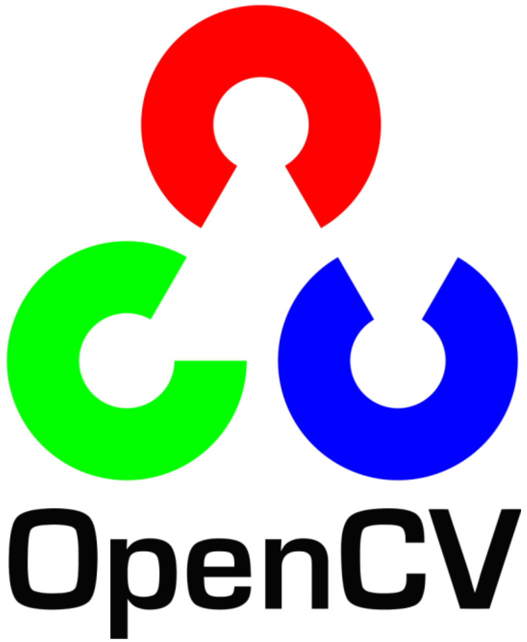
Mobile Applications for Sensing and Control

Sep Makhsous

Week 6

OpenCV

- OpenCV (Open Source Computer Vision Library) is an open-source library that includes hundreds of computer vision algorithms.
- It is written in C++ and can also be used in Python and Java.
- OpenCV is used for
    - Image processing
    - Facial recognition
    - Object detection



