# Installation

Download the ZIP file and install it using the normal Arduino IDE installation procedure

# Use in your own sketch

Add an include statement to your sketch:

#include <IoTT\_LocoNetESP32.h>

Define the pin numbers you want to use for transmit and receive:

#define pinRx 22 //pin used to receive LocoNet signals

#define pinTx 23 //pin used to transmit LocoNet signals

Define the logic level your interface is using. Set this to true if you are using inverse logic (most interface circuitry does), meaning your pins are high when LocoNet is low and vice versa.

For a possible interface schematics, see here: <https://github.com/tanner87661/IoTT-Video16/blob/master/Schematic_LocoNet-Interface_LocoNet-Interface-Classic_20190323114544.pdf>

#define InverseLogic true

Define and initialize the LocoNet library by adding it as variable to your sketch:

LocoNetESPSerial lnSerial(pinRx, pinTx, InverseLogic); //true is inverted signals

Make sure you add a processLoop command to your loop() function:

void loop() {

lnSerial.processLoop();

}

Add a callback function to your sketch:

void onLocoNetMessage(lnReceiveBuffer \* newData)

This function is called from the library every time a LocoNet command is received or a communication error occurs. newData has the following structure:

typedef struct {

uint8\_t lnMsgSize = 0; //number of valid data bytes in lnData

uint8\_t lnData[lnMaxMsgSize]; //byte buffer of incoming bytes

uint8\_t errorFlags = 0; //8 flags indicating erros. See below for meaning and value of flags

uint16\_t reqID = 0; //ID of the request that caused this received message

uint32\_t reqRecTime = 0; //the time the request was sent to the library

uint32\_t echoTime = 0; //time in microsecs between request and echo for the same message

uint32\_t reqRespTime = 0; //time in microsecs between request and reply message

} lnReceiveBuffer;

Error Flags meaning and values:

#define errorCollision 0x01

#define errorFrame 0x02

#define errorTimeout 0x04

#define errorCarrierLoss 0x08

#define msgEcho 0x10

#define msgIncomplete 0x20

#define msgXORCheck 0x40

#define msgStrayData 0x80

To send data, enter it into a lnTransmitMsg structure:

typedef struct {

uint8\_t lnMsgSize = 0;

uint8\_t lnData[lnMaxMsgSize];

uint16\_t reqID = 0; //temporarily store reqID while waiting for message to get to head of buffer

uint32\_t reqRecTime = 0;

} lnTransmitMsg;

And send it to the library:

void sendBDInput(uint16\_t bdNum, bool bdStat)

{ lnTransmitMsg myMsg;

myMsg.lnMsgSize = 4;

myMsg.reqID = random(2000);

myMsg.lnData[0] = 0xB2; //B2 1A 50 7

myMsg.lnData[1] = (bdNum & 0x7E) >> 1;

myMsg.lnData[2] = ((uint8\_t)bdStat<<5) | ((bdNum & 0x01)<<4);

myMsg.lnData[3] = myMsg.lnData[0] ^ myMsg.lnData[1] ^ myMsg.lnData[2] ^ 0xFF;

if (lnSerial.carrierOK())

{

int numBytes = lnSerial.lnWriteMsg(myMsg);

Serial.printf("Write %i bytes for ID %i Target %i\n", numBytes, myMsg.reqID, myMsg.lnMsgSize);

}

else

Serial.println("LocoNet not connected");

}

Always make sure you send a valid LcooNet message to the library, meaning correct OpCode, Data bytes, and XOR Check byte.