2013 National microMedic Contest Report

Team Name: Bauch & Allen

Team Members: Dr. Terry Bauch and Raymond Allen, Ph.D.

Submitted 31 July 2013

Table of Contents

Title Page .... **i**

Table of Contents .... **ii**

Contest Submission Enclosure List .... **iii**

List of Figures .... **iv**

Forward **1**

Introduction **1**

Medical Overview **2**

Electronics Overview **2**

SpO2 Sensor Circuit Details **5**

Nellcor SpO2 Sensors 5

Pulse Oximetry 6

SpO2 Circuit Description 6

Bridge Circuit 8

Transimpedance Amplifier 9

Red and IR Amplifier 9

Vdd/2 Buffer 11

Red/IR Switch 11

Analog-to-Digital Converter 12

Pressure Sensor Circuit Details **12**

Pressure Sensor 12

Instrumentation Amplifier 13

Pressure Amplifier and Filter Circuit 14

Pressure Sensor Calibration 15

Programming Overview **15**

Programming Details **16**

SpO2 Control and Data Acquisition Program 17

Baseline Control 18

Data Analysis and Display 18

LCD Display Driver Details 19

LCD Graphics 19

Data Analysis and Display Loop 20

Computer Program Description 21

Prototype Assembly **22**

Prototype Testing **23**

Conclusion and Future Work **24**

Appendix 1. Propeller Pin Usage **25**

Contest Submission Enclosure List

1. **Project Report (This document).**
2. **Source Code (complete source code is provided for both the Parallax Propeller microcontroller and the Windows C++ computer interface).**
3. **3D Cad models of 3D-printed enclosure.**
4. **Schematics (hardware schematics for all circuits is provided).**
5. **Pictures (several pictures of Terry Bauch and Raymond Allen with hardware are provided).**
6. **Videos (A public video of our prototype device in action is available on YouTube here:** http://www.youtube.com/watch?v=Dj23i4MWAHQ)

List of Figures

Figure 1. Photo of the prototype VTD during initial testing of the SpO2 circuit and program. **1**

Figure 2. Diagram of the main PCB boards and components in the VTD device. **3**

Figure 3. Photo of the Propeller Platform Express (PPE) board with components identified. **4**

Figure. 4. The microMedic shield comprising SpO2, EKG and pressure circuits. **4**

Figure. 5. An interface, “breakout” board to connect a 4.3” resistive touchscreen to the Propeller microcontroller. **5**

Figure. 6. Two SpO2 sensors used during VTD development, a disposable unit from a hospital (left) and a clip-on reusable clone from EBay (right). **6**

Figure. 7. Circuit block diagram for SpO2 measurement, showing how the four video switches are used to reduce the number of I/O pins and ADC channels required for SpO2 measurement. **7**

Figure. 8. Bridge circuits for the red and IR LEDs, which are connected in parallel as shown in the inset. **8**

Figure. 9. Transimpedance amplifier circuit with photodiode connection shown in the inset. **9**

Figure. 10. Red amplifier circuit (IR amplifier is identical). **10**

Figure. 11. Voltage divider and buffer amp to provide a Vdd/2 power supply. **11**

Figure. 12. Video switch used to switch between red and IR modes. **11**

Figure. 13. Four channel ADC circuit with SPI connection to MCU. **12**

Figure. 14. Pressure sensor circuit board from disposable pressure sensor used in a hospital (left), wire connection pad layout from 1620 datasheet (center) and equivalent circuit of the sensor (right). **13**

Figure 15. Pressure sensor instrument amplifier (U6) circuit with offset amp (U7) and a non-inverting amplifier (U9). **14**

Figure 16. Pressure filter and amplifier circuit. **15**

Figure 17. Usage of the eight Propeller cores in the VTD prototype. **16**

Figure 18. Block diagram of the Propeller program for the VTD showing the two main Spin codes (blue) with associated Spin drivers (green) and assembly drivers (peach). **16**

Figure 19. SpO2 control and data acquisition program flowchart. **17**

Figure 20. Data analysis and display program flowchart. **19**

Figure 21. Screenshot of computer program to display real-time VTD data, including all SpO2 signals; the red baseline (thin red trace), the red signal (bold red trace), the IR baseline (thin blue trace) and the IR signal (bold blue trace). **21**

Figure 22. Photos of the three enclosure parts; LCD bezel and holder (left and center) and box (right) **22**

Figure 23. Photos of the connections through holes in the box (left and center) and the completed device (right). **22**

Figure 24. Screenshot of Valsalva maneuver testing on a (hopefully) healthy human subject when blowing at a very high pressure level. **23**

Forward

Heart failure is a serious medical condition affecting about 6 million Americans, including adults and children. Although there is currently no cure, early diagnosis can help people live longer, more active lives through medicines and lifestyle changes. There are several existing procedures for detecting heart failure although many are expensive, time consuming or invasive, and not available for home use. One goal of this project is to create a low-cost device that can be used by unskilled individuals to quickly test for heart failure in a non-invasive method. Additionally, patients known to have heart failure are in need of an accurate self test to monitor their status, and keep them out of the hospital. The current standard is to have patients weigh themselves daily, but a more sensitive method is needed. Preliminary research suggests that a device using the Valsalva maneuver, described below, can meet this need, but no such device is commercially available. We would like to thank Parallax for hosting the 2013 National microMedic Contest and providing a forum that allowed us to meet and team up to develop a prototype of this device. Also, we would like to thank the US Army, TATRC and CMU for sponsoring the contest.

Introduction

Our project is a new medical device, referred to here as the VTD (Valsalva Test Device), to test for heart failure using the "Valsalva Maneuver". We bring a modern, high-tech approach to this old and well established test. To conduct the test, the patient "bears down" by blowing into a high resistance tube for 15 seconds while the doctor monitors the SpO2 signals which can show signs, indicating the presence of heart failure. A microcontroller is used to monitor the patient's attempt to blow and provide a feedback mechanism for the patient to maintain the desired pressure. The microcontroller also monitors a pulse plethysmographic ("pulse-ox") sensor to record pulse amplitude and display the pulse waveform changes on an LCD screen. At the same time, the microcontroller transmits test data over USB connection to a computer for recording and analysis. Cle1an, disposable and commercially available sensors are used to enable realistic application in the commercial, clinical setting. A photo of the working prototype device is shown in Fig. 1.

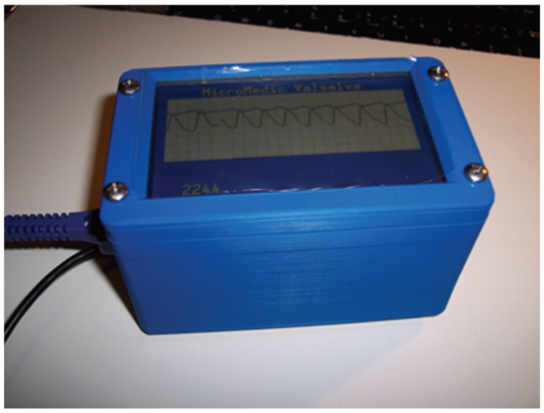


Figure 1. Photo of the prototype VTD during initial testing of the SpO2 circuit and program.

Medical Overview

In Medicine, physicians still struggle daily to estimate the internal fluid balance, or "volume status", of their patients. Common scenarios range from traumatic injuries with blood loss, to Congestive Heart Failure (CHF). When patients become too "dry", kidney failure can begin. When too "wet", pulmonary edema (water on the lungs), can suffocate them.

We still need a cheap, easy to use, accurate, non-invasive, portable technology to assess volume status.

Since the 1950's, researchers have noticed that the blood pressure change during a Valsalva Maneuver reflects the volume status of the patient. To perform a Valsalva Maneuver, the patient must hold their breath and "bear down", creating at least 25-30mmHg air pressure in their chest, for 10 to 15 seconds. Most of the research has focused on patients with CHF, since they are highly sensitive to volume status changes. It has been shown that measuring the blood pressure change during Valsalva can lead to medication adjustments which prevent repeat admissions to the hospital. Specifically, the ratio of the blood pressure Pulse Amplitude, Before and During a Valsalva maneuver, accurately estimates fluid balance when compared to invasive measurements of pressure inside the heart.

We wanted to create a device that can let physicians run this test in any environment, and make it easy enough that a medic could also run the test, or even the patient at home.

We also hypothesize that further research using this device may improve the treatment of shock-trauma in a field environment, as well as supporting more intensive care in field hospital and mass casualty scenarios.

There are two main components to measuring the Pulse Amplitude Ratio: measuring the blood pressure pulse wave amplitude, and properly performing a Valsalva maneuver. A simple, noninvasive method called Pulse Plethysmography, measures the pulse wave amplitude. A pressure transducer connected to disposable tubing enables the patient to blow hard enough to create 25-30mmHg air pressure and hold it for 10-15 seconds, producing a standardized Valsalva maneuver. Recording the pulse data allows easy measurement and calculation of the ratio, so the patient's fluid status can be determined in under 30 seconds.

Electronics Overview

The Valsalva Test Device (VTD) comprises two main medical instruments, a custom pulse plethysmographic sensor, also known as “pulse-ox” or SpO2 sensor, and a pressure sensor. As described in detail later, the VTD requires a special measurement of the SpO2 baseline, a signal not provided in commercially available units. The VTD analyzes and displays SpO2 and pressure data on a 4.3” LCD touchscreen. The Parallax Propeller microcontroller was chosen because it is easily programmed and can display color graphics and text on the 4.3” LCD with minimal circuitry, resulting in a low cost device. The VTD also includes an USB to serial interface used for programming, power and data transmission and a micro-SD card for data recording. The main components of the VTD are shown in Fig. 2. These components are distributed on three PCB boards. The main board is a Propeller Platform board with the Propeller microcontroller, USB to serial adapter and uSD card. The SpO2 sensor circuit and pressure sensor circuit are on the second, daughter PCB. The third PCB contains the circuitry required to drive the LCD display.

