ARCANE: A Navigational Aid for the Visually Impaired

Alex Davis | Anastasia Dodd | Abeer Javed
Lindsay Langford | Joseph Song | William Xie

TI Analog Design Competition
Spring 2014

Table of Contents

List of Figures

Project Design

Team ARCANE has successfully created a prototype navigation aid for the visually impaired. This product serves as a supplement to the white cane by detecting upcoming obstacles that could not be identified otherwise. Simply put, two sensors are attached to the user, which will identify upcoming objects based on their distance and direction. This information is then translated into a depth map, which is then sent to the haptic feedback system strategically placed on the forearm of the user and will vibrate with increasing intensity as the distance between an object and the user decreases.

Sensors

Our design uses two RGB cameras to estimate distance through triangulation. We use an Intel NUC as the main processor to generate depth maps in four steps: lens distortion, image rectification, stereo correspondence, and distance estimation through triangulation. The resulting depth data is then sent over to the haptic driver via serial protocol over Bluetooth.

Description:

The Intel NUC D54250WYK is a portable, lightweight Intel Core i5-4250U based development platform. We use it for all stages of image processing and communication with the haptic driver. The Logitech C310s were selected as the main sensors for our system. They are attached to a fixed platform to lock the relative orientation and distance between the two cameras. The C310s require no external power source and are connected directly to the NUC through two USB ports. The Intel NUC is powered by the main battery of our system and uses a version of Linux, Ubuntu 12.04 LTS, as its operating system.

Leveraging OpenCV, an open source computer vision library, our processor first represents raw image data as matrices. Then, lens distortion is removed by performing a transform on the image. The actual transformation is calculated by calibrating the cameras using an object of fixed geometry; an 8” x 6” chessboard was used for this purpose. The OpenCV calibration algorithm locates the corners of the chessboard and uses their locations to fix the distortion and calculate the focal length of each camera. This data is furthermore used to rectify the images; the translational and rotational relations between the two cameras are calculated from the same calibration data. The resulting left and right images are thus row aligned and rectified an important step in identifying the correspondences between points in each image.

These rectified images are then used to calculate the correspondence map via a semi-global block-matching algorithm. This method was selected due to its relatively light and parallelizable computational load, an important factor in attaining a real-time algorithm. The block-matching algorithm divides the two images into some number of blocks. For each block, the sum of absolute differences (SAD) was calculated. Using the SAD value for each block on the left image, the corresponding block is found by finding the block with the best matching SAD along the epipolar line to the left of the block's coordinate. To make the estimation more accurate, Mutual Information cost calculation was added to make the algorithm less sensitive to recording and illumination changes. In addition, constraints such as smoothness were added to penalize disparity changes and noise. The final costs were then used to select the corresponding blocks. The differences in pixels between each corresponding blocks form a disparity map.

This disparity map is used to calculate a depth map; using the disparity values in pixels (x_l - x_r), calculated focal length in pixels (f), baseline distance between the two cameras (T), each depth point was estimated using the equation presented in **Figure 1**. In **Figure 2**, the top portion is a pair of rectified images from the result of calibration. The green lines are selected epipolar lines used for the purpose of demonstration. In the lower right corner is a normalized depth map, with darker colors corresponding to lower distance values (black = 0). On the lower left is the rectified left image with nine overlaid sampling points.

Each of the nine samples is labeled by the average of multiple depth points surrounding it. The mapping between calculated and actual distances was completed by comparing measurements to calculated values. A best fit line is then used to model the relationship (as shown in **Figure 5**) and the gradient and intercepts of this line are used to calibrate the depth output.

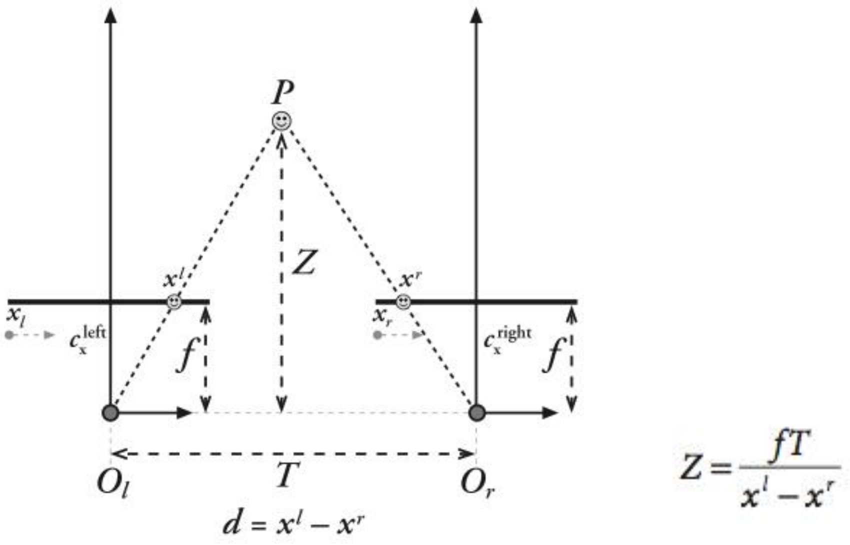


Figure 1: Distance estimation using similar triangles

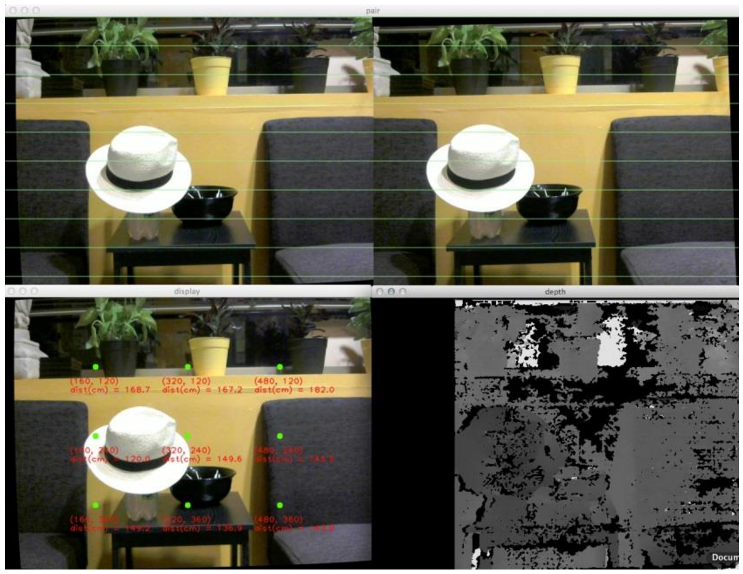

pair
display
depth
(160, 120) dist(cm) = 168.7
(320, 120) dist(cm) = 167.2
(480, 120) dist(cm) = 182.0
dist(cm) = 120.0
(320, 240) dist(cm) = 149.6
dist(cm) = 149.2
dist(cm) = 136.9
(480, 360)

