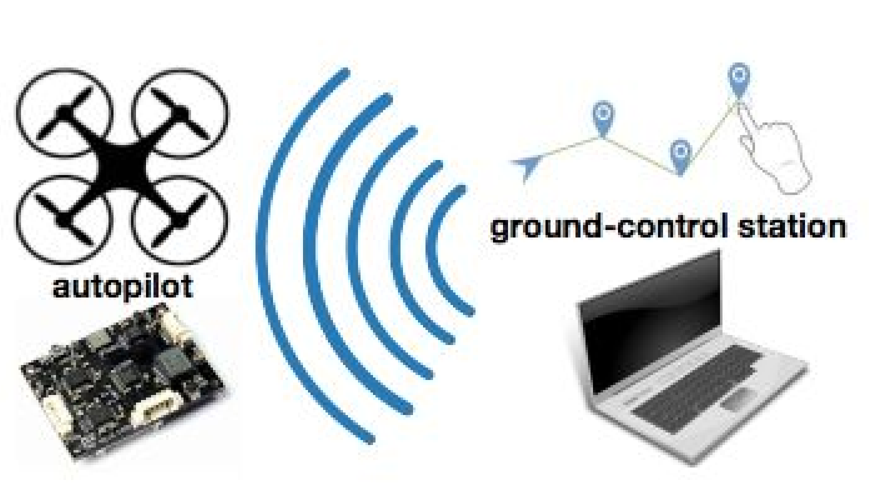
Reactive Control of Autonomous Drones

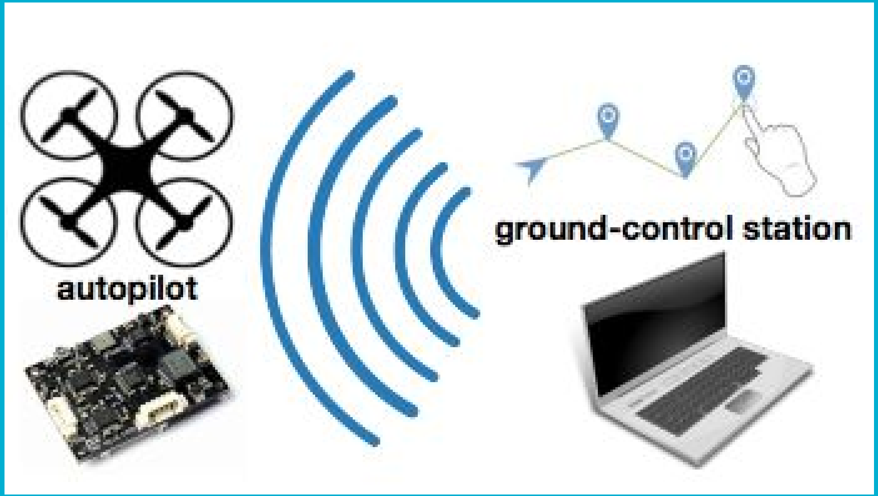
Hyo Jin Kim


Drones!
Explore near-inaccessible areas
High-resolution imagery
Low cost, Flexible

Background - Existing Platforms



Background - Existing Platforms



GCS

High-Level Control

- Waypoints to cover
- Actions to take at each waypoint

Question 1

What does autopilot do?

How it’s control is different from that of GCS?

Background - Existing Platforms

Autopilot

Low-Level Control

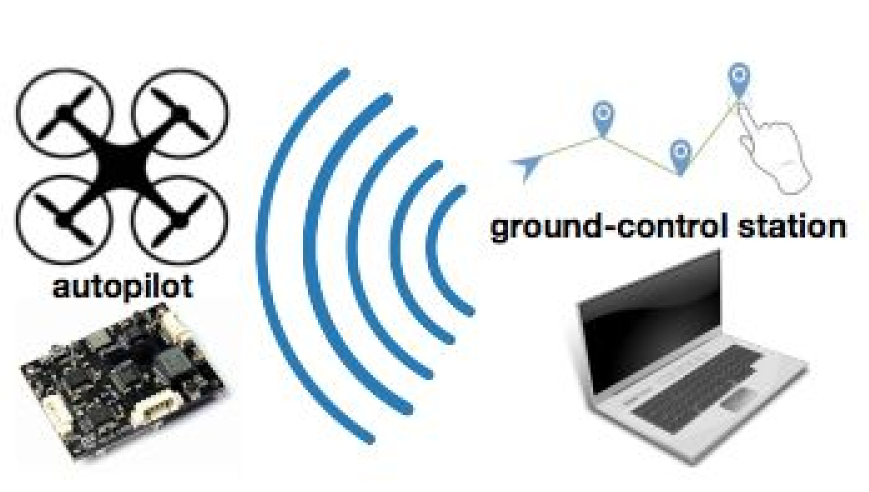
Sensor inputs

Accelerations, GPS

Operate actuators

Electrical motors

→ Set 3D orientation



GCS

High-Level Control

- Waypoints to cover
- Actions to take at each waypoint

Importance of Low-Level Control

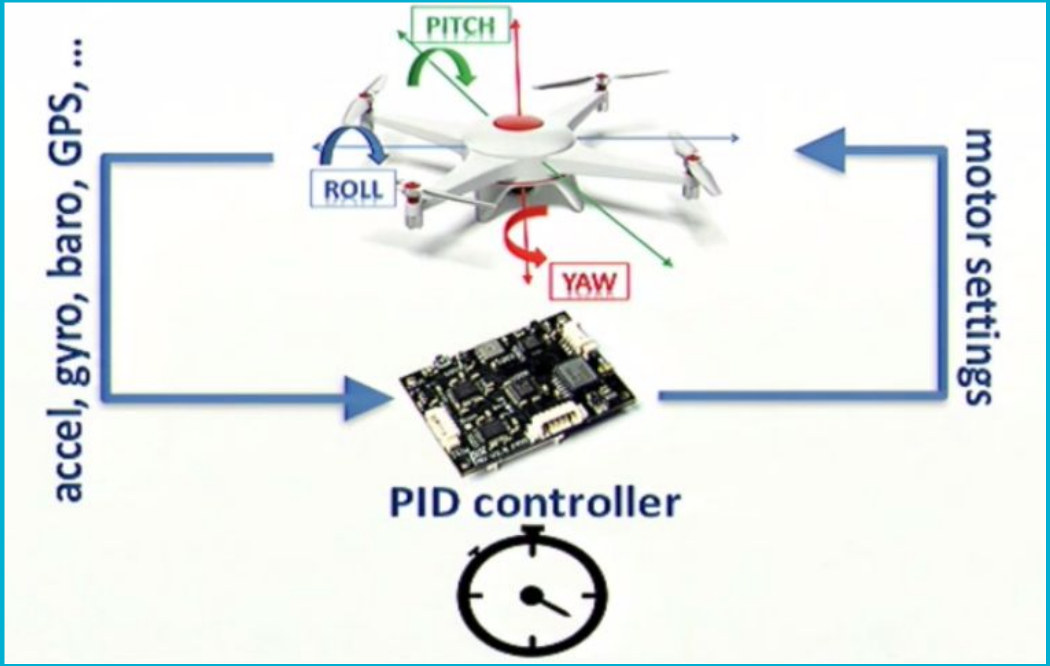
Determines the effectiveness of physical motion

- Quality of photos/videos

Affects how the energy is consumed

- Drone’s lifetime is often a result of how efficient is its operation

Autopilot in Time-Triggered Fashion



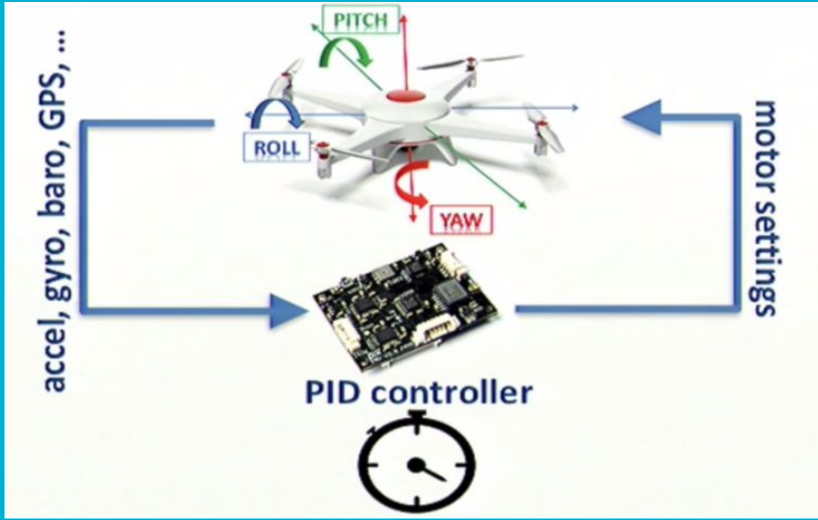
[Picture Credit: L. Mottola]

Question 2

How does time-triggered controller work?

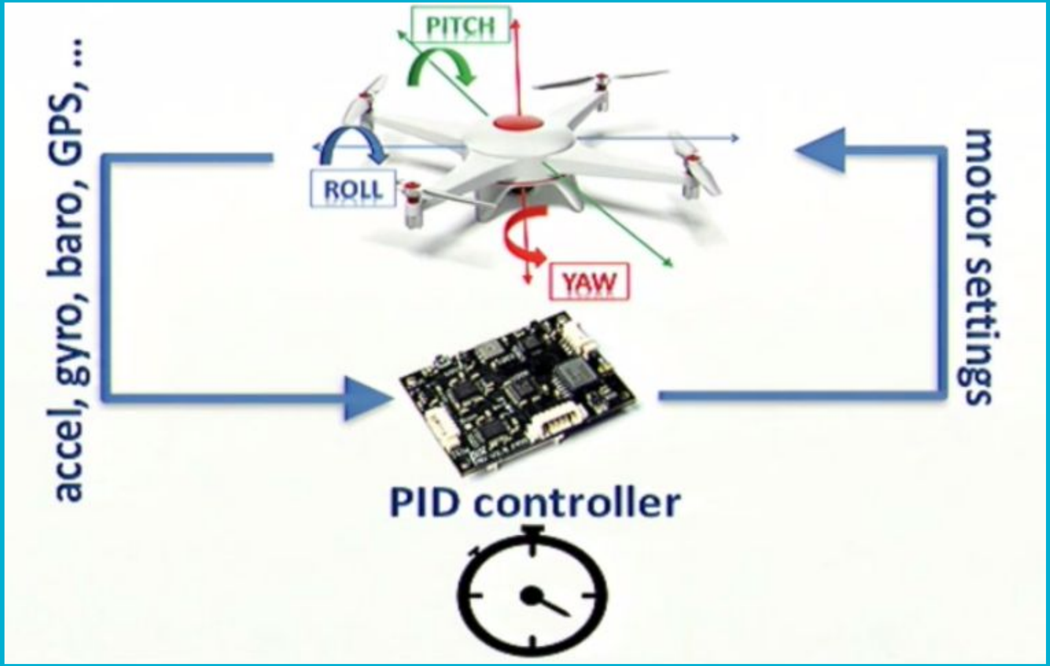
Autopilot in Time-Triggered Fashion

Every T time units, probe sensors, compute control decisions, and deliver commands to the actuators.



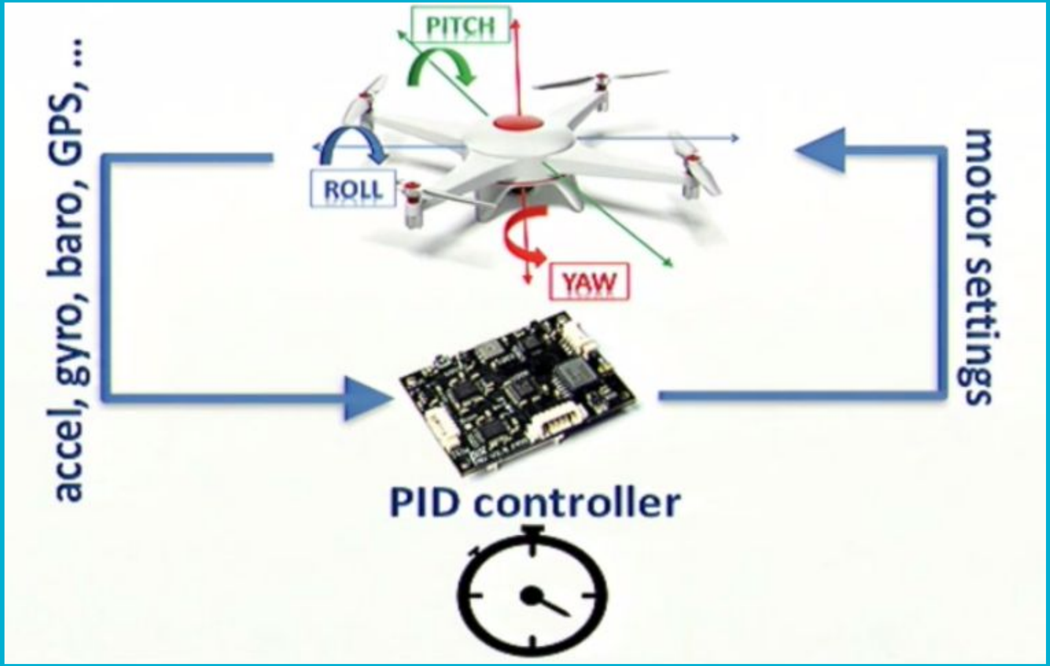
[Picture Credit: L. Mottola]

Autopilot in Change-Triggered Fashion



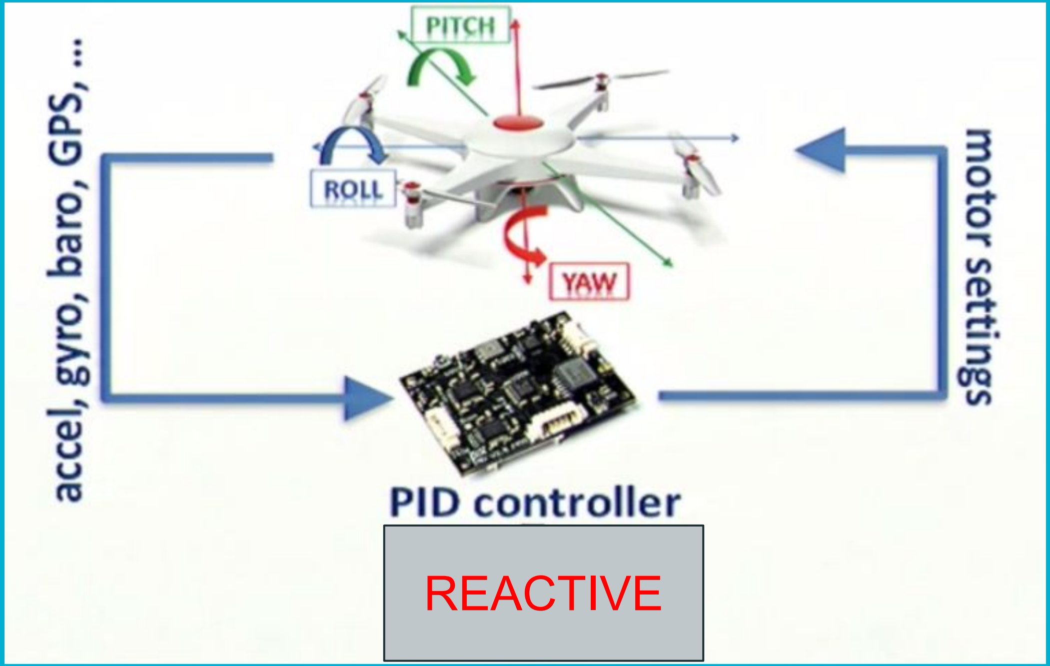
[Picture Credit: L. Mottola]

Autopilot in Change-Triggered Fashion



[Picture Credit: L. Mottola]

Autopilot in Change-Triggered Fashion



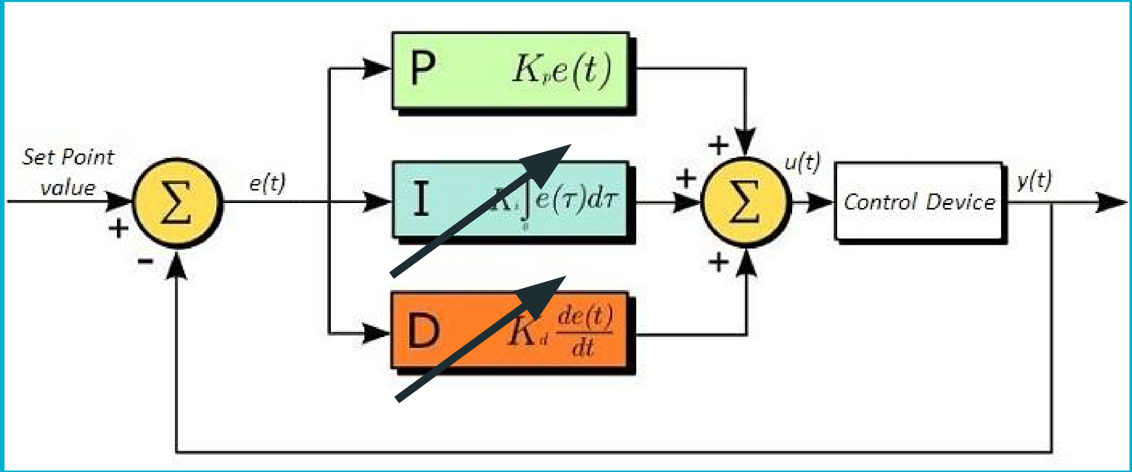
[Picture Credit: L. Mottola]

Motivation

Proportional component dominates

Similar sensor inputs results in similar output

→ Maintain current setting for similar sensor input



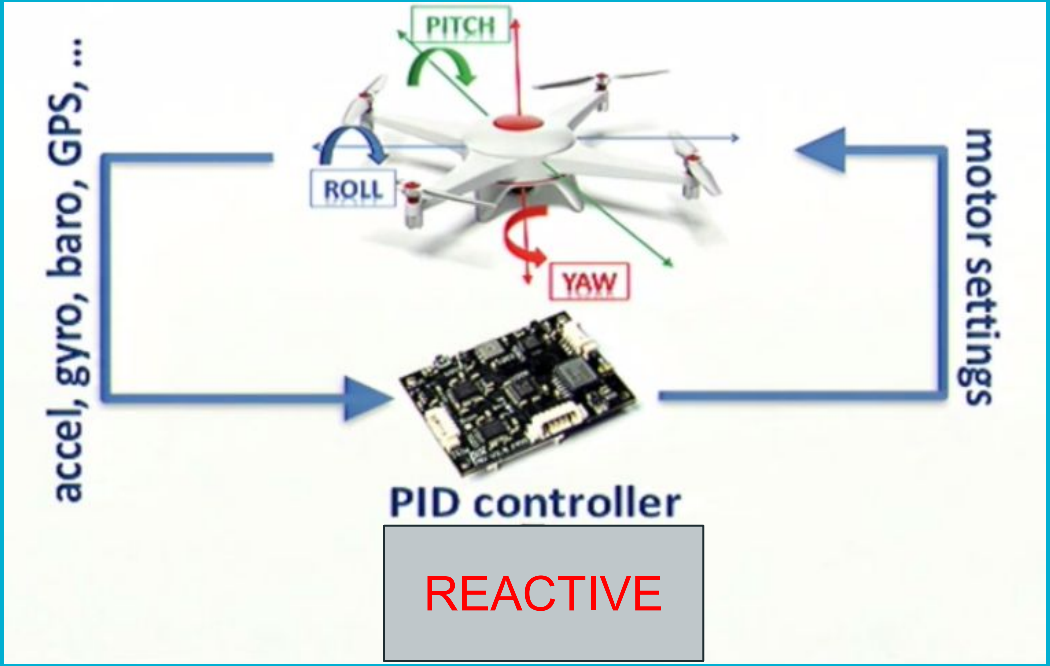
Motivation

Autopilot runs on hardware closely resembles mobile phones

Energy-efficient high frequency sensors

Have interrupt-driven modes: generate a value upon verifying certain conditions

Autopilot in Change-Triggered Fashion



[Picture Credit: L. Mottola]

Question 3

What are the benefits of reactive control?

Benefits

1. **Lessen the need to overprovision control rates**
   Run the control logic only upon recognizing the need to
2. **Improve hardware utilization**
   Spares unnecessary processing
3. **Attain more timely control decisions**
   If sensor inputs change often, control runs repeatedly

Statement

The efficiency of autopilot can be increased by executing control decisions only upon recognizing the need to, based on observed changes in the navigation sensors, that allows rate of execution dynamically adapt to the circumstances.

Subproblems (Challenges)

1. **What is a “significant” change in the sensor input?**
    - Difficult to generalize
      Depends on accuracy of sensor hardware, the physical characteristics of the drone, the control logic, and the granularity of actuator output.
2. **Handling Interleaving of Triggers**
    - Triggered by different sensors, at different rates, asynchronously
3. **Implementation issue**
    - Code quickly turns into a “callback hell” as the operation becomes inherently event-driven.

Related Work

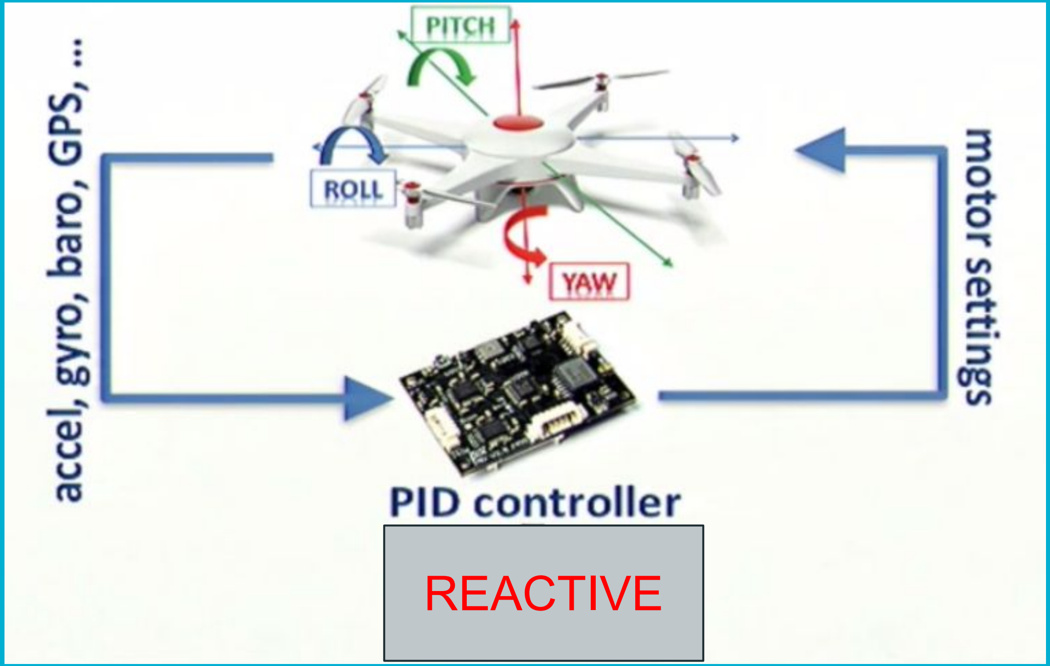
Event-based control (Astrom, 2007)

- Detect events → Generate control signal
- Control is not executed unless it is required

Difference:

- Different application
- Control logic is expressly redesigned
- Reactive control re-uses existing control logic

Autopilot in Change-Triggered Fashion

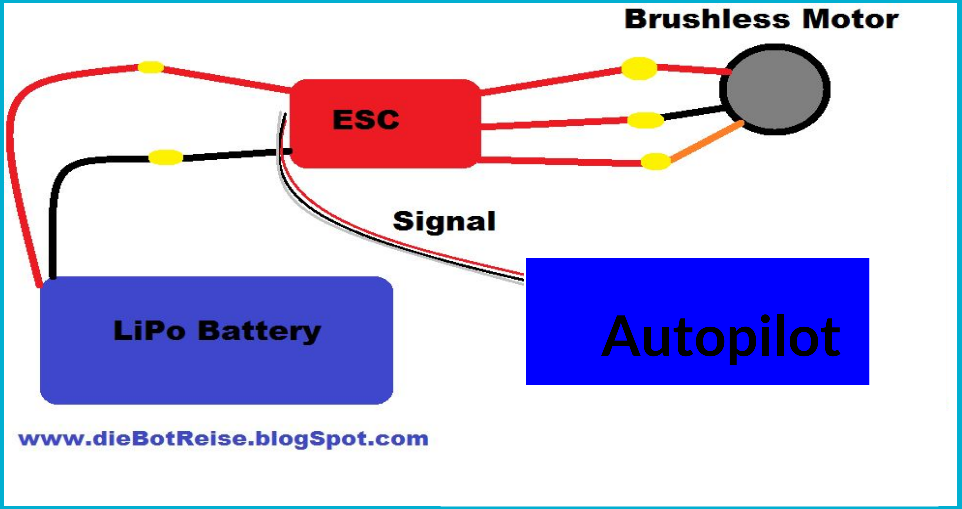


[Picture Credit: L. Mottola]

Experimental Demonstration of Problem

Verifying that some iterations of the control loop are unnecessary

Measure:
Output current of Electronic Speed Controllers (ESC)

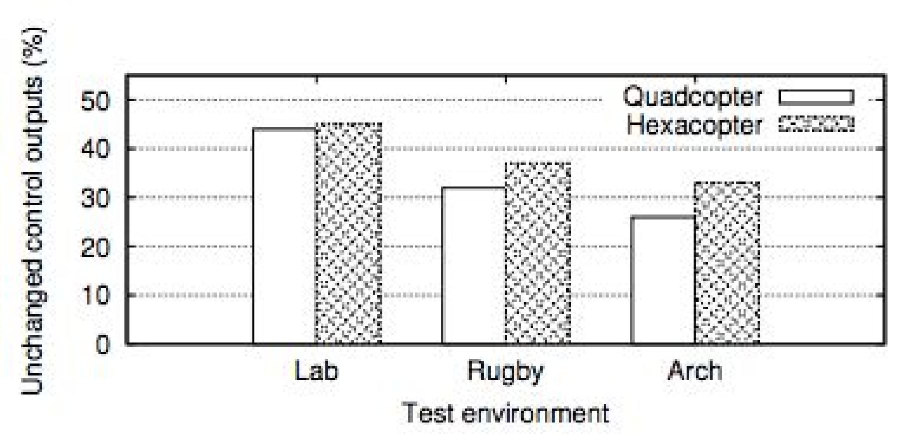


Experimental Demonstration of Problem

Verifying that some iterations of the control loop are unnecessary

Measure:
Output current of Electronic Speed Controllers (ESC)

