Indoor Exhaust System

By Sanika Dongre

and

Monish Nene

Test Plan:

Github repo link: https://github.com/monishnene/AESD_2

Project Description:

Our project is a simulation of an indoor exhaust system. We will use exhaust fans to remove harmful gases from the chamber and alert with a buzzer if the gases and/or temperature goes above a certain threshold.

Remote Node: Tiva TM4C1294XL

Control Node: Beagle bone green

The indoor exhaust system will be designed in a box that comprises of

Sensors:

1) Adafruit Temperature Si7021 Temperature and Humidity Sensor (I2C interface).
2) MQ7 gas sensor (detects CO value in ppm) sensor evaluation board that is used for detecting various gases in ppm level (ADC Interface).

Control outputs:

1) A set of four exhaust fans
2) A buzzer

The sensors will be placed in a box and we are planning to check it with an incense stick or a lighter. The count of exhaust fans on and their speed can be varied according to the feedback from the sensor.

Requirements:

1) State machine on the remote node to alert in case of emergency or failure of sensors, switch the fans on and off according to feedback (closed loop state machine).
2) LEDs on the remote node to indicate the current mode of operation.

Use cases:

1) Default: The value of temperature and the gas sensor is within the normal range and the exhaust fan is off
2) Smoke detected: Turn on the fans according to the level
3) High Temperature detected: Turn exhaust fans on based on the level
4) Failure of smoke sensor: Send alert with the buzzer
5) Failure of Temperature sensor: Send alert with the buzzer

System Configuration:

1) Tiva board: Remote Node
2) Beagle bone green: Control Node

3) The alerting mechanisms by the beagle bone green to indicate:

We will be using two buzzers – One buzzer for TIVA board and One for Beagle bone green to indicate the operational status of the system. The exhaust fans (actuators) will be connected on the TIVA board.

- Default/Normal: The value of temperature and the gas sensor is within the normal range and the exhaust fan is off.

- Degraded mode: Status LEDs on the TIVA board – LED 4 is blinking

- Failure mode: Status LEDs on the TIVA board – LED 4 is on

- Temperature and Humidity sensor fails – LED 2 on the TIVA Board blinks
- Gas sensor fails – LED 3 on the TIVA Board blinks
- Message queue is full – LED1 on the TIVA Board is ON

Failure:

- ➢ Smoke detected: Turn on the fans according to the set thresholds
- ➢ High Temperature detected: Turn exhaust fans on based on the set thresholds
- ➢ Failure of smoke sensor: Send alert with the buzzer
- ➢ Failure of Temperature sensor: Send alert with the buzzer

Control-Remote Node connectivity:

- ➢ UART for connecting the remote node (TIVA Board) to the control node (Beagle bone green)
- ➢ A message queue for communicating the control/sensor information for the transmission of the temperature and gas sensor data and logs.

System Functionality:

Automatic startup:

Control Node startup

Remote Node startup

Remote Node sensing:

State diagram:

1) 0 fan on/off
2) 1 fan on/off
3) 2 fan on/off
4) 3 fan on/off
5) 4 fan on/off
6) Buzzer

If the temperature and Humidity Sensor (Si7021) is disconnected, then the polling is done as it waits on acknowledgement for I2C and it stays in the degraded mode of operation and as soon as the sensor is connected back the data values is obtained.

If the Gas (MQ7) sensor is disconnected, then it continuously reads the ADC reading and by the time, the sensor remains disconnected the mode is in degraded mode of operation and once it is connected back, the sensor sends the reading.

The control algorithm – If the temperature goes above 20 degree Celsius then turn on one of the fan and if the temperature goes above 100 degrees – (alert condition): All the fans will be turned on and a buzzer will be turned on too. (Different threshold values will be set for different operations).

Failure and Fault detection Behavior:

If the temperature and Humidity Sensor (Si7021) is disconnected, then the polling is done as it waits on acknowledgement for I2C and it stays in the degraded mode of operation and as soon as the sensor is connected back the data values is obtained.

If the Gas (MQ7) sensor is disconnected, then it continuously reads the ADC reading and by the time, the sensor remains disconnected the mode is in degraded mode of operation and once it is connected back, the sensor sends the reading.

Testing based on threshold values for temperature and gas: by activating the exhaust fans and buzzers.

If the control node cannot detect the Remote node then the control node will keep trying to get connected until it can get connected to the remote node.

If the sensor fails or the temperature and gas sensor values are beyond a threshold then the fans will be turned on accordingly as a feedback mechanism.

