FishSense - Low Power Final Project Report

Rahul Polisetti
rpolisetti@ucsd.edu

Mohana Seelan
mseelan@ucsd.edu

Kyle Yang
kiyang@ucsd.edu

1 Abstract

The FishSense Project was created to modernize fish research and farming methods, using the latest in Machine Learning and Underwater Imaging. The current system consists of an Nvidia Jetson TX2 as the main compute resource and an Intel RealSense camera, which currently lasts around 2 hours. The team is looking to increase the deployment window, to around 2-3 weeks. Hence, we create a low power system for FishSense, using an external STM32 based sleep timer board to power on and off the main system in user defined cycles. This allows the system to be powered on a few times per day, while staying in almost zero power draw low power state for the rest of the day.

2 Introduction

2.1 Background

Fish are very important to the underwater ecosystem and are an essential food resource, and both conservationists and farmers try to measure fish activity and biomass in order to protect their population, but, current methods are a manual, labor intensive process. Solutions such as catch release, are invasive, often requiring fish to be removed from the sea for measurement.[7] [10]The FishSense project aims to automate this using underwater stereo and RBG cameras, along with machine learning techniques. The camera is fixed in a plexi-glass cylindrical module and is submerged with divers in order to detect fish movement. It is fully enclosed and houses electrical components, able to record footage and export that data into a 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.

2.2 Proposed Solution

At a high level, extended deployment periods 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. There were three approaches to the problem that were proposed. 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 and results of our bench-marking, we decided 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.

2.3 Our Contributions

To start scoping out the project details we first had to benchmark the power draw of the system in various modes. This allowed us to come to the conclusion that the only way to practically achieve a

Figure 1: Benchmarked power data. Note that the actual power draw may be lower due to us attaching a monitor and keyboard

long deployment window was by having an external board turn off the system completely when not required. After that, we made a plan to create the STM based timer board to act as a switch. Our contributions were broadly classified as follows:

- Bench-marking power draw of various system states.
- STM side firmware development
- NVIDIA Jetson side firmware development
- Interfacing both STM and Jetson hardware. This includes STM sleep timer carrier board development.

3 Related Works

There are different ways to enable an embedded system to last long periods of time. FPGAS are generally the hardware of choice when it comes to powering accelerators that are specific to ML features. [9] [4] However, we decided that the current iteration of the Fishsense module would do the ML heavy work while out of the water and not recording. So, priority would be to power the module on for certain periods of recording length.

Inspired by other research methods, [5], we explored trying to lower the power dissipation by making changes at a higher, system level. An idea to turn on the system only at certain intervals of the day came up. The STM family of micro-controller units is popular because of low cost and ease of use. According to the reference manual, the STM had several low-power modes that met the energy requirements of our system, and could be used reliably.[1] Additionally, there are HAL-layer tools [2] that could be used to abstract away the nitty gritty GPIO details that needed to be maintained.

Other low-power embedded research pointed to the importance of bench-marking and monitoring the current power consumption. [6]

Based on background research, we came to the conclusion that it would be best to potentially use an external MCU to control power to the general TX2 module [3], which could work as the brains to both control when the Realsense camera would start recording and process the ML heavy tasks.

4 Technical Material

This section will go into the technical details of our contribution. We will start with how the current system functions in order to get an understanding of what had to be changed. We will then explain how we went about reading power draw from the system in order to complete benchmarking. Then we will go into detail about our firmware code for both the STM32 and Jetson side programming. We will finally explain the interfacing process including the development of the sleep timer board.

4.1 Current System Architecture

FishSense is a handheld under- water RGBD imaging platform that measures fish length. This system can be used to derive fish biomass and health. The FishSense core technologies include depth cameras, compute platforms, software, and a sleek mechanical design. FishSense utilizes scientific advancements in underwater imaging and machine learning to collect and process data. The compute architecture consists of three parts:

- NVIDIA Jetson TX2
- Orbitty Carrier Board
- Custom Designed PowerIO module

The NVIDIA Jetson TX2 is the main compute unit of the system. Nvidia Jetson is a series of embedded computing boards from Nvidia. The Jetson TK1, TX1 and TX2 models all carry a Tegra processor (or SoC) from Nvidia that integrates an ARM architecture central processing unit (CPU). Jetson is a low-power system and is designed for accelerating machine learning applications. [3] The Nvidia Jetson TX2 board bears a Tegra X2 of microarchitecture GP10B[3] (SoC type T186 or very similar). This board and the associated development platform was announced in March 2017 as a compact card design for low power scenarios, e.g. for the use in smaller camera drones. Further a TX2i variant, said to be rugged and suitable for industrial use cases, is available for us, although it doesn't support FishSense Firmware.

