Stone Crushing Plant Monitoring System using Stm32

Problem Statement

Stone crushers emit fine dust particles into the atmosphere, causing environmental pollution and health risks. Exposure over long periods causes respiratory problems like asthma and silicosis, and damages crops by decreasing yields. Effective dust control is necessary to prevent these effects.

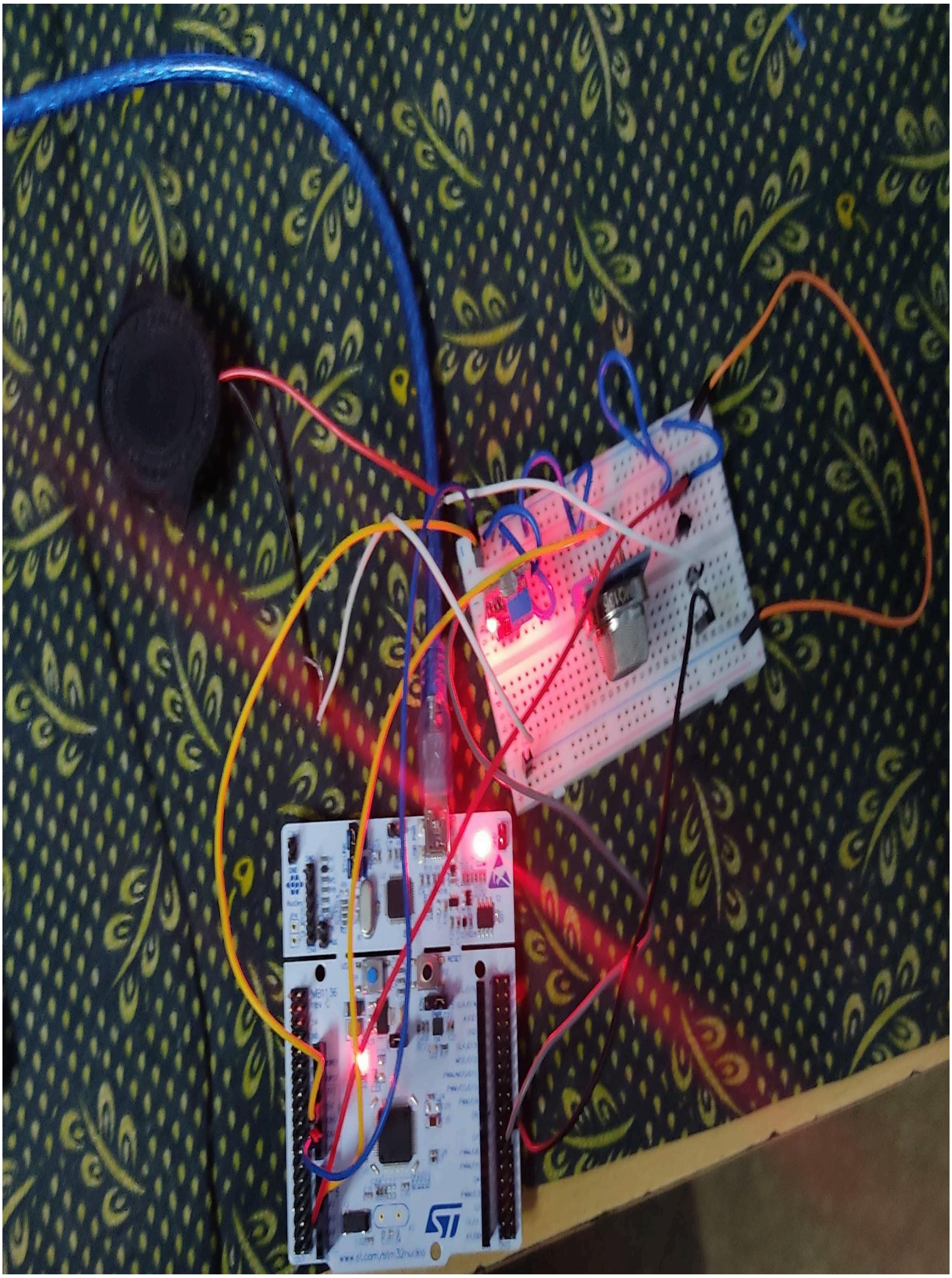
Solution:

