**MIDDLE EAST TECHNICAL UNIVERSITY**

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**

# EE447 TERM PROJECT

# TEMPERATURE-INITIATED OBJECT DETECTION

## I. INTRODUCTION

This report examines the term project for Introduction to Microprocessors course, where the primary task is to detect an object after temperature changes. The Temperature-Initiated Object Detection System is an embedded project designed to demonstrate the integration of multiple sensing technologies and microcontroller functionalities to achieve a multifunctional task. Utilizing the TM4C123G microcontroller, the system employs two temperature sensors (LM35 and BMP280) and an ultrasonic distance sensor to create a responsive and efficient object detection mechanism. The system operates primarily in deep sleep mode, conserving energy until an increase in temperature is detected. Upon activation, it transitions into a high-precision operational mode, leveraging I2C and SPI communication protocols for sensor interfacing and data presentation. The project serves as an advanced application of concepts such as analog-to-digital conversion, interrupt handling, and motor control. Additionally, it integrates user interface elements like the Nokia 5110 LCD, RGB LEDs, and a 4x4 keypad to provide comprehensive feedback and control. By implementing various hardware and software constraints, the project emphasizes the importance of designing energy-efficient and modular systems suitable for real-world applications.

## II. THE SYSTEM

In this section, we delve into the architecture and functionality of our temperature-initiated object detection system. The system is divided into six main parts, each playing a crucial role in achieving precise and efficient object detection based on temperature variations. The accompanying Figure 1 showcases the project, consisting of the integration of various sensors, the stepper motor, and the Nokia 5110 LCD display. Moreover, the pin diagram is in Appendix section Figure 7.

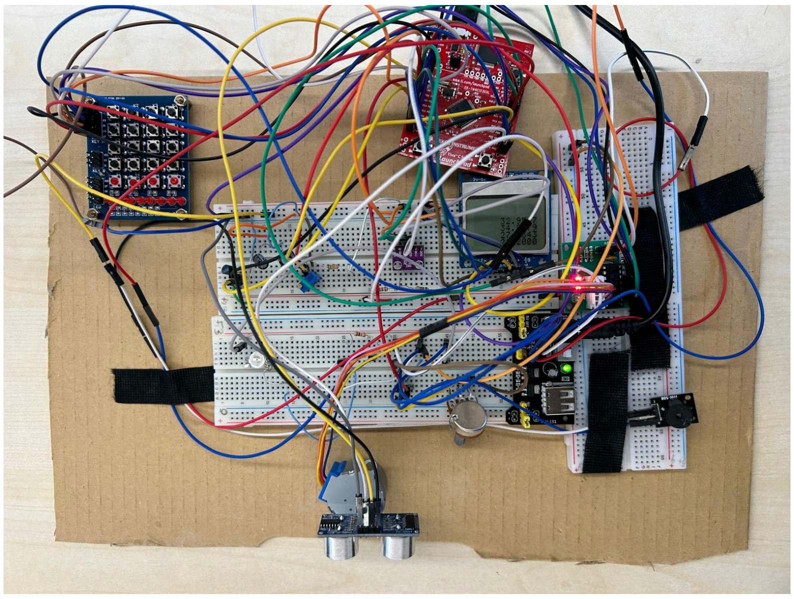
*Figure 1: The Overall Project*

**Subsystem 1: Deep Sleep**

The Deep Sleep Subsystem of our project focuses on minimizing power consumption while maintaining responsiveness to external events. In this mode, the microprocessor halts the system clock to conserve energy and enters a low-power state, utilizing the "WFI" (Wait For Interrupt) instruction. This ensures that the processor remains in deep sleep mode until an interrupt triggers a wake-up.

Interrupt subroutines are designed to efficiently manage state transitions. Upon being triggered, the appropriate interrupt handler transitions the microprocessor to the required operational state. At the end of each routine, the system reverts to the deep sleep mode, ready to respond to subsequent interrupts.