The Orbitty Carrier Board acts like a small motherboard for the TX2. The Orbitty Carrier for NVIDIA Jetson TX2, TX2 4GB, and TX2i is designed to match the NVIDIA Jetson TX2/TX2i module form factor. The Orbitty's design includes 1x USB 3.0, 1x USB 2.0 OTG, 1x HDMI, 1x GbE, 1x microSD, 2x 3.3V UART, I2C, and 4x GPIO. Ideal for robotics and unmanned applications or any small form factor rugged environment. [8]

The PowerIO board is a custom power supply unit created by the FishSense team, which taken battery power as input and powers the Orbitty along with passing through gpio input pins to various LEDs. The nominal voltage supplied from the battery and to the Orbitty is 12V.

4.2 Power Benchmarking

The Nvidia Jetson was benchmarked to observe its average amperage intake during various operating modes. Theses states include:

- stand-by
- power on with RGB and recording off
- power on with RGB on and recording off
- power on with both RGB on and recording on

The benchmarking was compared to the FishSense's average power requirement for per hour, and calculated for a 2 week recording duration. The Jetson's maximum power draw per hours for such a duration was 30.95 mA. This was calculated using the maximum amount of batteries the system can be equipped with. The amperage values were observed using a Tektronix PWS4205 Programmable DC Power supply and sampled per second during a 6 minute run time covering all states. The maximum allowed power draw of 30.95 mA set a threshold for future integrated physical system requirements. This benchmarking task helped us decide the path we had to take to implement the low power state, since such a low power draw compared to the active power draw requires a completely off low power state.

The post implementation low-power benchmark came down to around 30mA, which is just the power of the STM32. This could probably be reduced further by adjusting the STM32 clocks.

Note that the procedure to connect the bench power supply to the system is to just connect the leads to the battery input of the PowerIO board. Supply should be set to 12V, 5A. The PowerIO can also be bypassed using connecting the leads directly to the Orbitty.

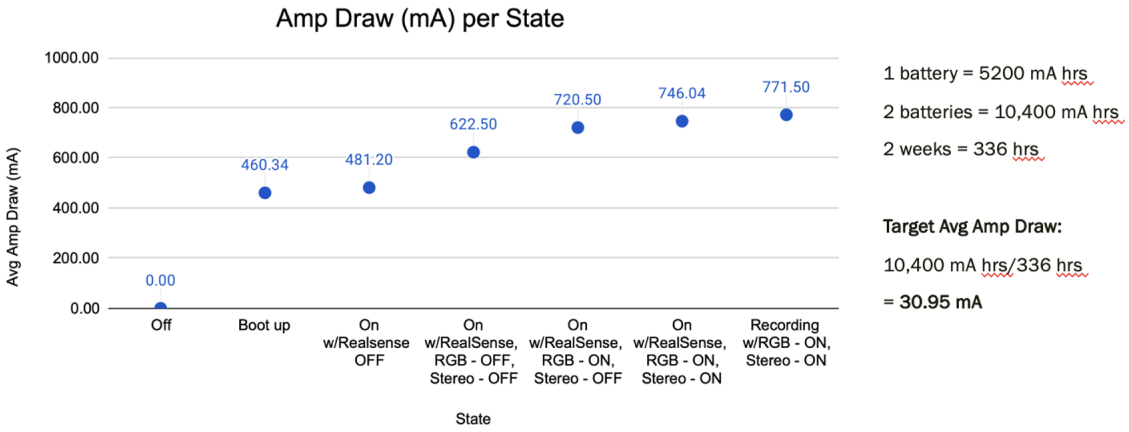

Figure 2: Benchmarked power data. Note that the actual power draw may be lower due to us attaching a monitor and keyboard

4.3 Changes to System Architecture

The changes we introduce to the system includes a new power states which FishSense can cycle between in user defined times. For example, in a two week deployment period, during the day, on the top of every hour, the system will power on from sleep and start recording. The system will continue recording for a set amount of time and then turn off completely. Again, after a set amount of time the system will power on again and start recording. This cycle continues.

