

Energy Efficient Ranging on NTB_v2

EE202B/CSM213B Winter 2017

Yan Zhang
Yi-Fan Zhang

“If I had an hour to solve a problem I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions.”

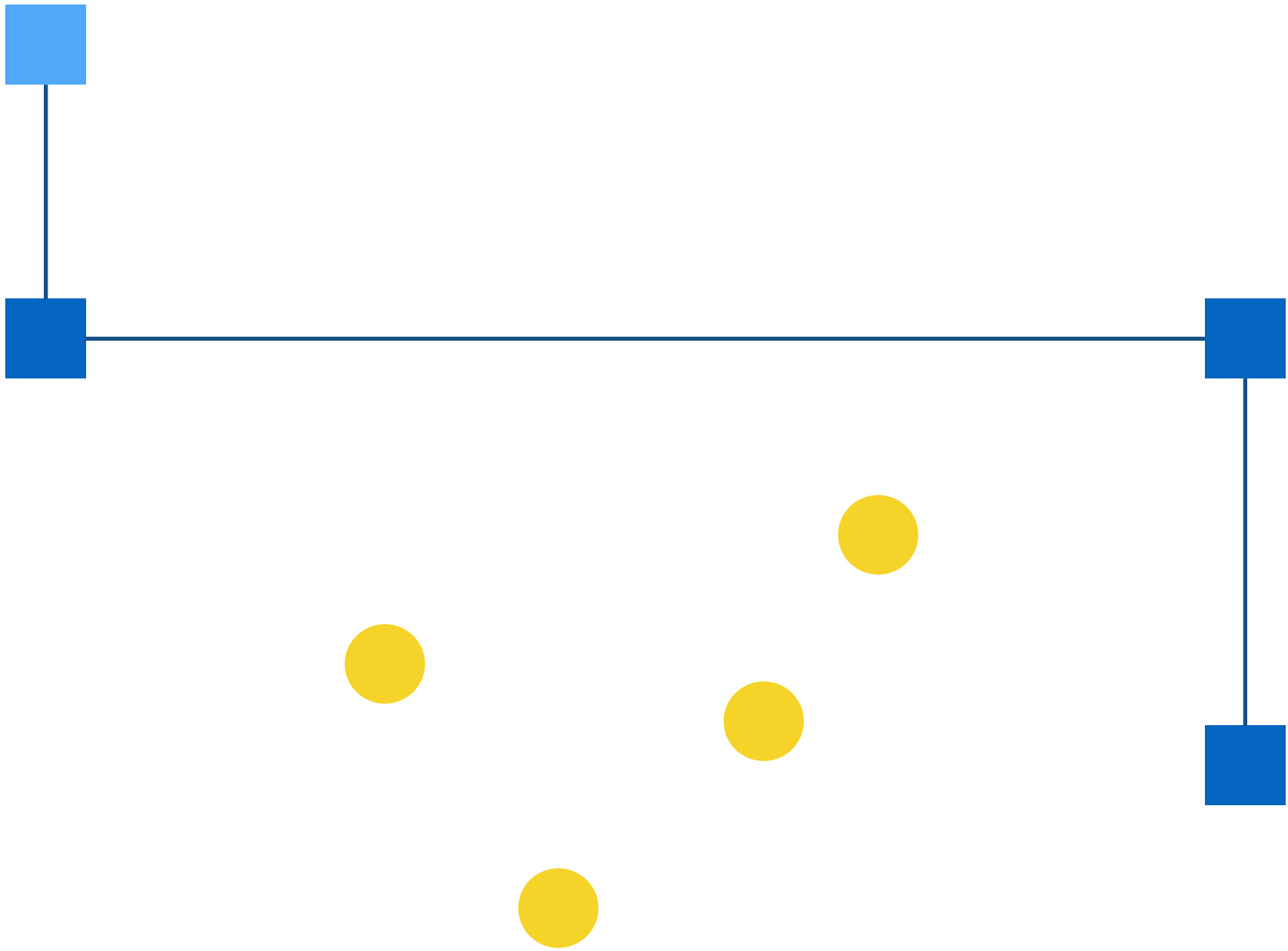
-Albert Einstein

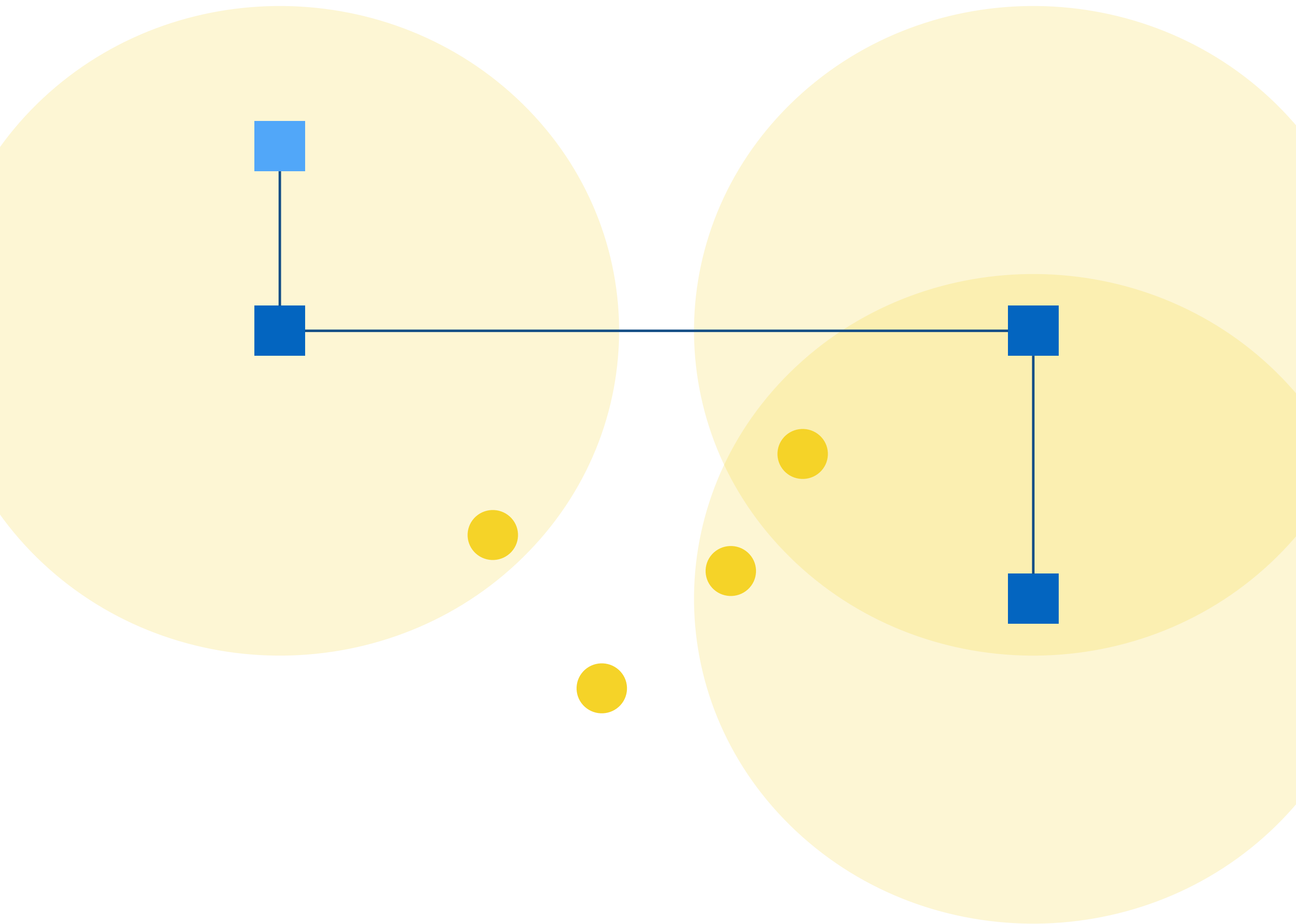
Precursor to PLoTs: groundwork and improvements made to the ntb_v2 testbed.

