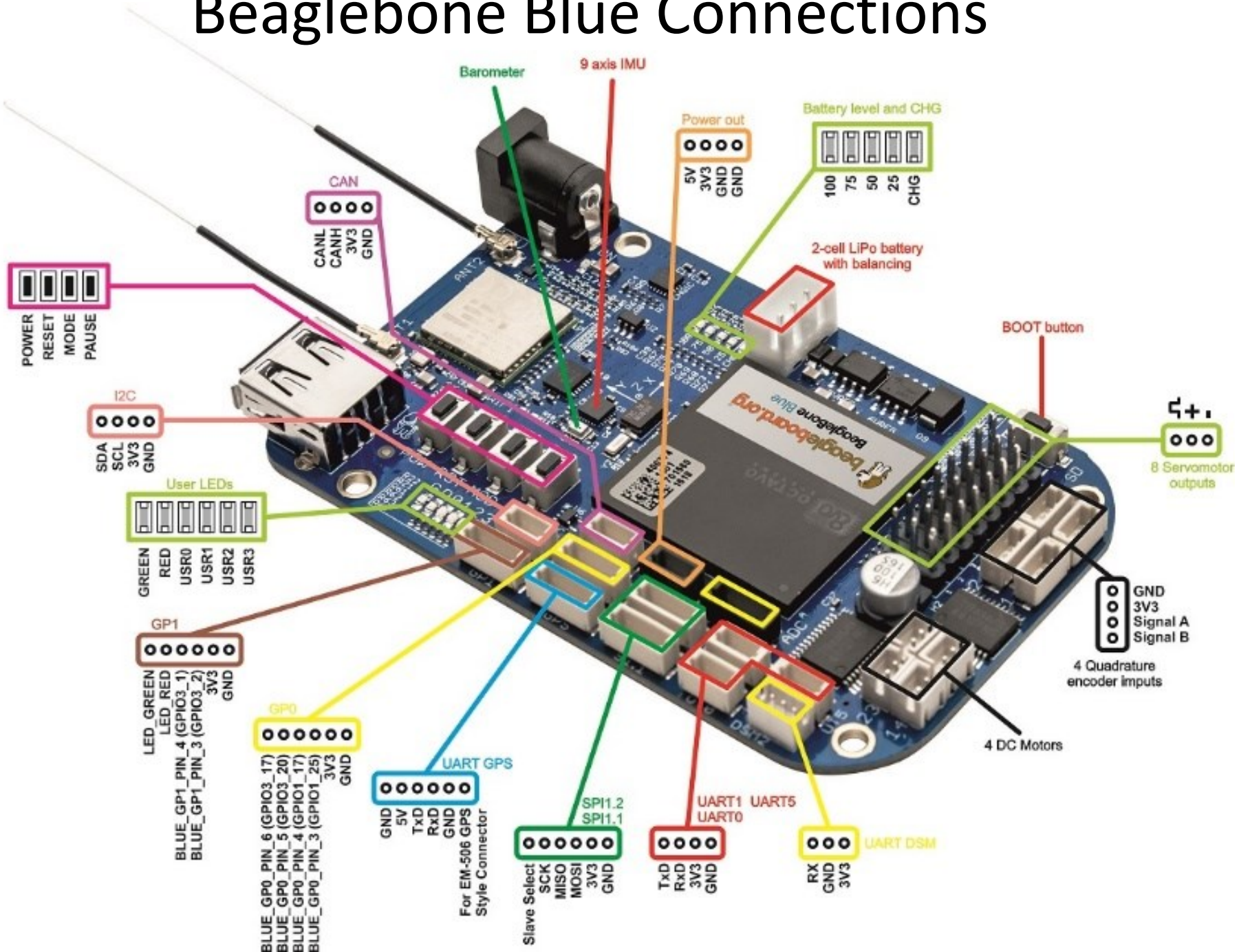
EECS192 Discussion Week 7
Programmable Real Time Unit and Linescan Camera

