Type	Bits	Base
Hex (Nibble)	4	2
Byte	8	16

Hex = 0 1111
bit = 1
nibble = 4 bits => 1111
byte = 8 bits => 1111 1111 = 2 Hex

Data Type	**32bit bytes**	**64bit**
Char	1	1
Short	2	2
int	4	4
Long	4	8
Long long	8	8
float	4	4
Double	8	8
Long Double	16	16

```
0b11111111   = 0xFF
0b11111111   >> 2 = 0b00111111    = 0x2F    // note:  truncates if allocd only for 1 byte
0b11111111   << 2 = 0b11111100    = 0xFA
```

Tell About Yourself

Tell us a little bit about yourself:

Program at a very early age beginning with a scripting language then c++. Most of my spare time is contributed to my own personal side projects. This can range from anything I can think of that would be useful or provides merits of learning. If I were to describe myself I would say I am a meticulous, methodical worker.

Describe yourself: meticulous, an excellent mentor, patient, willing to learn

Most proud of: instrumentation, rapid development with short schedule, coordinating with others

Describe When

What did you work on yourself: memory coordinator, prototyping the instrumentation in python then began reworking the code base and optimizing it. I had another new hire join and help me integrate new sensors. These new sensors were introduced as the result of requirement changes

Pursue in 5 Years: space platforms and I gained a huge interest in satellite systems due to my navigation experience.

Describe a Time Something went wrong: For our robotics competition, 2 weeks prior to the event we realized our dimensional measurements of the robot's clearance would disqualify us. Fortunately, I had tested another set of sensors that were the next likely candidate (only difference was slightly less range). So we took out the current sensors, fit the others in and

began testing the robot. Within several trials of adjusting thresholds of the new sensor's range profile the robot worked as intended. We were able to compete in the competition and won within our university competition.
Describe a time you had a conflict with a coworker: There was a time when I had a group assignment project. We would get together late at night and work on the project. We had one individual that would never show up to the night-hour group project sessions. This went on for couple weeks and then it began to affect our schedule because we really needed a hand. Our team lead didn't want to bug him. The individual noted that his classes were important to him and he didn't have time to contribute the hours. The others were not able to fit his time because of class conflicts. At first I felt like he disregarded everyone elses' effort but then I thought wow he was honest and upfront. So that opened up communication. So with that I asked if he would like me to adjust my schedule in the mornings to fit his to fill him in on how the project status. I set aside time for morning and late nights to sit down and work on the project. In the end the class

Software Experience: roughly 2years in grad school, 3 years' work experience, additional 1 year to open source project contributions

POSIX : Portable operating system compliant -> need not worry about porting and I can focus on the correctness of a programs implementation

UML diagrams, sequence diagrams, visualizing relationships, how to improve better design a system using Lucid Chart

Edwards Experience

UAVs
Instrumentation engineer – I am the engineer responsible for building a sensor system payload for our platforms. The system is designed to record air sensor data on board during flight performance maneuvers. The data is then sent to our data modeling experts to package it into a model to be used with simulators

Group 3 size of a car. Next platform is group 4 which will have a much greater wingspan but similar length in fuselage.

Instrumentation

Instrumentation

- features multithreading, a downlink stream GRPC
- sensors vary collecting INS data, position sensors, ECU and an air data boom
- each sensor has a driver code: connect, read, shutdown
- coordinator top level program: multithreaded, orchestrates when to begin recording, stop recording with a start signal
- a ground base station
- MPU4 Wave Relay Radio 900-922MHz as a wide area network

- **UART**
  - only one slave one master
  - 9600 to ~100k baud
  - within 10% timing error
  - parity checking
- - limited to 2-9 bits
-
- **Serial Communication**:
  sends data chunks at large burst

Producer Consumer

producer consumer
create 2 threads and attach routines()
share a queue
producer -> read data, package, store into queue then notify
consumer -> enters, block on notify, when notified unblock, grab from queue, process data

```
producer(){
    while(){
        create data
        grab lock
        push to queue
        unlock
        notify
    }
```

```
consumer(){
    while(){
        if queue is not empty{
            grab lock
            take from queue
            unlock
            process data
        }
    }
}
```

