Topics

- Checkpoints 1, 2, 3
- Memory Usage build/runtime
- Updated Skeleton code tour
- Peripheral interface: A/D, pulse count (intro)
- Sensor: line camera

Notes:
- 2/9 project proposal – upload to bcourses by Tues 5 pm
- Survey for Cory Courtyard
High resolution timing using built-in 64 bit counter

```c
uint64_t task_counter_value1, task_counter_value2;
timer_get_counter_value(TIMER_GROUP_0, TIMER_0, &task_counter_value1);
/
/* code to be timed here */
	timer_get_counter_value(TIMER_GROUP_0, TIMER_0, &task_counter_value2);
```
CP2- PWM for driving steering servo and ESC

Write code which performs the following sequence of functions:

• C2.1: Start wheels turning, and ramp up to full speed in 5 seconds and down to zero speed in another 5 seconds.

• C2.2: Set steering angle approximately half full-left and hold for 5 seconds. (For example, if full steering range is + - 20 degrees, set steering angle to +10 degrees.)

• C2.3: Set steering angle straight and hold for 5 seconds.

• C2.4: Set steering angle approximately half full-right, and hold for 5 seconds.

• C2.5: Set steering angle back to approximately straight.

• C2.6: Show steering changing and wheels turning at the same time

• C2.7: Report Data RAM and Instruction RAM usage. How much of each is left? [pio run –v in terminal window]

• C2.8: All members must fill out the checkpoint survey before the checkoff close. Completion is individually graded
ESC startup - generating timing options

1. `vTaskDelay(pdMS_TO_TICKS(3000));` [1 ms resolution]
   
