Winter 2009

Implementation of a dive computer

Andras Fekete
University of New Hampshire, Durham

IMPLEMENTATION OF A DIVE COMPUTER

BY

ANDRÁS FEKETE

BS, University of New Hampshire, 2006

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

in

Electrical Engineering

December, 2009

This thesis has been examined and approved.

Thesis Director, John R. LaCourse Ph.D

Chairman of Electrical and Computer Engineering department

Paul W. Latham II Ph.D

Affiliate Professor of Electrical and Computer Engineering

Allen D. Drake Ph.D

Associate Professor of Electrical and Computer Engineering

Jeff Bozanic, Ph.D

President, Next Generation Services

Date

ACKNOWLEDGEMENTS

I would like to give immense thanks to Dr.Paul Latham for his guidance and expertise in the development of this work. Without his continuous support and praise I would not likely have gotten this far.

I give great gratitude to Dr.John LaCourse, Dr.Allen Drake, and Dr.Jeff Bozanic, for being on my thesis committee for their advice and help during my research.

I would also like to thank L&L Engineering for lending me the equipment to test my thesis work and for giving funding to components that I need. My colleague Stew Kenly also has been very helpful in giving me ideas during times of trouble and headache.

My thanks go out to my parents Balázs and Györgyi, and my sister Zsuzsi whom have given me courage and the desire to succeed in my studies.

I also give my thanks to my advisor Dr.Andrew Kun, who believed in me when even I doubted myself on my road to getting a Masters degree at the University of New Hampshire.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ADC – Analog to Digital Converter

CCR – Closed Circuit Rebreather

FAT – File Allocation Table

FET – Field Effect Transistor

FSW – Feet of Sea Water

IC – Integrated Circuit

LCD – Liquid Crystal Display

LED – Light Emitting Diode

MUX – Multiplexer

OLED – Organic Light Emitting Diode

PLL – Phase Locked Loop

PWM – Pulse Width Modulator

SAC – Surface Air Consumption

SD – Secure Digital

TTS – Time To Surface

UML – Unified Modeling Language

ABSTRACT

IMPLEMENTATION OF A NOVEL DIVE COMPUTER

by

András Fekete

University of New Hampshire, December, 2009

Diving, whether it is a recreational sport or an occupation, can pose numerous safety concerns. Two of the principle concerns are drowning and decompression sickness. Drowning can be caused by many factors, one of them being: running out of air. This thesis is concerned with implementing a dive computer with real time dive planning capability. This allows for more freedom for the diver, without incurring additional risk of decompression sickness. In this thesis, I plan to present my algorithm which is differs from other dive computers in the way it calculates multiple ascent routes, allowing divers to alter their precomputed ascent path by determining alternate decompression schedules and breathing gas requirements.

CHAPTER 1

INTRODUCTION

Researchers and military groups have been recording for many years the effects of diving and pressures on the human body [1]. Normally, a diver makes a plan before diving, and decides on how long, and how deep he or she will dive, and what ascent path will be taken to get back up to safety. Typical dive planning is based on predetermining the dive beforehand and following an appropriate dive trajectory. This trajectory is designed to stay within the available gas supplies and minimize the occurrence of decompression sickness. The problem with this approach is that a diver may wish to deviate from this plan for both safety and enjoyment or for task reasons. The safety reason is that some unknown factor may force a change in the dive plan (getting lost, trapped in an over head environment or entanglement). The other reason for a change is something that the diver might wish to do that was not anticipated. This thesis explains the procedures taken in developing a dive computer that has the ability to compute ascent paths for the diver depending on the dive history and possible future choices.

1.1 Problem description

New dive computers are getting rather sophisticated. They have microprocessors that calculate the ascent rates and dive stops. However, they have a flaw. The gas supply and dive time is based on the fastest safe accent rate. What happens if a slower accent rate is desired? A larger gas supply is needed, but how much? When exploring a

shipwreck or coral under water, it may be better if during ascent the diver can freely chose an alternate safe ascent path instead of the precomputed path.

The author decided to implement a system which gives real-time calculations of ascent trajectories based on the Bühlmann algorithm [2]. This is a well known and proven algorithm. Other algorithms could be used as well; however, the basic integral planning concept would still be the same. The proposed dive computer would take into account the past history of the dive, compute the safest path to the surface, compute a curve of the current ascent rate, and show whether it would intersect the safe ascent path prescribed by the Bühlmann algorithm. With adequate gas supplies, the algorithm gives a prediction of how long each ascent path would take. The benefit is that divers could change their dive plan on the fly and not have to worry about running out of gas. It would also allow for reclaiming much of the dive time by exploring things on the ascent as well. For a decompression dive most time spent in is during decompression.

Another difficulty with dive computers is having waterproof buttons. It is proposed to have capacitive touch buttons to eliminate mechanical moving parts. With many devices becoming touch sensitive, it becomes apparent that having a similar interface would alleviate much of the engineering that goes into designing the user interface to dive computers, which is a major source of failure due to water leakage.

One of the serious issues with diving is the price of even a modest diving watch is out of reach for many. With processing power drastically becoming cheaper over the years, it should be very easy to afford a high-speed microprocessor. This would lessen

the burden on timing constraints in design, as well as give the final product the ability to control more devices and subsystems.

An additional feature of this dive computer is the integration of pressure monitoring and navigation with optional electronic control of oxygen levels for rebreathers. Future additions could include text messaging using the work of Bert Franzheim at the Center for Coastal and Ocean Mapping at UNH.

1.2 Approach

A new system is proposed that will be able to calculate and adapt to a diver's profile and in real-time calculate safe ascent paths. This system will have the form factor of a small diving watch. Time and pressure are the two variables used in the Bühlmann equations. The system uses an LCD display to show data statistics. It will be dynamically updated depending upon the actions of the diver. There is a microprocessor used for calculations and driving all the electronics. The display will also show the past dive profile and also show proposed ascent profiles with their respective Time-To-Surface values. A 32.768KHz oscillator is placed on-board for accurate measurement of time. This method has been proven to work well, consumes little power, and is the standard used in digital watches. The proposed system also has expandability options, such as more analog inputs than currently are necessary. It is required that this system be battery powered and should last at least 24 hours of continuous use with a single set of batteries.

Common features on dive computers include alarms. These may be audible or visual. The implementation of these features is simple. Besides a warning on the LCD, status indicator LEDs will be used as a heads up display to alert the diver's attention to the LCD. Buzzers are used on a dive computer to alert critical problems such as: decompression stop was missed, low gas supply, etc.

This platform will also have many expandability options that can be implemented in future work. These could include integration with a CCR system, the ability to have the form of a narrow bandwidth communication device, and perhaps even navigation features such as a magnetic compass.

Testing of the device was done in a laboratory environment. The two measured variables (time and pressure) were modified to validate the output of the system. The pressure sensor puts out an analog voltage to specify hydrostatic pressure. This can be replaced by a variable voltage source to simulate desired pressures.

CHAPTER 2

BACKGROUND

Exploration of the many water bodies of Earth has been a challenge for land walking humans. The human body as it is today is not meant for exposure to water for long periods of time or great depths. There have been numerous studies and devices created to make a shell or apparatus to extend the amount of time the body will last underwater.

Diving has evolved into a sport and hobby by many. Others use the opportunity to research and explore the deep for the benefit of others. In the early days of swimming under water, an air line was used to feed air to the person. Then with the idea of taking a tank of compressed air, in 1943 called Aqua-lung [3] by Emile Gagnan and Jacques Cousteau, the tether was removed. Aqua-lung was the first open-circuit scuba diving equipment. This device, which is still used regularly, works by using compressed air and a dive regulator, which makes air at ambient pressure available to the diver through a demand valve. The used air is exhaled through the same apparatus, but a valve diverts exhaled air to the exhaust hose, expelling the air near the tank so the bubbles don't obscure the vision of the diver.

These systems haven't been used since the 1970s as having a dual hose design was more cumbersome to put on as well as design. They have been replaced by a single hose scuba system that is still in use today which has a single hose from the tank with a

two stage regulator that makes air at ambient pressure. The first stage is at the tank which reduces the pressure from 3000 psi to about 145 psi, then the demand valve regulator makes the ambient pressure. The exhaled gas is released under the regulator.

With the increasing use of rebreather scuba systems since the 1990s, dive time can easily reach more than 3-6 hours. With these long dive times, dive plan flexibility is increasingly desirable. The way this can be accomplished is instead of having a so called Open Circuit dive where the exhaled air is released into the environment they have Closed Circuit Rebreathers (CCR) where the exhaled gas is passed through scrubbers to remove most of the carbon dioxide. This breathing gas is then replenished with more oxygen and ready for use. It is important to note that of the exhaled gas, 98% of the oxygen that was breathed in is exhaled. For this reason is why rebreathers allow for a drastic increase in time spent under water with the same volume of gas.

2.1 Dive computer history

The first analog dive computer was made in 1957 by the Foxboro Company for the US Naval Experimental Diving Unit [4]. After much research and measurement of people's dives, researchers in Toronto, Canada were able to test the design of a pneumatic dive computer [5]. Since then much has changed. Eventually, dive computers replaced depth gauges, diving watches and dive tables.

The more time a diver spends under water, the more nitrogen gets dissolved into human tissue. Dive computers display the total time remaining at the current depth without the need for decompression stops while ascending (no decompression limit), or

they can display the minimum depth ceiling when decompression is required. It is assumed that the dive has been properly planned so that sufficient breathing gas is available. If there is a significant change in the dive plan, this assumption may not be valid. As a result, a decompression dive must be preplanned, and the diver is required to "plan the dive" and "dive the plan".

Other manufacturers have been able to construct dive computers that are suited for a single decompression model in diving. This requires the diver to do much more planning before a dive. For example, an early device was the SOS Decompression Meter sold since 1959 by Scubapro (see Figure 1). It was also unofficially known as the Bends-O-Matic for its mediocre operation. The device was a pneumatic device with two air chambers that attempted to model the nitrogen absorption of tissue. This device was very popular into the late 70s.

Figure 1: Scubapro's SOS Decompression Meter

Farralon Decomputer started producing an electrical device in 1975 that had a color-coded display showing green to yellow to red, according to the danger of getting sick. A fundamental flaw with this device was the very simple decompression algorithm used: it became less reliable with repetitive dives.

The Orca EDGE was released in 1983 which used a microprocessor to estimate the nitrogen saturation of each of the 12 human tissue compartments it had in its algorithm. A smaller version of the same device was called the Skinny Dipper.

Oceanic Datamaster was a new device created in 1985 which combined a pressure gauge to determine how much air the diver used to adjust the algorithm. Orca Phoenix and the Dive Rite Bridge were the first dive computers that supported using different gas mixture, such as nitrox.

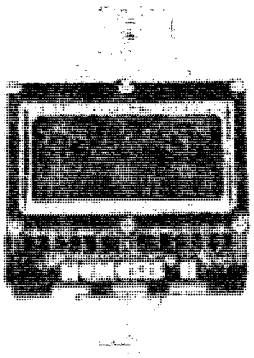

Figure 2: Cochran Nemesis II

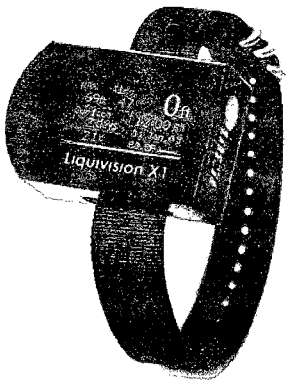

Figure 3: LiquiVision X1 dive computer

Soon after, in 1993, Cochran and UWATEC released dive computers that didn't require a hose from the computer to the tank, but instead involved attaching a device to the tank to transmit the pressure.

Cochran also introduced a dive computer (Nemesis II: see Figure 2) that was able to use multiple types of nitrox gas mixes to calculate dive time.

In early 2009, Liquivision introduced the X1 (Figure 3). This is a dive computer which is based on an open source platform. It has support for multiple decompression algorithms such as different versions of the varying permeability model. The innovation comes from the accelerometer based input system where the user flicks their wrist to achieve the effect of a button press.

2.2 Dive computer algorithms

There are many algorithms developed for diving. These include the Mares-Wienke Reduced Gradient Bubble Model [6]; Bühlmann ZHL-16 [2], ZHL-12 and the ZHL-8, Multi-Tissue Model, and the Varying Permeability Model [7]. These models are

based algorithms that have been fine tuned by averaged recordings of diver experiences. None have taken age, body type or any factors related to the physique of the diver into consideration. Nor have they taken a look at ambient temperature, though most dive computers now report this information. New research is being done to determine the effects of temperature. According to studies done by the US NAVY [8], having warmer conditions during decompression help reduce the effects of decompression sickness to a point. Heat stress can still occur in hotter water temperatures.

2.3 Bühlmann algorithm explained[9]

Nt1/2	Na	Nb	Ht1/2	Ha	Hb
4.0000	1.2599	0.5050	1.5000	1.7435	0.1911
8.0000	1.0000	0.6514	3.0000	1.3838	0.4295
12.5000	0.8618	0.7222	4.7000	1.1925	0.5446
18.5000	0.7562	0.7725	7.0000	1.0465	0.6265
27.0000	0.6670	0.8125	10.2000	0.9226	0.6917
38.3000	0.5933	0.8434	14.5000	0.8211	0.7420
54.3000	0.5282	0.8693	20.5000	0.7309	0.7841
77.0000	0.4701	0.8910	29.1000	0.6506	0.8195
109.0000	0.4187	0.9092	41.1000	0.5794	0.8491
146.0000	0.3798	0.9222	55.1000	0.5256	0.8703
187.0000	0.3497	0.9319	70.6000	0.4840	0.8860
239.0000	0.3223	0.9403	90.2000	0.4460	0.8997
305.0000	0.2971	0.9477	114.1000	0.4112	0.9118
390.0000	0.2737	0.9544	147.2000	0.3788	0.9226
498.0000	0.2523	0.9602	187.9000	0.3492	0.9321
635.0000	0.2327	0.9653	239.6000	0.3220	0.9404

Table 1: Bühlmann constants

The Bühlmann algorithm uses 16 compartments with different nitrogen and helium constants that have been derived. These constants are shown in Table 1. At each time step every one of the compartments ($P_{comp}(i)$) is recalculated using the present depth and the amount of time that has passed since the last calculation. In equation form the procedure is the following:

$$P_{comp}(i) = P_{comp}(i) + (P_{gas} - P_{comp}(i)) * (1 - 2^{(t_{exposure}/t_{halfTime})})$$

For the case when multiple inert gases are used, the algorithm is evaluated for each gas's partial pressure (P_{gas}) individually. A common mixed gas is Trimix, which is a blend of two inert gases nitrogen(N_2) and helium(He) along with oxygen(O_2). Oxygen may cause bubbles as well, but its effects are very small because oxygen is metabolized. To calculate the maximum ceiling the diver must stay below during the dive, one must calculate the 'a' and 'b' components for the compartments and combine them.

$$a_{cor}(i) = \frac{N2_a(i) * P_{compN2}(i) + He_a(i) * P_{compHe}(i)}{P_{compN2}(i) + P_{compHe}(i)}$$

$$b_{cor}(i) = \frac{N2_b(i) * P_{compN2}(i) + He_b(i) * P_{compHe}(i)}{P_{compN2}(i) + P_{compHe}(i)}$$

$$ceiling = (P_{compN2}(i) + P_{compHe}(i) - a_{cor}(i)) * b_{cor}(i)$$

$N2_a(i)$ represents the i^{th} compartment's 'a' value for nitrogen, $N2_b(i)$ represents the 'b' value. Similar concepts apply for He_b. $P_{compN2}(i)$ is the calculated pressure for the i^{th} nitrogen compartment.

