To obtain realistic sensor data, the smart parking system may be tested in a real-world environment. For this purpose, a small-scale test was conducted using my vehicle parked on a private driveway. This setting provides a controlled, accessible environment to simulate the conditions of a typical parking space, thereby allowing a fair assessment of system performance and monitoring.

The goal of this test is to evaluate the **accuracy, reliability, and range** of the ultrasonic sensor in detecting vehicle presence, as well as to monitor the **stability and effectiveness of MQTT communication** over Wi-Fi, while the device is running outside. By integrating the Arduino UNO R4 WiFi with the ultrasonic sensor, the system publishes vehicle status (parked or unparked) to Hive’s MQTT broker. The data is then monitored using an MQTT client such as MQTT-Explorer to subscribe to messages.

This test environment serves as a proof of concept, validating the system's functionality before scaling to a more complex, larger-scale parking environment. Additionally, it provides insight for identifying any sensor blind spots, delays in data transmission, or connection interruptions—factors critical to real-world deployment in larger car parks.

### **11.1 Testing Environment Setup**

The driveway location used for testing is located at my student accommodation; this is more convenient to use as I park my vehicle here all the time making testing easier for connectivity reasons and time management. The car used was my 2014 Hyundai i20 which is a relatively small car allowing me to reverse a good amount for good readings of the sensor. The device is placed near the front of the driveway where it will be classed as parked when in front of the sensor.

### **11.2 Hardware Configuration**

To achieve the necessary hardware configuration to enable my smart parking system for testing make sure the Wi-Fi connection was still present when using the device outside of my house on the driveway. This will allow a somewhat decent connection while still being connected to my MQTT Broker, so the messages can be published and subscribed to by the MQTT client. To use the device outside a power supply will need to be used as a USB cable is unusable at this point.

The main power source was a 9V battery supplied by Solent University, which is connected using a battery clip. 9V is perfect as the Arduino R4 Wi-Fi allows between 6-24W of external power so no loss of power or hiccups will show effect. When the battery clip and battery are connected it will keep the device powered on and connected to Wi-Fi even though it's not connected via USB because the ESP32-S3 uses its connection to the Wi-Fi independently. (Arduino 2025)

Finally, the sensor placement needed to be accurate enough to gather good sensor data, so to achieve this the placement of the device needed to be adjusted a few times. In the first instance of testing, I placed the device on the driveway and tested if it would detect my vehicle up close, however in this case it wouldn’t even detect the air dam, so I decided to use some spare boxes I had to prop the device up to be level with the front bumper which achieved an accurate reading when parked and not parked.

