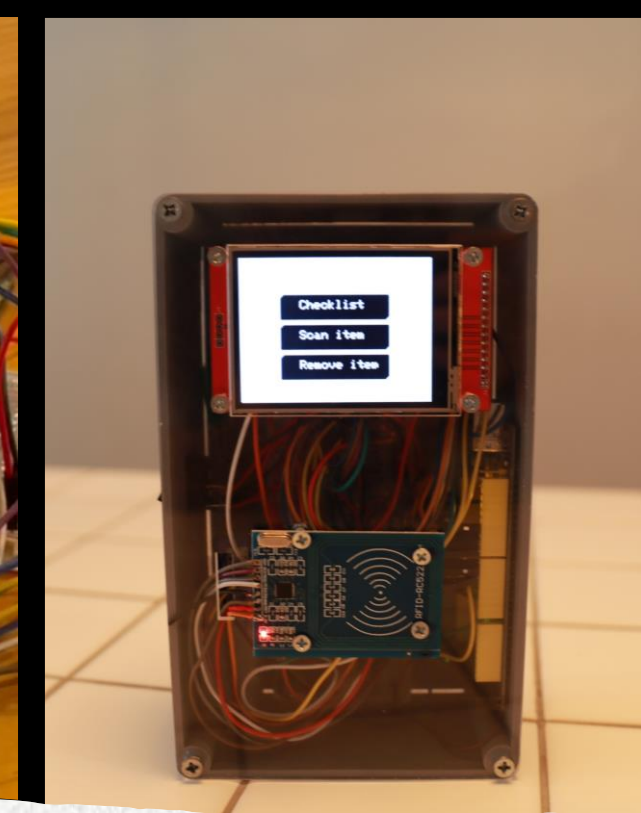
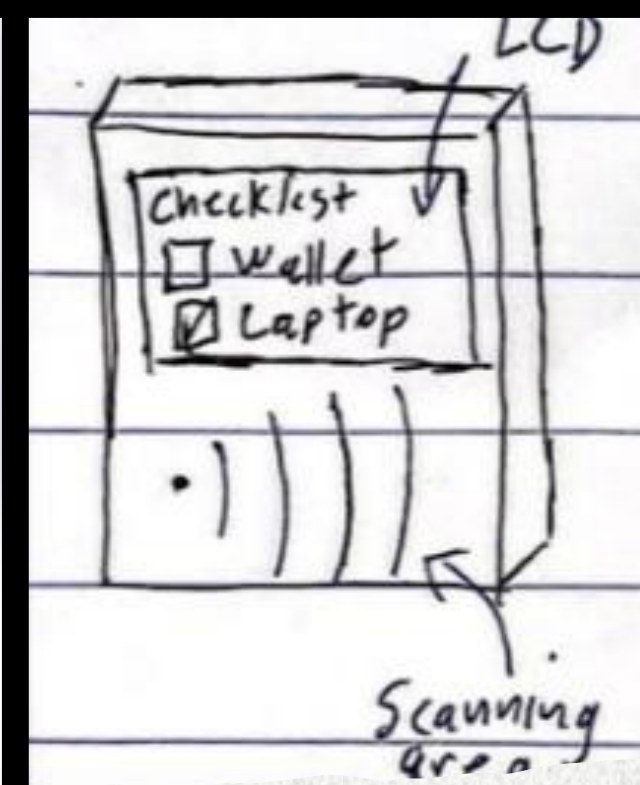
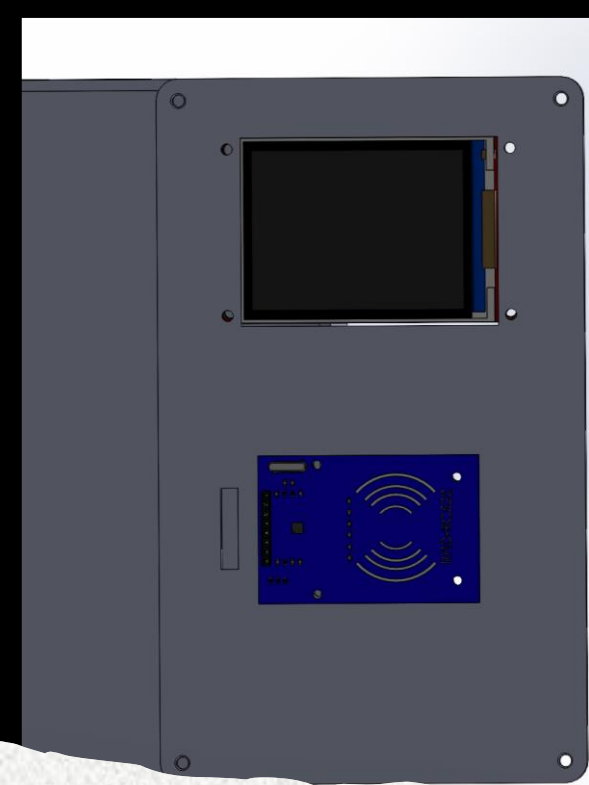
HNDI Summer 2023 - Rashid Abder





