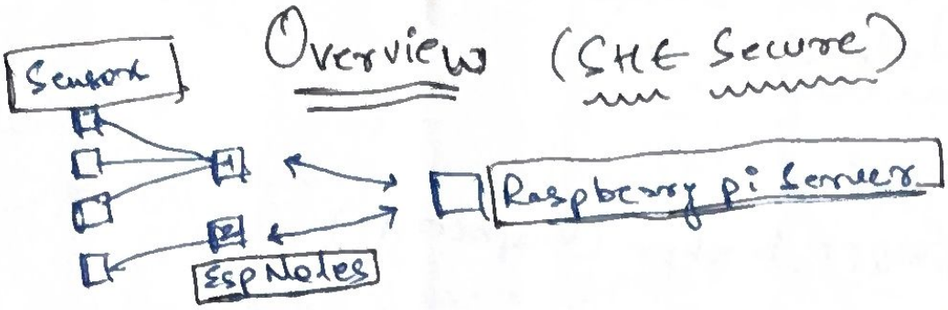
Overview (CHE Secure)



- MQTT Server → Raspberry Pi (Laptop) → Hosts MQTT (Mosquitto), Flask Web Server, SQL Lite Database.

Publishers + Subscribers

- Esp32 Node
  - ① → Alcohol Sensor + Panic Button + RFID Reader
    → Master Alert System. (LED, Buzzer, LCD)
  - ② → Noise Detection. (Microphone Sensor)

Server Setup (Raspberry Pi)

① MQTT Broker (Mosquitto)

② Flask API → Web browser
① Monitor incoming alerts
② Show current live system status.

③ SQLite DB

	MQTT Topics	Publisher	Subscriber	Purpose
①	/chesecure/rfid	ESP32 Node 1	Server	Publish student ID
②	/chesecure/panic	ESP32 Node 1	Server	publish panic event
③	/chesecure/alcohol	ESP32 Node 1	server	publish alcohol event
④	/chesecure/noise	ESP32 Node 2	server	publish noise event
⑤	/chesecure/alerts	Server	ESP32 Node 1 & 2	publish master alert flag.
⑥	/chesecure/custom_lcd	Server	ESP32 Node 1	publish custom LCD message.

(Explained later)
Since, we have to not trigger buzzer when alcohol detected but buzzer goes off whenever other alert, so we will make payload to alert topics

Esp32 Node I

① Connect to Wifi
② Connect to MQTT broker (Raspberry Pi)
③ Reads:
   ① RFID Scanner → Publisher studentID to /shescure/rfid
   ② Panic Button → Publisher to /shescure/panic
   ③ Alcohol Sensor → Publisher alcohol presence to /shescure/alcohol
④ Subscribes to /shescure/alerts
   On Recieving Alerts => Trigger Master Alert System
⑤ Display (16×2 LCD)
   => Student RFID Scanned
   => Panic button Pressed.
   => Alcohol Detected (5 sec)
   => High Noise Detected.
   => Normal Mode (System Secure)

* LCD Custom Message through /shescure/custom_lcd

	Publisher	Subscribers	Purpose
* /shescure/custom_lcd	Flask Server	Esp32 Node I	Customs Message

* ~~To be used~~ It can be used by the authorities to deliver message to the people inside the bus.

* If Server sends a message → LCD shows it.
  If Server sends "CLEAR" → LCD goes back to normal bus status.

ode-2

connect to Wifi

=> Connect to MQTT broker (Raspberry Pi)

=> Reads : ① Microphone sensor publish to (/classroom/noise)

=> Subscribes to (/classroom/alerts)

On Alert, blink External LED.

---

* The Server has other two main responsibilities, maintaining database.

i.e When a student scans RFID, the server has to verify the student from the database and then either authorise or disallow.

For this a separate table (specially from logs) will be created in the database. That will maintain pre entered database of students and will verify.

* Database of the students should be manually Entered into the database by the maintainer. (Commands further)

---

* Since, there are different types of Alerts like buzzer and no buzzer, so will also specify payload.

payload is just specifying the type of alert in the message. sent by the server

Then ESP32 Node will takecare about the Alert.

① Install and Update packages (MQTT, SQL, Flask,

② Project folder

```
she-score-server/
├── server.py        Main Server (MQTT + Flask)
├── database.db      SQLite
└── templates/
    │   └── dashboard.html   Webpage.
    └── static/
        └── style.css   Styles.
```

③ server.py

→ Connects to MQTT broker
→ Subscribes to sensor topics.
→ Logs data into database db
→ Publishes master alerts
→ Publishes custom LCD messages
→ Hosts flask dashboard.

