

# CEN479-Internet of Things

Project Name: Catching Santa Claus

Group Members - Student Numbers:

Nahide Sena Sabırlı - 20091000020

Elif Naz Kadayıfçı - 20091000038

Rawan ElShenieky - 21091000188

## Introduction

Having an unexpected motion detected while sleeping can be quite suspicious and can be a dangerous sign that somebody came uninvited, whether a thief or Santa Claus. That's why we designed a system to lock Santa when he comes while we are asleep and alarm us and maybe the neighborhood. Having a simple, clear, and critical purpose, we present to you our system in the upcoming sections.

## How does it work?

The user enters their sleeping hours and their deactivation code in our webpage. At that time we assume that the door of the living room is open. When motion is detected during the sleeping hours, the servo (which we considered as our door) turns 90 degrees to be closed and at the same time the buzzer turns on to wake up the user. When the user wakes up there will be a notification on their phone that says "There seems to be a movement in your house. Was this you?". From this point onwards we can classify the user actions into cases for clarification:

### Case 1

If the user responds with "yes" then the system gets deactivated, opening the door (servo moves back to 0 degree).

### Case 2

If the user responds with "no", another message comes up asking the user whether they would like to deactivate the system. At that point the system fetches the data from `sleeping_hours.xml` repeatedly to check the change in "codematch" element value from "none" or "no" to "yes".

## Case 2A (following up on Case 2)

If the user enters the wrong deactivation code, that does not match the one stored in the sleeping\_hours.xml file which was entered earlier. The door stays closed, meaning the servo stays in 90 degrees as it was.

## Case 2B (following up on Case 2)

If the user enters the correct deactivation code, that matches the one stored in the sleeping\_hours.xml file which was entered earlier. The system goes deactivated, the door turns open, meaning the servo moves back to 0 degree. By deactivating the system, any motion detected later will not cause any sensor to work.

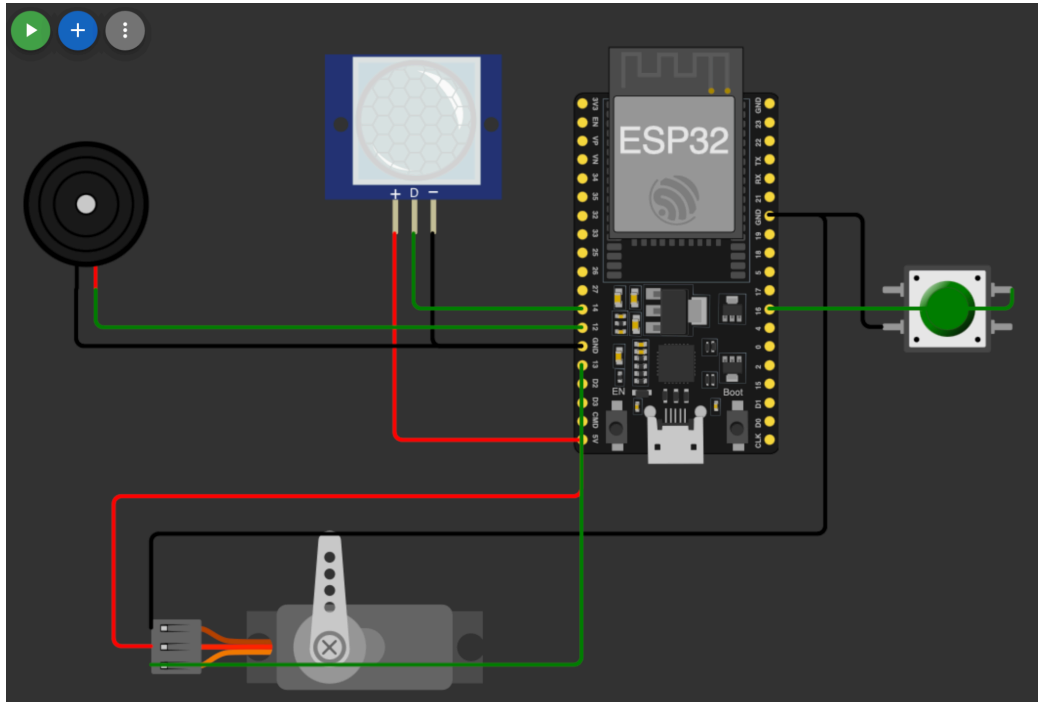
## Case 3

If the user wants to open the door temporarily without completely deactivating the door, this can be achieved using the button that is hypothetically placed outside the living room door. When the button gets a long press, the door turns open, meaning the servo moves back to 0 degree. However, if motion is detected again, since the system is still activated then whenever motion is detected, the servo (which we considered as our door) turns 90 degrees to be closed and at the same time the buzzer turns on again.

## Case 4

If there is motion detected outside of the sleeping hours set by the user, the system will ignore it and keeps refreshing on its own, by continuously fetching the data from sleeping\_hours.xml. This is to check if new sleeping hours are set which intersects with the current time which will cause the system to be active and start with Case 1.

## Parts Explained



### ESP32

In order to use web services and exchange data using HTTP with the xml files, we have found that ESP32 is more applicable to be used rather than arduino uno, thanks to its inbuilt compatibility with Wifi.