   or

2. Busy wait (may trigger watchdog for long delays)
   
   ```c
   while(task_counter_value < end_value)
   {
     timer_get_counter_value(TIMER_GROUP_0, TIMER_0, &task_counter_value);
   }
   
   or
   
   3. Interrupt alarm [usec resolution] (see timer-group-example.c in SkeletonHuzzah32)
      /* if timer interval > CONFIG_ESP_TASK_WDT_TIMEOUT_S, will get watchdog timeout*/
      timer_set_alarm();
      
CP3- remote control with UDP

(Intended to be easy since project proposal due Tues 2/9)

The car should be upside down, or lifted off the ground so it does not move.

CAUTION: setting PWM to 1.0 ms or 2.0 ms can drive servo to the end of its range and cause it to burn out.

• C3.1 From the remote client keyboard, send a speed command for the drive motor. Pick a range such as 0-100, and show that you can specify a range of values. The motor should remain turning until the next command.

• C3.2: Find the minimum ESC PWM value which causes the wheels to turn.

• C3.3: With motors turning, send an “Emergency Stop” command to the car (the motor should be turned off or braked, if available).

• C3.4: From the remote client, send a command to set the steering angle.

• C3.5: Find the range of steering angles for your car. (These values should be used for range checking in your code for the rest of the semester.)

• C3.6: Show that motor speed and steering can be set independently from the remote client keyboard.

• C3.7: Report Data RAM and Instruction RAM usage. How much of each is left?

        From VS terminal > pio run -v

• C3.8: All members must fill out the checkpoint survey before the checkoff close. Completion is individually graded.
EECS192 Lecture 3
Feb. 2, 2021

Notes:
• 2/9 project proposal – upload to bcourses by Tues 5 pm
• Survey for Cory Courtyard

Topics
• Checkpoints 1,2,3
• Memory Usage build/runtime
• Updated Skeleton code tour
• Peripheral interface: A/D, pulse count (intro)
• Sensor: line camera
Recap: Intro to stack, heap, malloc, free, etc

Memory model

- **stack**
- **heap**

DRAM

- uninitialized data
- **bss**
- initialized data
- **data**

IRAM

- **text**

Memory use at compile time

From VS terminal > pio run –v

<table>
<thead>
<tr>
<th>section</th>
<th>size</th>
<th>addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>.iram0.vectors</td>
<td>1027</td>
<td>1074266112</td>
</tr>
<tr>
<td>.iram0.text</td>
<td>88424</td>
<td>1074267140</td>
</tr>
<tr>
<td>.iram0.text_end</td>
<td>0</td>
<td>1074355564</td>
</tr>
<tr>
<td>.dram0.data</td>
<td>19912</td>
<td>1073414144</td>
</tr>
<tr>
<td>.dram0.bss</td>
<td>24560</td>
<td>1073434056</td>
</tr>
<tr>
<td>.flash.rodata</td>
<td>103292</td>
<td>1061158944</td>
</tr>
<tr>
<td>.flash.text</td>
<td>486215</td>
<td>1074593824</td>
</tr>
<tr>
<td>.debug_frame</td>
<td>91744</td>
<td>0</td>
</tr>
<tr>
<td>.debug_info</td>
<td>3444763</td>
<td>0</td>
</tr>
<tr>
<td>.debug_abbrev</td>
<td>287554</td>
<td>0</td>
</tr>
<tr>
<td>.debug_loc</td>
<td>696445</td>
<td>0</td>
</tr>
<tr>
<td>.debug_aranges</td>
<td>36008</td>
<td>0</td>
</tr>
<tr>
<td>.debug_ranges</td>
<td>40656</td>
<td>0</td>
</tr>
<tr>
<td>.debug_line</td>
<td>1623178</td>
<td>0</td>
</tr>
<tr>
<td>.debug_str</td>
<td>283103</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>7227598</strong></td>
<td></td>
</tr>
</tbody>
</table>
Heap memory (in Data RAM)- Before tasks

Used for stack, local variables, global variables
malloc() and free()

heap_caps_print_heap_info(MALLOC_CAP_8BIT);

Heap summary for capabilities 0x00000006:
At 0x3ffae6e0 len 6432 free 0 allocated 6300 min_free 0
  largest_free_block 0 alloc_blocks 25 free_blocks 0 total_blocks 25
At 0x3ffbada8 len 152152 free 136724 allocated 15284 min_free 135908
  largest_free_block 135908 alloc_blocks 26 free_blocks 2 total_blocks 28
At 0x3ffe0440 len 15072 free 15036 allocated 0 min_free 15036
  largest_free_block 15036 alloc_blocks 0 free_blocks 1 total_blocks 1
At 0x3ffe4350 len 113840 free 113804 allocated 0 min_free 113804
  largest_free_block 113804 alloc_blocks 0 free_blocks 1 total_blocks 1
Totals:
  free 265564 allocated 21584 min_free 264748 largest_free_block 135908
Heap memory (in Data RAM) - after tasks

Heap summary for capabilities 0x00000006:
At 0x3ffae6e0 len 6432 free 0 allocated 6300 min_free 0
  largest_free_block 0 alloc_blocks 25 free_blocks 0 total_blocks 25
At 0x3ffbada8 len 152152 free 75752 allocated 75616 min_free 75368
  largest_free_block 75412 alloc_blocks 184 free_blocks 4 total_blocks 188
At 0x3ffe0440 len 15072 free 15036 allocated 0 min_free 15036
  largest_free_block 15036 alloc_blocks 0 free_blocks 1 total_blocks 1
At 0x3ffe4350 len 113840 free 113804 allocated 0 min_free 113804
  largest_free_block 113804 alloc_blocks 0 free_blocks 1 total_blocks 1

Totals:
  free 204592 allocated 81916 min_free 204208 largest_free_block 113804

Double is how many bytes? (8)
double track_data[20000] would overflow

70K allocated for tasks
EECS192 Lecture 3
Feb. 2, 2021

Notes:
• 2/9 project proposal – upload to bcourses by Tues 5 pm
• Survey for Cory Courtyard

Topics
• Checkpoints 1,2,3
• Memory Usage build/runtime
• Updated Skeleton code tour
• Peripheral interface: A/D, pulse count (intro)
• Sensor: line camera
Possible starting point for proposal. However, missing inputs, outputs, Steering, velocity, etc

Note conventions- data flow left to right
msg = input('Enter string "command value": ')  

temp= msg.split() # get separate command and value elements  

msg_bytes = struct.pack('8si',str.encode(temp[0]),int(temp[1]))  
# use structure so there is no need for risky sscanf on Huzzah32  
sock.sendto(msg_bytes, addr)  # send cmd + value over UDP

Example input: SPEED 120

Command is up to 8 characters, value is a 32 bit int

If python struct.pack is changed, need to also change skeleton.h

struct cmd_struct_def {  
    char cmd[8];  
    int value;};
```c
size_read = recvfrom(sock, &cmd_struct, sizeof(cmd_struct), 0, (struct sockaddr *)&source_addr, &socklen);
if (size_read <= 0)
    { printf("socket read error\n");  }

printf("command: %s value = %d \n", cmd_struct.cmd, cmd_struct.value);
xQueueOverwrite(cmd_queue, &cmd_struct);
// put command on queue, over writing any previous cmd