Flask

→ web framework written in python. (Backend)
→ mainly used for building web applications — website, API, dashboards etc.

Why we choose flask ??
① Simple and Minimal by default.
perfect for IOT applications.
② Works easily with other Python and web technologies (WSGI compliant)
↳ web server standard for python.

Mosquitto

=> Message broker implements MQTT.
=> Lightweight, publisher-subscriber messaging protocol designed for low-bandwidth, high-latency or unreliable networks (IOT)

=> You don't write code in Mosquitto, but you send/receive messages through it.

:ribe to a topic (listen for messages)

:osquitto_sub -h localhost -t "topic-name"

Publish a message to that topic (send a message)

mosquitto_pub -h localhost -t "topic-name" -m "message".

---

install/upgrade

sudo apt update

sudo apt upgrade -y

sudo apt install -y mosquitto mosquitto-clients -y

sudo systemctl enable mosquitto

sudo apt install -y python3 python3-pip

pip3 install paho-mqtt flask flask-sqlalchemy

sudo apt install -y sqlite3

---

server code

① Define MQTT ← broker = localhost
← Port
← Topics

② Setup Flask app

③ Setup database

④ Handel MQTT — connect and reciever messages

⑤ Setup MQTT Client.

("____", 0)

↑

QOS

(Quality of Service)

0 → At most once

1 → At least once

2 → Exactly once

---

0 → fire and forget

→ no ack

→ message dilivered once or not at all

→ Very fast but no guarantee.

---

1 → Guarantees message arrival, but multiple arrivals

Entries in the Database :-

① Sqlite3 database.db

② INSERT INTO authorized_students (uid, name, gender)
VALUES ('23UCC611', 'Vedant Baldwa', 'Male');
'Female'

③ .exit

Once the server is running Two ip.

① http://127.0.0.1:5000
② http://192.168.186.128:5000

① → local host, will only work on same machine.
② → Local IP, for any device in same WiFi or network.

Esp32 Node 1

RFID Reader	SPI
Panic button	Digital GPIO input
Alcohol Sensor	Analog pin
Buzzer	GPIO Digital output
LED	GPIO Digital output
16x2 LCD	I2C

buzzer	26
Led	27
Panic-btn	25
Alcohol-pin	34
RST_PIN	22
SS_PIN	21

Libraries

PubSubClient (for MQTT)
SPI.h
Wire.h
LiquidCrystal_I2C.h (for 16x2 LCD)
MFRC522.h (for RFID)

Structure

Setup

MQTT Setup

- RFID Reading → Publishing UID to /shescure/rfid
- Panic button reading → Publishing alerts to /shescure/panic
- Alcohol sensing → Publishing alcohol detection to /shescure/alcohol
- Master Alerts → Listen to /shescure/alerts
- LCD Subscribe → Listen to /shescure/display_name and /shescure/custom_lcd
- buzzer & LED → Based on alerts
- Ideal Setup

Installing Libraries

PubSubClient
LiquidCrystal_I2C
MFRC522

WIFI_SSID ✓
WIFI_PASS ✓
RASPBERRY_PI_IP ✓
↓
without PORT number.
192.168.186.128

Network

Problem: If we use NAT (network address Translation) in server, ESP32 will not directly connect. We have to switch to Bridged mode.

NAT Mode → Raspberry Pi will share laptop's internet
will get a virtual IP
& Outdevices cannot directly see the rasp Pi

Bridged Mode → Your Raspberry Pi VM a real device on network
→ Actual IP address.
→ Discoverable.

Preferences → esp32 json link

Tools → Board → Board Manager → ESP32 by Espressif System.

Include pubsubclient folder in libraries of Arduino ide.

P.bsubclient by Nick O'Leary.

LiquidCrystal by Frank de Brabander

MFRC522 . Github community.

Esp32 NODE 2

Microphone sensor (Sound Detection)

Publish noise Alert

Subscribe to Master Alert

Since there are no requirement of showing anything, so this is just a basic model. further improvements can be done

---

Microphone Sensor → AnalogPin.

LCD → Blink On Master Alert

Change Alcohol / Noise threshold value accordingly.
(1800) (2000-3000)

Mention Auto Timeout System (LCD Buzzer Timeout)

Future Modification

(i) Improving Noise Detection.

(ii) simple notification module (via email or telegram)

(iii) counting number of Man/women from RFID. (Already added)