About me

- I am a sophomore studying Computer Engineering at UCSD
- I enjoy indoor bouldering, collecting music CDs and working on random small projects in my free time.



Internship project

Problem definition

- It is hard to find a convenient way for a user to check that they are bringing everything with them.
- Solutions usually rely on the user's memory or making traditional handwritten/digital checklists

Leaving for work in the morning...



5 minutes later, you realize you forgot something



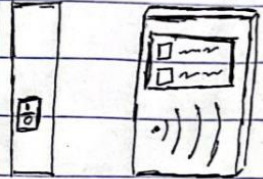
Idea

- An electronic device that acts like an enhanced checklist, hung on the door or wall of the user's home. The user would scan their items before leaving and update the checklist.
- Placing it on one of these locations incorporates the checklist in the routine of the user, making it more natural to use.

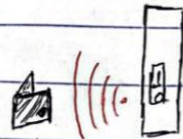
Concept sketches

PROCESS

- 1) Make sure device is on
- 2) Bring object close to reader
- 3) Have item checked off



Screen should be on



S-7 con



FIRST TIME SETUP

- | | | |
|--------------------------------------|--------------------------------------|---|
| 1) Device enters "discovering" state | 2) User enters name for item scanned | 3) User saves the item on the checklist |
|--------------------------------------|--------------------------------------|---|



If item already
in list:

- Do nothing

18 items not
in list'