Figure 2: Stereovision implementation

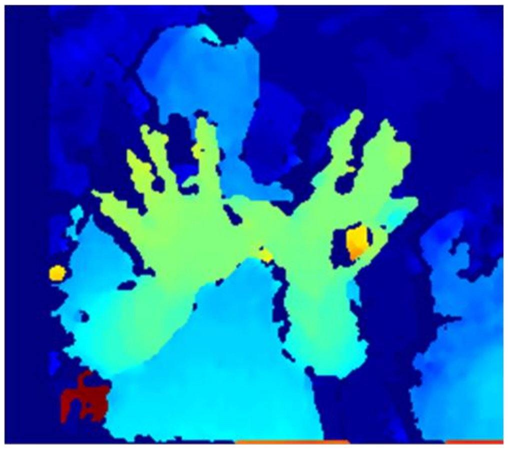


Figure 3: Color depth map

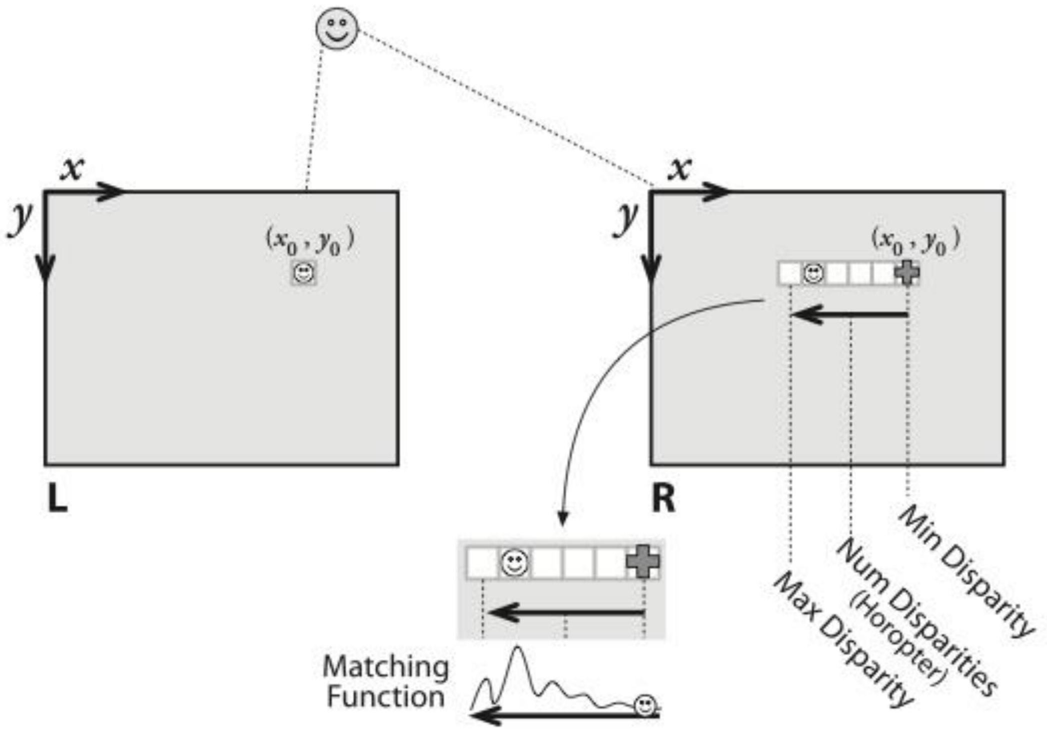


Figure 4: Block-matching algorithm for finding corresponding features

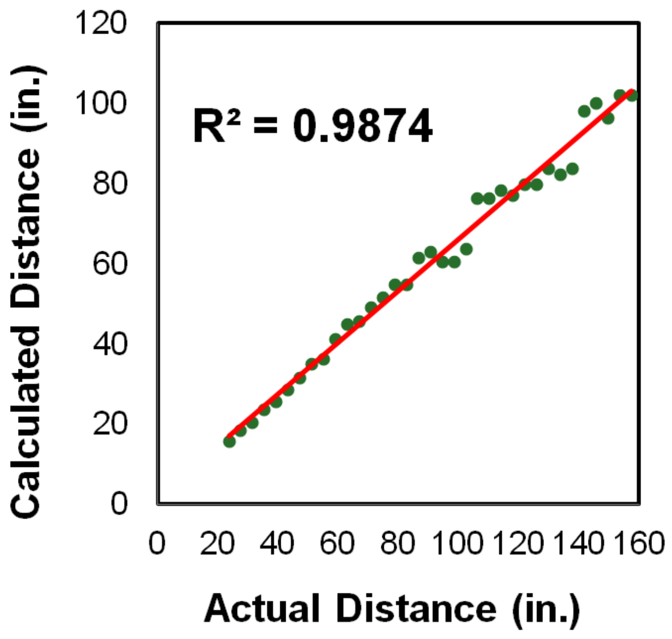


Figure 5: Best fit line modeling the relationship between the actual distance and the calculated distance

Camera Mount

While several locations were considered for the placement of the sensors, the final decision was that placement will be on the head of the user. The head was determined to be one of the body parts that moved the least while walking, and image stability is crucial for extracting depth maps from the user's environment.

The cameras are placed in a weather sealed case, which has two holes in the front to expose the lenses, and two in the back for wiring. This is then attached by Velcro onto the brim of a hat, which is worn by the user. It was determined that the weight of the camera case alone did not deform the structure of the hat, and it could be worn like any other cap. This allows for maximum control of the cameras, and does not obstruct the user's daily tasks.

A CAD model of the camera mount can be found in **Figure 6**:

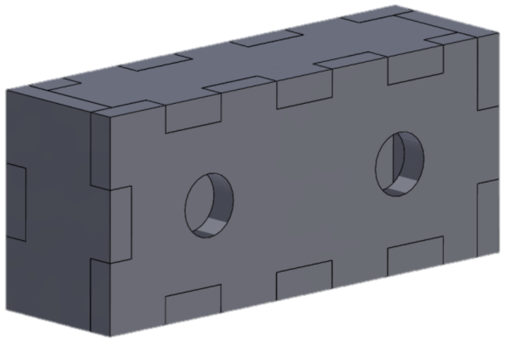


Figure 6: Case for the two cameras

Haptic Feedback

The goal of the haptic feedback portion of our design is to receive commands from the sensor subsystem's more powerful processor, translating and relaying them to the various vibrating motors. The motors are arranged in a 2x3 grid on top of the user's forearm. There is a specific motor numbering scheme which corresponds to the block matching points used in the depth map algorithm.

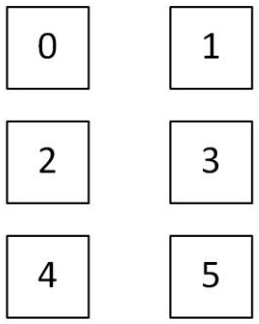


Figure 7: Top down view of the motor scheme

The haptic subsystem consists of four major components: the vibrating motors which provide the feedback, a motor driver which provides these with power, a compass module for navigation, and a microcontroller to handle communications between the various components.

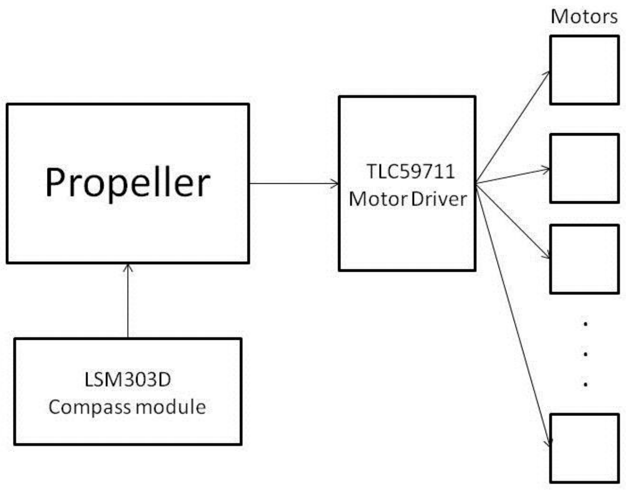


Figure 8: Components of haptic subsystem

Motor Driver:

The motor driver for our system is the TLC59711 from Texas Instruments. Its intended function is as an LED driver, but our motors have low enough current draw and voltage requirements that the TLC59711 is more than adequate. At 3.3 V, each pager motor requires only 40 mA of current to operate at a 100 percent duty cycle. Each channel on the motor driver can drive a current of up to 60 mA at 3.3V.

The only drawback to our use of the TLC59711 is its somewhat complex register structure. Since the chip is designed for use in LED displays and RGB lighting applications, it provides per-output PWM control as well as a global Red, Green, and Blue current scale. A handful of register bits control operation of the device, including PWM clock source (internal or external), pattern repeat, display blanking, and others. The first 12 16-bit registers control PWM output on each of the 12 output channels. The following 3 7-bit registers control maximum current available to each of the red, green, and blue groups. The remaining 5 bits control internal operation of the chip.

The TLC59711 receives this register data over an SPI-like 2-wire data bus consisting of a clock and data line. The device latches data on the clock signal's rising edge and may be operated at clock frequencies up to 10MHz. For transmitted data to be latched into the device's internal registers, an initial 6-bit write command must be sent by the Propeller ahead of the MSBs of the transmitted data.

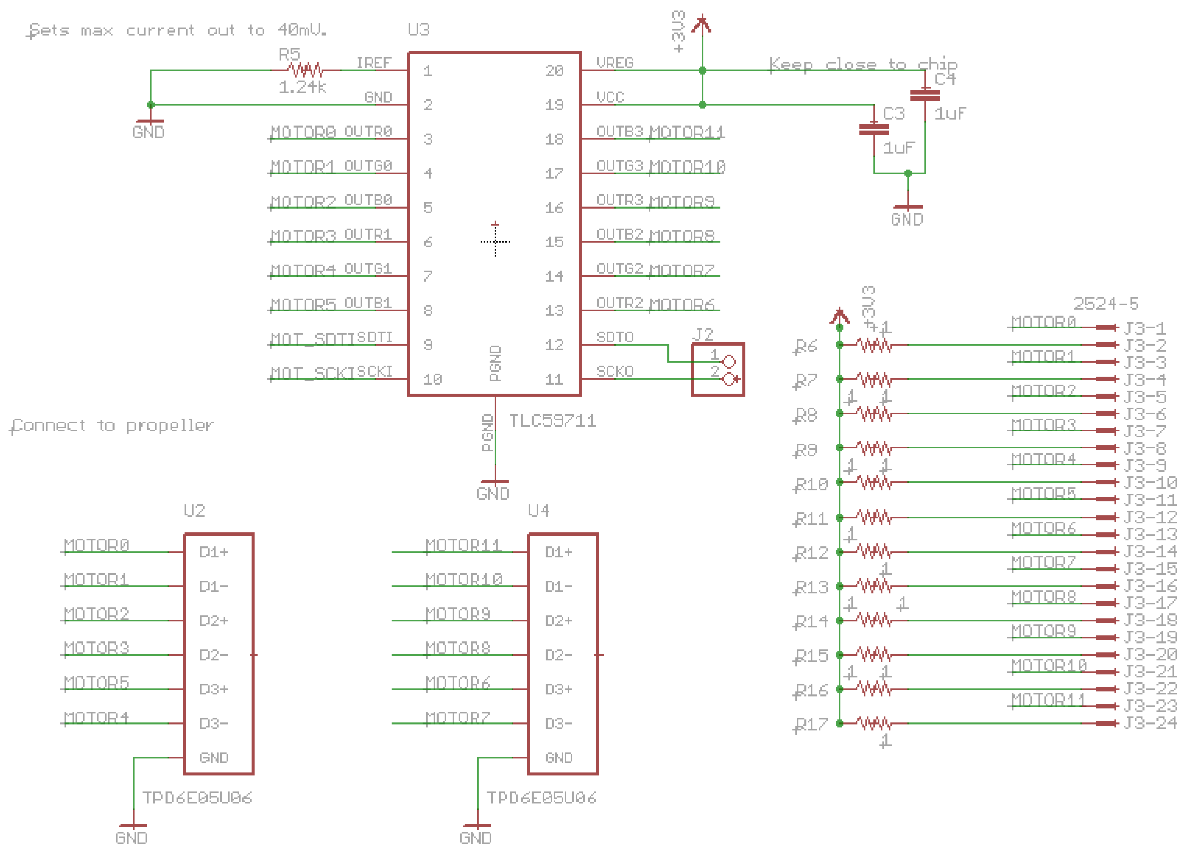


Figure 9: Schematic of motor driver circuitry

Processor:

To relieve some complexity from the code running on the stereo vision processor, as well as to make our system maximally platform-agnostic, C code to control the TLC59711 was developed for the Parallax Propeller microcontroller. The Propeller was selected due to its processing power, memory, and team members' prior experience with its use.

