\*\*(SHOW GITHUB PROFILE)\*\*

- “Hey there, welcome to my YouTube channel!

My name is Phuc, and today I’m super excited to show you a simple project that I’ve been working on.

It’s simple but pretty cool because it integrates some awesome technologies: a microcontroller, a cloud server, and a Qt application.

If you’re into IoT, cloud computing, or slick car dashboards, you’re definitely in the right place!”

\*\*(SHOW SHORT VIDEO INTRO)\*\*

\*\*(SHOW IMAGE OF THE ENTIRE PROJECT ON GITHUB)\*\*

- “This project is all about gathering real-time environmental data and then visualizing it in a car dashboard interface.

Well, let’s dive in!”

### \*\*Microcontroller\*\*

\*\*(SHOW IMAGE OF ESP32 MICROCONTROLLER)\*\*

- “Alright, so the first part of the project uses an ESP32 microcontroller, a DHT11 sensor, and an OLED display.

This trio is like the dream team of small, low-power IoT devices, making it perfect for projects where efficiency is key.

Let me break down more detail.”

\*\*(SHOW IMAGE OF DHT11 SENSOR)\*\*

- “The DHT11 sensor is responsible for measuring temperature and humidity, and I’m using the \*\*UART (Universal Asynchronous Receiver-Transmitter) protocol\*\* to communicate with it.

Specifically, I’ve connected the sensor to \*\*PORT D4\*\* on the ESP32.

UART is a serial communication protocol that transmits data one bit at a time without the need for a clock signal, which makes it simple and reliable for transmitting basic sensor data.

It’s like the good old walkie-talkies—one device talks, and the other listens!”

- “The ESP32 is constantly ‘listening’ for data coming from the DHT11, and once it receives the temperature and humidity readings, it processes the data.”

\*\*(SHOW IMAGE OF THE OLED DISPLAY)\*\*

- “Now, once the ESP32 has gathered the data, we need to display it.

That’s where our \*\*OLED display\*\* comes in.

I’m using the \*\*I2C (Inter-Integrated Circuit) protocol\*\*, which is an efficient, two-wire communication method.

I’ve wired the OLED to the ESP32 using the \*\*SCL\*\* and \*\*SDA\*\* pins.

The beauty of I2C is that it allows multiple devices to connect to the same bus, making it incredibly scalable.”

- “Unlike UART, which is one-to-one, I2C allows multiple devices to communicate with a single master device—in this case, the ESP32.

It’s like a group chat where everyone gets the same message without any confusion.”

- “After receiving the data from the DHT11 via UART, the ESP32 sends the information to the OLED display using I2C.

And just like that—bam! You can see the current temperature and humidity displayed on the screen. Pretty neat, right?”

### \*\*Code Explanation\*\*

\*\*(SHOW ARDUINO CODE ON SCREEN)\*\*

- “Using Arduino, I wrote some code that reads the sensor data every few seconds and then sends it to a cloud server via an API.”

### \*\*Humorous Aside\*\*

- “Maybe in a future version, my microcontroller will be able to brew coffee or even do some housework—how awesome would that be?

But for now, I’ll settle for real-time data updates, and honestly, that’s pretty cool already!”

### \*\*Cloud Server & APIs\*\*

\*\*(SHOW SERVER CODE ON SCREEN)\*\*

- “Now let’s move on to the second part: the cloud server.

This is where things get a little more technical, but I’ve kept it simple.

I built the server using \*\*PHP 7\*\* with \*\*MariaDB\*\* as the database.

The server exposes two APIs: one for the ESP32 to post data, and another for the Qt app to fetch real-time data.”

- “Here’s the code that handles the POST request from the ESP32.

Once the ESP32 sends the data, the server processes it and stores it in our MariaDB database.”

\*\*(SHOW DATABASE)\*\*

- “Here’s a quick look at the database where all the data is stored.”

\*\*(SHOW API LINKS)\*\*

- “Let’s test the APIs. First, we’ll run the POST link to send data from the ESP32 to the cloud.

And now, we’ll run the GET link to retrieve the latest data for the Qt app.”

### \*\*Qt Application\*\*

\*\*(SHOW QT IDE)\*\*

- “And finally, let’s talk about the user interface: the car dashboard application! I built this interface using \*\*Qt and QML\*\*.

Honestly, when it comes to creating sleek, responsive UIs, QML is the way to go. Plus, it makes integrating real-time data a breeze.”

\*\*(RUN AND DISPLAY THE DASHBOARD)\*\*

- “Here’s the dashboard in action!

It displays all the essential information—speed, fuel, temperature, humidity, and even the current time, all updated in real-time from the cloud server.

Whether you’re driving or just checking the data, this dashboard keeps you updated on everything.”

- \*\*(SHOW A FEATURE YOU ADDED)\*\*

- “I even added a little extra: a small robot that moves around on the dashboard, purely controlled with QML properties.

Just goes to show how powerful QML is, right?”

### \*\*Real-Time Data Updates\*\*

\*\*(SHOW THE DASHBOARD UPDATING IN REAL TIME)\*\*

- “Let’s see it in action!

Watch as the app fetches the latest data from the cloud server and updates the dashboard in real-time. That’s real-time data visualization at work!”

### \*\*Conclusion\*\*

\*\*(SHOW YOURSELF ON SCREEN AGAIN)\*\*

- “So, that’s the project in a nutshell—a seamless integration of an ESP32 microcontroller, a cloud server, and a Qt car dashboard.

It’s a simple example of how different technologies can come together to create a real-time system that’s not only functional but also fun to build.”

- “If you have any questions or want to learn more about any part of the project, feel free to drop a comment below.”

- “And if you enjoyed this video or found it helpful, don’t forget to hit the like button and subscribe for more content like this.

Thanks for watching, and see you in the next one!”