Automated Candle Safety System

Veeksha Pattikonda | Maya Oulhadj

Abstract

Our project is an automated candle safety system designed to reduce the risk of house fires caused by unattended candles. The system starts a 4-hour timer when a candle is lit, based on the recommended burn time for most candles. When the timer ends, visual and auditory alarms remind the homeowner to put out the candle. If no action is taken within 5 minutes, the system assumes the homeowner is away and extinguishes the candle automatically to prevent a potential fire. The system is compact, portable, and easy to use, effectively combining basic fire safety practices with technology to help people use candles safely.

Link to Project Demo Video with Narration

https://www.kapwing.com/videos/673abbbe1658825ecead165d

Project Description Pages

PROJECT OBJECTIVES

#	Goals	Goal Met?
1	Implement a digital fire sensor module to detect when a candle is lit.	Yes
2	Integrate a countdown timer on a four-digit LED display screen to track candle burn time.	No
3	Develop an alarm system with a buzzer and flashing LED to alert the homeowner when the candle's burn time expires.	Yes.
4	Include a squid button that allows the homeowner to manually deactivate the alarm.	Yes.
5	Successfully implement a fan that automatically extinguishes the candle if alarm is not responded to.	Yes.
6	Ensure the system is compact and portable.	Yes.

PROJECT APPROCH

We implemented the project by integrating several components from the Raspberry Pi kit to create a functional automated candle safety system. The system begins by waiting for a flame. We attached the fire sensor module to the top of our device to detect the flame from the candle below it. Once it detects that the candle is lit, a 4-hour countdown timer appears on the OLED graphical display screen. For demonstration purposes, however, we used 1-minute or even 30-second countdowns to simulate the 4-hour timer, as the actual time is too long for testing. When the timer reaches zero, a 5-minute alarm activates, with a flashing LED and buzzer synchronized to flash and sound simultaneously. (For the demonstration, we used a 1-minute countdown to simulate the 5 minutes.) The reason for this was to implement both a visual and auditory alarm, making the system more user-friendly. If the alarm is turned off by pressing the squid button before the 5 minutes are up, the whole system turns off, indicating that the homeowner has blown out the candle themselves. If it's not manually turned off within the 5 minutes, the system assumes nobody is home, and the Raspberry Pi cooling fan (located at the top of the device next to the heat detector) is automatically turned on for 20 seconds to blow out the flame. An aspect

we changed about the fan was that initially, we planned to have the fire sensor module check while the fan is running to determine whether the flame had been blown out. This would have been how the fan knew to stop. However, after testing, we realized that the device would still be quite hot even after the flame was blown out, so the fan would continue running for an extended period of time. To solve this, we conducted several tests to determine the optimal time frame for the fan to successfully extinguish the candle. After 20 tests, we found that the flame was blown out within 20 seconds, so we set the fan to run for this duration.

The recipes used in our device included 15.4 – Using an OLED Graphical Display, and we modified the ch_15_oled_clock.py code to control the countdown timer shown on the screen. This was a change from our original plan of using 15.1 – Using a Four-Digit LED Display and the ch_15_7seg.py code to implement the countdown. We weren't able to get a 4-digit LED display in time for the project, so we decided to use the OLED display, though the downside was that it's smaller and harder to read between steps in the process. We adapted the ch_15_oled_clock.py code to control the font size, text positioning, and display. We also used 16.7 – Making a Buzzing Sound and 11.1 – Connecting an LED to create the alarm for our project. We combined the ch_11_led_blink.py and ch_16_buzzer.py codes to sync the buzzer sound with the flashing LED. We turned both on simultaneously, then used sleep (0.5) between cycles to keep the alarm active for the required time. Additionally, we incorporated 10.11 – Using a Raspberry Squid Button and adapted the ch_10_button_test.py code to turn off the alarm by pressing the squid button. We integrated the `if button.is_pressed:` line of code into the alarm activation section and the `if not button.is_pressed:` line into the fan activation section, which worked well for us. We also used Lab 5, Question 4, which explained how to use the digital fire sensor module. We made only minor modifications to the code from the lab, taking advantage of the flame detection callback and event detection on the flame sensor pin. Initially, we planned to follow a YouTube tutorial to control the fan, but we later realized it only turned the fan on when the Raspberry Pi was overheating. However, since this was not aligned with our purpose, we switched to 12.4 – Controlling the Speed of a DC Motor and the ch_12_gui_slider.py code, which helped us understand how to control the fan more effectively.

We faced several challenges during the process but were able to overcome them. One major challenge was implementing the Raspberry Pi cooling fan. Since there were no resources or recipes that matched our needs, this took the most time. Initially, we tried using the L293D chip and a 5V 1A power supply to control the fan, but the Raspberry Pi overheated, so we had to find a new approach. After many attempts, we used the circuit from 12.4 – Controlling the Speed of a DC Motor to create a setup using a 2N3904 transistor and diode, replacing the DC motor with the cooling fan. Another challenge was keeping track of all the GPIO pins for the various components. We had to adjust the code repeatedly to ensure that each element was assigned the correct pin and that multiple components weren't connected to the same pin. To stay organized, we used a notebook to track the connections and modifications. Finally, we encountered challenges with 3D printing, as it was our first time using a 3D printer. One of our prints failed

just days before the project was due. To resolve this, we printed the device in parts, and once all the components were assembled, we used electrical tape to hold everything together. This method worked out even better because it allowed us to easily disassemble and reassemble the device, maintaining our goal of making it portable and compact.

CONCLUSION

In conclusion, creating the candle safety system was an educational experience. Combining our knowledge of hardware and software systems and integrating this to our raspberry pi to create a working model was a rewarding opportunity. By successfully designing and implementing a functional system that detects a candle flame, begin a countdown with an alarm system and extinguishes it by turning on a cooling fan. Even though we achieved this much we also encountered numerous challenges such as 3D printing, setting up the breadboard and calibrating the sensor. However, through a trial-and-error process we carefully troubleshooted and adapted our needs to excel in this presentation.

This project definitely strengthened our understanding of different types of systems and gave us experience to valuable skills such as improving our problem-solving and teamwork which would help us in future projects within this world of technology.