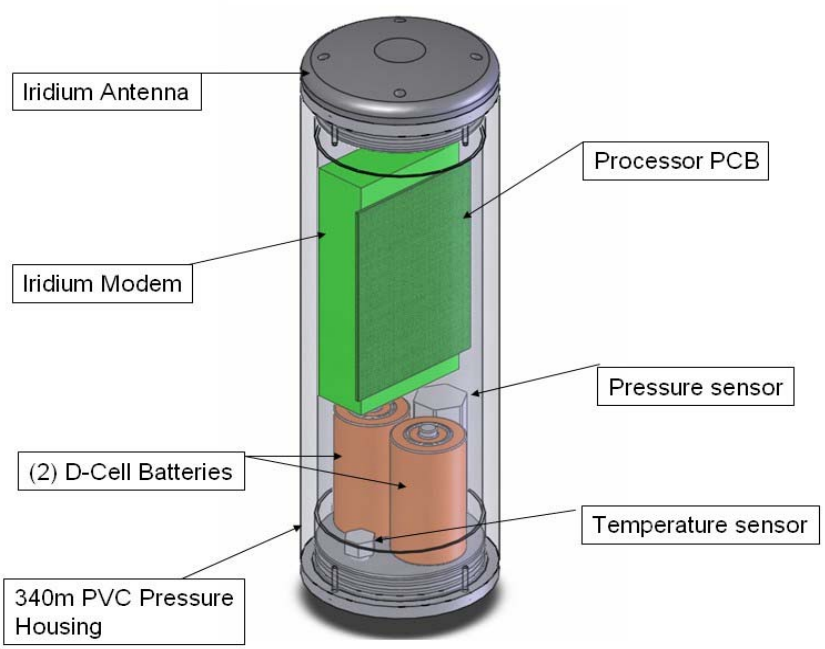
Turbidity Current Monitor



Final Documentation

Jack Baskin School of Engineering
University of California, Santa Cruz

SoE Sea Monkeys

Stefan Gadomski
sgadomsk@ucsc.edu
Electrical Engineering

Edgard Rincand
eggois@yahoo.com
Electrical Engineering

Table of Contents

Table of Contents Cont.

Project Overview

Problem

Throughout the world's oceans there are events called turbidity currents that occur in submarine trenches and canyons. As shown in figure 1, a turbidity current also known as a density current is a current of rapidly moving, sediment-laden water moving down a slope through air, water, or another fluid. The current moves because it has a higher density and turbidity than the fluid through which it flows[1]. Other than that not a lot more is known about these unpredictable events. Most importantly these events happen in the Monterey canyon located in the Monterey bay. The researchers at MBARI (Monterey Bay Aquarium Research Institute) were losing expensive research instruments trying to learn more about these events. This became very cost inefficient and caused them to think of a different strategy to learn more about these events.

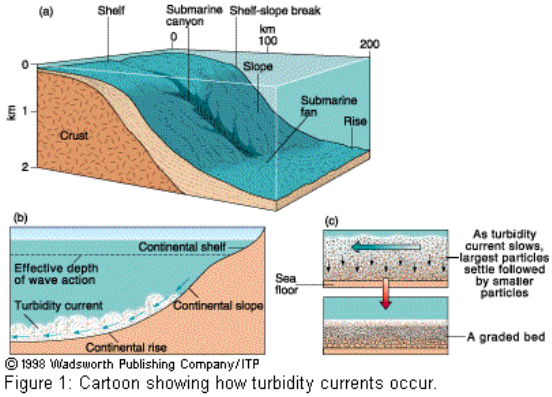


Figure 1: Cartoon showing how turbidity currents occur.

Currently MBARI has a device that records pressure and temperature attached to a mechanism that has the ability to detect a turbidity current. Once this mechanism detects a quickly moving current it releases the recording device. Surrounded by a flotation material the device will float to the surface and send a signal notifying MBARI it has surfaced. This device was developed for animal research rather than turbidity current research. As a result there are some drawbacks using this device for turbidity current research. One of these drawbacks is the fact that the device does not transmit a signal until it has been surfaced for twenty-four hours. Secondly the data cannot be retrieved until the device has been picked up. Picking up the device requires the use of a ship which is very expensive. Thirdly the device itself is very expensive (~$10,000). Overall this device is not efficient for use with researching turbidity currents.

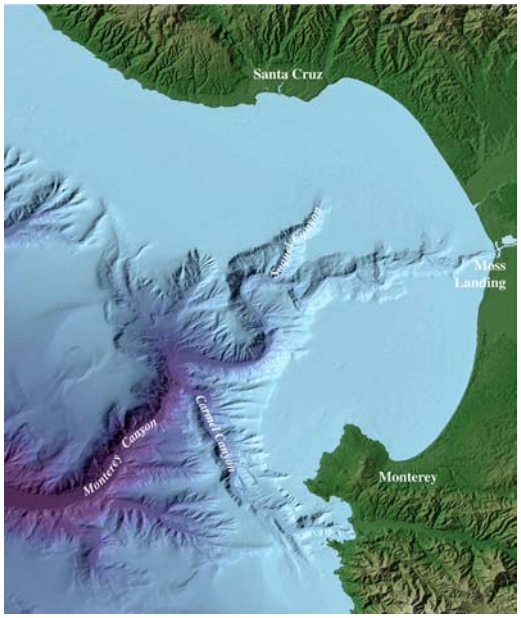


Figure 2: Map showing the Monterey Canyon.

[1] http://en.wikipedia.org/wiki/Turbidity_current
Figure 1: http://www.cliffshade.com/colorado/images/continental_margin2.gif
Figure 2 : http://coralnotesfromthefield.blogspot.com/2007_04_01_archive.html

Solution

MBARI needed a device that would improve upon the drawbacks of the existing device used. Brett Hobson, a mechanical engineer at MBARI proposed the creation of such a device. This device will work similar to the existing device recording temperature and pressure attached to the turbidity current detector. It will also record data until it reaches the surface. However, once surfaced it will transmit the last twenty-four hours of data via satellite to MBARI. Since the last twenty-four hours of data is the most critical MBARI can wait until the device has washed ashore to retrieve the rest of the data if necessary. Overall the device will help MBARI save money and make the retrieval of data easier.

Functional Requirements

1. Precisely determine the time a STEDS sensor package is released.
2. Record the depth, temperature, and water clarity level every 5 seconds for 1 hour prior to release until the unit surfaces (depth < 5m).
3. Transmit the recorded data ashore.

System Design Requirements

1. Measure pressure between 0 and 500 psi at an accuracy of 0.25 % of full scale.
2. Measure temperature to within 0.1 degrees Centigrade.
3. Use a real-time clock with a drift of less than 2 minutes/year.
4. Use an Iridium data radio with a pressure resistant antenna.
5. Battery pack must power all equipment for 6 months at 6 deg. C, with enough capacity left over to complete the Iridium data transfer.
6. Archive all data on flash memory with a 6 month capacity (at 10 second interval).
7. Everything must fit into a 3” schedule 80 PVC tube (2.864” ID).
8. Use a white LED and a photo diode to measure light attenuation as a function of water clarity.
9. Must cost $2000 or less.

The SoE Sea Monkey Team

Team Work Schedule

The work schedule planned out at the beginning the quarter allowed the team members to see what time each individual would be working. Due to the team only consisting of two people the scheduled working times were fairly similar. Work was scheduled for weekends as well but was only done if necessary. The table below shows the working times of each individual.

A.M.	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
11		Stefan					
P.M.							
12	Stefan/Edgard	Stefan		Stefan		Stefan	Stefan/Edgard
1	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard
2	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard
3	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard
4	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard	Stefan	Stefan/Edgard
5	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard
6	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard
7	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard	Stefan/Edgard
8		Edgard	Stefan/Edgard	Edgard	Stefan/Edgard	Edgard	
8		Edgard		Edgard		Edgard	
9		Edgard		Edgard		Edgard	
11		Edgard		Edgard		Edgard	

Table 1: Weekly schedule of team showing when a team member planned to work.

Hours Spent

Below are the various hours spent on the project by week. Notice the dip for week five (midterms) and the exponential push for week ten:

Week	Time Spent (Hours)
1	24
2	52.5
3	47
4	69.5
5	44.5
6	56.5
7	57.5
8	70.5
9	51
10	92
Total	565

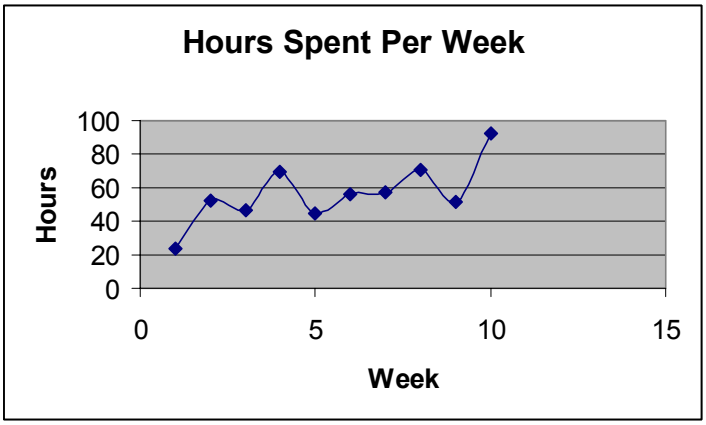


Table 2: Total hours worked per week by group. Figure 3:Line graph of table 2.

Below are the various hours spent on the project by task:

MBARI meetings	5
Documentation	30
Research	33
Schematic	82
Layout	113.5
Devkit	18
Modem	97
RTC	41
SD Card	17.5
Power Calculations	32.5
Ordering parts	21
ADC12	14.5
Software	24
Soldering	29
Poster	7

Table 3: Total hours worked on individual tasks.

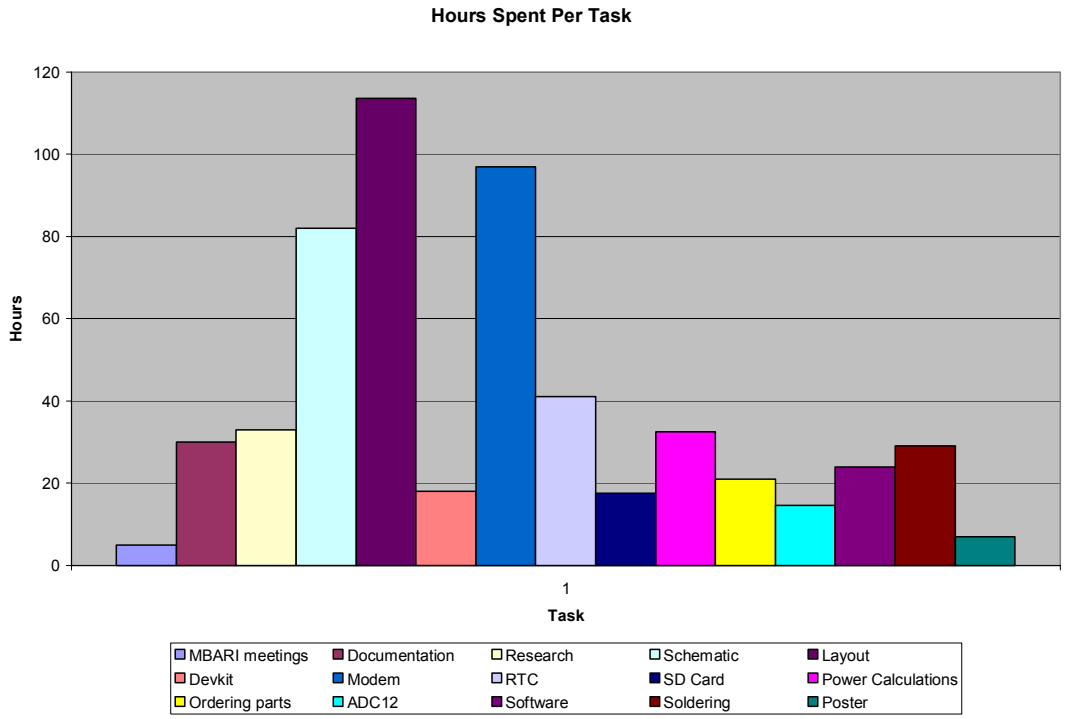
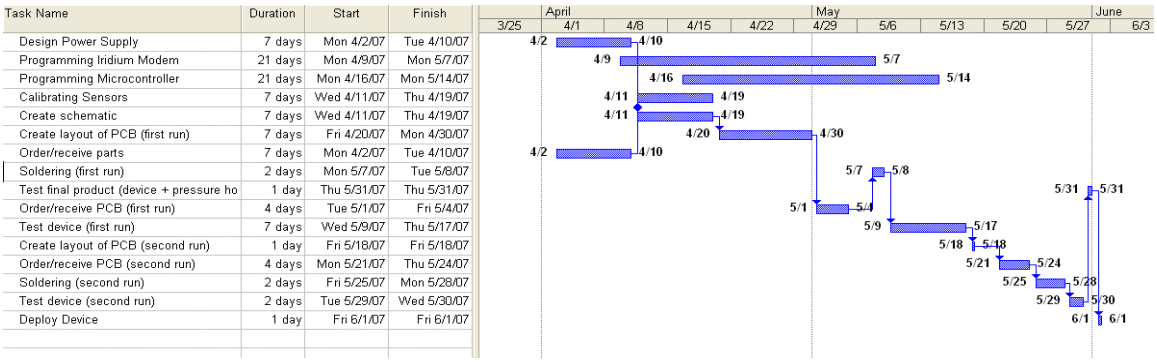


Figure 4: Bar graph of table 3.

