

# Intel Innovation Week

University of Johannesburg (23 – 27 Jun 2014)

Sensors



What will you make?



# Equipment required (per person)

Part	Qty	Cost	Purchase / Wiki link
LDR	1	1.80 EUR	<a href="#">Click here</a>
PIR motion sensor	1	5.60 EUR	<a href="#">Click here</a>
Piezo element	1	1.20 EUR	<a href="#">Click here</a>
Touch sensor	1	5.90 EUR	<a href="#">Click here</a>
Tilt switch	1	1.95 EUR	<a href="#">Click here</a>
LED	2	0.06 EUR	<a href="#">Click here</a>
Resistor (330ohm)	2	0.02 EUR	<a href="#">Click here</a>
Resistor (10k ohm)	2	0.02 EUR	<a href="#">Click here</a>
Jumper Wires (pack)	1	3.99 EUR	<a href="#">Click here</a>
Breadboard	1	4.59 EUR	<a href="#">Click here</a>

Total: 25.13 EUR

# Sensors – Introduction

# Introduction - What are Sensors?

- A sensor is a converter that measures a physical quantity and converts it into a signal.
- For accuracy, most sensors are calibrated against known standards.  
Eg calibrate temperature sensor against a thermometer (gold standard)
- Recent advances in Sensors  
MEMS – Micro Electro Mechanical Sensors  
Eg Accelerometers, Gyroscopes etc

Name some sensors in a Smartphone ?

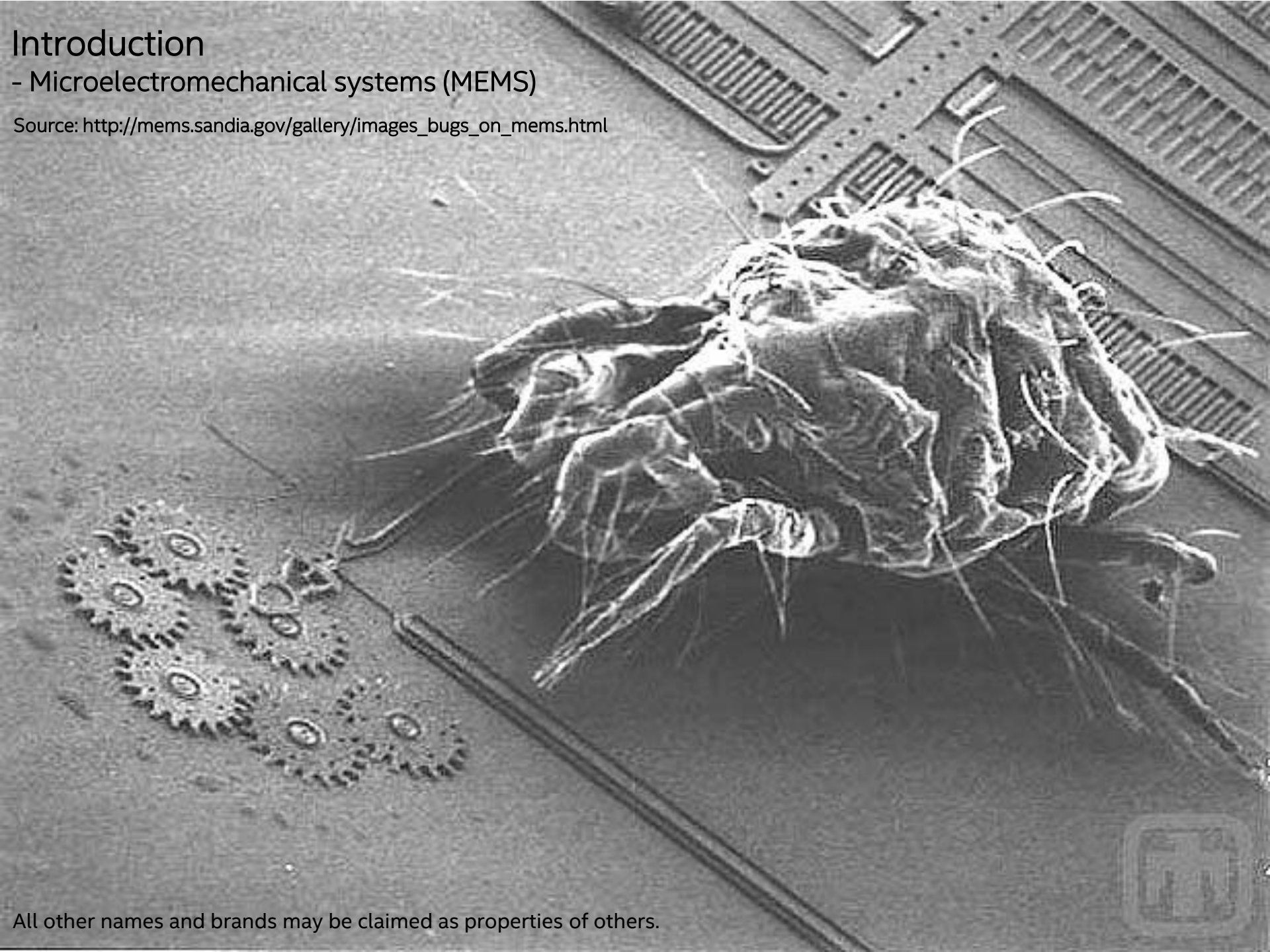


Proximity  
Touch/Tap  
Tilt  
Accelerometer  
Gyroscope  
Magnetometer  
GPS  
Temperature  
Light  
Altitude/Pressure  
Speech  
Camera

# Introduction

## - Microelectromechanical systems (MEMS)

Source: [http://mems.sandia.gov/gallery/images\\_bugs\\_on\\_mems.html](http://mems.sandia.gov/gallery/images_bugs_on_mems.html)



All other names and brands may be claimed as properties of others.

# Introduction - What can Galileo Sense?

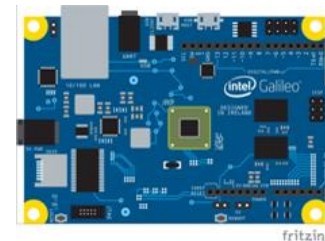
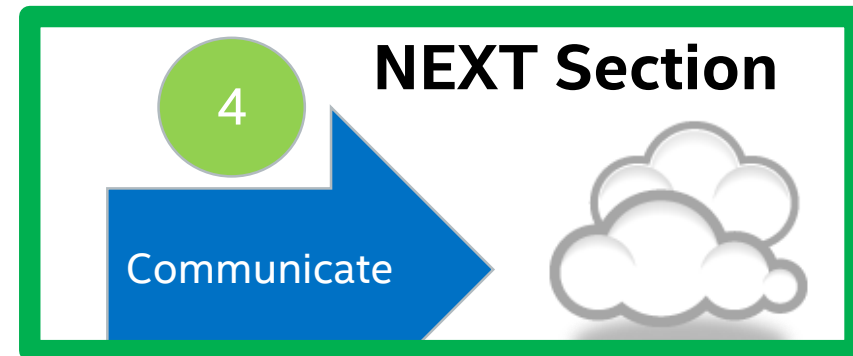
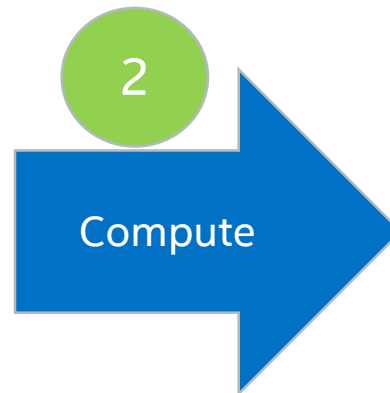
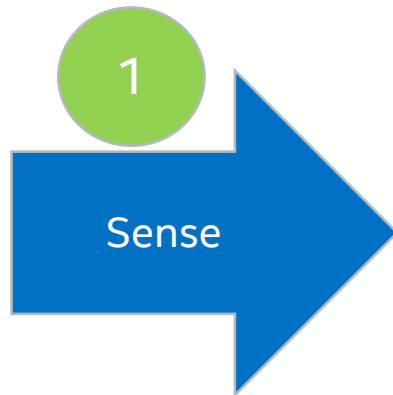
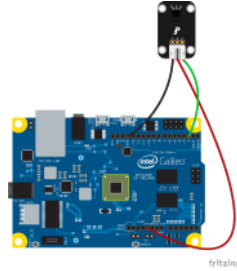
Light  
Distance  
Temperature  
Acceleration  
Vibration  
Motion  
Touch  
Gas  
Sound  
Etc....



Agricultural  
Automotive  
Medical  
Environmental  
Safety  
Security  
Transportation/Shipping  
Industrial Automation  
Home Automation

# Introduction – Sensor Data Flow

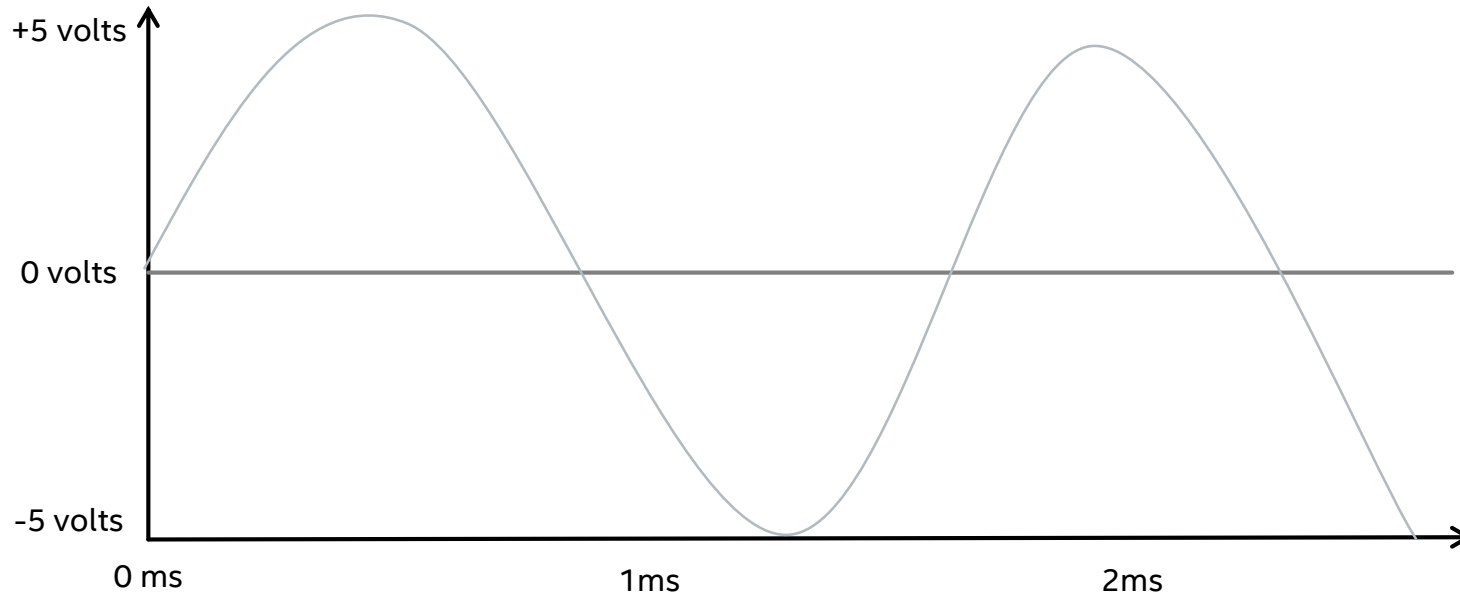
## Sensing Motion



All other names and brands may be claimed as properties of others.

# Introduction – Analog & Digital

What are analog & digital signals?