AC Mild wind Strong wind



The less influence, the more the control decisions remain the same

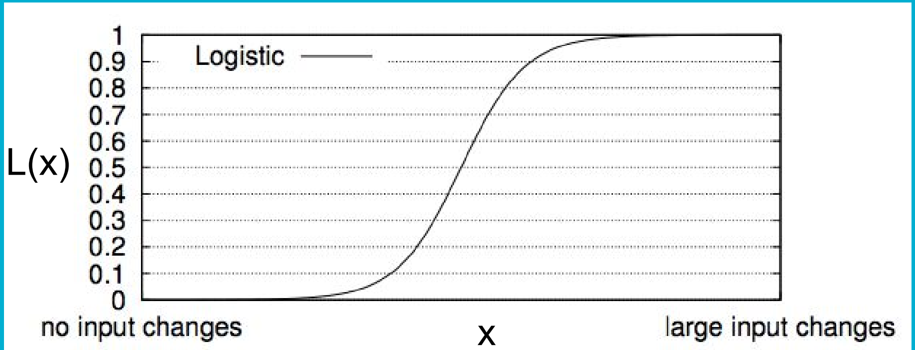
P1. Recognizing Change Alters Control Logic

For each sensors, perform logistic regression

x : difference between consecutive samples

y: whether control decisions changed {0,1}

$$L(x) = \frac{1}{1 + e^{-(\beta_1 + \beta_2 x)}}$$



P1. Recognizing Change Alters Control Logic

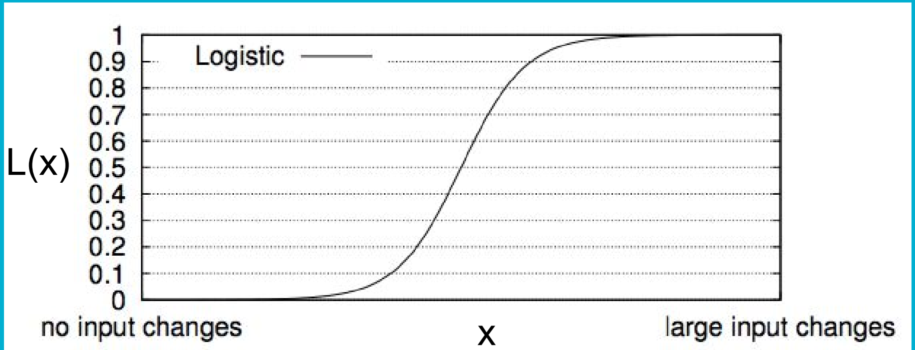
For each sensors, perform logistic regression

x : difference between consecutive samples

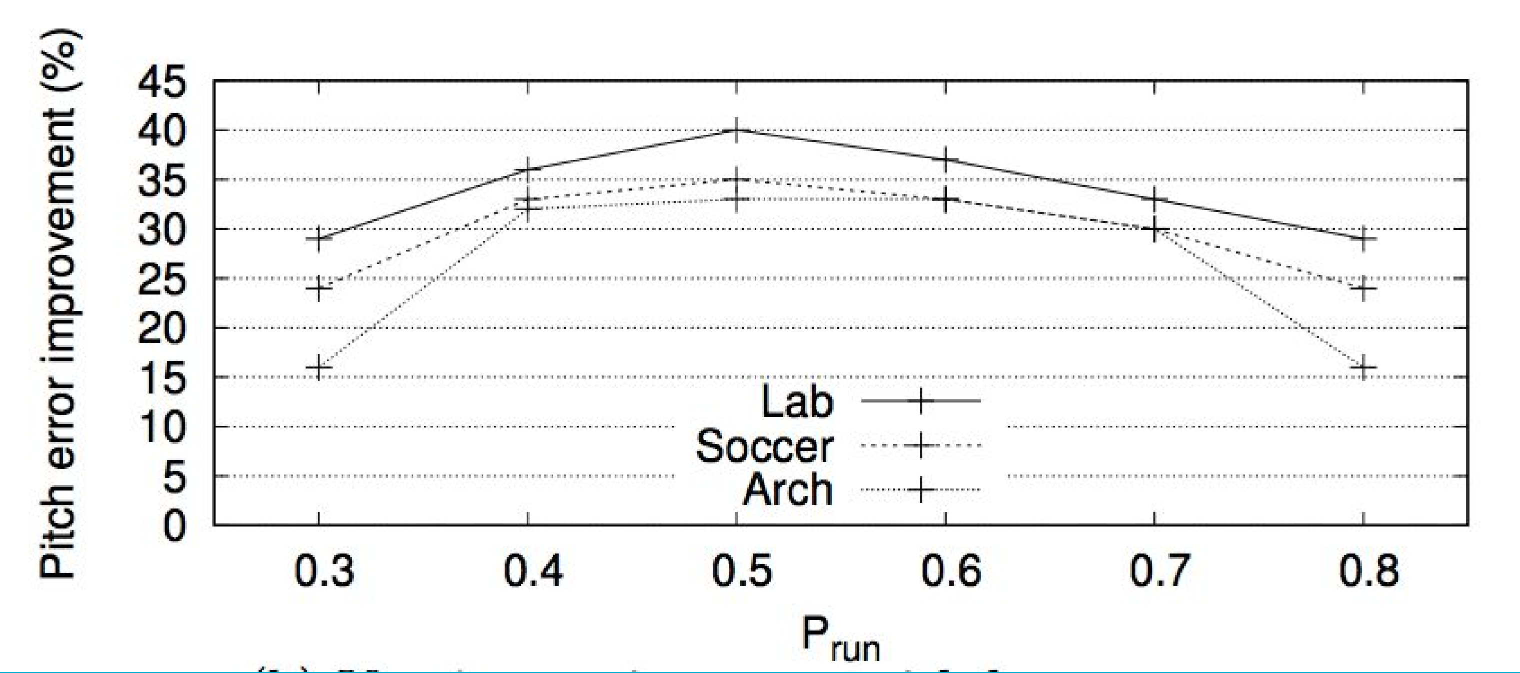
y: whether control decisions changed {0,1}

Whenever $L(x) > P_{run}$,
execute control logic
Trigger

$$L(x) = \frac{1}{1 + e^{-(\beta_1 + \beta_2 x)}}$$



Effect of P_{run}



P1. Recognizing Change Alters Control Logic

- Runtime Operation
  - For time T_{boot}, run in a time-triggered fashion for collecting data
  - Estimate parameters for L(x)
  - Perform reactive control

False positive/negative occurs

→ Add to data

→ Re-estimate L(x)

x

P2. Handling Interleaving of Triggers

- How to handle multiple sensors triggers close in time
- Must consider the unlucky case of missing a large number of consecutive triggers

Question 4.

- How to handle multiple sensors triggers close in time
- Must consider the unlucky case of missing a large number of consecutive triggers

How did authors solve this problem?

P2. Handling Interleaving of Triggers

- How to handle multiple sensors triggers close in time
- Must consider the unlucky case of missing many consecutive triggers

→ **Sample every sensor at the highest frequency**

Major energy drain aboard the drones is anyways due to the motors

→ **Failsafe**

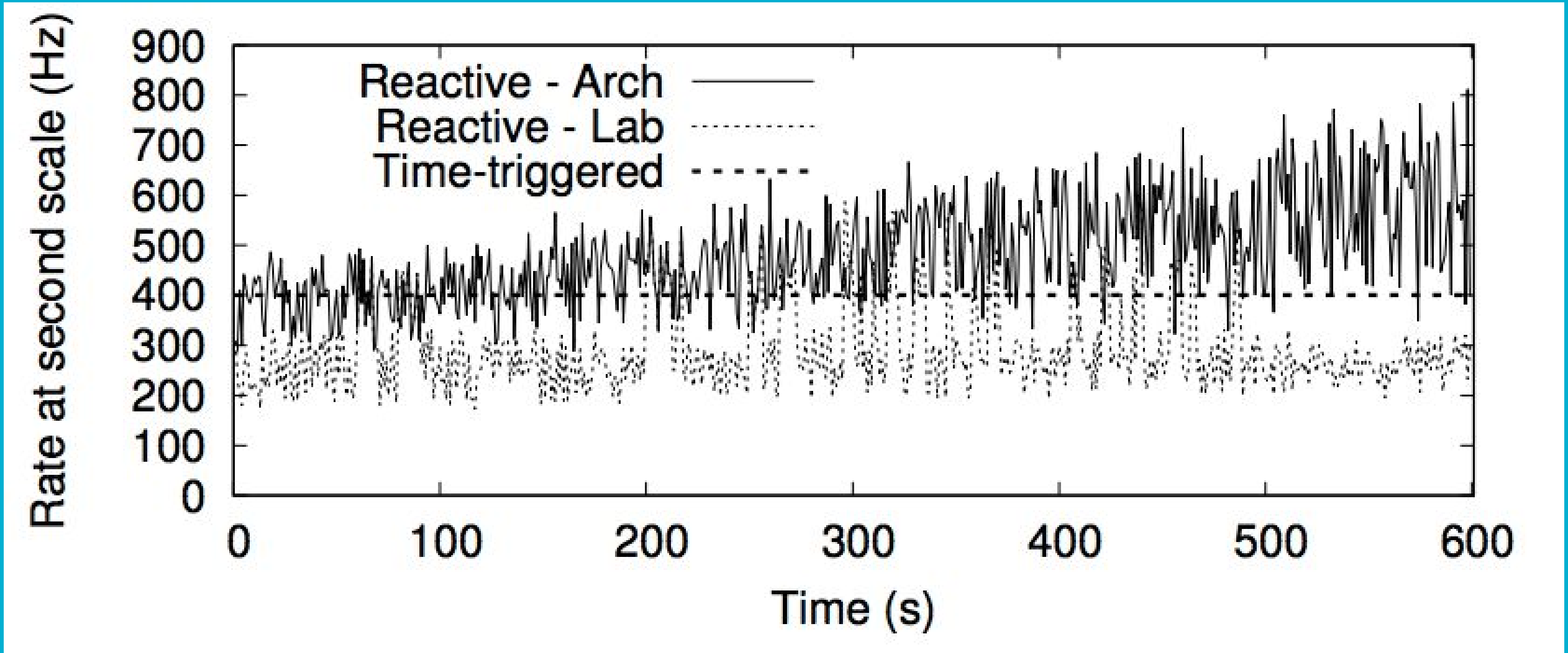
Execute control logic every $T_{failsafe}$

→ **Hyperperiod**

Wait before sampling of all sensors repeats

"Accumulates" all triggers possibly recognized on different sensors

In Action


Rate at second scale (Hz)
900
800
700
600
500
400
300
200
100
0
Reactive - Arch
Reactive - Lab
Time-triggered
0
100
200
300
400
500
600
Time (s)

P3. Implementaion - Callback Hell Problem

Two types of output:

- Immediately useful
- Updating global status

  → Every processing step need to execute upon recognizing change in inputs

  → Callback hell

```
var callback4 = function(){
    console.log("enough already")
}
var callback3 = function(){
    callback4();
}
var callback2 = function(){
    callback3();
}
var callback1 = function(){
    callback2();
}
async function(){
    callback1()
}
```

P3. Implementaion - Callback Hell Problem

Solution

Reactive Programming (Bainomugisha et al., 2013)

Question 5.

What is reactive programming?

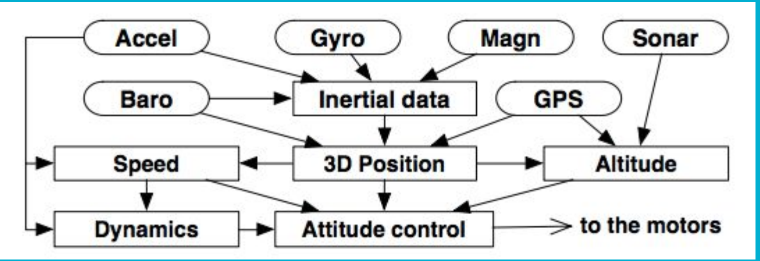
P3. Implementaion - Callback Hell Problem

Solution

Reactive Programming (Bainomugisha et al., 2013)

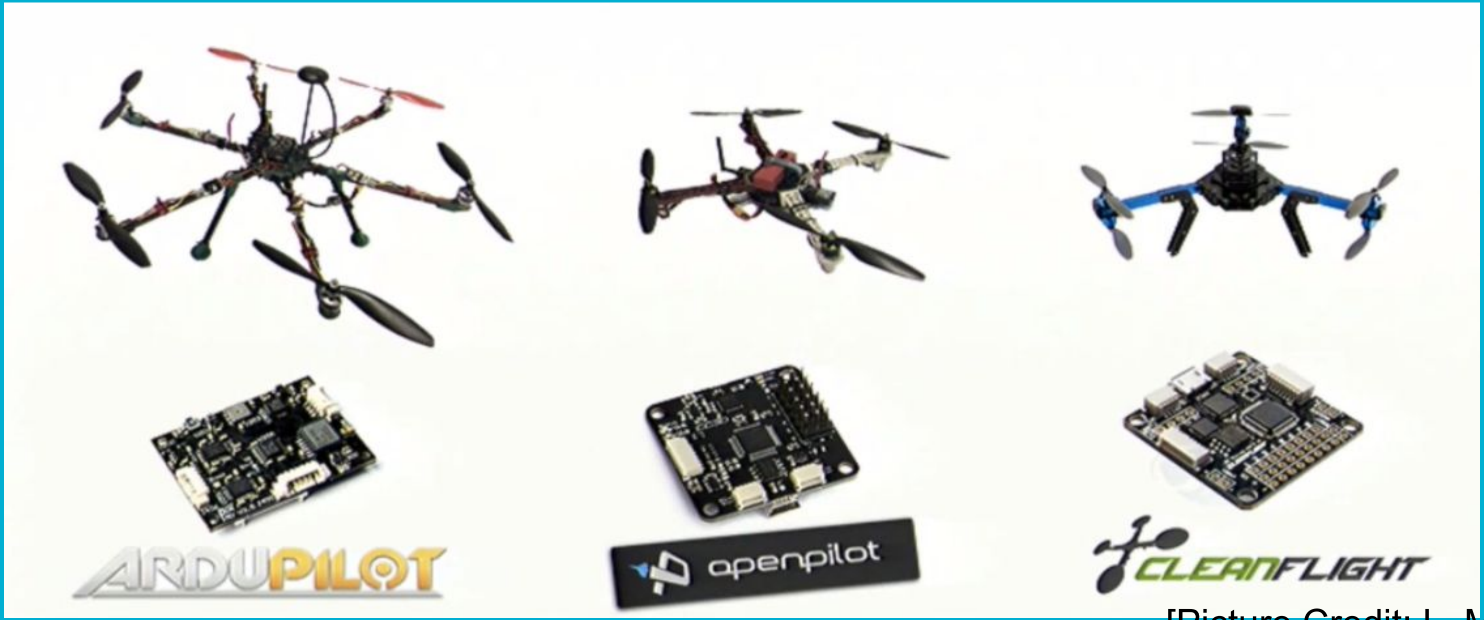
```
a= 2;
b= 3;
c= a + b;
```

- Declare data dependencies between variables
- Dependencies form acyclic graph
- Traverses the data dependency graph every time a data change occurs



Experimental Setup (1)

3x Drones, 3x Autopilots



[Picture Credit: L. Mottola]

Experimental Setup (Cont')

18 different flight paths

Each with 8 random waypoints

Repeat at least 3 times, until battery reaches 20%

3 Environments

Lab, Rugby, Arch

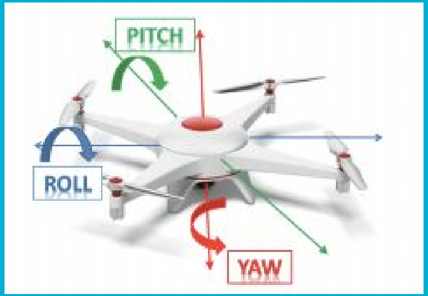
Parameters

$P_{run} = 0.6$, $T_{failsafe} = .1$ sec, $T_{boot} = 30$ sec

Experimental Setup (Cont')

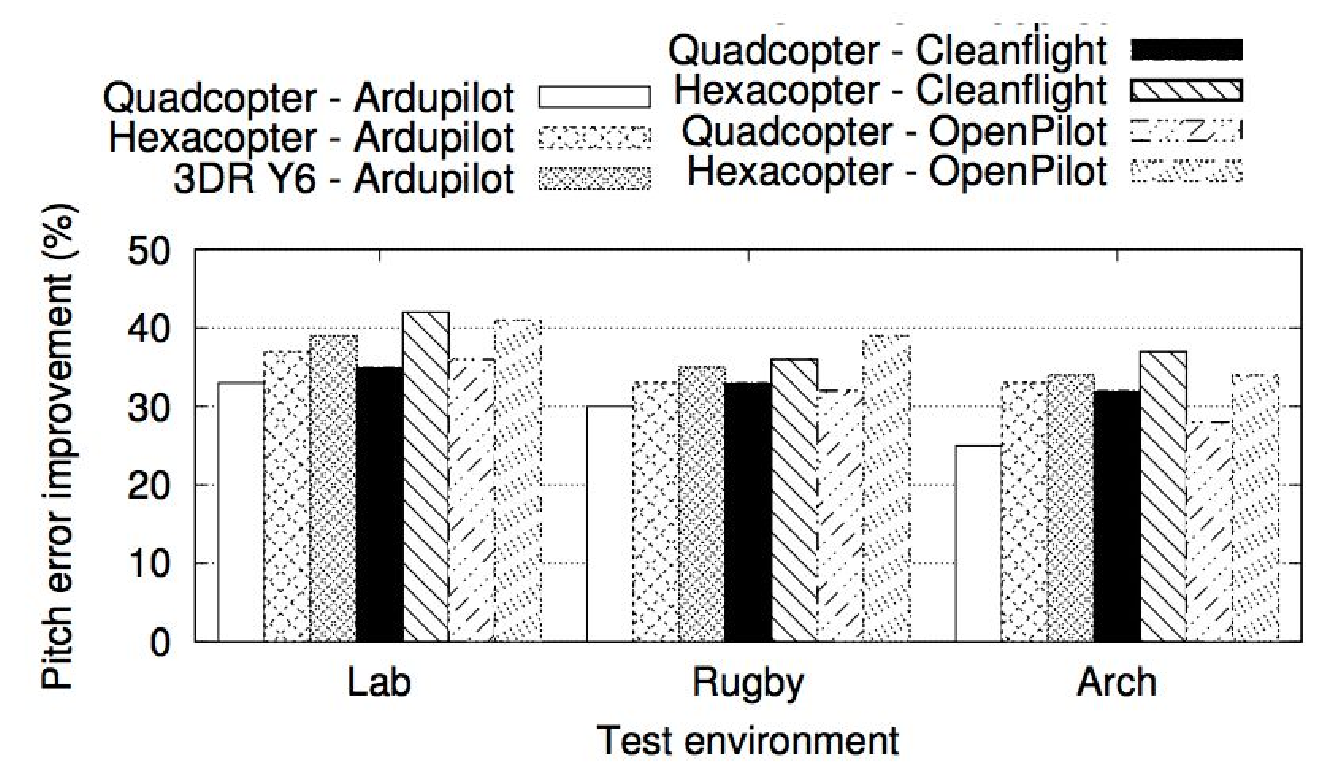
Measure

- **Attitude (motor output) error**
  Difference between the **desired** and **actual** attitude
  Autopilot (desired) Recorded (actual)



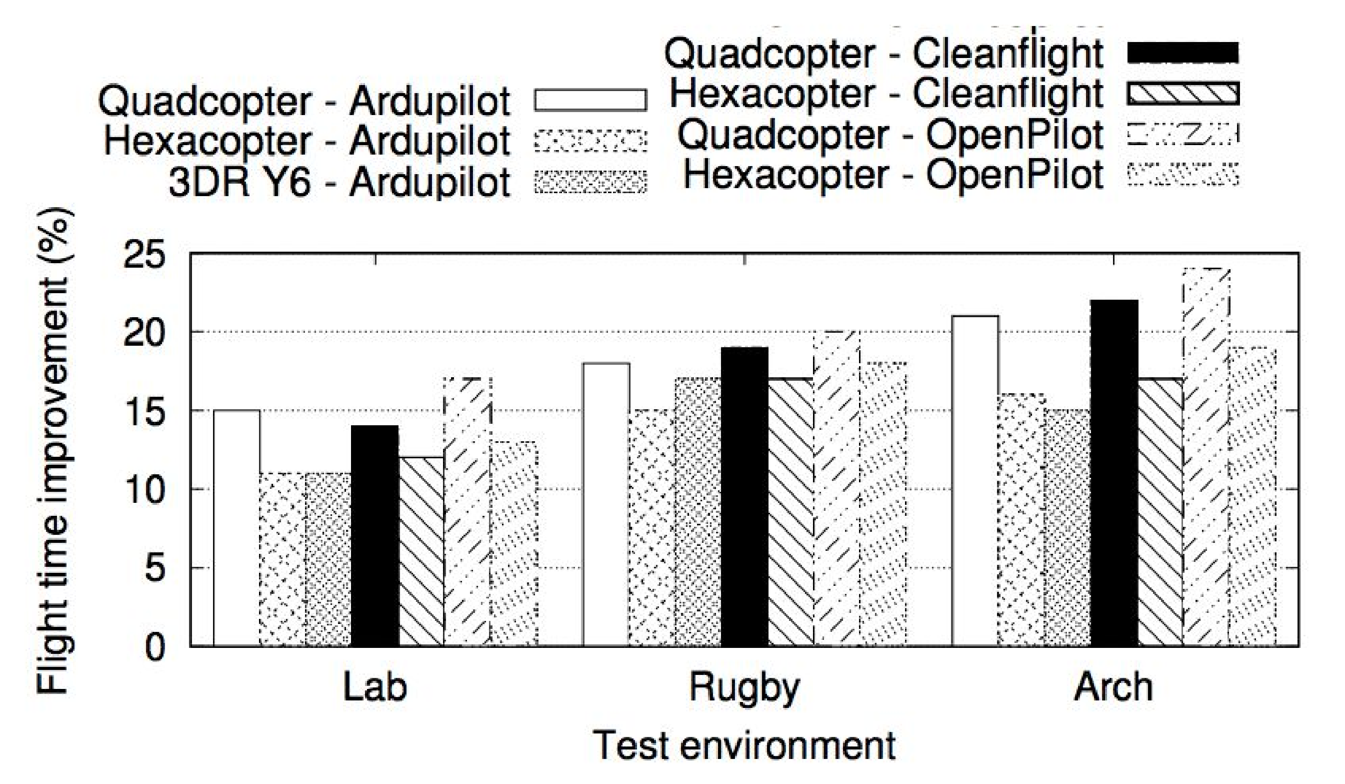
- **Flight time**
  Until battery falls below a 20% threshold

Evaluation



Quadcopter - Ardupilot
Hexacopter - Ardupilot
3DR Y6 - Ardupilot
Quadcopter - Cleanflight
Hexacopter - Cleanflight
Quadcopter - OpenPilot
Hexacopter - OpenPilot
Pitch error improvement (%)
50
40
30
20
10
0
Lab
Rugby
Arch
Test environment

Evaluation



Quadcopter - Cleanflight
Hexacopter - Cleanflight
Quadcopter - OpenPilot
Hexacopter - OpenPilot
Quadcopter - Ardupilot
Hexacopter - Ardupilot
3DR Y6 - Ardupilot
Flight time improvement (%)
25
20
15
10
5
0
Lab
Rugby
Arch
Test environment

Evaluation

3D Reconstruction (Structure from Motion)

30 target points to take pictures

The less stable, the more blurry the image becomes

Result: 29% dense cloud than time-triggered control

Evaluation

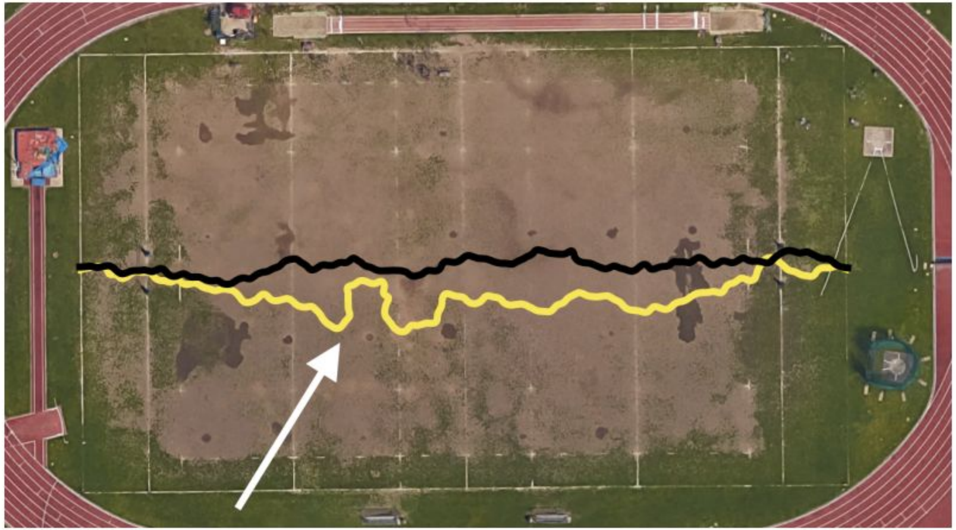


Figure 15: Example of ARVA-driven navigation when using reactive control (black) and time-triggered processing (yellow). *Time-triggered control occasionally produces highly inefficient paths, whereas we never observe similar behaviors with reactive control.*

Evaluation

Closing thoughts - Pros

- Exclusively works in software
  no hardware modification is required.
- Demonstrate that reactive control is applicable beyond waypoint navigation
- Nice solution to callback hell problem
- Impressive experimental results