Planning

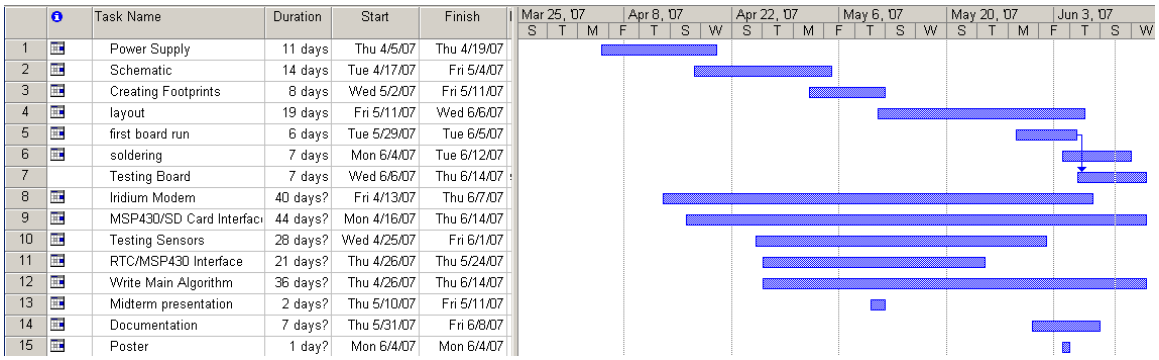
At the end of winter quarter a Gantt chart was created in order to identify all the tasks needed to be completed in order to successfully complete the project. This Gantt chart is shown below:



Task Name	Duration	Start	Finish
Design Power Supply	7 days	Mon 4/2/07	Tue 4/10/07
Programming Iridium Modem	21 days	Mon 4/9/07	Mon 5/7/07
Programming Microcontroller	21 days	Mon 4/16/07	Mon 5/14/07
Calibrating Sensors	7 days	Wed 4/11/07	Thu 4/19/07
Create schematic	7 days	Wed 4/11/07	Thu 4/19/07
Create layout of PCB (first run)	7 days	Fri 4/20/07	Mon 4/30/07
Order/receive parts	7 days	Mon 4/2/07	Tue 4/10/07
Soldering (first run)	2 days	Mon 5/7/07	Tue 5/8/07
Test final product (device + pressure ho	1 day	Thu 5/31/07	Thu 5/31/07
Order/receive PCB (first run)	4 days	Tue 5/1/07	Fri 5/4/07
Test device (first run)	7 days	Wed 5/9/07	Thu 5/17/07
Create layout of PCB (second run)	1 day	Fri 5/18/07	Fri 5/18/07
Order/receive PCB (second run)	4 days	Mon 5/21/07	Thu 5/24/07
Soldering (second run)	2 days	Fri 5/25/07	Mon 5/28/07
Test device (second run)	2 days	Tue 5/29/07	Wed 5/30/07
Deploy Device	1 day	Fri 6/1/07	Fri 6/1/07

Figure 5: Gantt chart created at the end of winter quarter showing predicted progress on the project.

This schedule was followed for the first few weeks but the team fell behind schedule around week three of the quarter. The Gantt chart below represents the actual durations and completion dates of various tasks related to the project:



	Task Name	Duration	Start	Finish
1	Power Supply	11 days	Thu 4/5/07	Thu 4/19/07
2	Schematic	14 days	Tue 4/17/07	Fri 5/4/07
3	Creating Footprints	8 days	Wed 5/2/07	Fri 5/11/07
4	layout	19 days	Fri 5/11/07	Wed 6/6/07
5	first board run	6 days	Tue 5/29/07	Tue 6/5/07
6	soldering	7 days	Mon 6/4/07	Tue 6/12/07
7	Testing Board	7 days	Wed 6/6/07	Thu 6/14/07
8	Iridium Modem	40 days?	Fri 4/13/07	Thu 6/7/07
9	MSP430/SD Card Interfac	44 days?	Mon 4/16/07	Thu 6/14/07
10	Testing Sensors	28 days?	Wed 4/25/07	Fri 6/1/07
11	RTC/MSP430 Interface	21 days?	Thu 4/26/07	Thu 5/24/07
12	Write Main Algorithm	36 days?	Thu 4/26/07	Thu 6/14/07
13	Midterm presentation	2 days?	Thu 5/10/07	Fri 5/11/07
14	Documentation	7 days?	Thu 5/31/07	Fri 6/8/07
15	Poster	1 day?	Mon 6/4/07	Mon 6/4/07

Figure 6: Gantt chart showing the actual progress of the team at the end of Spring quarter.

Budget

The requirement for the device to cost $2000 or less has been achieved. The most expensive component of the device is the Iridium modem. The table shown below represents the total amount spent while designing the first version of the device.

Item	Cost
PCB Parts	$211
PCB (2 boards)	$90.30
Modem	$512
Antcom S31R16RR-P-XT-U (modem Antenna	$160
MSP-600 (pressure sensor)	$96
Two Lithium Batteries	$170
Total Cost	$1,239.30

Table 4: Estimated cost of materials for project creation.

This total amount does not include the various chip samples received along with the thermistor that was traded for a UCSC t-shirt. The table below reflects the prices of the samples and thermistor received in order to estimate the total cost of one device.

Item	Cost
MSP430F169	$13.24
MAX3221EAE	$4.05
MAX603	$4.77
MAX1626	$2.06
2 PIN MICROFIT	$1.92
3 PIN MICROFIT	$0.52
DS3231SN	$2.00
MAX6029SO	$2.19
2mm EJECTOR HEADER	N/A
THERMISTOR	$120
Total Cost	$150.75

Table 5: Estimated cost of samples acquired.

Adding these two total costs together gives a total cost of $1390. This price reflects an estimate regarding how much one of these devices would cost to build.

Note: In order to send data through the Iridium network, a data plan must be purchased ($156/yr + $300 airtime). This cost is not included into the costs above.

The Project

Initial Design

At the end of winter quarter the hardware design proposed was the following block diagram:

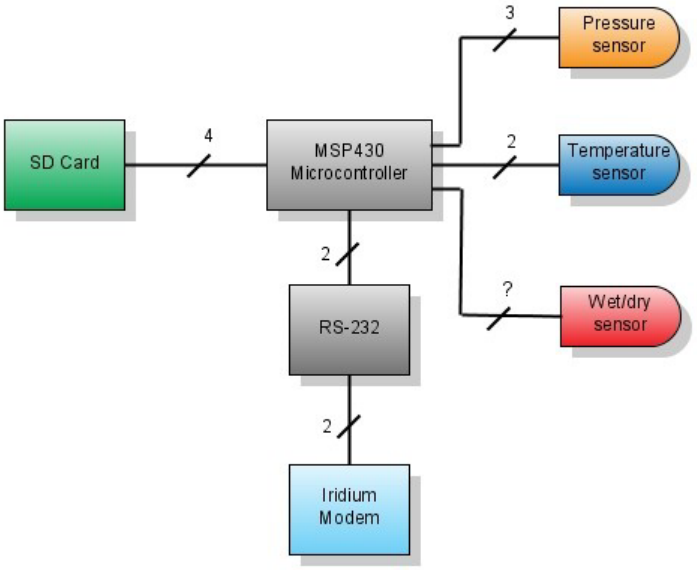


Figure 7: Initial hardware design at the end of winter quarter.

Modifications

After spending more time with the project some modifications were made to the initial design. The wet/dry sensor was removed after realizing the pressure sensor could be used for this purpose and turbidity sensor was added once the official requirements were received from Brett. After realizing it would be difficult to implement an accurate real-time clock with software given the group's minimal programming experience a real-time clock chip was chosen for use in the design. An external crystal and voltage reference chip were also added to the design once it was realized both of these features included on the MSP430 were unreliable and inaccurate.

Final Design

The block diagram shown below represents the final design being used for the device. This block diagram does not include the external crystal or voltage reference chip:

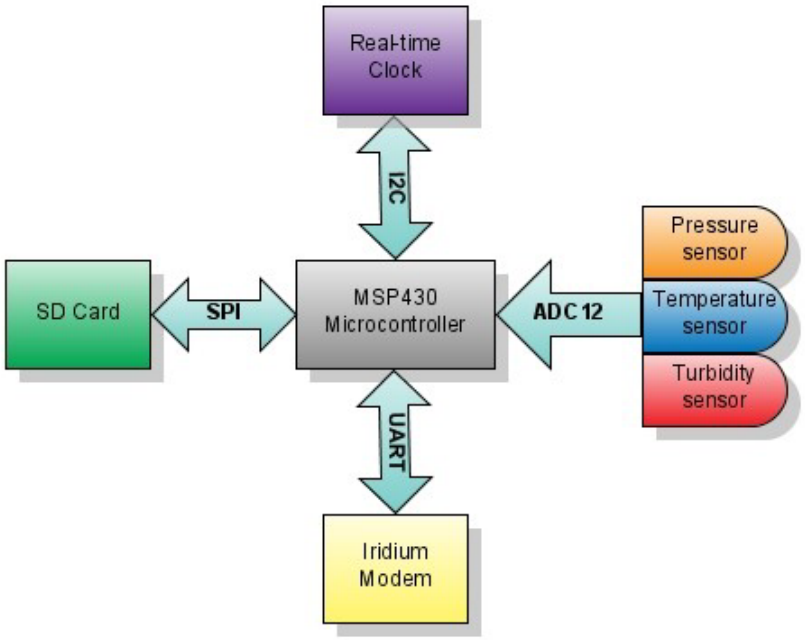


Figure 8: Final hardware design block diagram.

Hardware Technical Breakdown and Design

Microcontroller

The device required a component that could handle and perform the tasks required to implement all the features needed for the device. A microcontroller was the perfect choice for this task. The MSP430 microcontroller from Texas Instruments was chosen for use for its ultra-low power consumption, included features (USART, lots of I/O and ADC12), low cost development kit and the availability of help and support.

The MSP430 family consists of 107 different models, each with different features and internal memory size. The MSP430F149 was chosen from the family because it included various low power modes, ADC12, UART, SPI, 48 GPIO pins, 60KB of flash and 2KB of RAM on a 64LQFP (low profile quad flat pack) package. This choice was later changed to the MSP430F169 because it included the same features as the 149 but also had an I2C module implemented.

In order to prototype the design a development kit was required. Texas Instruments offers cheap and easy to use development kits for the MSP430 family. The MSP-FET430140 development kit was used due to it's compatibility with the 169. This dev kit is connected to a desktop PC via a 14-pin JTAG connector, which then plugs into either a USB or parallel port. The dev kit also comes with the IAR Embedded Workbench Kickstart software. This software is an integrated development environment for building and debugging code written for the MSP430.

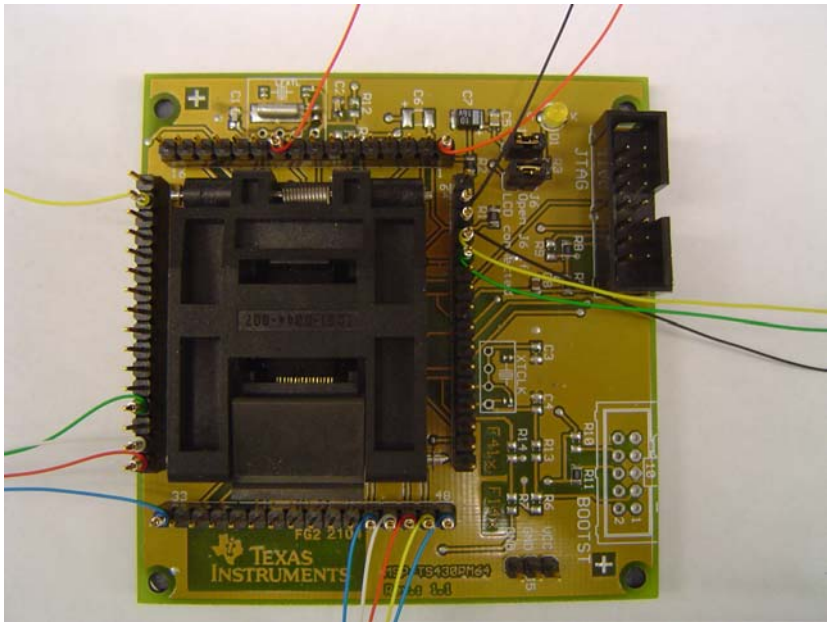


Figure 9: Picture of the MSP430 DevKit used.

Real-time Clock

Toward the beginning of the design process the decision was to implement a real-time clock module in software. It became apparent that it would be somewhat difficult to write code for an accurate real-time clock given the amount of previous coding experience in the group. Thus a design decision was made to use an external real-time clock chip. After researching various models the DS3231 from Maxim was chosen due to its low cost, small size and accuracy of ± 2 minutes a year. The DS3231 has the ability to maintain seconds, minutes, hours, day, date, month, and year information including corrections for leap year as is done through an I2C bidirectional bus.

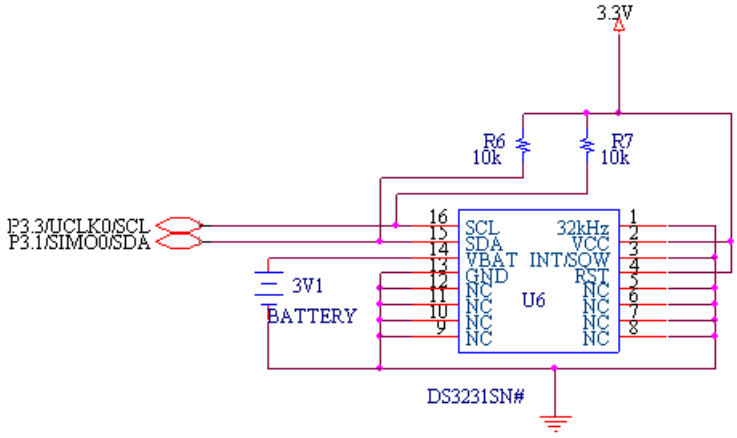


Figure 10: Schematic symbol and various wires of the DS3231.

Modem

In order to transmit data measurements a communication component was needed. The component would need to ability to send a message from anywhere within the Monterey bay at water level. Cell phone reception becomes scarce at water level so communication via satellite seemed like the next logical choice. A modem capable of transmitting short bursts of data (SBD) via the Iridium satellite network was chosen for this task.

The 9601-D-I from Nal Research was chosen mainly because of its compact size. Its dimensions of 4.16" x 2.21" x 0.51" make it perfect for use since space inside the pressure casing will be minimal. Data is sent to the modem through an RS232 interface while the modem itself is controlled with attention (AT) commands. The modem is capable of sending 205 bytes/message and receiving 135 bytes/message.

