Submarine

잠수함 만들기

- Start, Stop Shell
- 메인프로그램 SubMarine.py
- USB 카메라 프로그램 (Thread)
- PI 카메라 프로그램 (Thread)
- 메인 화면 index.html

Start, Stop Shell

start shell

```
#!/bin/bash

echo
echo '##### START Submarine  #####'
echo

wpa_cli -i wlan0 status
echo
echo

FILENAME=/home/hyunsu/submarine/nohup.out
if [ -f "$FILENAME" ] ; then
        echo "nohup.out delete !"
        rm /home/hyunsu/submarine/nohup.out
else
        echo "file not exist"
fi

echo '##### START CCTV #####'
cd /home/hyunsu/submarine
nohup python3 /home/hyunsu/submarine/SubMarine.py &
sleep 3

echo '##### START BROD SOUND #####'
hwString=`pactl list | grep -A 40 'Source' | grep -A 20 'device.bus = "usb"' | grep
'device.string' | grep -o '[0-9]'`
echo -e "\n Mic HW = [$hwString]"

nohup cvlc -vvv alsa://plughw:$hwString --sout
'#transcode{acodec=mp3,ab=64,channels=1}:standard{access=http,dst=0.0.0.0:8080/
out.mp3}' 1> /dev/null 2>&1 &

echo
echo '##### STARTed Submarine  #####'
echo
```

Stop shell

```
#!/bin/bash

echo
```

```
echo '##### STOP  Submarine #####'
echo


ps -ef | grep SubMarine.py | grep -v grep
KILLPID=`ps -ef | grep SubMarine.py | grep -v grep | awk '{print($2)}'`
echo "Stop Submarine Process = " $KILLPID
kill -9 $KILLPID
sleep 3

ps -ef | grep vlc | grep -v grep
KILLPID=`ps -ef | grep vlc | grep -v grep | awk '{print($2)}'`
echo "Stop Brod Sound Process = " $KILLPID
kill -9 $KILLPID
sleep 3


echo
echo '##### STOPed Submarine #####'
echo
```

