HOW TO BUILD A DUCK

User Manual | March 2020

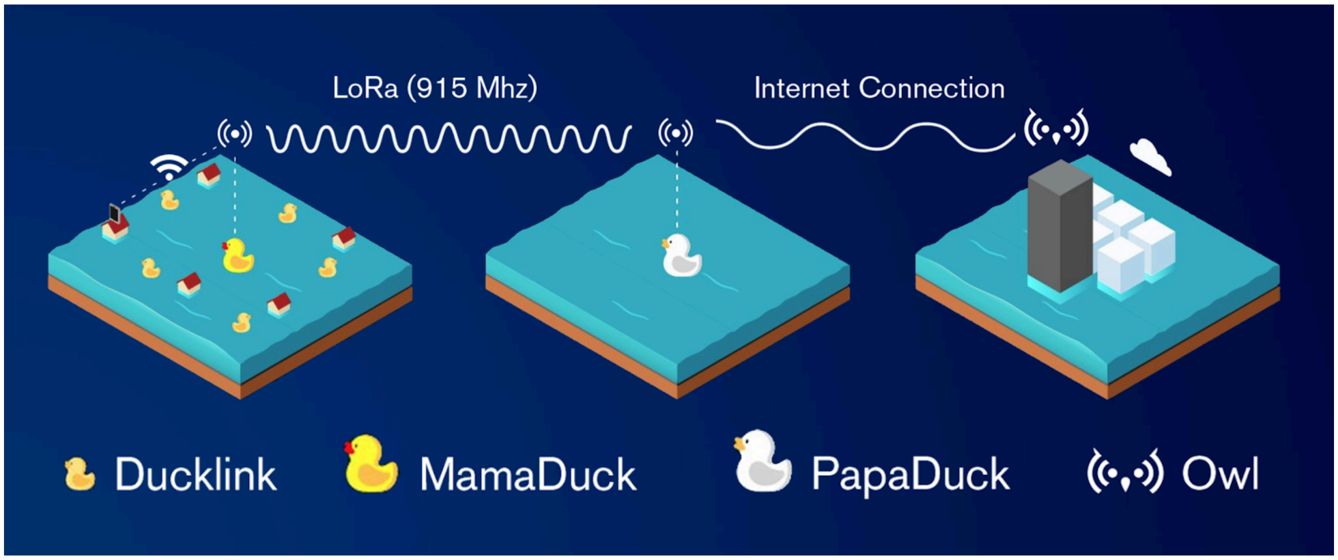
ClusterDuck Protocol V 1.0.2

PROJECT OWL

How to build a duck

Duck Descriptions

Ducks are IoT devices that connect together to form simple mesh networks. The Ducks utilize a combined network of LoRa (Long Range) technology, WiFi, Bluetooth, and sometimes other connectivities. When the Ducks need to communicate with each other they transmit over LoRa, a long-range and power efficient radio protocol. Often a user will need to communicate with the Ducks and may use WiFi. A networked cluster of Ducks - a *ClusterDuck* - is composed of several types of ducks: the DuckLink, MamaDuck, PapaDuck.



DuckLinks

These are the basic nodes of the mesh network. The DuckLinks create a WiFi network where users can connect to it and submit emergencies. The DuckLink collects that data and transmits it to the MamaDuck using LoRa (915 MHz in the United States, 433 MHz in Europe and Asia). Anyone with a working WiFi device such as a smartphone or laptop can connect to a DuckLink.

MamaDucks

MamaDucks act as central hubs to DuckLink groups. The MamaDuck is able to receive data over LoRa from the DuckLinks and transmit this data further into the network. This transmission can occur through other MamaDucks on the way towards the PapaDuck (once again using LoRa). The MamaDuck has most of the same properties as a Ducklink, though small changes in the device firmware help to optimize the architecture of the network.

PapaDuck

PapaDuck is the final Duck in the ClusterDuck and transmits network data to the internet. When communicating with other Ducks the PapaDuck similarly uses LoRa. The data that the PapaDuck receives gets pushed to the OWL Data Management System (DMS), the cloud platform, through the Internet. It acts like a gateway that collects data from MamaDucks and then upload it to OWL.

Raw Materials

Software

You will need the Arduino IDE to install the firmware on your Arduino devices.

The Ducklinks use our ClusterDuck Protocol. Project OWL's open source Duck firmware.

Depending on your computer setup, you may also need USB to UART drivers. If you have any problems, please reach out on the Project OWL Github or in the Project OWL slack.

Hardware

There are many combinations of electronics, batteries, and enclosures that can be used to build a Duck. The following are a recommended minimum set of materials commonly used to create a functional Duck:

- **LoRa Board:** Heltec ESP32 WiFi + LoRa board, such as this
- **Battery:** 3.7v 1.25 Micro JST battery, such as this
- **Data Transfer Cable:** USB Micro connector to USB for your computer, such as this
- **Enclosure:** Plastic Box (or 3D print, or rubber ducky, or anything you want really) if a case is necessary, this would work.