Closing thoughts - Cons

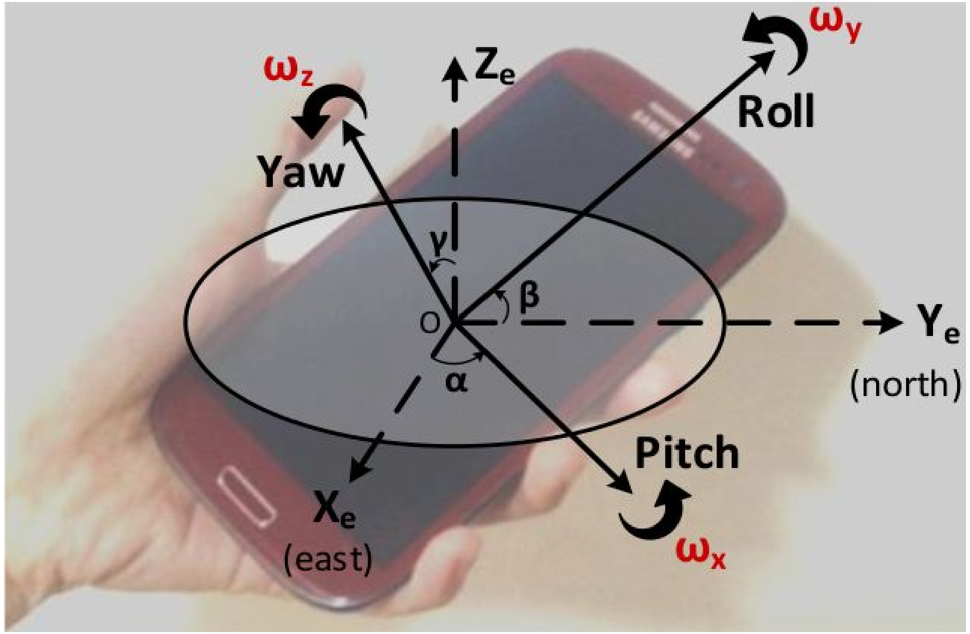
- Has to do periodic sensing & periodic computation of control decisions at highest possible frequency.
- Only execution of control logic is different from the time-triggered control
- Still dependent on time-triggered control (Failsafe)
- Very similar to event-based control

Discussion & Questions

Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things

Teng Wei and Xinyu Zhang

3D orientation using motion sensors (with batteries)



Output of the MEMS gyroscope: 3 angular velocities around the Roll, Y aw, and Pitch axis in the phone body-frame.

image from “Use It Free: Instantly Knowing Your Phone Attitude”, by Pengfei Zhou, Mo Li, Guobin Shen

Passive orientation using computer vision

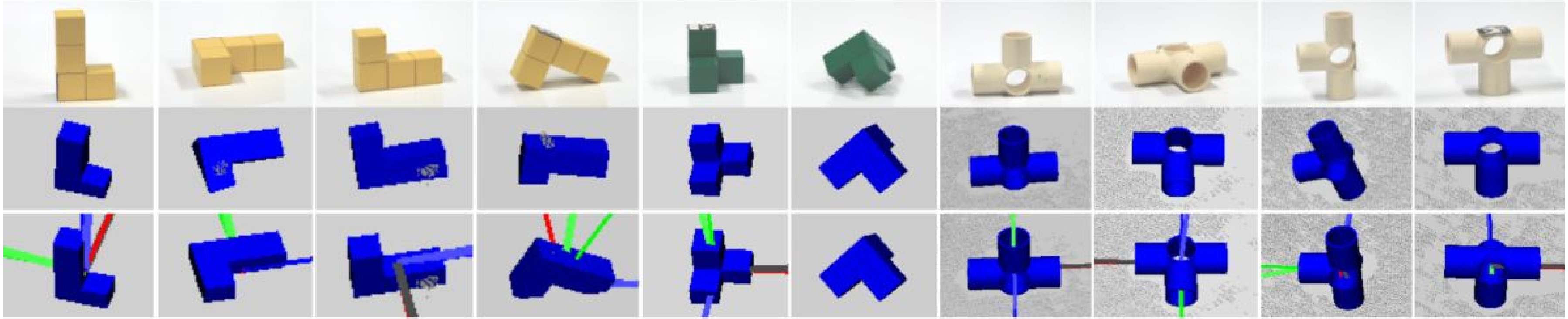


image from “Teaching Robots to Do Object Assembly Using Multi-MOdal 3D Vision”, by Weiwei Wan, Feng Lu, Zepei Wu, Kensuke Harada

Application of passive orientation sensing in IoT

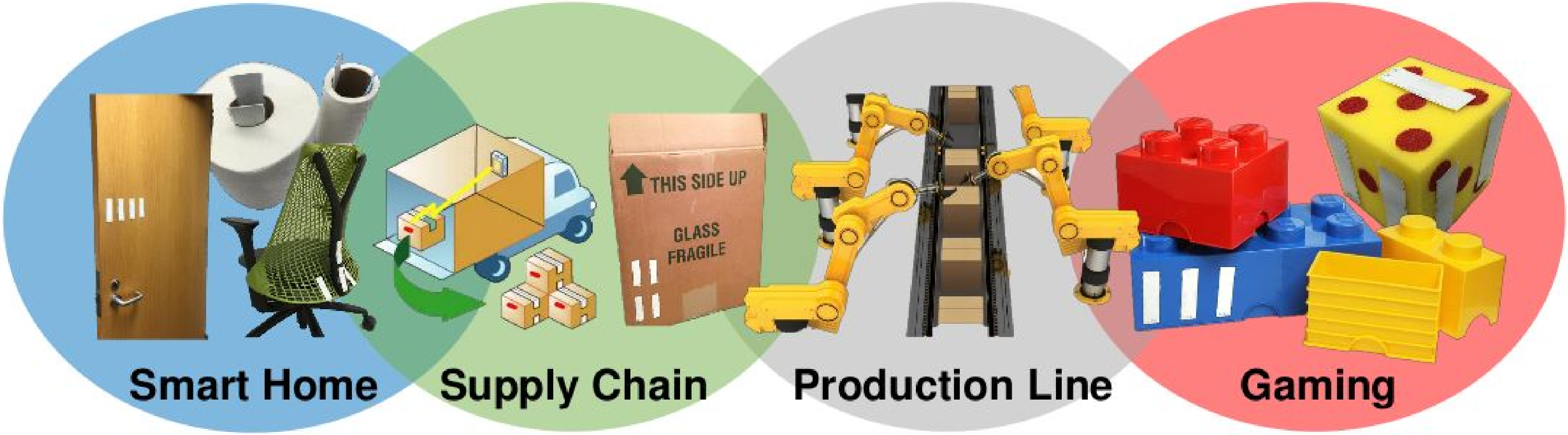
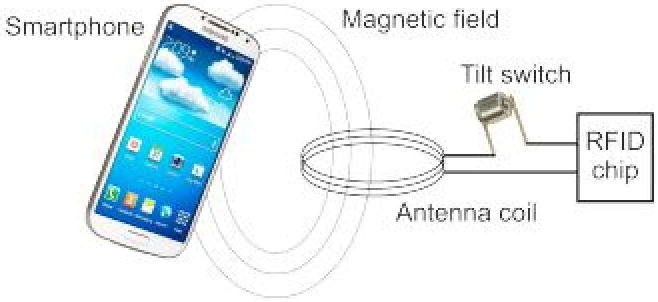


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Passive orientation using RFID tag



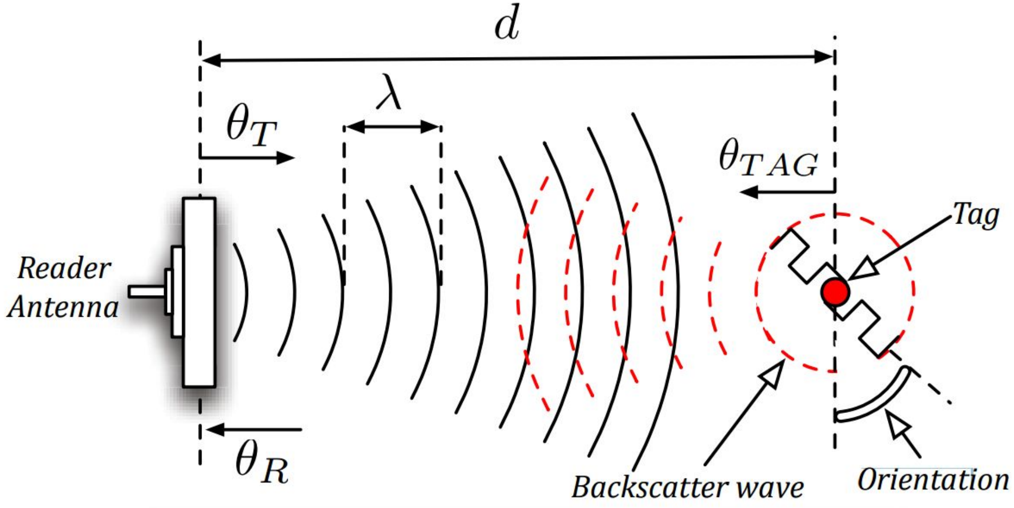
RFID-Die: a tile switch controls whether the RFID tag response.

image from “RFID-Die: Battery-Free Orientation Sensing Using an Array of Passive Tilt Switches”, by L. Büthe, M. Hardegger, P. Brülisauer, G. Tröster

Tagyro

- Build connection between 2 DoFs rotation and phase;
- Compute orientation spectrum from real-time and theoretic phase;
- Extend 2 DoFs to 3 DoFs using 2 RFID tag arrays.

Backscatter communication



$$\theta = \left(\frac{2\pi}{\lambda} \times 2d + \underbrace{\theta_T + \theta_R + \theta_{TAG}}_{\text{diversity term}}\right) \bmod 2\pi$$

$$\phi = \left(\frac{2\pi d}{\lambda} + \phi_{Reader} + \phi_{Tag}\right)\ mod\ 2\pi$$

image from “Tagoram: Real-Time Tracking of Mobile RFID Tags to High Precision Using COTS Devices”, by L. Yang, Y. Chen, X.-Y. Li, C.Xiao, M.Li, Y.Liu

3D coordinate system

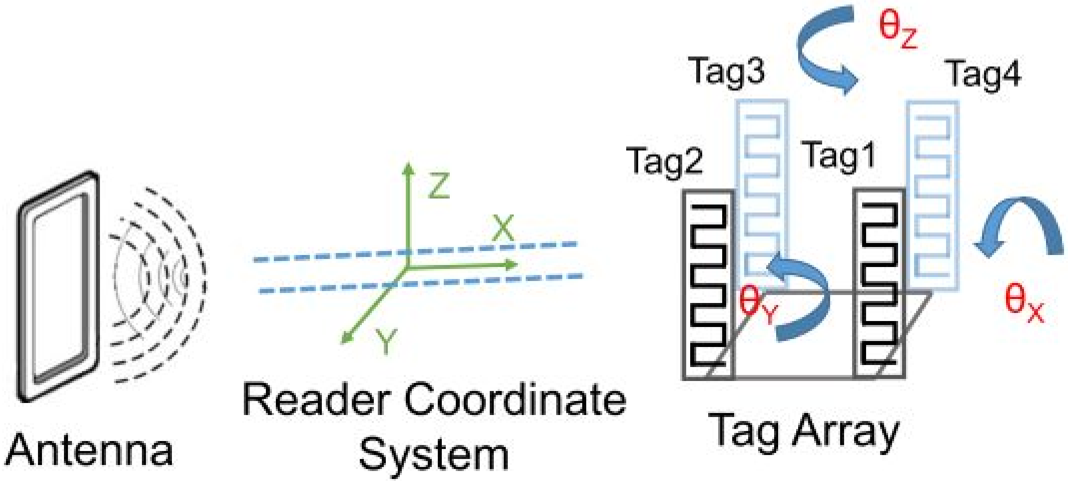
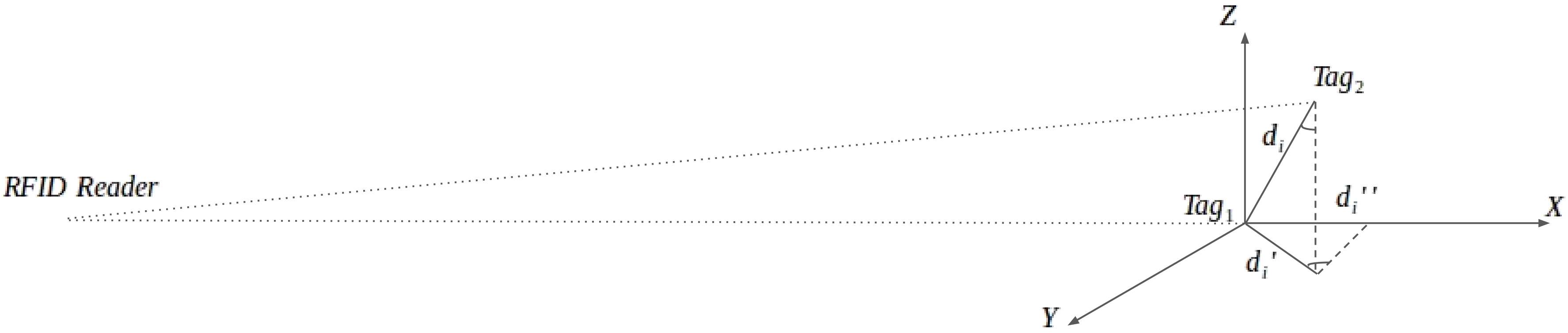


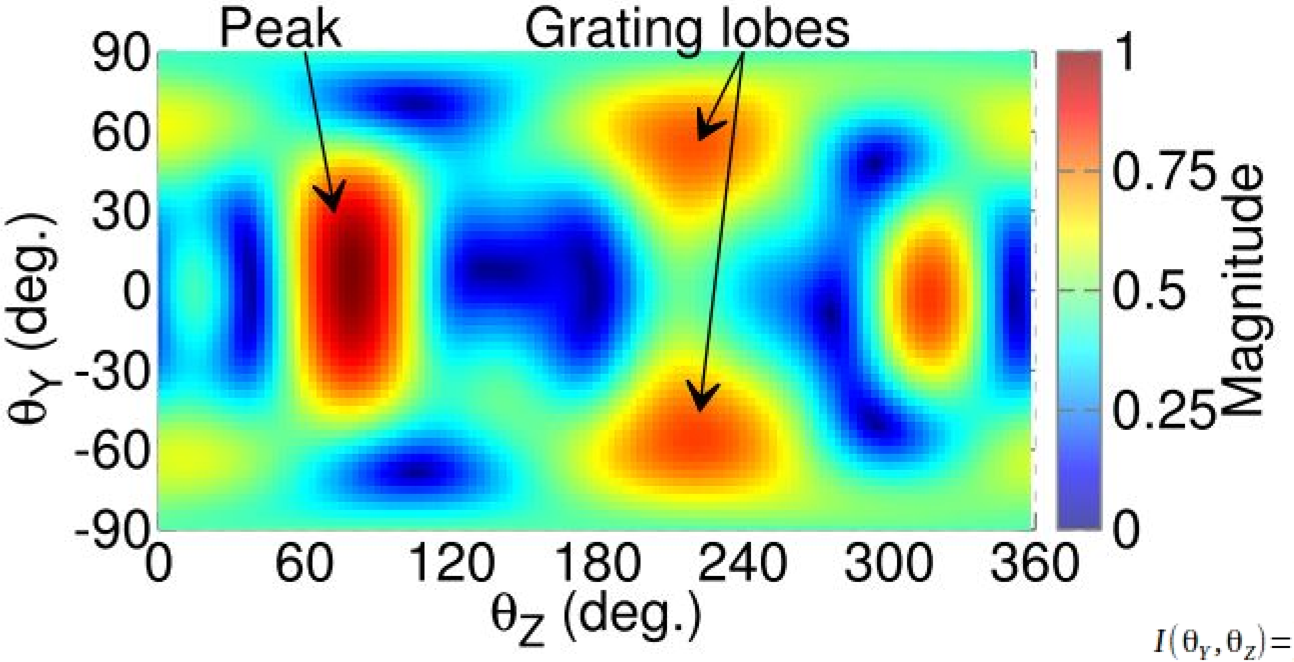
image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Phases of two RFID tags



$$\Delta(\phi_Y, \phi_Z, d_i) = \phi_i - \phi_1 = \left(\frac{2\pi \times 2 d_i''}{\lambda} + \Delta\phi_{Tag}^i\right) mod\ 2\pi = \left(\frac{4\pi d_i \sin(\theta_Y + \theta_Y^i)\sin(\theta_Z + \theta_Z^i)}{\lambda} + \Delta\phi_{Tag}^i\right) mod\ 2\pi$$

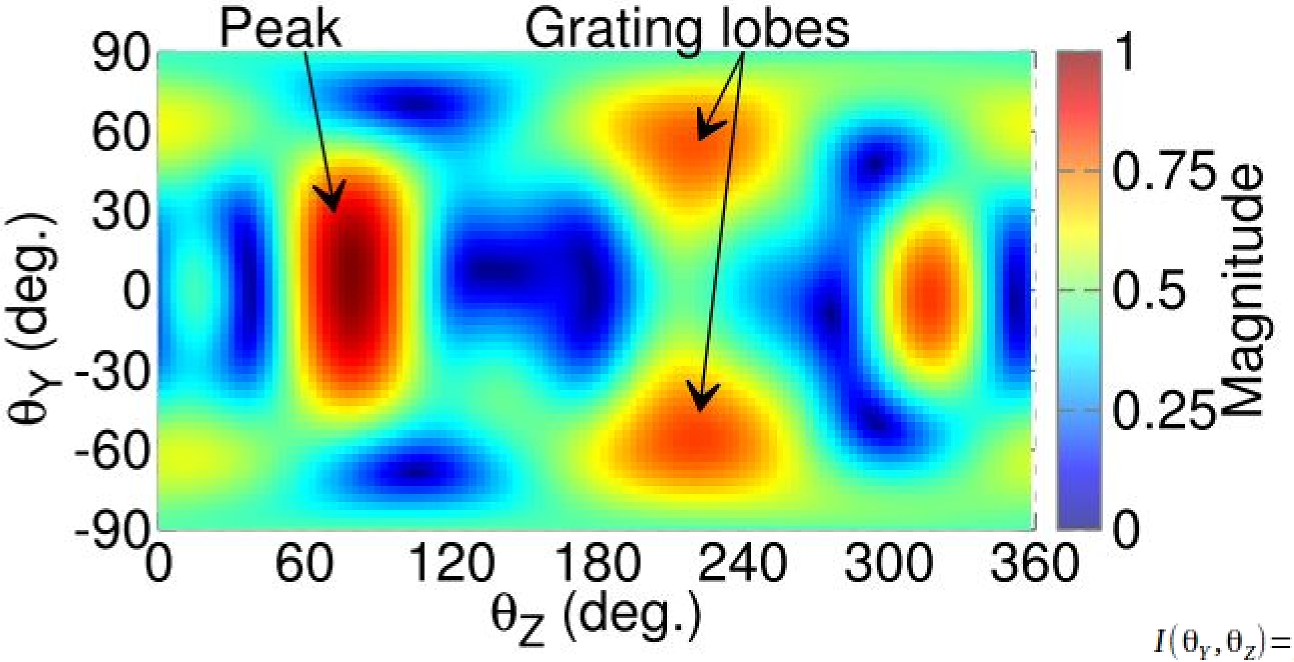
Orientation Spectrum



$$I(\theta_Y, \theta_Z) = \sum_{i=1}^{K} \exp\left(j\left(\Delta_i \phi(\theta_Y, \theta_Z, d_i) - \Delta\hat{\phi}_i\right)\right) / K$$

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Orientation Spectrum



$$I(\theta_Y, \theta_Z) = \sum_{i=1}^{K} \exp\left(j\left(\Delta_i \phi(\theta_Y, \theta_Z, d_i) - \Delta\hat{\phi}_i\right)\right) / K$$

How do they deal with grating lobes caused by spatial ambiguity?

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Spatial ambiguity

- Antennas need to be separated by less than half-wavelength;
  - In Tagyro, a quarter, thus 8.2 cm for 915MHz;
- Search for top three peaks;
- Take the one that is closest to the previous one.

Challenges

- A RFID tag doesn’t act as an isotropic point;
- RFID tags affect each other when deployed in close range;
- The computation needs to know the layout.

The orientation of an RFID tag

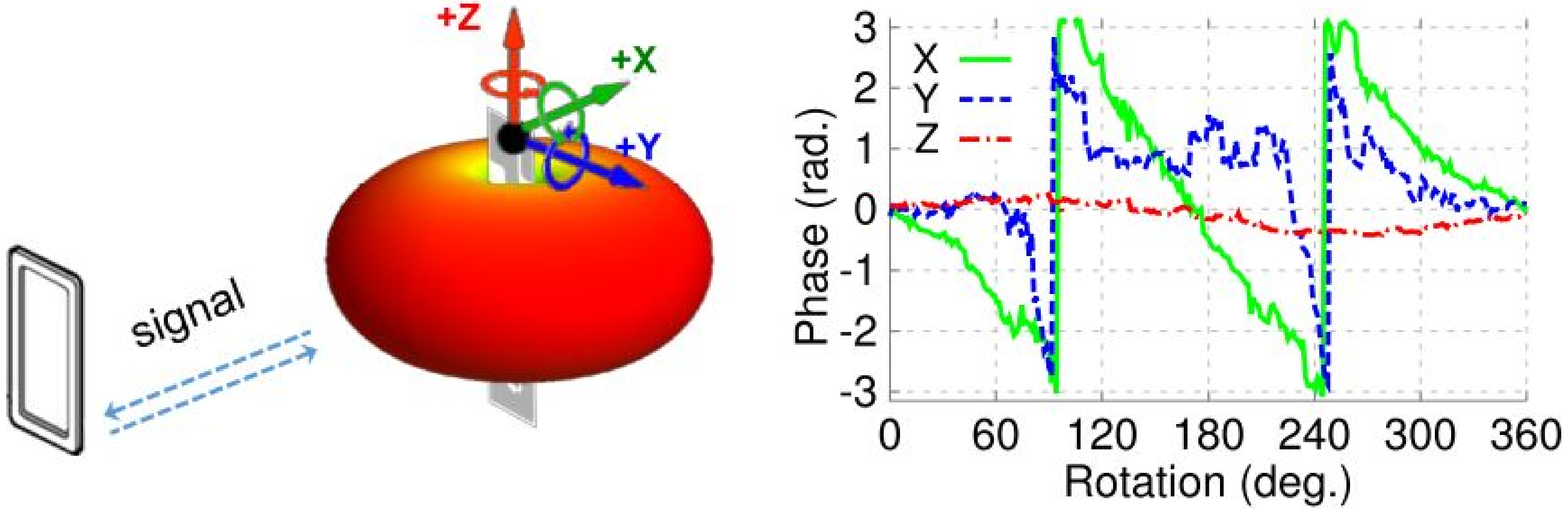
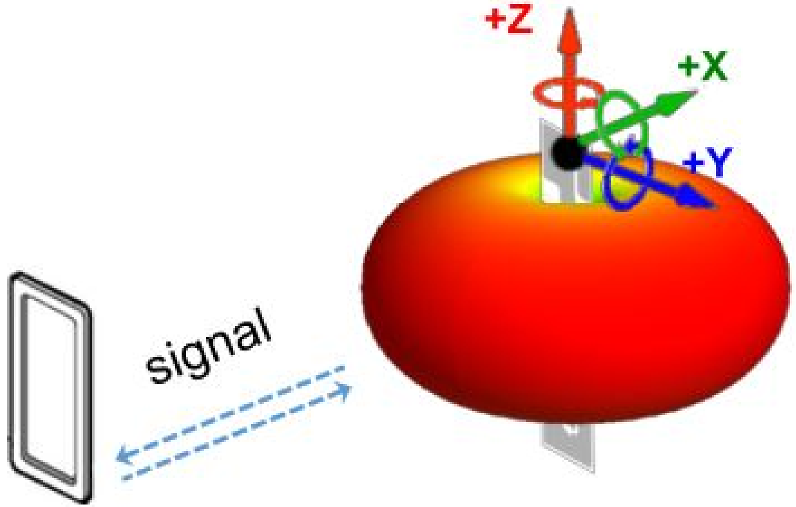
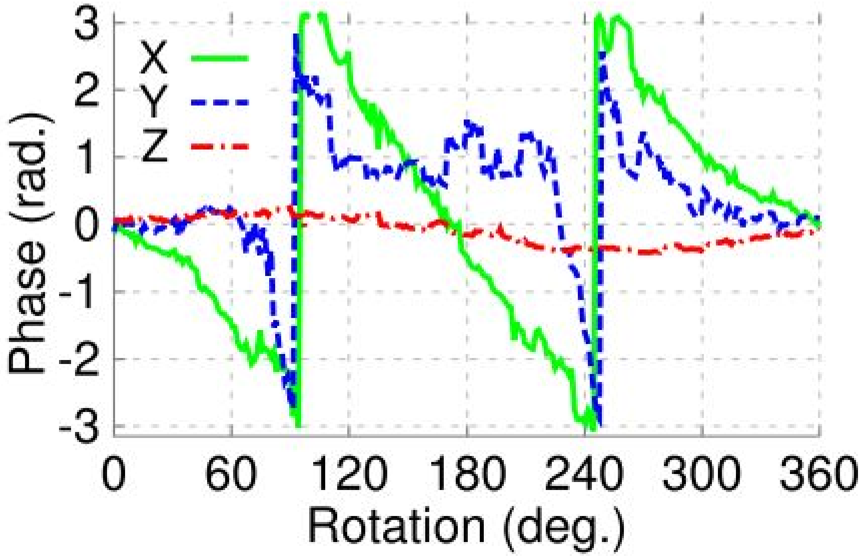


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

The orientation of an RFID tag

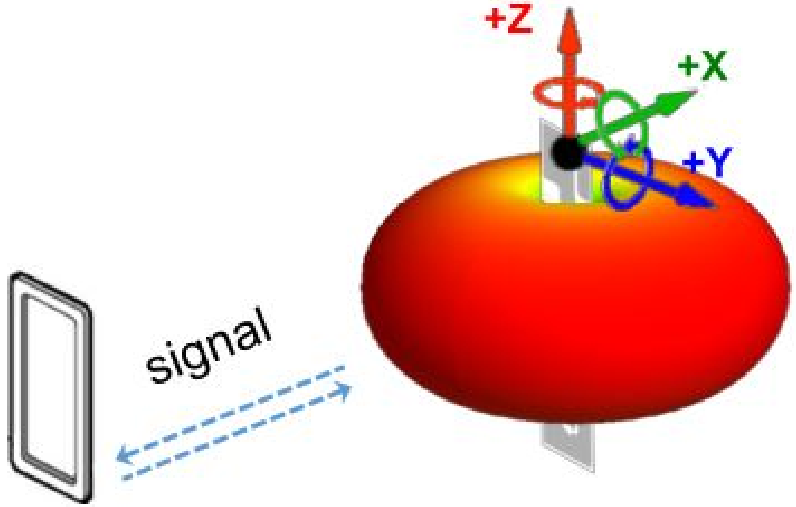




How do they deal with this limitation?

