

# ECE 105: Introduction to Electrical Engineering

Lecture 10

Sensors 2

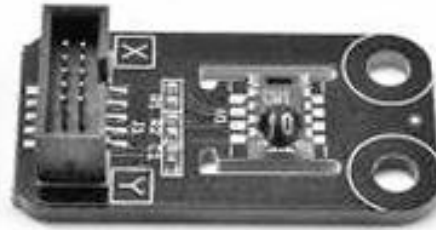
Yasser Khan

Rehan Kapadia

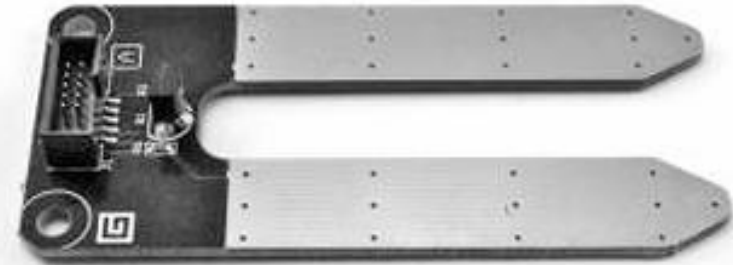
# Other sensors



Gas Sensor

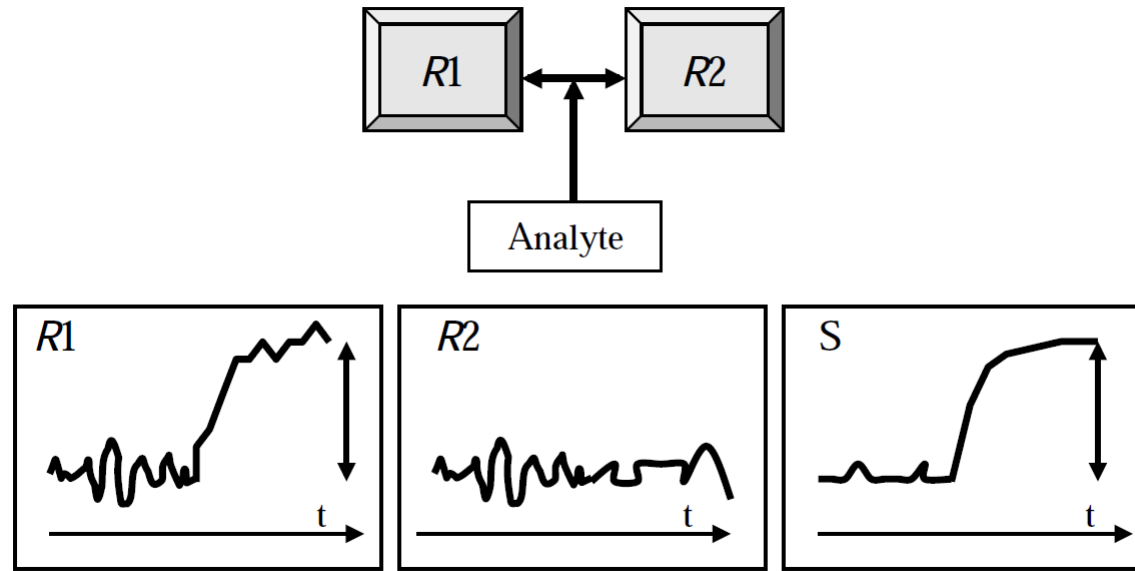


Humidity

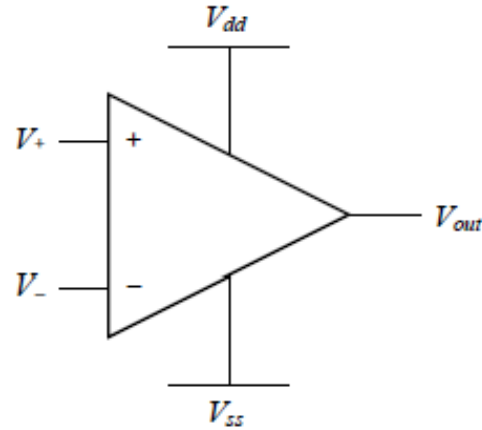


Moisture

# Differential measurement



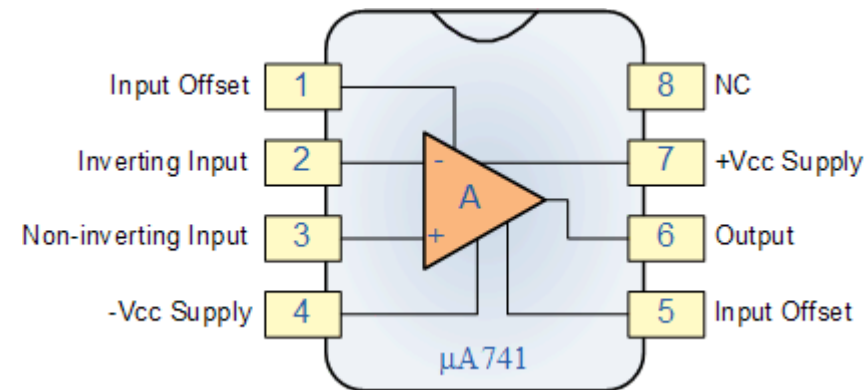
# Op Amp



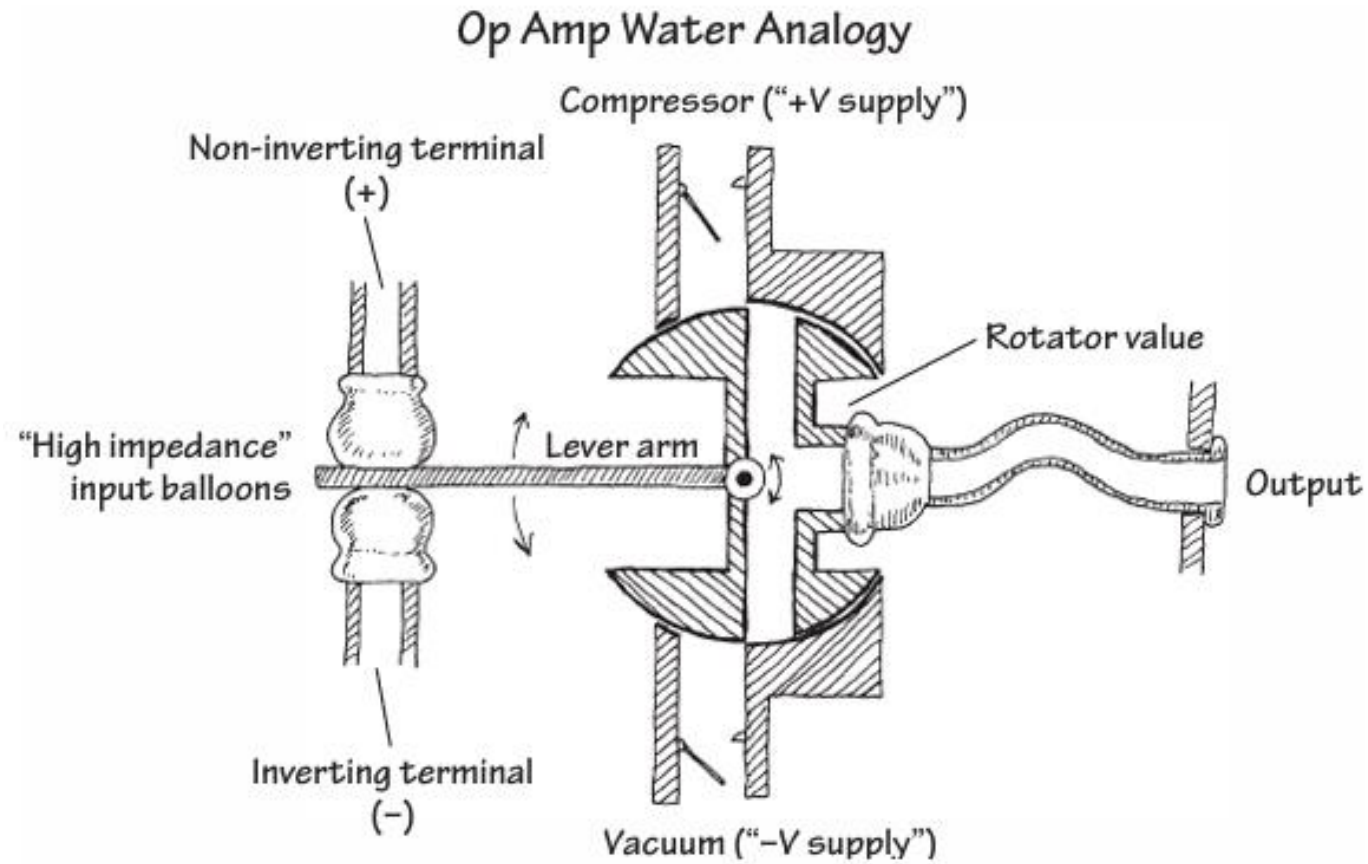
There are three Golden Rules for op-amp circuit design and analysis:

- (1) no current flows into the inputs,  $V_+$  and  $V_-$ ;
- (2) the input voltages are always equal, that is  $V_+ = V_-$ ;
- (3) the op-amp output can drive any current that is required.