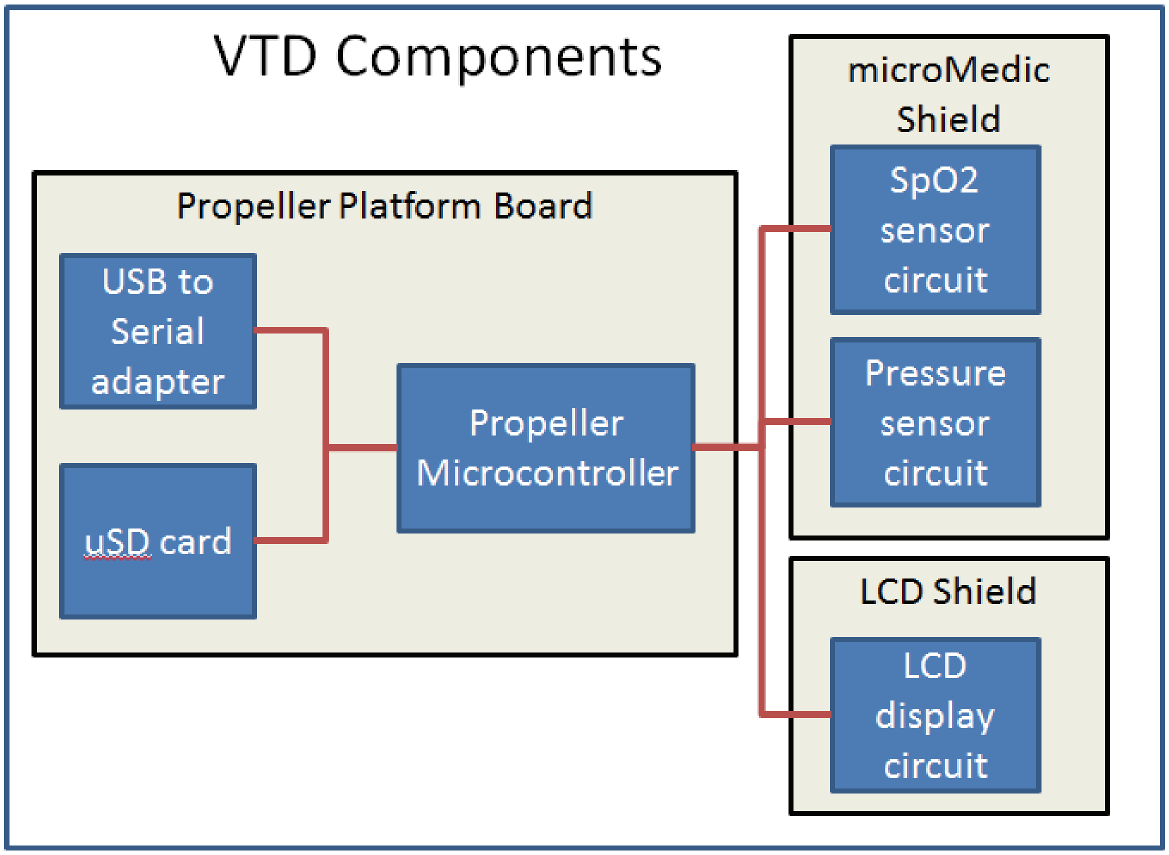


Figure 2. Diagram of the main PCB boards and components in the VTD device.

The Propeller microcontroller, USB to serial interface, and uSD card are all on one circuit board in a style known as the Propeller Platform. The Propeller Platform is popular in the user community as a base from which other boards, called "shields", can be easily connected. All of the Propeller's 32 I/O pins are brought out to headers on either side of the board with P0..P15 on one side and P16..P31 on the other. There are two rows of connected pins on either side allowing shields to be connected both above and below the platform. There are also two-row, four-pin power connectors on both side of the board. The particular Propeller Platform used for this project is known as "Propeller Platform Express" or PPE although other Propeller Platform boards would work equally well. The PPE also comprises 5 V and 3.3 V regulators, which provide power to the shields as shown in Figure 3. The PPE is an "Open Design" board and the schematic and PCB layout are freely available.

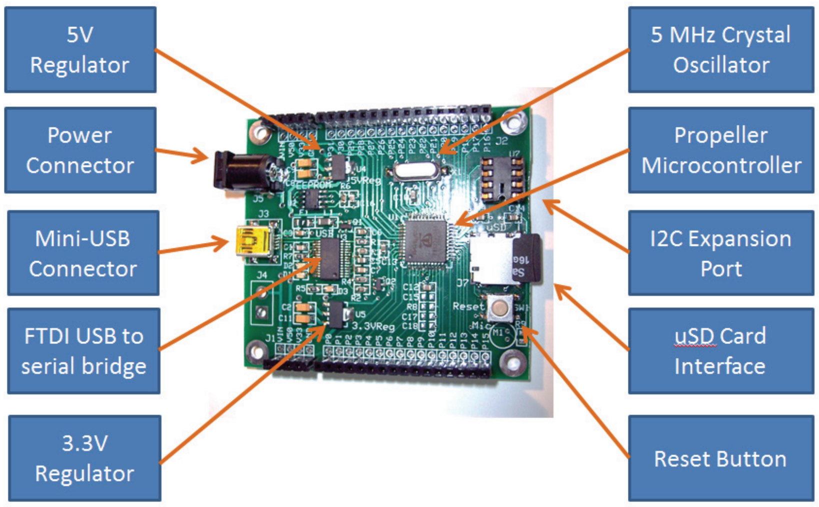


Figure 3. Photo of the Propeller Platform Express (PPE) board with components identified.

The SpO2 and pressure circuits are both contained in a "microMedic shield" that connects above or below the Propeller Platform. The SpO2 circuit connects to a Nellcor compatible disposable sensor with a regular DB9 connector, as shown in Fig. 4. The pressure sensor connects to a 4-pin connector on the other side of the board. This microMedic shield also has an EKG circuit in the extra space, but it is not needed for the Valsalva test described in this report and was not tested. The SpO2 and pressure circuits are described in great detail later in this report.

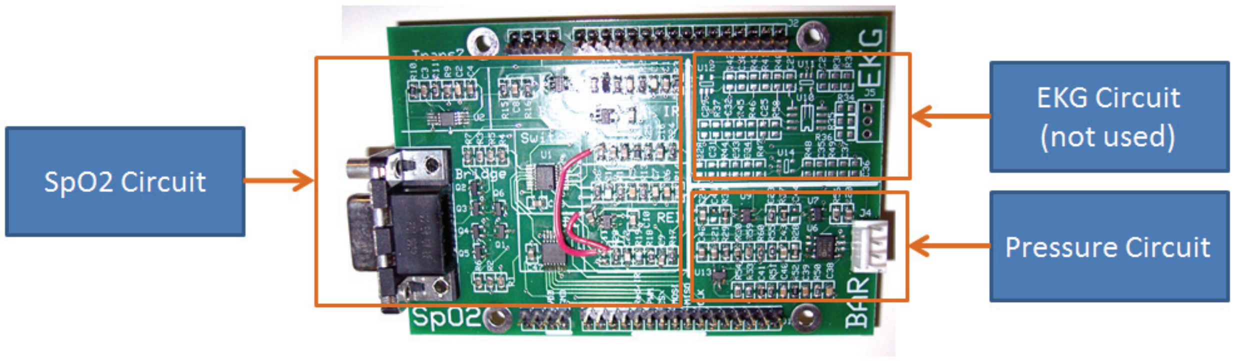


Figure. 4. The microMedic shield comprising SpO2, EKG and pressure circuits.

The third PCB circuit board used in the VTD is an adapter to drive a Samsung 4.3" LCD touchscreen. This 4.3" "breakout" board only has a few components, needed to display graphics on the LCD and respond to touch, as shown in Fig. 5. A resistive touchscreen controller chip, the TSC2003, provides touch information to the Propeller microcontroller over an I2C interface. The TSC2003 also provides an IRQ output that indicates if the screen is being touched. A simple boost circuit provides the ~15V power to drive the backlight of the LCD screen, which consists of several white LEDs in series. Six Propeller pins are used to provide 6-bit color data to the LCD screen (R0, R1, G0, G1, B0, B1). Six-bit color is the standard output of the Parallax VGA driver, which is used in a modified form to drive pixel data to the LCD. In the future, we may possibly use a variant of this board to interface with Newhaven Displays 4.3" touchscreens. The TSC2003 also provides two battery voltage measurements, which may be used in the future.

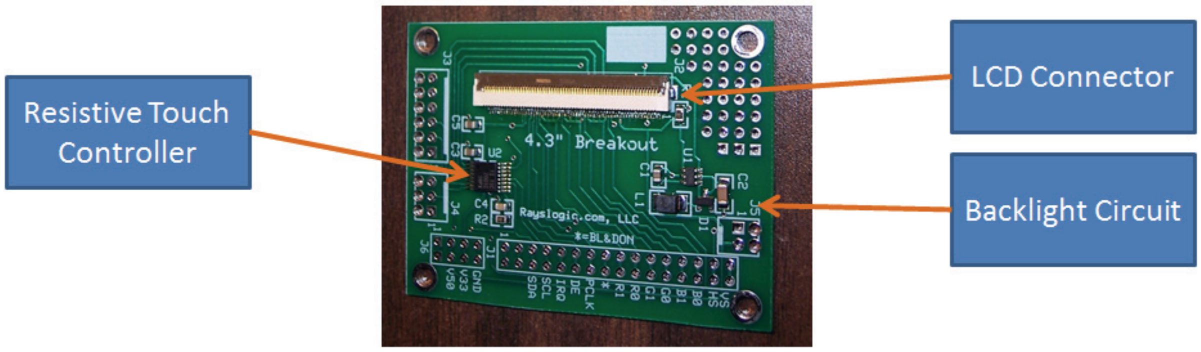


Figure. 5. An interface, "breakout" board to connect a 4.3" resistive touchscreen to the Propeller microcontroller.

SpO2 Sensor Circuit Details

Nellcor SpO2 Sensors

The SpO2 sensor circuit is designed to work with a Nellcor compatible sensors. The SpO2 sensors comprise a red LED, an infrared (IR) LED and a photodiode. They are arranged so that the LEDs are together and shine through the tip of the patient's finger to the photodiode on the opposite side. The two sensors used for initial testing of the VTD were an "official" Nellcor labeled disposable unit from a hospital and a clip-on reusable unit from China bought on EBay as shown in Fig. 6. Both units feature a convenient DB9 connector with the same pinout.

