RFI(N)D ME

Yu Guoguo
Cayuso Arthur
Dobretz Kevin

Sommaire

1. Le Projet
2. Caractéristiques des systèmes utilisés
3. Schéma bloc
4. Fonctions d’initialisation et utilité
5. Récupération de valeurs
6. Difficultés rencontrées
7. Interaction avec les commandes de scan
8. Interface et jeux

Le Projet

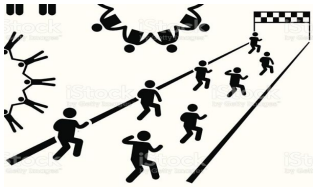
Conditions à respecter :

- Proposer un projet qui utilise la technologie RFID.
- Utiliser lecteur BlueBox, Antenne BlueBox, TAG RFID

Offrir une application ludique à l'utilisateur pour qu'il puisse jouer avec des TAG RFID.

Initialisation :

- Calibrate : Ajoute le nombre de joueur dans la partie



Jeux disponibles :

- Hide and Seek : Cache le TAG et cherche-le grâce à la réception RSSI.
- 1,2,3 Soleil : Départ d'une distance pour rejoindre l'antenne et toucher l'antenne sans se faire scanner.

Caractéristiques des systèmes utilisés part.1

Côté utilisateur :

- BLUEBOX READER (UHF INDUSTRIAL READER ADVANT LR 4CH)
  - Fréquence : 866.5 [MHz]
  - Puissance max transmise : 32 dBm donc 2dB ce qui fait 10^(2/10) = 1.58 [W]
  - Pt = 1.58 [W]

Electrical Specifications	
Power Supply	24 VDC ± 10 %
Power Consumption	3.5 W
Operating Frequency	865–960 MHz (ETSI & FCC)
Output Power	+32 dBm (SW adjustable)
Operating Distance	up to 12 m
Connectors	2x M12 (Power & Interface)
Antenna Connector	4x TNC-Female Integrated Multiplexer
Digital I/O	2 (opto-isolated)
Memory Extension	Micro SD-Slot
Interface options	RS232/485, Ethernet, Profibus, Profinet, Modbus (TCP)

- BLUEBOX ANTENNA (UHF LONG-RANGE ANTENNA)
  - Fréquence : 866.5 [MHz]
  - Gain de l’antenne : dBi = dBic donc 10^(8.5/10) = 7.07 [S.U]
  - Gt = 7.07 [S.U]

Electrical Specifications

Operating Frequencies:	865 – 867 MHz (EU) & 902 – 928 MHz (US)
Operating Distance:	Up to 12 m
Antenna Connector:	TNC-Female
Antenna Gain:	8.5 dBic
Beamwidth:	69° / 69°
Axial Ratio:	1 dB
Polarization:	Circular

Organismes de réglementations des transmissions :

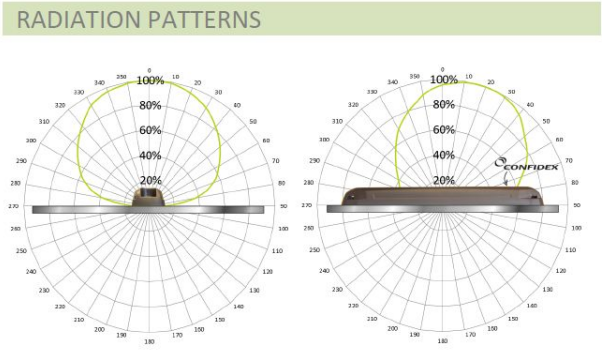
- ETSI (Europe) 865–868 MHz
- FCC (USA) 902–928 MHz

Caractéristiques des systèmes utilisés part.2

Côté TAG RFID :

- TAG Confidex Survivor™ avec Impinj M4QT
  - Fréquence : 866.5 [MHz]
  - Chip meilleure distance Impinj M4QT : 16m
  - Sensibilité à la réception : -17.4dBm , Pr_dB = -17.4 – 30 = -47.4 dB.

    Puissance reçu[W] = 10^(-47.4 /10) = 18.19 x 10^-6[W]



- Antenne du TAG avec Impinj M4QT
  - Fréquence : 866.5 [MHz]
  - Chip meilleure distance Impinj M4QT : 16m
  - Gt = 10 ^(8/10) = 6.31 [S.U].