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

The blind direction of an RFID tag



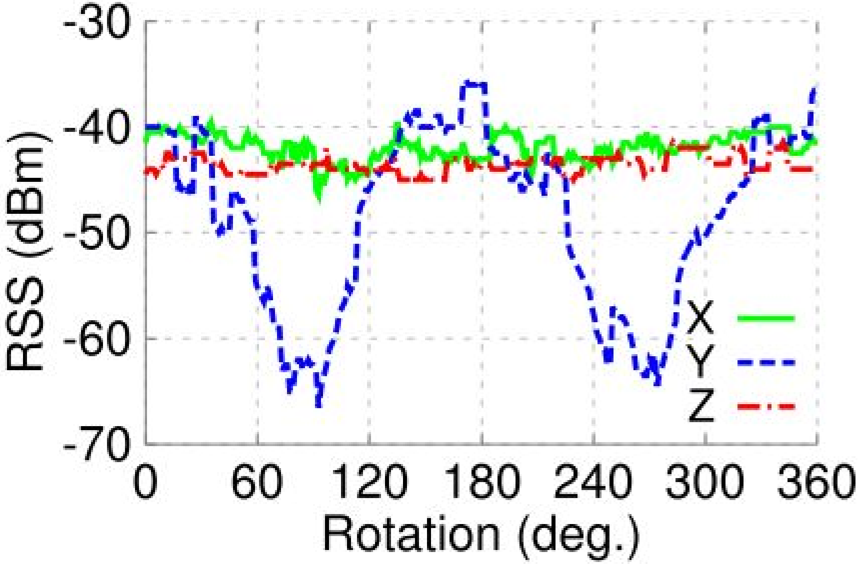


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

The coupling effect of RFID tags

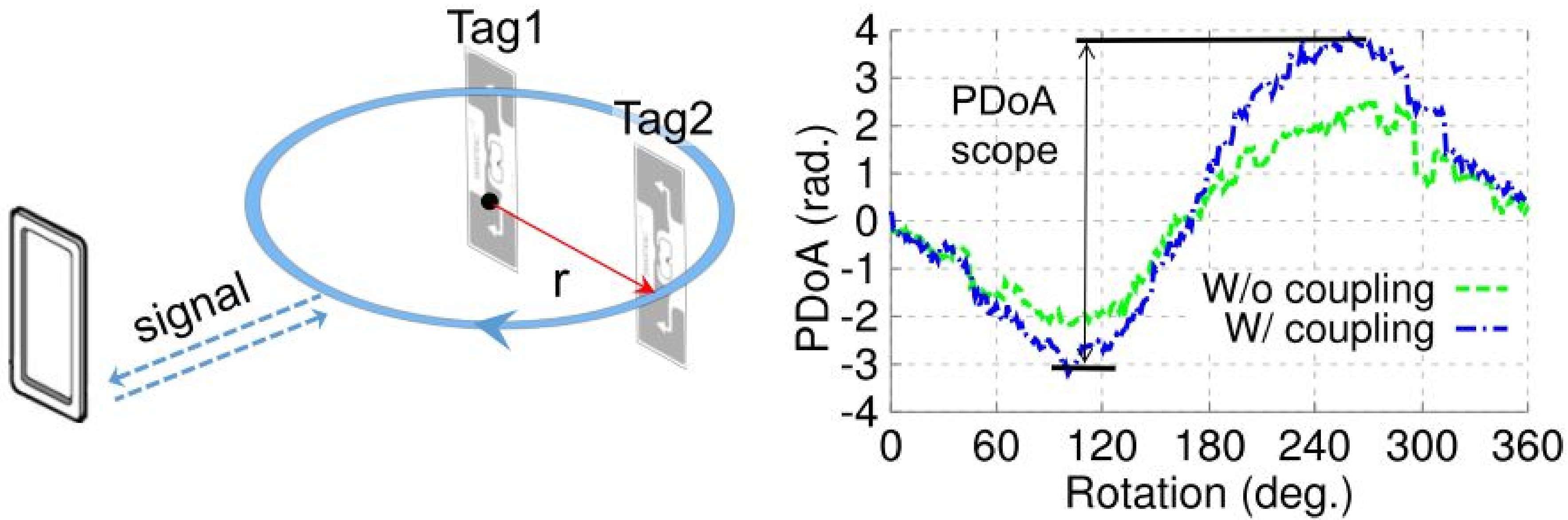


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

PDoA deviation over tag separation distance

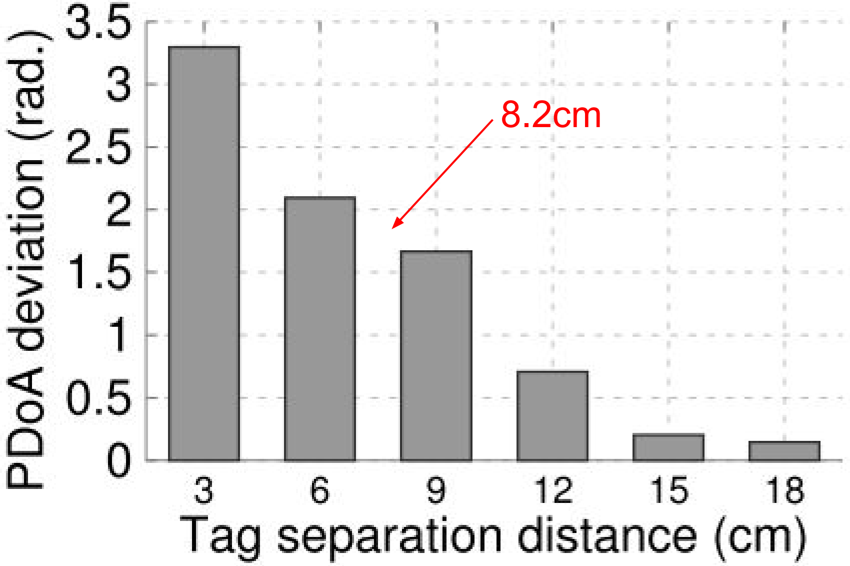
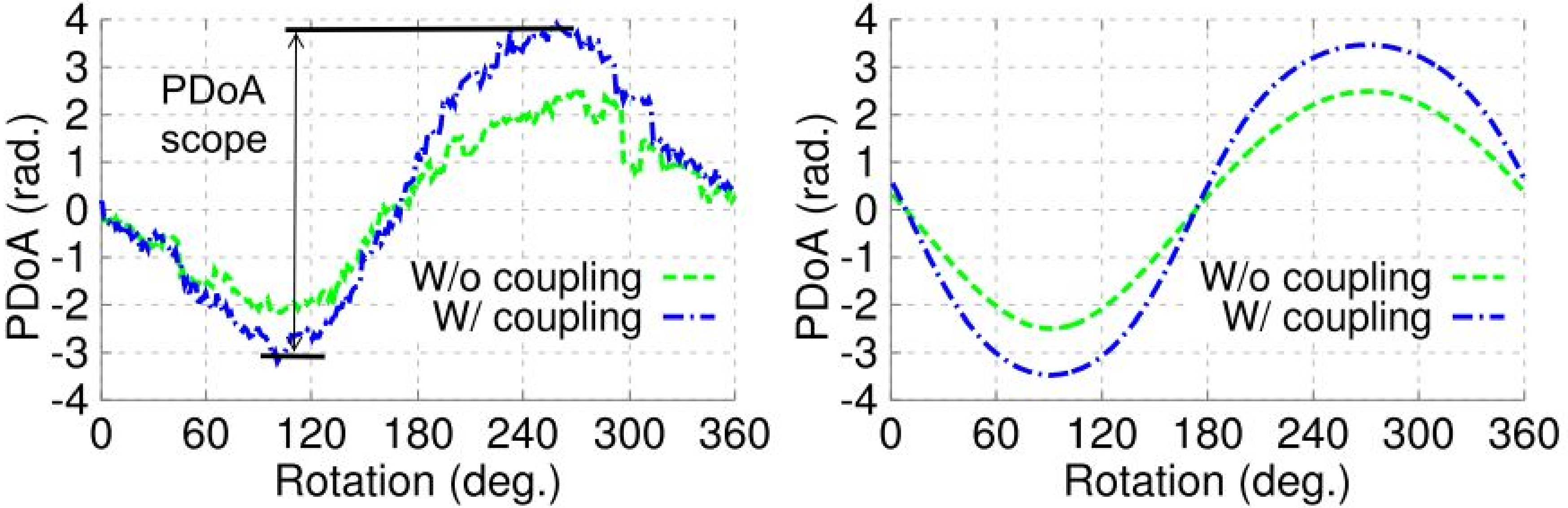


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

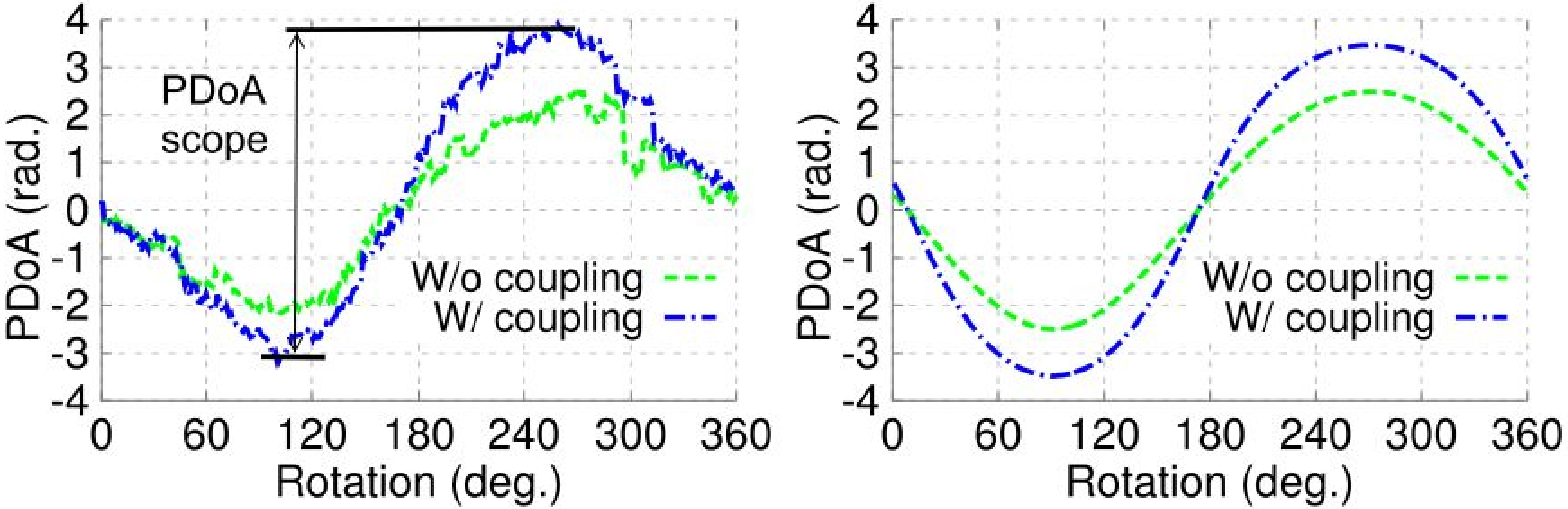
The resonant signal



$$s_1 = A_1 \exp\left(j 2\pi \frac{2d_1}{\lambda}\right), s_2^c = A_2^c \exp\left(j 2\pi \frac{d_1 + d_2 + r}{\lambda} + \phi^c\right), A_1 = 1, A_2^c = 0.5, \phi^c = \pi/2$$

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

The resonant signal



$s_1 = A_1 \exp(j 2\pi \frac{2 d_1}{\lambda}), s_2^c = A_2^c \exp(j 2\pi \frac{d_1 + d_2 + r}{\lambda} + \phi^c), A_1 = 1, A_2^c = 0.5, \phi^c = \pi/2$

What is the observation they make to simplify this situation?

image from "Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things", by Teng Wei, Xinyu Zhang

Effective distance

$$\{\hat{d}_i, 1 \leq i \leq K\} = \underset{d_i, 1 \leq i \leq K}{argmax} \left| \sum_{i=1}^{K} \exp(j(\Delta\phi_i(\theta_Y, \theta_Z, d_i) - \Delta\hat{\phi}_i)) \right|$$

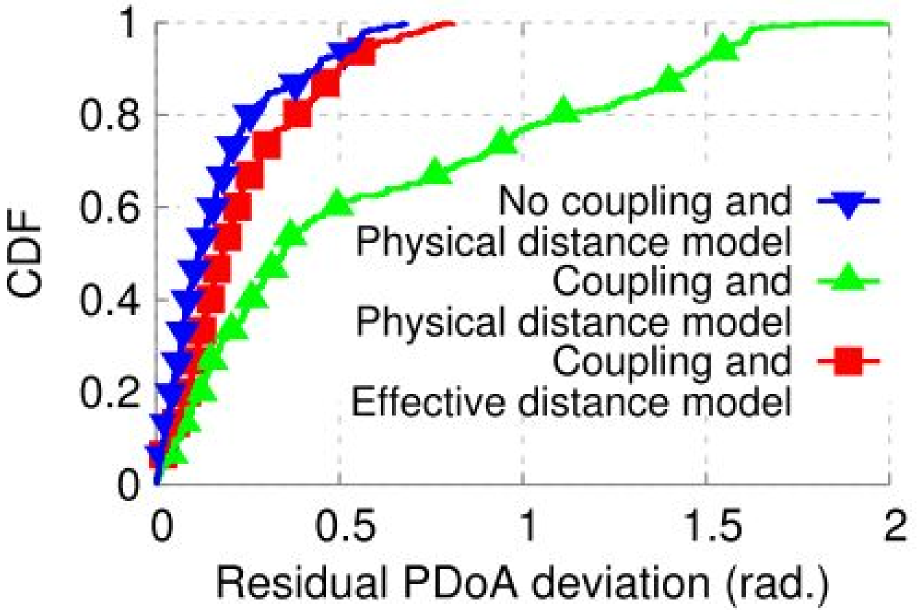
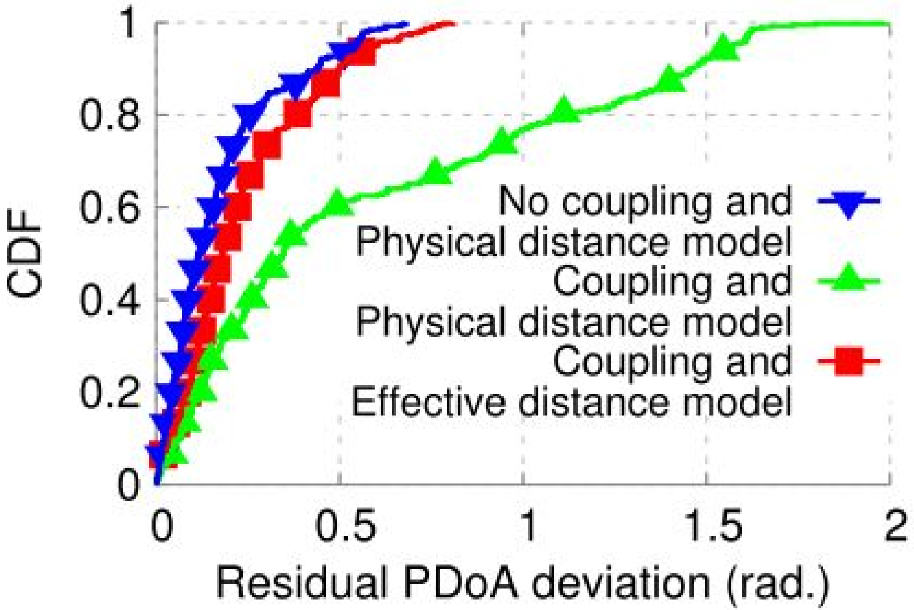


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Effective distance

$$\{\hat{d}_i, 1\le i\le K\} = \underset{d_i, 1\le i\le K}{argmax} \left|\sum_{i=1}^{K} \exp(j(\Delta\phi_i(\theta_Y,\theta_Z,d_i) - \Delta\hat{\phi}_i))\right|$$



$$\Delta(\phi_Y,\phi_Z,d_i) = \left(\frac{4\pi d_i \sin(\theta_Y+\theta_Y^i)\sin(\theta_Z+\theta_Z^i)}{\lambda} + \Delta\phi_{Tag}^i\right) \bmod 2\pi$$

What is the observation they make about the bound of PDoA?

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Array Layout Sensing (ALS) algorithm

- PDoA value is bounded within $[-4\pi\hat{d}/\lambda+\Delta\phi_{Tag}, 4\pi\hat{d}/\lambda+\Delta\phi_{Tag}]$ $\hat{d}=\frac{\lambda}{4}\frac{PDoAScope}{2\pi}$
- Rotate tag array by more than one cycle round each axis;
- Phase unwrapping: PDoA change greater than π or smaller than $-\pi$;
- Map the tags' pairwise effective distance to the entire tag array's effective layout;
    - Classical Multi-Dimensional Scaling problem;
    - Approximate algorithm solve in $O(KlogK)$
    - Take first 3 dimensions.

ALS algorithm

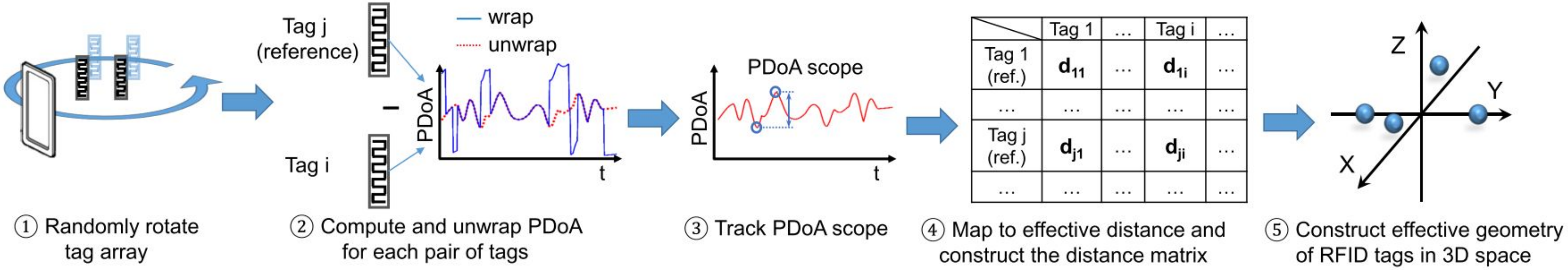


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

How to get tag’s initial phase offset

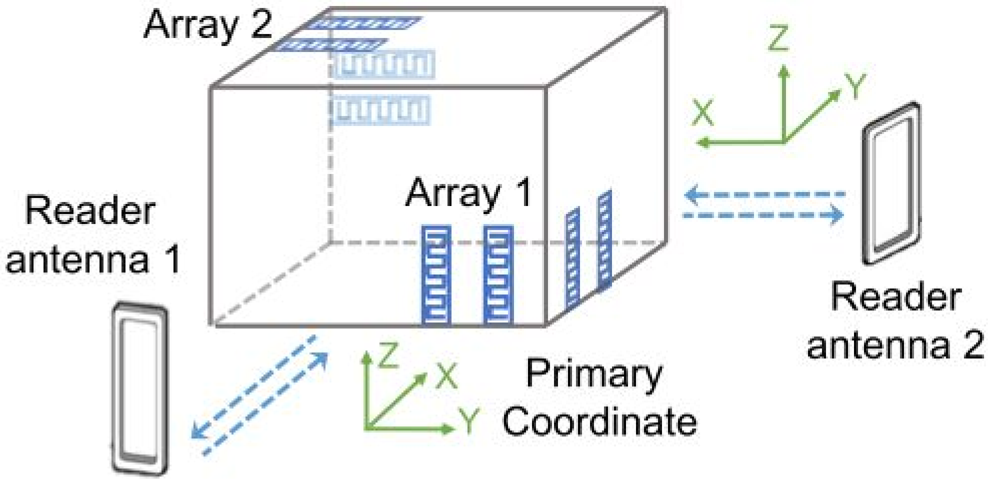
$$\Delta(\phi_Y, \phi_Z, d_i) = \left(\frac{4\pi d_i \sin(\theta_Y + \theta_Y^i)\sin(\theta_Z + \theta_Z^i)}{\lambda} + \Delta\phi_{Tag}^i\right) mod\ 2\pi$$

How to get tag's initial phase offset

$$\Delta(\phi_Y, \phi_Z, d_i) = \left(\frac{4\pi d_i \sin(\theta_Y + \theta_Y^i)\sin(\theta_Z + \theta_Z^i)}{\lambda} + \Delta\phi_{Tag}^i\right) \, mod \, 2\pi$$

$$[-4\pi \hat{d}/\lambda + \Delta\phi_{Tag}, 4\pi \hat{d}/\lambda + \Delta\phi_{Tag}]$$

Dual RFID tag arrays for 3-DoF orientation



- Orthogonal to each other;
- Also solve blind spot problem;
- 4 DoFs.

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Combo validator for blind direction problem

- At most one of the two arrays is in the blind direction (orthogonal);
- Set one array as blind when average RSS for tags is more than δ smaller than the other;
    - 5dB based on measurements (conservative);

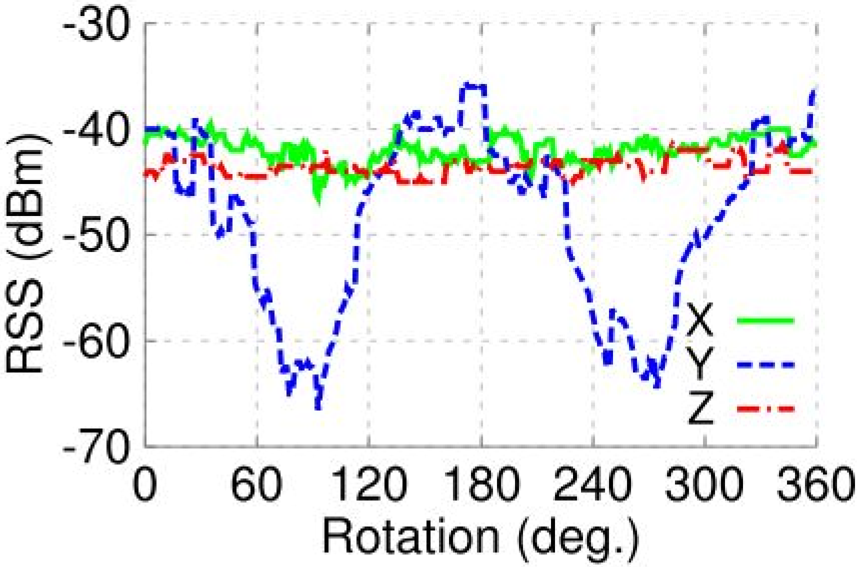


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

3D orientation

- Take first antenna's coordinate system as primary;
- $R_i A_j$ reader i and array j;
- $$I(\theta_X, \theta_Y, \theta_Z) = \sum_{j=1}^{2} I_{R1Aj}(\theta_Y, \theta_Z) + \sum_{j=1}^{2} I_{R2Aj}(\theta_X, \theta_Z)$$
  $$I(\theta_Y, \theta_Z) = \sum_{i=1}^{K} \exp(j(\Delta_i \phi(\theta_Y, \theta_Z, d_i) - \Delta \hat{\phi}_i)) / K$$

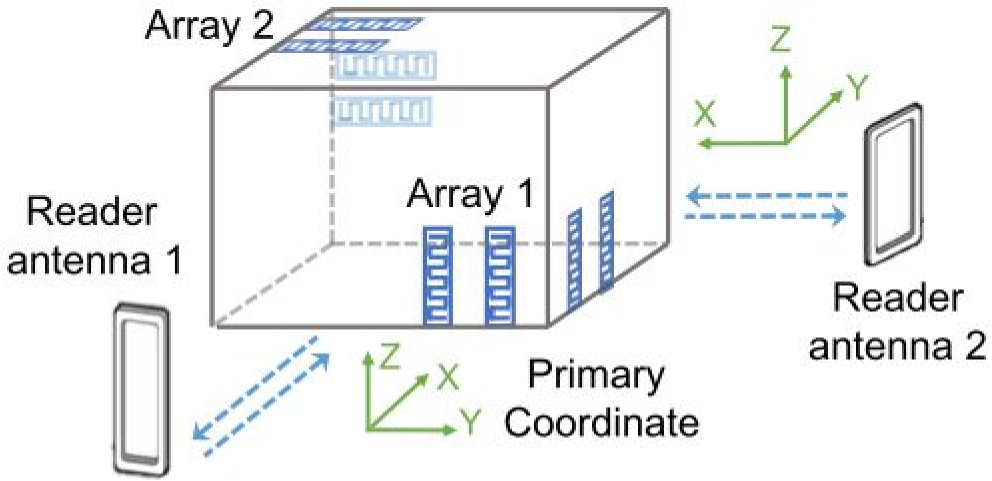


image from "Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things", by Teng Wei, Xinyu Zhang

Frequency-Hopping Readers

What is the problem?