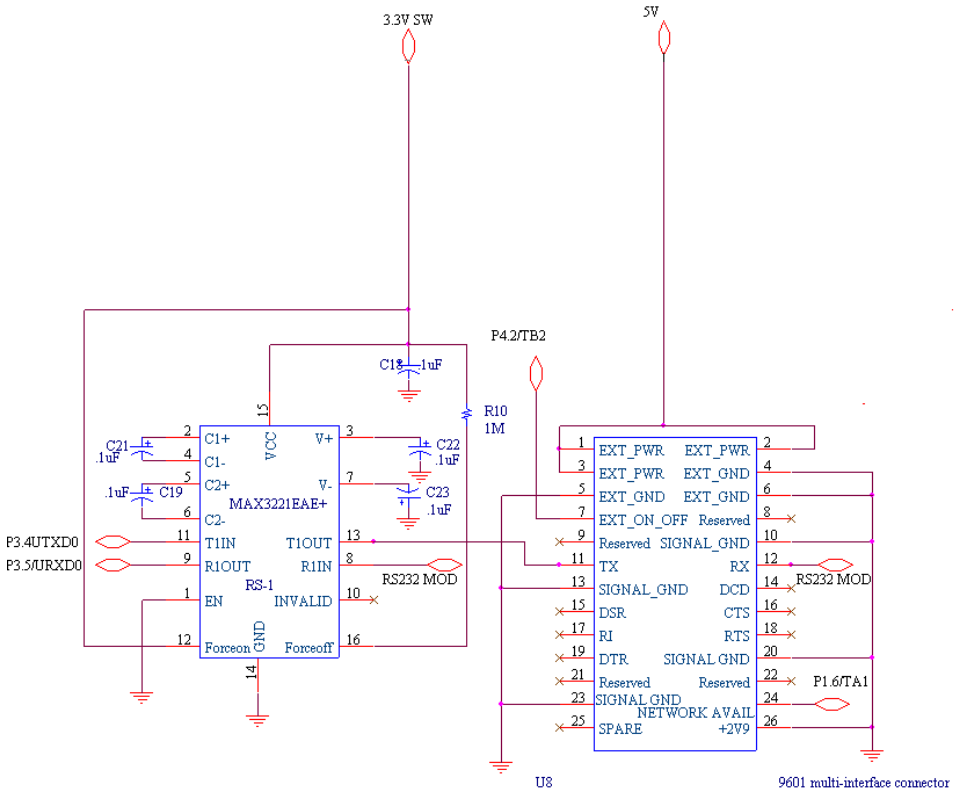


Figure 11: Schematic symbols and various wires for the Iridium modem and MAX3221 chip.

External Memory

The device needed the ability to store large amounts of data in a power-efficient manor. A secure digital (SD) card was chosen for this task due to its compact size, low power consumption and easy access. The main selling point for use of the SD card was the availability of an application note from Texas Instruments regarding instructions and source code to assist with interfacing a MSP430 to an SD card.

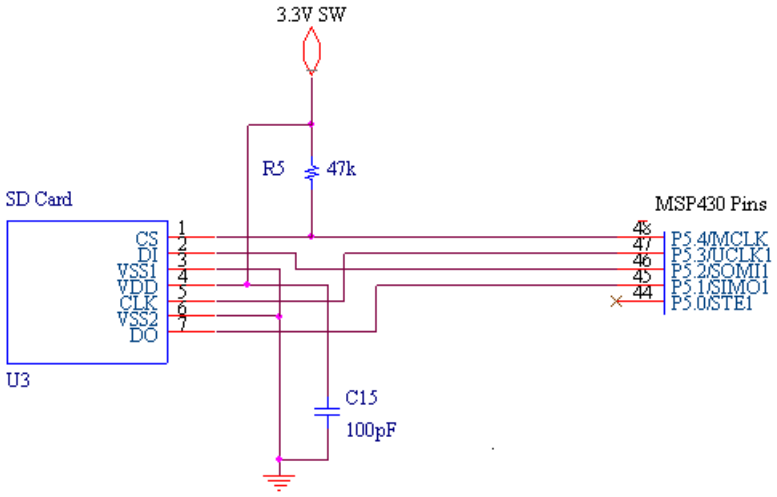


Figure 12: Schematic symbol and various wires for the SD card.

Voltage Reference

The device used an external voltage reference for the ADC12 because the MSP430 internal voltage reference is unreliable. Therefore using an external voltage reference enables the ADC12 reference to be exactly 2.5V. The voltage reference that the device uses is the MAX6029 from Maxim.

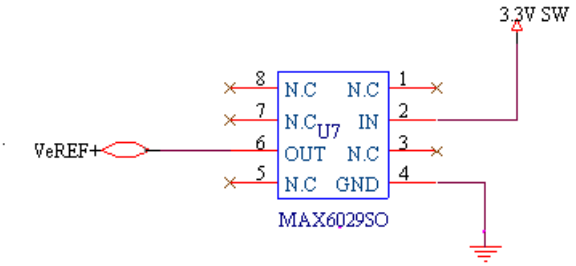


Figure 13: Schematic symbol and various wiring for the MAX6029.

External Crystals

Two external crystals are used for the MSP430. The device uses a 32 kHz crystal and a 1MHz crystal. The device needs an external crystal because the internal clock of the MSP430 is inaccurate. CMJ206T from Citizen Systems America Corp is the 32 kHz crystal used. The HC-49U from ECS INC is the 1MHz crystal used. The 1MHz crystal is used because it would be a beneficial feature to be able to increase the baud rate for the

UART. However the 1MHz Crystal is not necessary. If you refer to the appendix, one will notice that the pins of the 1MHZ crystal are connected to pins 52 and 53. If one decided not to implement the 1MHZ crystal, pin number 52 will be left open and pin number 53 will be tied to ground. The pins of the 32 kHz crystal are connected to pins 8 and 9 of the MSP430.

JTAG Connector

The JTAG (Joint Test Action Group) connector was implemented in the design of the device in order to be able to test, debug and program the MSP430 once it was attached to the PCB (Printed Circuit Board). The JTAG is connected using a 14-Pin stake header. When testing the board the JTAG worked with the MSP430.

Reset Button

The Reset Pushbutton is also used for testing. This was implemented because while testing, the device will be in one of the three states. Even if the power is cut off, the device will remain in that same state when it is turned back on. For example, if the device is in state 1 and you turn it off; when you turn the device back on it will still be in state 1. Therefore in order to fix that testing problem the device has a reset button in order to restart the device. The reset pushbutton that the device uses is the part number 71642 from Jameco.

The MSP430 requires a minimum pulse length of 2μs at the reset pin (58) in order to internally accept a software reset. Using the equation:

$$t = RC$$

the values of R and C can be solved for to satisfy this requirement. Using a 47k resistor requires the capacitor to be a minimum 42.5pF.

Pressure Sensor

The pressure sensor the device is using is the MSP-600-500-P-3-D-4 from Measurements Specialist, Inc. This pressure sensor was chosen because of its high accuracy, the 316 stainless steel enclosure and is able to measure up to 500 psi. The output voltage of the MSP-600 is between .5 to 4.5V at 5V. This caused a problem because the maximum input voltage the ADC12 can register is 2.5V. Any input voltage

above 2.5V will be registered as 2.5V. Therefore a voltage divider is used to make the output voltage of the MSP-600 to drop down to 2.5V or lower. This was done by using the equation:

$$R2 = (VoutR1)/(Vin-Vout)$$

The values of Vin and Vout are known and one can choose a reasonable value for R1, in this case it was 20kΩ. The calculated value for R2 turns out to be 25kΩ. Therefore in order to drop the voltage down to 2.5V, a voltage divider was created using a 20kΩ and a 25kΩ resistor. Figure 14 shows how the pressure sensor was implemented in the design.

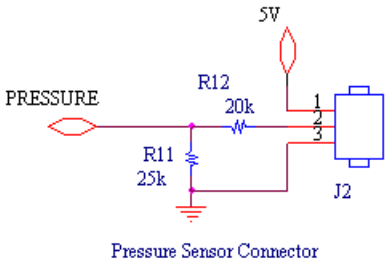


Figure 14: Schematic symbol and various wiring for the pressure sensor connector.

Temperature Sensor

The type of temperature sensor that is used for the device is a thermistor. A thermistor is an "electronic component that exhibits a large change in resistance with a change in its body temperature"[2]. The thermistor that was chosen for this device is the

[2] http://www.thermometrics.com/assets/images/ntcnotes.pdf

QT06001-055 from Quality Thermistor, Inc. This thermistor was chosen because of its operating temperature between -65°C to 150°C, the temperature surrounding the device will be about 6 °C. It is also in a 316 stainless enclosure with a NPT o-ring which was a requirement made by MBARI. The thermistor is implemented in the design by using its resistance with another resistor in order to make a voltage divider. The value of the other resistor was made 25k because the thermistor resistance is 25k at 5°C. The input voltage is 3.3V and by using the voltage divider equation one can solve for an output voltage of 1.65V allowing the ADC12 of the MSP430 able to register the correct value. When the thermistor is at room temperature the internal resistance is 10k and when the thermistor is at 5°C the internal resistance is at 25k. So the resistance of the thermistor changes due to the temperature, therefore one can calculate if the temperature is increasing or decreasing. For example if the MSP430 gets a reading of 1.8V, one can use the voltage divider equation to find the value of the thermistor resistance and then look at a table and figure out what the temperature was at that value. The figure below shows how the thermistor was implemented in our schematic design.

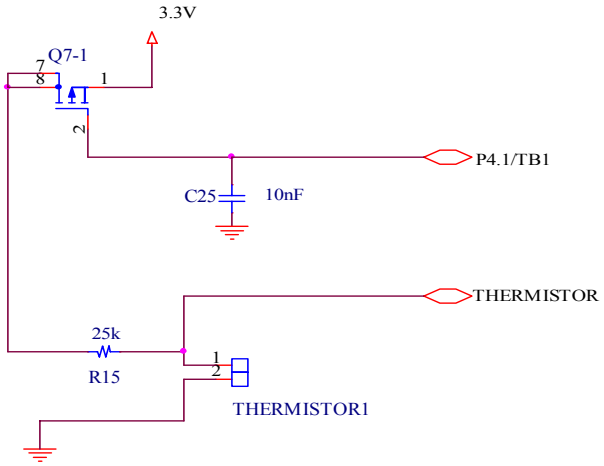


Figure 15: Schematic symbol and various wiring for the thermistor connector.

LED and Photodiode

The LED and Photodiode are used to measure turbidity. Turbidity is the haziness of water. The LED and photodiode will be placed parallel from each other in order for the photodiode to be able to measure the intensity of the LED. There is an 85 ohm resistor implemented with the LED and photodiode. This resistance was calculated using the circuit below. The forward current I_F = 20mA and the voltage drop through the diode is

3.3V. Therefore using KVL the equation 5V- R(20mA) -3.3V = 0 was created. Solving for R we get the value of 85 ohms. When I_F = 20mA the luminous intensity of the LED is 13000 MCD. Therefore when the device reads a intensity of 13000 MCD then data will show that there is no turbidity, However if the intensity decreases then the device will record the data and later determine that there was some turbidity. The figure below shows how the LED and Photodiode were implemented in the design.

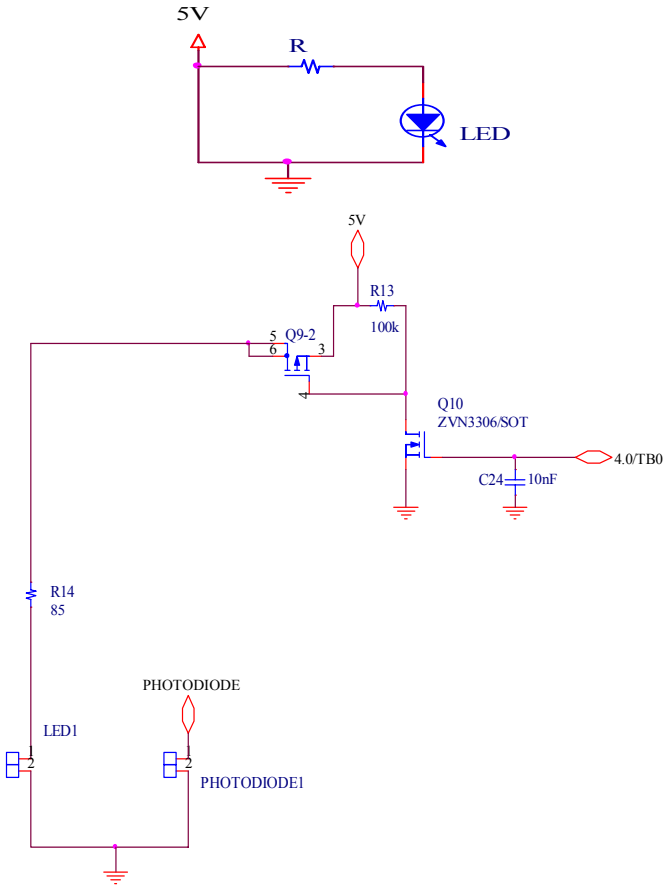


Figure 16: Schematic symbol and wiring for both the LED and photodiode.

Batteries

A portable and compact power source was needed. The logical solution for this decision is the use of batteries. However there are many types and sizes of batteries available so some research needed to be completed before a choice could be made.

The choice for the type of battery was narrowed down to a primary lithium battery. Primary lithium batteries feature a constant voltage while being used i.e. the

voltage of the battery will not change if the battery is discharged. The voltage will only start to decay when the battery is nearly dissipated. Primary batteries also do not require charging (they come charged when purchased) and operate well under a wide temperature range. This will be useful since the device will be subject to the temperature of deep ocean water. The only drawbacks to using primary lithium batteries are 1) they are not rechargeable and 2) they will be damaged if all the entire capacity of the battery is used. These drawbacks could be easily overcome with the use of secondary lithium batteries. Secondary lithium batteries are rechargeable and will not become damaged if the entire capacity of the battery is used. However, a chip capable of recharging lithium batteries is required due to the complicated charge procedure necessary.