Pourquoi calculer ces caractéristiques ?

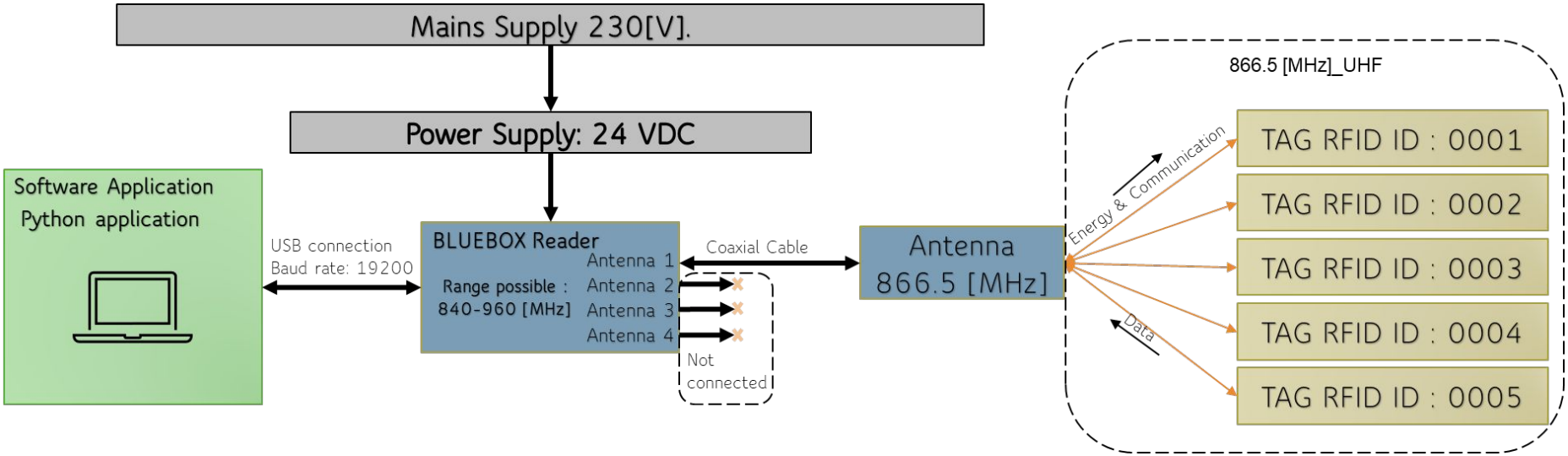
La partie réception RSSI n'était pas évidente à obtenir :

- Solution pour obtenir la distance entre l'antenne et le TAG RFID.
  - Équation de Friis (Quadratique)

Distance

$$R = \frac{\lambda}{4*\pi} * \sqrt{\frac{G_T * G_r * P_{T\_W}}{P_{r\_W}}}$$

Schéma bloc


Mains Supply 230[V].
Power Supply: 24 VDC
Software Application
Python application
USB connection
Baud rate: 19200
BLUEBOX Reader
Range possible :
840-960 [MHz]
Antenna 1
Antenna 2
Antenna 3
Antenna 4
Not connected
Coaxial Cable
Antenna
866.5 [MHz]
866.5 [MHz]_UHF
Energy & Communication
Data
TAG RFID ID : 0001
TAG RFID ID : 0002
TAG RFID ID : 0003
TAG RFID ID : 0004
TAG RFID ID : 0005

RFI(N)D me partie 2 - Guoguo

(Placeholder titles)

(Placeholder titles)

Programmation matérielle d'un scan RFID sur Python:

- Communication avec la bluebox
- Setup (fréquences, canaux…)
- Récupération des valeurs:
  - Firmware, température
  - tag ID
  - RSSI, reflected power

La partie technique du projet

- Les fonctions d’utilité (Guoguo)
- La logique du jeu (Kevin)