Frequency-Hopping Readers

- Commercial UHF RFID readers must randonly hop to one of 50 center frequencies within 902-928 MHz band every 200 ms, following FCC regulation;

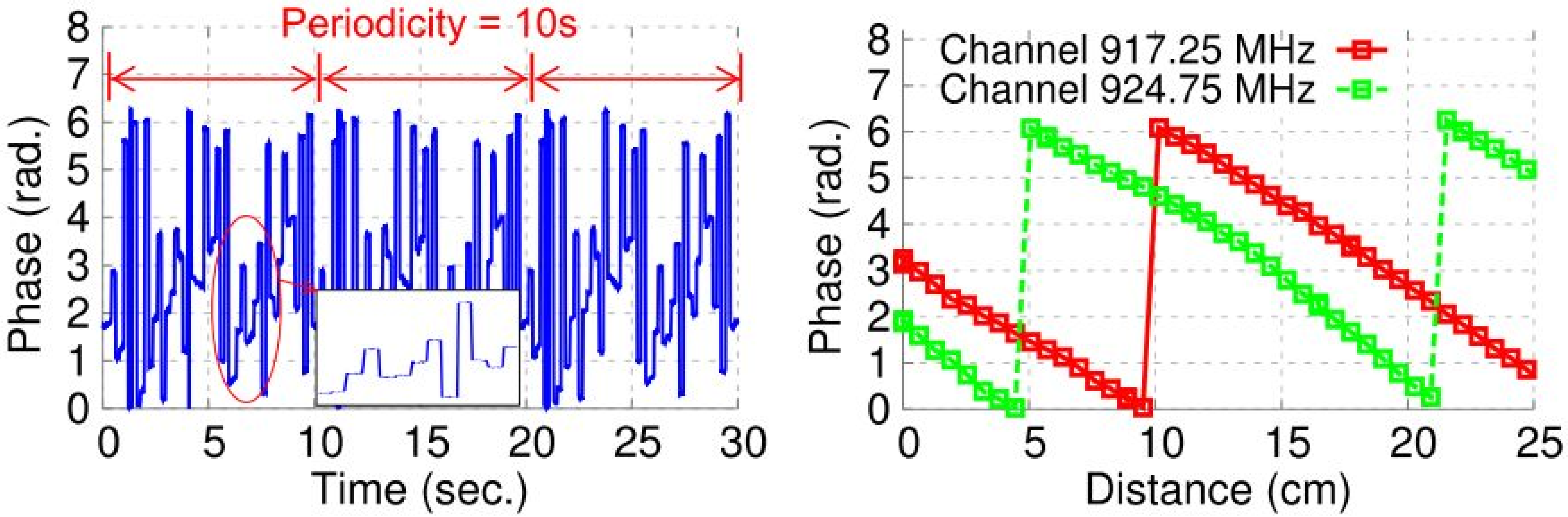


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Initial phase measurement

- 10 seconds for all frequencies;
- Map to a common frequency (default 915.25 MHz).

$$\phi(f_i, d_0) = (2\pi f_i d_0 / c + \beta_i)\ mod\ 2\pi$$

$$\phi(f_r, d) = (((f_i, d) - \phi(f_i, d_0))\frac{f_r}{f_i} + \phi(f_r, d_0))\ mod\ 2\pi$$

Asynchronous Phase Reading

- The EPC Gen2 RFID standard reads the tags asynchronously;
- Assume the tag array's rotation speed remains similar over consecutive phase readings.

$$\phi(t)=\phi(t_{i-1})+(\phi(t_i)-\phi(t_{i-1}))\frac{t-t_{i-1}}{t_i-t_{i-1}}$$

Experiments

- RFID readers:
  - Impinj R420;
- RFID tags:
  - ALN-9740;
  - SMARTRAC DogBone;
  - SMARTRAC ShortDipole;
  - Read distance 15 - 20 ft;
- Software:
  - 3D GUI;
  - RFID library;
  - Processing algorithm.

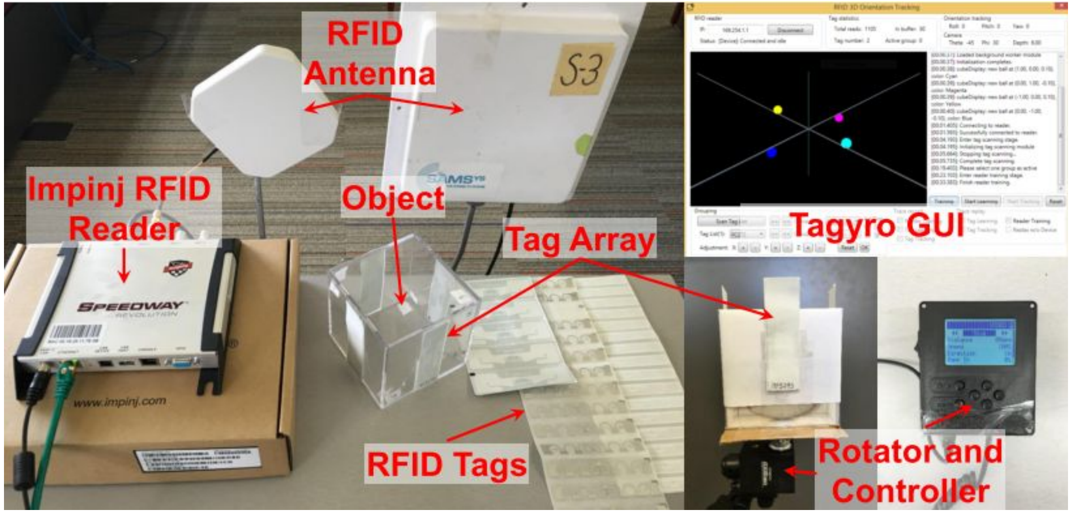


image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Evaluation

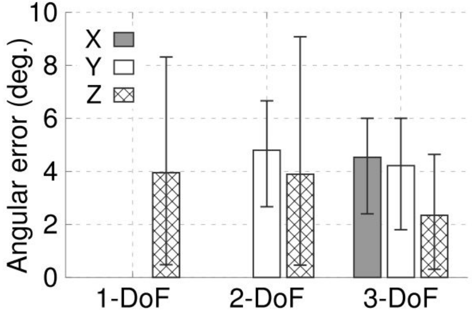


Figure 26: Accuracy vs. DoF. Error bar shows the 90^{th} error.

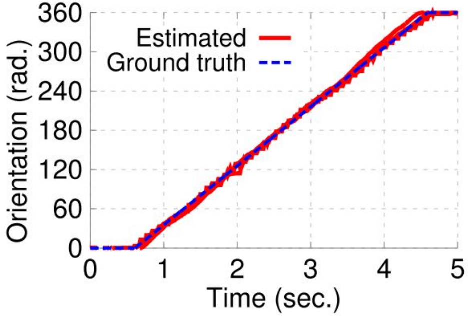


Figure 27: Estimation and ground-true for 1-DoF rotation.

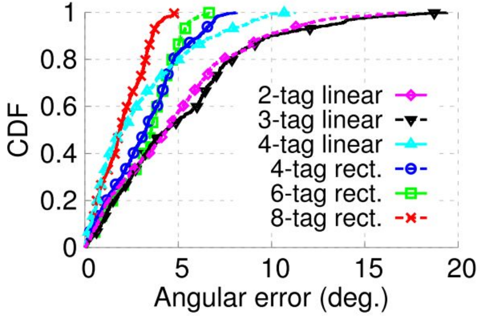


Figure 28: The CDF of orientation error vs. size of tag array.

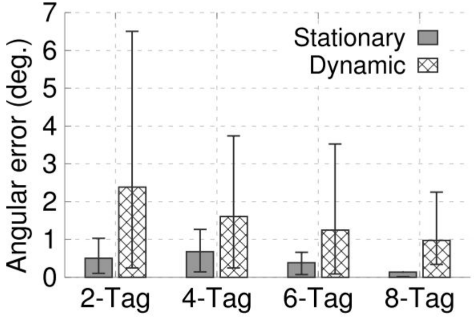


Figure 29: Orientation error under surrounding human activity. Error bar shows the 90^{th} error.

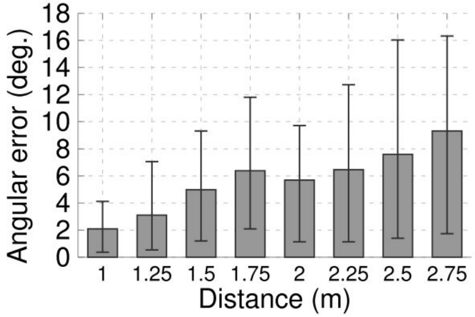


Figure 30: Orientation error over tag-to-antenna distance.

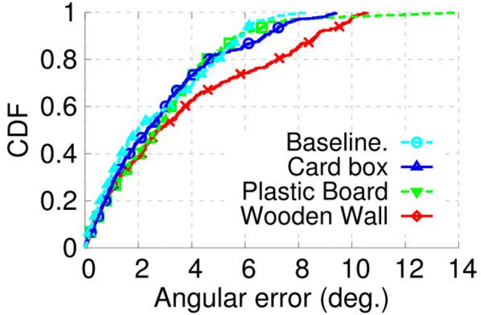


Figure 31: The CDF of orientation error under blockage.

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Evaluation

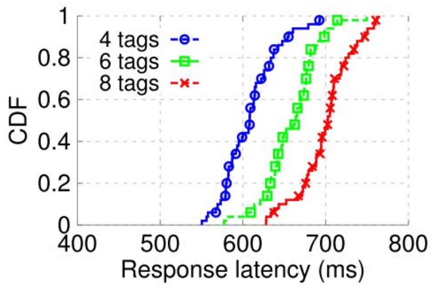


Figure 32: Latency of rotation response.

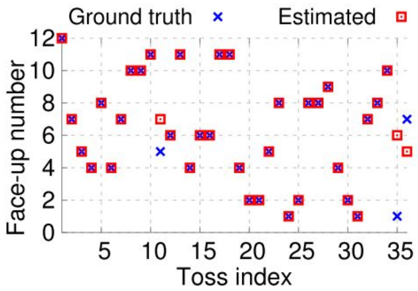


Figure 33: Detecting which side of the dice faces up.

Figure 34: Eight tags are attached to the surface of a 12-side dice.

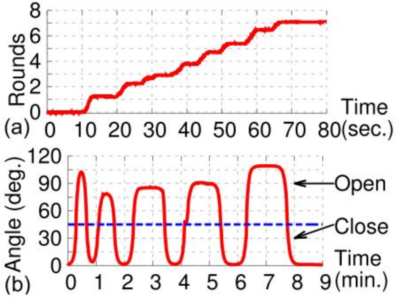


Figure 35: Detecting (a) usage of roll tissue and (b) open/close status of the door.

image from “Gyro in the Air: Tracking 3D Orientation of Batteryless Internet-of-Things”, by Teng Wei, Xinyu Zhang

Other topic

- Multipath Effects;
- Coupling Effect from nearby metallic objects;
- Size of tags;
- Tracking orientation of multiple objects.

https://www.youtube.com/watch?v=sxTKrBZXP7k

EkhoNet: High Speed Ultra Low-power Backscatter for Next Generation Sensors

COMP 790 Internet of Things

Shiwei Fang

Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

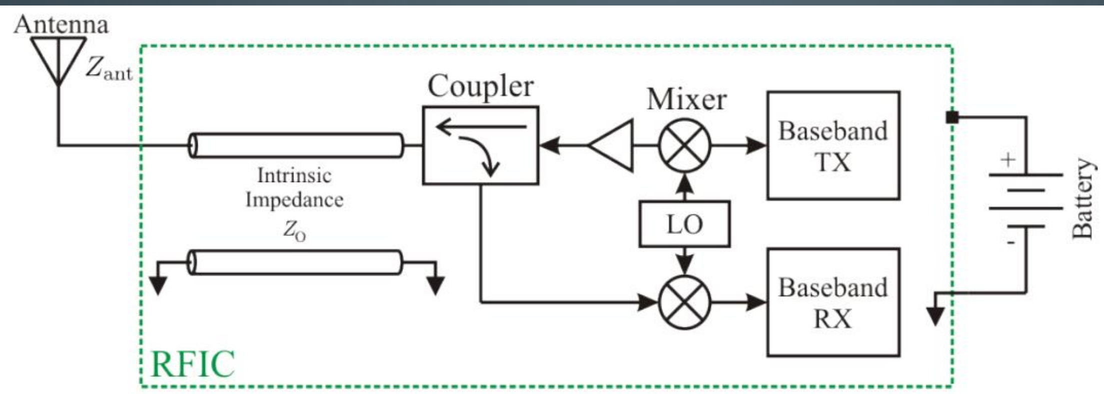
Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Background

- Active RF vs. Backscatter

What is Active RF?

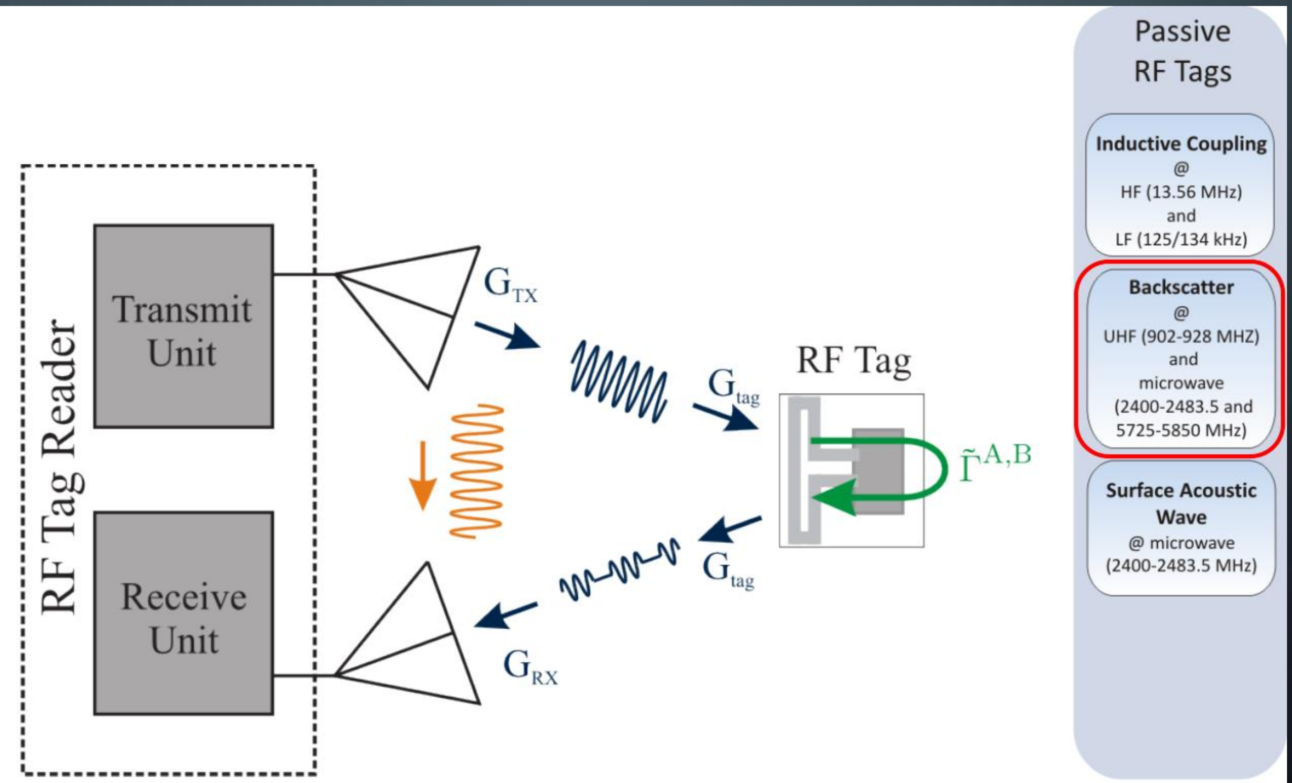


Active RF Tags

Conventional TX/RX
UHF (902-928 MHZ) and microwave (2400-2483.5 and 5725-5850 MHz)

Disney Research

So Passive?



Disney Research

Comparison

	Active RFID	Passive RFID
Power	Battery operated	No internal power
Required Signal Strength	Low	High
CommunicationRange	Long range (100m+)	Short range (3m)
Range Data Storage	Large read/write data (128kb)	Small read/write data (128b)
Per Tag Cost	Generally, $15 to $100	Generally, $0.15 to $5.00
Tag Size	Varies depending on application	"Sticker" to credit card size
Fixed Infrastructure Costs	Lower – cheaper interrogators	Higher – fixed readers
Per Asset Variable Costs	Higher – see tag cost	Lower – see tag cost
Best Area of Use	High volume assets moving within designated areas ("4 walls") in random and dynamic systems	High volume assets moving through fixed choke points in definable, uniform systems
Industries/Applications	Auto dealerships, auto manufacturing, hospitals – asset tracking, construction, mining, laboratories, remote monitoring, IT asset management	Supply chain, high volume manufacturing, libraries/bookstores, pharmaceuticals, passports, electronic tolls, item level tracking

Background Cont’d

- Power consumption
- CMOS
- Voltage
- Capacitance
- Clock cycle
- Duty cycle

Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Problem

- Communication and computation takes too much power when compared to sensor itself

So?

- Complete redesign the system so that the power consumption drop to the μW range.

Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

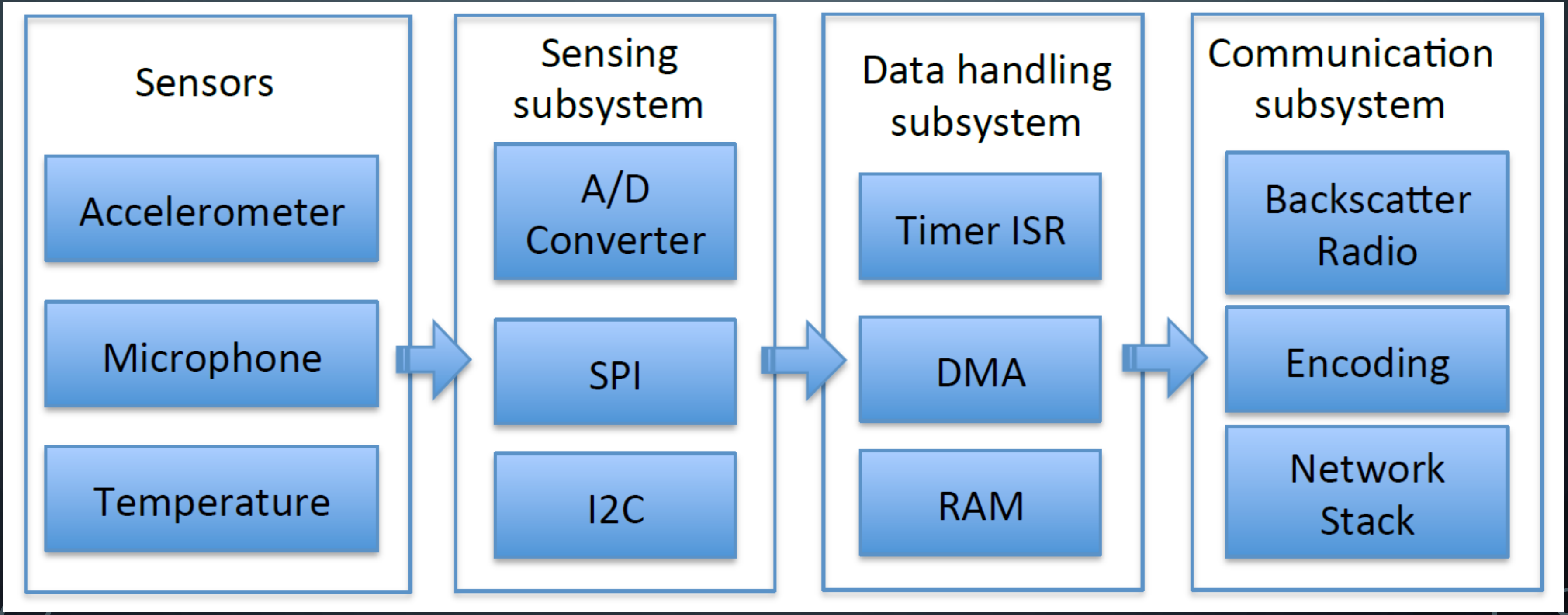
Existing System

- Sensor Data Acquisition
- Data Handling Subsystem
- Communication Subsystem
- Transmission Efficiency
- Summary

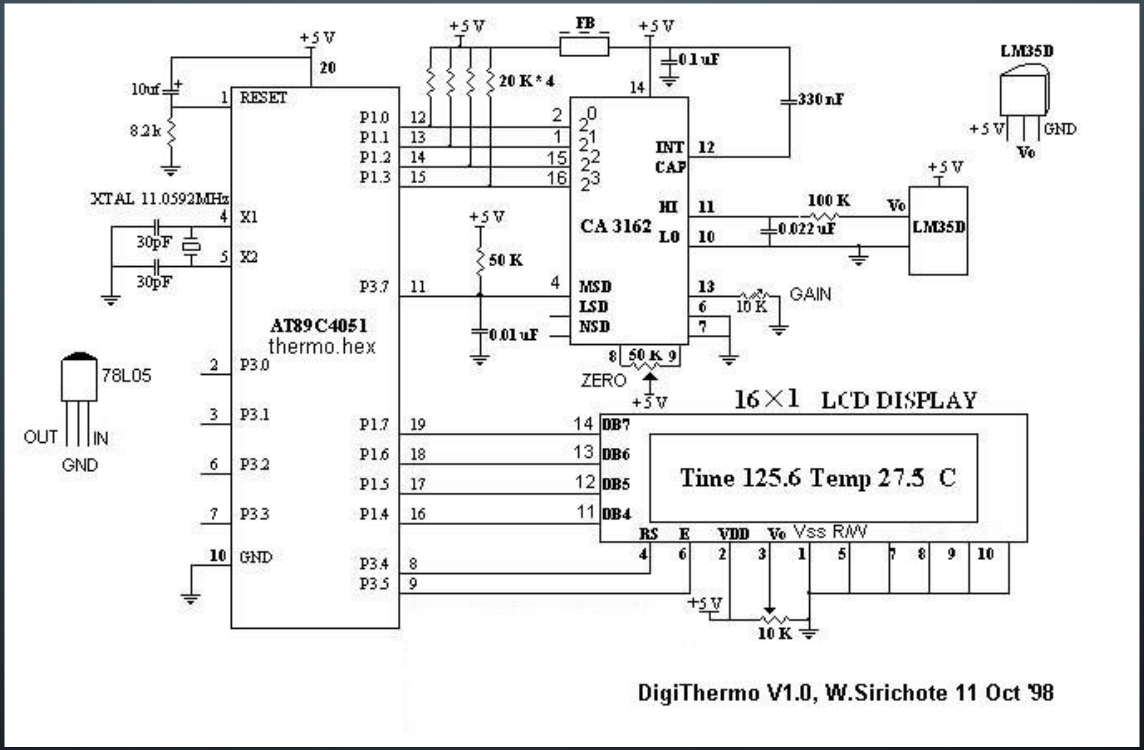
Sensor Data Acquisition

- Two types:
  - Sensor -> On-board ADC -> SPI/I2C -> Micro-controller
  - Sensor (analog) -> Micro-controller ADC
- Simple, Straight Forward.
- Yet not Efficient

Example


Sensors
Accelerometer
Microphone
Temperature
Sensing subsystem
A/D Converter
SPI
I2C
Data handling subsystem
Timer ISR
DMA
RAM
Communication subsystem
Backscatter Radio
Encoding
Network Stack

One Perspective

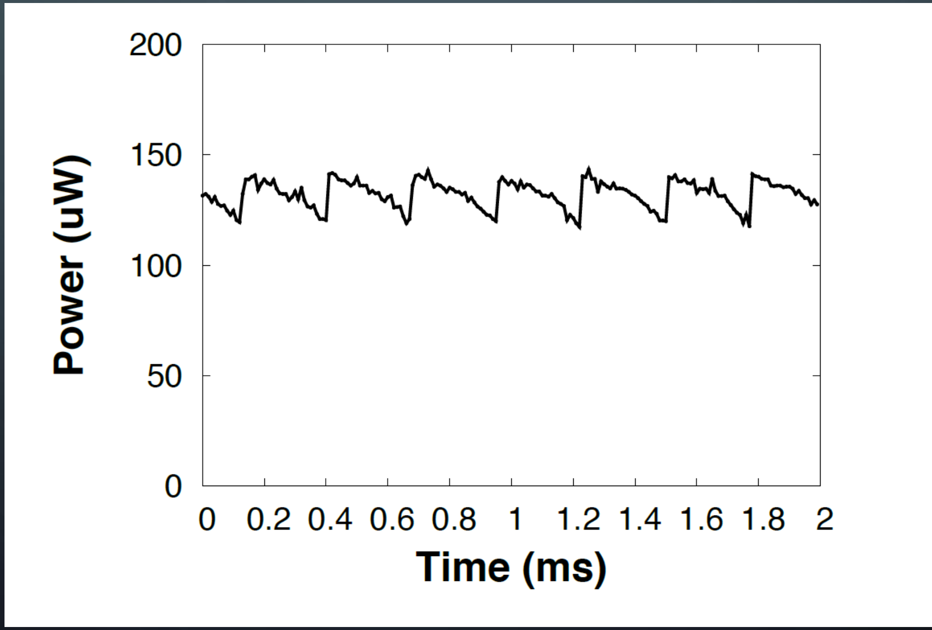


Data Handling Subsystem

- Processes the acquired sensor data
- Formats and packetize it
- Sends the data to the network stack

Some Optimization

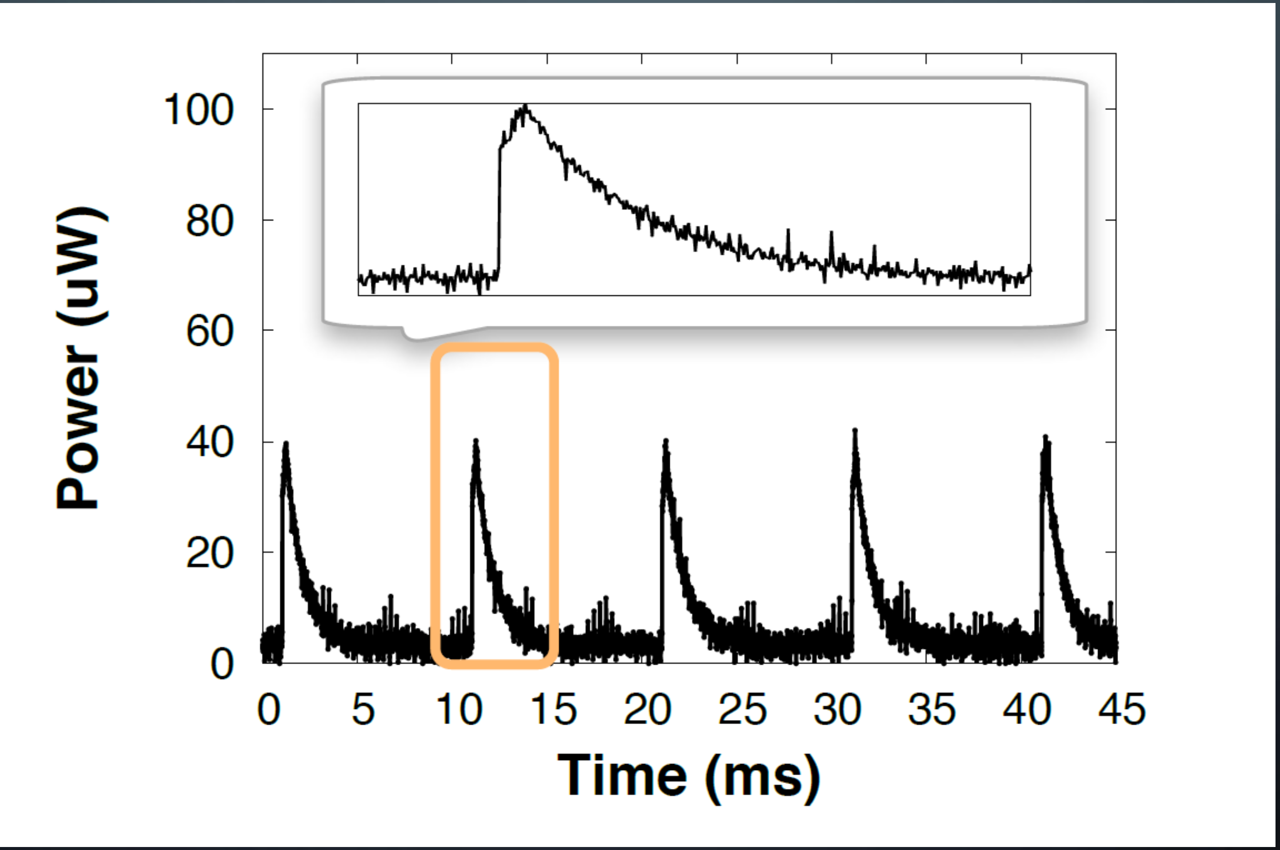
- Duty-cycled mode
- But fails when the rate is high



Direct Memory Access (DMA)

- Transfer data directly to memory without waking up the MCU
- What’s the Reality?
- Works at low rate
- Power consumption high with high rate

DMA cont'd



Communication Subsystem

- Includes different layers
- MCU needs to be on for processing messages if use software
- Hardware better?
- UART buffer needs to be filled with sensor data, wake up MCU or through DMA
- All in all, consumes a lot of power.