메인프로그램 SubMarine.py

SubMarine.py

메인 프로그램

```
from flask import Flask, render_template, send_from_directory, Response, send_file,
request, redirect, url_for
from flask_socketio import SocketIO
import argparse, logging, logging.config, conf
import os
from power import PowerStatus
from datetime import datetime
from commChecker import CommChecker
from gpioControl import GpioControl
from usbwebcamera import UsbWebCamera
from piwebcamera import PiWebCamera


app = Flask(__name__)
app.config['SECRET_KEY'] = '1764121q'
socketio = SocketIO(app)

logging.config.dictConfig(conf.dictConfig)
logger = logging.getLogger(__name__)

power = PowerStatus()
commChecker = CommChecker()
gpio = GpioControl()

usbcamera = UsbWebCamera(20,1)
usbcamera.run()

piwebcamera = PiWebCamera()
piwebcamera.run()

@app.after_request
def add_header(r):
    """
    Add headers to both force latest IE rendering or Chrome Frame,
    and also to cache the rendered page for 10 minutes
    """
    r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    r.headers["Pragma"] = "no-cache"
    r.headers["Expires"] = "0"
    r.headers["Cache-Control"] = "public, max-age=0"
```

```
    return r

@app.route("/")
@app.route("/index.html")
def index():
    logger.debug("Requested /")
    return render_template("index.html")

@app.route("/favorit.ico")
def favorit_ico():
    logger.debug("Requested favorit.ico image")
    filename = "favorit.ico"
    return send_file(filename)

@app.route("/VideoStatus")
def videostatus():
    logger.debug("Video Status change ")
    content = usbcamera.status()
    return Response(content, mimetype='text/xml')

@app.route("/SubmStatus")
def submStatus():
    logger.debug("Submarine Status change ")
    content = piwebcamera.status()
    return Response(content, mimetype='text/xml')

@app.route("/temperature")
def temperature():
    content = os.popen("vcgencmd measure_temp").readline()
    content = content.replace("temp=", "")
    powerstatus = power.getPowerStatus()
    return Response("내부온도:" + content+"  , 전원:"+powerstatus+" , 측정시
간:"+str(datetime.now()), mimetype='text/xml')

@app.route("/commCheck")
def commcheck():
    connstatus = str(commChecker.checkQuality())
    return Response(connstatus+" , "+str(datetime.now()), mimetype='text/xml')

@app.route('/lightOnOff', methods=['GET', 'POST'])
def lightOnOff():
    gpio.light()
    logger.debug('message KK Fired !!!')
    return Response('Light ON', mimetype='text/html')

''' ##############   MOVE    #############  '''

@app.route("/floating", methods=['GET', 'POST'])
def floating():
    if request.method == 'POST':
```

```
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.Floating(int(moveOrder))
        print("Move floating ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/diving", methods=['GET', 'POST'])
def diving():
    if request.method == 'POST':
        moveOrder = request.form['moveCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.Diving(int(moveOrder))
        print("Move diving ..." + moveOrder)
    return Response(reString, mimetype='text/html')



@app.route("/forward", methods=['GET', 'POST'])
def forward():
    if request.method == 'POST':
        moveOrder = request.form['moveFBLFCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.Forward(int(moveOrder))
        print("Move Forward ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/backward", methods=['GET', 'POST'])
def backward():
    if request.method == 'POST':
        moveOrder = request.form['moveFBLFCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.Backward(int(moveOrder))
        print("Move Backward ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/right", methods=['GET', 'POST'])
def right():
    if request.method == 'POST':
        moveOrder = request.form['moveFBLFCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
```

```
            reString = gpio.TurnLeftF(int(moveOrder))
        print("Move Right ..." + moveOrder)
    return Response(reString, mimetype='text/html')

@app.route("/left", methods=['GET', 'POST'])
def left():
    if request.method == 'POST':
        moveOrder = request.form['moveFBLFCnt']
        if not moveOrder :
            reString = "Do not move"
        else :
            reString = gpio.TurnRightF(int(moveOrder))
        print("Move Left ..." + moveOrder)
    return Response(reString, mimetype='text/html')


@app.route("/video_usb_feed")
def video_usb_feed():
    return Response(genusb(usbcamera),
        mimetype="multipart/x-mixed-replace; boundary=frame")

def genusb(usbcamera):
    logger.debug("Starting USB stream")
    while True:
        frame = usbcamera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/png\r\n\r\n' + frame + b'\r\n')


@app.route("/video_pi_snapshot")
def video_pi_snapshot():
    return Response(genpi(piwebcamera),
        mimetype="multipart/x-mixed-replace; boundary=frame")

def genpi(piwebcamera):
    logger.debug("Starting PI capchure")
    while True:
        frame = piwebcamera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/png\r\n\r\n' + frame + b'\r\n')


if __name__=="__main__":
    logger.debug("Starting Submarine start")
#    socketio.run(app, log_output=True, host='0.0.0.0', port=8081, debug=True,
use_reloader=False, allow_unsafe_werkzeug=True)
    socketio.run(app, log_output=True, host='0.0.0.0', port=8081,
allow_unsafe_werkzeug=True)
```

USB 카메라 프로그램 (Thread)

USB 카메라 프로그램 (Thread)

```
import cv2
import threading
import time
import logging
import os

logger = logging.getLogger(__name__)
archive_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'archive')

thread = None
status = True

class UsbWebCamera:

    def __init__(self,fps=20,video_source=0):
        logger.info("Initializing usb camera class with {fps} " + str(fps) + " and
video_source={" + str(video_source) + "}")
        self.fps = fps
        self.video_source = video_source
        self.camera = cv2.VideoCapture(self.video_source)

        self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

        self.max_frames = 5*self.fps
        self.frames = []
        self.isrunning = False

    def run(self):
        logging.debug("Perparing Usb Camera thread")
        global thread
        if thread is None:
            logging.debug("Creating thread")
            thread = threading.Thread(target=self._capture_loop,daemon=True)
            logger.debug("Starting thread")
            self.isrunning = True
            thread.start()
            logger.info("Thread started")

    def _capture_loop(self):
        global status
        dt = 1/self.fps
```

```
        logger.debug("Observation started")
        while self.isrunning:
            if(status):
                v,im = self.camera.read()
                im = cv2.flip(im, 1)
                #im = cv2.flip(im, 0)
                if v:
                    if len(self.frames)==self.max_frames:
                        self.frames = self.frames[1:]
                    self.frames.append(im)
            time.sleep(dt)
        logger.info("Thread stopped successfully")

    def status(self):
        global status
        logger.debug("Status isrunning" + str(status))
        if(status):
            content = "STOP THREAD !"
            status = False
        else:
            content = "START THREAD !"
            status = True
        return content

    def get_frame(self, _bytes=True):
        if len(self.frames)>0:
            if _bytes:
                img = cv2.imencode('.png',self.frames[-1])[1].tobytes()
            else:
                img = self.frames[-1]
        else:
            logger.info("Change cam source 1")
            self.video_source = 1
            self.camera = cv2.VideoCapture(self.video_source)
            self.camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
            self.camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
            with open(archive_path+"/not_found.jpeg","rb") as f:
                img = f.read()
        return img
```