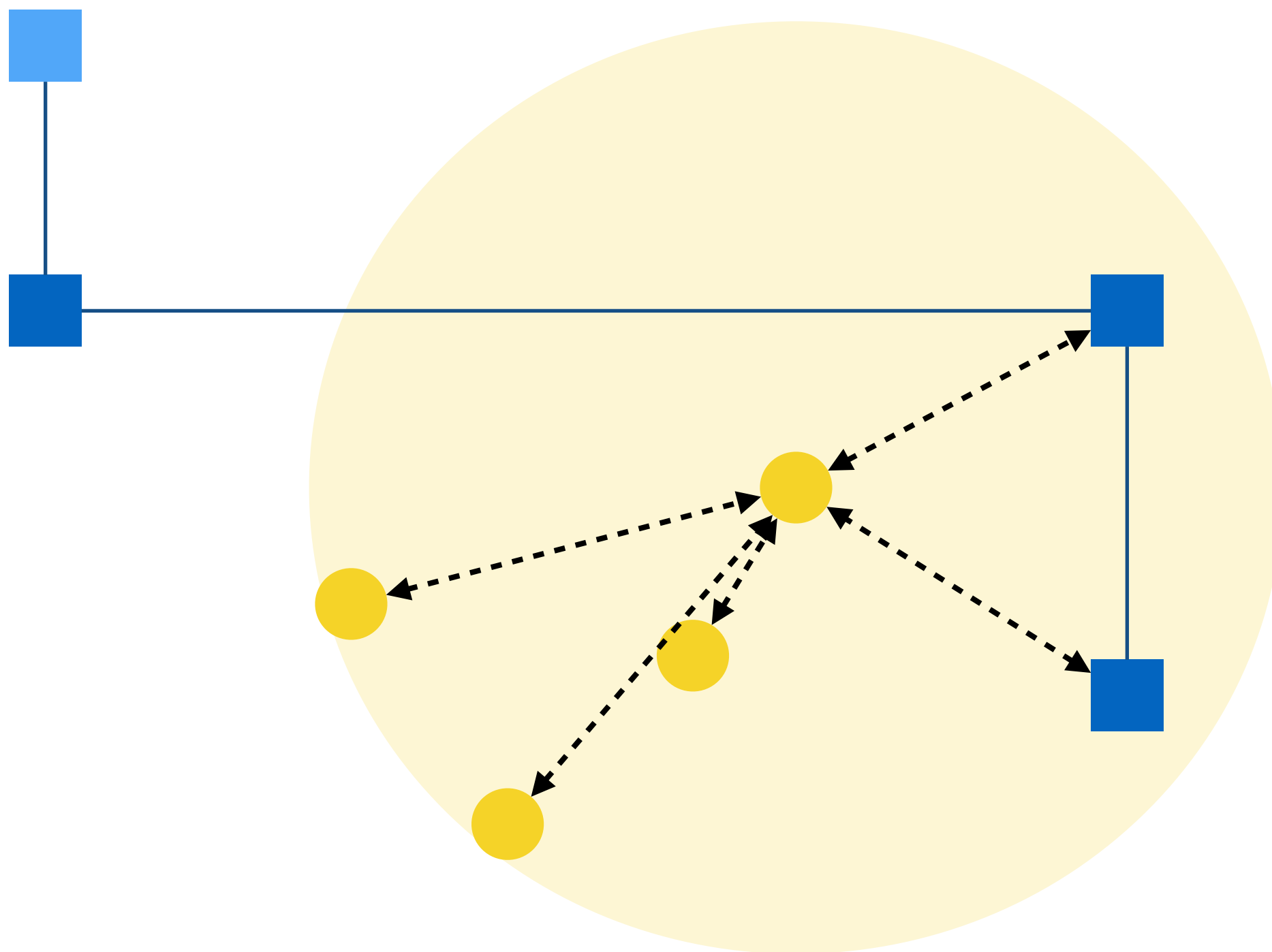
PLoTS: Power Efficient Localization and Time Synchronization

