Firefly Synchronisation System

Generated by Doxygen 1.9.4

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Firefly Class Reference

Public Member Functions

- **Firefly** ()

    Construct a new Firefly object.
- void init (unsigned int minBoost, unsigned int maxBoost, unsigned int period, unsigned int syncWindow)

    Initialise the Firefly object with the necessary clock period, boost range, and sync window.
- void **cycle** ()

    A function that cycles through the firefly's clock and handles the boosting of the clock when LoRa messages are detected. This function also calls for the flashing functions when the end of the clock cycle is reached.
- void setAllowedToStart (bool isAllowedToStart)

    Set the value of the allowedToStart variable.
- bool getAllowedToStart ()

    Get the value of allowedToStart.
- bool isSynced ()

    Return if the firefly has synchronised with its neighbours or not.

3.1.1 Member Function Documentation

3.1.1.1 getAllowedToStart()

```
bool Firefly::getAllowedToStart ( )
```

Get the value of allowedToStart.

Returns

true if allowed to start, false otherwise.

3.1.1.2 init()

```
void Firefly::init (
            unsigned int minBoost,
            unsigned int maxBoost,
            unsigned int period,
            unsigned int syncWindow )
```

Initialise the Firefly object with the necessary clock period, boost range, and sync window.

Parameters

minBoost	The minimum amount to boost the firefly's clock by.
maxBoost	The maximum amount to boost the firefly's clock by.
period	The length of the clock cycle.
syncWindow	The size of the window from the end of the clock cycle that can be determined as the firefly being in sync.

3.1.1.3 isSynced()

```
bool Firefly::isSynced ( )
```

Return if the firefly has synchronised with its neighbours or not.

Returns

true if the firefly has synchronised with its neighbours, false otherwise.

3.1.1.4 setAllowedToStart()

```
void Firefly::setAllowedToStart (
            bool isAllowedToStart )
```

Set the value of the allowedToStart variable.

Parameters

isAllowedToStart	Whether the synchronisation process is allowed to start.

The documentation for this class was generated from the following file:

- Firefly.h

3.2 LoRaComms Class Reference

Public Member Functions

- **LoRaComms** ()

  Construct a new LoRa Comms object.
- void init (int txPower, int syncWord, uint32_t spiFreq, long frequency, int CS)

  Initialises the LoRa module and LoRa library with the specified parameters.
- bool isPacketDetected ()

  Return if a packet is available.
- void broadcastMessage (String message)

  Broadcast a message containing the specified data.

3.2.1 Member Function Documentation

3.2.1.1 broadcastMessage()

```
void LoRaComms::broadcastMessage (
             String message )
```

Broadcast a message containing the specified data.

Parameters

message	The message to broadcast.

3.2.1.2 init()

```
void LoRaComms::init (
             int txPower,
             int syncWord,
             uint32_t spiFreq,
             long frequency,
             int CS )
```

Initialises the LoRa module and LoRa library with the specified parameters.

Parameters

txPower	The LoRa module output power.
syncWord	The sync word used to ensure that packets can only be read from LoRa modules using the same sync word.
spiFreq	The frequency of the SPI communication between the TinyPICO.
frequency	The frequency of the LoRa communication.
CS	The select pin connected from the TinyPICO to the LoRa module.

3.2.1.3 isPacketDetected()

```
bool LoRaComms::isPacketDetected ( )
```

Return if a packet is available.

Returns

true if a message was received, false otherwise.

The documentation for this class was generated from the following file:

- LoRaComms.h

3.3 OTAComms Class Reference

Public Member Functions

- **OTAComms** ()

  Construct a new OTAComms object.
- void init (int otaPort, const char ∗dnsName)

  Initialises the attributes of this class and configures the callback functions to each stage of the over-the-air update process.
- void **begin** ()

  Calls for the ArduinoOTA library to begin.
- void **handle** ()

  Calls for the ArduinoOTA library to handle a request for an over-the-air update.

3.3.1 Member Function Documentation

3.3.1.1 init()

```
void OTAComms::init (
             int otaPort,
             const char * dnsName )
```

Initialises the attributes of this class and configures the callback functions to each stage of the over-the-air update process.

Parameters

otaPort	The port to be used for over-the-air updates.
dnsName	The hostname to be used for over-the-air updates.

The documentation for this class was generated from the following file:

- OTAComms.h

3.4 UDPHandler Class Reference

Public Member Functions

- **UDPHandler** ()

  Construct a new UDPHandler object.
- void init (const char ∗ssid, const char ∗pass, unsigned int port, Firefly ∗firefly)

  Initialises the ESP32 WiFi module and the WiFiUDP object, and creates a pointer to the original Firefly object.
- void **handle** ()

  Responds to incoming messages with the correct akcnowldegement.

3.4.1 Member Function Documentation

3.4.1.1 init()

```
void UDPHandler::init (
            const char * ssid,
            const char * pass,
            unsigned int port,
            Firefly * firefly )
```

Initialises the ESP32 WiFi module and the WiFiUDP object, and creates a pointer to the original Firefly object.

Parameters

ssid	The name of the router to connect to.
pass	The password of the router to connect to.
port	The port to use for the UDP connection.
firefly	A pointer to the Firefly object used to provide status updates to the software platform.

