令和元年度卒業論文

魚群探知機と水域ネットワークを用いた定置網漁業の効率化

令和 2 年 2 月 5 日

学籍番号：153022
氏名：榊原 萌

指導教官：田房 友典

弓削商船高等専門学校情報工学科

<目次>

第1章　　はじめに

近年，世界の水産資源への関心が日に日に高まっている．欧米での健康志向の高まりや，中国等の経済発展により，世界の食用水産物消費量は年々増加を続けており，世界の 1 人当たりの年間水産物消費量は，約 50 年間で 2 倍に増加している．[1]しかし，水産資源の需要が高まるにつれ消費量も増え，水産資源が枯渇していくことが問題視されている．今後の水産資源の需要を支えると予想されている養殖業にも，養殖適地に限りがあること，収容できる魚の密度に限りがあること，魚粉を中心とした餌の供給にも限界があることなどの要因から，限界がある可能性がある．

水産資源の保護のために，持続可能な漁法が注目されている．現在よく行われている底引き網漁などの持続不可能な方法は，魚を必要以上に獲りすぎてしまうなど，水産資源が枯渇する原因となっているためである．水産資源は，自然の再生産システムの中で産卵，成長，世代交代が行われ，適切な量の漁獲を行えば永続的な利用が可能になるという性質を持っている．増加分だけを漁獲することで永続的に利用することが可能である．[1]

昔から日本で行われている持続可能な漁法に，定置網漁法がある．定置網漁法とは，海上の一定の場所に定置網を設置し，定期的に網起こしを行って，網に滞留した魚を獲るという漁法である．この方法では魚を獲りすぎることはなく，水産資源の保護に繋がる．しかし，網起こしの前に網にいる魚の数や種類が分からないため，無駄な手間や費用を費やしてしまい，場合によっては損失を与えてしまう可能性がある．よって，定置網漁法は環境によいが効率が悪い漁法だと言える．そこで，陸上から定置網の状態が確認できれば，定置網に魚が多くかかっているときに漁をすることで効率的に利益を得ることができるため，定置網漁業が効率化できると考えた．

本研究では，海上の定置網上に魚群探知機を設置し，そのデータを陸上からモニタリングするシステムを開発する．当初は省電力無線 LAN の代わりに XBee wi-fi を使用する予定であった．しかし，より遠隔に wi-fi 接続するためには，多くの電力量が必要となり，海上で長時間の運用は困難である．そこで，システムの構成を画像転送に特化したものに変更した．魚群探知機の様子をカメラで撮影し，より省電力で稼働する XBee wi-fi で送信する．陸上ではカメラから送られた画像を処理し，定置網のデータをモニタリングする．今年度は，カメラからの画像を取得し，XBee wi-fi を通して片方の PC に送信するプログラムを開発した．

第2章　　システム構成

本章では，昨年度のシステムとその問題点，問題点を解決した新しいシステムについて説明し，2 つのシステムを比較する．ここでは，昨年度のシステムを旧システム，今年度に開発したシステムを新システムとして扱う．

2.1 旧システムの構成

旧システムの構成図を図 1 に示す．2 台の魚群探知機を，省電力無線 LAN を用いて接続する．無線LAN を通してデータを共有し，陸上と海上で同じデータを表示させることで，定置網をモニタリングする仕組みになっている．

