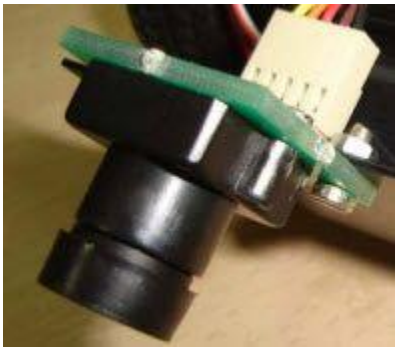
Line Scan Camera Use

The line scan camera module consists of a CMOS linear sensor array of 128 pixels and an adjustable lens. This camera has a 1x128 resolution.



Line Scan Camera Board Details

Schematics, BOM, and datasheets:

Line Scan Camera featuring TAOS 1401

Mounting Options

You will likely want to mount the camera on a mast or boom above the car to ensure the greatest field of view. Determining the angle of orientation about the pivot at the top of the boom will change the “look ahead” distance of the camera and enable more efficient steering algorithms

DIY Camera Mounts

Solution Overview

One method of implementation is to take the entire readout of the camera and store it in the memory. Then a line detection algorithm can be used to locate the position of the black line. Due to varying lighting conditions, some level of pixel thresholding may be necessary as the intensity differences across the data may not always produce a clear indication of the line location. A good approach is to use an algorithm that looks for changes in the magnitude of voltage from one portion of the array to another, since the camera’s AO magnitude is directly related to the brightness the pixel array senses. If the microcontroller finds a significant decrease in magnitude followed by large increase in magnitude this would give us a good indication of the location of the line. For this a derivative function can be utilized.

Once we have successfully determined the position of the black line, immediately adjust the wheels to adjust the direction of the car so that the black line will remain in the center of the camera’s view.

Sample camera output (for illustrative purposes only) The camera outputs an analog signal from 0 to 5V depending on the grey-scale value of the image. to simplify our sample we will assume that we have set limits for the line and have transformed the data to digital bits using a threshold value. 0’s are high intensity (non-line locations), 1’s are low intensity (black or line locations)

10000000000000000000000000000000011111010000000000000000001000000000000000

Since the camera provides a 128x1 bit picture, and the camera will be pointing down at the track which is a fixed width. A control algorithm should be developed to line up the 1’s in the center of the 128 bits. The center of the field of view will be require calibration and testing, but it is

assumed that the camera will remain in a fixed location pointing down the center of the forward looking axis of rotation.

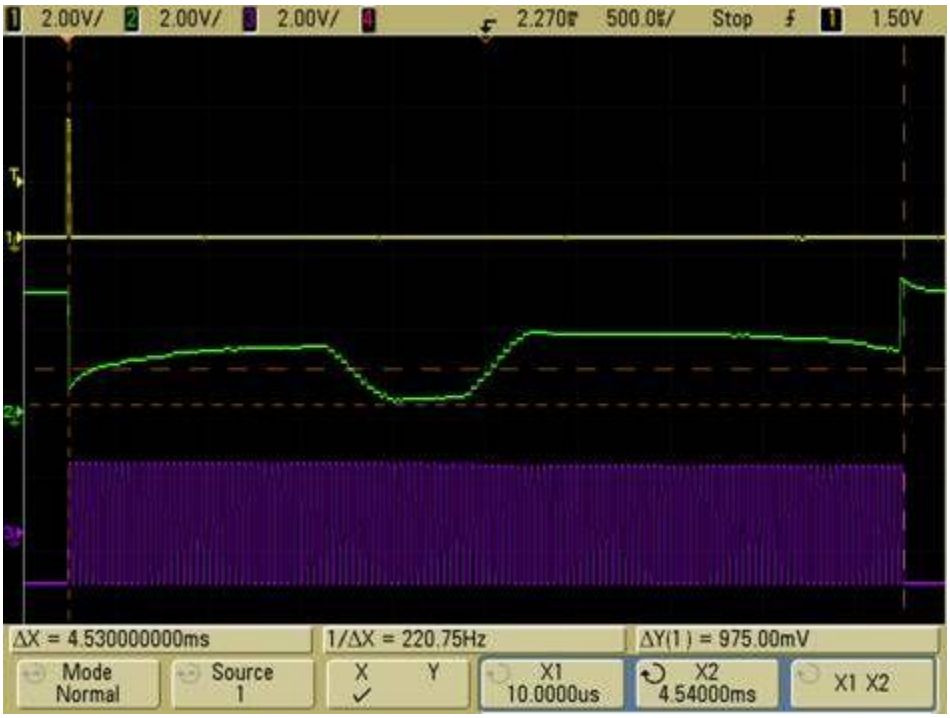
Signals

For normal operation of the camera, the following signals must be produced and processed:

1. **CK (clock)** - latches SI and clocks pixels out (low to high) continuous signal
2. **SI (serial input to sensor)** begins a scan / exposure discrete pulses, pulse must go low before rising edge of next clock pulse
3. **AO (analog output)** - Analog pixel input from the sensor (0-Vdd) or or tri-stated

The CK and SI signals are simple ON/OFF signals which can be produce using a GPIO Pin, setting the pin high and low corresponding to the desired exposure time of the camera. The only other requirement is to read the Analog Output of the camera which requires the initialization of the Analog Module and setting it to the proper pinout.

