FLAMES – Fire Localization and Monitoring for Emergency Situations

Neil Singh, Lauren Caccamise, Ronex Faustin

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

Abstract — FLAMES (Fire Localization and Monitoring for Emergency Situations) is a compact, optoelectronic system designed to detect and classify fire events in enclosed, dark environments. By combining visible-light optics, broadband photodiodes, and embedded processing using an FPGA, microcontroller, and Raspberry Pi, FLAMES triggers visual and audio alerts. Housed in a 3D printed enclosure, it differentiates flames from other stochastic light sources such as sparks. Detection in FLAMES is achieved through a three-stage real-time analysis of light intensity, flicker frequency, color characteristics, and temporal patterns, using custom hardware and signal processing algorithms. The system scans its environment using a rotating lens assembly and halts upon detection. FLAMES offers an affordable, scalable solution for fire monitoring in sensitive indoor spaces. Conventional smoke detectors are ineffective in spark-prone, enclosed environments such as CNC machining enclosures, where frequent sparking can lead to false alarms or missed detections. These environments are ideal for the FLAMES system, which is specifically designed to differentiate between flames and non-hazardous light sources like sparks.

I. Introduction

FLAMES is a multidisciplinary engineering project designed to address the need for reliable, low latency fire detection in confined indoor environments. At the core of FLAMES is a Xilinx Artix-7 FPGA on the BASYS3 development board, responsible for flagging real-time signal processing outputs from the ESP32-WROOM-32D and Raspberry Pi 5. These include intensity-based detection using adaptive thresholding for ambient lighting from our TSL2591 light sensor, temporal flicker analysis from our photodiode, and YCbCr color feature extraction from the Raspberry Pi 5 camera with a custom lens system. Together, these computations detect flame-specific characteristics in the visible spectrum. The system is mounted on a stepper motor system that enables 180 degrees scanning of the surrounding environment. Supporting components include the ESP32 microcontroller for peripheral coordination for our stepper motor, LUX-based thresholding via the TSL2591 light sensor, and system-wide alert management that turns on a LED and sounds a passive buzzer. This project demonstrates the feasibility of using off-the-shelf components to create an intelligent, responsive fire detection system tailored for sensitive environments such as machining rooms and/or small laboratories.

The optical system includes a multi-element lens configuration. At the forefront, a plano convex lens collects light from the source of fire or spark and focuses it onto a silicon photodiode. A secondary lux sensor positioned behind a fisheye lens assembly is implemented to trigger a motor stop when a specified threshold is crossed. A custom imaging system (camera) with a field of view of approximately 66 degrees has been designed to add multiple layers of detection redundancy to increase the detection confidence

II. System Components

To gather a better understanding of the entirety of the system, the individual components will be presented.

A. Lux Sensor & Supporting Optics

The TSL2591 lux sensor has been selected for its exceptional ability to measure luminous flux, which quantifies the intensity of light falling on a specific area. The TSL291 lux sensor from Adafruit will be utilized to collect luminous flux (lux) data from the surrounding environment. The sensor is paired with a fish-eye lens assembly to effectively direct light onto the dual infrared and visible wavelength photodiode located on the TSL2591 development board. A 5 mm 3D Printed aperture has been added to the front surface of the fisheye lens. The system initiates by scanning its surrounding environment, collecting baseline light flux information through the fisheye aperture-limited lens. When the optical emission falls within the same field of view, the motor halts and conducts analysis employing three methods of detection confirmation. The introduction of the 5mm aperture limits the FOV to approximately 18 degrees allowing the system to achieve better precision halting accuracy. As stated earlier, this lux sensor is crucial in the initial stages of determination of a fire or sparking source. The specified field of view has been calculated by the following formula below. The Sensor plane (photodiode) is approximately 1mm^2 and the effective focal length of the wide field of view lens is approximately 3 mm.

$$FOV = 2 \cdot tan^{-1}\left(\frac{Sensor\ Dimension}{2 \cdot EFL}\right). \quad (1)$$

B. *Photodiode & Supporting focusing Optics*

The FLAMES unit employs the FDS1010 photodiode from Thorlabs, which boasts an active area of 100 mm². The diode's material is silicon, which in testing has proven to be the optimal choice for detecting wavelengths spanning from 400 nm to 1100 nm. For our project, we are primarily focused on the visible wavelength components of fire and sparks commonly found in machining environments. Silicon's responsivity at our operating wavelength, coupled with the diode's extensive active area, enables us to establish an ideal imaging plane for the incident light or spark to be imaged onto. A 100 mm focal length plano-convex lens is positioned approximately 40 mm away from our diode.

