Software Formalization

Year: 2018 **Semester:** Fall **Team:** 6 **Project:** Garbage Collecting Boat
Creation Date: October 01, 2018 **Last Modified:** October 04, 2018

Author: Qianli Ma **Email:** ma160@purdue.edu

Assignment Evaluation:

Item	Score (0-5)	Weight	Points	Notes
Assignment-Specific Items				
Third Party Software		x2		
Description of Components		X3		
Testing Plan		x3		
Software Component Diagram		x4		
Writing-Specific Items				
Spelling and Grammar		x2		
Formatting and Citations		x1		
Figures and Graphs		x2		
Technical Writing Style		x3		
Total Score				

5: Excellent 4: Good 3: Acceptable 2: Poor 1: Very Poor 0: Not attempted

General Comments:
Relevant overall comments about the paper will be included here

1.0 Utilization of Third Party Software

We are using no third-party software on our microcontroller, outside of the C standard library and HAL library.

The computer-based user interface libraries being used are as follows:

Table 1. User Interface Libraries

Name	License	Description	Use
PyQt5	PSF [1]	"PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android." [2]	We use this library to write the graphical user interface to interact with the users.
socket	WIDE Project [1]	"This module provides access to the BSD socket interface. It is available on all modern Unix systems, Windows, MacOS, and probably additional platforms." [3]	We use this library to communicate with the boat through TCP.
Compass Widget	GPL	"On the #pyqt channel on Freenode, Epifanio was creating a compass widget. Although he eventually decided to use Graphics View for this, the following code could be used as the starting point for a simple custom widget with a custom signal and a property for controlling the angle." [4]	We modified this library to work with PyQt5. This library draw the compass on the GUI when angle provided.
Google Map API	Google Cloud Platform (GCP) API Key [5]	"The Maps JavaScript API lets you customize maps with your own content and imagery for display on web pages and mobile devices. The Maps JavaScript API features four basic map types (roadmap, satellite, hybrid, and terrain) which you can modify using layers and styles, controls and events, and various services and libraries." [6]	We obtained a GCP API Key for a one-year free trial. We use Google Map JavaScript interface to allow the user to set up routes and display the boat position.

threading	PSF [1]	"This module constructs higher-level threading interfaces on top of the lower level _thread module. See also the queue module." [7]	We use this library to allow continuous data transfer using TCP.
select	PSF [1]	"This module aims to wait for I/O completion. This module provides access to the select() and poll() functions available in most operating systems" [8]	We use this library to perform timeout when the boat is not sending any message.
time	PSF [1]	"This module provides various functions to manipulate time values."[11]	We use this library to wait for a second to reconnect socket after the previous one failed.
logging	PSF [1]	"This module defines functions and classes which implement a flexible event logging system for applications and libraries. The key benefit of having the logging API provided by a standard library module is that all Python modules can participate in logging, so your application log can include your own messages integrated with messages from third-party modules." [20]	We use this library to allow the program to record useful information for debugging and developing purpose.
os	PSF [1]	"This module provides miscellaneous operating system interfaces. This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. " [21]	We use this library to check the existence of files.

The video streaming interface libraries being used are as follows:

Table 2. Video Streaming Interface Libraries

Name	License	Description	Use
opencv-python	MIT [9]	"Wrapper for OpenCV python bindings" [9]	We use this library to encode our Raspberry Pi camera raw data into a JPEG format picture and then to decode the JPEG picture and display it as a picture stream on the user's screen.
threading	PSF [1]	"This module constructs higher-level threading interfaces on top of the lower level _thread module. See also the queue module." [7]	We use this library to enable multithreading on the Raspberry Pi in order to speed up camera data processing achieving a higher frames per second for our video stream.
picamera	BSD [10]	"This package provides a pure Python interface to the Raspberry Pi camera module for Python 2.7 (or above) or Python 3.2 (or above)." [10]	We use this library to enable python to access the Raspberry Pi camera module and obtain numpy arrays from the camera output.
time	PSF [1]	"This module provides various functions to manipulate time values."[11]	We use this library to enable wait timers in our python code to wait until out Raspberry Pi camera has turned on.
socket	WIDE Project [1]	"This module provides access to the BSD socket interface. It is available on all modern Unix systems, Windows, MacOS, and probably additional platforms." [3]	We use this library to enable TCP sockets in our video streaming interface so that we can continuously stream frames over TCP between our Raspberry Pi and our user interface.
pickle	PSF [1]	"The pickle module implements binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream, and "unpickling" is the inverse operation,	We use this library to convert our numpy array from the Raspberry Pi camera module into a byte stream so it could be sent over TCP and then transfer the byte stream back to a numpy array.

		whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as "serialization", "marshaling," or "flattening"; however, to avoid confusion, the terms used here are "pickling" and "unpickling"." [12]	
zlib	ZLIB [13]	"zlib is designed to be a free, general-purpose, legally unencumbered -- that is, not covered by any patents -- lossless data-compression library for use on virtually any computer hardware and operating system."[14]	We use this library to compress our byte stream before sending over TCP and to decompress after receiving the byte stream.
struct	PSF [1]	"This module performs conversions between Python values and C structs represented as Python strings. This can be used in handling binary data stored in files or from network connections, among other sources. It uses Format Strings as compact descriptions of the layout of the C structs and the intended conversion to/from Python values."[15]	We use this library to pack our frame byte stream and data size into a new byte stream when sending over TCP and to unpack when we receive the data stream over TCP.
numpy	BSD[16]	"Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. "[17]	We use this library to manipulate the numpy array received from the TCP socket to convert the frames into RGB color space.
paramiko	GNU LPGL v2.1[18]	"Paramiko is a Python (2.7, 3.4+) implementation of the SSHv2 protocol [1], providing both client and server functionality. While it leverages a Python C extension for low-level cryptography (Cryptography), Paramiko itself is a pure Python interface around SSH networking concepts."[19]	We use this library to remotely start our video stream coming from the Raspberry Pi since our TCP socket has to be opened first before the video stream starts.

2.0 Description of Software Components

The software components for the project includes boat steering logic, the video streaming interface, the function of receiving data in the microcontroller and transmitting data to the user interface, and battery monitoring.

The boat steering logic component is responsible for calculating the expecting PWM duty cycle values for both left and right turbojet motors in order to steer the boat with the desired heading. The boat steering logic takes current GPS information, current compass information, and the desired route coordinates as inputs. The working duty cycle for the motor ranges from 7% to 12%. When going straight, both left and right values will be set to 9. When turning left, the left PWM value will be set to 7 and the right PWM value will be set to 11. Likewise, when turning right, the left PWM value will be set to 11 and the right PWM value will be set to 7. On the microcontroller side, when a new PWM is received, it will reach the desired value by gradually add or subtract 1% every second.

There are 14 ADC channels, using DMA and constant sampling cycle, used for break beam circuit and battery monitor circuit. The ADC sampling is always being processing and once the voltage drop below the warning value, warning information will be sent to the user interface.

The data from GPS is in NMEA format [22], and the major information utilized is $GPRMC (Recommended minimum specific GPS/Transit data), which includes coordinates and the speed in knots. In the limited micro RAM, a 512-byte buff is used to store the coming NMEA message, the program extracts the useful information in $GPRMC message from the whole and then send it to the user interface.

The communication tool between micro and user interface is ESP8266 which connect micro with UART and communicate in TCP protocol. When rebooting the micro, the initialization of ESP8266 is done. Once successfully reset, UART2 is hearing from any coming message from the user interface in the infinite loop, and once received anything, send back the information of GPS coordinates, compass, filling situation, and battery charge.

The compass talks with the micro by either I2C or SPI. Since the order package is still on the way, two peripheral are both left for testing. The principle to deal with the information received by compass is the same to that of GPS.

3.0 Testing Plan

The testing plan for all the ADC channel is just to check whether the value stored in the buff changes. The temporary LDR circuit can be used to test the voltage drop once receive luminance. The range of battery voltage is known, thus can be easily transferred and tested in digital value.

To test GPS communication, we plan to take the board outside the room. By checking the value stored in corresponding variables, we can know whether the GPS module talks to the microcontroller properly and whether useful information is extracted. The compass information testing works in the same way as that of GPS, just using different variables and memory.

The Wi-Fi communication testing is the most significant one. First, the reset command of the ESP8266 is tested by checking "OK" message received and only success can lead to the later on the process. It is constantly hearing from the user interface, and once receive "command" from the user interface, the value of some variable representing the speed will change to the received value. Then, it sends a series of information including GPS, compass, battery, and fullness of storage to the user interface. If the interface cannot connect to ESP8266, any error or exception happens during the communication, it will print "connection error". If interface successfully sends the message but cannot hear from the microcontroller, it will try to reconnect the Wi-Fi module and keep sending a message.