Transmission Efficiency

- Low clock utilization
  - Software implemented
  - EPC Gen 2 PHY-layer encoding
  - EPC Gen 2 MAC layer not designed for high bandwidth data transfer

Summary

- Many operation involves MCU
- Hardware implementations (DMA, UART) does not solve the problem
- Inefficient utilization of the clock cycle reduce the throughput

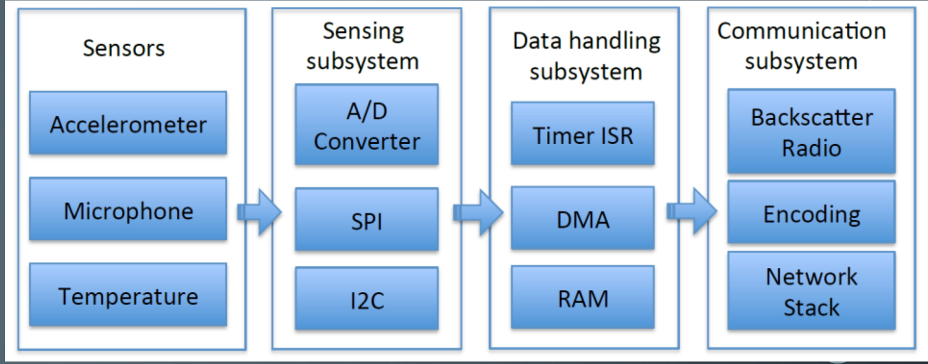
What To Do?

- Clean-slate redesign of a backscatter-based sensor platform

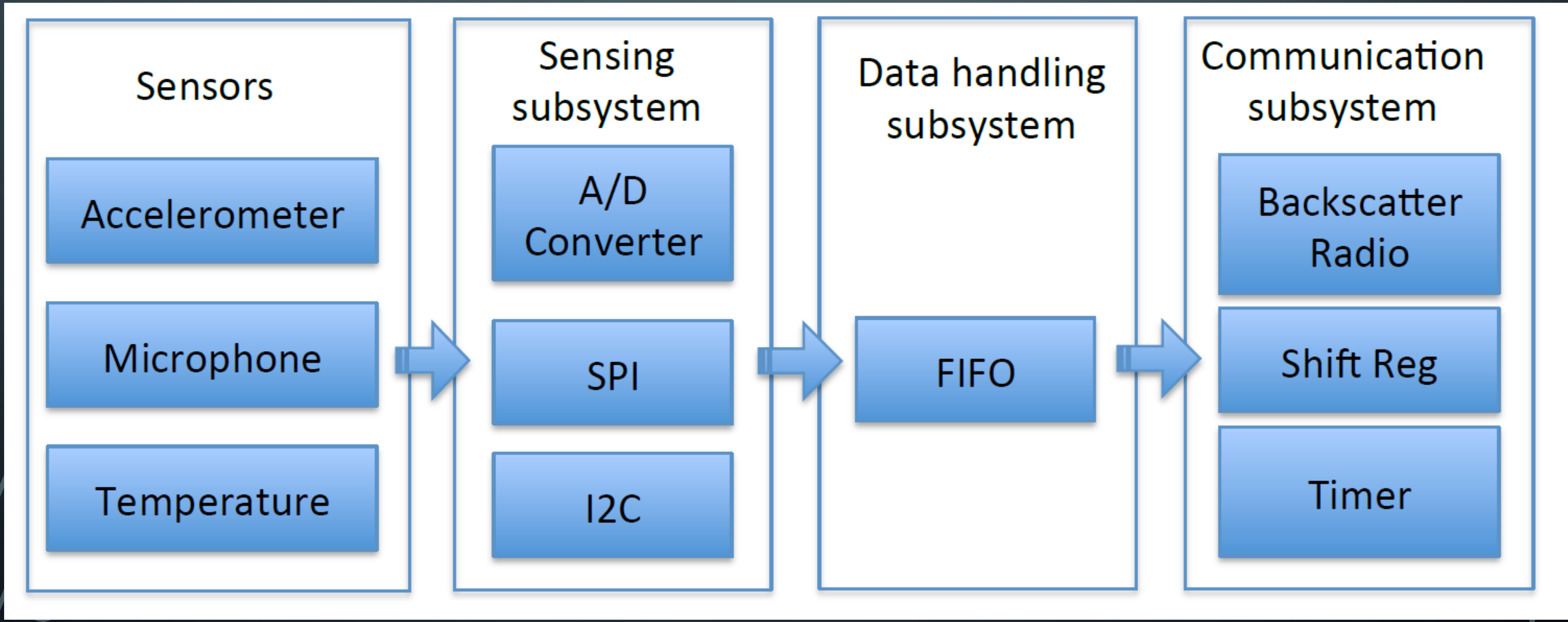
Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Ekho Platform



- Minimalist design



No Computation!

- Use a FIFO buffer
- Why need the buffer?
- Deal with short delay
- Side benefit: No software

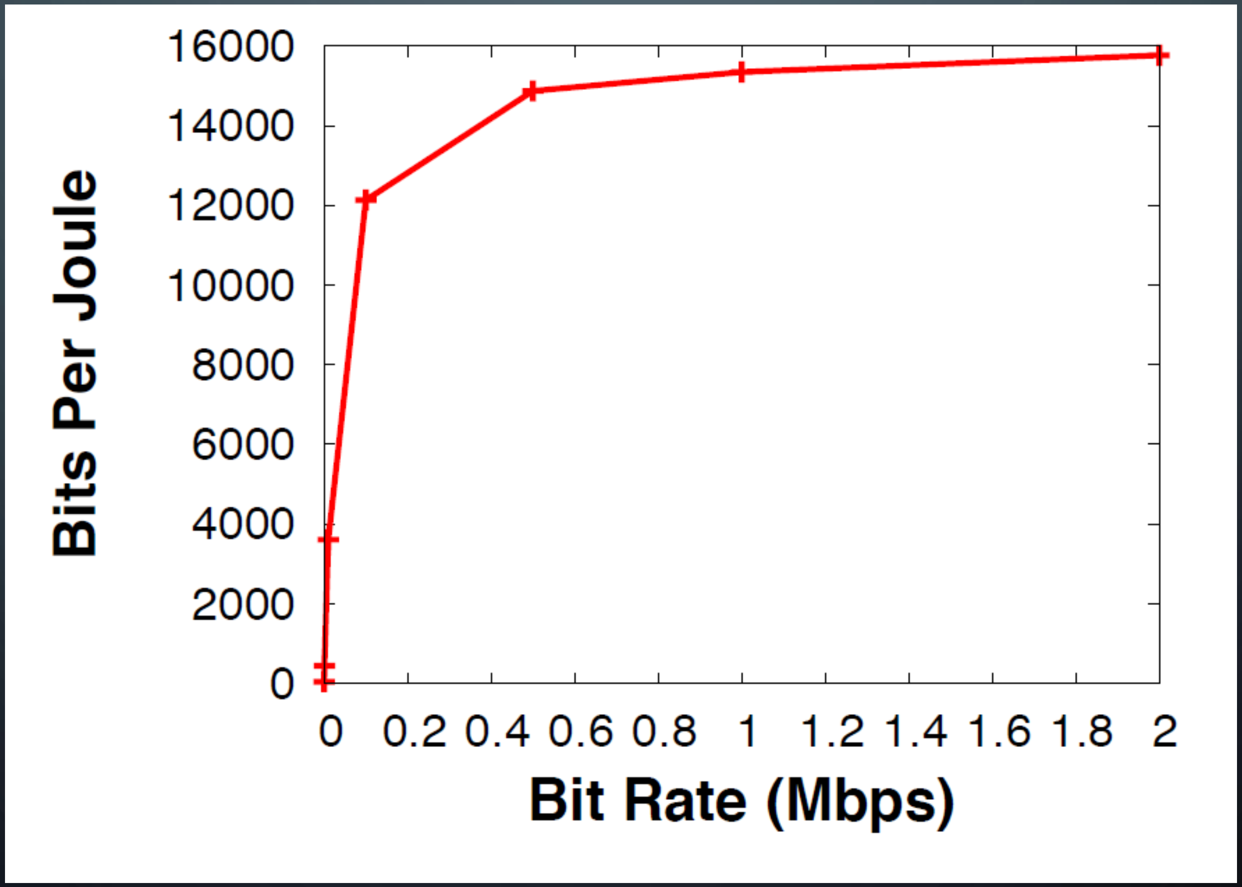
Simplified Communication

- Designed for sensor data only
- Reader informs each node of a timer, period, and a rate for transfer
- Only contains a timer and shift register
- No encoding

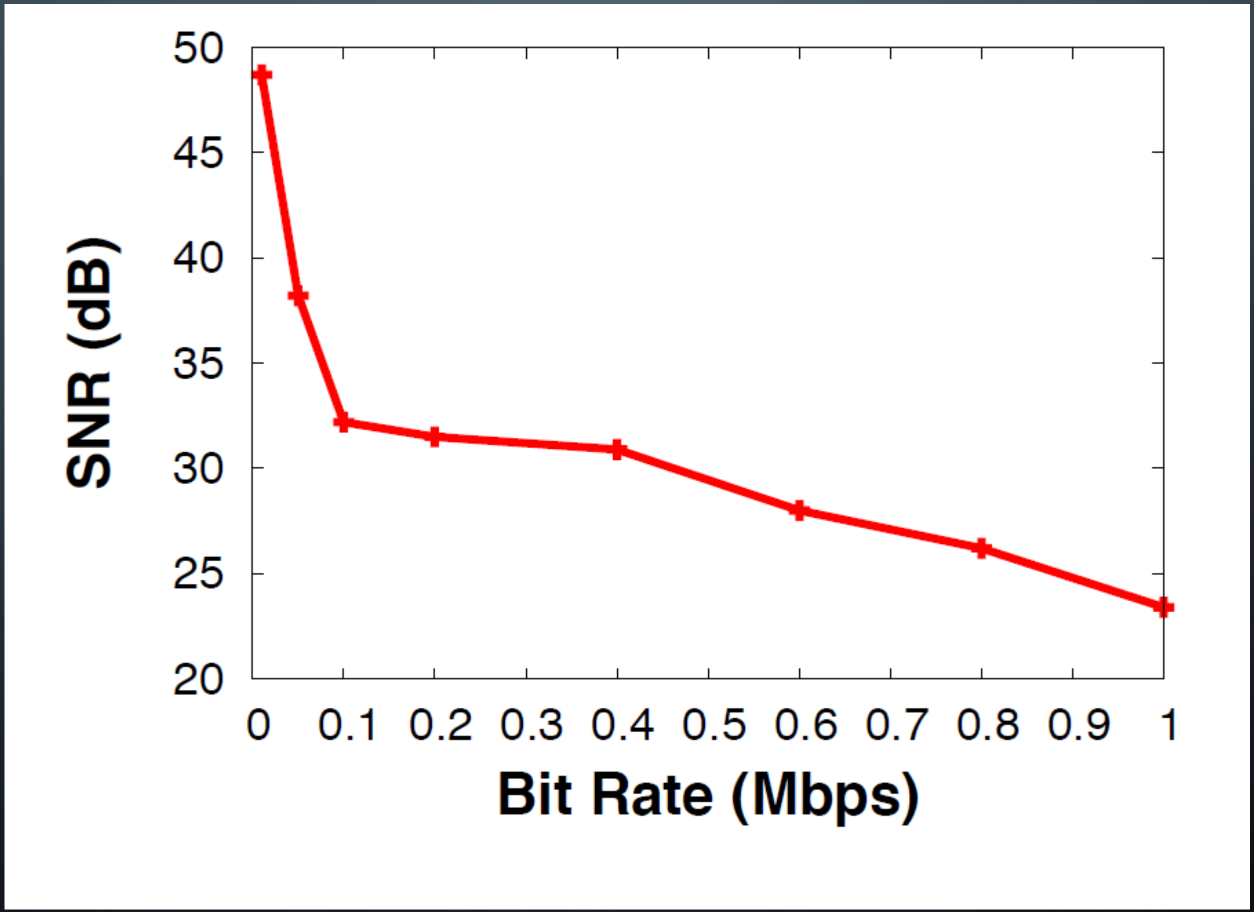
MAC Layer

- A high speed MAC layer for raw data transfer
- Design considerations?
  - Bits/Joule
  - Signal to Noise Ratio
  - Utility of data
  - Clock Drift
  - Buffer size

Efficiency of Backscatter

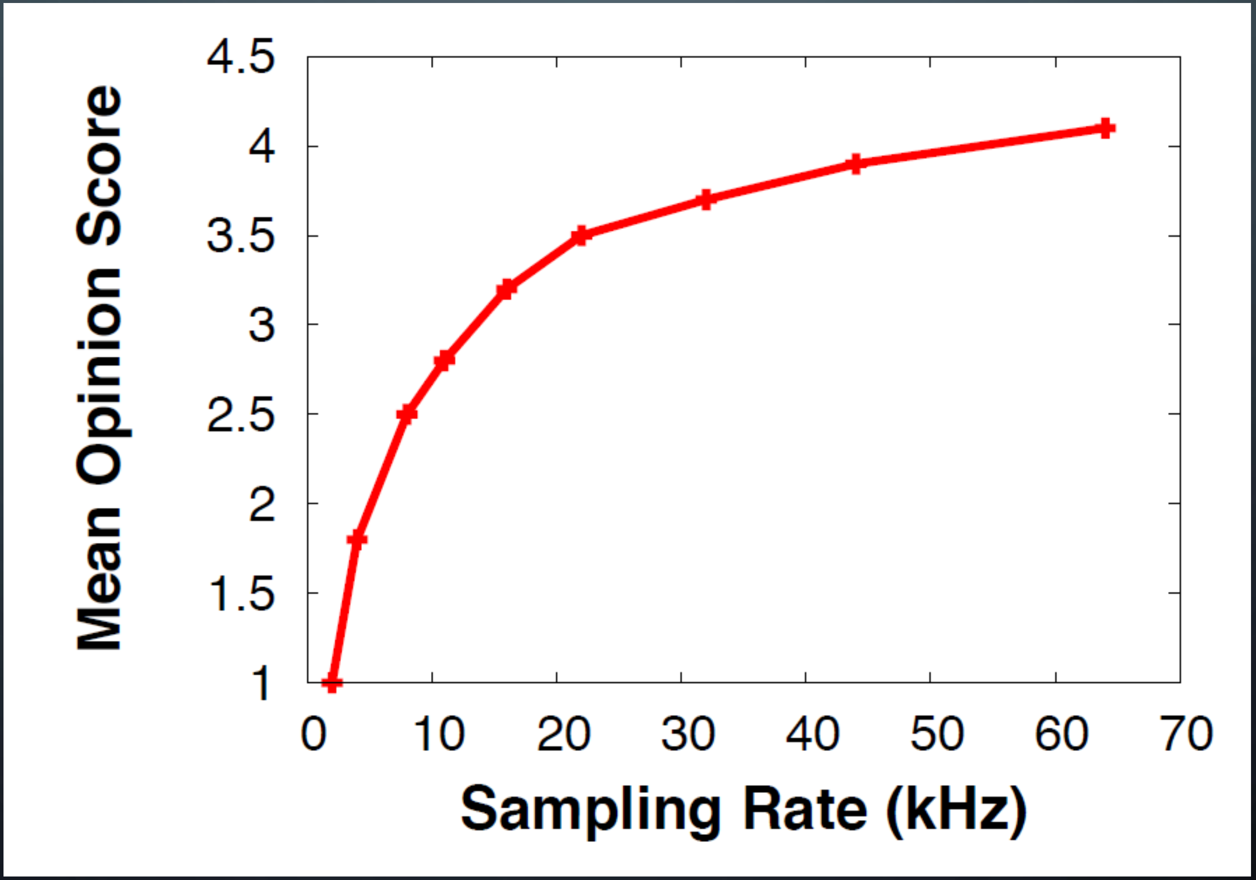


SNR



SNR (dB)
50
45
40
35
30
25
20
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Bit Rate (Mbps)

Mean Opinion Score (MOS)



Reader Design

- Select the optimal bit rate and slot size such that aggregate utility of received data is maximized
- Also aggregate energy consumption is minimized
- Subject to constraints on the buffer sizes, SNR, and guard bands

The Equation

$$
\begin{aligned}
\underset{\mathbf{s},\mathbf{t}}{\text{maximize}} \quad & \mathbf{1}^{\mathbf{T}} U(\mathbf{s}) \\
\text{subject to} \quad & \mathbf{t}^{\mathbf{T}} \mathbf{1} \leq 1 \\
& \mathbf{s} \preceq s_{max} \mathbf{1} \\
& (1 - \delta) \operatorname{diag}(\mathbf{t}) \mathbf{r} = b\, \mathbf{s}
\end{aligned}
$$

Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Implementation

- Hardware
- Software Defined Backscatter Reader
- MAC Layer Protocol

FPGA

- Sensing, data handling and communication subsystems
- Maximum size of the FIFO determined
- How big is the buffer?
- 32K bits (2KB)

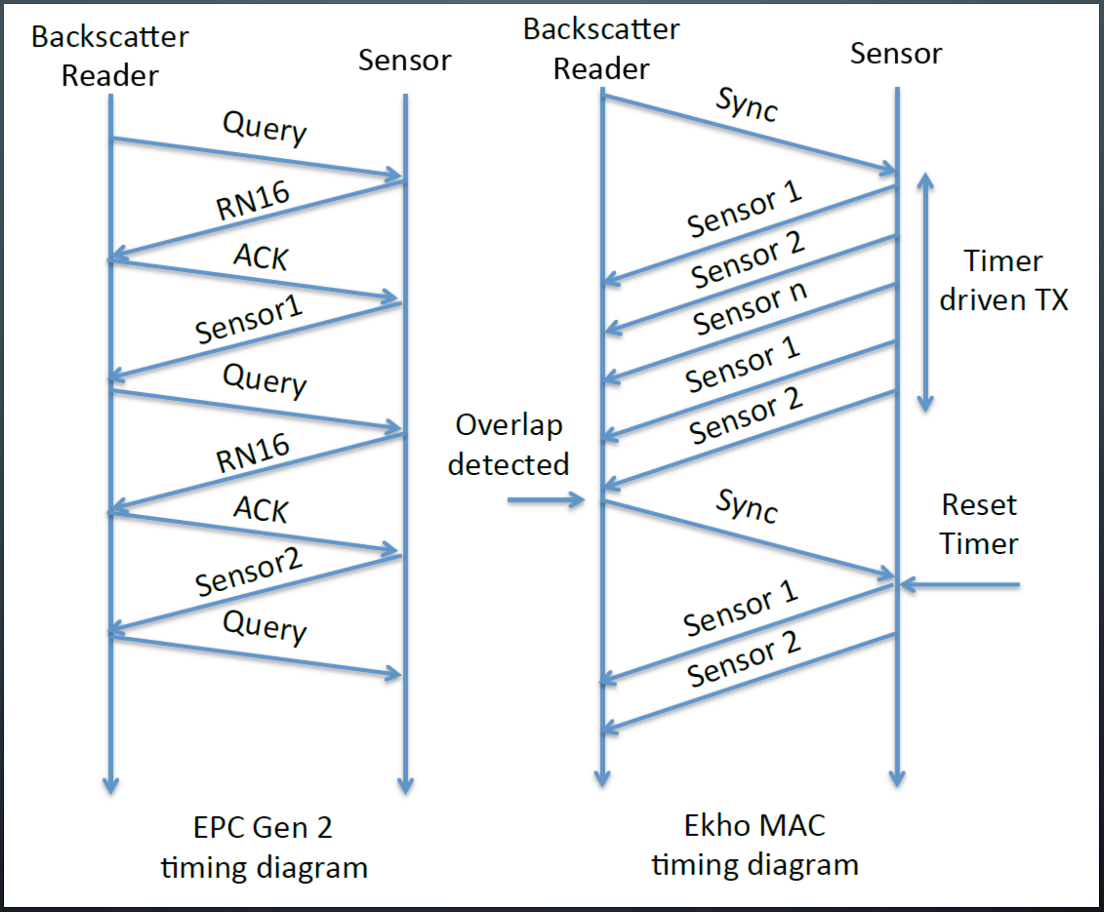
Backscatter

- Existing systems unstable
- How they solved it?
- Use a small bias current for shaper edge

Reader

- Directly sent the data via OOK and no encoding
- Track the amplitude of the signal

MAC Layer



Backscatter Reader
Sensor
Query
RN16
ACK
Sensor1
Query
RN16
ACK
Sensor2
Query
EPC Gen 2 timing diagram
Overlap detected
Backscatter Reader
Sensor
Sync
Sensor 1
Sensor 2
Sensor n
Sensor 1
Sensor 2
Timer driven TX
Sync
Reset Timer
Sensor 1
Sensor 2
Ekho MAC timing diagram

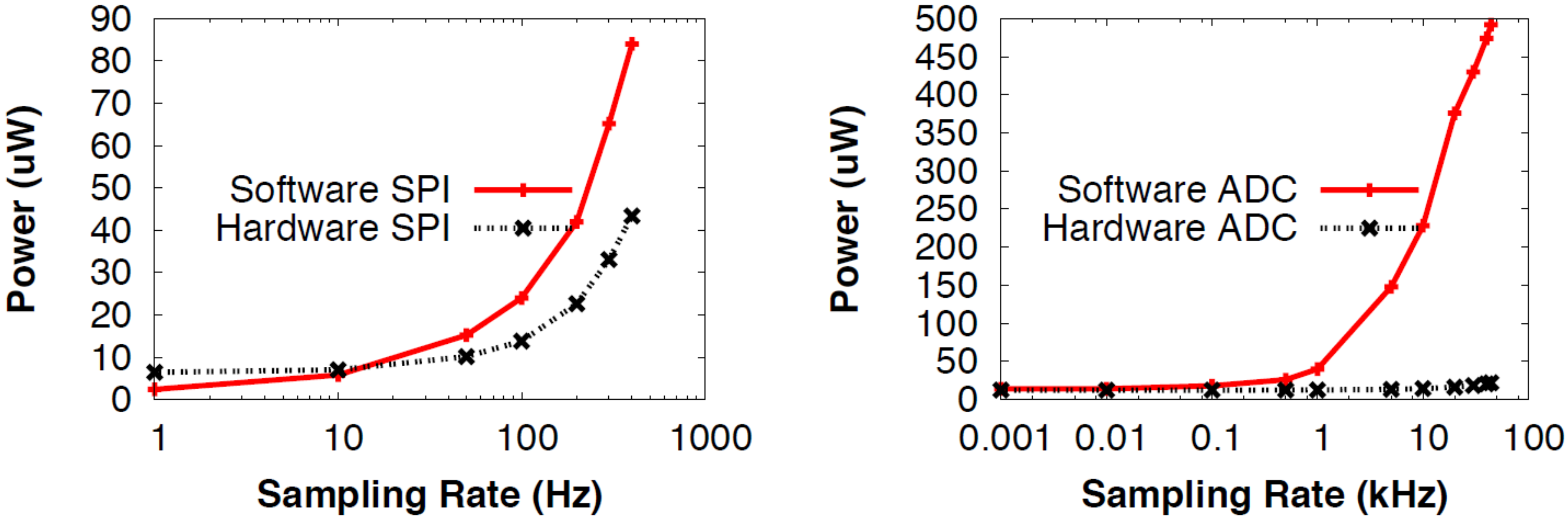
Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Evaluation