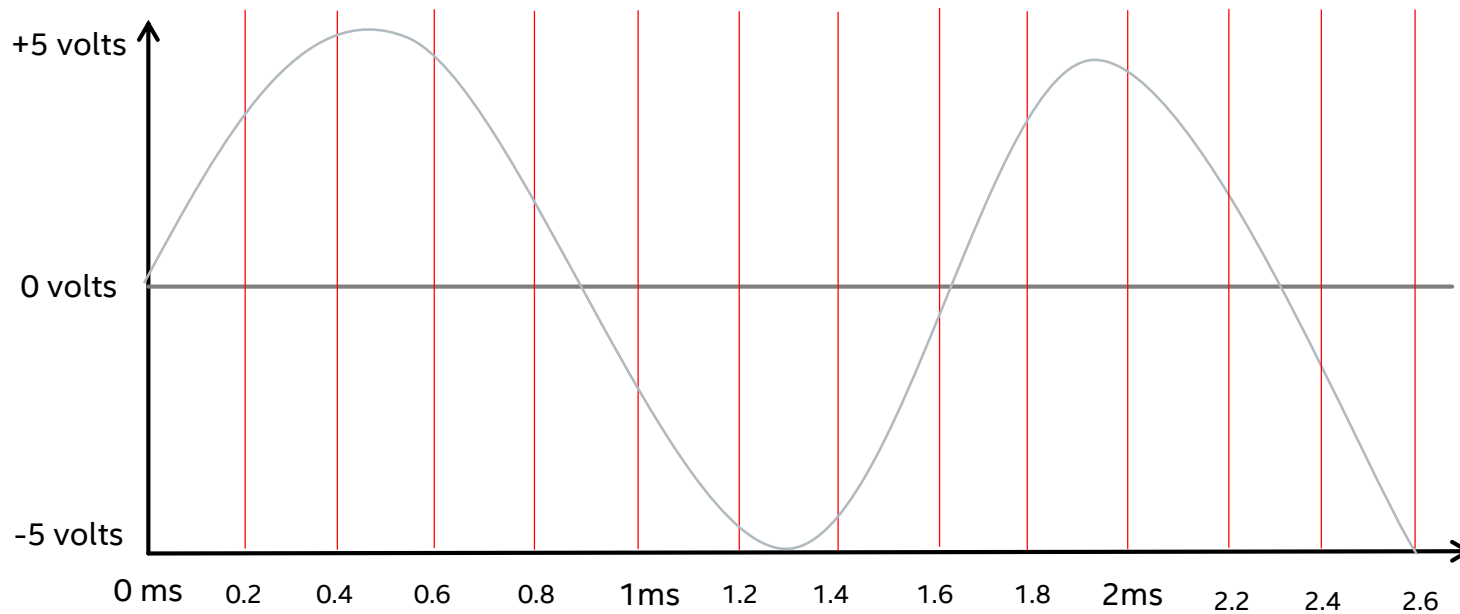


Analog signal  
Signal passes through infinite number of voltage levels  
between -5 /+5 volts.



# Introduction – Analog & Digital

What are analog & digital signals?



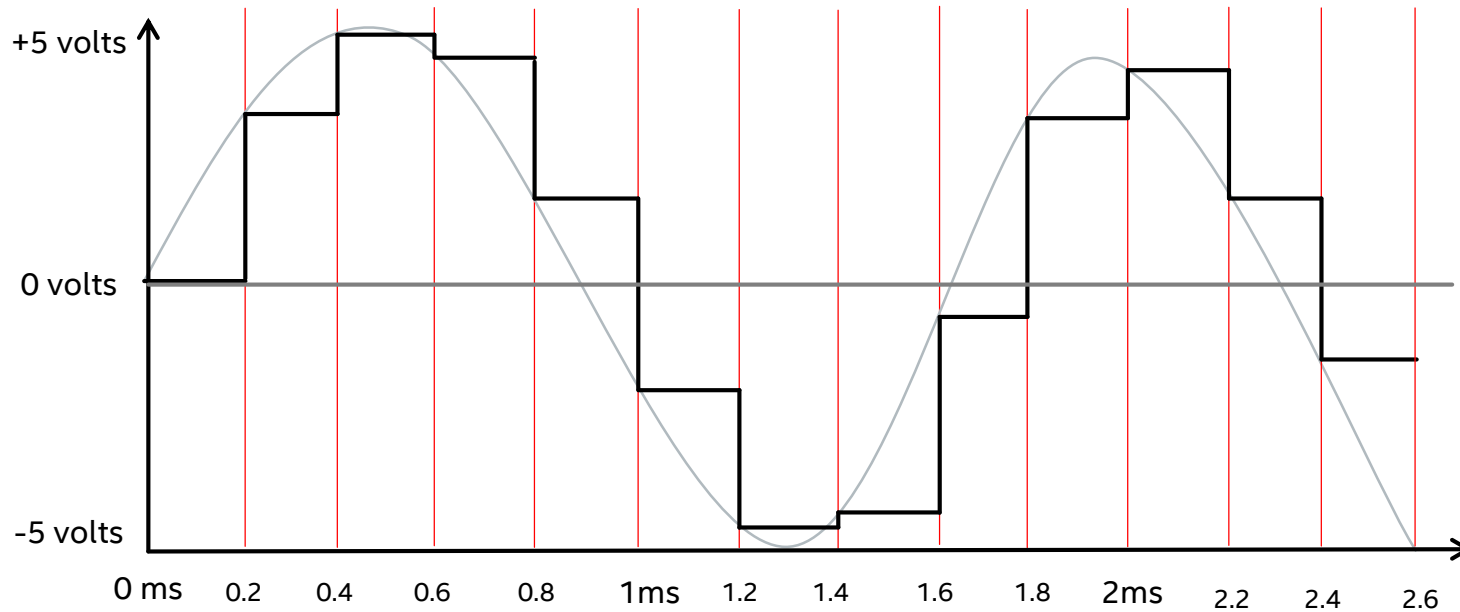
Analog signal

Signal passes through infinite number of voltage levels between -5 /+5 volts.

ADC sampling at 5kHz- take a 'snapshot' of the voltage level every 200uS

# Introduction – Analog & Digital

What are analog & digital signals?



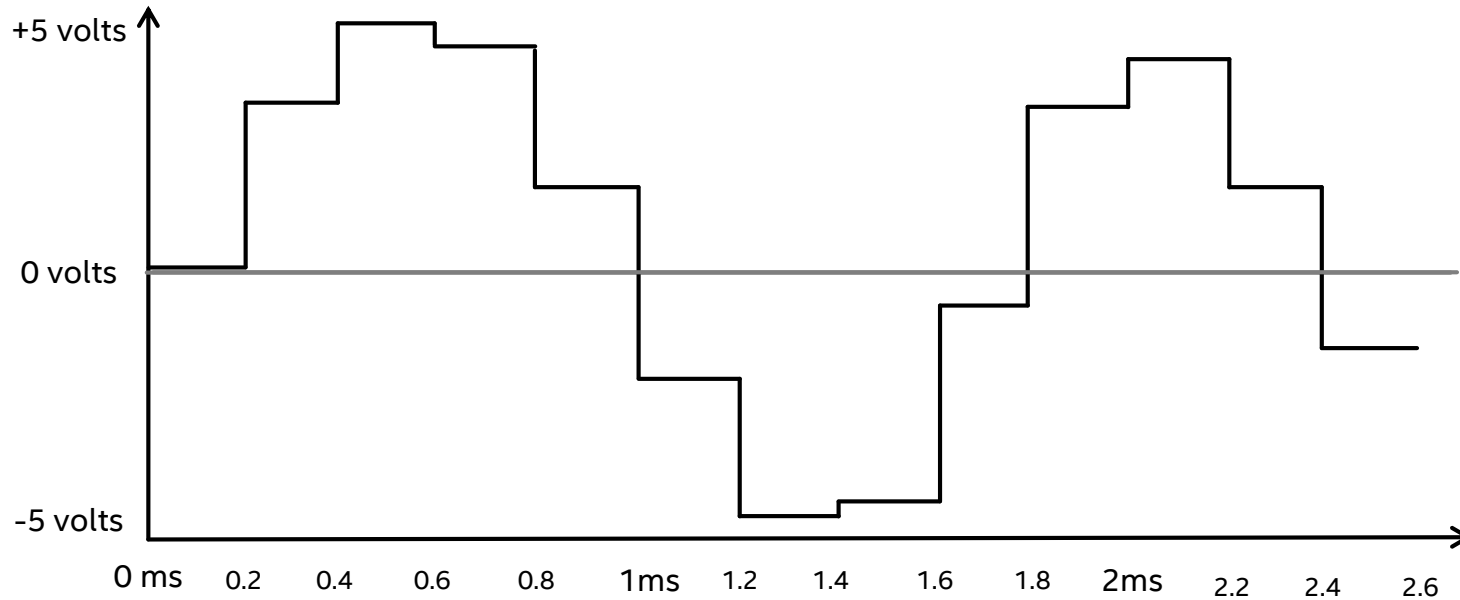
Analog signal

Signal passes through infinite number of voltage levels between -5 /+5 volts.

ADC sampling at 5kHz- take a 'snapshot' of the voltage level every 200µs

# Introduction – Analog & Digital

What are analog & digital signals?



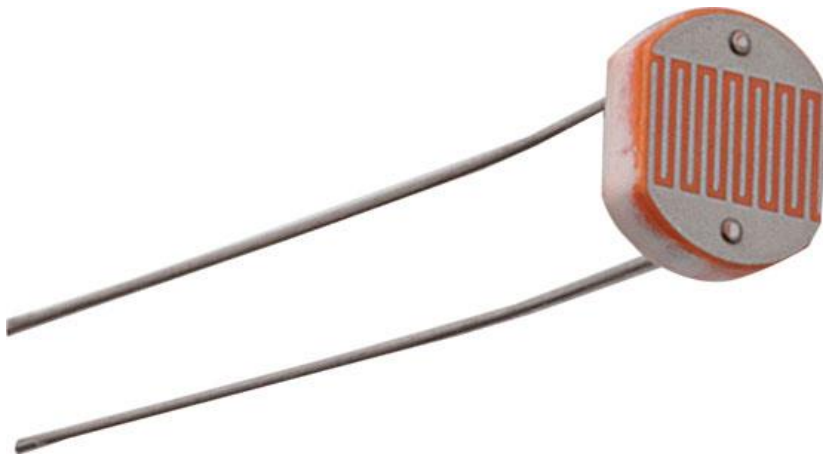
**Now we have a digital signal, which contains a discrete range of values.**

**We can store these values in some digital storage medium, and use some tricks to approximate the analog 'curve' when we want to re-create the analog signal.**

# Sensors – Light

# Light Sensing - Light Dependent Resistor (LDR)

- An LDR is very similar to an ordinary resistor; it is made of a semi-conductive material that will partially restrict the flow of electricity through the device.
- Variable resistor - proportional to how much light the LDR is exposed to.
- Good enough for differentiating between light & dark



# Light Sensing – Basic LDR



Project Name: LDR (*Instructor Lead*)

**Objective:** read the value of an LDR connected to an analog pin, and print it out on the Serial port.

## Software Elements

- `analogRead()`
- `analogWrite()`
- `Serial`
- `delay()`

## Components

- Intel Galileo Board (Qty 1)
- Breadboard (QTY1)
- Jumper Wires
- 5mm LED (Qty 1)
- 220 Ohm Resistor (Qty 2)
- LDR (Qty 1)

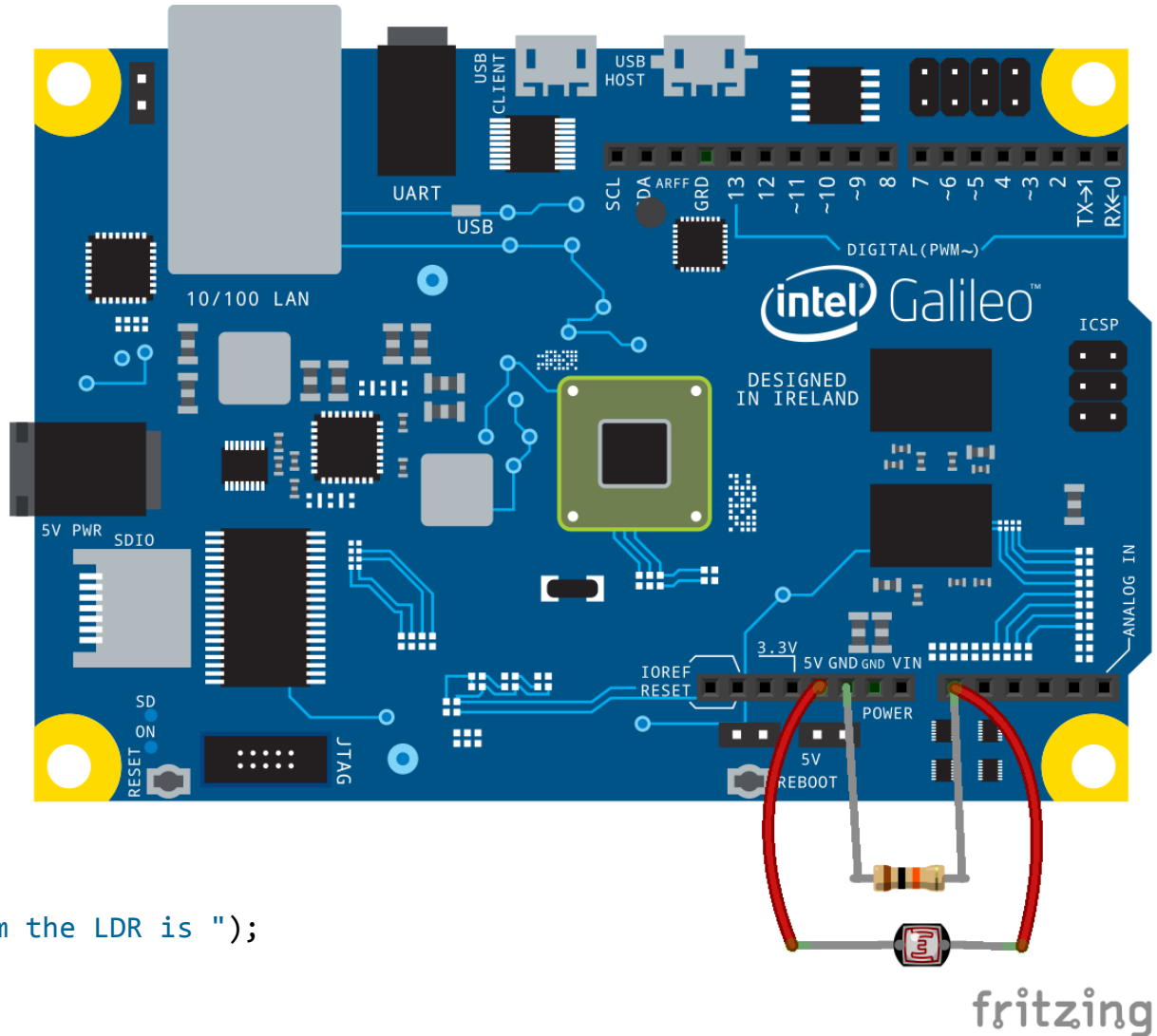
# Light Sensing – Reading the LDR

LAB

```
#define PIN 0
int value;

void setup() {
    Serial.begin(115200);
}

void loop() {
    value = analogRead(PIN);
    Serial.print("The value from the LDR is ");
    Serial.println(value);
    delay(200);
}
```



Project Files: Lesson3-Sensing\Section1-Light\LDR\_basic

# Light Sensing - Engineering Challenge



Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

## Challenge:

Add an LED to the circuit via a PWM-enabled pin, and use values read from the LDR to control the brightness of the LED.