Arduino Software Preparation

Open Arduino IDE. If you do not have this developer environment yet, download the Arduino IDE here

Download USB to UART Bridge VCP Driver from here

Install VCP Driver

MacOS:

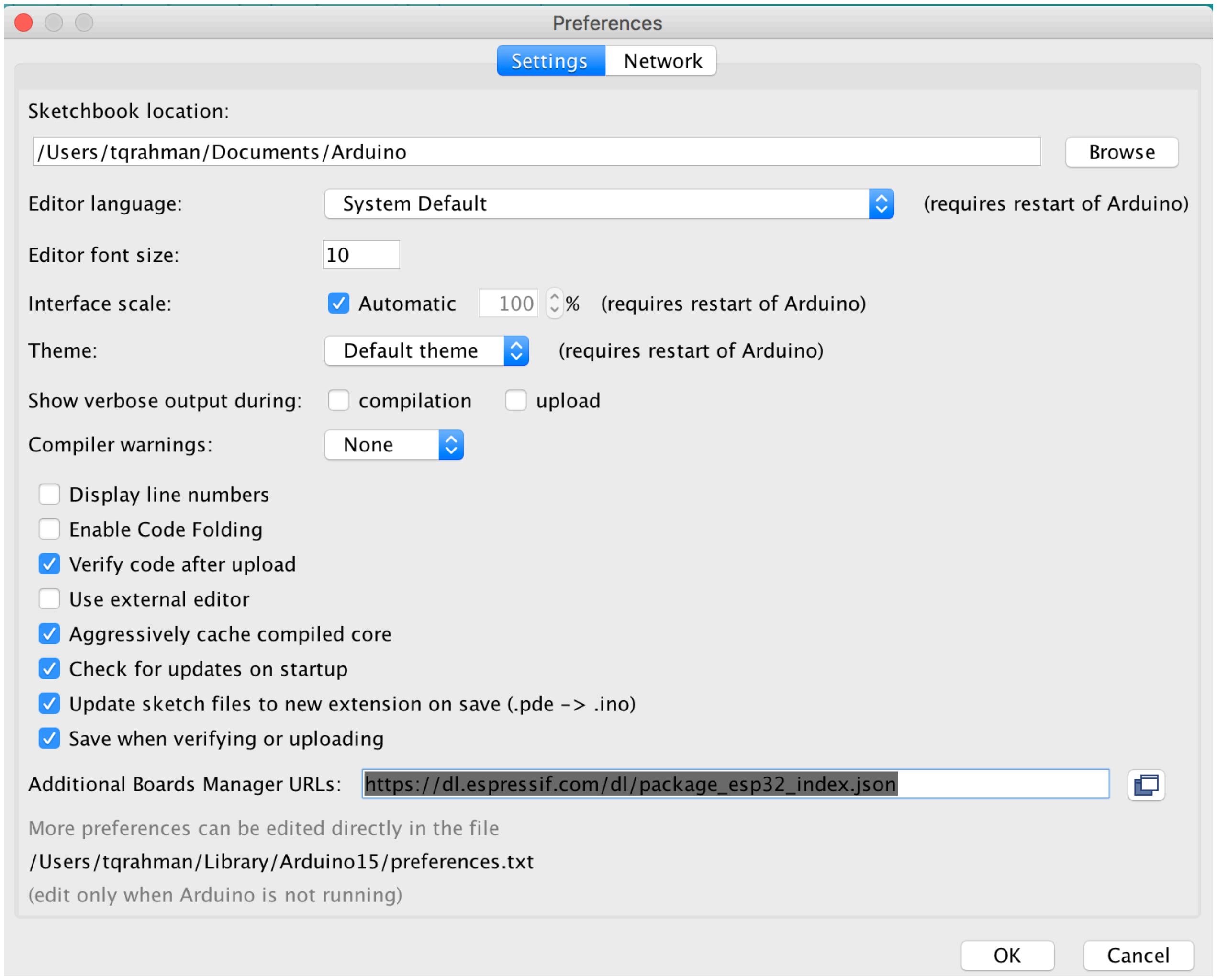
- Make sure you click on the folder that says Legacy MacVCP Driver and then click on ‘Silicon Labs VCP Driver.pkg’
- Once it is finished installing, go to Mac System Preferences -> Security and Privacy -> General. Make sure Silicon Labs is allowed.

Install the necessary libraries that work with ClusterDuck Protocol:

Add ESP32 Board Library

1. Open Arduino
2. Go to Preferences, and in the “Additional Boards Manager URLs” please add the following string:

   https://dl.espressif.com/dl/package_esp32_index.json,https://adafruit.github.io/arduino-board-index/package_adafruit_index.json


Preferences

Settings | Network

Sketchbook location:
/Users/tqrahman/Documents/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 10

Interface scale: ✓ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

- ☐ Display line numbers
- ☐ Enable Code Folding
- ✓ Verify code after upload
- ☐ Use external editor
- ✓ Aggressively cache compiled core
- ✓ Check for updates on startup
- ✓ Update sketch files to new extension on save (.pde -> .ino)
- ✓ Save when verifying or uploading

Additional Boards Manager URLs: https://dl.espressif.com/dl/package_esp32_index.json