Additionally, we incorporated a dedicated button that forces the microprocessor into deep sleep immediately, allowing manual control over power states. This design ensures both energy efficiency and operational reliability, seamlessly integrating deep sleep functionality with the system's overall state management.

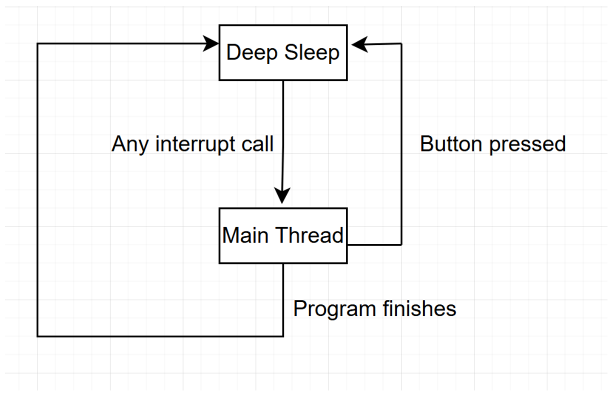

*Figure 2: Flowchart of Deep Sleep Subsystem*

## Subsystem 2: Temperature Detection (LM35 and BMP280) and Trimpot

The Temperature Detection subsystem is designed to monitor environmental temperature using two sensors: the LM35 and the BMP280. The LM35, an analog temperature sensor, is connected to the analog comparator module of the TM4C123 microcontroller. It operates during the deep sleep mode of the system to detect a rise in temperature above a set threshold, which can be adjusted using a multiturn trimpot. The temperature from the LM35 and trimpot is captured via the ADC by converting its analog output voltage into a digital value. The ADC samples the sensor's output and converts it to a corresponding temperature value using a linear conversion formula. If the LM35 detects a temperature increase beyond this threshold, an analog comparator interrupt wakes the system. Upon waking, the BMP280, a digital temperature sensor, measures the temperature with higher precision using the I2C protocol. The BMP280 communicates its temperature readings digitally, eliminating the need for additional ADC conversions. For improved accuracy, the BMP280 takes 128 readings, averages these values, and confirms if the temperature exceeds the digital threshold set by the keypad.

The I2C operation for the BMP280 involves establishing a communication link between the microcontroller and the sensor using two dedicated lines: SDA (data) and SCL (clock). The microcontroller acts as the I2C master, initiating communication and generating the clock signals, while the BMP280 operates as the slave device. The process begins with the master sending a start condition, followed by the sensor's address to select it for communication. Once the BMP280 acknowledges the address, the master sends read or write commands to request data or configure the sensor. For temperature measurement, the BMP280 responds with its readings in digital format. These readings are retrieved as multiple bytes, which the microcontroller combines to compute the precise temperature value. The communication ends with a stop condition, signaling the completion of the I2C operation. This protocol ensures reliable and efficient data exchange with minimal pin usage.

This subsystem also integrates with the user interface; when the LM35 detects a temperature rise, a power LED, driven by a transistor and an external power source, is turned on and remains active until the system re-enters deep sleep mode. If the BMP280 confirms the temperature exceeds the threshold, a speaker emits a two-second square wave alert.