The only values necessary to store are the table of half times, 'a' and 'b' values, and the current percent inert gas in each compartment of each gas. These a and b parameters are scaled to incorporate additional conservatism by so called gradient factors. They serve as factors used by the Bühlmann algorithm to determine the tolerated ambient pressure. They can be calculated from the following formulas:

$a = 2 * t_{ht}^{-1/3}$ $\quad b = 1.005 - t_{ht}^{-1/2}$

where t_{ht} is the half-time for the compartment (see Table 1).

CHAPTER 3

OBJECTIVES

Existing diving hardware has many limitations. They are bulky, use a fixed maximum accent rate trajectory, and they are often inaccurate if the internal functionality is not fully understood by the operator. Additional problems include: cost, standardization, ease of use. The objective of this project is to incorporate low-cost, easy to use, reliable, open-source software system that incorporates many of the features commonly found in dive computers (such as: watch, depth gauge, decompression status, thermometer) enhanced with alternative ascent paths, non-volatile dive history storage, and solid-state design.

3.1 What is the problem?

The least expensive dive computers are about $150 and they have severely limited functionality. They are essentially a glorified watch and depth gauge. The watch cannot possibly contain many expensive components. Dive computers can be as expensive as $2400. This hefty entrance fee turns off many people who would be interested in taking up diving for sport or researching underwater life.

Standardization is another key problem with these devices. For example, a newly acquired watch or car can be operated within minutes if one is familiar with the operation of other watches and cars. On the other hand, every diving watch differs in its interface

design. Like a watch or a car, every dive computer should employ similar operating procedures.

3.2 Why should this be done?

From preliminary research, it was determined that it is feasible to create a more elaborate dive computer for a much reduced cost. The advantage of doing it as a university project is that it can be easily published as an open source project, which will pull manufacturers in a single direction and give opportunities for new groups to form and use a canned solution. Then, like many projects in the open source community, this too will be able to benefit from the collective knowledge of many individuals.

A problem with diving is that divers must carry with them several pieces of instrumentation, especially in CCR applications. These include: compass, depth gauge, chronometer, compression calculator, thermometer, and with a CCR, oxygen control electronics. When surfacing and waiting to decompress after a dive there is rarely any provision for the entertainment of the diver. Since there isn't likely anything to look at or explore, it would be nice to have text messaging, an e-book reader, games or better yet an MP3 player, which are already available in separate products. One can bring such along, but that is another piece of equipment that can get lost or get tangled and cause problems for the diver. The more equipment, the greater the weight, complexity and risk. In an ideal world, there would be one simple integrated device.

Another major reason for this project was to increase the safety of individual divers. When underwater, the dive computer is the lifeline of the diver. With built-in

redundancies and on-the-fly recalculation of decompression plans, even an advanced diver can be more at ease.

3.3 What is available?

One of the current high end dive computers, the VR3, has many of these features. The unit presently costs $1400, and one must spend even more for "PIN codes" to unlock these additional features. Even for that much money, they don't have the ability to recalculate a safe ascent path in case the diver deviates from the plan suggested by the dive computer.

On the other hand, the VR3 also is the only dive computer found that has a graphical display. Most others have a custom made display. In recent years, display technologies have greatly improved and have become much more energy efficient. If cell phones with a graphical displays can last many days and even weeks, then it should be trivial for a small watch to achieve similar results.

The Liquivision X1 is currently available for $1750. Since it uses open source software, upgrades will be free and can be loaded by the end-user. It also uses a 128x64 pixel organic LED screen with a battery life of up to 20 hours. It only displays the safe ascent path of a single dive.

CHAPTER 4

HARDWARE IMPLEMENTATION

The design and development of this device started with research into existing diving watches. This was discussed in the background section. With knowledge about what exactly needs to be accomplished, a preliminary schematic was drawn out as shown in Figure 4.

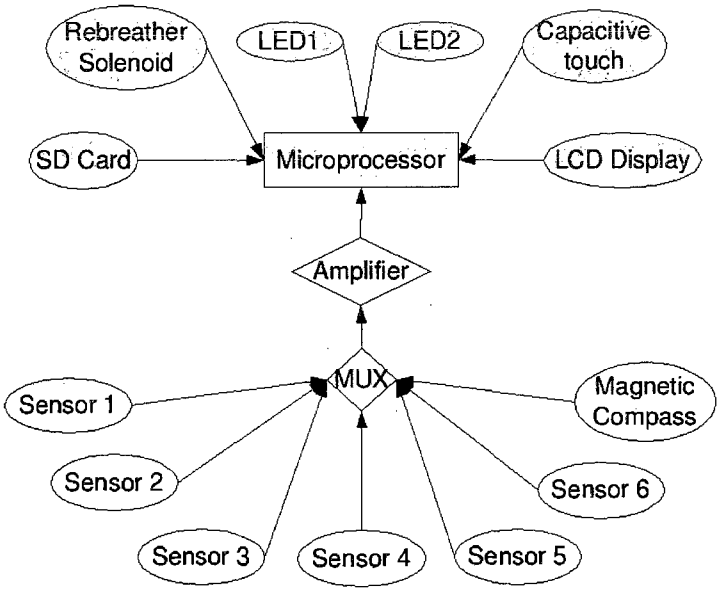

Figure 4: General system overview

4.1 General implementation

This project incurred two board revisions. However, the system has maintained the same general concept as presented in Figure 4. The microprocessor is the "brains" of the system, with the sensors multiplexed and amplified into one port. This obviates

having duplicates of amplification components. Since the sensors don't need to be sampled rapidly, reasonable multiplexer delays are acceptable.

The sensors are configured to have 3 different depth sensors. This is done for redundancy; the two measurements that are closest to each other are taken as the depth measurement. Special instructions to warn of a potential sensor failure are added in case the third measurement differs by a certain amount.

In the case of a CCR, one would have 3 oxygen sensors attached and monitoring in the same fashion as a depth measurement. In an open circuit dive, a single sensor would be hooked up to measure the gas consumption to use in the algorithm for estimating the time remaining.

4.1.1 Display selection

The device that took the longest amount of design time was the LCD screen, since it is important to have low power consumption but bright display. Cellular phone displays were considered, as well as monochrome displays simply because of their efficiency. Size was of big concern, as the size of the display will determine the size of the dive computer. A company that makes a small organic LED (OLED) color display was found. This was useful for two reasons; OLED displays are by far much more energy efficient, and secondly, the display has multiple colors. This allows for display of data in a more organized fashion. However, it turned out that the display was much too small for it to be visible under water. Human eyesight diminishes greatly underwater,

and the small pixels were hard to see. It was decided to design a new board for ergonomics and other reasons later described.

4.1.2 Microprocessor

The chosen processor for this device was the ARM STM32F101. This processor is capable of running at 3.3V, and it has very little current consumption. The internal clock speed can also be rather fast. With an 8MHz clock, it could scale up to 128MHz with an internal PLL. As with nearly all low power microprocessors, it employs an external 32KHz crystal for keeping track of time and date. This is to be used in conjunction with low power modes so the device can shut down while it is idle.

Internally, the processor also has a temperature and a voltage sensor. It will be used to measure the ambient temperature and battery life respectively. Battery life is important to monitor as many of the other chips on the computer require a minimum of 2.5V to operate.

4.1.3 Device outputs

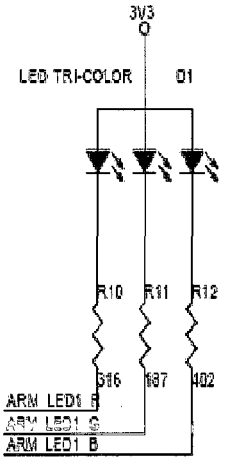

Figure 5: Connection diagram of the LED

Two tri-color LEDs (one for heads up display) were also placed on board to use as status indicators when the LCD is not powered up. They are for alerting the diver to danger or status updates when in idle mode. A buzzer was also placed in the watch to have an audible alarm in case of emergencies. A solenoid actuator circuit was also incorporated to be used in conjunction with an air regulator in rebreathers. A MicroSD card slot was installed to store the dive history. This will make it easy to transfer the captured data to a computer after a dive for analysis.

4.1.4 Device inputs

A series of sensors were planned with this device. It was designed in a way to have the ability to expand the capabilities of the dive computer later on as ideas and

opportunities present themselves. The dive computer has a bank of small signal devices. For example; the design is to have three pressure sensors measuring ambient pressure. Then the two that are closest in measurement are taken as the measured pressure so that in case one of the sensors has gone bad, there will be backups.

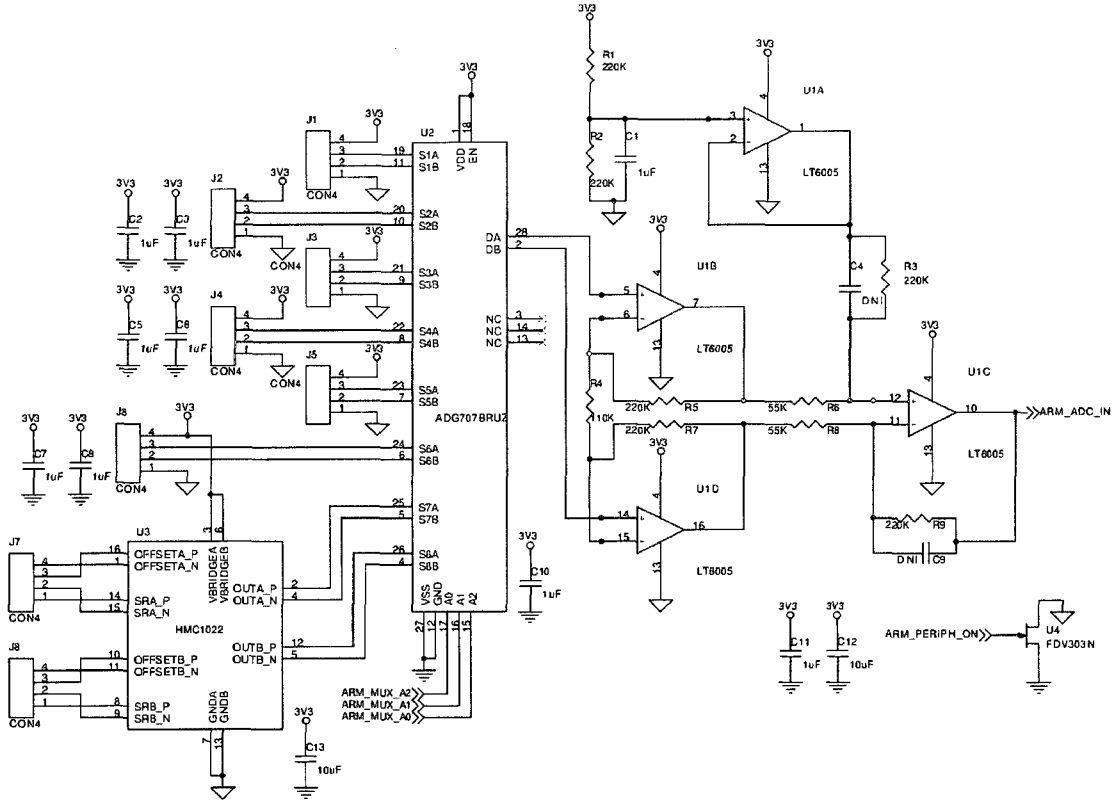

Figure 6: The input signal selector and the instrumentation amplifier

The measurement of pressure is the most critical part of the dive computer. The output voltage from the sensors is assumed to be from about 60mV to 120mV. For the

other three low-voltage inputs, different air pressure sensors are planned. With today's divers, it is not uncommon to go down with more than 1 kind of breathing gas.

Other inputs include a two dimensional compass. All these analog inputs mentioned thus far are switched by a 8:1 multiplexer IC. The output of the MUX is fed into the amplifier circuit which is connected to the microprocessor's ADC (see Figure 6).

Three push buttons and three capacitive touch buttons were also designed in. Capacitive touch is the way one communicates with the dive computer under water. Making things water tight is a very tough thing to do, especially if wires or moving parts must be accessible on the outside. Capacitive touch is used in many devices like cell phones, and automotive devices. Having such a feature will likely increase the durability of the device. Figure 7 shows the capacitive touch test board with a 9V battery as a size comparison. In production models, it is recommended to have this built on a flex circuit

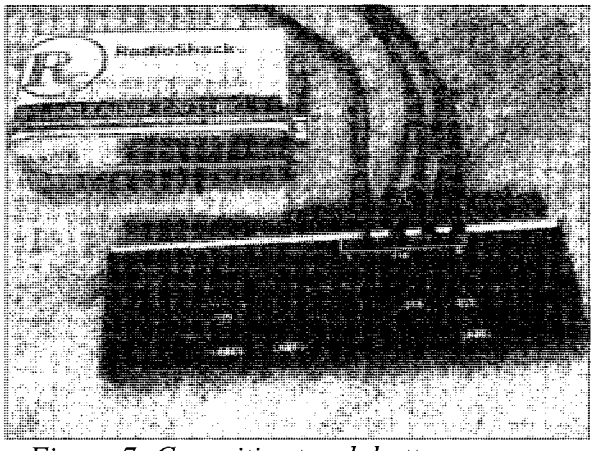

Figure 7: Capacitive touch buttons

board as it is easier to seal around a flex circuit than a ribbon cable connector.

4.1.5 Power supply

The input power to the dive computer comes from any type of battery capable of producing 2.8V-3.3V. The supply voltage is measured and can be applied to the readings from the analog inputs as a scale as the battery depletes the voltage drops and the instrumentation amplifier also droops. A boost converter is also added to drive the backlight of the display. This system uses a simple inductor-diode-FET system to boost up the voltage with input and output capacitors to store charge.

4.1.6 Sub-circuits

The ground connections are switched on or off by FETs for certain circuits, those that consume significant amounts of power. The analog ciruits, display backlight and capacitive touch sensors are all switched in this way. When in low power mode, the analog circuits and the backlight can be both switched off. Capacitive touch sensors are intended to be measured periodically, so they too can be turned off while not in use.