During testing, we have observed that when placed at the nominal focal length (100 mm), a smaller spot size is produced, and the spot position on the image plane becomes increasingly dependent on the distance of the source. This is done to intentionally increase the spot size from the incident light on the photodiode while maintaining the ability to vary the distance. By observation, we can clearly discern that fire flickers at a lower frequency compared to sparks generated by a grinding or abrasive cutting process. The rate at which the optical source flickers and, consequently, images onto the photodiode plane is utilized as the primary detection method for determining the type of optical source. The photodiode will generate a photocurrent on the order of microamperes. The output photocurrent is then inputted into the transimpedance circuit and analog-to-digital converter, where a digital voltage is generated. The figure below demonstrates how the light will be focused onto the image plane of the diode.

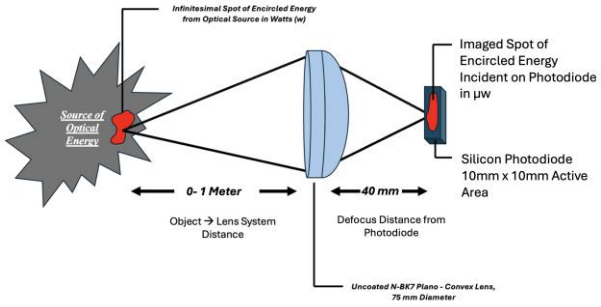

Fig. 1 . Figure 1 illustrates how samples of the optical source will be imaged onto the diode plane via the photodiode.

The relative distances between source and lens, lens and diode have been highlighted.

Defocusing the lens introduces larger spot size on detector plane enabling increased illumination on the diode plane. The power incident on the photodiode can be calculated using the equation below. The power of the optical source being used to demonstrate the systems capabilities has been measured to be 13 mW of optical power.

$$p_{detector} = \frac{P_{Source}}{4\pi r^2} \quad (2)$$

The photocurrent generated from the photodiode is calculated using the following equation.

$$I_{Photocurrent} = P_{Detector} \cdot \mathcal{R} \quad (3)$$

C. *Transimpedance Amplifier Circuit*

The transimpedance amplifier (TIA) in our design converts the photocurrent from the FDS1010 photodiode into a measurable voltage signal for the ESP32. The FDS1010, operating in reverse bias, provides sufficient signal for our TIA's current-to-voltage conversion. The OPA380 precision op-amp, selected for its transimpedance amplifier capabilities, operates in reverse bias with the FDS1010. The OPA380 offers high-speed operation, including a 90 MHz Gain Bandwidth Product (GBW) and exceptional precision [1]. A feedback resistor (R_F) determines the TIA's gain, which dictates the output voltage.

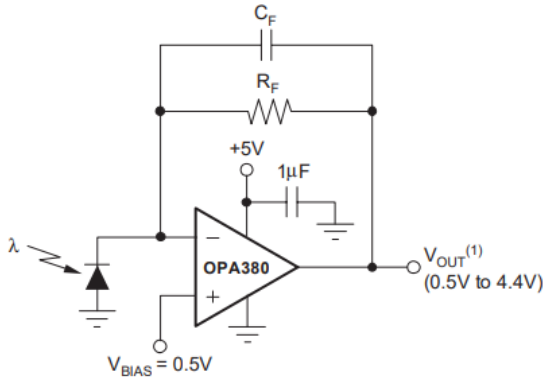

Fig. 2. OPA 380 Transimpedance Amplifier Configuration. [1].

Light incident on the diode will generate a photocurrent dependent on the wavelength of incident light. Photocurrent is then converted into voltage via the transimpedance amplification circuit. In typical transimpedance amplifier designs, the gain is often set in the megaOhm (MΩ) range to convert very small

photocurrents into measurable voltages. However, during testing with an optical power of 13 mW at 1 meter, we observed that the output voltage from our amplifier was significantly higher than expected, enough to saturate the op-amp. To address this, we reduced the transimpedance gain by selecting a 5.6 kΩ feedback resistor, which brought the output voltage within a usable range. Since the OPA380 op-amp supports a supply voltage between 2.7 V and 5.5 V [1] , we chose to power it at 3.3 V. This ensures that the amplifier's output voltage remains within the ESP32's input limits, maintaining compatibility.