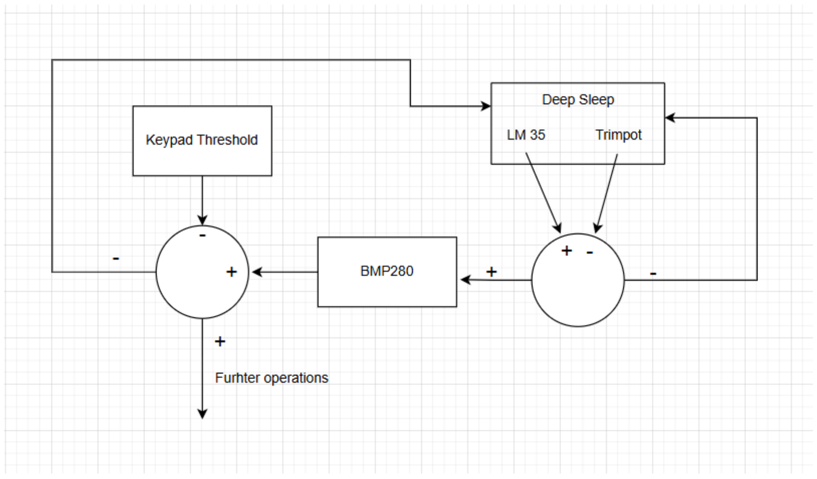

*Figure 3: Flowchart of Temperature Sensing Subsystem*

**Subsystem 3: Motor and Sensor**

The motor and sensor subsystem works by combining stepper motor control with ultrasonic distance sensing to detect objects within a scanning range of -90° to 90°. The process begins with initializing the necessary components, including GPIOs for motor control, SysTick for timing, and Timer2A for capturing echo pulse width from the ultrasonic sensor. Once initialized, the motor starts scanning from -90°, moving in fixed steps and incrementally updating the angle as it progresses. At each step, the ultrasonic sensor is triggered to send a pulse, and the system waits for the echo to return. The echo's pulse width is captured using Timer2A, and the distance to an object is calculated based on the duration of the echo. The distance values are stored in a buffer for further processing.

As the motor completes its forward scan to 90°, it reverses direction back to -90°, continuing to measure distances. If an object is detected within 1000 mm during the scan, the distance and angle are recorded. Once the scan is complete, the system calculates the average distance from the buffer and displays the results on the Nokia 5110 LCD. If no object is detected, a "Object is not detected." message is displayed. The system then resets the motor position and distance buffer, reinitializing the SysTick timer for the next scanning cycle. This systematic approach ensures accurate detection and clear feedback for the user.

The implementation requires two distinct SysTick initializations, necessitating reinitialization of SysTick in each case. Since only one SysTick initialization and its corresponding handler function can be utilized, specific cases are defined for the motor and sensor operation versus other tasks. The SysTick configuration for the motor is significantly faster than for the other components to ensure the required high-speed motor operation. However, modifying the general SysTick settings for the motor disrupts the functioning of other system parts. To address this, an if-else condition is incorporated within the SysTick initialization and handler functions. The system operates by first determining whether it is in the motor case. If it is not, the general SysTick initialization is applied. If it is, the SysTick configuration is updated specifically for the motor and sensor operations. Once the motor completes its movement, the SysTick settings are reverted to their initial values, allowing the system to resume operations for the other tasks. The flowchart follows:

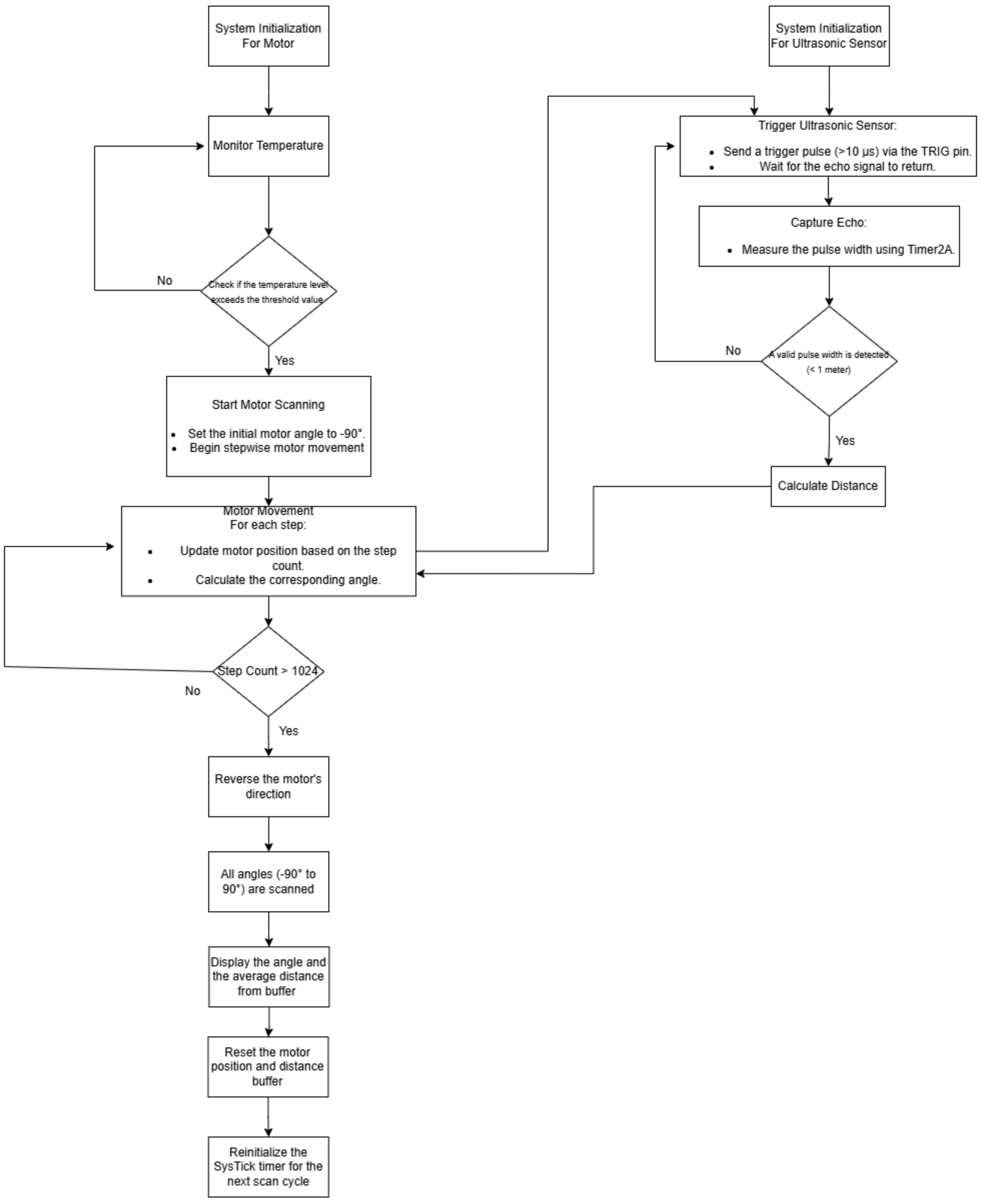
System Initialization For Motor
Monitor Temperature
Check if the temperature level exceeds the threshold value
No
Yes
Start Motor Scanning
Set the initial motor angle to -90°.
Begin stepwise motor movement
Motor Movement For each step:
Update motor position based on the step count.
Calculate the corresponding angle.
Step Count > 1024
No
Yes
Reverse the motor's direction
All angles (-90° to 90°) are scanned
Display the angle and the average distance from buffer
Reset the motor position and distance buffer
Reinitialize the SysTick timer for the next scan cycle
System Initialization For Ultrasonic Sensor
Trigger Ultrasonic Sensor:
Send a trigger pulse (>10 μs) via the TRIG pin.
Wait for the echo signal to return.
Capture Echo:
Measure the pulse width using Timer2A.
A valid pulse width is detected (< 1 meter)
No
Yes
Calculate Distance

*Figure 4: Flowchart of Motor and Sensor*

## Subsystem 4: Display (NOKIA 5510 LCD)

The LCD screen algorithm in this project is designed to visually display critical system information such as temperature readings, object detection results, and a graphical representation of detected objects. The Nokia 5110 LCD is utilized for its simplicity and ability to display both text and graphics. The algorithm efficiently manages the transition between different operational states of the system, ensuring that the displayed information is relevant and updated in real-time.

