IoT Competition using Raspberry Pi

Intel IoT Club, Amrita Vishwa Vidyapeetham

Date: August 19, 2023 (Saturday) – 10 AM to 4 PM

Event: Online

Group No – 2

Team Member:

1. GEETH VISHNU G – CB.EN.U4CSE21119
2. CHAITANYA VARMA R – CB.EN.U4CSE21148
3. ZAKEER AHAMAD SHAIK – CB.EN.U4CSE21155

Abstract:

Embarking on an unprecedented journey into coal mine safety, our project unveils an ingenious evacuation system powered by Raspberry Pi and Grove Pi. Picture a network of cutting-edge sensors and smart devices working harmoniously to safeguard miners from potential dangers. From monitoring temperature fluctuations and air quality to alerting in case of sound anomalies, this system creates a fortified shield of protection. When emergencies strike, piercing alarms kick into action, guiding miners to safety. Imagine a dynamic LCD display providing real-time information, pinpointing the locations of life-saving emergency kits. Not stopping there, our system even adapts to ambient lighting. Witness a game-changer in safety technology that sets the new standard for hazardous environments, all wrapped up in a customizable package tailored to specific needs.

Introduction:

Venturing into the depths of coal mines, our project emerges as a technological sentinel for safety. Welcome to the realm of the "Coal Mine Evacuation System" - an innovation that fuses Raspberry Pi and Grove Pi prowess to reimagine safety within mining. Through an intricate web of sensors and intelligent alerts, we navigate miners through challenges, ushering in an era of real-time protection and responsive guidance. Join us on a journey that marries cutting-edge technology with unwavering dedication to miners' safety.

Hardware Required:

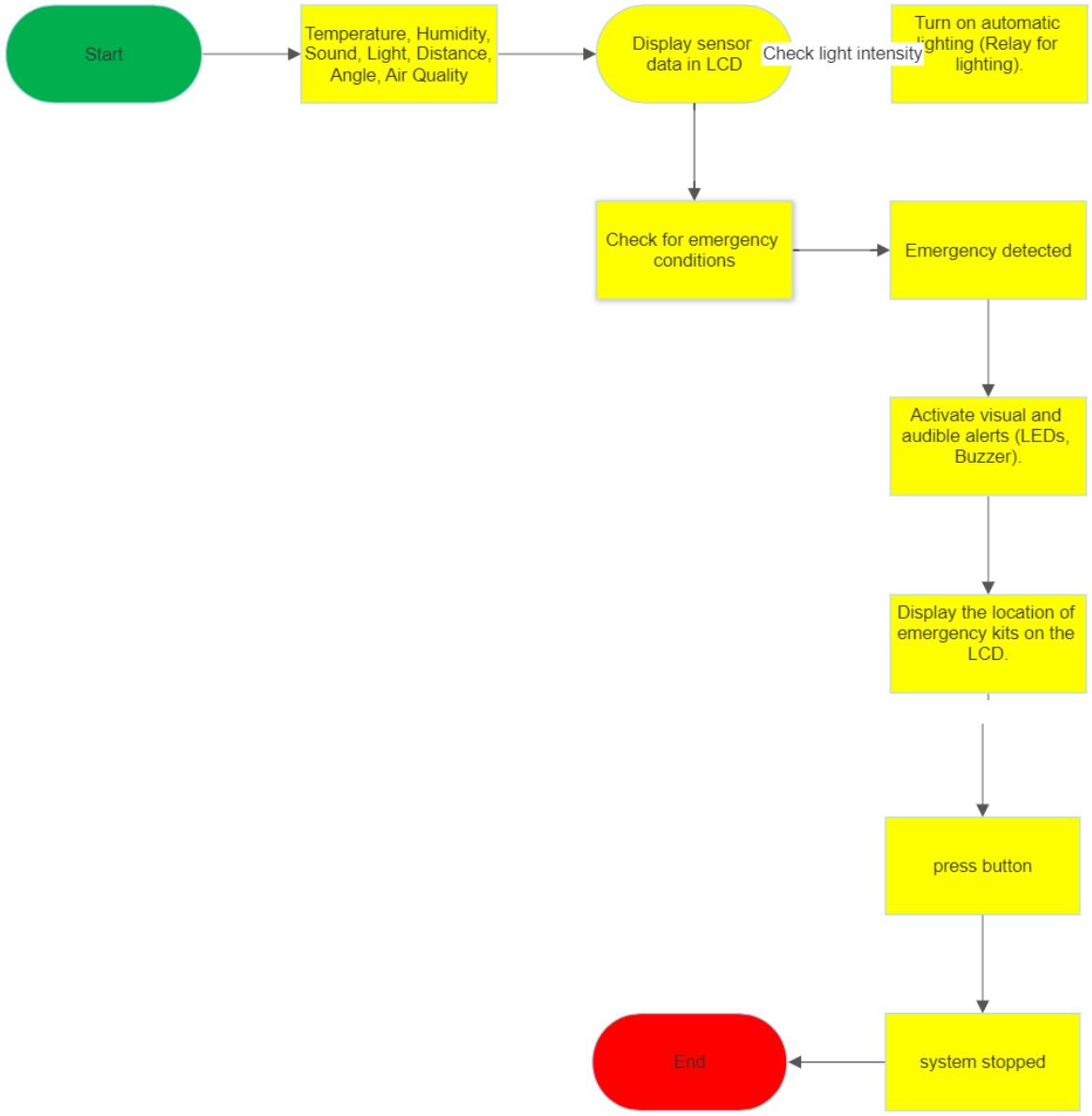
1. Raspberry Pi 3
2. Grove Pi Sensors

Sensors and Actuators:

- LEDs (Red, Blue, Green)
- Sound Sensor
- Temperature and Humidity Sensor
- Light Sensor
- Button
- Ultrasonic Sensor
- Rotary Angle Sensor
- Buzzer
- RGB LCD Display
- Air Quality Sensor

- Relay
- Power Supply
- Jumper Wires(In case of ethernet)

Flow chart:



Initialization:
Initialize sensors (LEDs, Sound, Temperature/Humidity, Light, Button, Ultrasonic, Rotary, Buzzer, LCD Display, Air Quality).
Initialize communication (Email, LCD, etc.).
Set constants (Thresholds, Emergency Kits Location, etc.).
Display system startup message on the LCD.
System Activation:
Wait for the button press.
Display "System Activated" on the LCD.
Enter the main loop.
Main Loop:
Read sensor data (Temperature, Humidity, Sound, Light, Distance, Angle, Air Quality).
Display sensor data on the LCD.

Check for emergency conditions (Temperature, Sound, Distance, Air Quality).
3.1. Emergency Detected:
- Activate visual and audible alerts (LEDs, Buzzer).
- Display emergency message on the LCD.
- Log the emergency event.
- Send emergency email notification.
- Execute specific emergency actions based on the detected condition.
3.2. No Emergency:
- Display safe status indicators (LEDs, LCD).
- Display specific sensor data.
- Check if light intensity requires automatic lighting control.
Automatic Lighting Control:
If light intensity is low:
Turn on automatic lighting (Relay for lighting).
Display "Automatic Lighting: ON" on the LCD.
If light intensity is sufficient:
Turn off automatic lighting.
Display "Automatic Lighting: OFF" on the LCD.
Emergency Kits Location:
Display the location of emergency kits on the LCD.
User Interaction:
Provide information and feedback on the LCD.
Allow users to view sensor data, emergency history, etc.
Loop Iteration:
Repeat the loop at a specified interval (e.g., every second).
User Interaction (Button Press):
If the button is pressed:
Display "System Stopped" on the LCD.
Turn off LEDs, Buzzer, and automatic lighting.
Close communication channels.
Terminate the program.

Purpose of each sensor:

LEDs (Red, Blue, Green):
Purpose: Visual indication of system status and emergency alerts.
Red LED: Indicates emergency conditions.
Blue LED: Indicates system activation and status.
Green LED: Indicates safe conditions.

Sound Sensor:
Purpose: Detects unusual or loud noises that might indicate hazardous situations, such as collapses or machinery malfunctions.

Temperature and Humidity Sensor:
Purpose: Monitors the temperature and humidity levels in the mine to ensure optimal working conditions and to detect overheating or excessive moisture.

Light Sensor:
Purpose: Measures ambient light intensity to adjust lighting controls and ensure optimal visibility for miners during evacuations.

Button:
Purpose: Initiates the activation of the system. Miners press the button to start the monitoring and emergency response functions.

Ultrasonic Sensor:
Purpose: Measures distances to detect obstacles or obstructions in the path, helping miners navigate safely.

Rotary Angle Sensor:
Purpose: Measures the angle of rotation. It can be used for tasks like detecting the position of doors, hatches, or other movable objects.

