1. ***Describe what the problem this project or part of a team project addresses***

In factories, especially in plants which deal with fumes like petroleum and pharmaceutical plants, safety of the plant and the people in it is heavily dependent on good smoke and gas leak detection. Situations arise where an undetected gas leak in one part of the plant can cause a fire outbreak over the entire plant.

For example, Gas leak explosion that killed 23 at ChemChina Plant.(link: <https://www.insurancejournal.com/news/international/2018/12/03/510793.htm>)

Another example is, Explosion and fire caused due to gas leakage from high-pressure ethylene piping at an ethanol manufacturing plant (link: <http://www.shippai.org/fkd/en/cfen/CC1200098.html>).

The main problems are the following:

1. Lack of sensors to detect the leakage.
2. Lack of good communication to alert the entire plant in time.
3. No – centralized decision making center to handle the situation.
4. Steps taken to evacuate personnel for example, unlocking factory doors which might be locked otherwise.

This project addresses the issue of establishing a suitable communication method within the plant and a centralized decision making node.

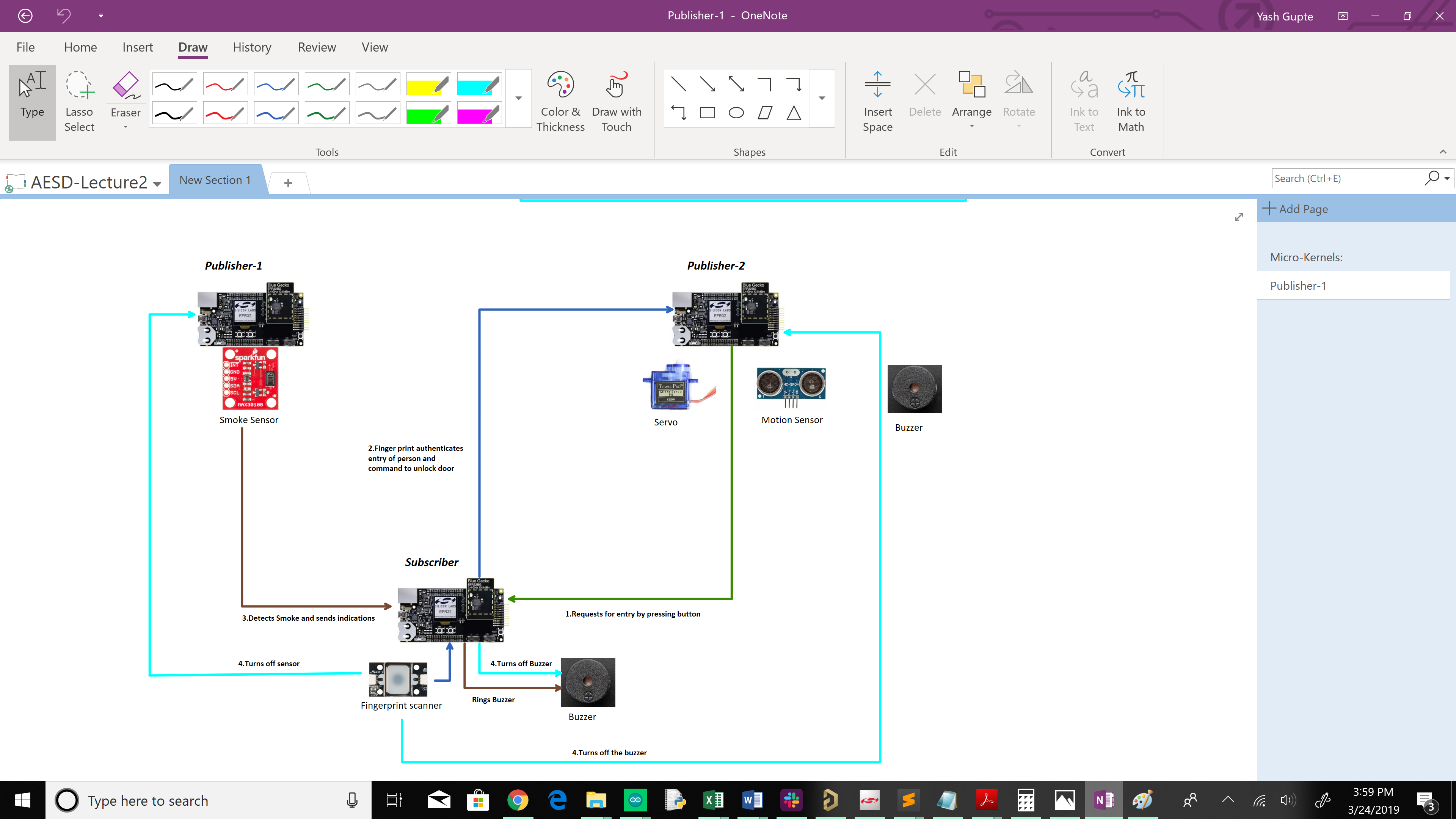
1. ***How does this project alleviate or solve the problem?***

This part of the project is part of Subscriber of the main project. The roles of this part are:

* + 1. Allow the entry of a person into the plant when a request is detected by sending command to Publisher-2.
    2. Have a centralized decision to make when the smoke signal is detected. This allows the turning off the Buzzers and re-enabling the locking of doors.