The other key elements to a transimpedance design include the total input capacitance (C_{TOT}), which consists of the photodiode capacitance (C_{DIODE}) (for the FDS1010 it is 375 pF) [2] plus the parasitic common-mode (3pF) and the differential-mode input capacitance (1.1pF) for the OPA380, the feedback capacitor (C_F) which can be set to control the frequency response. To achieve a maximally flat, 2nd order, Butterworth frequency response, we used the following equation:

$$\frac{1}{2\pi R_F C_F} = \sqrt{\frac{GBW}{4\pi R_F C_{TOT}}} \tag{4}$$

The TIA bandwidth is calculated using the following equation:

$$f_{-3dB} = \sqrt{\frac{GBW}{2\pi R_F C_{TOT}}}\ \text{Hz} \tag{5}$$

By substituting our known values, we were able to calculate the approximated feedback capacitance which equals 0.1621pF and a transimpedance amplifier bandwidth of 2.72 MHz. By setting the optimal of C_F such that the TIA has 2nd order Butterworth response and performs with a flat frequency response, crucial for accurate signal amplification in high-speed optical detection applications.

D. Custom Imaging System (Camera)

The camera system utilized in the FLAMES unit has been customized with a 66-degree field of view in mind. This field of view was chosen to accommodate larger fires and add an additional layer of redundancy to increase detection accuracy. The field of view has been calculated using the following formula.

$$EFL = \frac{Diagonal\ of\ Sensor}{2\ \tan\left(\frac{FOV}{2}\right)} \tag{6}$$

Once the field of view has been calculated, the effective focal length for the entirety of the imaging system shall be calculated using equation 6.

$$HFOV = 2\ \tan^{-1}\left(\frac{Scene\ Height}{2{*}Obj\ Distance}\right) \tag{7}$$

The diagonal of the sensor measures 6.9 mm. The specified FOV at this stage of the calculation is maintained at 66 degrees. The effective focal length of the system is calculated to be 4.85 mm. The imaging system utilizes an aspheric and plano convex lens coupled with a 400 µm pinhole which effectively increases the depth of field. In testing alongside the Sony global shutter sensor, this combination of lenses proved to capture and identify fires up to 3.7 meters away when tested. The imaging system can detect fires as small as the flame produced by a common household lighter, where the flame measures approximately 2 x 1 inches. It is crucial to note that the image recognition confidence and size of the flame have a proportional relationship. As the size of the flame increases, the distance at which the imaging system can detect the threat also increases. A figure of the imaging system is shown below.

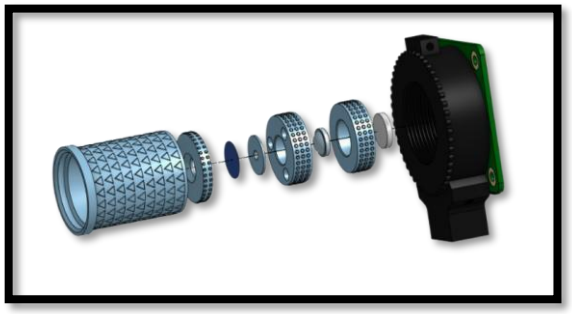

Fig. 3. Exploded View of Custom Imaging System

Figure 3 showcases the exploded view of the imaging system. All lenses have been packaged within a 3D printed housing which slots into the C-mount groove from the Raspberry Pi 5 camera.

E. Microcontroller

The microcontroller used in the FLAMES system is the ESP32-WROOM-32D, selected for its robust processing capabilities and built-in FreeRTOS support for real-time task management. FreeRTOS is utilized to coordinate motor control, sending step and direction signals to the DRV8255 motor driver to achieve precise 180-degree rotations and stopping of the motor. Additionally, the ESP32 performs frequency-domain analysis using the

Arduino FFT library, converting photodiode intensity data into energy magnitude across frequencies to isolate the 5–10 Hz flicker range characteristic of flame behavior [3]. The resulting data is used to assert a detection flag, which is transmitted to the FPGA. With the ESP32 managing motor control, FFT analysis, and alert signaling, the FPGA plays a critical role in offloading detection coordination and flag handling.

F. Raspberry Pi

The Raspberry Pi 5 was selected for its compatibility with the Raspberry Pi Global Shutter Camera and its ability to perform YCbCr color feature extraction using OpenCV. This analysis runs in parallel with the photodiode-based processing on the ESP32, enabling multi-modal flame detection. OpenCV was chosen due to its extensive library support and flexibility in image processing tasks. The YCbCr color space is particularly effective in low-light conditions, making it well-suited for detecting flame-specific features [4]. Upon detection, the Raspberry Pi transmits a flag to the FPGA to support synchronized, parallel decision-making.