In order to completely power off the system we need a way to turn off power supplied by PowerIO board, and also set a Real Time Clock to time when the system turns back on again. This is accomplished by a low power STM32 G0 board. The STM has a low power RTC module which can accurately keep time and trigger the system to turn on.

The STM board act as a clock master while the NVIDIA TX2 acts as a timing master in the following manner:

- Tx2 sends sleep signal to STM32
- STM32 turns of entire system
- STM32 wakes up entire system after set amount of time
- Tx2 records for certain time and sends sleep signal to STM32 again

This is because we have to ensure that all recording processes are complete before stopping the system. That is why the TX2 sends a signal to the STM to turn the system off.

Now coming to how the STM actualy is interfaced to the PowerIO board in order to turn off the system. The current revision of the PowerIO board doesn't have an enable pin which would have allowed the STM to be directly connected to the PowerIO using the GPIO pins. Therefore we had to have a separate carrier board designed to help interface the STM to the PowerIO. This is explained in the section below in Sleep Timer Board.

4.4 Sleep Timer Board

The Sleep Timer Board is a PCB which the STM32 G0 board in physically placed on, and creates a switch to to turn power flow on and off between the PowerIO and the Orbitty. This is done by way of the PD9 GPIO pin on the STM32. This is traced out to a series of MOSFETS that are toggled on by GPIO pull up. These MOSFETS close the circuit delivering 12V power from the PowerIO to the Orbitty.

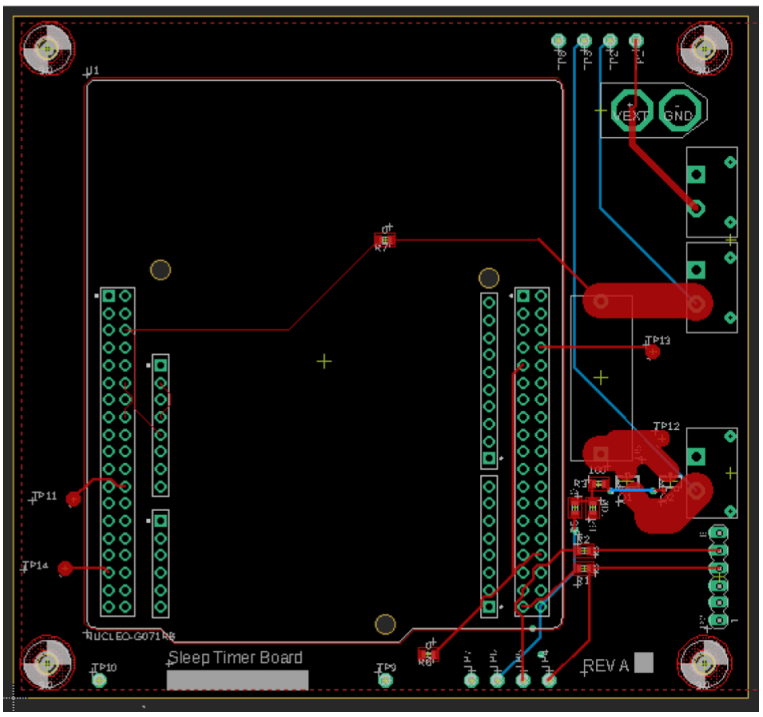

Figure 3: Sleep timer board. The first part highlighted in red towards the right is where the power IO board connects. The bottom two red highlighted parts in the shape of a ”v” is where the board connects to the Orbitty carrier. The light blue line in between the red ”v”s are where the MOSFETs in the next figure sit.