PI 카메라 프로그램 (Thread)

PI 카메라 프로그램 (Thread)

```
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2
import threading
import time
import logging
import os

logger = logging.getLogger(__name__)
archive_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'archive')

thread = None
status = False

# Set up camera constants
#IM_WIDTH = 1024
#IM_HEIGHT = 768
IM_WIDTH = 640
IM_HEIGHT = 480

class PiWebCamera:
    def __init__(self,fps=20):
        logger.info(f"Initializing pi camera class with {fps} fps ")
        self.fps = fps
        self.camera = PiCamera(framerate=24)
        self.camera.resolution = (IM_WIDTH,IM_HEIGHT)
        # self.camera.rotation = 180
        self.rawCapture = PiRGBArray(self.camera, size=(IM_WIDTH,IM_HEIGHT))
        self.rawCapture.truncate(0)
        self.max_frames = 5*self.fps
        self.frames = []
        self.isrunning = False

    def run(self):
        logging.debug("Perparing Pi Camera thread")
        global thread
        if thread is None:
            logging.debug("Creating PI thread")
            thread = threading.Thread(target=self._capture_loop,daemon=True)
            logger.debug("Starting PI thread")
            self.isrunning = True
```

```
            thread.start()
            logger.info("Thread PI started")

    def _capture_loop(self):
        global status
        dt = 1/self.fps
        logger.debug("Observation started")
        while self.isrunning:
            if(status):
                self.camera.capture(self.rawCapture, format="bgr",
use_video_port=True)
                self.rawCapture.truncate(0)
                image = self.rawCapture.array
                if len(self.frames)==self.max_frames:
                    self.frames = self.frames[1:]
                self.frames.append(image)
            time.sleep(dt)
        logger.info("Thread stopped successfully")

    def status(self):
        global status
        logger.debug("Status isrunning" + str(status))
        if(status):
            content = "STOP THREAD !"
            status = False
        else:
            content = "START THREAD !"
            status = True
        return content


    def get_frame(self, _bytes=True):
        if len(self.frames)>0:
            if _bytes:
                img = cv2.imencode('.png',self.frames[-1])[1].tobytes()
            else:
                img = self.frames[-1]
        else:
            with open(archive_path+"/not_found.jpeg","rb") as f:
                img = f.read()
        return img
```

