Fish Sense - Low Power System

Project Specification - 237D

Rahul Polisetti, Mohana Seelan, Kyle Yang

1. Project Charter
   1. **Project Overview**

FishSense is an underwater fish detection module that can detect fish activity at a reasonable distance away from an embedded RealSense camera. The camera is fixed in a plexiglass cylindrical module and is submerged with divers in order to detect fish movement. It is fully enclosed and houses electrical components which are programmable with an adaptable firmware, able to record footage and import that data into an integrated solid-state drive. The footage is used to implement a machine learning algorithm which classifies fish from the images as well as calculates their size. At present, the module can last up to 2 hours of constant recording, but modifications are being done to lengthen this time. Our project is to introduce a low-power system to extend deployment periods so more footage can be captured. At a high level this would be accomplished by creating active and sleep (low-power) states which the system will be able to toggle between based on set parameters in order to extend the operating window of the device.

* 1. **Project Approach**

There are currently three approaches to the problem that we have identified. The first two approaches are to switch between active and sleep states based on a timer. This is the simplest setup and can be achieved either with only the existing NVIDIA Jetson TX2 board itself, or by using an external STM32 board as a timer to completely cut power to the main system. The third way would be to use an external sensor to detect the presence of fish and toggle the active and sleep states.

Based on review with our project leads, we are going to be focusing on the external STM32 board approach, since this would have the most minimal power draw during sleep if the main system has its power completely cut. It would also be easier to implement during the quarter compared to the sensor-based approach, since the parts are already available in SEALAB.

The project will include multiple opportunities for data logging and graphing since we will have to profile power usage for different states and state transitions, along with various changes in hardware/software made to said states.

* + 1. **Jetson Only Low-Power-System:**

An initial strategy we will implement is developing and programming a fixed state change operation which starts and stops recording at fixed intervals. The programming will be firmware written onto an NVIDIA Jetson TX2 board which will be fixed internally into the module. The module currently holds the Real Sense camera and a Sandisk SSD memory card within a plexiglass casing and mechanical hardware which holds an NVIDIA Jetson TX2.

The existing firmware on the NVIDIA Jetson will be modified to switch between states at specific times. This could be, for example, active for 1 minute and sleeping for 29 minutes. The Jetson will still have to be online for the sleep state too in order to keep track of time, and therefore we would need to find ways to reduce power consumption during this state. This could include lowering clocks and power collapsing unused cores during sleep.

The state-change firmware will be written in C++, provided by a GitHub repository from Engineers for Everyone. Three C++ files are provided to program the camera: “ply\_saver.cpp,” “rs-save-to-disk.cpp,” and “rs\_save.cpp.” ply\_saver.cpp is a program which analyzes the point-cloud representation of the fish in each frame of a video and segments the footage into the elements of the video frame. It also initializes a pipeline to stream the Real Sense footage in order to segment the images. The rs-save-to-disk.cpp file is a file which saves the RealSense data footage to the SSD disk from the module. This firmware writes the frames onto the Sandisk. The other program, “rs\_save.cpp” is another program which saves the frames of the camera footage onto the Sandisk SSD storage drive. Rs\_save.cpp is primarily used as a firmware which is to be modified for our low power system.

* + 1. **STM32 G0 Board Low-Power-System:**

Another strategy to implement the low-power-system is to program directly onto an alternate G0 board which is capable of turning on and off the main system completely, which includes the RealSense, NVIDIA Jetson TX2, SSD, and USB hub. The board utilizes very lower-power to run a timer, which can be interfaced with the system's existing power-IO board. Our power goal for this implementation is to draw less than 30 milliamps to power the STM board while the RealSense camera is in sleep mode. This would ensure enough power is available during the full duration of its deployment. We plan to accomplish this optimal power draw by utilizing firmware that directs power usage to the camera. We plan to implement the firmware by utilizing C++ libraries, such as librealsense, that integrate directly with the RealSense camera.