The LCD algorithm begins by initializing the screen and defining a structured layout for displaying data. During normal operation, the algorithm first checks the system's state and dynamically updates the display content based on the ongoing tasks. In the initial state, temperature readings from the LM35 sensor and threshold values are shown. These values are formatted and displayed using the Nokia5110_SetCursor and Nokia5110_OutUDec functions, ensuring proper alignment and readability. When the motor starts scanning for objects, the algorithm clears the screen to prepare for new information. If an object is detected within a 1000 mm range, the algorithm displays a message indicating detection, along with the distance and angle of the object relative to the system. This information is updated dynamically for every step of the motor. If no object is detected during the scan, the screen shows a message stating "Object is not detected." For graphical representation, the algorithm calculates pixel coordinates on the LCD using the object's angle and distance. The GetPixelCoordinates function maps these polar coordinates into the Cartesian plane of the LCD screen, scaling the distance appropriately for display. The algorithm then sets the corresponding pixel on the screen using Nokia5110_SetPxl. This graphical feature provides a clear visualization of object positions within the scanned area. The algorithm handles transitions between states, such as clearing the screen when necessary or redrawing graphics after significant changes. It ensures that the user interface remains intuitive and responsive by updating the display content in synchronization with the motor and sensor operations. The flowchart of the display unit can be seen in Figure 5.

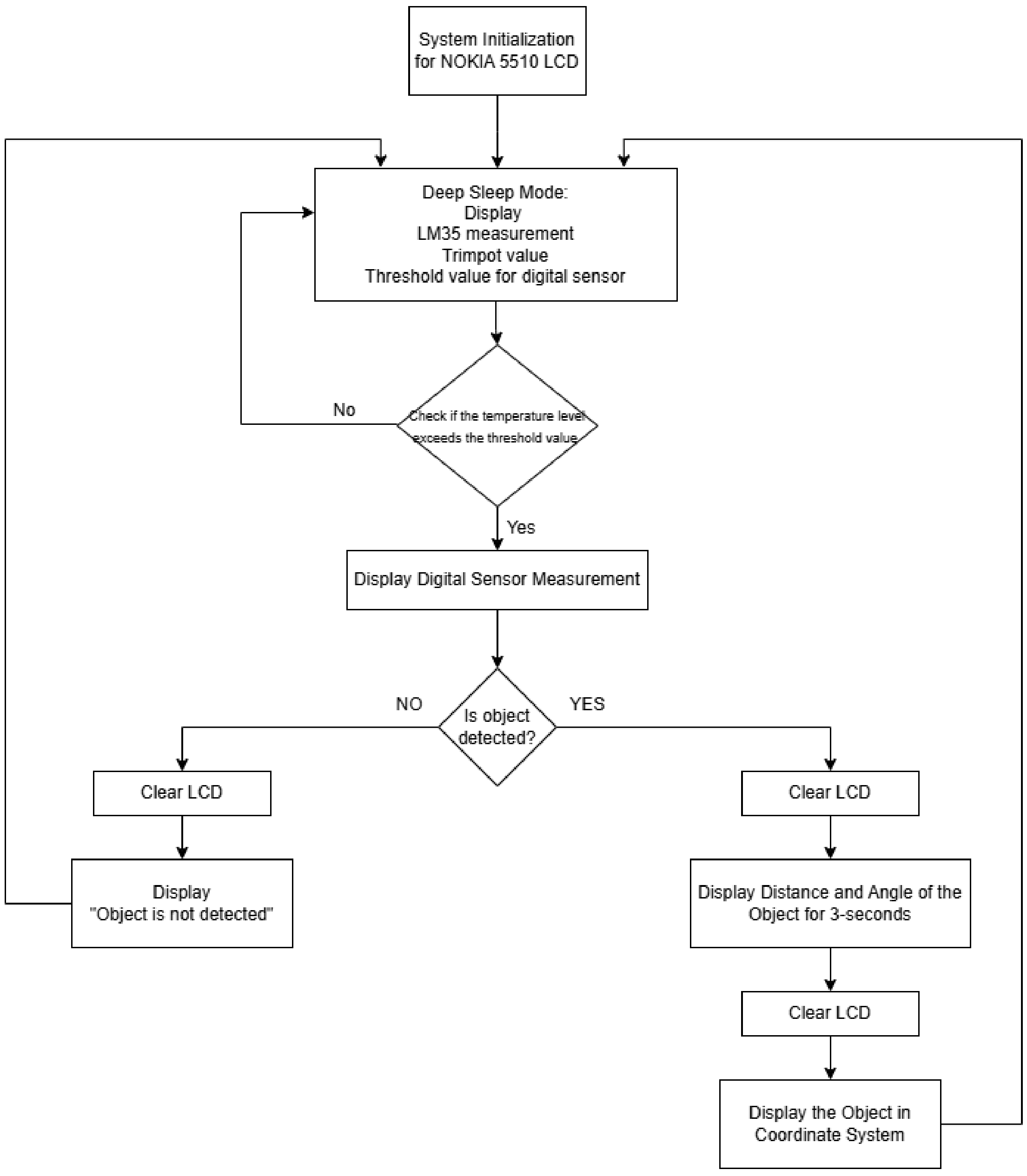
System Initialization for NOKIA 5510 LCD
Deep Sleep Mode: Display LM35 measurement Trimpot value Threshold value for digital sensor
Check if the temperature level exceeds the threshold value
No
Yes
Display Digital Sensor Measurement
Is object detected?
NO
YES
Clear LCD
Display "Object is not detected"
Clear LCD
Display Distance and Angle of the Object for 3-seconds
Clear LCD
Display the Object in Coordinate System

