technicalSolution

February 23, 2025

1 Technical Solution

1.1 Contents

Source files

- File Structure
- Dependancies
- Organisation
- Diagram ##### Packages
- Utils
- Control
- Services
- Transducers
- Web
- Main.py

File Structure: The following file structure is used to organise the final iteration of my code:

```
iteration2
├── control
│   ├── __pycache__
│   ├── __init__.py
│   ├── autoPilot.py
│   ├── boatSim.py
│   ├── pidModule.py
│   ├── simpleModel.py
│
├── services
│   ├── __pycache__
│   ├── slapStore.py
│   ├── slapStoreTest.py
│
├── transducers
│   ├── __pycache__
│   ├── gps.py
│   ├── tillerActuator.py
│
├── utils
│   ├── __pycache__
│   ├── nmea
│   ├── __init__.py
│   ├── liveCompass.py
│
├── web
│   ├── __pycache__
│   ├── static
│   ├── templates
│   ├── app.py
│
└── main.py
```

Dependencies Install dependencies

Organisation I utilised a status.md to keep track of progress

Packages I have split the code into five seperate packages: - control - services - transducers - utis - web

Control Package

This package contains the PID and boat simulator systems:

Services

This package contains the database and storage

Transducers

This package contains componants of the system which take an input and provide an output. There are two transducer files, gps.py and tillerActuator.py gps.py simply returns a NMEA string for heading. As we are using a simulated boat and GPS, the gps.py must encode the actual heading from the boat into a NMEA string to emulate the real world GPS.

The tiller actuator is used to turn the maths and processes occuring into the system into a usable motor control signal to turn the rudder.

[ ]:
```
# %load ../../../src/iteration2/transducers/gps.py

# gps.py


from control.boatSim import BoatSim
from utils.nmea.nmeaEncoder import Encoder
class Gps:
    def __init__(self, boat_sim: BoatSim):
        # Import the instance of the boat simulator
        # Import the encoding functions I made
        self.boat_sim = boat_sim
        self.encoder = Encoder()

    def getHeading(self):
        # Returns encoded heading
        return self.encoder.encode_Angle(self.boat_sim.getHeading())
```

Web

This package contains all the componants to run the web server and allows it to interact with the rest of the system: This module contains all the HTTP files to be served to the client, the javascript and all images used. The web server main is called app.py, this contains all the service functions of the server:

[ ]:
```
# app.py

# %load ../../../src/iteration2/web/app.py
from flask import Flask, request, render_template, jsonify, g
from services.slapStore import SlapStore
from services.slapStore import Boat
```

```python
from control.autoPilot import AutoPilot
import threading
import queue
import time

class WebServer:

    def __init__(self, auto_pilot: AutoPilot):
        # Importing the Auto Pilot instance
        self.auto_pilot = auto_pilot


    def create_server(self):

        # Creating instances of Flask and the Database service
        app = Flask(__name__)
        store = SlapStore()

        @app.route("/")
        def home():
            # Default Route
            return render_template('changeangle.html')

        @app.route("/config")
        def config():
            # Route for adding a boat
            return render_template('config.html')

        @app.route('/api/setDirection', methods=['PUT'])
        def setDirection():
            try:
                # Sets the target heading based on users input in web page
                # Get data from request
                heading = request.get_data().decode('utf-8')
                print("Received data:", heading)

                # Update desired heading, returns the new actual heading
                heading = self.auto_pilot.setHeading(int(heading))

                # Return JSON response
                response_data = {"angle": str(heading)}
                print("Heading", response_data)

                return jsonify(response_data), 200

            except Exception as e:
                print(f"Error processing setDirection request: {str(e)}")
```

```
                return jsonify({"error": str(e)}), 400

        @app.route('/api/headings', methods=['GET'])
        def get_headings():
            # Returns Headings (Target and Actual)
            return jsonify(self.auto_pilot.getHeadings())


        @app.route('/api/addDirection', methods=['PUT'])
        def addDirection():
            try:
                # increments the target heading based on users input in web page
                # Get data from request
                change = request.get_data().decode('utf-8')
                print("Received data:", heading)

                # Update desired heading, returns the new actual heading
                heading = self.auto_pilot.getHeading()
                heading += change
                heading = self.auto_pilot.setHeading(heading)

                # Return JSON response
                response_data = {"angle": heading}
                print("Heading", response_data)

                return jsonify(response_data), 200


            except Exception as e:
                print(f"Error processing request: {str(e)}")
                return jsonify({"error": str(e)}), 400


        return app
```

HTTP File

This is served to the client:

[ ]:
```
<!-- %load ../../../src/iteration2/web/templates/changeangle.html -->
<html>

<script src="{{ url_for('static', filename='js/angle.js') }}"></script>


<head>
    <title>Change Value</title>
```

```
</head>

<body>
    <div>
        Angle = <div id="angle">0</div>
    </div>
    <button onclick="changeValue(10)">+10</button>
    <!-- These buttons run the changeValue function in the javascript-->
    <button onclick="changeValue(-10)">-10</button><br><br>

    <label for="directionSet">Set Direction:</label>
    <input type="int" id="directionSet" name="directionSet">
    <button onclick="setValue(parseInt(document.getElementById('directionSet').
 ↪value))">Set Direction</button><br><br>
    <!-- This the is the SVG to display the compass and the two lines -->
    <svg width="200" height="200" xmlns="http://www.w3.org/2000/svg" >
       <image href="{{ url_for('static', filename='images/compass-rose.png')␣
 ↪}}" x="0" y="0" width="200" height="200"/>
        <line id = target
              x1="100" y1="100"
              x2="100" y2="50"
              style="stroke:rgb(255, 0, 0);stroke-width:2"
              transform="rotate(0, 50, 50)" />
        <line id = actual
              x1="100" y1="100"
              x2="100" y2="50"
              style="stroke:rgb(0, 0, 255);stroke-width:2"
              transform="rotate(0, 50, 50)" />
      </svg>
</body>

</html>
```

Main This file starts athe system running, It starts up and instance of each module, importing them into the other modules to ensure only a single instance of each is used:

[ ]:
```
# %load ../../../src/iteration2/main.py
from services.slapStore import SlapStore
from control.autoPilot import AutoPilot
from control.boatSim import BoatSim
from web.app import WebServer
from transducers.gps import Gps
from transducers.tillerActuator import TillerActuator
import threading
import time
import atexit
```

```
# Used for the decay equation in boatSim
TIMECONSTANT = 0.5

class Main():

    def __init__(self):
        # Init an instance of needed componants
        self.boat_sim = BoatSim(TIMECONSTANT)
        self.gps = Gps(self.boat_sim)
        self.tiller_actuator = TillerActuator(self.boat_sim)
        self.auto_pilot = AutoPilot(self.gps, self.tiller_actuator)

    def main(self):

        # Init Control system
        self.auto_pilot.start()

        # Ensure the controller stops when the application exits
        atexit.register(self.auto_pilot.stop)

        #Create and start Flask web server
        webserver = WebServer(self.auto_pilot)
        app = webserver.create_server()
        app.run(debug=True, use_reloader=False)



if __name__ == '__main__':

    main = Main()
    main.main()
```