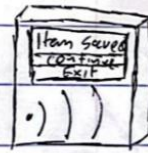
- Move to 2



- Next button

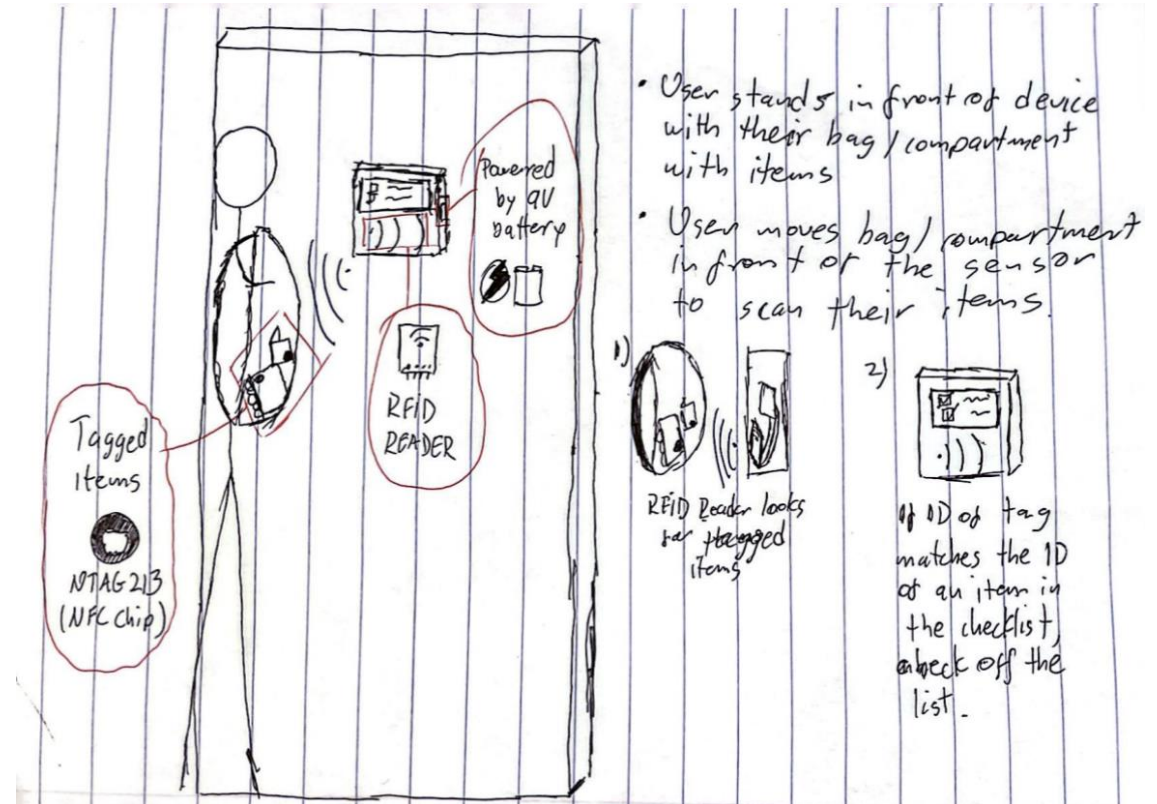
On-screen Keyboard

allows user to enter alphanumeric characters

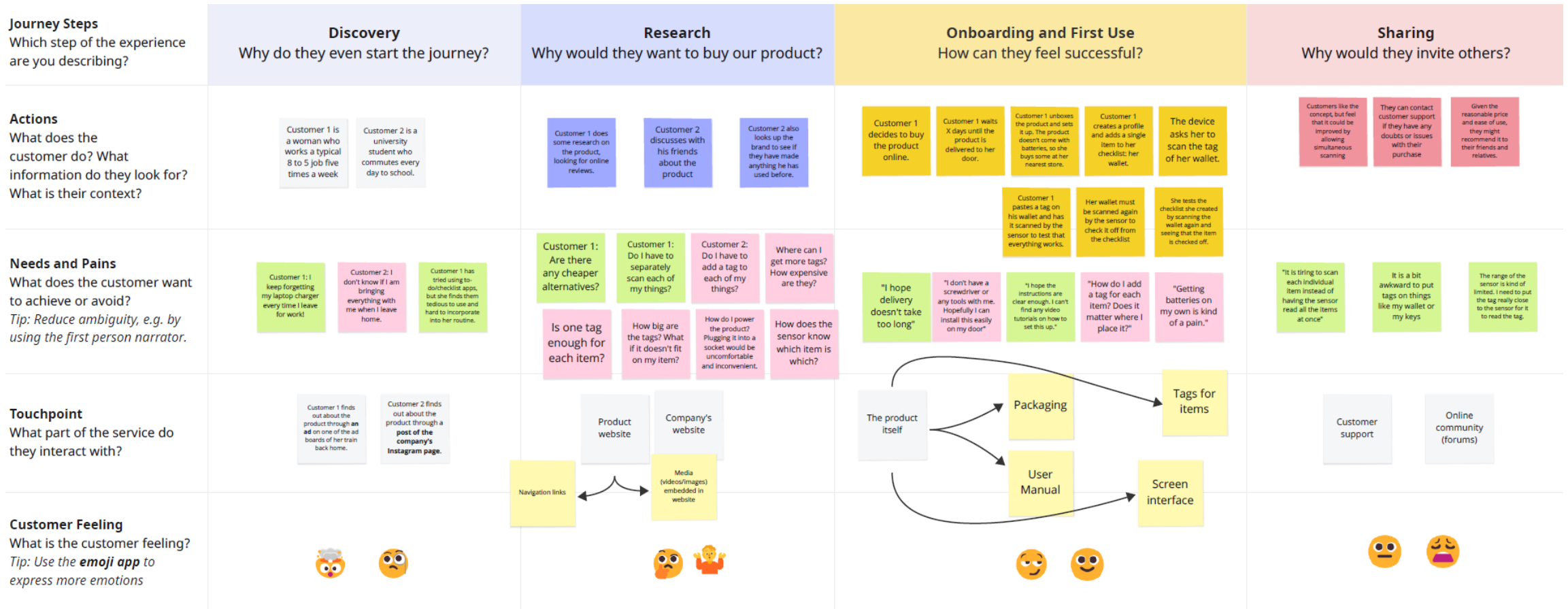


Repeat