The choice for size, including voltage and capacity, depends on the input voltage range of the voltage regulators and power calculations for the device. The largest minimum voltage required for the voltage regulators is 4.3V (MAX 604). The average open circuit voltage at room temperature of a primary lithium battery is 3.6V. This voltage will drop slightly with temperature. Therefore two primary lithium batteries must be connected in series resulting in an open circuit voltage of 7.2V.

As for capacity size, various power calculations were needed in order to determine how many ampere-hours (Ah) were required to have the device deployed for six months with enough power left over to send data via satellite. After consulting the completed calculations it was determined that roughly 16 Ah would be required for the device to have enough power for one deployment.

Ultimately, the battery of choice is the CSC93 from Electrochem. These primary lithium batteries have an open circuit voltage of 3.9V at room temperature with a rated capacity of 30 Ah. The batteries have a cell diameter of 1.32" and a length of 4.39" making them a perfect size for the pressurized enclosure.

Power Management

In designing the device the main concern was the conservation of power. The device has to be underwater for at least 6 months with enough power to send the data. Therefore all components used for the design were low power.

MOSFETs

To conserve power MOSFETs (metal–oxide–semiconductor field-effect transistor**)** are used to control power to components that do not require 100 percent of the time, which are the pressure sensor, LED, thermistor, SD card, Iridium modem, and RS-232. The pressure sensor, LED, and thermistor are only used when taking measurements. The SD card is only used when it is reading or writing. The modem and RS-232 are only used when the device is transmitting data via satellite. The MOSFETs are also used to control the switching regulators because they supply power to components that are being turned off and on. It makes no sense leaving the switching regulators on if the power they supply is not being used. The MOSFETs that were chosen are the TPS1120 P-channel MOSFET and the ZVN 3306 N-channel MOSFET.

TPS1120

The TPS1120 was chosen because it is "the ideal high-side switch for low-voltage portable battery-management system, where maximizing battery life is a primary concern".[3] This component works perfectly with the voltages that are being used for the device. An example of how the TPS1120 is implemented can be seen in figure 17. The TPS1120 max gate-to-source threshold voltage (V_{GS}) is -1.5V. This mean when V_{GS} is -1.5V or greater the MOSFET is completely on and will deliver power to the temperature sensor. Therefore when the MSP430 gives logic 0 to the TPS1120 it will give power. When it gives logic 1 no power is delivered through the TPS1120. The TPS1120 is used similarly for other components.

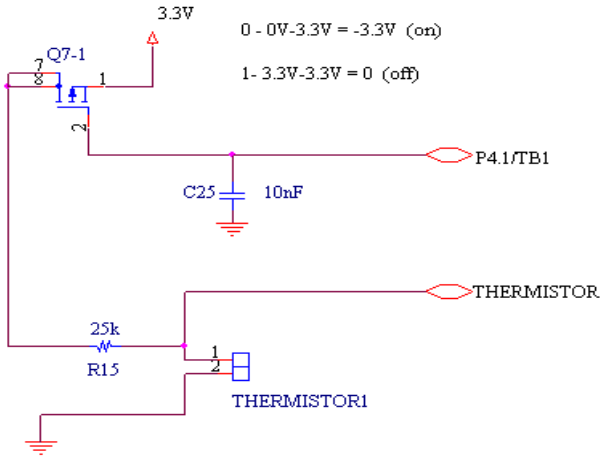


Figure 17: Schematic symbol showing the TPS1120 controlling power to the thermistor.

ZVN3306

Since 5V is required for the device the TPS1120 will not work with 5V. If the source is 5V and the gate is 3.3V, using the equation $V_{GS}=V_G-V_S$ one can calculate what V_{GS} is when the MSP430 gives a logic 0 or a logic 1. When the MSP430 gives logic 1 then V_{GS} will be -5V and when it gives a logic 0 then V_{GS} will be -1.7V. When the gate-to-source threshold voltage is -1.5 the TPS1120 is completely on, therefore with a 5V source the TPS1120 is always on which causes a problem. To solve this problem the ZVN3306 N-channel MOSFET was chosen. The ZVN3306 solves the problem because since the source is tied to ground, the gate-to-source voltage is the voltage given by the MSP430. This means that when the MSP430 gives logic 1 the gate will receive 3.3V and when it gives logic 0 the gate will receive 0V. The VGS(th) max of the ZVN3306 MOSFET is 2.5V. Therefore when the gate of the ZVN3306 gets 0V, it is off. This makes the gate-to-source voltage of the TPS1120 0V allowing the TPS1120 to be turned off. Now when the gate of the ZVN3306 receives 3.3V this turns it completely on allowing the gate of the TPS1120 to be tied to ground. This causes the TPS1120 to be turned on and to give 5V to a certain component. The figure below shows how the ZVN3306 and the TPS1120 are used together to supply the LED with 5V.

[3] TPS1120 Data Sheet, http://focus.ti.com/lit/ds/symlink/tps1120.pdf

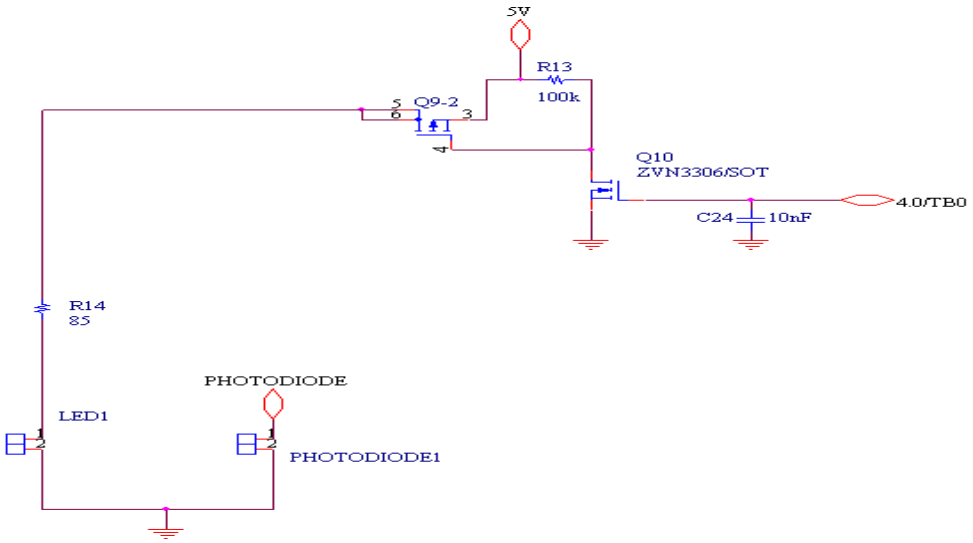


Figure 18: Schematic symbol showing both the TPS1120 and the ZVN3306.

Power Supply

The power supply for this device converts 7.2V from two lithium batteries in series to 5V using a switching regulator and to 3.3V using a linear regulator and a switching regulator. The 5V switching regulator supplies voltage to the Iridium modem, pressure sensor, and the LED. The 3.3V linear regulator supplies voltage to MSP430 microcontroller, real time clock, JTAG connector (Joint Test Action Group) and the temperature sensor. The 3.3V switching regulator supplies voltage to the SD card, RS-232, and the external voltage reference for the MSP430.

Why use both linear and switching regulators?

In the first stages of designing the device, a 3.3V linear regulator was going to be used to supply all the components that needed to be supplied by 3.3V. Linear regulators are beneficial because they have low quiescent current which is very favorable when a component is on all the time. However if you look at the table below you will notice

Components	Max Current Draw (mA)	Power consumed by components using 3.3V (mW)	Power consumed by linear Regulator using 7.2V (mW)	Power Lost (mW)
Microcontroller	0.66	2.178	4.752	2.574
Real Time Clock	0.2	0.66	1.44	0.78
Voltage Reference	0.0066	0.02178	0.04752	0.02574
Temperature Sensor	0.081	0.2673	0.5832	0.3159
RS-232	1	3.3	7.2	3.9
SD Card (Read)	40	132	288	156
SD Card (Write)	45	148.5	324	175.5
			Total Power Lost	339.09564

Table 6: Calculations showing the amount of power wasted by running several components off a linear regulator.

that the linear regulator consumes more power than the power consumed by the components it supplies power too. So by using a linear regulator to supply all the components the device will be losing 339mW of power. The device needs to conserve all the power that it can because of the duration it will remain underwater. Therefore this caused a problem for the initial design of the device.

To solve this problem the device had to include a switching regulator because of its high efficiency. The switching regulator loses very little power however it has a high quiescent current. Therefore the final design of the device has two switching regulators to supply different voltages to components that are being turned on and off. Since the switching regulators have a high quiescent current it would not be a good idea to use switching regulators for components that are on all the time. Therefore the device uses a linear regulator for the components that are on all the time and switching regulators for the components that are turning on and off. Also since the components the switching regulator supply voltage to are turning on and off the device also turns the switching regulator on only when it needs to supply voltage to those components.

Using two switching regulators and one linear regulator helps the device conserve power. The switching regulator that the device uses is the MAX1626 from Maxim. It was chosen because it is able to supply 5V if a specific pin was driven high and 3.3V if the pin is driven low, therefore the device is able to use the same chip for the two different voltages. The linear regulator that the device uses is the MAX603 from Maxim. It was chosen because of its low quiescent current.

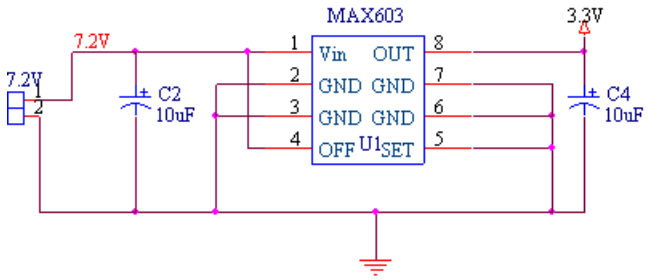


Figure 19: Schematic symbol and wiring of the MAX603

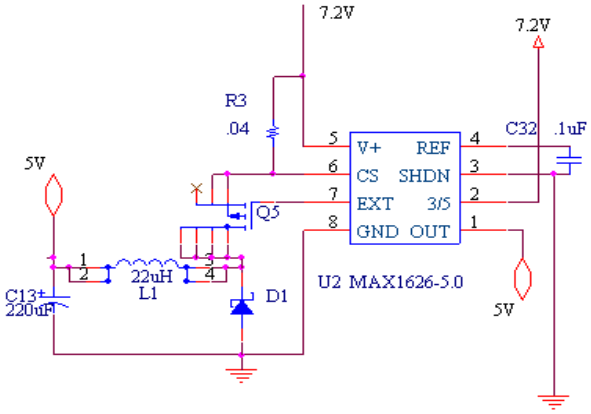
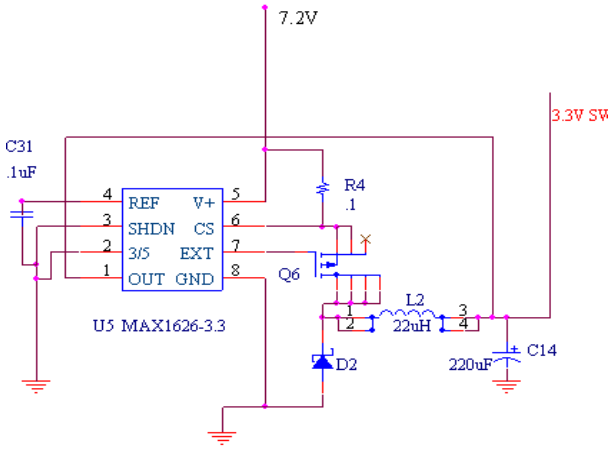


Figure 20: Schematic symbol and wiring of the MAX1626 which supplies 5V



Max1626 3.3V

Figure 21: Schematic symbol and wiring of the MAX1626 which supplies 3.3V

Layout Design

The board is 2-layers with the bottom layer as a ground plane. The board went through two different designs. The first design did not work out because the part placement made routing very difficult. Therefore a new design was created with logical reasoning behind where the parts were placed. Referring to the appendix one will notice that the connectors for the battery, and the sensors are all located near the edge of the board. This was done because it makes it easier to connect those components to the board and it makes sure that there are not a lot of other components in the way to interrupt the connection. The RS-232 chip and the 9601 modem connector are close to each other because the RS-232 chip is needed in order for the modem to communicate with the MSP430. The final layout design of the board places parts that should be near each other as close as possible. The Linear regulator should be close to the battery connector so it can get the input voltage that it needs however it is located near the top of the board because the components it supplies power too are all located in the top half of the board. The SD card holder is also located in the top of the board because it was the biggest

component and the most space available was in that area. The MSP430 is located close to the middle of the board because it is the main component. The board has most of its components on the top layer; however the real time clock, 4 resistors and a capacitor are located on the bottom layer.

