

Lecture 6: Working with registers

MPU5060 Demo

Vikram Iyer

Administrative

April				
Monday	Tuesday	Wednesday	Thursday	Friday
9:30-11:30 OH (Deeksha) 08 ECE 345 11:30-13:30 OH (Alex) ECE 345	13:00-15:00 OH (Zach) 09 ECE 345	9:30-11:30 OH (Deeksha) 10 ECE 345 12:30-14:20 Lecture MOR 230 <i>Lecture 5: Hardware and Machine Organization</i> 14:00-15:00 OH (Vikram) ECE 345	10:00-12:00 OH (Zach) 11 ECE 345 12:30-14:30 OH (Alex) ECE 345	12:30-14:20 Lecture 12 MOR 230 <i>Lecture 6: Working with Registers and IMU Demo</i> 14:00-15:00 OH (Vikram) ECE 345 23:59 C Programming 1 due
9:30-11:30 OH (Deeksha) 15 ECE 345 11:30-13:30 OH (Alex) ECE 345	13:00-15:00 OH (Zach) 16 ECE 345	9:30-11:30 OH (Deeksha) 17 ECE 345 12:00-13:00 Remote OH (Vikram) <i>Zoom</i>	10:00-12:00 OH (Zach) 18 ECE 345 12:30-14:30 OH (Alex) ECE 345 23:59 Lab 1 due	12:30-14:20 Lecture 19 MOR 230 <i>Lecture 7: ATmega 2560 Datasheet and Timers</i> 14:00-15:00 OH (Vikram) ECE 345
9:30-11:30 OH (Deeksha) 22 ECE 345 11:30-13:30 OH (Alex) ECE 345	13:00-15:00 OH (Zach) 23 ECE 345	9:30-11:30 OH (Deeksha) 24 ECE 345 12:30-14:20 Lecture MOR 230 <i>Lecture 8: Reading Analog Data and Intro to Interrupts</i> 14:00-15:00 OH (Vikram) ECE 345	10:00-12:00 OH (Zach) 25 ECE 345 12:30-14:30 OH (Alex) ECE 345 23:59 C Programming 2 due	12:30-14:20 Lecture 26 MOR 230 <i>Lecture 9: Tasks, Threads, Scheduling I, Interrupts</i> 14:00-15:00 OH (Vikram) ECE 345

Last time

- Hardware architecture- what's inside a processor?
 - Registers
 - Data bus and signaling
 - How a processor works
 - Example of code execution
- Getting started with Arduino

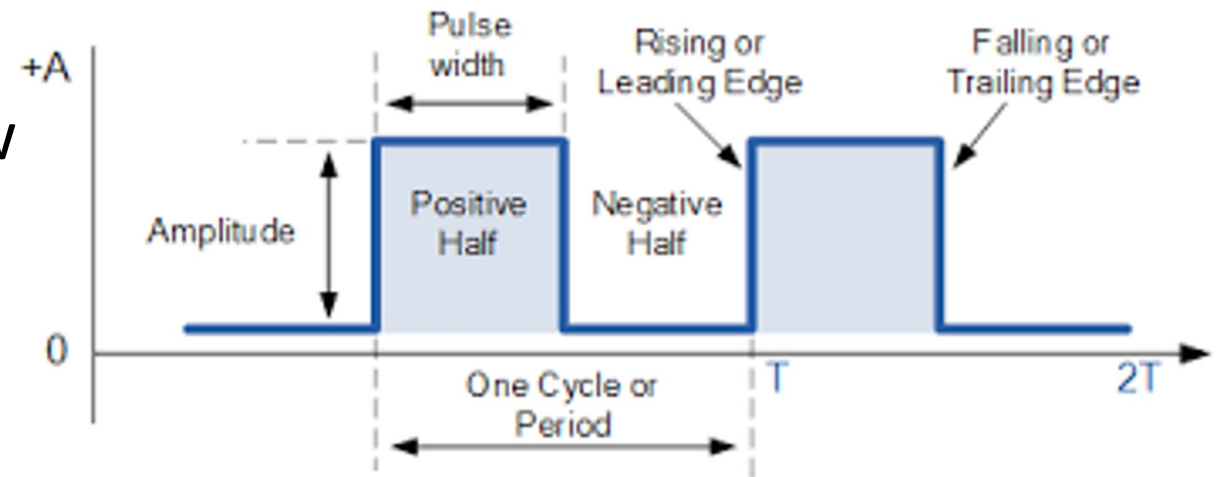
Plan for today

- Lab 1 background
 - Frequency
 - Timing functions
 - Brief intro to circuits
- Working with registers: MPU6050 demo
 - What is an IMU and how does it work?
 - Reading datasheets
 - Demo of reading and writing registers to control sensor

Lab 1 hints: Frequency and period

- Frequency = $1/\text{Period}$
- Similarly, Period = $1/\text{Frequency}$
 - Example: a 440 Hz frequency wave has a period of $1/440$ seconds = 2.27 ms period.

- Period for square wave: high-low



Lab 1 hints: Handy Arduino functions

- Do NOT use `tone()`: hard to manage 2 different frequencies simultaneously, can cause issues
- `delay(1)`: Pause the program by 1 millisecond
 - If called for every loop, pauses 1 ms after execution
 - ATmega2560 clock speed = 16MHz, 1 ms is slow comparatively

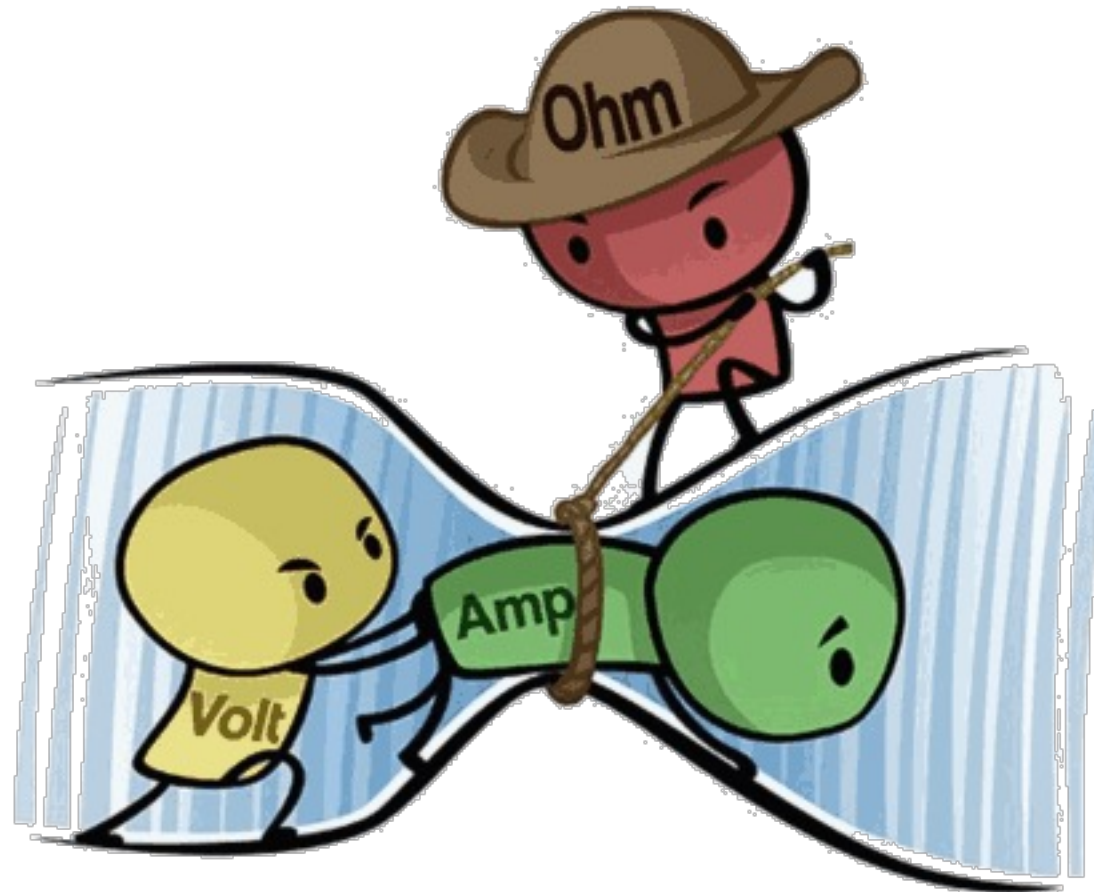
Lab 1 hints: Handy Arduino functions

- Do NOT use `tone()`: hard to manage 2 different frequencies simultaneously, can cause issues
- `delay(1)`: Pause the program by 1 millisecond
 - If called for every loop, pauses 1 ms after execution
 - ATmega2560 clock speed = 16MHz, 1 ms is slow comparatively
- `millis()`: returns the current time in milliseconds - relative to the system
 - What does `'millis() % 1000'` do when called every loop?
 - Return type: `unsigned long`
 - What is the maximum value we can get?

Lab 1 hints: Handy Arduino functions

- Do NOT use `tone()`: hard to manage 2 different frequencies simultaneously, can cause issues
- `delay(1)`: Pause the program by 1 millisecond
 - If called for every loop, pauses 1 ms after execution
 - ATmega2560 clock speed = 16MHz, 1 ms is slow comparatively
- `millis()`: returns the current time in milliseconds - relative to the system
 - What does '`millis() % 1000`' do when called every loop?
 - Return type: `unsigned long`
 - What is the maximum value we can get?
 - $(2^{32} - 1) = 4,294,967,295$ ms = around 49.7 days

A very brief circuits review/overview



A very brief circuits review/overview

Quantity	Symbol	Unit	Abbreviation
Current	I	Ampere (Amp)	A
Voltage	V	Volt	V
Resistance	R	Ohm	Ω

Ohm's Law:

$$V=I*R$$

A very brief circuits review/overview

Quantity	Symbol	Unit	Abbreviation
Current	I	Ampere (Amp)	A
Voltage	V	Volt	V
Resistance	R	Ohm	Ω