- Camera Connections
- PRU overview
- PRU <-> Linux interface
- PRU debugging

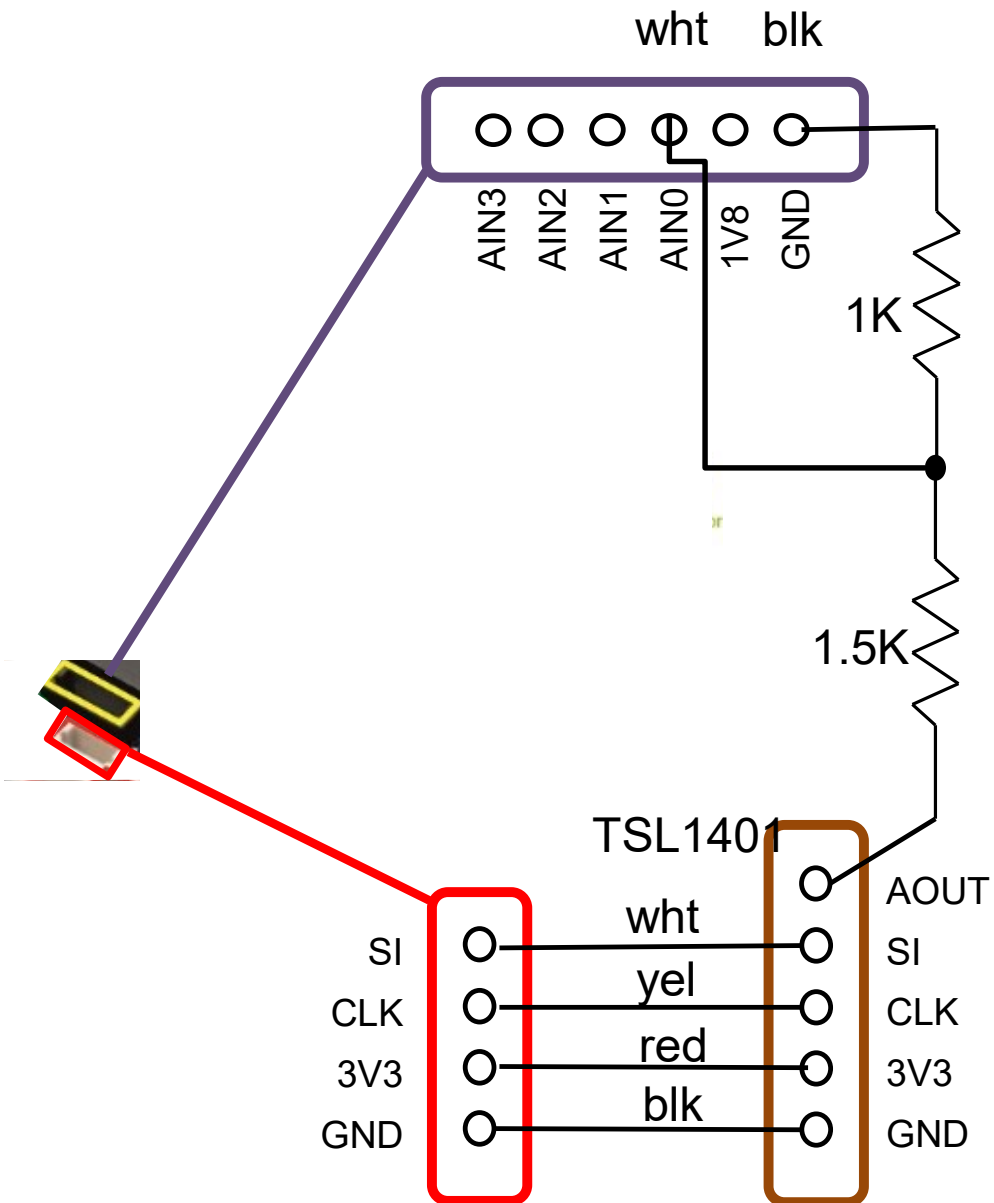
Beaglebone Blue Connections



GPIO015=SI=UART1_TX
GPIO014=CLK = UART1_RX

TSL1401 line camera and BeagleBoneBlue Interface

Note AIN < 1.8 V!