### Buzzer

It is essential to use a buzzer for alarming the user and possibly the neighbors - if it is loud enough - when the system is activated. We found it also a reasonable and practical way that matches our suggested scenario, in which the user will be sleeping and might not be alarmed with the normal mobile's notification sound.

### Push Button

It puts the system in idle for a short period of time until motion is detected once again. This is to provide the user with a secondary option in case they want to let, for example, the police inside to engage with whoever is inside, keeping them locked and preventing Santa Claus from running.

## Servo

In our simulation, we thought of the servo's rotations and movements equivalent to the door's in reality. This was to bring a more realistic perspective while inspecting the changes when running the simulation.

## PIR Motion Sensor

We thought of the PIR sensor used as what it is called to be "pet friendly" or "pet immune". Those are the terms called for the PIR sensors that do not put the movements of the pets into their accounts, focusing solely on the human movements.

## Notes & Comments

### Suggested scenario

While planning for our project we had a specific setup set for increasing the optimal outcome of our system. We assumed that the sleeping hours are set on a daily basis, meaning that the user can change their sleeping schedule. Also, it is designed for one user only, meaning that if person A set up their sleeping schedule and the motion got detected in A's house and at the same time another person B enters the system, they will directly be in Step 2 with person A's information saved.

### Link with XML project

This project is prepared to be used in the XML & Web Services course with code CEN467. The reports of each subject contain differences in terms of content as each one focuses more on its relevant course core.

## Alternative Scenarios

Our system can be implemented in different environments other than in home while sleeping. For example, it can also be implemented in companies, offices, malls, etc... The hours that will be set can be working hours or break times.

## Hosting Details

Based upon your suggestion of using infinityfree.com, we used it to host our webpage. You can check it out from the following links:

The main webpage: [http://santaclaus.lovestoblog.com/sleeping\\_hours.php](http://santaclaus.lovestoblog.com/sleeping_hours.php)

The sleeping\_hours.xml: [http://santaclaus.lovestoblog.com/sleeping\\_hours.xml](http://santaclaus.lovestoblog.com/sleeping_hours.xml)

The all\_logs.xml: [http://santaclaus.lovestoblog.com/all\\_logs.xml](http://santaclaus.lovestoblog.com/all_logs.xml)

The Wokwi project: <https://wokwi.com/projects/418091855523480577>

Note that the links are accessed only through the computer and it does not work on the phone.

## Possible Improvements

In terms of security, we can also add a keypad next to the button (in real life) which has an activation code that the user can enter to keep the system activated. Also, the volume of the buzzer's alarm can be increased so that it can be alerting enough. There can also be a camera set, to capture photos or show a live-streaming video inside the living room.

## Conclusion

To sum it all up, we have successfully created a virtual environment that simulated a realistic scenario that can be life-threatening in case it was not a false alarm. Combining our project with XML was a positive step towards showing our work implemented in the nearest actual application for our setup. Approaching our aim with planning and designing led us into satisfactory results that we hope are reasonable to you.

# Code Snippets

```
1  #include <ESP32Servo.h>
2  #include <WiFi.h>
3  #include <HTTPClient.h>
4
5  // WiFi Credentials
6  const char* ssid = "Wokwi-GUEST";
7  const char* password = "";
8
9  // Web Service URL
10 const char* serverURL = "http://santaclaus.lovestoblog.com/sleeping_hours.xml";
11
12 // Pin Definitions
13 const int pirPin = 14;    // PIR sensor OUT pin
14 const int buzzerPin = 12; // Buzzer pin
15 const int servoPin = 13;  // Servo signal pin
16 const int buttonPin = 16; // Push button pin
17
18 Servo doorServo; // Servo object
19
20 // State Variables
21 bool motionDetected = false;
22 bool doorClosed = false;
23 bool buttonPressed = false;
24 String codematch = "none"; // Initialize codematch value
25
26 // Hardcoded Current Time (24-hour format, e.g., 23 for 11:00 PM)
27 int currentHour = 23; // Change this to simulate different times
28
29 // Sleeping Hours Range (start and end)
30 int sleepStart = -1;
31 int sleepEnd = -1;
32
33 // Timer for periodic XML fetching
34 unsigned long lastFetchTime = 0;
35 const unsigned long fetchInterval = 2000; // Check every 2 seconds
36
37 void setup() {
38     Serial.begin(115200);
39 }
```

```

40 // Initialize Pins
41 pinMode(pirPin, INPUT);
42 pinMode(buzzerPin, OUTPUT);
43 pinMode(buttonPin, INPUT_PULLUP);
44
45 // Attach Servo
46 doorServo.attach(servoPin);
47
48 // Initial Positions
49 doorServo.write(0); // Door open
50 digitalWrite(buzzerPin, LOW);
51
52 // Connect to WiFi
53 WiFi.begin(ssid, password);
54 while (WiFi.status() != WL_CONNECTED) {
55     delay(500);
56     Serial.print(".");
57 }
58 Serial.println("\nWiFi connected.");
59
60 // Retrieve sleeping hours from the XML file
61 fetchSleepingHours();
62 }
63
64 void loop() {
65     // Periodic XML fetching when door is closed
66     unsigned long currentTime = millis();
67     if (doorClosed && (currentTime - lastFetchTime >= fetchInterval)) {
68         fetchSleepingHours();
69         lastFetchTime = currentTime;
70
71         // Check if codematch changed to "yes" while door was closed
72         if (codematch == "yes") {
73             Serial.println("Code match changed to YES - Opening door");
74             doorServo.write(0); // Open the door
75             doorClosed = false;
76             return; // Skip the rest of the loop
77         }
78     }