# Engineering Challenge Review

## Challenge 1: Add LED, use LDR value to control PWM signal

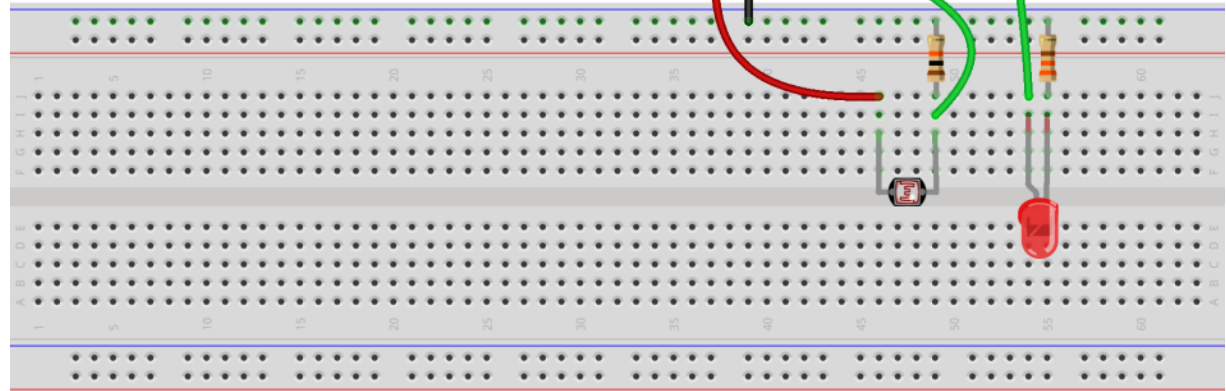
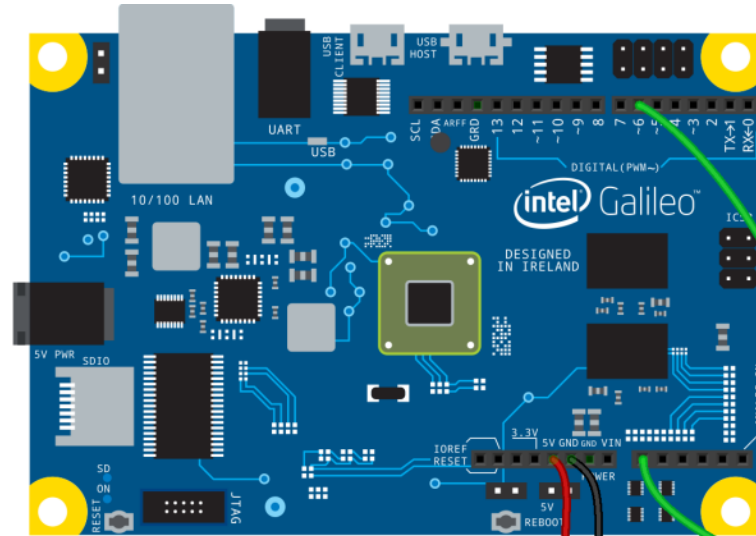
Challenge Review

```
#define A_PIN 0    //analog pin for LDR
#define D_PIN 6    //digital pin for LED

int value;

void setup() {
  pinMode(D_PIN, OUTPUT);
}

void loop() {
  value = map(analogRead(A_PIN), 0, 1023, 0, 255);
  analogWrite(D_PIN, value);
}
```

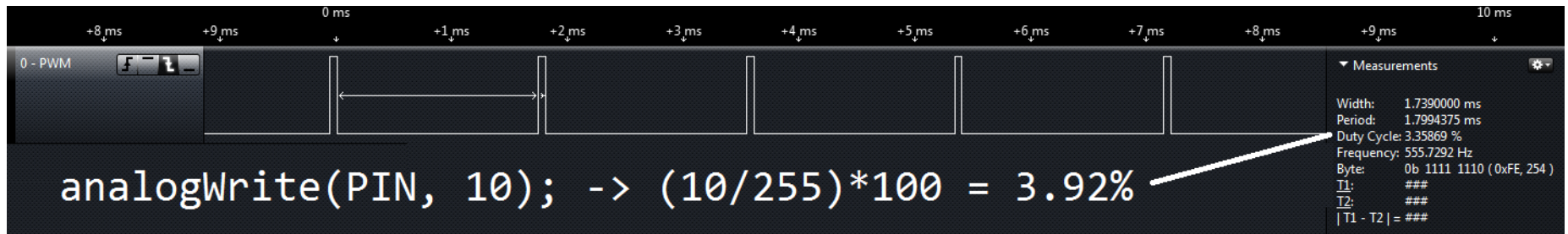


fritzing

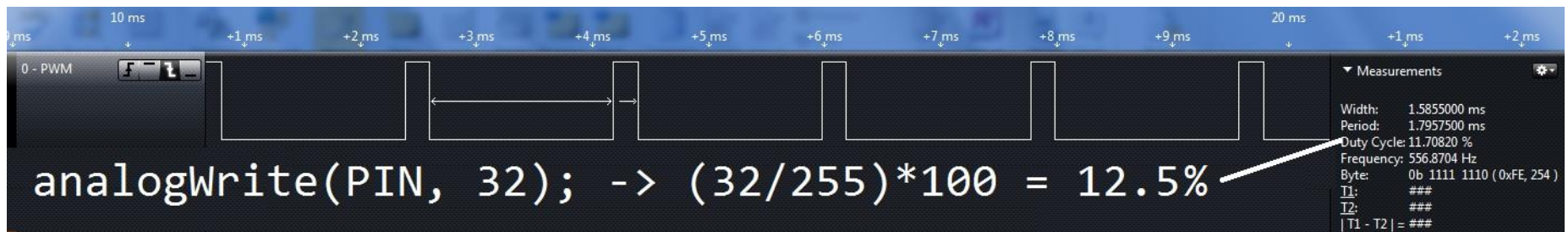
Project Files: Lesson3-Sensing\Section1-Light\LDR\_challenge

# PWM duty cycle on Logic Analyser

Darker – finger over LDR (PWM= ~10)



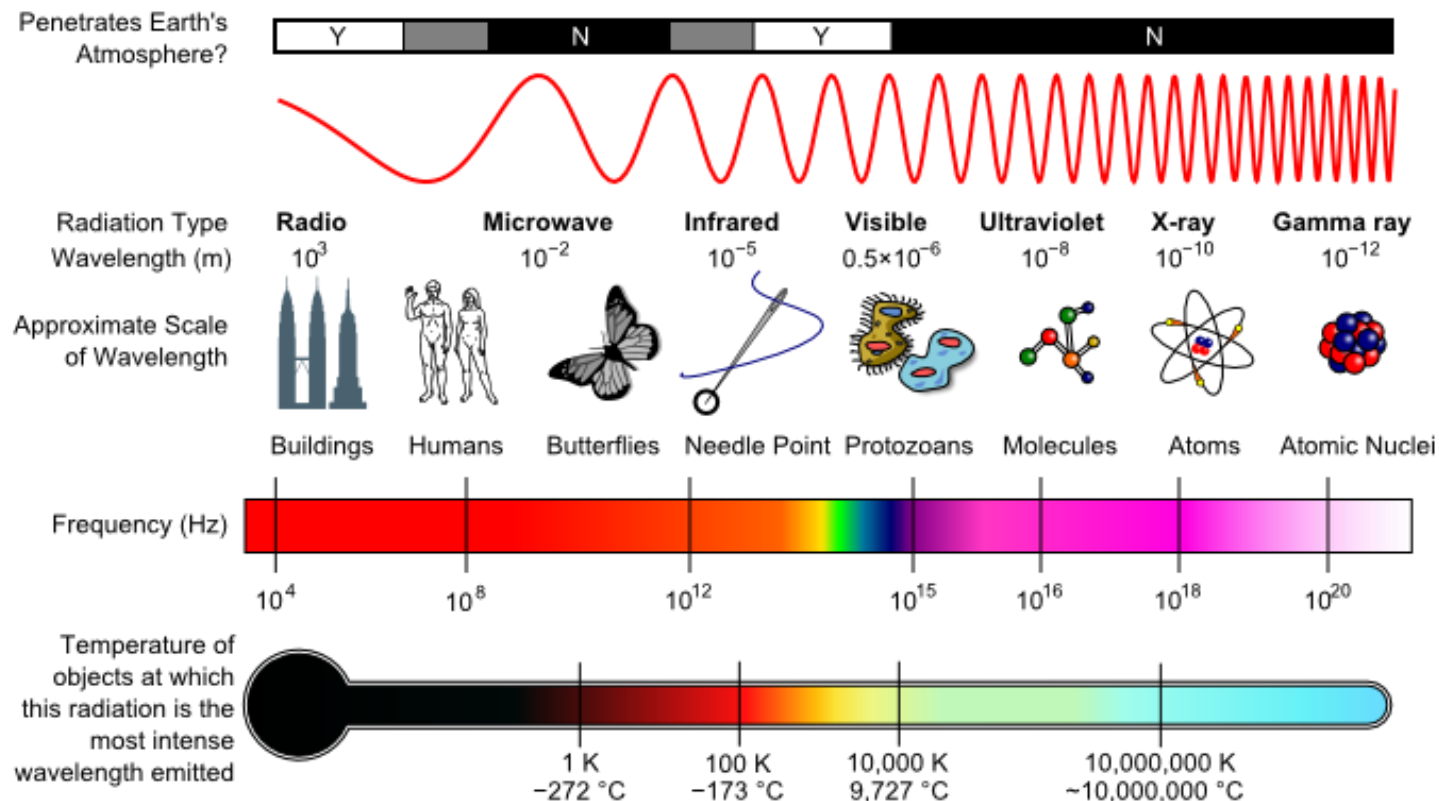
Brightness – no finger over it (PWM = ~32)



# Sensors – Motion

# PIR Motion Sensor

- PIR (**P**assive **I**nfra-**R**ed) motion sensors work by sensing changes in infra-red energy.
- All objects with a temperature above absolute zero (which is about -270 degrees celcius, so pretty much everything) emit heat energy in the form of infra-red radiation.



All other names and brands may be claimed as properties of others.

# PIR Motion Sensor

A PIR sensor is not a temperature / heat sensor

Most PIR sensors are very sensitive

- Will detect movement even if the moving object has no change in temperature.
- Detects the smaller changes in the various sources of infra-red radiation around it.

The PIR sensor has a digital output; high or low.

- High = motion has been detected.
- Low = Normal and no motion is being detected.

**Simple, right? Let's try an example.....**

# PIR Motion Sensor

Project Name: PIR motion sensor

Objective: Read the value from the motion sensor, print a message when motion is detected.