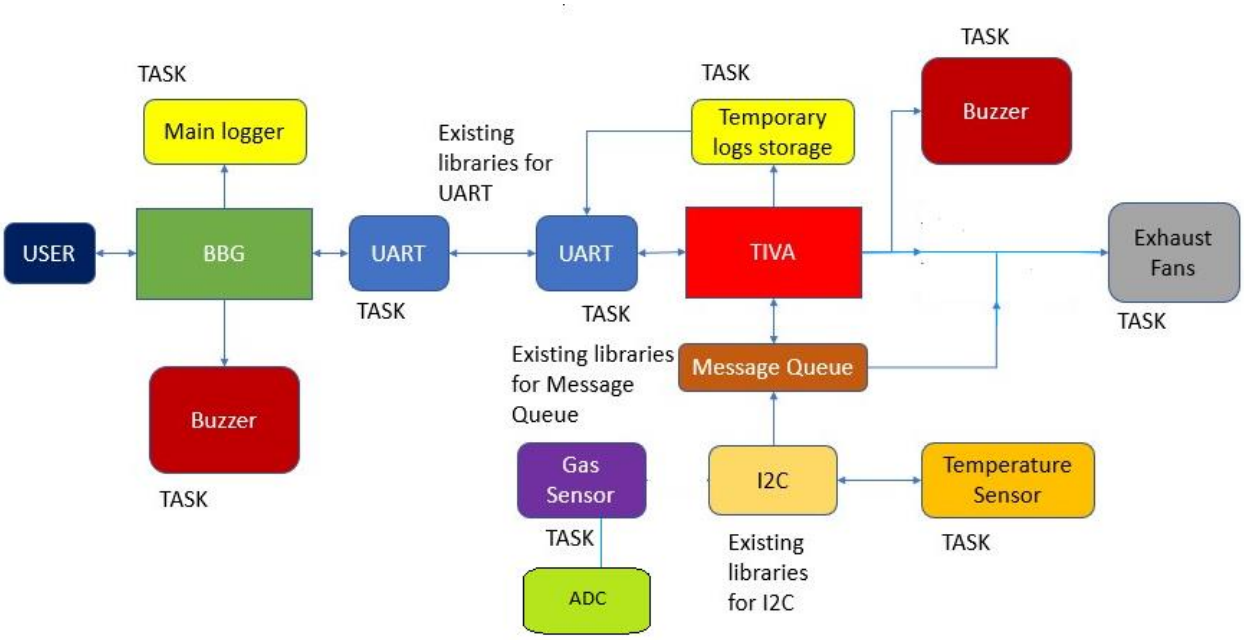
If the control node and the remote node connection fails then the buzzer on the control node side will turn on.

Logging:

The logging will be done on a logfile in the control node side. The data will be collected by the remote node and sent to the Beagle bone via UART. If the connection between the control node and remote node fails, then the real time data from the sensors will be saved on a logfile on remote node side and once the connection is retrieved then the temporary file will be sent to the beagle bone.

Architecture Description:

Software Diagram:



Closed loop Control:

1) 0 fan on/off
2) 1 fan on/off
3) 2 fan on/off
4) 3 fan on/off
5) 4 fan on/off
6) Buzzer on/off

Tasks for Remote Node:

1) Threshold task: To check the threshold values and accordingly turn on the buzzer and/or exhaust fans
2) Logger task: For collecting data and sending it over UART and back up data in case of connection failure
3) UART task: uart send and receive options
4) Temperature sensor task: To get the temperature values from the sensor using I2C interface
5) Gas sensor task: To get the gas readings from the sensor using I2C interface

Tasks for Control Node:

1) Threshold task: To turn on/off the buzzer and the fans based on threshold values
2) Logger task: To save the data in a log file
3) UART task: To send/receive data over UART
4) Gas Task: To get the CO value in ppm for the MQ7 Gas sensor.
5) Temperature and Humidity task: To get the readings from the temperature and the humidity (Si7021 sensor)

API Description:

Remote-Node:

1) void exit_handler(): To exit the handler
2) void uart_send(uint8_t* ptr, uint32_t size): To send the UART data
3) void uart_receive(uint8_t* ptr, uint32_t size): To receive the UART data
4) void queue_adder(queue_data_t* data_send): To add the data in the queue and send the message queue
5) void buzzer_control(void): To turn the buzzer on or off
6) void Fan_update(int8_t value): fans update based on the values – turn on or off
7) void i2c_init(void): To initialize the I2C
8) void gpio_init(void): To initialize the GPIO pins for the output devices
9) void uart_init(void): To initialize the uart and configure it
10) void UARTFxn(void* ptr): Task to send and receive data over UART from the remote node to the control node.
11) void loggerFxn(void* ptr): Task to log the data
12) void gasFxn(void* ptr): Task to send the gas data readings in ppm to the control node from the remote node
13) void thresholdFxn(void* ptr): Task to turn on/off the fans and the buzzer based on the set threshold values
14) void __error__(char* pcFilename, uint32_t ui32Line): Assert error() function FREERTOS

Control-Node:

fork() operation in main to separate the logger task and the GUI designed

1) void* logger(void* ptr): Logger function to log the data
2) void logfile_setup(void): To setup the logfile function
3) int32_t timer_init(void): To initialize the timer – periodic timer for 10 seconds
4) void system_end(int sig): To change the condition for system_end
5) void uart_init(void): uart initialization
6) void termios_init(void): To initialize the terminal for saving the current port settings and setting the conditions for uart using proper flags

.

Project Plan:

Using Excel Gantt:

AESD_PROJECT_2