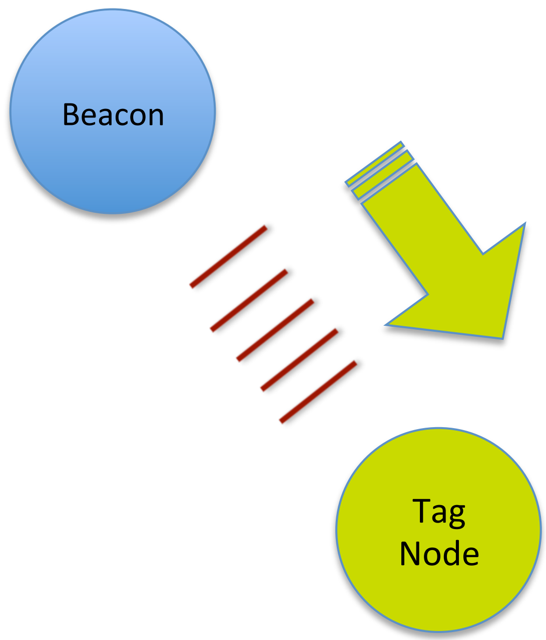
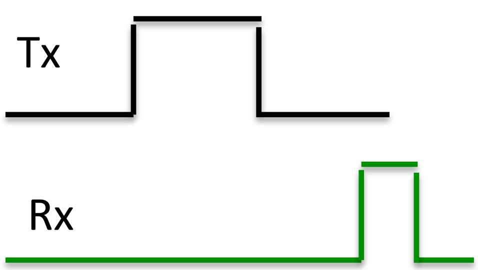
CSSE4011 Lab 4, Part 1: Ultrasound Localization

Ultrasound Localisation

- Use ultrasonic rangers to determine distance from nodes.
- If you know location and orientation of the beacon, you can estimate location of the tag
- Ultrasonic rangers:
  - Generate ultrasonic pulse (5 us)
  - PWM output