Acutal camera output image:



Yellow = SI, Green = Camera Signal, Purple = clock

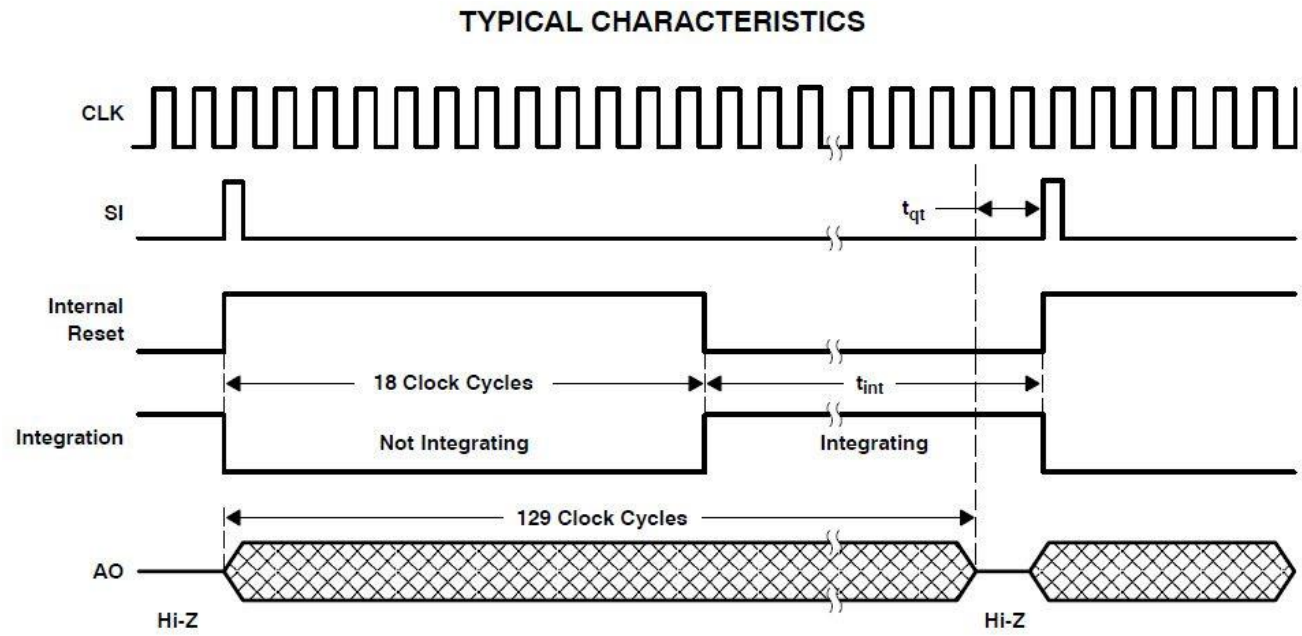
[More camera waveforms and information (Power Point) available here]

This link shows a video of the camera connected to the oscilloscope
http://www.youtube.com/watch?v=YOAd3ERnXiQ

To obtain this signal, connect channel 1, 2 and 3 of an oscilloscope to the SI pulse (Trigger off this signal), CLK, and AO signals.

Timing

The timing for creation and read of the signals is crucial and is detailed in the diagram below. This information can also be found in the Line Scan Datasheet.



Analog Read:

The Analog Output (AO) signal from the camera needs to be processed and read by the microcontroller's Analog to Digital Converter (ADC). This ADC device converts a continuous signal into a discrete number which is proportional to the signal voltage. An 8 bit ADC has 256 discrete levels (2^8). If a analog signal between 0 and 5 volts is sampled, a digital discrete number of 0 would correspond to zero volts, and a digital discrete number of 255 would correspond to 5 volts. A number such as 145w would correspond to about 2.8 volts. The maximum signal sample rate is limited by the microcontroller. Proper configuration of the ADC peripheral and the multiplexer of the chip will configure a pin to read in an analog signal when calling the function. More details on analog to digital converters can be found on the wikipedia site here.

Read/Write

In write mode, the GPIO pin can be set, cleared, or toggled via software initiated register settings.

Microcontroller Reference Manual: Analog to Digital Converter

You will find high level information about GPIO usage in several different areas of a reference manual. See the reference-manual article for more general information.

- Relevant Chapters: (see GPIO chapters for clock and SI Creation)
    - Introduction:
    - System Modules: System Integration Modules (SIM) - provides system control and chip configuration registers
    - Chip Configuration:
    - Signal Multiplexing: Port control and interrupts

Focusing the camera:

Once the sensor is perfectly working the next step is to find the best position of the lens that will generate the clearest images.