G. FPGA

The FLAMES system is centered around the BASYS3 FPGA development board, featuring a Xilinx Artix-7 chip. The FPGA enables parallel processing of key detection flags, including Fast Fourier Transform (FFT) analysis from the photodiode and YCbCr color feature extraction from the Raspberry Pi 5. Leveraging a 100 MHz system clock, the Artix-7 concurrently executes these modules, facilitating efficient integration between the ESP32 microcontroller and the Raspberry Pi. Acting as an intermediary, the FPGA synchronizes detection events from both sources. Once both flags are asserted, the ESP32 enters a standby state, ready to trigger the alert mechanism, which includes LED and buzzer activation.

III. SYSTEM CONCEPT

Figure 4 illustrates the overall system workflow of the FLAMES system. The user initiates monitoring by starting the system, which begins by continuously collecting data from the TSL2591 light sensor. This sensor performs baseline calibration using algorithms such as Exponential Moving Average with Adaptive Thresholding and Z-Score Anomaly Detection to identify sudden changes in ambient lighting.

Next, the FDS1010 photodiode collects raw ADC data, which undergoes baseline calibration and Fast Fourier Transform (FFT) to detect flicker patterns within the 5–10 Hz frequency range, characteristic of fire. The system also performs color feature extraction in the YCbCr color space by our Raspberry Pi CMOS sensor, with a customized lens system, to differentiate sustained flames from other light sources, such as welding sparks.

If abnormal flicker and color patterns are detected, the system triggers an alarm and notifies the user. A manual reset option allows the user to restart the monitoring cycle. This closed-loop system enables real-time, responsive fire detection while maintaining adaptability and user control in dynamic lighting conditions.

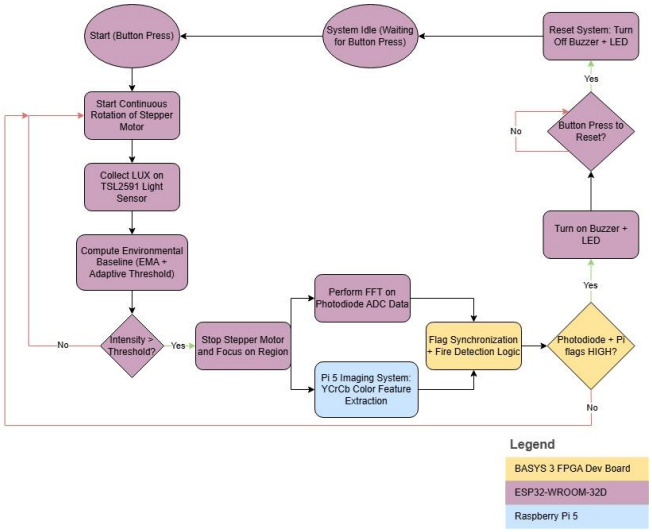

Fig. 4. Software Diagram illustrating FLAMES workflow

IV. HARDWARE DETAILS

In this section, the hardware components of the project will be discussed in further detail. Some components covered in section II are omitted as to not be repeated. The PCB designs will be given its own section and discussed in further detail.

A. Photodiode and Lens System

The photodiode employed in the final prototype of the system is the FDS100. As previously mentioned, the diode was selected for its extensive active area and extended anode and cathode leads. Visible fire emission occurs within the 600–800 nm wavelength range. While flames tend to emit strongly in the near infrared region, we have opted for a silicon diode due to its accessibility and ease of integration with the overall system. During the initial stages of the project, the team intended to utilize an infrared diode, as a significant portion of fire

emission falls within the infrared spectrum. However, infrared diodes with large active areas are typically slim to none due to increased capacitance, resulting in a slower response time. Material and cost constraints also influenced the decision-making process. The responsivity curve of the FDS1010 silicon diode from Thorlabs is presented below.

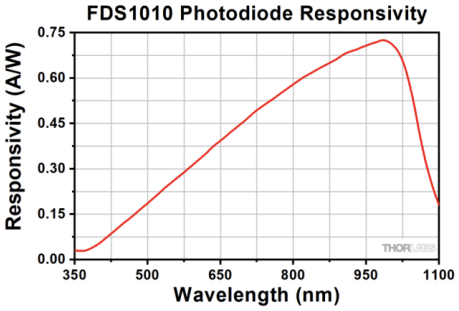

Fig. 5. Shown above is the responsivity curve for the FDS1010 photodiode sourced from the FDS1010 datasheet. [2].

