Risks of Trusting the Physics of Sensors

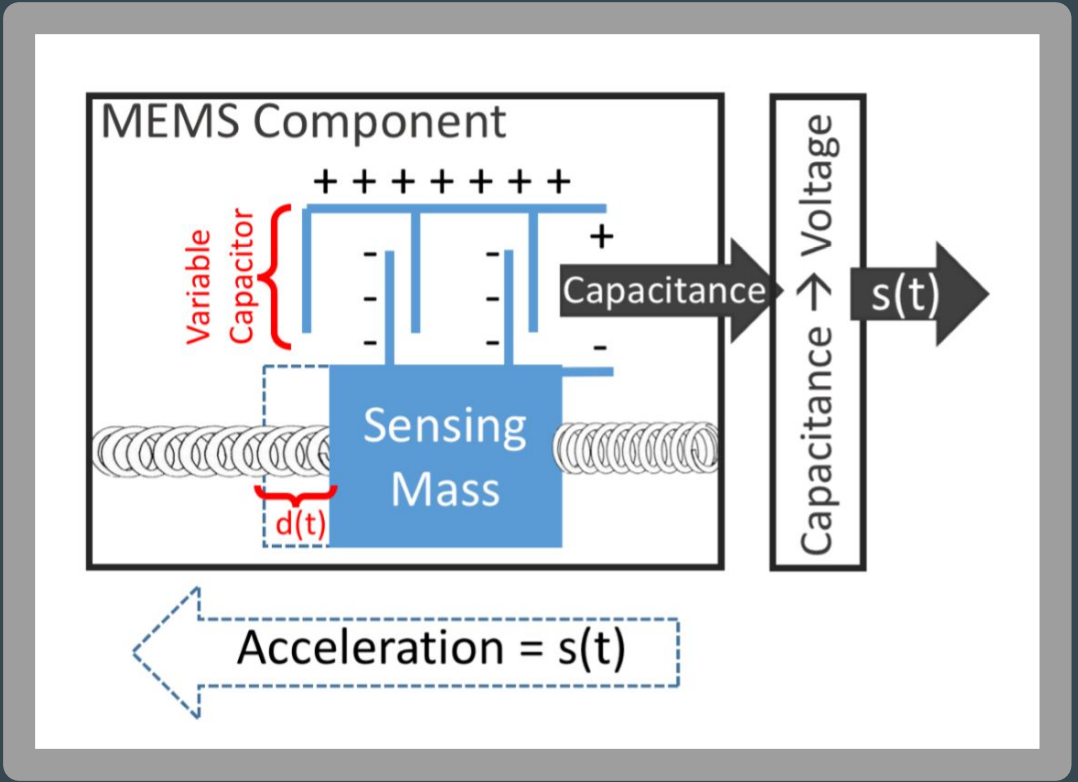
Kevin Fu and Wenyuan Xu

Presented by: Rick Sear

Introduction and Background

- IoT devices rely on sensors
- Sensors rely on physics
- Big surprise: this can be a problem

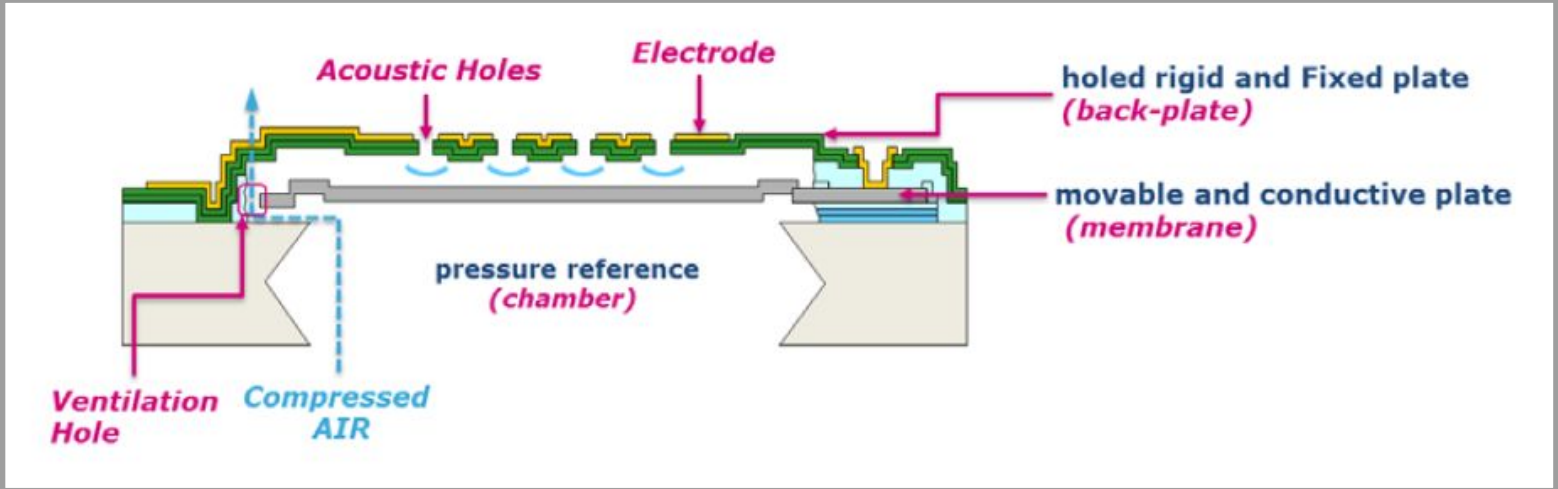
Your Favorite Sensors: How do they work?



