**DOCUMENTAÇÃO DO TRABALHO ARDUÍNO – João Victor Jacques Campos Lessa   
LIP – Manhã**Proposta Inicial:  
Após pesquisar sobre quais automações fazer no arduíno para o trabalho, me deparei com diversos projetos muito avançados que utilizavam diversos recursos de acesso mais restrito como por exemplo: estruturas robóticas e chassis de veículos. Portanto optei por fazer um sistema de alarme, contudo utilizando de mais recursos para melhor eficiência.  
  
Montagem:  
As estruturas necessárias para a montagem do sistema são:

. Arduino Uno

. LDR (Light Dependent Resistor)

. Resistor para o LDR

. Buzzer

. LEDs (vermelho para alerta)

. Botões (para inserir o código de segurança)

. Sensor de temperatura TMP36

. Display LCD (16x2)

. Protoboard e fios de conexão

. Resistores para LEDs

. Potenciômetro para o controle de contraste do LCD  
  
OBS: Tive que montar e remontar várias vezes por conta de erros meus ao conectar cabos em entradas erradas, utilizar a resistência errada e estruturar os cabos mais organizadamente.

Código:  
O código contém dois gatilhos para o disparo do alarme, que é a luminosidade e a variação brusca de temperatura. Além disso, possui um sistema de senha que, ao disparar o alarme, ele pode ser desativado com a sequência correta de botões. A temperatura é mostrada no LCD e quando ela atinge pontos críticos é mostrado um aviso junto ao alarme. No mesmo LCD e mostrado a luminosidade que disparou o alarme.  
  
