**IRManager**

**IRManager()**

Constructeur qui initialise l’attribut irSender avec l’aide de la classe IRSenderBigBang, qui reçoit l’attribut Application \*lconfig->irPin.

|  |
| --- |
| class IRSenderBitBang : public IRSender |
|  |

|  |
| --- |
| { |
|  |

|  |
| --- |
| public: |
|  |

|  |
| --- |
| IRSenderBitBang(uint8\_t pin); |
|  |

|  |
| --- |
| void setFrequency(int frequency); |
|  |

|  |
| --- |
| void space(int spaceLength); |
|  |

|  |
| --- |
| void mark(int markLength); |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| protected: |
|  |

|  |
| --- |
| int \_halfPeriodicTime; |
|  |

};

**IRManager(Application \*c)**

Constructeur qui initialise la configuration de IRManager par l’application .

**void init(String server, int zone)**

Comme le constructeur, la méthode init initialise le IRSender. Ce constructeur est utilisé pour l’IDE Arduino.

**uint8\_t getHPState()**

Retourne l’état de le pompe à chaleur, 0 si éteint, 1 sinon.

**uint8\_t getHPTemp()**

Retourne la temperature de la pompe à chaleur.

**uint8\_t IRManager::sendIR(String hpmodel, String power, String fmode, String fan, String temp, String vair, String hair)**

Retourne result, qui vaut 1 si l’envoi d’IR marche, 0 sinon.

Si l’utilisateur envoie des valeurs valides, cette méthode initialise un tableau de modèles de pompes à chaleur. Ce tableau est ensuite parcouru : on cherche le modèle de pompe à chaleur entré par l’utilisateur. On fait ensuite appel à la méthode send() qui envoie un signal IR avec les paramètres spécifiés par l’utilisateur.

|  |
| --- |
| IRManager |
| -\*irSender : IRSender  - \*lconfig : Application  - HPState : unsigned int;  -HPTemp : float |
| +getHPState() : uint8\_t  +getHPTemp() : uint8\_t  +IRManager()  +IRManager(Application \*c)  +sendIR(String hpmodel, String power, String fmode, String fan, String temp, String vair, String hair) : uint8\_t  +init() : void |