# Kiosk Code Instruction (**Nattapong\_Pun/vitalSign/Kiosk** Folder)

Kiosk.ipynb: Import hardware and communication protocol (**commProtocol**) from config file in **config** folder

**Construct Kiosk Object** -> Kiosk(configFile)

\*\*\*configFile is python dictionary as follow\*\*\*\*\*\*\*

configFile={"hardware":"config/banbangkhae.ini",

"commProtocol":"config/emetWorksCommProtocol.ini"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The object would import associated class for controlling the vital sign **hardware** and **commProtocol**

Choose **commProtocol** by calling Kiosk.commProtocol

eg. kiosk = Kiosk(configFile) >> kiosk.mqtt() would initiate mqtt server

**Construct Mqtt Object** -> mqtt(configFile)

Pass down mqtt parameter and connected to broker when called

On\_message created publisher thread to pass down the message to associated device class

**Hardware**

Method: **start**, **stop**, **reset**, **status** (These take no argument)

* **start**: start the measurement, eg: thermalGun.start()
* **stop**: stop the measurement, eg: thermalGun.stop()
* **status**: status of devices, eg: thermalGun.status()
* **reset**:Disconnect and Reconnect Bluetooth device. For Serial Port, re-constructing the pyserial object, eg.thermalGun.reset()

Flag: **isRunning**, **msgReady**

* if **isRunning** flag is True, return the thread and terminate, eg. thermalGun.isRunning
* if **msgReady** flag is True, publish to broker server, eg. thermalGun.msgReady

Attribute: **msg**

* **msg**: message to publish to broker, eg. thermalGun.msg