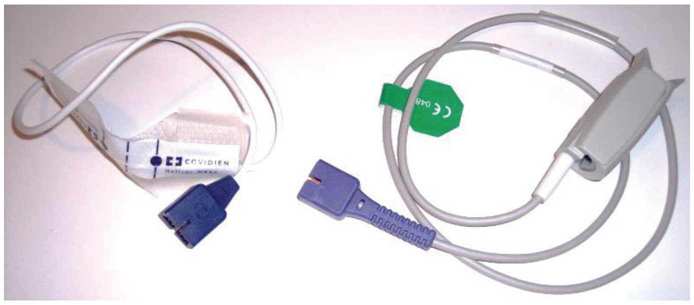


Figure. 6. Two SpO2 sensors used during VTD development, a disposable unit from a hospital (left) and a clip-on reusable clone from EBay (right).

Pulse Oximetry

The basics of pulse oximetry can be found in great detail on the internet, so we will provide only a brief explanation. The photodiode is sensitive to both the red and IR LEDs, so we alternate between shining only the red LED and measuring photodiode current and then shining only the IR LED and taking a reading. The signal that we are looking for is due to the heart pulsing blood in and out of the finger. When there is more blood, there is more absorption and the signal goes down. When there is less blood, there is less absorption and the signal goes up. The amount of oxygen in the blood can be determined by comparing the amplitudes of the red and IR signals. This is mostly due to the absorption of the red signal being more sensitive to the amount of oxygen in the blood than the IR signal. Oxygenated blood absorbs red much less than oxygen deprived blood. The IR signal is less sensitive to oxygenated blood and responds in the opposite way. When the average or DC value of both the red and IR signals are the same, one can simply use the ratio of the amplitudes of red to IR signals to estimate the percentage of oxygen saturation in the blood. If the red signal is much bigger than the IR signal, then the oxygen level is low. If the IR signal is bigger than the red signal, then the oxygen level is high.

SpO2 Circuit Description

There are several SpO2 circuits that can be found on the internet. Our circuit is largely based on that described by Freescale Semiconductor in their Application Note AN4327. A block diagram of the circuit is shown in Fig. 7. Two special features of this circuit are a 4-channel, 2-way video frequency switch that is used to switch between control and monitoring of red and IR signals and also separate red and IR signal amplifiers. The video switch reduces the number of microcontroller pins needed and allows red and IR signals to share the same ADC input channels. The circuit also features several high and low pass filters and a 60 Hz (line frequency) notch filter. These filters eliminate the need for software filtering, reducing the processing power required to analyze the signals. The high pass filters have long RC decay times, making it a good idea to have separate circuits for red and IR as we switch between them rapidly.

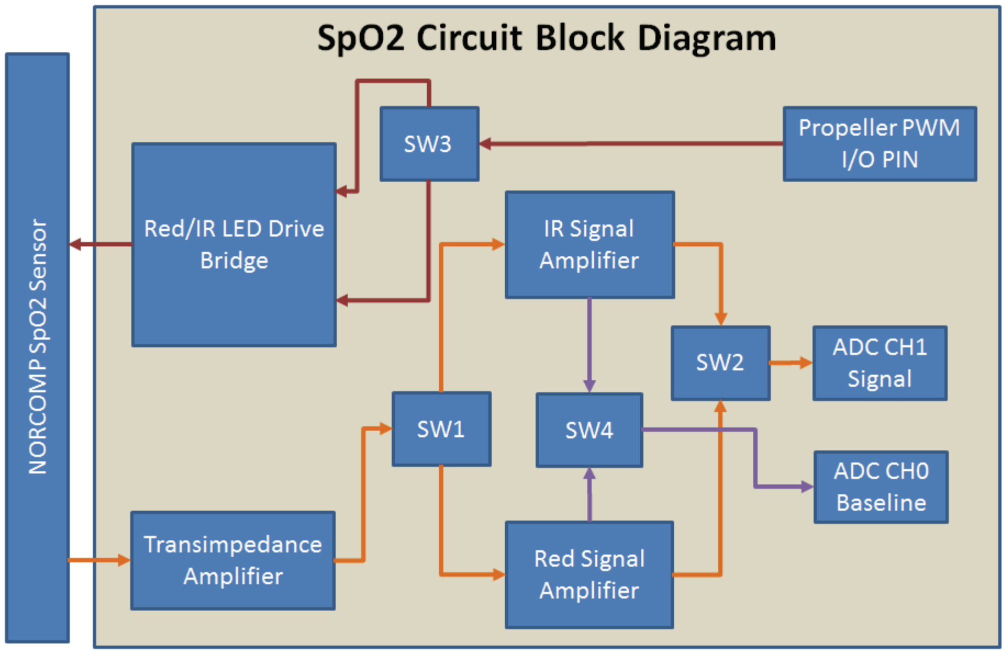


Figure. 7. Circuit block diagram for SpO2 measurement, showing how the four video switches are used to reduce the number of I/O pins and ADC channels required for SpO2 measurement.

Only two Propeller MCU I/O pins are required as inputs to control the circuit. The first pin is used to toggle the switches between red and IR mode. The second pin is used to do PWM (pulse-width modulation) on the red/IR LEDs to control their intensity. In red mode, SW3 (switch #3 of the four-channel video switch) sends the PWM output of a Propeller I/O pin to the bridge, driving the red LED. Also in red mode, the output of the transimpedance amplifier, which converts the current signal from the SpO2 sensor's photodiode into a voltage signal, is sent through SW1 to the red signal amplifier. The first part of the red signal amplifier is just a low pass filter, which is tapped by SW4 and sent to channel #0 of the ADC (analog to digital converter) chip. The signal is very small at this point, but we use the DC level of the baseline to decide if the LED intensity is at the right level for the best signal. The signal continues through the red signal amplifier where the DC level is removed and the AC signal is greatly filtered and amplified. The red signal out of the amplifier is sent through SW2 to channel #1 of the ADC chip. For IR mode, the switches send the controls and signals to the IR portions of the circuit. In summary, because of the video switch only 2 Propeller I/O pins are needed for control and two ADC channels are needed for data collection. The Propeller does not have an onboard ADC circuit, so we use a dedicated ADC chip with an SPI interface. The SPI interface requires 4 Propeller I/O pins, bringing the total required I/O pins to six.

Bridge Circuit

The red and IR LEDs are driven by a bridge circuit, as shown in Fig. 8. Bridge circuits are often used with DC motors to direct current either forward or backward through the motor to control its direction. Here, we use a bridge circuit to switch current between Red and IR LEDs and also control their brightness. One of the video switches inputs a PWM signal from the Propeller and outputs to either the "PwmRed" line or the "PwmIR" line, depending on the switch control being in red or IR mode. The red and IR LEDs inside the Nellcor SpO2 sensor are connected in parallel with the anodes on opposite ends. The signal, "Red", in Fig. 8 connects to the red LED's anode and the signal, "IR", connects to the IR LED's anode. This LED drive circuit is arranged as a bridge, using four switches (Q2, Q3, Q4 and Q5) to direct current either from Red to IR or from IR to Red. The direction of the current determines whether the red or IR LEDs are lit. When in red mode and when the PwmRed signal is high, the transistor, Q1, turns on switches Q2 and Q4. This directs current from the power supply, through R7, through Q2, through the red LED's anode, out the red LED's cathode, then through Q4 to ground. When in red mode and the PwmRed signal is low all these transistors are off. When in IR mode, the circuit works in the mirror image way with Q6, Q5 and Q3 turned on when PwmIR is high. The resistor, R7, is there to limit the maximum possible current in the LEDs, preventing damage. Both PWM signals alternate quickly between the high and low state. The percentage of time high compared to low determines the brightness of the LED being driven. As described later, the Propeller programming will increase or decrease the PWM on time to achieve the desired brightness level. The PWM frequency is much faster than the response time of the photodiode circuit, so this PWM modulation does not appear in the filtered photodiode signal.

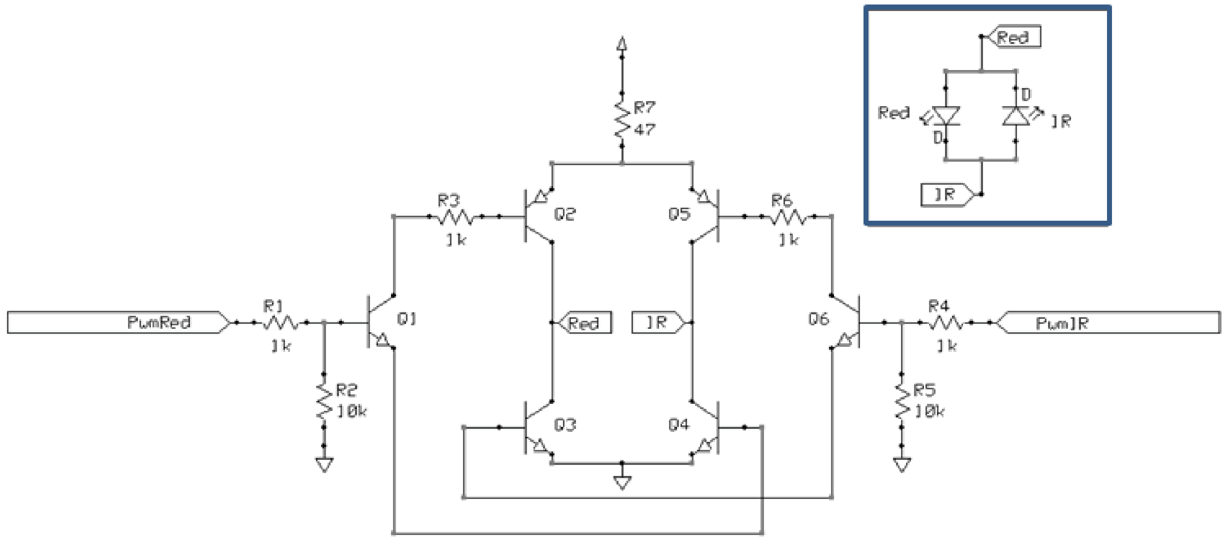


Figure. 8. Bridge circuits for the red and IR LEDs, which are connected in parallel as shown in the inset.