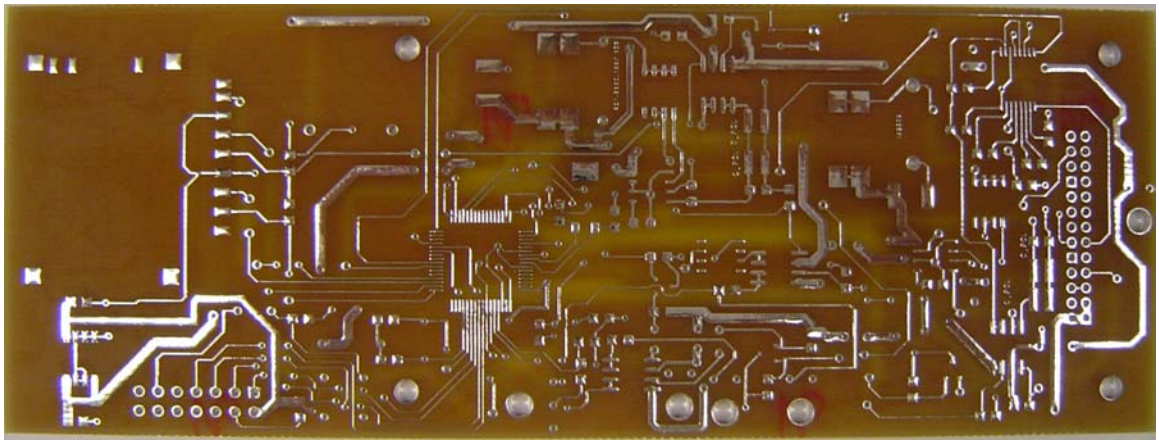


Figure 22: PCB without components

First Board Run Errors

The first board made for this device had multiple errors. The errors of the first board are listed below:

- Drill holes for reset button were too small. Initial size was 28 mils, correct size is 35 mils.
- Drill holes for sensor connectors and battery are too small. Initial size is 28mils correct size drill is 38 mils.
- Reset pushbutton is over one of the mount holes; therefore pin 1 needs be rotated to be located in the top left and pin 2 in the bottom left of the four pins. This will make the reset pushbutton face towards the edge of the board.
- Drill holes for the IN5818 diode are too small. Initial size is 35mils, correct size is 38mils. Also the diode is too close to the Inductor therefore must be moved down away from the inductor.
- Pin 13 and pin 8 of the MAX3221 are connected to the 9601 modem connector incorrectly. Pin 13 of the MAX3221 needs to be connected to pin 11 of the modem connector. Pin 8 of the MAX3221 needs to go pin 12 of the modem.
- Drill holes for R3 the .04 ohm current sense resistor is too small. Initial size was 28mils, correct size 35 mils.
- Drill holes for 1MHz crystal too small. Initial drill size 20, correct size is 28mils.
- JTAG pin 2 and pin 4 are routed incorrectly. Pin 2 needs to be tied to Vcc and pin 4 needs to be tied to ground. No need for 0 ohm resistor.
- Photodiode and LED connector are too close to each other, unable to mount to the board. Need to move parts around in order to make space for those connectors.
- C17 capacitor is too close to thermistor connector. Needs to be moved up away from the thermistor connector.
- The mounting holes do not match with the modem mounting holes.

- Footprint for battery holder for the real time clock is wrong. Circular ring should be around pin 2 not pin 1. However this does not affect the routing for the component.

Modifications

The first board had some components that did not cause any errors, however with some slight modification it would make these components easier to use for the second board run. Those modifications are listed below:

- Make the connector for the battery and pressure sensor face towards the edge of the board.
- On the first board the battery for the real time clock is located on the bottom layer. It will be better if it is located on the top layer.
- The 1MHz crystal should be placed on the bottom layer instead of the top layer.
- The IN5820 diode should be moved down slightly.

Testing

As of this writing the power supply for the device is working properly and the also the JTAG used to program the MSP430 is also functional. With the time allotted until the final presentation the board should be in a fully functional state.

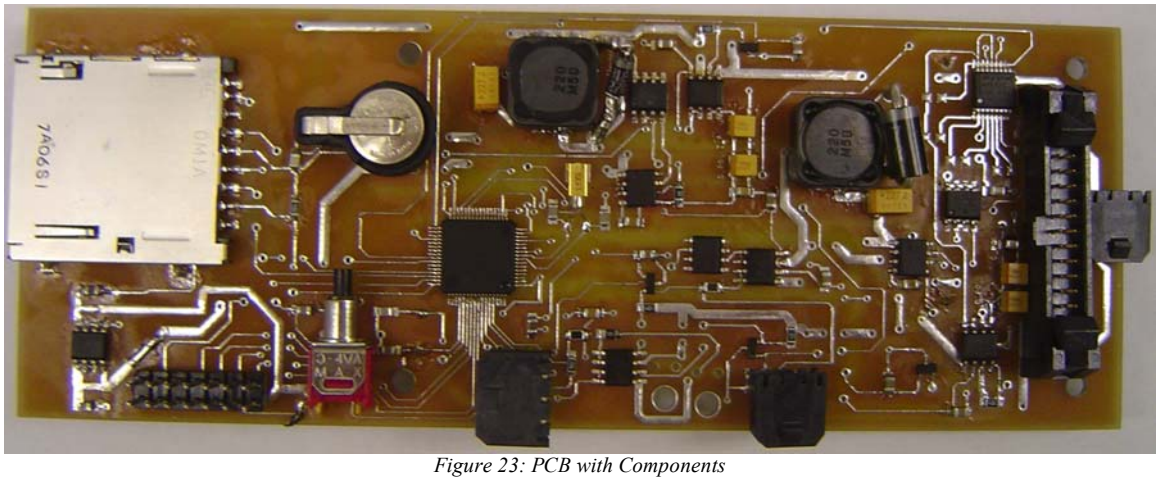


Figure 23: PCB with Components

Prototype

A hardware prototype was created using the MSP430 development kit, a bread board, DIP versions of the real-time clock and RS232 chips, the two sensors and the modem. This was required in order to develop software for the device.

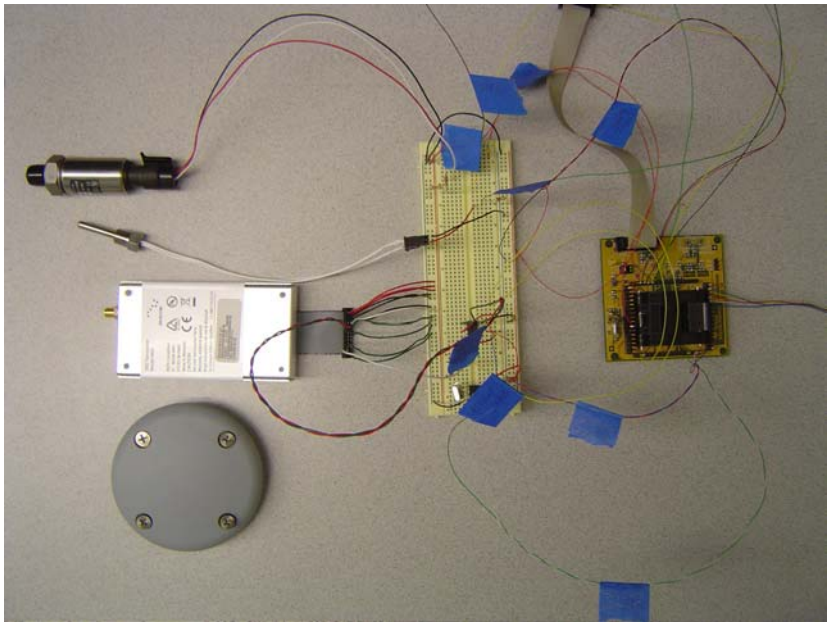


Figure 24: Prototype used in lab to develop software

Software Technical Breakdown and Design

Interfaces

Digital I/O

The MSP430F169 has six digital I/O ports included on the chip (P1-P6), which consist of eight I/O pins. Every I/O pin can be individually configured for input or output directions meaning each I/O pin can be individually read/written to. This is done with an eight-bit register, each bit in the register corresponding to an I/O pin (P1.0 – P1.7). A two-digit hexadecimal can be used to assign each bit a 0 or a 1. The digit on the right controls P1.0 – P1.3 while the digit on the left controls P1.4 – P1.7. For example:

P1IN = 0x31;

will set P1.0, P1.4 and P1.5 to a 1.

Configuration

The configuration of a port (e.g. P1) can be done in two steps with software:

1. Configure the P1SEL register. Each bit can be set as I/O function (0) or peripheral module (1). Note: More information regarding the peripheral module setting is discussed in the corresponding module section.
2. Configure the P1DIR register. Each bit can be set as input (0) or as output (1).

Use

Once these two registers have been correctly configured for either input or output, the user can use two registers to read/write those pins. The inputs can be read from with the P1IN register. If the corresponding bit is 0 then the input is low while if the bit is 1 then the input is high. The outputs can be written to with the POUT register. A 0 will set the output low while a 1 will set the output high.

USART Peripheral Interface, I2C

The MSP430F169 has the ability to use the universal synchronous/asynchronous receive/transmit peripheral interface (USART) for I2C communication in USART 0. This is done with the use of two pins (SCL and SDA) and thirteen different eight-bit registers controlling various pieces of the I2C module. Ten of the twelve registers can be read/written to while the I2C data control register and the I2C interrupt vector can only be read from.

Configuration

The configuration of the I2C module must be done with a specific procedure in order to correctly switch the USART from UART or SPI mode to I2C mode. Failure to follow this procedure could result in unpredictable operation:

1. Select I2C mode by setting the I2C bit = 1 and the SYNC bit = 1 with SWRST bit = 1 in register U0CTL.
2. Disable I2C module by setting the I2CEN bit = 0 in register U0CTL.
3. Configure the I2C module settings for use with the device(s) that will be connected with I2CEN bit = 0.
4. Enable the I2C module by setting the I2CEN bit = 1.
5. Configure the I2C pins, SCL and SDA for peripheral use by setting P3SEL = 0x0A.

USART Peripheral Interface, UART Mode

The MSP430F 169 has the ability to use the USART peripheral interface for the operation of an asynchronous UART mode in USART 0 or USART 1. This is done with the use of two pins (UTXDx and URXDx) and eleven different eight-bit registers controlling various parts of the UART module. Ten of the eleven registers can be read/written to while the receive buffer register (URXBUF) can only be read from.

Configuration

The configuration of the UART module must be done with a specific procedure if the user is re-configuring the USART module for UART operation from I2C operation:

1. Set the I2C, SYNC and ICEN bits = 0 in register UxCTL.
2. Set the SWRST bit = 1 in register UxCTL.
3. Configure the UART settings for use with device that will be connected.
4. Configure the UART pins, UTXDx and URXDx for peripheral use.

Use

Writing to the transmit buffer register (UTXBUF) will send the connected device data while the receive buffer register contains what has been sent from the connected device. Reading this register allows the user to see what data has been sent. This is most easily done with an interrupt that is called every time a new byte of data is received.

USART Peripheral Interface, SPI Mode

The MSP430F169 also has the ability to use the USART peripheral interface for the operation of a synchronous SPI mode in USART 0 or USART 1. This is done with the use of three or four pins (SIMO, SOMI, UCLK and STE) and eleven different eight-bit registers controlling various parts of the SPI module. Like the UART module ten of the eleven register can be read/written to while the receive buffer register (URXBUF) can only be ready from.

Configuration

The configuration of the SPI module must be done with a specific procedure if the user is initializing or re-configuring the USART module for SPI operation:

1. Set the SWRST bit = 1 in register UxCTL.
2. Configure the SPI settings for use with device that will be connected.
3. Enable the USART module by setting the MEx bit in register USPIEx.
4. Set the SWRST bit = 0 in register UxCTL.
5. Configure the SPI pins, SIMOx, SOMIx, UCLKx and STEx for peripheral use.

ADC12

The MSP430F169 includes a 12-bit analog-to-digital converter with eight individually configurable external input channels and software selectable internal or external reference voltage. It also includes fifteen individually configurable ADC12 memory registers that can be used to store conversions from the various input channels.

Configuration

The configuration of the ADC12 module can be done in five simple steps:

1. Configure the ADC12 channel pin to be used for peripheral use.
2. Turn on the ADC12 and set the sampling time.
3. Choose what sampling timer to use.
4. Configure the ADC12 channel to be used and the reference to be used with the corresponding memory register.

Use

After starting a conversion, the user can read from the configured memory channel (ADC12MEMx) to obtain the conversion result. This result will be a 4-digit

hexadecimal number which can be converted to the corresponding analog voltage by following these steps:

1. Convert the 4-digit hexadecimal number to the corresponding decimal number.
2. Multiply the decimal number by the voltage reference level.
3. Divide the result from the previous step by the resolution of the ADC (4095).

Similarly, the ADC12 conversion expected given an analog input voltage can be found by using the following formula:

$$N_{ADC} = 4095 \times \frac{Vin - V_{R-}}{V_{R+} - V_{R-}}$$

where Vin is the input analog voltage, V+ is the positive voltage reference, V- is the negative voltage reference and Nadc is the conversion result expected.

Using the Real-time Clock

Setting the Time

The DS3231 has a simple setup regarding how time is kept. Each time-unit is stored in an eight-bit register with the hexadecimal representation of the register being the actual time unit value (i.e. 0x30 in the seconds register corresponds to 30 seconds). In order to set one of these registers, the user must simply write to the register. To select a particular register a register pointer is used. The first byte written to the real-time clock sets the register pointer to point to that particular register. This register pointer will move to the next time-unit register once the current register is written to. This allows the user to set the register pointer once and then set the time units one by one in a specific order without having to point the register pointer to a certain register.

Reading the Time

Reading the time from the DS3231 is just as simple as setting the time. First the register pointer must be pointed to the correct register. Then once this register is read from the pointer will increment to the next register. If the pointer register is pointing to the seconds register the user can then read seconds, minutes, hours, day, date, month and year in that order without having to reset the register pointer.

Using the Iridium Modem

Configuring the Modem for Use with MSP430

In order for the 9601-D-I to correctly communicate with the MSP430 through a 3-wire serial interface, steps must be taken to change the default configuration:

1. Ignore the DTR input by sending the 9601-D-I the command AT&D0. The modem will then return OK.
2. Disable RTC/CTS flow control by sending the command AT&K0. The modem will then return OK.
3. Change the data rate of the 9601-D-I from 19200 bps to 9600 bps by sending the command AT+IPR=5. The modem will then return OK. (Note: The change in data rate takes into effect after the OK has received by the modem. The settings of the device/program being used to communicate with the 9601-D-I will need to be changed in order to continue communication.)
4. Disable the echo of characters from the 9601-D-I by sending the command "ATE0". The modem will then return OK.
5. Store the current configuration as profile 0 by sending the command AT&W0. The modem will then return OK.
6. Store profile 0 as the power-up default by sending the command AT&Y0. The modem will then return OK.

The modem is now correctly configured for communication with the MSP430 microcontroller.

Sending a Message

Sending a message with the 9601-D can be done with two AT commands. The first command allows the transfer of an X-byte message to the message buffer located inside the modem. The second command will send the message currently in the buffer if the Iridium network is visible. If the network is unavailable the modem will still attempt to send the message but will be unsuccessful.