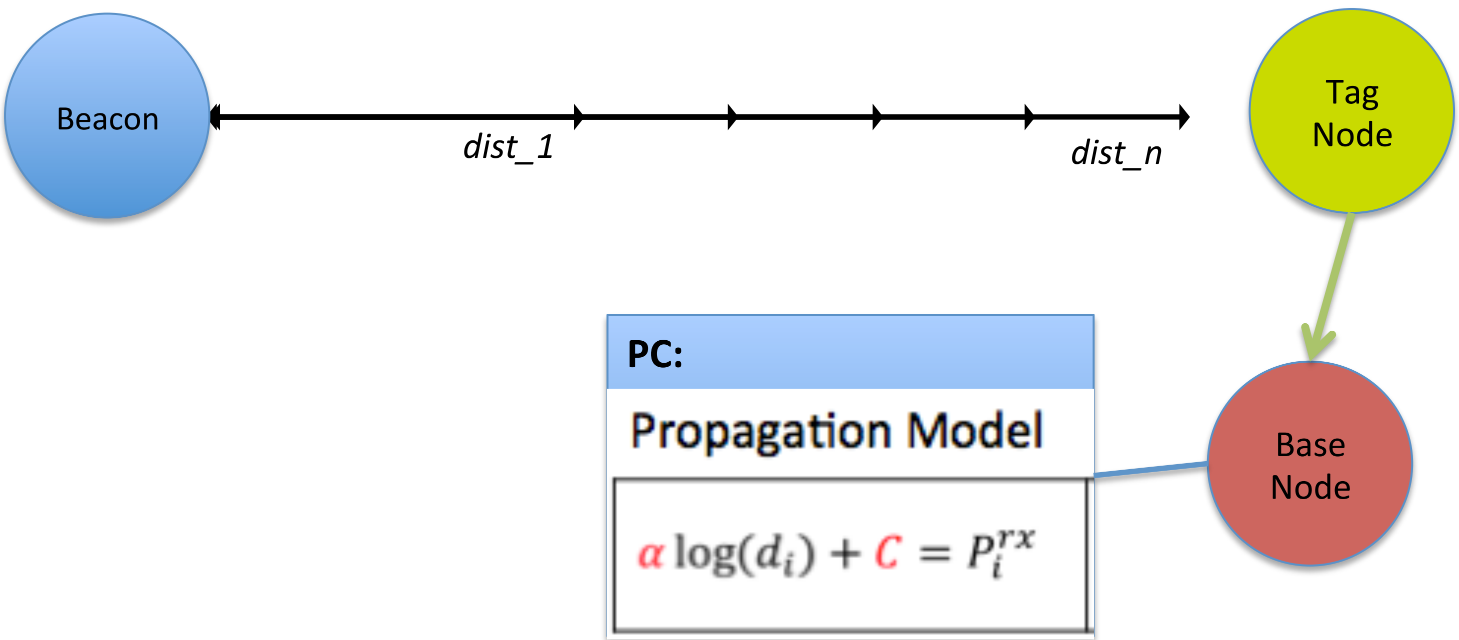


Ultrasonic Ranger Waveform

CSSE4011 Lab 4, Part 2:
RF Localization

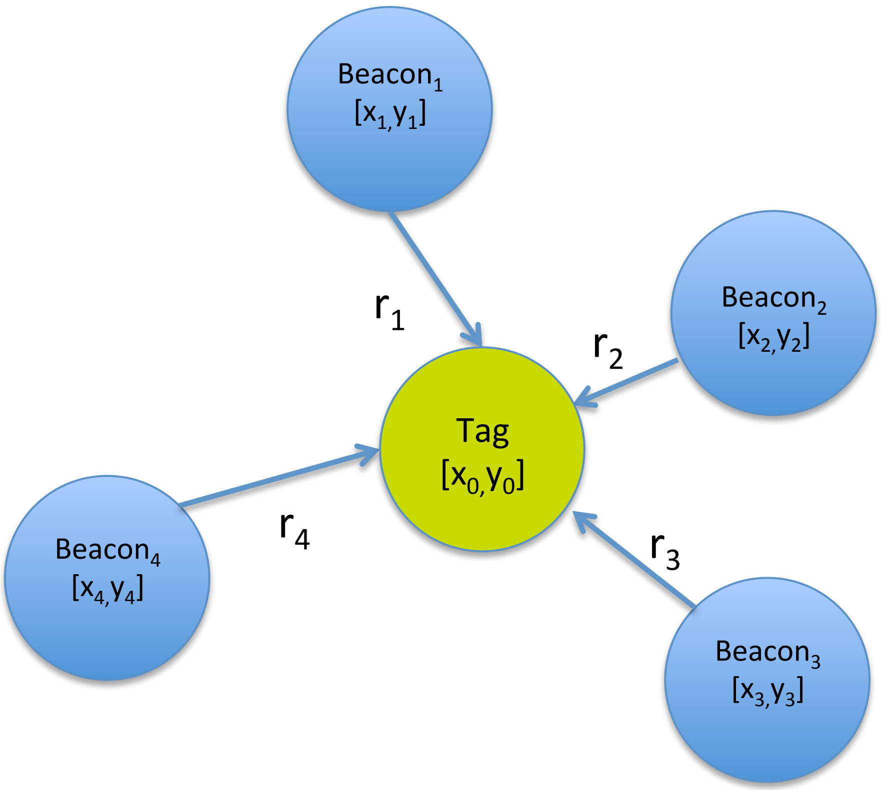
RSSI Ranging

1. Use calibrated propagation model from the previous lab
2. Model was built using at least 10 distances
3. Propagation model gives you relationship between distances and RSSI value



Rssi Localization

- Tag estimates its distance from Beacons by measuring RSSI of the radio links
- Location of the Tag is at an intersection of circles centered at beacons



PC: Localization

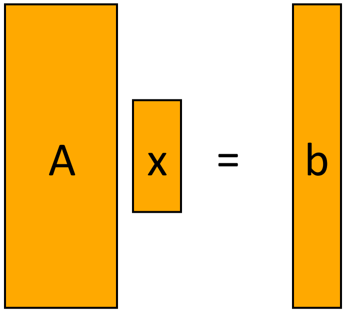
- Localize a tag inside an area of interest using multilateration:
  - Place beacons at known coordinates $[x_1,y_1]\ldots[x_4,y_4]$
  - Measure ranges between the tag and all beacons $r_1,\ldots,r_4$ (convert RSSI in [dBm] to distance in [m])
  - Create matrix **A** and vector **b** of the linearized multilateration problem (see lecture) and use least squares equation to estimate tag location
  - Visualize the location in GUI

Help with Linear Least Squares

Propagation Model	Multilateration
$\alpha \log(d_i) + C = P_i^{rx}$	$2x_0(x_k - x_i) + 2y_0(y_k - y_i) = r_i^2 - r_k^2 - x_i^2 - y_i^2 + x_k^2 + y_k^2$

Linear least squares problem formulation:

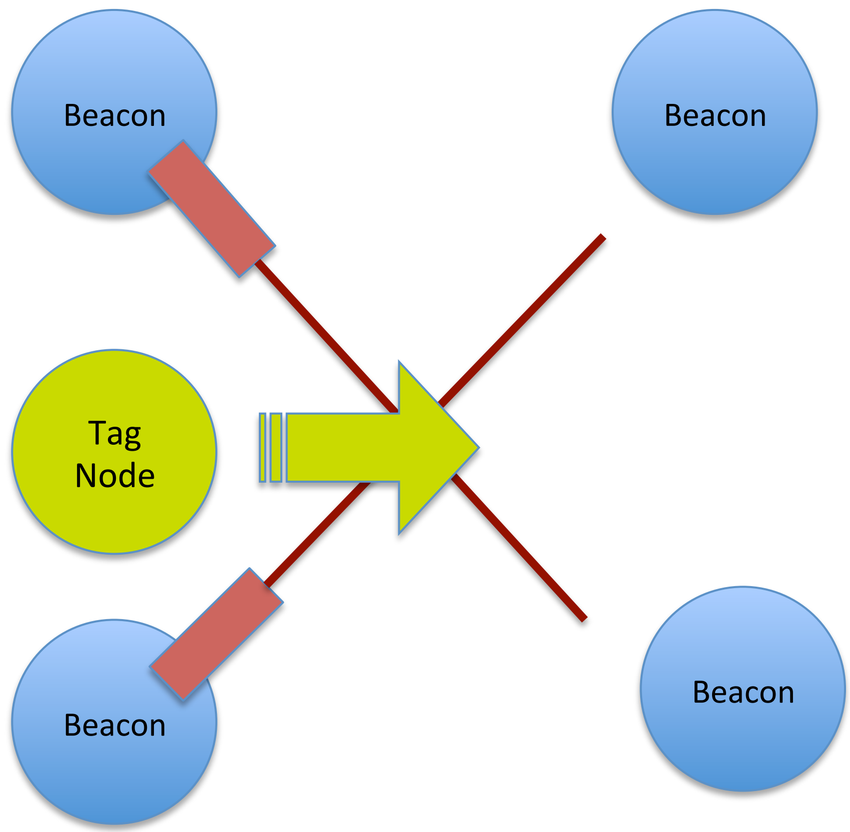
$$Ax = b$$



CSSE4011 Lab 4, Part 3: Tracking of Mobile Nodes

Track Mobile Tag Node

- Move the Tag Node during the experiment and visualize its changing location on a PC
- Simplifications:
  - Assume constant velocity and known initial location
- Build Kalman Filter to fuse information:
  - RF location
  - Ultrasound location
  - Tag motion model



Examples We Provide

kalman.py Example **kalman.py**	**Assignment** **tracking.py**
Estimate 1-dimensional constant function	Estimate location of a tag node
STATE **x**=[const] - Value of the constant function	**x** = $[x, y, v_x, v_y]$ - location and velocity
Transition Matrix **A** [1]	[1 0 dt 0] [0 1 0 dt] [0 0 1 0] [0 0 0 1]
Observation Matrix **H** [1]	[1 0 0 0] [0 1 0 0]
Process/Observation Noise Cov Matrix **Q** = eye(1)*proc_noise **R** = eye(1)*observ_noise	 **Q** = eye(4)*proc_noise **R** = eye(2)*observ_noise

Kalman Filter Notes

- Use tracking.py as a starting point
- Your tracking system will
  - Fuse across sensors
    - Location estimate L_1 from ultrasound localization (part 1 of the prac)
    - Location estimate L_2 from RSSI based multilateration (part 2 of the prac)
  - Fuse across time
    - Assume a simple motion model (e.g., linear motion)
- You'll need to
  - define a new observation matrix H that incorporates measurements from both localization subsystems
  - define an observation error matrix R that takes into consideration different errors of the two subsystems (ultrasound is more accurate)
  - solve the case when ultrasound or RSSI location is not available

Prediction (Time Update)

(1) Project the state ahead

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t \mathbf{u}_t$$

(2) Project the error covariance ahead

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^{\mathrm{T}} + \mathbf{Q}_t$$

Correction (Measurement Update)

(1) Compute the Kalman Gain

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t^{\mathrm{T}} (\mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^{\mathrm{T}} + \mathbf{R}_t)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1})$$

(3) Update Error Covariance

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t \mathbf{H}_t \mathbf{P}_{t|t-1}$$

Source: R. Faragher, "Understanding the basis of the Kalman filter via a simple and intuitive derivation", IEEE Signal Process. Mag., vol. 29, no. 5, pp. 128-132, Sep. 2012.