Transimpedance Amplifier

The transimpedance amplifier, shown in Figure 9, takes the very small currents generated by light hitting the photodiode and turns it into a buffered voltage signal. R9 and R11 form a voltage divider to give us an offset bias voltage, "TransImpedanceOffset", of about 0.35 V. This keeps the output above ground. This circuit is nearly identical to one in the OPA381 datasheet with the exception that the photodiode anode is not connected to ground. If the photodiode anode were grounded the photodiode would be slightly reverse biased by the offset bias voltage and the time response would be slightly better. In this circuit, however, the photodiode's voltage is kept at zero. The current generated in the photodiode by incident light is the same, but the response is slower, which is fine for this application. In steady state, any current produced by light on the photodiode must be exactly balanced by current in the feedback resistor, R10. The large value of R10, 4.7 MΩ, means that tiny photodiode currents produce large voltages in the output of the transimpedance output, "TransImpedanceOut". Again, this is the function of a transimpedance amplifier, taking small currents and turning them into voltage signals. The steady state output voltage can be related to the photodiode current by

$$V_{TransImpedanceOut} = V_{TransImpedanceOffset} + 4.7 \cdot 10^{6} \cdot I_{photodiode} \quad (1)$$

where $I_{photodiode}$ is the current through the photodiode due to red or IR light from the LEDs. The feedback capacitor, C3, filters out high frequency noise.

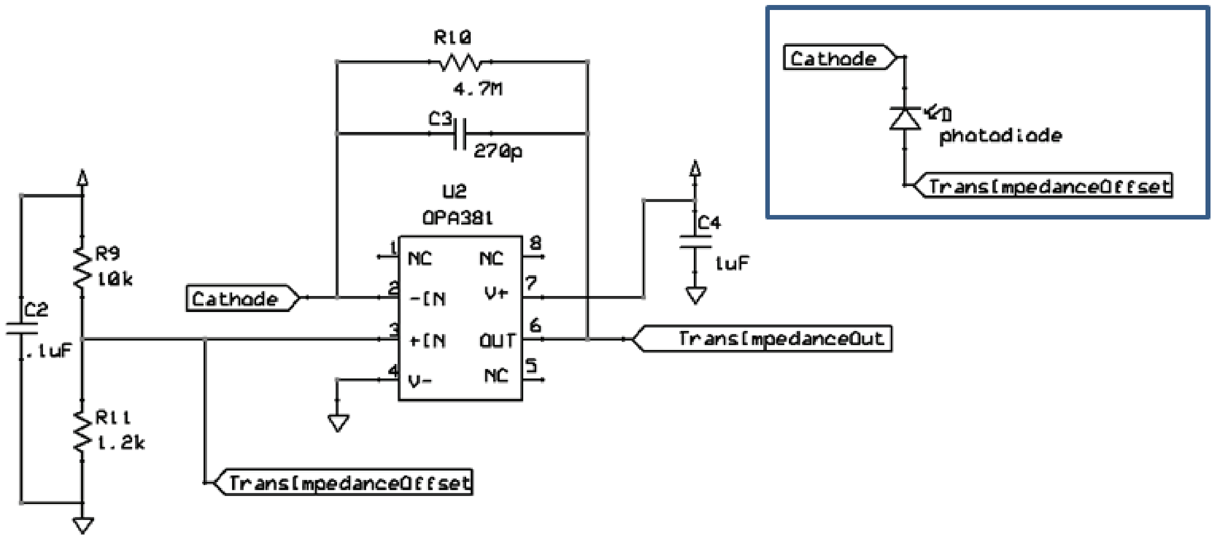


Figure. 9. Transimpedance amplifier circuit with photodiode connection shown in the inset.

Red and IR Amplifiers

The output of the transimpedance amplifier is directed through the video switch to either the red or the IR amplifier, depending on which mode we are in. The amplifier does three things; filters out unwanted frequencies, removes the DC component of the signal and makes the signal big enough to be easily digitized. The red and IR amplifiers are identical and if it weren't for the very long time constants being used in the RC filters here, we could have used the same amplifier for both red and IR, as many other versions of SpO2 sensors do. But, we have chosen this approach with hardware rather than software filtering. The red amplifier is shown in Fig. 10 where the input is "TransImpedanceRed", which is the output of the transimpedance amplifier after going through the video switch in red mode.

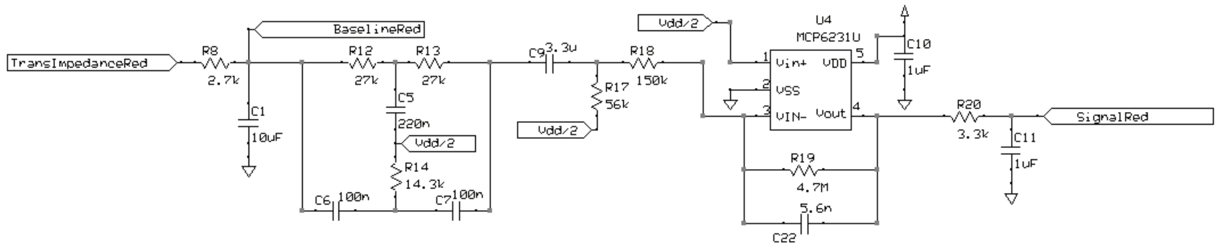


Figure. 10. Red amplifier circuit (IR amplifier is identical).

The components R8 and C1 form a simple low-pass RC filter with a cut-off frequency, f_c, of 5.9 Hz given by

$$f_c = \frac{1}{2\pi RC} \quad . \qquad (2)$$

This is a very low frequency, but the frequencies we want are mainly in the range from about 1 to 6 Hz and this removes frequencies above this. Before going further, we tap into the amplifier and send this slightly filtered signal, "BaselineRed", through the video switch to one channel of the ADC to be digitized. This baseline signal still contains the DC level, which we need to determine how much red or IR light is getting to the photodiode. The reason for this will be described in more detail later.

Next, the signal goes through a notch filter (R12, R13, C5, R14, C6 and C7) meant to remove any 60 Hz noise from power lines. Then, the signal goes through a high-pass filter made up of C9 and R17 with a cut-off frequency of 0.86 Hz. This removes unwanted low frequencies in the signal, including the DC level. One end of R17 is connected to "Vdd/2", which is a buffered DC voltage equal to half of Vdd or 1.65 Volts. We do this so that our AC signal now rides on top of a midrange DC level so that when it gets to the ADC, even a large signal will stay within the range we can digitize, 0 to 3.3 Volts.

Next, the red signal goes through an amplifier made up of R18, U4, R19 and C22. For the opamp, U4, we have chosen the general purpose MCP6231U. The DC gain (amplification factor) of this inverting amplifier is just given by R19/R18, equal to 31.3. The capacitor, C22, is placed in parallel to R19 to act as another low pass filter, with cut-off frequency of 6.0 Hz.

Finally, the red output signal goes through one final low pass filter, made up of R20 and C11 with a cut-off frequency of 48 Hz. This output signal, "SignalRed", goes to the video switch where in red mode it is sent to the ADC for digitization.

Vdd/2 Buffer

Another MCP6231U opamp is used to provide the Vdd/2 power supply used in the red and IR amplifiers. It is simply arranged as a non-inverting buffer driven by a voltage divider of equal resistors (R15&R16) to feed it ½ of the 3.3 V power supply voltage as shown in Figure 11.

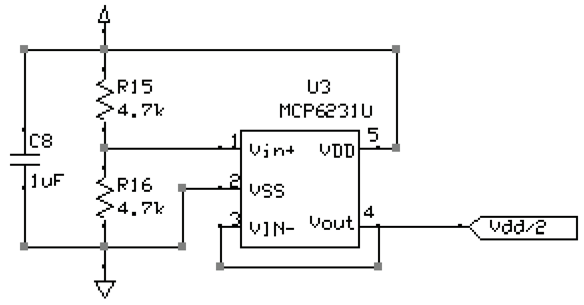


Figure. 11. Voltage divider and buffer amp to provide a Vdd/2 power supply.

Red/IR Switch

The red/IR switch is simply the FSAV430 chip with four single-pole, double-through (SPDT) switches that connect each of four A channels to either their B1 or B2 channels, depending on the switch control input. One Propeller I/O pin is used for the "SwitchControl" signal shown in Figure 12, allowing us to switch signals between red and IR modes. When "SwitchControl" is low, we are in red mode and, for example, the switch's 1A input ("TransImpedanceOut") is connected to the 1B1 ("TransImpedanceRed") input. In IR mode, 1A is connected instead to 1B2 ("TransImpedanceIR"). The switch is also used for the red and IR baseline, signal and PWM signals.

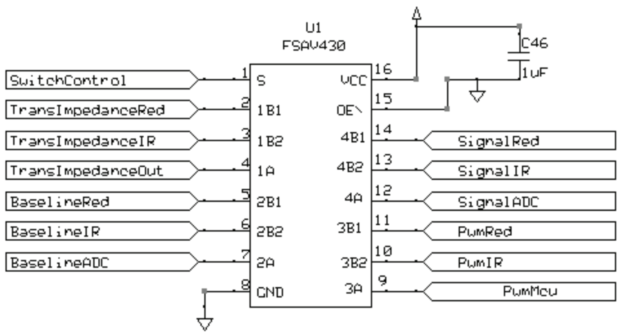


Figure. 12. Video switch used to switch between red and IR modes.

Analog-to-Digital Converter

The last part of the SpO2 circuit is the ADC (analog to digital converter). The four channel ADC chip, MCP3204, is used to digitize the baseline and amplified SpO2 signals as shown in Fig. 13. The Propeller microcontroller gets the digital data from MCP3204 ADC over an SPI (serial peripheral interface) bus made up of four control signals; clock ("CLK"), data output ("DOUT"), data input ("DIN") and chip select ("CSn"). The first two of the MCP3204's four analog inputs are used for SpO2 measurements. The third analog input is used for the output of the pressure measurement circuit, described later. The fourth analog input is connected to the EKG circuit, but is not used in this application. We have the reference voltage input, VREF, tied to Vdd, so the 12-bit outputs will range from 0 with 0 V applied to 4095 with 3.3 V (Vdd) applied.