Avionics Engineer

- avionics discipline engineer
- focus on navigational systems like an onboard AHRS system (Attitude Heading Reference System) planning test points, maneuvers
- Ground base stations
- Data visualization
- Provide technical reports, findings, deficiencies

Memory Coordinator

Memory Coordinator
Hypervisor – (hardware/software) that sits on top of the kernel which manages Guest Virtual Machines: create/destroy. This uses a software hypervisor LibVirtManager

HV: provides a Memory Balloon mechanism to manage memory
balloon expands/contracts to provision memory to VMs
-HV is the host OS, Guest is VMs OSes
-LibVirt (virtualization API) which provides an interface for hypercalls to manage memory.
getMemoryStat(VM)

// assumes VMs are already running
The program works like so:
Connect to VMs through localhost:port

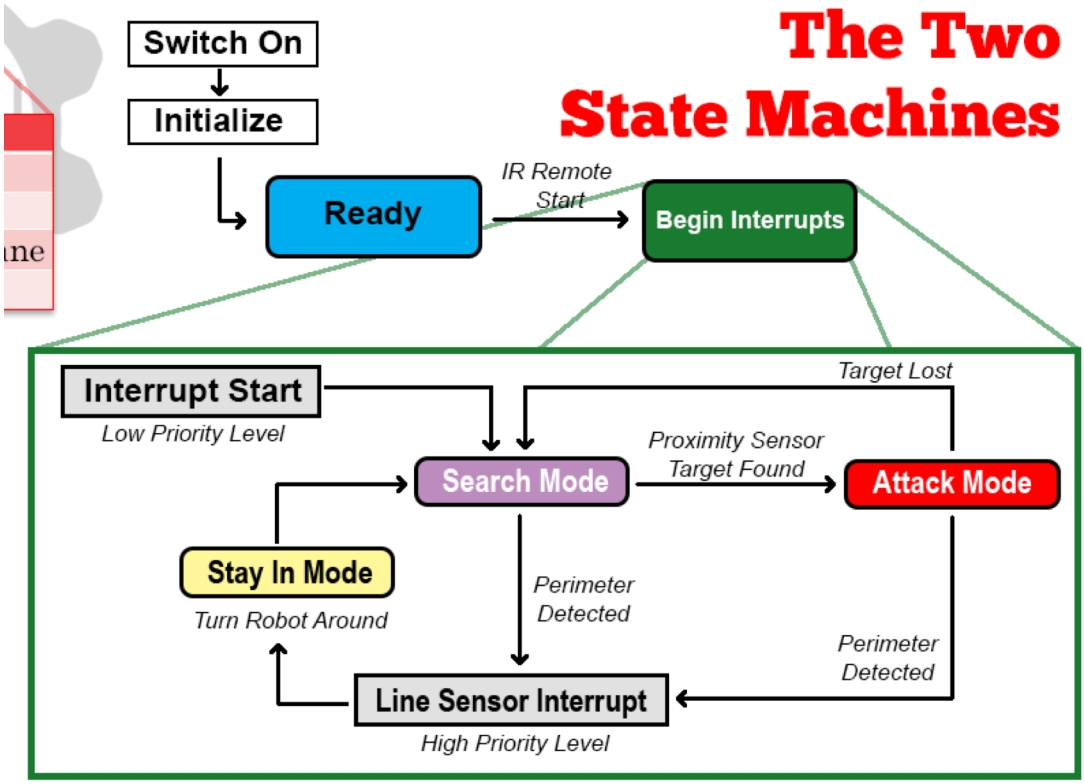
Poll VM
continuously poll memory statistics memory usage/free for all VMs
we create some rules for our threshold: starved vs wasteful VMs

HV will poll constantly to get statistics
HV sees Wasteful OSx (not actively using memory) = inflate balloon (increase pressure)
forces OS to free memory and give back to HV
HV recv memory and adds it to its free pool of memory
HV sees Starving OS
HV tells balloon to deflate (reduces memory pressure)