https://opencv.org/

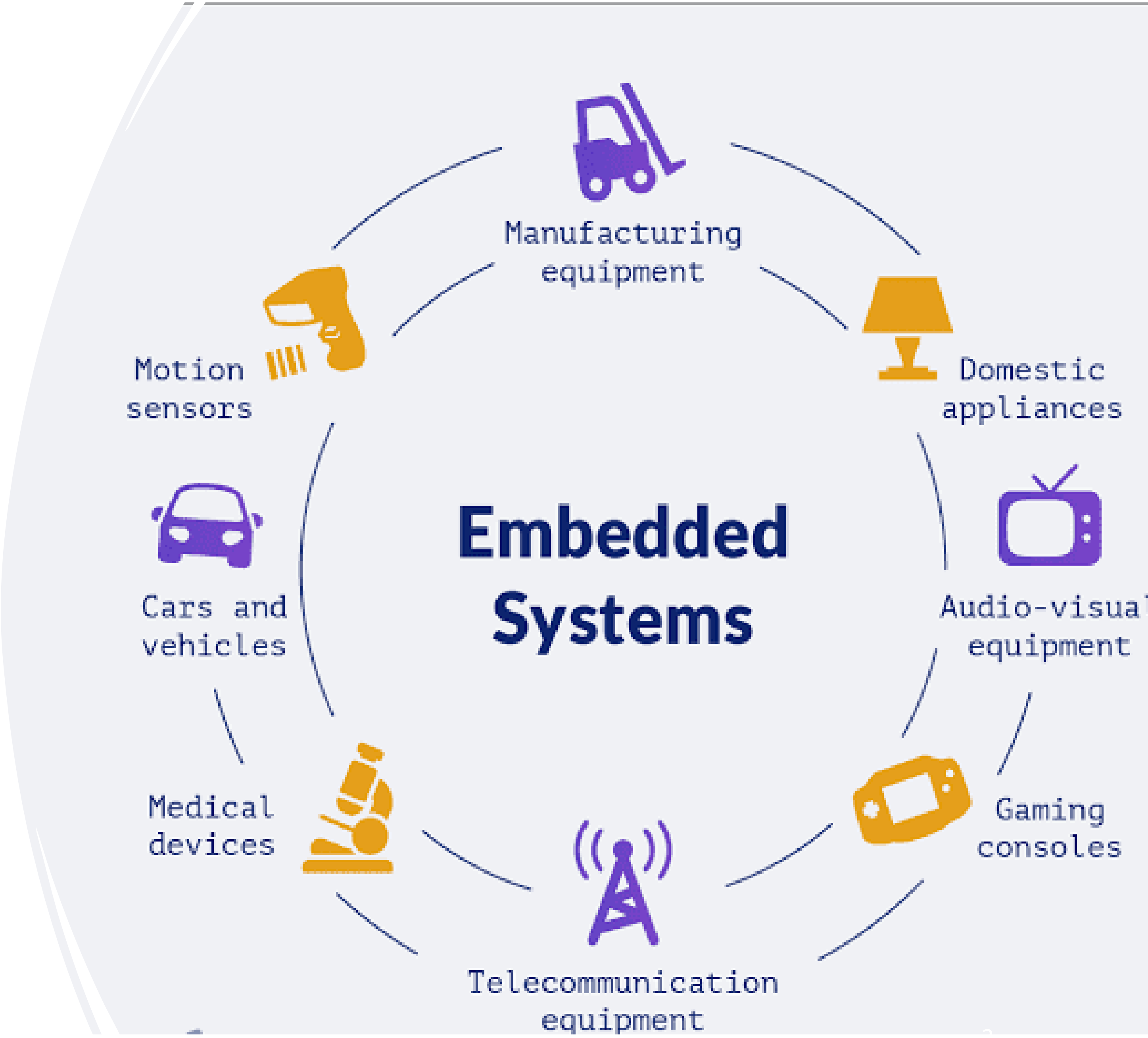
How to setup OpenCV

- Download OpenCV SDK:
  - Visit the official Sourceforge page (search online).
  - Download the latest version of OpenCV
- Import OpenCV into Android Studio:
  - Open your project in Android Studio.
  - Go to File -> New -> Import Module.
  - Select the sdk directory from the extracted OpenCV SDK folder.
  - Rename the imported module to opencv for clarity (Right-click on the sdk folder, select Refactor -> Rename -> Rename Module).
- Add OpenCV to Build Process:
  - In your app's build.gradle file, add implementation project(":opencv").
- Initialize OpenCV in Your Application:
  - Add OpenCVLoader.initDebug() in the onCreate() function of your main activity to load OpenCV libraries at runtime.
- Example Function - Convert Image to Grayscale:
  - Create a function to convert a Bitmap to grayscale using OpenCV.
- Handle Library Conflicts in Gradle:
  - In your app's build.gradle, under android -> packagingOptions, use pickFirst for conflicting libraries, e.g., pickFirst 'lib/*/libc++_shared.so'.

https://philipplies.medium.com/setting-up-latest-opencv-for-android-studio-and-kotlin-2021-edition-259be404b133

