Underwater P.O.V. Camera Tag

Arthur Bondar, EET

Benjamin Sabean, B.Sc

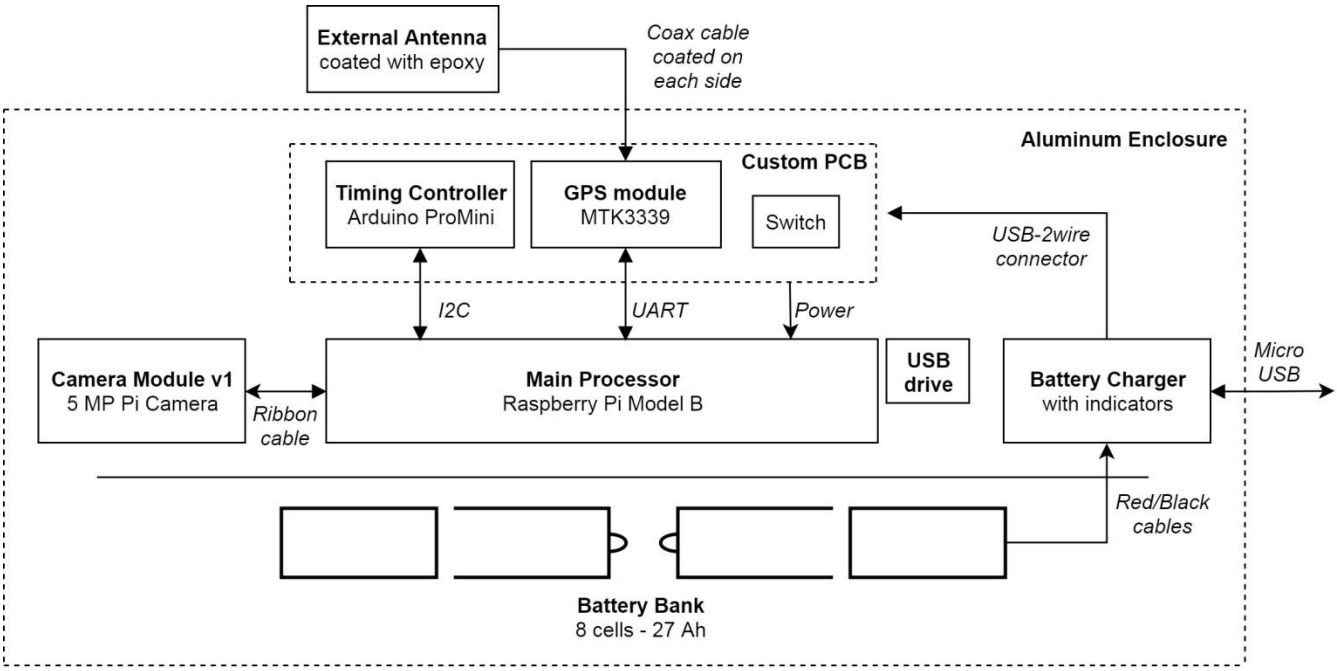
October 10, 2018

Contents

Overview 3
System Components 4
Battery *4*
Camera *5*
Raspberry Pi *6*
GPS *7*
Timing Control *8*
Interface 9
Deployment 11
Log file 13
Calculations 14
Parts List 15
Software Architecture 16

Overview

Underwater Point of View Camera Tag is designed for marine data collection. This device can be used as a POV tag for aquatic turtles, or in stationary monitoring applications. The system is equipped with a small video camera and a GPS. For enclosure, aluminum tube housing from Blue Robots rated up to 500m (1640ft) is used. The system has been deployed previously and proved robustness underwater. To power the electronics, array of Lithium-ion batteries and a charge controller are used. Camera Tag's video lapse characteristics can set by the user. The main interface for the system is a removable USB-flash drive. On-board computer reads the drive during boot and stores all recorded data on it. During long deployments, more than one day, the Tag is designed to enter sleep mode to extend battery life. For trouble-shooting purposes, the Tag has LED indicators and fault logging feature.