Process

Finish
C/ten



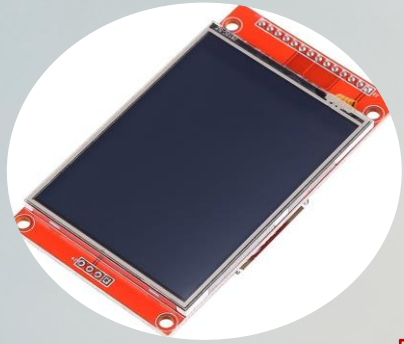
UX Map



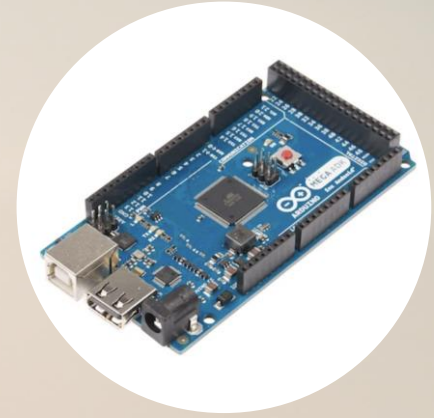
Prototype



Critical components

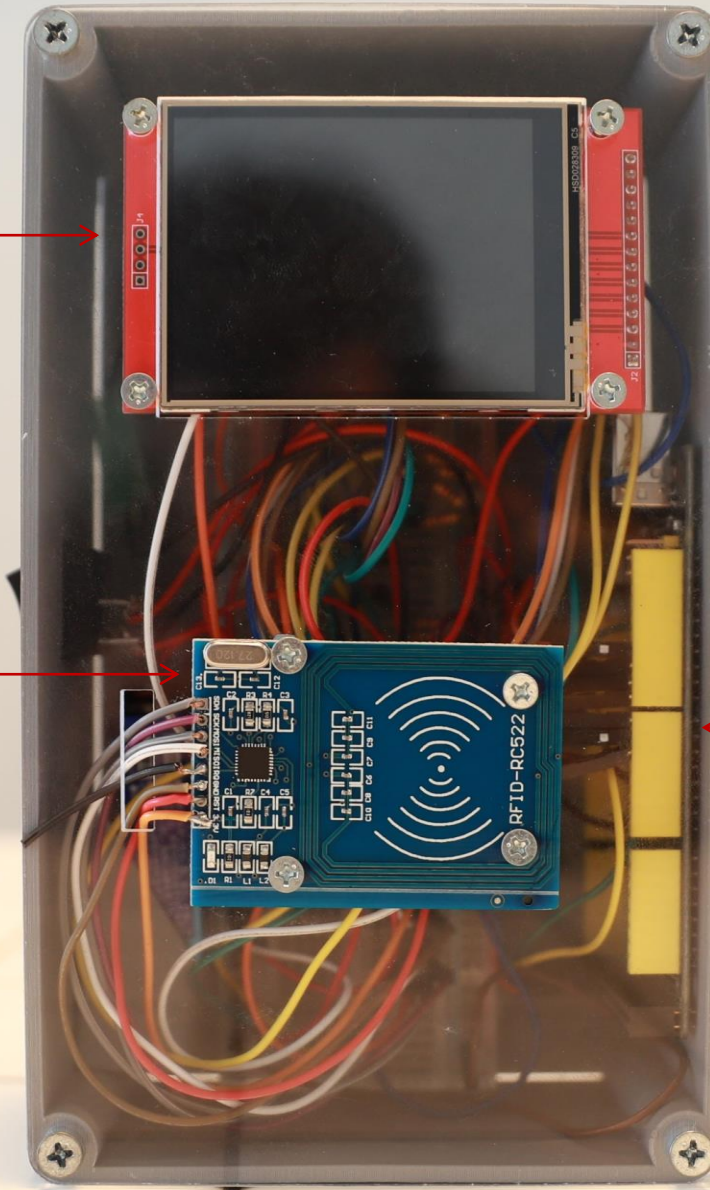
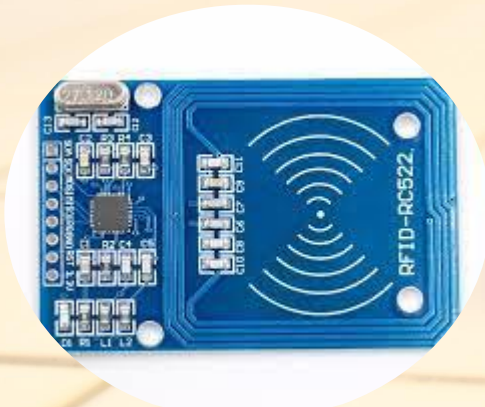