While Parallax does not provide native support for C code on the Propeller, there are a few open source compilers which do, Ross Higson's Catalina project and gcc being two examples. For this project, we chose Catalina due to its ease of use, integration with a full-featured IDE (CodeBlocks) generally good documentation, and compatibility with the Windows operating system.

Multicore Architecture:

At present, we are utilizing 7 of the Propeller's 8 cores to run code. At startup, a main function is launched in core 0. This in turn launches C code to run in four other cores before entering a continuous loop. Within this loop is code which facilitates inter-core communication between the other running cores. A visual overview of our software's operation is provided in **Figure 10**.

The first core launched by the main function is a terminal driver described in greater detail later on. This driver receives characters from a separate assembly-language RS232 driver running in another core, allowing the device to be controlled by a separate computer. The terminal code also provides a mutually-exclusive printf implementation allowing other cores to print debug information without contention.

The next core to be launched runs the compass code. This loops, checking a messaging variable which enables or disables compass updates. If compass updates are required, the compass code will compute the current heading, placing it into a second messaging variable to be read by the main loop. The compass code relies heavily upon floating-point math, so an additional core is launched to run an assembly-language floating-point math engine to accelerate computation. Additional information about the compass's operation is provided later in this document.

A sixth core is launched to run the motor driver code. The core of this code simply bit-bangs the SPI-like protocol of the TLC59711. This is done by a simple loop which shifts each successive bit of data out on the data line, waits for about 50us, pulls the clock high, waits for about 50us, then repeats the cycle with the next MSB to be transmitted, giving an effective data rate of 10kbps. The driver polls a status variable in memory, waiting for an update. When an update is indicated, the driver transmits the new values to the TLC59711.

The final core simply polls the front-panel buttons, indicating their state to the main loop. Dedicating a core to this task was necessary due to the Propeller's lack of hardware interrupts.

The main loop polls the terminal for updates. If updated data is present, it is copied locally. If the user has pressed the compass button, the compass activates for approximately 10 seconds, causing the main loop to merge both terminal-set motor levels and compass data before transmitting the updates to the TLC59711. Wherever possible, wait statements are used to reduce power consumption of cores, giving a power reduction over a naive busy-waiting implementation.

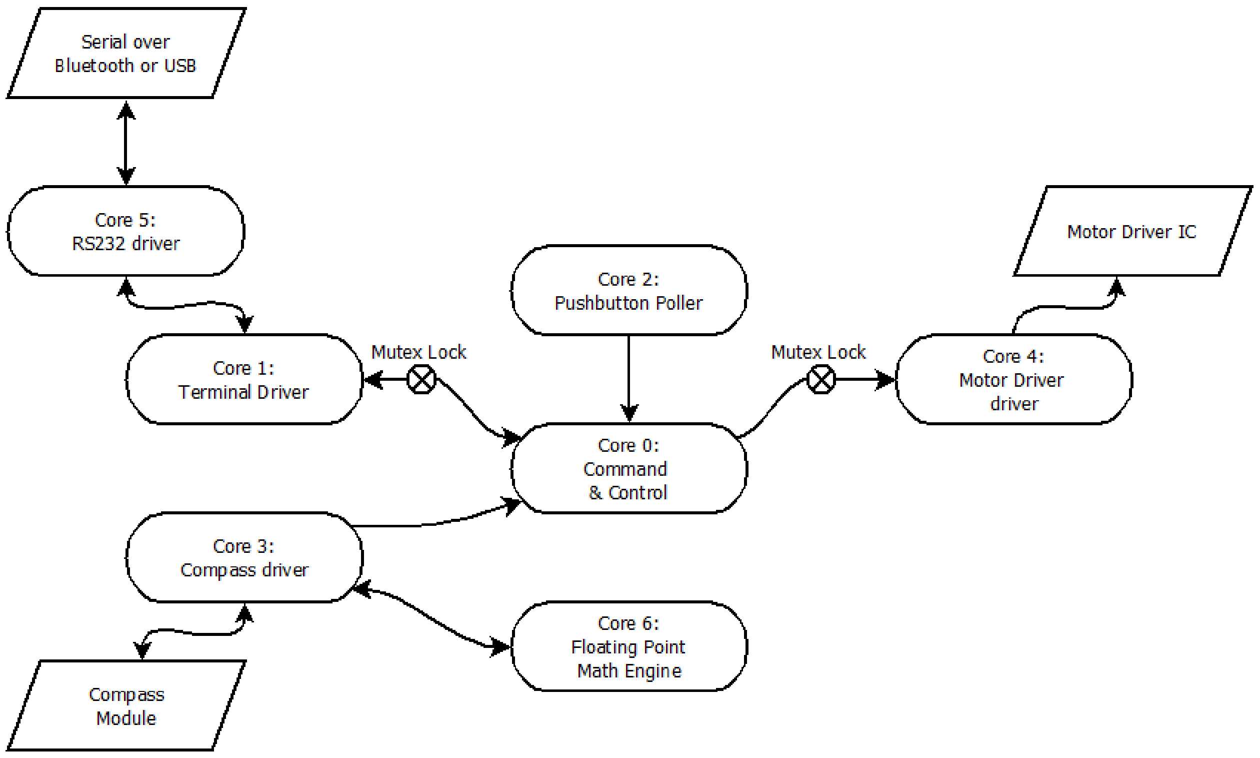

Serial over Bluetooth or USB
Core 5: RS232 driver
Core 2: Pushbutton Poller
Motor Driver IC
Core 1: Terminal Driver
Mutex Lock
Mutex Lock
Core 4: Motor Driver driver
Core 0: Command & Control
Core 3: Compass driver
Core 6: Floating Point Math Engine
Compass Module

Figure 10: Overview of multicore software implementation

Terminal:

In order to maximize compatibility across platforms, a TTY-like terminal was implemented on the Propeller, connected over a USB to serial converter to a controlling device (e.g. a PC running terminal software).

Terminal input is expected to be a single line (ending with a carriage return character) containing the command and up to two arguments separated by spaces. An example would be “s 65535 1”. This command is to set (s) channel 1 of the TLC59711 to level 65535 (full duty cycle).

Input to the terminal is parsed into three separate strings by replacing spaces within the string with null (0) characters and noting the substring start addresses. So, for the previous example, “s 65535 1\r” becomes the strings “s”, “65535” and “1\r” with \r denoting the carriage return character sent by the terminal. The last two of these strings are converted to integers using the C string library’s `atoi` conversion function. At present, the first string containing the command is assumed to be a single character long and is simply passed on to a switch-case statement to handle various command types, at which point the two (or fewer) arguments are handled appropriately.

At present, only three commands are supported, as follows:

Set - Usage: “s <intensity> [<channel>]”. Set can take one or two arguments. If followed by a single numeric argument, it sets all channels on the motor driver to the specified level (“s 0” will turn all motors off, for instance). If provided with two arguments, the command will alter the intensity of the channel specified by the second argument to the value specified by the first argument.

Mode - Usage: “m <live/blind>”. Mode allows the user to enable or disable the command line interface provided in the code. By typing “m live” feedback on commands will be provided. For example, when in live mode, typing “s 32767” results in the Propeller echoing back “All channels set to level 32767”. In cases when command-line feedback is undesirable (when the haptic controller is connected to the sensor system, for example), typing “m blind” will suppress these statements.

Two commands, “h” and “?” are provided to display a help message documenting all of these commands. Both of these override the mode set by the “m” command, printing an output regardless of whether command-line mode is enabled.

Upon receiving and parsing the command string, data to be sent to the TLC59711 is created by modifying the selected channel’s intensity value based on the provided arguments. All other data is left unmodified. This new data is then transmitted (along with existing unmodified channel states and configuration data) MSB first at around 10 kbps.

Vibratory Motor Coupons:

The coupon boards provide a mounting point for our vibrating pancake motors, providing strain relief for their fine (~28ga) wire leads. Each board is about an inch long and half an inch wide and has four components on it: the pager motor itself, a schottky diode, a .1uF ceramic capacitor, and a two-pin header. The diode is reverse-biased across the motor’s power leads and is intended to dissipate any inductive spikes caused by the motor commutating. The capacitor helps with this as well, smoothing out fast transients and switching noise which would otherwise cause problems for the motor driver. A male header provides a quick disconnect point for prototyping while keeping the coupon’s overall profile low.