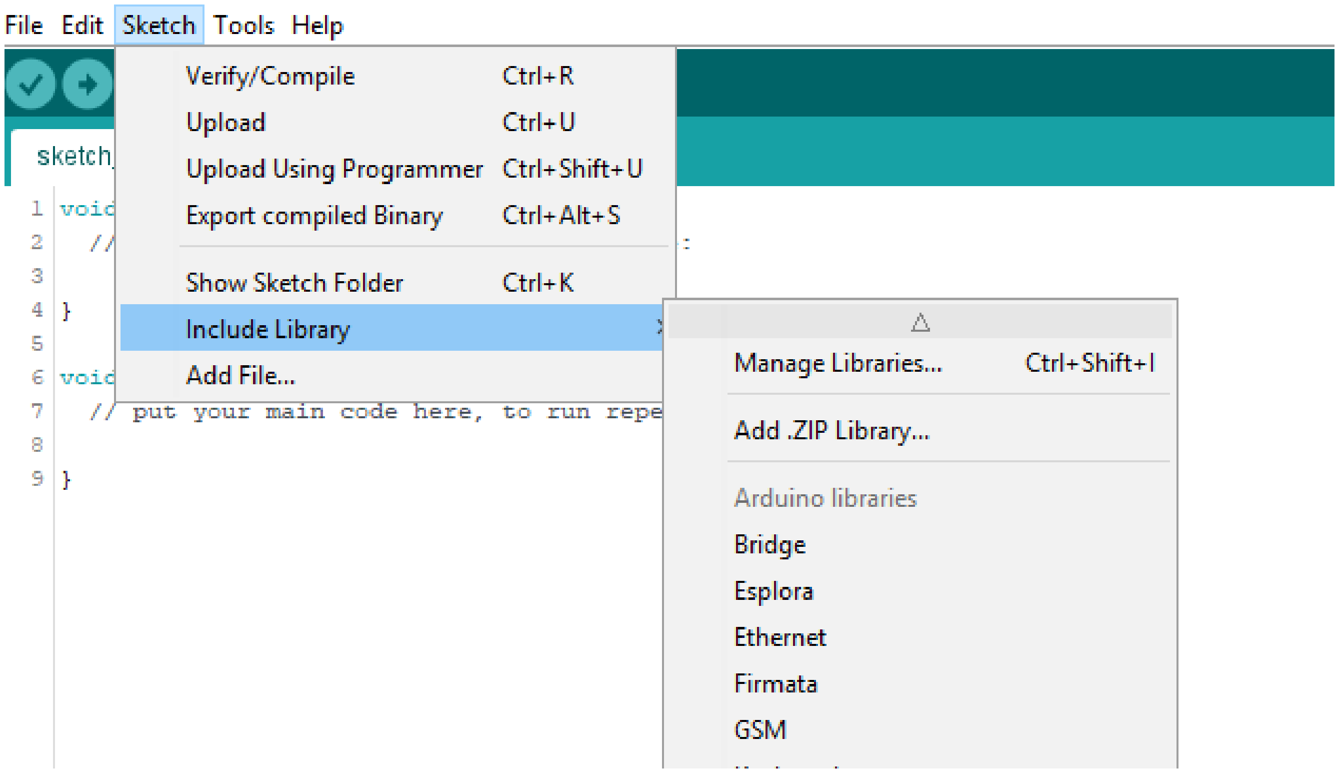
More preferences can be edited directly in the file
/Users/tqrahman/Library/Arduino15/preferences.txt
(edit only when Arduino is not running)

Install Clusterduck Protocol

Using the Library Manager "COMING SOON"

Importing as a .zip Library

In the Arduino IDE, navigate to *Sketch > Include Library > Add .ZIP Library*. At the top of the drop down list, select the option to "Add .ZIP Library".


File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Bridge
Esplora
Ethernet
Firmata
GSM

```
void
  //
}

void
  // put your main code here, to run repe
}
```

Navigate to the downloaded ClusterDuck Protocol Folder and select.

Return to the *Sketch > Include Library menu.* menu. You should now see the library at the bottom of the drop-down menu. It is ready to be used in your sketch. The zip file will have been expanded in the *libraries* folder in your Arduino sketches directory.

Manual Install

You can also copy the library manually into your Arduino libraries folder as follows.

1. Copy ClusterDuck folder (Download Zip or Clone to local machine)
2. Navigate to your Arduino folder. This can be found in your default Documents folder.
3. Navigate to the library folder
4. Paste into library folder
5. Restart Arduino
6. You should now be able to see examples by going to *File -> Examples -> ClusterDuck*

You should be able pull new commits directly to this folder in your Arduino library.

Add required libraries

The required libraries to work with the ClusterDuck protocol can be installed in multiple ways. There is a folder in the source code that is called "Libraries" and contains all the required libraries to work with the ClusterDuck Protocol. Or you can download all the libraries manual and import into Arduino by .zip. Some libraries are available in the Arduino Libraries manager.

Installing libraries by included libraries folder.

In the source code there is a folder with the required libraries you will need to copy and paste these libraries into the arduino Libraries folder so they can be found.

Go to your copy of the ClusterDuck Protocol source code and find the Libraries folder. Copy all the libraries in that folder. Navigate to your Arduino folder. This is usually found in your *Documents* for windows *C:/Users/USER/Documents/Arduino.* for MacOS */Users/USER/Documents/Arduino/*. In the Arduino Libraries folder drag or paste all the libraries.

Installing libraries through Library Manager

Some Libraries can be installed with the Arduino Library manager.