MEMS Accelerometer

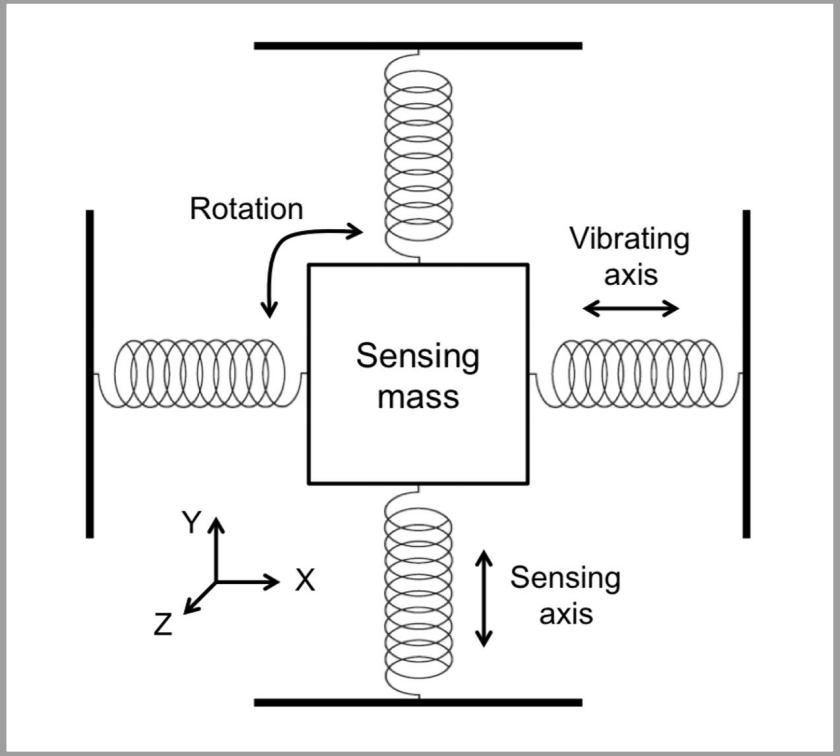
Your Favorite Sensors: How do they work?

MEMS Microphone



Audible sound: 20 Hz - 20 kHz
This device has a *fast* refresh rate

Your Favorite Sensors: How do they work?



MEMS Gyroscope

This Seems Exploitable

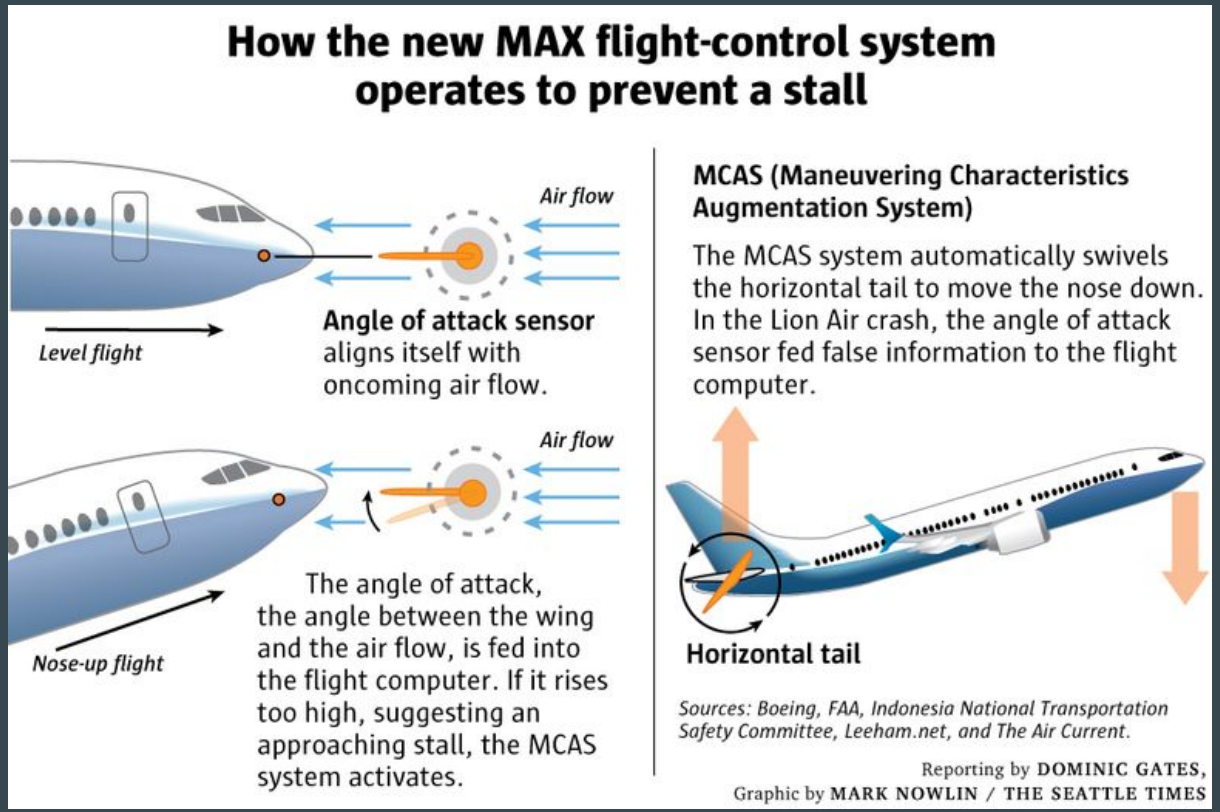
Transduction attack - using the actual physics of a sensor (transducer) to create intentional altered output