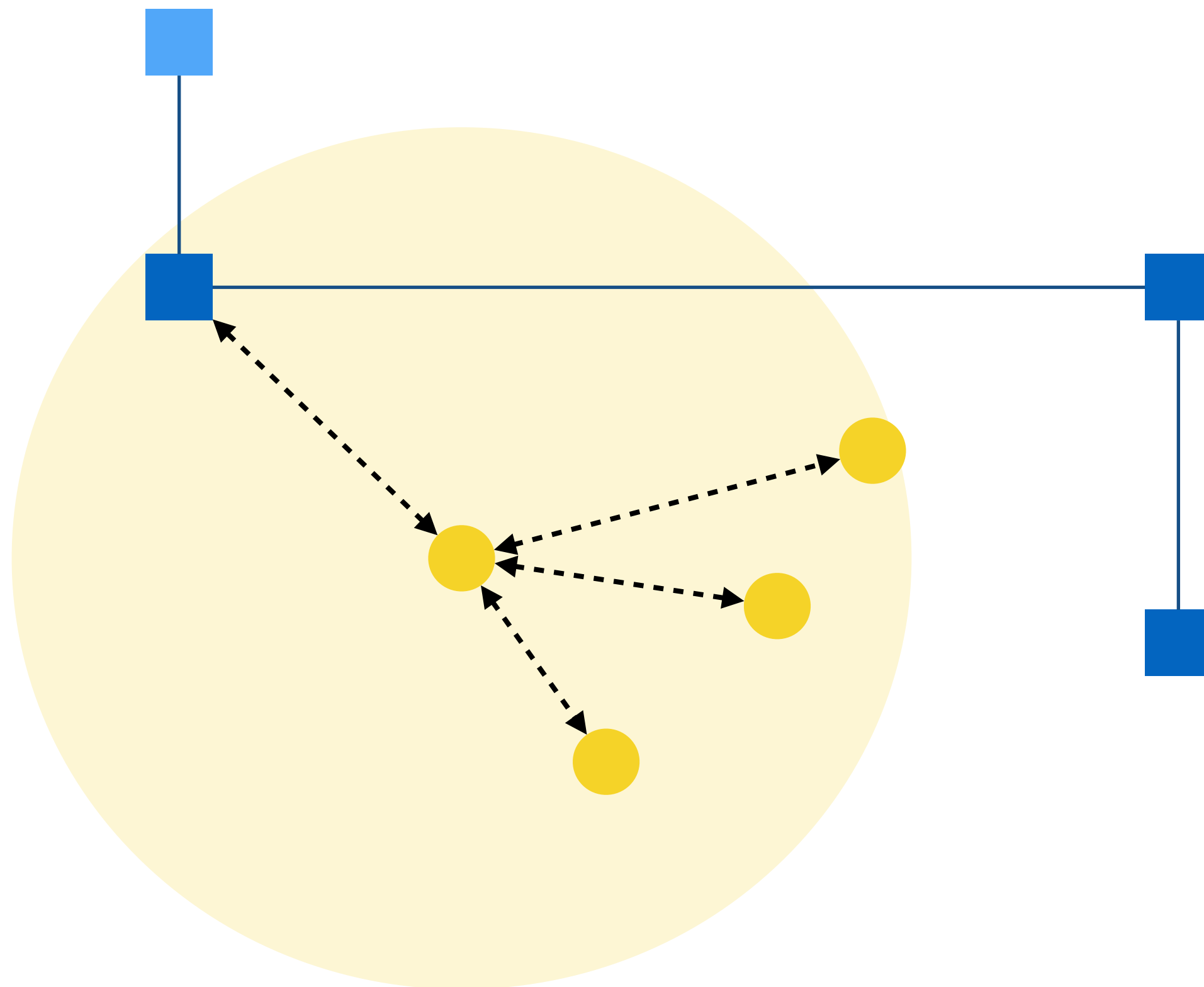
Hani Esmaeelzadeh, Amr Alanwar

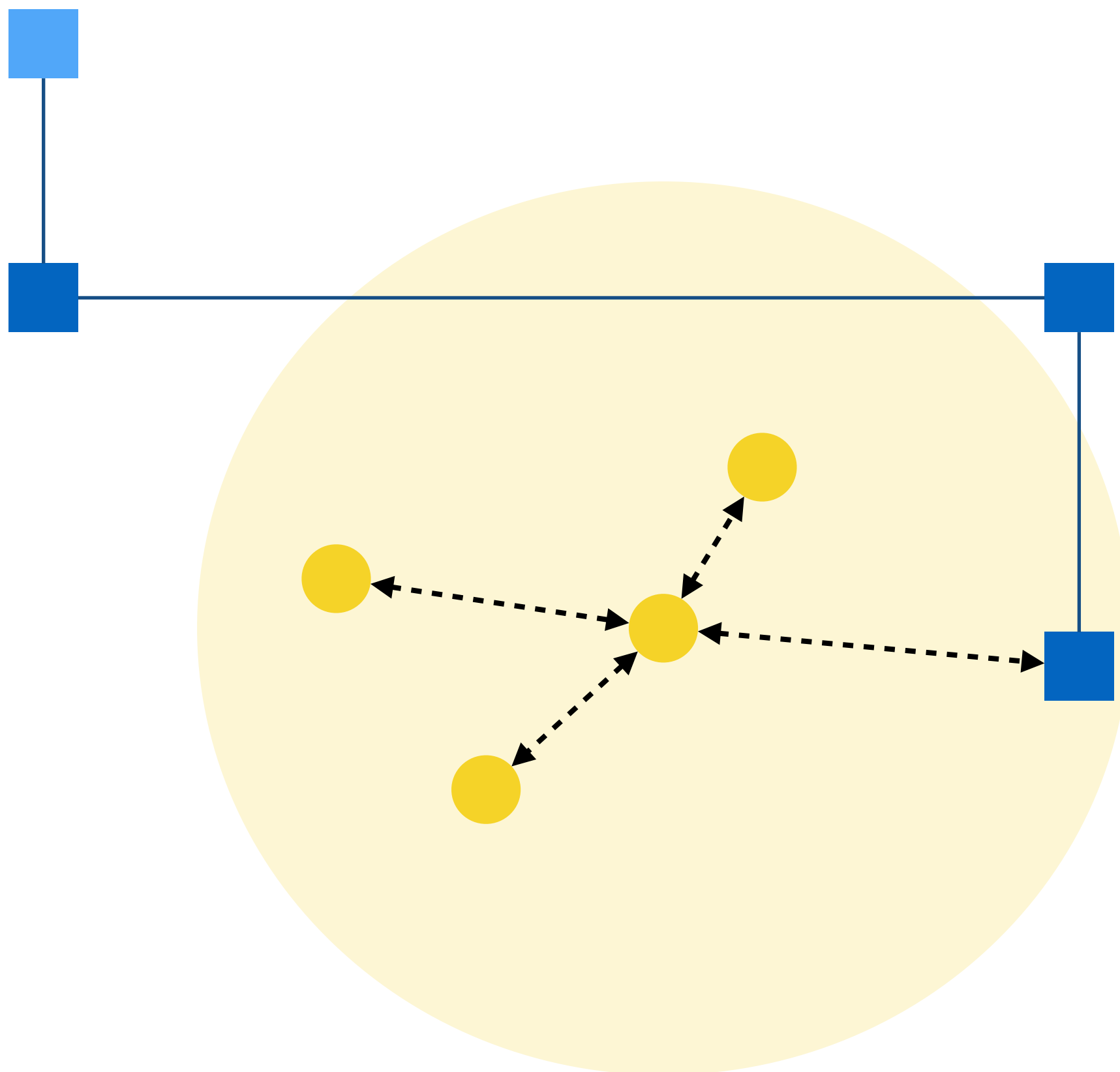
Abstract—Last decade has witnessed a widespread in the Internet of Things (IoT) devices which cause a dramatic increase in power consumption. Therefore, IoT technology is driving energy innovation towards a better solution to achieve high performance and low power consumption. Moreover, localization and time synchronization are two important challenges in IoT field. However, these three problems namely, localization, time synchronization, and power saving are traditionally treated separately. Therefore, we propose *PLoTS*: a power efficient distributed framework for localization and time synchronization for IoT devices. *PLoTS* aims to introduce an efficient IoT device in terms of power consumption and capable of accurate simultaneous localization and time synchronization. *PLoTS* exposes Kalman filter capabilities towards power saving using efficient duty cycling. Furthermore, a fast start-up crystal oscillator using precisely-timed energy injection technique is proposed to facilitate duty-cycling and increase the battery life. Our initial measurement results demonstrate a 15x faster start-up over prior state-of-the-art. Also, our preliminary results achieved one-meter localization error and sub-microseconds synchronization error.

