Development Diary: The Parallax Laser Range Finder

by Joe Grand (@joegrand)
Grand Idea Studio
About

electrical engineer.
hardware hacker.
daddy.
runner.
(former) tv host.
Designer of Parallax Things
Agenda

Introduction

Triangulation Theory

Early Attempts & Development

Camera/Image Processing

Demonstrations
Design Goals

- Low cost
- Small footprint
- Easy to use
- Simple serial interface
- Open source/customizable
Application Ideas

- Distance or liquid level measurements
- Object detection and/or avoidance
- Item counting
Typical Range Finding Methods

- Time-of-Flight
- Phase Measurement
- Optical Triangulation
Optical Triangulation

\[ D = \frac{h}{\tan \theta} \]

http://sites.google.com/site/todddanko/home/webcam_laser_ranger/laser_ranger_drawing.gif
Optical Triangulation 2

- Relationship between $pfc$ and $\theta$ is a SLOPE-INTERCEPT linear equation

Early Attempt 1

Recreation of Todd Danko's Webcam Based DIY Laser Rangefinder
Not very accurate, but a good starting point to prove the concept
http://sites.google.com/site/todddanko/home/webcam_laser_ranger
Early Attempt 2

CMUcam2 + Freescale MC9S08QG8
Resolution 176x255, Accuracy ~1/4", Range 7-40"
Early Attempt 3

CMUcam2 + Propeller
Resolution 176x255, Accuracy ~1/4", Range 7-40"
Development Hardware

Propeller Proto Board + OVM7690 Eval. Board + Custom PCB
LRF Module: Front

635nm Laser Diode w/ APC Arima APCD–635–02–C3–A

640x480 CMOS Camera OmniVision OVM7690
LRF Module: Back

- Prop Clip
- Unused GPIO
- Propeller
- Bi-Color LED
- Laser Diode Driver (LDO)
LRF Module: Schematic
LRF Module: PCB Layout

Parallax Laser Range Finder

DANGER
LASER RADIATION
AVOID DIRECT EYE EXPOSURE
DIODE LASER
3 mW MAX OUTPUT at 635 nm
CLASS IIIa LASER PRODUCT

Parallax® Laser Range Finder

Grand Idea Studio

#580-4 Rev. A
2011 P. 3.0 US
Propeller

- Completely custom, ground up, open source
- Multicore: 8 parallel 32-bit processors (cogs)
- Code in Spin, ASM, or C

*** INFORMATION: www.parallax.com/propeller/
*** DISCUSSION FORUMS: http://forums.parallax.com
*** OBJECT EXCHANGE: http://obex.parallax.com
Propeller 2

- Clock: DC to 128MHz (80MHz recommended)
- Global (hub) memory: 32KB RAM, 32KB ROM
- Cog memory: 2KB RAM each
- GPIO: 32 @ 40mA sink/source per pin
- Program code loaded from external EEPROM on power-up
• Standard development using Propeller Tool & Parallax Serial Terminal (Windows)
• www.parallax.com/downloads/propeller-p8x32a-software
• Programmable via serial interface
**Cogs**

- Spin Interpreter (Cog 0)
- Auto-Baud Detection (start-up only)
- Full-Duplex Serial (JDCogSerial)
- Floating Point (F32)
- I2C for OVM7690 SCCB interface (pasm_i2c_driver)
- OVM7690 Frame Grabbers (on request)
# OVM7690 Camera Interface

- **DVP[7:0]** (Digital video port)
- **VSYNC** (Vertical Sync)
- **HREF** (Horizontal Reference)
- **PCLK** (Pixel Clock)

YUV422 color space @ 16 bits/pixel

<table>
<thead>
<tr>
<th>Byte Ordering (lowest byte)</th>
<th>Y0</th>
<th>U0</th>
<th>Y1</th>
<th>V0</th>
<th>Y2</th>
<th>U2</th>
<th>Y3</th>
<th>V2</th>
<th>Y4</th>
<th>U4</th>
<th>Y5</th>
<th>V4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macropixel = 2 image pixels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[www.fourcc.org/yuv.php#YUY2](http://www.fourcc.org/yuv.php#YUY2)
OVM7690 Frame Grabber

- Custom frame grabbers written in PASM
- Launched on demand depending on command
- Used PASD to help debug
  - www.insonix.ch/propeller/prop_pasd.html
1. Full (ovm7690_fg)
   - 160 x 128 x 8bpp (greyscale)
   - Useful for testing, taking low-res photos

2. ROI (Region of Interest, ovm7690_fg_roi)
   - 320 x 16 x 8bpp (greyscale)
   - Better resolution for actual range finding
   - Handles preliminary image processing (on request)
     - Double frame grab w/ laser off/on
     - Background subtraction, thresholding, column sum
1. Start cog
2. Grab frame
   • 8 bits at a time
3. Preliminary image processing (if requested)
4. When done, set flag in hub RAM to non-zero
5. Cog self-destruct
- Extremely timing sensitive
- Propeller overclocked to 96MHz
- Only had 24 cycles to grab/store each byte
- 6 instructions @ 4 cycles each!