The documentation for this class was generated from the following file:

- UDPHandler.h

Chapter 4

File Documentation

4.1 Firefly.h File Reference

This file handles all the inner workings of the firefly simulation system, including running a clock cycle, flashing an LED, and calling the broadcasting and receiving LoRa messages to and from neighbouring TinyPICOs.

```
#include <TinyPICO.h>
#include "LoRaComms.h"
```

Classes

- class Firefly

4.1.1 Detailed Description

This file handles all the inner workings of the firefly simulation system, including running a clock cycle, flashing an LED, and calling the broadcasting and receiving LoRa messages to and from neighbouring TinyPICOs.

Author

Sean Coaker (seancoaker@gmail.com)

Version

1.0

Date

14-04-2022

Copyright

Copyright (c) 2022

4.2 Firefly.h

Go to the documentation of this file.

```

#include <TinyPICO.h>
#include "LoRaComms.h"

class Firefly {

private:

    bool allowedToStart = false;
    bool syncStarted = false;
    unsigned int packetNotDetectedCounter = 0;
    bool synced = false;
    TinyPICO tp = TinyPICO();

    unsigned int minBoost;
    unsigned int maxBoost;
    unsigned int period;
    unsigned int syncWindow;
    LoRaComms lora;

    void flashAndSendPacket();

    void flashWithoutSendingPacket();

public:

    Firefly();

    void init(unsigned int minBoost, unsigned int maxBoost, unsigned int period, unsigned int
    syncWindow);

    void cycle();

    void setAllowedToStart(bool isAllowedToStart);

    bool getAllowedToStart();

    bool isSynced();

};
```

4.3 LoRaComms.h File Reference

This file contains the LoRaComms class that handles the recieving and broadcasting of LoRa messages.

```
#include <LoRa.h>
```

Classes

- class LoRaComms

4.3.1 Detailed Description

This file contains the LoRaComms class that handles the recieving and broadcasting of LoRa messages.

Author

Sean Coaker (seancoaker@gmail.com)

Version

1.0

Date

14-04-2022

Copyright

Copyright (c) 2022

4.4 LoRaComms.h

Go to the documentation of this file.

```
#include <LoRa.h>

class LoRaComms {

private:

    int txPower;
    int syncWord;
    uint32_t spiFreq;
    long frequency;
    int CS;

public:

    LoRaComms();

    void init(int txPower, int syncWord, uint32_t spiFreq, long frequency, int CS);

    bool isPacketDetected();

    void broadcastMessage(String message);

};
```

4.5 OTA_LoRa_Sync.ino File Reference

A file that handles all the parts to firefly synchronisation and communication with the software platform.

```
#include "OTAComms.h"
#include "UDPHandler.h"
```

Macros

- #define **DNS** "esp3"

  The DNS name of this TinyPICO.
- #define **CS** 5

  The clock select pin used by the LoRa module.
- #define **SSID** "Seans_Router"

  The name of the router to connect to.
- #define **PASSWORD** "daefa5eb2f"

  The password of the router to connect to.
- #define **UDP_PORT** 7375

  The port to use for the UDP connection.
- #define **OTA_PORT** 3232

  The port to use for over-the-air uploads.

Functions

- void **setup** ()

  The Arduino setup function. This function initialises the WiFi connection, the UDP connection, the OTA connection, and the Firefly object.
- void **loop** ()

  The Arduino loop function. This function handles requests for over-the-air uploads, UDP messages and running a cycle of the firefly system.

Variables

- OTAComms **ota**

  An object of OTAComms to provide functionality for over-the-air updates.
- Firefly **firefly**

  An instance of Firefly to simulate firefly synchronisation.
- UDPHandler **udp**

  An object of UDPHandler to provide functionality for UDP communication.

4.5.1 Detailed Description

A file that handles all the parts to firefly synchronisation and communication with the software platform.

Author

Sean Coaker (`seancoaker@gmail.com`)

Version

1.0

Date

14-04-2022

Copyright

Copyright (c) 2022

4.6 OTAComms.h File Reference

A file that handles the over-the-air uploads from the software platform.

```
#include <ArduinoOTA.h>
```

Classes

- class OTAComms

4.6.1 Detailed Description

A file that handles the over-the-air uploads from the software platform.

Author

Sean Coaker (seancoaker@gmail.com)

Version

1.0

Date

14-04-2022

Copyright

Copyright (c) 2022

4.7 OTAComms.h

Go to the documentation of this file.

```
#include <ArduinoOTA.h>

class OTAComms {

private:

    int otaPort;
    const char *dnsName;

public:

    OTAComms();

    void init(int otaPort, const char *dnsName);

    void begin();

    void handle();

};
```

4.8 UDPHandler.h File Reference

This file handles the incoming and outgoing UDP messages. It communicates with the Firefly class to provide status updates to the software platform.

```
#include <WiFi.h>
#include <WiFiUdp.h>
#include "Firefly.h"
```

Classes

- class UDPHandler

4.8.1 Detailed Description

This file handles the incoming and outgoing UDP messages. It communicates with the Firefly class to provide status updates to the software platform.

Author

Sean Coaker (seancoaker@gmail.com)

Version

1.0

Date

14-04-2022

Copyright

Copyright (c) 2022

4.9 UDPHandler.h

Go to the documentation of this file.

```

#include <WiFi.h>
#include <WiFiUdp.h>
#include "Firefly.h"

class UDPHandler {

private:

    const char *ssid;
    const char *password;
    unsigned int port;
    WiFiUDP udp;
    Firefly *firefly;

public:

    UDPHandler();

    void init(const char *ssid, const char *pass, unsigned int port, Firefly *firefly);

    void handle();

};
```