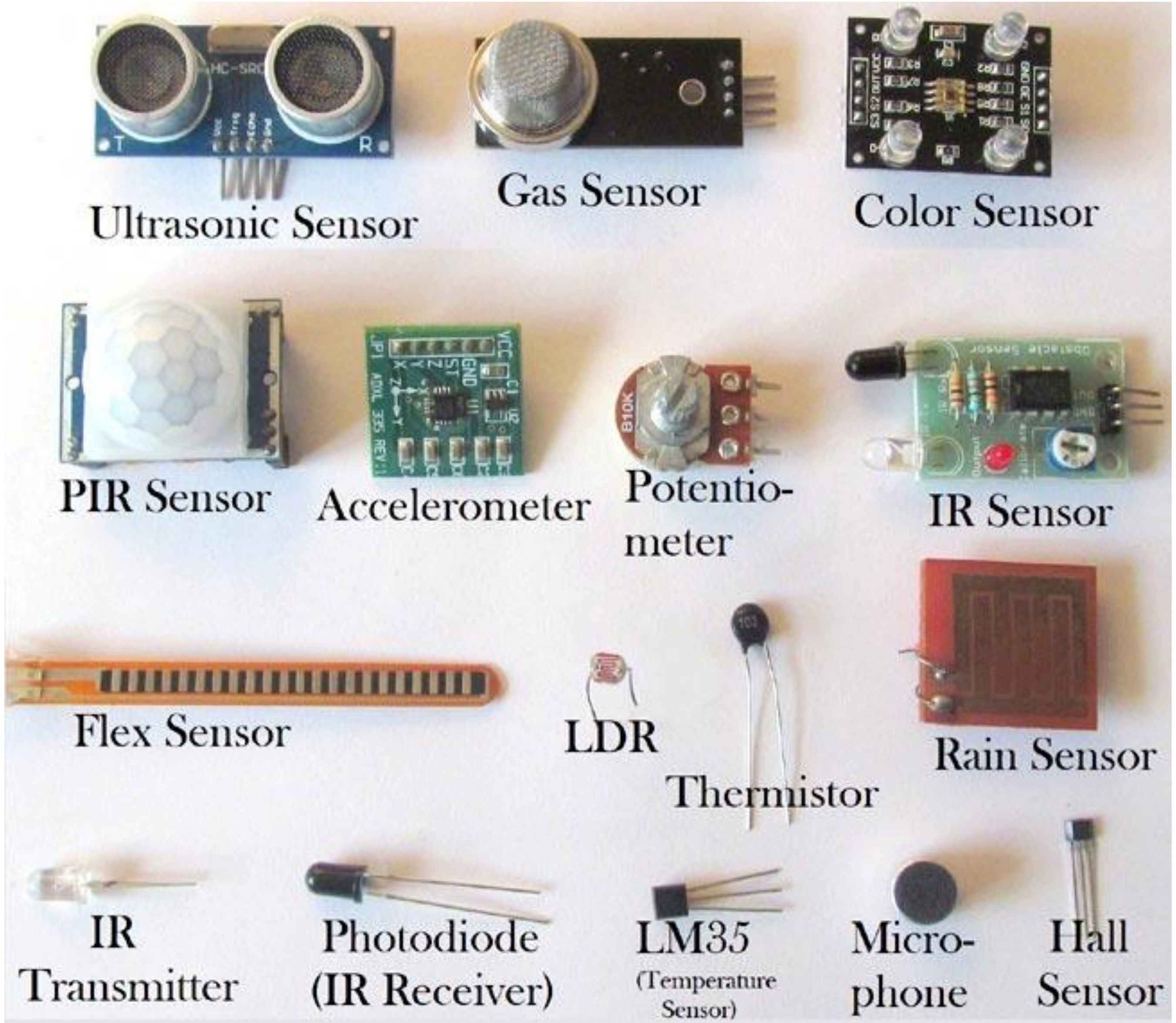
Embedded Systems: Sensor Connectivity + BLE

- Importance of Embedded Systems in Mobile Apps
  - Definition of embedded systems
  - Role of embedded systems in mobile apps
  - Common use cases of embedded systems in mobile apps (e.g., IoT devices, wearables, and smart home appliances)