Go to *Sketch > Include Library > Manage Libraries*:

- Search for LoRa and install *LoRa by Sandeep Mistry*
- Search for u8g2 and install *u8g2 by Oliver*
- Search for ArduinoJson and install *ArduinoJson by Benoit Blanchon*

Installing libraries by adding .Zip files

To add Libraries by zip files. Go to *Sketch > include Library > add .zip Library…*

Go to the following websites and download the Libraries:

- adafruit/Adafruit-BMP085-Library: A powerful but easy to use BMP085/BMP180 Library
- adafruit/Adafruit_BMP280_Library: Arduino Library for BMP280 sensors
- https://arduinojson.org/v6/doc/installation/ Version 6
- me-no-dev/AsyncTCP: Async TCP Library for ESP32
- adafruit/DHT-sensor-library: Arduino library for DHT11, DHT22, etc Temperature & Humidity Sensors
- https://github.com/ThingPulse/esp8266-oled-ssd1306
- https://github.com/me-no-dev/ESPAsyncWebServer
- https://github.com/FastLED/FastLED
- https://github.com/sandeepmistry/arduino-LoRa
- https://github.com/swatish17/MQ7-Library
- https://github.com/knolleary/pubsubclient
- https://github.com/olikraus/u8g2
- https://github.com/mkarawacki/wavesharesharpdustsensor
- adafruit/Adafruit_Sensor: Common sensor library
- Arduino-timer Version 2.0.1
- espressif/arduino-esp32: Arduino core for the ESP32

Load the Heltec ESP32 Board to your Arduino IDE:

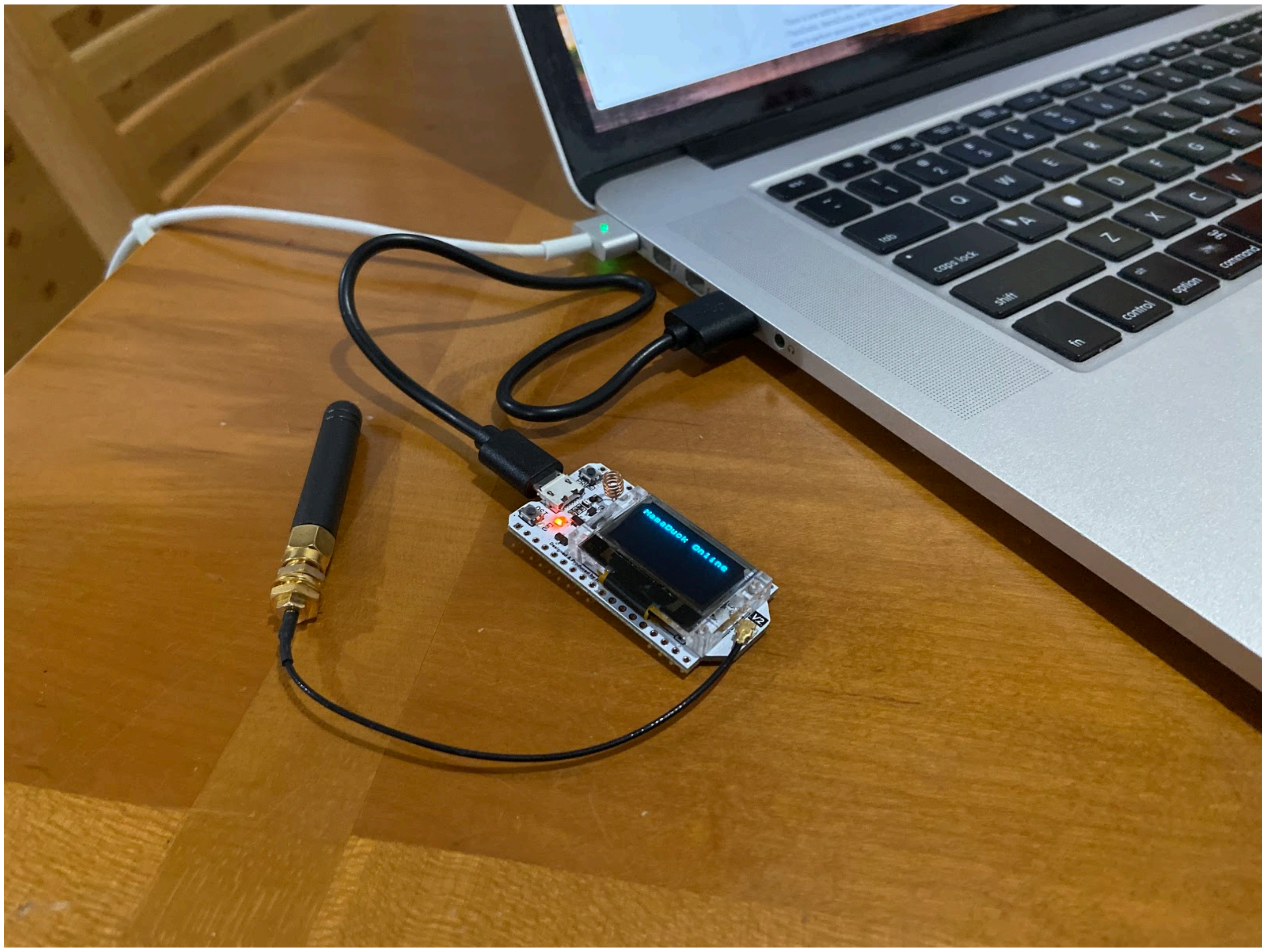
In Arduino IDE, select *Tools > Board > Boards Manager*, and in the pop up window type “esp32” into the search field. You should see the “esp32 by Espressif Systems” library. Install this library.