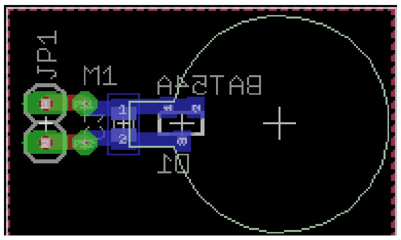


Figure 11: A coupon board layout

Compass Operation

The compass module we are using is actually a combination of an accelerometer and magnetometer in a single chip, the LSM303D designed by ST Microelectronics. This combination can be used to create a digital tilt-compensated compass, which allows us to give the user accurate compass heading information while they are wearing the device, regardless of how their arm is positioned.

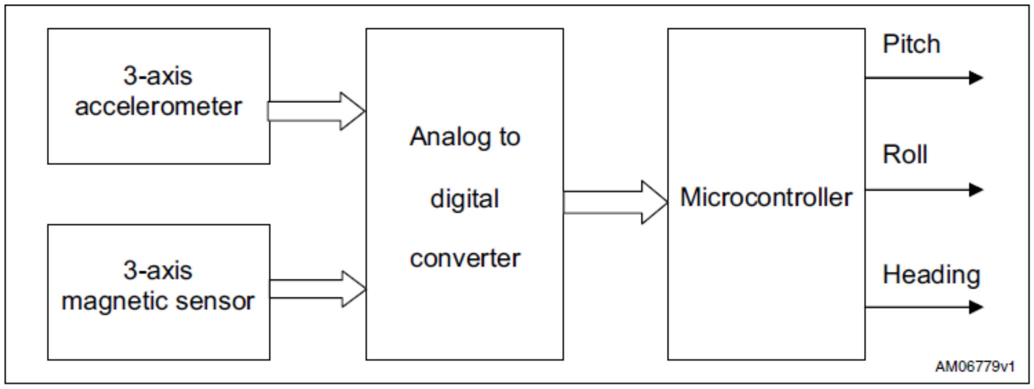


Figure 12: An overview of getting a compass reading from the LSM303D

The LSM303D already contains an analog to digital converter, so we can simply transmit data from the chip to the Propeller. We chose to use the standard 4-wire SPI protocol due to the ease of implementing it in software, its stability, and the fact that we have a plethora of free pins on the Propeller. The LSM303D is an SPI slave with a fairly simple communication scheme. SPI transmissions are 16 clock pulses long, with each pulse corresponding to a bit, and follow a set pattern. The first half of each transmission occurs over the master-out slave-in (MOSI) pin and is consistent for both reads and writes: The first bit indicates whether a read or write is taking place, the second is used to determine if the address should be incremented in multiple read/write commands or not, and the next six bits are used to indicate which register the operation is accessing. If the operation is a write, the next eight bits will be written by the Propeller to the LSM303D over MOSI, and if it is a read the compass will send the next eight bits over the master-in slave-out pin (MISO) to be received by the Propeller. A diagram of the overall protocol is below, with the MISO line being referred to as SDO and MOSI being called SDI. The chip select (CS) pin is held low to indicate that a transmission is in progress.

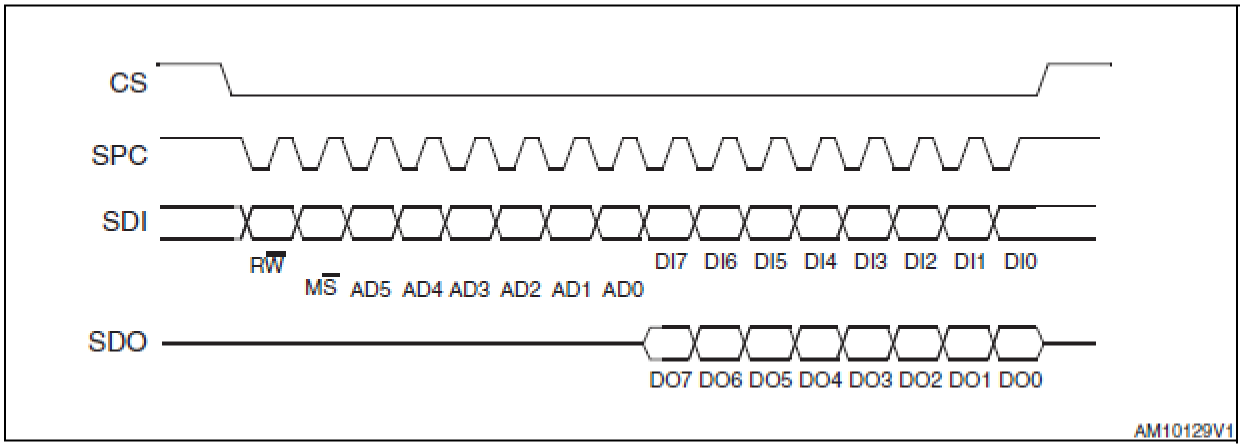

CS
SPC
SDI
RW
MS
AD5 AD4 AD3 AD2 AD1 AD0
DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
SDO
DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0
AM10129V1

Figure 13: Four line SPI on the LSM303D

After obtaining raw accelerometer and magnetometer data in the X, Y, and Z directions, the Propeller has to transform them into useful readings for pitch, roll, and heading, the definitions of which can be seen in the diagram below. Its equations can be found in the **Appendix:**

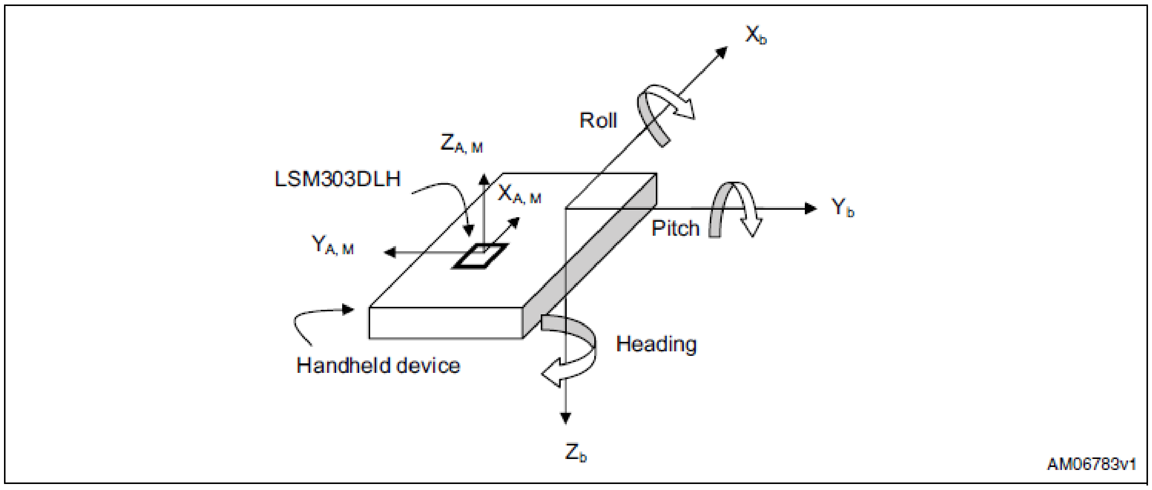


Figure 14: Coordinate system for compass readings from LSM303D

Integrated Haptic System

In order to make our product wearable, we combined many of the individual components of the haptic feedback system into a single custom printed circuit board.

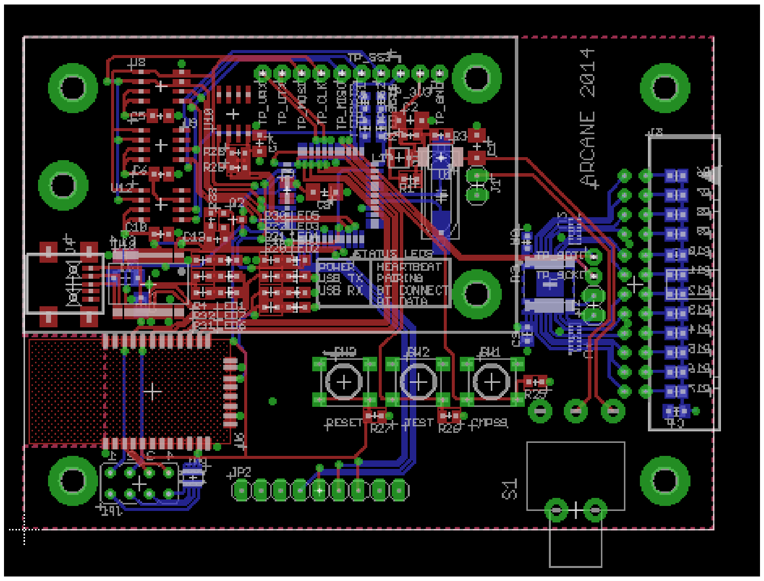


Figure 15: Integrated board

The Propeller and TLC59711 are both included on this board, and there is a 9-pin header for the pre-assembled compass module. The Propeller has the required 128 Kbit EEPROM for program storage, and we also include two 256 Kbit SRAM chips and a 4 Mbit Flash memory chip in order to provide additional storage should it become necessary. The SRAM and flash memory chips are addressed using 4-wire SPI protocol like the compass, and the EEPROM is addressed using I2C.

Our custom PCB also gives us two options to communicate serially with the NUC used by the sensor side of the system - over a USB cable or via Bluetooth. An FT232RL USB-to-serial converter sits between a USB mini B port and the Propeller, which allows for serial communications between the Propeller and a given computer over USB. An RN-42 Bluetooth module contains the entire Bluetooth stack within it, meaning that the Propeller can interface with it the exact same way it does with the FTDI chip - the only change we have to make to use Bluetooth is changing the pins used for serial communication. The code currently supports using only one of the serial communication methods at a time, and the decision about which to use must be made when compiling the embedded code for the Propeller.