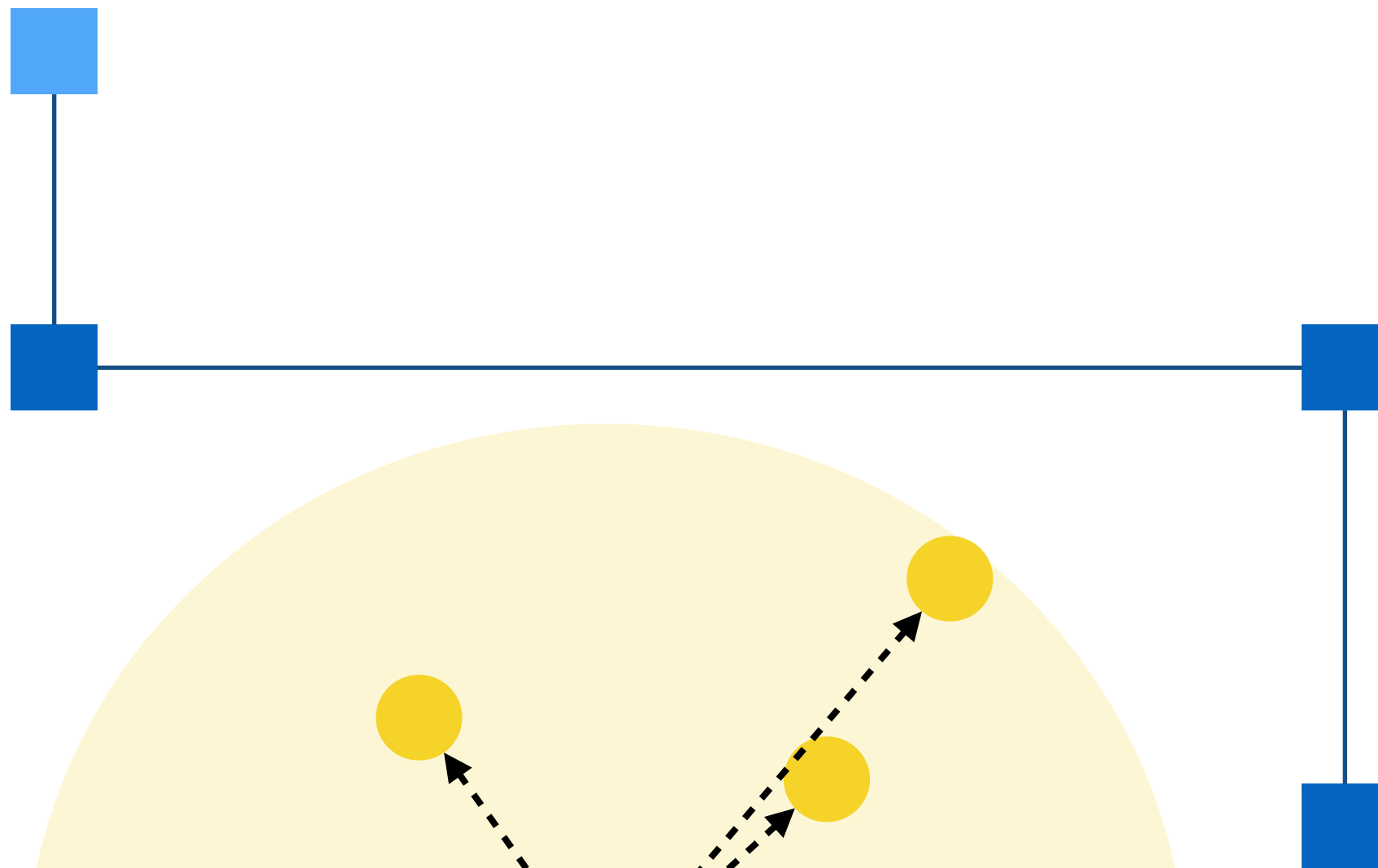




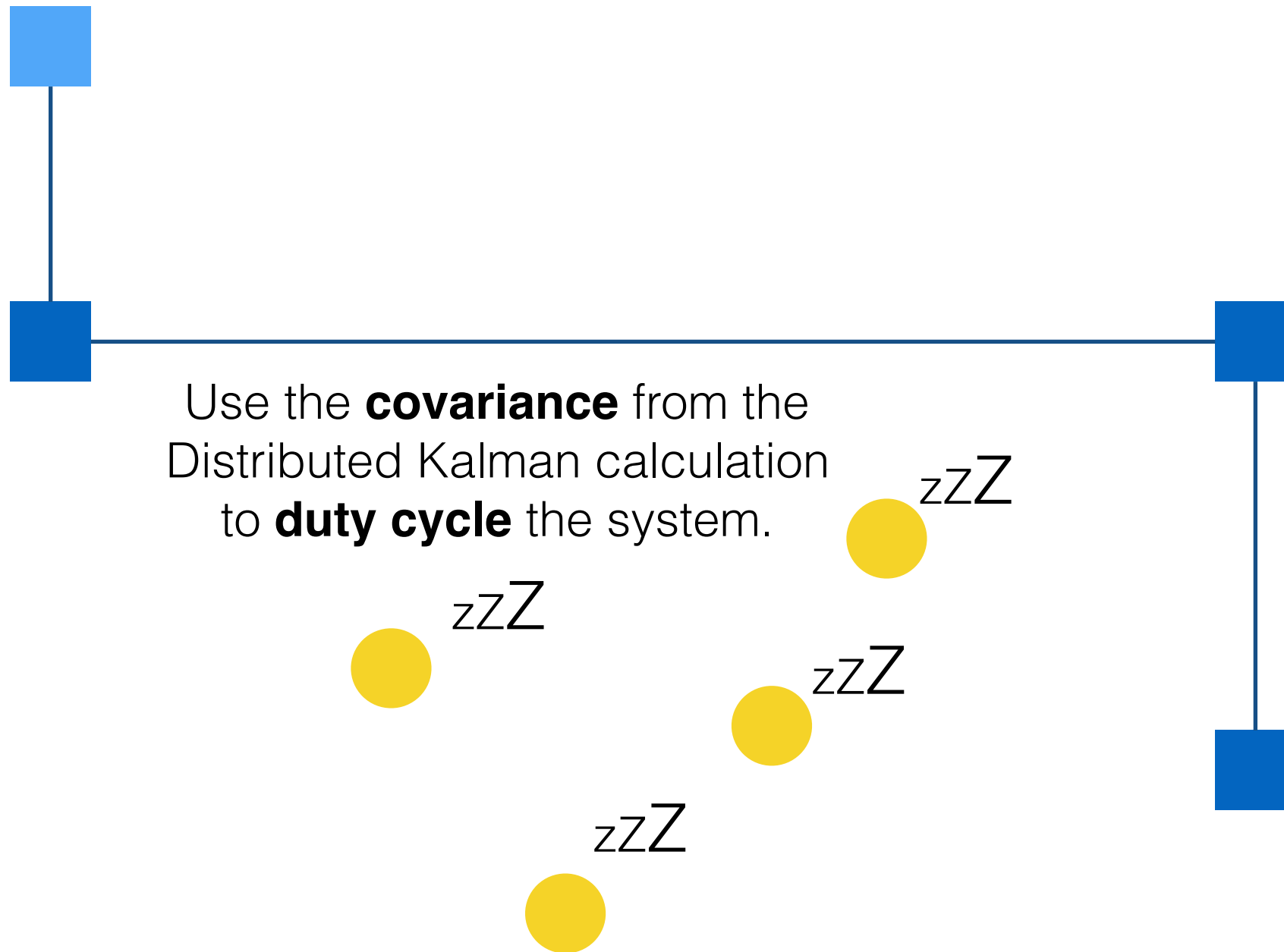








After multiple iterations,
every node eventually learns
the global state of the system.



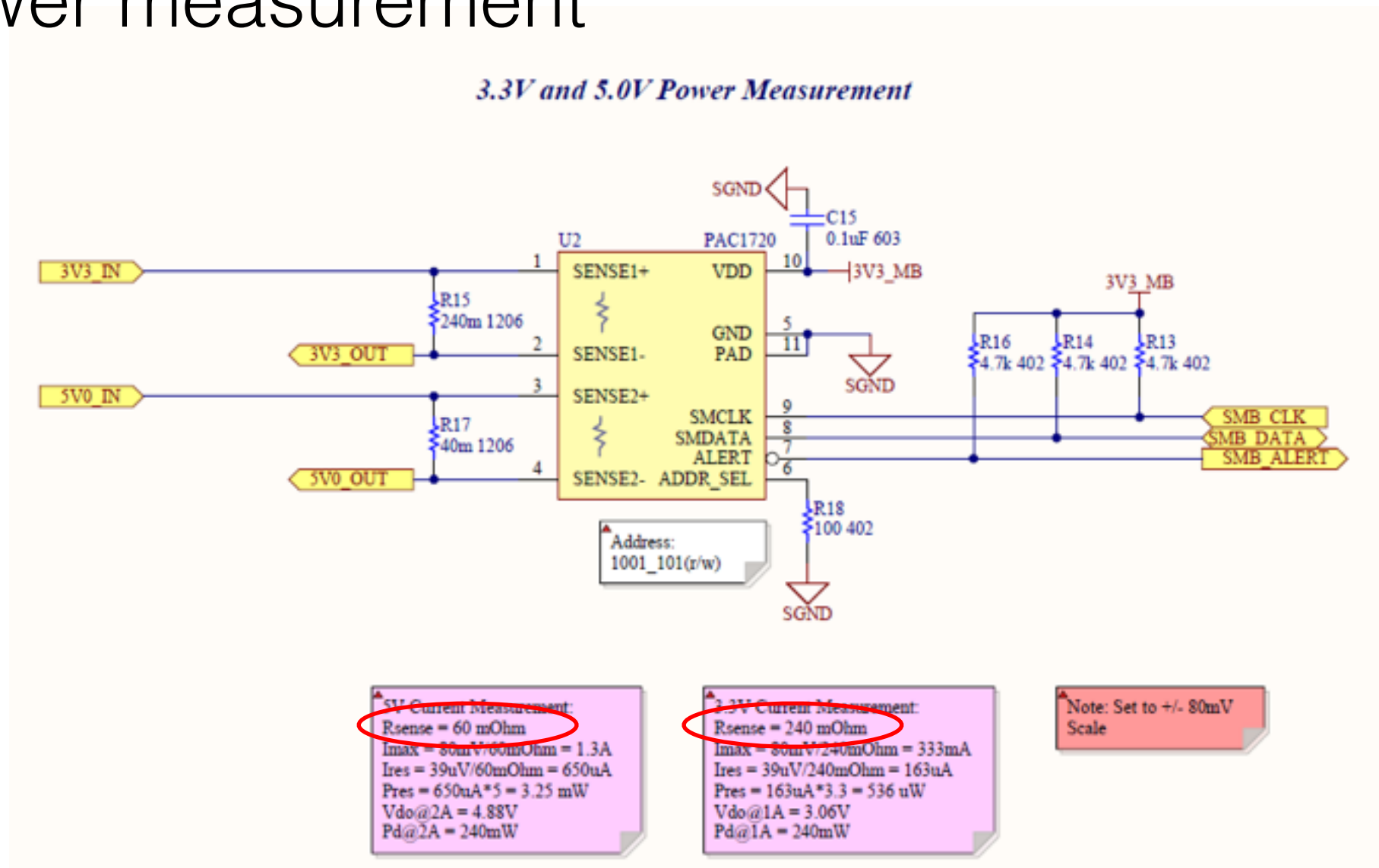
- 1. Hardware/Firmware (How to duty cycle?)**
- 2. Software/Algorithms (When to duty cycle?)**

Hardware Modification

- Objectives
 - Be able to measure MCU power
 - Incorporate oscillator design with “crystal pre-energization”
- Additional Objectives
 - Minimal impact to original design and performance
 - Options, options, and options