```

```

79
80 // Check motion detection
81 motionDetected = digitalRead(pirPin);
82
83 // Check if current hour falls within the sleeping hours range
84 bool withinSleepingHours = isWithinSleepingHours(currentHour);
85
86 // Check button press and update codematch
87 if (digitalRead(buttonPin) == LOW) { // Button pressed
88     Serial.println("Button Pressed! Checking code status...");
89     buttonPressed = true;
90     fetchSleepingHours(); // Update codematch value
91
92     // If codematch is "yes" and door is closed, open it
93     if (codematch == "yes" && doorClosed) {
94         Serial.println("Code match is YES - Opening door and disabling motion detection");
95         doorServo.write(0); // Open the door
96         doorClosed = false;
97         delay(500); // Debounce delay
98         return; // Skip the rest of the loop
99     } else {
100         Serial.println("Button Pressed! Opening the door...");
101         doorServo.write(0); // Open the door
102         doorClosed = false;
103         delay(500); // Debounce delay
104     }
105 }
106
107 // If codematch is "yes", keep the door open regardless of motion
108 if (codematch == "yes") {
109     if (doorClosed) {
110         Serial.println("Code match is YES - Opening door");
111         doorServo.write(0); // Open the door
112         doorClosed = false;
113     }
114     return; // Skip the rest of the loop
115 }
116

```

```

117 // Normal motion detection behavior when codematch is not "yes"
118 if (motionDetected && withinSleepingHours && !doorClosed) {
119     Serial.println("Motion Detected within sleeping hours! Closing the door...");
120     // Activate Buzzer
121     digitalWrite(buzzerPin, HIGH);
122     delay(500);
123     digitalWrite(buzzerPin, LOW);
124
125     // Close the Door
126     doorServo.write(90); // Move Servo to 90° (door closed)
127     doorClosed = true;
128     lastFetchTime = millis(); // Initialize the fetch timer when door closes
129 } else if (motionDetected && !withinSleepingHours) {
130     Serial.println("Motion Detected but outside sleeping hours. No action taken.");
131 }
132
133 delay(1000); // Delay for stability
134 }
135
136 void fetchSleepingHours() {
137     if (WiFi.status() == WL_CONNECTED) {
138         HTTPClient http;
139         http.begin(serverURL);
140         int responseCode = http.GET();
141
142         if (responseCode > 0) {
143             String payload = http.getString();
144             Serial.println("XML Data Received:");
145             Serial.println(payload);
146
147             // Extract sleeping hours from XML
148             sleepStart = parseIntFromXML(payload, "<start>", "</start>");
149             sleepEnd = parseIntFromXML(payload, "<end>", "</end>");
150             codematch = parseStringFromXML(payload, "<codematch>", "</codematch>");
151
152             Serial.print("Sleeping Hours: ");
153             Serial.print(sleepStart);
154             Serial.print(":00 to ");
155             Serial.print(sleepEnd);
156             Serial.println(":00");
157

```

```

157         Serial.print("Codematch Value: ");
158         Serial.println(codematch);
159     } else {
160         Serial.println("Failed to fetch XML file.");
161     }
162     http.end();
163 }
164 }
165
166 bool isWithinSleepingHours(int currentHour) {
167     if (sleepStart == -1 || sleepEnd == -1) {
168         // Sleeping hours not yet set
169         return false;
170     }
171
172     if (sleepStart < sleepEnd) {
173         // Regular range (e.g., 22:00 to 6:00)
174         return currentHour >= sleepStart && currentHour < sleepEnd;
175     } else {
176         // Overnight range (e.g., 22:00 to 6:00)
177         return currentHour >= sleepStart || currentHour < sleepEnd;
178     }
179 }
180
181 int parseIntFromXML(String payload, String startTag, String endTag) {
182     int startIndex = payload.indexOf(startTag);
183     int endIndex = payload.indexOf(endTag);
184
185     if (startIndex != -1 && endIndex != -1) {
186         startIndex += startTag.length();
187         String value = payload.substring(startIndex, endIndex);
188         return value.toInt();
189     }
190     return -1; // Return -1 if tag not found
191 }

```

```

193 String parseStringFromXML(String payload, String startTag, String endTag) {
194     int startIndex = payload.indexOf(startTag);
195     int endIndex = payload.indexOf(endTag);
196
197     if (startIndex != -1 && endIndex != -1) {
198         startIndex += startTag.length();
199         return payload.substring(startIndex, endIndex);
200     }
201     return "none"; // Return "none" if tag not found
202 }

```