Directions: pru_firmware and LineCamera

https://github.com/ucb-ee192/SkeletonBeagle/README.md

Skeleton code for reading TSL1401 Line Sensor on Beagle Bone Blue for EE192
Using PRU0 for Fast A/D read (about 8.2 us) Feb. 27, 2019.
1. Clone whole repo. git clone https://github.com/ucb-ee192/SkeletonBeagle
2. Compile and install PRU code i) cd pru_firmware ii) make all iii) sudo make install
3. compile and run LineCamera i) cd LineCamera ii) make iii) sudo ./LineCamera (must be root as accesses /dev/mem, etc) This reads TSL1401 through shared memory buffer and saves to file linescans.csv LEDs USR0...USR3 will flash when reading A/D. This particular .csv format is compatible with Python linescanplot.py

Notes:

- The PRU0 has direct access to GPIO addresses without any device driver protection.
- UART1 RX is used as CLK. UART1 TX is used as SI.These GPIO pins are setup using rc_pinmux and rc_gpio_init, and should avoid Linux conflicts.
- The analog output from the line camera is in the range 0...3.3 V. A voltage divider is needed to keep the input voltage less than 1.8V on the A/D input.
- See LineCamera/TSL1401-BBBLue-interface.png for wiring connections.

GPIO Access Linux vs PRU

http://inst.eecs.berkeley.edu/~ee192/sp19/files/BeagleBone Blue Pin Table.xlsx

Use on Blue	config-pin name	Ball	MODE0	MODE1	MODE2	MODE3	MODE4	MODE5	MODE6	MODE7
UARTs										
UART1_TX	P9.24	D15	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL	NC	pr1_uart0_txd	pr1_pru0_pru_r31_16	gpio0[15]
UART1_RX	P9.26	D16	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA	NC	pr1_uart0_rxd	pr1_pru1_pru_r31_16	gpio0[14]