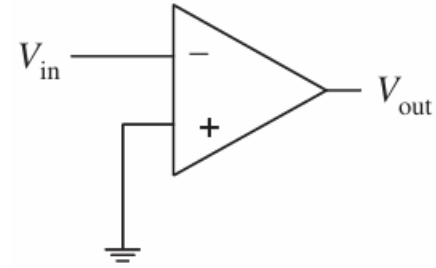
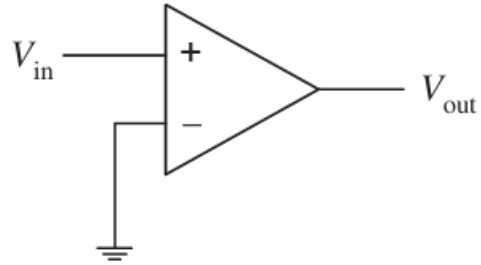
Circuit schematic symbol for an operational amplifier.



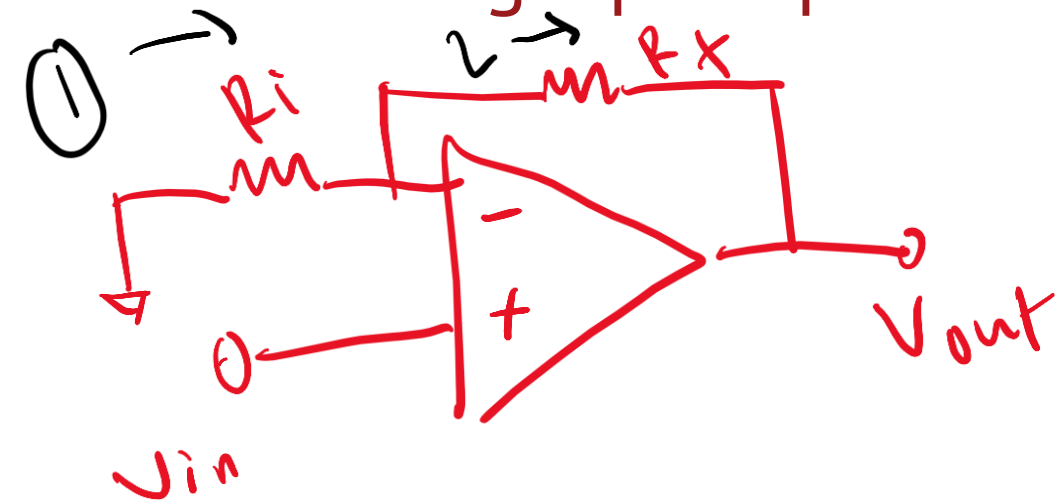
# Op Amp Water Analogy



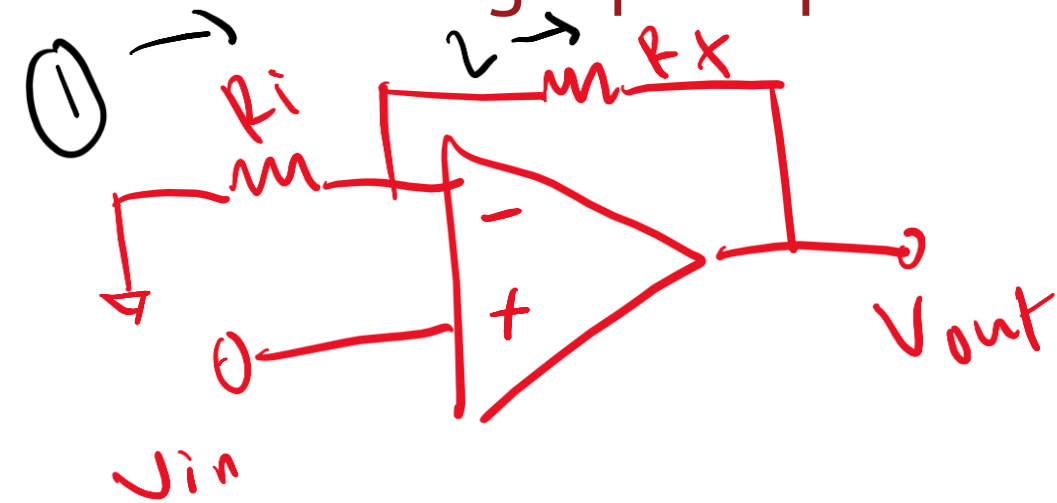
# Non-inverting and inverting



# Non-inverting Op Amp



# Non-inverting Op Amp



②

$$\frac{0 - V_{in}}{R_i} = \frac{V_{in} - V_{out}}{R_f}$$

$$\frac{-V_{in}}{R_i} = \frac{V_{in}}{R_f} - \frac{V_{out}}{R_f}$$

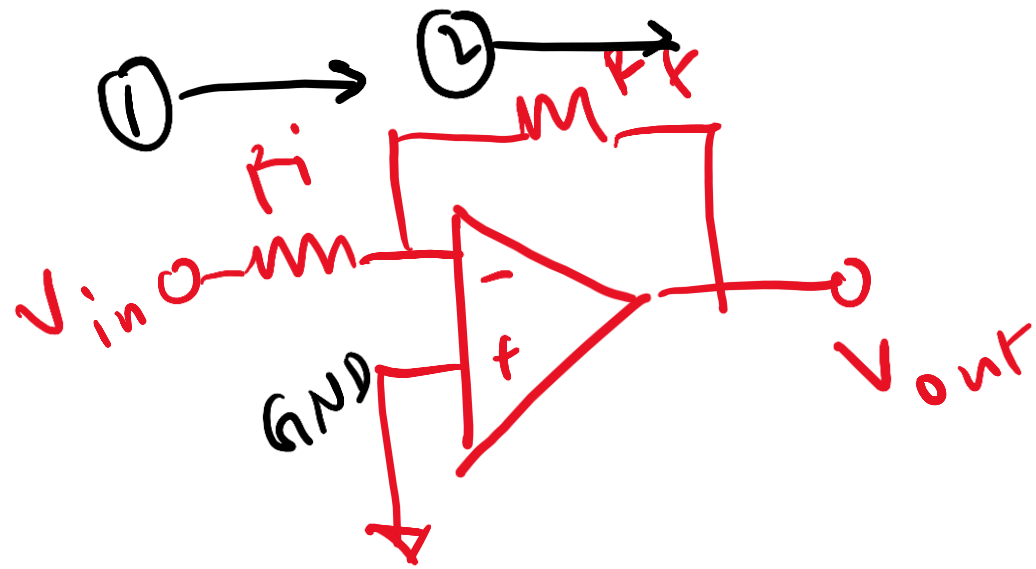
$$\frac{-V_{out}}{R_f} = \frac{-V_{in}}{R_i} - \frac{V_{in}}{R_f}$$

$$V_{out} = V_{in} \cdot R_f \cdot \left( \frac{R_f + R_i}{R_i \cdot R_f} \right)$$