This diagram illustrates the main hardware building blocks of the system. Each building block will be further discussed in this document. Physical connection or protocol of communication is shown between each block. Block - *Custom PCB*, refers to a stacking board that is mounted on top of the Raspberry Pi. Battery bank is mounted on the bottom of the platform illustrated by the black separating line. Lastly, Micro USB cable is used to charge the battery bank.

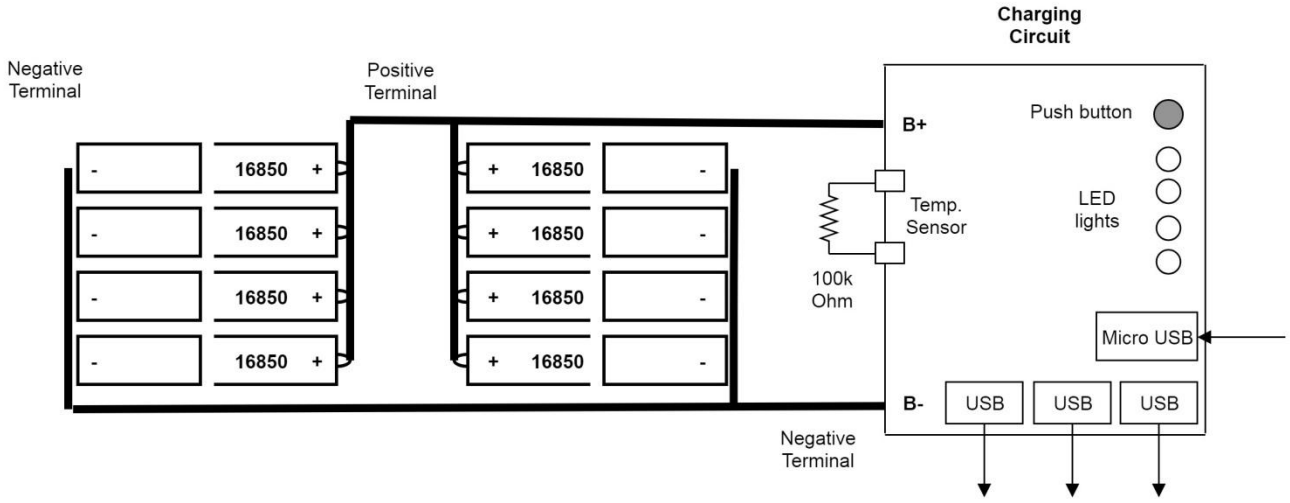
System Components

This section contains the information about each hardware component. Brief explanation, together with troubleshooting information, will be given for each component.

Battery

To support the on-board computer and extra circuitry a large battery bank has been selected. Battery bank from the manufacturer RAVPower provides 26800mAh storage capacity. It combines 8 genuine Lithium-ion cells (18650) together with a charge control circuit that can deliver up to 2.4A of current per USB output. The charger is also capable of charging the battery bank at 2A maximum current, provided that the power supply is ***2A capable***.

The charging circuit has 4 LED indicators in a bar configuration, and a push button. Each LED corresponds to 25% increment in charge and will flash during charging. Push button is used to display state of charge when the bank is not charging. During charge, the lights will stay on and the last LED will flash. Although it is possible to charge the bank and run the system for testing at the same time, it is not advised for best performance.



Note: after opening the enclosure, a temperature sensing resistor (thermistor) attached to the cells has been removed. In order for the battery bank to function, a 100kOhm resistor had to be soldered instead.

The battery bank has a single ***Micro-USB*** port for charging.

- Charging time using 1A charger, 27Ah / 1A = 27 hours
- Charging time using 2A charger, 27Ah / 2A = 13.5 hours

For safety and longevity of the battery, battery cell voltage should be monitored regularly. Simply connect the ***Red*** lead of the volt-meter to the ***Positive*** Terminal, and ***Ground (Black)*** to the ***Negative***. Loosely speaking, voltage of the cells will correspond to the amount of charge present in them. Cell voltage should be no lower than 3.0v (0%) and should not exceed 4.2v (100%).