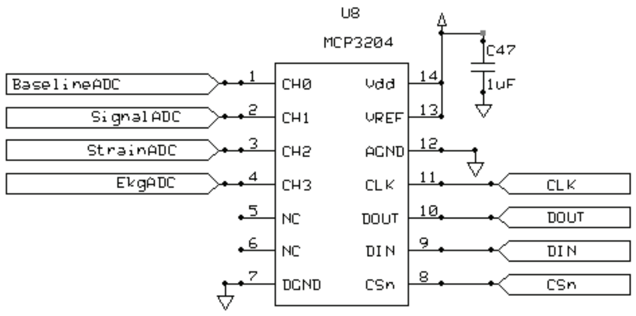


Figure. 13. Four channel ADC circuit with SPI connection to MCU.

Pressure Sensor Circuit Details

Compared to the SpO2 circuit, the pressure sensor circuit is simple. The pressure sensor we are using provides a tiny voltage that is proportional to the pressure on the sensor. The circuit just amplifies this signal so it can be digitized by the ADC, described earlier.

Pressure Sensor

An actual disposable pressure sensor from a hospital was used for this project. One difficulty we encountered is that we could not find any specifications for the sensor. So, we took one of the sensors apart in an attempt to identify it. After breaking open the plastic case to expose the circuit board shown in Fig. 14, we believe we identified the sensor as model 1620 from Measurement Specialties. The 1620 sensor is a silicon piezoresistive pressure sensor, normally used for invasive blood pressure monitoring. One interesting thing about this sensor is that it contains resistors that are a film deposited directly on the circuit board (black areas in Fig. 14). A laser is used to trim these resistors to the correct value by drilling holes in the film.

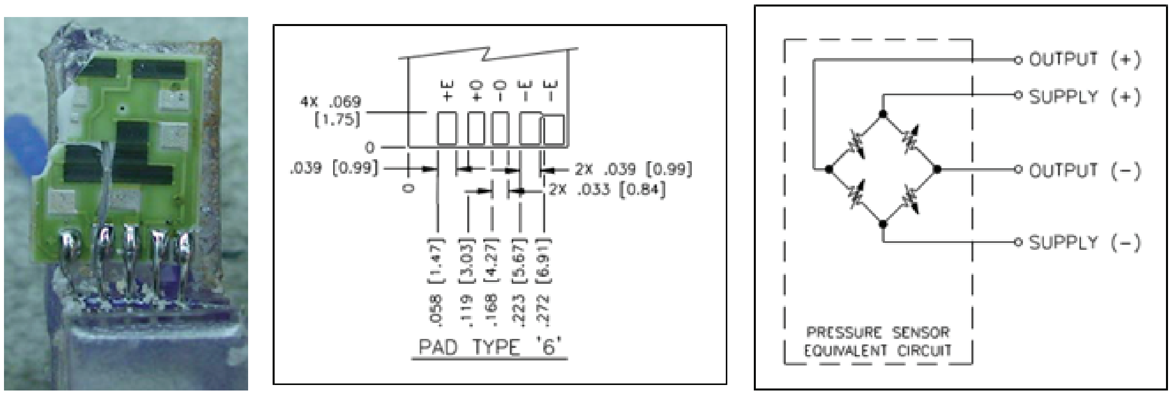


Figure. 14. Pressure sensor circuit board from disposable pressure sensor used in a hospital (left), wire connection pad layout from 1620 datasheet (center) and equivalent circuit of the sensor (right).

Another interesting thing about this sensor is that it has a 5-wire connection although as seen in Fig. 14, only 4 wires are needed. It is speculated that the fifth wire is just used to verify connection to the pressure sensor. Another difficulty we encountered with this sensor is that we could not find a mating electrical connector. It is speculated that the connector is a proprietary design. So, for this project, we cut the connector off and wired in a standard 4-pin header instead to connect to our microMedic circuit board.

Instrumentation Amplifier

The main component in this pressure circuit is the AD627 instrumentation amplifier. Instrumentation amplifiers are specially designed to amplify the output of a resistive bridge, like the strain gauge sensor of the 1620. We could have used the AD627 alone to provide the signal we needed, but we have added additional offsets, buffers and filters to the circuit, just in case it proved necessary. The AD627 instrument amplifier circuit is shown in Fig. 15 along with the 4-pin connection to the sensor, J4. With no pressure applied, the strain gauge pressure sensor's outputs, "Strain+" and "Strain-", are both equal to Vdd/2. When positive pressure is applied, such as from blowing into the tube connected to the sensor, the "Strain+" voltage will increase and the "Strain-" voltage will decrease the same amount. The AD627 amplifies the input voltage difference by an amount determined by the external gain resistor, R28. In this circuit, we have R28 as 2.1 kΩ for a gain of 100 and then we use additional amplifiers to further boost the signal before sending to the ADC. However, as noted earlier, we could have dramatically simplified this circuit by using the lowest gain resistor allowed, 200 Ω, for a gain of 1000 and then sending the AD627's output directly to the ADC without filtering or offset from ground.

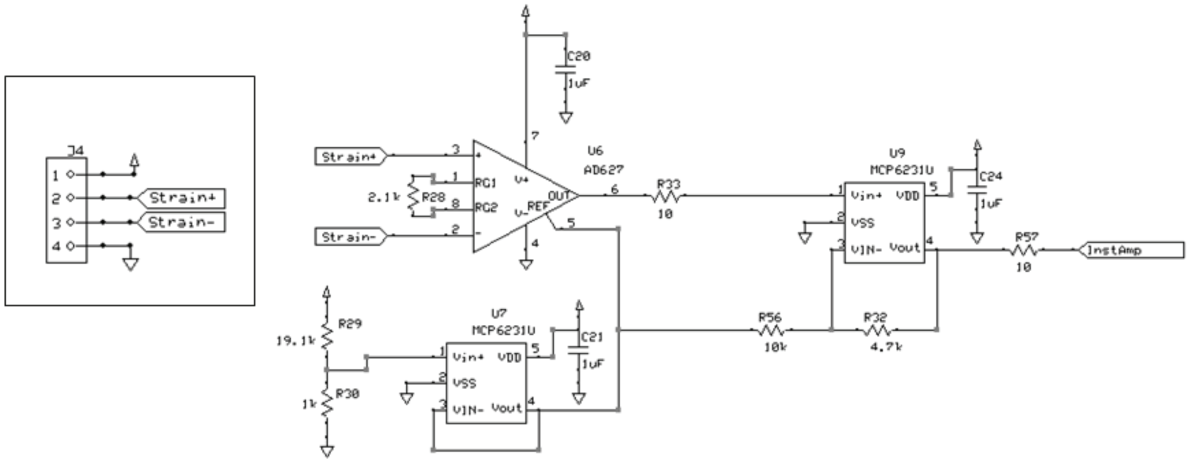


Figure 15. Pressure sensor instrument amplifier (U6) circuit with offset amp (U7) and a non-inverting amplifier (U9).

The opamp, U7, provides a buffered reference voltage to the AD627 in order to bring the output up above ground when no pressure is applied. R29 and R30 form a voltage divider that provides this 0.16 V level to this non-inverting buffer amplifier. We continue to use the MCP6231 opamps for this general purpose application. The output of the AD627 is sent to the non-inverting amplifier, U9, where it is boosted with a gain given by (R56+R32)/R32 or 3.1.

Pressure Amplifier and Filter Circuit

The output of the instrument amplifier circuit is then sent rough a filter and amplifier circuit shown in Fig. 16. First, there is the same 60 Hz line filter used in the SpO2 red and IR circuits, made up of R50, R51, C38, R52, C39 and C40. Then, there is the non-inverting amplifier made up of U13, R60 and R59. To calculate the signals amplification through this circuit, we first recognize that the signal is resistively divided with R50, R51, R61 and R53 by a factor of 56k/(56k+54k) = 0.51, or roughly half. Then, the feedback resistors of the opamp, R60 and R59, give a gain of (100k+10k)/10k = 11. The output of the opamp then goes through low pass RC filter made from R55 and C43 with a cutoff frequency of 4.8 Hz before going through the video switch to the third channel of the ADC, described. Again, this circuit contains elements that turned out to be not really needed, particularly R61 and R54.

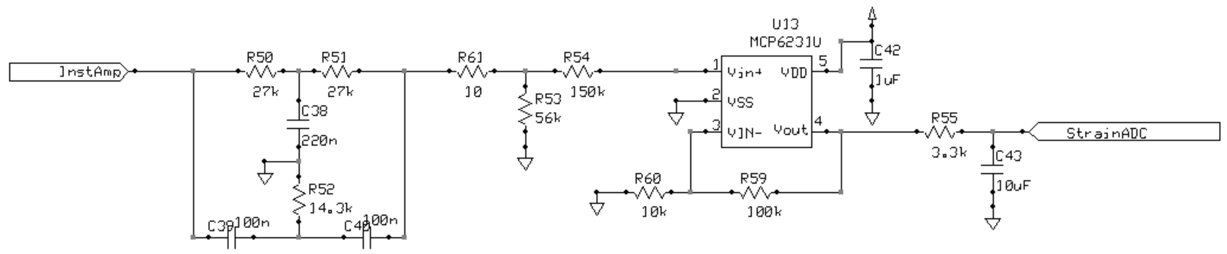


Figure 16. Pressure filter and amplifier circuit.