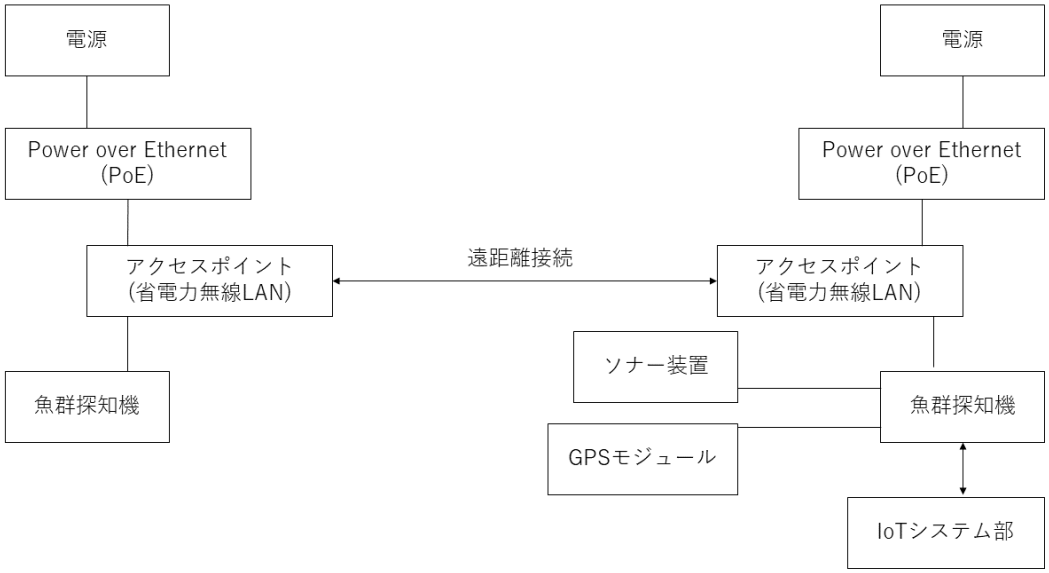

図 1　旧システムの構成図

2.2 旧システムの課題

旧システムには，消費電力が大きく実用的ではないという問題点があった．システムを稼働させるには約 700W の電源が必要である．使用するバッテリー1 台では約 1 時間しか稼働しないため，複数のバッテリーを用いなければならない．また，バッテリーの充電にはソーラーパネルを用いる．日照時間を 5 時間とし，発電能力 30W，1 日平均充電電流 4.14Ah のソーラーパネルを使用すると仮定すると，バッテリー1 台を完全に充電するにはソーラーパネルが 7 枚必要になる．複数のバッテリーを使用するため，ソーラーパネルもその分必要になる．よって，システムの規模も大きくなり，実用的ではない．さらに，魚群探知機の価格も，1 台が約 20 万円と高額であるため，導入しにくいことも問題点である．

これらの問題点を解決するため，システムの構成を大幅に変更することにした．

2.3 新システムの構成

図 2 に新システムの構成図を示す．図中の実線はケーブル等で直接接続されている部分を，点線は無線接続されている部分を示す．魚群探知機の画面をカメラで撮影し，XBee wi-fi を通して陸上の PC に送信する．陸上では送られてきた画像を処理することで定置網をモニタリングする．

旧システムからの変更点として，モニタリングに必要な魚群探知機が 1 台になり，XBee wi-fi と PC を無線で同じネットワークに接続した．また，現在は海上側に PC を使用しているが，Raspberry Pi と USB カメラを用いることでさらに省電力で稼働させることができるため，実装時にはそちらを用いる．

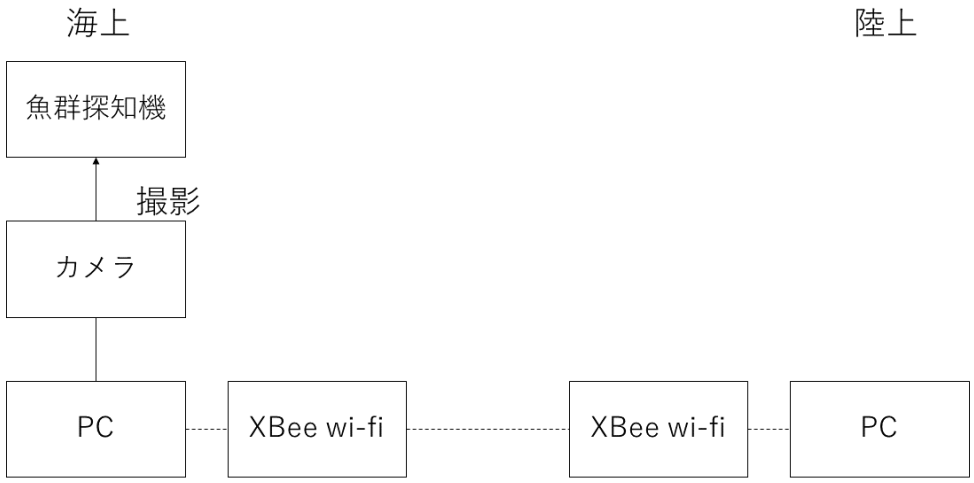

図 2　新システムの構成図

2.4 システムの比較

2つのシステムを，必要な電力，バッテリー1台での稼働時間，必要な魚群探知機の台数の3点で比較したものを表1に示す．新システムの消費電力は，実装時を想定しRaspberry Piの消費電力を用いて計算した．新システムの消費電力は約510Wである．魚群探知機の消費電力が500Wのため，それ以外の消費電力がかなり小さくなったことがわかる．さらに，消費電力が小さくなったことにより，バッテリー1台で動く時間が旧システムに比べて2倍に伸びた．よって，旧システムの課題であった消費電力の問題を解決できたと考えられる．さらに，消費電力が下がったことで，システムの稼働に必要なバッテリーやその充電に必要なソーラーパネルの台数も減り，設備も旧システムに比べて小規模になる．また，必要な魚群探知機が1台になったため，導入に必要な価格も下がった．よって，旧システムより導入もしやすくなった．