Troubleshooting

If the Tag is not booting properly (no lights are ON), the battery/charger is the likely cause:

- Measure battery voltage. When voltage falls below 3.0v, the charger will not work. In this case, cells need to be replaced or charged using a constant-current power supply.
- Press the button on the charging circuit. If no LEDs light up, the charger is damaged or battery voltage fell below threshold.
- Connect micro-USB charging cable and use the push button to determine if charging circuit is working properly.

Camera

The current camera used in the Tag is the ***NoIR Raspberry Pi Camera module v1***. It is a 5MP camera based on OV5647 sensor. This module supports 1080p video resolution with up to 30 frames per second. The specific camera currently used (coloured black) is a ***NoIR*** Night-light camera that filters Infra-Red light. It is optimized for low light recordings. During daylight colors may appear filtered.

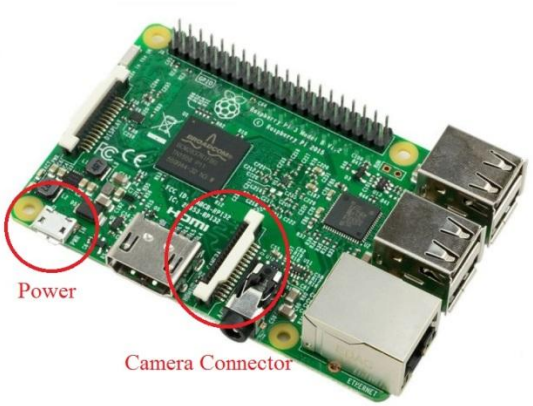
Camera is connected to the Pi using a ribbon cable (SCI connector). It can be easily swapped for higher resolution or daylight camera module without changes to the software. To change the camera, gently lift top of the connector and remove the old camera module. Place the new module and close the connector to secure the cable tightly. To make sure the direction of the cable is correct, look at the non-insulated part of the cable (copper end) and make sure they align with the copper traces inside the connector. Newer version of the camera - ***PiCamera v2 8MP,*** is available on the market and will be included in the parts list.

Troubleshooting

If the camera module is faulty, the system will not run and return to a sleep state. In that case no lights will be present on the camera itself and on the board. This premature shutdown will be reflected in system log file.

Raspberry Pi

Main computer of the system is Raspberry Pi model B. It is a powerful single board computer used in variety of embedded applications. In the Tag, it is used for processing video data, communication with the GPS, interacting with the timing controller and storing all data to a USB drive. The on-board computer has a Linux operating system with a command line interface.



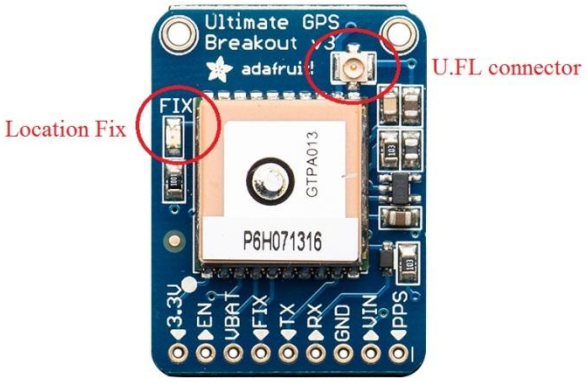
Although Raspberry Pi is a very capable embedded computer, it has a limited amount of resources. Thus, resource and heat monitoring is important during long deployments. There are number system parameters being recording in system log file. Tag's processor work load will vary with different recording conditions. During daylight, when Pi has to process an image with large number of pixels, performance reaches 50% of maximum capacity. However, when more shadow is captured by the camera lens, the processing task becomes less intensive and the system can rest at 5-10% of full load.

Troubleshooting

If raspberry Pi is faulty, the system won't run and no LED with light up on the board. Raspberry Pi can be replaced, provided that it has the software package – ***system image***. To trouble shoot the Pi by itself, unplug the battery connector (2-wire connector) and connect a micro USB cable (capable of supplying 2A) to the ***Power*** connector as labeled in the image above. The system should be able to boot and perform all necessary tasks if the Pi is functional.