  
Figure 25

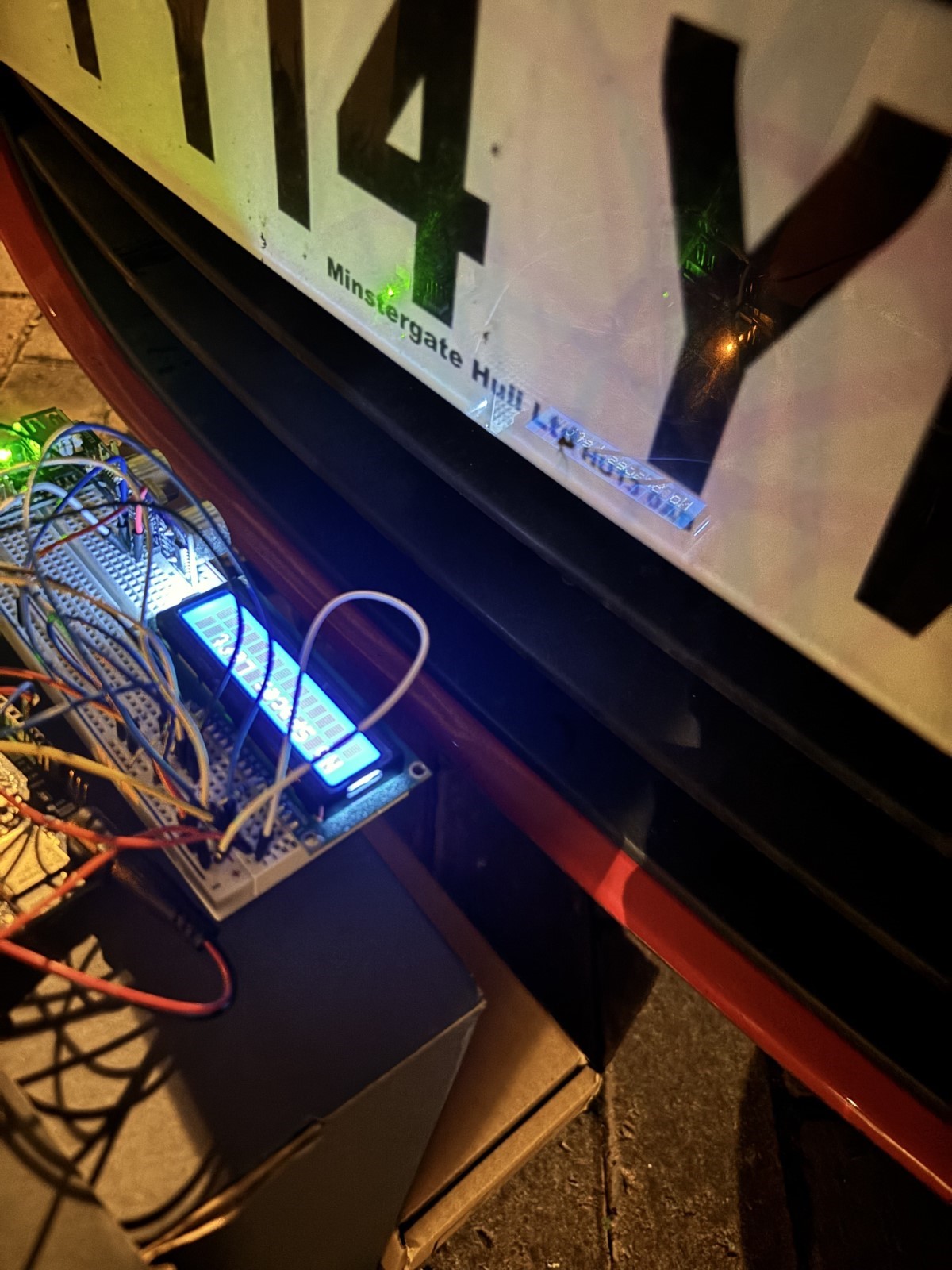


Figure 26

### **11.3 Sensor Accuracy and Range Testing**

The Ultrasonic sensor measured the data as accurately as it should, as I couldn't look at both screens at the same time, to monitor incoming messages, I would have to view my laptop screen inside to see if it was detecting accurately or not, then back outside which was very inefficient. However, the MQTT client, MQTT-Explorer timestamps every message published so this gives a good insight into how accurately and quickly the ultrasonic sensor responds when the vehicle is parked and not parked.

### **11.4 Real-Time Data Monitoring**

To monitor sensor data using the smart parking device, I decided to use a 16 x 2 LCD which is used to show real-time updates from the codebase of the Arduino where it displays ‘Space Available’ or No Spaces Left’ while showing the integers ‘1’ meaning its occupied or ‘0’ meaning its ‘taken’. This data being sent to the LCD screen is different from the data being sent to the serial monitor as its codebase only shows if the car is parked or not. In the serial monitor however, it shows the distance in CM whenever an object is less than 10cm near the sensor or more than 10cm, so whenever the distance is changed it shows ‘Distance changed’ in the serial monitor to alert the user as shown in the figure below, on the other hand, the LCD is converted to only show the user when the vehicle is parked or not.