Since the visible flame and spark emissions are strongly observed in the wavelength range of 600 to 800 nm, indicating that the responsivity varies between 0.3 and 0.6 amps/watt. During testing, we aimed to collect the output photocurrent as a function of distance using a NBK7 plano convex lens. It is important to note that at this stage of testing, the diode had not been electrically connected to the OPA 380 operational amplifier. This testing was conducted to verify that photocurrent in the microampere range is being produced. The results yielded a photocurrent in the microampere range, which is sufficient for our application and enables us to have confidence that the OPA 380 will provide adequate gain to the optical signal. *Fig. 6* presents the results.

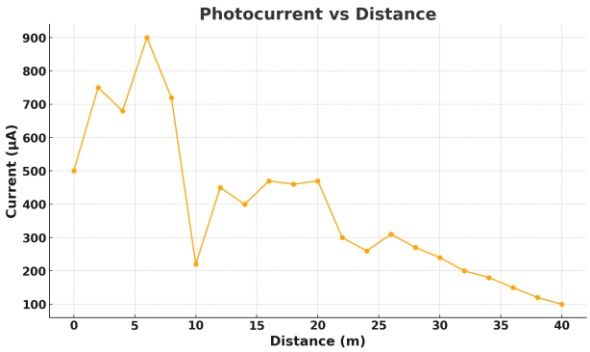

Fig. 6. Photocurrent as a function of distance testing results.

From the data, we can comfortably conclude that there is sufficient photocurrent being produced in the microampere range with the introduction of the 75mm diameter and 100 mm focal length lens. [5].

B. Analog to Digital Converter

After amplification, the analog signal is sent to the ADS8319 analog-to-digital converter (ADC) for digitization. Although the ESP32 has a built-in ADC, we use the external ADS8319 for its higher 16-bit resolution and faster 500 kHz sampling rate [6]. This ensures more accurate capture of subtle signal changes. The ADS8319 connects to the ESP32 via SPI, allowing fast and reliable data transfer for precise signal analysis.

C. Nema 23 Stepper Motor

The Nema 23 was chosen for its excellent performance-to-cost ratio. It provides enough power to drive the scanning mechanism of the fire detection system without sacrificing stability and precision. While we are aware that stepper motors like the Nema 23 are generally not efficient in power consumption, especially when maintaining a holding position, this concern is addressed by implementing a low-power mode within the control system. When the motor is maintaining idle positioning, the motor will enter a sleep mode in conjunction with the selected motor driver so it consumes less power which will allow us to generate less heat.

The selected model of the Nema 23HS22-1006S stepper motor is the most practical choice in terms of cost-effectiveness, its attainable power needs, and assure torque output. Its 0.9 Nm holding torque is sufficient to smoothly rotate our mounted system with enough force to overcome mechanical resistance without compromising the precise scanning alignment. Its rated current of 1A per phase is sufficient for application in more power efficient motor drivers. The moderate mass of 0.7 kg and nominal frame size of 57 x 57 mm of the motor permit easy installation and incorporation into the scanning mechanism without imposing undue mechanical load or requiring overly heavy-duty mechanical structures [7]. This balance between weight, torque, and current draw ensures that the system can perform continuous or on-demand scanning operations with smooth, efficient motion.

D. Stepper Motor Driver

We selected the DRV8825 motor driver due to its balance of performance and control accuracy. One of its key advantages is the support for up to 1/32 micro stepping, which enables extremely smooth and precise motor control. This level of resolution is critical for our

application, as it allows for accurate and controlled scanning of the environment during the fire detection process. High precision ensures minimal position error, which is essential for maintaining sensor alignment and reliability across a 180-degree field of view.

However, during testing we encountered a failure in the DRV8825 motor driver chip, which we suspect was due to thermal issues. The DRV8825 motor driver has a thermal shutdown threshold between 150-180 degrees Celsius. The chip relies on its thermal pad for proper heat dissipation, and thermal resistance plays a critical role in this. Specifically, the junction-to-ambient resistance is 31.6 degrees C/W and the junction-to-case thermal resistance is as low as 1.4 degrees C/W when properly heatsinked [8]. Without sufficient copper area or thermal vias under the pad to transfer heat away from the device. The internal temperature can quickly rise during operation.

This suggests that our custom motor driver board layout may not be adequately dissipating heat through the bottom pad, which is essential for preventing thermal shutdown or permanent damage. Moving forward, we plan to improve the thermal design by incorporating multiple vias to connect to bottom layer copper planes. We also incorporated a parallel driver configuration by soldering 3 DRV8825 driver boards together. Placing the drivers in parallel effectively distributes the total current load between the 3 boards, reducing the thermal and electrical stress on each individual driver. This approach allows each DRV8825 to operate at a lower current level, staying within its safe operating range and improving overall heat dissipation. By sharing the load, the parallel configuration not only improves reliability and extends the lifespan of the drivers but also provides a practical workaround for the thermal limitations of a single chip-especially in applications where active cooling or larger heatsinks may not be feasible. We also combined this with limiting the motor current with reference voltage tuning:

$$Current\ Limit = VREF \cdot 2 \tag{8}$$

Since our selected motor is rated for 1A, we configured VREF to be 0.5V. These enhancements should mitigate overheating and ensure long-term reliability of the DRV8825 within our system.