GPS

The GPS module used in the Tag is the MTK3339, purchased from Adafruit (Ultimate GPS). It has a small form factor and integrates a very precise Real Time Clock. To support the clock during power-down, a small coin-cell battery (CIR2032) can be found on the bottom of the module's printed circuit board. On the upper side of the board a small UFL antenna connector is located. The GPS operated at 1.5GHz frequency and will work great with any antenna that is optimized for that band. This module communicates with the Rapsberry Pi by UART protocol thought the stacking board.



A great extra feature of the GPS is the LED light indicator– labeled ***fix***. This light provides visual indication when GPS has detected satellites and able to provide location.

- No position fix – LED flashes once every 2 seconds.
- Location found – LED flashed once every 10-15 seconds.

This feature can be used to visually check if satellite broadcasting can reach the module. Signal will depend greatly on the outside weather. With clear line of sight (clean sky), the module can establish a fix within 30 seconds.

<u>Troubleshooting</u>

The on-board LED can be an indication of a faulty GPS module. If the light doesn't come on within 30 second interval, the GPS is either damaged or not getting power.

Timing Control

In order to perform sleep cycles of the main computer (Raspberry Pi), a separate micro-controller has to stay awake and keep track of the time. A small Arduino ProMini controller is used for this purpose.

It communicates with Raspberry Pi through the pins of the stacking board and uses I2C communication protocol. This board has its own software written specifically for the timing purposes. Extra features such as battery voltage monitoring and release valve trigger can be added to this controller in the future. The main purpose of this unit is to send a wakeup signal and start the recording cycle.

The board has two main indicators:

- Blue LED on top of the PCB – indicates when sleep command is received and a sleep cycle is running.
- Small flashing red LED – flashes every second to indicate normal operation of the firmware.

Troubleshooting

When timing controller is faulty, wakeup interval will not be set correctly. In this case no wake up trigger will be generated and the main computer will not begin new recording cycle. For troubleshooting purposes, the on-board red LED can be used to determine operation. Additional troubleshooting step can be a test run and log file reading. At the end of each recording period, the Blue LED will come on to indicate Sleep cycle. If the LED didn't come on, the communication link between two boards is broken or the controller is damages. Controller can be replaced, provided that is has the software written to it.

Interface

User Interface is established thought removable USB flash drive. Any large USB storage device will be suitable for the Tag, provided that it is formatted with ***exFAT*** file system in order to accommodate large video files (FAT32 maximum file size is 4GB). File size calculations will be given in the following sections. Interface file is a ***text*** file called ***setup.txt*** or simply ***setup*** on some Windows OS.

Various unpredicted events can occur during the Tag's deployment, such as battery depletion, over temperature, end of disk memory and connection failure. To get the most information video data out of the system, video files have been split into small video ***sections***. This way, each section can be written permanently to the USB drive rather than corrupting one large video file.

The following is the content of the setup file and explanation of each parameter:

```
release = 30:00
sections = 20
start = 14:00
end = 16:30
video mode = 4
fps = 30
rotation = 0
gps interval = 55
```

Editing the file

The file can be edited with Notepad or preferably Notepad++. Each parameter has its name format that needs to be maintained. Parameter's name and value are separated by equal sign. White spaces before and after the equal sign do not affect file interpretation. Likewise, any extra spaces or new lines should not affect the interpretation. Tag's program opens the file, looks for known parameters and check for the value after equals sign, any extras will be discarded. Please make sure to save the file after each edit.

Release

This parameter is currently not in used. It is maintained in the software for potential upgrade with a release system for the Tag. This time specifies when a release valve or a burn wire will be triggered after beginning of the deployment.
Format: ***release = [hours]:[minutes]***

Sections

This parameter specifies the duration of the video sections mentioned earlier. To avoid video file corruption, the full duration has been split into smaller section. When a video is interrupted by unexpected even, only the last video section will be lost. Very small section time might put more load onto the USB drive and be inconvenience to work with. Large sections might result in loss of larger recording time (recommended section time is between 20 to 50 minutes). Videos sections can be stitched together using any video editing software. Note: there will be loss in recording time while each section is being moved to USB drive, for 20 min section transfer time is estimated to be around 1 minute.
Format: ***sections = [minutes]***

Start/End

Timing parameters – start and end, affect the Tag's sleep/recording duration. The camera will ***record*** for the interval between start and end (recording = end – start). The camera will ***sleep*** until a new start time is reached (sleep = start – end). The Tag is able to communicate with the GPS module and the exact precise current time. Even when the GPS cannot establish location (no fix), it is still able to provide the system with current time.

