Match the Stray function to the functions in BurglarAlarm.ino!

Cut and paste the text between the tildas~

|  |
| --- |
| **Function A:** |
| **~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~** |
| LogInfo("Try to invoke method %s", methodName); |
| char \*responseMessage = "\"Successfully invoke device method\""; |
| int result = 200; |
|  |
| if (strcmp(methodName, "stop") == 0){ |
| alarm\_is\_triggered = false; |
| } else { |
| LogInfo("No method %s found", methodName); |
| result = 404; |
| } |
|  |
| \*response\_size = strlen(responseMessage) + 1; |
| \*response = (unsigned char \*)strdup(responseMessage); |
|  |
| return result; |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| **Function B:** |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| int distance = FindSensorDistance(); |
| char message[128]; |
| if (distance < 70 && !alarm\_is\_triggered && alarm\_is\_enabled){ |
| alarm\_is\_triggered = true; |
| snprintf(message, 128, "{\"alarm\_status\":\"triggered\"}{\"status\":\"online\"}"); |
| SendMessageToAzure(message, has\_iot\_hub); |
| } else if (millis() - send\_interval\_ms > 10000) { |
|  |
| if (alarm\_is\_enabled){ |
| snprintf(message, 128, "{\"status\":\"online\"}"); |
| } else { |
| snprintf(message, 128, "{\"status\":\"offline\"}"); |
| } |
| SendMessageToAzure(message, has\_iot\_hub); |
| } |
|  |
| if (alarm\_is\_triggered){ |
| M5.Lcd.drawBitmap(0, 0, 320, 240, (uint16\_t \*) alarm\_triggered\_img); |
| PlayAlarmRing(); |
| } else { |
| M5.Lcd.drawBitmap(0, 0, 320, 240, (uint16\_t \*) alarm\_warning\_img); |
| TurnAlarmRingOff(); |
| } |
| delay(100); |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| **Function C:** |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| RunAlarm(); |
| Esp32MQTTClient\_Check();  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| **Function D:** |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| portENTER\_CRITICAL\_ISR(&mux); |
| alarm\_is\_triggered = false; |
| portEXIT\_CRITICAL\_ISR(&mux);  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| **Function E:** |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| char \*temp = (char \*)malloc(size + 1); |
| if (temp == NULL) return; |
| memcpy(temp, payLoad, size); |
| temp[size] = '\0'; |
|  |
| JsonObject& json\_obj = json\_buffer.parseObject(temp); |
| bool alarm\_status = json\_obj["Enable/Disable"]["value"]; |
| const char \*desired = json\_obj["desired"]["Enable/Disable"]["value"]; |
|  |
| if (desired != NULL) alarm\_status = json\_obj["desired"]["Enable/Disable"]["value"]; |
|  |
| if (alarm\_status){ |
| // Enable the alarm. |
| alarm\_is\_enabled = true; |
| } |
| else{ |
| // Disable the alarm. |
| alarm\_is\_enabled = false; |
| alarm\_is\_triggered = false; |
| } |
|  |
| free(temp); |
| json\_buffer.clear(); |
| send\_interval\_ms = millis();  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| **Function F:** |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
|  |
| M5.Lcd.setTextColor(WHITE); |
| M5.Lcd.setTextSize(4); |
| M5.Lcd.printf("Connecting to WiFi..."); |
| M5.update(); |
| Serial.println("Starting connecting WiFi."); |
| delay(10); |
| WiFi.begin(ssid, password); |
| while (WiFi.status() != WL\_CONNECTED) { |
| delay(500); |
| Serial.print("."); |
| Serial.println("WiFi connected"); |
| Serial.println("IP address: "); |
| Serial.println(WiFi.localIP()); |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| **Function G:** |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

|  |
| --- |
| digitalWrite(TRIGGER\_PIN, LOW); |
| delayMicroseconds(2); |
|  |
| // Sets the TRIGGER\_PIN on HIGH state for 10 micro seconds. |
| digitalWrite(TRIGGER\_PIN, HIGH); |
| delayMicroseconds(10); |
| digitalWrite(TRIGGER\_PIN, LOW); |
|  |
| // Reads the ECHO\_PIN and returns the sound wave travel time in microseconds. |
| int duration = pulseIn(ECHO\_PIN, HIGH); |
| return duration\*0.034/2; |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ |
| **Function H:** |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

|  |
| --- |
| M5.Speaker.tone(NOTE\_1); |
| M5.update(); |
| delay(200); |
|  |
| M5.Speaker.tone(NOTE\_2); |
| M5.update(); |
| delay(200); |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ | |
| **Function I:** | |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

|  |
| --- |
| if (has\_iot\_hub){ |
| Serial.println(message); |
| if (Esp32MQTTClient\_SendEvent(message)){ |
| Serial.println("Sending data succeed"); |
| } else { |
| Serial.println("Failure..."); |
| } |
| } |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ | |
| **Function J:** | |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

|  |
| --- |
| M5.Speaker.mute(); |
| M5.update(); |
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ | |
| **Function K:** | |

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

|  |
| --- |
| M5.begin(); |
| Wire.begin(); |
| Serial.begin(115200); |
| json\_buffer.clear(); |
|  |
| // Connect to WiFi. |
| ConnectToWiFi(); |
| randomSeed(analogRead(0)); |
|  |
| // Connect to Azure IoT Central and setup device callbacks. |
| Esp32MQTTClient\_Init( (const uint8\_t\*)connection\_string, true); |
| has\_iot\_hub = true; |
| Esp32MQTTClient\_SetSendConfirmationCallback(SendConfirmationCallback); |
| Esp32MQTTClient\_SetDeviceTwinCallback(DeviceTwinCallback); |
| Esp32MQTTClient\_SetDeviceMethodCallback( DeviceMethodCallback); |
|  |
| // Setup interrupts to be triggered on the button pins. |
| pinMode(interrupt\_pin\_button\_a, INPUT\_PULLUP); |
| attachInterrupt( digitalPinToInterrupt(interrupt\_pin\_button\_a), HandleInterruptStopAlarm, FALLING); |
| pinMode(interrupt\_pin\_button\_b, INPUT\_PULLUP); |
| attachInterrupt( digitalPinToInterrupt(interrupt\_pin\_button\_b), HandleInterruptStopAlarm, FALLING); |
| pinMode(interrupt\_pin\_button\_c, INPUT\_PULLUP); |
| attachInterrupt( digitalPinToInterrupt(interrupt\_pin\_button\_c), HandleInterruptStopAlarm, FALLING); |
|  |
| // Configure echo and trigger pins used by the ultrasonic sensor. |
| pinMode(TRIGGER\_PIN, OUTPUT); |
| pinMode(ECHO\_PIN, INPUT); |