V. PCB Details

Our project features a custom-designed PCB system comprising several interconnected modules. At the core is the MCU board, which houses the ESP32 microcontroller and serves as the central hub for communication and control. This board will interface with multiple development platforms, including the BASYS3 FPGA board and the Raspberry Pi 5 development board, as well as several custom-designed components. These include the DRV8825 motor driver board for precise stepper motor control, a transimpedance amplifier board (previously mentioned) for processing photodiode signals, and three daughter boards that serve as voltage regulators to distribute power to different sections of the system. Each of these PCBs has been tailored to meet the specific performance, communication, and power requirements ensuring modularity, reliability and efficient integration.

VI. Software Details

To explain the software portion fully, understanding FLAMES' intent is crucial. FLAMES is utilizing three layers of analysis before a final confirmation of a fire is sounded. Each layer can respond with a rejection of a fire and that will deem the detection as not a fire.

Keep in mind that the system will not be able to detect anything other than a fire, if there is a spark or a new constant light source, the system will deem there is no fire but not detect specifically what this light source is.

A. C++ with the ESP32

Due to the system's multi-modal detection architecture, the software is implemented in three distinct programming languages, each corresponding to the hardware platform it runs on. The ESP32-WROOM-32D is programmed in C++ using Arduino-specific libraries, including those shown below:

```
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2591.h>
#include <SPI.h>
#include "arduinoFFT.h"
```

Fig. 7. FLAMES Arduino IDE Libraries Used

(1) The Adafruit_Sensor.h provides a standardized interface for reading sensor data, enabling seamless integration with the TSL2591 light sensor.
(2) The Adafruit_TSL2591 library interfaces with the TSL2591 sensor to capture high-resolution ambient light data. Adjustable gain and timing settings allow it to detect sudden lux spikes, serving as the system's initial trigger.

(3) The SPI library facilitates communication between the ESP32 and the ADS8319 ADC, which digitizes photodiode output for real-time FFT analysis. While SPI is manually toggled, the library ensures proper pin configuration.
(4) The arduinoFFT library performs FFT on ADC data to detect 5–10 Hz flicker frequencies typical of fire. Results are analyzed using EMA and adaptive thresholding to confirm fire presence.

Our C++ implementation consists of 358 lines of code, utilizing the libraries previously discussed. To leverage these libraries effectively, we have structured the system around the key functions outlined:

```
void setup()
void loop()
void handleButton()
void stepperTask(void* parameter)
void sensorTask(void* parameter)
float readLux()
uint16_t readADS8319()
bool runFftDetection()
void resetToStartPosition()
```

Fig. 8. FLAMES C++ ESP32 Function Calls

(1) The setup() function initializes all hardware components and configurations. It sets up motor, sensor, and communication pins; initializes SPI for ADC communication; configures the TSL2591 light sensor; calculates FFT frequency bin ranges; and creates two FreeRTOS tasks: stepperTask for motor control and sensorTask for sensor monitoring. The system begins in an idle state, waiting for a button press.
(2) The loop() function runs continuously as the main control loop. It checks for user input through handleButton() and blinks an LED when fire is detected. If the system is idle, it yields control to background tasks, keeping the system responsive.
(3) The handleButton() function monitors the physical button using software debouncing. Pressing the button toggles the system between active and idle. Activation resets key flags, recenters the motor, silences the buzzer, and resets the FPGA. Deactivation stops all motion and alerts, returning the system to standby.
(4) The stepperTask() function is a FreeRTOS task that handles bidirectional sweeping of the stepper motor across 180 degrees. It moves outward, then returns to center before switching directions. The task pauses when fire is detected or the system is shut off, allowing consistent directional coverage.
(5) The sensorTask() function is a FreeRTOS task that continuously reads data from the TSL2591 light sensor. It calculates a dynamic baseline using an Exponential Moving Average (EMA) and compares it to current readings. If a sudden change is detected, it pauses the motor, triggers the Raspberry Pi to begin camera analysis, and runs runFftDetection() to check for flicker patterns. If fire is confirmed, it activates the LED, buzzer, and stops the motor until manually restarted.
(6) The readLux() function retrieves the visible light intensity from the TSL2591 sensor by removing the infrared portion from the raw signal. This lux value is used to detect abnormal changes in ambient lighting.
(7) The readADS8319() function performs SPI communication with the 16-bit ADS8319 ADC to digitize the analog output from the photodiode. This data is later used for frequency-based fire detection.
(8) The runFftDetection() function analyzes photodiode data for flickering in the 5–10 Hz range, which is characteristic of fire. It collects samples, runs FFT with a Hamming window, and compares the energy in the fire band to an adaptive threshold. If detection conditions are met, it sends a signal to the FPGA via a GPIO pin and waits for confirmation before triggering a full fire alert.
(9) The resetToStartPosition() function moves the stepper motor back to its center position based on its tracked step count. This ensures consistent orientation and avoids drift over time, especially after resets or manual interruptions.