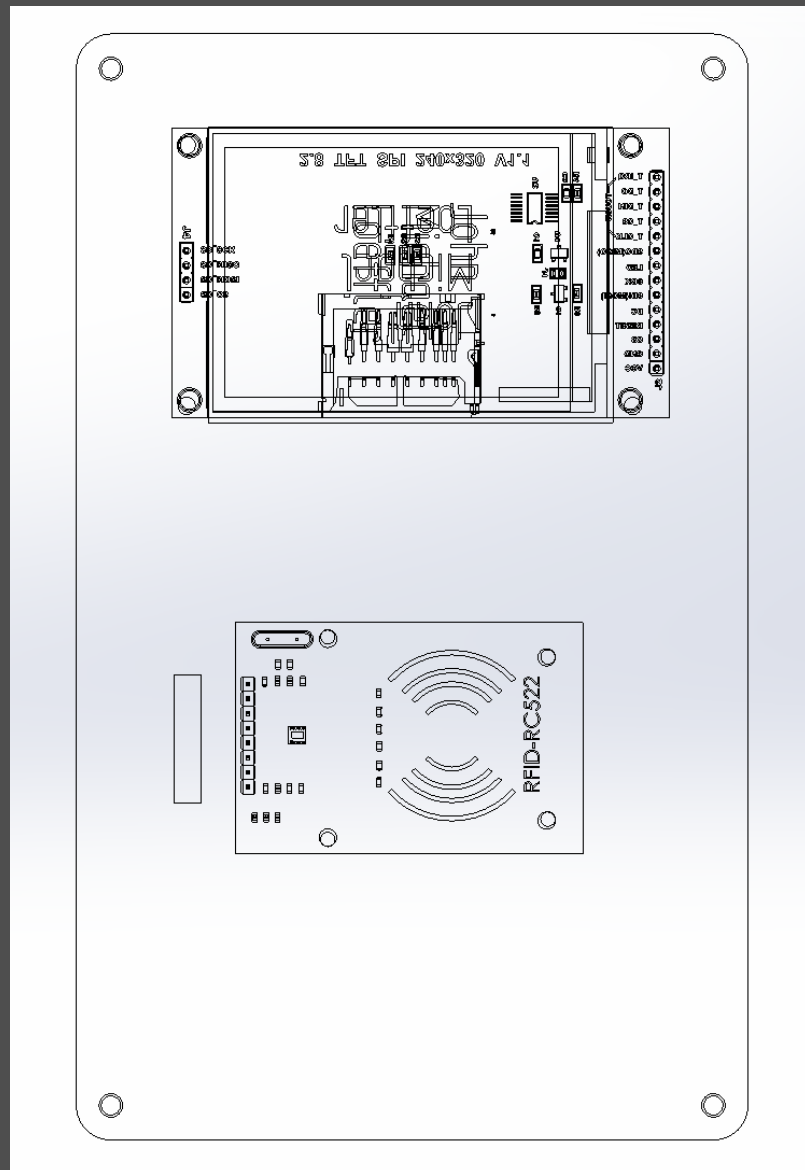
2.8" Touchscreen



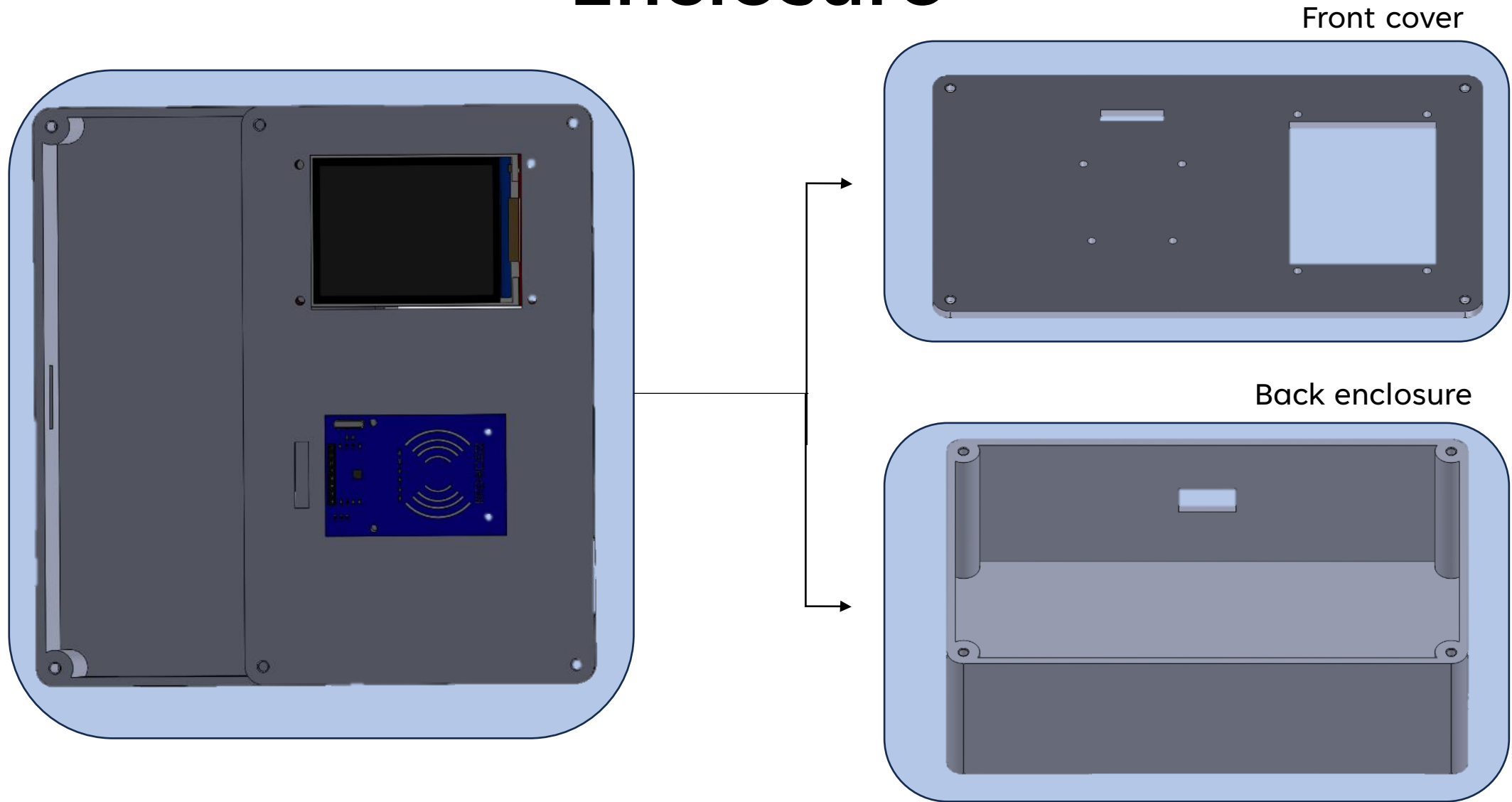
Arduino Mega

RFID Reader





Enclosure




```

5 #include <Arduino.h>
6 #include <Adafruit_NeoPixel.h>
7 #include <Adafruit_SSD1306.h>
8 #include <Adafruit_GFX.h>
9
10 // Pin that controls backlight with help of MOSFET
11 #define backlightControl 5
12
13 String UID;
14 int totalItems = 0;
15 int timeToSleep = 10000;
16 bool lowPower = false;
17
18 // Different states for the device
19 enum device {
20   MENU,
21   M1C0M,
22   M0M,
23   SCAM,
24   LOW_Power,
25   CHECKLIST,
26   REMOVE
27 };
28
29 // Starting state
30 enum device state = MENU;
31
32 void resetSPI() {
33   pinMode(SS, OUTPUT);
34   digitalWrite(SS, LOW);
35   digitalWrite(SS, HIGH);
36   digitalWrite(SS, LOW);
37   digitalWrite(SS, HIGH);
38 }
39
40 void setup() {
41   pinMode(backlightControl, OUTPUT);
42   digitalWrite(backlightControl, HIGH);
43
44   Serial.begin(9600);
45   Serial.print("JUST STARTED");
46   SPI.begin();
47   WDT.reset();
48   TFT.begin();
49
50   // Disable unnecessary features to save power
51   power_wdt_disable();
52   power_uart_disable();
53   power_timer1_disable();
54   power_timer2_disable();
55   power_timer3_disable();
56   power_timer4_disable();
57   power_timer5_disable();
58   power_timer6_disable();
59 }
60
61 void loop() {
62   Serial.println("LOOP RUNNING");
63 }

```

MAIN

```

344 bool flag;
345 String text = "Items";