4.2 Hardware layout v1.0

Design decisions for layout of hardware were complicated. A ground plane is critical for noise reduction as this device has both digital and analog circuits. Decoupling capacitors were placed for each VDD pin on each device for greater noise immunity on the supply line. Large input capacitors were placed at the battery terminals, so that in case of large current spikes, the batteries would not have to work as hard. Sources of electrical noise like the crystal oscillators and the boost converter were placed on the

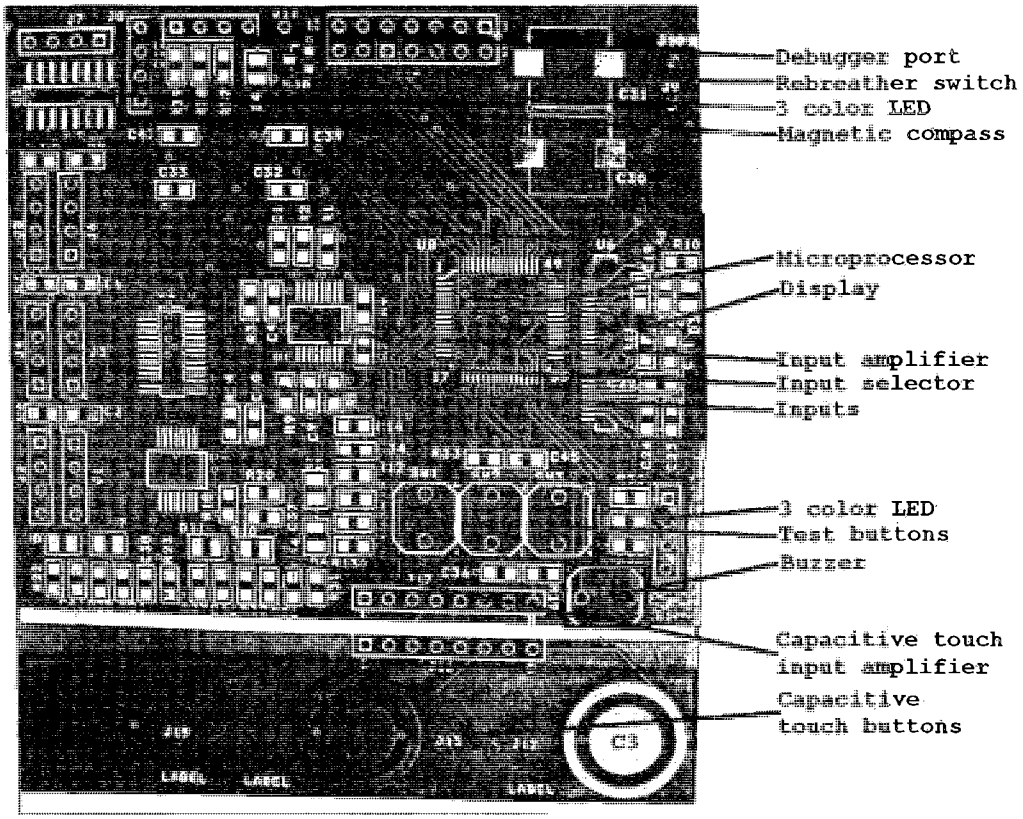

Figure 8: Development board PCB

bottom side of the board as far from analog and sensitive measurement devices as possible.

The circuit board was made to be as compact as possible to fit the form factor of a typical digital watch. The capacitive buttons were made to be detachable for the purposes of demonstration and development. The area of the inner pad is roughly equivalent to the area of the outer pad. The way the system works is that a square wave generated by a PWM is launched into one of the pads, and the voltage out is measured. The measurements are amplified and fed into the ADC of the microprocessor.

To conserve space, the MicroSD card slot was placed on the back. The display can be folded around the rear. The analog inputs all have dedicated supply and ground connections with decoupling capacitors. The ground plane is split so that it can be centrally enabled and disabled by the FET for analog signals. The current is steered around central hot spots; however, there should not be much current flowing through the sensor circuitry. These hot spots are also placed as far as possible from sensitive ICs. The multiplexer is always on as it does not consume much current when not switching.

For debugging, a 14pin debugging header is populated for programming the ARM. This can be easily removed for deployment.

4.3 Hardware layout v2.0

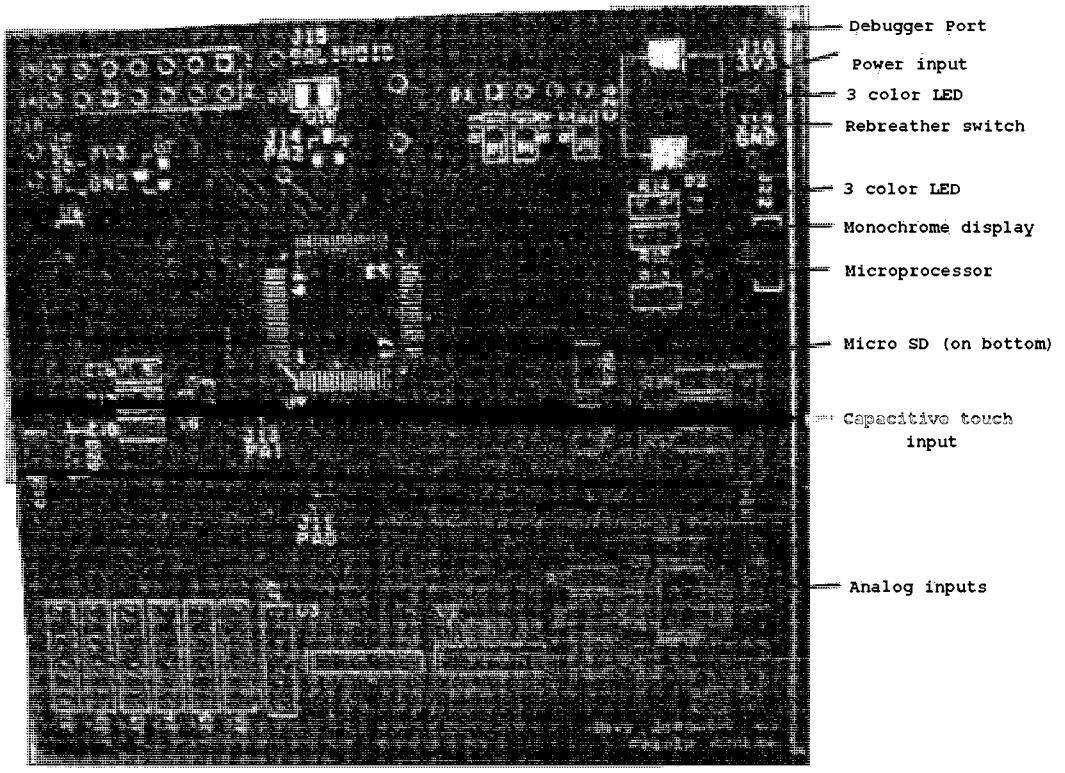

Figure 9: Development board V2

This layout is much simplified since the capacitive touch is a simple RC circuit. This represents a large reduction in power usage as well as complexity. The theory of operation for this is that all the buttons are pulled high at the same time and each button is measured to see how long it takes for the RC circuit to charge up past the logic threshold of the ARM. With the internal system clock, this can be done very accurately. Placing a finger on the contacts measurably changes the capacitance. This is explained in the results section.

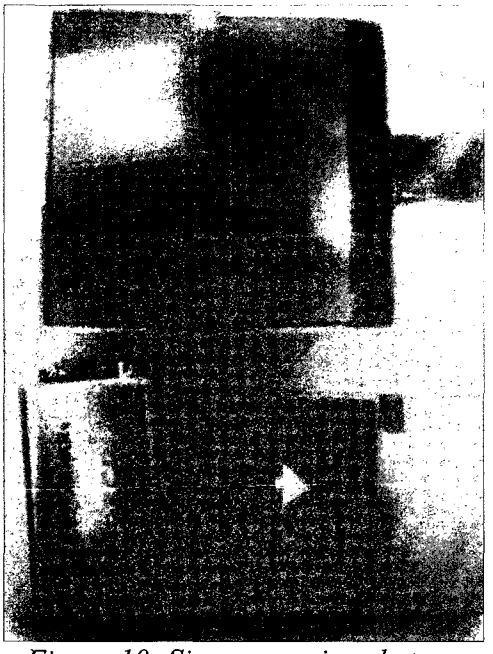

Figure 10: Size comparison between LCDs

The other major difference with the new layout is that the LCD no longer requires its own 12V supply. This has the added benefit of reducing the noise on the board generated by the switching boost converter as well as freeing up a timer on the ARM. In effect this represents a power savings so the ARM has less work to do. The boost converter in the LCD is also likely to have better efficiency. The change of the LCD was mainly to solve the ergonomic problem with the small display. The difference in size is shown in Figure 10. An added benefit of the new display is that it is possible to turn off the backlight and still see the graphics displayed on the screen.

4.4 Cost of the hardware

The costs are shown in Table 2. These costs are for single quantities, so that if one were to mass produce such a product, the actual costs will be much lower per unit. As it stands, the total cost of $168.58 is still much less than the price of the average dive computer. This platform also provides support for much more functionality than others on the market today.

The capacitive touch sensors can be fabricated out of FLEX connectors. In single quantities the etchings are very expensive as the tooling setup costs are most of the costs. To have it manufactured, $1200 would have to be spent on the tooling. Individual button circuitry will only cost $30 per piece. Therefore in bulk the buttons will be also affordable.

Idx	Qty	Part Number	Description	Unit Price	Total
1	22	445-3463-1-ND	CAP 1.0UF 50V Y5V 0805	$0.10	$2.18
2	4	490-3347-1-ND	CAP 10UF 16V Y5V 0805	$0.26	$1.05
3	1	PCE3097CT-ND	CAP 100UF 10V ELECT FC SMD	$0.49	$0.49
4	2	478-3735-1-ND	CAP 20PF 5% 100V NP0 0805	$0.23	$0.46
5	2	399-1110-1-ND	CAP 12PF 50V NP0 0805	$0.08	$0.16
6	7	P220KADCT-ND	RES 220K OHM 5% 0805	$0.17	$1.16
7	1	P110KADCT-ND	RES 110K OHM 5% 0805	$0.17	$0.17
8	2	RHM54.9KCRCT-ND	RES 54.9K OHM 1/8W 1% 0805 SMD	$0.04	$0.07
9	2	RHM316CCT-ND	RES 316 OHM 1/8W 1% 0805 SMD	$0.04	$0.08
10	2	RHM187CRCT-ND	RES 187 OHM 1/8W 1% 0805 SMD	$0.04	$0.07
11	2	RHM402CRCT-ND	RES 402 OHM 1/8W 1% 0805 SMD	$0.04	$0.07
12	1	RHM4.75KCCT-ND	RES 4.75K OHM 1/8W 1% 0805 SMD	$0.04	$0.04
13	3	RHM1.0MARCT-ND	RES 1.0M OHM 1/8W 5% 0805 SMD	$0.03	$0.10
14	1	LT6005CGN#PBF-ND	IC OPAMP R-R QUAD 16-TSSOP	$3.50	$3.50
15	1	ADG707BRUZ-ND	IC MUX 16CH 1.8-5.5V 28-TSSOP	$6.30	$6.30
16	1	342-1005-1-ND	LINEAR MAGN 2 AXIS 16SOIC	$17.00	$17.00
17	4	FDV303NCT-ND	MOSFET N-CH 25V 680MA SOT-23	$0.28	$1.10
18	1	785-1078-1-ND	MOSFET N/P-CH 20V 6-TSOP	$0.53	$0.53
19	1	497-6441-ND	MCU ARM 512KB FLASH 64-LQFP	$14.33	$14.33
20	1	HR1940CT-ND	CONN MICRO SD R/A SMD	$2.02	$2.02
21	1	CTX506-ND	CRYSTAL 8.0000MHZ 20PF	$1.31	$1.31
22	1	SE2405CT-ND	CRYSTAL 32.7680KHZ 12.5PF SMD	$0.95	$0.95
23	1	OR723CT-ND	CONN FPC 30POS 0.5MM SMD	$1.52	$1.52
24	1	OR746CT-ND	CONN FPC 4POS 0.5MM SMD	$0.53	$0.53
25	1	458-1096-2-ND	BUZZER 3.65KHZ 1-30V SMT	$2.77	$2.77
26	1	NHD-C128128BZ-FSW-GBW-ND	LCD COG GRAPH 128X128 WH TRANSFL	$28.00	$28.00
27	1	350-2078-1-ND	LED R/G/B 1208 SMD	$2.63	$2.63
28	1	Circuit board	Dive computer main board	$80.00	$80.00
					$168.58

Table 2: Hardware bill of materials

CHAPTER 5

SOFTWARE IMPLEMENTATION

5.1 General overview

As with most other dive computers, the significant part of the device is the software that it runs. For this project, the ARM is programmed using the Raisonance Ride toolkit which is built upon the GCC compiler. The program is separated into pseudo classes for each sub-device or operation. This fact is represented in the UML diagram[10] (see Figure 11). They are called pseudo classes because in C there is no such thing as a class, merely structures, and those pseudo classes don't have constructors or destructors. By careful programming, a class can be emulated and implemented even in C. Each device pseudo class has a classname_config() function, which sets up the ARM to have the correct input/output pins and timers used by that device.

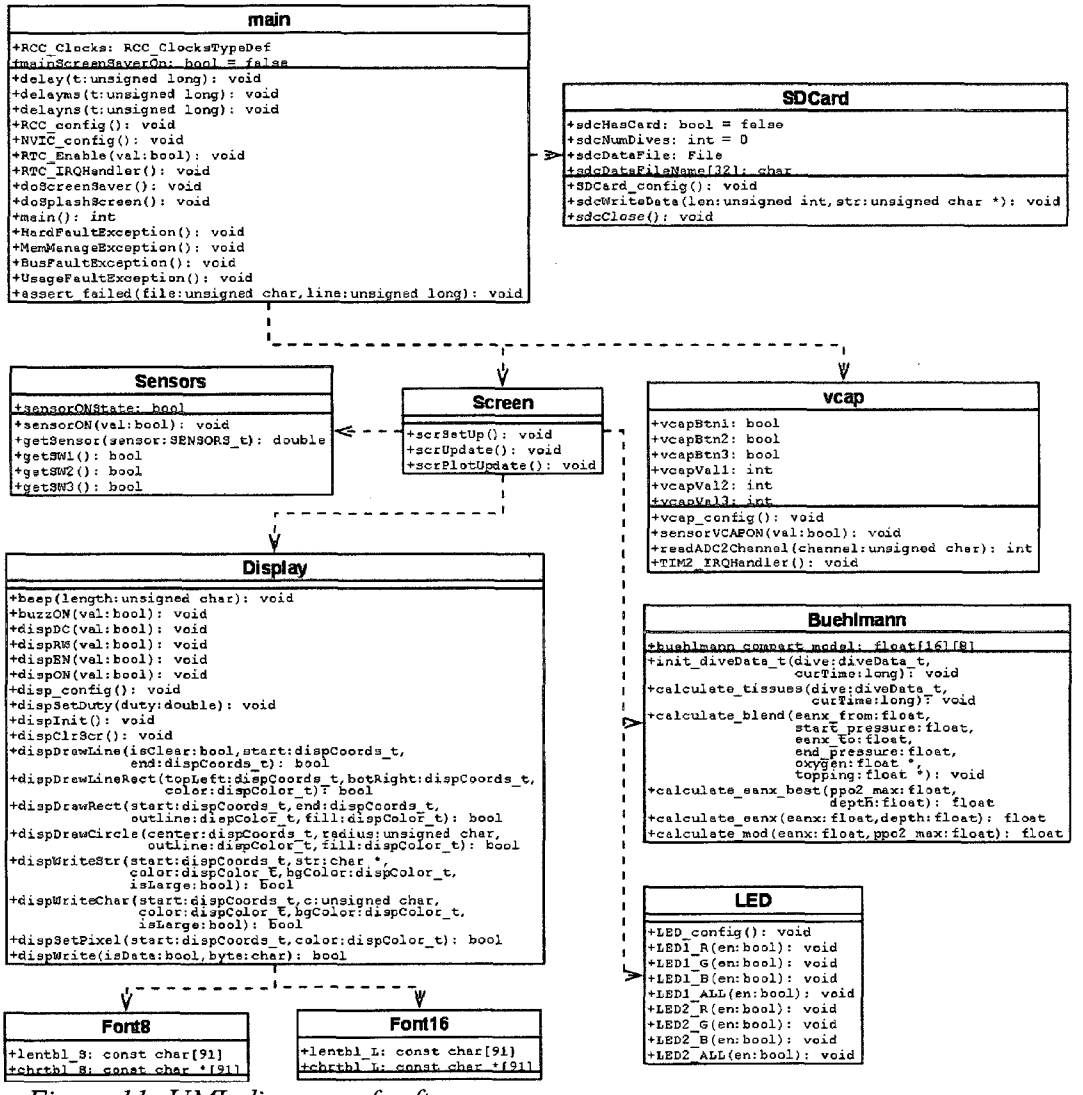

Figure 11: UML diagram of software

5.1.1 Buehlmann.c

The Bühlmann algorithm was originally adapted from the open source program GTKDive. It has been highly modified to match the constraints of this project. Many of the calculations were simplified. There is now a single call to the

*calculate_tissues(diveData_t *dive, long curTime)* function. This function does all the calculations necessary and uses the time stamp of the last execution of the function and the current time given by the second parameter. As mentioned earlier, the Bühlmann algorithm depends on time and pressure. Now that we have a delta time, we can use that in our computations to predict how much in-gassing the tissue compartments have done. The typical delta time is 1 second measured by the 32.768KHz crystal oscillator, currently set up as a timer-based event to take measurements. In the predictive ascent rates, the delta time is 1 second, but this can be varied by software by passing a *curTime* that is one second in advance. In testing the system, the time calculations were accelerated so that one second of test time was equivalent to one minute of dive time.