메인 화면 index.html

메인 화면 index.html

```
{% extends "base.html" %}

{% block content %}
<div class="container">
    <center>
    Submarine 0.1
    </center>
    <form name="frm">
    <table>
        <tr>
            <td colspan=3>
            Temperature = <span id="temperature"> 내부 온도 </span>
            </td>
        </tr>
        <tr>
            <td colspan=3>
      메시지:<span id="resultmessage">...</span><br><BR>
            <input type="button" value="VideoStatus" id="btnVideoStatus" /> Video
Status=<span id="videostatus">STARTING</span>
            <audio autoplay controls>
                <source src="http://web.kkorack.com:19080/out.mp3" type="audio/mp3">
            </audio>
            <BR>
            </td>
        </tr>
        <tr>
            <td>
                <table>
                    <tr>
                        <td></td>
                        <td><input type="button" value=" LIGHT ON " id="btnLightOn" /
></td>
                        <td><br><br><br></td>
                    </tr>
                    <tr>
                        <td></td>
                        <td><input type=text id="moveFBLFCnt" maxlength="5" size="5"
value=1 />
                        <td><br><br><br></td>
                    </tr>
                    <tr>
                        <td></td>
```

```
                        <td><center><input type="button" value=" ^ " id="btnForward" /
></center></td>
                        <td></td>
                    </tr>
                    <tr>
                        <td><input type="button" value=" < " id="btnLeft" /></td>
                        <td></td>
                        <td><input type="button" value=" > " id="btnRight" /></td>
                    </tr>
                    <tr>
                        <td></td>
                        <td><center><input type="button" value=" v "
id="btnBackward" /></center></td>
                        <td></td>
                    </tr>
                </table>
            </td>
            <td>
                <img src="{{ url_for('video_usb_feed') }}" class="border border-dark">
            </td>
            <td>
                <table>
                    <tr>
                        <td colspan=3>
                            <input type="button" value="SubmarineStatus"
id="btnSubmStatus" /> Submarine Status=<span id="submstatus">STOPED</span><br>
                            <img src="{{ url_for('video_pi_snapshot') }}" width=400
class="border border-dark">
                            <br>
                            <span id="communicationlevel"> 통신 정보 </span><br>통신이 중지되
면 부상
                        </td>
                    </tr>
                    <tr>
                        <td colspan=3>
                            <input type=text id="moveCnt" maxlength="5" size="5"
value=1 />
                        </td>
                    </tr>
                    <tr>
                        <td></td>
                        <td><input type="button" value=" UP " id="btnFloating" /></td>
                        <td></td>
                    </tr>
                    <tr>
                        <td></td>
                        <td><input type="button" value=" DOWN " id="btnDiving" /></td>
                        <td></td>
                    </tr>
                </table>
```

```
            </td>
        </tr>
        <tr>
            <td colspan=3>
                <span id="alertmessage"> 알림 메시지 </span><br>
            </td>
        </tr>
    </table>
    </form>
</div>
<script type="text/javascript">


var tempId = null;
tempId = setInterval(getTemperatureFunc, 1000*60);
var commCheck = null;
commCheck = setInterval(getCommStatus, 1000*30);
//clearInterval(timerId);

    function getTemperatureFunc() {
        var fName = "/temperature";
        xhr1 = new XMLHttpRequest();
        xhr1.open("get", fName, true);
        xhr1.onreadystatechange = function() {
            if (xhr1.readyState == 4) {
                    if (xhr1.status == 200) {
                            document.getElementById("temperature").innerHTML =
xhr1.responseText;
                        } else {
                            document.getElementById("temperature").innerHTML = "
temperature ERROR : " + xhr1.status;
                        }
                }
        }
        xhr1.send(null);
    }


    function getCommStatus() {
        var fName = "/commCheck";
        xhr2 = new XMLHttpRequest();
        xhr2.open("get", fName, true);
        xhr2.onreadystatechange = function() {
            if (xhr2.readyState == 4) {
                    if (xhr2.status == 200) {
                            document.getElementById("communicationlevel").innerHTML =
xhr2.responseText;
                        } else {
                            document.getElementById("communicationlevel").innerHTML =
" communicationlevel ERROR : " + xhr2.status;
```

```
                    }
                }
            }
            xhr2.send(null);
        }

        ////////////////////////////////////////////////////////////////////////////////
///////////

        window.onload = function() {
            document.getElementById("btnLightOn").onclick = setLightOnOff;
            document.getElementById("btnFloating").onclick = getFloatingFunc;
            document.getElementById("btnDiving").onclick = getDivingFunc;
            document.getElementById("btnForward").onclick = getForwardFunc;
            document.getElementById("btnBackward").onclick = getTurnBackwardFunc;
            document.getElementById("btnLeft").onclick = getLeftFunc;
            document.getElementById("btnRight").onclick = getRightFunc;
            document.getElementById("btnVideoStatus").onclick = getVideoStatusFunc;
