Light Sensor Alarm Code

const int buttonRed = 8;

const int buttonBlue = 9;

const int buttonWhite = 10;

const int buttonGreen = 11;

const int ledRed = 6;

const int ledGreen = 7;

const int buzzerpin = 13;

const int LEDpin = 5;

const int displayButton = 12;

const int photoSensorState = 4;

int password[4] = {1, 2, 3, 4};

volatile int entered[4] = {0, 0, 0, 0};

volatile int state = LOW;

volatile int pushButtonCounter = 0;

bool boxNeverOpen = 1;

bool alarmDisabled = 0;

unsigned long correctPasswordEntered = 0;

unsigned int hack = 0;

volatile int eventNumber = 0;

struct timerInfo {

 String action;

 unsigned int timeOfEvent;

};

timerInfo timerInfoArray[20];

unsigned int timer = 0;

void setup() {

 Serial.begin(9600);

 pinMode (buttonWhite, INPUT);

 pinMode (buttonGreen, INPUT);

 pinMode (buttonRed, INPUT);

 pinMode (buttonBlue, INPUT);

 pinMode (ledRed, OUTPUT);

 pinMode (ledGreen, OUTPUT);

 pinMode (buzzerpin, OUTPUT);

 pinMode (LEDpin, OUTPUT);

 pinMode (displayButton, INPUT);

 pinMode (photoSensorState, INPUT);

 attachInterrupt (digitalPinToInterrupt(2), buttonPressed, RISING);

 attachInterrupt (digitalPinToInterrupt(3), photoResistor, CHANGE);

 cli();

 TCCR1A = 0; // clears register

 TCCR1B = 0; // clears register

 TIMSK1 |= (1 << TOIE1);  //enables overflow interrupt

 TCCR1B |= \_BV(CS12);  //prescales to 16MHz/256 = 62500 Hz.

 sei();

}

ISR(TIMER1\_OVF\_vect) // timer 1 can count to 65535 and the clock frequency is set to 62500hz or 0.000016 seconds per timer increment. 0.000016 \* 65535 = 1.04856 seconds

{

timer++; // timer increments every 1.04856 seconds

}

long timerr(){

 unsigned long timerValue = 0;

 timerValue = ((timer \* 65535) + TCNT1) / 62500;

 return (timerValue);

}

void loop() {

 if ((state == HIGH) && (((millis() - correctPasswordEntered) > 10000) || (correctPasswordEntered == 0))) {

   detachInterrupt(digitalPinToInterrupt(2)); //so other lights and buttons cannot be pressed thus risking change of state

   digitalWrite(LEDpin, state);

   digitalWrite(buzzerpin, state);

   tone(buzzerpin, 100); // Send 1KHz sound signal...

   delay(1000);        // ...for 1 sec

   noTone(buzzerpin);     // Stop sound...

   delay(1000);        // ...for 1sec

   if ((digitalRead(displayButton)) == HIGH) { //hold down button due to delays

     displayTimedEvents(); //polling used for display events because it should override anything and interrupts were detached

   }

 }

 if (pushButtonCounter == 4) {

   bool check = compareArrays();

   if (check == 1) {

     correctPasswordEntered = timerr();

     timerInfoArray[eventNumber].action = "Correct Password Entered - Alarm Disabled";

     timerInfoArray[eventNumber].timeOfEvent = correctPasswordEntered;

     eventNumber ++;

     while ((timerr() - correctPasswordEntered) < 10) {

       detachInterrupt(digitalPinToInterrupt(3));

       digitalWrite (ledGreen, HIGH); // turns green LED on for a 10 second time period after correct password is entered

     }

     digitalWrite (ledGreen, LOW); // green LED turns off after the 5 second safe zone

     timerInfoArray[eventNumber].action = "Alarm Enabled";

     timerInfoArray[eventNumber].timeOfEvent = timerr();

     eventNumber ++;

     hack = millis();

     attachInterrupt (digitalPinToInterrupt(3), photoResistor, CHANGE);       //      Serial.print("state after attach interrupt: ");

     if (digitalRead(photoSensorState) == HIGH) { //if the box if left open after safe zone expires, alarm will sound

       state = HIGH;

     }

     //timerInfoArray[eventNumber-1].action = "err";