## Software Elements

- `analogRead()`
- `Serial`
- `delay()`

## Components

- Intel Galileo Board (Qty 1)
- Breadboard (QTY1)
- Jumper Wires
- PIR motion sensor

# PIR Motion Sensor

LAB

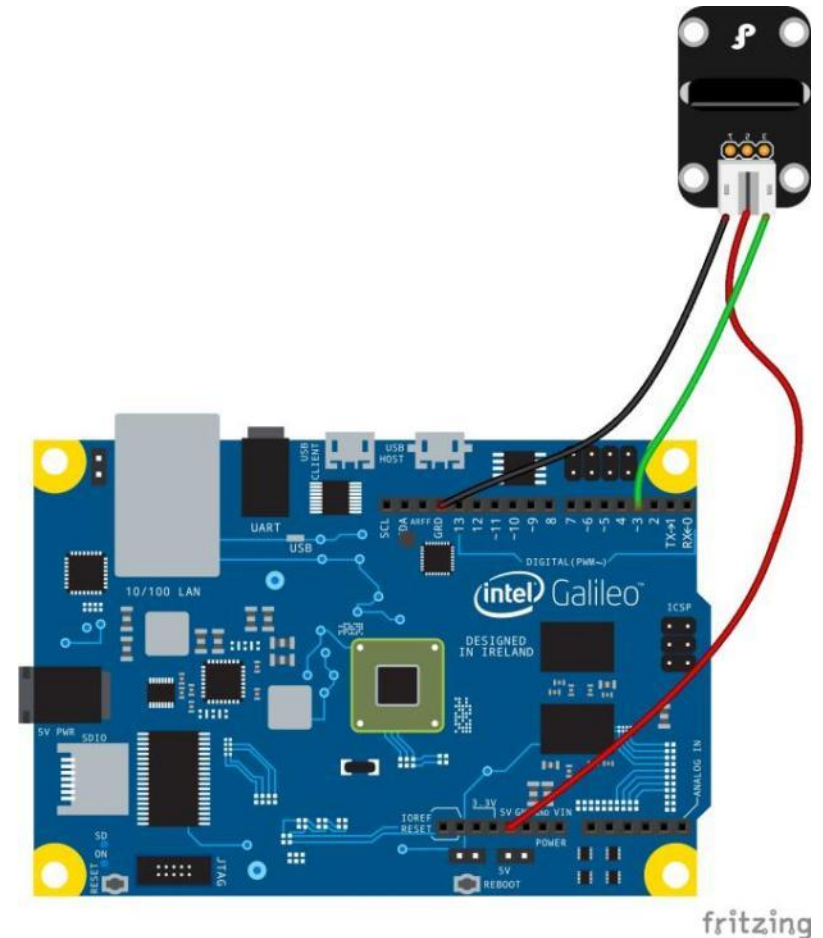
## Sensor Reference Page:

[http://www.dfrobot.com/wiki/index.php/Digital\\_Infrared\\_motion\\_sensor\\_\(SKU:SEN0018\)](http://www.dfrobot.com/wiki/index.php/Digital_Infrared_motion_sensor_(SKU:SEN0018))

```
short pin = 3;

void setup()
{
    Serial.begin(115200);
    pinMode(pin, INPUT);
}

void loop() {
    if (digitalRead(pin) == 1)
    {
        Serial.println("Motion detected!");
    }
}
```



Project Files: Lesson3-Sensing\Section2-Motion\PIR\_motion\_sensor\_basic

# PIR Motion Sensor



Well, it sort of works, but there's one problem...





# PIR Motion Sensor



```
#define MOTION 1
#define NOMOTION 0

short state = NOMOTION;
short pin = 3;
short value;

void setup()
{
    Serial.begin(115200);
    pinMode(pin, INPUT);
}

void loop() {
    value = digitalRead(pin);
    if (value == MOTION && state == NOMOTION)
    {
        state = MOTION;
        Serial.println("Motion detected!");
    }

    else if (value == NOMOTION && state == MOTION)
    {
        state = NOMOTION;
        Serial.println("No more motion.");
    }
}
```

To stop the message printing continuously while motion is detected, we'll add an extra variable called `state`, to keep track of the state of the pin we are reading.

Initially, `state` will contain a value of 0. After we take a reading from the motion sensor, inside the condition portion of the `if` statement, we check if the motion sensor's pin is high **and** if the state variable is equal to 0. If both of these are true, then set state to 1 and do the `Serial.print()`.

Now, we also have to add a second `if` statement – if the signal from the motion sensor goes low and `state` is still equal to 1, we must set it back to 0 to the next high signal can be detected.

# PIR Motion Sensor - Engineering Challenge

Using your own circuit design and sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

## Challenge:

Add an LED to the previous circuit. When motion is detected, fade the LED from dark up to full brightness. Keep the LED lit until motion is not detected, at which point the LED should fade down, and remain off until motion is detected again.

## Challenge 1: PIR Motion sensor with LED

```
#define MOTION 1
#define NOMOTION 0
#define INPIN 3
#define OUTPIN 5

short state = NOMOTION;
short value;

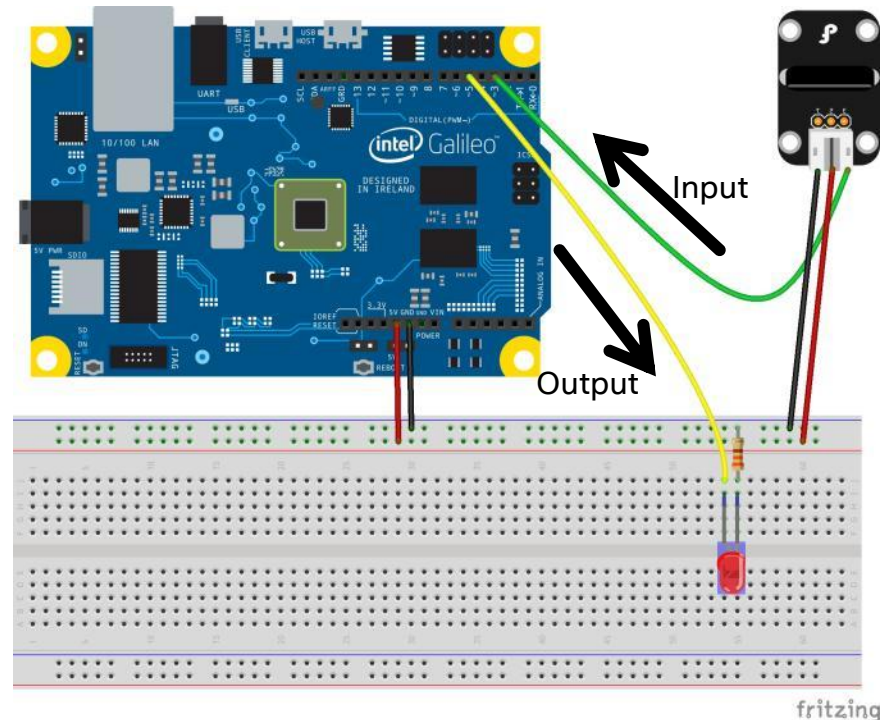
void setup()
{
    Serial.begin(115200);
    pinMode(INPIN, INPUT);
}

void fadeUp(int pin)
{
    for (int i = 0; i <= 255; i++)
    {
        analogWrite(pin, i);
    }
}

void fadeDown(int pin)
{
    for (int i = 255; i >= 0; i--)
    {
        analogWrite(pin, i);
    }
}

void loop()
{
    value = digitalRead(INPIN);
    if (value == MOTION && state == NOMOTION)
    {
        state = MOTION;
        fadeUp(OUTPIN);
    }

    else if (value == NOMOTION && state == MOTION)
    {
        state = NOMOTION;
        fadeDown(OUTPIN);
    }
}
```



**Project Files:** Lesson3-Sensing\Section2-Motion\PIR\_motion\_sensor\_challenge

Break Time 😊

# Sensors – Vibration

# Vibration sensor

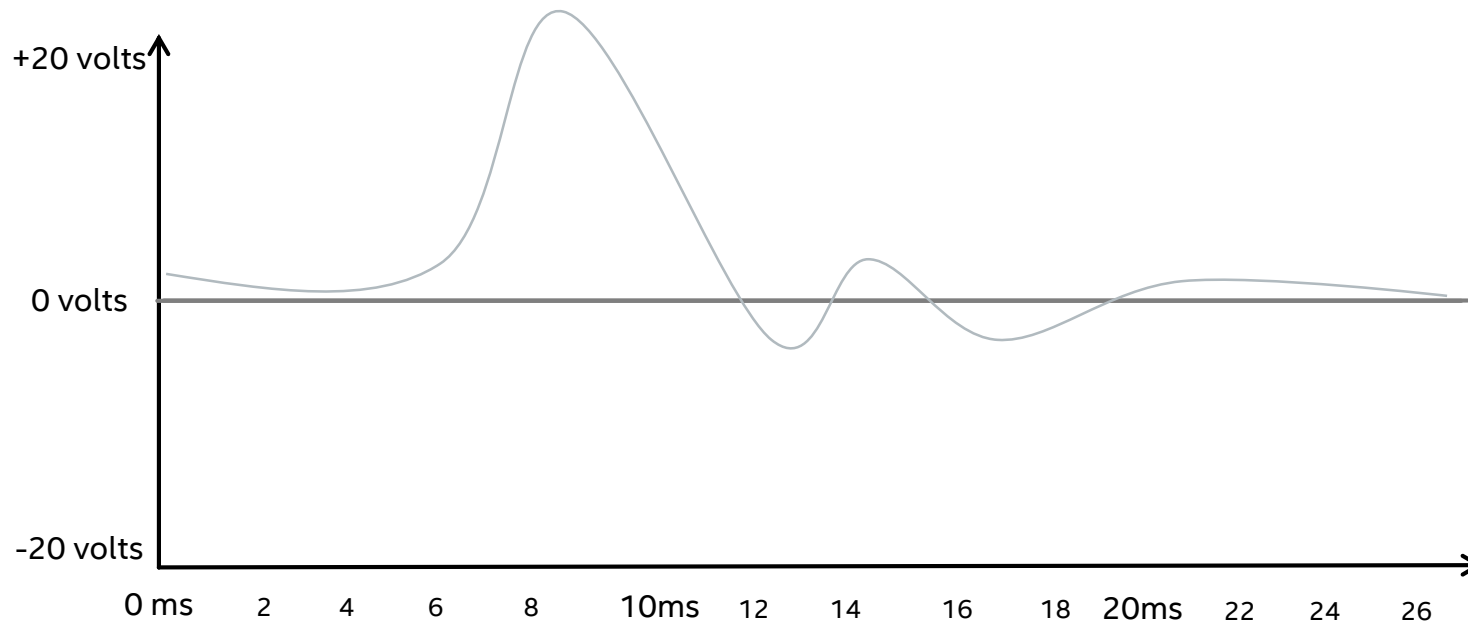
Commonly known as a '**piezo element**', '**piezo transducer**' or simply '**piezo disc**'.

- It consists of a small slice of piezo crystal, coupled with thin copper disc.
- Voltage Spike across the copper disc and piezo element occurs when something impact the disc (i.e. if you drop it, tap it, slap it etc),
- This sensor is passive, meaning no power source is required for it to operate.

**Now, let's look at some of the problems we'll meet in trying to read this sensor, and how we can work around them....**

# Vibration sensor

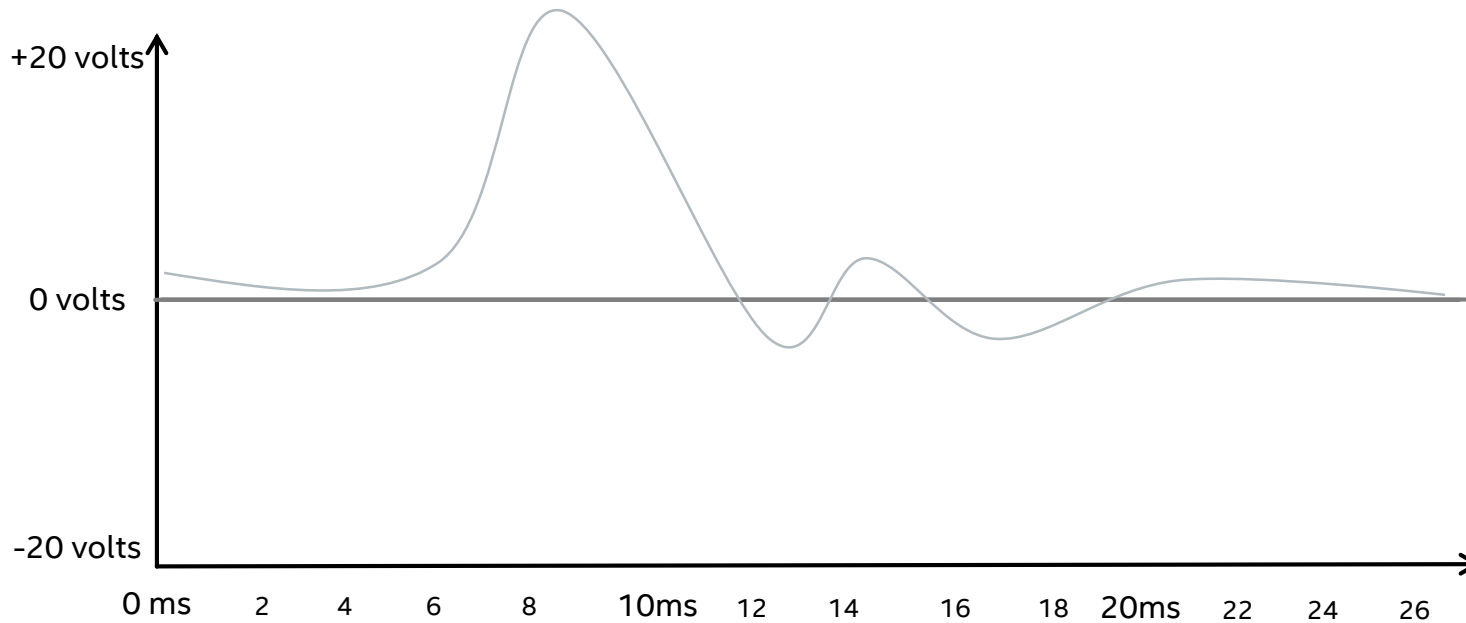
here's an approximation of what the voltage spike looks like when you give the piezo a single tap.