4.5 STM32 Firmware

The STM's primary goal was to utilize functions to establish a timer. The STMCubeIDE uses multiple functions, included in the HAL manual. The several subsections of the module include an RTC timer and LPTIM1 timer, including other timer modification sections which were observed but not used. Primarily, the RTC and LPTIM1 were observed, though the RTC was the primary timer source. Among the system core requirements were the GPIO and SYS pins, both of which were important in the configuration to alternative sources. Analog timers were included but not used, as well as connectivity, multimedia, computing, and middleware settings. Each setting had adjustable modes and configurations.

The primary function of the HAL APIs are to implement a user call-back function mechanism. Through this mechanism, peripheral level initialization/initialization functions can be performed using clocks, GPIOs, interrupts, and DMA (direct-memory access). Interrupt events and error events are also part of the working STM function.

The real-time clock (RTC), in comparison to the low-power timer (LPTIM1) relieved the entire system of a power capacity requirement, as the STM itself was sufficiently low-power enough to uphold a firmware ready to power the FishSense module. Important functions used in the code were BSP_LED_On, which turned on an LED, and BSP_GPIO_Off, which turned off an LED. The STM was configured initially using an LED to test the functionality of the firmware.

The STM Firmware turns off the system by setting the PD9 GPIO pin to 0. This turns off the power as explained in the previous section. The firmware then sets an alarm to turn on after a user defined amount of time. This alarm triggers the same GPIO pin to go to 1.

To look for the power down event trigger coming from the TX2, the STM has an input GPIO check for a 1 signal to clock down to 0. This is a downward edge trigger. When the signal is detected the system powers down again continuing the cycle. The current prototype has the STM32 powered by a laptop, since we haven't yet designed the battery connector to the STM32.

4.6 Nvidia Jetson TX2 Firmware

The TX2 is programmed to start immediately after being powered on. When the system is turned on, it starts booting into the Ubuntu OS. The FishSense firmware is then automatically started. The main part of the firmware code is found in the rs_save.cpp file. This consists of a few overhead functions such as file pointer initialization, and GPIO. This is then followed by a while loop which runs until the set duration to continuously record and save clips as bag files.

The original code had firmware implemented to handle the magnetic reed switch which is used by the diver to manually start and stop video recording. This code was removed to allow the recording to start immediately.

The firmware starts a clock once the recording start and keeps a track of duration. After the set duration, the firmware stops recording, and closes all file pointers. It then sends a signal to the STM32 board to power off the system. This is in the form of a GPIO pin on the TX2 being set from 1 to 0. This GPIO pin is always set to 1 when power on.

5 Milestones

5.1 Previous Milestones

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

- Milestone 1.1: Contact researches to find out how long and often FishSense needs to be active and collecting data from the RealSense Camera
- Milestone 1.2: Benchmark the current power consumption in FishSense's active mode
- Milestone 1.3: Given the frequency of active mode (using deliverable 2 information), calculate the maximum power consumption that low power mode can have

Deliverable 2: Create a low power mode for the Nvidia Jetson

- 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
- Milestone 2.2: Create a low power mode on the Nvidia Jetson by turning off cores slowing down clock speed, configuring different RealSense modes
- Milestone 2.3:Extra: try to make active mode more low-power by changing RealSense settings

Deliverable 3: Assess how well low power mode does

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

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

5.2 Previous Milestone Progress Changes

Deliverable 1 - Week 5:

- Milestone 1.1: Contact researchers to find out how long and often FishSense needs to be active and collecting data from the RealSense Camera - Based on our project mentor, Nathan's, guidance, we postponed this milestone to the last week of the project. This is because if we were to contact researchers at the beginning of the project , they would be giving us a long list of either unreasonable or unviable requirements which we would not know how to handle before we implement the MVP. These requirements would be essentially useless for us at the moment. Our current goal is then to get the MVP up and running and present to the researchers what our system is capable of, and give them a list of parameters, such as wake up time, cycle time, etc. that we would be able to change for them.