OMUS Sumo Robot

Omus
pic32 processor
8 IR sensors (4 edge, 4 proximity)

-Developed requirements for sensors: analog or digital, noise, range, response times
-Selected based on past competitions most competitors moved fast enough -> analog sensors for faster response time
- test the sensors to measure voltage (purchased bulk and selected with lowest error ex.3m = X volts)
- hooked up to PIC32 analog pins and read values
- hooked up pic32 with signal noise generator and ensued values were reading properly
- used signal noise generator to simulator sensors and devise the logic of state machine



OMUS:
Enable GPIO Pins (input sensors and Output motors)
Enable multi_vectory_mode (allows multiple interrupts)
Enable interrupts (Interrupt Enable Control Register, Interrupt Priority Register)
Assign Interrupt Service Routines (ISR) -> Line sensor has higher priority
Proximity sensors ran in main

Src: Example 8-4
```
IPC0CLR = 0x0000001C; // clear the priority level
IPC0SET = 0x00000008; // set priority level to 2
```

Build around state machine: ->search , attack, stay-in-arena

Code Pseudo:
setup init pins and timers and interrupts

```
while(){
    // poll sensors
    poll ground sensors -> if sees line then set flag=1, perform ISR, set flag to 0
    poll proximity sensors -> set analog values
      wait
      seek  // if analog > 300 object detected, rotate motors
      escape
      attack1 // if analog > 300 (close) attempt to push 80% speed
      attack2 // if analog > 700 (far) full speed push at 100%
```

Context Switching

Context Switching Form Process A to B

Copy context of process A
Update Page Table Block Register to point to A
Push PCB to Ready Queue of Processes/Tasks
Schedule Process B
Unload data structures from memory for process B
Update PTBR to B
Load State of Process B
Set Process B to Is Running then Execute

Thread Pool

thread pool:

```
// init
for( threads attach routine)
   add vector threads

// dismiss
notify all threads that are blocked on notify
join each

// thread has a routine
while(!dismissed){
       if(queue not empty){
             get item in front
             pop
             return
       }
       block on notify
}

template <type function>
// queue <task>
queue.enqueue(<task>)

// routine
thread loop(){
     while(running){
          task = getNextItem()
          task();
     }
}
```

C vs C++ vs Python

C "function driven language"	C++ "object oriented	Python
-Procedure oriented - procedure - record - module - procedure call * data and functions are separated	-Object oriented & Procedural -method - object - class - message (call) * data and functions are encapsulated in objects	Interpreted Language like Java (requires the Python Interpreter similar to how Java requires the Java Virtual Machine) does translation to byte code for machine to understand
- no support for abstraction, inheritance, polymorphism, Encapsulation	Has support for all single and multiple inheritance	Supports all types of inheritance
No namespaces	Has namespace	

Favorite C++ stuff

Favorite C++:

smart pointers: unique_ptr, shared_ptr - owns and manages another object through a pointer and disposes the object when the pointer goes out of scope

RAII – Resource Acquisition is Initialization

P threads

```
while(){
    lock(mutex)
    //
    unlock(mutex)
}
```

C++ threading(lock is wrapped around an object)

```
while(){
   {
       std::unique_lock<std::lock> locker<mutex>
    }
   // when goes out of scope is destroyed
}
```

Multithreading Dead lock vs Race Condition

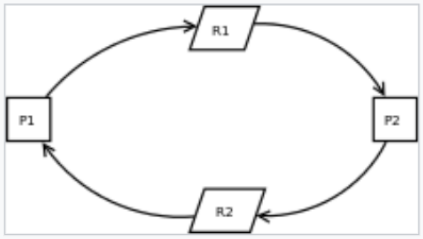
Deadlock: conflict of shared resource ex 2 threads trying to grab a lock.
Race Condition:

data race

```
// Shared variable
var count = 0

func incrementCount() {
  if count == 0 {
    count ++
  }
}

func main() {
  // Spawn two "threads"
  go incrementCount()
  go incrementCount()
}
```

!data race

```
Thread 1    Thread 2

lock(l)     lock(l)
count=1     count=2
unlock(l)   unlock(l)
```



Both processes need resources to continue execution. *P1* requires additional resource *R1* and is in possession of resource *R2*, *P2* requires additional resource *R2* and is in possession of *R1*; neither process can continue.