# Vibration sensor

here's an approximation of what the voltage spike looks like when you give the piezo a single tap.



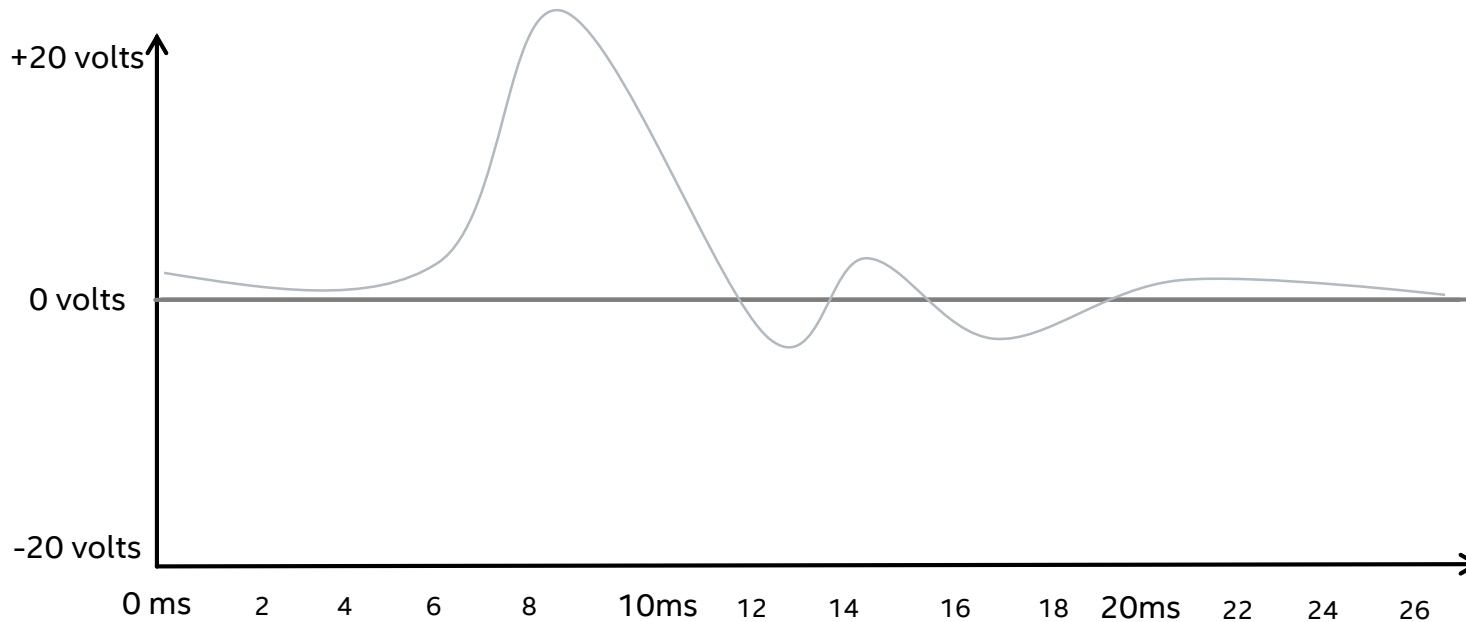
Let's write a quick sketch to read this sensor and detect an impact:

```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  if (analogRead(0) > 600)
  {
    Serial.println("Impact!");
  }
}
```

# Vibration sensor

here's an approximation of what the voltage spike looks like when you give the piezo a single tap.



Let's write a quick program to read this sensor and detect an impact:

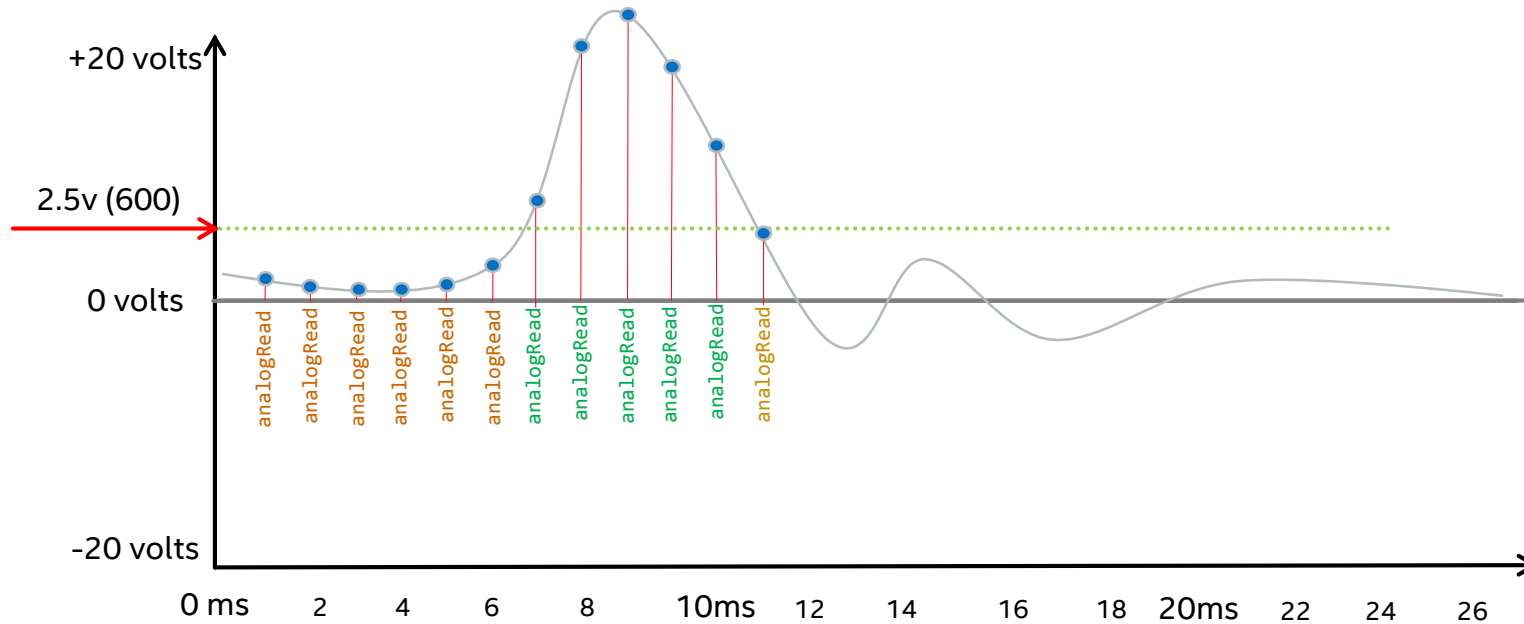
```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  if (analogRead(0) > 600)
  {
    Serial.println("Impact!");
  }
}
```

**Can anyone see a  
problem with this  
program?!**

# Vibration sensor

here's an approximation of what the voltage spike looks like when you give the piezo a single tap.



Let's write a quick sketch to read this sensor and detect an impact:

```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  if (analogRead(0) > 600)
  {
    Serial.println("Impact!");
  }
}
```

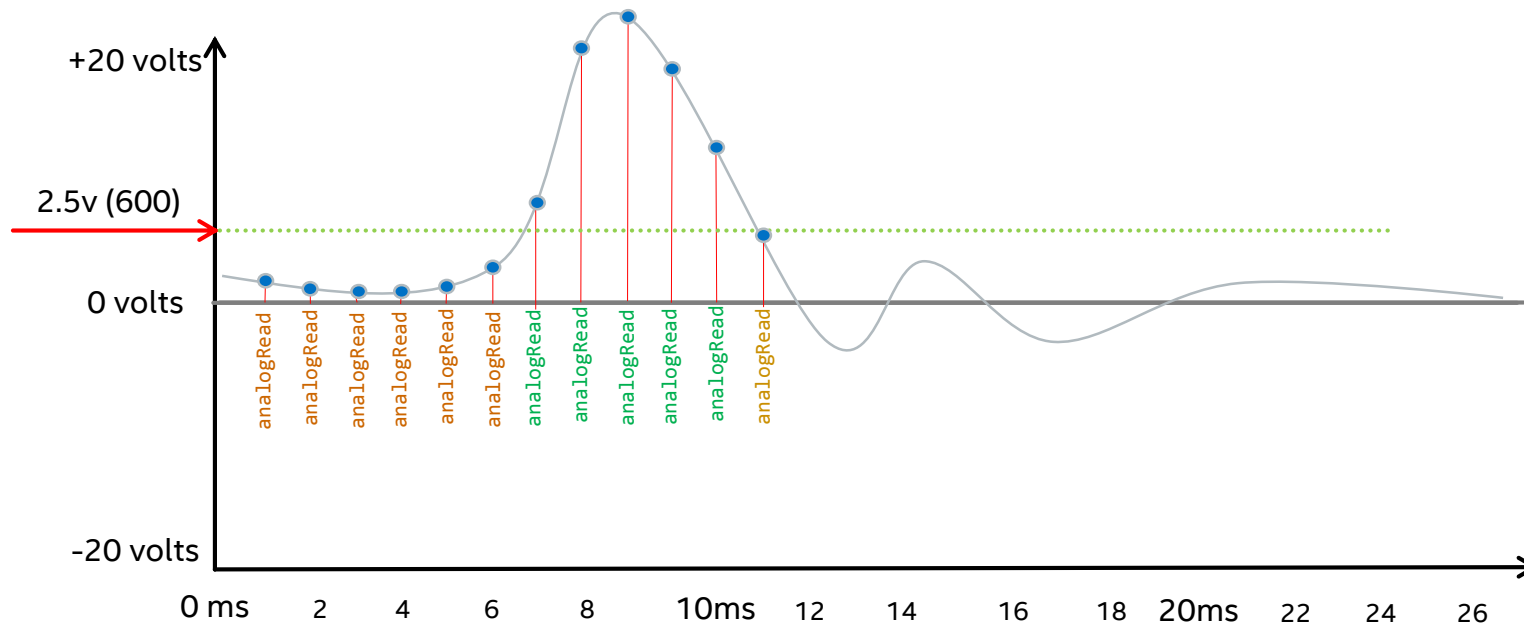
Output:

Impact!  
Impact!  
Impact!  
Impact!  
Impact!

**Can anyone see a  
problem with this  
sketch?!**

# Vibration sensor

here's an approximation of what the voltage spike looks like when you give the piezo a single tap.



Let's write a quick sketch to read this sensor and detect an impact:

```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  if (analogRead(0) > 600)
  {
    Serial.println("Impact!");
  }
}
```

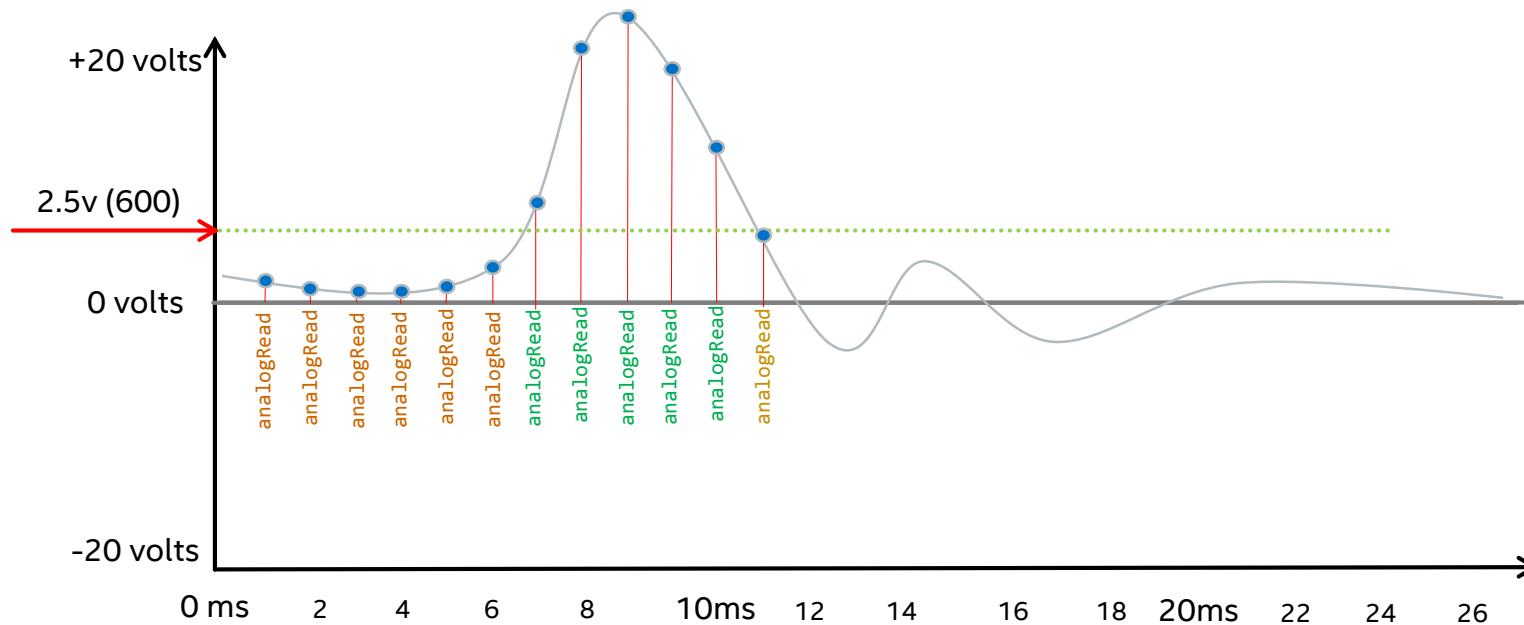
Output:

Impact!  
Impact!  
Impact!  
Impact!  
Impact!