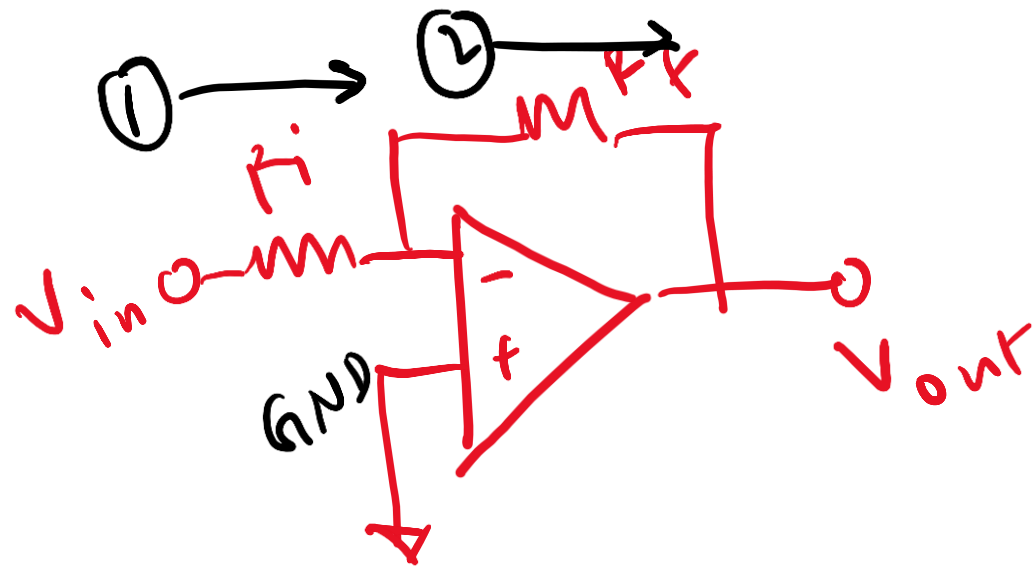
$$V_{out} = V_{in} \frac{R_i + R_f}{R_i}$$



# Inverting Op Amp



# Inverting Op Amp

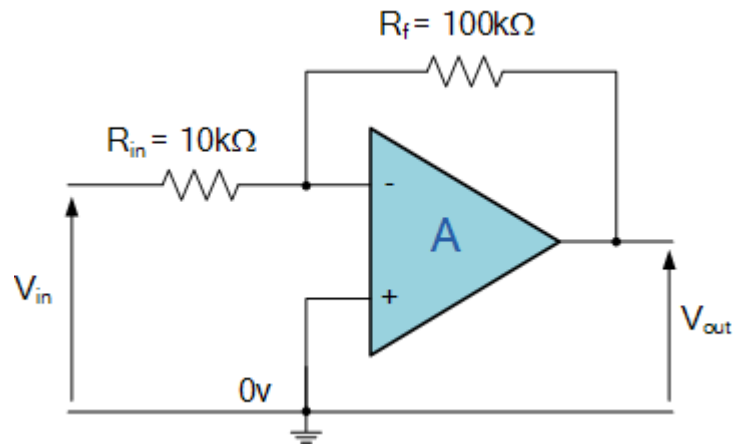


$$\textcircled{1} \frac{V_{in} - 0}{R_i} = \textcircled{2} \frac{0 - V_{out}}{R_f}$$

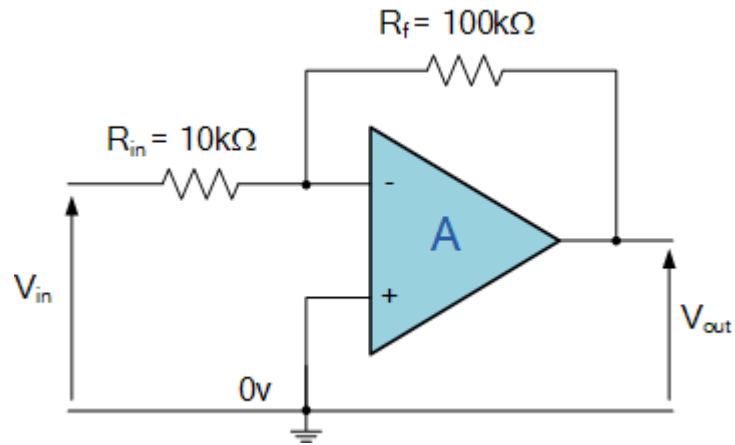
$$\frac{V_{in}}{R_i} = \frac{-V_{out}}{R_f}$$