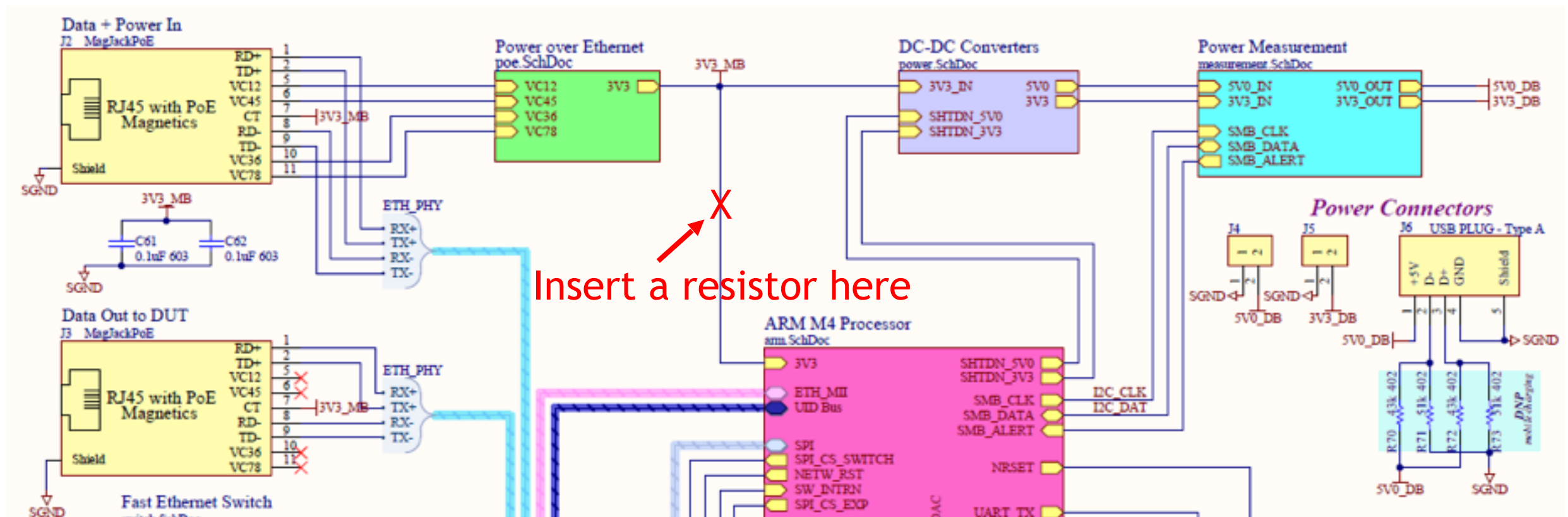
Measure MCU Power

- Similar to what Paul has done on the NTB board, current sensing points can be established for power measurement



Measure MCU Power

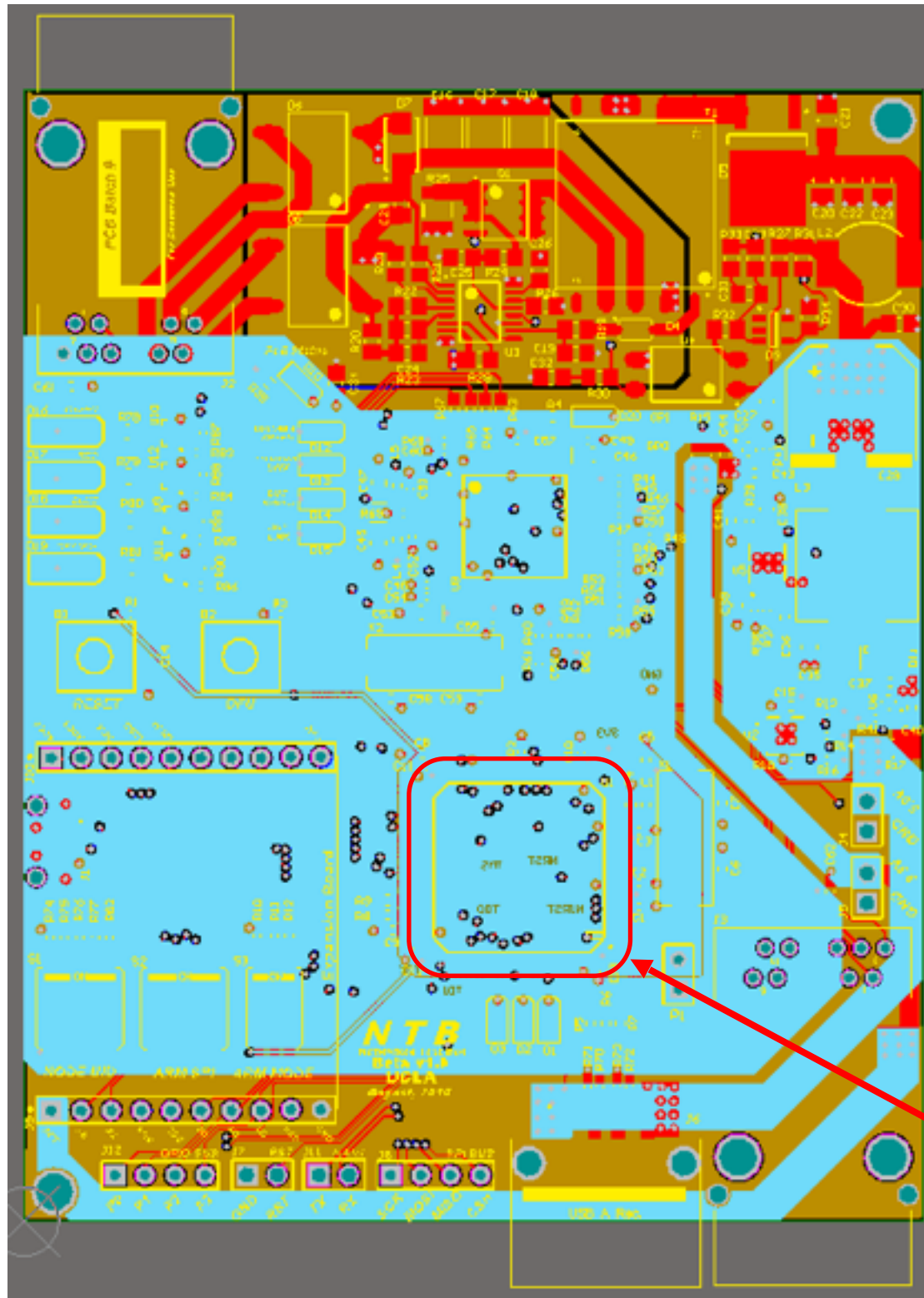
- Looks simple, but as Paul put it
 - “Isn't the 3v3_MB a large polygon on the power layer?”



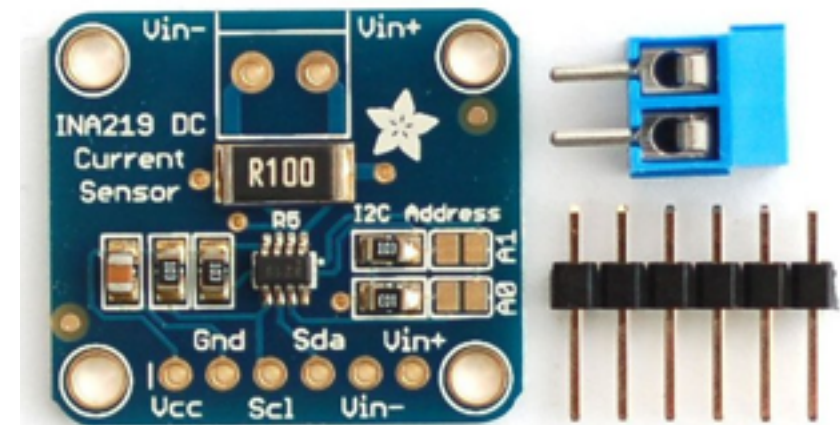
Measure MCU Power

- A smaller polygon is carved out of the original 3.3V_MB domain and sits on a different layer as a separate net.
- A cross-layer jumper, instead of a resistor, is used for options to do power measurement.
- The jumper can be used with INA219B breakout board + BBB for continuous streaming
- The resistor on INA219B can be changed for different range and resolution (0.1 Ω default)