The current pressure is preset in the *dive* variable before the function is called. This is because when the ascent rates are calculated, the depth has to be a variable to determine what the model would give as the solution to yield time and pressure.

5.1.2 Display.c

This file is the low level driver for the display. When the software for the first version was written, it was verified that good coding style was followed, so that when a new display was connected, the only file needing change was the driver.

This class also utilizes the two font sizes where important information is in the center with large font. In a real dive computer, there normally aren't any labels as the user is expected to know what each number represents. At other times the labels are

printed on the casing to save the processor from having to write that information to the screen.

5.1.3 Screen.c

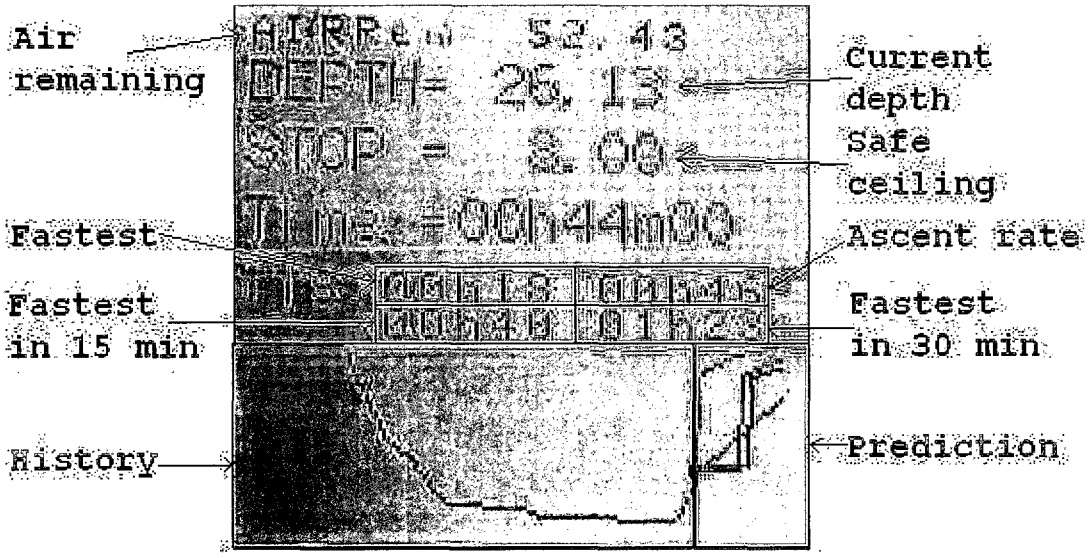

Figure 12: Explanation of the dive display

This class describes what will be printed on the screen when a call to update happens. There are two update methods: one for updating simple displays such as the current depth, time, and remaining air and one for plotting the past dive profile and making the predictions. For simplicity in computation and screen updates, the two update methods operate on separate sections of the screen. On the upper half are the simple pieces of information. The bottom half is reserved for the plot and predictions. This separation can be seen in Figure 12. In the picture, the various pieces of data displayed are outlined. In this example, a large history profile was selected.

The plot data is stored in local memory. Depth data is periodically captured and

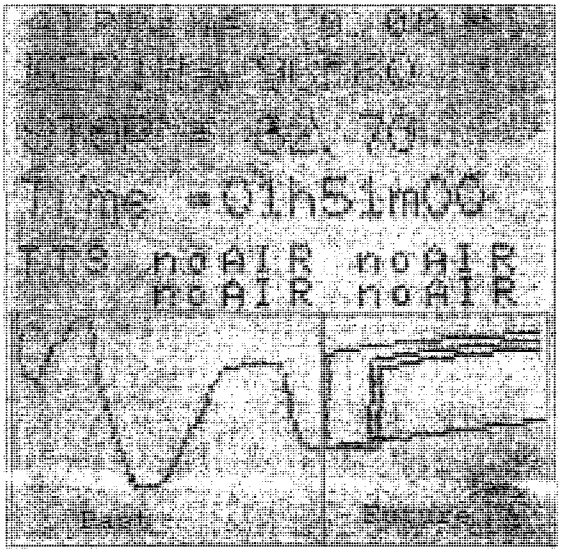

Figure 13: Demonstration of variable profile lengths

placed in a circular buffer. Since there are only 100 pixels available for dive history representation, the data points are averaged before being displayed. The code is written in such a way that this is user definable allowing for a longer prediction display or longer history display. Having the current dive data stored in local memory provides an accurate representation of the dive profile without the need for constantly accessing data from an external memory source, which can be slow. Figure 13 shows an example when the dive profile length is decreased in favor of a longer ascent prediction. This also shows a case where a 'bounce dive' (multiple ascents and descents before surfacing) is performed at deep depths.

5.1.4 Vcap.c

When vcapMeasure() is called, Timer2 is turned on, and it waits until Timer2 halts itself. This means that while the processor is making the 142 capacitance measurements, the main loop is stalled. Since the measurements take roughly $1/60^{th}$ of a second to complete, this might be a rather long time with respect to clock cycle time, but in reality it is not as big of a problem as one might think. The main loop is constantly interrupted by many events, even so only a small percentage of the time is used by handling events.

The measurements are conducted by using the processor's SysTick counter. First it is verified that the sensors are in fact registering a logic level 0. This is just a sanity check to make sure nothing is broken but more importantly so that the RC network has had enough time to settle since the last measurement. Next, voltage is applied to all three sensors at the same time, and the time is measured from then to when the voltage on the RC circuit surpasses the logic level 1.

When the values have all been collected, each is multiplied by the corresponding Hamming window coefficient. Since the window function is mirrored, it is only necessary to store half the constants in the processor.

5.1.5 Sensors.c

This pseudo class was extended by using the onboard temperature and voltage sensors in addition to the eight inputs available from the ADC input. Each input was

labeled in code by its function. On initialization, the ADC calibration routine is run. When the getSensor(...) function is called, a decision tree is executed to determine the source of the data being measured. Different set up procedures take place depending on the sensor. If it is external to the processor, then the MUX is adjusted accordingly.

5.1.6 Main.c

In the system as it stands now, the ARM runs constantly. A main() function initializes the device, then stays in a while(1) loop. The ascent curves are calculated continuously. Periodically it is interrupted by procedures to check the button states and to record the current depth.

Originally, at the end of the while loop, the code simply looped around to check if an interrupt occurred. This was highly inefficient, since any time a clock or data path switches, maximum energy is consumed. When the processor was placed in a standby mode at the end of the loop, the power consumption was reduced to 63% while waiting for an interrupt.

To save even more power, instead of continuously calculating graphs, the system can be placed in low power standby and the graphs recalculated every minute. This would extend battery life to be longer than about 1 day. In testing, two Radio Shack Alkaline AA batteries were used. These lasted for more than 1 day of continuous use of the v1.0 hardware. After 1 day, the system was still operational; however, there was insufficient voltage to light up the blue LED. All the other systems were seemingly functional.

5.2 Ascent trajectory calculations

When the processor is not doing anything, it reverts back to continuously updating the graphical display of the dive profile. This profile consists of 4 ascent trajectories:

- Fastest possible safe ascent
- stay at current depth for 15 minutes then start fastest safe ascent
- stay at current depth for 30 minutes then start fastest safe ascent
- continue up at the present rate of change.

The system displays the proposed profiles overlaid on top of each other. It will also inform the diver of the time that each ascent profile will take. These calculations take a variable amount of time to complete, depending on the amount of time each ascent takes to complete. No reasonable dive takes more than 5-10 seconds to completely finish calculating, which is sufficiently fast for this application. In a production model, the ascent trajectory calculations would probably be done every minute or so. To enable the calculations to make more efficient use of processor cycles, another ending point could be when the current path would result in the diver running out of air before surfacing. This allows the processor to finish all of the calculations in under a second.

Calculation of remaining air can be done in multiple ways. Currently, it is assumed that the diver is using an Aluminum-80 which is a commonly used diving tank which contains 80 cubic feet of air. A surface air consumption (SAC) value of $0.72 ft^3/min$ was assumed. Both these variables can be easily changed by means of a user

interface or could be measured by a pressure sensor on the tank. The sensor would take an initial pressure measurement, and with knowledge of the tank size it could tell how much air is left. Then taking periodic measurements and calculating the rate of change, the SAC value can be determined and constantly updated. For example, with heavy exertion it is necessary to use more air. The way this dive computer was designed, it would be trivial to add such a feature for production. The measurements could be double checked by using the current method, which is to take the current amount of air and subtract the SAC value multiplied by the current depth.

$$Air_{tank} = Air_{tank} - SAC * (P_{current} / P_{surface}) * t_{delta}$$

It is known that for each atmosphere of pressure the diver is under, the SAC value is proportional to the total pressure. At 33 feet of sea water (FSW) the consumption rate is twice the surface rate, at 66 FSW, it is three times. It is important to realize that the SAC value also depends on how much work the diver is doing. When swimming hard, the value can increase significantly.

A series of screen shots can be seen in Figure 14 demonstrating the system in action over the course of a simulated dive. The screen, since it is a graphical display, can be made to display different pieces of information in different areas.

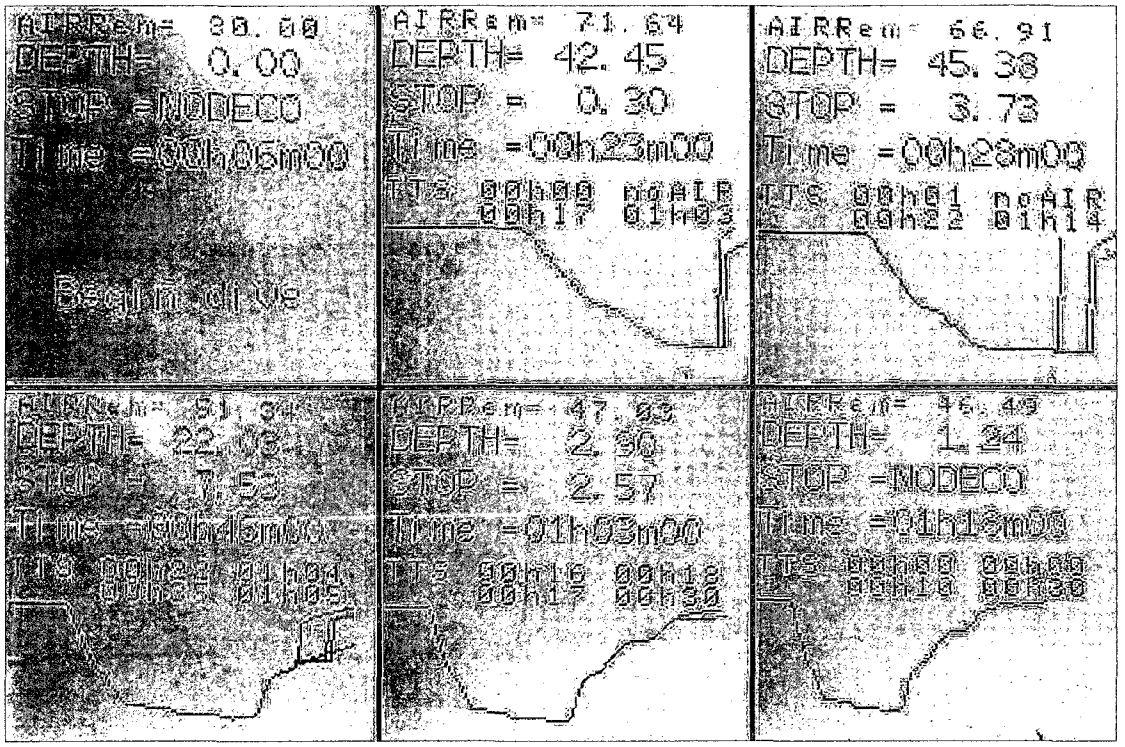

Figure 14: Example of a dive

5.3 Dive profile extraction

Each time the system makes a measurement, if there is an secure digital (SD) card inserted, the system will save each measurement onto the card. The card is stored as a File Allocation Table (FAT) file system so that it can be removed from the dive computer and easily read with any computer. All data is stored as a text file so it makes parsing very simple. This can be used in additional debugging strategies, but more importantly divers typically wish to have this feature. At the time of this writing, a 2GB micro SD card is typical in memory size, and is very inexpensive. This amount of memory will likely be enough for a lifetime of dives, but it gives the ability to have as many things

recorded as possible (e.g.: current values of depth, sensor readings, temperature, time, etc.). The system makes sure not to overwrite previous dive data, and it numbers the files in sequence. A configuration file can be read off the card for changing the behavior of the system. Currently it is used to record the number of dives done.

The code is fast enough to be able to record multiple measurements per second

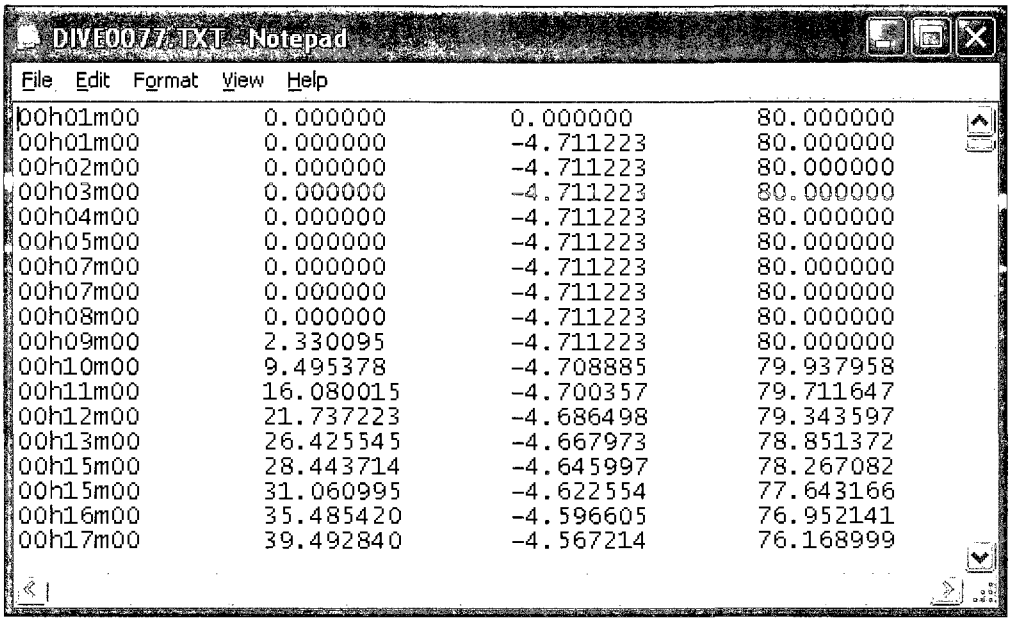

Figure 15: Example contents of a captured dive profile

and not have a noticeable impact on other subsystems. This is obviously overkill, but it gives the opportunity to have more advanced features developed without worrying about speed constraints.