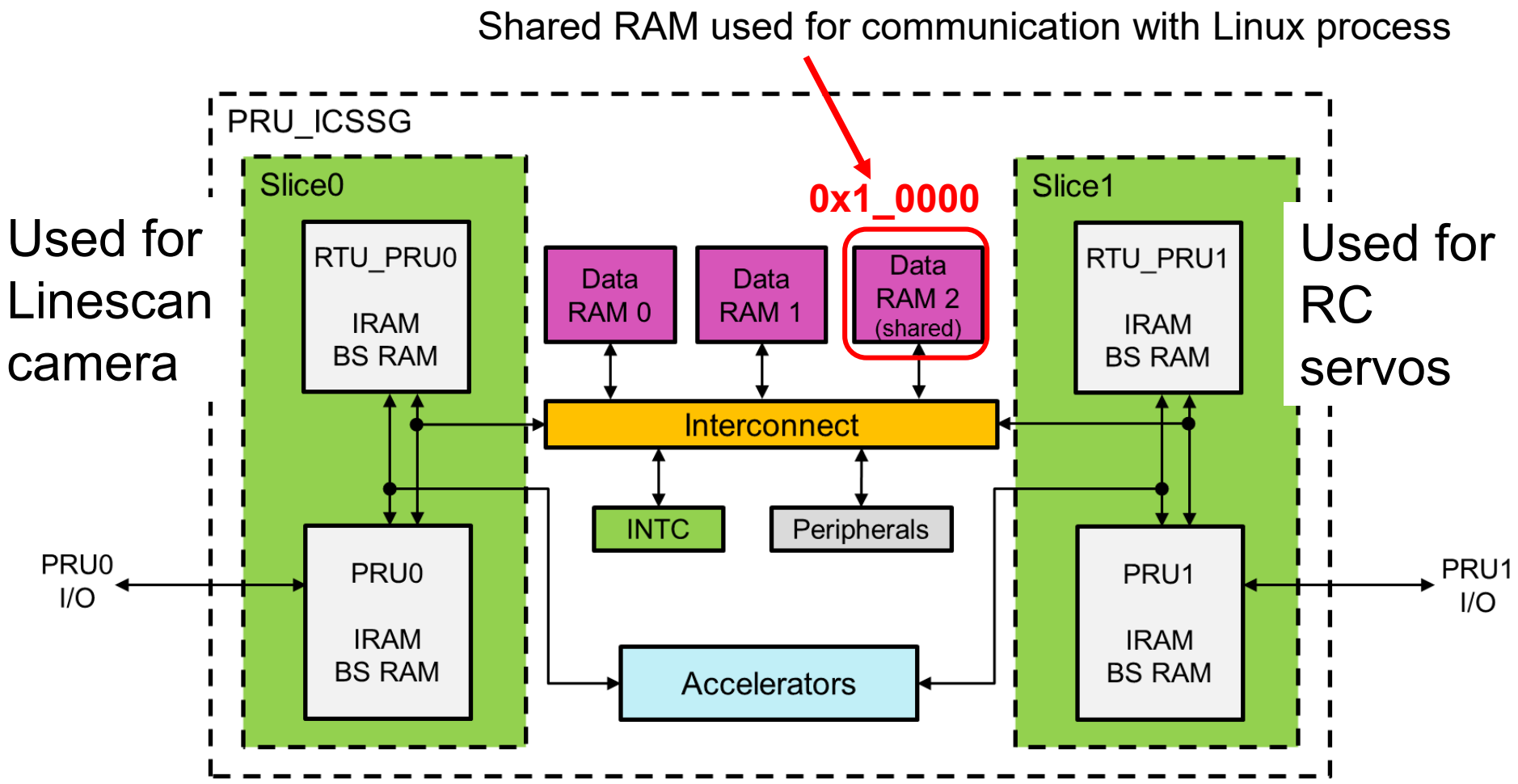
Linux GPIO lines need to be set to mode 7 then output direction:

```
int pin1 = UART1_HEADER_PIN_3;    ///< P9_26, normally UART1 RX
rc_pinmux_set(pin1, PINMUX_GPIO);
rc_gpio_init (CHIP, pin1, GPIOHANDLE_REQUEST_OUTPUT);
```