Measure MCU Power



+

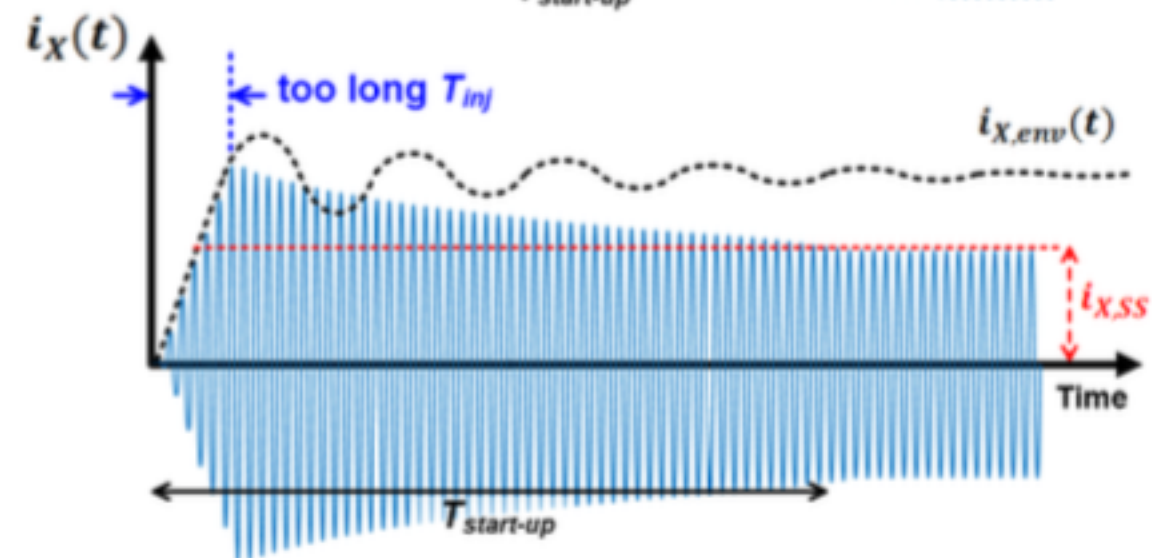
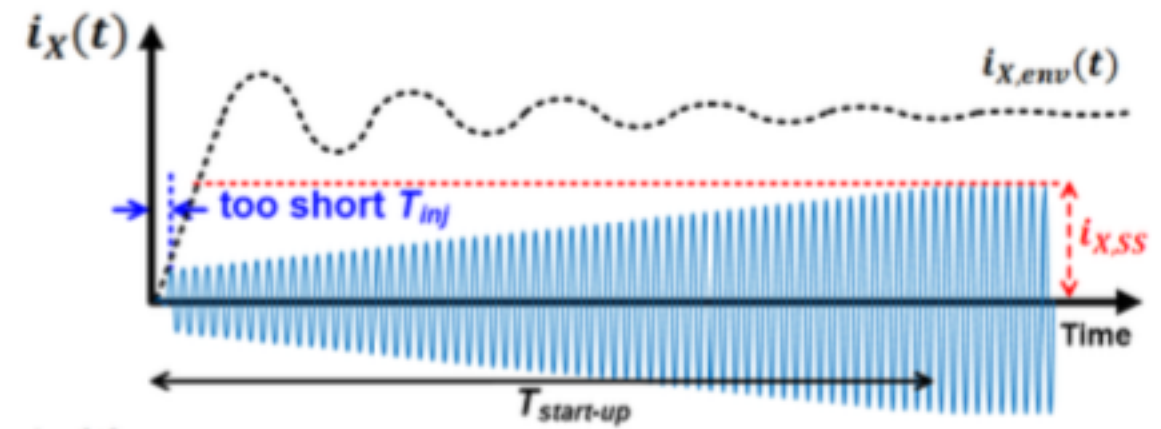
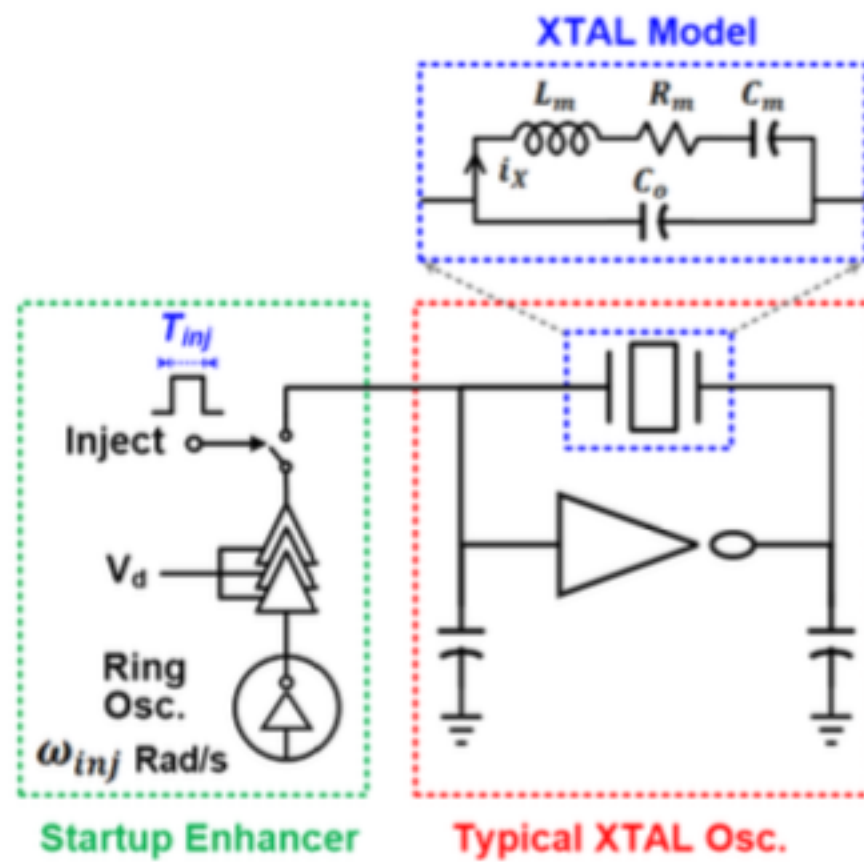


- Modular approach
- Isolated monitoring system
- Board design ready for fabrication

Subnet Layer for MCU supply

Incorporate the oscillator

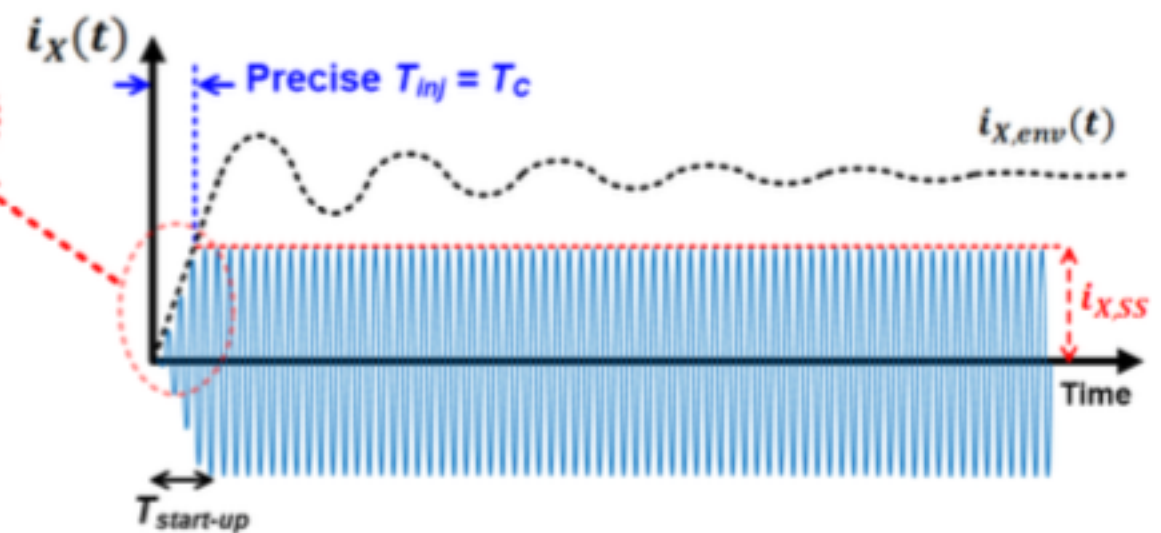
- “The most effective way to speed up the crystal oscillator is to provide it with a surge of initial energy during start-up using a low-Q, integrated “injection” oscillator”
- Controlled injection time to avoid underdamped or overdamped startup behavior as main innovation to achieve fast startup



$$\text{Slope} \cong \frac{V_d}{2L_m} \longrightarrow T_c = \frac{2L_m i_{X,SS}}{V_d}$$