Once the board library is installed, you are ready to use the Arduino IDE.

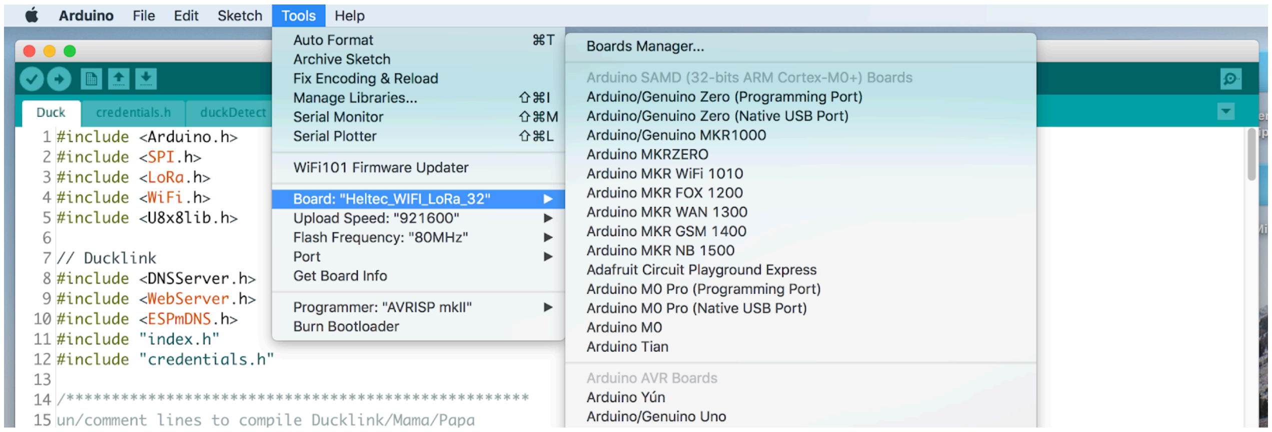
Install The firmware

DuckLink Hardware and Firmware Assembly

1. Connect ESP32 board to your computer via USB cable.



2. In Arduino IDE, select Tools > “board” to Heltec_WiFi_LoRa_32



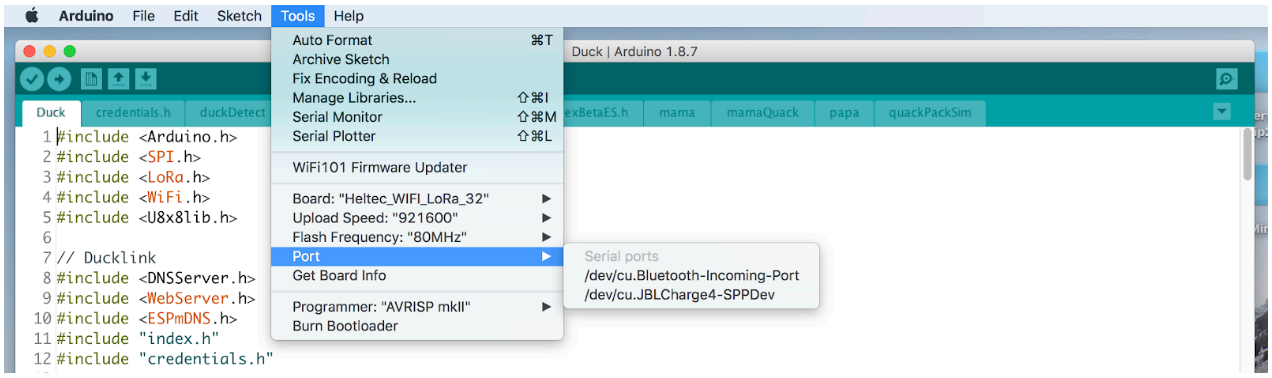
```
Board: "Heltec_WIFI_LoRa_32"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Programmer: "AVRISP mkII"
```

```
#include <Arduino.h>
#include <SPI.h>
#include <LoRa.h>
#include <WiFi.h>
#include <U8x8lib.h>

// Ducklink
#include <DNSServer.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include "index.h"
#include "credentials.h"

/***************************************************
un/comment lines to compile Ducklink/Mama/Papa
```