<table>
<thead>
<tr>
<th>Specification</th>
<th>640x480 (VGA) @ 10fps (6MHz PCLK)</th>
<th>@96MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSYNC width</td>
<td>= 782.5uS</td>
<td>75095 cycles</td>
</tr>
<tr>
<td>Time from VSYNC low to HREF high</td>
<td>= 3.9325mS</td>
<td>377399 cycles</td>
</tr>
<tr>
<td>Time in between lines/HREF</td>
<td>= 35uS</td>
<td>3358 cycles</td>
</tr>
<tr>
<td>Time from last HREF in frame to next VSYNC</td>
<td>= 1.555mS</td>
<td>149232 cycles</td>
</tr>
<tr>
<td>Pixel clock (PCLK)</td>
<td>= 0.125uS</td>
<td>12 cycles/bit</td>
</tr>
</tbody>
</table>

(must grab data within 6 cycles of PCLK going high)

Timing diagram @ 96MHz Propeller
12 cycles/bit
Data valid when PCLK is HIGH

\[ \begin{array}{cccccc}
Y & U/V & Y & \ldots \\
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
t=0 & 6 & 12 & 18 & 24 & \text{cycles}
\end{array} \]
Image Processing

1. Background Subtraction
2. Thresholding
3. Column Sum
4. Blob Detection
5. Mass/Centroid Calculation(s)
6. Select Primary Blob
Command Interface

- TTL-level serial interface
- ASCII commands/responses
- Auto-baud rate detection (300-115.2kbps)
- Four physical connections:
  1. GND
  2. VCC
  3. SOUT (Serial Out)
  4. SIN (Serial In)
Basic Commands (FW 2.0)

- Single range measurement (in mm, decimal)
- Single range measurement (in mm, binary)
- Repeated range measurement
- Adjust camera for current lighting conditions
- Reset camera to initial settings
- Toggle laser on/off
- Display version information
- Display available commands
Advanced Commands (FW 2.0)

• Display coordinate, mass, and centroid for all blobs
• Calibrate camera system for range finding
• Adjust blob detection parameters
• Capture & send single frame (160x128)
• Capture & send single frame (320x16) w/ laser enabled
• Capture & send processed frame (320x16) w/ background subtraction
**Calibration**

- Required during production to account for manufacturing variances (camera and laser diode alignment)
- Required after major firmware update
- Done "automatically" using 'X' command
  1. Take a number of measurements from known distances
  2. Record \( pfc \) value and actual angle at each distance
  3. Calculate slope & intercept values
  4. Store calibration data in unused portion of boot Serial EEPROM
## Measurement Results (cm)

<table>
<thead>
<tr>
<th>Actual Distance to Target (cm)</th>
<th>Calculated Distance (cm)</th>
<th>Difference (Δ)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>19.9</td>
<td>0.1</td>
<td>-0.50</td>
</tr>
<tr>
<td>30</td>
<td>29.7</td>
<td>0.3</td>
<td>-1.00</td>
</tr>
<tr>
<td>40</td>
<td>40.1</td>
<td>-0.1</td>
<td>0.25</td>
</tr>
<tr>
<td>50</td>
<td>50.3</td>
<td>-0.3</td>
<td>0.60</td>
</tr>
<tr>
<td>60</td>
<td>60.2</td>
<td>-0.2</td>
<td>0.33</td>
</tr>
<tr>
<td>70</td>
<td>70.8</td>
<td>-0.8</td>
<td>1.14</td>
</tr>
</tbody>
</table>

**Average % Error**

0.64

Prototype unit, serial #0
# Measurement Results (in)

<table>
<thead>
<tr>
<th>Actual Distance to Target (in)</th>
<th>Calculated Distance (in)</th>
<th>Difference (Δ)</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9.9</td>
<td>0.1</td>
<td>-1.00</td>
</tr>
<tr>
<td>20</td>
<td>20.1</td>
<td>-0.1</td>
<td>0.50</td>
</tr>
<tr>
<td>30</td>
<td>30.7</td>
<td>-0.7</td>
<td>2.33</td>
</tr>
<tr>
<td>40</td>
<td>40.3</td>
<td>-0.3</td>
<td>0.75</td>
</tr>
<tr>
<td>50</td>
<td>48.8</td>
<td>1.2</td>
<td>-2.40</td>
</tr>
<tr>
<td>75</td>
<td>70.3</td>
<td>4.7</td>
<td>-6.27</td>
</tr>
</tbody>
</table>

Average % Error: 2.21

Prototype unit, serial #0
Key Specifications

• Optimal measurement range: 6-48 in. (4 ft.)
• Accuracy error: < 5% (typically much better)
• Sample rate: 5Hz
• Power: 5V @ 150mA
• Operating temperature: 32-122 °F (0-50°C)
• Dimensions: 3.95" W x 1.55" H x 0.67" D
Limitations

- **Range**
  - Longer distances will result in a noticeable reduction in accuracy due to very slight changes in angle
  - Firmware limits maximum distance to 100"

- **Environment**
  - Works best in a controlled environment w/minimal changes in brightness (e.g., indoors)
  - Not reliable against bright targets, as background subtraction will remove the bright spot from the frame (including the laser)
Demonstrations

- Terminal Program
- LRF Image Viewer (VB.Net)
- BASIC Stamp II
- Arduino
- FSLBOT (MCF52259)
- LRF + Nintendo Game Boy Printer
Get It

parallax.com/product/28044
*** Assembled units, example code, documentation

grandideastudio.com/portfolio/laser-range-finder
*** Schematics, BOM, videos, other documentation

github.com/grandideastudio/laser-range-finder
*** Source code
The End.