4.0 Sources Cited:

[1] "History and License," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3.7/license.html. [Accessed: 04-Oct-2018].

[2] "PyQt5," *PyPI*. [Online]. Available: https://pypi.org/project/PyQt5/. [Accessed: 04-Oct-2018].

[3] "socket - Low-level networking interface," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3/library/socket.html. [Accessed: 04-Oct-2018].

[4] "Compass widget," *Python Wiki.* [Online]. Available: https://wiki.python.org/moin/PyQt/Compass%20widget. [Accessed: 04-Oct-2018].

[5] "Authentication Overview | Authentication | Google Cloud," *Google*. [Online]. Available: https://cloud.google.com/docs/authentication/. [Accessed: 04-Oct-2018].

[6] "Overview | Maps JavaScript API | Google Maps Platform," *Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/tutorial. [Accessed: 04-Oct-2018].

[7] "threading — Thread-based parallelism," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3/library/threading.html. [Accessed: 04-Oct-2018].

[8] "select — Waiting for I/O completion," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3/library/select.html. [Accessed: 04-Oct-2018].

[9] "opencv-python 3.4.3.18," *Python Software Foundation.* [Online]. Available: https://pypi.org/project/opencv-python/. [Accessed: 04-Oct-2018].

[10] "picamera," *Sphinx.* [Online]. Available: https://picamera.readthedocs.io/en/release-1.13/. [Accessed: 04-Oct-2018].

[11] "time — Time access and conversions," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3/library/time.html. [Accessed: 04-Oct-2018].

[12] "pickle — Python object serialization," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3/library/pickle.html. [Accessed: 04-Oct-2018].

[13] "Zlib License," *zlib software.* [Online]. Available: http://www.zlib.net/zlib_license.html. [Accessed: 04-Oct-2018].

[14] "zlib 1.2.11," *zlib software.* [Online]. Available: http://www.zlib.net/. [Accessed: 04-Oct-2018].

[15] "struct — Interpret bytes as packed binary data," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3.7/library/struct.html. [Accessed: 04-Oct-2018].

[16] "NumPy," *NumPy developers.* [Online]. Available: http://www.numpy.org/. [Accessed: 04-Oct-2018].

[17] "NumPy," *wikipedia.* [Online]. Available: https://en.wikipedia.org/wiki/NumPy. [Accessed: 04-Oct-2018].

[18] "paramiko/LICENSE," *paramiko.* [Online]. Available: https://github.com/paramiko/paramiko/blob/master/LICENSE. [Accessed: 04-Oct-2018].

[19] "Paramiko," *y Sphinx 1.6.7 & Alabaster 0.7.11.* [Online]. Available: http://www.paramiko.org/. [Accessed: 04-Oct-2018].

[20] "logging — Logging facility for Python," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3/library/logging.html. [Accessed: 04-Oct-2018].

[21] "os — Miscellaneous operating system interfaces," *Python Software Foundation.* [Online]. Available: https://docs.python.org/3/library/os.html. [Accessed: 04-Oct-2018].

[22] "NMEA data," *gpsinformation.net.* [Online]. Available: http://www.gpsinformation.org/dale/nmea.htm. [Accessed: 04-Oct-2018].

