Unmanned Underwater Vehicle Software Documentation

Olympic College Engineering Club

NOTE: *This is a "**living document**", and as such will be updated on a semi-regular basis. Please ensure you have the **latest version** of this document at all times.*

Table of Contents

Overview....3
Arduino....4
  Overview....4
  List of Tasks....4
Raspberry Pi....5
  Overview....5
  List of Tasks....5

Overview

This document serves as a comprehensive reference for all software programs necessary for the operation of Olympic College Engineering Club's (OCEC) Unmanned Underwater Vehicle (UUV). This is a "living document", and as such will be updated on a sem-regular basis. Please ensure you have the most recent version of this document at all times. This document consists only of a list and explanation for the software of the project, and does not contain any source code or compiled binary code. Please see the appropriate section on the project's GitHub repository for the source code.

Software tasks on board the UUV are handled by two main components: A miniature computer called a Raspberry Pi (RasPi), and a microcontroller called an Arduino. The Raspberry Pi is the more powerful of the two, so the majority of computational tasks are run there. Programs for the RasPi will be written primarily in Python 3.0, which is considered a beginner-friendly programming language. The Arduino is responsible for controlling servos and motors by sending the control signals that those components expect, as well as receive data from sensors and reporting it to the Raspberry Pi. Programs for the Arduino are written in C++, which is a somewhat more advanced language that may be challenging for beginners to learn. Arduinos do not have and operating system to handle multiple programs running at once, so only a single program can be uploaded to and run by the Arduino at any one time. To make the software easier to write, maintain, and understand, the Arduino program will be broken down into smaller parts that the "main" program will compile into a single binary, which will be uploaded to the Arduino.

The RasPi and Arduino communicate over a serial connection between the two. Arduino has built-in functionality to send and receive data over serial, and the Raspberry Pi software will use the Python serial library to read and send data. Messages between the two will be in a standardized form to make communication predictable and modular.

Arduino

Overview

The UUV uses an Arduino Mega 2560 for most of the interaction between the software layer and the electronics layer. The digital and analog pins on the Arduino connect to sensors (eg. thermistors, gyroscope, accelerometer, etc), thruster motors via electronic speed controls (ESCs), servos, and any other peripheral devices on board the vehicle. The Arduino communicates with the UUV's onboard computer (the Raspberry Pi) to report sensor data and receive commands for motor power and servo angle, etc. Programs for the Arduino are written in C++, and the Arduino can only have one program uploaded & running at any time. To make programming for the Arduino easier to manage and more collaborative, its software will be divided into smaller sub-programs, each of which contain the functionality for a single aspect of the Arduino's responsibilities, and which will be combined into the "main" program at compile time using the "include" keyword in C++.

List of Tasks

The following is a list (in no particular order) of all the tasks that the Arduino program needs to be able to complete. Each item on this list can be partitioned into its own sub-program.

- Send and receive serial data to and from the Raspberry Pi.
- Interpret received commands from the RasPi and call corresponding functions.
- Package data from sensors into messages to send to RasPi.
- Set motors (via ESCs) to a specific speed value.
- Set servo(s) to a specific angle value.
- Read sensor data from analog sensors and digital sensors. Each sensor will have its own sub-program that interfaces with the sensor according to the datasheet and pinout of the sensor.

Each of these tasks is elaborated on below.

Raspberry Pi

Overview

The Raspberry Pi is the main computer onboard the UUV. It is responsible for receiving commands either from the laptop computer of the operator on the surface if operating tethered, or creating commands (possibly? Depends on how we decide to make this work) onboard. These commands come mainly in the form of directional input from a controller, so the RasPi computes what power level each motor should be at to achieve the desired motion. Motor commands are then sent to the Arduino, which sends the signals to the motors. The RasPi also reads information from the serial connection to the Arduino, mainly sensor data from the vehicle's sensors. The sensor data is used to adjust the control of the vehicle, for example using gyroscope information to correct for unexpected roll, etc. Sensor data will also be sent to the laptop of the operator to be displayed. Connected to the RasPi is the UUV's camera, and the video feed from the camera is sent to the operator computer to give the operator a first-person view of the vehicle. When running autonomously, the RasPi will also be responsible for running the computer vision program that recognizes objects in the field of view of the camera.

List of Tasks

The following is a list (in no particular order) of all tasks the Raspberry Pi needs to be able to complete. Each item on this list can be its own separate Python program, but it needs to be included in the "main" program's "import" list at the start in order to be used.

- Receive commands from surface operator's laptop computer.
- Interpret and process commands from operator computer and send appropriate instructions to Arduino.
- Send commands to Arduino over serial connection.
- Receive sensor data from Arduino over serial connection.
- Process sensor data, and use it to inform the desired system state of the vehicle when calculating motor speed values, etc.
- Read camera feed from onboard camera and stream the feed to operator computer.
- Send sensor data to operator computer.
- (When in Autonomous Mode) Analyze environment data from sensors & camera to create virtual map of surroundings.
- (When in Autonomous Mode) Use virtual map for object avoidance & mission objective completion.

Each of these tasks is elaborated on below.