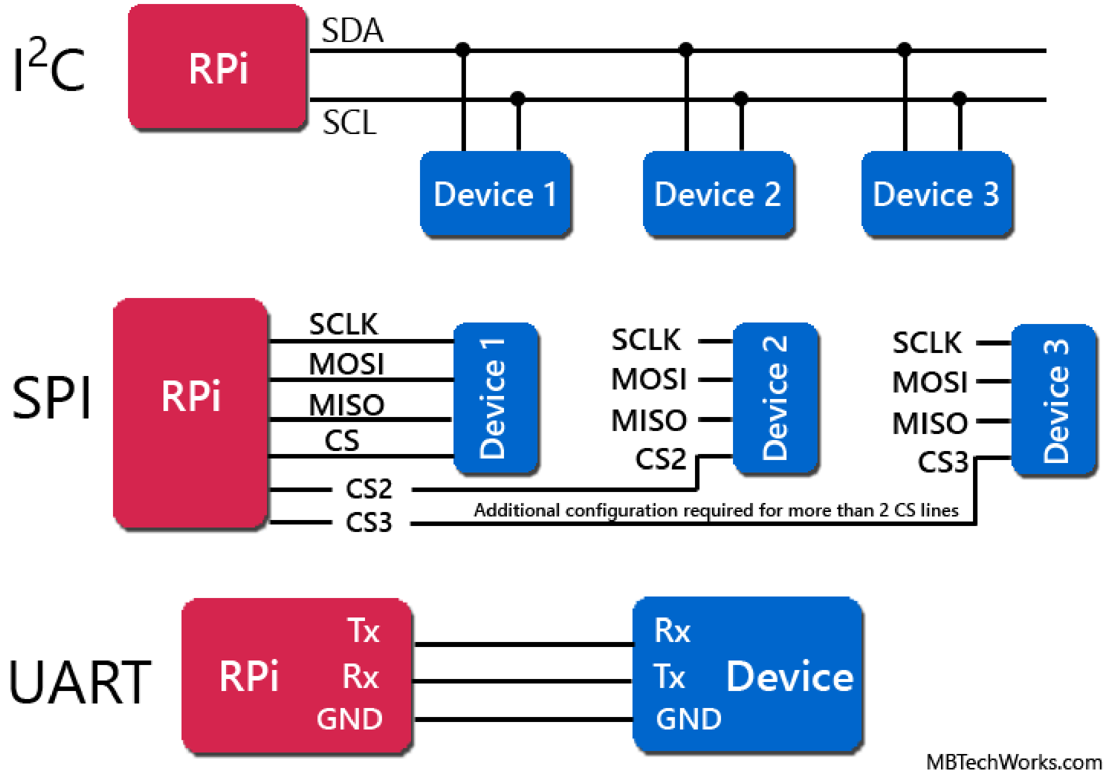
Sensor Connectivity and Communication

- Definition of sensors
- Types of sensors (e.g., temperature, humidity, proximity, accelerometer, and gyroscope)
- Sensor data processing in mobile apps



Sensor Communication Protocols

- Definition of communication protocols
- Common communication protocols for embedded systems (e.g., I2C, SPI, UART, and CAN)
- Factors affecting the choice of communication protocol (e.g., data rate, distance, power consumption, and cost)

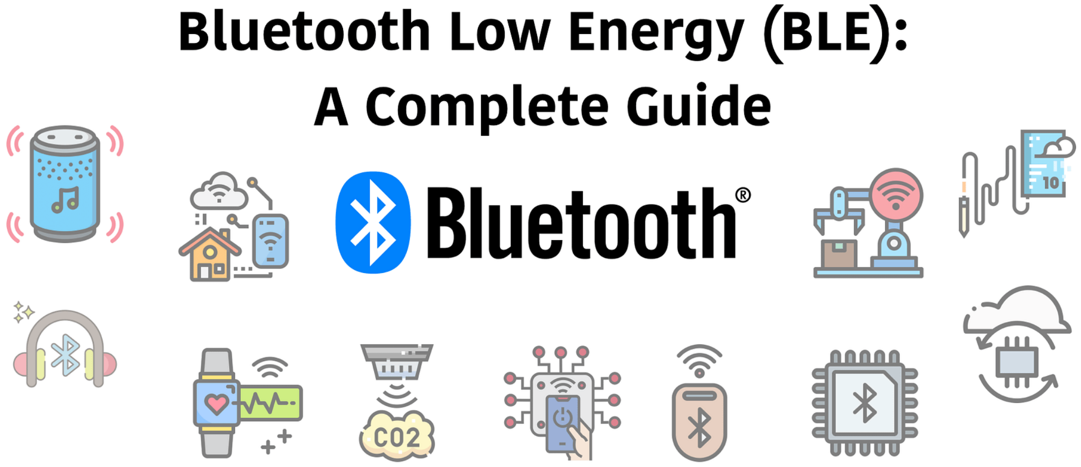


Connecting Sensors to Mobile Devices

- Direct connection (e.g., USB or GPIO)
- Indirect connection (e.g., using a microcontroller or a module)
- Wireless connection (e.g., Wi-Fi, Bluetooth, and Zigbee)

