import 'dart:async';

import 'dart:io';

import 'dart:typed\_data';

import 'd\_artnet\_4.dart';

InternetAddress \_deviceToUpgrade;

ArtnetServer \_server;

int \_validDeviceCount = 0;

bool \_upgradeStarted = false;

bool \_firmwareLoaded = false;

bool \_upgradeDone = false;

Uint16List \_firmware;

int \_firmwareLastPos = 0;

int \_firmwareCurPos = 0;

int \_blockNumber = 0;

int \_dataSent = 0;

void main(List<String> arguments) async {

\_server = ArtnetServer(connectionCallback, pollCallback, packetCallback);

\_firmware = Uint16List.view(Uint8List.fromList(await File("./build/Blizzard\_Bridge.bin").readAsBytes()).buffer);

print("Firmware loaded in - Length ${\_firmware.length}");

\_firmwareLoaded = true;

}

void connectionCallback(){

print("We are connected");

}

void pollCallback(){

if(!\_upgradeStarted) print("Sent Poll");

}

void packetCallback(Datagram gram){

if(!\_upgradeStarted){

if(ArtnetGetOpCode(gram.data) == ArtnetPollReplyPacket.opCode){

print("got poll from ${gram.address} $\_validDeviceCount");

if(++\_validDeviceCount >= 3){

\_upgradeStarted = true;

\_deviceToUpgrade = InternetAddress("192.168.1.61");

startOTA();

}

/\*if(gram.address == \_deviceToUpgrade){

if(++\_validDeviceCount >= 3){

\_upgradeStarted = true;

startOTA();

}

} else {

\_deviceToUpgrade = gram.address;

\_validDeviceCount = 1;

}\*/

}

} else {

if(ArtnetGetOpCode(gram.data) == ArtnetFirmwareReplyPacket.opCode){

ArtnetFirmwareReplyPacket reply = ArtnetFirmwareReplyPacket(gram.data);

print("Got firmware Reply - ${reply.blockType}");

if(reply.blockType == 0x00 || reply.blockType == 0x01 && !\_upgradeDone){

\_firmwareLastPos = \_firmwareCurPos;

ArtnetFirmwareMasterPacket more = ArtnetFirmwareMasterPacket();

more.blockType = ArtnetFirmwareMasterPacket.blockTypeOptionFirmCont;

more.firmwareLength = \_firmware.length;

more.blockId = \_blockNumber;

if(\_dataSent + 512 >= \_firmware.length){

print("here!");

more.blockType = ArtnetFirmwareMasterPacket.blockTypeOptionFirmLast;

more.data = \_firmware.toList().sublist(\_firmwareLastPos, \_firmware.length - 1);

print(more);

\_upgradeDone = true;

\_server.sendPacket(more.udpPacket, \_deviceToUpgrade);

\_dataSent+=(\_firmware.length - \_firmwareLastPos);

print("Data sent: $\_dataSent");

exit(0);

} else {

more.data = \_firmware.toList().sublist(\_firmwareLastPos);

\_firmwareCurPos = (++\_blockNumber)\*512;

}

\_server.sendPacket(more.udpPacket, \_deviceToUpgrade);

\_dataSent+=512;

print("Sent $\_firmwareCurPos of ${\_firmware.length} block: $\_blockNumber last pos: $\_firmwareLastPos, data sent: $\_dataSent");

} else if (reply.blockType == 0xFF){

print("ERROR");

exit(0);

}

}

}

}

void startOTA(){

ArtnetFirmwareMasterPacket packet = ArtnetFirmwareMasterPacket();

if(!\_firmwareLoaded){

Timer(Duration(seconds: 1), startOTA);

print("Firmware still loading");

return;

}

print("OTA started");

packet.blockType = ArtnetFirmwareMasterPacket.blockTypeOptionFirmFirst;

packet.firmwareLength = \_firmware.length;

packet.blockId = \_blockNumber;

packet.data = \_firmware.toList();

\_firmwareLastPos = \_firmwareCurPos = 512;

\_blockNumber++;

\_server.sendPacket(packet.udpPacket, \_deviceToUpgrade);

\_dataSent+=512;

print(packet);

print("Sending first OTA packet");

}

class ArtnetServer{

/\*Internals\*/

InternetAddress \_ownIp = InternetAddress.anyIPv4;

Function \_connectionCallback, \_packetCallback, \_pollCallback;

RawDatagramSocket \_socket;

bool \_connected = false;

int \_uuid = 0;

ArtnetServer(this.\_connectionCallback, this.\_pollCallback, this.\_packetCallback){

startServer();

}

void \_handlePacket(RawSocketEvent e){

Datagram gram = \_socket.receive();

var packet;

if (gram == null) return;

if(!ArtnetCheckPacket(gram.data)) return;

\_packetCallback(gram);

}

void startServer(){

if(\_connected) return;

RawDatagramSocket.bind(InternetAddress.anyIPv4, 6454).then((RawDatagramSocket socket){

\_socket = socket;

print('UDP ready to receive');

print('${socket.address.address}:${socket.port} - $\_uuid');

\_connected = true;

\_socket.broadcastEnabled = true;

\_socket.listen(\_handlePacket);

\_connectionCallback();

//Kick off Timers!

\_tick();

});

}

void stopServer(){

if(!\_connected) return;

\_connected = false;

\_socket.close();

}

void sendPacket(List<int> packet,[InternetAddress ip, int port]){

InternetAddress ipToSend = (ip == null) ? InternetAddress("255.255.255.255") : ip;

int portToSend = 6454;

if(\_connected) \_socket.send(packet, ipToSend, portToSend);

}

void \_tick(){

ArtnetPollPacket packet = ArtnetPollPacket();

\_pollCallback();

sendPacket(packet.udpPacket);

if(\_connected){

Timer(Duration(seconds: 1), \_tick);

}

}

List<int> populateOutgoingPollReply(){

ArtnetPollReplyPacket reply = ArtnetPollReplyPacket();

reply.ip = \_ownIp.rawAddress;

reply.port = 0x1936;

reply.versionInfoH = 0;

reply.versionInfoL = 1;

reply.universe = 0;

reply.oemHi = 0x12;

reply.oemLo = 0x51;

reply.ubeaVersion = 0;

reply.status1ProgrammingAuthority = 2;

reply.status1IndicatorState = 2;

reply.estaManHi = 0x01;

reply.estaManLo = 0x04;

reply.shortName = "Blizzard Wizzard";

reply.longName = "Blizzard Wizzard";

reply.nodeReport = "!Enjoy the little things!";

reply.packet.setUint8(ArtnetPollReplyPacket.nodeReportIndex, 0); //Sometimes you have to look for the little things

reply.numPorts = 1;

reply.portTypes[0] = ArtnetPollReplyPacket.portTypesProtocolOptionDMX;

reply.style = ArtnetPollReplyPacket.styleOptionStNode;

reply.status2HasWebConfigurationSupport = true;

reply.status2DHCPCapable = true;

return reply.udpPacket;

}

static String internetAddressToString(InternetAddress address){

var temp = address.rawAddress;

return temp[0].toString() + "." + temp[1].toString() + "." + temp[2].toString() + "." + temp[3].toString();

}

}