PRU direct bit/register access

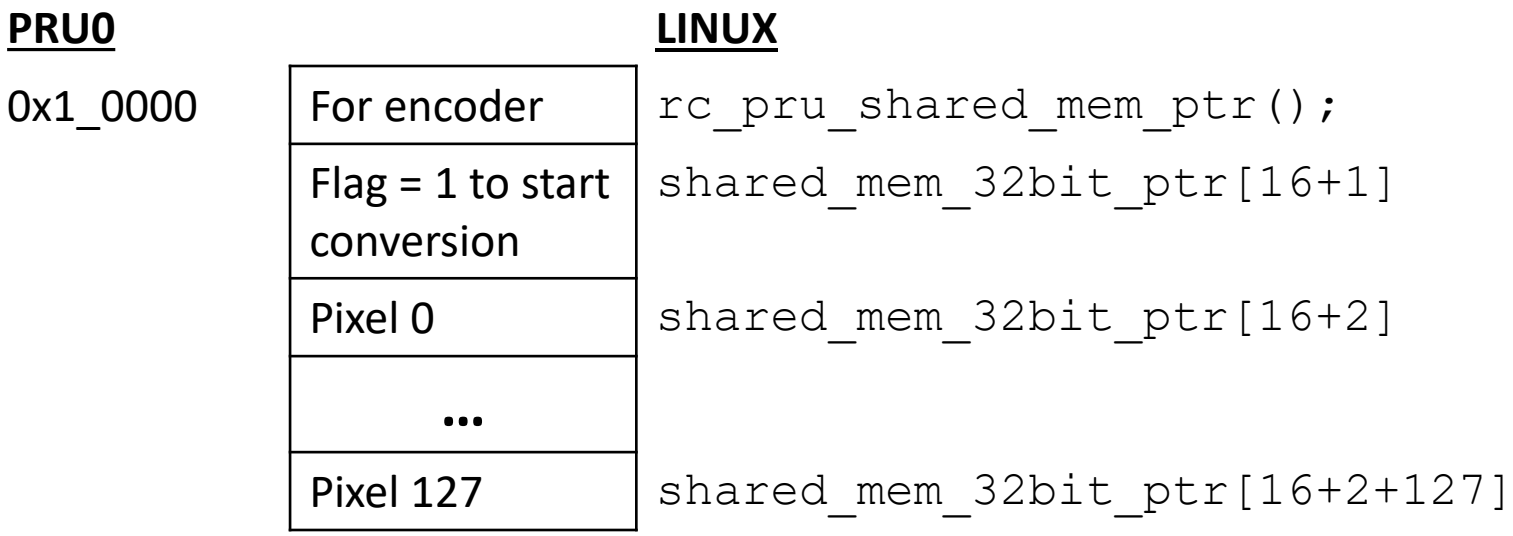
```
#define GPIO0 0x44e07000
#define GPIO_CLEARDATAOUT 0x190
#define GPIO_SETDATAOUT 0x194
#define CLK (1 << 14) // gpio[14] UART1_RX
#define SI (1 << 15) // gpio[15] UART1_TX
*GPIO0_CLEAR = CLK; //  reset CLK line
*GPIO0_SET = CLK;
```

Programmable Real-time Unit



Used on BeagleBone Blue for RC servo and quad encoders (hopefully real-time A/D for line camera)

Shared 12 kb Address Space



Linux LineCamera.c line 116:

```
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 1;
// set flag to start conversion by PRU
while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 1);
```

PRU main_pru0.c line 117:

```
while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 0);
// loop until command
//… read 128 pixels …
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 0;
// reset to zero
```