The first command is the Short Burst Data: Write Binary Data to the Module command. The AT command is AT+SBDWB=<SBD message length> where the <SBD message length> parameter represents the length in bytes of the SBD message to be sent. The minimum SBD length is 1 byte and the maximum is 205 bytes. Once the 9601-D receives this command it will indicate that it is prepared to receive the message it will return READY. The message now must be sent to the 9601-D within 60 seconds followed by an extra 2-bytes. These extra 2-bytes are called the 2-byte checksum and are the least significant 2-bytes of the summation of the entire SBD message with the high order byte being sent first. If the 9601-D accepts the message then a 0 will be returned. If a 1 is returned then the message and 2-byte checksum were not completely written to the 9601-D in the allotted 60 seconds. If a 2 is returned then the checksum sent to the 9601-D does not match the checksum calculated by the modem.

After a message has been successfully transferred to the buffer (9601-D returns a 0) the message can be sent through the Iridium network. The command to accomplish this is the Short Burst Data: Initiate an SBD Session. The AT command is AT+SBDI and will initiate an SBD session between the 9601-D and the Gateway SBD Subsystem (GSS). Once this is complete the 9601-D will return the following command response:

+SBDI: <MO status>, <MOMSN>, <MT status>, <MTMSN>, <MT length>, <MT queued>

- <MO status> provides the user with the status of mobile originated transaction. If a 0 is returned no SBD message is available to send. If a 1 is returned the SBD message was successfully sent. If a 2 is returned an error occurred while sending the message.
- <MOMSN> is incremented each time a message is successfully sent with the 9601-D. This value provides the user with how many messages have been successfully sent with the 9601-D and will count up to 65535 before wrapping back to 0.
- <MT status> provides the user with the status of a mobile terminated transaction. If a 0 is returned no SBD message was received from the GSS. If a 1 is returned an SBD message was successfully received from the GSS. If a 2 is returned an error occurred while attempting to perform a mailbox check or receive a message from the GSS.
- <MTMSN> is incremented each time a message is successfully forwarded to the 9601-D. This value provides the user with how many messages have been successfully received by the 9601-D and will count up to 65535 before wrapping back to 0.
- <MT length> is the length in bytes of the message received from the GSS and will read zero if no message was received.
- <MT queued> is the number of messages waiting at the GSS to be transferred to the 9601-D.

(Note: Other AT commands can also be used to send a message with the 9601-D.)

For example if the message "hello" encoded in ASCII were successfully sent with no message received by the 9601-D, the commands and command responses would be as follows:

AT Command/Data to 9601-D	Command Response from 9601-D
AT+SBDWB=5	READY
68 65 6C 6C 6F 02 14	0
AT+SBDI	+SBDI: 1, 1, 0, 0, 0

Table 7: AT commands and command responses to send a message

Decoding a Message

A message sent with the Iridium modem will have a consistent format. The data to be sent can be considered as a "packet" of data. A message will contain one packet of data. A packet of data includes a time stamp followed with the corresponding temperature ADC reading and pressure ADC reading summing to 10 bytes structured as follows:

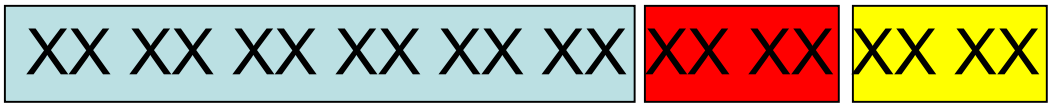


Figure 25: Format of message received from modem:
Blue: Time-stamp
Red: Temperature reading
Yellow: Pressure reading

The time stamp includes month, date, year, hour, minute and seconds summing up to 6 bytes of the message, with each byte corresponding to a 2-digit hexadecimal number. Since the RTC keeps time in this format these six bytes are the actual time stamp in the format:

Month/Day/Year Hours: Minutes: Seconds

For example if the 6 bytes read 06 08 07 11 12 30 then the corresponding time would be June 8th 2007 11:12:30.

The temperature reading is a 4-digit hexadecimal number corresponding to the ADC12 temperature reading from the MSP430. In order to convert this number to the analog voltage of the thermistor first this 4-digit decimal number must be converted to the corresponding decimal number. Multiplying the result with the ADC voltage reference (2.5) and divide this by the resolution of the ADC (2^12 = 4095). The pressure reading is also a 4-digit hexadecimal that can be converted the same way.

The message will be sent via email to steds@mbari.org with a subject title "SBD Msg from Unit: 300034012511710". The email will contain several pieces of information in addition to the message from the device. These include the amount of messages sent by the modem, the amount of messages received by the modem, the time the message was sent, the session status, the message size (in bytes) and the latitude and longitude of the modem at the time of transfer.

The message itself is included as an attachment. The .SBD file will be named as the modem unit number followed by the message number. This file must be opened with a hex editor in order to see the corresponding hex values. If this file is opened with a text editor the ASCII characters associated with those hex values will be displayed, which will result in an undecipherable text message.

Software Development Priorities

Before software was started each component that needed code written for implementation into the design was given a priority. These priorities were assigned based on how important the component was to the final design. The priority list is as follows:

1. Iridium Modem
2. Pressure sensor
3. Real-time clock
4. SD Card
5. Temperature sensor

The Iridium modem was given priority over everything else since the device would be useless without being able to send the data via satellite. The pressure sensor was given second priority since pressure readings are necessary to detect if the device is above or below water. The real-time clock is third on the priority since the only information MBARI is interested in is the time of a turbidity current. SD card was placed

fourth on the priority list since the MSP430 has internal memory that can be used to store data measurements. Finally the temperature sensor was placed last on the list since the operation of the device does not require temperature readings.

The code written for each component was actually completed in a different order than the priority list, which is listed below:

1. Real-time clock
2. Temperature sensor
3. Pressure Sensor
4. Iridium Modem

The Iridium modem was worked on first but frustration was the driving factor in putting it aside and starting work with the real-time clock. Once work with the real-time clock was finished work on the modem began again but frustration set in. Work with the ADC12 began and was completed with the use of the temperature sensor since the connector for the pressure sensor was being shipped. Once the connector for the pressure sensor arrived work was completed to implement it into the design. Finally work was again continued on the modem until it a packet of data could be sent through the Iridium network. The SD card has been worked with but as of this writing it is not functional. Work with the SD card will continue in the time allotted until the final presentation and will hopefully be working with the rest of the design by the final presentation.

Software States

Software for the device is divided into three different states, each corresponding to the device completing different tasks. In STATE 0 the device will be checking for a

pressure reading equivalent to five meters underwater. In STATE 1 the device will be taking temperature, pressure and turbidity readings with a time stamp every five seconds. Finally in STATE 2 the device will transfer data taken via satellite. The flowcharts shown below show each state in more detail.

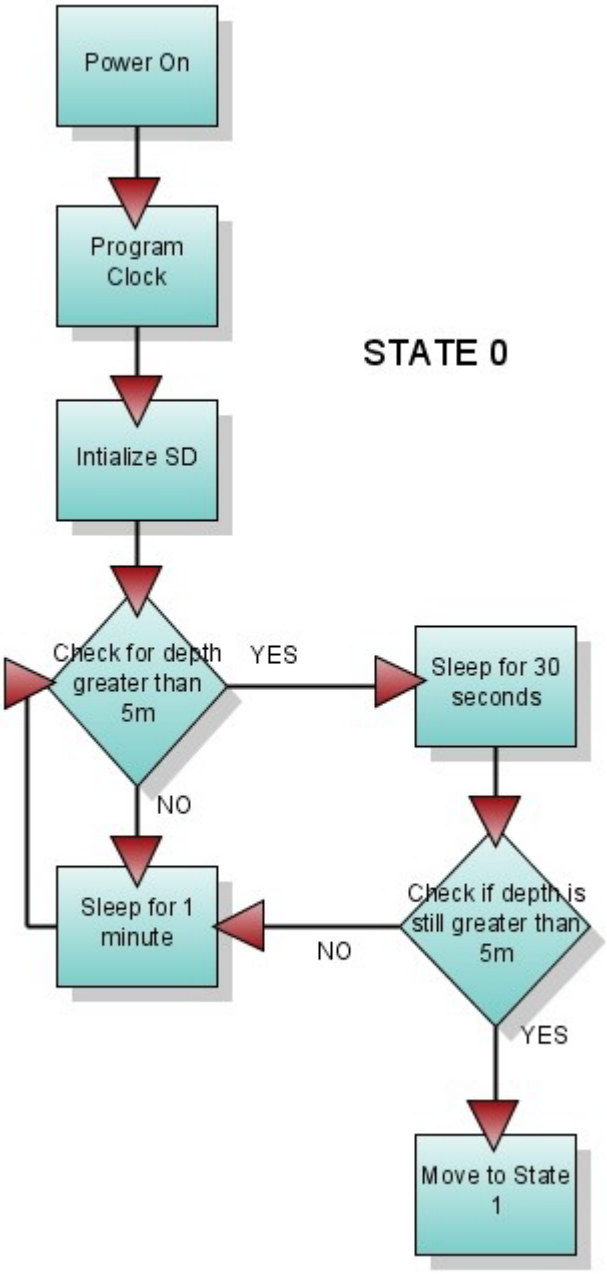


Figure 26: flow chart for state 0

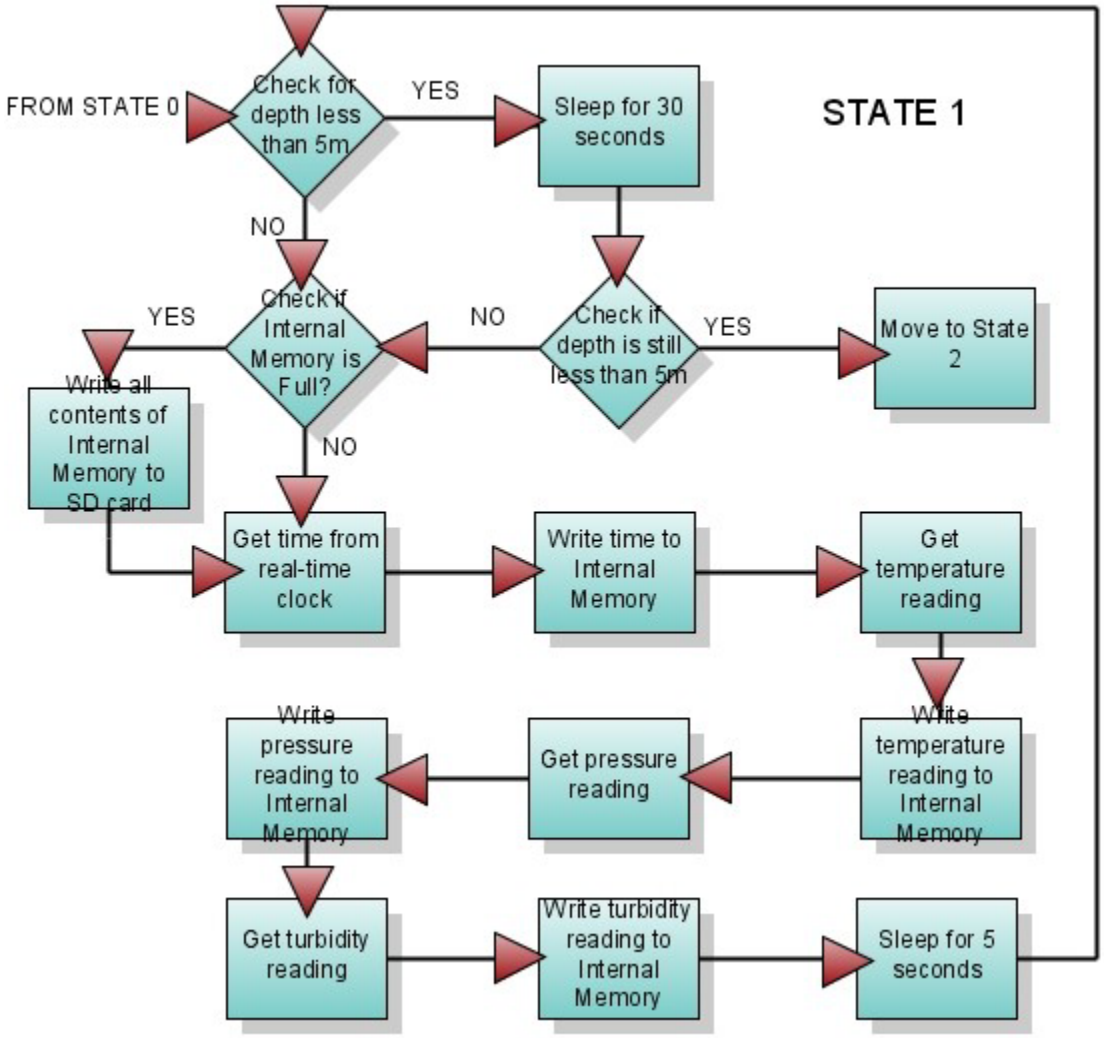


Figure 27: Flow chart for state 1

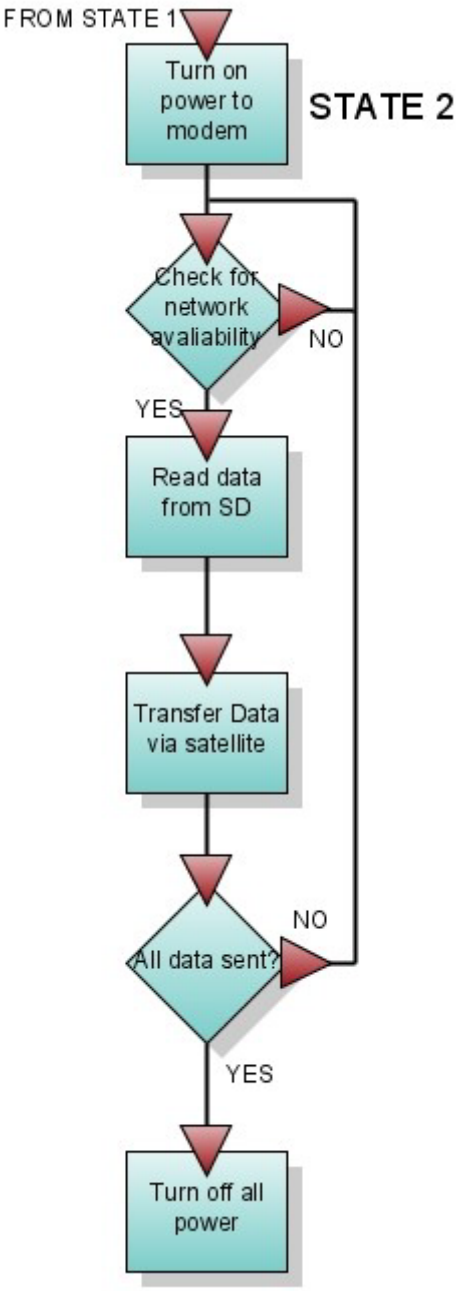


Figure 28: Flow chart for state 2

Mechanical Design

The device will be enclosed in a 340m PVC (Polyvinyl chloride) pressure housing. The two batteries will be located at the bottom of the casing, along with the pressure sensor, temperature sensor and the turbidity sensor. The temperature sensor and pressure sensor will have access to the outside water without compromising the internal pressure. The modem and the PCB (Printed Circuit Board) will be placed above the batteries. The PCB is mounted onto the modem using screws from the modem. The modem requires an antenna to communicate; therefore the antenna is placed on top of the enclosure to ensure satellite visibility.