これらの点から，新システムは旧システムに比べて実用的なシステムになったと言える．

表1　新システムと旧システムの比較

比較項目	旧システム	新システム
電源に必要な電力	約700W	約510W ※Raspberry Piの消費電力で計算
バッテリー1台で動く時間	約1時間	約2時間
必要な魚群探知機	2台	1台

第3章　　XBee wi-fi

本章では，システムのネットワーク構築に用いた XBee wi-fi の概要，Xbee wi-fi の設定方法について説明する．

3.1 XBee wi-fi の概要

XBee wi-fi とは，ディジ インターナショナルから発売されている wi-fi モジュールである．XBee シリーズの特徴として，他の無線通信に比べてスリープ時の消費電力が少なく，スリープからの復帰が早いことが挙げられる．これらの特徴から，一定時間だけ起動させて通信を行い，残りの時間はスリープさせるといった使用方法に向いている．また，低速通信でもあるため，センサからのデータなど，小さいデータの送受信に適している．

XBee wi-fi は，通信モードを変更することで無線アクセスポイントとして使用できる．これにより，スマートフォンやタブレットなどの wi-fi 対応デバイスからのデータを収集・送信することが可能である[2]．さらに，DeviceCloud 機能を搭載しており，クラウドでデバイスを管理することもできる．短所として，通常の Xbee より消費電力が大きいこと，wi-fi ネットワークの知識がないと導入が難しいことが挙げられる．

また，XBee wi-fi は屋外で最大 1500m 間の通信が行える．本システムは上島町近海での使用を想定しているため，1km 前後あれば十分な通信距離が確保できていると考えた．

3.2 XBee wi-fi の設定

XBee wi-fi の設定は，X-CTU という XBee シリーズ専用のソフトウェアで行う．X-CTU は，ディジ インターナショナルの公式サイトから無料でダウンロードできる．X-CTU を起動した際の画面を図 3 に示す．また，今回変更した項目を表 1 に示す．画面の左側に現在 PC に接続されている XBee wi-fi が表示される．右側には，設定できる項目が表示される．これらの項目を目的に合わせて変更することで設定を行う．

今回変更した項目は，SSID，通信方法，IP アドレス，通信プロトコルである．本システムでは，XBee wi-fi を無線のアクセスポイントとして利用するため，通信方法を Soft AP モードに設定した．

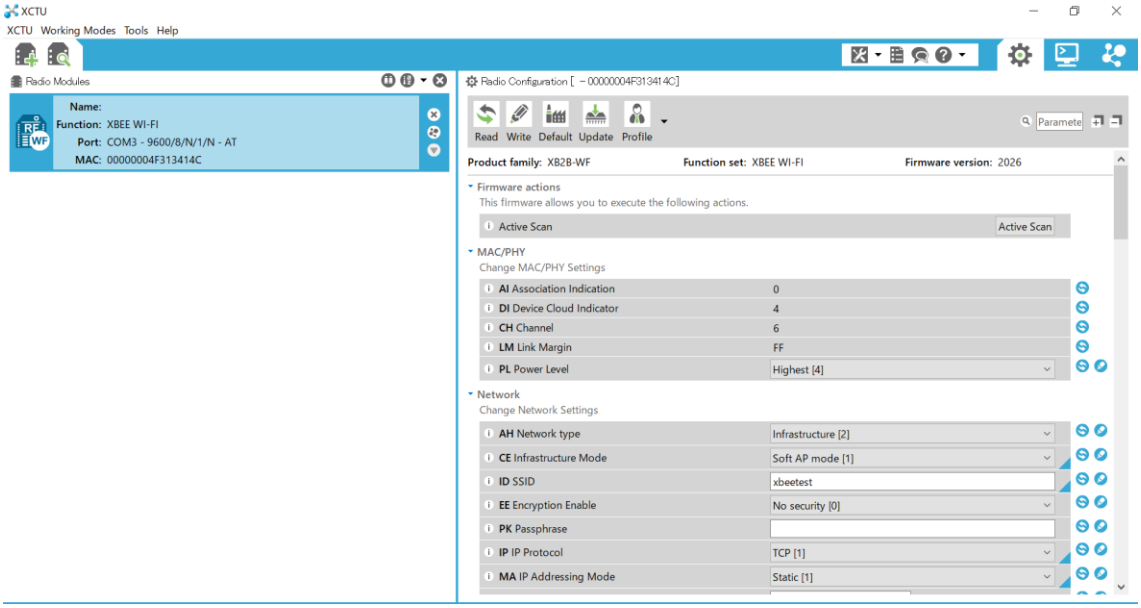

図 3　X-CTU を起動した際の画面

表 2　変更した項目

変更した項目	モジュール 1	モジュール 2
SSID	Xbeetest	Xbeetest
通信方法	Soft AP モード	Soft AP モード
IP アドレス	192.168.0.12/24	192.168.0.13/24
通信プロトコル	TCP	TCP