Linescan telemetry file format

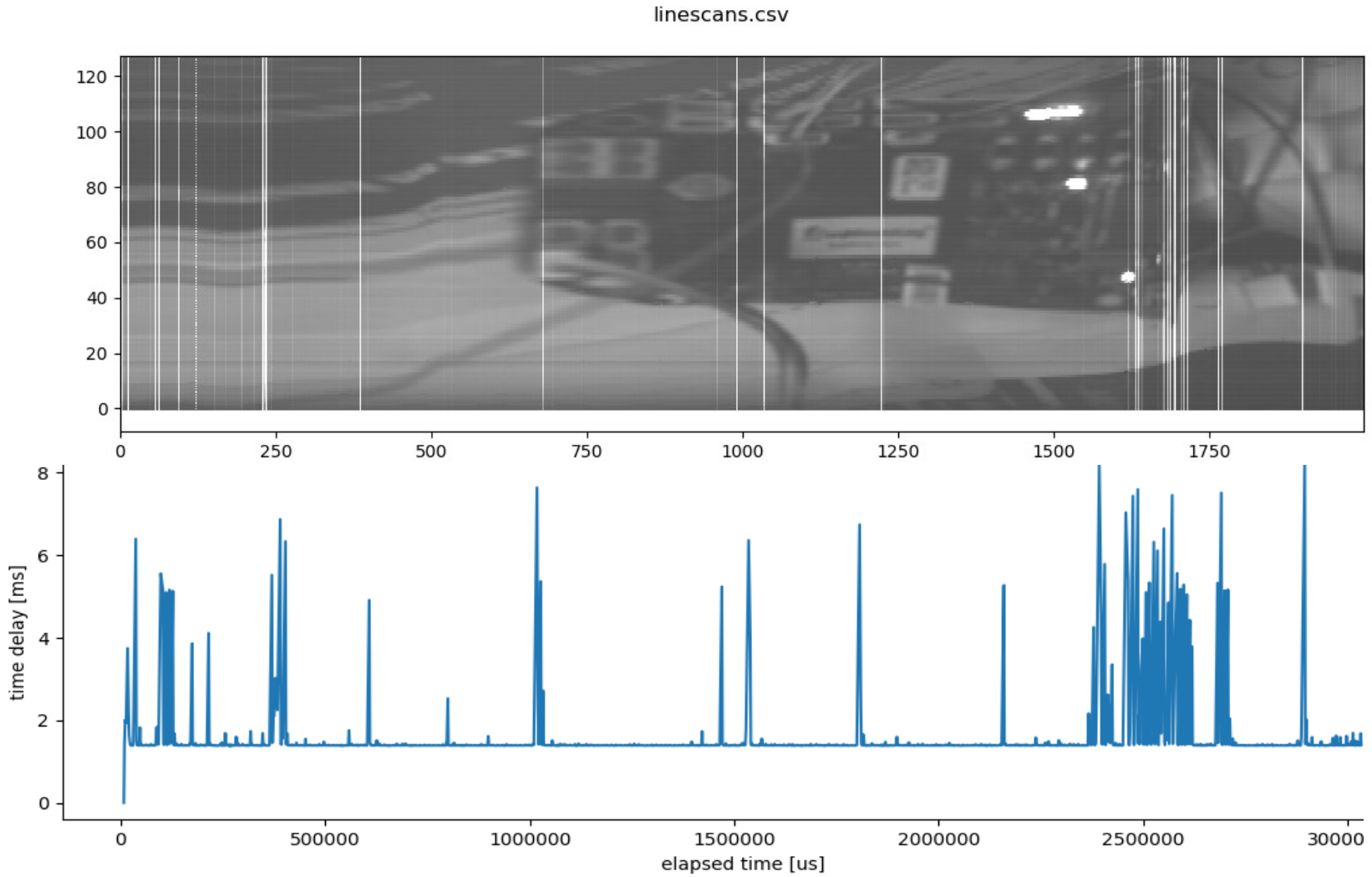
Caution with .csv format: spaces also act as delimiters

linescans.20190302-235143.csv

time (us) linescan_near velocity
2957
[114,114,120,115,128,122,123,119,120,117,124,119,124,119,124,120,122,125,174,167,174,171,176,184,196,190,195,176,162,142,145,143,150,151,159,159,167,165,166,156,159,163,173,171,179,174,175,172,178,175,180,174,178,174,174,170,179,168,176,173,176,171,175,171,175,169,174,169,174,169,172,168,174,173,173,168,173,164,167,160,165,158,159,156,156,142,146,137,136,141,151,146,146,142,145,141,143,138,140,135,140,133,136,128,128,122,131,133,146,167,188,183,183,174,179,177,186,178,181,173,177,167,170,164,162,156,155,152] 0
5857
[177,168,178,173,193,191,181,176,180,176,186,181,186,180,185,182,185,188,328,316,326,324,336,365,389,381,385,340,288,241,237,241,257,268,282,293,306,311,304,278,281,308,327,328,338,337,336,333,337,335,340,337,337,338,328,328,339,321,331,330,329,325,326,323,327,323,326,319,322,319,319,317,326,326,325,317,318,304,302,296,298,289,287,279,275,244,245,225,210,239,250,249,246,240,241,236,234,227,227,220,220,212,212,197,182,178,195,213,239,321,376,364,360,342,342,346,358,352,350,340,334,320,318,304,294,281,271,266] 0

Display Linescan file

On laptop or desktop: `python linescanplot.py`



Note: Variable timing from Linux causes overexposure

Delay measurement using rc library

Note: Variable timing from Linux causes overexposure

Linux LineCamera.c

```
start_time = rc_nanos_since_boot();

   for(j = 0; j< 1000; j++)
   {shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] = 1;
// set flag to start conversion by PRU
   while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 1);
// wait for PRU to zero word
      for(i = 0; i< 128; i++){
linescan[i]=
    (int) shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+2+i];
// copy data
             }
      }
end_time = rc_nanos_since_boot();
run_time = end_time - start_time;
```