Note 1: System time is local time in Nova Scotia - ***Atlantic Daylight Time (UTC -3)***.

Note 2: A delay of up to one video section is expected past the ***end*** time or switch termination.

Format: ***start = [hours]:[minutes]***

Video mode

To maintain proper aspect ratios and resolution, default video settings are used. This parameter specifies the mode for camera recordings. Modes are taken from Raspberry Pi documentation settings and correspond to the following table:

#	Resolution	Aspect Ratio	Framerates	Video	Image	FoV	Binning
1	1920x1080	16:9	1-30fps	x		Partial	None
2	2592x1944	4:3	1-15fps	x	x	Full	None
3	2592x1944	4:3	0.1666-1fps	x	x	Full	None
4	1296x972	4:3	1-42fps	x		Full	2x2
5	1296x730	16:9	1-49fps	x		Full	2x2
6	640x480	4:3	42.1-60fps	x		Full	4x4
7	640x480	4:3	60.1-90fps	x		Full	4x4

Modes 1 and 2 are not suited for video recordings since they can't offer high framer-rates. Modes 4 and 5 would be ideal for typical use. Mode 1 has a higher resolution of 1080p at 30 frames per second, however the Field of View (***FoV***) in this mode is partial, meaning image will appear slightly "zoomed in". Higher quality recording will result in more stress on the system and larger file sizes.

Format: ***video mode = [number between 1 to 7]***

Detailed documentation about video parameters can be found here:
https://picamera.readthedocs.io/en/release-1.12/fov.html

FPS

This parameter refers to Frames per Second of the video recording. With higher frame-rates object movement appears smoother in the video. As with resolution, higher frame rates will result in longer transfer time, larger file sizes and stress on the system.
Note: Please make sure to not exceed the maximum ***Frame-rates*** for each video mode specified in previously shown table.
Format: ***fps = [number]***

Rotation

Rotation parameter is used to set the angle for camera video recording. With the Tag placed horizontally (battery on the bottom) the 0 degrees mark will be the correct orientation for the video.
Format: ***rotation = [number between 0 and 359]***

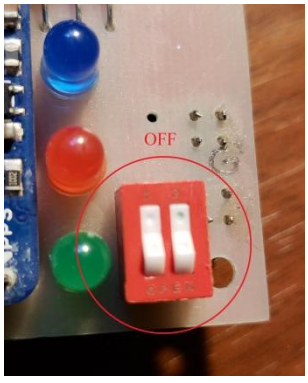
GPS interval

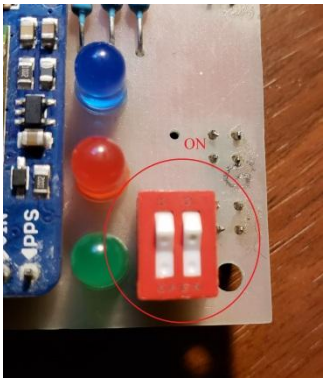
This parameter specifies the interval between GPS data samples in seconds. Location data is written on the USB as a CSV spreadsheet with the date as a filename. Data will only be written when a location fix is established and the interval between samples has past.
Format: ***gps interval = [seconds]***

Deployment

It is advised that you become familiar with the hardware circuitry before deployment. Perform a couple of simulations prior to deployment for testing system performance. Observe the log files to make sure each test finished successfully. To perform a test or deployment, follow these steps:

- Edit and save the ***setup*** file with preferred parameters.
- Make sure the red ***Switch*** on top of board is in OFF position.





If the switch is in ON position, the system will stop recording and shutdown. The switch is used to terminate video recording before the end of video duration interval. Activation of the switch will be tracked in the log file and might help troubleshooting. When switch is triggered, the last video section will be transferred to the USB drive and might take up to ***1-2 minutes***.