$$V_{out} = \left( -\frac{R_f}{R_i} \right) V_{in}$$

# Example



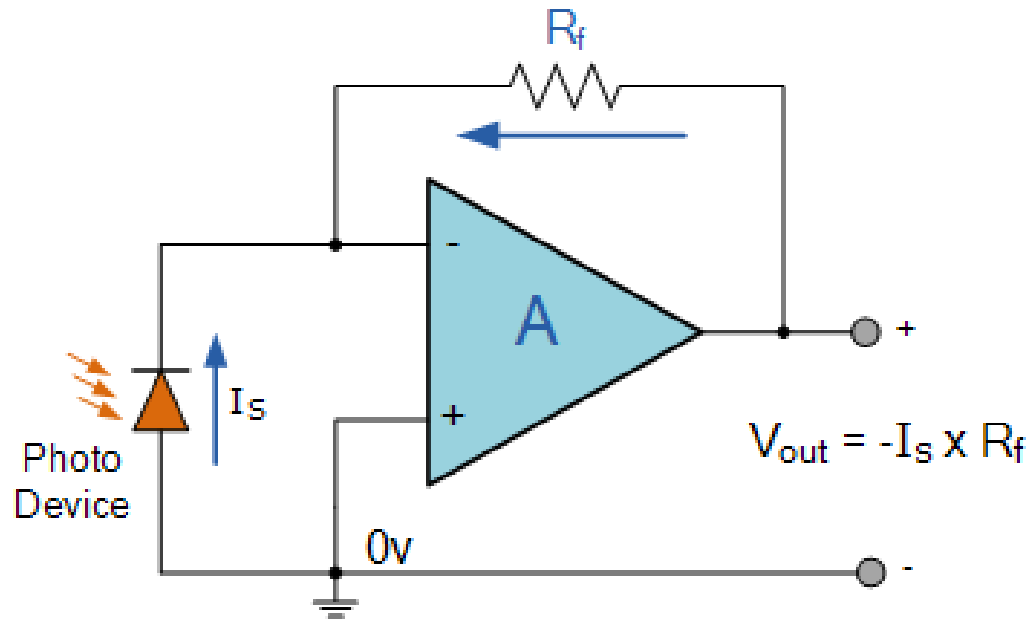
# Example



$$V_{out} = -\frac{R_f}{R_i} \cdot V_{in}$$
$$= -\frac{100k}{10k} V_{in}$$

$$V_{out} = (-10) V_{in}$$

# PD readout circuit



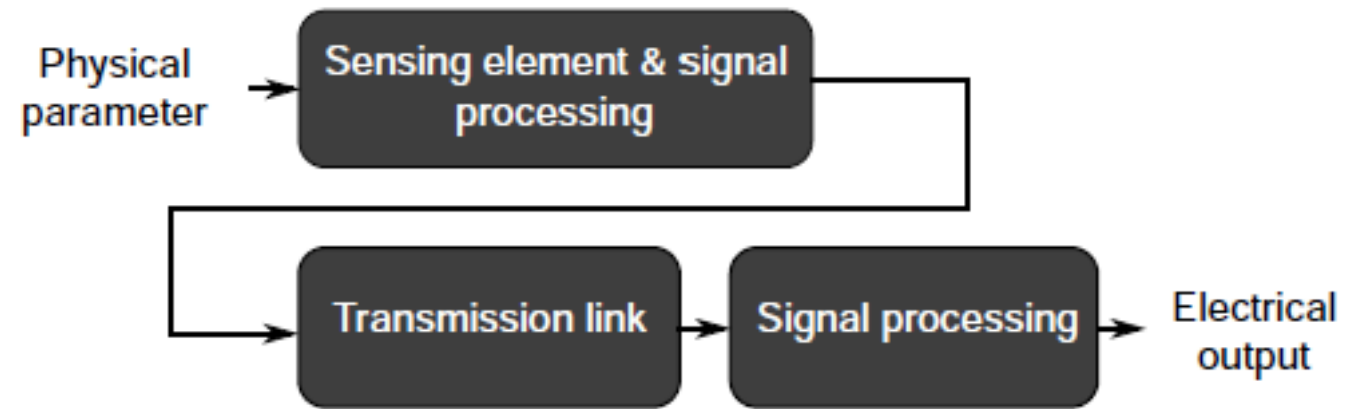
$$\frac{0 - V_{out}}{R_f} = I_s$$

$$V_{out} = -I_s \cdot R_f$$

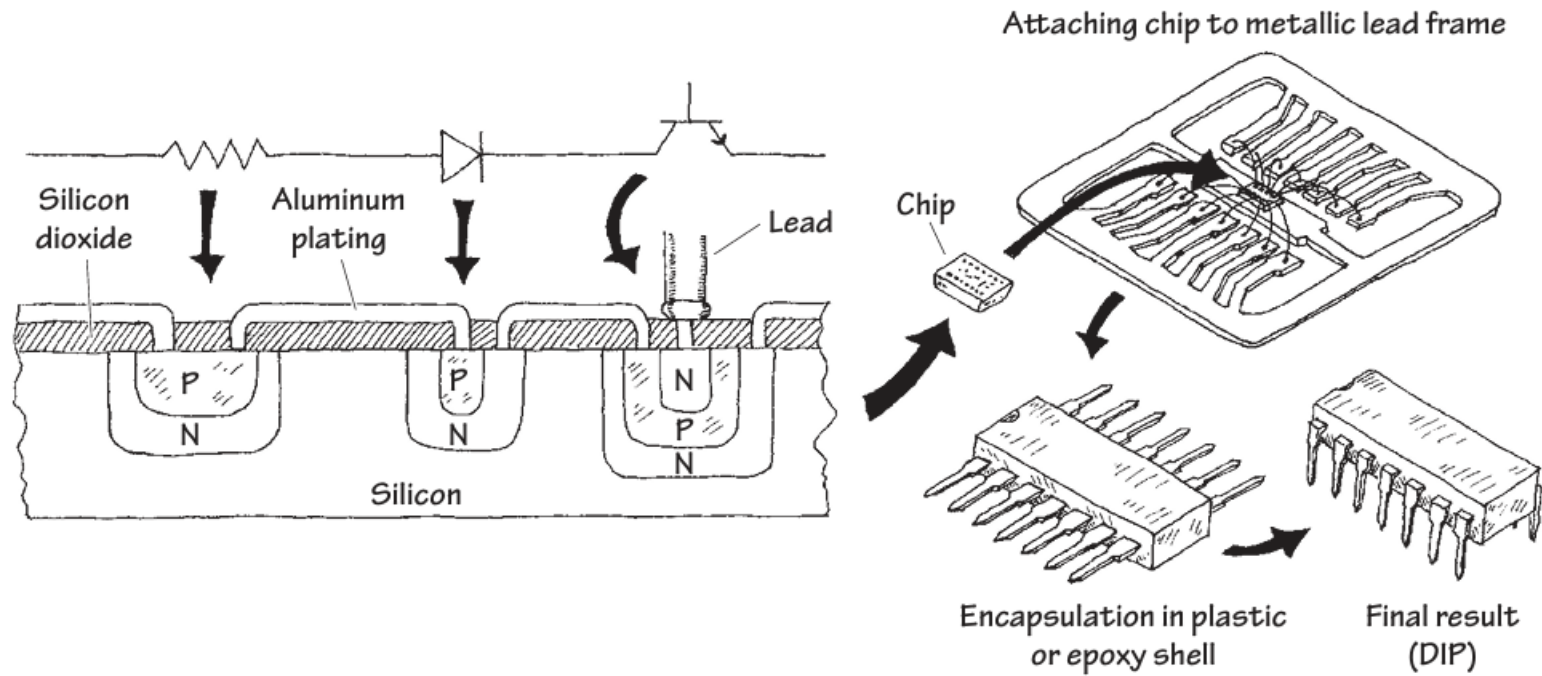
We need  $V_{out}$  (  $-3V$  ).  $I_s = 10 \mu A$   $\left\{ \begin{array}{l} -3V = 10 \mu A \cdot R_f \\ R_f = \frac{3}{10 \mu} \\ = \frac{3 \times 10^6}{10} \Omega \end{array} \right.$

What  $R_f$  we should use?

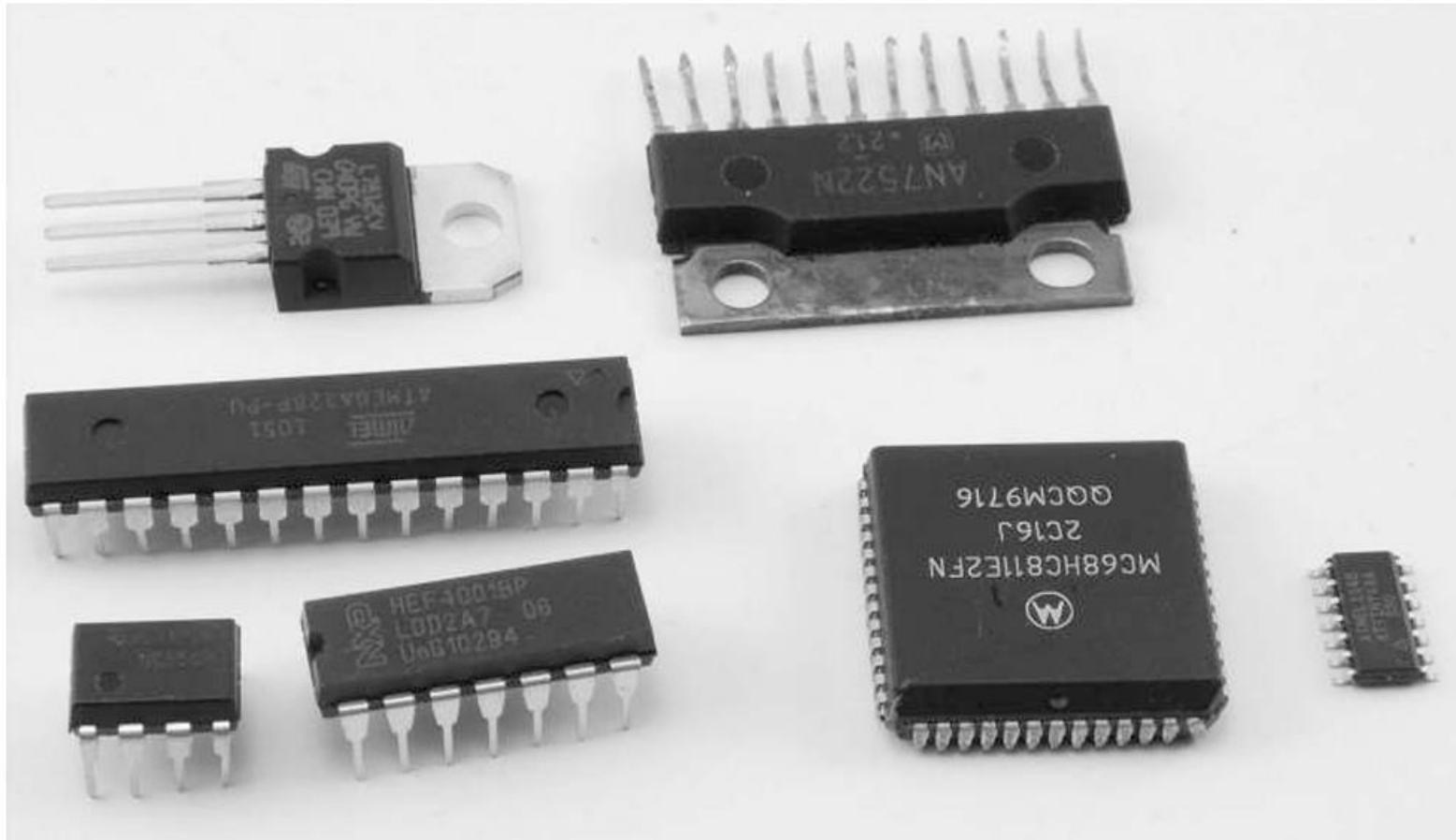
# From physical world to getting data on a computer