B. Python with the Raspberry Pi

The next stage of the system involves the Raspberry Pi 5, which interfaces with a Raspberry Pi Camera Module and a custom lens configuration. This component is programmed in Python with OpenCV and handles image-based analysis. The primary functions used in this subsystem are outlined below:

```
def start_camera()
def stop_camera(cap)
main while loop
```

Fig. 9. FLAMES Raspberry Pi 5 Python Functions

(1) The start_camera() function initializes the Raspberry Pi camera using a GStreamer pipeline

compatible with cv2.VideoCapture, configured for 640×480 resolution at 30 FPS in BGR format. If the camera fails to open, it returns None, allowing the system to skip detection safely.

(2) The stop_camera() function releases the camera resource and prints a debug message. It is called when fire is not detected for a set time or after a 10-second alert window, ensuring a proper shutdown and returning the system to an idle state.

(3) The main while loop serves as the Pi's main execution flow and operates in three phases:
   - Idle phase: Waits for a signal from the ESP32 (via TRIGGER_PIN) indicating the TSL2591 has detected a sudden change in light.
   - Detection phase: Starts the camera and analyzes enhanced frames in the YCbCr color space. Fire is identified based on brightness, color differences, and flicker. If confirmed, a fire flag is latched.
   - Post-detection phase: When fire is latched, FPGA_FLAG_OUT is set HIGH to notify the FPGA. The flag remains active for 10 seconds. After this or if no fire is found during a timeout, the camera is stopped and the system returns to idle.

The Raspberry Pi 5 performs color-based analysis using the YCbCr color space to extract features from the light source when the TSL2591 light sensor flags a potential anomaly.

C. *Verilog with the FPGA*

In parallel, the BASYS3 FPGA development board is programmed in Verilog to handle flag synchronization and execute fire detection logic. This logic determines whether the system should respond to the detected anomaly. The associated hardware modules are outlined below:

```
module top
module fire_alert_logic
```

Fig. 10. FLAMES FPGA Verilog Modules

(1) The top module connects the FPGA to the ESP32 and Raspberry Pi, coordinating signal flow between devices. It manages incoming fire detection flags from both sources and passes them to the fire_alert_logic module for evaluation. It also routes debug outputs to onboard LEDs to visually indicate which subsystems are active. This module plays a key role in synchronizing asynchronous flag signals from different devices into the FPGA's unified clock domain.

(2) The fire_alert_logic module handles the core detection logic and flag synchronization. It ensures that both the ESP32 and Raspberry Pi fire signals are valid and stable within the FPGA's timing constraints before issuing a final alert. Only when both flags are confirmed high simultaneously does it assert the final_alert_out signal to the ESP32. It also outputs internal debug signals for visual confirmation and system testing.

You can see the FPGA hardware diagram here:

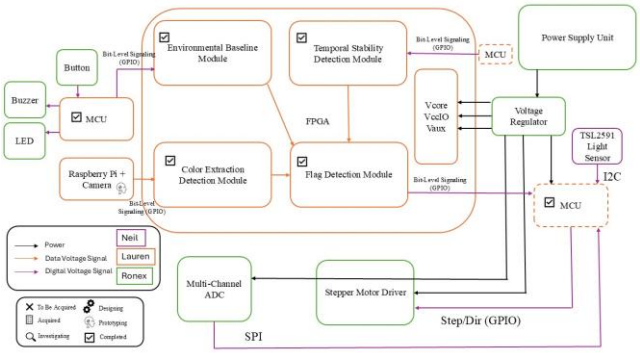

Fig. 11. BASYS 3 FPGA Dev. Board Hardware Diagram

As shown in Fig. 11, the system is conceptually divided into four distinct modules. While each of these could have been implemented as separate Verilog modules, they were consolidated into a single module, fire_alert_logic, for simplicity and clarity in the final implementation. This organization maintains the conceptual separation while streamlining the code structure. The GitHub for our system can be found here [9].