- Milestone 1.2: Benchmark the current power consumption in FishSense's active mode - This milestone has been completed, and results have been presented in our oral project update.
- Milestone 1.3: Given the frequency of active mode, calculate the maximum power consumption that low power mode can have - Based on the current FishSense setup and our power consumption figures from the previous Milestone, we have calculated the maximum power consumption that low power mode can have. Based on the results we have found that the parameters placed by us in Deliverable 2, that is to create a low power mode purely using the NVIDIA Jetson alone, is not going to work. The NVIDIA Jetson board cannot realistically be placed in a low enough power state to hit our targeted current numbers, and cannot be turned off on its own since it does not have an internal RTC to start itself up again.

  The calculations we did were explained in our presentation for the oral project update, which is below for reference.

  As can be seen in the image below, the target average current draw if we want the system to last for 2 weeks permanently in a low power state should be around 31mA, whereas the current draw for the Jetson alone is 460mA in idle. Even without considering that we have to switch to a high power state, this setup is obviously not feasible without completely powering off the system.

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

- Milestone 2.1, 2.2: As discussed above, the method of achieving low power mode in this deliverable is unfeasible. We have thus switched this Deliverable out with our Extra Deliverable 4 that we proposed in our Project Specification Document. This method is to program directly onto an alternate STM32G0 board the capability 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 FishSense camera is in sleep mode. This would ensure enough power is available during the full duration of its deployment.

Extra Deliverable 4 (Replaced Deliverable 2): Use the RTC onboard the STM32 to turn off the FishSense System completely- Week 6 7

- The milestones for this deliverable have been updated and have been split up into more detailed steps after we pivoted to using the STM32 board. These details can be seen in the next section on Update Milestones.
- Milestone 4.1: Create a basic RTC timer using the STMCube IDE and flash to the STM32G0 for proof of concept testing - We needed to start learning the STM32 development environment along with creating a proof of concept by flashing a LED light on the STM32 based on an RTC timer interrupt that we programmed. The video link below shows our program running on the STM board in debug mode. The program sets up the RTC on the STM32, switches to the LEC oscillator crystal, and sets up the onboard LED4 to light up after 25 seconds have passed. The program sets on program start, the RTC to be 12:00:00, and then starts counting.

5.3 Updated Milestones

Deliverable 1: Program the STM32 to act as an external timer with an RTC, Priority 1 Scheduled Deadline: 5/24

- Milestone 1.1: Read documentation on how to create an RTC time on the STM32 [2] [1]
- Milestone 1.2: Create a basic RTC timer using the STMCube IDE and flash to the STM32 to test that it works
- Milestone 1.3: Create a low power RTC timer using the HAL library in a .ioc file.
  - Set the LSE oscillator on the STM

  - Set the time correctly
  - Configure GPIO to supply power to carrier board on PD9
    * Custom carrier board already designed by Nathan
    * Carrier board will bypass the power specs for power io to cut and supply power to TX2
  - Flash to the STM

Deliverable 2: Test that the STM32 is supplying power during timed intervals, Priority 2 Scheduled Deadline: 5/28

- Milestone 2.1: Connect the STM32 to the custom power IO carrier board and time when power is on and off using power supply

Deliverable 3: Set up communication between the TX2 and STM32, Priority 1 Scheduled Deadline: 5/24

- Milestone 3.1: Figure out how the TX2 can communicate with the STM32 and modify the current firmware code to do so
  - TX2 tells recording to stop, unmounts SSD
  - Send timing data to STM32, so it can resume timer
  - STM32 or TX2 cuts power to TX2

Deliverable 4: Connect TX2, custom carrier board, and STM, Priority 2 Scheduled Deadline: 5/30

- Milestone 4.1: Test that TX2 and STM32 are communicating at correct time intervals by observing when the TX2 turns on and off with the power supply
- Milestone 4.2: Test that system is also recording during on interval

5.4 Milestone Problems and Successes

All deliverables except for 3 and 4 were completed by the team. The STM firmware code was completed and tested. The STM code correctly writed to PD9 after a fixed amount of time.

Deliverables 3 and 4 were not completed by the team. Deliverable 3 could not be completed because the existing TX2 firmware and flashing process could not be completed on an Nvidia TX2i, which was the only module available for most of the quarter due to the TX2 modules being shipped to Florida for testing. A module became available and flashed the day before the video was due. Prior to that, many of the TX2 modules had a Linux OS flashed that could not properly run the firmware. There were mounting and savescript problems that required reflashing of the OS. The firmware script was running on a Linux system that seemed to have problems on Therefore, the team had around a day to flash the TX2 code and debug the code. The firmware code was completed prior to the day it could be tested however, so the Fishesen team will have TX2 code to work with in order to communicate with the STM.