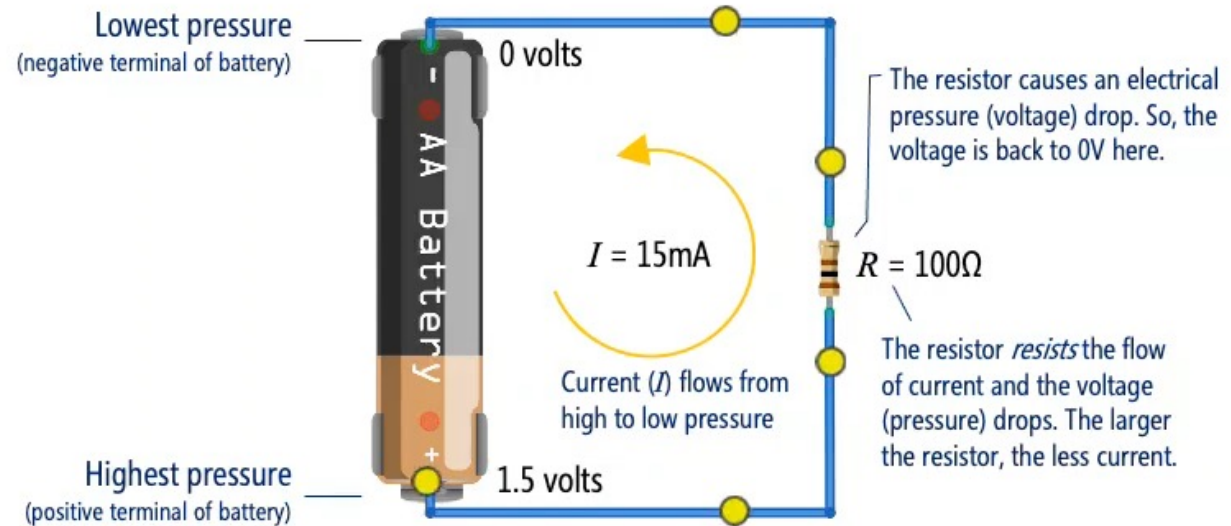
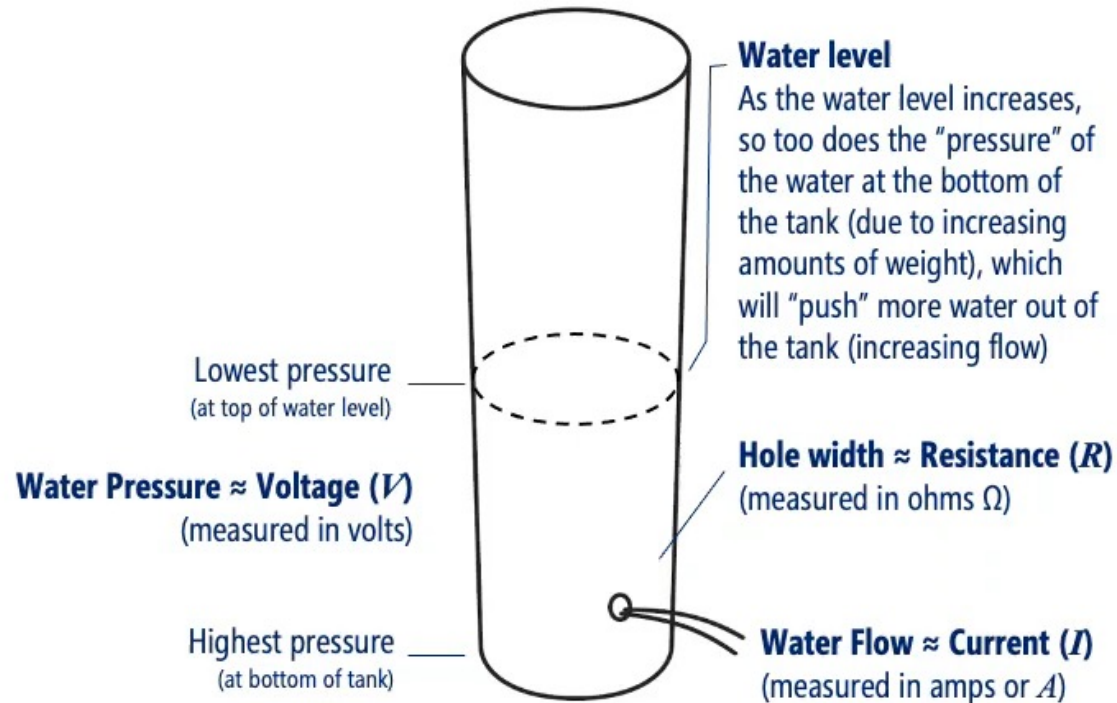
Ohm's Law:

$$V=I*R$$

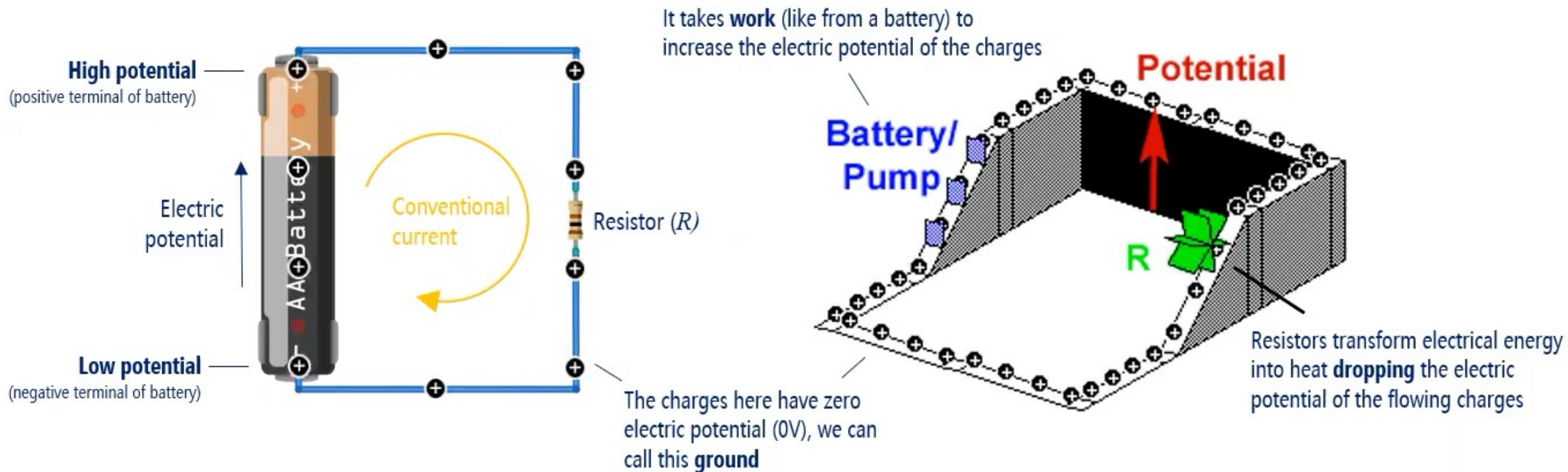
Water flow (hydraulic) analogy

	Electric	Hydraulic
Flow rate	Current, <i>amps (coloumbs/sec)</i>	Flow rate, <i>GPM (gallons/minute)</i>
Potential	Voltage, <i>volts</i>	Pressure, <i>psi (pound per square inch)</i>
Resistance	Resistance, <i>ohm (volts/amp)</i>	Resistance, <i>psi/gpm</i>

HYDRAULIC—ELECTRIC ANALOGY



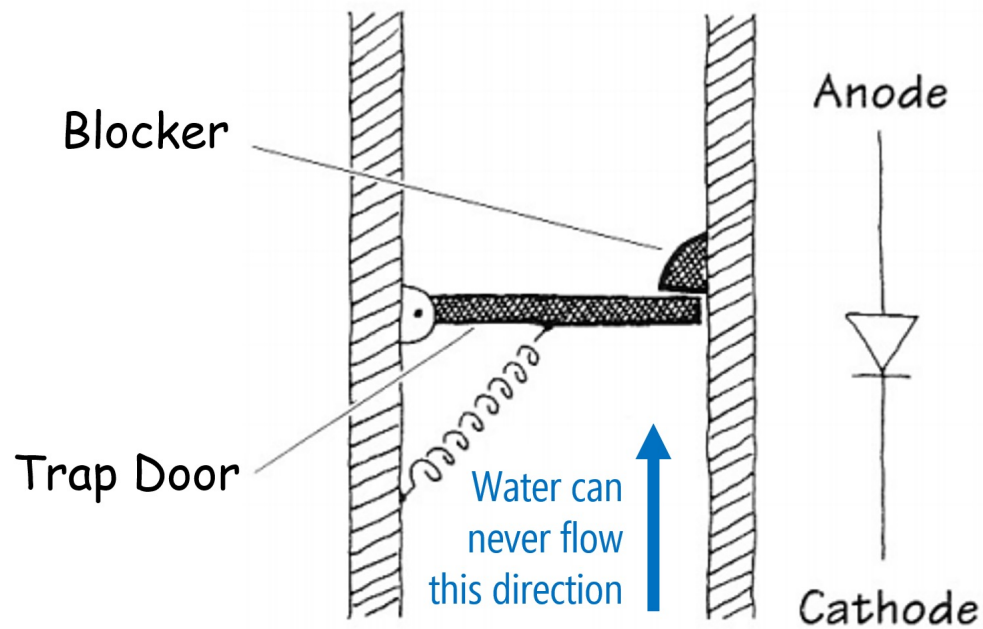
$1 \text{ V} = 1 \text{ joule (of work)} / 1 \text{ coulomb (of charge)}$



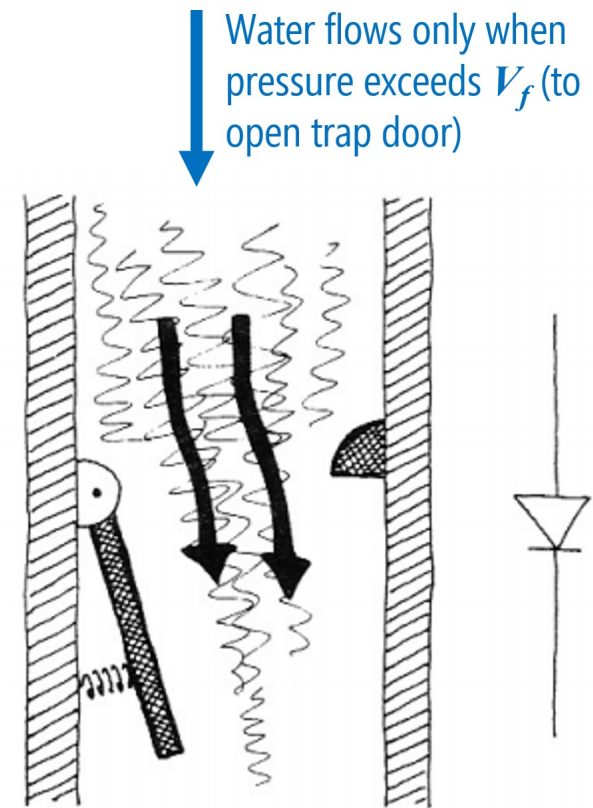
Power = work / time, $1 \text{ W} = 1 \text{ joule} / \text{second}$

Watts = **Voltage** (joules / coulomb) x **Current** (coulombs / second)