3. In Arduino IDE, select Tools > “port” to

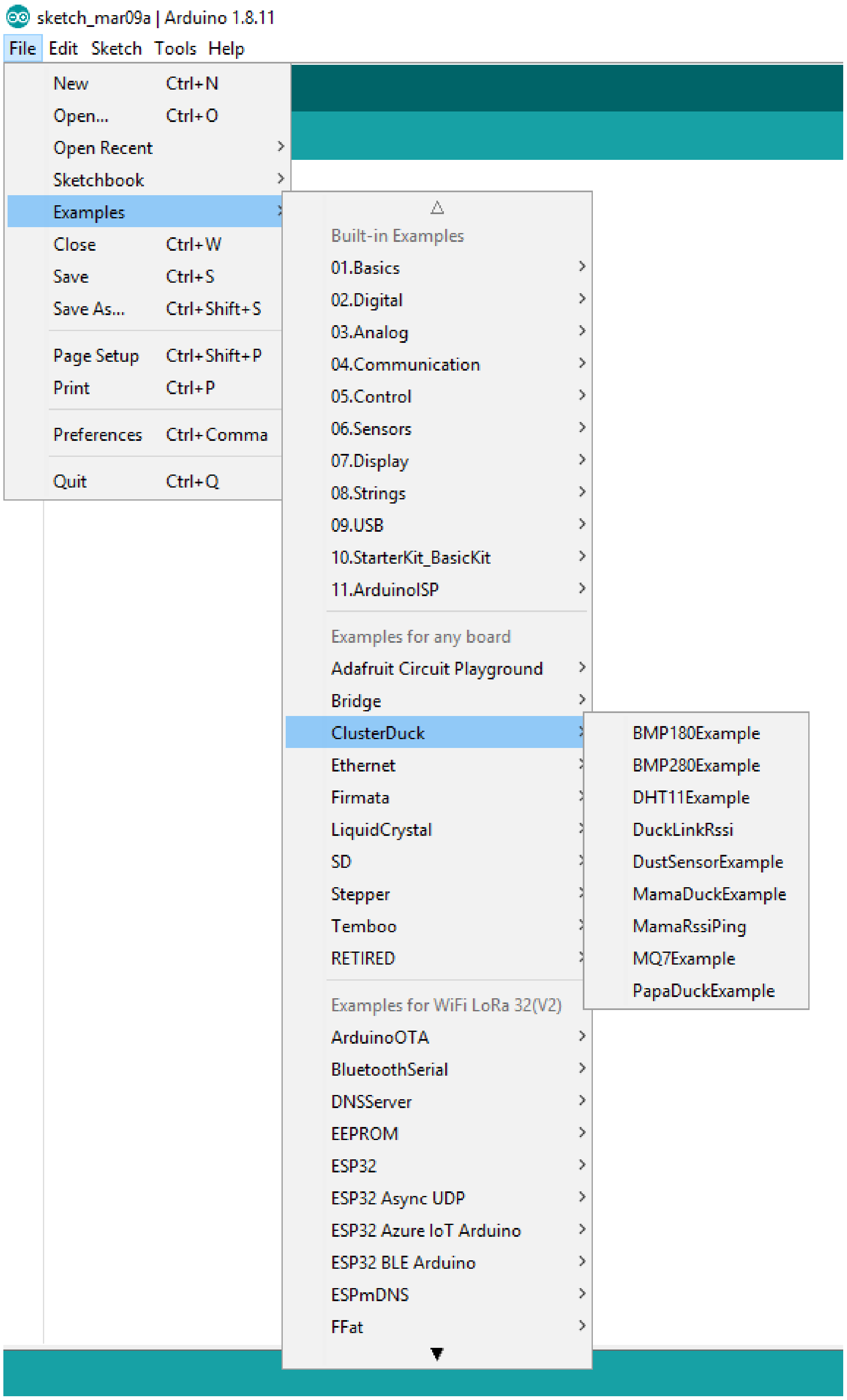


```
#include <Arduino.h>
#include <SPI.h>
#include <LoRa.h>
#include <WiFi.h>
#include <U8x8lib.h>

// Ducklink
#include <DNSServer.h>
#include <WebServer.h>
#include <ESPmDNS.h>
#include "index.h"
#include "credentials.h"
```

4. Select Duck:

   At this point you will need to choose what kind of Duck you want to make. There are Examples for different Ducks included in the ClusterDuck Protocol. Go to File -> Examples -> Clusterduck.


sketch_mar09a | Arduino 1.8.11
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
Built-in Examples
01.Basics
02.Digital
03.Analog
04.Communication
05.Control
06.Sensors
07.Display
08.Strings
09.USB
10.StarterKit_BasicKit
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
ClusterDuck
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for WiFi LoRa 32(V2)
ArduinoOTA
BluetoothSerial
DNSServer
EEPROM
ESP32
ESP32 Async UDP
ESP32 Azure IoT Arduino
ESP32 BLE Arduino
ESPmDNS
FFat
BMP180Example
BMP280Example
DHT11Example
DuckLinkRssi
DustSensorExample
MamaDuckExample
MamaRssiPing
MQ7Example
PapaDuckExample

Choose an Example File

Note: every Clusterduck network needs at least 1 Mama and 1 Papa

MamaDuckEcample: Choose a DuckID and modify the code for your needs for api references and more explanation see down below or go to GitHub

PaPaDuckExample: For the papa you need to setup a WiFi connection

```
#define SSID        ""
#define PASSWORD    ""
```

Setup your SSID and Password between the “” in the code.