# Packaged chips



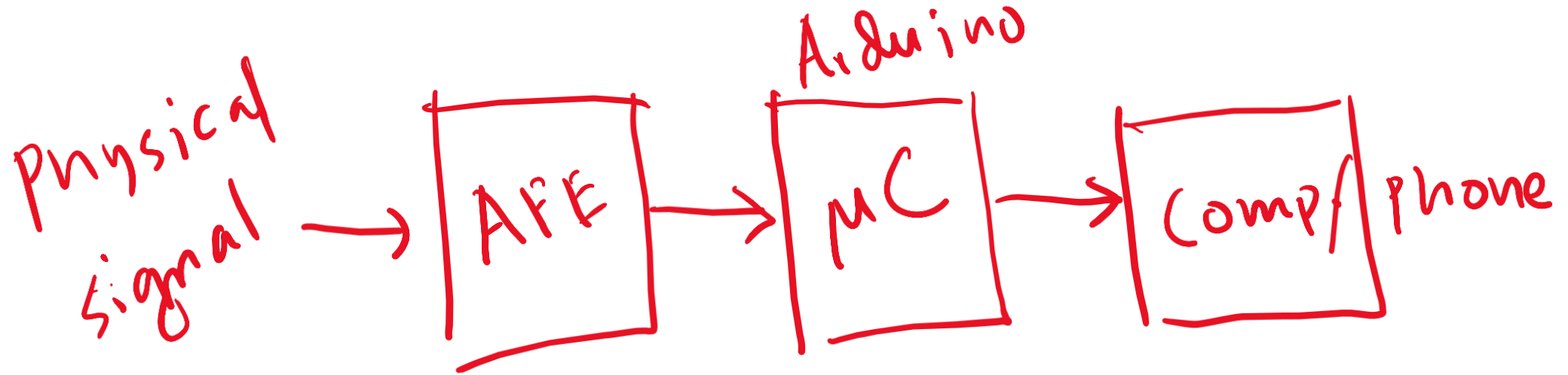
# IC packages



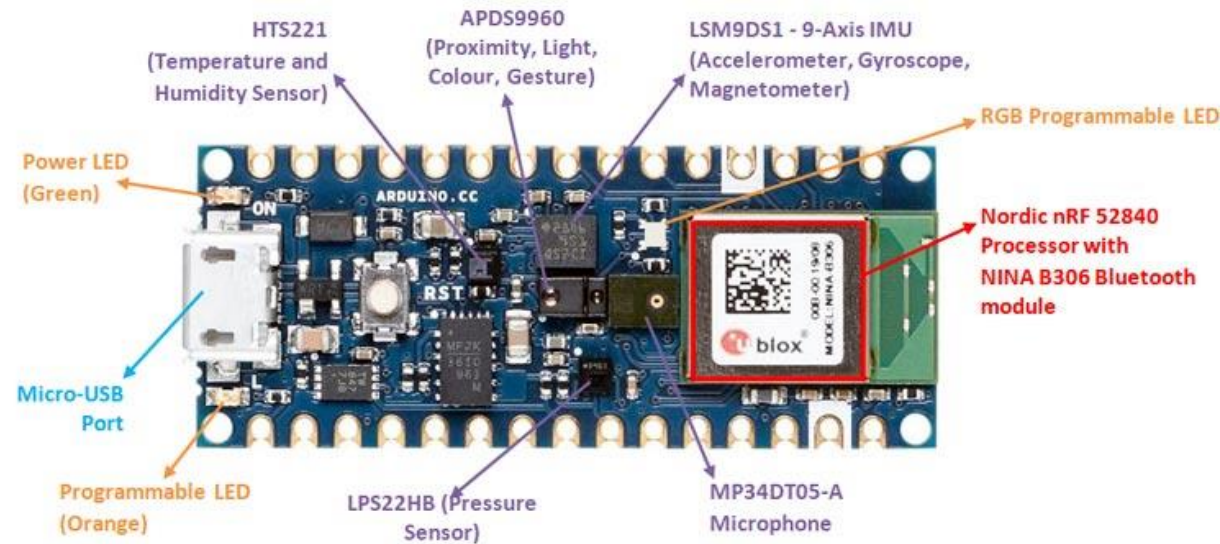


# Analog front end

An analog front-end (AFE) is a set of analog signal conditioning circuitry that uses sensitive analog amplifiers, often operational amplifiers, filters, and sometimes application-specific integrated circuits for sensors block needed to interface a variety of sensors to analog-to-digital converter or, in some cases, to a microcontroller.