How will the device be placed underwater?

The enclosure that the device uses will be surrounded by floatable foam. Then the floatable foam and the device will be attached to a STED (Self Triggering Event Detector). The two are attached by a lever. Therefore as long as an event does not occur the device will remain attached to the STED. However once the STED detects an event by the increasing current, it lifts the lever and releases the device. Then the device will float to the surface and transmit the data when it detects the Iridium satellite.

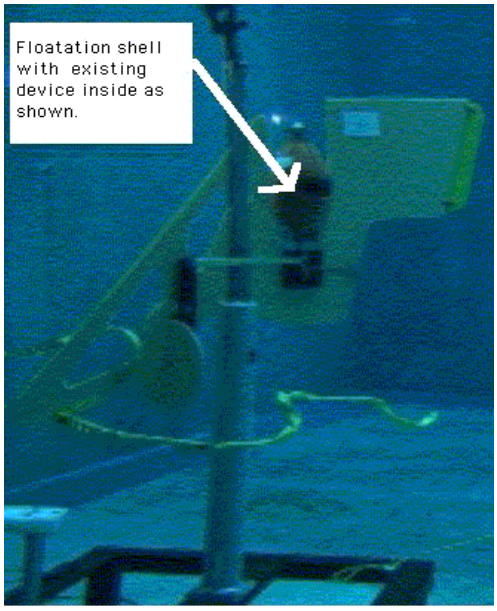


Figure 29: Current detector with flotation device attached

Potential Features

Battery voltage Reader

Lithium battery voltage remains at constant level until the voltage begins to decrease. Once the voltage begins to decrease the voltage drops rapidly. This causes a problem because if an event does not occur by the time the battery starts to discharge it may not have enough power to transmit the data via satellite. Therefore being able to read the battery voltage will be a great feature because if the voltage begins to drop then the device can shut down certain components in order to conserve power.

Early Release

Another great feature that would be great to add to the device is the ability to release itself from the STED without an event happening. The only use for this feature will be if the battery begins to discharge and the device only has enough power to transmit the data. If the device does not have enough power to transmit data then the device is a waste, therefore this feature will be a great add on.

More Sensors

The current device only uses three ADC12 pins of the MSP430. Therefore the device is capable of having up to five more different sensors. This is beneficial because it allows the device to be used for other kinds of underwater research.

Personal Experience/Reflection

Edgard Rincand

EE123A/B has been a great experience. I feel that this class has really prepared me in what to expect after college. The beginning of this course was very difficult because I did not know what to expect. This course was not like any other course where you have a problem and you know exactly what textbook to look in for the answer. That is why this course was very difficult for me because I was given a problem, but I had no idea where to start in order to solve that problem. Now at the end of this course and about to graduate I feel that I have learned the steps that I needed in order to have a project given to me and being able to complete that project.

Designing the project was very difficult and the most challenging. It was the most challenging part because every time I felt that the design was complete a new problem would surface and the design would have to be changed. This was frustrating because we had a deadline to meet and every time we determined a new problem we would be farther from completion. Creating a schematic and the layout was time consuming because I had to learn how to use the programs. I am glad that I had the opportunity to be able to create a PCB. The board I created, however, had a lot of errors due to my inexperience. I feel that my next board will be more successful due the experience that I gained not only of

the program, but of the design process in creating a PCB. This course has prepared me for the next step after my undergraduate career. The knowledge that I have gained in this course has prepared me for the first step in my engineering career.

Stefan Gadomski

First of all I would like to say that the last ten weeks have been the most grueling ten weeks of my entire life. I currently have no social life, eat out almost every day and still don't have enough time to complete what needs to be done. This isn't a bad thing though. I feel like I've learned more in the past ten weeks than in my entire undergraduate career.

I feel very grateful to have had the opportunity to take part in this class. At first I was going to write a Senior Thesis instead of taking 123A/B. Now after having completed 123A/B I realize what a bad mistake this would have been. There is no way I would have learned or gained as much from completing a Senior Thesis compared to what I've learned/gained in this class. This class alone has prepared me for the real world as an engineer. I now know to expect 70-hour work weeks when a deadline is fast approaching, something I was not expecting before.

One thing I would change about the last ten weeks is the classes I had to take in addition to this class. If I could take the time I had to spend on my other classes and put all of it towards the project I feel like I could have produced a much more refined and complete product for MBARI. During week five when I was studying for midterms it felt very odd to put the project on hold; like I was torn from something I enjoyed doing.

Overall the last ten weeks have been a very positive experience. I'm actually glad I was assigned with software because it gave me the opportunity to learn something I should have been taught earlier in my undergraduate career. I still would have learned a great deal even if I was assigned hardware but now I have the ability to do both hardware and/or software on the next project I work on.

Appendix

MSP430 Pin Assignments

MSP430 Pin #	MSP430 Pin Name	CONNECTS TO:	CONNECTING PIN
1	Dvcc	3.3V	
2	P6.3/A3	NO CONNECT	
3	P6.4/A4	NO CONNECT	
4	P6.5/A5	NO CONNECT	
5	P6.6/A6/DAC0	NO CONNECT	
6	P6.7/A7/DAC1/SVSIN	NO CONNECT	
7	VREF+	MAX6029 (voltage ref), GND	4
8	XIN	MS1V-T1K (crystal)	1
9	XOUT	MS1V-T1K (crystal)	2
10	VeREF+	MAX6029	6
11	VREF-/VEREF-	GND	
12	P1.0/TACLK	NO CONNECT	
13	P1.1/TAO	ZVN3306 (N-MOSFET)	3 (GATE)
14	P1.2/TA1	NO CONNECT	
15	P1.3/TA2	ZVN3306 (N-MOSFET)	3(GATE)
16	P1.4/SMCLK	NO CONNECT	
17	P1.5/TA0	TPS1120 (P-MOSFET)	2(GATE)
18	P1.6/TA1	9601 D-MODEM CONNECTOR	24
19	P1.7/TA2	NO CONNECT	
20	P2.0/ACLK	NO CONNECT	
21	P2.1/TAINCLK	NO CONNECT	
22	P2.2/CAOUT/TA0	NO CONNECT	
23	P2.3/CA0/TA1	NO CONNECT	
24	P2.4/CA1/TA2	NO CONNECT	
25	P2.5/ROSC	NO CONNECT	
26	P2.6/ADC12CLK/DMAE0	NO CONNECT	
27	P2.7/TA0	TPS1120 (P-MOSFET)	4(GATE)
28	P3.0/STE0	NO CONNECT	
29	P3.1/SIMO0/SDA	DS3231(RTC)	15
30	P3.2/SOMI0	NO CONNECT	
31	P3.3/UCLK0/SCL	DS3231(RTC)	16
32	P3.4/UTXD0	MAX3221EAE (RS-232)	11
33	P3.5/URXD0	MAX3221EAE (RS-232)	9
34	P3.6/UTXD1	NO CONNECT	
35	P3.7/URXD1	NO CONNECT	
36	P4.0/TB0	ZVN3306 (N-MOSFET)	3(GATE)
37	P4.1/TB1	TPS1120 (P-MOSFET)	2(GATE)
38	P4.2/TB2	9601 D-MODEM CONNECTOR	7
39	P4.3/TB3	NO CONNECT	

40	P4.4/TB4	NO CONNECT	
41	P4.5/TB5	ZVN3306 (N-MOSFET)	3(GATE)
42	P4.6/TB6	TPS1120 (P-MOSFET)	4(GATE)
43	P4.7/TBCLK	NO CONNECT	
44	P5.0/STE1	NO CONNECT	
45	P5.1/SIMO1	SD CARD CONNECTOR	7
46	P5.2/SOMI1	SD CARD CONNECTOR	2
47	P5.3/UCLK1	SD CARD CONNECTOR	5
48	P5.4/MCLK	SD CARD CONNECTOR	1
49	P5.5/SMCLK	NO CONNECT	
50	P5.6/ACLK	NO CONNECT	
51	P5.7/TBOUTH/SVSOUT	NO CONNECT	
52	XT2OUT	HC-49U (CRYSTAL)	2
53	XT2IN	HC-49U (CRYSTAL)	1
54	TDO/TDI	14 PIN JTAG	1
55	TDI/TCLK	14 PIN JTAG	3
56	TMS	14 PIN JTAG	5
57	TCK	14 PIN JTAG	7
58	RST/NMI	14 PIN JTAG	2,9,11
59	P6.0/A0	PHOTODIODE CONNECTOR	1
60	P6.1/A1	THERMISTOR CONNECTOR	1
60	P6.0/A0	TPS1120 (P-MOSFET)	7,8 (DRAIN)
61	P6.2/A2	PRESSURE SENSOR CON.	2
62	AVSS	GND	
63	DVSS	GND	
64	Avcc	RESET PUSHBUTTON	2
64	Avcc	MSP430	58

Table 8: MSP430 pin assignments

Bill of Materials

PCB Parts

Item	Quantity	Reference	Part
1	3	C2,C4,C29	10uF
2	4	C3,C5,C10.C11	68uF
3	2	C6,C7	12pF
4	2	C1,C9	
5	9	C8,C12,C17,C24, C25,C26,C27,C28,C35	10nF
6	2	C13,C14	220uF
7	2	C15,C20	100pF
8	7	C18,C19,C21,C22,C23,C31,C32	
9	1	C30	100nF
10	2	C33,C34	22pF
11	1	D1	IN5820
12	1	D2	IN5818
13	4	LED1,PHOTODIODE1,THERMISTOR1,7.2V	2 PIN MICRO-FIT HEADER
14	1	J2	3 PIN MICRO-FIT HEADER
15	1	J3	14 PIN HEADER
16	2	L1,L2	22uH
17	4	Q1,Q4,Q7,Q9	TPS1120
18	4	Q2,Q3,Q8,Q10	ZVN3306
19	2	Q5,Q6	MMSF3PO2HD
20	4	R1,R2,R8,R13	100K OHM
21	1	R3	.04 OHM CURRENT SENSE
22	1	R4	.1 OHM CURRENT SENSE
23	2	R5,R9	47K
24	2	R6,R7	10K OHM
25	1	R10	1M OHM
26	2	R11,R15	25K OHM
27	1	R12	20K OHM
28	1	R14	85 OHM
29	1	RS-1	MAX3221EAE
30	1	SW1	SW PUSHBUTTON
31	1	U1	MAX603
32	2	U2,U5	MAX1626
33	1	U3	SD CARD HOLDER
34	1	U4	MSP430F169
35	1	U6	DS3231SN#
36	1	U7	MAX6029SO
37	1	U8	26-PIN SAMTEC CONNECTOR
38	1	Y1	32KHz CRYSTAL
39	1	Y2	1MHz CRYSTAL
40	1	3V1	3V BATTERY HOLDER

Table 9: Bill of Materials for PCB parts

Other Parts

Item	Quantity	Part
1	1	SD Card
2	1	IRIDIUM MODEM
3	1	IRIDIUM ANTENNA
4	2	LITHIUM BATTERIES
5	1	PRESSURE SENSOR
6	1	TEMPERATURE SENSOR
7	1	LED
8	1	3V BATTERY 10MM CELL
9	2	PCB
10	1	MSP-FET 430140 DEV KIT
11	1	MALE TO FEMALE COAXIAL CABLE
12	1	PHOTODIODE
13	1	340M PVC PRESSURE HOUSING
14	4	2-PIN MICRO-FIT CONNECTOR
15	1	3-PIN MICRO-FIT CONNECTOR
16	9	MICRO-FIT WIRE CRIMP
17	1	26-PIN CABLE
18	1	PACKARD CONNECTOR