Diodes

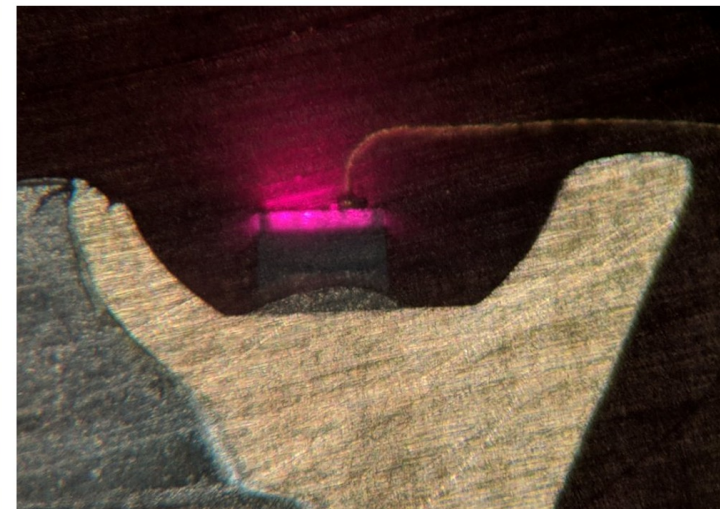
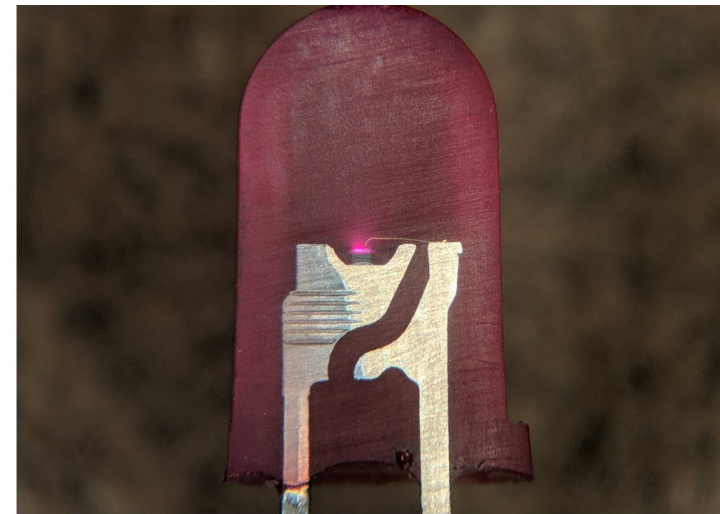
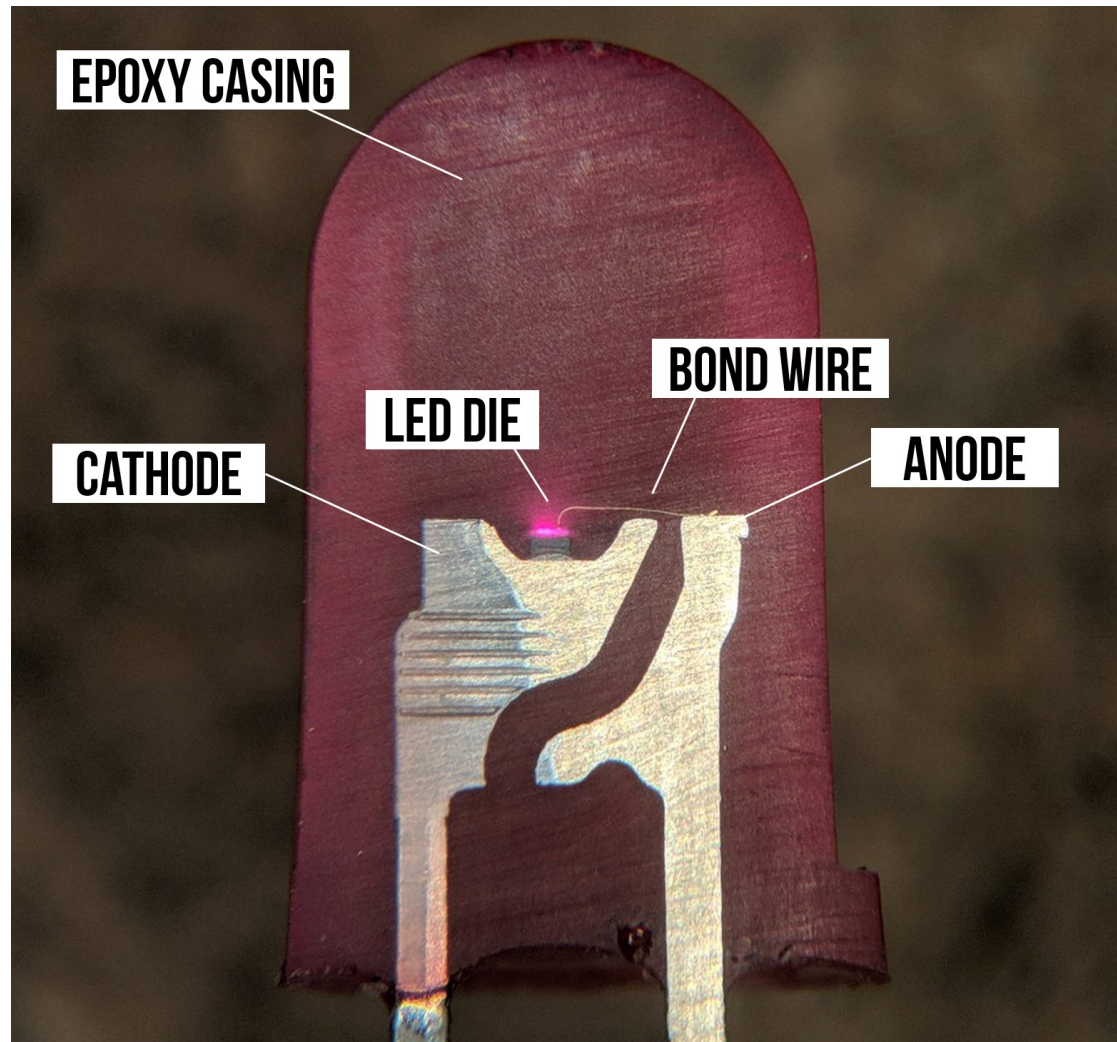


Imagine this trap door won't open unless a certain water pressure threshold is met (this is the forward voltage V_f , which is sometimes called the "on" voltage V_{on}).



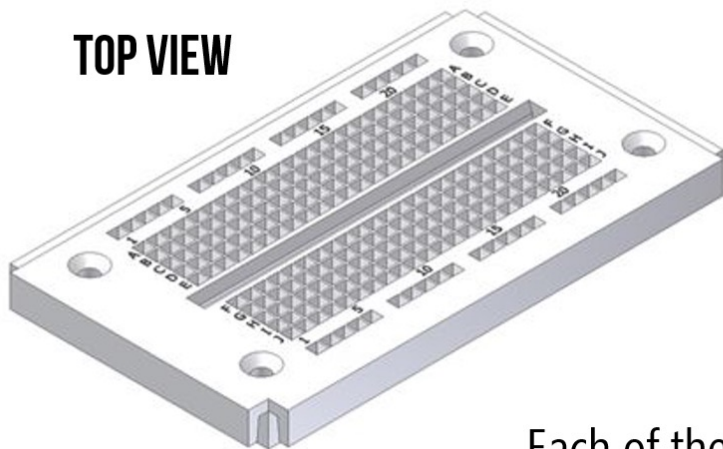
Once the V_f threshold is met, the trap door opens and water (current) passes through.

Light Emitting Diodes (LEDs)

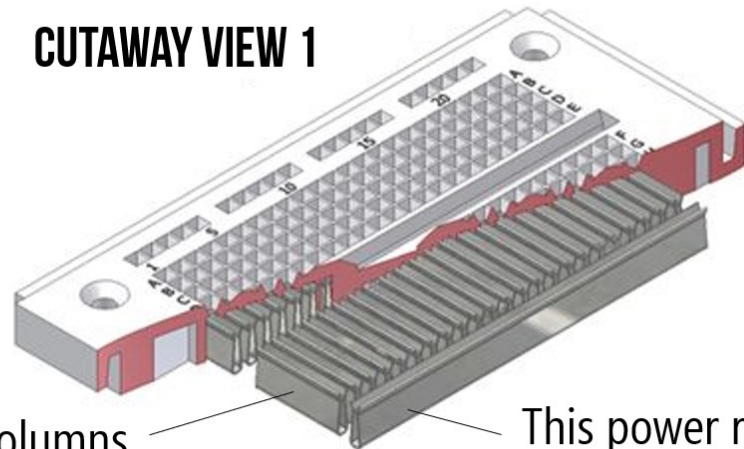


Prototyping with breadboards

TOP VIEW



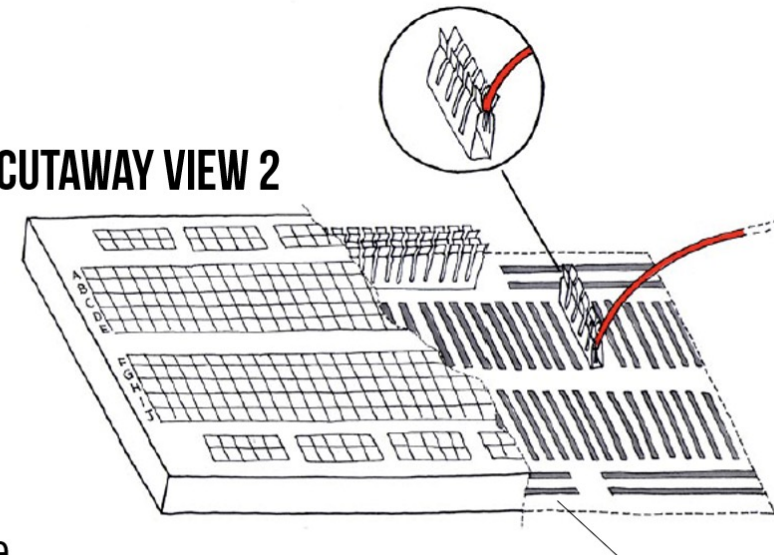
CUTAWAY VIEW 1



Each of these columns are independent **nodes**

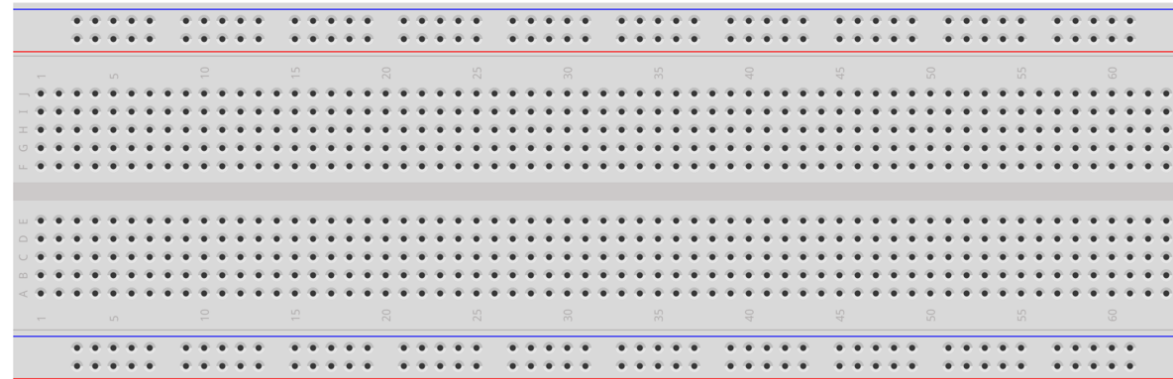
This power rail is one giant wire (or **node**)