Example input: SPEED 120

cmd_struct.cmd = "SPEED", cmd_struct.value = 120

struct cmd_struct_def {
    char cmd[8];
    int value;};
```
// check for command waiting in queue
cmdready = uxQueueMessagesWaiting(cmd_queue);
if (cmdready >= 1) // get item from cmd queue
    if(xQueueReceive(cmd_queue, &cmd_struct,
                     portMAX_DELAY) != pdPASS) {
        printf("error reading from cmd_queue\n");
    }
    udp_cmd(cmd_struct.cmd, cmd_struct.value);
}

// handle commands from UDP
void udp_cmd(char command[], int value) {
    if (strcmp(command,"TIME") == 0) {
        get_time();
    }
    ...
Add other commands here, use string compare to parse
EECS192 Lecture 3  
Feb. 2, 2021

Notes:
• 2/9 project proposal – upload to bcourses by Tues 5 pm
• Survey for Cory Courtyard

Topics
• Checkpoints 1,2,3
• Memory Usage build/runtime
• Updated Skeleton code tour
• Peripheral interface: A/D, pulse count (intro)
• Sensor: line camera
ESP32 Hardware block diagram-peripherals

+GPIO
+ (Ch 18 T.R.M. Pulse Counter)
Sensors

**128 x 1 LINEAR SENSOR ARRAY WITH HOLD**

- 128 x 1 Sensor-Element Organization
- 400 Dots-Per-Inch (DPI) Sensor Pitch
- High Linearity and Uniformity
- Wide Dynamic Range . . . 4000:1 (72 dB)
- Output Referenced to Ground
- Low Image Lag . . . 0.5% Typ
- Operation to 8 MHz
- Single 3-V to 5-V Supply
- Rail-to-Rail Output Swing (AO)
- No External Load Resistor Required
- Replacement for TSL1401R–LF
- RoHS Compliant

---

**Description**

The TSL1401CL linear sensor array consists of a 128 x 1 array of photodiodes, associated charge amplifier circuitry, and an internal pixel data-hold function that provides simultaneous-integration start and stop times for all pixels. The array is made up of 128 pixels, each of which has a photo-sensitive area of 3.524 x 3 square micrometers. There is 8-μm spacing between pixels. Operation is simplified by internal control logic that requires only a serial-input (SI) signal and a clock.

**Functional Block Diagram**

Line camera: 128 pixels, 200 Hz

[Image of the Functional Block Diagram]

**Other options? Gyro sensor?**

https://www.sparkfun.com/tutorials/283
Velocity sensor mounting (preview - week 4)

3.3 V DC


Fig.9 Test Circuit for Response Time

? Analog or pulse count input?
ADC Converter (Ch. 30)