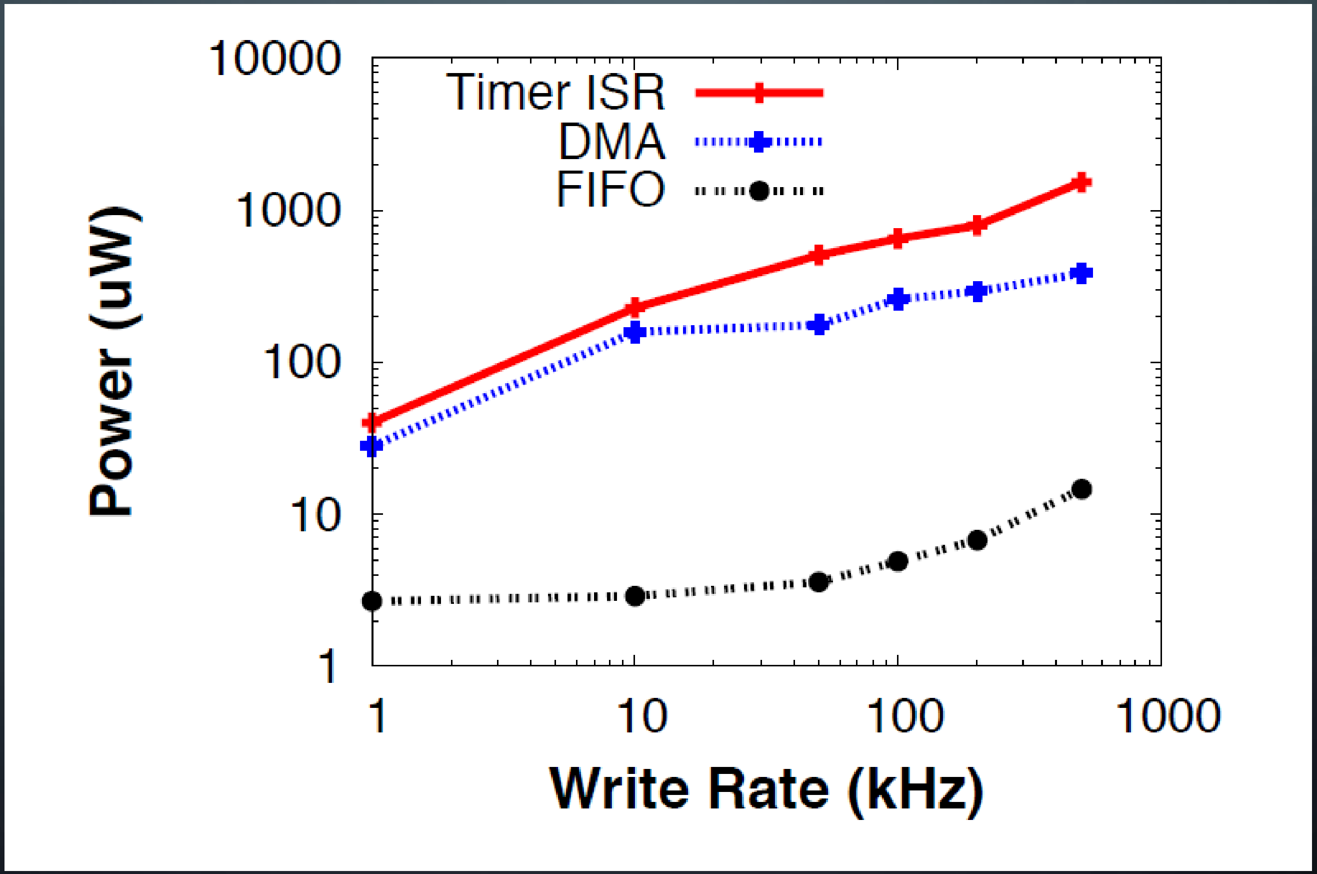
- Power Consumption for Each Subsystem
- Power Consumption for Whole System
- Throughput

Sensing Subsystems

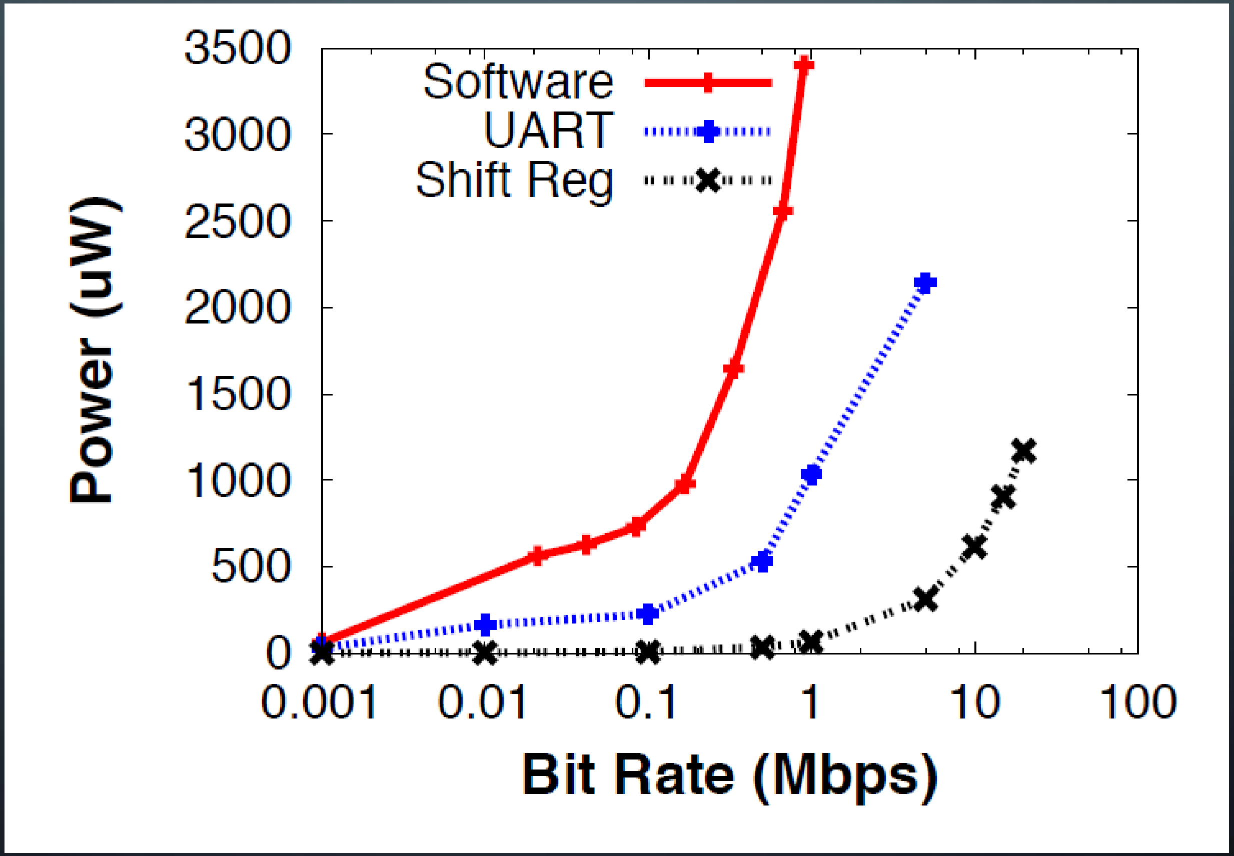


(a) Accelerometer. (b) Microphone.

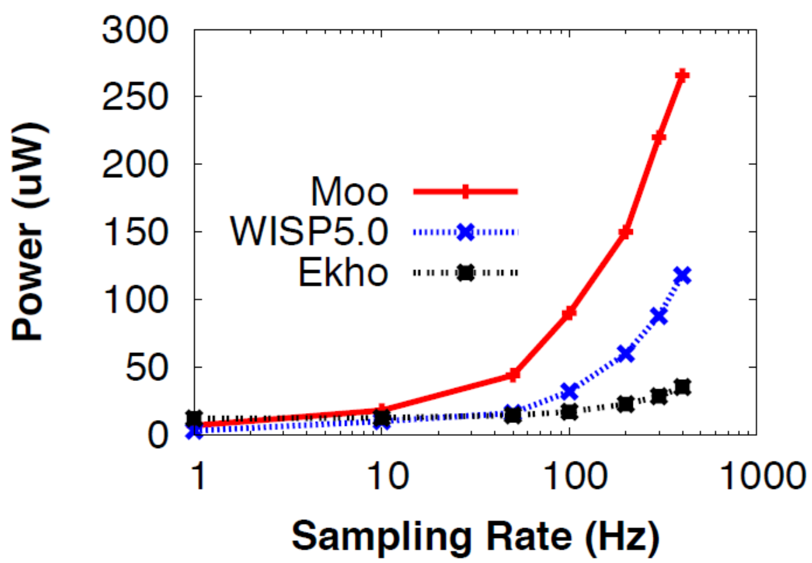
Data Handling Subsystem



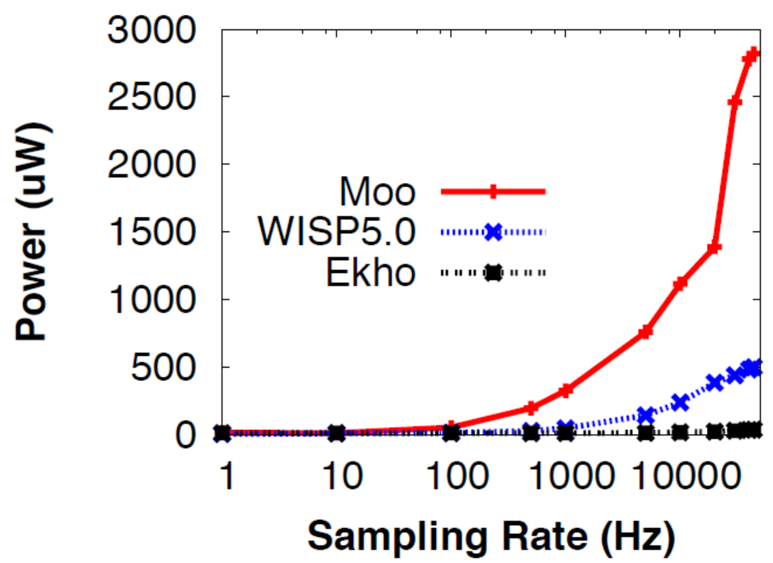
Communication Subsystem



Whole System

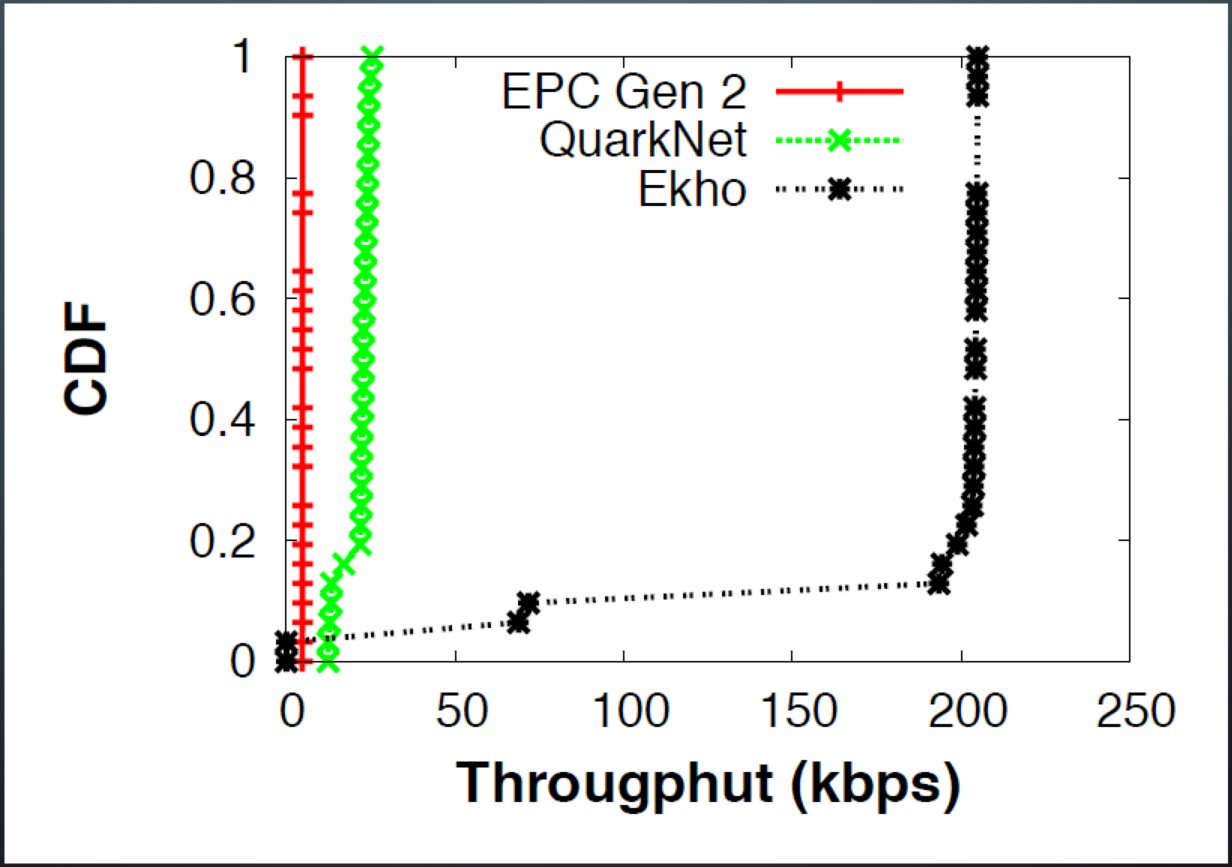


Accelerometer Sensor



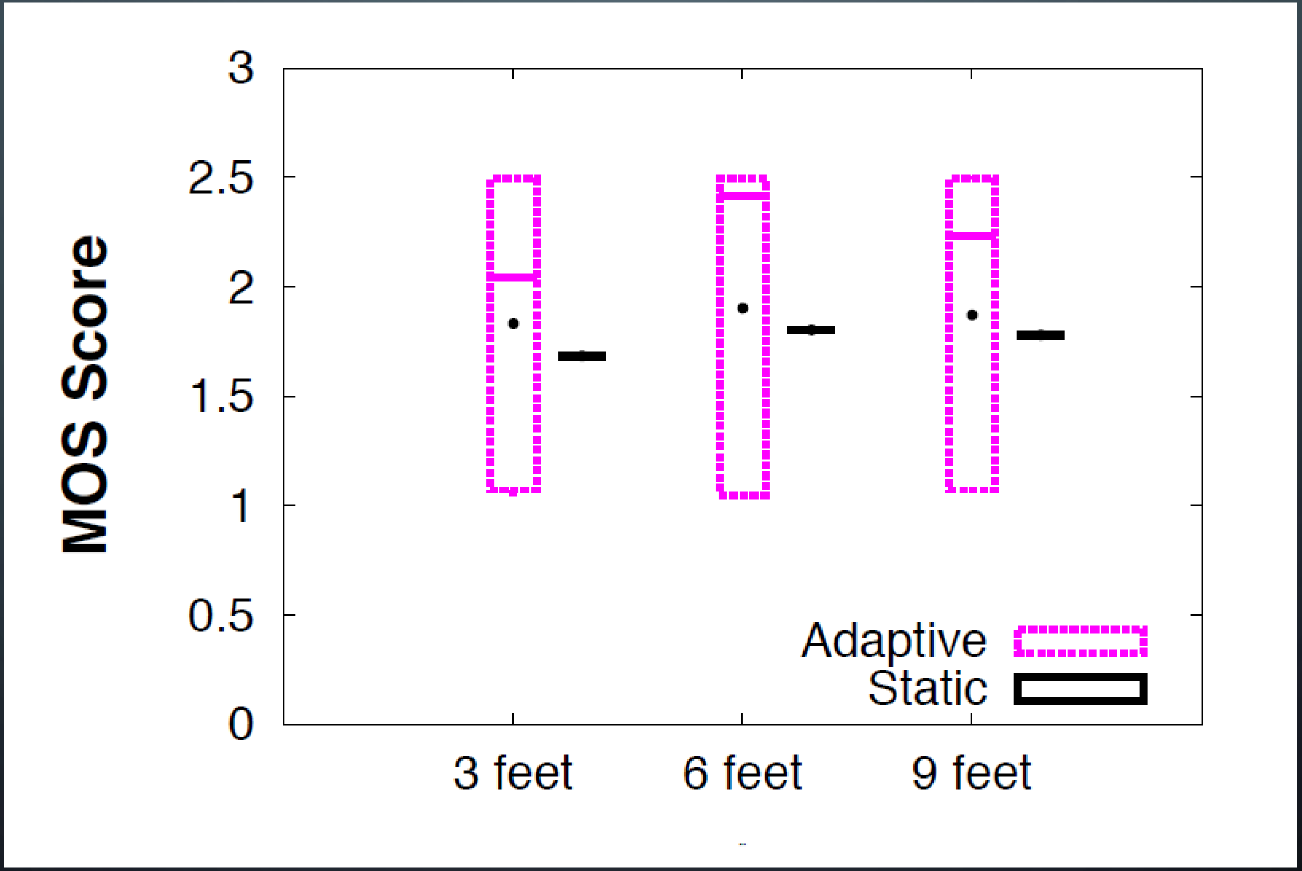
Audio Sensor

Throughput



EPC Gen 2
QuarkNet
Ekho
CDF
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
250
Througphut (kbps)

MOS Score

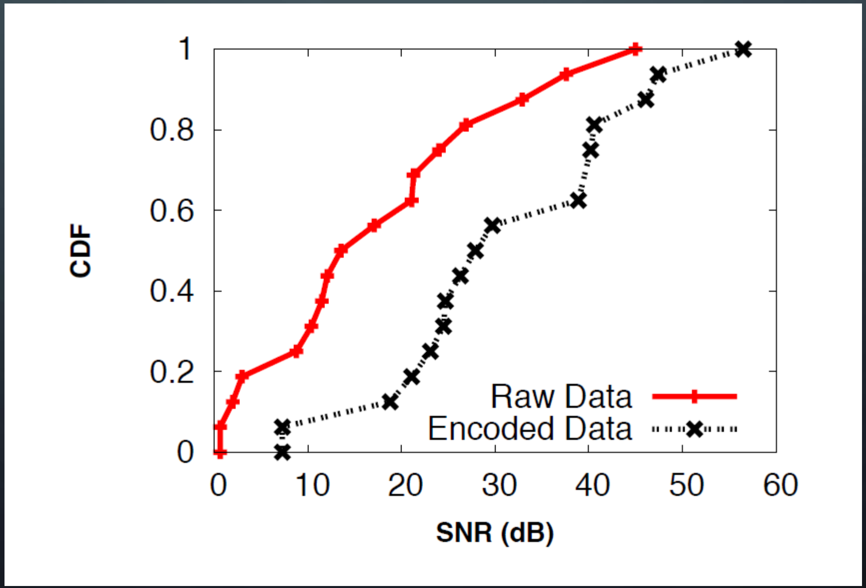


Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Discussion

- FPGA is harder to work with
- Less power consumption for Ekho
- No encoding hurts



Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Conclusion

- Enabler for new applications
- Low power consumption

Outline

- Background
- Problem
- Existing Systems
- Methods
- Implementation
- Evaluation
- Discussion
- Conclusion
- Personal Opinion

Personal Opinion

- Pros:
  - Energy Efficient
  - High throughput
  - Cheaper for sensor
- Cons:
  - No Encoding
  - Firmware Update?
  - Limited Throughput (also quality)
  - Expensive reader
  - Range

Ekonet: High Speed Ultra Low-power Backscatter for Next Generation Sensors

COMP 790 Internet of Things

Shiwei Fang

PRACTICAL BLUETOOTH TRAFFIC SNIFFING:
SYSTEMS AND PRIVACY IMPLICATIONS

Wahhab Albazrqaoe[1,2], Jun Huang[1], Guoliang Xing[1]

[1] Department of Computer Science and Engineering, Michigan State University, USA

[2] University of Karbala, Karbala City, Iraq

Presented by Marc Eder

COMP 790: Internet of Things

September 30, 2016

Overview

- Introduction
  - Motivation
  - Challenges
  - Proposed Solutions
- Bluetooth
- The BlueEar
  - System Overview
  - Clock Acquisition
  - Subchannel Classification
  - Selective Jamming
- Implementation
- Performance
- Privacy Implications
- Discussion

INTRODUCTION

Motivation

- Bluetooth is increasingly popular platform for wireless communication
  - Particularly useful due to low power requirements and high bandwidth

- With increased popularity comes increased threat
  - How secure is Bluetooth communication really?
  - Complex encryption is ignored in favor of lower power draw
  - Standard E_0 encryption used between paired devices is susceptible to brute force attacks

- Nevertheless, implementation details of the communication scheme makes it difficult to passively intercept Bluetooth signals
  - Existing methods are expensive and actively pair with a transmitter

- Can we cheaply and passively eavesdrop (sniff) a Bluetooth signal?

Challenges

- Bluetooth's rapid channel switching makes it difficult to continuously monitor packet streams
- Additional adaptive channel hopping (due to bad channels) adds a layer of complexity to the existing switching patterns
- Without pairing to a transmitter, sniffing devices are highly susceptible to channel interference which reduces ability to intercept packets

Proposed Solutions

- Pattern matching
  - Sit on a channel and watch the pattern of transmitted packets
  - Match the packet pattern as a sliding window across all hopping phases

- Probabilistic matching
  - To catch adaptive hops, choose a pattern with ≥95% confidence

- Subchannel classification
  - Classify subchannels as good or bad to determine whether the transmitter will hop to them
  - Selectively jam the bad channels to force the transmitter elsewhere

BLUETOOTH

How does Bluetooth work?

- Generally, Bluetooth devices transmit signals in the 2.402-2.480 GHz spectrum
- This *channel* is further broken into 79 1MHz *subchannels* (further specifying signal paths)
- The transmitter switches transmission *subchannels* every 625µs (1,600 hops / second)
- **Why perform the channel switching?**

How does Bluetooth work?

- Bluetooth communication performed by *pairing* multiple Bluetooth capable devices
  - *Master-slave* dynamic

- Resulting network called a *piconet* and specified by unique *address*

- Channel *hopping* is dictated as a function of the piconet address and its clock

$$Channel\ index = H(A, c)$$

How does Bluetooth work?

- Bluetooth Classic uses 27-bit clock → 2^{27} selection states, or *phases*
- *Basic hopping sequence* $\{i_1 \ldots i_{2^{27}-1}\}$ determined by hopping function, and *phase* is the current index of the sequence
- **Is this a random pattern?**

How does Bluetooth work?

- The world is a noisy place – most devices use *adaptive channel hopping* to avoid bad channels
- Channel quality classified by a *subchannel map*
  - Bad ones get skipped if necessary
- *Remap* function (dependent on address and phase) determines where to go
- Device-dependent implementation
- *Indiscoverable mode* hides piconet address, clock, and subchannel map from unpaired devices

THE BLUEEAR

System Overview

- Employ 2 Bluetooth radios, *scout* and *snooper*
  - *Scout* surveys channel conditions
  - *Snooper* tracks the target device

- Simple 3 step process to sniff packets
  1. Filter packets corresponding to target device
  2. Match clocks
  3. Manipulate subchannel hopping

- Performing the clock matching is where it gets tricky

Clock Acquisition

- Without adaptive hopping, matching clocks is fairly trivial

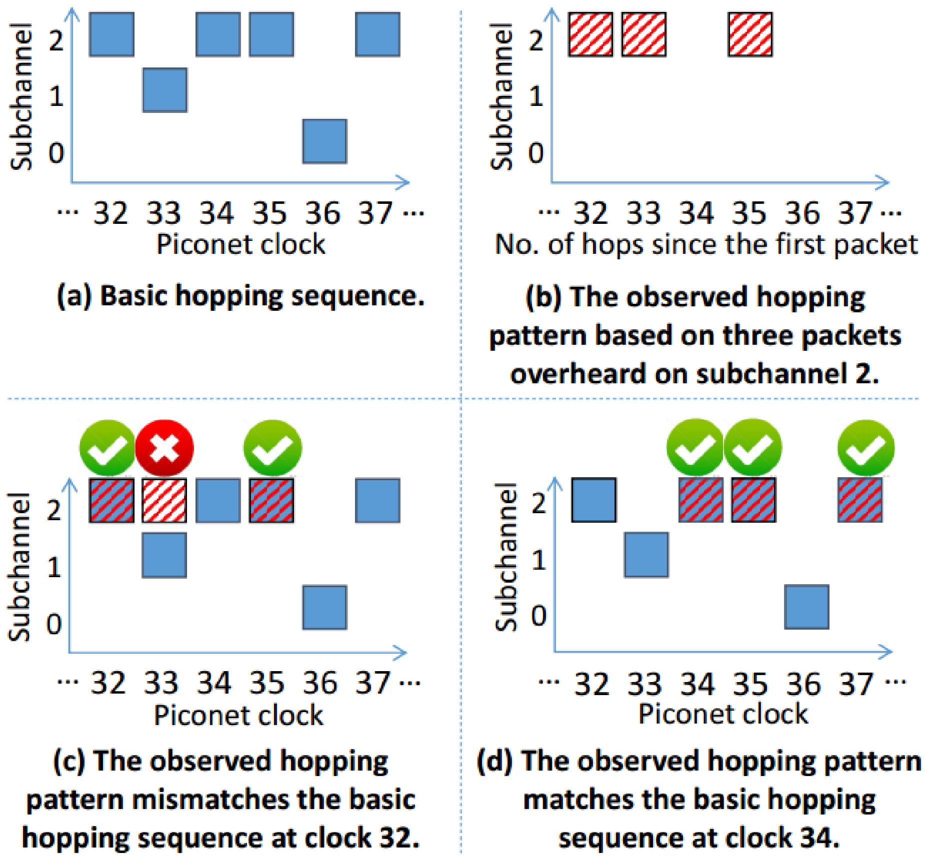
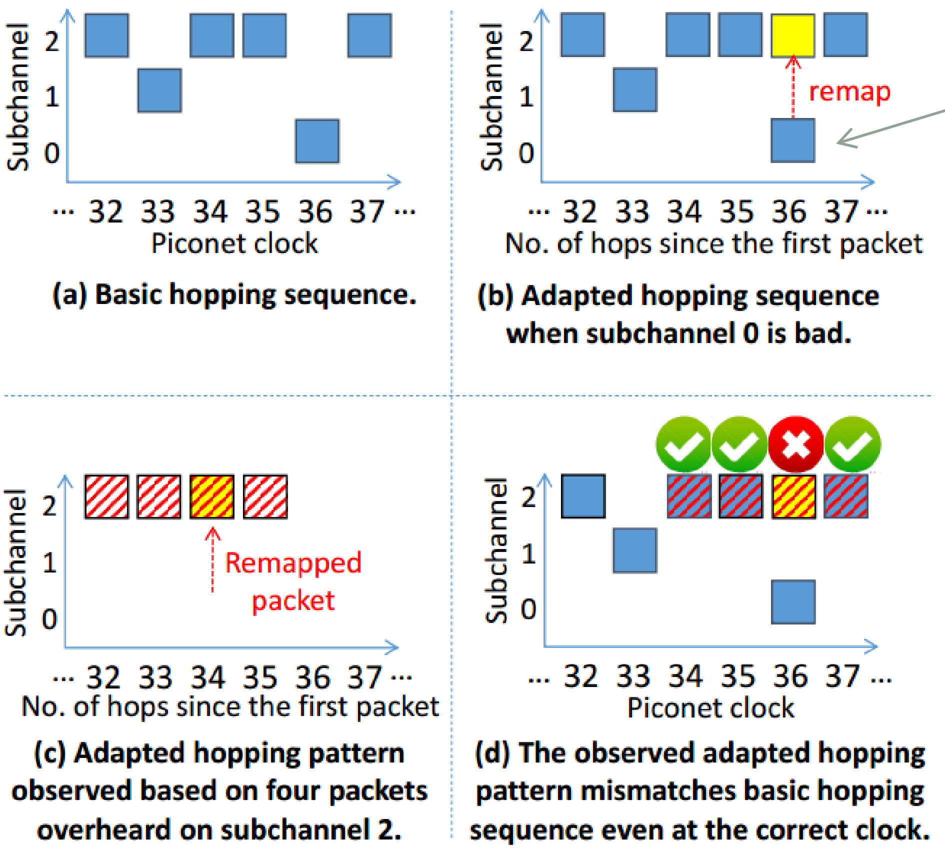


