Software Formalization

Year: \_2019\_ Semester: \_Spring\_ Team: \_\_17\_\_ Project: \_Face Tracking Drone\_\_\_\_\_\_

Creation Date: \_\_\_\_Feb 18, 2019\_\_\_\_\_\_\_\_\_\_ Last Modified: March 3, 2015

Author: Xufei Gao\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Email: [gao288@purdue.edu](mailto:_gao288@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:

1.0 Utilization of Third Party Software

|  |  |  |  |
| --- | --- | --- | --- |
| Name | License | Description | Use |
| flash loader demonstrator | GPLv3 | The STM32 Flash loader demonstrator (FLASHER-STM32) is a free software PC utility from STMicroelectronics, which runs on Microsoft® OSs and communicates through the RS232 with the STM32 system memory bootloader.[1] | The tool will be used to program the STM32F4 microcontroller via UART |
| OpenFace | GPLv3 | OpenFace is the ﬁrst toolkit capable of facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation with available source code for both running and training the models.[2] | We will be using FaceLandMarkVid to detect human faces and its command line API to output the data in a proper way. We will change the source code in order to process the data and send it to the microcontroller. |
| Mission Planner | GPLv3 | ArduPilot (including Copter, Plane, Rover, Sub and Antenna Tracker) and the ground control software (including Mission Planner, APM Planner 2 and MAVProxy) are free software: you can redistribute it and/or modify it under the terms of the GNU General Public License version 3 as published by the [Free Software Foundation](http://www.fsf.org/). [3] | The tool will be used to tweak the parameter for our drone and testing purpose. |

Table 1. Third Party Software

2.0 Description of Software Components

Our application requires communications between the ground station and our on-drone microcontroller. Thus, our software components are separated into two parts in the following.

**2.1 Ground station:**

**2.1.1 Image Processing Component**

The image processing component is built with a third-party library named ‘Openface’. It provides us with an important feature called Landmark Detection and Tracking.

The image processing component is responsible for recognizing faces and analyzing facial landmarks. The component produces head pose data stored in the following format (X, Y, Z, rot\_x, rot\_y, rot\_z). The translation is in millimeters with respect to camera centre. The rotation is in radians around X, Y, Z axises with the convention R = Rx\*Ry\*Rz.

**2.1.2 Facial Data to Control Data Component**

We will add one more module along with the image processing component which turns the facial data into control commands. Since the drone needs to know how to adjust positions in the air. This component is responsible for analyzing the head pose and producing control data.

**2.1.3 UART**

A UART module is used to send pc data to the microcontroller which is responsible for communicating with the drone. PC will use ‘/dev/tty’ to send data to the UART.

**2.1.4 Ground Station Telemetry Firmware:**

**2.1.4.1 UART**

A UART module is used to acquire data from a PC. Details are explained in 2.2.3.

**2.1.4.2 RF Transmitting**

A RF module will be used to send data to the on-drone microcontroller. It will communicate with the ground station telemetry using SPI3. A more specific scheme for interfacing will be described in the following 2.2.1. The only difference is that on the ground station, an external interrupt is not necessary.

**2.2 On-drone Embedded Firmware:**

**2.2.1 RF Receiving:**

Our RF module will communicate using SPI3 on the microcontroller, with an external interrupt pin. When data is received, the RF module will fire an interrupt and set the interrupt pin. When the interrupt enters the micro, the IRQ handler is called and read control signals from the RF. The import functions under this component include:

* nRF24l01\_Init(uint8\_t channel, uint8\_t payload, uint8\_t datarate, uint8\_t output power): Initialization process of the RF module.
* nRF24l01\_SetAddr(uint8\_t\* addr): Set the address for the device with a 5 bytes array.
* nRF24l01\_SetTxAddr(uint8\_t\* addr): Set the address for the transmitter with a 5 bytes array.
* nRF24l01\_GetData(uint8\_t\* addr): Read data from RF module into a array.
* nRF24l01\_TransmitData(uint8\_t\* addr): Transmit data as feedback to the transmitter.

**2.2.2 PPM:**

In order to communicate with the flight control, we are going to implement the Pulse Position Module.