código fonte:  
**#define LDR\_PIN A0**

**#define TMP36\_PIN A1**

**#define BUZZER\_PIN 8**

**#define LED\_PIN 9**

**#define BUTTON\_PIN\_1 2**

**#define BUTTON\_PIN\_2 3**

**#define BUTTON\_PIN\_3 4**

**#define BUTTON\_PIN\_4 5**

**#define LCD\_RS 10**

**#define LCD\_EN 11**

**#define LCD\_D4 12**

**#define LCD\_D5 13**

**#define LCD\_D6 A2**

**#define LCD\_D7 A3**

**const int correctCode[4] = {1, 2, 3, 4}; // Código de segurança correto**

**int enteredCode[4] = {0, 0, 0, 0}; // Código inserido**

**int codeIndex = 0; // Índice do código inserido**

**bool codeCorrect = false;**

**int threshold = 500; // Valor de luminosidade para detectar abertura**

**float criticalHighTemp = 40.0; // Temperatura crítica alta**

**float criticalLowTemp = 0.0; // Temperatura crítica baixa**

**void setup() {**

**pinMode(BUZZER\_PIN, OUTPUT);**

**pinMode(LED\_PIN, OUTPUT);**

**pinMode(BUTTON\_PIN\_1, INPUT\_PULLUP);**

**pinMode(BUTTON\_PIN\_2, INPUT\_PULLUP);**

**pinMode(BUTTON\_PIN\_3, INPUT\_PULLUP);**

**pinMode(BUTTON\_PIN\_4, INPUT\_PULLUP);**

**pinMode(LCD\_RS, OUTPUT);**

**pinMode(LCD\_EN, OUTPUT);**

**pinMode(LCD\_D4, OUTPUT);**

**pinMode(LCD\_D5, OUTPUT);**

**pinMode(LCD\_D6, OUTPUT);**

**pinMode(LCD\_D7, OUTPUT);**

**digitalWrite(BUZZER\_PIN, LOW);**

**digitalWrite(LED\_PIN, LOW);**

**lcdBeginInator();**

**lcdPrintInator("Sistema FIEMT");**

**Serial.begin(9600);**

**}**

**void loop() {**

**int ldrValue = analogRead(LDR\_PIN);**

**float temperature = readTemperature(TMP36\_PIN);**

**lcdSetCursor(0, 0);**

**lcdPrintInator("Temp: ");**

**lcdPrintInator(temperature);**

**lcdPrintInator(" C ");**

**bool alarmTriggered = false;**

**if (ldrValue > threshold && !codeCorrect) {**

**alarmTriggered = true;**

**lcdSetCursor(0, 1);**

**lcdPrintInator("Alerta! LDR: ");**

**lcdPrintInator(ldrValue);**

**lcdPrintInator(" ");**

**} else if (temperature > criticalHighTemp || temperature < criticalLowTemp) {**

**alarmTriggered = true;**

**lcdSetCursor(0, 1);**

**lcdPrintInator("Temp. Critica ");**

**} else {**

**lcdSetCursor(0, 1);**

**lcdPrintInator("Seguro ");**

**}**

**if (alarmTriggered) {**

**tone(BUZZER\_PIN, 500); // Emitir som contínuo com frequência de 500Hz**

**digitalWrite(LED\_PIN, HIGH);**

**} else {**

**noTone(BUZZER\_PIN);**

**digitalWrite(LED\_PIN, LOW);**

**}**

**if (digitalRead(BUTTON\_PIN\_1) == LOW) {**

**enteredCode[codeIndex++] = 1;**

**delay(200);**

**} else if (digitalRead(BUTTON\_PIN\_2) == LOW) {**

**enteredCode[codeIndex++] = 2;**

**delay(200);**

**} else if (digitalRead(BUTTON\_PIN\_3) == LOW) {**

**enteredCode[codeIndex++] = 3;**

**delay(200);**

**} else if (digitalRead(BUTTON\_PIN\_4) == LOW) {**

**enteredCode[codeIndex++] = 4;**

**delay(200);**

**}**

**if (codeIndex == 4) {**

**codeCorrect = checkCode();**

**codeIndex = 0; // Resetar índice para próxima tentativa**

**}**

**delay(100);**

**}**

**float readTemperature(int pin) {**

**int reading = analogRead(pin);**

**float voltage = reading \* 5.0 / 1024.0;**

**float temperatureC = (voltage - 0.5) \* 100;**

**return temperatureC;**

**}**

**bool checkCode() {**

**for (int i = 0; i < 4; i++) {**

**if (enteredCode[i] != correctCode[i]) {**

**// Código incorreto, resetar o código inserido**

**for (int j = 0; j < 4; j++) {**

**enteredCode[j] = 0;**

**}**

**return false;**

**}**

**}**

**// Código correto, resetar o código inserido**

**for (int j = 0; j < 4; j++) {**

**enteredCode[j] = 0;**

**}**

**return true;**

**}**

**void lcdSendNibble(byte data) {**

**digitalWrite(LCD\_D4, data & 0x01);**

**digitalWrite(LCD\_D5, data & 0x02);**

**digitalWrite(LCD\_D6, data & 0x04);**

**digitalWrite(LCD\_D7, data & 0x08);**

**digitalWrite(LCD\_EN, HIGH);**

**delayMicroseconds(1);**

**digitalWrite(LCD\_EN, LOW);**

**}**

**void lcdSendByte(byte data, byte mode) {**

**digitalWrite(LCD\_RS, mode);**

**lcdSendNibble(data >> 4);**

**lcdSendNibble(data);**

**delayMicroseconds(100);**

**}**

**void lcdBeginInator() {**

**delay(15);**

**lcdSendNibble(0x03);**

**delay(5);**

**lcdSendNibble(0x03);**

**delayMicroseconds(150);**

**lcdSendNibble(0x03);**

**lcdSendNibble(0x02);**

**lcdSendByte(0x28, LOW); // Function set: 4-bit, 2 line, 5x8 dots**

**lcdSendByte(0x08, LOW); // Display off**

**lcdSendByte(0x01, LOW); // Clear display**

**lcdSendByte(0x06, LOW); // Entry mode set**

**lcdSendByte(0x0C, LOW); // Display on, cursor off, blink off**

**}**

**void lcdSetCursor(byte col, byte row) {**

**byte rowOffsets[] = {0x00, 0x40, 0x14, 0x54};**

**lcdSendByte(0x80 | (col + rowOffsets[row]), LOW);**

**}**

**void lcdPrintInator(const char \*str) {**

**while (\*str) {**

**lcdSendByte(\*str++, HIGH);**

**}**

**}**

**void lcdPrintInator(float value) {**

**char buffer[10];**

**dtostrf(value, 6, 2, buffer);**

**lcdPrintInator(buffer);**

**}  
  
Conclusão:  
Os primeiros resultados começaram com dois leds estourando, por conta da resistência errada, e também com o buzzer disparando independentemente dos sensores de luz e temperatura. O erro estava na conexão dos cabos, no qual em nenhum dos dois sensores havia uma corrente passando pelo GND. Após resolvidos, o próximo problema foi o display do LCD que não mostrava os dados. Foi corrigido adicionando um potenciômetro ao LCD.**