NYU | TANDON SCHOOL OF ENGINEERING

NSF

Promoting robotic design and entrepreneurship experiences among students and teachers

Lesson 15: Arduino Advanced Session - Sensors

Innovative Technology Experiences for Students and Teachers (ITEST), Professional Development Program, July 2017-19

Mechatronics, Controls, and Robotics Laboratory, Department of Mechanical and Aerospace Engineering, NYU Tandon School of Engineering

CONTENTS

- Living and robotic systems analogy
- Sensing the environment
- Choosing a sensor
- Ultrasonic sensor
- Analog to digital conversion
- How ADC works?
- Light sensor (Photoresistor)

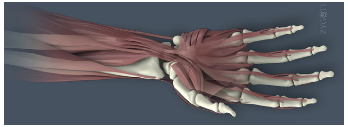
- **TASK/ACTIVITY**: Using Sensors with Arduino

LIVING AND ROBOTIC SYSTEMS ANALOGY

- Living organisms are a major source of inspiration for robotics

Source

Source



Source

Living systems	Analogy	Robotic systems
Sensory organs	⟷	Sensors
↓		↓
Brain	⟷	Microcontroller
↓		↓
Body (musculoskeletal structure)	⟷	Actuators & physical structure (links, joints, etc.)

Note: Many bio-inspired robots have a physical structure designed to mimic living organisms

Humidity sensor (DHT 22)

Air quality sensor (MQ 135)

Temperature sensor (LM 35)