- Power up the system. Either through battery charge circuit (2wire-USB), or power the Pi directly using micro USB (2A) adapter for land testing.
- Observe the on-board indicators. The system has a row of 3 LEDs that provide information on system status (technically there is a fourth LED on the side of the board, it is designed for system release valve and is not used currently).

  Green – This light is used to indicate normal operation of the system when no video is recorded. For example when a video section is being moved, this LED will be on. Additionally, this light will blink 20 times during boot and during shutdown of the Raspberry Pi. Blinking during boot can be a good indication of normal performance. Likewise, when the program is stopped using the switch, blinking feature can let the used know that Pi is shutting down. After blinking the Green LED will go off and it is safe disconnect the power and remove the storage drive.

  Red – This light is used to indicate video recording. This light blinks every 1-2 seconds during video recording. The Tag is programmed to turn either the Green or the Red light, but not both at the same time. Blinking of the Red LED provides indication of whether the system is stalled, or running normally.

  Blue - This light is turned on by the Timing controller (Arduino) to indicate sleep state of the Tag. It is a bright blue light that stays solid throughout the sleep cycle. It comes off when Pi turns on for new recording.

During normal performance, the following pattern can be observed:

1. Green LED blinks fast - after power on Green light blinks 20 times.
2. Red LED blinks - recording starts, Green turns off and Red light blinks every 1-2 seconds.
3. Green LED solid - video section is finished, Red turns off and solid Green light turns on.
4. Red LED blinks - Section is moved and a new recording starts – Green off and Red blinks.
5. Green LED blinks fast - When recording cycle finished, or termination Switch was used, the Green LED will blink 20 times and go off.
6. Blue LED solid - at the same time as Green LED is blinking and Pi is shutting down, solid Blue light comes on to indicate beginning of the sleep count-down.
   Note: if the system was terminated using the switch, blue LED won't come on.

Additional lights that can be used to trouble-shoot the system are small camera light and Pi power light. Generally speaking, the red light on the camera should stay on while the red light on the board is blinking. Pi's red power light should stay on at all times while power is present.

Log file

The most important troubleshooting tool is the log file that is generated after each run. The file is called **log.txt** and is created on the USB after each deployment. The software logs major events during deployment to the log file. It provides insights into timing, parameters, successes and failures of the software. Let us examine step by step each parameter in the log file:

`START`	Beginning of code execution
`setting release time: 30hr 0min`	Release time as specified in setup file
`release set: SUCCESS`	Release time sent to Arduino successfully
`recording time: 128 min, each section: 20`	Recording cycle in minutes calculated from setup file and section time in minutes
`gps sample interval: 55 sec`	Logging interval for GPS readings
`video param: ['raspivid', '-t', '1200000', '-o', '/home/pi/Video/noname.h264', '-a', '12', '-md', '4', '-rot', '0', '-fps', '30', '-n']`	Video parameters: ***-md = video mode, -rot = rotation, -fps = frame rate***
`section started, camera pid: 1057`	Logs the beginning of a video section
`section finished`	Logs the end of a video section
`moving file: /home/pi/Video/2018-10-10_14_22_43.h264 to usb ...`	Tracks the beginning of file moving procedure
`move finished in: 1min 8sec`	Logs the end and time it took to move the file
`CPU temp=55.8'C`	Measures processor temperature (***below 85'C is normal***)
`CPU load average: 2.66, 1.00, 0.4`	Logs system stress. First number current load. Second average over 5 min. (***4.00 = 100%, 2.0 = 50%***)
`CPU throttled=0x`	Shows whether internal errors due to bad power supply or over temperature have occurred (***0x – normal, 0x50005 – error***)
`free RAM 423MB`	Shows free memory available for the system. (***750MB max, less than 50MB not desired***)
`CPU frequency(45)=60006200`	Shows the frequency of the processor (***60– normal, 120 and higher – under stress)***
`recording time left: 108 min`	After each section, the remaining recording time is logged
`code TERMINATED using switch`	Indicates termination using switch
`FINISH`	**Important**! If critical error occurred during deployment the log will not end with this line. Good indication on whether system encountered failures

Calculations

In this section, you can find roughs estimation on deployment duration, USB memory and section transfer rates. Please not that numbers provided here are only estimates, since exact duration will largely depend on how intensive processing of the video image is. More colourful image will result in more processing and larger file sizes. The user is encouraged to record the file sizes and transfer times after each deployment to build a more accurate profile.

