**EXPERIMENT NO. 5**

**Aim:**

To design and implement an anti-theft security system for a locker using capacitive touch sensors that detect unauthorized access attempts and notify the owner via Bluetooth Classic communication using the ESP32.

**Apparatus:**

1. ESP32 Development Board
2. LED (Light Emitting Diode)
3. Potentiometer (10kΩ or similar)
4. Breadboard
5. Jumper Wires
6. Resistor (220Ω to limit current to the LED)
7. USB cable (for power and programming)
8. Computer with Arduino IDE (or other programming environment for ESP32)

**Theory:**

An anti-theft security system aims to detect unauthorized access attempts and notify the rightful owner. This project uses capacitive touch sensors to sense the touch (or proximity) of someone trying to access the locker. If the sensor detects a touch in an unauthorized manner (such as repeated failed attempts or suspicious behavior), the ESP32 triggers an alert system.

The ESP32 will be programmed to:

1. Monitor the capacitive touch sensor for touch events.
2. Notify the owner of any unauthorized attempts by sending a Bluetooth message to a paired mobile device.
3. Optionally activate local alarms such as buzzers or LEDs to warn the intruder.

Bluetooth Classic communication allows a low-power, wireless method of sending alerts directly to a nearby device, such as a smartphone, without the need for an internet connection.

**Procedure:**

1. **Circuit Setup:**
   * Connect the capacitive touch sensor to an available GPIO pin of the ESP32 (e.g., GPIO 33).
   * Optionally, connect a buzzer or LED to another GPIO pin to act as a local alert mechanism (e.g., GPIO 25).
   * If using a relay for a locking mechanism, connect it to a digital output pin of the ESP32.
2. **Bluetooth Setup:**
   * ESP32 has built-in Bluetooth Classic capabilities, which we will use to communicate with the owner’s mobile device.
   * Prepare the ESP32 to initiate and manage Bluetooth communication.
3. **Programming:**
   * In Arduino IDE (or any other compatible development environment), write a program that performs the following:
     + Initialize the capacitive touch sensor input.
     + Monitor the touch sensor’s state.
     + If an unauthorized touch is detected, send a notification to the owner via Bluetooth.
     + Trigger the local alert (buzzer or LED) when unauthorized access is attempted.
4. **Upload and Test:**
   * Upload the program to the ESP32.
   * Pair the ESP32 with a mobile phone or other Bluetooth device. Ensure Bluetooth on the phone is turned on.
   * Open a Bluetooth terminal app on the mobile device (e.g., “Bluetooth Terminal” from the app store) to receive the alerts from the ESP32.
5. **Testing:**
   * Touch the capacitive sensor to simulate unauthorized access to the locker.
   * Observe that an alert is sent via Bluetooth to the paired mobile device.
   * Check that the local alert (buzzer or LED) is activated during the unauthorized access event.

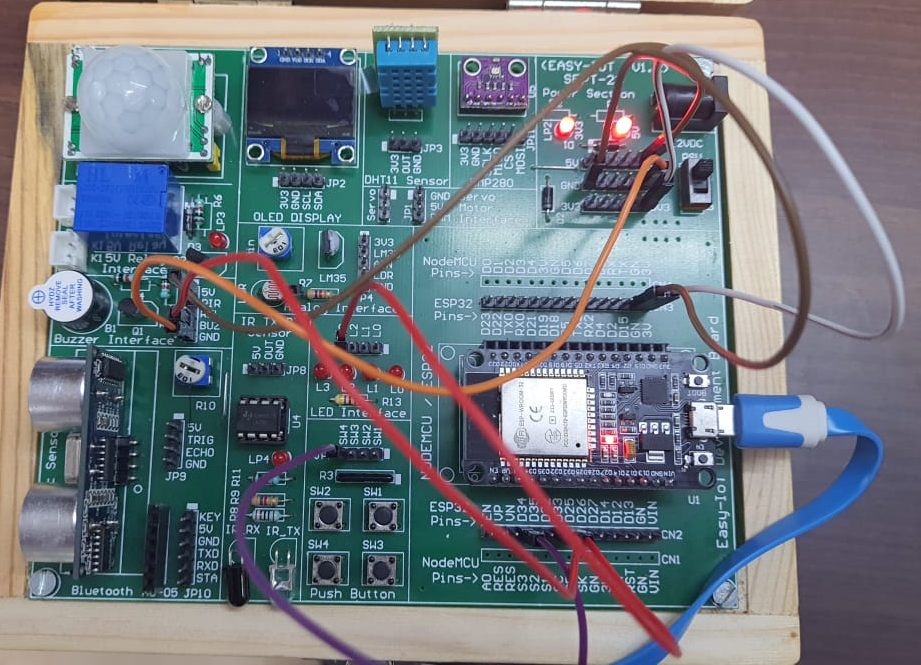
**Observation:**

* When the capacitive touch sensor detects an unauthorized touch, an alert message is sent to the owner's Bluetooth-enabled device, and a local alarm (buzzer or LED) is triggered.
* The sensitivity of the capacitive touch sensor can be adjusted by varying the threshold value in the code.

**Conclusion:**

This project successfully demonstrated how capacitive touch sensors can be used to detect unauthorized access attempts in a locker security system. The ESP32’s Bluetooth Classic module effectively communicated alerts to the owner’s mobile device, ensuring immediate notification of any potential security breaches. The system can be expanded with additional features such as integrating a relay-controlled lock or adding multiple sensors for enhanced security.

**Result:**



#include <BluetoothSerial.h> // For Bluetooth communication

BluetoothSerial SerialBT;

const int correctCode = 975301;

int enteredCode = 0 ;

unsigned long startTime;

bool alarmTriggered = false;

int touchPin = 12;

int buzzerPin = 26; // Pin for the buzzer

int relayPin = 23; // Pin for the locker control

int stopButtonPin = 35; // Pin for the button to stop the alarm

void setup() {

Serial.begin(96200); // Start the serial communication

SerialBT.begin("LockerSystem");

pinMode(buzzerPin, OUTPUT);

pinMode(relayPin, OUTPUT);

pinMode(stopButtonPin, INPUT\_PULLUP);

digitalWrite(buzzerPin, LOW);

digitalWrite(relayPin, LOW); // Locker initially locked

}

void loop() {

// Start code entry

if (touchRead(touchPin)< 20){

if (enteredCode == 0) {

startTime = millis();

SerialBT.println("Enter 6-digit security code: ");

}

// Check if there is input from the Serial monitor

if (Serial.available()) {

enteredCode = Serial.read();

}

}

// If the entered code is 6 digits or time limit is exceeded

if (enteredCode != 0 || millis() - startTime > 30000) {

if (enteredCode == correctCode) {

openLocker();

} else {

triggerAlarm();

}

}

// Stop alarm if owner presses the button

if (alarmTriggered && digitalRead(stopButtonPin) == LOW) {

stopAlarm();

}

}

void openLocker() {

digitalWrite(relayPin, HIGH); // Unlock locker

delay(5000); // Locker remains open for 5 seconds

digitalWrite(relayPin, LOW); // Relock locker

enteredCode = 0; // Reset code entry

}

void triggerAlarm() {

digitalWrite(buzzerPin, HIGH); // Sound the alarm

alarmTriggered = true;

sendAlert(); // Notify owner/security via Bluetooth/Wi-Fi

}

void stopAlarm() {

digitalWrite(buzzerPin, LOW); // Turn off the alarm

alarmTriggered = false;

enteredCode = 0; // Reset code entry

}

void sendAlert() {

SerialBT.println("Intruder alert! Incorrect code or timeout.");

}