第4章　　画像の転送

本章では，カメラで撮影した画像を転送するプログラムについて説明する．

本研究では，画像を転送するためのプログラムを Python と OpenCV で記述した．通信は socket を，画像の処理は OpenCV を用いる．

プログラムの流れを図 4 に示す．プログラムは送信側と受信側に分かれている．送信側では，まずカメラからの画像をフレームごとに取得する．取得した画像を送信できるように文字列データに変換し，socket を用いて受信側に送信する．受信側では，受信した文字列データを画像データに再変換し，PC の画面に表示させる．なお，今回は画像処理を行わず，画像のみを送信する．画像データを表示させる理由は，情報のみを送信するより視覚的に分かりやすく，画像を見て漁へ行くか判断できるからである．また，画像が取得できれば，後に画像処理などを使った活用がしやすいことも理由として挙げられる．

使用した PC の設定項目を表 2 に示す．プログラム中で IP アドレスが指定しやすいように，IP アドレスの割り当て方式を手動にし，それぞれの PC に IP アドレスを設定した．

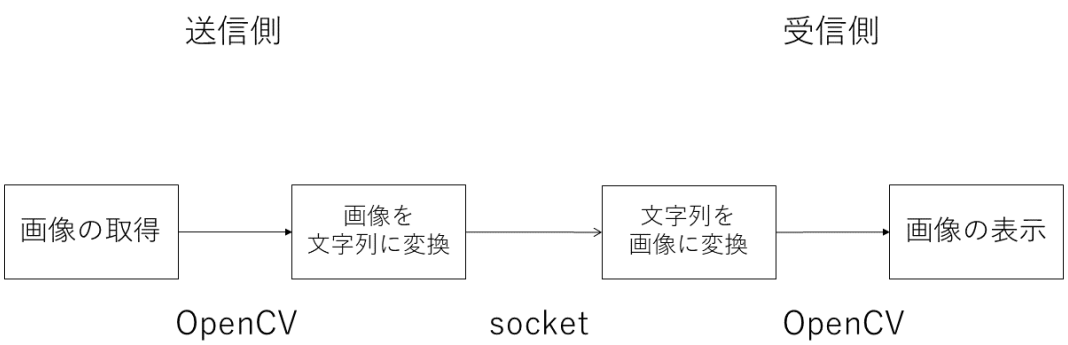

図 4　プログラムの流れ

表 3　PC の設定項目

設定した項目	送信側 PC	受信側 PC
割り当て方式	手動	手動
IP アドレス	192.168.0.14/24	192.168.0.15/24

4.1 送信側のプログラム

送信側のプログラムでは，画像の取得，文字列データへの変換，データの送信を行う．

プログラムが起動されると，カメラの画像を取得し，文字列データへと変換する．その後，接続待ち状態になる．受信側から接続要求があると，接続を確立し変換した文字列を送信する．受信側と接続が確立した後は常に接続されているが，受信側で通信を切る操作がされると，接続を閉じ，プログラムが終了される．

リスト 1 に，画像を送信するまでのプログラムの一部を示す．また，送信側のプログラムの全体を付録 1 に示す．カメラからの画像の取得を videoCap.read()で行い，imencode で取得した画像を文字列に変換する．

```
import socketserver
import cv2
import numpy as np
class TCPHandler(socketserver.BaseRequestHandler):
  #read するたびにフレームを取得
    ret, frame = videoCap.read()
  #文字列に変換
    jpegsByte = cv2.imencode('.jpg', frame, encode_param)[1].tostring()
    self.request.send(jpegsByte)
```

リスト 1　送信側のプログラム(一部)

4.2 受信側のプログラム

受信側のプログラムは，文字列データの受信，画像データへの変換，画像の表示を行う．

プログラムが起動されると，送信側に接続要求を行う．接続が確立されると受信した文字列を画像に変換し，表示させる．また，キーボードの q キーを押すことで通信を終了させることができる．