This way a centralized system is in place to control the response to what happens in the plant.

1. ***Functional block diagram of the individual project***



**Individual Portion in blue box**

I will be working on the part included in the blue box above. It includes:

* 1. Configuring the node as friend.
  2. Commutation’s with other 2 nodes.
  3. Interfacing the finger print sensor.
  4. Interfacing the buzzer.

***3b. Additional Details***

Fingerprint Sensor (UART) Pins: P30(Tx), P31(Rx)

Buzzer (digital) Pin s:P10

Fingerprint sensor

Fingerprint sensor is UART based and USART-2 is used. This is because USART0 is being used for ‘LOG\_INFO’ functions. USART1 is connected to buttons which need to be used in this project.

The sensor has documented steps for storing finger prints and then checking the database of 3000 samples to check for the samples.

Custom APIs were written to implement the storing of new finger prints, validation etc.

Functions Used with Fingerprint sensor – Custom Made

1. **FingerPrintInit():** Initializes the sensor with baud of 9600 baud. Assigns pins P30 and P32 which are UART2 pins turns on LED to check if sensor receives commands.
2. **void CommandSend(uint8\_t param,uint8\_t cmd\_code):**

@**brief** :Sends an array of 12 bytes to the senor.

@**param** :It is the parameter for the respective command.

@**cmd\_code**: This is the hex value of the command to be sent.

1. **uint32\_t CommandResponse(uint8\_t param):**

@**brief** : Receives the value from the sensor and checks if valid. A value of 0x30 is success and 0x31 indicates failure.

@**param:** Performs operations for separate params received.

1. **void** **CheckFingerPrint**(**void**):

**@brief:** Checks if the placed finger print is valid or not.

**@Functions called:**

* + 1. Turns on LED for taking the readings.
    2. Checks if finger has been placed in the sensor.
    3. Waits up-to 5s for finger to be placed.
    4. If valid, turns off the led and calls appropriate function.
    5. If not, returns an error “invalid ID” to be displayed on the log. The function can be called again to restart this operation.

5. **void** **AddFingerPrint**(**void**):

@brief: Adds a new finger print to the data-base.

@functions called:

* + 1. Turns on LED for taking the readings.
    2. Checks if finger has been placed in the sensor.
    3. Waits for finger to be placed.
    4. Checks the current enroll count and adds 1 to increment it to store a new id.
    5. Captures image of finger.
    6. Starts enrollment-1.
    7. Checks of finger is placed. And asks user to lift finger.
    8. Waits for user to place finger again.
    9. Captures finger.
    10. Starts enrollment-2.
    11. Checks of finger is placed. And asks user to lift finger.
    12. Waits for user to place finger again.
    13. Captures finger.
    14. Starts enrollment-2.
    15. Displays if finger was successfully added.

Buzzer

The buzzer is a piezo sensor. It is run at a frequency of approximately,300Hz. A function which sets and clears a GPIO at 300Hz is used. A custom blocking wait function was written for this purpose. The function used is blocking because it needs the system to not do anything else at that instant. When an interrupt is received, the function is exited.

1. ***Bluetooth Commands***

***[Attached excel sheet]***

1. ***List of sensors for this project***
   * 1. Finger Print Scanner (GT-521F32) <https://www.sparkfun.com/products/14518>
     2. Buzzer
2. ***What exposed services and client profiles will be implemented?***

Exposed services will be:

* + 1. state PB1 and PB0.

The node will be listening to:

* + 1. Indications sent from Node with smoke sensor.

1. ***Persistent Data Used***

The last state of each button pressed will be stored in flash memory. This can be helpful to determine what the last state of the button was before reset. If a command was in progress, it can be sent again based on the state.

1. ***Final development schedule for this project broken down to discrete components including target implementation dates.***

***Link:*** [***https://drive.google.com/open?id=1HzVqCI9Xvxpw5bhBRqhw\_fRMOPwq48\_g***](https://drive.google.com/open?id=1HzVqCI9Xvxpw5bhBRqhw_fRMOPwq48_g)