An example of such a data file is shown in Figure 15. The first column represents the time, the second the current depth, the third the tolerated ceiling (negative numbers mean above sea level), and the fourth column is a record of the remaining gas in the tank.

CHAPTER 6

RESULTS

6.1 Capacitive touch

Throughout development the features of the proposed dive computer were tested. After a set of the ARM processor code was completed, the capacitive touch sensors were tested. In principle, the function is simple. Engineers who debug small signal electronics circuits know that if something is not working the way it should, that they put a finger (literally) on the component to see if it fixes the problem. This is because a finger has pico Farads of capacitance. In essence this is how the dive computer operates. Using this principle, a capacitor with capacitance in the pico Farad range can be placed on the circuit board. Then, when the user places a finger on the pad, the capacitance increases by a measurable amount. With an LCR (Inductance Capacitance and Resistance) meter, the pads were measured to see what the actual capacitance was (see Figure 17). Obviously, the dive computer would be used underwater, so the capacitance in wet conditions was also measured (see Figure 16 and Figure 18).

The three conditions done were "No Press" when only the test points were measured, and "Full Press" when the pad was pressed as hard as possible. The task of detecting button presses was accomplished by setting a dynamic threshold value, and if any of the voltage measurements surpassed such a value, the button would be considered pressed. The dynamic threshold was determined by noting that if all three buttons were

registering a pressed state then it is likely that there was a change in the environment which caused the capacitance to go out of range. Being submerged or taken out of water caused such changes.

In testing, it was also discovered that the circuitry was especially prone to 60Hz noise from appliances on the power grid like fluorescent light fixtures. This may not be a

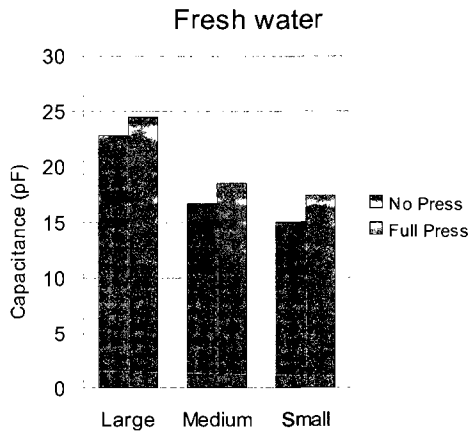

Figure 16: Submerged in fresh water

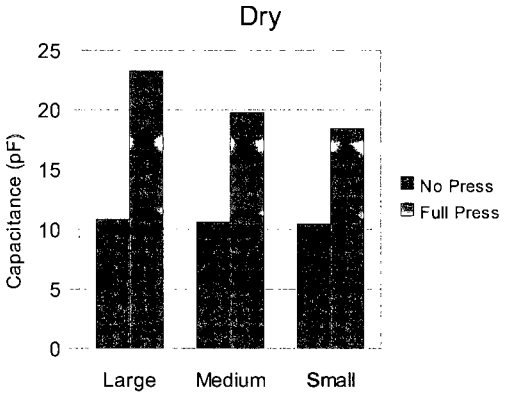

Figure 17: Dry land measurements

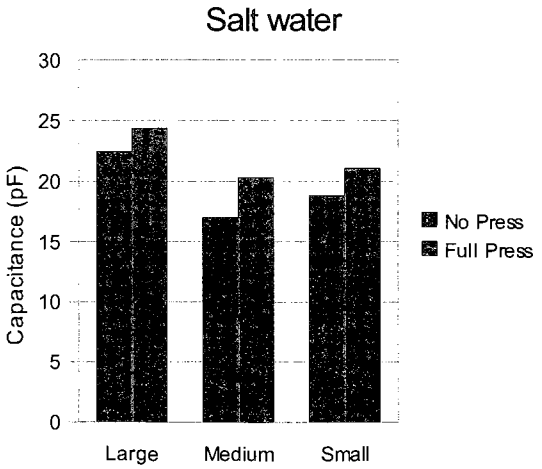

Figure 18: Submerged in salt water

big problem underwater, but on land it may make the controls erroneous. To remedy this problem a software notch filter was implemented. Taking 142 measurements spaced 116.8uS apart gives an interval of about one oscillation of a 60Hz wave. Then each data point is multiplied with an element of a 142 length Hamming window and collected into a sum, which is the internal representation of the capacitance measured.

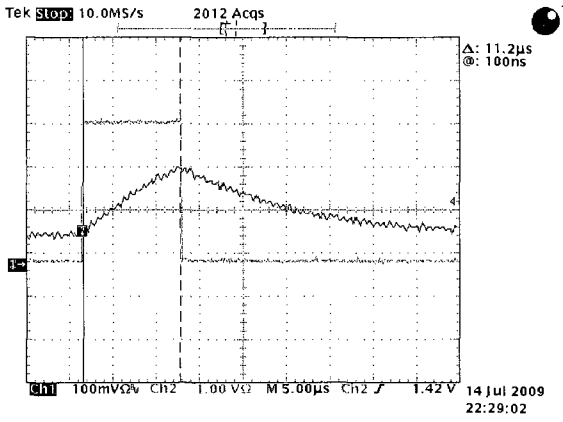

Figure 20: "No press" measurement of signals on a button

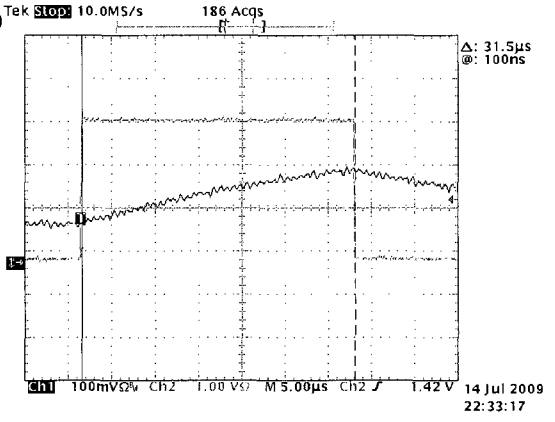

Figure 19: "Full press" measurement of signals on a button

To validate the capacitive touch, measurements on an oscilloscope were taken. Channel 2 was set as the measurement of the signal that was applied to the RC network. Channel 1 was the voltage on the medium sized capacitor. It can be observed (in Figure 20) that the rise time is 11.2 microseconds when the buttons are dry and not being pressed. When a button was being pressed (Figure 19), the rise time was roughly 31 microseconds. This corresponds to the measurements done with the LCR meter.

To know whether a button was pressed, the system uses the running average of the rise time for each button and compares it to the currently measured rise time. Then,

the button that had the greatest deviation is the one that is selected. If all three buttons are past a certain threshold, then the average values are recalculated. This is in the case when the buttons are either taken out or put into a body of water. Using this system has proven fool proof and there are no false positives.

6.2 Input signals

Using a laboratory power supply and a 10:1 voltage divider circuit, the input signals were tested to the dive computer. In final testing a single pressure sensor was simulated. The multiplexer was set to measure the input from a single input, but it is possible to have any combination of the inputs measured, or all of them sampled in sequence. There is working and tested code to accomplish that. The test was accomplished by having a single screen display the values read in at each MUX input.

When the electrical signal simulating the pressure was fed into the pressure input, the dive computer took the value as a pressure and proceeded to use it in the Bühlmann algorithm to calculate the safe ascent profile. To test the algorithm in the microprocessor, a third party Matlab script [11] was run with a certain dive profile (see 9.2), and the same profile was injected into the algorithm using C code (see 9.1). The compartments were each compared, and matched exactly.

Many dive simulations were also performed by setting one of the MUX inputs as a pressure sensor input in code. Then using a laboratory power supply with the 10:1 voltage divider circuit, the Bühlmann code was tested in a near realistic environment. Tests such as rapid changes in depth were undertaken, as a diver could have a machine

propelling him faster than possible to swim. To make these simulations less time consuming, one second in real life was set to be one minute in code. Having such a feature in the software isn't necessarily a bad thing as it could make it possible to have dive plans generated underwater for theoretical dives the diver might want to make.

6.3 Ergonomics of the LCD

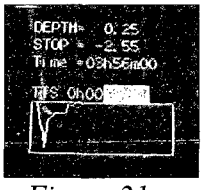

Figure 21: Hardware v1.0 display

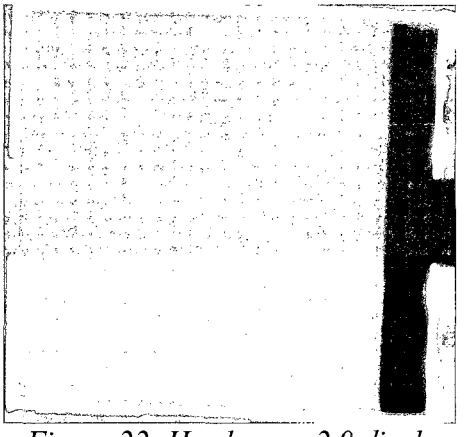

Figure 22: Hardware v2.0 display

Admittedly, small isn't always the best. The original display used was a 128x128 pixel display that measured 1"x1" (Figure 21). It has been replaced with the 2.8"x3" monochrome display (Figure 22) while keeping the resolution constant. Removing the ability to have colors is not a very costly change as it was likely that the original display would have only used black, white, red, green, and cyan. Any other colors would become indistinguishable when diving. Red was used to show danger such as ascending too quickly. In the monochrome display, however, one could just as simply flash the backlight on and off. Readability was greatly increased with the larger display.

CHAPTER 7

CONCLUSION

As I have shown throughout this document, the implementation of a dive computer that can adapt to a diver's swimming profile is feasible. In the diving community it would be well received as it takes less time planning the dive and allows more concentration on the interesting parts of being under water (e.g.: exploration). It would also decrease the steepness of the learning curve associated with diving and calculating from dive tables.

The new system of user inputs would also make the device more robust and easier to manufacture, thus reducing cost in production and later cost to the end user. The capacitive touch has been widely explored in above-water applications, but not as a sub-surface interface.

This prototype dive computer has been designed with low power consumption in mind. The processor needs to run only for short bursts of time. The display was selected to have minimal power requirements with the most visibility in all environments for a dive. The processor could cut power to parts of the system when not in use, guaranteeing minimal power consumption.

This platform also has many expandability options. One of these is the ability to add in more sensors. In the case of a CCR one would want a redundant set of oxygen sensors, say three, and employ majority voting algorithms to take the two that are most

closely matched to each other. If the third sensor is reading measurements that are unreasonable, a warning can be flagged by the processor that it may be time to replace the sensors. Same thing can be done with the depth gauge.

The solenoid actuator circuit is also available for controlling the CCR actuator. Though not necessary for open circuit divers, this feature is there to support a single platform for almost all cases.

A commonly desired feature by divers is connectivity with a PC to analyze statistics about the dives that have been conducted. This system offers that feature by having a common microSD card store the data gathered in a file system that is commonly used in today's operating systems. Since data is stored in a clear text format, parsing can be accomplished easily by any programming language or software package.

CHAPTER 8

FUTURE WORK

8.1 Low power modes

In the future the processor's low power modes should be explored more. As it stands, the processor is almost always doing something, but if the display updates are made to occur less frequently then the processor could shut down and consume less power. Right now the execution of the Bühlmann trajectories takes up much of the processing time. This means that if the update rate were to be reduced, the processor could be in standby mode for a very long time. The software is written so that the display can have other data updated independently of the graphs. Other algorithms may be considered as well to decrease CPU usage.

8.2 Magnetic compass

Implementing the magnetic compass and fully debugging it would add more valuable features to this device. This was outside the scope of this research. All the hardware is available on the circuit board. The necessary steps to get it working are testing the power up sequence of the compass and understanding the readings out of the device.

8.3 User interface

Now that there are working non-mechanical buttons that can reliably work underwater, it would be ideal to have them perform some action other than lighting up LEDs. This is a task for someone who is going to implement this system as a final product that will be sold to consumers. More importantly it would be useful to try to get the capacitive touch buttons built up as a flex circuit board to see if it makes a difference. Most likely it will not, but the advantage of flex circuit is that in manufacturing the final product, it will make it easier to get the buttons out to a non-water-resistant area with a flex connector.

8.4 Capacitive touch

The capacitive touch buttons should be tested with different gloves that divers may use to see how well the system responds. Allowing changes to the sensitivity of the button would make the device be more robust under a wider range of conditions.

8.5 Entertainment features

Since the dive computer already has an SD card connector on it, the ability is there to store books as text files which can be read by the diver when there is really nothing else to do while surfacing. The platform proposed here already integrates a large graphical display, which would make it truly ideal for finishing up that book while out-gassing. It is also possible to get underwater MP3 players, so it should be possible to integrate one into the dive computer easily. The processor selected is perfectly capable to decode MP3 files from an SD card and send them to an output.

8.6 Communication

Another important feature that could be added to a dive computer is the ability to send short messages while diving. This can be automatic, for example the computer periodically transmitting its designated ID number, and where it physically is depth-wise, along with readings from the compass. It could also be made intelligent enough to notice changes in air consumption indicating distress. Alternatively, it could be also used as a way of informing the boat crew of status in terms of findings and plans of action.

8.7 Navigation

With the addition of an accelerometer, a crude underwater positioning system can be implemented on this platform. Maps can be stored on the SD card and displayed on the screen. As a start, a trace of the path traveled from a birds eye view can help one get back to the boat. Such a feature will be priceless to cave divers.