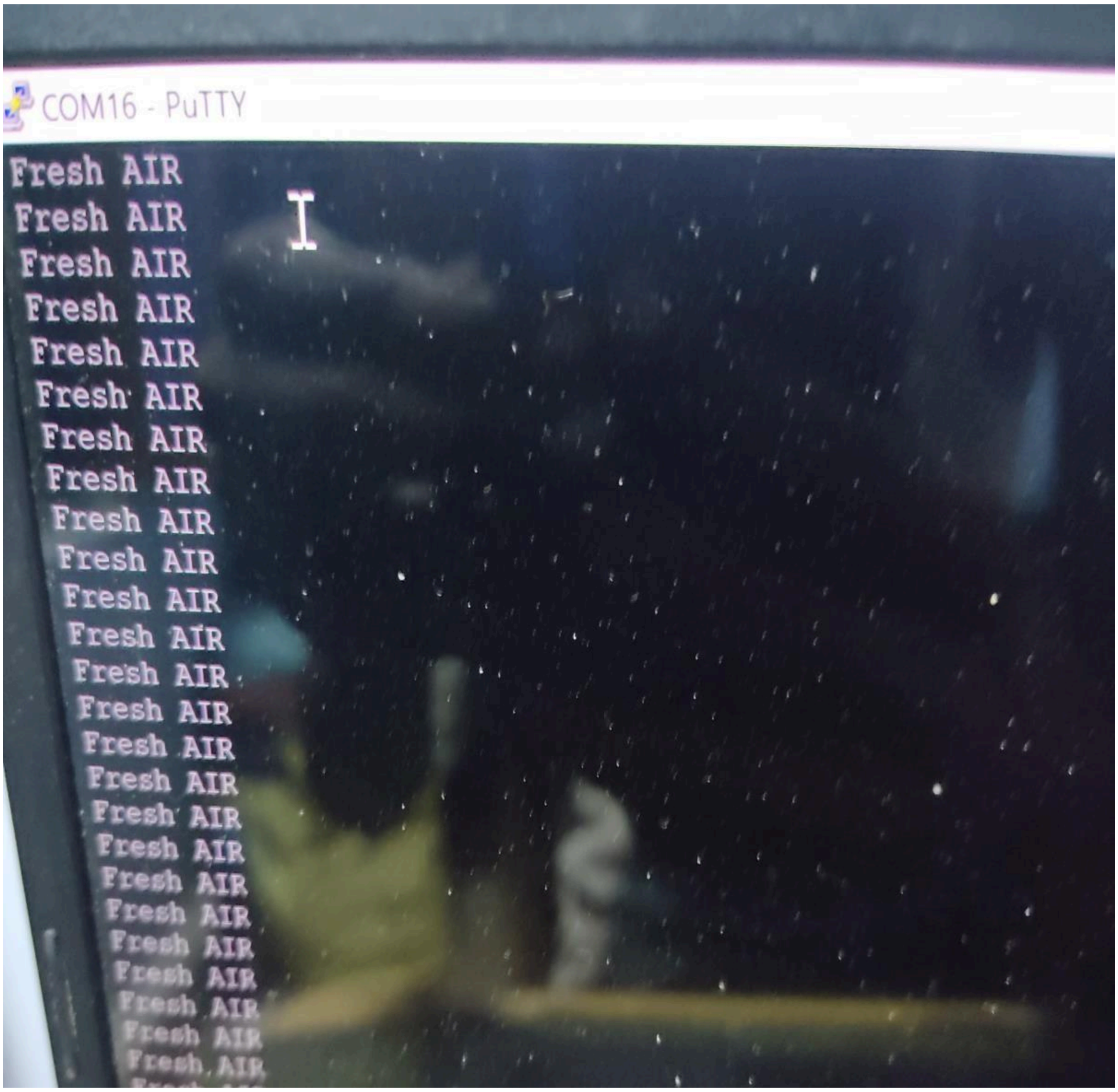
- We are designing an intelligent air pollution sensor for stone crushers with a NUCLEO microcontroller.
- It measures air quality, temperature, and light near the crusher using sensors.
- The information is transmitted to a computer via USART communication, and live data can be viewed with PuTTY software.
- A buzzer will sound if the pollution exceeds safe levels, alerting individuals in the nearby.
- This system makes it easy to monitor pollution and ensures the surroundings and workers are safer.