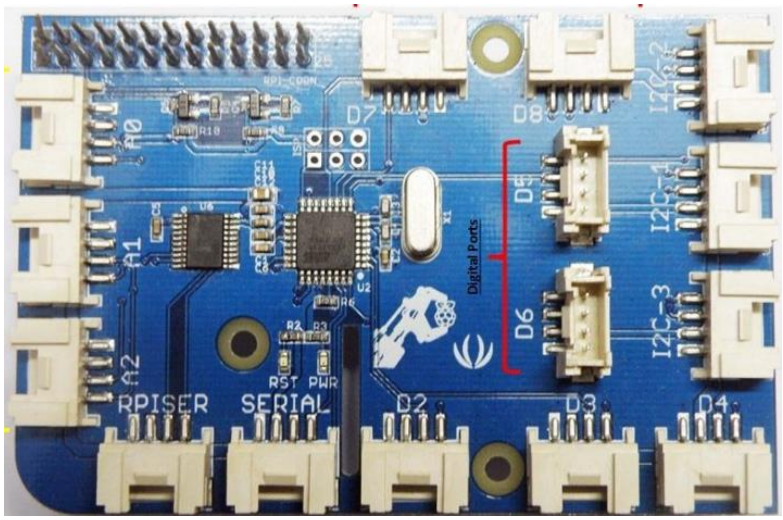
Buzzer:
Purpose: Provides audible alerts and alarms during emergency situations to ensure miners are alerted even if they can't see visual indicators.

RGB LCD Display:
Purpose: Displays real-time sensor data, system status, emergency alerts, and any other relevant information for miners.

Air Quality Sensor:
Purpose: Measures the air quality by detecting the presence of harmful gases, ensuring miners are not exposed to dangerous gas levels.
Relay (Optional for Automatic Lighting):
Purpose: Controls external devices like lights, enabling the system to automatically adjust lighting conditions based on the ambient light intensity.

Each sensor plays a crucial role in detecting potential hazards, communicating information to miners, and facilitating a prompt response during emergencies. Their collective integration forms a comprehensive system designed to enhance safety measures within coal mines.

Schematic diagram:



#Since time is not there we are not able to keep things properly instead we keep just kept which pin to keep where

LEDs (Red, Blue, Green):
Red LED: Connect to digital pin D5.
Blue LED: Connect to digital pin D6.
Green LED: Connect to digital pin D7.
Sound Sensor: Connect to analog pin A1.
Temperature and Humidity Sensor: Connect to digital pin D2.
Light Sensor: Connect to analog pin A0.
Button: Connect to digital pin D8.
Ultrasonic Sensor:
Connect the trigger pin to digital pin D3.
Connect the echo pin to digital pin D4.
Rotary Angle Sensor: Connect to analog pin A0.
Buzzer:
Connect to digital pin D4.
RGB LCD Display:
Connect the I2C SDA pin to the SDA pin on Grove Pi.
Connect the I2C SCL pin to the SCL pin on Grove Pi.

Air Quality Sensor:
Connect to analog pin A1.

Conclusion:

In the heart of coal mining's challenges, the "Coal Mine Evacuation System" emerges as a sentinel of safety. By fusing innovation and sensors, we've erected a fortress against danger, guiding miners through hazards with precision. This project isn't just about electronics; it's a testament to our commitment to human lives. As we close this chapter, the system's LEDs, alerts, and LCD fade, leaving a profound impact on safety standards. In the mine's depths, our creation echoes the harmonious blend of technology and empathy, underlining the future of safety protocols.