```c
adc1_config_width(ADC_WIDTH_BIT_12);
adc1_config_channel_atten(ADC1_CHANNEL_0,ADC_ATTEN_DB_0);
int val = adc1_get_raw(ADC1_CHANNEL_0);
```

https://github.com/espressif/esp-idf/tree/release/v4.2/examples/peripherals/adc

Claim: 2 M samples per second (Mfps) = 0.5 us
Pulse Counting (Ch 18)

github.com/espressif/esp-idf/tree/release/v4.2/examples/peripherals/pcnt

Figure 122: PULSE_CNT Upcounting Diagram

Figure 121: PULSE_CNT Architecture
Huzzah32 Pinouts

HEADER - 1X16_MIN

RESET

A0_DAC2 16
A1_DAC1 15
A2_134 14
A3_139 13
A4_1036 12
A5_104 11
SCK 10
MOSI 9
MISO 8
IO16 7
IO17 6
IO21 5

RST 3V NC GND A0 A1 A2 A3

3V

BAT EN USB13

BATTERY

ESP32

SCK

A4

A5

SCK

M0

M1

RX

TX

21

IO13_A12
IO12_A11
IO27_A10
IO33_A9
IO15_A8
IO32_A7
IO14_A6
SCL
SDA

HEADER

JP3

VPAT

VBUS

EN
# ESP32-WROOM Module Connections

<table>
<thead>
<tr>
<th>Name</th>
<th>No.</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>1</td>
<td>P</td>
<td>Ground</td>
</tr>
<tr>
<td>3V3</td>
<td>2</td>
<td>P</td>
<td>Power supply</td>
</tr>
<tr>
<td>EN</td>
<td>3</td>
<td>I</td>
<td>Module-enable signal. Active high.</td>
</tr>
<tr>
<td>SENSOR_VP</td>
<td>4</td>
<td>I</td>
<td>GPIO36, ADC1_CH0, RTC_GPIO0</td>
</tr>
<tr>
<td>SENSOR_VN</td>
<td>5</td>
<td>I</td>
<td>GPIO39, ADC1_CH3, RTC_GPIO3</td>
</tr>
<tr>
<td>IO34</td>
<td>6</td>
<td>I</td>
<td>GPIO34, ADC1_CH6, RTC_GPIO4</td>
</tr>
<tr>
<td>IO35</td>
<td>7</td>
<td>I</td>
<td>GPIO35, ADC1_CH7, RTC_GPIO5</td>
</tr>
<tr>
<td>IO32</td>
<td>8</td>
<td>I/O</td>
<td>GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9</td>
</tr>
<tr>
<td>IO33</td>
<td>9</td>
<td>I/O</td>
<td>GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8</td>
</tr>
</tbody>
</table>

**JP1-8**

**JP1-9**

**JP1-10**

**Vref?**

**JP3-9**

**JP3-7**

*P=power  
I= input only  
I/O = either*
# ESP32-WROOM Module Connections

## Huzzah32

<table>
<thead>
<tr>
<th>Name</th>
<th>No.</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP1-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO25</td>
<td>10</td>
<td>I/O</td>
<td>GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0</td>
</tr>
<tr>
<td>JP1-12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO26</td>
<td>11</td>
<td>I/O</td>
<td>GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1</td>
</tr>
<tr>
<td>JP3-6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO27</td>
<td>12</td>
<td>I/O</td>
<td>GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV</td>
</tr>
<tr>
<td>JP3-10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO14</td>
<td>13</td>
<td>I/O</td>
<td>GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2</td>
</tr>
<tr>
<td>JP3-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO12</td>
<td>14</td>
<td>I/O</td>
<td>GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3</td>
</tr>
<tr>
<td>JP3-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>15</td>
<td>P</td>
<td>Ground</td>
</tr>
<tr>
<td>JP3-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO13</td>
<td>16</td>
<td>I/O</td>
<td>GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER</td>
</tr>
<tr>
<td>SPI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCK/CLK*</td>
<td>20</td>
<td>I/O</td>
<td>GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS</td>
</tr>
<tr>
<td>SDO/SD0*</td>
<td>21</td>
<td>I/O</td>
<td>GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS</td>
</tr>
<tr>
<td>SDCI/SD1*</td>
<td>22</td>
<td>I/O</td>
<td>GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS</td>
</tr>
<tr>
<td>Flash</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P=power  
I= input only  
I/O = either
## ESP32-WROOM Module Connections

<table>
<thead>
<tr>
<th>Huzzah32</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JP3-8</td>
<td>IO15</td>
<td>23</td>
<td>I/O</td>
</tr>
<tr>
<td>RTS?</td>
<td>IO2</td>
<td>24</td>
<td>I/O</td>
</tr>
<tr>
<td>DTR?</td>
<td>IO0</td>
<td>25</td>
<td>I/O</td>
</tr>
<tr>
<td>JP1-7</td>
<td>IO4</td>
<td>26</td>
<td>I/O</td>
</tr>
<tr>
<td>JP1-3</td>
<td>IO16</td>
<td>27</td>
<td>I/O</td>
</tr>
<tr>
<td>JP1-2</td>
<td>IO17</td>
<td>28</td>
<td>I/O</td>
</tr>
<tr>
<td>JP1-6</td>
<td>IO5</td>
<td>29</td>
<td>I/O</td>
</tr>
<tr>
<td>JP1-5</td>
<td>IO18</td>
<td>30</td>
<td>I/O</td>
</tr>
<tr>
<td>JP1-4</td>
<td>IO19</td>
<td>31</td>
<td>I/O</td>
</tr>
<tr>
<td>NC</td>
<td>32</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JP1-1</td>