Conclusion:

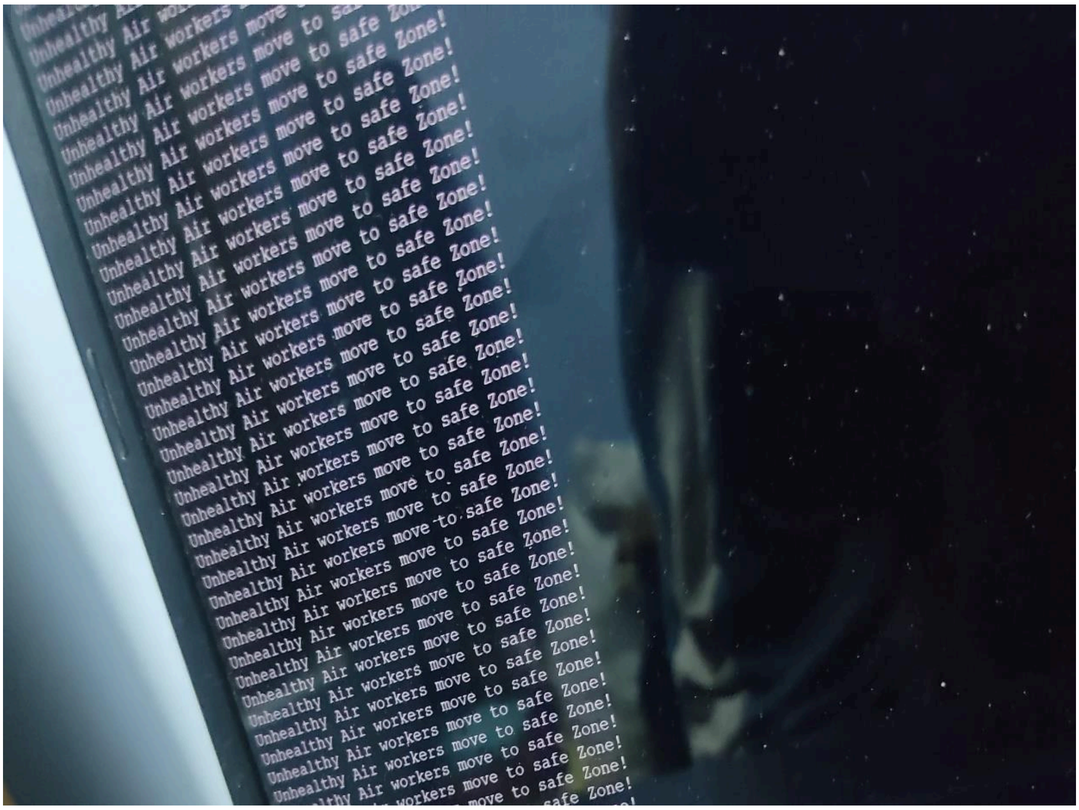
This intelligent monitoring system enhances pollution management at stone crusher sites by providing real-time data, immediate alerts, and accessible monitoring through USART communication. It serves as a practical solution to improve environmental conditions and worker safety in industrial environments prone to high dust generation.

OP:





```
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
```



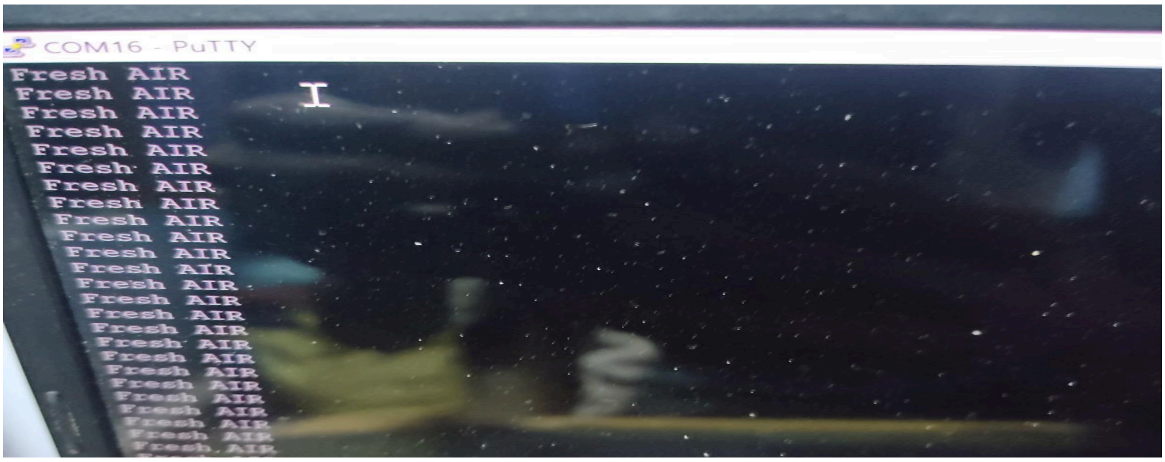
```
Unhealthy Air wor
Unhealthy Air workers move to sa
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
Unhealthy Air workers move to safe Zone!
lthy Air workers move to safe Zone!
```