Source

Source

Source

HUMAN- ROBOT ANALOGY

Eye

Source

Camera

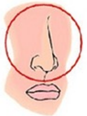
Source

Ear

Microphone

Source

Nose



Gas sensor

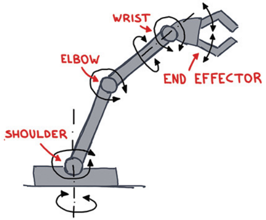
Source

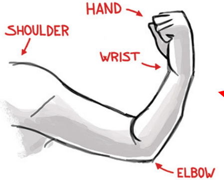
Skin

Touch sensor

Source

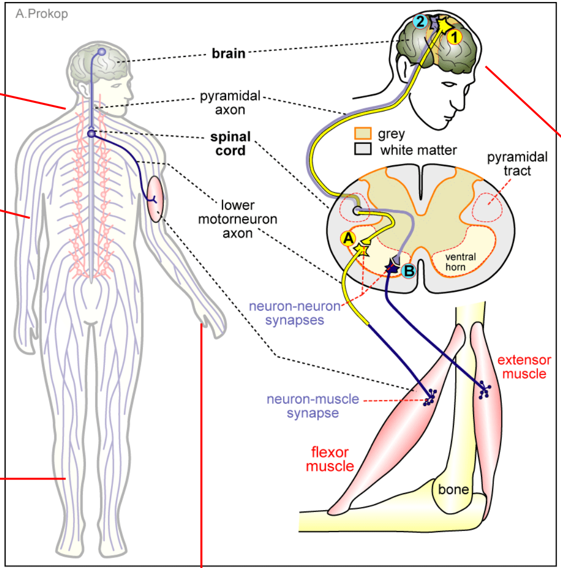
Sensory organs → Sensors



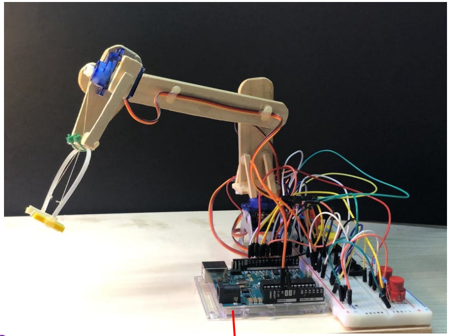


Source

Arm → Actuators

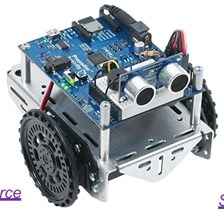


Source

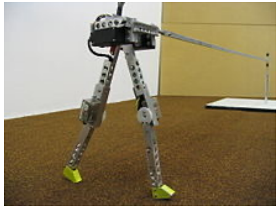


Source

Brain → Microcontroller

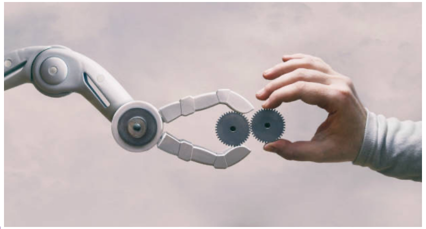


Source



Source

Leg → Mobile mechanism with actuators



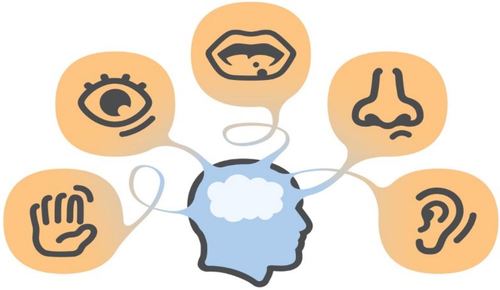
Source

Hand → End-effector

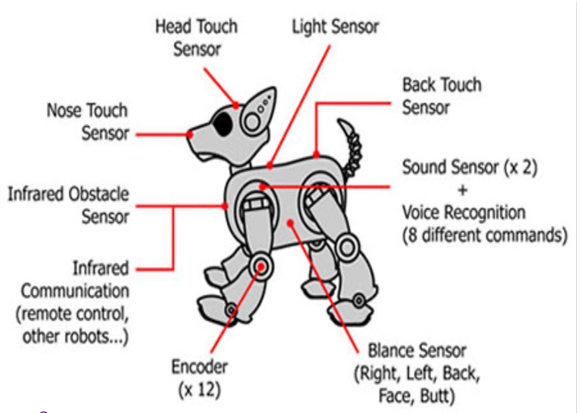
SENSING THE ENVIRONMENT

Analogy: *Human sensory organs → Sensors*

- The human body has sensory organs that help us perceive and respond to external environment
- Robots have **sensors** that capture the changes in the environment (sound, force, light, etc.) and itself (velocity, acceleration, etc.) by providing **variations in electrical signals** (may be voltage or current changes)
- These variations can be processed by a microcontroller to control the actuators of a robot as a response to the changes in the environment



Source



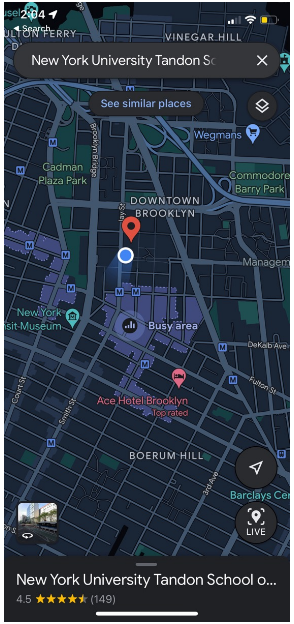
Source

SENSORS USED IN EVERYDAY LIFE

Accelerometer

Source

GPS



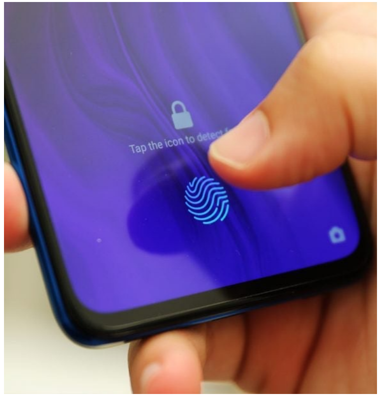
Temperature

Source

Heart rate

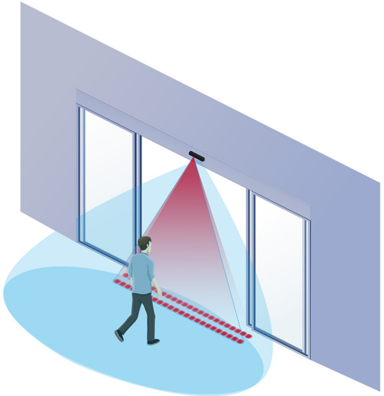
Source

Fingerprint



Source

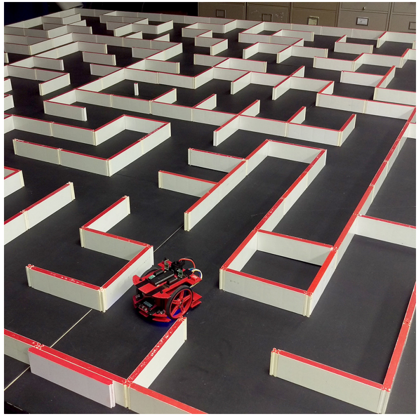
Proximity (Infrared)



Source

WHY DO ROBOTS NEED SENSORS?

