IOT DEVICE DEVELOPMENT AND PYTHON SCRIPT DEVELOPMENT

IOT DEVELOPMENT:

Invest in device design, prototyping and pre-deployment testing

IoT is here to stay with billions of devices now active. But it is also now clear that the substantial complexities around deployment have been glossed over. In fact, Cisco Systems found that more than 75% of IoT deployments fail and Microsoft estimates that 30% of IoT projects fail at the Proof of Concept (PoC) stage.

As a starting point, success is about planning for a variety of technical bottlenecks and cost challenges around long device development cycles. For example, prototyping devices can present a substantial delay – of up to 18 months – and if decisions around device design must be made so far in advance, the development cycle will be continually hampered, to the cost of innovation.

This is a first major point of failure for IoT businesses because of the 'miss' in failing to be first to market and the unacceptable limitations on creativity. Device design must consider the need to standardise and simplify production and deployment and 'versioning' of appliances, to meet global market-specific requirements, is counter productive.

Instead, appliances should be future-proofed, incorporating IoT devices that have a single stock-keeping unit, one SIM and are suited to global deployment.Further, when embarking on an IoT project, it's important to fully explore and understand the possible technical constraints and risks relating to the project requirements and how the device will be used in service.The type and configuration of the IoT device and associated connectivity solution affects the build, so making the right procurement choices early in the project are fundamental to its long-term success.

The key to success is ensuring that your device performs in a predictable way. Devices not only need to connect to any network, but also adapt to network variances automatically and stay connected.

During five or ten (or more) years of deployment, unexpected events can affect your devices. For example, new legislation could restrict roaming in a particular geography, a mobile operator may sunset a critical service or a network might suffer performance degradation. IoT connectivity onboarding services must include rigorous process to ensure that a device is connecting correctly, well before reaching the critical phases of your project.

Choose the right connectivity partner

Navigating fragmented IoT technologies, global geographies and industry regulations and standards have all posed significant connectivity challenges, creating a multitude of complex considerations.

Firstly, early connectivity considerations around whether to use cellular, low power or satellite technology, and coverage availability must be made at the outset, during device build.

Secondly, the complexity of working across a fragmented global network of multiple Mobile Network Operators (MNOs), with inflexible contract terms and varying standards, permanent roaming restrictions (making global coverage commercially unviable) all mean that economies of scale simply are not available to businesses beyond the larger connectivity providers. These have all proven costly challenges to operational efficiency and major barriers to global IoT innovation over the past decade,

which have left enterprises cautious about moving to large-scale deployments, particularly given the patchwork of operators cannot offer anywhere near 100% global connectivity.

Easy to manage, ubiquitous, global connectivity has been missing until now, but today it is enabling business to move beyond these prohibitive conditions and remains the foundation to enabling successful large-scale global IoT deployment that will ultimately help the IoT market to 'cross the chasm'. The emergence of the embedded universal integrated circuit card (eUICC), that opens multiple profiles to SIMs and enables seamless network switching, has revolutionised the sector and promises a bright and expansive future for global IoT deployments.

Enable scalable data management & analytics

As the core value proposition of IoT services, data, the process of its transmission to the cloud and large-scale analytics and security capabilities must go hand-in-hand to deliver on IoT's promise of analysis and action. But managing the volume and velocity of data at hyperscale can be both challenging and costly.

It is far easier to perform analytics on well-managed data where only the most relevant data is stored and analysed and it is easier to secure because anomalies, frauds and attacks can be more clearly identified. It's also easier to ensure that data for time-specific services that require instantaneous action will be handled as priority.

Given that exabytes of data will make up the foundations of IoT services, ensuring outcomes match inputs and that costs are not incurred wastefully is critical and this can only be achieved through careful relationship management with suppliers.

Design-in security as part of a holistic strategy

At its foundation, security in successful IoT deployments relies upon four principal factors: tamper-proof physical hardware with an embedded SIM, that is write-protected with no programming interfaces and no debug code; application security enabled by transport layer security (TLS) and HTTPS; network security via the GSM network or VPN security; and side band security such as via SMS.

A common failing in IoT is that security strategies have been approached retrospectively and retrofitted to devices and associated systems, post-manufacture, when in fact optimal security provision should be designed-in from the outset.

With 'over the air' (OTA) and zero touch functionality, a device simply connects to the service provider and downloads the security certificate on power up. This is not only more secure, but it also means that security certificates don't have to be manually assigned. Through OTA software security can also be strengthened and updated through time as required.