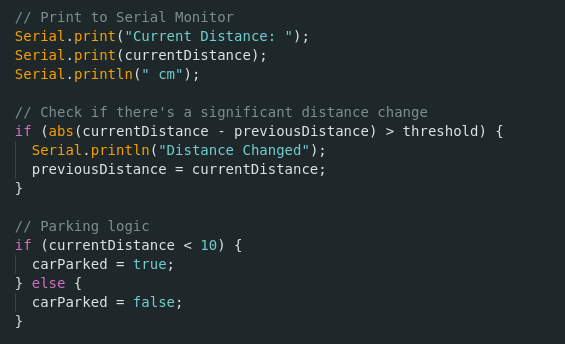


Figure 27

### **12. MQTT Connectivity Testing**

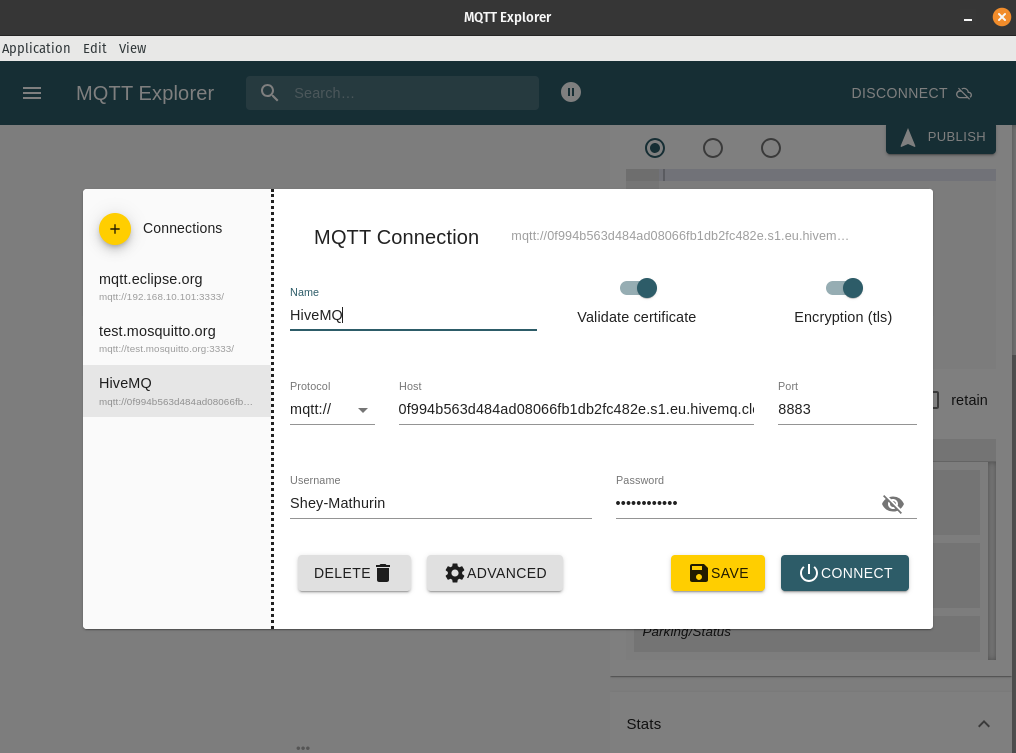


Figure 28

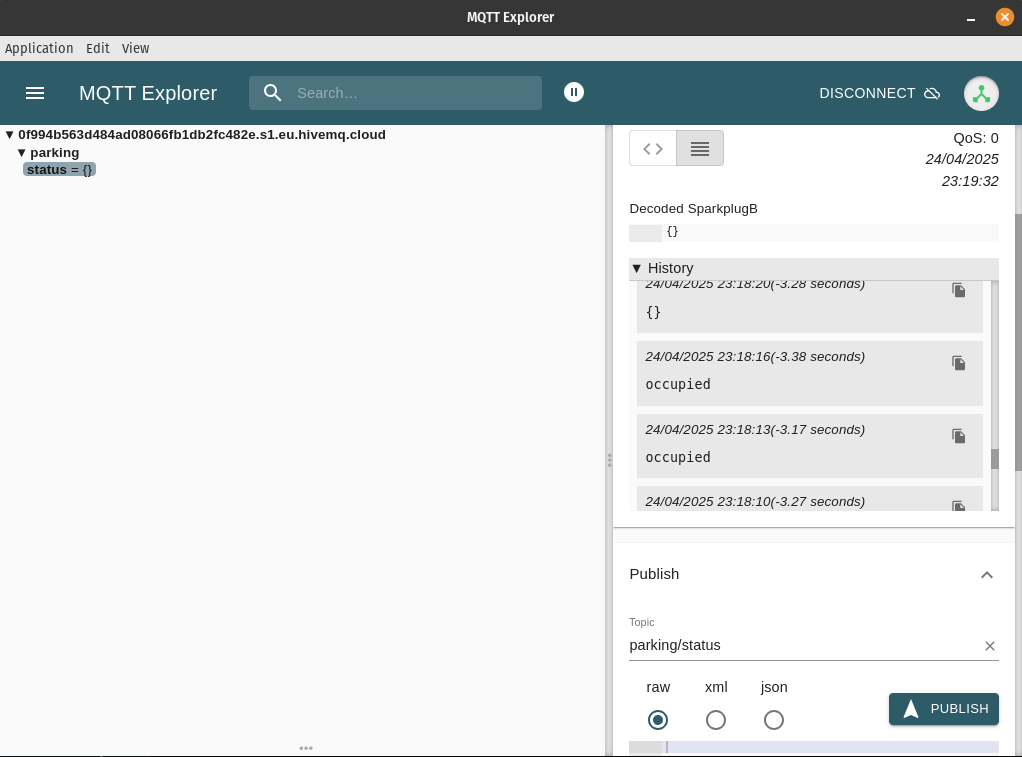


Figure 29

When connecting the MQTT Broker to my Arduino codebase, was a simple process, where I inserted the credentials given from the Hive MQ cluster so I could run and serverless setup and ensure connection to the broker while it was running. Once connected, when the smart parking codebase is running it is run under the variable ‘Parking’ and the published messages are sent in the variable ‘status’ which converts the messages sent to ‘occupied’ for one the car is parked and ‘{}’ which means the space is empty.