$i_{X,env}(t)$: XTAL current envelope when it is driven by the ring osc

$i_{X,SS}$: XTAL osc steady-state current

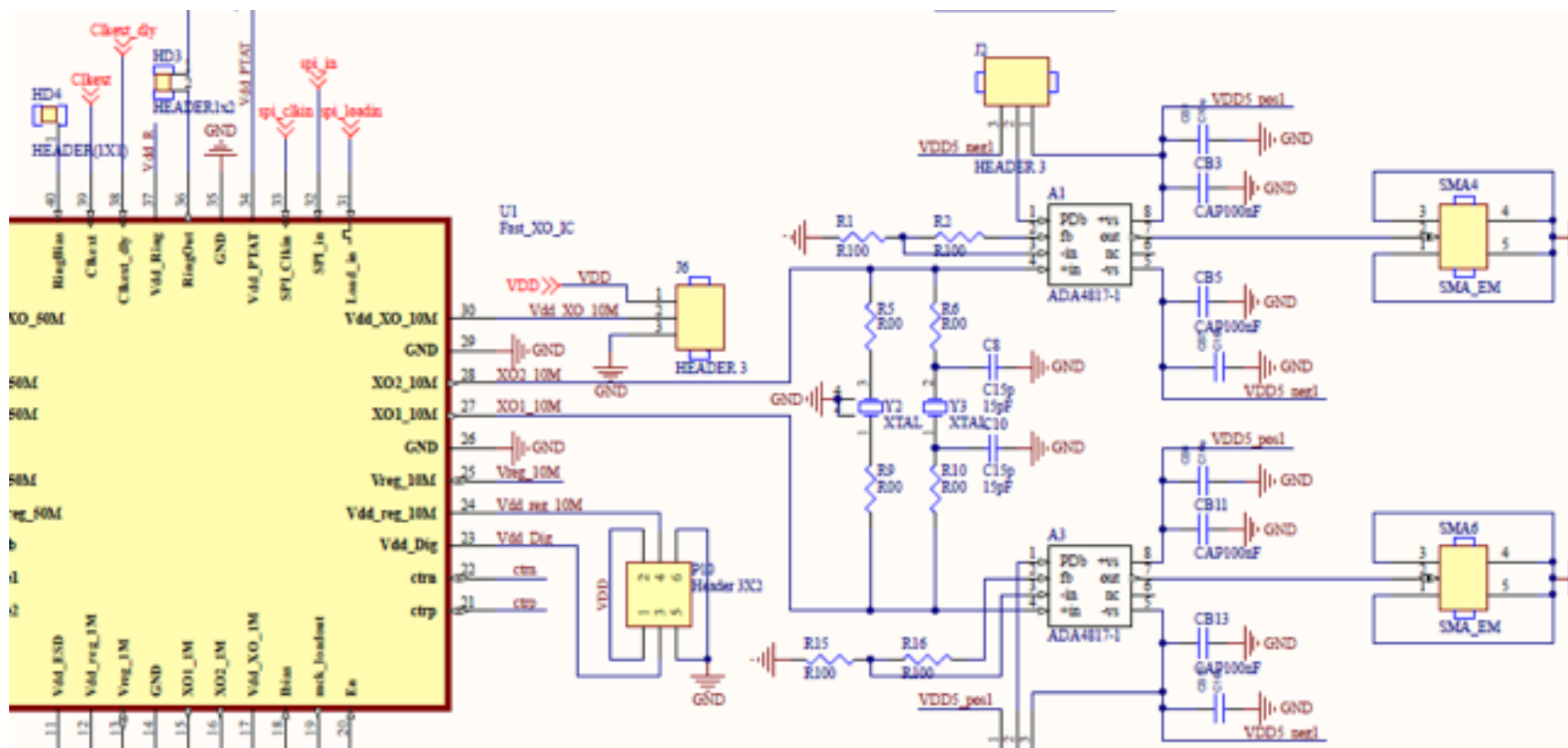


The SMALL Print...

- Chip is not packaged yet...
- Off-chip logic needed to ensure a completely OFF state for the XOs
- SPI interface needed to set up the oscillator (no spare SPI ports – must reuse)
- Multiple power supplies needed (the original design almost uses one supply per circuit)
- The outputs of the oscillators have relatively small amplitude (0.2-0.3Vp-p) - need amplifiers to bring it up to the voltage level needed by the MCU
- Okay...as a student chip designer

The weird one...

- “Clkext and Clkext_dly are two 90degrees **out of phase clocks at the oscillation frequency of the XO**. I used delay ICs to generate the ~90degrees phase shift. You can use any other techniques you know to generate the phase shift. **These two signals need to be available during the start-up**, or maybe before that so MCU might not be able to provide these. You may wanna provide them **externally using SMAs**.”



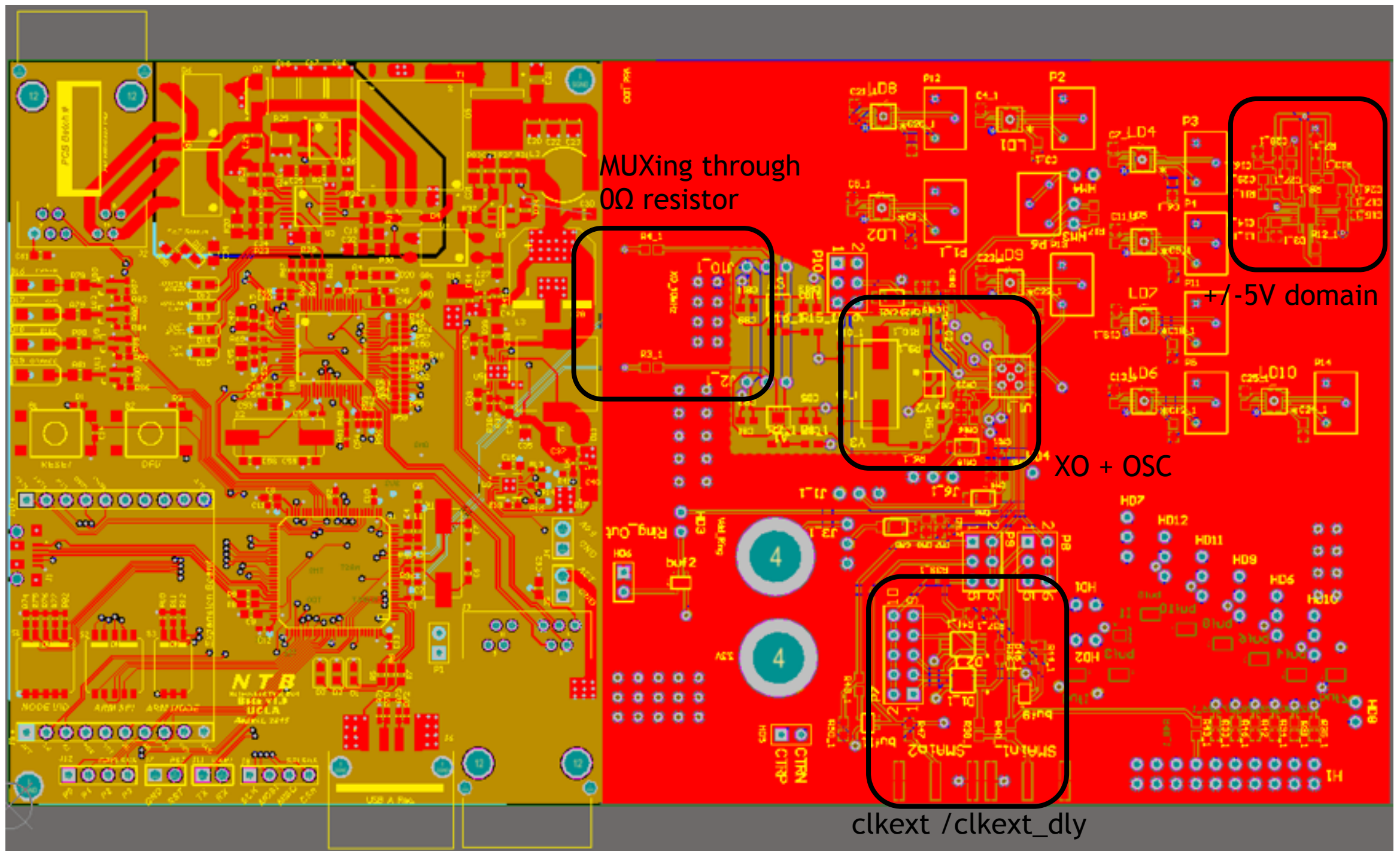
Despite the above

- A tapeout is more expensive than a board design, so make the most out of one...