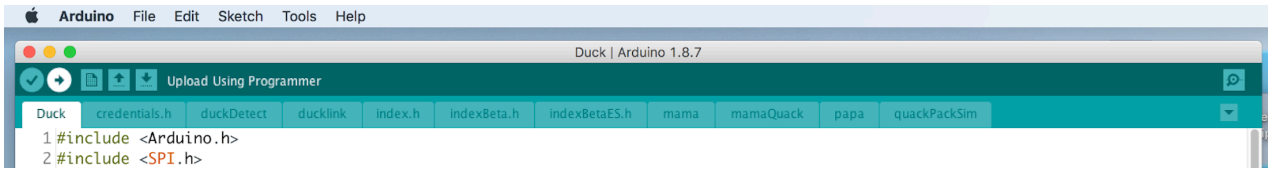
You can setup your own MQTT server or connect to the OWL DMS (Coming Soon!) here:

```
#define ORG         ""          // "quickstart" or use your organisation
#define DEVICE_ID   ""
#define DEVICE_TYPE "PAPA"
#define TOKEN       ""

char server[]           = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[]            = "iot-2/evt/status/fmt/json";
char authMethod[]       = "use-token-auth";
char token[]            = TOKEN;
char clientId[]         = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
```

5. Upload the firmware

   Finally, upload this code to the Duck device by hitting the right-pointing arrow in the top left corner.



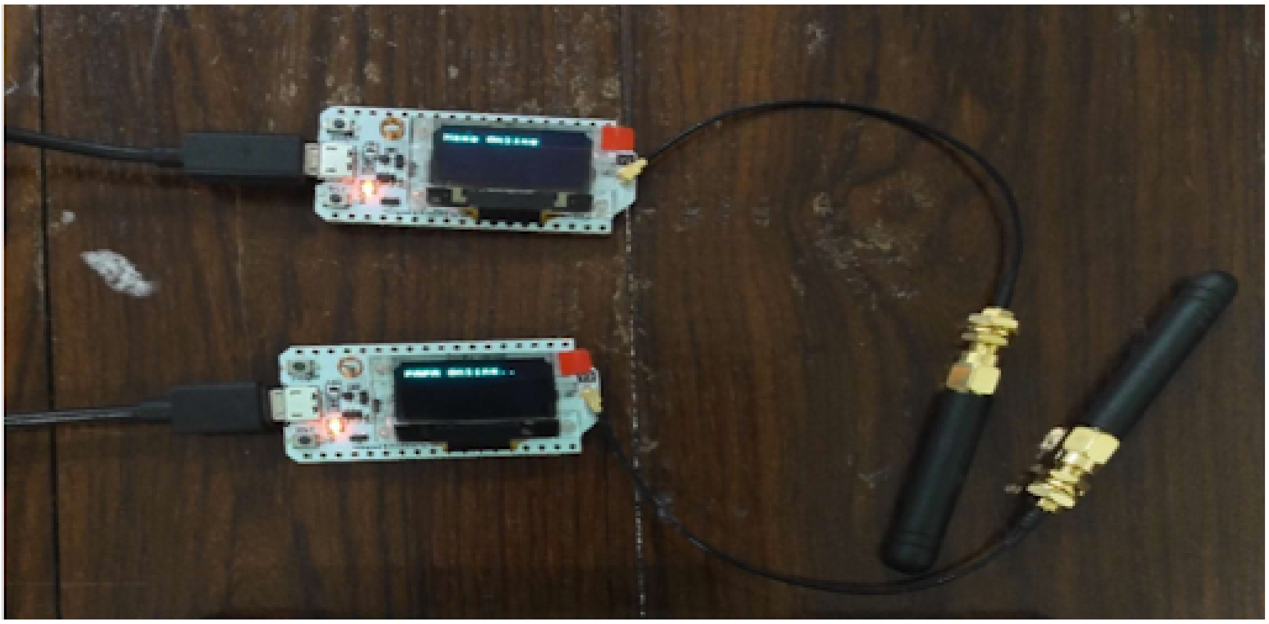
```
#include <Arduino.h>
#include <SPI.h>
```

This will show a lot of activity in the console, and if successful will say complete. If this process fails for any reason, contact us and we can help debug whatever is going wrong.

Disconnect the USB cable and connect the battery to the board.

If everything worked properly, the OLED screen on the board may light up and you should see the Wifi network available if you check wifi settings on your phone or computer.

Finished Duck:



FAQs

General

How many ducks do I need to set-up to test if it works?

If you only want to test if it works, you need at least two ducks, a papa duck and mama duck. You can have a point to point network with the protocol with just 2 ducks. Adding more ducks will extend the network.

Errors when compiling

fatal error: PubSubClient.h: No such file or directory

The cause of this error because the library is missing. To install the library, go to Sketch -> Include Library -> Manage Libraries. In the search box, type PubSub. Find and install PubSubClient by Nick O'Leary.

Multiple libraries were found for "WiFi.h"

This issue occurred because when the program tried to compile, it came across two "Wifi.h" files and was not sure which one to choose. The Wifi.h that is needed is

Setup Issues

Board not displaying in Port

Make sure that the LoRa board is connected to the computer using a DATA TRANSFER CABLE. Not all cables are created equal. One of the cables transfer data from computer to computer. The other cable just charges devices but cannot transfer/receive data. Make sure you are using a data transfer cable.

Cloud Errors

Data not uploading to the internet

Double check to see if the credentials (network and password) for WiFi are correctly inputted in the credentials.h file.