<td>IO21</td>
<td>33</td>
<td>I/O</td>
</tr>
<tr>
<td>UART</td>
<td>RXD0</td>
<td>34</td>
<td>I/O</td>
</tr>
<tr>
<td>JP3-11</td>
<td>TXD0</td>
<td>35</td>
<td>I/O</td>
</tr>
<tr>
<td>JP3-12</td>
<td>IO22</td>
<td>36</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>IO23</td>
<td>37</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>GND</td>
<td>38</td>
<td>P</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I/O = either</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>P = power</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>I = input only</th>
</tr>
</thead>
</table>
EECS192 Lecture 3
Feb. 2, 2021

Notes:
• 2/9 project proposal – upload to bcourses by Tues 5 pm
• Survey for Cory Courtyard

Topics
• Checkpoints 1,2,3
• Memory Usage build/runtime
• Updated Skeleton code tour
• Peripheral interface: A/D, pulse count (intro)
• Sensor: line camera
TSL 1401 line sensor

Functional Block Diagram

SI: serial input. SI defines the start of the data-out sequence.
AO: analog output
CLK: clock
TSL 1401 line sensor

Figure 1. Timing Waveforms

Figure 2. Operational Waveforms

From PWM?
From PWM or GPIO?
TSL 1401 line sensor - option exposure control

Exposure time (>25 us)

SI

129 clocks

CLK

A/D read
~12 us

AO

Clock only
no A/D read

Line sensor
Automatic Gain Control

- Choose exposure time based on average illumination
- Keep frame rate constant e.g. read sensor twice $1+4 \Rightarrow 4 +1 \text{ ms}$
- (Constant time is important for control- will see later)

In all the discussion that follows, we will be using one-shot imaging.
Possible algorithms for line detection

e.g. prototype with scipy.signal.filter. Many options. Here are 3 suggestions:

• Subtraction- to find left and right edge of line (ok if not noisy, somewhat lighting invariant)

• Difference of Gaussians (idea is to smooth then differentiate)

• Correlation (best match position for known features)
  – scipy.signal.correlate

Try out algorithms in HW1
Alternative #1 frame subtraction

TSL 1401 line sensor 8 bit

Frame 0

Frame 1

Frame 1-Frame 0

Notes: peak shows edge of track. Noisy, only 1 pixel resolution.
Alternative #2 Difference of Gaussians

Laplacian of Gaussian

\[ \Delta[G_\sigma(x, y) \ast f(x, y)] = [\Delta G_\sigma(x, y)] \ast f(x, y) = \text{LoG} \ast f(x, y) \]

Convolve with Difference of Gaussians kernel (approx. to LoG)

\[ \Gamma_{\sigma_1, \sigma_2}(x) = I \ast \frac{1}{\sigma_1 \sqrt{2\pi}} e^{-(x^2)/(2\sigma_1^2)} - I \ast \frac{1}{\sigma_2 \sqrt{2\pi}} e^{-(x^2)/(2\sigma_2^2)}. \]

Consider \( \frac{\partial^2}{\partial x^2} (h \ast f) \)

Note: zero crossing is edge location
Alternative #3 Correlation

\[ \text{arg min} \; \| I(y) - f(y - \Delta y) \|_2 \]

Notes: normalize, find by least squares or search. Can use \( \Delta y(n-1) \) to initialize.
Summary

Memory Usage build/runtime
Updated Skeleton code tour
Peripheral interface: A/D, pulse count (intro)
Sensor: line sensor

Keep in mind limited time and memory
Extra Slides
DC Motor Physical Model

\[ \vec{F} = il \times \vec{B} \]

\[ \tau = \vec{r}_1 \times \vec{F}_1 + \vec{r}_2 \times \vec{F}_2 \]
Motor Electrical Model
(neglect inductor)

Motor Electrical Model
Back EMF
Motor electromechanical behavior

Continued on board
Also- see motor worksheet……

Note: missing e-stop!
For this problem, consider a DC permanent magnet motor (as used in your car). The car is on a carpet and moves in a straight line with no slip between the wheels and the carpet. The car is initially moving at a speed of 2 meters per second.

You can assume a motor model as shown below. The qualitative shape of the curves is more important than magnitudes.

\[ e = k_e \frac{dv}{dt} \]

\[ v(t) \]

\[ i_m(t) \]

\[ x(t) \]

\[ \text{Let peak speed} = 5 \text{ m/sec} \]
\[ \text{Accel} = 5 \text{ m/s}^2 \]
\[ k_e = 1 \text{ v/(m/sec)} \]

On board

(For answer see sp99 final solution)
\[ <i_m> = \left( \frac{T}{T_0} \right) i_{\text{max}} \]

Is \( i_{\text{max}} \) constant?
Motor Electrical Model

Back EMF
Motor electromechanical behavior

Also- see motor worksheet……

\[ i_m = \frac{V_{BAT} - k_e \dot{\theta}_m}{R_m} \]

Conclusion:
\[ \langle i_m \rangle = ? \]
Motor Model

On board....

http://inst.eecs.berkeley.edu/~ee192/sp18/files/NiseAppendixI.pdf

http://inst.eecs.berkeley.edu/~ee192/sp13/pdf/motor_modeling.pdf