# Let's use an analog front end



## APDS-9960

Digital Proximity, Ambient Light, RGB and Gesture Sensor

## Data Sheet

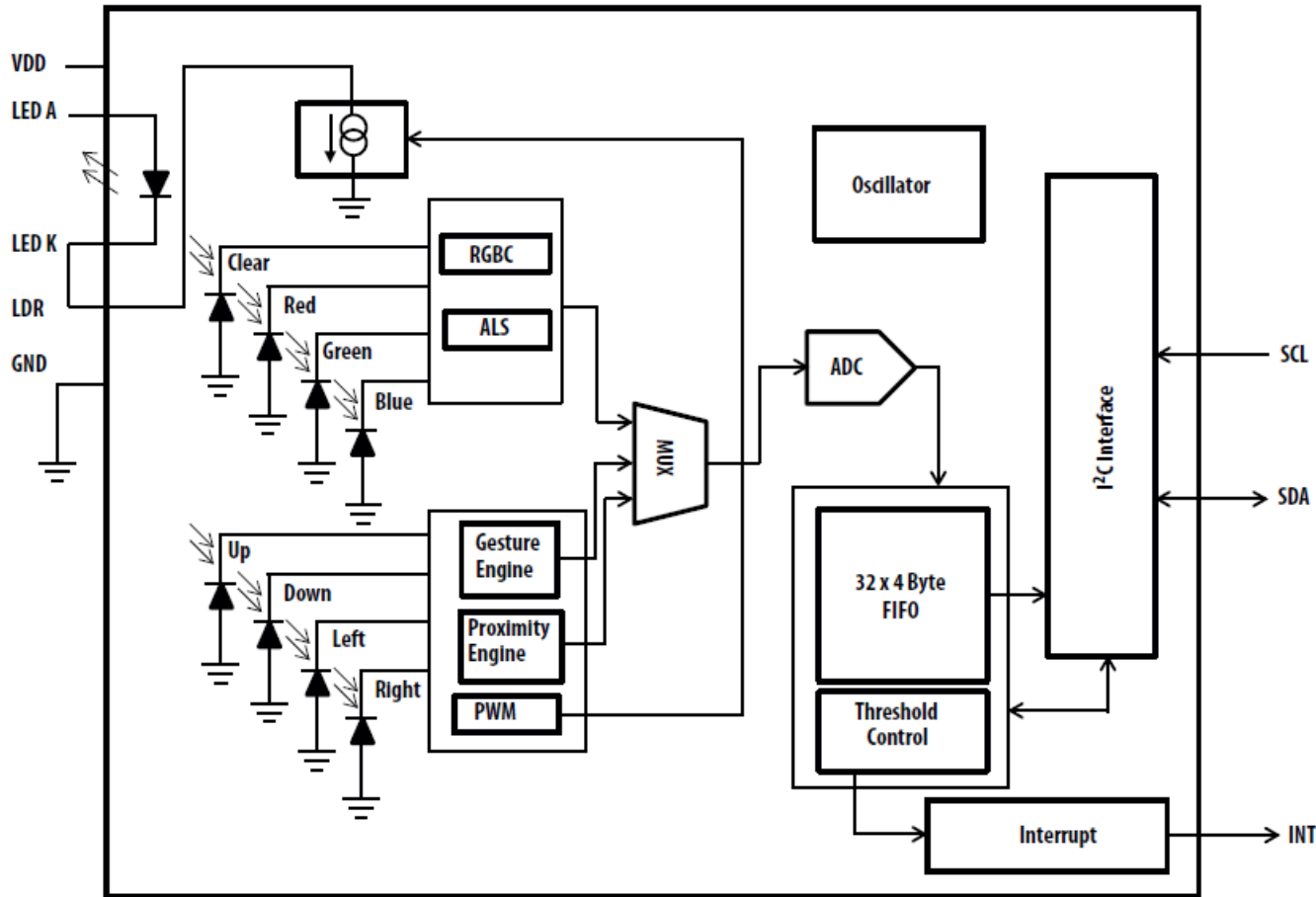


<https://docs.broadcom.com/doc/AV02-4191EN>



# Block diagram of the AFE

Functional Block Diagram



# Pin Outs of the AFE

I/O Pins Configuration

Pin	Name	Type	Description
1	SDA	I/O	I <sup>2</sup> C serial data I/O terminal - serial data I/O for I <sup>2</sup> C-bus
2	INT	O	Interrupt - open drain (active low)
3	LDR		LED driver input for proximity IR LED, constant current source LED driver
4	LEDK		LED Cathode, connect to LDR pin when using internal LED driver circuit
5	LEDA		LED Anode, connect to V <sub>LEDA</sub> on PCB
6	GND		Power supply ground. All voltages are referenced to GND
7	SCL	I	I <sup>2</sup> C serial clock input terminal - clock signal for I <sup>2</sup> C serial data
8	V <sub>DD</sub>		Power supply voltage



Absolute Maximum Ratings over operating free-air temperature range (unless otherwise noted)\*

Parameter	Symbol	Min	Max	Units	Conditions
Power supply voltage <sup>[1]</sup>	V <sub>DD</sub>		3.8	V	
Input voltage range	V <sub>IN</sub>	-0.5	3.8	V	
Output voltage range	V <sub>OUT</sub>	-0.3	3.8	V	
Storage temperature range	T <sub>stg</sub>	-40	85	°C	

\* Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

Note 1. All voltages are with respect to GND.

# AFE to Arduino communication protocol

## I<sup>2</sup>C-bus Protocol

Interface and control are accomplished through an I<sup>2</sup>C-bus serial compatible interface (standard or fast mode) to a set of registers that provide access to device control functions and output data. The devices support the 7-bit I<sup>2</sup>C-bus addressing protocol.

The device supports a single slave address of 0x39 Hex using 7-bit addressing protocol. (Contact factory for other addressing options.)

- A Acknowledge (0)
- N Not Acknowledged (1)
- P Stop Condition
- R Read (1)
- S Start Condition
- Sr Repeated Start Condition
- W Write (0)
- ... Continuation of protocol
- Master-to-Slave
- Slave-to-Master

1	7	1	1	8	1	8	1	1	
S	Slave Address	W	A	Register Address	A	Data	A	...	P

I<sup>2</sup>C-bus Write Protocol

1	7	1	1	8	1	8	1	1	
S	Slave Address	R	A	Data	A	Data	A	...	P

I<sup>2</sup>C-bus Read Protocol

1	7	1	1	8	1	1	7	1	1	8	1
S	Slave Address	W	A	Register Address	A	Sr	Slave Address	R	A	Data	A

I<sup>2</sup>C-bus Read Protocol - Combined Format

8	1	1	
Data	A	...	P

The I<sup>2</sup>C-bus standard provides for three types of bus transaction: read, write, and a combined protocol. During a write operation, the first byte written is a command byte followed by data. In a combined protocol, the first byte written is the command byte followed by reading a series of bytes. If a read command is issued, the register address from the previous command will be used for data access. Likewise, if the MSB of the command is not set, the device will write a series of bytes at the address stored in the last valid command with a register address. The command byte contains either control information or a 5-bit register address. The control commands can also be used to clear interrupts.

The I<sup>2</sup>C-bus protocol was developed by Philips (now NXP). For a complete description of the I<sup>2</sup>C-bus protocol, please review the NXP I<sup>2</sup>C-bus design specification at <http://www.i2c-bus.org/references/>.