*Figure 5: The Flowchart of the LCD unit.*

**Subsystem 5: Keypad Module**

The Keypad Module subsystem is implemented to allow user interaction for adjusting the digital temperature threshold and issuing system commands. This subsystem utilizes a 4x4 matrix keypad, interfaced with the TM4C123 microcontroller. The keypad operation is implemented in assembly language, ensuring optimized performance and direct control over the microcontroller's GPIO pins. The rows and columns of the keypad are connected to specific GPIO pins configured for input and output modes. The algorithm for scanning the keypad involves setting each row as an output while the other rows are set as inputs, and then reading the column pins for a pressed key. This process is repeated for each row in a loop until a key press is detected. Debouncing logic is included to ensure accurate keypress detection by ignoring transient signals caused by physical keypresses. When a key is pressed, its corresponding value is determined based on the active row and column combination. The detected key value is then written to a specific memory location in the microcontroller's SRAM. This allows the main program, running in C, to read the keypad input from the memory location and process it accordingly. The interaction between the assembly code and the main C program is achieved through a well-defined memory mapping strategy, ensuring smooth data exchange. The assembly code responsible for the keypad operation writes the detected key value to a predefined memory address using store instructions. The main program continuously monitors this address and retrieves the key value when updated. This mechanism ensures seamless integration of the keypad functionality into the larger system, enabling dynamic threshold adjustment and other user commands. To handle keypad inputs efficiently, the GPIOB interrupt is configured for the keypad's lower row. The lower row pins of the keypad matrix are connected to GPIOB, with each pin configured to detect edge-triggered interrupts. This setup ensures that the microcontroller promptly responds to any key press without continuous polling, saving processing resources.

**Subsystem 6: Buzzer and Power LED**

The buzzer system in this project is implemented using Timer 2 on the TM4C123G microcontroller, where the frequency of the sound varies based on the temperature detected by the BMP280 sensor. The *pulse_init* function initializes the Timer 2B module and configures it to generate periodic square pulses that drive the buzzer. The frequency of these pulses, and hence the sound produced, is determined by adjusting the *HIGH* and *LOW* values, which correspond to the ON and OFF times of the pulse.