346 tft.setCursor(0, 0);
347 tft.setTextColor(ILI9341_BLACK);
348 int16_t x = 10;
349 int16_t y = 5;
350
351 for (int i = 0; i < totalItems; i++) {
352   tft.setCursor(x + 50, y + 10);
353   tft.println(KEYS[i]);
354   Serial.println(KEYS[i]);
355   tft.setCursor(x + 50, y);
356   drawButton(x, y, 30, 30, ILI9341_BLACK, ILI9341_BLACK, ILI9341_BLACK);
357   y += 40;
358 }
359
360 removeItemScreen();
361
362 bool removeItemScreen() {
363   int16_t startX = 5;
364   int16_t x = 10;
365   int16_t buttonX = 310 - 30;
366   int16_t buttonY = 230 - 30;
367   drawButton(buttonX, buttonY, 30, 30, ILI9341_BLACK, ILI9341_BLACK, ILI9341_BLACK);
368   while (true) {
369     if (TouchButton(buttonX, buttonY, 30, 30)) {
370       state = MENU;
371       break;
372     }
373     for (int i = 0; i < totalItems; i++) {
374       if (TouchButton(x, startY + 40 * i, 30, 30)) {
375         return confirmRemoval(i);
376       }
377     }
378   }
379 }
380
381 bool confirmRemoval(int i) {
382   clearScreen();
383   tft.setTextSize(2);
384   tft.setTextColor(ILI9341_BLACK, ILI9341_WHITE);
385   tft.setCursor(0, 0);
386   String line1 = "Remove " + KEYS[i] + " ?";
387   clearScreen();
388   tft.setCursor(0, 0);
389   tft.println(line1);
390   if (confirmationButtons()) {
391     Serial.println("YES");
392     totalItems--;
393     Serial.println("REMOVED");
394     return true;
395   } else {
396     Serial.println("NOTHING");