Power calculation [Run time = Battery / Total consumption]

Recording time [hours]	Recording power consumption [A]	Sleep time [hours]	Sleep power consumption [A]	Consumption per day [Ah]	Battery capacity [Ah]	Run time [days]	Run time [hrs]
12	0.6	12	0.15	9	26.7	2.97	71.20
10	0.6	14	0.15	8.1	26.7	3.30	79.11
8	0.6	16	0.15	7.2	26.7	3.71	89.00

Storage [MB/s = File size / Video duration]

Video Mode	MB per second	Recoding time [hrs] 128GB USB	Recoding time [hrs] 256GB USB
1 at 30fps	2	17.8	35.6
4 at 30fps	1.75	20.3	40.6
5 at 30fps	1.2	29.6	59.3

Section transfer [mode 4, speed is 20MB/s]

Section length [min]	File size [MB]	Transfer time [sec]	Transfer time [min]
20	2100	105	1.75
40	4200	210	3.5
60	6300	315	5.25

Parts List

Bill Of Material - October 12, 2018					
Component	**Supplier**	**Quantity**	**Price per unit**	**Full price**	**Link**
Mechanical parts					
Aluminum Tube 8.75"	BlueRobotics	1	$200.00	$200.00	Aluminum Tube
Clear Acrylic End Cap	BlueRobotics	1	$13.00	$13.00	Acrylic End Cap
Aluminum End Cap	BlueRobotics	1	$12.00	$12.00	End Cap
O-Ring Flange (3" series)	BlueRobotics	2	$24.00	$48.00	O-Ring Flange
Cable Penetrator 6mm	BlueRobotics	2	$4.00	$8.00	Cable Penetrator
Cable Penetrator Blank	BlueRobotics	1	$4.00	$4.00	Penetrator Blank
Enclosure Vent and Plug	BlueRobotics	1	$8.00	$8.00	Vent and Plug
Spare O-Ring Set	BlueRobotics	1	$3.00	$3.00	O-Ring Set
Silicon Lube	BlueRobotics	1	$3.00	$3.00	Silicone Grease
			TOTAL:	$299.00	
Custom parts					
Suction cup holder	NSCC	2			
GPS mount	NSCC	1			
VHF holder	NSCC	1			
Sensor holder	NSCC	1			
Suction cups	NSCC	2			
Mounting Platform	NSCC	1			
Printer Circuit Board	NSCC	1			-
GPS mold	NSCC	1			
Electronic parts					
Raspberry Pi 3 B+	Amazon-CanaKit	1	$70.00	$70.00	Rasboerry Pi 3 B+
Pi Camera V2	Amazon-Smraza	1	$34.00	$34.00	PiCamera v2 8MP
Adafruit GPS	Adafruit	1	$55.00	$55.00	Ultimate GPS
Arduino Pro Mini	Amazon-NooElec	1	$11.00	$11.00	Arduino Pro Mini
USB flash disk 256gb	Amazon-SanDisk	1	$89.00	$89.00	SanDisk 3.0
Battery Bank	Amazon-RAVPower	1	$66.00	$66.00	26ah Battery Bank
Resistors, LED's, Buttons	Amazon-Kuman	1	$17.00	$17.00	Electronics Kit
JST Connectors	Amazon-daier	1	$9.00	$9.00	JST Connectors
			TOTAL:	$351.00	
Summary					
			TOTAL:	$650.00	

Software Architecture

This section is designed for developers. The following diagram illustrates the software components involved in development of the Tag (Class diagram).
Full software can be found here: https://github.com/ArthurBondar/Underwater-POV-Camera-Tag