Local power management for the haptic PCB is provided by a TPS62160 buck converter from Texas Instruments. We chose this particular IC because it can accept an input voltage in the range of 3 - 17 V, which means our battery voltage of 6.4V falls well within spec, and because it can provide up to 1A of current, which is suitable for our system's needs. The motors are the most current-demanding aspect of our project, and they only demand 40 mA of current each, or 240 mA total. Each core in the Propeller typically draws current on the order of uA, and the other ICs in our system typically draw tens of mA at maximum. For example, the FT232RL draws 15 mA in normal operation, the RN-42 draws 3 mA when connected and 30 mA when transmitting, and the TLC59711 draws under 32 mA. We set the TPS62160 to output a constant 3.3V.

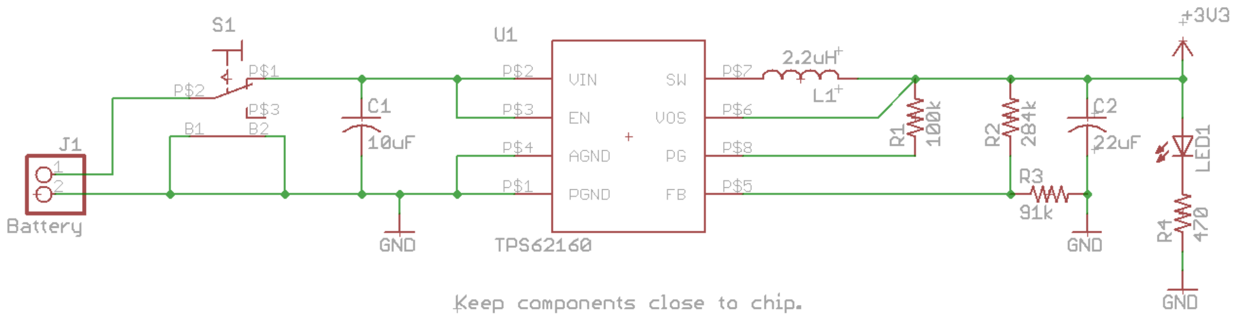


Figure 16: Power management schematic

Battery Management

To simplify use of the device and prevent the user from having to change batteries on-the-go, an integrated lithium-iron-phosphate (LiFePO4) battery charger was designed. LiFePO4 batteries were selected primarily for safety reasons, as their less-volatile chemistry is more tolerant of abuse than the slightly more compact and more energy-dense lithium-polymer chemistry.

To give ample runtime without causing the 3.3V regulator on the integrated haptic board to enter a brownout condition, a 2-cell (6.4V working voltage) Lithium Phosphate battery was chosen. This pack is capable of supplying 1200mAh at a peak discharge rate of 4.2A and is internally protected against overcharging (Vbat > 7.6V), over-discharging (Vbat < 4.46V), and short circuits. Even at our worst-case peak current demand of 500mA, a 1200mAh battery should last a healthy 2.4 hours. Under real-world usage, this can easily stretch much farther.

The core of the battery charger is the Texas Instruments bq24630 stand-alone synchronous switch-mode lithium phosphate battery charger IC. The bq24360 was chosen due to its support for lithium phosphate packs of up to 4 series cells with no need for supervisory circuitry or microcontroller monitoring. This allowed us to design a fully independent battery management board, segregating our design into two easily debuggable sections. The only real drawback to the bq24630 is its lack of internal MOSFETs, instead requiring 3 discrete P-channel SI7617DN MOSFETs and 2 discrete N-channel SIS412DN devices. While these are bulky, they do support ~35A continuous drain currents, adding a short-circuit safety margin to our design.

The bq24630 is essentially a buck-boost regulator with sensitive current monitoring. Typical battery charging for lithium phosphate batteries consists of two phases: an initial constant-current phase and a final constant-voltage phase. During the constant-current phase, battery pack voltage is monitored. Once the pack voltage reaches its final value, the charger enters the second phase, maintaining a constant voltage across the pack and monitoring the charging current. Once this current drop below a set threshold, the battery is considered charged and the charge cycle is terminated. In our application, the constant-current phase is limited to 1A, the maximum pack voltage is set to be 7.2V, and charging is terminated once charge current drops below 100mA. These values were selected to be well within safety margins of our chosen battery pack. During the entire charging process, pack temperature is monitored via an attached thermistor. If the pack reaches an unsafe temperature, charging is terminated to prevent fire or pack damage.

The bq24630 is able to switch attached loads between wall power and battery. When an external power supply is connected, the bq24630 disconnects the battery before switching to external power. This break-before-make switching prevents inadvertent damage to the battery from excessive shoot-through current and is mirrored when the charger is disconnected from external power (i.e. MOSFET for external power is turned off before battery MOSFET is turned on). The bq24630 is also able to limit current draw from the external supply, throttling its own charging current if necessary to do so. Our charger is designed to operate from a 12V 2A wall-wart type supply, so a 2A limit was set on the bq24630 to remain within the wall-wart's capabilities.

PCB Case

To house the custom circuit boards and battery, Team ARCANE has created a 3D printed PCB case which will be worn on the bicep of the user. This case was 3-D printed to maximize the precision of our design, since we are dealing with a PCB that requires a near perfect

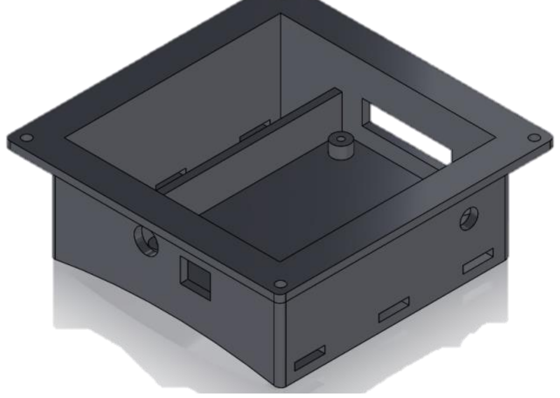


Figure 17: PCB Case

fit. The case is divided into two compartments – one for holding the PCB into place, and another for isolating the heat of the battery. Additionally, there are four holds for components of the PCB: the connector, power switch, USB port, and DC input port.

Arm Sleeve

The arm sleeve is made of flannel, to minimize coupling between motors, and Velcro, to make it adjustable to all arm sizes and levels of comfort as well as to allow the easy removal and replacement of pager motors if necessary. Additionally, the soft material allows for the vibrations of the individual motors to be felt in specific parts of the user's forearm. This is unlike a rigid sleeve, which would make the entire forearm vibrate with a slight vibration in one specific area. The layout for the sensors was determined through thorough testing, as described in the **Testing and Results** section. A picture of the arm sleeve can be found in **Figure 18:**

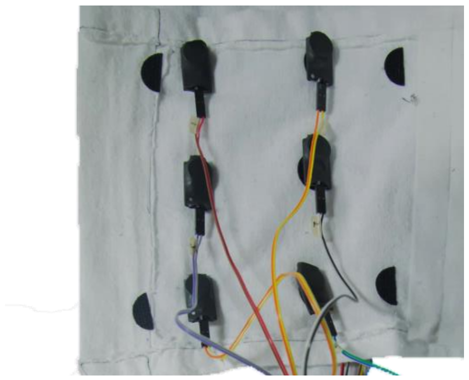


Figure 18: Arm sleeve with a 3x2 motor layout

Sensor Testing

For our sensor testing, we need to ensure that we can accurately translate the environment into a depth map. Then that depth map needs to be converted into inputs to the haptic feedback system of our design. To that end, we tested our depth map generation in a number of different environments to ensure that they are portrayed correctly. For each of these different environments, we changed the position of one item after the initial depth map, to ensure that the algorithm can detect changes when most things in the environment stay constant. Further, to ensure that the sensor correctly identifies objects at all the sizes and distances we set out in our design criteria, we will placed objects of various sizes and at a variety of distances in all of our testing environments. This means that we included obstacles as low as a foot off the ground and as high as seven feet, along with obstacles as far away as twelve feet.

We tested our sensors by shrinking a piece of brown, square packaging paper down an inch at a time to see if the sensors would still identify it as a distinct object.