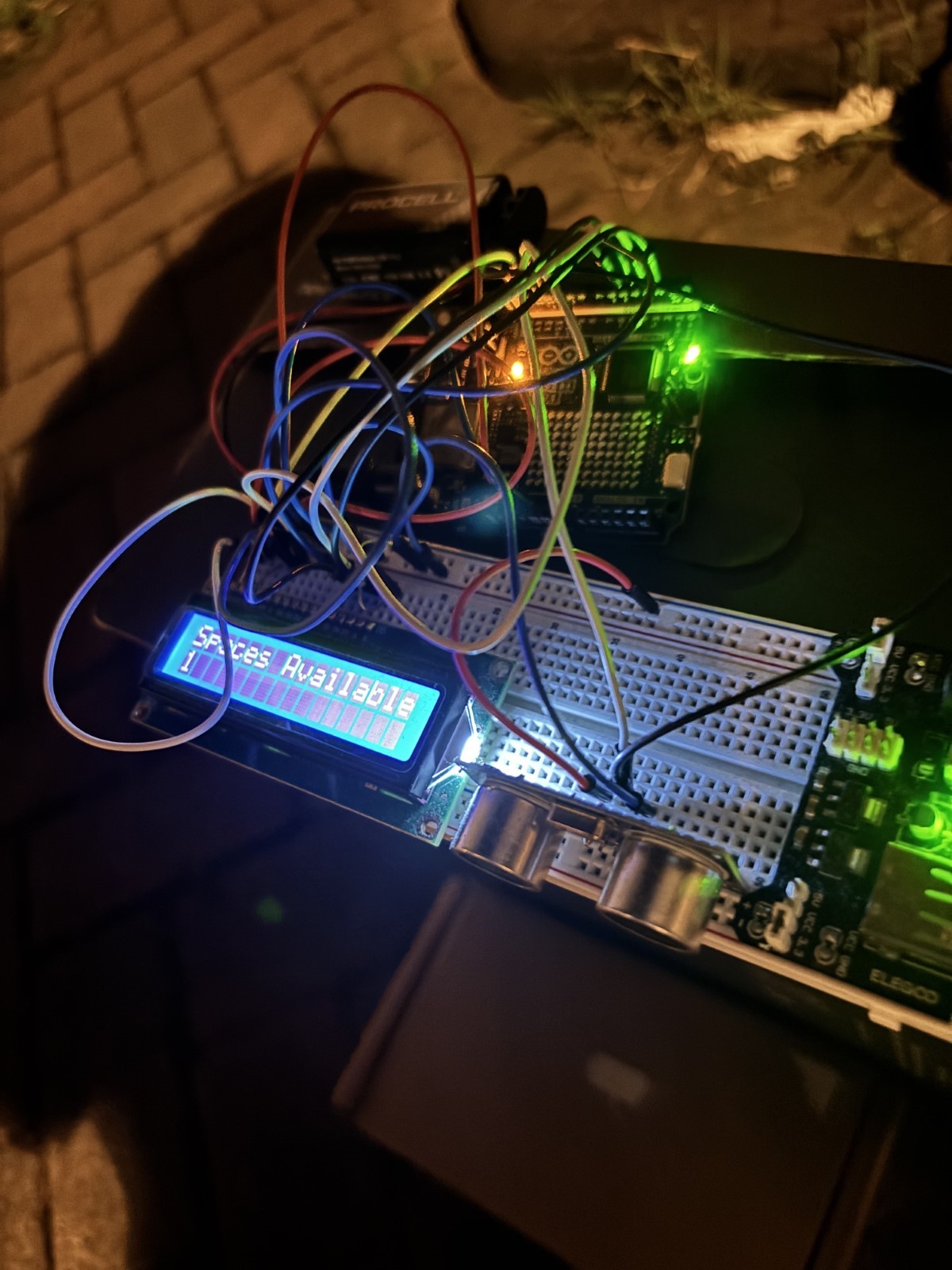


Figure 30

The system performed very well, with no connection interruptions or cutouts. While running inside my house and outside on the driveway where the initial testing was taking place it performed well allowing me to note down the key sensor data that can be later put into a graph to analyse how long it takes to detect if something is in range, the accuracy of identifying a space and a taken space, the actual distance of the vehicle from the sensor and finally, the MQTT messages sent.

### **12.1 System Response and Feedback**

When monitoring the smart parking device only on battery power running of the ESP32-S3, it ran at a reasonable speed and updated the parking availability accurately and quickly making it easy to visualise for the user and determine if a space is taken or occupied. In the MQTT client, MQTT-Explorer the time and date is published with each message making it effortless to read and clear to understand.

### **12.2 Edge Case Scenarios**

Conducting tests on my driveway did cause a bit of hassle as I had to decide whether during the day or the nighttime was appropriate. I had to consider that during the day there may be a lot more people driving around, so cars or people can be obstacles for my testing. The increase in Heat and UV as we come to summer can also affect my hardware and could cause certain parts to malfunction or lose power, so I later decided to conduct my testing at night as there are no pedestrians or obstacles that could affect me as much as if it was during the day.

### **12.3 Limitations Observed**

One limiting factor that came into play when testing my smart parking device on my driveway, was that a housemate of mine also drives and uses the other driveway space. This limited the number of spaces I had to manoeuvre in if necessary and limited how far I could reverse or go forward as I was parked on the lifted curb part of our driveway and closed off by a small wall dividing our house from the other house.