Why block on While and not If = Spurious Wake Ups

[While blocks] is better than [if blocks] for busy waiting threads to protect against spurious wakeup (when a waiting thread awoken even without been notified)

Atomic

Atomic – in concurrent programming atomic is an operation is guaranteed to be executed by one thread. no other threads will see it (even at partial state).

Mutex

Mutex = Mutual Exclusion – locking mechanism, allow multiple threads to share access but not simultaneously.

Semaphores

Sempahore - signaling mechanism. Threads waiting on semaphore can be signaled by another thread. Different to mutex because Mutex can only be called by thread calling wait() function

1. Counting Sempahore = number = numAvailableResources
   if <= 0 no operation performed
2. Binary Semaphore – similar to counting but restricted to 0 or 1

Mutex:

```
while(){
   Wait for mutex (grab lock){
      // critical section
   }
   cond notify
}
```

Sempahore Counting

```
while(s){
     while (S<=0) //wait
     S--
}

signal(S){
    S++;
}
```

Sempahore Binary

```
while(){
     while(S==0)  //wait
     S--
}

signal(S){
    S++;
}
```

Kernel Trap

Kernel Trap:

```
//user mode executing code
1. file write traps to kernel
2. passes control to the kernel
3. a bit in the kernel is reset (for next trap)
//kernel validates,
4. if !valid then return (fail silently)
5. perform file write I/O
6. hand back to user control
7. continue where trap occurred
```

Hypervisor Explained

Src:https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/VMware_paravirtualization.pdf

Hypervisor – is OS that sits on top of bare metal hardware
    It can spawn/manage/remove VMs
    does binary translation on the fly

In fully virtualized OS
-guest OS requires no modification
OS manages a page table
VPN-> PPN->MPN which uses a shadow table where OS thinks it sees direct hardware but it does not
OS -> maintains pagetable (VPN->PPN)
Hypervisor maintains hardware page table (PPN->MPN)

if memory is referenced in program then it will reference its page table
walk page table
if address exist (TLB hit)
    bypass translation and use direct mapping
    return direct mapping PPN-> machine address
no exist fetch address (TLB miss)
    make i/o call to fetch page from memory
    block for return
    update page table to map to the machine address

Paravirtualized (assisted virtualization)
-requires OS to be modified in code (Xen uses modified kernel and I/O calls)
-Uses hypercalls() installed to communicate kernel interface
-OS is unaware it is virtualized, hypercalls() are trapped by binary translation
-VMWARE IS NOT Paravirtualized (but is lightly paravirtualized and uses some techniques of para but truly isnt)

Also new form Hardware Assisted
provides new flags and hyper calls

OOP Concepts

Encapsulation	Private Public Accessors Protected only modifiable by inherited class ex. Private attributes and public accessors/methods
Abstraction	Class interfaces provide intent and abstract implementation Ex. Class shouldn't need to know the implementation of another class, just use it
Inheritance	**"**is a" or "has a" relationship between 2 objects single – 1 base 1 class multiple – derive multiple times going from generic to more specification (but one base similar to single inheritance)

	hierarchical inheritance – from multiple brands hybrid – everything combined
Polymorphism	Provide implementation of multiple definitions Static: method overloading – compile time Compiler knows as it checks the type and number of args same name but different ex void print(float) void print(int)
	Dynamic: method overriding – runtime method set as virtual then override keyword for new method to define the implementation concrete implementation of the method of the virtual method (may differ per class such such as read() we know sensor reads but the protocol and implementation is not given) Ex Virtual Class Sensor Virtual init virtual read Virtual shutdown() Inherit from Sensor : Gps() void init() void read() void shutdown()

Questions:

Collaboration Tools:
What kind of language:
How big is the team on a project:
How are projects typically managed
Next Steps