CUTAWAY VIEW 2

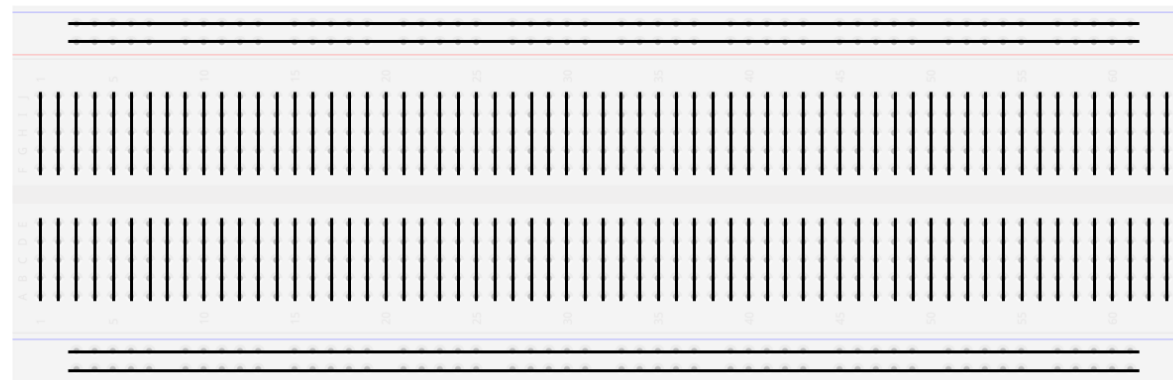


This breadboard has **two power rails** on each side

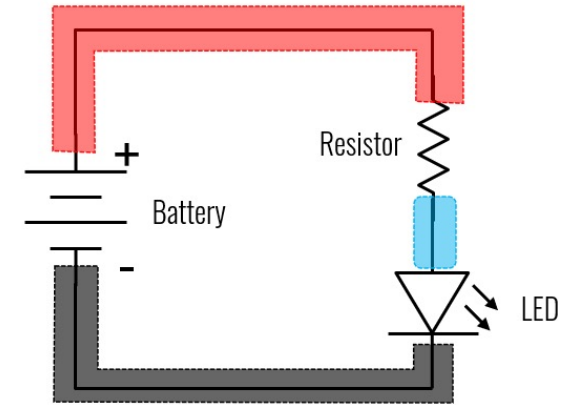
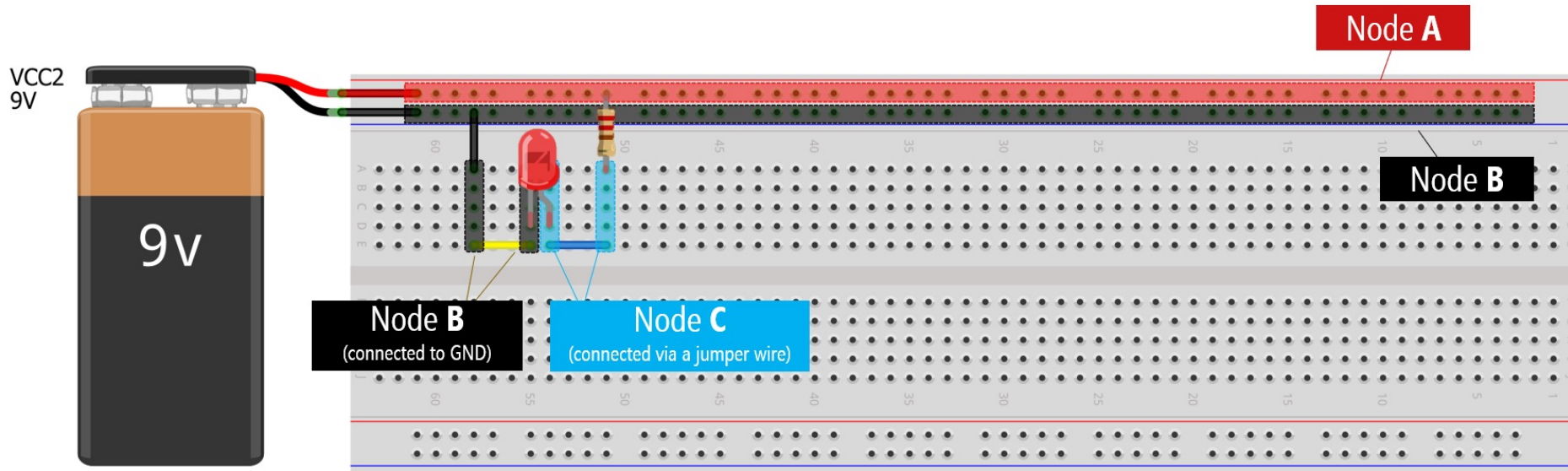
Prototyping with breadboards



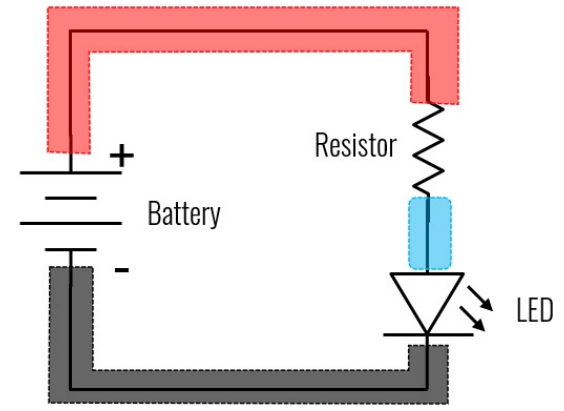
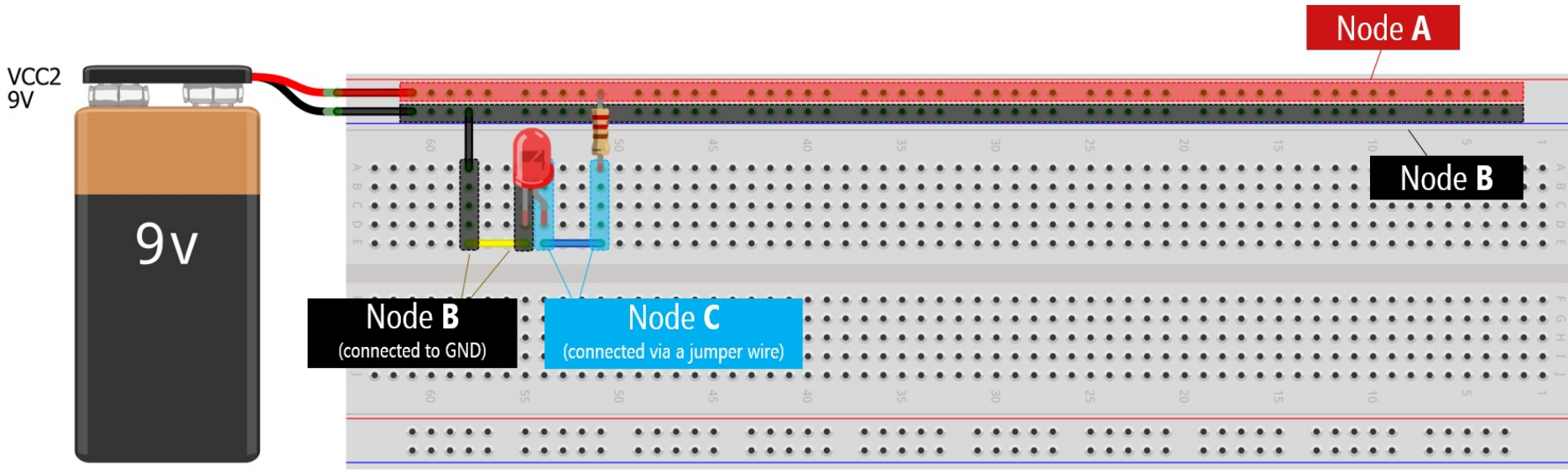
Internal connections



Prototyping with breadboards



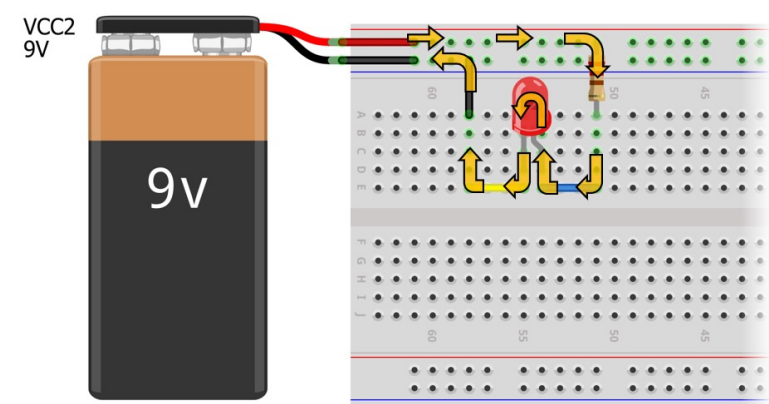
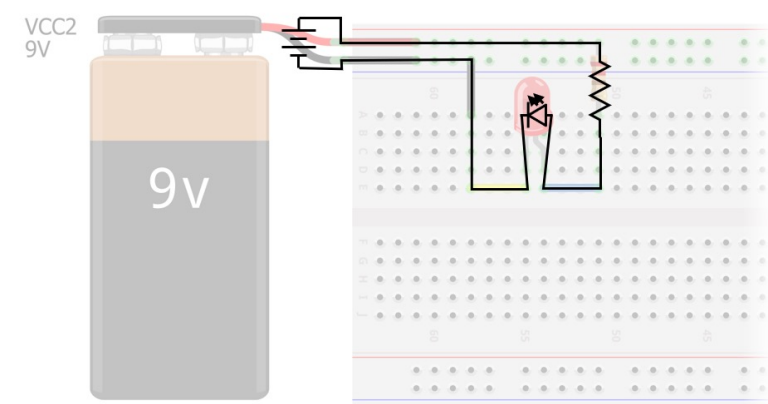
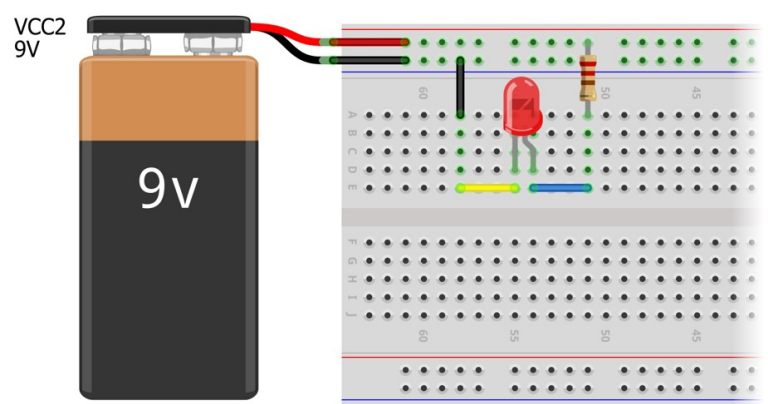
Prototyping with breadboards