1. ***Configuration of Friend Node***



Fig 1.Node On

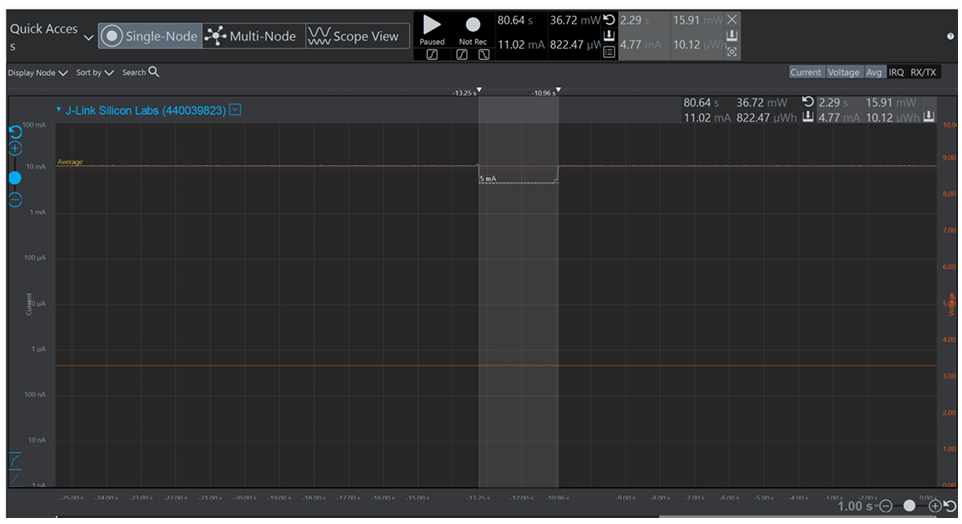


Fig 2. Finger-print sensor On.

Components of node:

1. Fingerprint Sensor
2. Buzzer
3. LED

Working of Node:

1. On receiving a request from LPN-1, an LED turns on. In a response, the user can press PB0,

Followed by a finger-print validation to unicast a "Door Open" message to LPN-1.

1. This node also receives "Smoke Alert" from LPN-2 when smoke crosses threshold on LPN-2.
2. On pressing PB1, it publishes an "Override" signal to all nodes.

Design Considerations and justifications:

1. This node uses a finger-print sensor for validation of finger prints. The sensor is turned On only when either PB0 or PB1 is activated. This prevents a continuous polling of the sensor for read values. As seen in the figure 2 above, the sensor requires about 5mA of current when checking for finger-prints. Over let's say 20s, it would consume (5mA X 20) = 100mA more!
2. The finger-print sensor utilizes commands which are sent over UART. The commands used are mentioned in the Datasheet found here : <https://cdn.sparkfun.com/assets/learn_tutorials/7/2/3/GT-521F52_Programming_guide_V10_20161001.pdf>. However, the API and design of the communication process is to be developed individually.
3. The sensor utilizes functions to Add fingerprints, check valid finger prints etc. The UART used is USART2. Pins P30 and P32 are used as Tx and Rx for communication at a baud of 9600. Between communications, a write command is sent and a read command is immediately received. Example,

CommandSend(1,CMD\_CMOS\_LED);

CommandResponse(CMD\_CMOS\_LED);

1. Here, practically the time between execution of these 2 commands is one clock cycle. Which is approximately (4/32765)s. There were 2 possible alternatives to this approach.

**State Machine** - A

state machine could have been implemented to jump from command to command. However, for an operation like Checking a fingerprint, which utilizes about 6 different pairs of commands, it would take 6x2 = 12 , separate states to implement. Increasing the complexity and time of operation to complete the operation.

**Interrupt based** - This approach would have helped reduce the current consumption to a certain extent. For example, the dip in the fig2 above would be replaced by smaller peaks over the same duration. However, it would increase complexity in code such that, there would be a UART interrupt after each transfer and receive. According to the data-sheet, <https://www.silabs.com/documents/public/reference-manuals/efr32xg13-rm.pdf> Page 620. "RX Buffer Full Interrupt Flag" and "TX Complete Interrupt Flag" indicate transmit complete and receive complete respectively.

To send any command, 12 separate bytes need to be sent. The will trigger 12 separate interrupts for transmit alone. The same would repeat for receive. This would sum up-to 24 times interrupt would be triggered for just 1 pair of commands. In case of Checking for a finger-print, there are 12 separate states. This would result in 12x24 = 288 times that the ISR would be invoked.

Hence, my method of implementation is used here because, firstly the node is a friend node with no hard need for low power. Secondly, the sensor starts to read values from the sensor only when, a button interrupt is received this prevents the sensor from utilizing the power off the processor at all times. Moreover, the window to place the finger is only 2s and even faster if the finger is placed as soon as the button is pressed.

1. ***Lessons learnt:***
2. Using a UART based Fingerprint sensor.
3. Configuration of server commands and events.
4. Finding the app-key for unicast purposes.
5. Using PRAGMA in code to temporarily lower the optimization.
6. Utilizing flash commands for use of persistent data.
7. Blue LED mentioned is replaced with a Green one.
8. Github Link Project: <https://github.com/CU-ECEN-5823/course-project-yash1595>
9. GitHub Link Provisioner: <https://github.com/yash1595/ecen5823-provisioner-update>
10. Command Table link: <https://drive.google.com/open?id=1-SFyK7YWxBniqz34lYFpCNsJd_xLTbrs>