When the temperature exceeds the threshold, the system activates the buzzer by setting the PB1 pin high and toggles its state every period. The timer interrupt ensures precise timing for the pulse generation and keeps the buzzer active for a maximum duration of 2 seconds. After 2 seconds, the timer is disabled, and the buzzer stops. This functionality effectively provides both audible feedback for high temperatures and a dynamic response, where the pitch of the sound changes with the detected temperature, enhancing user awareness. The flowchart can be seen in Figure 6.

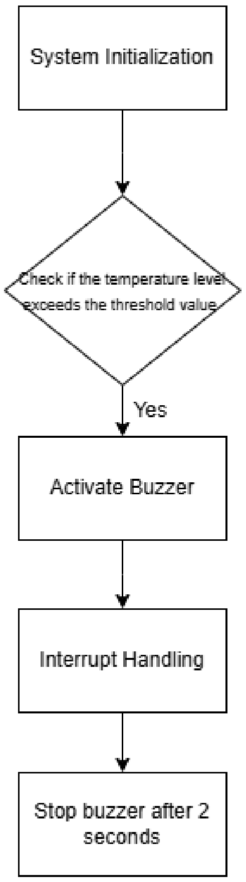
System Initialization
Check if the temperature level exceeds the threshold value
Yes
Activate Buzzer
Interrupt Handling
Stop buzzer after 2 seconds

*Figure 6: Flowchart of the buzzer*

The power LED in the project is a key visual indicator that the system has detected a temperature rise above the threshold and exited deep sleep mode. The LM35 analog temperature sensor monitors the ambient temperature, and its output is fed into an analog comparator within the TM4C123G microcontroller. When the temperature exceeds the threshold, defined by a reference voltage adjustable via a multiturn trimpot, the comparator generates an interrupt signal to wake the system. Upon waking, the microcontroller activates a GPIO pin connected to a transistor, which drives the power LED. This transistor-based circuit ensures that the LED receives adequate current without overloading the GPIO pin. The LED remains illuminated throughout the system's active state, serving as a continuous indicator of operation. It turns off only when the user presses a designated button to re-enter deep sleep mode, at which point the GPIO pin driving the LED is set low. This design provides a reliable and efficient means of signaling the system's status.

**Subsystem 7: On board RGB LEDs**

The Onboard RGB LEDs subsystem provides visual feedback based on the system's operations and object detection results. The TM4C123 microcontroller directly controls the onboard Red, Green, and Blue LEDs, which are connected to specific GPIO pins. These LEDs are used to indicate the proximity of detected objects within predefined distance ranges during the scanning operation of the ultrasonic sensor. Each LED is connected to a GPIO pin configured as an output. The microcontroller sets the appropriate pin high to turn on the corresponding LED, or low to turn it off. The RGB LEDs operate based on the following logic:

- If the detected object's distance is less than 50 cm, the Red LED is turned on, while the Green and Blue LEDs are turned off.
- If the detected object's distance is between 50 cm and 75 cm, the Blue LED is turned on, while the Red and Green LEDs are turned off.
- If the detected object's distance is between 75 cm and 100 cm, the Green LED is turned on, while the Red and Blue LEDs are turned off.
- If no object is detected or the distance exceeds 100 cm, all RGB LEDs remain off.

## III. CONCLUSION

The Temperature-Initiated Object Detection System successfully combines temperature sensing, object detection, and user interaction into an efficient solution. By incorporating deep sleep mode, the system demonstrates the ability to conserve energy while maintaining readiness for activation. The seamless integration of components such as the LM35, BMP280, and HC-SR04 sensors, along with the effective use of communication protocols, showcases the versatility and capability of the TM4C123G microcontroller.

Through careful implementation of interrupt-driven tasks, stepper motor control, and visual feedback mechanisms, the project meets its objectives of accurate heat detection, object localization, and user-friendly operation. This work not only fulfills the requirements of the laboratory but also provides valuable insights into the development of embedded systems for practical and industrial use. It highlights the potential for further enhancement, such as dynamic threshold adjustment and advanced data visualization, paving the way for future advancements in intelligent detection systems.