SIMPLE BREADBOARDED LED CIRCUIT

UNDERLYING CIRCUIT AND CONNECTIONS

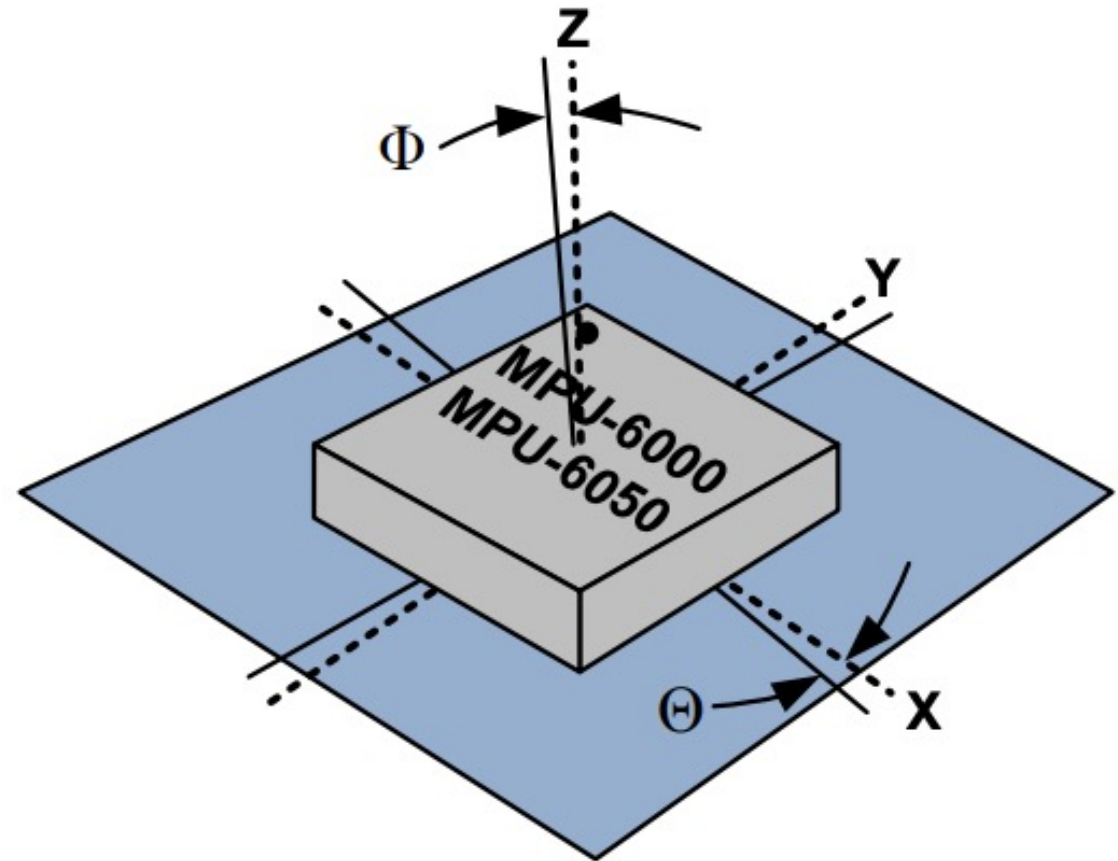
CURRENT FLOW THROUGH BREADBOARD



MPU6050: Inertial measurement unit (IMU)

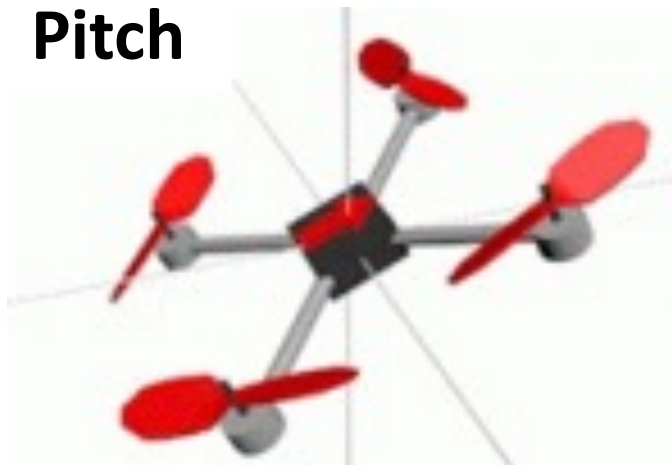
Includes:

- 3 axis Accelerometer- sensor that measures acceleration
- 3 axis Gyroscope- sensor that measures angular acceleration
- Temperature sensor



IMU use case 1: Drones and flying robots

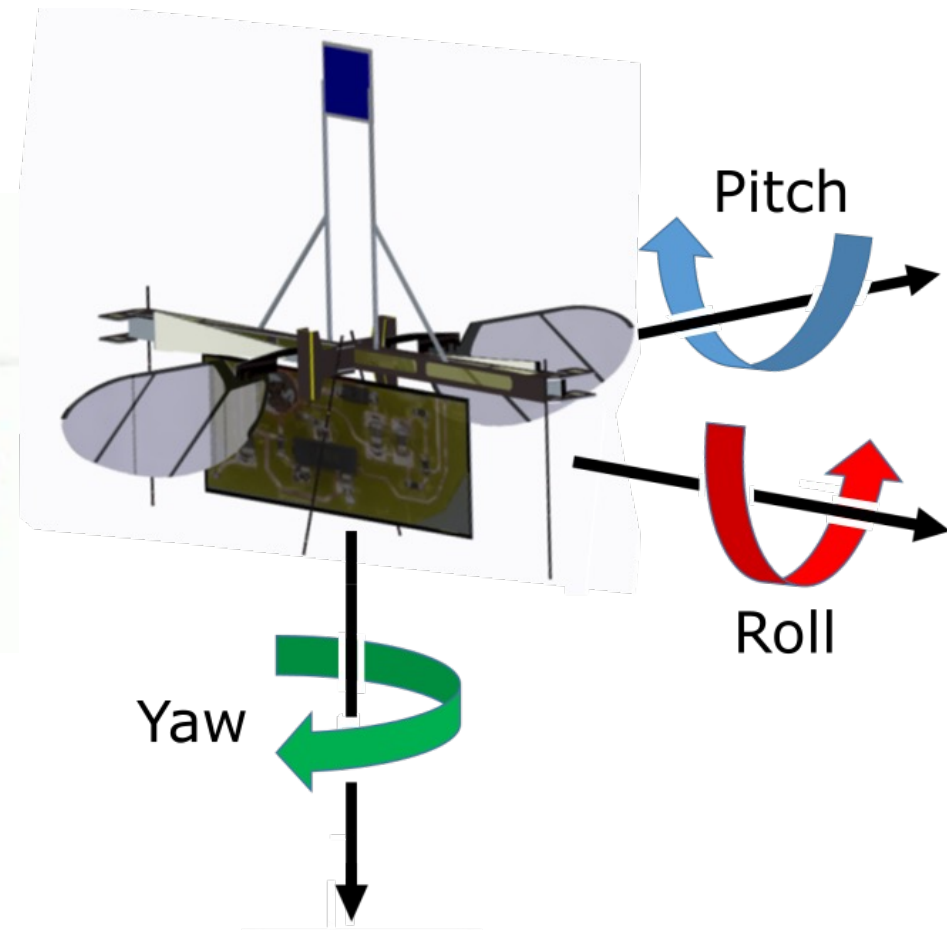
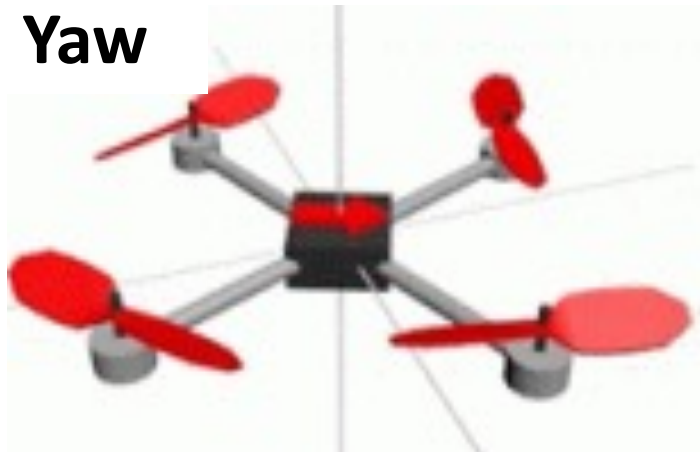
Pitch