Another limiting factor noticed while testing my smart parking device was that when monitoring the device outside, I had to leave my laptop indoors so the connection to the broker wouldn’t time out or lose connection due to the distance from the router. To conclude, having another house member watching the published messages as I park and view the device LCD outside would be more efficient than running back and forth as they can make me aware of how quickly and accurately the published messages are coming in considering my door isn't too far from my room.

### **12.4 Conclusions and Next Steps**

The smart parking device was tested on a driveway at my student accommodation using a 2014 Hyundai i20. The setup included an Arduino UNO R4 WiFi, ultrasonic sensor, and MQTT communication via a HiveMQ cloud broker. A 9V battery powered the device, allowing it to operate independently of USB power while being tested outside on my private driveway and to maintain Wi-Fi connectivity. Sensor placement was thought out by using some boxes I had lying around which were elevated to align with the vehicle’s bumper for accurate readings. Real-time feedback was displayed on an LCD screen, while detailed distance data was logged in the serial monitor and MQTT-Explorer with timestamps. Testing confirmed strong system responsiveness and reliable MQTT connectivity both indoors and outdoors. However, limitations included restricted parking space due to another resident’s vehicle and the inefficiency of monitoring messages indoors while operating the device outdoors. Environmental factors such as heat and daytime activity also posed minor challenges.

**12.5 Future Improvements**

To enhance efficiency and accuracy, future testing could involve a helper to monitor MQTT data in real time, improving coordination. Relocating the Wi-Fi router or using a range extender could improve outdoor connectivity. Additionally, integrating mobile-based feedback/MQTT application or developing a custom web UI could eliminate the need for physical proximity to the laptop during testing.