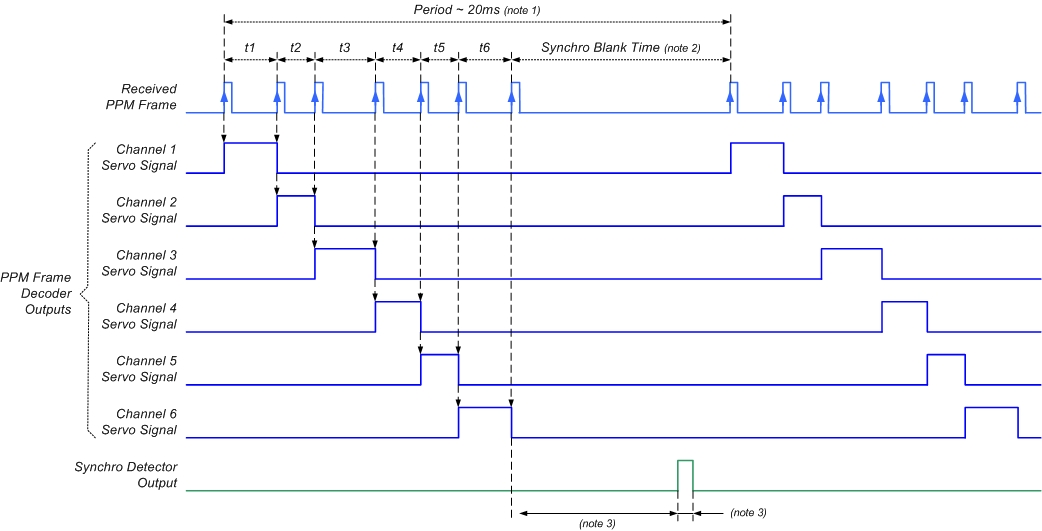


Figure 1. PPM signal [4]

Since STM32 cannot generate this kind of signal natively and there is no library online, we are going to implement this by ourselves starting from a low level with timer interrupt. With structure defined as follows:

* unsigned int pwm\_channel\_val[8]: This variable will store the desired pwm value for each channel, typically between 1000 and 2000.
* unsigned int channel\_inuse: This variable will indicate how many channels are in use.
* unsigned int next\_channel: This variable will record who is the next channel in the frame. if next\_channel >= channel\_inuse, the output will simply be low and this variable will be reset when the data frame ends.

With the structure above, we are able to output PPM signals with a single GPIO pin and timer interrupts.

**2.2.3 UART:**

The onboard UART module will be used as a data transmitter for debugging purpose while as programming port in boot mode. With important functions below:

* UART\_Reads(uint8\_t\* data, uint8\_t buf\_size): Reads a string from UART until CR or buffer full.
* UART\_Writes(uint8\_t\* data): Writes string through uart.

**2.2.4 Gimbal Controlling:**

The Gimbal is simply controlled by three analog channels. Unfortunately, we only have two DAC onboard. So, we decided to use a PWM with a low-pass filter as the third channel. The following interface will be provided:

* Gimbal\_move(uint8\_t channel, uint8\_t val): setting the x/y/z channel to the desired position

3.0 Testing Plan

**3.1 UART:**

* Read one frame of the UART data and display the frame on the putty in PC.
* Compare the message displayed on the putty with the message designated by the program.

**3.2 RF Transmitting and RF Receiving:**

The RF transmitting part will be tested with RF receiving part:

* Designate a message in the microcontroller program.
* Send the message via the RF transmitting part.
* Receive the message using RF receiving part and display the message received to putty in PC by UART.
* Check whether the message received matches the message transmitted.

**3.3 Gimbal Controlling:**

* Power the gimbal to realize the functionality of self-stabilizing.
* Change the inputs of three-axises rotation control voltage to rotate the gimbal on three axises.

**3.4 Pulse Position Modulation:**

PPM will be tested with the flight control. By connecting the flight control with the Mission Planner, we are able to view the value of each channel of the PPM signal. With simple interface (like using the signal from a potentiometer as input), we will be able to verify the correctness of our Pulse Position Modulation.

**3.5 Image Processing Component**

By using a general camera, the image processing component should be able to detect faces successfully. By giving multiple different head poses and check its output data is the only way to test this third-party source code. Also for accuracy, the processing speed should at least reach 15 frames per second. By adjusting camera settings and image quality we are able to fulfill the fps test.

**3.6 Facial Data to Control Data Component**

The test for this component should be unit testing. The way to test this component is by changing the head poses data produced by the Image Processing Component and see if the output falls into the correct range/acceptable range. When the human face is still and right in the center of the camera with rotations are zero, the output change for this case should be zero means that the drone should stay still and hold the altitude.

**3.7 UART**

This module is tested by sending a certain string to the UART and see if the receiver acquires the same data with no package lost.

4.0 Sources Cited

[1] “FLASHER-STM32 ,” *STM32F0DISCOVERY - Discovery kit with STM32F051R8 MCU - STMicroelectronics*. [Online]. Available: https://www.st.com/en/development-tools/flasher-stm32.html. [Accessed: 22-Feb-2019].

[2] TadasBaltrusaitis, “TadasBaltrusaitis/OpenFace,” GitHub. [Online]. Available: https://github.com/TadasBaltrusaitis/OpenFace/wiki/API-calls. [Accessed: 22-Feb-2019].

[3] “License (GPLv3)¶,” Planning a Mission with Waypoints and Events - Mission Planner documentation. [Online]. Available: http://ardupilot.org/dev/docs/license-gplv3.html. [Accessed: 22-Feb-2019].

[4] f2425252f2425252 61, “STM32F107 generate PPM signal,” Electrical Engineering Stack Exchange, 01-Jul-1967. [Online]. Available: https://electronics.stackexchange.com/questions/253684/stm32f107-generate-ppm-signal. [Accessed: 22-Feb-2019].