Appendix 1: Software Component Diagram

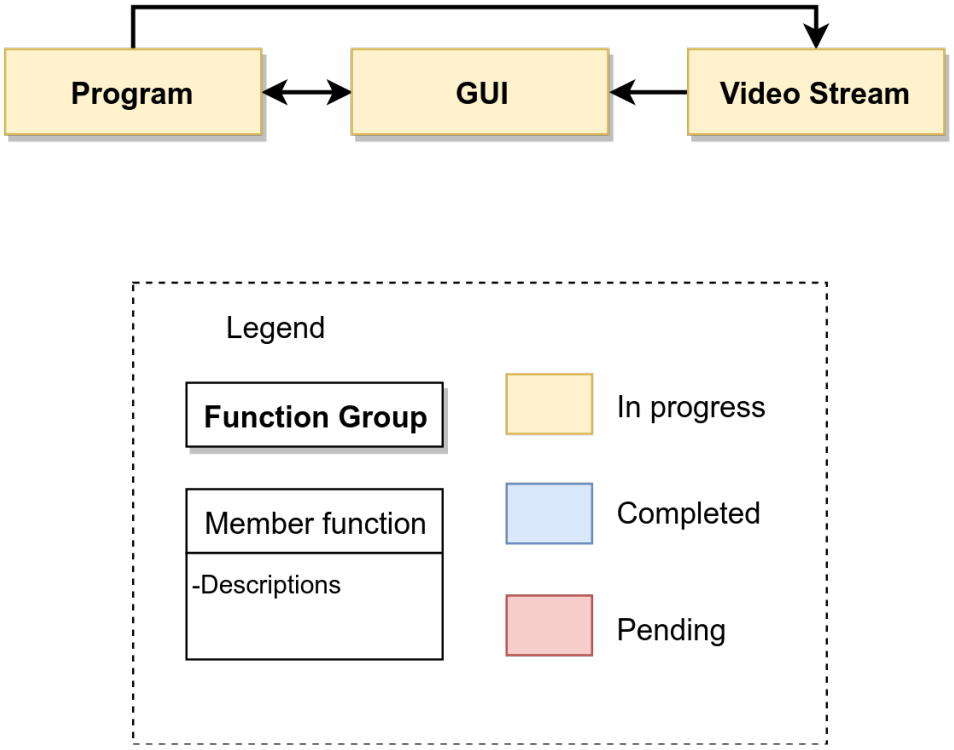

Figure 1. Overall Code Structure