→ Sensors should be treated as untrustworthy

Problem Definition

Software (currently) blindly trusts data from sensors, even though these can be easily tampered with

Problems with Trusting Sensors



- AoA sensor malfunctions linked to 200+ FAA incident reports
- Aircraft was not tested in a “faulty sensor” setting

Vulnerabilities in MEMS Hardware: Ultrasound

- MEMS components all have vibrating pieces in them
- Find a way to control the vibrations

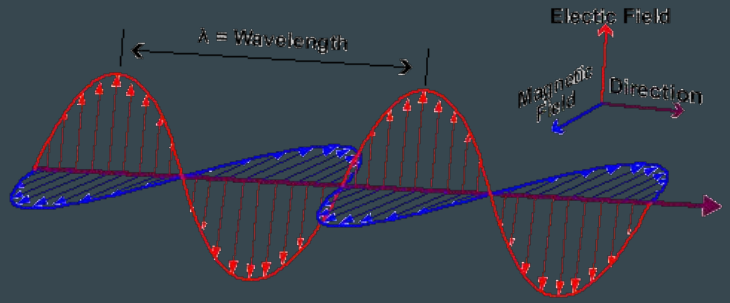
Vulnerabilities in MEMS Hardware: Lasers

13:46 - 14:42

More recent than this paper, but along the same vein

Malicious Back-Door Coupling

- Interfering with the system through wiring
- Lots of devices on a plane interfere with radio antenna wiring
- High voltage power wire interfering with data wires
- Bending an optical cable to distort data
- This doesn't have to be done with special equipment



How do we stop this?

- Avoid “component-centric” security: don’t make it a “component” in a pipeline
- Make sensor output “continuously checkable”: allow software to access more data
  - Constraint: performance and physical space constraints
- Specify physical security: manufacturers suggest how a piece of hardware might be secured

Some Concrete Solutions

- Simple redundancy: add a second sensor for comparison!
- Build a dynamic model to predict how a system should behave

Other ideas?

Education Soapbox

Interdisciplinary Teams

- Embedded systems teams should include people from MechE, EE, and CS
- Not *everyone* has to master sensor physics, but all should be aware of what kinds of risks come with sensors

Education Soapbox

Opportunities in Embedded Security

- Engineering schools should offer more classes on cyberphysical security: what are the risks of making computational abstractions?
- Each layer of abstraction has security risks which should be explored

Education Soapbox

Back to Basics

- CS programs shouldn't lose sight of the importance of hardware
- Moving forward, it's important to have software engineers that understand both the hardware and software of computers

"If a department eliminates computer architecture, students may seek comfort in hiding behind a beautiful Java facade rather than facing the ugly limitations of computing machinery."

Critique

- I would like to have seen more specifics into hardware risks and less general discussion about education
- More specifics on proposed solution frameworks
- Education section
  - Most students see/need/want the following path: college → internship → job, so “immediately marketable programming skills” can’t be dismissed so easily
  - Notes that CS programs are already jam-packed, assumes that engineering disciplines *do* have room for extra cyberphysical security classes
  - What else should schools do? Lacks specifics in the “schools should” area

Further Discussion Questions

- Who is responsible for making sure hardware is secure?
- Is it reasonable for students to get the good interdisciplinary education the paper calls for?
- What can be done at the manufacturing stage to improve security?