Roll



Yaw



IMU use case 2: Tracking hummingbirds



Alejo Rico-Guevera

DEPARTMENT
OF BIOLOGY

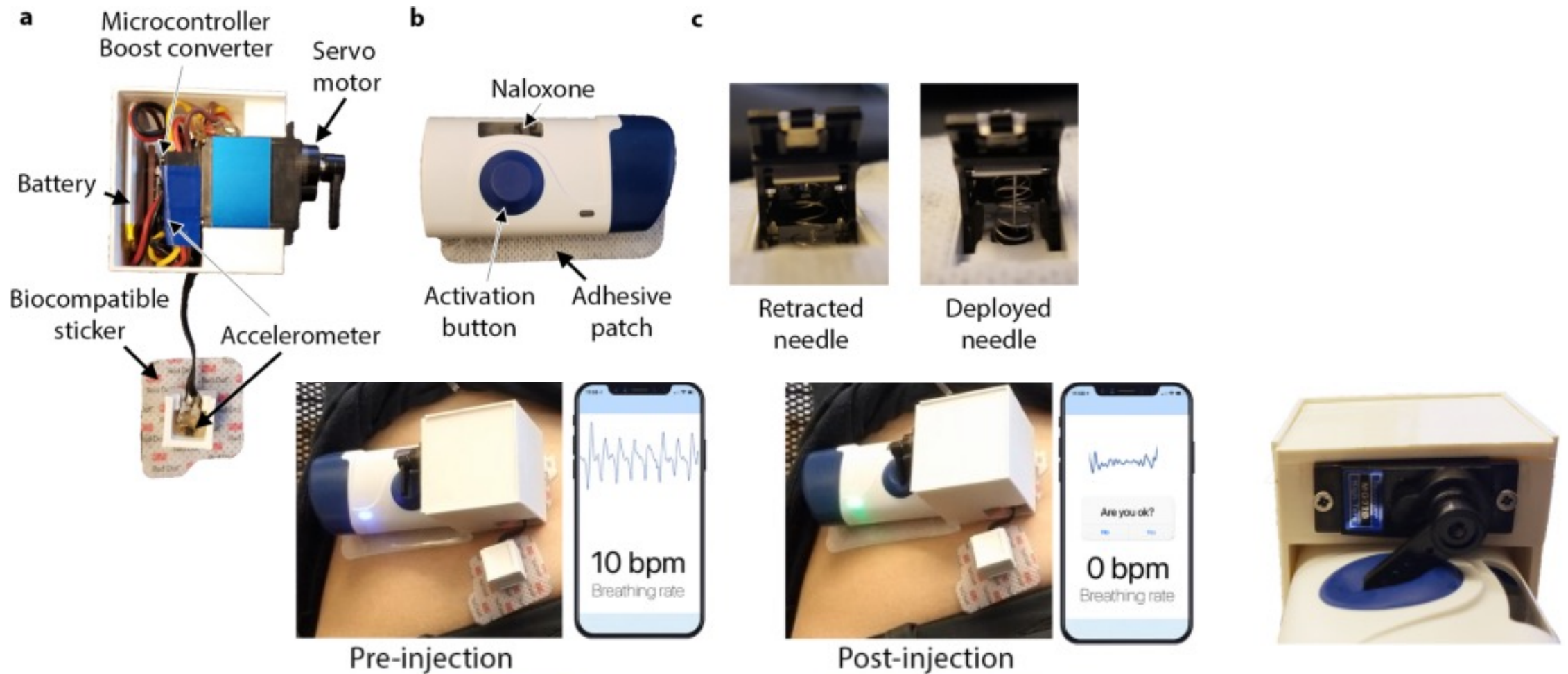


Alyssa Sargent



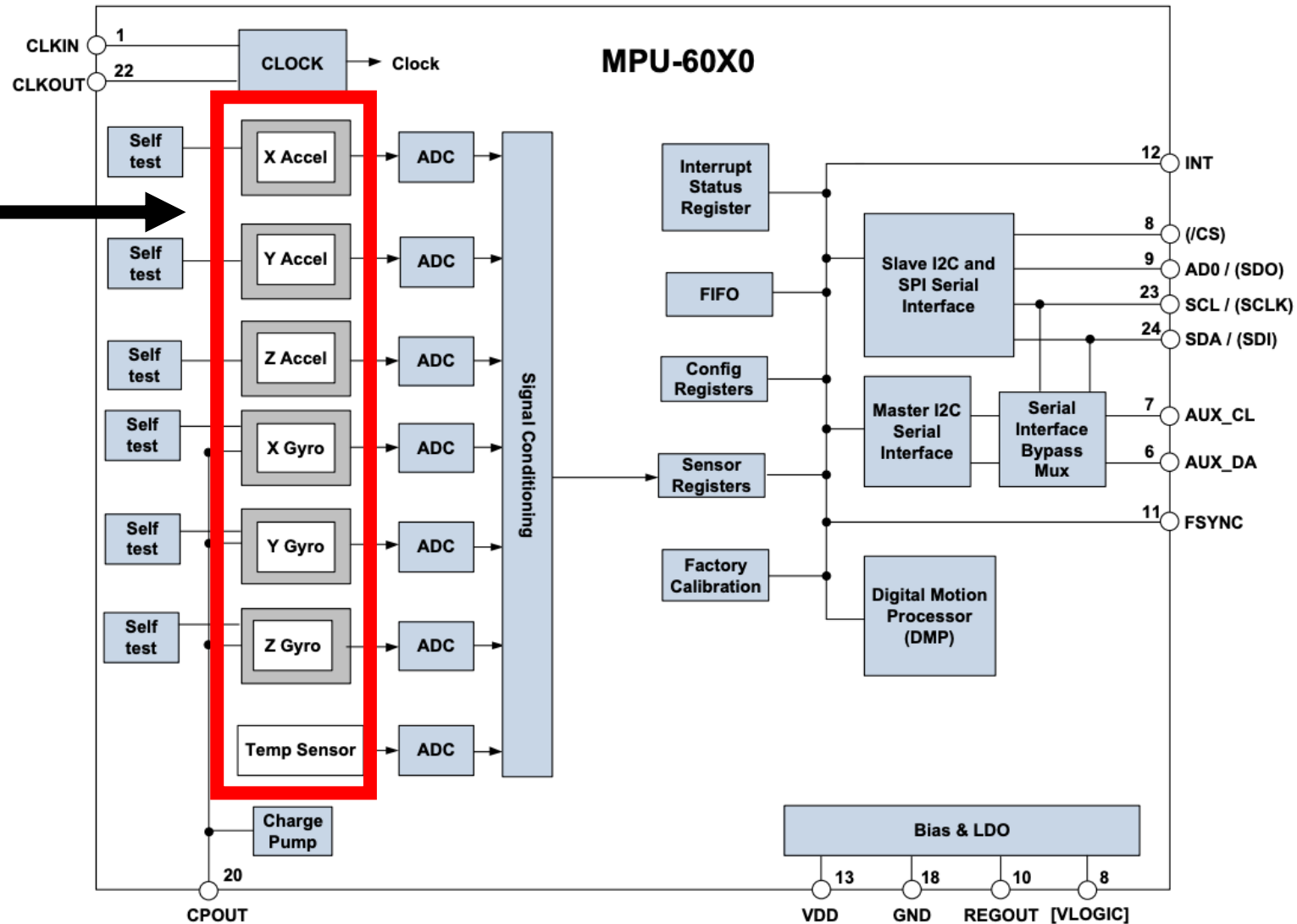
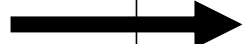
Yash Talwekar

IMU use case 3: Measuring breathing



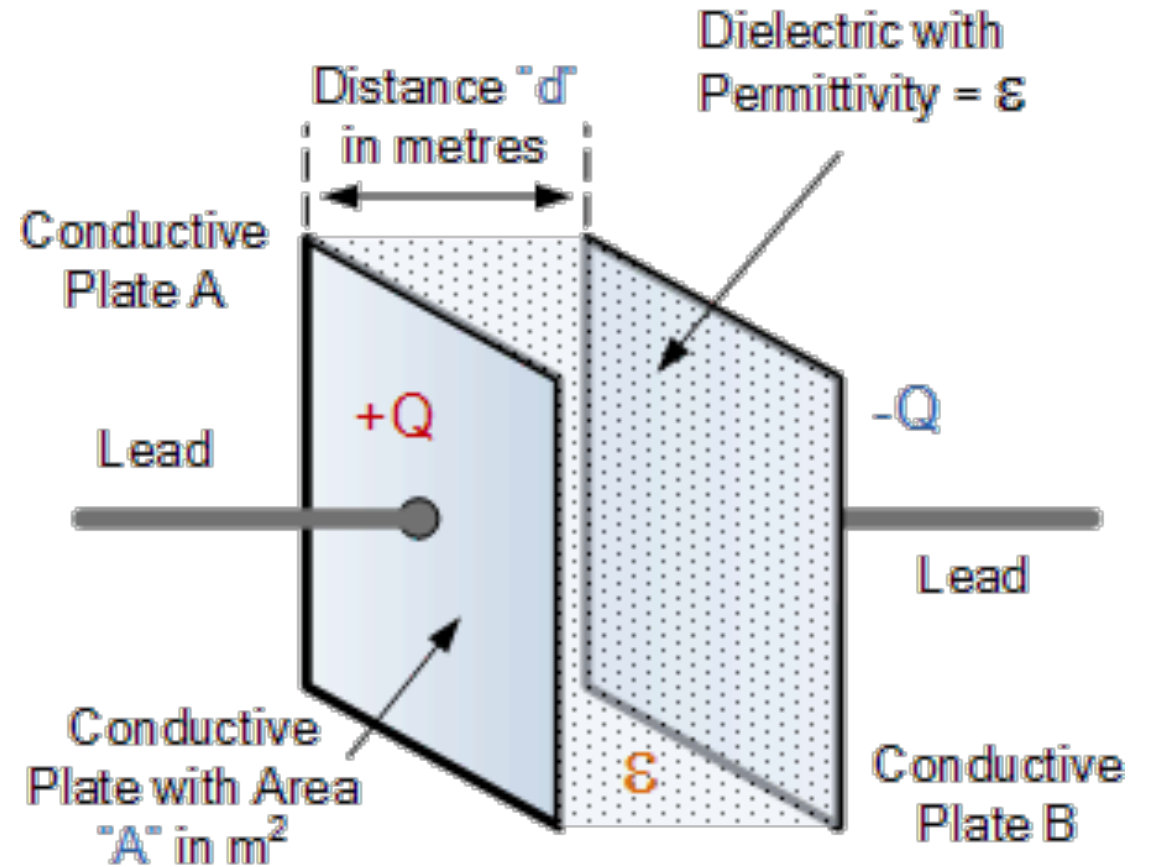
What's inside the chip?

