indoor-climate-raspi-files

What, where

Root directory

- **src/** - source code of the project
  - **krecik_iot_controller/** - contains files related to the **main controller** and its services
  - *main.py* - main file which is run on startup
- *activate.sh* - script which is run on startup, called from /etc/rc.local. It runs src/main.py.

src/krecik_iot_controller

- **krecik_iot_controller/**
  - **services/** - services used by the main controller
  - *krecik_iot_controller.py* - the main controller, responsible for:
    - datasource initialization (using bluetooth module)
    - handling Wi-Fi (connecting, checking if connected)
    - handling data sending to the server (includes reading from sensors)

Services are described below.

src/krecik_iot_controller/services

- **services/**
  - **bluetooth/**
  - *aes_cipher.py* - contains class responsible for encrypting and decrypting data using AES algorithm.
  - *krecik_sensor.py* - contains class responsible for reading data from the sensor.
  - *datasource.py* - contains class responsible for handling data: wifi ssid, wifi password, backedn url, backend auth token. The controller uses this class to get the data, If not configured, the krecik_iot_controller starts initialization process and configures it using the data from the bluetooth module. If configured, the krecik_iot_controller uses the data from the datasource that is stored in the file.

Bluetooth services is described below.

src/krecik_iot_controller/services/bluetooth

Contains files related to bluetooth module.

- **_bluetooth/**
  - **cputemp/** - contains files related to the bluetooth module based on *Douglas6's cputemp* lirary (later mentioned as the library).

- ble_config_example.py_ - Douglas6's example of the configuration file for the bluetooth module.
- *krecik_ble_config.py* - configuration file for the bluetooth module. Defines **Advertisments**, **Characteristics**, **Services** and **Descriptors** (that inherit after base classes from the library).
- *krecik_ble_service.py* - contains class which represents a bluetooth service for our purposes. It consists of the **bluetooth server** (provided by the library) object that is initialized with the configuration file and the **bluetooth service** object that is initialized with the classes from the *krecik_ble_conf* file. It also contains methods for starting/ stopping and reading/writing data to the characteristics (**encryption included**).

How

1. Ubuntu runs the /etc/rc.local script on startup, it runs our activate.sh script.
2. activate.sh:
   - activates system bluetooth and configures it to be discoverable and pairable
   - inits the environment variables for the encryption
   - runs the main.py script from src directory.
3. main.py:
   - creates the controller object *KrecikIOTController*
   - runs the controller's main loop
4. Controller object creation:
   - checks if datasource is configured (from file)
   - if not, it starts the datasource initialization process (using bluetooth module)
     - it uses KrecikBleServer to open the bluetooth server
     - it reads the data KrecikBleServer got (decryption is done by the server),
     - if data is valid, it saves it to the file
     - if not, it awaits for the next data
   - when datasource is configured, KrecikIOTController tries to connect to the wifi.
   - if connection is successful, KrecikIOTController tries to send the data to the server.
   - if sending is successful, KrecikIOTController is ready to work.
   - if connection or sending is not successful, KrecikIOTController restarts the datasource initialization process.
5. KrecikIOTController main loop:
   - KrecikIOTController reads the data from the sensor
   - KrecikIOTController tries to send the data to the server
   - if sending is not successful, KrecikIOTController puts the data to the queue

- if sending is successful, KrecikIOTController tries to send the data from the queue, until it is empty or sending is not successful
- KrecikIOTController sleeps for the given time