Code

```
import time
import smtplib
from grove.grove_led import GroveLed
from grove.grove_button import GroveButton
from grove.grove_ultrasonic_ranger import GroveUltrasonicRanger
from grove.grove_rotary_angle import GroveRotaryAngle
from grove.grove_buzzer import GroveBuzzer
from grove.grove_rgb_lcd import GroveRgbLcd
from grove.grove_sound_sensor import GroveSoundSensor
from grove.grove_temperature_humidity_sensor import GroveTemperatureHumiditySensor
from grove.grove_light_sensor_v1_2 import GroveLightSensor
from grove.grove_air_quality import GroveAirQuality
from grove.grove_relay import GroveRelay # Import the Grove Relay module

# Initialize sensors and actuators
red_led = GroveLed(5)
blue_led = GroveLed(6)
green_led = GroveLed(7)
button = GroveButton(8)
ultrasonic = GroveUltrasonicRanger(3)
rotary = GroveRotaryAngle(0)
buzzer = GroveBuzzer(4)
lcd = GroveRgbLcd(0x04, 2, 3)
sound_sensor = GroveSoundSensor(1)
temp_humidity_sensor = GroveTemperatureHumiditySensor(2)
light_sensor = GroveLightSensor(0)
air_quality_sensor = GroveAirQuality(1)
automatic_light_relay = GroveRelay(2) # Relay for automatic lighting

# Define constants
TEMPERATURE_THRESHOLD = 30
SOUND_THRESHOLD = 80
DISTANCE_THRESHOLD = 30
AIR_QUALITY_THRESHOLD = 200

# Email configuration (replace with your own details)
SMTP_SERVER = "smtp.example.com"
SMTP_PORT = 587
SMTP_USERNAME = "your_email@example.com"
SMTP_PASSWORD = "your_password"
RECIPIENT_EMAIL = "recipient@example.com"

# Define emergency kits location
EMERGENCY_KITS_LOCATION = "Near Exit B"

def display_message(message, color=(255, 255, 255)):
lcd.set_text_n_color(message, *color)

def log_event(event):
with open("event_log.txt", "a") as log_file:
log_file.write(f"{time.strftime('%Y-%m-%d %H:%M:%S')} - {event}\n")

def send_email(subject, message):
with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
server.starttls()
server.login(SMTP_USERNAME, SMTP_PASSWORD)
server.sendmail(SMTP_USERNAME, RECIPIENT_EMAIL, f"Subject: {subject}\n\n{message}")

def main():
```

```
try:
lcd.set_text("Coal Mine Evacuation System")
time.sleep(2)
lcd.set_text("Press the button to start")
# Wait for button press to activate
while not button.is_pressed():
pass
lcd.set_text("System Activated")
while True:
# Read sensor data
temperature, humidity = temp_humidity_sensor.read()
light_intensity = light_sensor.read()
sound_level = sound_sensor.read()
distance = ultrasonic.get_distance()
rotation_angle = rotary.angle()
air_quality = air_quality_sensor.read()
# Display sensor data on the LCD
lcd_text = f"Temp: {temperature:.2f}C Humidity: {humidity:.2f}%\n"
lcd_text += f"Light: {light_intensity} Sound: {sound_level}\n"
lcd_text += f"Distance: {distance}cm Angle: {rotation_angle}°\n"
lcd_text += f"Air Quality: {air_quality}\n"
lcd_text += f"Emergency Kits: {EMERGENCY_KITS_LOCATION}\n" # Display emergency kits location
# Control automatic lighting based on ambient light intensity
if light_intensity < 100: # Adjust threshold as needed
automatic_light_relay.on()
lcd_text += "Automatic Lighting: ON\n"
else:
automatic_light_relay.off()
lcd_text += "Automatic Lighting: OFF\n"
display_message(lcd_text, (0, 128, 0)) # Display in green
# Check for emergency conditions and activate indicators
emergency_triggered = False
if (
temperature > TEMPERATURE_THRESHOLD or
sound_level > SOUND_THRESHOLD or
distance < DISTANCE_THRESHOLD or
air_quality > AIR_QUALITY_THRESHOLD
):
red_led.on()
blue_led.off()
green_led.off()
buzzer.on()
lcd.set_text("Emergency! Evacuate!")
emergency_triggered = True
log_event("Emergency Triggered")
send_email("Coal Mine Emergency", "An emergency has been triggered in the coal mine.")
else:
red_led.off()
blue_led.off()
green_led.on()
buzzer.off()
# Display specific emergency messages
if emergency_triggered:
if temperature > TEMPERATURE_THRESHOLD:
lcd.set_text("High Temperature Detected")
elif sound_level > SOUND_THRESHOLD:
lcd.set_text("Loud Noise Detected")
elif distance < DISTANCE_THRESHOLD:
lcd.set_text("Obstacle Detected")
elif air_quality > AIR_QUALITY_THRESHOLD:
lcd.set_text("Low Air Quality")
# Wait before next iteration
time.sleep(1)
except KeyboardInterrupt:
# Cleanup and exit gracefully
lcd.set_text("System Stopped")
red_led.off()
blue_led.off()
green_led.off()
buzzer.off()
automatic_light_relay.off() # Turn off the automatic lighting relay
lcd.set_rgb(0, 0, 0) # Turn off LCD backlight

if __name__ == "__main__":
```

main()