BIBLIOGRAPHY

1: National Oceanic and Atmospheric Administration, NOAA Diving Manual: Diving for Science and Technology, Best Publishing Company, 2001, ISBN:0-941332-70-5

2: A. A. Bühlmann, Tauchmedizin, Berlin: Springer-Verlag, 1995, ISBN:3540555811

3: Chris Acott, A BRIEF HISTORY OF DIVING AND DECOMPRESSION ILLNESS, SPUMS Journal, 1999, ISBN:0813-1988

4: Searle, WF, Jr, Foxboro Decomputer Mark I., PANAMA CITY FL: NAVY EXPERIMENTAL DIVING UNIT, 1957

5: Stubbs, RA and Kidd, DJ, A pneumatic analogue decompression computer, Canadian Institute of Aviation, 1965

6: Bruce R. Wienke and Timothy R. O'Leary, REDUCED GRADIENT BUBBLE MODEL: DIVING ALGORITHM, BASIS, AND COMPARISONS, Tampa, Florida: NAUI Technical Diving Operations,

7: Corrado Bonuccelli, Calculating Deco schedule with VPM, 2009, http://www.hhssoftware.com/images/vpm_expl.pdf

8: Gerth, WA; Ruterbusch, VL; Long, ET, The Influence of Thermal Exposure on Diver Susceptibility to Decompression Sickness, PANAMA CITY FLA: NAVY EXPERIMENTAL DIVING UNIT, 2007

9: Paul Chapman, An Explanation of Professor A.A. Buehlmann's ZH-L16 Algorithm, 2008, http://njscuba.net/gear/trng_10_deco.html

10: Satish Mishra, Visual Modeling & Unified Modeling Language (UML), 1997

11: Vlad Pambucol, Dive Computers, 2002,
http://www.pambucol.com/scuba/dive_computers_final_presentation_html/dive_computers_main.htm

CHAPTER 9

APPENDIX

9.1 Bühlmann code in C

```
// Buehlmann.c
//=============================================================
// Arithmetic stuff for calculation the buehlmann model
#include <math.h>
#include <string.h>
#include "buehlmann.h"

diveData_t myDive;

float buehlmann_compart_model[16][3] = {
//p.a.tol p.i.g.  p.i.g
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
```

```
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0},
        {0.0, 0.0, 0.0}};

const float buehlmann_compart_const[16][8] = {
// Nt1/2        Na              Nb      Ht1/2  Ha              Hb
        {4.0,   1.2599,         0.5050,         1.5,    1.7435,         0.1911},
        {8.0,   1.0000,         0.6514,         3.0,    1.3838,         0.4295},
        {12.5,  0.8618,         0.7222,         4.7,    1.1925,         0.5446},
        {18.5,  0.7562,         0.7725,         7.0,    1.0465,         0.6265},
        {27.0,  0.6667,         0.8125,         10.2,   0.9226,         0.6917},
        {38.3,  0.5933,         0.8434,         14.5,   0.8211,         0.7420},
        {54.3,  0.5282,         0.8693,         20.5,   0.7309,         0.7841},
        {77.0,  0.4701,         0.8910,         29.1,   0.6506,         0.8195},
        {109.0,0.4187,          0.9092,         41.1,   0.5794,         0.8491},
        {146.0,0.3798,          0.9222,         55.1,   0.5256,         0.8703},
        {187.0,0.3497,          0.9319,         70.6,   0.4840,         0.8860},
        {239.0,0.3223,          0.9403,         90.2,   0.4460,         0.8997},
        {305.0,0.2971,          0.9477,         115.1,  0.4112,         0.9118},
        {390.0,0.2737,          0.9544,         147.2,  0.3788,         0.9226},
        {498.0,0.2523,          0.9602,         187.9,  0.3492,         0.9321},
        {635.0,0.2327,          0.9653,         239.6,  0.3220,         0.9404}};

void init_diveData_t(diveData_t *dive, long curTime){
        int i;

        // my_dive members init
        dive->deco      = 0;
        dive->depth     = 0.0;
```

```
        dive->depth_old   = 0.0;
        dive->lastTime    = curTime;
        dive->p_amb       = 0.0;
        dive->pp_n2       = 0.0;
        dive->pp_o2       = 0.0;
        dive->pp_he       = 0.0;
        dive->perc_n2     = 0.78084;
        dive->perc_o2     = 0.209476;
        dive->perc_he     = 0.00000524;
        dive->te_min      = 999.0;
        dive->depth_max   = 0.0;
        dive->depth_tol_min = 0.0;

        memcpy(dive->compart, buehlmann_compart_model,
sizeof(buehlmann_compart_model));

        // calculate pressure
        dive->p_amb = P_AMB_AIR + dive->depth;
        dive->pp_n2 = dive->perc_n2 * (dive->p_amb - P_STEAM);
        dive->pp_o2 = dive->perc_o2 * (dive->p_amb);
        dive->pp_he = dive->perc_he * (dive->p_amb - P_STEAM);

        // inert gas pressure in balance, t0=te in all tissues
        for (i=0; i<16; i++) {
                dive->compart[i][M_PIG_N2] = dive->pp_n2;
                dive->compart[i][M_PIG_HE] = dive->pp_he;
        }
}

void calculate_tissues(diveData_t *dive, long curTime) {
```

```
unsigned char i;
float *curCompart, acor, bcor;
const float *curCompartConst;

// calculate pressure
dive->p_amb = P_AMB_AIR + dive->depth;
dive->pp_n2 = dive->perc_n2 * (dive->p_amb - P_STEAM);
dive->pp_o2 = dive->perc_o2 * (dive->p_amb);
dive->pp_he = dive->perc_he * (dive->p_amb - P_STEAM);

dive->depth_old = dive->depth;

if(!dive->deco) dive->te_min = 999.0;
dive->pp_amb_tol_max = 0.0;
dive->def_cmpt = 0;

// store max. depth
if (dive->depth > dive->depth_max) dive->depth_max = dive->depth;

float deltaTime = ((float) (dive->lastTime - curTime))/60;
for (i=0; i<16; i++) {
        curCompart = dive->compart[i];
        curCompartConst = buehlmann_compart_const[i];
        // tissue saturation
        // pt. i.g. (tE) = pt. i.g. (t0) + [pI i.g. - pt. i.g. (t0)]*[1 - 2^(-tE / t1/2)]

        curCompart[M_PIG_N2] = curCompart[M_PIG_N2] + (dive->pp_n2 -
curCompart[M_PIG_N2]) * (1-pow(2,deltaTime/curCompartConst[C_N_T1_2]));
        curCompart[M_PIG_HE] = curCompart[M_PIG_HE] + (dive->pp_he -
curCompart[M_PIG_HE]) * (1-pow(2,deltaTime/curCompartConst[C_H_T1_2]));
```

```
                // tolerated ambient pressure
                // pamb. tol. = (pt. i.g. - a) * b
                acor = (curCompartConst[C_N_A]*curCompart[M_PIG_N2] +
curCompartConst[C_H_A]*curCompart[M_PIG_HE]) / (curCompart[M_PIG_N2] +
curCompart[M_PIG_HE]);
                bcor = (curCompartConst[C_N_B]*curCompart[M_PIG_N2] +
curCompartConst[C_H_B]*curCompart[M_PIG_HE]) / (curCompart[M_PIG_N2] +
curCompart[M_PIG_HE]);
                curCompart[M_PAMBTOL] = (curCompart[M_PIG_N2] +
curCompart[M_PIG_HE] - acor) * bcor;

                if (curCompart[M_PAMBTOL] > dive->pp_amb_tol_max) {
                        dive->pp_amb_tol_max = curCompart[M_PAMBTOL];
                        dive->def_cmpt = i + 1;
                }
        }

        // minimum tolerated depth
        dive->depth_tol_min = dive->pp_amb_tol_max - P_AMB_AIR;

        // store time calculation was completed
        dive->lastTime = curTime;
}

//Buehlmann.h
//=====================================================================
#ifndef BUEHLMANN_H
#define BUEHLMANN_H

#include <stdlib.h>
```

```
//#define BUEHL_ACCEL_TIME
#define WITH_STEAM_PRESSURE
#ifndef _Bool
#define _Bool
typedef enum { false = 0, true = 1} bool;
#define false false
#define true true
#endif

typedef struct {
 float depth;        // depth
 float depth_old;    // depth(t-dt)
 float depth_max;    // maximum depth
 long  lastTime;     // time last measurements were calculated
 float p_amb_air;    // air pressure
 float p_amb;        // ambient pressure
 float pp_n2;        // inert gas pressure
 float pp_he;
 float pp_o2;
 float perc_n2;
 float perc_o2;
 float perc_he;
 float te_min;       // smallest no decompression limit
 float pp_amb_tol_max; // max tol. ambient pressure
 float depth_tol_min;  // min tol. depth
 int   def_cmpt;     // defining compartment

 float compart[16][3]; // compart model array
 bool  deco;         // indicates if this is a deco dive
```

```
                // in terms of pp_amb_tol_max > p_amb_air
} diveData_t;

#define C_N_T1_2    0
#define C_N_A       1
#define C_N_B       2
#define C_H_T1_2    3
#define C_H_A       4
#define C_H_B       5
#define M_PAMBTOL   0
#define M_PIG_N2    1
#define M_PIG_HE    2
#define M_TE_N2     3
#define M_TE_HE     4

#define P_AMB_AIR 1.01325
#ifdef WITH_STEAM_PRESSURE
#define P_STEAM 0.0567
#else
#define P_STEAM 0.0
#endif

extern diveData_t myDive;

#define calculate_eanx_best(ppo2_max,  depth) { ppo2_max / (depth/10 + 1) * 100 }
#define calculate_ead(eanx, depth) { ((1-(eanx/100))*(depth+10)/0.79 ) - 10 }
#define calculate_mod(eanx, ppo2_max) { ((1000 * ppo2_max) / eanx) - 10 }

// function prototypes
```

```
void init_diveData_t(diveData_t *dive, long curTime);
void calculate_tissues(diveData_t *dive, long curTime);
void calculate_blend(float eanx_from, float start_pressure, float eanx_to, float
end_pressure, float *oxygen, float *topping);

#endif

//Main.c
//=============================================================================
#include <stdio.h>
#include "buehlmann.h"

#define NUMCOMPART 5
void printCompart(diveData_t *dive) {
        int i,j;
        printf("P_AMB:    %f\n",dive->p_amb);
        printf("COMPARTNM:");
        for(j = 0; j < NUMCOMPART; j++) printf(" %f",(double)j);
        printf("\n");
        printf("P_IG_N2:  ");
        for(j = 0; j < NUMCOMPART; j++) printf(" %f",dive->compart[j][M_PIG_N2]);
        printf("\n");
        printf("P_IG_HE:  ");
        for(j = 0; j < NUMCOMPART; j++) printf(" %f",dive->compart[j][M_PIG_HE]);
        printf("\n");
        printf("P_AMB_TOL:");
        for(j = 0; j < NUMCOMPART; j++) printf(" %f",dive->compart[j]
[M_PAMBTOL]);
        printf("\n");
}
```

```
#define TIMEINTERVAL 0.5
int main() {
        double diveDepths[] = {1.01325, 2.0265, 3.03975, 4.053, 5.06625, 6.0795,
7.09275, 8.106, 9.11925, 10.1325, 11.14575, 11.14575, 11.14575, 10.1325, 9.11925,
8.106, 7.09275, 7.09275, 6.0795, 5.06625, 4.053, 3.03975, 2.0265, 1.01325};
        int i;
        init_diveData_t(&myDive,0);
        printCompart(&myDive);

        for(i = 1; i < 24; i++) {
                myDive.depth = diveDepths[i] - P_AMB_AIR;
                calculate_tissues(&myDive,60*TIMEINTERVAL*i);
                printCompart(&myDive);
        }
        return 0;
}
```

9.2 Bühlmann code in Matlab

```
%Buhlmann ZH-L16 algorithm
%by Emiliano Rial Verde
%emiliano(at)rialverde(dot)com
close all; clear all; clc;

%ALL PRESSURES ARE IN BAR!!!
%1.01325 bar = 1 atm = 33 fsw = 10msw = 101325 Pa

pa=1.01325; % Atmosferic pressure at sea level (default). Change if diving at altitude!
pwl=0.0567; %Water vapour pressure in the lungs to correct the partial pressures
```

```
%Initial variables (a simulated dive with pressure readings taken every time interval)
timeinterval=0.5; %Time interval at wich the pressure was measured in minutes.
depth=[pa:pa:pa*11 pa*11 pa*11:-pa:pa*7 pa*7:-pa:pa] %Pressure measurements at the
indicated time intervals.
disp(mat2str(depth))
%Air composition at 15 Degrees C, at 101325 Pa of pressure (sea level).
%This is the default
%Modify this composition if breathing a mixture other than air!
N2=0.78084;
O2=0.209476;
%Ar=0.00934;
%CO2=0.000314;
%Ne=0.00001818;
%CH4=0.000002;
He=0.00000524;
%Kr=0.00000114;
%H2=0.0000005;
%Xe=0.000000087;

%Pressure of gasses at starting altitude
%Only nitrogen and helium are considered.
paN2=pa*N2; %Presure of nitrogen.
ppN2=(pa-pwl)*N2; %Partial pressure of nitrogen in the lungs.

paHe=pa*He; %Presure of helium.
ppHe=(pa-pwl)*He; %Partial pressure of helim in the lungs.


%The 16 tissues from Buhlmann's algorithm
```

```
%Column 1 is nitrogen half time, followed by a and b values
N2half=1;
N2a=2;
N2b=3;
%Column 4 is helium half time, followed by a and b values
Hehalf=4;
Hea=5;
Heb=6;

tissues=[4 1.2599 0.5050 1.5 1.7435 0.1911;
8 1.0000 0.6514 3.0 1.3838 0.4295;
12.5 0.8618 0.7222 4.7 1.1925 0.5446;
18.5 0.7562 0.7725 7.0 1.0465 0.6265;
27 0.6667 0.8125 10.2 0.9226 0.6917;
38.3 0.5933 0.8434 14.5 0.8211 0.7420;
54.3 0.5282 0.8693 20.5 0.7309 0.7841;
77 0.4701 0.8910 29.1 0.6506 0.8195;
109 0.4187 0.9092 41.1 0.5794 0.8491;
146 0.3798 0.9222 55.1 0.5256 0.8703;
187 0.3497 0.9319 70.6 0.4840 0.8860;
239 0.3223 0.9403 90.2 0.4460 0.8997;
305 0.2971 0.9477 115.1 0.4112 0.9118;
390 0.2737 0.9544 147.2 0.3788 0.9226;
498 0.2523 0.9602 187.9 0.3492 0.9321;
635 0.2327 0.9653 239.6 0.3220 0.9404];

%The dive is calculated in a repetitive fashion at short intervals
%Therefore the instantaneous equation is used instead of Shreiner's
%See Dive Computers by Vlad Pambucol at:
```

```
%http://www.aut.utt.ro/~vlad/scuba/dive_computers_final_presentation_html/dive_computers_final_presentation.pdf

%Nitrogen and Helium dynamics throughout the dive defined in the variable "depth"
ppN2dyn=zeros(size(tissues,1), max(size(depth)));
ppHedyn=zeros(size(tissues,1), max(size(depth)));
ceiling=zeros(size(tissues,1), max(size(depth)));
nodecotime=zeros(1, max(size(depth)));
decostops=zeros(1, max(size(depth)));
ppN2dyn(:,1)=repmat(ppN2, size(tissues,1),1); %Partial pressure of nitrogen at the begining of the dive for every tissue.
ppHedyn(:,1)=repmat(ppHe, size(tissues,1),1); %Partial pressure of helium at the begining of the dive for every tissue.
nodecotime(1)=NaN;
disp(['Depth:     ' num2str(0)])
disp(['COMPARTMNT: ' mat2str([0:4])])
disp(['P_IG_N2:    ' mat2str(ppN2dyn(1:5,1))])
disp(['P_IG_HE:    ' mat2str(ppHedyn(1:5,1))])
disp(['P_AMB_TOL:  ' mat2str(ceiling(1:5,1))])
for i=2:max(size(depth))
  ppN2dyn(:,i)=ppN2dyn(:,i-1)+((depth(i)-pwl).*N2-ppN2dyn(:,i-1)).*(1-2.^(-timeinterval./tissues(:,N2half)));
  ppHedyn(:,i)=ppHedyn(:,i-1)+((depth(i)-pwl).*He-ppHedyn(:,i-1)).*(1-2.^(-timeinterval./tissues(:,Hehalf)));


      %Ascent ceiling (Decompression stop) calculation
  acor=(tissues(:,N2a).*ppN2dyn(:,i)+tissues(:,Hea).*ppHedyn(:,i))./(ppN2dyn(:,i)+ppHedyn(:,i));
  bcor=(tissues(:,N2b).*ppN2dyn(:,i)+tissues(:,Heb).*ppHedyn(:,i))./(ppN2dyn(:,i)+ppHedyn(:,i));
  ceiling(:,i)=((ppN2dyn(:,i)+ppHedyn(:,i))-acor).*bcor;
```

```
        disp(['Depth:      ' num2str(depth(i))])
        disp(['COMPARTMNT: ' mat2str([0:4])])
        disp(['P_IG_N2:    ' mat2str(ppN2dyn(1:5,i))])
        disp(['P_IG_HE:    ' mat2str(ppHedyn(1:5,i))])
        disp(['ACOR:       ' mat2str(acor(1:5))])
        disp(['BCOR:       ' mat2str(bcor(1:5))])
        disp(['P_AMB_TOL:  ' mat2str(ceiling(1:5,i))])

end
```