Pressure Sensor Calibration

To calculate the actual pressure on the sensor from the voltage digitized by the ADC, we need the calibration factor of the sensor, give in the datasheet as 5.00 uV/V/mmHg. Since we are providing 3.3 V power to the sensor, the calibration is then 16.5 uV/mmHg. (Our target pressure of 25 mmHg then gives a 412.5 uV signal at the input to the instrumentation amplifier.) With a gain of 100, the output of the instrumentation amplifier is 1.65 mV/mmHg. After the filters and buffer amps, the signal is then boosted by a factor of 3.1*0.51*11 = 17.4 for a final output of 28.7 mV/mmHg. At our target pressure of 25 mmHg, we then have a signal of 0.71 Volts, which is comfortably in the range of the ADC, from 0 to 3.3 V. Note that since we provided an offset voltage to the instrumentation amplifier the final output will be somewhat above zero with no pressure applied. So, as described later, the code will need to subtract off the reading with zero pressure applied to determine the actual pressure.

Programming Overview

The Propeller microcontroller was programmed in its native language, called Spin. Fortunately, most of the code required to implement the VTD was already available as “Objects” from the Parallax web site in a place called OBEX for “object exchange”. The Propeller is rather unique in that it has eight identical cores that can be used to run independent Spin code or to act as soft peripherals running assembly code. For example, the serial communications is done with assembly programmed cores, instead of dedicated peripheral hardware found in other microcontrollers. For some of the soft peripheral drivers, such as SPI and I2C bus interfaces, a driver with assembly speed was not required and so was just accessed with Spin code. The way the VTD prototype used the cores is shown in Fig. 17. Only six of the eight available cores were used in the original prototype. The remaining cores will likely be utilized in the final version for uSD card data recording, for example. The Propeller can have up to 32 kB of programming code and the prototype only uses 12 kB, so there is plenty of room to add features. The source code is split into several files with one main file “MicroMedic2e.Spin” and several object files that contain Spin or assembly drivers. The main file contains the Spin code for two cores, the “SpO2 control and data acquisition” core and the “Data analysis and display” core. These two cores launch the assembly driver code that runs in the other four cores that are used.

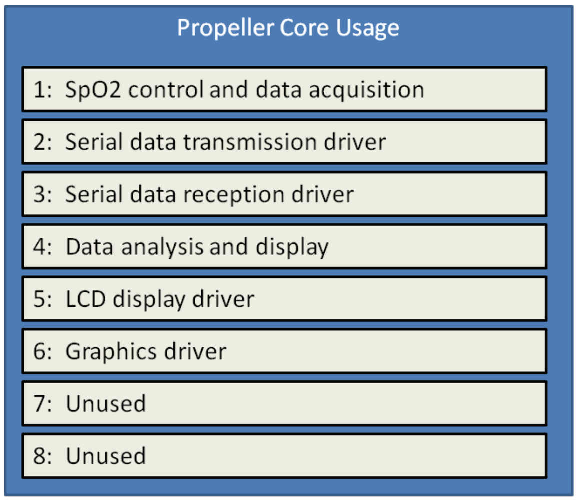

Propeller Core Usage
1: SpO2 control and data acquisition
2: Serial data transmission driver
3: Serial data reception driver
4: Data analysis and display
5: LCD display driver
6: Graphics driver
7: Unused
8: Unused

Figure 17. Usage of the eight Propeller cores in the VTD prototype.

Programming Details

A block diagram of the VTD programming code is shown in Fig. 18. There are two main Spin programs running at the same time in separate cogs, “SpO2 control and data acquisition” and “Data analysis and display”. The “SpO2 control and data acquisition” can be thought of as the primary part of the code. It controls the SpO2 sensor by switching between red and IR modes and setting the PWM LED brightness. It also gets the SpO2 and pressure data from the ADC using the SPI driver and then transmits the data using the serial driver. In the early stages of code development, there was only this code and the serial data was captured and analyzed over USB connection to a computer.

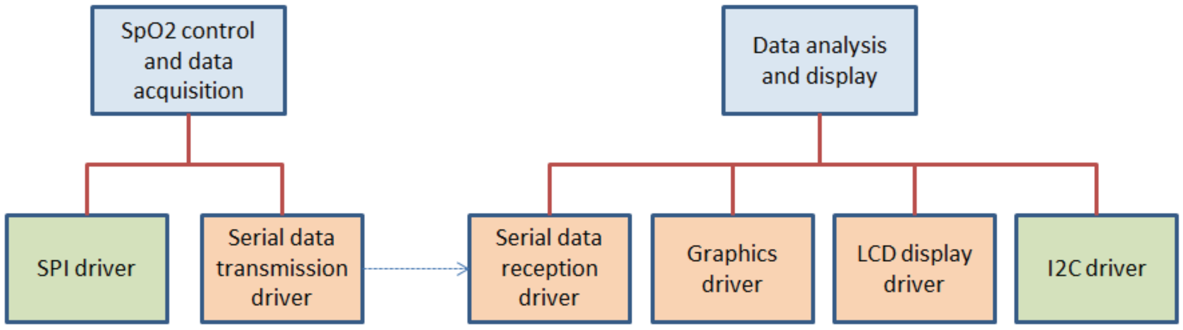

SpO2 control and data acquisition
Data analysis and display
SPI driver
Serial data transmission driver
Serial data reception driver
Graphics driver
LCD display driver
I2C driver

Figure 18. Block diagram of the Propeller program for the VTD showing the two main Spin codes (blue) with associated Spin drivers (green) and assembly drivers (peach).

The other main program “Data analysis and display” was then added to display data locally on a 4.3” LCD touchscreen. The two main programs, although running independently, share main memory and so main memory can be used to exchange data between them. However, for this application, we mostly share data a different way, using another serial driver. A second serial driver is started to capture the data stream being transmitted by the “SpO2 control and data acquisition” program. One benefit of this approach is that we are certain that both the USB connected computer and the “Data analysis and display” program are receiving the exact same data. This program then takes the data, does some analysis, and the displays it on the 4.3” LCD through the assembly driver. The SpO2 data is displayed as a graph using the “Graphics driver”. The touchscreen of the LCD is interfaced using the Spin “I2C driver” so that on-screen buttons can be used to control the VTD baseline functions, described later.

SpO2 Control and Data Acquisition Program

A flowchart for this part of the program is shown in Fig. 19. After starting the serial and SPI drivers, it goes into an infinite loop where it switches the SpO2 between red and IR modes while transmitting SpO2, pressure and EKG (not used) data over the serial interface in every loop. Also, it continuously monitors the SpO2 baseline signal and adjusts it, as required, by varying the PWM of the Red or IR LEDs.

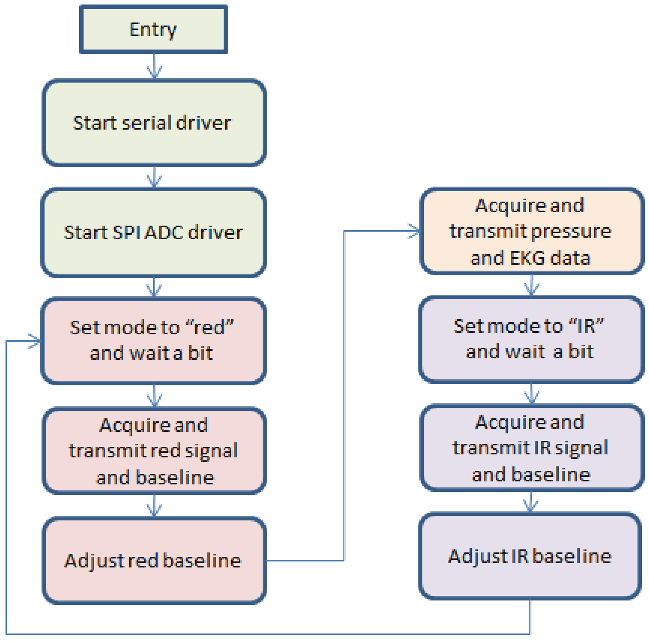

Entry
Start serial driver
Start SPI ADC driver
Set mode to “red” and wait a bit
Acquire and transmit red signal and baseline
Adjust red baseline
Acquire and transmit pressure and EKG data
Set mode to “IR” and wait a bit
Acquire and transmit IR signal and baseline
Adjust IR baseline

Figure 19. SpO2 control and data acquisition program flowchart.

Switching between red and IR modes simply involves setting the video switch control pin to 0 (for red) or 1 (for IR) and also setting the respective PWM duty cycle. We have independently adjustable settings for red and IR PWM in case the two LEDs do not give the same photodiode response for the same PWM setting. After switching modes, we use the “waitcnt” command to insert a delay to allow the amplified signal to settle before sampling the voltage with the ADC. Each waitcnt command waits for 10 milliseconds since the last waitcnt ended. So, there is a complete cycle every 20 milliseconds, resulting in 50 data sets per second being transmitted. Since the heart usually beats at about once per second, we get 50 points of data for every heart cycle, which is enough for a smooth curve, especially with the heavy hardware filtering. For pressure, the program just takes a reading and transmits it. There is no processing or control done with the pressure signal.

Baseline Control

Control of the red and IR baseline levels is the most complex part of the code. Especially since the VTD does something unique by allowing the option of not controlling the baseline level during the Valsalva measurement. What baseline control does is either increase or decrease the PWM controlled intensity of the red or IR LED until the respective baseline signal gets within the desired range. This is needed to get the optimal light intensity on the photodiode to give a signal level that is not too small or too big. If the baseline level is too small, the signal won't be big enough to accurately digitize. If the baseline level is too large, the photodiode and/or amplifiers will be in a non-linear regime, distorting the signal. One reason baseline control is needed is because factors like the thickness of the finger and amount of light that gets through the finger varies from patient to patient, requiring more or less LED light to get the same response from the photodiode. The PWM value is adjusted between 2000 levels using the hardware counter and the "frqa" register, as described in the Parallax Appnote AN001.

The ADC is 12-bit, so the measured baseline signal can range in value from 0 to 4095. Our testing has shown that a midrange value of 1000 for the baselines works well for our circuit. However, the Valsalva measurement requires that the baseline be kept steady. So, we have use a Boolean control variable, "bAutoBaseline" that enables or disables automatic baseline control. The code compares every measured baseline level with the desired level in every cycle. If "bAutoBaseline" is true and the baseline is out of the desired range (700 to 1300), the code adjusts the PWM level. Once the baseline gets within the desired range, baseline control stops so that it doesn't affect the signal. If "bAutoBaseline" is false, which we set when making the Valsalva measurement, the code will not change either of the PWM levels.