Physical sensors

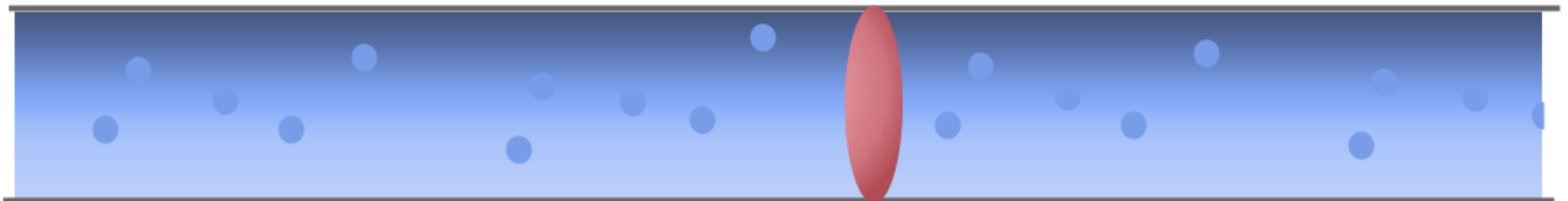


Capacitors

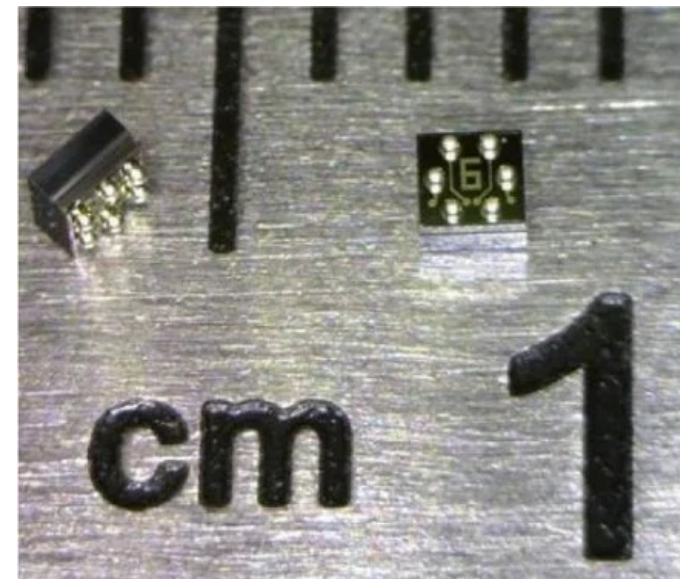
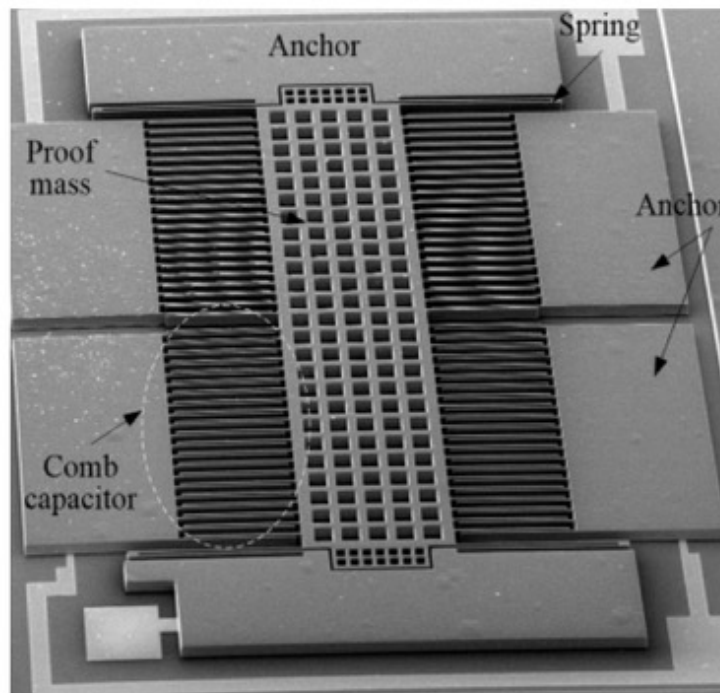
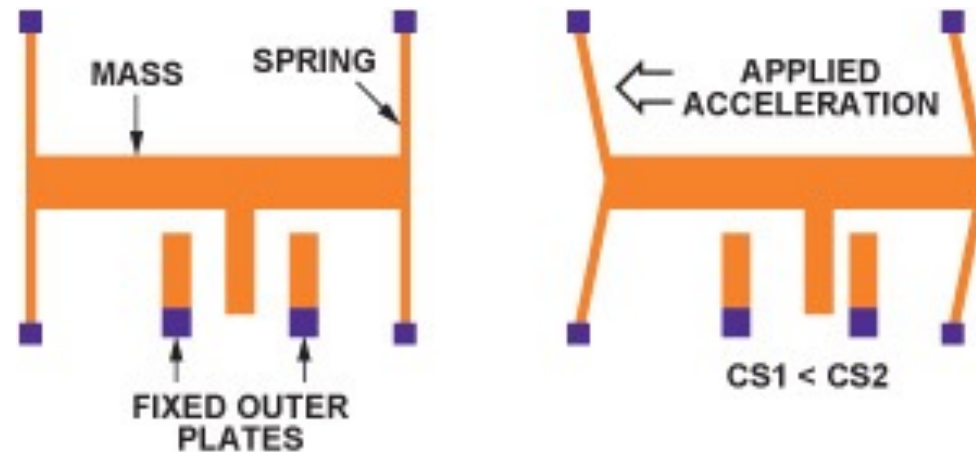
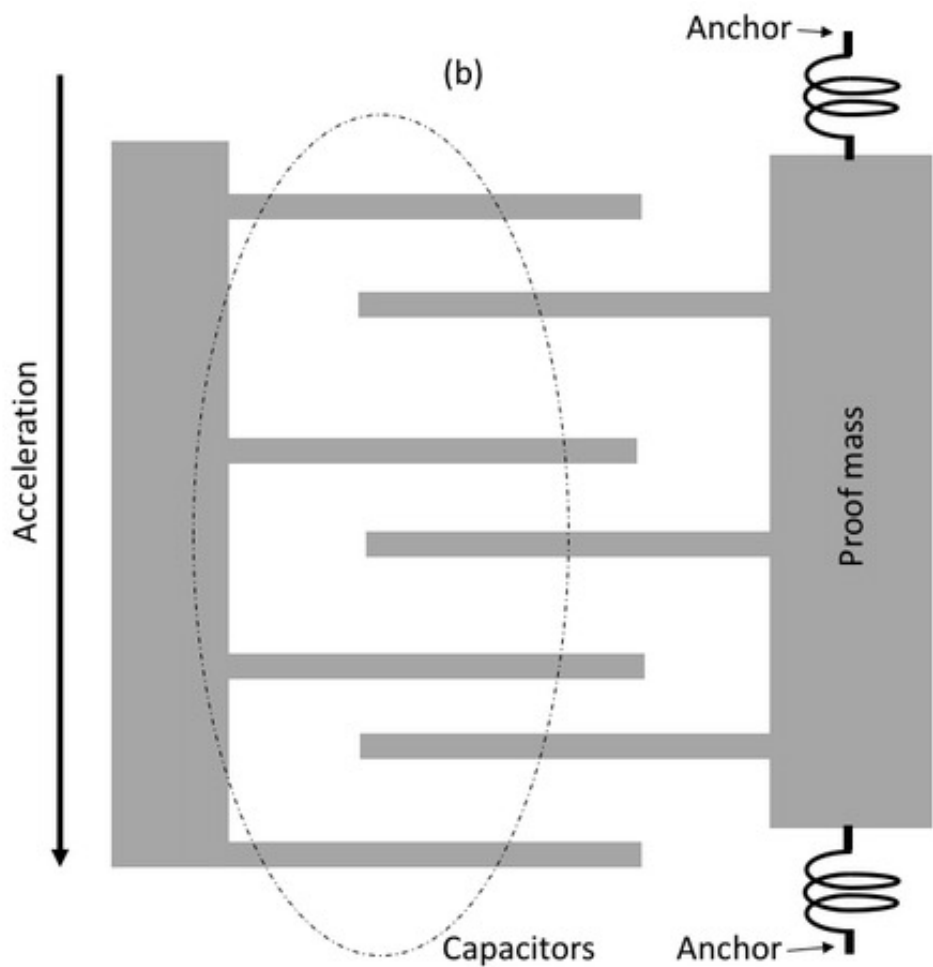
$$C = \frac{\epsilon A}{d}$$



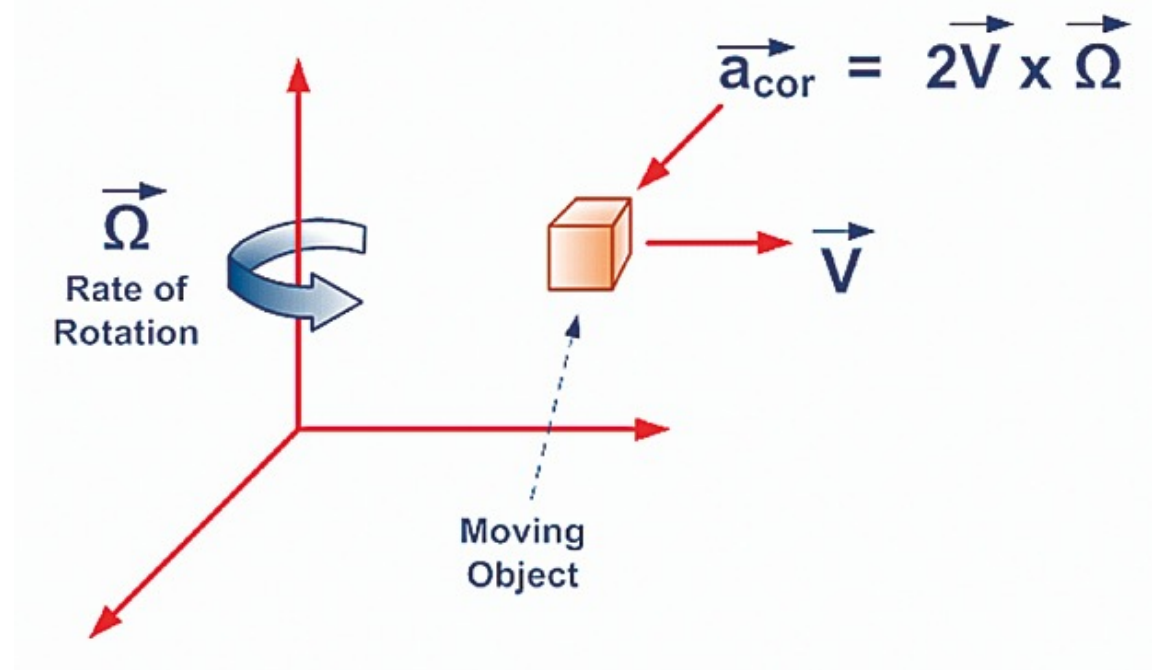
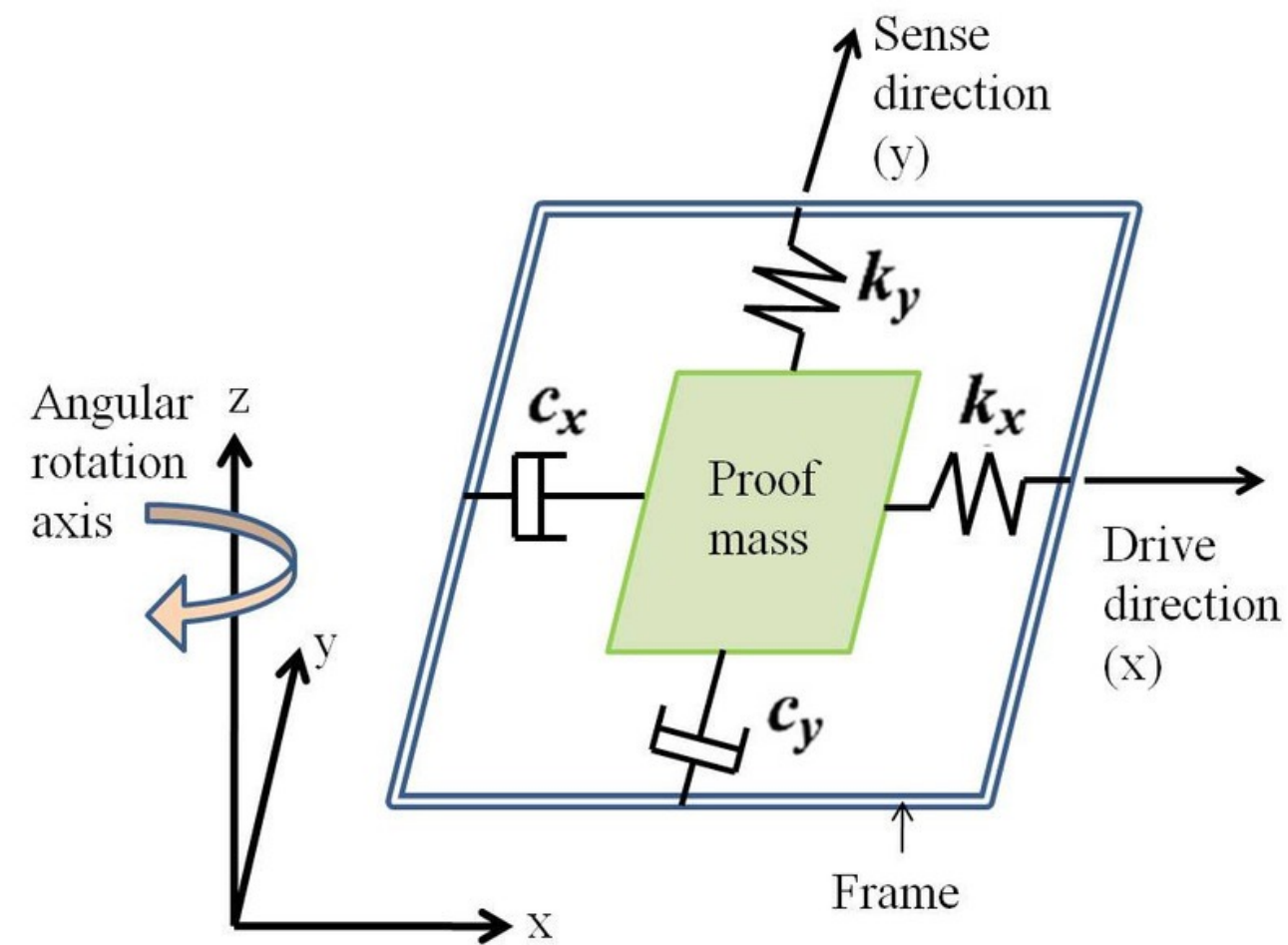
Analogy: rubber membrane