Figure from paper

Clock Acquisition

- Adaptive hopping makes the problem trickier



This packet would have normally been transmitted at channel 0, but was instead remapped to channel 2, throwing off our pattern matching on channel 2

Figure from paper

Clock Acquisition

- **What was the key observation the authors made that led to a probabilistic solution?**

- Solution lies in observation that ratio of errors in pattern should equal ratio of remapped subchannels
  - Bluetooth standard requires ≥20 subchannels for frequency hopping
  - Remapped subchannels ratio upper bounded by $\frac{59}{79}$
  - If $\frac{errors}{packets} > \frac{59}{79}$, then the candidate clock is likely incorrect

- Using the Central-Limit Theory, clock can be determined with ≥95% accuracy as

sample standard deviation

number of mismatches for clock candidate c

$$\frac{d_c}{n} - 2\frac{\sigma}{\sqrt{n}} \geq \frac{59}{79}$$

number of samples

Subchannel Classification

- Once clock is acquired, all packets *theoretically* should be able to be intercepted
- In practice, noise and interference in the channel can lead to missed packets
- BlueEar aims to identify which subchannels are good for listening in on

Subchannel Classification

- 3 types of subchannel classifiers evaluated

1. Packet-rate-based classifier
    - Determine good/bad based on rate of all packets from target device through the channel
    - **Pros?**
    - **Cons?**
2. Spectrum-sensing-based classifier
    - Determine good/bad based on interference statistics of the channel
    - **Pros?**
    - **Cons?**
3. Hybrid classifier
    - Use both metrics to determine if a channel is good or bad

Subchannel Classification

- Hybrid classifier ultimately implemented in BlueEar
- Trains an SVM to classify subchannel quality

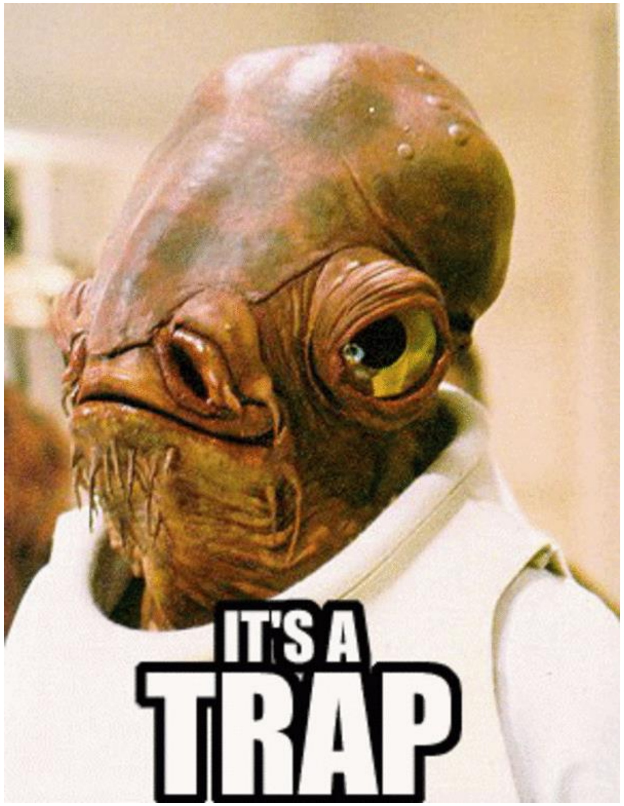
  Training GT: Packet-based classifier output

  Training Observations: Interference conditions of channel *i* as sampled by the scout

- Claims to learn the device's classification model
- Outputs a log-likelihood confidence score $\lambda_i = log\frac{\rho}{1-\rho_i}$
  - **Where is ρ_i, the probability that channel *i* is good, coming from?**
- **Is this a valid approach?**

Selective Jamming

- BlueEar also manipulates the target device to go to channels conducive for eavesdropping
- Adds extra interference to subchannels classified as bad
- Adaptive hopping scheme skips bad subchannels and moves to a better subchannel for sniffing



IMPLEMENTATION

Hardware

- 2 Ubertooths
  - Hop selection

- Linux laptop
  - Clock acquisition
  - Subchannel classification

Ubertooth Firmware

- Each hardware component's clock can skew, leading to drift error
  - Scout and snooper tick 1 μs before target to avoid missing packets
- Snooper implements hop selection kernel
- Task scheduling is priority based
  - Hop selection and subchannel switching are most important

PERFORMANCE

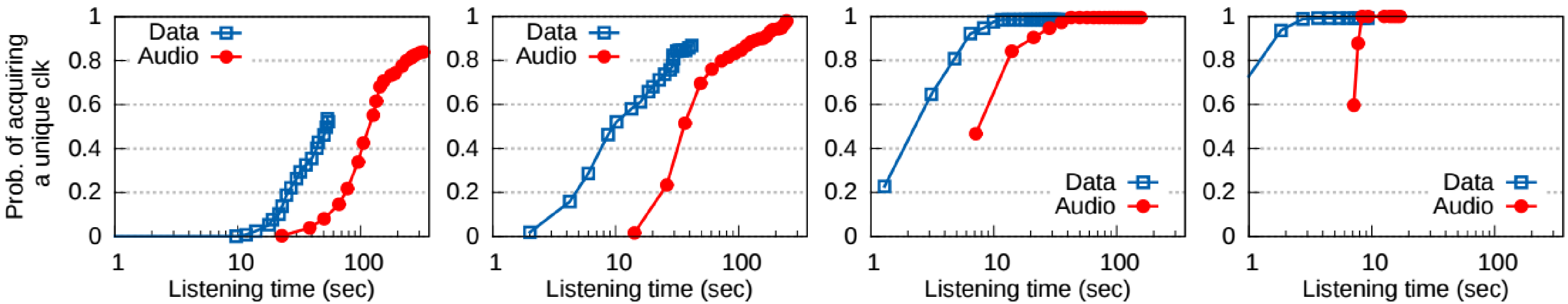
Evaluations

- Tested on data and audio traffic
  - Audio → playing an audio file transmitting to Bluetooth headset
  - Data → Data file trasmission via Broadcom dongle
  - Tested in office setting nearby 802.11 WLAN access points

- Check
  - Synchronization delay
  - Subchannel classification accuracy
  - Packet capture rate
  - Subchannel classification and packet capture rates in interference conditions
  - Subchannel classification and packet capture rates in crowded spectrum
  - Ambient interference conditions

Synchronization Delay

- 3 WLAN access points crowd channels



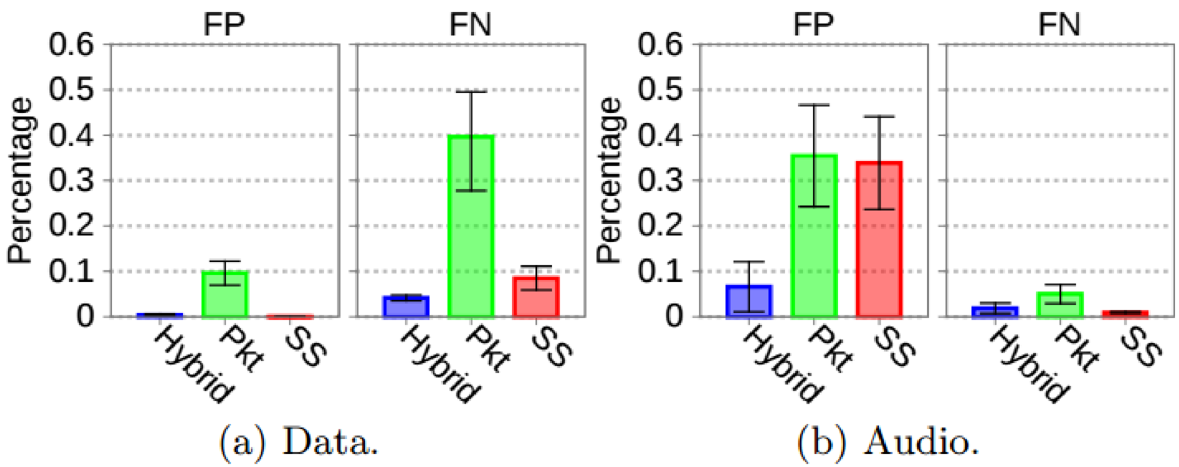
(a) No bad subchannels. (b) 25% bad subchannels. (c) 50% bad subchannels. (d) 75% bad subchannels.

- Audio sniffing has longer clock acquisition delay than for data – lower packet rate for audio
- More interference, faster clock acquisition → **Why?**

Figure from paper

Fast-Varying Spectrum Context

- When subchannel map is modified frequently (i.e. interference conditions change a lot)



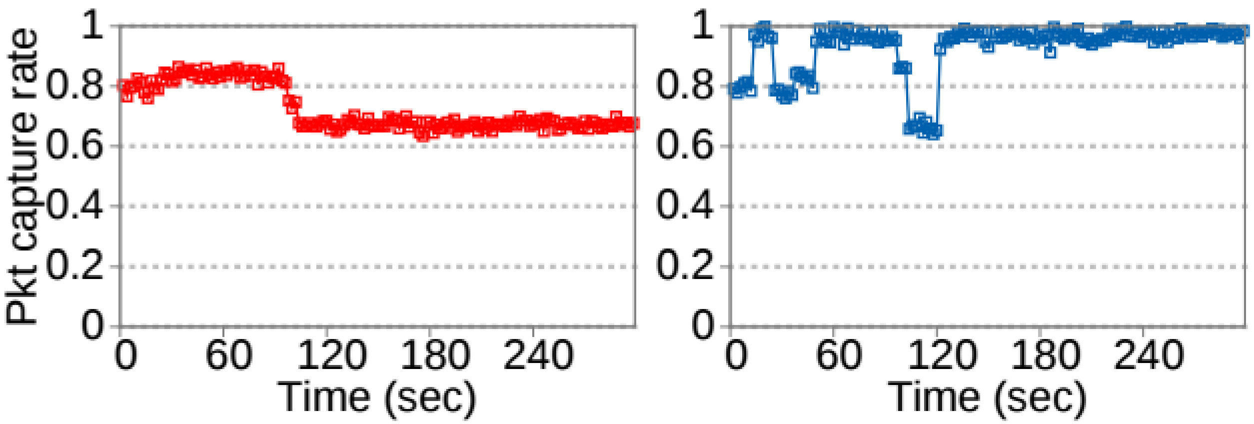
(a) Data. (b) Audio.

- Packet rate performs poorly → **Why?**
- **Why does spectrum-sensing approach fare worse with audio?**

Figure from paper

Packet Capture Rate

- Evaluates selective jamming usefulness



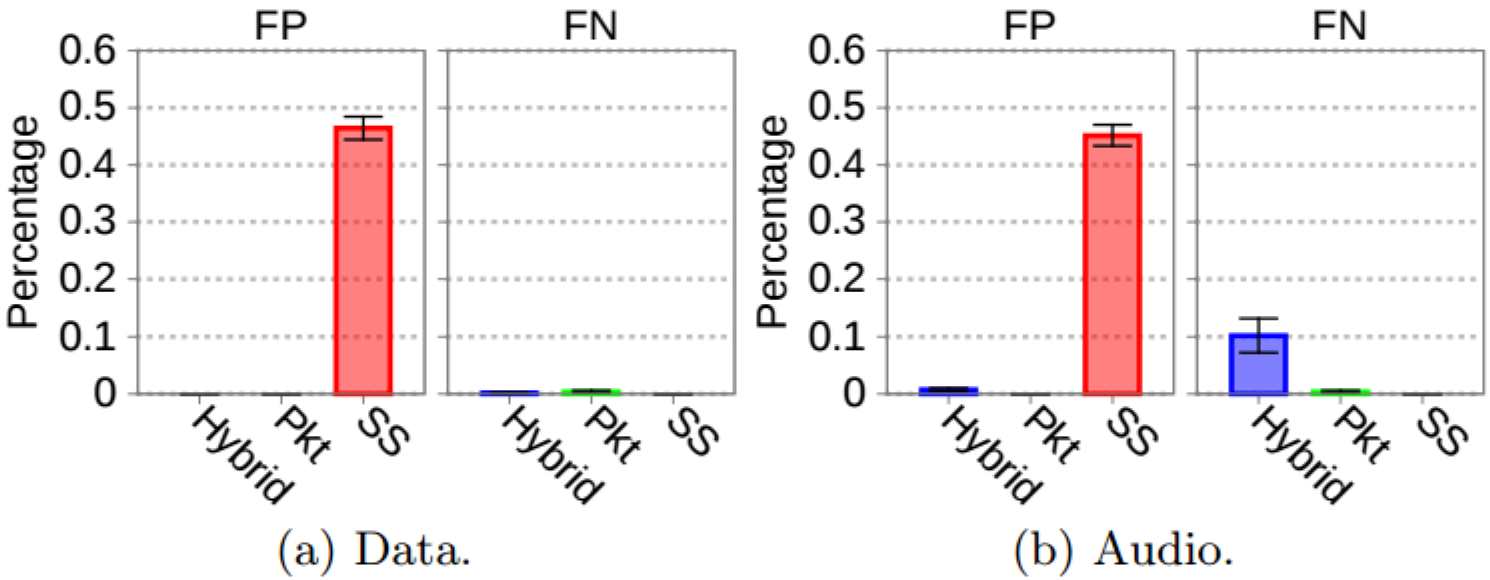
(a) Selective jam disabled. (b) Selective jam enabled.

- **Why does selective jamming improve capture rate?**

Figure from paper

Interference Conditions

- Interference close to target, not much near scout (remember interference is spatially-dependent)

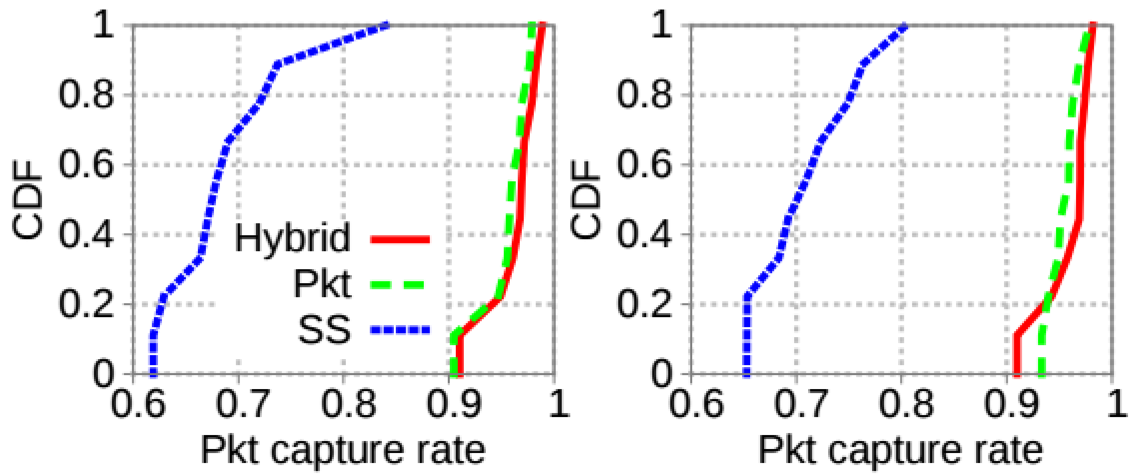


- **Why such terrible performance from spectrum-sensing approach?**

Figure from paper

Interference Conditions

- Interference close to target, not much near scout (remember interference is spatially-dependent)



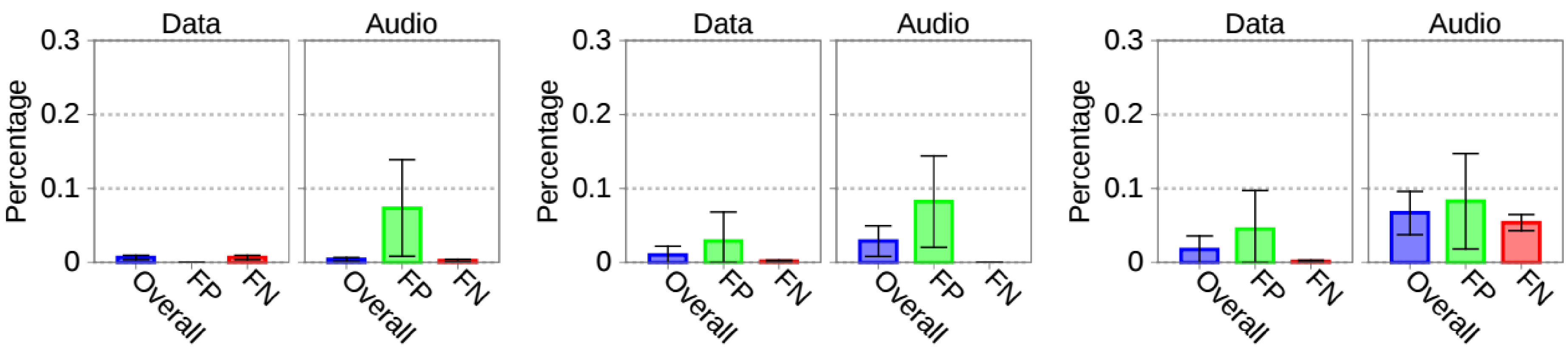
(a) Data. (b) Audio.

- Packet capture rate still very good for hybrid and packet-rate methods

Figure from paper

Crowded Spectrum

- High channel use by 802.11 WLAN networks (causing “bad” subchannels)



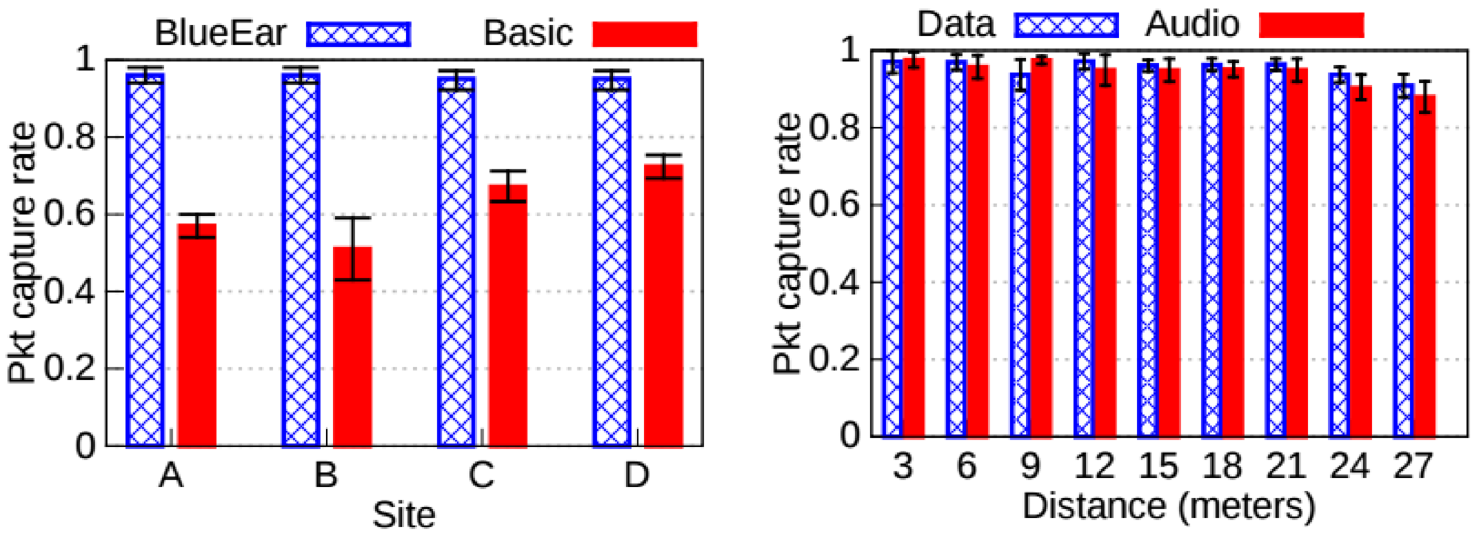
(a) No bad subchannels. (b) 25% basd subchannels. (c) 50% bad subchannels.

- Still very good performance (low FP and FN rates)

Figure from paper

Ambient Interference

- Packet capture rate at varied locations in environment with ambient interfering sources



- Left image compares BlueEar to basic Ubertooth sniffer at different locations
- Right image shows that packet capture rate stays high even at long distances from target

Figure from paper

PRIVACY IMPLICATIONS

Practical Evaluation

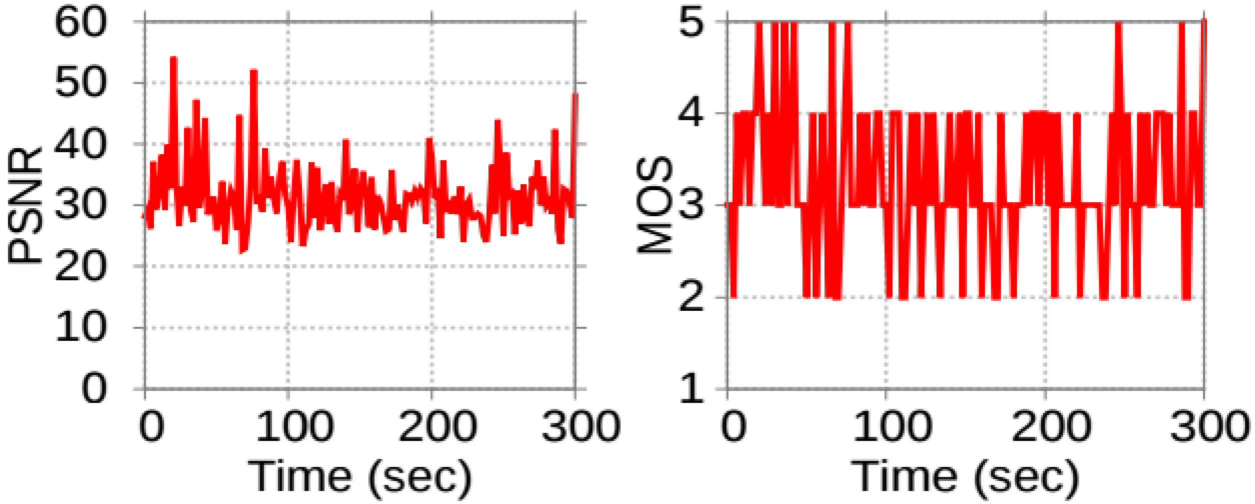
- Attempt to eavesdrop on speech conversation
  - Tricky because audio streams highly susceptible to packet loss

- Experiment:
  1. Collect real packet loss rates
     - Remove lost packets from test audio stream
  2. Establish piconet for speech streaming
  3. Deploy BlueEar
  4. Log all missed packets

- Evaluate using PSNR for stream quality

Results

- PSNR maps to Mean Opinion Score (MOS) which described quality of signal



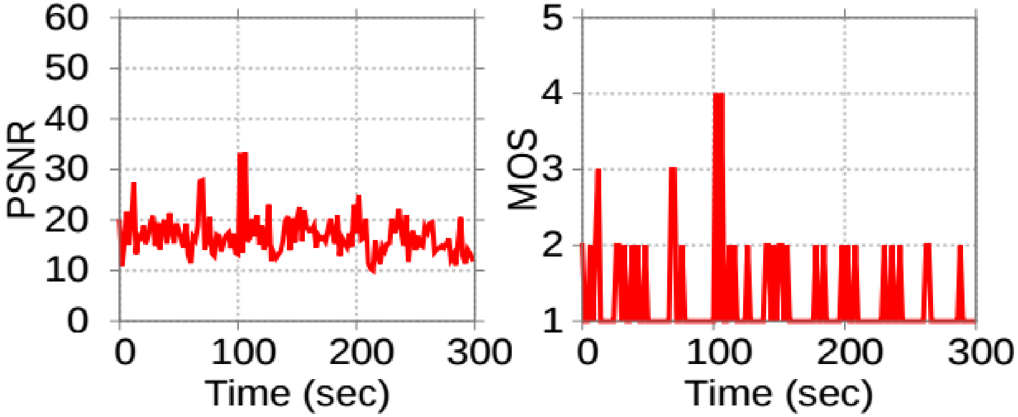
(a) PSNR. (b) Mean opinion score.

- 81% of sniffed stream scores higher than “fair” score

Figure from paper

Countermeasure

- These results are scary so far – say goodbye to privacy
- But wait! If we randomly mask the subchannel classifications by avoiding a “good” channel or knowingly hopping to a “bad” channel, we can break the learned adaptive hopping pattern.



(a) PSNR. (b) Mean opinion score.

- PSNR results degrade to “poor” in 95% of audio stream

Figure from paper

DISCUSSION

Discussion

- **Is the countermeasure practical? Is it easily circumvented as well?**
- **Are there any holes in the BlueEar device? Are there any practical situations that may arise that would render it useless?**
- **What sort of modifications to the Bluetooth system could prevent these attacks?**