API Reference And Quickstart CLusterDuck Protocol

CLUSTERDUCK PROTOCOL ™

Github

Quick Start

Open new sketch in Adruino IDE and include the ClusterDuck library

```
#include "ClusterDuck.h"
```

Create ClusterDuck object

```
ClusterDuck duck;
```

Initializes the ClusterDuck class object

In setup()

```
duck.begin(baudRate);
```

Initializes the baude rate for serial printing and messaging. You can adust to your desired baude rate.

- int baudRate – Default is 115000

Set device ID and captive portal form length.

```
duck.setDeviceId(String deviceId, const int formLength);
```

- String deviceId – input the device ID used to identify your registered device on the web – do not leave null or empty string
- const int formLength – (optional) define the number of captive portal form fields – Default is 10 to match our default captive portal template

Setup DuckLink

`duck.setupDuckLink();`

or

`duck.setupMamaDuck`

can also be used here to setup a MamaDuck, however you cannot use both in the same sketch.

In loop()

Add corresponding Duck run code. Must be of the same device type as used in `setup()`. (e.g. if `duck.setupMamaDuck()` is used in `setup()` use `duck.runMamaDuck()`)

`duck.runDuckLink();`

Your sketch should look something like this:

```
#include "ClusterDuck.h"
ClusterDuck duck;

void setup() {
// put your setup code here, to run once:
    duck.begin();
    duck.setDeviceId("Z", 10);
    duck.setupDuckLink();
}

void loop() {
// put your main code here, to run repeatedly:
    duck.runDuckLink();
}
```

Now compile and upload to your device. If using a Heltec LoRa ESP32 board you should see a Duck Online message on the LED screen. You can now open your phone or laptop's wifi preferences and connect to the `SOS DuckLink Network`!

API

`setDeviceId(String deviceId, const int formLength)`

- Set device ID and captive portal form length. Do not leave deviceId *null* or as an empty string. formLength defaults to 10. Use in `setup()`.

`void begin(int baudRate)`

- Initialize baude rate for serial. Use in `setup()`.

`void setupDisplay(String deviceType)`

- Initializes LED screen on Heltec LoRa ESP32 and configures it to show status, device ID, and the device type. Use in `setup()`.

`void setupLoRa(long BAND, int SS, int RST, int DIO, int TxPower)`

- Initializes LoRa radio. If using Heltec LoRa ESP32 set SS to , RST to and DIO to . TxPower corresponds to the the transmit power of the radio (max value: 20). Use in `setup()`.

`void setupPortal(const char *AP)`

- Initializes the captive portal code. *AP is the value that will be displayed when accessing the wifi settings of devices such as smartphones and laptops. Use in `setup()`.

`bool runCaptivePortal()`

- Processes requests coming through the captive portal. Returns `true` if there is a new submission. Use this in your `loop()` function to run continuously.

`void setupDuckLink()`

- Template for setting up a DuckLink device. Use in `setup()`

`void runDuckLink()`

- Template for running core functionality of a DuckLink. Use in `loop()`.

`void setupMamaDuck()`

- Template for setting up a MamaDuck device. Use in `setup()`.

`void runMamaDuck()`

- Template for running core functionality of a MamaDuck. Use in `loop()`.

`String * getPortalDataArray()`

- Returns webserver arguments based on formLength as an array of `Strings`.

`String getPortalDataString()`

- Returns webserver arguments based on formLength as a single String with arguments separated by *

`void sendPayloadMessage(String msg)`

- Packages msg into a LoRa packet and sends over LoRa. Will automatically set the current device's ID as the sender ID and create a UUID for the message.

`void sendPayloadStandard(String msg, String senderId = "", String messageId = "", String path = "")`

- Similar to and might replace `sendPayloadMessage()`. senderId is the ID of the originator of the message. messageId is the UUID of the message. ms is the message payload to be sent. path is the recorded pathway of the message and is used as a check to prevent the device from sending multiple of the same message.

`void couple(byte byteCode, String outgoing)`

- Writes data to LoRa packet. outgoing is the payload data to be sent. byteCode is paired with the outgoing so it can be used to identify data on an individual level. Reference `setDeviceId()` for byte codes. In addition it writes the outgoing length to the LoRa packet.
- Use between a `LoRa.beginPacket()` and `LoRa.endPacket()` (note: `LoRa.endPacket()` will send the LoRa packet)

`String readMessages(byte mLength)`

- Returns a String. Used after `LoRa.read()` to convert LoRa packet into a String.

`bool checkPath(String path)`

- Checks if the path contains deviceId. Returns bool.

`String * getPacketData(int pSize)`

- Called to iterate through received LoRa packet and return data as an array of Strings.

- Note: if using standerd byte codes it will store senderId, messageId, payload, and path in a Packet object. This can be accessed using `getLastPacket()`

`void restartDuck()`

- If using the ESP32 architecture, calling this function will reboot the device.

`void reboot(void *)`

- Used to call `restartDuck()` when using a timer

`void imAlive(void *)`

- Used to send a '1' over LoRa on a timer to signify the device is still on and functional.

`String duckMac(boolean format)`

- Returns the MAC address of the device. Using `true` as an argument will return the MAC address formatted using ':'

`String uuidCreator()`

- Returns as String of 8 random characters and numbers

`String getDeviceId()`

- Returns the device ID

`Packet getLastPacket()`

- Returns a Packet object containing senderId, messageId, payload, and path of last packet received.
- Note: values are updated after running `getPacketData()`