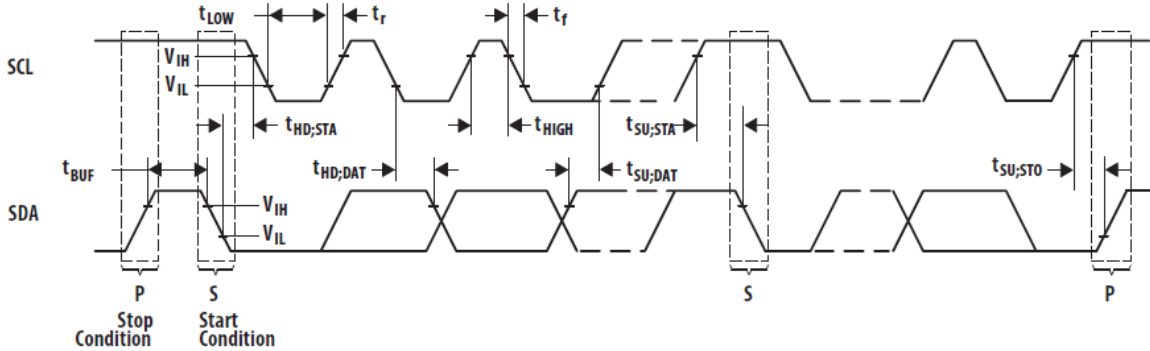


Figure 1. Timing Diagrams

# Spectral response of the optical sensor

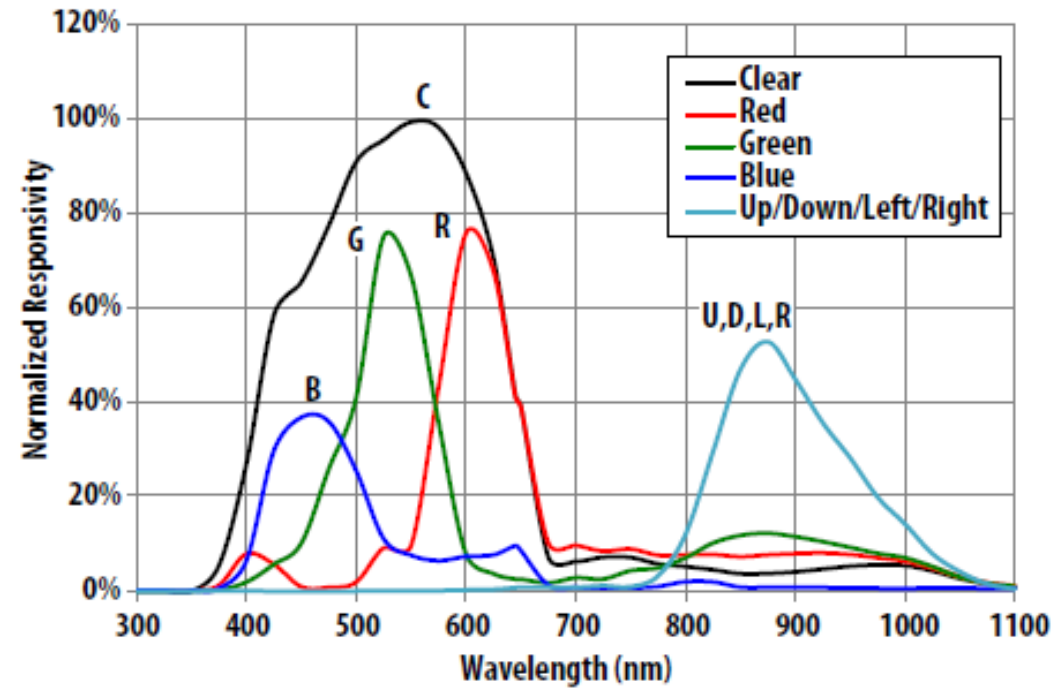


Figure 2. Spectral Response

# How to connect the AFE to the Arduino

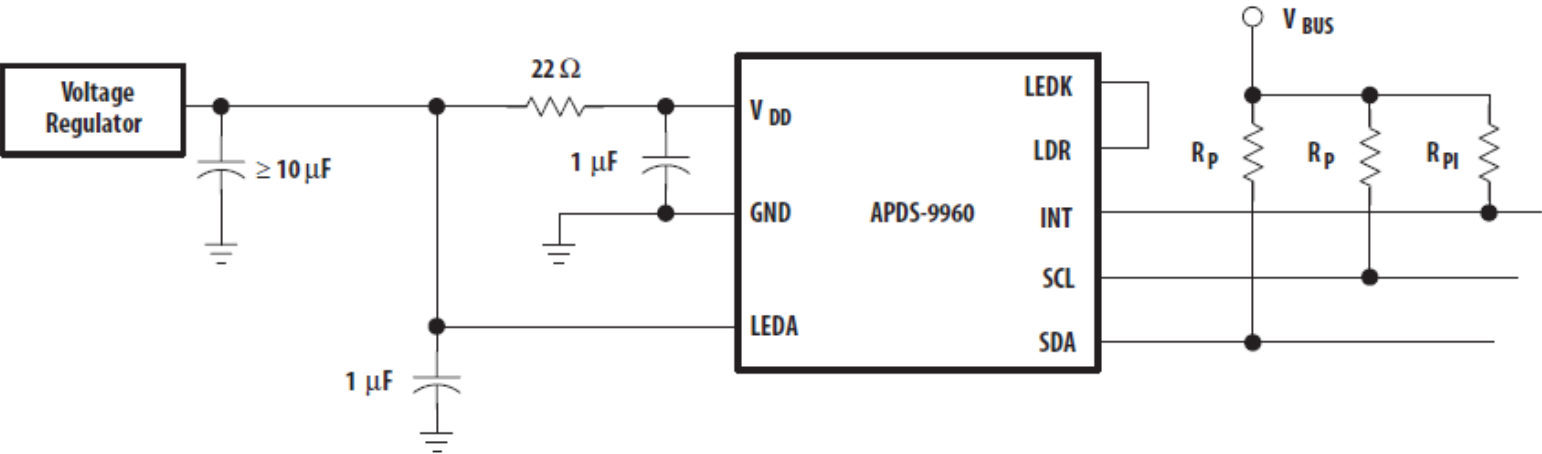


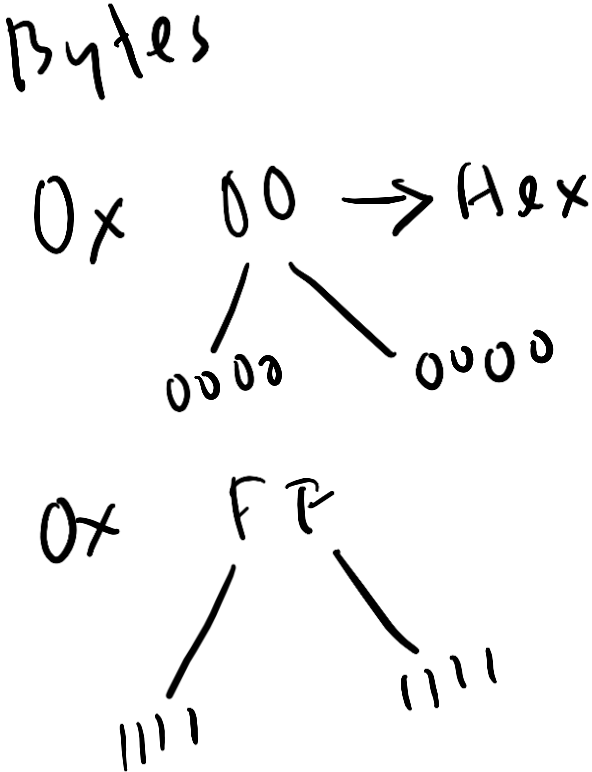
Figure 14b. Circuit Implementation using Single Power Supply

# How data is stored in the AFE or the Arduino

## Register Set

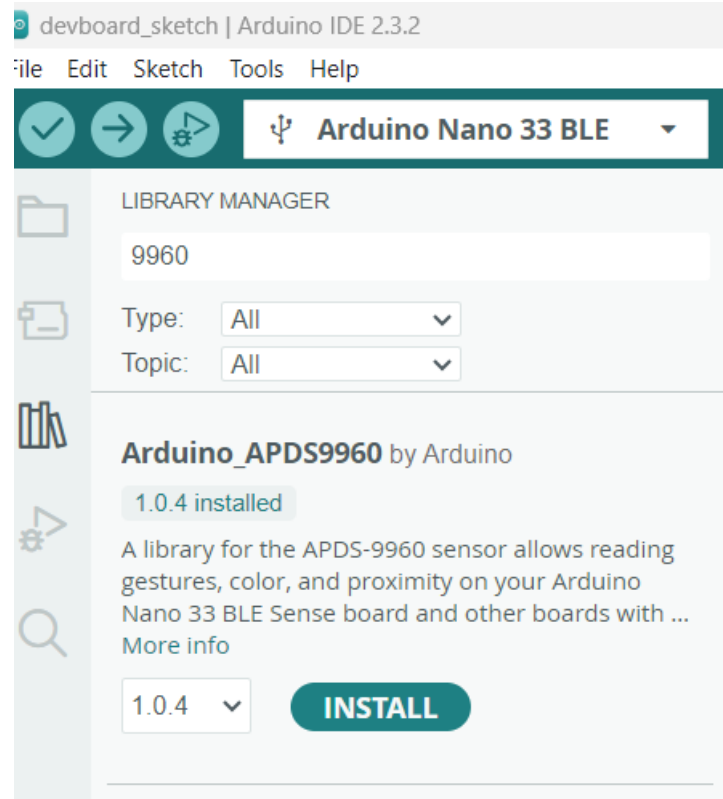
The APDS-9960 is controlled and monitored by data registers and a command register accessed through the serial interface. These registers provide for a variety of control functions and can be read to determine results of the ADC conversions.

Address	Register Name	Type	Register Function	Reset Value
0x00 – 0x7F	RAM	R/W	RAM	0x00
0x80	ENABLE	R/W	Enable states and interrupts	0x00
0x81	ATIME	R/W	ADC integration time	0xFF
0x83	WTIME	R/W	Wait time (non-gesture)	0xFF
0x84	AILTL	R/W	ALS interrupt low threshold low byte	--
0x85	AILTH	R/W	ALS interrupt low threshold high byte	--
0x86	AIHTL	R/W	ALS interrupt high threshold low byte	0x00
0x87	AIHTH	R/W	ALS interrupt high threshold high byte	0x00
0x89	PILT	R/W	Proximity interrupt low threshold	0x00
0x8B	PIHT	R/W	Proximity interrupt high threshold	0x00
0x8C	PERS	R/W	Interrupt persistence filters (non-gesture)	0x00
0x8D	CONFIG1	R/W	Configuration register one	0x60
0x8F	PPULSF	R/W	Proximity pulse count and length	0x40

