Example Codes

Sender Node

Below is an example code of a node which broadcasts a message to every other node in the mesh network every 10 seconds. A sender node usually behaves as child nodes.

```cpp
#include <painlessMesh.h>

// Mesh network parameters
#define MESH_PREFIX     "yourMeshNetwork"
#define MESH_PASSWORD   "yourMeshPassword"
#define MESH_PORT       5555

painlessMesh mesh;

// FreeRTOS Task Handle for mesh updates
TaskHandle_t meshUpdateTaskHandle;

// Function to handle received messages
void receivedCallback(uint32_t from, String &msg) {
    Serial.printf("Received message from node %u: %s\n", from, msg.c_str());
}

// Task to continuously update the mesh network
void meshUpdateTask(void *pvParameters) {
    while (true) {
        mesh.update();
        vTaskDelay(10 / portTICK_PERIOD_MS);  // Short delay to yield to other tasks
    }
}

void setup() {
    Serial.begin(115200);
```

```
  // Initialize mesh network
  mesh.setDebugMsgTypes(ERROR | STARTUP | CONNECTION);  // Debug message types
  mesh.init(MESH_PREFIX, MESH_PASSWORD, MESH_PORT);
  mesh.onReceive(&receivedCallback);  // Register the message receive callback

  // Create FreeRTOS task for updating mesh
  xTaskCreate(
    meshUpdateTask,           // Function to implement the task
    "MeshUpdateTask",         // Task name
    8192,                     // Stack size
    NULL,                     // Task input parameter
    1,                        // Priority
    &meshUpdateTaskHandle     // Task handle
  );
}


void loop() {

}
```

Receiver Node

Below is an example code of a node which receives messages from every other node in the mesh network that are within range. A receiver node usually behaves as parent nodes.

```
#include <painlessMesh.h>

// Mesh network parameters
#define MESH_PREFIX     "yourMeshNetwork"
#define MESH_PASSWORD   "yourMeshPassword"
#define MESH_PORT       5555

painlessMesh mesh;

// FreeRTOS Task Handle for mesh updates
TaskHandle_t meshUpdateTaskHandle;

// Function to handle received messages
void receivedCallback(uint32_t from, String &msg) {
```

```
    Serial.printf("Received message from node %u: %s\n", from, msg.c_str());
}


// Task to continuously update the mesh network
void meshUpdateTask(void *pvParameters) {
    while (true) {
        mesh.update();
        vTaskDelay(10 / portTICK_PERIOD_MS);  // Short delay to yield to other tasks
    }
}


void setup() {
    Serial.begin(115200);

    // Initialize mesh network
    mesh.setDebugMsgTypes(ERROR | STARTUP | CONNECTION);  // Debug message types
    mesh.init(MESH_PREFIX, MESH_PASSWORD, MESH_PORT);
    mesh.onReceive(&receivedCallback);  // Register the message receive callback

    // Create FreeRTOS task for updating mesh
    xTaskCreate(
        meshUpdateTask,          // Function to implement the task
        "MeshUpdateTask",        // Task name
        8192,                    // Stack size
        NULL,                    // Task input parameter
        1,                       // Priority
        &meshUpdateTaskHandle    // Task handle
    );
}


void loop() {

}
```

Root Node with MQTT Bridge

```
#include <Arduino.h>
#include <painlessMesh.h>
#include <PubSubClient.h>
#include <WiFiClient.h>
```

```cpp
#define   MESH_PREFIX     "whateverYouLike"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT     5555

#define   STATION_SSID     "YourAP_SSID"
#define   STATION_PASSWORD "YourAP_PWD"

#define HOSTNAME "MQTT_Bridge"

// Prototypes
void receivedCallback( const uint32_t &from, const String &msg );
void mqttCallback(char* topic, byte* payload, unsigned int length);

IPAddress getlocalIP();

IPAddress myIP(0,0,0,0);
IPAddress mqttBroker(192, 168, 1, 1);

painlessMesh  mesh;
WiFiClient wifiClient;
PubSubClient mqttClient(mqttBroker, 1883, mqttCallback, wifiClient);

void setup() {
  Serial.begin(115200);

  mesh.setDebugMsgTypes( ERROR | STARTUP | CONNECTION );  // set before init() so that you can see startup
messages

  // Channel set to 6. Make sure to use the same channel for your mesh and for you other
  // network (STATION_SSID)
  mesh.init( MESH_PREFIX, MESH_PASSWORD, MESH_PORT, WIFI_AP_STA, 6 );
  mesh.onReceive(&receivedCallback);

  mesh.stationManual(STATION_SSID, STATION_PASSWORD);
  mesh.setHostname(HOSTNAME);

  // Bridge node, should (in most cases) be a root node. See [the
wiki](https://gitlab.com/painlessMesh/painlessMesh/wikis/Possible-challenges-in-mesh-formation) for some
background
  mesh.setRoot(true);
```

```cpp
  // This node and all other nodes should ideally know the mesh contains a root, so call this on all nodes
  mesh.setContainsRoot(true);
}

void loop() {
  mesh.update();
  mqttClient.loop();

  if(myIP != getlocalIP()){
    myIP = getlocalIP();
    Serial.println("My IP is " + myIP.toString());

    if (mqttClient.connect("painlessMeshClient")) {
      mqttClient.publish("painlessMesh/from/gateway","Ready!");
      mqttClient.subscribe("painlessMesh/to/#");
    }
  }
}

void receivedCallback( const uint32_t &from, const String &msg ) {
  Serial.printf("bridge: Received from %u msg=%s\n", from, msg.c_str());
  String topic = "painlessMesh/from/" + String(from);
  mqttClient.publish(topic.c_str(), msg.c_str());
}

void mqttCallback(char* topic, uint8_t* payload, unsigned int length) {
  char* cleanPayload = (char*)malloc(length+1);
  memcpy(cleanPayload, payload, length);
  cleanPayload[length] = '\0';
  String msg = String(cleanPayload);
  free(cleanPayload);

  String targetStr = String(topic).substring(16);

  if(targetStr == "gateway")
  {
   if(msg == "getNodes")
   {
     auto nodes = mesh.getNodeList(true);
     String str;
     for (auto &&id : nodes)
```

```
      str += String(id) + String(" ");
      mqttClient.publish("painlessMesh/from/gateway", str.c_str());
    }
  }
  else if(targetStr == "broadcast")
  {
    mesh.sendBroadcast(msg);
  }
  else
  {
    uint32_t target = strtoul(targetStr.c_str(), NULL, 10);
    if(mesh.isConnected(target))
    {
      mesh.sendSingle(target, msg);
    }
    else
    {
      mqttClient.publish("painlessMesh/from/gateway", "Client not connected!");
    }
  }
}


IPAddress getlocalIP() {
  return IPAddress(mesh.getStationIP());
}
```