Appendix 1: Software Component Diagram

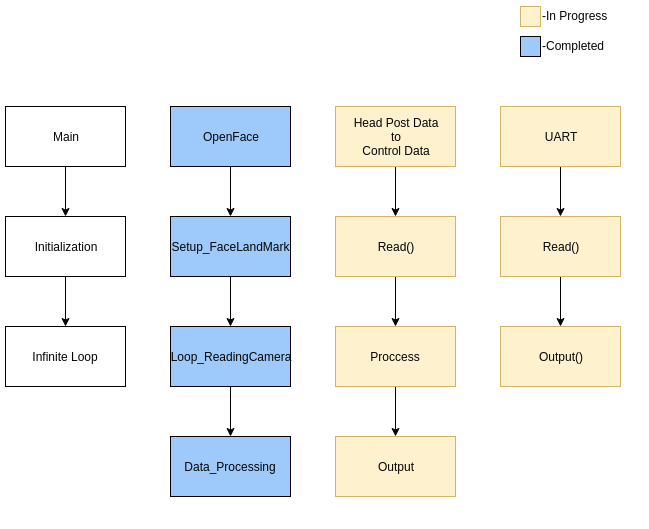
**

Figure 2. Ground Station Code Structure

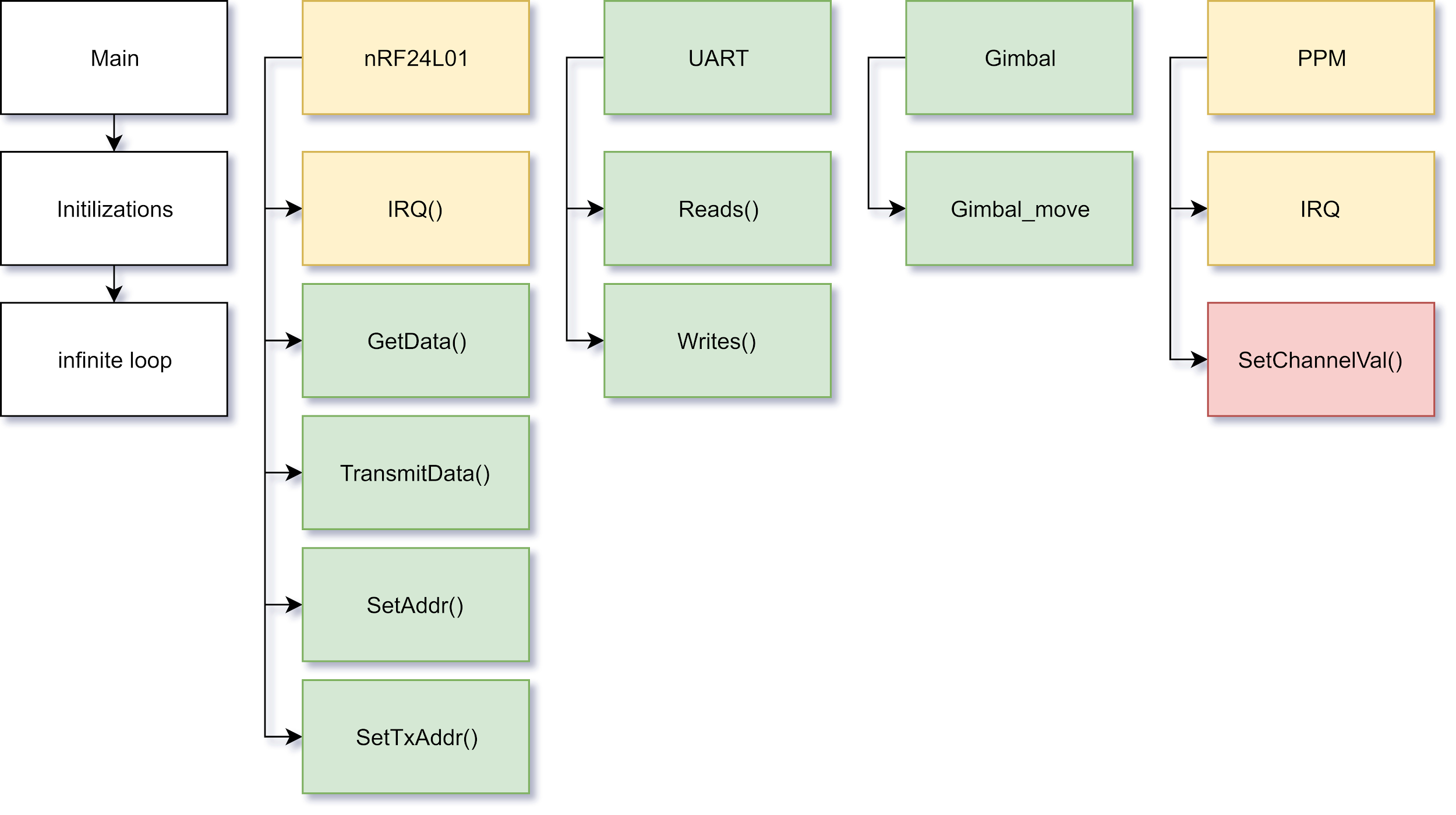
**

Figure 3. On-drone Embedded Firmware Code Structure

Green: Completed

Yellow: In progress

Red: Pending