Talha Hamza
Student ID: 5876131

Lab Report EE 1301

Description:

My mini project achieves the control of a siren using a button. When the button is pressed, the tone changing siren goes off. When the button is pressed a second time, the siren stops immediately.

Process:

My project did not work the first time I connected it.

I made a couple of mistakes. I had an additional wire connection that was causing a short circuit. I also had to rewrite a part of my program as pressing the button was not effectively switching the siren on and off.

For the connection issue, I scrutinized my circuit until I located the problem. For the programming issue, I consulted ChatGPT which informed me that time delays in my program were causing all operations, including registering of button presses, to halt. Hence, instead of using delays, I used measurements of the current time to change the siren's tone.

Code:

```
// Include Particle Device OS APIs
#include "Particle.h"

// Let Device OS manage the connection to the Particle Cloud
SYSTEM_MODE(MANUAL);

// Run the application and system concurrently in separate threads
SYSTEM_THREAD(ENABLED);

// Show system, cloud connectivity, and application logs over USB
// View logs with CLI using 'particle serial monitor --follow'
SerialLogHandler logHandler(LOG_LEVEL_INFO);

// Setting variables for button and speaker pins
int ButtonPIN = D2;
int speakerPin = D0;
```

Talha Hamza
Student ID: 5876131

```
// setting tracking variables for button state and siren state
bool ButtonNow = false;
bool ButtonLast = false;
int ButtonCount = 0;
bool buttonState = false;
bool sirenPlaying = false;

// setting variables for siren timing to help with tone changes
unsigned long sirenStartTime = 0;
unsigned long sirenDuration = 0;
unsigned long sirenToneChangeTime = 0;
bool isHighTone = true;

/*
This function controls the siren sound.
It takes a boolean parameter, play, which determines whether the siren should be
played or stopped.
Inside the function: currentTime is used to track the current time using
millis().
If play is true and the siren is not currently playing, it starts the siren with
alternating high and low tones, changing every 500 ms.
If the siren is already playing, it checks the elapsed time and changes the tone
every 500 ms.
If play is false, it immediately stops the siren if it's currently playing.
*/
void siren(bool play) {
    unsigned long currentTime = millis();

    if (play) {
        if (!sirenPlaying) {
            // Start playing the siren
            tone(speakerPin, isHighTone ? 880 : 440);
            isHighTone = !isHighTone;
            sirenToneChangeTime = currentTime;
            sirenPlaying = true;
        } else if (currentTime - sirenToneChangeTime >= 500) {
            tone(speakerPin, isHighTone ? 880 : 440);
            isHighTone = !isHighTone;
            sirenToneChangeTime = currentTime;
        }
    } else if (sirenPlaying) {
        noTone(speakerPin); // Stop the siren immediately if button state is off
        sirenPlaying = false;
    }
}
```

Talha Hamza
Student ID: 5876131

```
// in the setup function, I set the button pin as an input and set the serial
baud rate to 9600
// This is to ensure that I can see the button count and button state in the
serial monitor

void setup() {
    pinMode(ButtonPIN, INPUT_PULLDOWN);
    Serial.begin(9600);
}

/*
The loop function continuously checks the state of the button using digitalRead.
If the button transitions from low to high, it registers the button press,
increments ButtonCount, toggles buttonState, and logs the button state ("Button
On" or "Button Off").
It also updates ButtonLast to track the previous button state.
Finally, it calls the siren function with the current buttonState, controlling
whether the siren plays or stops.
*/
void loop() {
    ButtonNow = digitalRead(ButtonPIN);
    if (ButtonNow == HIGH && ButtonLast == LOW) {
        // Button was pressed
        ButtonCount++;
        buttonState = !buttonState; // Toggle button state
        Serial.print("Button Count = ");
        Serial.print(ButtonCount);
        Serial.print(", Button State = ");
        if (buttonState) {
            Serial.println("Button On");
        } else {
            Serial.println("Button Off");
        }

        ButtonLast = HIGH;
    } else if (ButtonNow == LOW) {
        ButtonLast = LOW;
    }
    siren(buttonState);
}
```