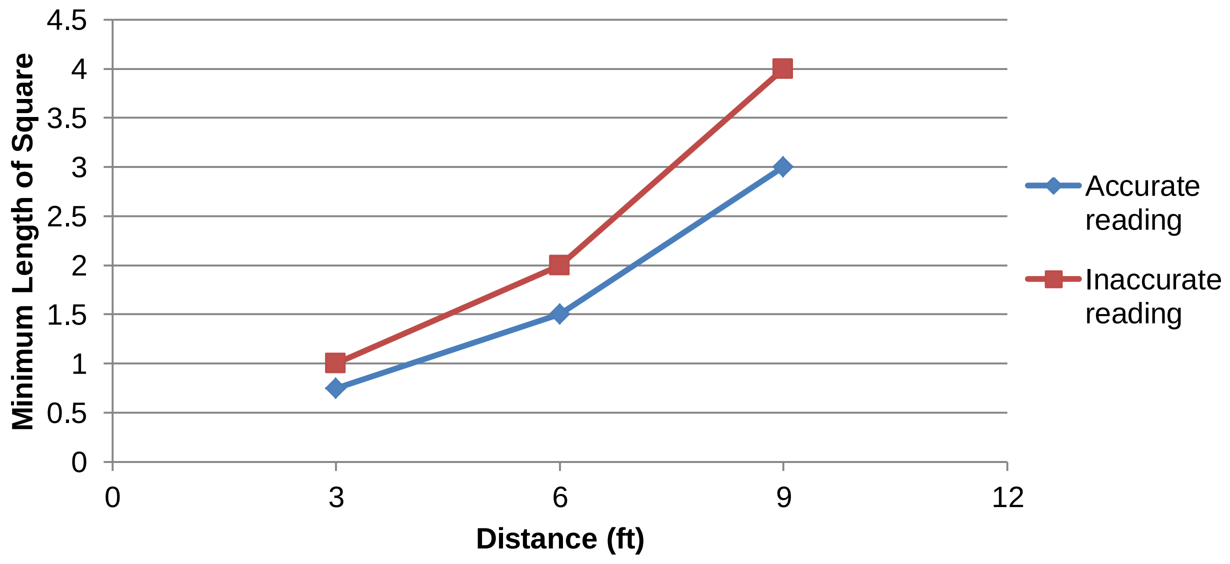


Figure 19: Minimum length of square with reading versus distance from sensors

With a block size of 3x3 pixels giving a resolution of 160 x 120 pixels, the farther the object is from the sensors, the smaller the area of the object. Only when the object is very large, or very close to the 3x3 pixel size is the distance accurate. Anything smaller causes the value to change, but with the cost function and other noise making the data point inaccurate in finding the corresponding distance.

Haptic Testing:

There are two main focuses for the haptic testing. Firstly, we need to determine the best layout for our haptic vibration on both or only one of the hands or arm. We need to ensure that each vibration is felt as distinct from the others and that the intensity at each location is comfortable and safe for the user. Secondly, we need to ensure that the vibrations we feed back to the user can be interpreted into meaningful information about the environment. For example, the vibrations corresponding to a tree branch need to be understood as indicating an overhead obstacle instead of a wall or some other lower-height object.

"Virtual" Obstacle Course

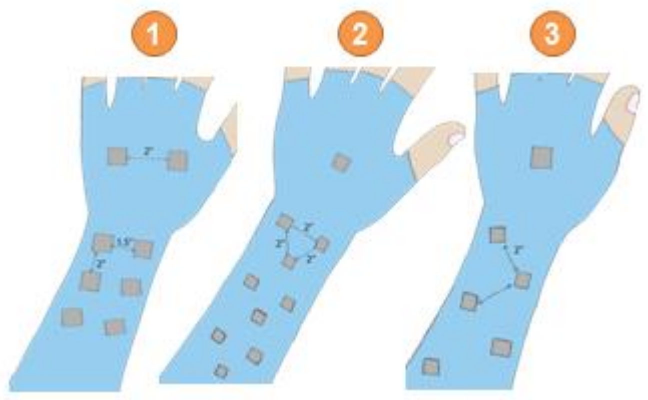


Figure 20: Three unique layouts for haptic feedback

To ensure that we have successfully accomplished these two goals, we tested a number of layouts and run the user through a virtual obstacle course. Because there will be a learning curve for each layout, we will run each virtual obstacle course multiple times in order to get an accurate read on how intuitive the layout really is. We will average the percentage of obstacles correctly identified for the last three of five obstacle courses for each layout and compare the accuracy count and learning curve for each layout in order to determine our best layout option.

In order to meet our design specifications, our virtual obstacle course must include the haptic feedback cues for an obstacle to the left, to the right, overhead, down low, and moving closer. To ensure that our test subjects are learning the haptic cues for each layout instead of memorizing the order of the virtual obstacles, we will vary the order of the obstacles in the virtual obstacle course. If at any point the vibrations become uncomfortable or indistinguishable, we know that we need to scale back the intensity of the vibrations or that the layout is unusable.

As we began running our tests, we discovered that the three by three layout of pager motors (Layout #2) was an overwhelming amount of information and it was impossible to distinguish distinct motors' locations. The motors were too close to identify which one was vibrating, which further confused the mental image of the environment we were trying to create, especially when multiple motors were running at the same time. Keeping this in mind, we decided to rule Layout #2 out, and continued testing with Layouts #1 and #3.

From there, the purpose of the rest of the testing was to determine whether the first or the third layout would be clearer: whether it is easier to understand the haptic feedback when the right and the left are even with each other or offset from each other.

The results from the four users are as follows:

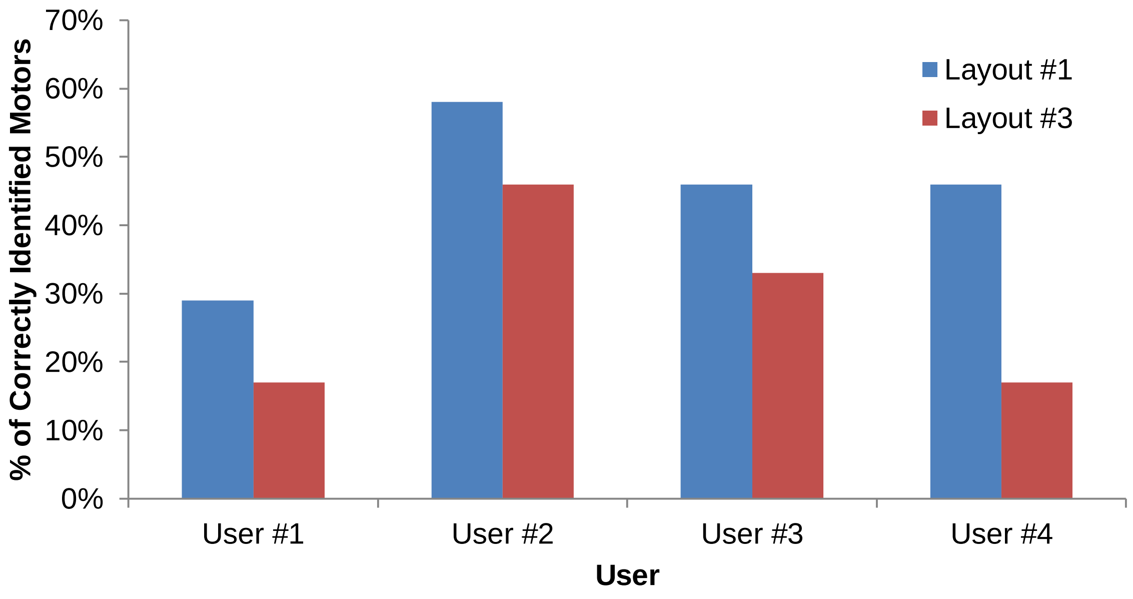


Figure 21: Comparison in pager motor differentiation between Layouts #1 and #3

From a comparison of the average success rates, Layout 1's accuracy rate of 45% was much better than Layout 3's 28%. From this preliminary testing, it appears that Layout 1 is a better choice for our purposes.

Integrated Prototype Testing

When we completed our integrated prototype, we began testing on blindfolded individuals. The individual was put into the arm sleeve according to how tight was most comfortable for them while the sensors were left attached to a monitor so we could see which of the 6 regions detected an obstacle. Then, before the individual was blindfolded, we explained the layout of the motors: which would indicate an obstacle to the right vs. left, overhead, waist-height or ground-level. We explained that only one would be on at a time, so if more than one was on at a time, it was our mistake and not to bother trying to figure out which motors were vibrating. Additionally, we reassured test subjects that the inability to identify the motor vibrating or a poor score was not a reflection on them, but on us and our ability to communicate information to them as a user. The individual was then blindfolded and the testing began. The only training our test subjects received was that we stimulated each of the six regions of the arm sleeve and told them which region was being triggered in order for them to feel where each of the six regions were. This only lasted around ~2 minutes.

We stimulated specific regions of the arm sleeve by holding a box lid in the region of the sensors that would trigger the specific region we wanted. We did each of the six regions four times for each test subject in a random order. The results were as follows:

		Motor Accuracy (Correct / Total)						Overall Accuracy
		Top Left	Top Right	Middle Left	Middle Right	Bottom Left	Bottom Right	
User #1	Correct	5	4	4	4	3	3	92%
	Total	5	4	4	4	4	4	
User #2	Correct	3	3	4	3	3	3	73%
	Total	4	5	4	4	5	4	
User #3	Correct	4	3	2	1	1	4	63%
	Total	4	4	4	4	4	4	
User #4	Correct	4	2	4	1	3	1	60%
	Total	4	4	4	4	4	5	
User #5	Correct	4	4	3	3	1	4	79%
	Total	4	4	4	4	4	4	
User #6	Correct	3	4	3	2	4	4	83%
	Total	4	4	4	4	4	4	
User #7	Correct	3	5	5	2	4	4	92%
	Total	3	5	5	4	4	4	

Figure 22: Accuracy of vibration layout by user

In order to interpret these results more easily, we can also look at an accuracy map:

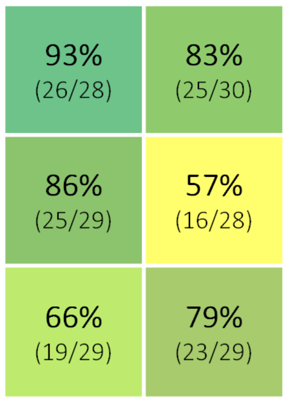


Figure 23: Percent accuracy of vibration layout

Our vibration layout resulted in a 78% success rate of correctly identifying the location of an object in front of the user. Many users commented that it became easier to identify the motor vibrating over time as they became more familiar with the device. A few commented specifically that it was more difficult to identify the motor vibrating when it had been vibrating for a long time as they felt that they got used to the vibrations and didn't notice them anymore.