1. The best way to do it is using an oscilloscope:
2. Connect the SI and AO signals to the oscilloscope
3. Set the SI pulse so that it can be clearly seen and then trig the AO signal with the SI signal using the trig function
4. Fix the camera looking at a sheet of paper with a black line in the center
5. The image of the black line will appear on the oscilloscope screen
6. Screw the camera until you find the position where the line seems the clearest

Camera Circuit

5 wires must be connected

- ground
- power
- SI
- CLK
- AO

Camera Limitations

According to the datasheet:

" The sensor consists of 128 photodiodes arranged in a linear array. Light energy impinging on a photodiode generates photocurrent, which is integrated by the active integration circuitry associated with that pixel. During the intergration period, a sampling capacitor connects to the output of the integrator through an analog swith. The amount of charge accumulated at each pixel is directly proportional to the light intensity and the integration time."

Integration Time: T

T = (1/fmax)*(n-18)pixels + 20us, where n is the number of pixels

Minimum integration time: 33.75us
Maximum integration time: capacitors will saturate if exceeding 100ms
frequency range 5 Khz - 8 Mhz (8 Mhz is fmax in equation above)

The integration time is the following: It occurs between the 19th CLK cycle and the next SI pulse. The CLK frequency itself has little to do with the integration time. One each rising edge, the clock outputs one of the previously sampled intensity values. This means that integration time should be set by varying the time between SI pulses, not changing the clock frequency. Make the CLK frequency high, and have as much time as needed between the two SI pulses to obtain the desired intensity value.

Pro Tips:

#1 - Avoid Light Noise

- Light can be transmitted through the pcb on the back of the camera. This unwanted extra light shining on the CMOS linear sensor can induce significant errors into your signals received. A shroud or housing for the camera unit can easily eliminate this problem. One of the easiest solutions is to place a piece of electrical tape across the back of the camera in the highlighted area indicated in the picture below.

#2 - Know your Settings

When testing the car on the track or transporting it, it is not uncommon for the focus on the camera to loosen or change. Therefore it is recommended that after adjusting your camera focus for maximum performance you make mark (ex. metallic sharpie) between the lens and its body so you can realign the camera lens to it's proper position easily if it does shift.

#3 - Know your Zone

When hooking up the linescan camera, regardless of position or focus there is a drop off at each end of the image data. This is easily viewed with an oscilloscope. This effect is undesirable, particularly when you are finding your line position utilizing a derivative approach. These fallouts cause erroneous derivative values, and hence a poor line position solution. Two solutions we found useful were: (1) Ignoring the first 10-15 pixels and last 10-15 pixels of the image data array, and then determining the line position; (2) Often when making decisions in the code as to where the line was at it was found useful to use a threshold value for the difference in the derivative position, and secondly a binary threshold on the camera data. Note that the falloff depends on camera focus, position, etc. Therefore, these threshold values and pixels in which to ignore are relative to a specific instance. The problem however is common to the camera.

#4 - Buffer Data

Since the camera can read the line very quickly while the servo can only update every 20ms, there are multiple camera reads before the servo can update, if you are reading the camera fast and then overriding without saving them in some form then those camera reads are being wasted and are better off not having occurred. What can help is to create some sort of filter by bringing new values into an array with previous values and preforming some sort of averaging. The following code will take the new line position value and place it in a 1xA array where A is defined by CAMERA_AVG. NO AVERAGING IS OCCURRING HERE all that is happening is the camera values are being saved in a simple array, what is done with them is up to you.
The way this works is that it shifts the entire array so the oldest data point is discarded in order to make room for the new line position at the other end of the array. It will only adds the new value if there is one available if not it copies the previous first position value to the new first position value.

CAMERA_AVG => an integer value for how long the averaging length will occur
gfpLineAverage => global floating point array of camera center line values
fpLinePos => returned from read camera this is the center line position
ReadCamera() => is the read camera function call returns a floating point value of fpLinePos

// this will shift the values up and throw away the oldest value
// then add a new reading

```
for (i=CAMERA_AVG;i>0;i—)
{
gfpLineAverage[i]=gfpLineAverage[i-1];
}
// if no line was detected the previous camera value will be passed on
if (fpLinePos=ReadCamera())
{
gfpLineAverage[0]= fpLinePos;
}
```

For example an array of of center line position values ranging from 0-127 could look like.
Initial values
[51 50 52 54 58 55]
New position of 45 read
[45 51 50 52 54 58]
New position of 44 read
[44 45 51 50 52 58]
No value read
[44 44 45 51 50 52]
No value read
[44 44 44 45 51 50]
New position of 50 read
[50 44 44 44 45 51]

Program Exercise

Specifics of how to configure the K40 ADC, to create the delay code is covered in the K40: Line Scan Camera Tutorial.

Additional Training Resources

Freescale Motor Control Tutorial

Freescale Lecture 1: Introduction and Motor Basics
Freescale Lecture 2: Pulse Width Modulaiton

Freescale Lecture 3: Control Design

Freesacle Lecture 4: Speed and Position

Freescale Lecture 5: MPC5607B Overview

Additional Resources

- Freescale app note on interfacing with a linescan camera
- Freescale app note on interfacing with an RCA camera

Labels

- camera
- camera_mount
- freescale_cup
- robotics
- sensor

Attachments

Dataset for Camera Algorithm.pptx