After the initial impact, the piezo signal takes some time to decay- only a few milliseconds. But the Galileo's ADC is taking readings fast enough that, by the time it comes around for the next reading, the signal is still decaying and it registers **another** impact (even though it's still decaying from the **first** impact)

# Vibration sensor

here's an approximation of what the voltage spike looks like when you give the piezo a single tap.



Let's write a quick sketch to read this sensor and detect an impact:

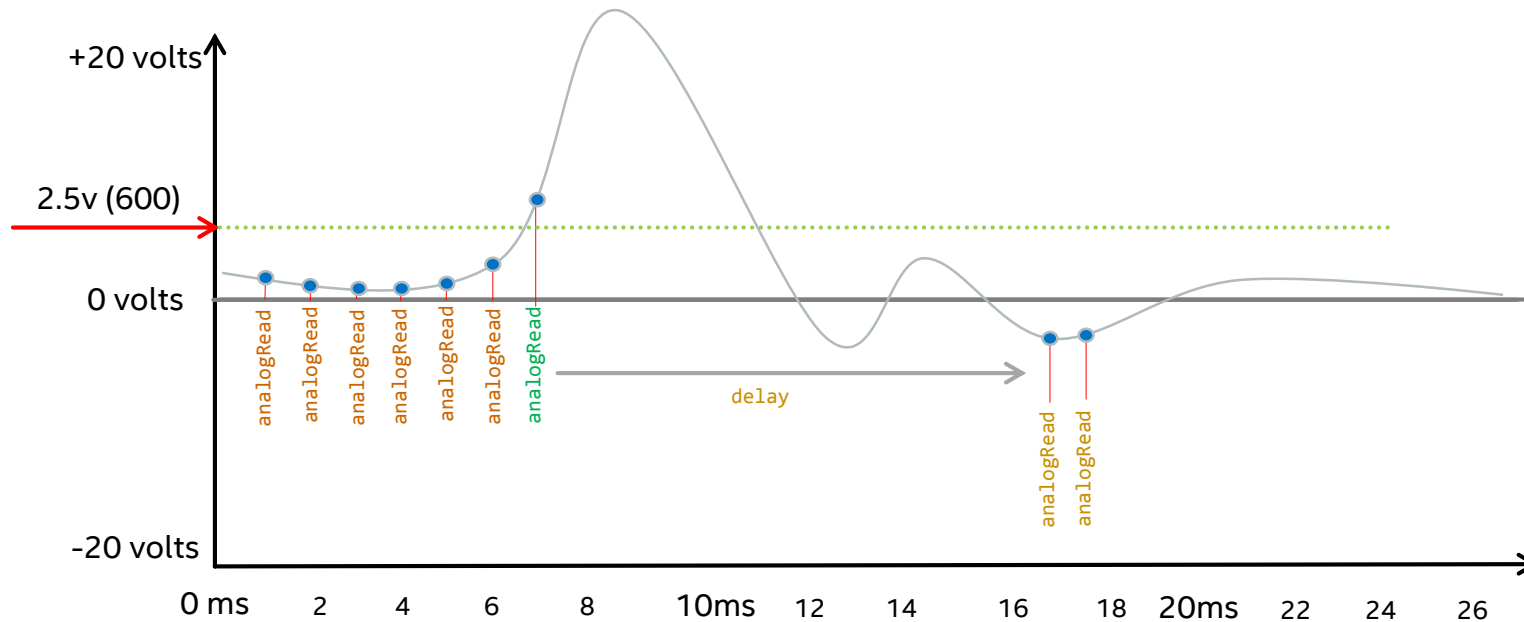
```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  if (analogRead(0) > 600)
  {
    Serial.println("Impact!");
  }
}
```

Before we move on to the lab, see if you can fix this sketch by adding **one more line**

# Vibration sensor

here's an approximation of what the voltage spike looks like when you give the piezo a single tap.



Let's write a quick sketch to read this sensor and detect an impact:

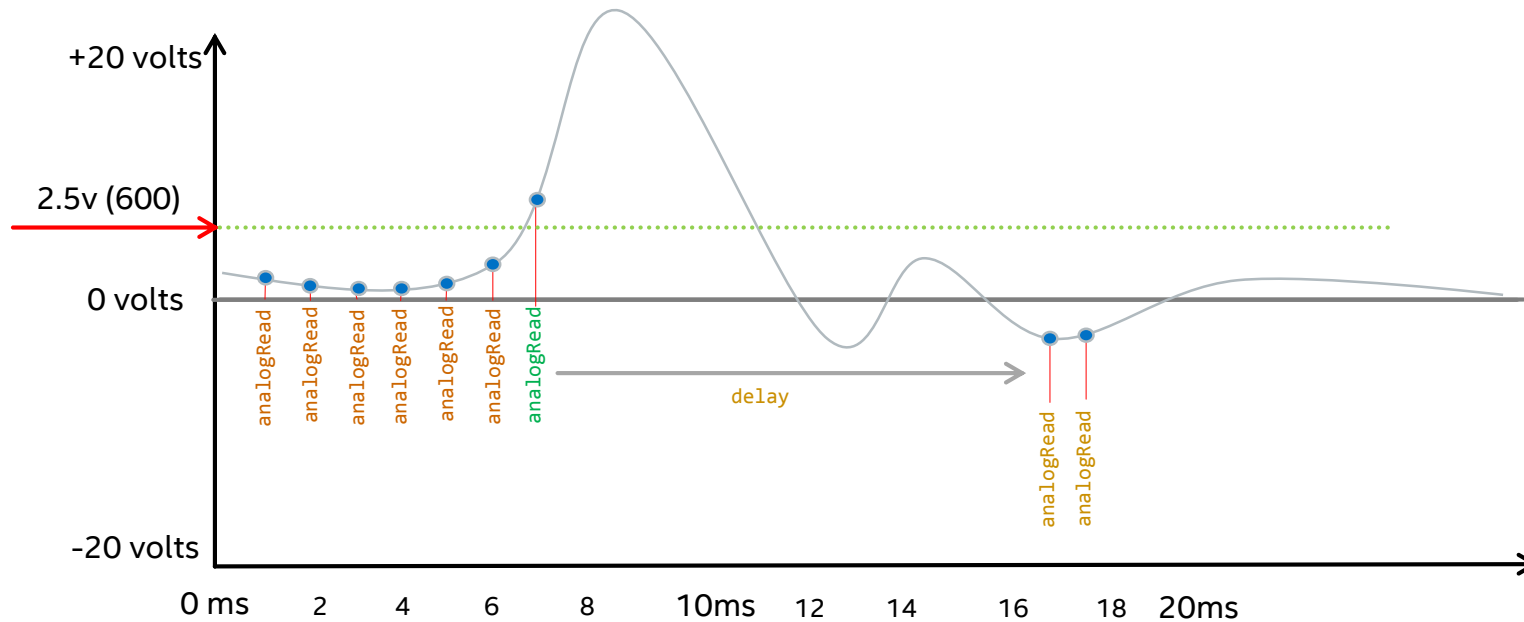
```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  if (analogRead(0) > 600)
  {
    Serial.println("Impact!");
    delay(10);
  }
}
```

Before we move on to the lab, see if you can fix this program by adding **one more line**

# Vibration sensor

here's an approximation of what the voltage spike looks like when you give the piezo a single tap.



Let's write a quick sketch to read this sensor and detect an impact:

```
void setup()
{
  Serial.begin(115200);
}

void loop()
{
  if (analogRead(0) > 600)
  {
    Serial.println("Impact!");
    delay(10);
  }
}
```

Before we move on to the lab, see if you can fix this program by adding **one more line**

NOTE: This technique works in our simple case, but it's not the best option since **delay()** blocks execution (program will wait for 10 ms before continuing)

# Sensors - Piezo element

Project Name: piezo debouncing (*Instructor Lead*)

**Objective:** read the analog voltage from a piezo element, and use **non-blocking** debouncing techniques to determine if there has been a 'hit'

## Software Elements

- `analogRead()`
- `millis()`
- `Serial`
- `delay()`

## Components

- Intel Galileo Board (Qty 1)
- Breadboard (QTY1)
- Jumper Wires
- 1 Mega Ohm Resistor (Qty 1)



# Piezo debouncing

LAB

## Sensor Reference Page:

<http://ie.farnell.com/multicomp/abt-448-90-rc/piezo-element-35mm-2-900hz-leaded/dp/1675549>

Or you can use this one this one if you have a sensor kit. I found threshold of dfrobot sensor to be ~110

[http://www.dfrobot.com/wiki/index.php?title=Analog\\_Piezo\\_Disk\\_Vibration\\_Sensor\\_\(SKU:DFR0052\)](http://www.dfrobot.com/wiki/index.php?title=Analog_Piezo_Disk_Vibration_Sensor_(SKU:DFR0052))

```
#define THRESH 300
//values higher than this value register a 'hit'

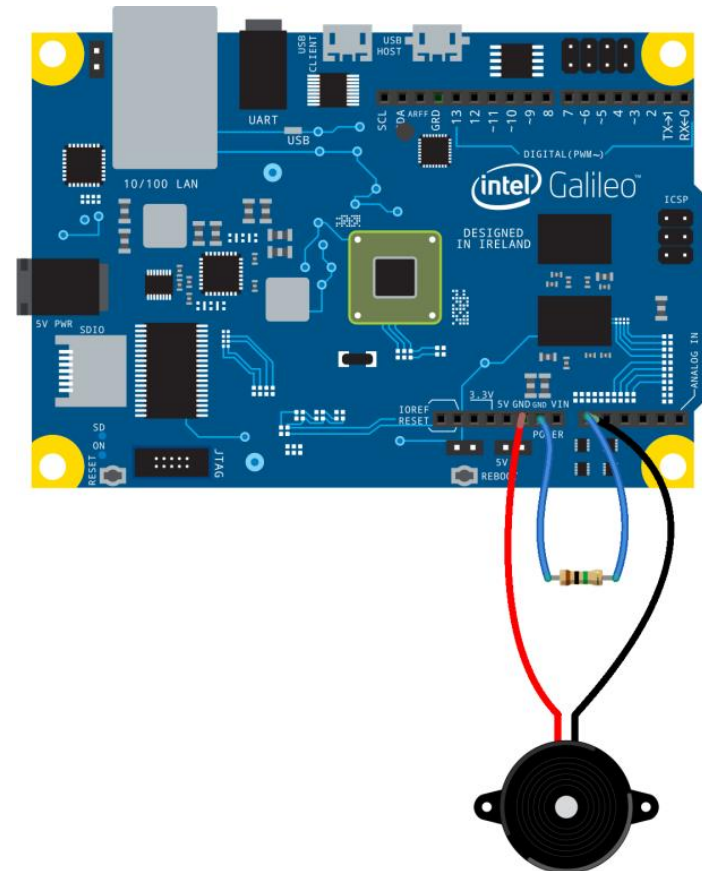
#define WAIT 150
//how long we wait before reading again

#define PIN 0
int value;
long timer = 0;

void setup() {
    Serial.begin(115200);
}

void loop() {
    value = analogRead(PIN);
    if (value > THRESH && timer == 0) {
        timer = millis();
        Serial.println("That's a hit!");
    }

    else if ((millis() - timer) >= WAIT && value < THRESH) {
        timer = 0;
    }
}
```



Project Files: Lesson3-Sensing\Section3-Vibration\vibration\_sensor\_basic

fritzing

# Sensors – Touch

# Touch sensor

- This capacitive touch sensor works kind of like a capacitor, except your body acts as one half of the capacitor.
- So when you touch your finger (or any other body part) to the metal surface, the significant change in capacitance indicates that a touch has occurred.
- This underlying concept is the basis for touch screens on phones and tablets.



Image source: <http://www.dfrobot.com>

**Before we move on to the lab, take 10 minutes and see if you can adapt the code from the 'PIR motion sensor' lesson to work for this sensor.**

All other names and brands may be claimed as properties of others.

# Touch Sensor

Project Name: Touch sensor (*Instructor Lead*)

Objective: Read the touch sensor and detect when somebody is touching it.