            document.getElementById("btnSubmStatus").onclick = getSubmStatusFunc;
        }

        function setLightOnOff() {
             var fName = "/lightOnOff";
             xhr = new XMLHttpRequest();
             xhr.open("POST", fName, true);
             xhr.onreadystatechange = function() {
                  if (xhr.readyState == 4) {
                       if (xhr.status == 200) {
                            document.getElementById("alertmessage").value =
xhr.responseText;
                       } else {
                            alert("ERROR : " + xhr.status);
                       }
                  }
             }
             xhr.send(null);
        }

            function getFloatingFunc() {
                var fName = "/floating";
                data=document.getElementById("moveCnt").value;
                xhr = new XMLHttpRequest();
                xhr.open("POST", fName, true);
                xhr.onreadystatechange = function() {
                    if (xhr.readyState == 4) {
                        if (xhr.status == 200) {
                            document.getElementById("resultmessage").innerHTML =
xhr.responseText;
                        } else {
                            alert("ERROR : " + xhr.status);
```

```
                    }
                }
            }
            xhr.setRequestHeader('content-type', 'application/x-www-form-
urlencoded;charset=UTF-8');
            xhr.send("moveCnt=" + data);
        }

        function getDivingFunc() {
            var fName = "/diving";
            data=document.getElementById("moveCnt").value;
            xhr = new XMLHttpRequest();
            xhr.open("POST", fName, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        document.getElementById("resultmessage").innerHTML =
xhr.responseText;
                    } else {
                        alert("ERROR : " + xhr.status);
                    }
                }
            }
            xhr.setRequestHeader('content-type', 'application/x-www-form-
urlencoded;charset=UTF-8');
            xhr.send("moveCnt=" + data);
        }

        function getForwardFunc() {
            var fName = "/forward";
            data=document.getElementById("moveFBLFCnt").value;
            xhr = new XMLHttpRequest();
            xhr.open("POST", fName, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        document.getElementById("resultmessage").innerHTML =
xhr.responseText;
                    } else {
                        alert("ERROR : " + xhr.status);
                    }
                }
            }
            xhr.setRequestHeader('content-type', 'application/x-www-form-
urlencoded;charset=UTF-8');
            xhr.send("moveFBLFCnt=" + data);
        }

        function getTurnBackwardFunc() {
            var fName = "/backward";
```

```
            data=document.getElementById("moveFBLFCnt").value;
            xhr = new XMLHttpRequest();
            xhr.open("POST", fName, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        document.getElementById("resultmessage").innerHTML =
xhr.responseText;
                    } else {
                        alert("ERROR : " + xhr.status);
                    }
                }
            }
            xhr.setRequestHeader('content-type', 'application/x-www-form-
urlencoded;charset=UTF-8');
            xhr.send("moveFBLFCnt=" + data);
        }



        function getRightFunc() {
            var fName = "/right";
            data=document.getElementById("moveFBLFCnt").value;
            xhr = new XMLHttpRequest();
            xhr.open("POST", fName, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        document.getElementById("resultmessage").innerHTML =
xhr.responseText;
                    } else {
                        alert("ERROR : " + xhr.status);
                    }
                }
            }
            xhr.setRequestHeader('content-type', 'application/x-www-form-
urlencoded;charset=UTF-8');
            xhr.send("moveFBLFCnt=" + data);
        }

        function getLeftFunc() {
            var fName = "/left";
            data=document.getElementById("moveFBLFCnt").value;
            xhr = new XMLHttpRequest();
            xhr.open("POST", fName, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        document.getElementById("resultmessage").innerHTML =
xhr.responseText;
```

```
                    } else {
                        alert("ERROR : " + xhr.status);
                    }
                }
            }
            xhr.setRequestHeader('content-type', 'application/x-www-form-
urlencoded;charset=UTF-8');
            xhr.send("moveFBLFCnt=" + data);
        }

        function getVideoStatusFunc() {
            var fName = "/VideoStatus";
            xhr = new XMLHttpRequest();
            xhr.open("get", fName, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        document.getElementById("videostatus").innerHTML =
xhr.responseText;
                    } else {
                        alert(" status ERROR : " + xhr.status);
                    }
                }
            }
            xhr.send(null);
        }

        function getSubmStatusFunc() {
            var fName = "/SubmStatus";
            xhr = new XMLHttpRequest();
            xhr.open("get", fName, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                    if (xhr.status == 200) {
                        document.getElementById("submstatus").innerHTML =
xhr.responseText;
                    } else {
                        alert(" status ERROR : " + xhr.status);
                    }
                }
            }
            xhr.send(null);
        }
```

```
</script>

{% endblock %}
```