9.3 Hardware schematics v1.0

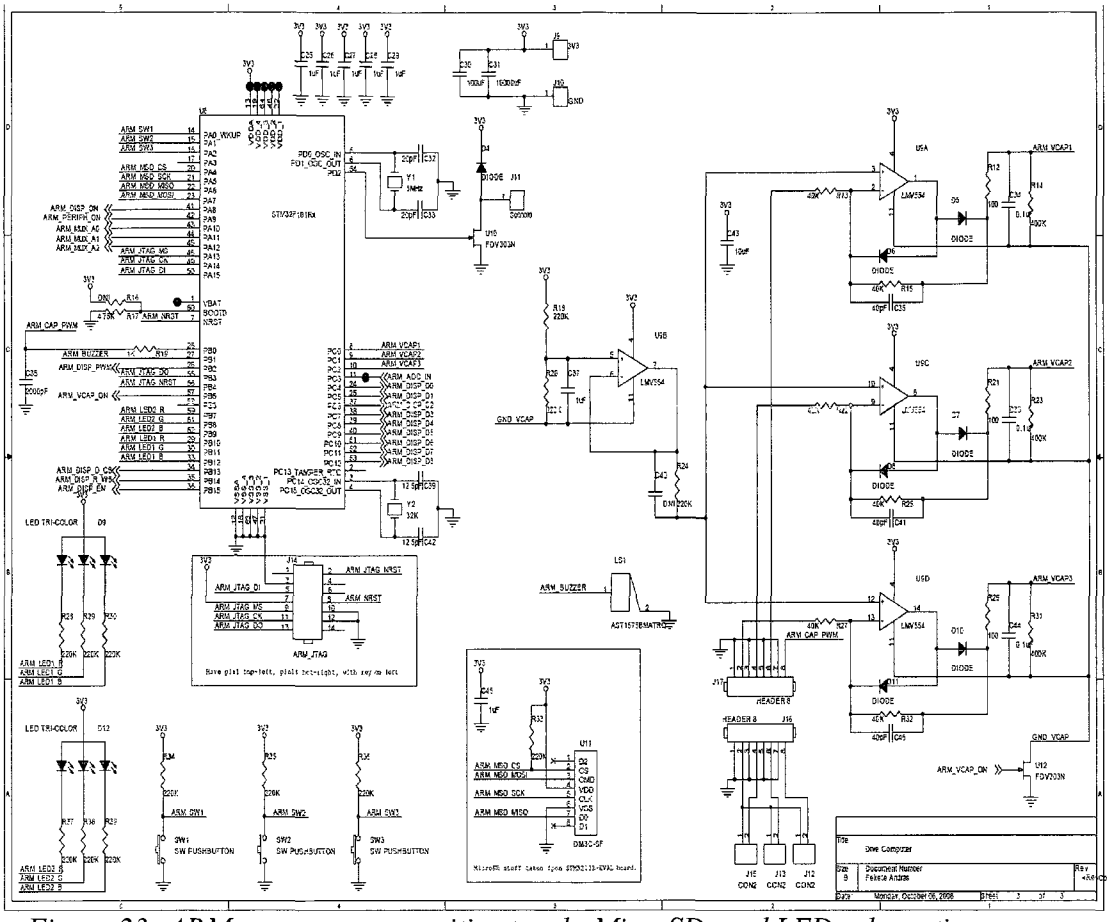

Figure 23: ARM processor, capacitive touch, MicroSD, and LED schematics

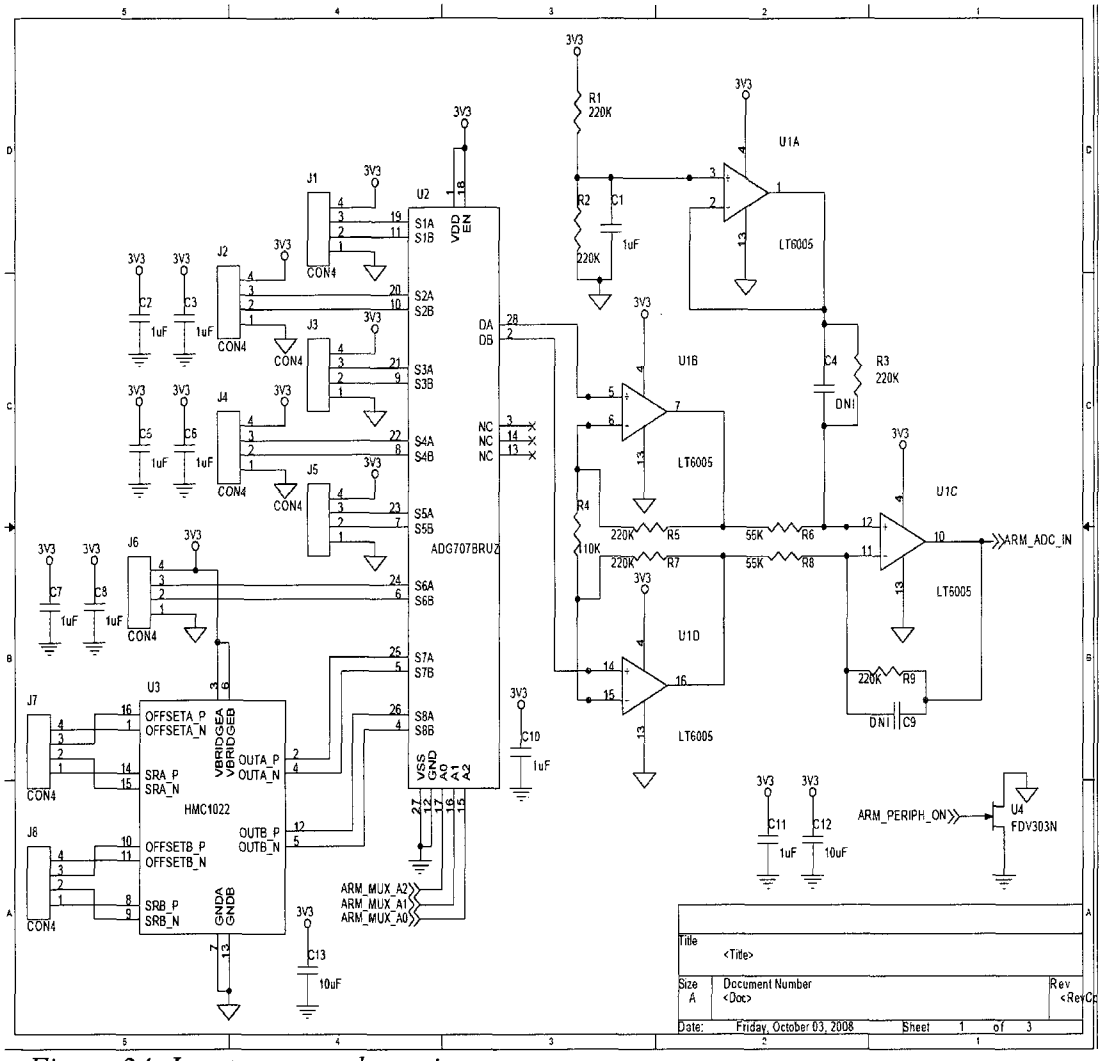

Figure 24: Input sensor schematics

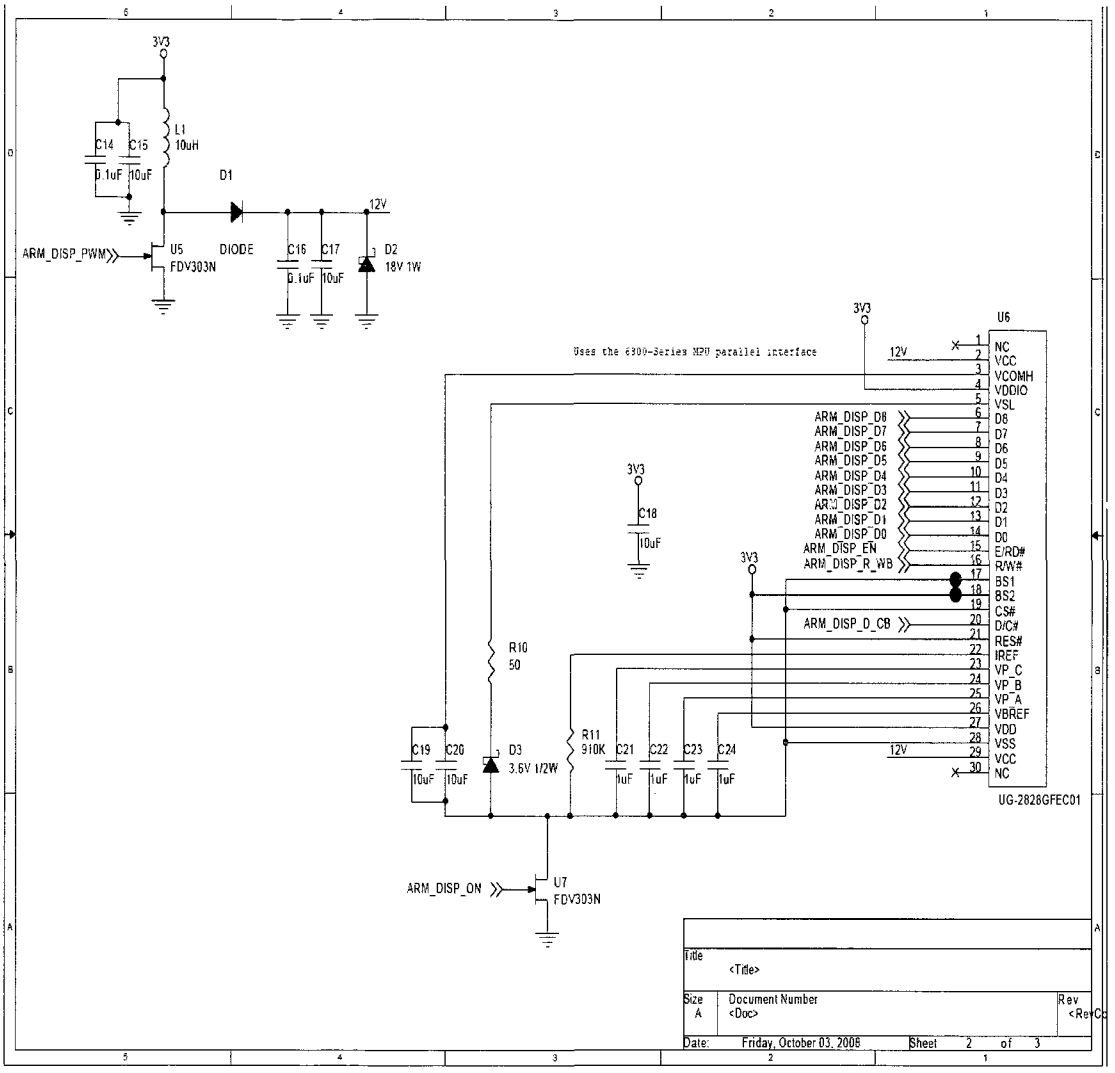

Figure 25: Display schematics

9.4 Hardware layout v1.0

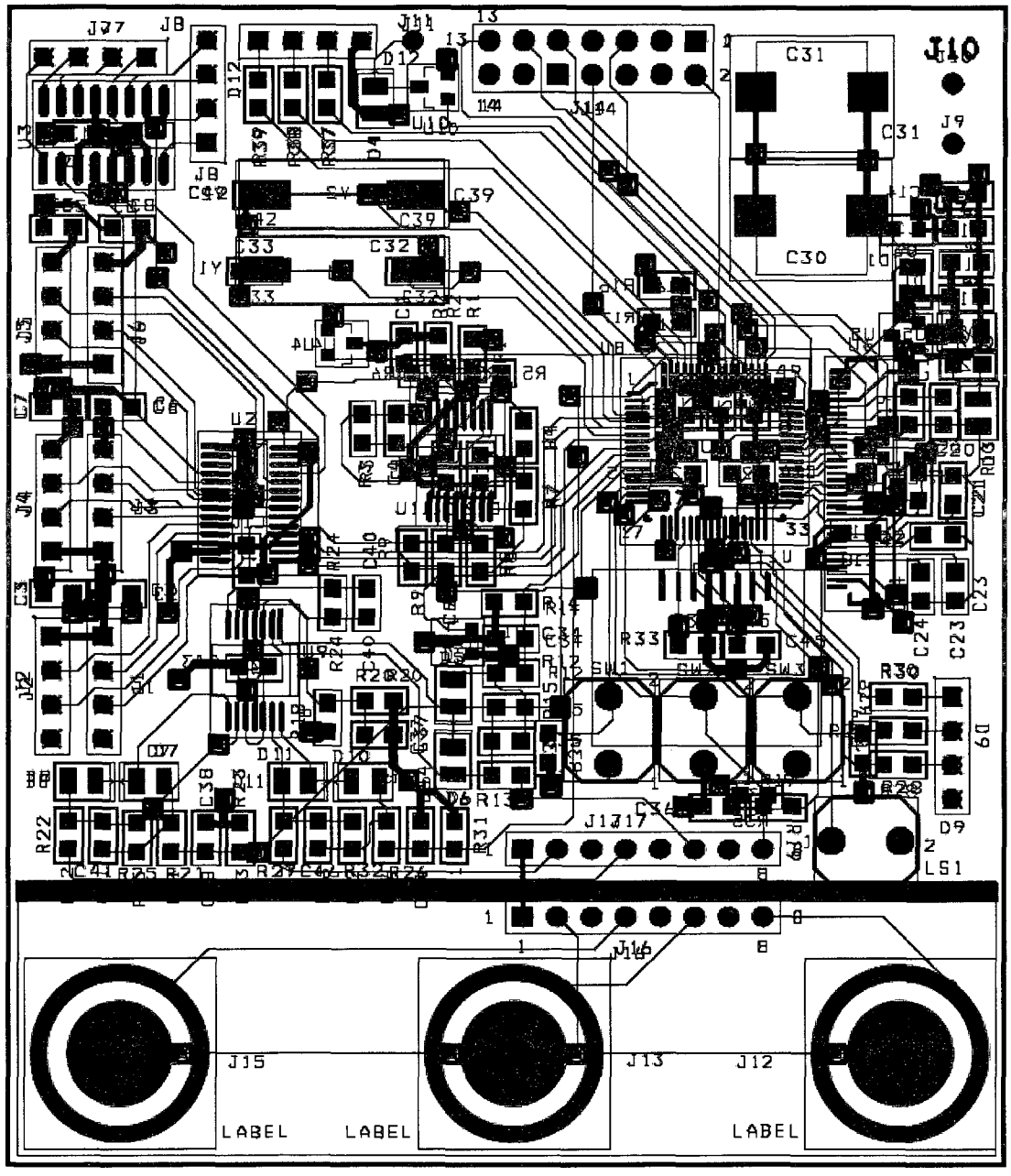

Figure 26: Development board layout