A to D using PRU pru_firmware/src/main_pru0.c

```
#define ENCODER_MEM_OFFSET 16
#define PRU_SHAREDMEM 0x10000 //shared memory with Cortex A8
#define GPIO0 0x44e07000
#define GPIO_CLEARDATAOUT 0x190
#define GPIO_SETDATAOUT 0x194
#define CLK (1 << 14) // gpio[14] = TSL1401 clock, UART1_RX
#define SI (1 << 15) // gpio[15] = start integration, UART1_TX
// access shared memory with Linux:
shared_mem_32bit_ptr = (volatile unsigned int *) PRU_SHAREDMEM;

// signal to Linux process that PRU is ready by clear []
shared_mem_32bit_ptr[ENCODER_MEM_OFFSET] = 0x0;...

for(i = 0; i< 130; i++)
{ *GPIO0_CLEAR = CLK; //  reset CLK line
  shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+2+i] = read_adc(0);
  *GPIO0_SET = CLK; // rising edge of CLK gives next pixel
      __delay_cycles(200); // allow hold time - 1 us
}
```

Need at least 129 clocks for proper operation

PRU Debugging details (for reference)

Debugging PRU

1. `git clone https://github.com/RRvW/prudebug-rl`
2. missing readline.h

   `sudo apt-get install libreadline-dev`
3. `make`
4. `sudo ./prudebug`

https://markayoder.github.io/PRUCookbook/04debug/debug.html

Assembly code: `build/main_pru0.lst`

Symbol table: `file.map`

```
page   address   name
----   -------   ----
0      00000200  main
0      00000324  read_adc
```

Maybe need Beaglebone reset if get:
ERROR: failed to run rc_encoder_pru_init

Prudebug main_pru0.c

main_pru0.c line 117:

```
116  while(1)
117  {while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1]==0);
119  *GPIO0_SET =  SI ; // set SI
```

---

main_pru0.lst line 697:

```
697; 117 | {      while(shared_mem_32bit_ptr[ENCODER_MEM_OFFSET+1] == 0); // loop
698;     |  until command to read 128 times
701;----------------------------------------------------------------------
702 00000078 000000F1002081         LBBO      &r1, r0, 0, 4         ; [ALU_PRU] |117| $O$v1
703 0000007c 0000005700E1FF         QBEQ      ||$C$L6||, r1, 0x00   ; [ALU_PRU] |117|
706
```

PRU Instruction offset address = 0x200 (from file.map) +0x78 = 0x278
Since 4 bytes per instruction, PRU instruction memory address = 0x278/4 = 0x9e

```
PRU0> dis 0x9e
Absolute addr = 0xd09e, offset = 0x009e, Len = 16
[0x009e] 0xf1002081 >> LBBO R1, R0, 0, 4
[0x009f]  0x5700e1ff       QBEQ -1, R1, 0
[0x00a0] 0xe1043297  SBBO R23, R18, 4, 4
```

Breakpoint after while()

Prudebug main_pru0.c

PRU0> dis 0x9e
Absolute addr = 0xd09e, offset = 0x009e, Len = 16
[0x009e] 0xf1002081 >> LBBO R1, R0, 0, 4
[0x009f] 0x5700e1ff QBEQ -1, R1, 0
[0x00a0] 0xe1043297 SBBO R23, R18, 4, 4


```
PRU0> br 1 0xa0
PRU0> gss
Running (will run until a breakpoint is hit or a key is pressed)....
```

Now linecamera will wait for gss to read next frame

Errata: LineCamera.c line 64:

```
if(rc_encoder_pru_init()){…}
```

PRU needs to be running, e.g. reset flag to pass this init.