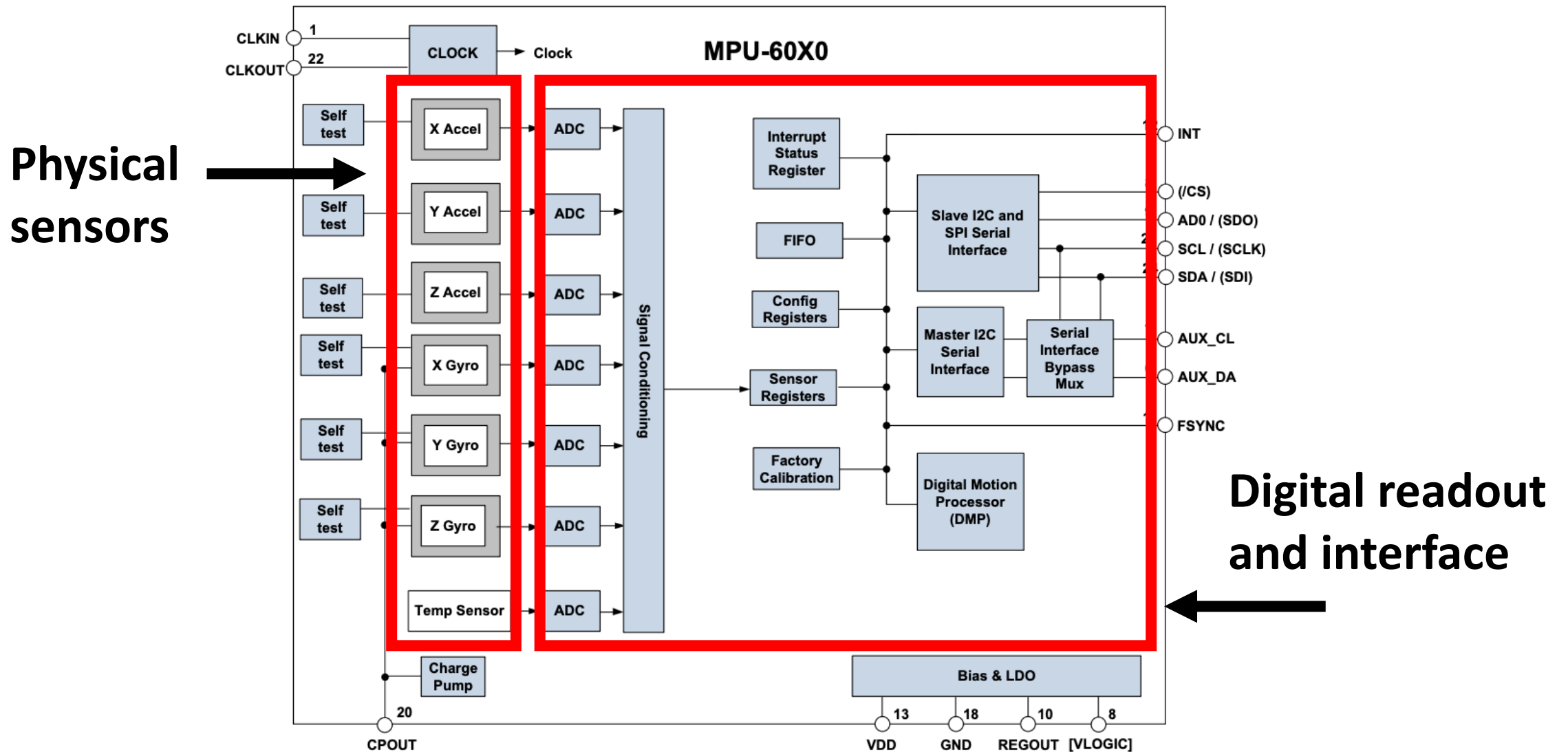
MEMS Accelerometer



MEMS Gyroscope



What's inside the chip?

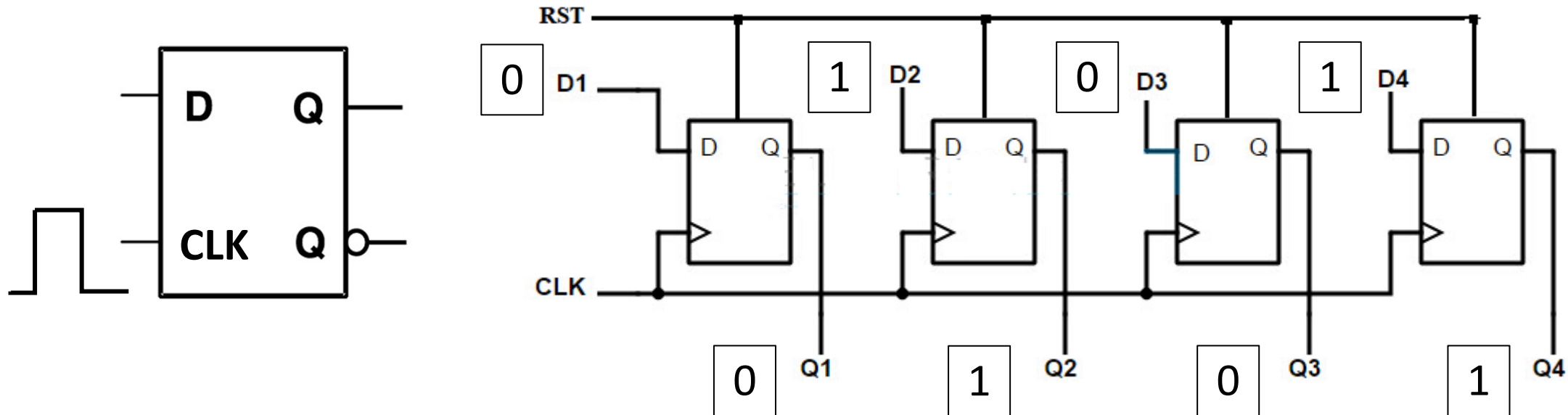


Registers

Register- an array of D flip-flops which can store a collection of bits

An n bit register has n inputs, n outputs, and one clock line

Flip flop: Memory element that stores 1 bit



Hardware control registers



ON

OFF

Hardware control registers

ON

OFF



Fcn 7	Fcn 6	Fcn 5	Fcn 4	Fcn 3	Fcn 2	Fcn 1	Fcn 0
7	6	5	4	3	2	1	0

MPU6050 Register table

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

MPU6050 Register table

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

4.28 Register 107 – Power Management 1

PWR_MGMT_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Description:

This register allows the user to configure the power mode and clock source. It also provides a bit for resetting the entire device, and a bit for disabling the temperature sensor.

4.28 Register 107 – Power Management 1

PWR_MGMT_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Parameters:

- DEVICE_RESET* When set to 1, this bit resets all internal registers to their default values. The bit automatically clears to 0 once the reset is done. The default values for each register can be found in Section 3.
- SLEEP* When set to 1, this bit puts the MPU-60X0 into sleep mode.
- CYCLE* When this bit is set to 1 and *SLEEP* is disabled, the MPU-60X0 will cycle between sleep mode and waking up to take a single sample of data from active sensors at a rate determined by *LP_WAKE_CTRL* (register 108).
- TEMP_DIS* When set to 1, this bit disables the temperature sensor.
- CLKSEL* 3-bit unsigned value. Specifies the clock source of the device.

4.18 Registers 65 and 66 – Temperature Measurement

TEMP_OUT_H and TEMP_OUT_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
41	65	TEMP_OUT[15:8]							
42	66	TEMP_OUT[7:0]							

Description:

These registers store the most recent temperature sensor measurement.

The temperature in degrees C for a given register value may be computed as:

$$\text{Temperature in degrees C} = (\text{TEMP_OUT Register Value as a signed quantity})/340 + 36.53$$