Bluetooth short-range wireless technology enables two devices to connect directly without requiring supporting network infrastructure such as a wireless router or access point.

- Wirelessly connect peripherals (mouse, headphones… ect) to electronic devices
- Pairing: connecting two bluetooth device
    - each device registers the pairing information, including the security key, of its partner device.
    - Saved on both devices so they can auto-reconnect
- Operates on radio frequency in the 2.4 GHz range
- Range: 30 to 300 feet
- To operate bluetooth: each device registers the pairing information, including the security key, of its partner device.
- 

- Use Raspberry pi's terminal command line to connect

Bluetooth Communications

- How to connect to Bluetooth between your Raspberry Pi and ground station
    - `Bluetoothctl`
    - `pair [MAC address]`
    - `trust [MAC address]`
    - `connect [MAC address]`
    -
- How to measure RSSI (and what is RSSI)
    - "Received Signal Strength Indicator"
    - a measurement of how strong a radio signal is being received by a device, typically expressed in decibels (dBm)
    - Use built in WIFI capabilities on raspberry pi to access RSSI `Iwconfig` to terminal
    - Look for the line corresponding to the network you're connected to, and within that line, find the "Link Quality" field which will display the RSSI value in dBm.

- What is your transmit power
  - Uses Cypress CYW43455 chipset
  - maximum transmit power of **+4 dBm (2.5 mW)**.
  - To check in terminal: hciconfig hci0
- How to send data over Bluetooth
  - Import bluetooth
  - ```
    # Target Bluetooth address
    target_device = "00:11:22:33:44:55"

    # Create a socket
    sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)

    # Connect to the target device
    sock.connect((target_device, 1))

    # Data to send
    data_to_send = "Hello from Raspberry Pi!"

    # Send data
    sock.send(data_to_send.encode())

    # Close the socket
    sock.close()
    ```
- Location and description of Bluetooth antenna
  - Integrated in printed circuit board and is not visible
  - Uses bluetooth 5.0
  - Located at the top left corner of the board
- Change your Bluetooth settings
  - sudo nano /etc/bluetooth/main.conf
  - ^^ to edit
  - sudo systemctl restart bluetooth
  - ^^ restart for changes to take effect
- Send a picture over Bluetooth
  - Identify the Bluetooth device address with `bluetoothctl devices`.
  - Use commands like `sendfile <device_address> <image_file_path>` to transfer specific image files.
- Send encrypted data over Bluetooth
  - Implement cryptography library
    ```

```
import bluetooth
from cryptography.fernet import Fernet



# Generate a key
key = Fernet.generate_key()



# Function to encrypt data
def encrypt_data(data, key):
    f = Fernet(key)
    encrypted_data = f.encrypt(data.encode())
    return encrypted_data



# Bluetooth connection setup (replace with your target device details)
target_device = "00:11:22:33:44:55"
port = 1



# Establish Bluetooth connection
sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
sock.connect((target_device, port))



# Example data to encrypt
data_to_send = "Secret message"
```

```
# Encrypt data
encrypted_data = encrypt_data(data_to_send, key)



# Send encrypted data over Bluetooth
sock.send(encrypted_data)



sock.close()
```

Step 1: Set Up Bluetooth on Raspberry Pi

Install Bluetooth Tools: Run the following commands to ensure the Bluetooth utilities are installed:

```
sudo apt update
sudo apt install bluetooth bluez python3-pip
```

Enable and Start the Bluetooth Service:

```
sudo systemctl enable bluetooth
sudo systemctl start bluetooth
```

Step 2: Install Python Libraries

Install `pybluez`: This library allows Python to communicate with Bluetooth devices.
bash

```
pip3 install pybluez
```

(Optional) Install `bluetooth` or `bleak` for BLE Devices:

- `bluetooth`: For classic Bluetooth connections.
- `bleak`: For Bluetooth Low Energy (BLE) devices.

```
pip3 install bleak
```

Step 3: Scan for Bluetooth Devices

Here is a Python script to discover nearby Bluetooth devices:

```
import bluetooth

print("Scanning for devices...")
devices = bluetooth.discover_devices(lookup_names=True,
duration=8)

if devices:
    print("Found devices:")
```

```
    for addr, name in devices:
        print(f"  {name} - {addr}")
else:
    print("No devices found.")
```

Step 4: Connect to a Bluetooth Device

Classic Bluetooth Example: Use `bluetooth.BluetoothSocket` for sending and receiving data.

```
import bluetooth

# Specify the target device's MAC address
target_address = "XX:XX:XX:XX:XX:XX"  # Replace with your
device's MAC

try:
    # Create a Bluetooth socket
    sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
    sock.connect((target_address, 1))  # Channel 1 is commonly
used

    print("Connected. Sending data...")
    sock.send("Hello from Raspberry Pi!")
    sock.close()
except bluetooth.BluetoothError as e:
    print(f"Bluetooth connection failed: {e}")
```

1.

BLE Example with `bleak`:

python

Copy code

```
from bleak import BleakClient

# Replace with the BLE device's MAC address
address = "XX:XX:XX:XX:XX:XX"
```

```
async def connect_ble():
    async with BleakClient(address) as client:
        print(f"Connected: {client.is_connected}")
        await client.write_gatt_char("your_characteristic_uuid",
b"Hello BLE!")

import asyncio
asyncio.run(connect_ble())
```

2.

Step 5: Additional Notes

Pairing Devices: You may need to pair devices via the terminal:
Inside the `bluetoothctl` shell:

```
scan on
pair XX:XX:XX:XX:XX:XX
trust XX:XX:XX:XX:XX:XX
connect XX:XX:XX:XX:XX:XX
```

-
- **Device-Specific Communication:**
    - Check the documentation for the specific Bluetooth device you're connecting to for protocols (e.g., Serial Port Profile, GATT for BLE).

Project Description

In this project, you will learn how to interface with and use the power system, Bluetooth communications, IMU, and camera. Together as a team, you are responsible for knowing how to connect, power, and program all of these components. For each component, choose one or two team members to focus on that component and write an

interface document explaining its use. The interface document should include mechanical, electrical, and software interfaces. At the end of the project, each member will present their component's interface and teach the team how to use it.

Interface Document

Your interface document should include the following:

- Mechanical interfaces: simple mechanical drawing, size of screws used, keying or alignment
- Electrical interfaces: power (voltage and current levels), pinout, type of cables used
- Software interfaces: python code demonstrating relevant functions, description of software settings and options

- Code should be uploaded to GitHub or your team's preferred version control system as a separate .py file

Presentations

Presentations should have two main components:

1. Overview of the interface document, including an explanation of the component's functionality and the code to control it
2. Demo of how to connect and use the component

You may also include time for each team member to try out the other components.