## Software Elements

- `digitalRead()`
- `millis()`
- `Serial`
- `delay()`

## Components

- Intel Galileo Board (Qty 1)
- Breadboard (QTY1)
- Jumper Wires
- 1 Mega Ohm Resistor (Qty 1)

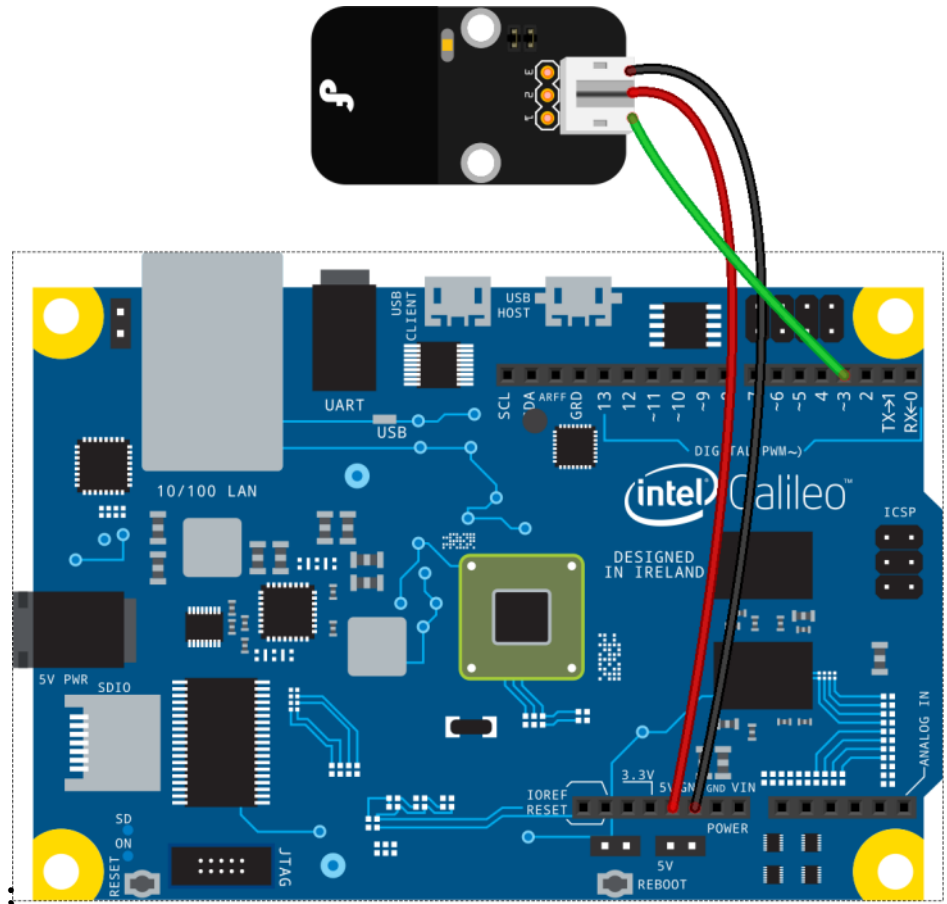
# Touch Sensor

## Sensor Reference Page:

[http://www.dfrobot.com/index.php?route=product/product&product\\_id=78#.U5nw7vk7uSp](http://www.dfrobot.com/index.php?route=product/product&product_id=78#.U5nw7vk7uSp)

```
#define PIN 3
short state = 0;
short value;
void setup()
{
    Serial.begin(115200);
    pinMode(PIN, INPUT);
}

void loop()
{
    value = digitalRead(PIN);
    if (value == 1 && state == 0)
    {
        state = 1;
        Serial.println("Touch");
    }
    else if (value == 0 && state == 1)
    {
        state = 0;
        Serial.println("Finger lifted");
    }
}
```



fritzing

Project Files: Lesson3-Sensing\Section3-Touch\touch\_sensor\_basic

# Touch Sensor- Engineering Challenge



Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

## Challenge:

Add an LED to your circuit, and use the touch sensor to control the brightness of the LED in steps. Use 4 steps, with each step brighter than the last. When the highest step is reached, the next touch will send it back to the first step.

# Engineering Challenge Review

## Challenge 1: Cycle through LED brightness levels

```
#define TOUCH 3
#define LED 5
short state = 0;
short value;
const int step = 80;
const int PWM_MAX = 255;
short brightness = 0;

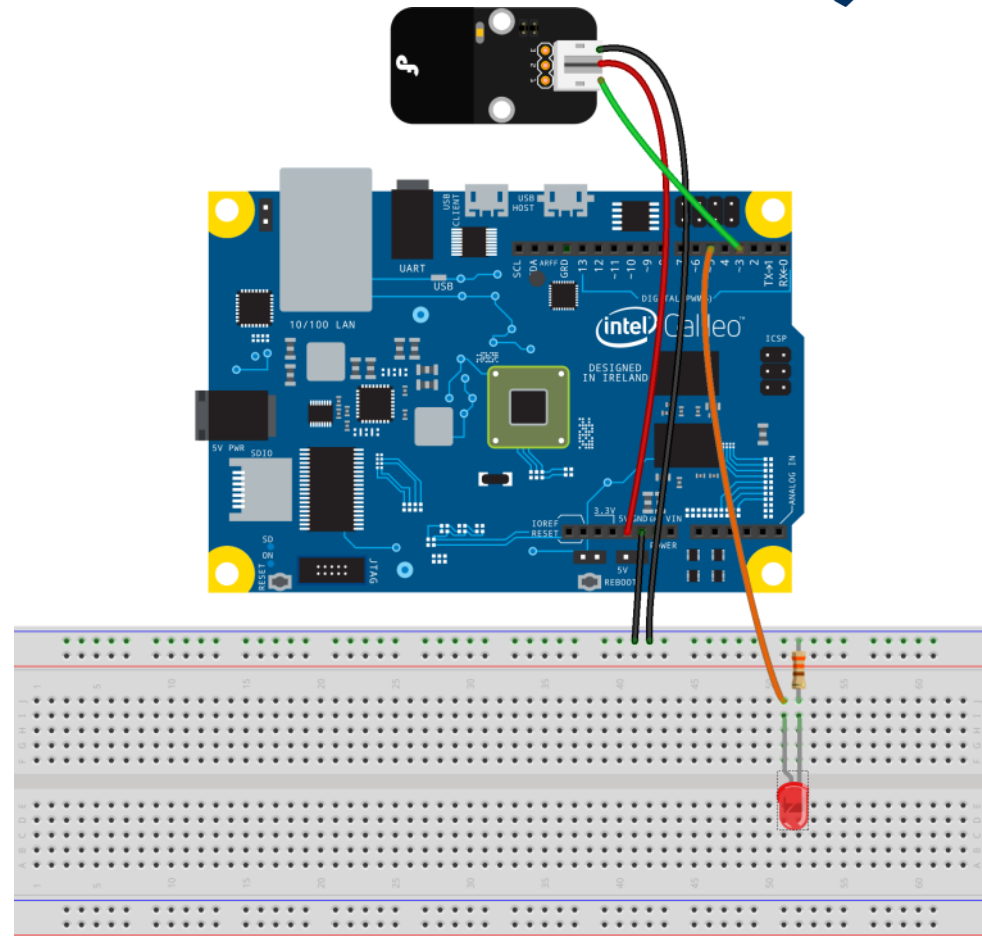
void setup()
{
    Serial.begin(115200);
    pinMode(TOUCH, INPUT);
}

void loop()
{
    value = digitalRead(TOUCH);
    if (value == 1 && state == 0)
    {
        state = 1;
        if (brightness >= PWM_MAX)
            brightness = 0;
        else
            brightness += step;

        analogWrite(LED, brightness);
    }
    else if (value == 0 && state == 1)
    {
        state = 0;
    }
}
```

Project Files: Lesson3-Sensing\Section4-Touch\touch\_sensor\_challenge

Challenge  
Review



fritzing

# Sensors – Tilt

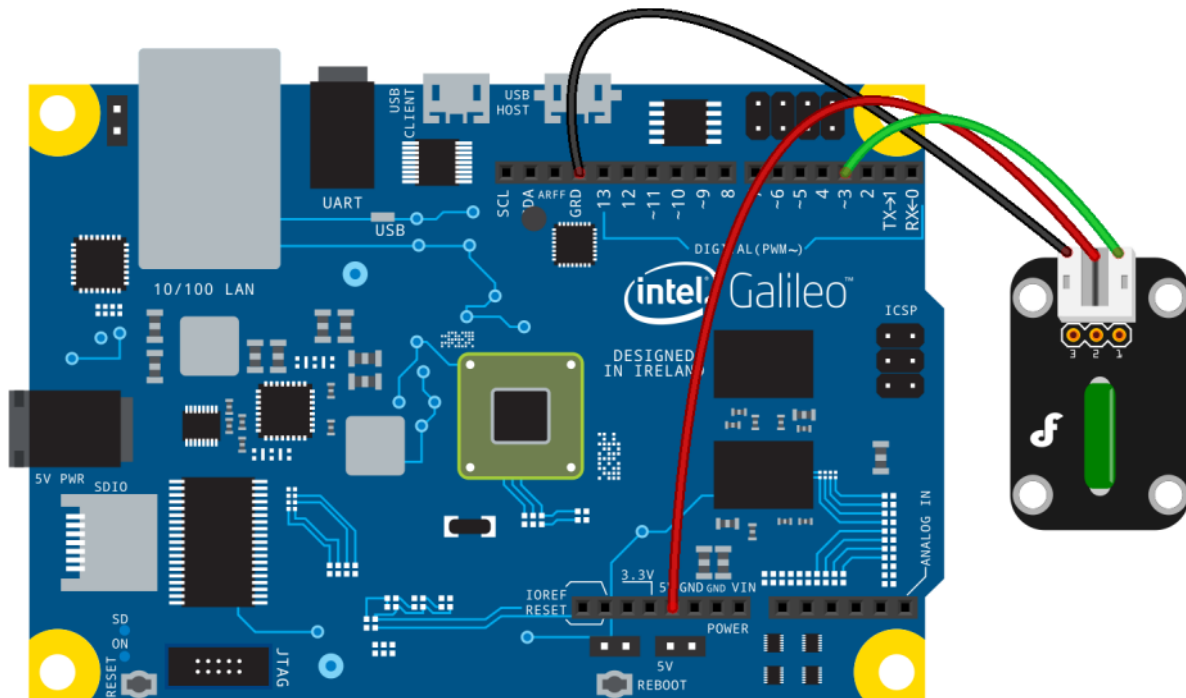


# Tilt sensor

- For this last sensor, you're going to be completely on your own. No code or explanation is provided for this sensor- see if you can figure out what it does and how to use it!

## Sensor Reference Page:

[http://www.dfrobot.com/index.php?route=product/product&product\\_id=77#.U5n10fk7uSo](http://www.dfrobot.com/index.php?route=product/product&product_id=77#.U5n10fk7uSo)



fritzing

Break Time 😊

# Sensors – Sensor Logger

# Sensor Logger – Touch enabled

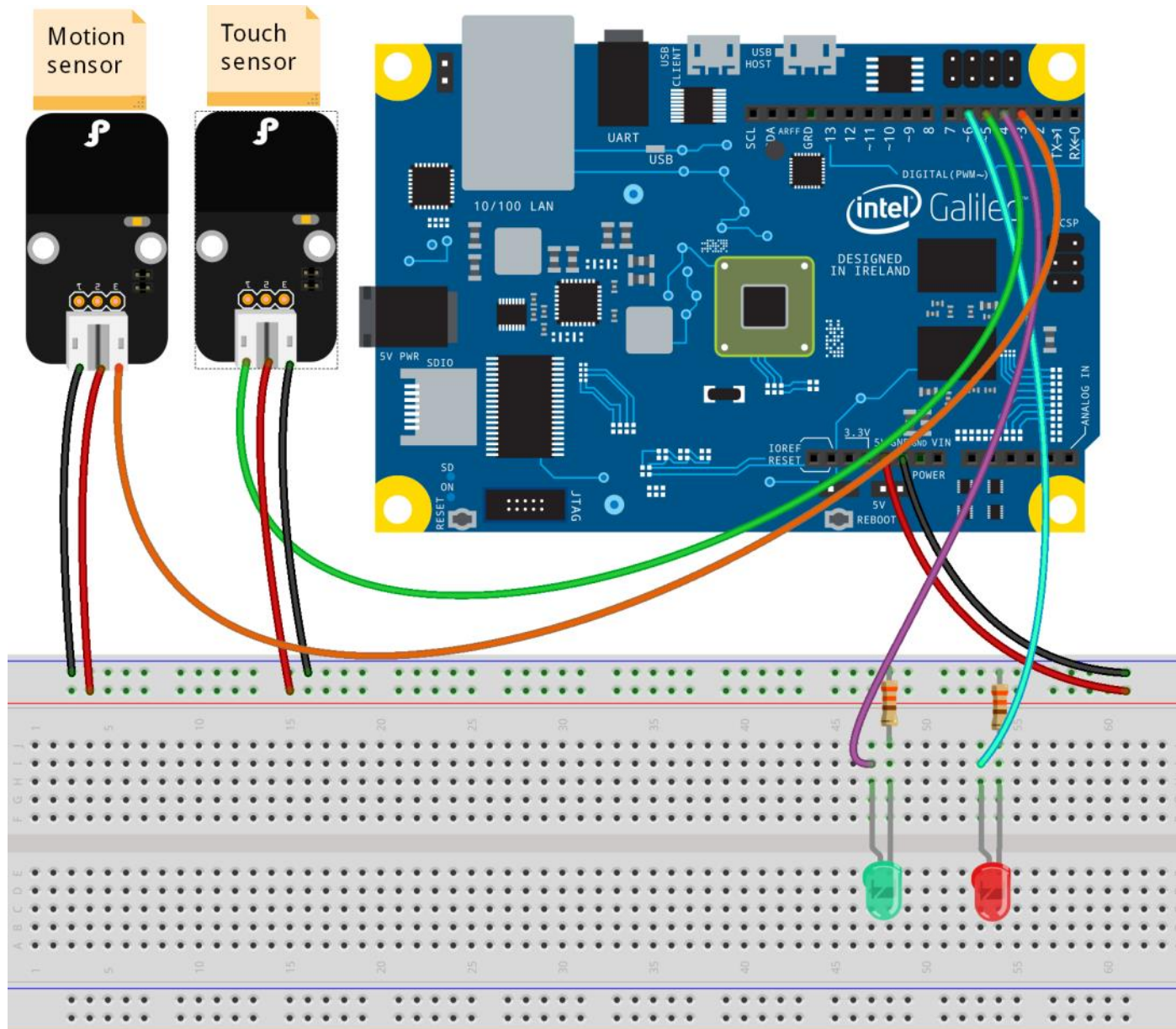


Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

## Challenge:

Build a motion detection system that is enabled/disabled through touch. When enabled it will log (record) all motion events to the SD card on your Galileo.