Code:

```
#include<stdint.h>
#include<stm32f4xx.h>
#include<string.h>
#include<stdio.h>
#include<string.h>

void LDR_init() //digital ip based sensor

{

    RCC->AHB1ENR|= (1<<1); //GPIOB clk enable

    GPIOB->MODER&= ~(0xc); //Reset Pb1 00 set as ip
    mode GPIOB->MODER|= (0x00010); //Set Pb2 op mode

    GPIOB->PUPDR|= (0x8); //Set Pb1 acts as Pull down configuration avoid floating

}
void delay()

{

    for (uint32_t i=0; i<=70000;i++); //Empty loop

}
```

```
void MQ135_init() //Analog based sensor

{

        RCC->AHB1ENR|= (1<<0); //GPIOA Peripheral
        Enable RCC->APB2ENR|= (1<<8); //ADC peripheral
        Enable
        GPIOA->MODER|= (0xc0);//Set as analog mode 1100 = 12 at
        PA3 GPIOA->MODER|= (0x3); //set as pA2 temp
        ADC1->CR2|= (1<<0); //ADON
        enable ADC1->SQR3=3; //ADc chnl 3
        ADC1->SQR3=2; //ADc chnl 2 temp

        for(volatile i=0; i<=20000; i++)

        ADC1->CR2|= (1<<30); //Conversion Starts
        ADC1->CR2|= (1<<1); // contineous
        Conversion
}

void USART_init()

{

      RCC->APB1ENR|= (1<<17); //enable usart clk

      GPIOA->MODER|= (0xA0); //Enable PA2 &A3 1010 -A

      GPIOA->AFR[0]|= (0x700); //tx 0111 pA2 transmit the data

      for(volatile int i=0; i<=20000;i++);
      // Set baud rate to 9600 for 16 MHz clock -> BRR = 16000000 / 9600 = ~1666 =
      0x0682 USART2->BRR = 0x0682;
      USART2->CR1|= (1<<13); //Usart
      enable USART2->CR1|= (1<<3); //TX
      enable

}

void USART2_SendString(const char *str)

{

   while (*str)

   {



      while (!(USART2->SR & (1<<7))); // Wait until transmit data register is
      empty USART2->DR = (*str & 0xFF);
```

```
     str++;
  }

}



int main(void)



{

    LDR_init();
    MQ135_init();
    USART_init();


    char buffer[60];

    uint32_t adc_Value, temp;

    float voltage; //MV
    int temperature;
    while(1)

    {

        while(!(ADC1->SR & (1<<1))); // waiting for the conversion completing
        adc_Value = ADC1->DR;
        delay();
        while(!(ADC1->SR & (1<<1))); // waiting for the conversion
        completing temp = ADC1->DR;
        voltage = (temp * 3300) / 4095;
        temperature = voltage / 10.0;
        delay();

        if (((GPIOB->IDR & (1 << 1)) != 0) && (adc_Value >= 1100 || temperature > 40))

        {
            sprintf(buffer,"Unhealthy AIR Move To Safe Zone \r\n");
            for(volatile int i=0;i<=10000;i++);
            USART2_SendString(buffer);
      GPIOB->ODR|= (1<<2);

        }

        else

        {

            GPIOB->ODR&= ~(1<<2);
            sprintf(buffer,"Fresh AIR \r\n");
            for(volatile int i=0;i<=10000;i++);
            USART2_SendString(buffer);

        }

        delay();

}}
```



```
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
Fresh AIR
```