Figure 10: Bill of Materials for all other components

Layout Views

Top view

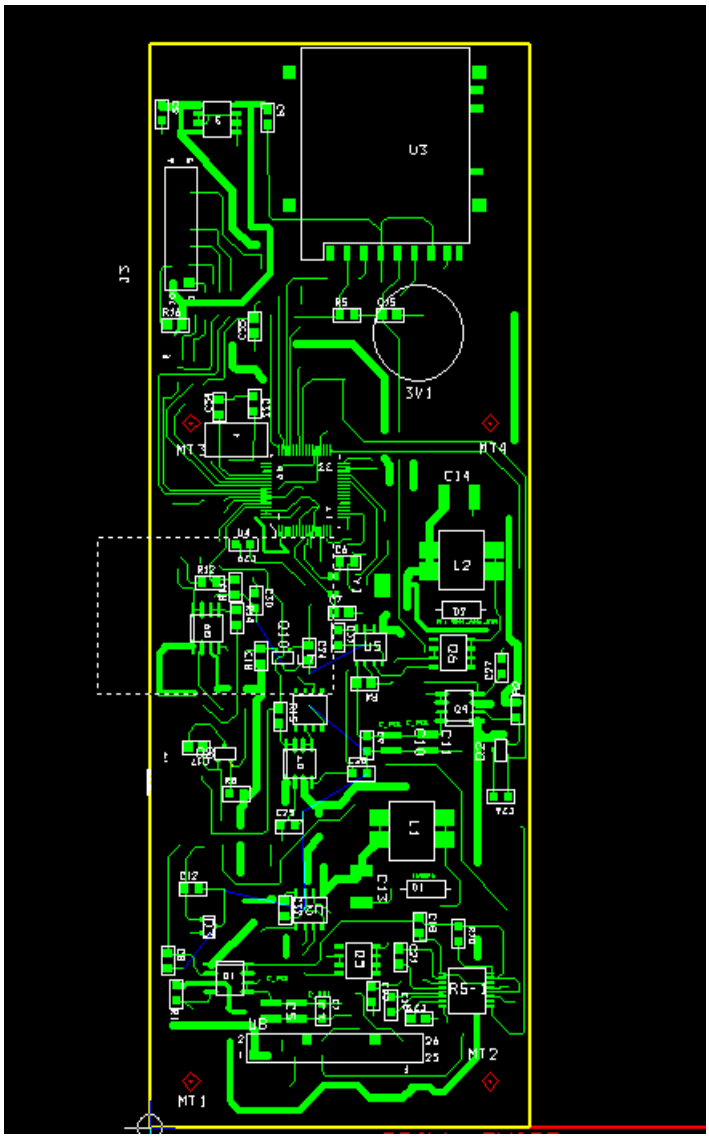


Figure 30: Top view of PCB layout

Bottom View

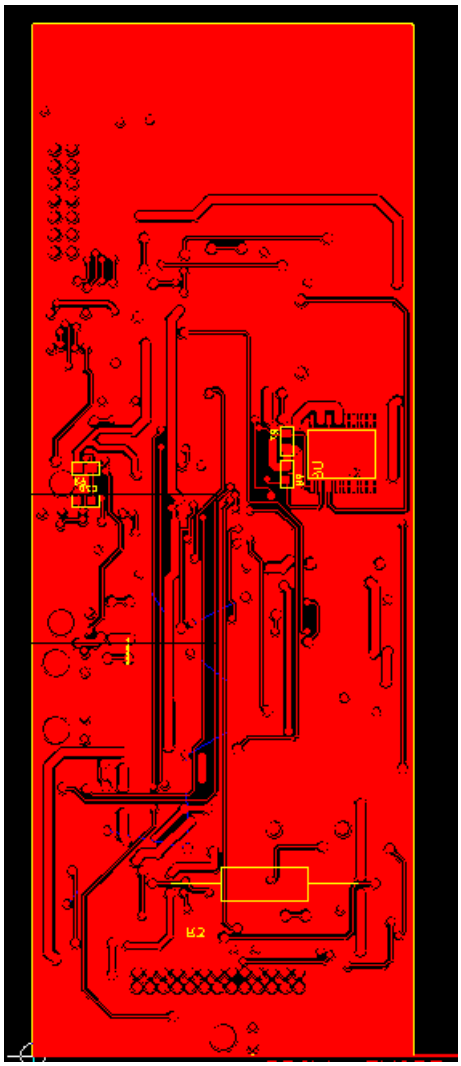


Figure 31: Bottom view of PCB layout

Top and Bottom View

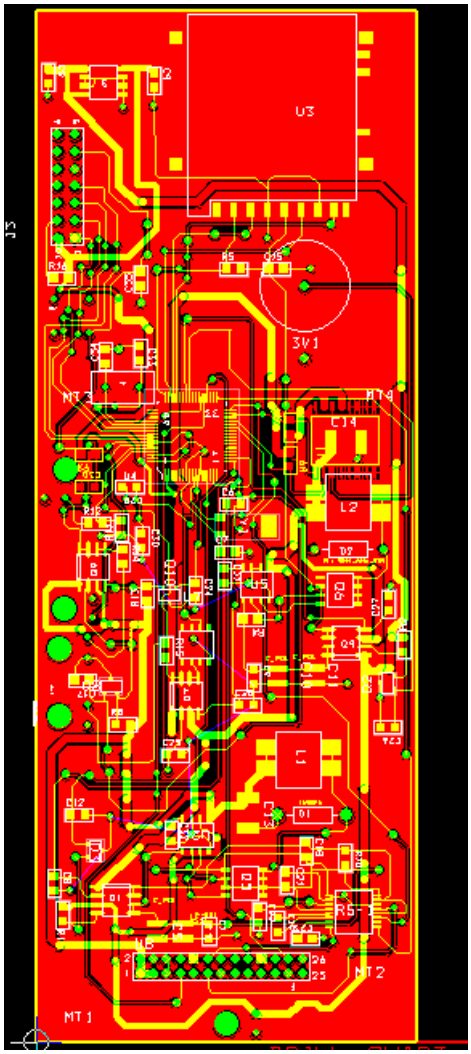


Figure 32: Top and Bottom view of PCB Layout

Schematic Design

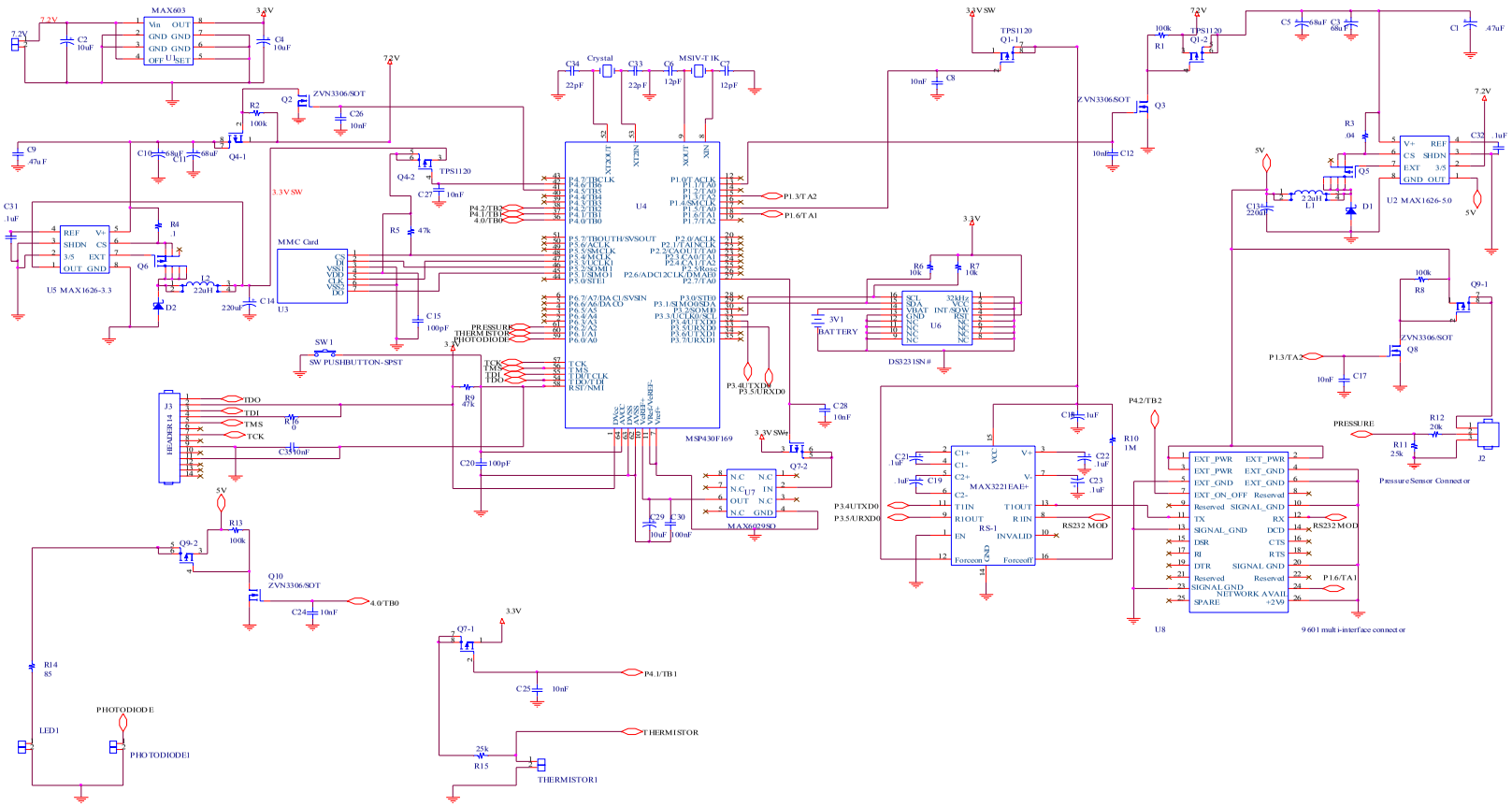


Figure 33: Schematic Design

Power Calculations

State 0

Component	Voltage (V)	Nominal Current Draw (uA)	Maximum Current Draw (uA)	Nominal Power Consumption (uW)	Maximum Power Consumption (uW)	% Time Used	Nominal Power Used (uW)	Max Power Used (uW)
3.3V Linear Regulator	7.2	15	35	108	252	1	108	252
3.3V Switching Regulator	7.2	70	90	504	648	0.031	16	20
5V Switching Regulator	7.2	70	90	504	648	0.016	7.9	10.13
Microcontroller (Active)	3.3	563	663	1857.9	2187.9	0.063	116.119	136.744
Microcontroller (Standby Mode 0)	3.3	75	90	247.5	297	0.938	232.031	278.438
Real-time Clock (Active)	3.3	200	200	660	660	0.016	10.313	10.313
Real-time Clock (Standby)	3.3	110	110	363	363	0.984	357.238	357.238
Voltage Reference	3.3	6.65	6.65	21.945	21.945	0.031	0.680	0.680
Pressure Sensor	5	10000	10000	50000	50000	0.016	781.25	781.25
SD Card (Read)	3.3	40000	40000	132000	132000	0.016	2062.5	2062.50
Minimum		206.65	241.65	740.445	933.945		697.950	888.356
Maximum		40694.65	40814.65	134350.845	134824.845		2644.537	2809.162
Total		51109.65	51284.65	186266.345	187077.845		3691.630	3909.375

Table 11: Power calculations for state 0

State 1

Component	Voltage (V)	Nominal Current Draw (uA)	Maximum Current Draw (uA)	Nominal Power Consumption (uW)	Maximum Power Consumption (uW)	% Time Used	Nominal Power Used (uW)	Maximum Power Used (uW)
3.3V Linear Regulator	7.2	15	35	108	252	1	108	252
3.3V Switching Regulator	7.2	70	90	504	648	0.4	201.6	259.2
5V Switching Regulator	7.2	70	90	504	648	0.2	100.8	129.6
Microcontroller (Active)	3.3	565	663	1864.5	2187.9	0.5	932.25	1093.95
Microcontroller (Standby Mode 0)	3.3	75	90	247.5	297	0.5	123.75	148.5
Real-time Clock (Active)	3.3	200	200	660	660	0.1	66	66
Real-time Clock (Standby)	3.3	110	110	363	363	0.9	326.7	326.7
Voltage Reference	3.3	6.65	6.65	21.945	21.945	0.32	7.0224	7.0224
SD Card (Write)	3.3	45000	45000	148500	148500	0.08	11880	11880
SD Card (Read)	3.3	40000	40000	132000	132000	0	0	0
Pressure Sensor	5	10000	10000	50000	50000	0.2	10000	10000
Temperature Sensor	3.3	65	81	214.5	267.3	0.1	21.45	26.73
LED	5	15000	30000	75000	150000	0.02	1500	3000
Minimum		200	235	718.5	912		558.45	727.2
Maximum		45650	45788	150976.5	151587.9		13121.85	13485.15
Total		111176.65	126365.65	409987.445	485845.145		25267.572	27189.702

Figure 12: Power calculations for State 1

State 2

Component	Voltage (V)	Nominal Current Draw (uA)	Maximum Current Draw (uA)	Nominal Power Consumption(uW)	Maximum Power Consumption (uW)	% Time Used	Nominal Power Used (uW)	Maximum Power Used (uW)
3.3V Linear Regulator	7.2	15	35	108	252	1	108	252
3.3V Switching Regulator	7.2	70	90	504	648	1	504	648
5V Switching Regulator	7.2	70	90	504	648	1	504	648
Microcontroller (Active)	3.3	565	663	1864.5	2187.9	1	1864.5	2187.9
RTC (standby)	3.3	110	110	363	363	1	363	363
SD Card (Read)	3.3	40000	40000	132000	132000	0.11	14520	14520
RS-232	3.3	300	1000	990	3300	1	990	3300
Iridium Modem (Active)	5	350000	1000000	1750000	5000000	0.333	582750	1665000
Iridium Modem (Standby)	5	66000	66000	330000	330000	0.667	220110	220110
Minimum		67130	67988	334333.5	337398.9		224443.5	227508.9
Maximum		351060	1001898	1753829.5	5006750.9		586579.5	1671750.9
Total		457130	1107988	2216333.5	5469398.9		821713.5	1907028.9

Figure 13: Power calculations for state 2

Team Charter

Below is the team charter that was created by both members. The creation of this document allows rules to be set and followed by each member.

TEAM CHARTER

Team Name: *SOE Sea Monkeys*

Members:
Edgard Rincand
Stefan Gadomski

Roles:
Edgard Rincand (Team supervisor #1, Engineer)
Stefan Gadomski (Team supervisor #2, Engineer)

Mission Statement:
"To provide our client MBARI with a product that meets all of the requirements given while treating this as a rare learning experience gaining the most knowledge possible".

Goals:
To have an operational product by mid-May, in order to be able to deploy the product into the Monterey bay.

Ground Rules:

1. *To come to class and team meetings on time and fully prepared.*
2. *Treat each other with respect.*
3. *Keep each other inform of any changes or problems that pertain to ones certain task.*
4. *Be open minded regarding new ideas.*
5. *Give reasonable time to inform member if not able to attend meeting.*
6. *Encourage and help each other even if the certain task is not your own.*
7. *Ask questions when you do not understand something.*
8. ***DON'T WASTE VALUABLE TIME***

(Failure to abide to any of the above rules will result in a group meeting with Cyrus and Professor Peterson. If problems persist the member must consider the possibility of being terminated from the project with the consent of both professors and the other team member.)

Signatures:
Edgard Rincand _____ DATE___/____/___
Stefan Gadomski _____ DATE___/____/____