Data Analysis and Display

The second main program is responsible for data analysis and display on the 4.3" touchscreen. It also checks for and responds to touches on the screen. This program is located in the top of the "VTD1.Spin" file and the previously described SpO2 control and data acquisition program is located at the bottom of the file. The flowchart for this program is shown in Fig. 20. One of the first things the data analysis and display program does is start the SpO2 control and data acquisition program using the "Cognew" command. Next, after doing some setup tasks, the program goes into an infinite loop where it gets new data from the serial driver's receive buffer, processes the data and then displays it on the LCD screen.

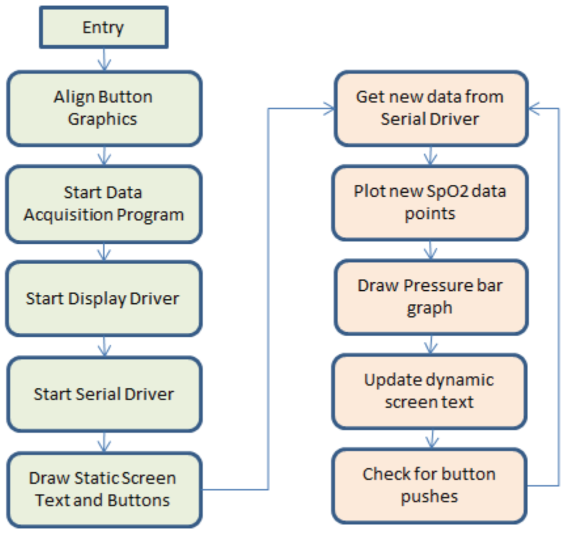

Entry
Align Button Graphics
Start Data Acquisition Program
Start Display Driver
Start Serial Driver
Draw Static Screen Text and Buttons
Get new data from Serial Driver
Plot new SpO2 data points
Draw Pressure bar graph
Update dynamic screen text
Check for button pushes

Figure 20. Data analysis and display program flowchart.

LCD Display Driver Details

The LCD display driver, as mentioned earlier, is an adaptation of the standard VGA driver from Parallax and is contained in the file, "PSB_LcdDriver". The main changes that were made in the adaptation where changing the resolution from 640x480 to 480x272, changing the vertical and horizontal timings, and adding a pixel clock and data enable signal. There are long threads in the Parallax Propeller user forum describing the driver and how to use it in great detail. The driver, just like the VGA driver, is tile based where the screen is divided into an array of 16x16 pixel tiles and the word array "screen" contains one word (16-bits) for every tile. Each word in the screen array contains a 10-bit pointer to somewhere in memory where the pixel data is located. Usually, this is a place in the ROM where the font data resides. The other 6 bits of the word are used as an index to a 64 member array of color options that can be used to paint the tile. The driver works in 2-bit color, meaning that only four colors can be used to paint each tile. Each member of the 64 color option array is a long with four bytes, one byte describing each of the four possible colors that can be used to draw the tile. Normally, the driver is used in text mode, using the ROM font. The ROM font data is made up of 64 member sets of longs, each of these sets describes two characters with their 1-bit data interleaved into 2-bits. Which character is drawn is determined by the color chosen to draw the tiles. So, for any particular choice of foreground and background colors, we need to use two of the 64 color sets to draw the font. One color set draws the odd numbered characters and the other draws the even number character.

LCD Graphics

Although the display driver is normally used with ROM font text, there is a way to use it for dynamic graphics using RAM also using a graphics driver provided by Parallax in the file "Graphics.Spin". The Propeller does not have enough RAM to do this for the entire screen, so we limit the graphics area to a

28x10 tile area where we will draw the SpO2 signal chart. A second method of drawing 2-bit graphics is used to draw static graphical buttons on the screen. The button graphics data was created on a computer and saved as the files, “Button_AB_On2.dat” and “Button_AB_Off2.dat”. One difficulty with this approach is that the program must move this graphics data so that it is aligned in memory so that the lower 10 bits of its address are all zeros. So, the first thing the program does is calculate the address where the graphics data must be moved to, “user_charbase”, and then use the “longmove” command to move the graphics data into position. Later, the function “Bitmap2Bit” is used to draw the buttons on the screen.

Data Analysis and Display Loop

The core of this program is in the function, “PlotData”, which contains in infinite “repeat” loop that does the data analysis and display. The timing is controlled only by using the “rx” function of the serial driver to wait for a new byte to arrive in the receive buffer before returning it. One limitation here is that this loop has to be faster than the 20 ms data transmission loop time, or the receive buffer will quickly overflow. The data stream contains a special character, “$”, after every data set. This program checks for this character in the stream to make sure we are in sync. Next, the program calculates a new average value for red (“RedAverage”) and IR (IrAverage) signals using the new data. This is simply done with these calculations

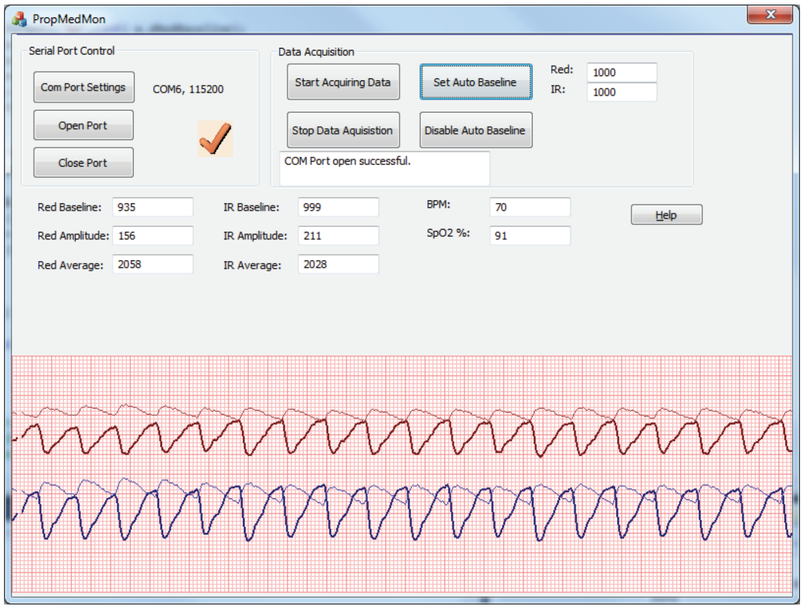
$$\text{RedAverage} = (\text{RedAverage}*99+\text{SignalRed}) / 100$$

$$\text{IrAverage} = (\text{IrAverage}*99+\text{SignalIr}) / 100 \tag{3}$$

where the old average is given a weight of 99 and the new value a weight of one. This is a fast and simple way of calculating an average of the last many samples. This average value is used to draw the SpO2 signal in the graphics area. Next, we update the graphics area by drawing the grid chart background and then plotting the SpO2 data on top. To make it simple, we draw only two traces. One trace is the sum of red and IR signals minus their average values. The other trace is the sum of red and IR baselines minus their expected averages of 1000 each. We do not calculate an average for the baseline signals so that we can see the Valsalva action in the baseline. Next, we draw a bar graph under the SpO2 chart to show the current pressure reading. Instead of using pixel graphics, we draw the bar graph with text graphics to conserve memory and time. This is done by simply displaying space characters in two different colors. We start at the left side of the screen drawing space characters in red and continue drawing spaces to the right until we get to the current pressure, then we draw space characters in the background color all the way to the right side of the screen. At two places we draw a vertical pipe character “|” instead of space to show the target range of 20 to 30 mmHg. Next, the program updates text values for the current pressure and the on/off status of the “AutoBaseline” function. Finally, the program touchscreen controllers IRQ output (connected to a Propeller pin) to see if the screen is being touched. If it is being touched and being touched not too hard and not too soft, we use a function “CheckButtons” to see if the x and y coordinates of the touch correspond to one of the buttons. This code is not fast enough for us to do this check on every loop, so we only do it once every 10 loops, or 5 times per second. Currently, there are only two buttons, one turns the automatic baseline feature on and one turns it off. In normal SpO2 mode, we want the automatic baseline on. But, during the Valsalva measurement, we need it to be off. The two main programs both have access to the variables in hub memory that control the automatic baseline feature, "bAutoBaseline", "TargetRed", "TargetIR", "bBaselineRed" and "bBaselineIR".

Computer Program Description

For more in-depth real-time data analysis, the VTD can be connected to a PC running a specially made program. The MFC C++ program runs in Windows and shows a single dialog screen, shown in Fig. 21. This program provides a much larger graph area on which all four SpO2 signals (red baseline, red signal, IR baseline and signal) are displayed for many cycles. The program also shows numerical levels for the red and IR baselines, signal amplitudes and average values. Additionally, the beats per minute (BPM) and blood oxygenation percentage (SpO2%) are calculated and displayed. To use the program, you first use the "Com Port Settings" button to select the com port that the VTD is connected to (You can use the Propeller Tool's "Identify" function to determine which com port is assigned). Then, click "Open Port" and "Start Acquiring Data" to start data acquisition and display. For the Valsalva measurement, you can use the "Disable Auto Baseline" button to tell the VTD to stop adjusting the baseline level. You can restart auto baseline and set custom baseline values with the "Set Auto Baseline" button.



COM6, 115200

Red: 1000
IR: 1000

COM Port open successful.

Red Baseline: 935
Red Amplitude: 156
Red Average: 2058
IR Baseline: 999
IR Amplitude: 211
IR Average: 2028
BPM: 70
SpO2 %: 91

Figure 21. Screenshot of computer program to display real-time VTD data, including all SpO2 signals; the red baseline (thin red trace), the red signal (bold red trace), the IR baseline (thin blue trace) and the IR signal (bold blue trace).