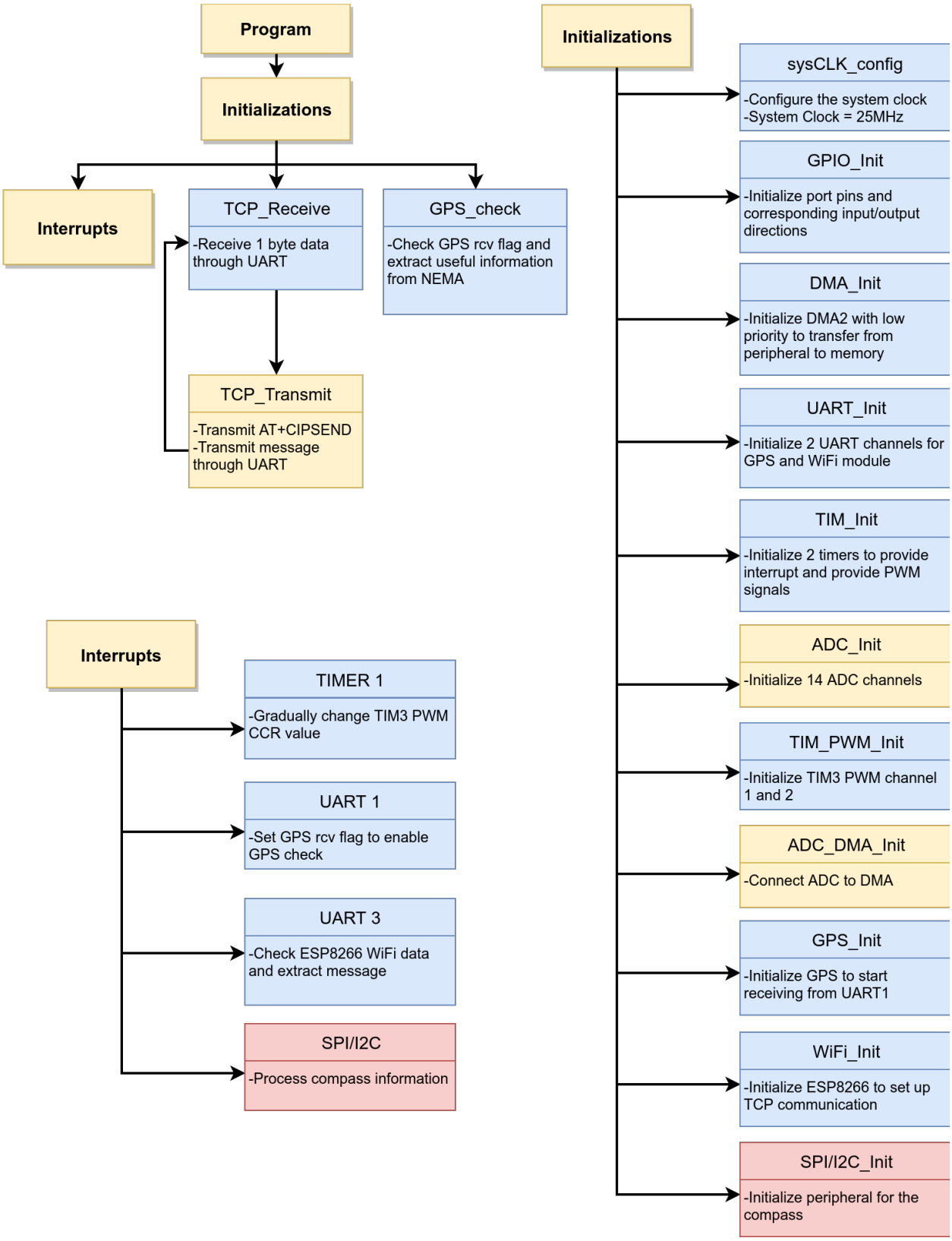

Figure 2. Micro Code Structure

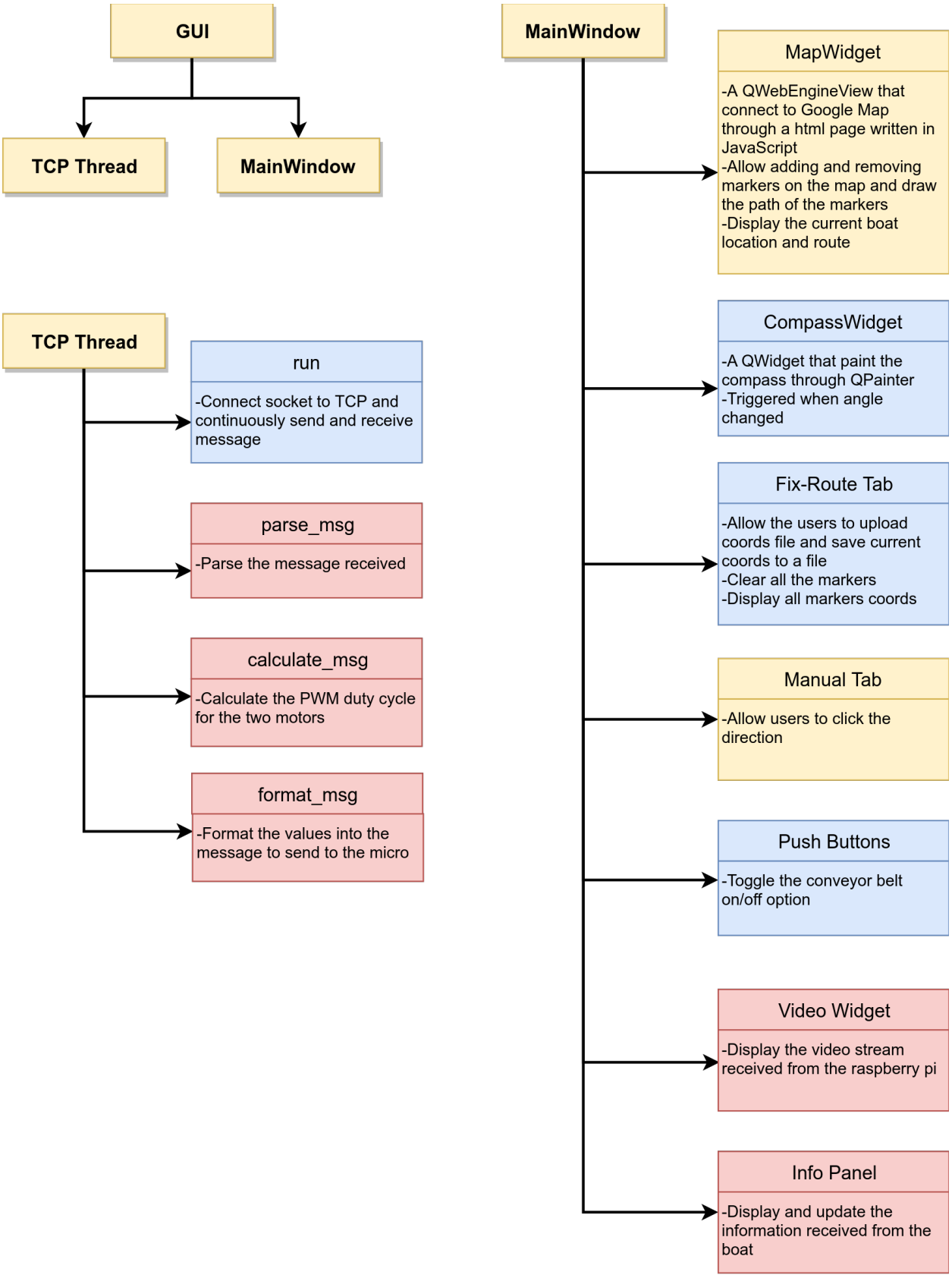