# Motion data logged to Galileo SD card

The image shows two windows from a computer screen. The left window is the Arduino IDE, titled 'sketch\_jun13a | Arduino 1.5.3'. It displays a C++ sketch for logging motion data to an SD card. The sketch includes headers for `<SD.h>`, defines pins for a PIR sensor (IN\_MOTION) and a touch sensor (IN\_TOUCH), and sets up an SD card. The `setup()` function initializes the serial port at 115200 baud and the SD card. The `init_SD()` function delays 3000ms before starting the SD card. The sketch is uploaded to an Intel Galileo on COM19, as indicated by the status bar. The right window is a PuTTY terminal titled 'COM80 - PuTTY'. It shows the output of the `cat /media/mmcblk0p1/datalog.txt` command, displaying a list of motion events with timestamps. Below the terminal, another window titled 'COM19' shows the serial output of the Arduino, including 'delete old logfile, new one created!', 'Alarm enabled', 'Motion logged' (repeated six times), and 'Alarm disabled'.

```
sketch_jun13a | Arduino 1.5.3
File Edit Sketch Tools Help

sketch_jun13a $
#include <SD.h>

#define MOTION 1
#define NOMOTION 0
#define IN_MOTION 3 //input from PIR sensor
#define OUT_TOUCH 4 //touch LED
#define IN_TOUCH 5 //input from touch sensor alarm on(1) or alarm off(0)
#define OUT_MOTION 6 //motion LED

short mvalue = 0;
short mstate = NOMOTION; //for debouncing PIR sensor
short tvalue = 0;
short tstate = 0; //for debouncing touch sensor
short alarm_enabled = 0; //alarm enabled or not

void setup()
{
  Serial.begin(115200);
  pinMode(IN_MOTION, INPUT);
  pinMode(OUT_TOUCH, OUTPUT);
  pinMode(IN_TOUCH, INPUT);
  pinMode(OUT_MOTION, OUTPUT);

  digitalWrite(OUT_TOUCH, LOW);
  digitalWrite(OUT_MOTION, LOW);
  init_SD();
}

void init_SD(void)
{
  delay(3000);
  //if (/SD begin/)

Done uploading.
Transfer complete

Intel® Galileo on COM19
```

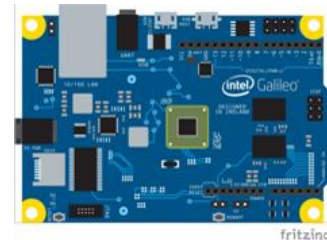
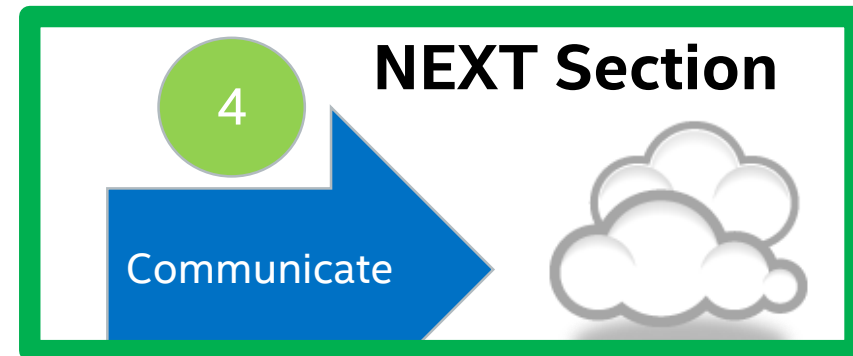
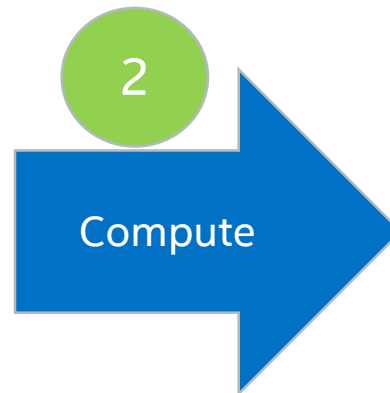
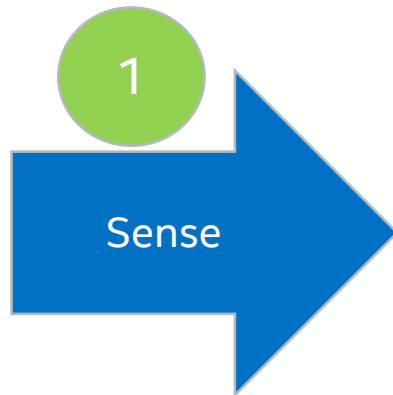
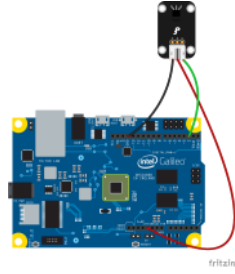
```
COM80 - PuTTY
root@clanton:~# cat /media/mmcblk0p1/datalog.txt
Mon Jan  1 00:43:18 UTC 2001
Motion on Mon Jan  1 00:43:20 UTC 2001
Motion on Mon Jan  1 00:43:23 UTC 2001
Motion on Mon Jan  1 00:43:24 UTC 2001
Motion on Mon Jan  1 00:43:29 UTC 2001
Motion on Mon Jan  1 00:43:30 UTC 2001
Motion on Mon Jan  1 00:43:31 UTC 2001
root@clanton:~#
root@clanton:~#
root@clanton:~#
root@clanton:~#
root@clanton:~#
root@clanton:~#
root@clanton:~#
root@clanton:~#
root@clanton:~#
root@clanton:~#

COM19
delete old logfile, new one created!
Alarm enabled
Motion logged
Motion logged
Motion logged
Motion logged
Motion logged
Motion logged
Motion logged
Alarm disabled

Autoscroll
No line ending
115200 baud
```

# Conclusion – We have covered 1, 2 and 3

## Sensing Motion



Congratulations 😊



# Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm> Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Celeron, Intel, Intel logo, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel SpeedStep, Intel XScale, Itanium, Pentium, Pentium Inside, VTune, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel® Active Management Technology requires the platform to have an Intel® AMT-enabled chipset, network hardware and software, as well as connection with a power source and a corporate network connection. With regard to notebooks, Intel AMT may not be available or certain capabilities may be limited over a host OS-based VPN or when connecting wirelessly, on battery power, sleeping, hibernating or powered off. For more information, see <http://www.intel.com/technology/iamt>.

64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.

No computer system can provide absolute security under all conditions. Intel® Trusted Execution Technology is a security technology under development by Intel and requires for operation a computer system with Intel® Virtualization Technology, an Intel Trusted Execution Technology-enabled processor, chipset, BIOS, Authenticated Code Modules, and an Intel or other compatible measured virtual machine monitor. In addition, Intel Trusted Execution Technology requires the system to contain a TPMv1.2 as defined by the Trusted Computing Group and specific software for some uses. See <http://www.intel.com/technology/security/> for more information.

Hyper-Threading Technology (HT Technology) requires a computer system with an Intel® Pentium® 4 Processor supporting HT Technology and an HT Technology-enabled chipset, BIOS, and operating system. Performance will vary depending on the specific hardware and software you use. See [www.intel.com/products/ht/hyperthreading\\_more.htm](http://www.intel.com/products/ht/hyperthreading_more.htm) for more information including details on which processors support HT Technology.

Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, virtual machine monitor (VMM) and, for some uses, certain platform software enabled for it. Functionality, performance or other benefits will vary depending on hardware and software configurations and may require a BIOS update. Software applications may not be compatible with all operating systems. Please check with your application vendor.

\* Other names and brands may be claimed as the property of others.

Other vendors are listed by Intel as a convenience to Intel's general customer base, but Intel does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of these devices. This list and/or these devices may be subject to change without notice.

Copyright © 2014, Intel Corporation. All rights reserved.



# Backup Section

# Piezo debouncing- Engineering Challenge



Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

Challenge:

Build a circuit with **two** piezo sensors, and maintain **two** timers to keep track of hits.

# Engineering Challenge Review

## Challenge 1: detect hits from two piezo elements

Challenge  
Review

```
#define THRESH 300
//values higher than this value register a 'hit'

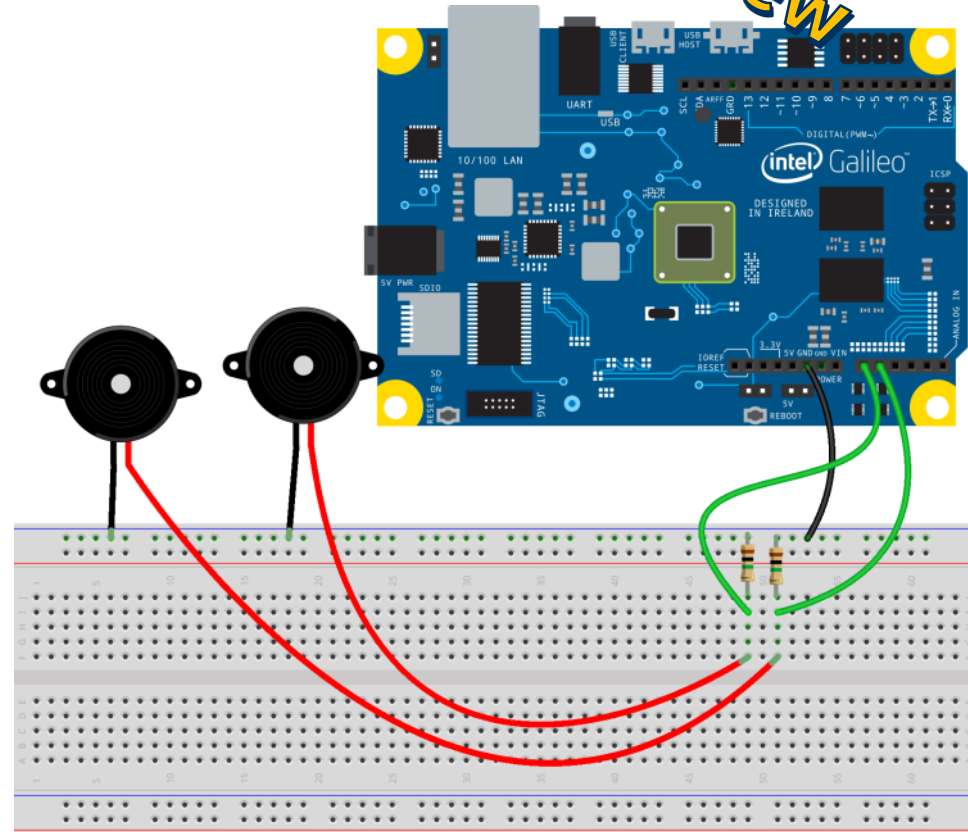
#define WAIT 150
//how long we wait before reading again

int value;
long timer[2] = {0, 0};

void setup() {
  Serial.begin(115200);
}

void loop() {
  for (int i = 0; i < 2; i++)
  {
    value = analogRead(i);

    if (value > THRESH && timer[i] == 0)
    {
      timer[i] = millis();
      Serial.println(i);
    }
    else if ((millis() - timer[i]) >= WAIT && value < THRESH)
    {
      timer[i] = 0;
    }
  }
}
```



fritzing

Project Files: Lesson3-Sensing\Section3-Vibration\vibration\_sensor\_challenge