Les fonctions d’utilités

- Les fonctions d’initialisation
  - “RF activation”
  - “Antennas Auto-tuning”
- La récupération des valeurs:
  - calcul de checksum
  - split_rx (parsing)
  - La détection et la récupération de tag
    - Merci Giovanni
  - RSSI (dans notre cas reflected power)
  - la température
  - la version de firmware
  - etc

```
device # 219515420048
firmware: BB_TWO_4U  2.04
temp: 2.25°C
status: 4
parametters: x022AFF4810000001A0
RAM config pp 1-7: b''
ROM config pp 1-7: x15
str(rx):b'\x01FF\x020000000000\x03\x00\r'
str(rx,utf8):FF0000000000
tag: x020000000000
RF power: b''
RF activation response: x06
autotuning response: b''
RF sensitivity: b''
Reflected power: b''
str(rx):b''
str(rx,utf8):
RSSI power: b''
show_first_read_tag:
print tag;
in while loop 0
str(rx):b''
str(rx,utf8):
tag: b''
RF power: b''
RF activation response: b''
autotuning response: b''
```

https://github.com/oragegu/rfid-game

	5121L, 5131L	5121H, 5131H	5121U, 5131U	5221L, 5222L, 5	5221H, 5222H, 5	5221U-S, 5222U	5231U, 5232U
Device Reset	✓	✓	✓	✓	✓	✓	✓
Read Device Serial Number	✓	✓	✓	✓	✓	✓	✓
Read Ethernt MAC Address				✓	✓	✓	✓
Read Bluetooth MAC Address							
Read Firmware Version	✓	✓	✓	✓	✓	✓	✓
Firmware Upgrade	✓	✓	✓	✓	✓	✓	✓
Read Temperature				✓	✓	✓	✓
Read Date/Time				✓	✓	✓	✓
Write Date/Time				✓	✓	✓	✓
Write ROM General Parameters	✓	✓	✓	✓	✓	✓	✓
Write RAM Configuration Parameters							
Write ROM Configuration Parameters	✓	✓	✓	✓	✓	✓	✓
Write ROM Default Parameters	✓	✓	✓	✓	✓	✓	✓
Read RAM General Parameters	✓	✓	✓	✓	✓	✓	✓
Read RAM Configuration Parameters	✓	✓	✓	✓	✓	✓	✓
Read ROM Configuration Parameters							
Sleep Mode							
'RF Reading' Test			✓			✓	✓
'RF Power' Test			✓			✓	✓
'RF Sensitivity' Test			✓				
Read Reflected Power			✓			✓	✓
Read RSSI Power			✓				
Digital Output Activation	✓	✓	✓	✓	✓	✓	✓
Read Device Status	✓	✓	✓	✓	✓	✓	✓

Les fonctions d'initialisation

- "RF activation"

In 'continuous' mode, this command is used to resume the activity of the RF antennas connected to the **BLUEBOX**; see also 'RF deactivation' command.

The 'master' sends the following command:

SOH <add h> <add l> STX '3' '9' ETX <bcc> CR

- "Antennas Auto-tuning"

2.27 Antennas Auto-Tuning

This command is used to start an auto-tuning procedure on the RF output channels to improve the reading performances of the **BLUEBOX**.

The 'master' sends the following command:

SOH <add h<add l> STX 'D' '4' ETX <bcc> CR

La récupération des valeurs:

- calcul de checksum
- split_rx (parsing)
- La détection et la récupération de tag
- RSSI (dans notre cas reflected power)
- la température
- la version de firmware

L’exemple de l’envoie: la détection et la récupération de tag

- Cette commande est utilisée pour obtenir une étiquette RFID à partir du lecteur RFID.
- SOH <add h> <add l> ENQ <bcc> CR -> `SOH,ord('F'),ord('F'),ENQ,ENQ,CR`

Bluebox:

- SOH <add h> <add l> STX <code 1 h> <code 1 l> ... <code i h> <code i l> ... <code n h> <code n l> ETX <bcc> CR

```
b'\x01FF\x023000E20051634301019822314549AA0A\x03s\r'
tag = 3000E20051634301019822314549AA0A
```

Une réponse typique

- '\x01FF\x023000200423000000000000000423624D01\x03t\r' en tant que réponse d'un RFID Bluebox
- Décomposition du message binaire
  - \x01 - caractère de début d'en-tête (SOH)
  - FF - données probablement retournées, en hexadécimal
  - \x02 - caractère de début de texte (STX)
  - 3000200423000000000000000423624D01 - charge utile de données, la signification des valeurs n'est pas claire
  - \x03 - caractère de fin de texte (ETX)
  - t - caractère ASCII, signification non claire
  - \r - caractère de retour chariot (CR)