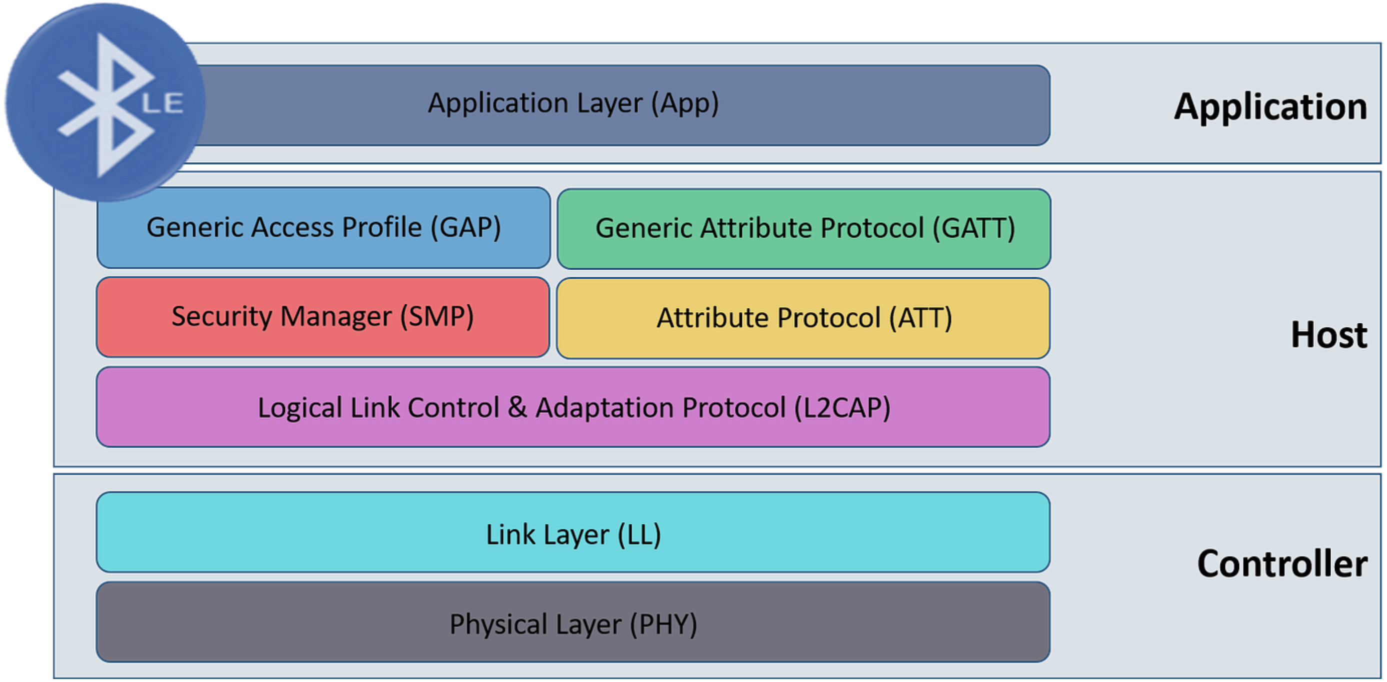
Bluetooth Low Energy (BLE) in Embedded Systems

- Definition of Bluetooth Low Energy
- Comparison between Classic Bluetooth and BLE (e.g., power consumption, data rate, and range)
- Common use cases for BLE in mobile apps (e.g., wearables, beacons, and smart home devices)



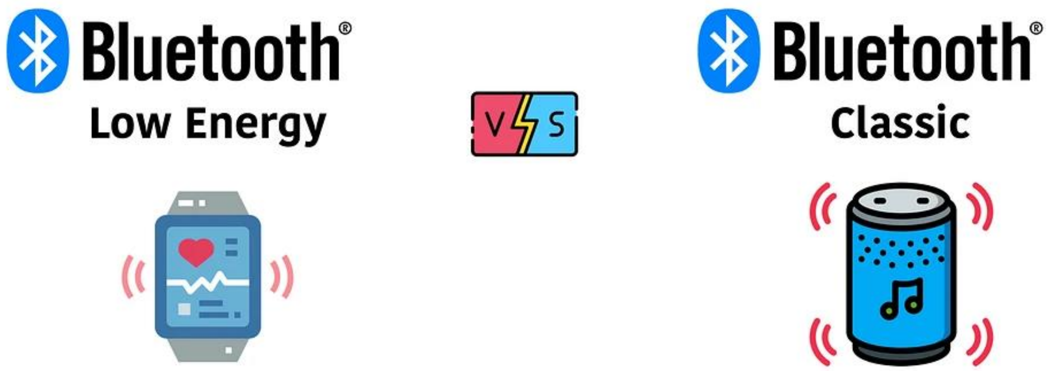
BLE Architecture

- aBLE stack components (e.g., GAP, GATT, L2CAP, and HCI)
- BLE roles (e.g., central, peripheral, observer, and broadcaster)
- BLE services and characteristics



Best Practices - BLE in Mobile Apps

- Adaptive connection interval
- Optimizing advertising and scanning parameters
- Utilizing sleep modes