* + 1. **Alternative Sensor Based Low-Power-System:**

A sonar will be implemented as an external triggering system between power states that can detect when fish are active or not active near or around the module. The Ping Sonar Altimeter and Echosounder is a sonar available on the market which can be used and attached to a daughter PCB board with an alternate firmware. We have to write this firmware to achieve active and sleep states. The sonar is a single-beam echosounder that can measure distances up to 50 meters underwater, with a 30 degree beam width and 300 meter depth rating, and is sold by BlueRobotics and is included with a programmable interface and open-source software. Our further steps to take will be to research this software and modify it to work with our RealSense implementation.

* 1. **Minimum Viable Product**

Our minimum viable product for FishSense would consist of benchmarking the power draw in different states and creating a low power mode on the Jetson. Currently, FishSense is deployed underwater by a trained diver, turned on underwater, and then left on until the module is picked back up. During this mode, which we will refer to as active mode, The RealSense camera and the NVIDIA Jetson TX2 are powered up and collect footage the entire duration. Unfortunately, the external SSD fills up with video footage within a few hours and FishSense no longer collects data after that time period. It is not clear yet whether the battery runs out of the SSD card or if the battery becomes full first. Regardless, we know the battery cannot last 2 weeks.

For FishSense to become more valuable to oceanographers and other researchers who leverage the device, it should last underwater for around 2 weeks. We do not believe that there is a need for FishSense to be in active mode for the entirety that is underwater. Therefore, our MVP would consist of:

* Benchmarking the current power consumption of the active mode
* Using the necessary sampling frequency given by researchers of how often the RealSense needs to be collecting data:
  + Calculating the maximum power consumption the future low power mode would need to be under
* Turning the NVIDIA Jetson on and off at the necessary frequency and measuring the power consumption over time → in this case, low power mode is just “off”
* Creating a low power mode on the NVIDIA Jetson that may or may not achieve the max power consumption calculated
* A low power and active mode that can switch states with each other at a certain frequency

The short term goal of our project is to collect as much data about power consumption on FishSense, and try out a method of achieving the 2 week data collection goal underwater. Long term, the data we collect and the attempts we will make can inform the FishSense team of what works and what will not. Also, we hope that the data power collection methods we use will also help the FishSense team try out different solutions in the future.

We can also exceed the requirements of the MVP by connecting the STM32 IO board to the NVIDIA Jetson, and use the STM to toggle the Jetson between active and off states. This may be required for the long-term goals of the FishSense team if the Jetson cannot reach a state that is “low power enough.” However, configuring the STM board is not part of our proposed MVP.

* 1. **Constraints, Risk, and Feasibility**

Since we have a time constraint to complete the project in around eight weeks, we have ruled out using an auxiliary sensor to provide state change input. This is because finding a sensor, delivering the sensor, calibration and testing, and integration of the sensor will take significant time. This is if we can find a sensor which is low power enough in the first place. Therefore we are focused on using a timer based approach to state changes.

Our other constraint in this project is the physical battery capacity of the device. In order for the system to be operational for at least two weeks, we have to have to have almost zero power draw during the sleep state. This rules out using only the existing NVIDIA Jetson board to act as a timer, since it would be improbable to run the board at such a low power draw as to be negligible. The most viable and realistically feasible option is to use an external STM32 G0 board in low power RTC mode as a timer to cut power supply to the main system completely during the sleep state.

One of the stumbling blocks for this approach is being able to easily interface the STM32 with the existing Power IO board, without modifications. Our current idea is to connect the STM32 GPIO directly to the Power IO’s power bypass header. Another stumbling block is how the main system would behave to such power cycling, and if there must be any modifications to the main firmware to protect against any data corruption. We also are currently not aware of how long the active state and sleep state have to be to collect any meaningful data, and will have to consult with fish experts to get reasonable numbers.

Risks in this project include not being able to reach a low enough power on the STM32 board in a sleep state to last an entire week. We could add additional batteries to the device if this is the case, but this would be out of the scope of the project. There could also be issues with powering the STM32 board itself from the main systems batteries, which can be worked around by using a separate power supply just for the prototype. There is also the risk of damaging the batteries and the Jetson board due to constant power cycling.