Introduction à l'indicateur de force du signal reçu (RSSI)

Définition : Indicateur de force du signal reçu Communément utilisé pour estimer la distance entre le lecteur RFID et la étiquette

Les difficultés logiciels

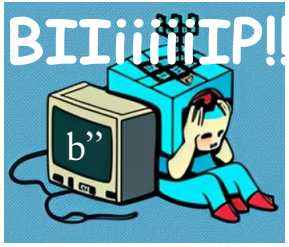
- Suppression des données de la mémoire buffer pour éviter la saturation
- Détermination de la signification de trois fréquences pour le reflected power
  - La plage de fréquences pour la lecture du lecteur RFID
- La détermination de checksum pour chaque commande envoyé
- La manipulation de ROM write
  - e.g activation de la lecture de reflected power (sans succès)
  - La réponse est toujours un NAK pour la lecture du ROM config

Les défis matériels - Facteurs affectant l’estimation de la distance RSSI

- Perte de chemin
- Obstacles
- Estompage multi-chemin
- Bruit
- Gain d'antenne

Impossibilité d'avoir le RSSI ou la sensibilité pour les modèles 5232U !!!

(et aucun réglage du son)



	5121L, 5131L	5121H, 5131H	5121U, 5131U	5221L, 5222L, 5231L, 5232L, 5241L, 5242L	5221H, 5222H, 5231H, 5232H, 5241H, 5242H	5221U-S, 5222U-S, 5237U-S, 5238U-S	5231U, 5232U	5224U, 5225U	532xU, 533xU, 534xU	532xU-RTC, 533xU-RTC, 534xU-RTC	5346U-BL	542xU, 552xU	542xU-RTC, 552xU-RTC	5721U
✖ Read RAM Configuration Parameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Read ROM Configuration Parameters								✓	✓	✓	✓	✓	✓	✓
Sleep Mode														
✔ 'RF Reading' Test			✓			✓	✓		✓	✓	✓	✓	✓	✓
✖ 'RF Power' Test			✓			✓	✓	✓	✓	✓	✓	✓	✓	✓
'RF Sensitivity' Test			✓					✓	✓	✓	✓	✓	✓	✓
✖ Read Reflected Power			✓			✓	✓	✓	✓	✓	✓	✓	✓	✓
Read RSSI Power			✓					✓	✓	✓	✓	✓	✓	✓
Digital Output Activation	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
✔ Read Device Status	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RF Deactivation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RF Activation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Antennas Auto-Tuning							✓		✓	✓	✓	✓	✓	
✔ Buffer Data Request	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Queue Data Request	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Programmation jeu sur Python

- Interaction avec les commandes de scan
- Création d’interface pour scan
- Jeux

Interaction avec les commandes de scan

Sur python le code importe les commandes de scan comme des modules, à partir de là plusieurs applications sont possibles, le langage est assez polyvalent

Accessoires de jeu (tags + PC)

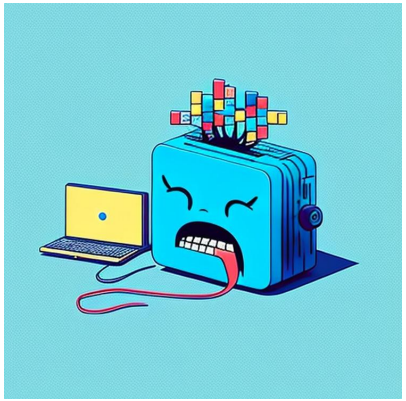
Programme interactif, jeu

```
        with open(players_textfile, "r") as file:
            file_content = file.read()
            players_list = [item.strip() for item in file_content.split(",")]
    except FileNotFoundError:
        with open(players_textfile, 'w') as fp:
            players_list=[]
    return players_list

def calibrate(nbplayers:int):
    #add prints here for prompts
    players={}
    for i in range(0,nbplayers+1):
        pname="player #"+str(i)
        print("player #"+str(i)+", please write your name")
        b = input("$>")
        if (len(b)!=0):
            pname=b
        print( pname, " please pass your tag in front of the antenna short1s")
        tag = capture()
        players.append([pname,tag])
        with open('players.txt', 'w') as fp:
            for item in players:
                # write each item on a new line
                fp.write("%s\n" % item)

    return players

def get_SB_settings():
    Pt = 1.5E
    Gr = 1.64
    Gt = 7.07
    freq = 866.5*10**6
    return Pt,Gr,Gt,freq
'''
def hide_and_seek(players):
    print("press a key when ready to start the timer")
    b = input("$>")
    timer(3)
    caught=[]
    Pt,Gr,Gt,freq= get_SB_settings()
    while len(caught)!=len(players)-1:
        print("tag")
```