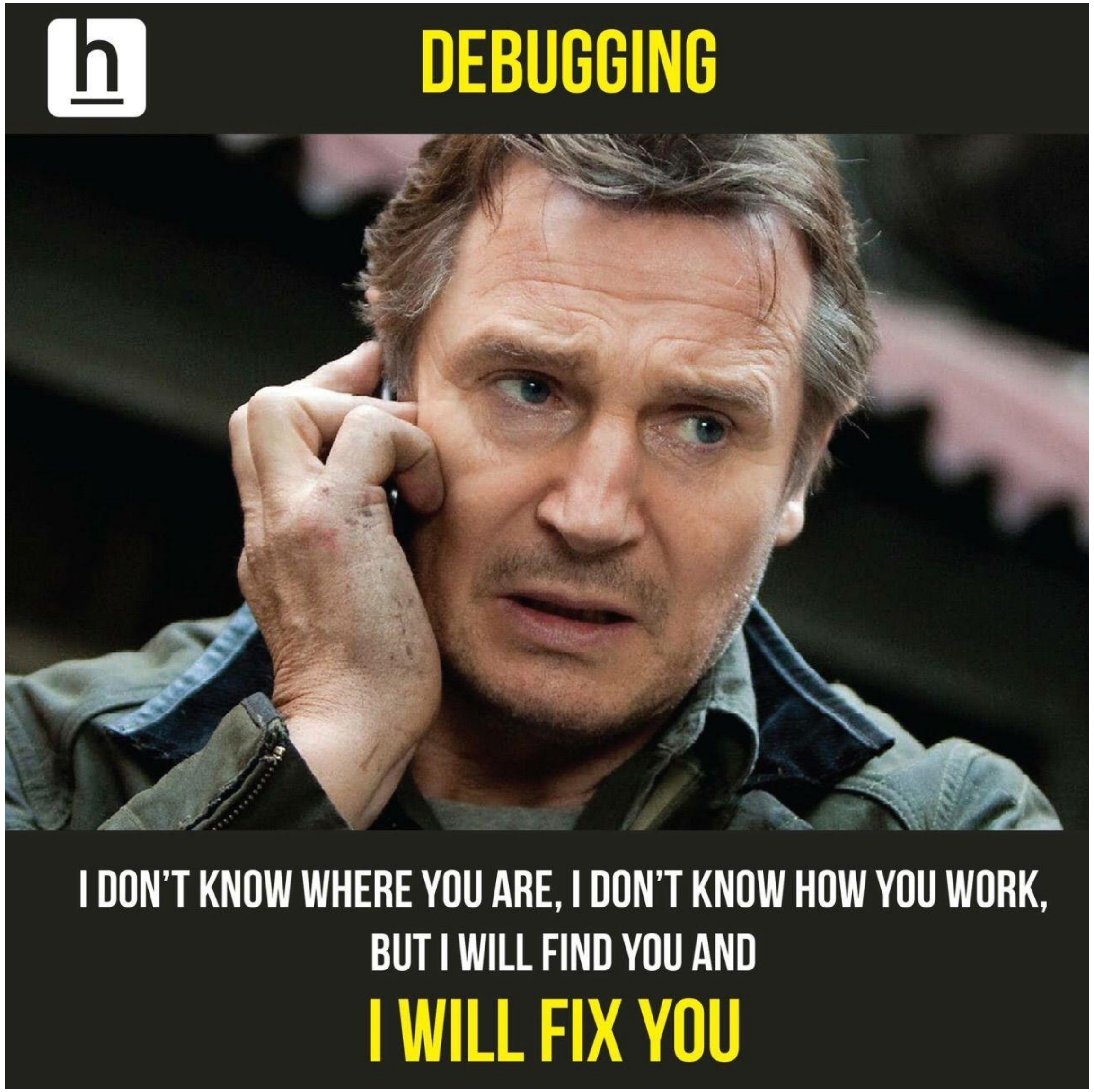


Security Considerations

- BLE security features (e.g., pairing, bonding, and encryption)
- Security risks and mitigations (e.g., man-in-the-middle attacks and eavesdropping)

Debugging

- BLE debugging tools (e.g., nRF Connect and LightBlue)
- Tips for troubleshooting common BLE issues



Figma Tutorial

ICTE 7

- Build your first Bluetooth app