     //state = LOW; // hack for button

     // boxNeverOpen = 1;

     for (int i = 0; i < 4; i ++) {

       (entered[i] = 0); //resets password after correct entry

     }

   }

   else if (check == 0) {

     timerInfoArray[eventNumber].action = "Wrong Password Entered";

     timerInfoArray[eventNumber].timeOfEvent = timerr();

     eventNumber ++;

     wrongPassword();

   }

   pushButtonCounter = 0;

 }

}

void buttonPressed () {

 static unsigned long last\_interrupt\_time = 0; // changes to interrupt time at end of ISR. Thus if it glitches and triggers multiple times successively, it won't go into following loop because that glitch happens in less than 200ms

 unsigned long interrupt\_time = millis();

 if (interrupt\_time - last\_interrupt\_time > 200) {

   if ((digitalRead(buttonRed)) == HIGH) {

     entered[pushButtonCounter] = 1;

     pushButtonCounter ++;

     timerInfoArray[eventNumber].action = "Red Button Pressed";

     timerInfoArray[eventNumber].timeOfEvent = timerr();

     eventNumber ++;

     Serial.println("red");

   }

   if ((digitalRead(buttonBlue)) == HIGH) {

     entered[pushButtonCounter] = 2;

     pushButtonCounter ++;

     timerInfoArray[eventNumber].action = "Blue Button Pressed";

     timerInfoArray[eventNumber].timeOfEvent = timerr();

     eventNumber ++;

     //currentState();

     Serial.println("blue");

   }

   if ((digitalRead(buttonWhite)) == HIGH) {

     entered[pushButtonCounter] = 3;

     pushButtonCounter ++;

     timerInfoArray[eventNumber].action = "White Button Pressed";

     timerInfoArray[eventNumber].timeOfEvent = timerr();

     eventNumber ++;

     //currentState();

     Serial.println("white");

   }

   if ((digitalRead(buttonGreen)) == HIGH) {

     entered[pushButtonCounter] = 4;

     pushButtonCounter ++;

     timerInfoArray[eventNumber].action = "Green Button Pressed";

     timerInfoArray[eventNumber].timeOfEvent = timerr();

     eventNumber ++;

     //currentState();

     Serial.println("green");

   }

   if ((digitalRead(displayButton)) == HIGH) {

     displayTimedEvents();

     while (1 == 1) {};

   }

   last\_interrupt\_time = interrupt\_time;

 }

}

void photoResistor() {

 static unsigned long last\_interrupt\_time = 0;

 unsigned long interrupt\_time = millis();

 if ((interrupt\_time - last\_interrupt\_time > 200) && ((millis() - hack) > 100)) {    // && ((millis() - hack) > 100))

   if (boxNeverOpen == 1) { // causes the alarm to never stop buzzing if you open it WITHOUT entering the correct password

     state = HIGH;

     boxNeverOpen = 0;

     timerInfoArray[eventNumber].action = "Alarm triggered";

     timerInfoArray[eventNumber].timeOfEvent = timerr();

     eventNumber ++;

   }

 }

 last\_interrupt\_time = interrupt\_time;

}

bool compareArrays () {

 for (int i = 0; i < 4; i ++) {

   if (password[i] != entered[i]) {

     return false;

   }

 }

 return true;

}

void currentState() {

 Serial.print("the button has been pushed this many times: " );

 Serial.println(pushButtonCounter);

 Serial.print("the password so far is: " );

 for (int i = 0; i < 4; i ++) {

   Serial.print(entered[i]);

 }

 Serial.println();

}

void wrongPassword() {

 digitalWrite (ledRed, HIGH);

 delay (500);

 digitalWrite (ledRed, LOW);

 delay (500);

 digitalWrite (ledRed, HIGH);

 delay (500);

 digitalWrite (ledRed, LOW);

 for (int i = 0; i < 4; i ++) {

   (entered[i] = 0);

 }

}

void displayTimedEvents() {

 for (int k = 0; (k < 20); k++) {

   if (timerInfoArray[k].action == "err") {

     Serial.print(k + 1);

     Serial.print("      ");

     Serial.println("0");

   }

   else {

     Serial.print(k + 1);

     Serial.print("      ");

     Serial.print(timerInfoArray[k].action);

     Serial.print("      ");

     Serial.println(timerInfoArray[k].timeOfEvent);

   }

 }

}