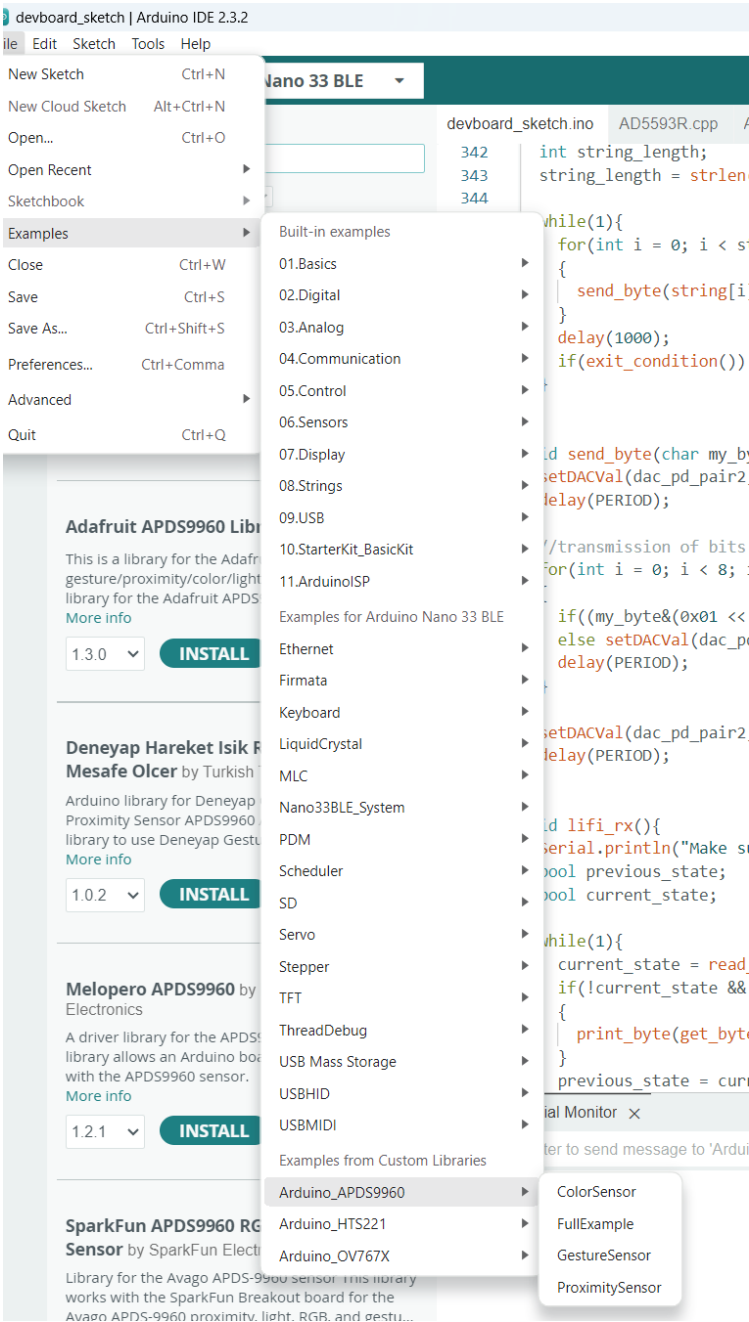




# Search and install library for the AFE



# Run an example



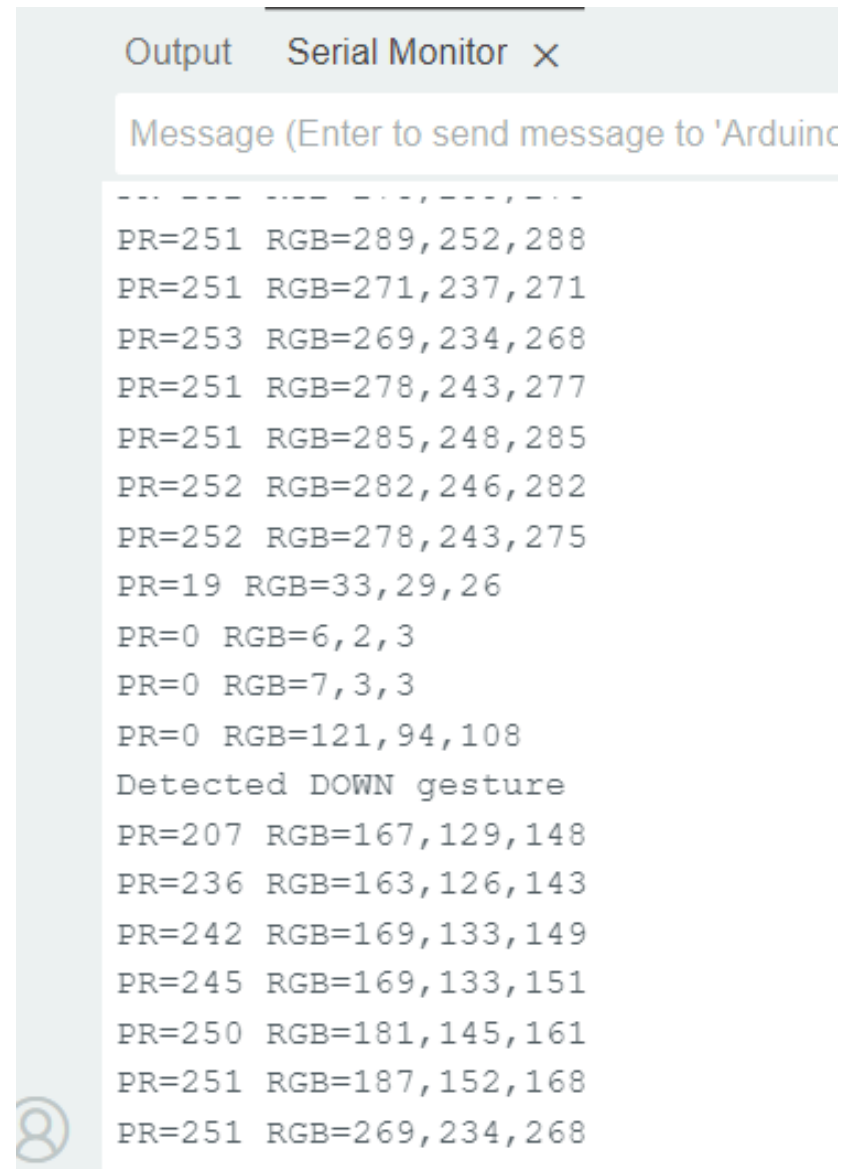
# Upload code to Ard.



```
FullExample | Arduino IDE 2.3.2
File Edit Sketch Tools Help

FullExample.ino
1  /*
2   APDS-9960 - All sensor data from APDS-9960
3
4   This example reads all data from the on-board APDS-9960 sensor of the
5   Nano 33 BLE Sense:
6   - color RGB (red, green, blue)
7   - proximity
8   - gesture
9   and prints updates to the Serial Monitor every 100 ms.
10
11  The circuit:
12  - Arduino Nano 33 BLE Sense
13
14  This example code is in the public domain.
15  */
16
17  #include <Arduino_APDS9960.h>
18
19  void setup() {
20    Serial.begin(9600);
21    while (!Serial); // Wait for Serial Monitor to open
22
23    if (!APDS.begin()) {
24      Serial.println("Error initializing APDS-9960 sensor.");
25      while (true); // Stop forever
26    }
27  }
28
29  int proximity = 0;
30  int r = 0, g = 0, b = 0;
31  unsigned long lastUpdate = 0;
32
33  void loop() {
34
35    // Check if a proximity reading is available.
36    if (APDS.proximityAvailable()) {
37      proximity = APDS.readProximity();
38    }
39
40    // Check if a gesture reading is available
41    if (APDS.gestureAvailable()) {
42      int gesture = APDS.readGesture();
43      switch (gesture) {
44        case GESTURE_UP:
45          Serial.println("Detected UP gesture");
46          break;
47
48        case GESTURE_DOWN:
49          Serial.println("Detected DOWN gesture");
50          break;
51
52        case GESTURE_LEFT:
53          Serial.println("Detected LEFT gesture");
54          break;
55
56        case GESTURE_RIGHT:
57          Serial.println("Detected RIGHT gesture");
58          break;
59      }
60    }
61  }
```

# Serial monitor



```
Output  Serial Monitor  ×
Message (Enter to send message to 'Arduino')
--- -- -- -- --
PR=251 RGB=289,252,288
PR=251 RGB=271,237,271
PR=253 RGB=269,234,268
PR=251 RGB=278,243,277
PR=251 RGB=285,248,285
PR=252 RGB=282,246,282
PR=252 RGB=278,243,275
PR=19 RGB=33,29,26
PR=0 RGB=6,2,3
PR=0 RGB=7,3,3
PR=0 RGB=121,94,108
Detected DOWN gesture
PR=207 RGB=167,129,148
PR=236 RGB=163,126,143
PR=242 RGB=169,133,149
PR=245 RGB=169,133,151
PR=250 RGB=181,145,161
PR=251 RGB=187,152,168
PR=251 RGB=269,234,268
```

# Serial plotter