397     return false;
398   }
399 }
400
401 void clearScreen() {
402   tft.fillRect(0, 0, tft.width(), tft.height(), ILI9341_WHITE);
403   tft.setCursor(0, 0);
404   tft.println("");
405 }

```

SCREEN

```

210 tft.print(textBuffer);
211
212
213 bool confirmationButtons() {
214   tft.setTextColor(0xffff, 0x03e0);
215   drawButton(120, 120, 60, 30, 0x8888, 0xffff, 0x03e0);
216   tft.setCursor(130, 130);
217   tft.print("Yes");
218   tft.setTextColor(0xffff, 0xf800);
219   drawButton(120, 160, 60, 30, 0x8888, 0xffff, 0xf800);
220   tft.setCursor(135, 170);
221   tft.print("No");
222   return getKeyPressConfirmation();
223 }
224
225 bool getKeyPressConfirmation() {
226   while (true) {
227     if (TouchButton(120, 120, 60, 30)) {
228       Serial.println("YES");
229       return true;
230     } else if (TouchButton(120, 160, 60, 30)) {
231       Serial.println("NO");
232       return false;
233     }
234   }
235 }
236
237 byte TouchButton(int x, int y, int w, int h) {
238   int X, Y;
239   // Retrieve a point
240   TS_Point p = ts.getPoint();
241   if (p.z > 10) {
242     Y = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);
243     X = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
244     Serial.print("Pressure = ");
245     Serial.print(p.z);
246     Serial.print(", x = ");
247     Serial.print(X);
248     Serial.print(", y = ");
249     Serial.print(Y);
250     delay(30);
251     Serial.println();
252   }
253 }