VII. POWER MANAGEMENT SYSTEM

TABLE I
POWER MANAGEMENT

Component	Operating Voltage
BASYS 3 Artix-7 FPGA	5V
ESP32-WROOM-32D	3.3V
Raspberry Pi 5	5V
ADS 8319	3.3V
DRV8825	3.3V/12V
OPA 380	3.3V

The entire system will be powered using a 12V, 10A wall adapter, which serves as the primary power supply. This 12V input is routed to a set of custom-designed voltage regulator PCBs, each responsible for providing the appropriate voltage to various components within the system. To achieve the necessary voltage levels, we utilized the LM2679 adjustable switching voltage regulator, which efficiently steps down the 12V input to both 5V and 3.3V outputs. This allows us to meet the operating voltage requirement of all components, as summarized in Table 1. The DRV8825 motor driver requires both 3.3V, for the reset and sleep pins of the IC, and 12V to enable operation. This power distribution strategy ensures stable, efficient, and isolated voltage regulation across all subsystems.

THE ENGINEERS

Neil Singh is an optical engineering student at CREOL. He will be working as an optical designer for L3 Harris's Space Imaging Division in Boston, Massachusetts, after graduating. He is planning on pursuing his graduate degree online through CREOL.

Ronex Faustin is an undergraduate of Electrical Engineering planning to obtain hands on experience within the industry. After graduation, he will be pursuing his Professional Engineering License in Electrical Engineering and plans to remain in the industry in electrical design.

Lauren Caccamise is an undergraduate Computer Engineering student with a minor in mathematics. After graduation, she is set to pursue a Ph.D. in Electrical and Computer Engineering at Purdue University.

ACKNOWLEDGEMENT

We'd like to express our deepest gratitude to Dr. Arthur Weeks, Dr. Chung Yong Chan, and Dr. Aravinda Kar for their invaluable support and guidance. Their leadership has been instrumental in helping us step outside our comfort zones and embrace new challenges. As for our review committee, we'd like to express our deepest gratitude for giving us the chance to showcase our senior capstone project.

REFERENCES

[1] "opa380.pdf." Accessed: Jul. 06, 2025. [Online]. Available: https://www.ti.com/lit/ds/symlink/opa380.pdf?ts=1751796579604&ref_url=www.mouser.com

[2] "FDS1010-SpecSheet.pdf." Accessed: Jul. 05, 2025. [Online]. Available: https://www.thorlabs.com/drawings/5e2fbe2cd15e0c46-9D6702B3-EC61-FC3F-002DB79FC642E54F/FDS1010-SpecSheet.pdf

[3] "7.3 SW_Flame_School_Fire_Radiation," *Sense-Ware*, p. 1.

[4] "YCbCr Color Space: Understanding Its Role in Digital Photography," PRO EDU. Accessed: Jul. 06, 2025. [Online]. Available: https://proedu.com/blogs/photography-fundamentals/ycbcr-color-space-understanding-its-role-in-digital-photography

[5] "Group 6 Senior Design 1 Final Paper."

[6] "ADS8319 data sheet, product information and support | TI.com." Accessed: Mar. 17, 2025. [Online]. Available: https://www.ti.com/product/ADS8319?utm_source=google&utm_medium=cpc&utm_campaign=asc-dc-null-44700045336317134_prodfolderdynamic-cpc-pf-google-ww_en_int&utm_content=prodfolddynamic&ds_k=DYNAMIC+SEARCH+ADS&DCM=yes&gad_source=1&gclid=Cj0KCQjwkN--BhDkARIsAD_mnIphZ_g2CAzGmwVd7Vbo9hbTm3yCz7ja5x3U179n4RT4quilqDM7mUIaAp5LEALw_wcB&gclsrc=aw.ds#params

[7] "Stepper Motor Frame Sizes," Oriental Motor U.S.A. Corp. Accessed: Mar. 12, 2025. [Online]. Available: http://www.orientalmotor.com/stepper-motors/stepper-motor-frame-sizes.html

[8] "Pololu - DRV8825 Stepper Motor Driver Carrier, High Current." Accessed: Mar. 17, 2025. [Online]. Available: https://www.pololu.com/product/2133

[9] L. Caccamise, *laurencacc/FLAMES*. (Jul. 06, 2025). C++. Accessed: Jul. 06, 2025. [Online]. Available: https://github.com/laurencacc/FLAMES