9.5 Hardware Schematics v2.0

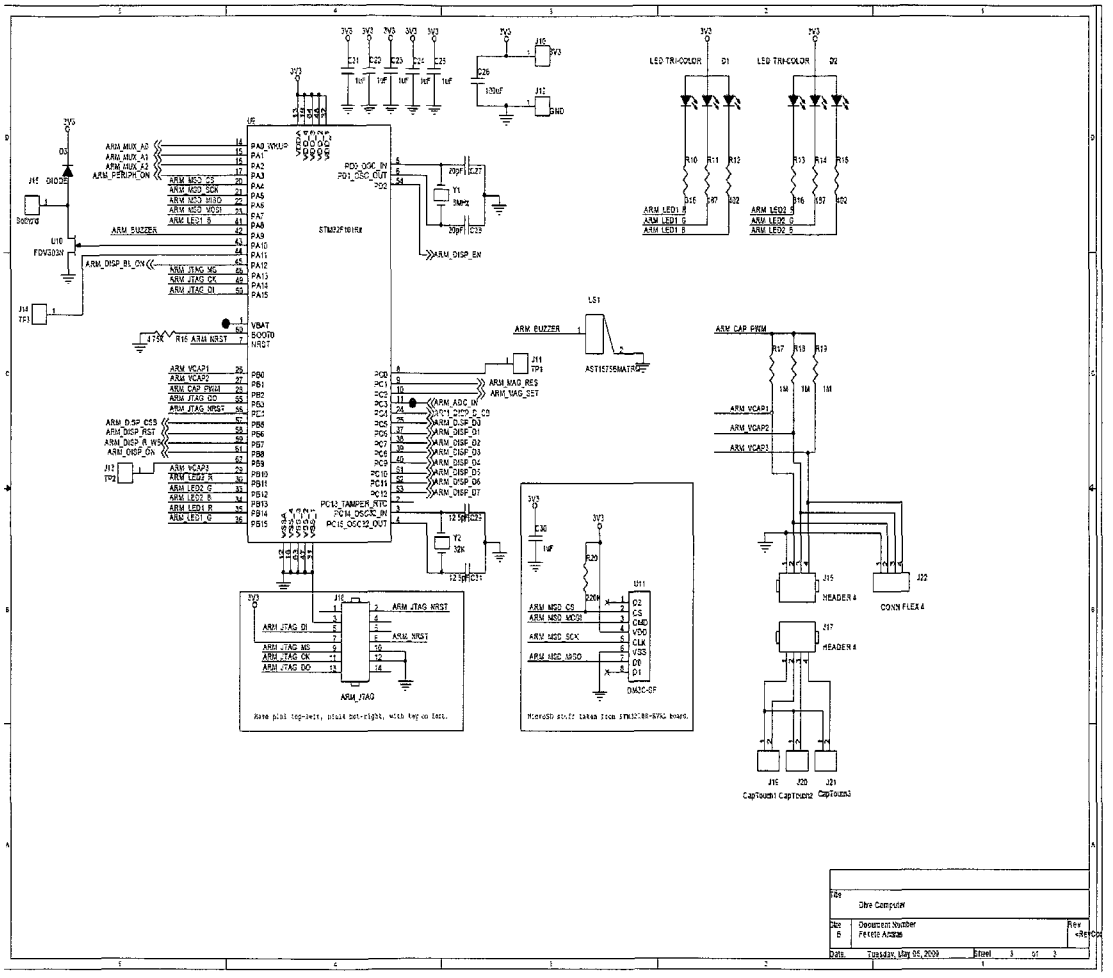

Figure 27: ARM, SD Card, Cap. touch, and other circuits

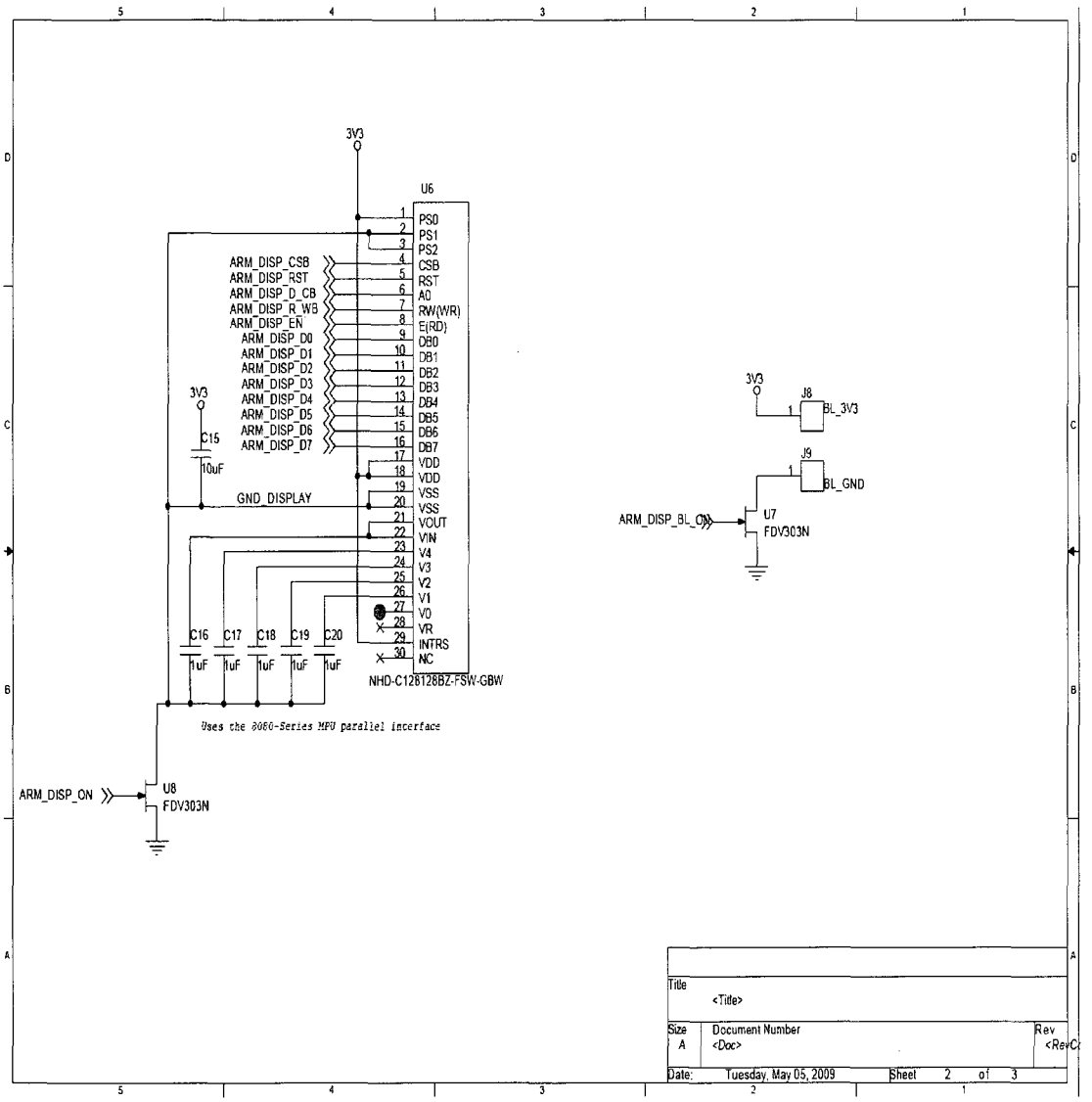

Figure 28: Monochrome LCD display

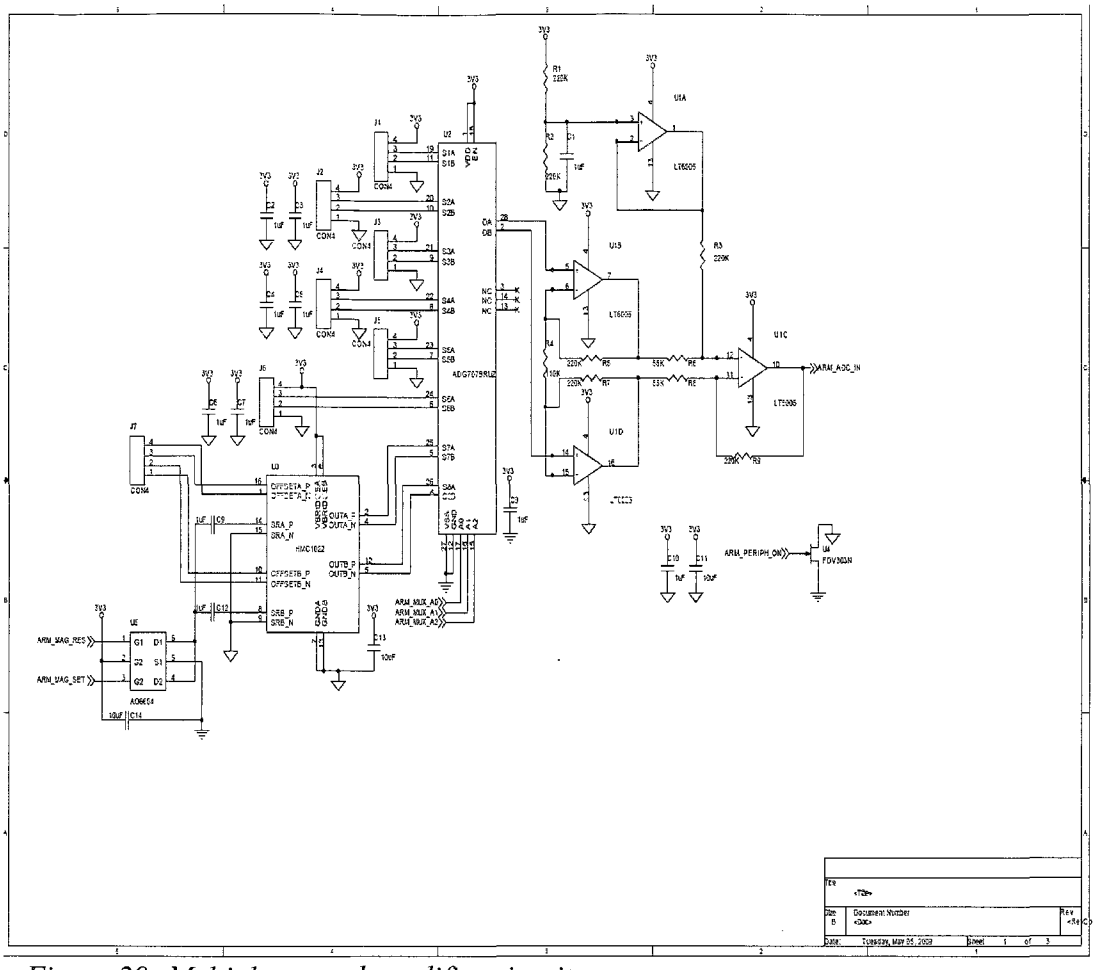

Figure 29: Multiplexer and amplifier circuitry

9.6 Hardware Layout v2.0

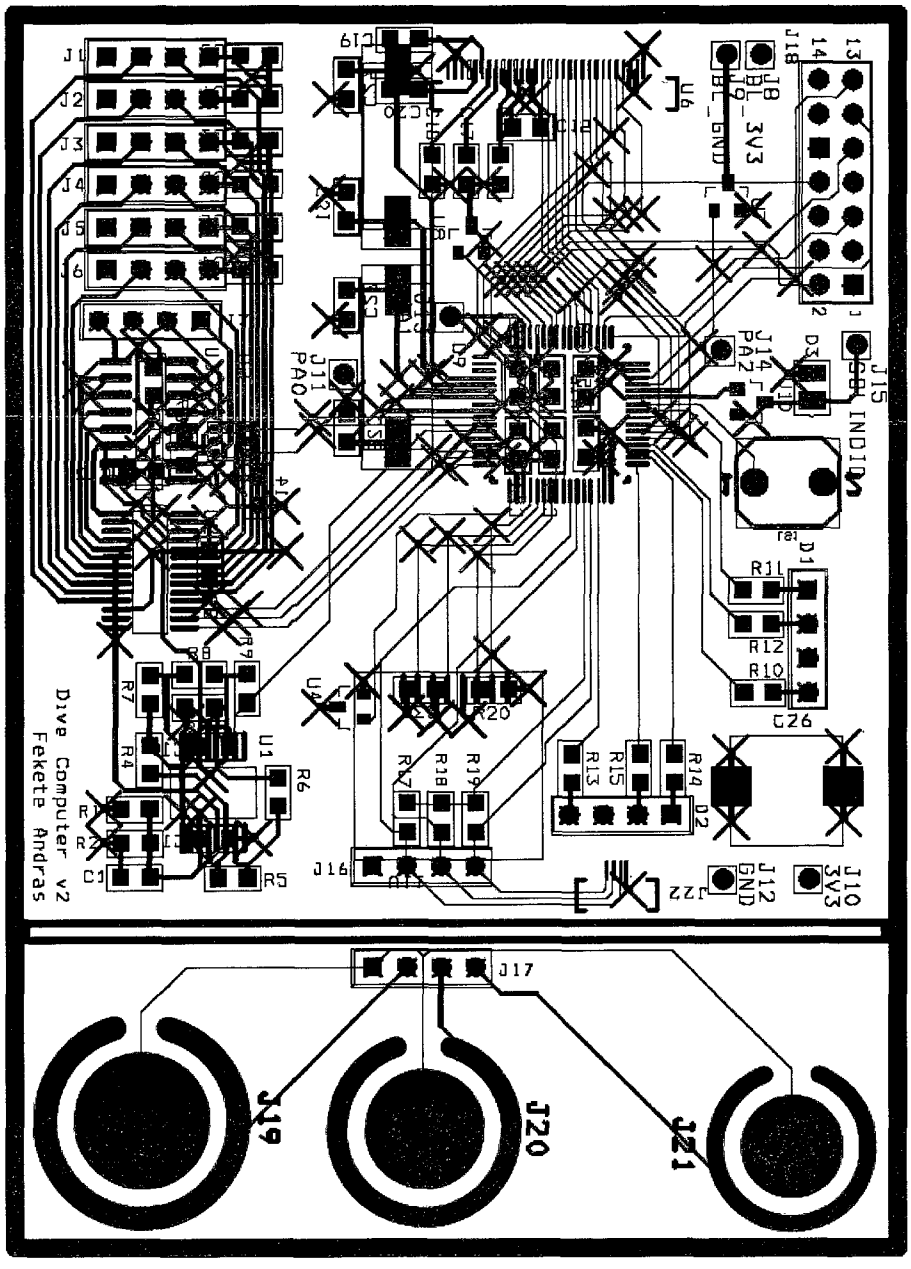

Figure 30: Second development board layout.