The computer program gets new data by setting a timer with a period of 10 ms that calls an update function. This function checks the serial port receive buffer for new data. If it finds new data, it processes it and tells Windows that part of the graph needs updating by calling the system function "Invalidate" with the rectangle that needs updating. The update function repeats this process until there is no more data points in the buffer and ends. This process repeats when the timer is next activated. More processing is currently being done by the computer program. It also calculates the maximum and minimum in the signal so it can determine the amplitude of red and IR signals. Also, it uses the time when the maximums occur to determine the heart rate in BPM. The blood oxygenation percentage, SpO2%, is currently calculated in a very basic way based on information from the Freescale Appnote. We just take the ratio of red to IR signal amplitude and linearly interpolate between their empirically determined points; SpO2(0.4)=100%, SpO2(1)=85% and SpO2(3.4)=0%. Here we are counting on the red and IR baseline levels being approximately equal, but we could use their actual values in the calculation to be more precise.

Prototype Assembly

A plastic enclosure was 3D printed to house the circuit boards and give the prototype a more professional appearance. The ABS printed enclosure has 3 parts, an LCD bezel, an LCD holder, and a box as shown in Fig. 22. The LCD bezel and holder sandwich the LCD display. At the bottom of the box, the Propeller Platform circuit board was attached. The microMedic board plugs into the top of this board and then the touchscreen board plugs into the top of the microMedic board.

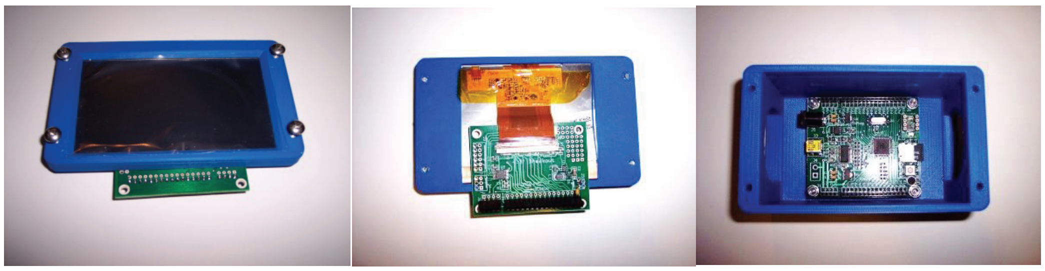


Figure 22. Photos of the three enclosure parts; LCD bezel and holder (left and center) and box (right).

Large holes in the sides of the box allowed for the SpO2 sensor, pressure sensor, USB and power cables to be connected as shown in Fig. 23. The result is a nice looking, portable assembly.

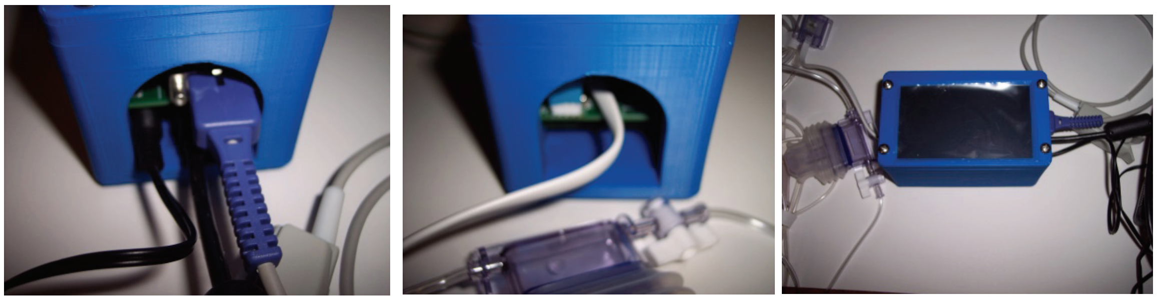
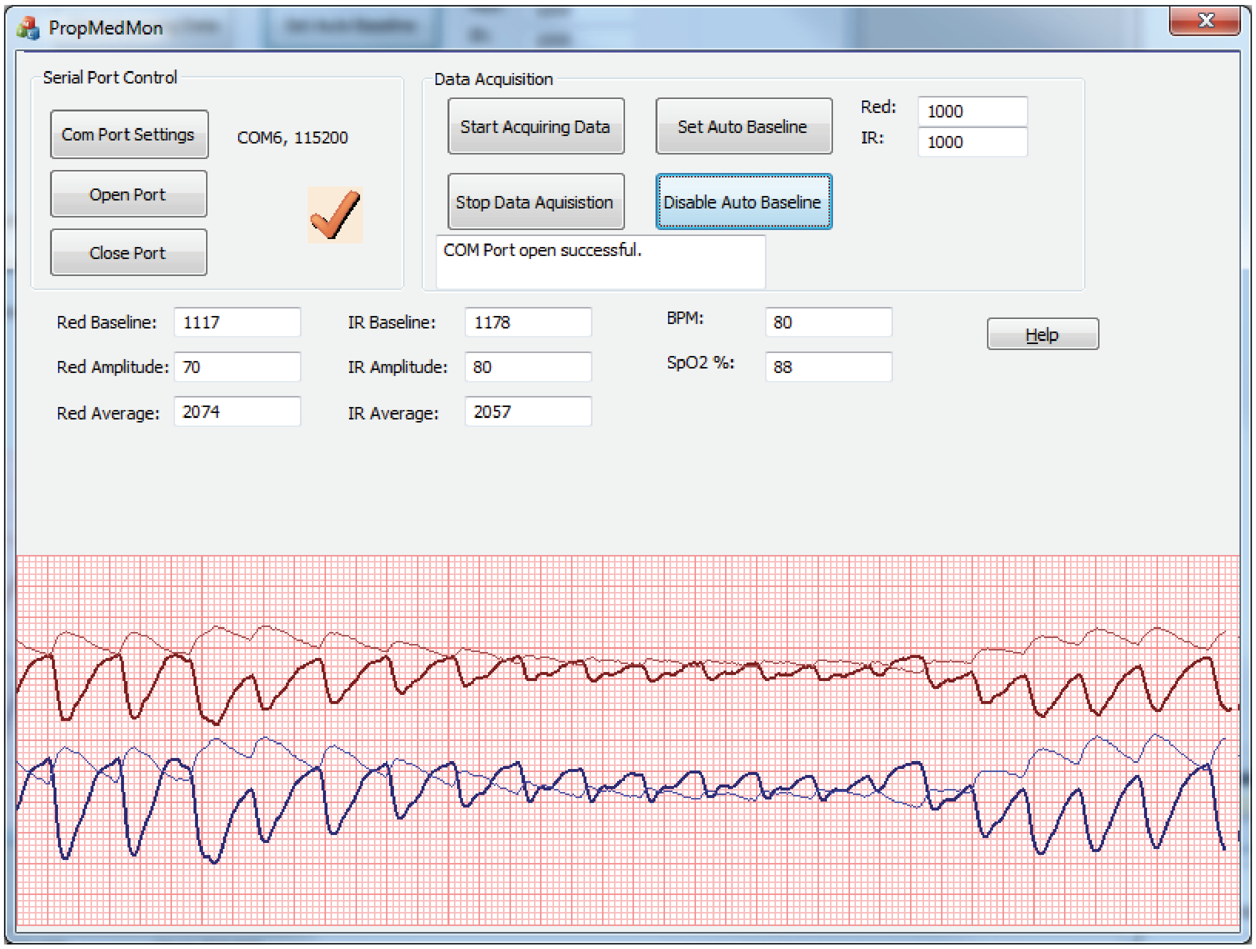


Figure 23. Photos of the connections through holes in the box (left and center) and the completed device (right).

Prototype Testing

We have just started initial patient tests with this device and so do not yet have data from heart failure patients to present. However, the device was extensively tested on ourselves. The Valsalva maneuver does not give very much response to the SpO2 baseline signal in healthy people. But, when we blow very, very hard at much higher pressures, we can see the baseline respond. A screenshot of the PC application when blowing at very high pressure is shown in Fig. 24. There you can see that the baseline trace first goes up slightly for two beats when blowing begins. Then, it drops noticeably below the original level. Also, you can see the SpO2 signal decreasing during this time. At the right, you can see baseline and signal levels returning to normal when blowing stops.



COM6, 115200

Red: 1000
IR: 1000

COM Port open successful.

Red Baseline:	1117	IR Baseline:	1178	BPM:	80
Red Amplitude:	70	IR Amplitude:	80	SpO2 %:	88
Red Average:	2074	IR Average:	2057		

Figure 24. Screenshot of Valsalva maneuver testing on a (hopefully) healthy human subject when blowing at a very high pressure level.

Conclusion and Future Work

We have successfully constructed a novel medical device, the VTD (Valsavla Test Device), which simultaneously measures SpO2 signals and air pressures while performing the Valsalva maneuver and displays information on a 4.3” touchscreen LCD. The VTD uses actual disposable sensors used at a hospital. The SpO2 pulse oxygenation circuit is specially programmed to suspend adjustment of the red and IR LED intensities during the Valsalva maneuver so that changes in the blood pressure can be observed. In addition to displaying SpO2 data, the LCD screen displays the pressure in an easy to read bar graph to provide the patient with feedback so that they can maintain the desired pressure for the test. The VTD can be used on its own, or be connected via USB cable to a computer where a program was written to display and analyze the data in more detail. Future work involves tested the device on patients to assess its efficacy and also making improvements in the device. Some improvements we would like to add are battery power and a data recording function.

Appendix 1. Propeller Pin Usage

The table below describes how the 32 I/O pins of the Propeller microcontroller are being used in the VTD. There are six pins still free, P10...P15.

I/O PIn	**Usage**
P0	uSD: MISO
P1	uSD: CLK
P2	uSD: MOSI
P3	uSD: CS
P4	uMedic: SwitchControl
P5	uMedic: PwmMcu
P6	uMedic: CSn
P7	uMedic: DIN
P8	uMedic: DOUT
P9	uMedic: CLK
P10	
P11	
P12	
P13	
P14	
P15	
P16	LCD: VSync
P17	LCD: HSync
P18	LCD: Blue0
P19	LCD: Blue1
P20	LCD: Green0
P21	LCD: Green1
P22	LCD: Red0
P23	LCD: Red1
P24	LCD: Backlight Control and DON
P25	LCD: Pixel Clock
P26	LCD: Data Enable
P27	LCD: Touch IRQ
P28	I2C: SCL
P29	I2C: SDA
P30	Serial: RX
P31	Serial: TX