Fonctions de capture RFID et contrôle de l’antenne/bluebox

```
#!/usr/bin/python3
import serial
from time import sleep
receiver = serial.Serial('/dev/ttyACM0', 19200)
#receiver = serial.Serial('/dev/ttyUSB3', 115200)
#receiver = serial.Serial('/dev/ttyUSB0', 19200)

# Define the frequency range to EU standards (-doesn't work
freq_1_h = (864 >> 8) & 0xff
freq_1_l =  864 & 0xff
freq_2_h = (865 >> 8) & 0xff
freq_2_l =  865 & 0xff
freq_3_h = (868 >> 8) & 0xff
freq_3_l =  868 & 0xff

#Define channels for ETSI
channel_h = 0x01
channel_l = 0x01

# Define the address of the RFID tag
add_h = 0x00 # ord('F')
add_l = 0x46

# Define the antenna number
antenna = 0x01

# Calculate the BCC value

SOH = 0x01
STX = 0x02
```

Bluebox et antenne

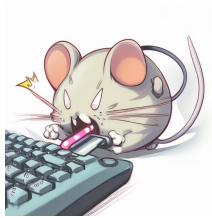


Création d’interface pour scan RFID

```
Welcome to RFIND ME
our players of today:


Arthur with tag: 30003039606283C67080000A18BA6A8A
Guoguo with tag: 3000E2005163430101982231454 9AA0A
Kevin with tag: 300020042300000000000000000423624D
```

la première interface était 100% en ligne de commande au clavier

Ajout progressif d’une interface clickable:



Les fonctions exploitent des formules et des communications avec la bluebox

```
def capture():
    tag=''
    while (tag==''):
        tag=scan.show_first_read_tag()
    return tag
```

```
def evaluate_distance(Pr,Pt,Gr,Gt,freq,eps_r=1,mu_r=1):
    lmbda=(3*10**8)/(freq*mu_r*eps_r)
    R = (lmbda/(4*math.pi)) * math.sqrt((Pt*Gt*Gr)/Pr)
    return(R)
```

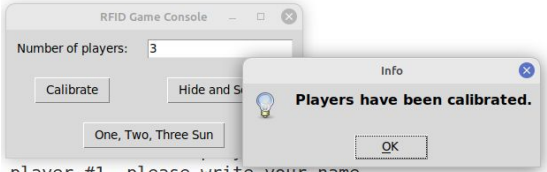
Les mesures sont importées et servent de variables pour le jeu

Jeux

Calibration

Enregistrement des joueurs et leur tag

Sauvegarde du fichier pour reprendre la partie




```
player #1, please write your name
$>Arthur
Arthur  please pass your tag in front of the antenna shortly
player #2, please write your name
$>Guoguo
Guoguo  please pass your tag in front of the antenna shortly
player #3, please write your name
$>Kevin
Kevin  please pass your tag in front of the antenna shortly
Calibration finished
```

Cache cache

Se balader avec l'antenne pour trouver les tags !! 100% fun

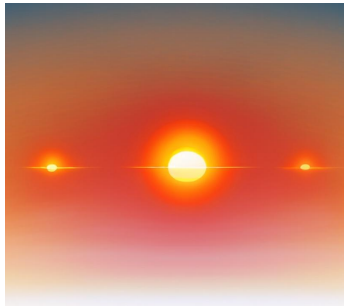
-disponible pour les schtroumpfs surtout-

https://www.youtube.com/shorts/O-weXa6pOCY

1-2-3 soleils

Saurez-vous échapper au regard du scan en cachant votre tag?