Deliverable 4 could not be completed due to problems with the sleep timer board. The sleep timer board connects the Orbitty carrier board to the power IO board through a series of MOSFETS. The STM32 PD9 pin acts as a switch to close the MOSFETS and allow power to flow through the power IO board to the Orbitty. Using an oscilloscope, we were able to test and see that the STM board was correctly writing to PD9 at set intervals of time. We could also see that the power IO board was working correctly to provide power to the sleep timer board. The sleep timer board, however, had MOSFETs in the wrong mode. Therefore, the circuit could never be closed, even when PD9 was high.

6 Conclusion

In conclusion our main goal was to create a low power mode for the FishSense system in order to allow for longer deployments. We did this by creating a sort of duty cycle for the system, where it would power on every hour for a few minutes and then sleep for the rest of the hour. This was done by using an external STM32 board to act as a RTC and cut and restore power to the system. The interface to cut power was made by designing a sleep timer carrier board for the STM.

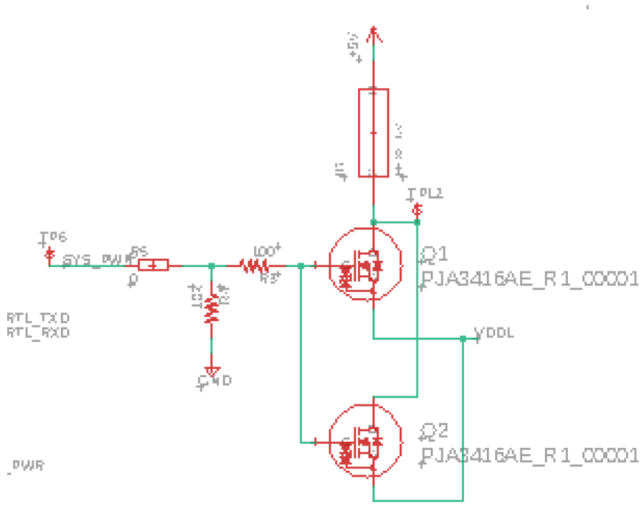

Figure 4: MOSFETs on the Sleep timer board that were in the wrong mode.

References

[1] Description of stm32g0 hal and low-layer drivers, um2319 - rev $2_2020, Nov2020$.

[2] Stm32g0x1 advanced arm®-based 32-bit mcus, $rm0444_2020, Nov2020$.

[3] Nvidia jetson, May 2022.

[4] Shyam Jagannathan, Kumar Desappan, Pramod Swami, Manu Mathew, Soyeb Nagori, Kedar Chitnis, Yogesh Marathe, Deepak Poddar, Suriya Narayanan, and Anshu Jain. Efficient object detection and classification on low power embedded systems. In *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pages 233–234, 2017.

[5] Jihong Kim and Tajana Simunic Rosing. Power-aware resource management techniques for low-power embedded systems, 2006.

[6] Dongkun Shin, Hojun Shim, Yongsoo Joo, Han-Saem Yun, Jihong Kim, and Naehyuck Chang. Energy-monitoring tool for low-power embedded programs. *IEEE Design Test of Computers*, 19(4):7–17, 2002.

[7] N.J.C. Strachan. Length measurement of fish by computer vision. *Computers and Electronics in Agriculture*, 8(2):93–104, 1993.

[8] Connect Tech. Orbitty carrier for nvidia® jetson™ tx2/tx2i, Oct 2021.

[9] Onur Ulusel, Christopher Picardo, Christopher B. Harris, Sherief Reda, and R. Iris Bahar. Hardware acceleration of feature detection and description algorithms on low-power embedded platforms. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–9, 2016.

[10] D. J. White, C. Svellingen, and N. J.C. Strachan. Automated measurement of species and length of fish by computer vision. *Fisheries Research*, 80(2-3):203–210, September 2006.