```

TOUCHSCREEN FEATURES

```

9 int sizeScanned = 0;
10
11 void writeEEPROM(int addrOff, const String data) {
12   byte length = str.length();
13   EEPROM.write(addrOff, length);
14   for (int i = 0; i < length; i++) {
15     EEPROM.write(addrOff + 1 + i, str[i]);
16   }
17 }
18
19 String readStringFromEEPROM(int addrOffset) {
20   int newAddrOff = addrOffset;
21   int maxLength = EEPROM.read(addrOffset);
22   char data[newAddrOff + 1];
23   for (int i = 0; i < newAddrOff; i++) {
24     data[i] = EEPROM.read(addrOffset + 1 + i);
25   }
26   data[newAddrOff] = '\0';
27   return String(data);
28 }
29
30 void insert(String key, String value) {
31   // Check if the key already exists
32   for (int i = 0; i < size; i++) {
33     if (KEYS[i].equals(key)) {
34       VALUES[i] = value; // Update value if the key already exists
35       return;
36     }
37   }
38   // Key doesn't exist, add it to the arrays
39   KEYS[size] = key;
40   VALUES[size] = value;
41   size++;
42 }
43
44 String lookup(String key) {
45   for (int i = 0; i < size; i++) {
46     if (KEYS[i].equals(key)) {
47       return VALUES[i]; // Return the value if the key is found
48     }
49   }
50   // Key not found, return a space
51   return " ";
52 }
53
54 void delete(String value) {
55   for (int i = 0; i < size; i++) {
56     if (VALUES[i].equals(value)) {
57       // Shift elements to the left to fill the gap left by the deleted key
58       for (int j = i; j < size - 1; j++) {
59         KEYS[j] = KEYS[j + 1];
60         VALUES[j] = VALUES[j + 1];
61       }
62       size--;
63       return; // Exit the function after deletion
64     }
65   }
66 }

```

DATA STRUCTURE CODE FOR STORAGE

```

42 UID = "";
43 delay(5000);
44 clearScreen();
45 return false;
46 }
47 }
48 }
49 }
50 }
51 bool readTagChecklist() {
52   if (RFID_READER.PICC_IsNewCardPresent()) {
53     Serial.println("No tag found!");
54     return false;
55   }
56 }
57 if (RFID_READER.PICC_ReadCardSerial()) {
58   Serial.println("Can't get data on tag!");
59   return false;
60 }
61
62 else {
63   Serial.print("UID:");
64   for (byte i = 0; i < RFID_READER.uid.size; i++) {
65     // If byte less than 0x10, print '0'. Print ' ' otherwise
66     Serial.print(RFID_READER.uid.uidByte[i] < 0x10 ? "0" : " ");
67     Serial.print(RFID_READER.uid.uidByte[i], HEX);
68     UID += String(RFID_READER.uid.uidByte[i], HEX);
69   }
70
71   if (checkIfDuplicate(UID, VALUES) == -1 || totalItems < 1) {
72     Serial.println();
73     RFID_READER.PICC_HaltA();
74     return true;
75   }
76   else if (checkIfDuplicate(UID, VALUES) >= 0) {
77     Serial.println(checkIfDuplicate(UID, VALUES));
78     clearScreen();
79     itemAlreadyExistsScreen();
80     UID = "";
81     delay(5000);
82     clearScreen();
83     return false;
84   }
85 }
86 }
87
88 void scanChecklist() {
89   int16_t x = 5;

```

RFID SCANNER

Code

Code is split into 5 different files, each containing the functionality of a different module/feature of the prototype.

Video