Incorporate the oscillator

- Original design targets three frequencies, only 10MHz is preserved
- Different power supplies are preserved as the sensitivity to supplies noise and coupling are unknown
- The SPI are left as header pins
- Amplifiers are needed, which mandates +/-5V supply conversion
- The clkext/clkext_dly signals are preserved

Incorporate the oscillator



Incorporate the oscillator

- Design is a complex merge of NTB board and Hani's test board (some components and pinout approaches are recycled but since two boards use different layer/plane strategy the internal wiring is redesigned. New components are also added)
- Board area is more than doubled, due to the bells and whistles needed to deal with the “small prints”
- If there is a future revision addressing the “small prints”, it is possible to keep the original board area but need a redesign
- The board design is on Github, ready for fabrication

Distributed Kalman Filter

Two sub-projects: **libdkf** and **dkf_sim**

libdkf

- Cross platform C++ (arm, x86) DKF library
 - Built as a static library.
 - Exposes and “extern C” API so can be called directly from C code.
- Enhancements to build system
 - Incorporate C++11 toolchain as separate step in build.
 - Added `syscalls.c` to ntb firmware for supporting C++ environment

dkf_sim


- Runs DKF on PC with offline measurement data.
- Provides better debugging and logging facilities.

Implementation Challenges

```
function [P_next,x_next]=dif_ekf_p3(fstate ,P,Q,G,x)
% diffusion Kalman
```

Unknown and dynamic types (real, complex, integer, lambda, etc)

Dynamic matrix dimensions



```
function [P_next,x_next]=dif_ekf_p3(fstate ,P,Q,G,x)
% diffusion Kalman
```

The diagram consists of two black arrows. The first arrow originates from the text 'Dynamic matrix dimensions' and points to the parameter 'P' in the function signature. The second arrow originates from the text 'Unknown and dynamic types (real, complex, integer, lambda, etc)' and points to the parameter 'x' in the function signature.

```

function [z,A]=jaccsd(fun,x)
% JACCSD Jacobian through complex step differentiation
% [z J] = jaccsd(f,x)

...

z=fun(x);
n=numel(x);
m=numel(z);
A=zeros(m,n);
h=n*eps;
for k=1:n
    x1=x;
    x1(k)=x1(k)+h*1i;
    A(:,k)=imag(fun(x1))/h;
end
end

```

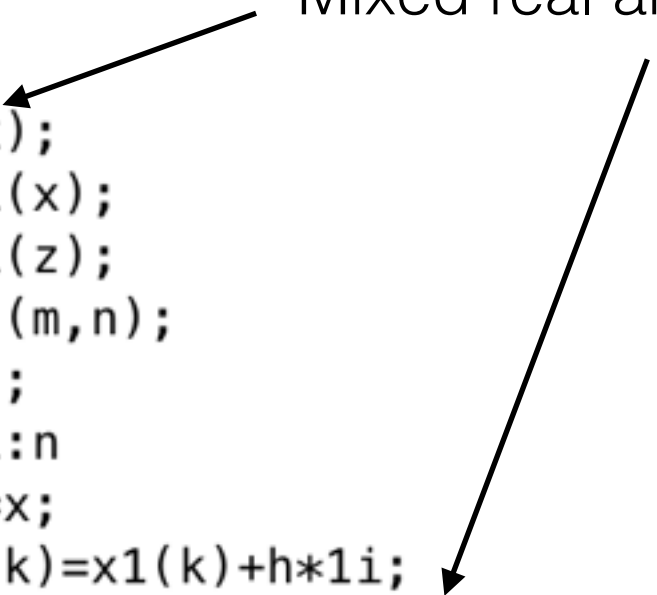


```
function [z,A]=jaccsd(fun,x)
% JACCSD Jacobian through complex step differentiation
% [z J] = jaccsd(f,x)
```

```
...
```

Mixed real and complex

```
z=fun(x);
n=numel(x);
m=numel(z);
A=zeros(m,n);
h=n*eps;
for k=1:n
    x1=x;
    x1(k)=x1(k)+h*1i;
    A(:,k)=imag(fun(x1))/h;
end
end
```



```
function efk_part2(obj,fstate,Q)
    obj.x=0;
    for i =1:length(obj.c)
        obj.x = obj.x+ obj.eital{i}*obj.c(i) ;
    end
    ekf_part3(obj,Q,fstate);
end
```

Scalar on first iteration

```
function ekf_part2(obj,fstate,Q)
    obj.x=0;
    for i =1:length(obj.c)
        obj.x = obj.x+ obj.eital{i}*obj.c(i) ;
    end
    ekf_part3(obj,Q,fstate);
end
```

Becomes vector subsequently

535 Lines of MatLab code to **1062** Lines of C++.

Power Management Firmware

MCU Sleep

- Sleep mode (WFI).
- Deep sleep mode (power off oscillator).
 - Wake-up using RTC.
- Challenge is making sure all peripherals are in the proper state.

Challenges with the DW1000 Radio Sleep

- Learning the HAL and the underlying hardware registers.
- Developing a good abstraction for the radio is an entire project on its own.

Modified the LwIP TCP server on the ntb boards to allow peer-to-peer communication.

'Cnl' lists the IP addresses of all neighbors.

'Cna<ip_addr>' adds a neighbor IP address.

'Cnc' deletes all neighbors.

'Cnb<message/command>' sends command or data to configured neighbors.

Lessons Learned

Three Projects

1. DKF Algorithm Implementation
2. Hardware Design (Oscillator and Power measurement)
3. Firmware for power management

Three Projects

1. DKF Algorithm Implementation

Underestimated the scope.

Consider alternative: Matlab Coder.

2. Hardware Design (Oscillator and Power measurement)
3. Firmware for power management

Three Projects

1. DKF Algorithm Implementation

Underestimated the scope.

Consider alternative: Matlab Coder.

2. Hardware Design (Oscillator and Power measurement)

Prototype != off-the-shelf part

3. Firmware for power management

Three Projects

1. DKF Algorithm Implementation

Underestimated the scope.

Consider alternative: Matlab Coder.

2. Hardware Design (Oscillator and Power measurement)

Prototype != off-the-shelf part

3. Firmware for power management

Start with commercial development board (mbed, beaglebone, etc).

Focus entirely on making the radio easier to use.

References

1. Hani Esmaeelzadeh and Amr Alanwar. "PLoTS: Power Efficient Localization and Time Synchronization"
2. Zhen, ChengFang, et al. "Energy-efficient sleep/wake scheduling for acoustic localization wireless sensor network node." International Journal of Distributed Sensor Networks 2014 (2014).
3. Wu, Yan, Sonia Fahmy, and Ness B. Shroff. "Optimal QoS- aware sleep/wake scheduling for time-synchronized sensor networks." 2006 40th Annual Conference on Information Sciences and Systems. IEEE, 2006.
4. Federico S. Cattivelli and Ali H. Sayed. "Distributed Nonlinear Kalman Filtering With Applications to Wireless Localization." Department of Electrical Engineering University of California, Los Angeles.

Questions?