The sensor and NUC subsystem has a battery life of about 5 hours, more than enough to sustain a user throughout their daily movement. We tested the battery life of the sensors and NUC by timing how long it took for the battery to die; thus we have a very accurate measurement of its battery life. Unfortunately, it was not possible to test the haptic subsystem's battery in this manner. However, we measured the average power load of the system at 0.88W. The battery has a 7.2Wh lifespan, which we calculate to give us about an 8 hour expected battery life. While we have not been able to fully test the battery life of the haptic subsystem, we believe that the haptic battery life is significantly more than the sensor and NUC battery life, such that the sensor and NUC battery life will always be the limiting factor.

Summary & Recommendations

As of 4/20/2014, the ARCANE is a fully functional prototype.

The current status of our design:

- Sensors are fully functional and housed in an acrylic case placed on the brim of a hat.
- NUC is fully functional with our stereovision algorithm and housed in a backpack.
- The haptic controller is fully functional and housed in a 3D printed case fastened on the bicep of the user. Additionally, the haptic controller receives Bluetooth information from the NUC without error and can successfully receive information from the compass.
- The pager motors are fully functional and correctly receive information from the haptic controller.

Overall, our design accomplishes its major objectives. We are able to detect objects at a distance of 9 feet as well as at a range of heights.

Things we might have done differently include:

- Using a different microcontroller than the Parallax Propeller, since it is more of a development tool rather than a final product
- Implementing and experimenting with a different feedback location, such as the hand
- Using discrete pulses rather than continuous vibration for haptic feedback
- Increasing the frequency of pulses of vibrations rather than intensifying vibrations to indicate distances

While the core of our design works, there are changes we would make in hindsight to make the work easier and the product more intuitive for visually impaired users. Many visually impaired users felt that a constant vibration became difficult to understand after a time, recommending instead that we change the frequency of pulses of vibrations instead of intensifying a constant vibration.

Future work on the ARCANE might include:

- Object recognition for emergency alerts about fast moving objects or stepping off a curb
- Use of wider angle cameras for wider field of view and more information
- Use of wireless cameras or a different form factor such that there are no cords between sensors and the NUC
- Use of lighter, more compact components

Despite our progress this past year, there is much left to do in order to make a fully fleshed out supplement. Many visually impaired users recommended object recognition for stepping off a curb and an alert to warn the user about a fast moving object such as a bicycle wilI increase the usefulness of our product. Additionally, our current prototype is in four parts that the user must put on separately. Future work might condense these four parts into fewer parts and make them lighter and more wearable. Other improvements would increase the durability, wearability, and usefulness of our product for visually impaired users.

References

[1] R. A. Jarvis, “A Perspective on Range Finding Techniques for Computer Vision,” IEEE Trans. Pattern Anal. Mach. Intell, vol. 5, no. 2, pp. 122-138, Mar. 1983. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4767365

[2] S. Birchfield and C. Tomasi, “Depth Discontinuities by Pixel-to-pixel Stereo,” Interational Journal of Computer Vision, vol. 35, no. 3, pp. 269-293, Aug. 1999. http://www.ces.clemson.edu/~stb/publications/p2p_ijcv1999.pdf

[3] D. Scharstein and R. Szeliski, “High-Accuracy Stereo Depth Maps Using Structured Light,” Microsoft Corp., Redmond, WA, 2003. http://research.microsoft.com/pubs/75606/scharstein-cvpr03.pdf

[4] A. Levin et al, “Image and Depth from Conventional Camera to Coded Aperture,” M.I.T. Comp. Sci. and Arti. Intell. Lab, Cambridge, MA, 2007. https://graphics.stanford.edu/courses/cs448a-08-spring/levin-coded-aperture-sig07.pdf

[5] M. Grosse, “Coded Aperture Projection,” ACM Transaction on Graphics. vol. 29, no. 3, pp.1-19, 2010. http://web.media.mit.edu/~gordonw/publications/CodedApertureProjectionTOG.pdf

[6] L. Jones et al. Vibrotactile pattern recognition on the arm and back. Perception. 52-68, 2009.

[7] V. Hayward and K. MacLean. “Do-It-Yourself Haptics, Part I.” IEEE Robot. Automat. Mag. 88-104, 2007.

[8] K. MacLean and V. Hayward. “Do-It-Yourself Haptics, Part II: Interaction Design.” IEEE Robot. Automat. Mag. 104-119, 2008.

[9] L. Jones and M. Berris “The Psychophysics of Temperature Perception and Thermal-Interface Design.” Proc. of the 10th Symp. Haptic Interfaces for Virtual Environments and Teleoperator Systems, 2002 : IEEE Computer Society. http://midas.inrf.uci.edu/groups/bats/uploads/Xsense/Jones_HIfVEaTS_2002.pdf

[10] R. Sodhi et al. 2013 “AIREAL: Interactive Tactile Experiences in Free Air” ACM Trans. Graph. 32, 4, Article 134 (July 2013), 10 pages. DOI = 10.1145/2461912.2462007 http://doi.acm.org/10.1145/2461912.2462007

[11] S. Kim et al. 2013 “Tactile Rendering of 3D Features on Touch Surfaces” Proc. 26th annua. ACM symp. User interface software and technology pp.531-538. DOI = 10.1145/2501988.2502020 http://doi.acm.org/10.1145/2501988.2502020

[12] G. Robles-De-La-Torre. "International Society for Haptics: Haptic technology, an animated explanation". [Online]. Available: http://Isfh.org.

[13] C. Hogan. “Analysis of blind pedestrian deaths and injuries from motor vehicle crashes, 2002-2006.” Direct Research, LLC., Vienna, VA. 2008 http://files.meetup.com/211111/analysis-blind-pedestrian-deaths.pdf

[14] A. Mandal. “What is visual impairment?” [Online]. Available: http://www.news-medical.net/health/What-is-visual-impairment.aspx

[15] World Health Organization (WHO). (2013, October). “Visual impairment and blindness.” [Online]. Available: http://www.who.int/mediacentre/factsheets/fs282/en/

[16] D. Héiligh. (2012, June). “Review of the ultracane.” [Online]. Available: http://www.digitaldarragh.com/2012/06/20/review-of-the-ultracane/

[17] S. Hoefer. (2011, August). “Meet the tacit project: it’s sonar for the blind.” [Online]. Available: http://grathio.com/2011/08/meet-the-tacit-project-its-sonar-for-the-blind/

[18] E. Berdinis. (2012, October). “Kinecthesia: haptic belt for the blind.” [Online]. Available: http://www.kinecthesia.com/

[19] M. Kendrick. (2009, August). “Tasting the light: device lets the blind ‘see’ with their tongues.” [Online]. Available: http://www.scientificamerican.com/article.cfm?id=device-lets-blind-see-with-tongues

[20] Texas School for the Blind and Visually Impaired. (2010, February 19). “Visual Impairment - What is it Like?” [Online] Available: http://www.tsbvi.edu/instructional-resources/1912-visual-impairment-what-is-it-like

[21] ST Microelectronics, “Ultra compact high performance e-Compass 3D accelerometer and 3D magnetometer module,” LSM303D datasheet, Nov. 2013.

[22] ST Microelectronics, “Using LSM303DLH for a tilt compensated electronic compass,” Application Note AN3192, Aug. 2010.

Images: *(Figures not listed were created by Team ARCANE)*

Figure 1: http://robotica.dc.uba.ar/public/papers/ras2012.pdf

Figure 4:

http://217.92.194.243/fws2010/presentation/FWS16_SimGeoColCamCalib_Kapusi_ZBS_Slides.pdf

Figure 11:

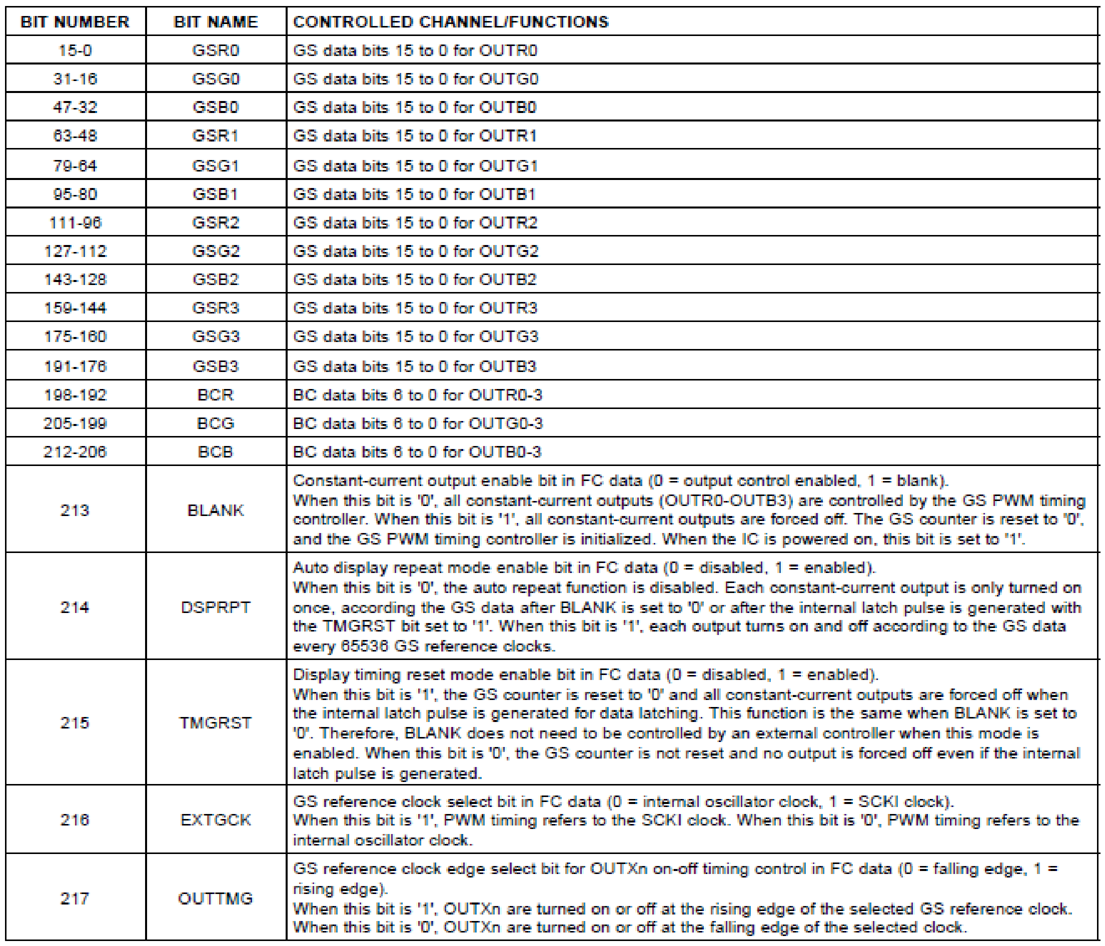
http://www.st.com/web/en/resource/sales_and_marketing/presentation/product_presentation/Smart_street_lighting_marketing_pres.pdf

Figure 12:
http://www.st.com/web/en/resource/sales_and_marketing/presentation/product_presentation/Smart_street_lighting_marketing_pres.pdf

Appendix

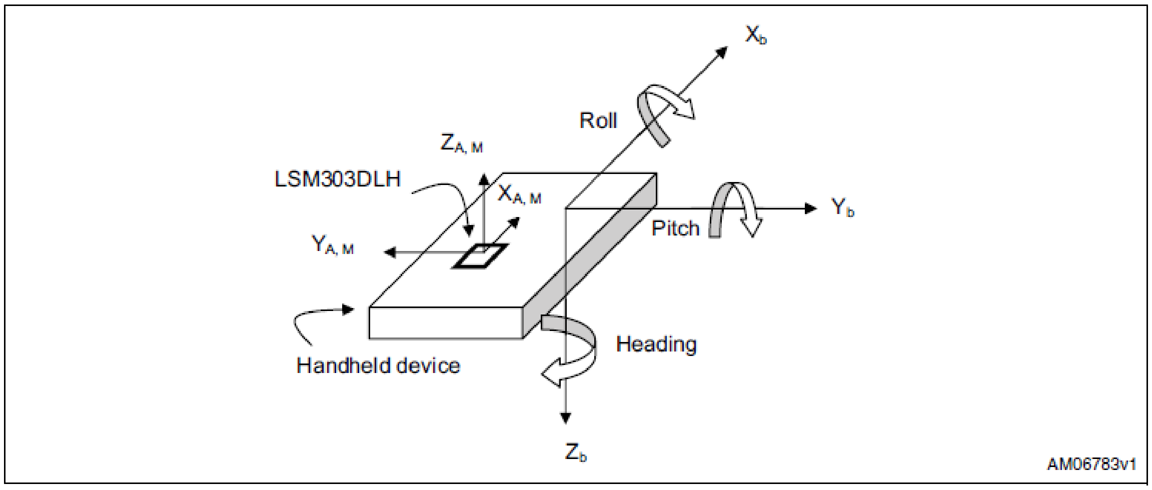
Appendix A: Table from the datasheet of the TLC59711, detailing its register structure

Table 5. Data Latch Bit Assignment



BIT NUMBER	BIT NAME	CONTROLLED CHANNEL/FUNCTIONS
15-0	GSR0	GS data bits 15 to 0 for OUTR0
31-16	GSG0	GS data bits 15 to 0 for OUTG0
47-32	GSB0	GS data bits 15 to 0 for OUTB0
63-48	GSR1	GS data bits 15 to 0 for OUTR1
79-64	GSG1	GS data bits 15 to 0 for OUTG1
95-80	GSB1	GS data bits 15 to 0 for OUTB1
111-96	GSR2	GS data bits 15 to 0 for OUTR2
127-112	GSG2	GS data bits 15 to 0 for OUTG2
143-128	GSB2	GS data bits 15 to 0 for OUTB2
159-144	GSR3	GS data bits 15 to 0 for OUTR3
175-160	GSG3	GS data bits 15 to 0 for OUTG3
191-176	GSB3	GS data bits 15 to 0 for OUTB3
198-192	BCR	BC data bits 6 to 0 for OUTR0-3
205-199	BCG	BC data bits 6 to 0 for OUTG0-3
212-206	BCB	BC data bits 6 to 0 for OUTB0-3
213	BLANK	Constant-current output enable bit in FC data (0 = output control enabled, 1 = blank). When this bit is '0', all constant-current outputs (OUTR0-OUTB3) are controlled by the GS PWM timing controller. When this bit is '1', all constant-current outputs are forced off. The GS counter is reset to '0', and the GS PWM timing controller is initialized. When the IC is powered on, this bit is set to '1'.
214	DSPRPT	Auto display repeat mode enable bit in FC data (0 = disabled, 1 = enabled). When this bit is '0', the auto repeat function is disabled. Each constant-current output is only turned on once, according the GS data after BLANK is set to '0' or after the internal latch pulse is generated with the TMGRST bit set to '1'. When this bit is '1', each output turns on and off according to the GS data every 65536 GS reference clocks.
215	TMGRST	Display timing reset mode enable bit in FC data (0 = disabled, 1 = enabled). When this bit is '1', the GS counter is reset to '0' and all constant-current outputs are forced off when the internal latch pulse is generated for data latching. This function is the same when BLANK is set to '0'. Therefore, BLANK does not need to be controlled by an external controller when this mode is enabled. When this bit is '0', the GS counter is not reset and no output is forced off even if the internal latch pulse is generated.
216	EXTGCK	GS reference clock select bit in FC data (0 = internal oscillator clock, 1 = SCKI clock). When this bit is '1', PWM timing refers to the SCKI clock. When this bit is '0', PWM timing refers to the internal oscillator clock.
217	OUTTMG	GS reference clock edge select bit for OUTXn on-off timing control in FC data (0 = falling edge, 1 = rising edge). When this bit is '1', OUTXn are turned on or off at the rising edge of the selected GS reference clock. When this bit is '0', OUTXn are turned on or off at the falling edge of the selected clock.

Appendix B: Calculating the Compass Readings from an LSM303D



Let Mx, My, and Mz be magnetometer readings, and Ax, Ay, and Az be accelerometer readings. It can be shown that in order to get pitch and roll from the raw accelerometer readings, the following calculations can be used:

Pitch = arcsin(-Ax)

Roll = arcsin(Ay/cos(pitch))

One can then solve for the X and Y components of the heading (Hx and Hy) by utilizing:

Hx = Mx * cos(Pitch) + Mz * sin(Pitch)

Hy = Mx * sin(Roll) * sin(Pitch) + My * cos(Roll) - Mz * sin(Roll) * cos (Pitch)

The heading can then simply be found as arctan(Hy/Hx).

Appendix C: Critical IC Bill of Materials

Name	**Description**	**Quantity**	**Package**	**Digikey Part Number**
BQ24630	LiFePO4 battery charger	1	PVQFN-24	296-25761-1-ND
P8X32A-Q44	Propeller microcontroller	1	P8X32A-Q44	P8X32A-Q44-ND
RN-42	Bluetooth module	1	RN-42	740-1038-ND
TLC59711	12-channel motor driver	1	TLC59711	296-36745-1-ND
TPS62160	Step-down regulator	1	VSSOP	296-30318-1-ND
FT232RL	USB to serial converter	1	SSOP28	768-1007-1-ND

The complete bills of materials for the main haptic PCB and the battery charger PCB can be found at our GitHub repository at https://github.com/ricearcane/ti_contest/tree/master/Hardware/BOMs.

Appendix D: Hardware CAD files

All of our hardware design was done in EAGLE. The original CAD files for the integrated haptic feedback PCB, the motor coupon boards, and the battery charger board along with PDF copies of the schematic and boards designs may be found at our GitHub repository at https://github.com/ricearcane/ti_contest/tree/master/Hardware. There is also a text file to indicate what circuit corresponds to each page of the main haptic schematic.

Appendix E: Software

Our project involved writing software for two different computing platforms, the Intel NUC and the Parallax Propeller. There are multiple files for each platform, so we felt it would be excessive to provide all of the code in this appendix. Our sensor side code contains our stereo vision algorithm, some calibration data, and a built script, and it is located in our GitHub repository at https://github.com/ricearcane/ti_contest/tree/master/Software/Sensors.

Our haptic side code was written for the Parallax Propeller and contains code for driving the motors and utilizing data from the compass module. It can be found in our GitHub repository at https://github.com/ricearcane/ti_contest/tree/master/Software/Haptics.