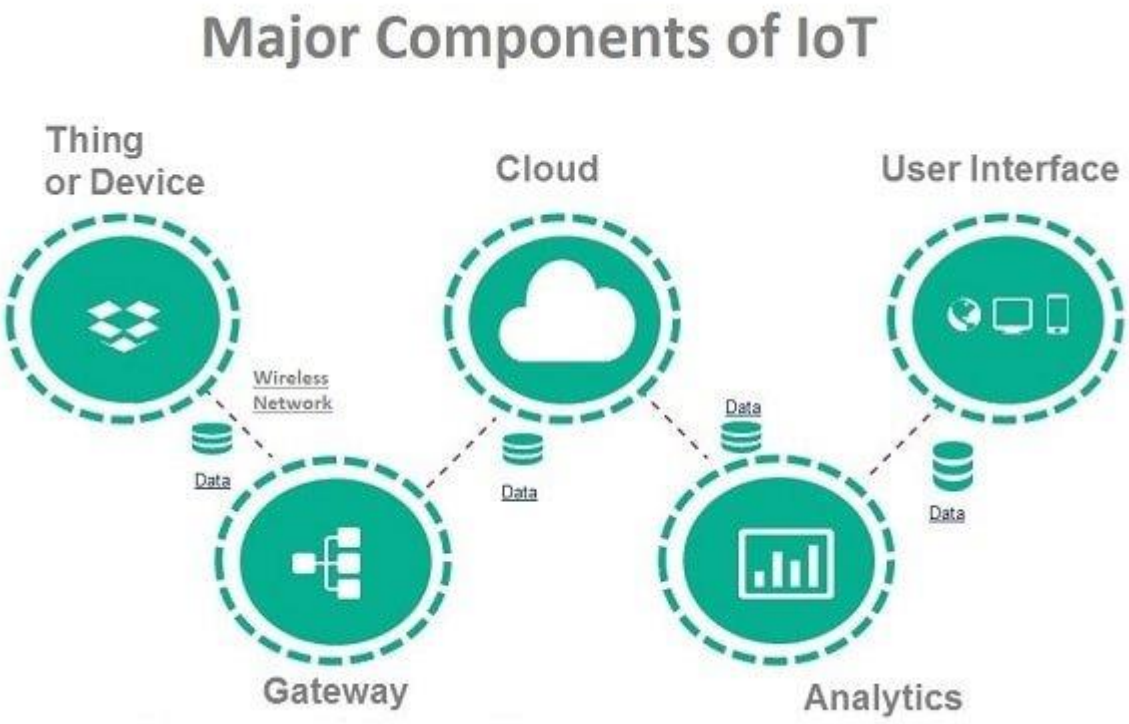
Plan for flexible device management

One of the greatest challenges of a successful IoT deployment is that it becomes harder to manage with scale and organisations need a better understanding of the impact of larger volumes of devices on cost and resources. Decisions around whether connected devices can be rolled out across multiple fragmented regions and whether the device will work in its current format across all geographies or whether versioning will come into play are all critical.

So, organisations must design for growth from the outset, with the flexibility to rapidly scale up if the service is successful. Here, visibility and control of the entire device estate, the connectivity

and the data are essential, with potential issues ranging from predictive maintenance to more urgent fault identification.

Managing this through automation is an essential ingredient of IoT success. Organisations should focus on using IoT managed services suitable for large scale success right from the initiation of a project to ensure it runs smoothly so that benefits are realised and time to market is reduced.



PYTHON :

```
  import time
import some_iot_library  # Import the library for your IoT devices


# Define your IoT device configuration
device_id = "your_device_id"
sensor_id = "your_sensor_id"
connection_params = {...}  # Connection parameters for your IoT devices


# Function to connect to the IoT device
def connect_to_device(device_id, connection_params):
    try:
```

```
        # Initialize and connect to the IoT device using provided parameters
        device = some_iot_library.connect(device_id, connection_params)
        return device
    except Exception as e:
        print(f"Error connecting to the IoT device: {e}")
        return None

# Function to read data from the sensor
def read_sensor_data(device, sensor_id):
    try:
        # Use the library to read data from the sensor
        data = device.read_sensor(sensor_id)
        return data
    except Exception as e:
        print(f"Error reading data from the sensor: {e}")
        return None

# Function to send data to a remote server or cloud
def send_data_to_server(data):
    try:
        # Implement code to send data to a remote server or cloud service
        # This might involve making HTTP requests or using a specific API
        # Example:
        # response = requests.post("your_server_url", json=data)
        # Handle the response accordingly
        pass
    except Exception as e:
        print(f"Error sending data to the server: {e}")

# Main function to run the script
def main():
device = connect_to_device(device_id, connection_params)
```

```
    if device is not None:
        while True:
            sensor_data = read_sensor_data(device, sensor_id)
            if sensor_data is not None:
                # Process the sensor data (if needed)
                # For example, perform calculations, filtering, or data formatting
                processed_data = sensor_data

                # Send the data to a server or cloud
                send_data_to_server(processed_data)

            # Define the data collection interval
            time.sleep(5)  # 5 seconds

if __name__ == "__main__":
    main()
```