1. Group Management

In our group management, the choices of our subproject contribution will be decided through consensus and the decisions of general implementation will be made by our project leader.

We will be communicating through Discord, through multiple chats for correspondence of each of the subprojects and the whole. We’ve also been using Discord for note taking and keeping track of assignment guidelines.

We will be following a Gant chart, and share it with our project leader.

1. Project Development
   1. **Development Roles**

The development of the low power system will be split into three parts: STM32 based timer development, NVIDIA Jetson side interfacing and firmware solution, and power benchmarking and testing. The three group members will be able to handle one of the three tasks each.

STM32 based timer development includes setting the board up into its lowest power state with only its RTC module active, writing the firmware for the board to toggle the interfacing mechanism on and off at certain time intervals, and if needed, developing a method to notify the Jetson a few seconds before shutting down the main system in order to prevent data corruption.

The NVIDIA Jetson side interfacing and firmware solution includes finding out how to interface the STM32 with the existing Power IO board and the Jetson hardware. It also includes modifications to the firmware to handle such power cycling. This could be stopping the recording process and closing all active file write handlers when a sleep notification is sent by the STM32, and then starting the recording process automatically when powered on.

The power benchmarking and testing role will include connecting the setup to the bench power supply instead of the batteries and logging power draw for different states and state transitions. The tester will also have to check data integrity during long term operations with power cycling involved. The role may also include testing the system underwater at SIO.

* 1. **Hardware/Software Used and Availability**

Thehardware used is the NVIDIA Jetson TX2 along with its development carrier board, STM32 G0 microcontroller, Intel RealSense camera, and SSD. The software used is the Jetsons preinstalled ARM64 version of Ubuntu, and the SDM32 programming environment. Programming will be done in C/C++, and will be based on existing firmware code. All hardware is already available in the SEALAB and currently nothing has to be purchased.

* 1. **Testing**

Power testing will be done using a lab bench power supply to measure the power draw for different states and state transitions. The data integrity of the captured video files will also be checked due to the addition of constant power cycling. Final testing will be done in the last two weeks by running the system for an extended period of time.

* 1. **Documentation**

This project's documentationis to be done on the existing Fish Sense documentation platform. This includes a shared Google drive where all setup instructions and logged data are stored. We will be following the same formatting and style guidelines set by the previous contributors. All the code will be commented and uploaded to the Fish Sense github repository.

1. Project Milestones and Schedules

**Deliverable 1:** Find out the necessary power goals given the current FishSense system and researcher use cases - Week 5

**Milestone 1.1:** Contact researches to find out how long and often FishSense needs to be active and collecting data from the RealSense Camera

Documentation: Emails correspondence between researchers, meeting notes, Zoom calendar invites

**Milestone 1.2:** Benchmark the current power consumption in FishSense’s active mode

Documentation: Graph of data over time, including boot up, power down

**Milestone 1.3:** Given the frequency of active mode (using deliverable 2 information), calculate the maximum power consumption that low power mode can have

Documentation: Graph, Data tables, Python/Matlab code for calculations

**Deliverable 2:** Create a low power mode for the NVIDIA Jetson - Week 6 & 7

**Milestone 2.1:** Turn the NVIDIA Jetson on and off at the researcher-given sampling frequency and measure the power consumption. In this case, “off” is the low power mode

Documentation: Github C++ code with programmed timer

**Milestone 2.2:** Create a low power mode on the NVIDIA Jetson by turning off cores slowing down clock speed, configuring different RealSense modes

Documentation: Github C++ code

**Extra Milestone 2.3:** Try to make active mode more low-power by changing RealSense settings

Documentation: Github C++ code

**Deliverable 3:** Assess how well low power mode does - Week 8

**Milestone 3.1:** Measure low power mode consumption and state change power consumption

Documentation: Graph, Data tables, Python/Matlab code for calculations

**Extra Deliverable 4:**

1. Connect the STM32 IO board to the Jetson and have that turn on and off the active mode on the Jetson
2. Measure power consumption after configuring STM IO Board