リスト 2 に，画像を受信し表示させるプログラムの一部を示す．また，受信側のプログラムの全体を付録 2 に示す．送信側の PC から送信されてきた文字列の受信には sock.recv を用いる．そして，送られてきた文字列を OpenCV で処理できるように numpy 形式に格納し，imdecode で画像データに再変換する．再変換された画像は，imshow で q キーを押すまで表示される．

```
import socket
import numpy
import cv2

def getimage():
   sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
   sock.connect((HOST,PORT))
   #文字列の受信
   while recvlen > 0:
      receivedstr = sock.recv(1024*8)
   sock.close()
   #画像にデコード
   narray = numpy.fromstring(buf, dtype = 'uint8')
   return cv2.imdecode(narray, 1)

While True:
   #画像の表示
   img = getimage()
   cv2.imshow('Capture', img)
```

リスト 2　受信側のプログラム(一部)

第5章　おわりに

XBee wi-fi と魚群探知機を使って，魚群探知機の画像を用いて定置網の様子をモニタリングするシステムを開発した．昨年度のシステムと違い，消費電力が小さく，魚群探知機が 1 台あればモニタリングを行うことができる．

このシステムは，元々2 台の魚群探知機に無線 LAN を通して同じデータを表示させ，陸上から定置網をモニタリングできるようにするという構成であった．2 台の魚群探知機に搭載されている無線機能の IP アドレスが同じで，変更することができなかった．そのため，省電力無線 LAN の代わりに XBee wi-fi を用いた際，2 台を同時にネットワークに接続することが出来なかったと考えられる．これを考慮した結果，魚群探知機の様子をカメラで撮影し，撮影した画像を XBee wi-fi で陸上に転送する方式に変更した．

本システムの問題点として，消費電力が大きいという点があった．このシステムは，Raspberry Pi と USB カメラを用いることで，魚群探知機以外をモバイルバッテリーで動作させることができる．よって，全体的な消費電力を減らすことができる．

今年度中に実現できなかったこととして，画像処理を用いてのモニタリング，画像の通信速度や実際の通信距離の調査，PC を Raspberry Pi に置き換えての実装の 3 点が挙げられる．1 点目は，画像処理を用いて魚群探知機の画面を処理することで，魚がどれだけいるかなどの情報を画像と同時に送信することができるため，導入を検討している．2 点目は，実際の環境でこのシステムが満足に動作するのかの確認ができていないため，システムが完成してからの実験を検討している．3 点目は，まだプログラムを PC で実行しているため，実際のシステムに導入する予定の Raspberry Pi での動作確認ができていない．これらは，来年度以降の課題として引き続き調査していく．

参考文献

[1]特集　私たちの水産資源｜農林水産省
https://www.jfa.maff.go.jp/j/kikaku/wpaper/h22/pdf/h22_hakusyo5_2.pdf
[2]XBee wi-fi｜ディジ　インターナショナル
http://www.digi-intl.co.jp/products/wireless-wired-embedded-solutions/zigbee-rf-modules/zigbee-mesh-module/xbee-wi-fi.html

付録 1　送信側のプログラム

```
import socketserver
import cv2
import numpy as np
class TCPHandler(socketserver.BaseRequestHandler):
    videoCap = ''
    #リクエストを受け取るたびに呼ばれる関数
    def handle(self):
        self.data = self.request.recv(1024).strip()
    #read するたびにフレームを取得
        ret, frame = videoCap.read()
    #jpeg の圧縮率　0~100 値が高いほど高品質 何も指定しなければ 95
        encode_param = [int(cv2.IMWRITE_JPEG_QUALITY), 100]
    #文字列に変換
        jpegsByte = cv2.imencode('.jpg', frame, encode_param)[1].tostring()
        self.request.send(jpegsByte)
#送信元の設定
HOST = "192.168.0.14"
PORT = 8080
#画像の設定
videoCap = cv2.VideoCapture(0)
if not videoCap:
    print("画像が開けませんでした")
    sys.exit()
socketserver.TCPServer.allow_reuse_address = True
server = socketserver.TCPServer((HOST, PORT), TCPHandler)
try:
    server.serve_forever()
except KeyboardInterrupt:
    pass
server.shutdown()
sys.exit()
```

付録２　受信側のプログラム

```
import socket
import numpy
import cv2

def getimage():
    #送信先の IP アドレス
    HOST = "192.168.0.14"
    PORT = 8080
    sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    sock.connect((HOST,PORT))
    sock.send(b'test')
    buf = b''
    recvlen = 100
    #文字列の受信
    while recvlen > 0:
        receivedstr = sock.recv(1024*8)
        recvlen = len(receivedstr)
        buf += receivedstr
    sock.close()
    #画像にデコード
    narray = numpy.fromstring(buf, dtype = 'uint8')
    return cv2.imdecode(narray, 1)

While True:
    img = getimage()
    cv2.imshow('Capture', img)
    #q を入力して終了
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```