-to be implemented-



Cahier des charges

1. Familiarisation avec le matériel à l'aide du logiciel bluebox pour microsoft Windows
   - ✓ 1.1: Essais de capture de tags à plusieurs mètres
   - 1.2: Récolte des données de mesures de puissance du signal si disponible.
   - ✗ 1.3: Premières calibrations (évaluer les puissances à 1m 2m 5m … si 1.2 possible)
   - ✗ 1.4: produire des représentations graphiques des atténuations observées et attendues (selon Friis)
2. Développement d'un code python utilisable sur linux pour contrôler la bluebox
   - ✓ 2.1: Compréhension des codes déjà tout faits disponible
   - ✓ 2.2: Compilation du code et Utilisation pour l'acquisition de signaux de détection des tags
   - ✗ 2.3: Récolte des données de mesures de puissance du signal (si nous n'y arrivons pas le projet devra être changé)
   - ✗ 2.4: Premières calibrations (idéalement nous devrions retrouver les mêmes valeurs qu'avec le logiciel bluebox)
   - 2.5: Reproduire l'étape 1.4
3. ✗ Design de l'aspect ludique
   - ✓ 3.1: Adapter le code pour proposer une interface (au moins un prompt)
   - ✗ 3.2: Décider (et coder) le moyen d'information du programme vis à vis des distances des tags
   - ✓ 3.3: Choisir comment le jeu se joue (tourner l'antenne, bouger/cacher les tags, 1-2-3 soleil etc…)
   - ✓ 3.4: Adapter le code pour les aspect ludiques trouvés
4. ✓ Wrap-Up
   - ✗ 4.1: Premiers essais avec sujets tests
   - ✗ 4.2: Écouter les retours et proposition pour rendre le jeu plus amusant
   - ✓ 4.3: Développer un discours de valorisation du produit
   - ✓ 4.4: Perspectives futures, autres applications possibles.

Difficultés / Limites

Windows:
Statut et Settings de la machine: ETSI, fréquences.
Pas de contrôle du son affreux de la machine.
Pas d'exportation des ID tags vers un fichier/programme
.

Python:
Perte de temps avec les interférences d'import
La documentation indique qu'il est possible de récupérer le RF power et le reflected power avec ce modèle mais les commandes retournent des vides.

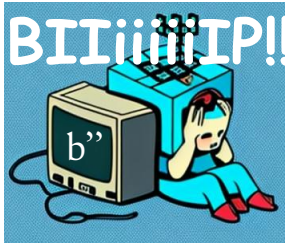
Sans valeurs autre que l'ID, impossible de déterminer les distances

Impossibilité de couper/réduire le son de la BBox -> Torture et voisins qui tappent au plafond

https://demo.hedgedoc.org/s/Up0J0ltCT

Impossibilité d'avoir le RSSI ou la sensibilité pour les modèles 5232U !!!

(et aucun réglage du son)



	5121L, 5131L	5121H, 5131H	5121U, 5131U	5221L, 5222L, 5231L, 5232L, 5241L, 5242L	5221H, 5222H, 5231H, 5232H, 5241H, 5242H	5221U-S, 5222U-S, 5237U-S, 5238U-S	5231U, 5232U	5224U, 5225U	532xU, 533xU, 534xU	532xU-RTC, 533xU-RTC, 534xU-RTC	5346U-BL	542xU, 552xU	542xU-RTC, 552xU-RTC	5721U
✖ Read RAM Configuration Parameters	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Read ROM Configuration Parameters								✓	✓	✓	✓	✓	✓	✓
Sleep Mode														
✔ 'RF Reading' Test			✓			✓	✓		✓	✓	✓	✓	✓	✓
✖ 'RF Power' Test			✓			✓	✓	✓	✓	✓	✓	✓	✓	✓
'RF Sensitivity' Test			✓					✓	✓	✓	✓	✓	✓	✓
✖ Read Reflected Power			✓			✓	✓	✓	✓	✓	✓	✓	✓	✓
Read RSSI Power			✓					✓	✓	✓	✓	✓	✓	✓
Digital Output Activation	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	
✔ Read Device Status	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RF Deactivation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
RF Activation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Antennas Auto-Tuning							✓		✓	✓	✓	✓	✓	
✔ Buffer Data Request	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Queue Data Request	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