## APPENDIX

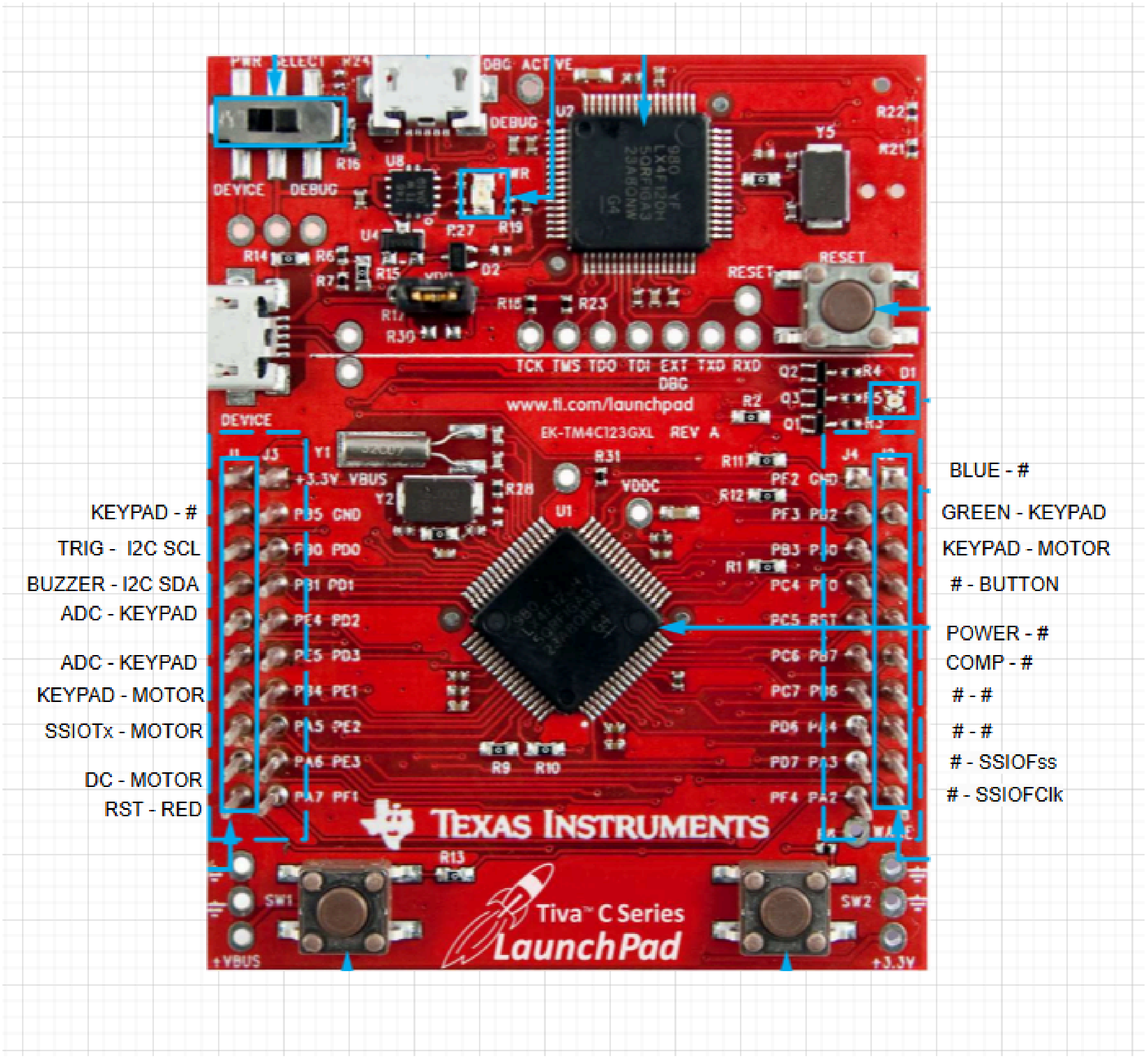

*Figure 7: PIN diagram*