Figure 3. GUI Code Structure

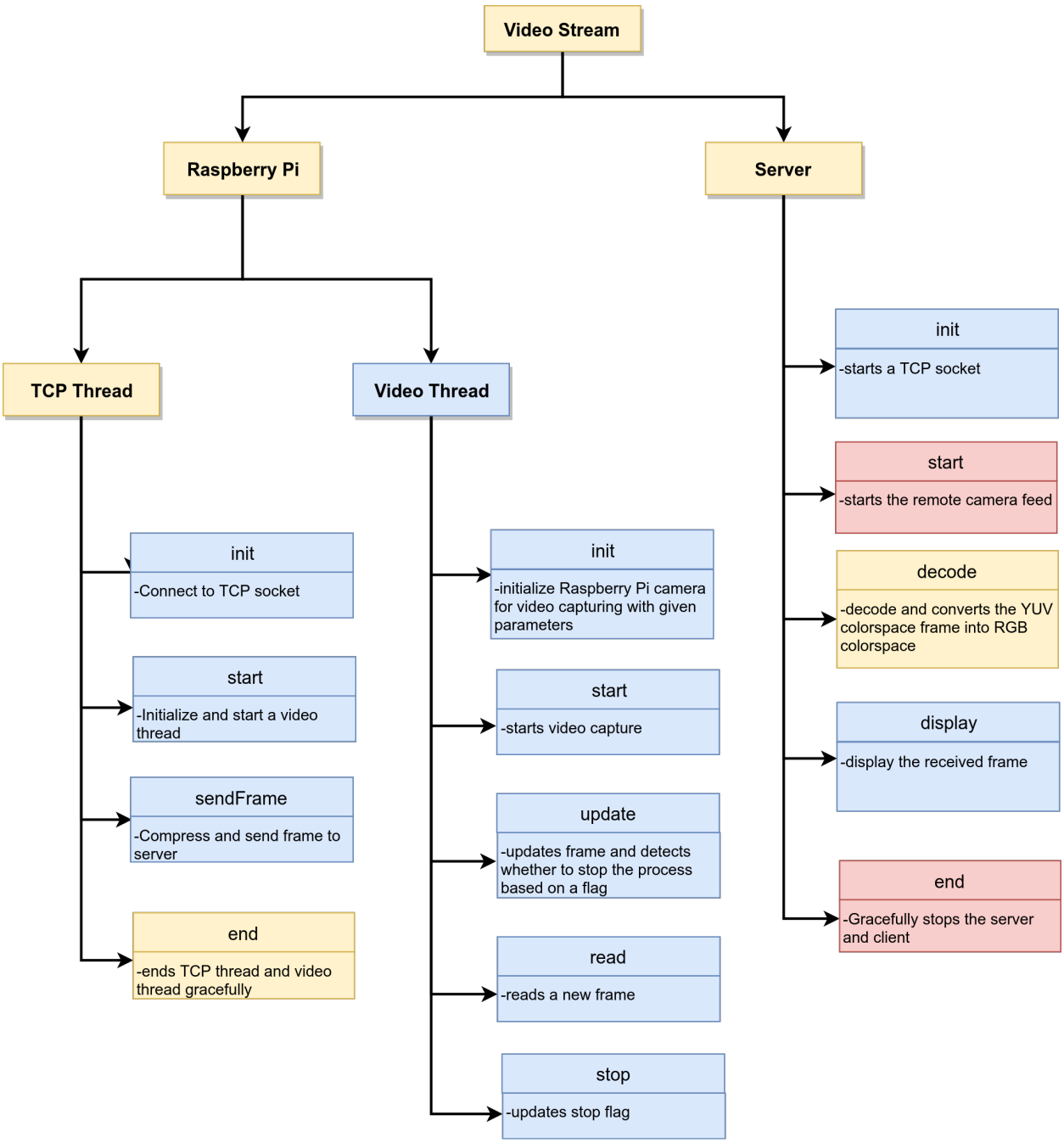

Figure 4. Video Stream Code Structure