Robots need sensors for perception of the surroundings to accomplish tasks involving interaction with the surroundings such as detecting obstacles, following a line, solving a maze, etc.



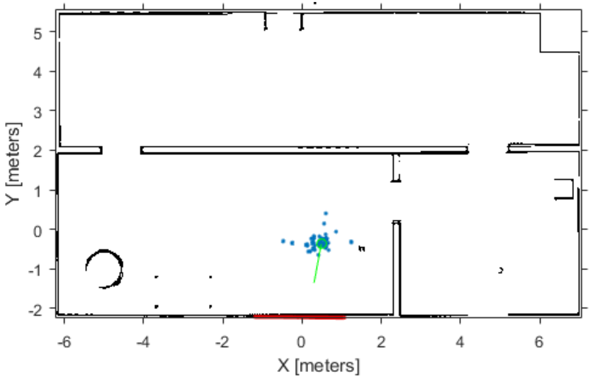
Source

A robot requires sensors to locate itself in a maze

Example 1:

Localization

A mobile robot uses sensors data to estimate its location with respect to the surroundings



Source

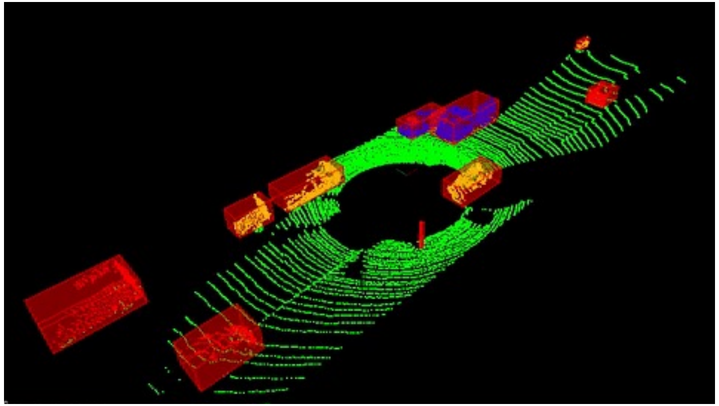
Estimation of robot location in a simulation using sensor data and a localization algorithm

WHY DO ROBOTS NEED SENSORS?

Example 2:

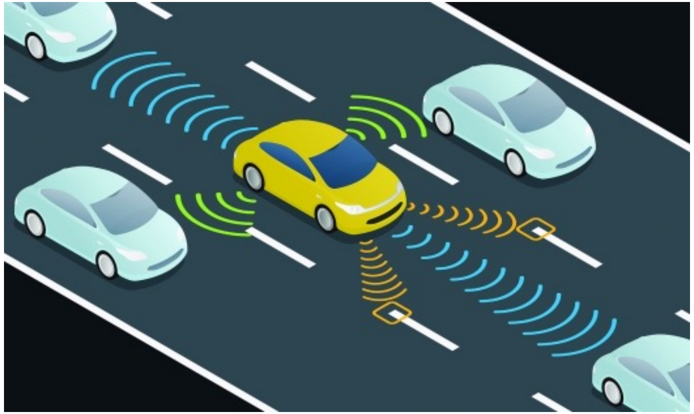
Obstacle detection

Self-driving cars use a combination of LiDAR, radar and ultrasonic sensors for mapping the environment in real time to ensure robust functionality and safe operation



Source

Mapping the surroundings (using LiDAR)



Source

Obstacle avoidance

WHAT ARE SENSORS?

- **American National Standards Institute (ANSI) definition:** A sensor (or “transducer”) is a device which provides a usable output in response to a specific measurand

Source

Input Signal → Sensor → Output Signal

- **Sensors** collect information about the world, they are electrical/mechanical/chemical devices that map an environmental attribute to a quantitative measurement

Source

Automated Weather Station

Source

DETECTABLE PHENOMENON

Stimulus	Quantity
Biological and chemical	fluid concentrations (gas or liquid), flow rate,…
Electric	charge, voltage, current, resistance, capacitance, electric field (amplitude, phase, polarization),…
Magnetic	magnetic field (amplitude, phase, polarization), flux, permeability,…
Optical	refractive index, reflectivity, absorption,…
Thermal	temperature, flux, specific heat, thermal conductivity,…
Mechanical	position, velocity, acceleration, force, strain, stress, pressure, torque,…

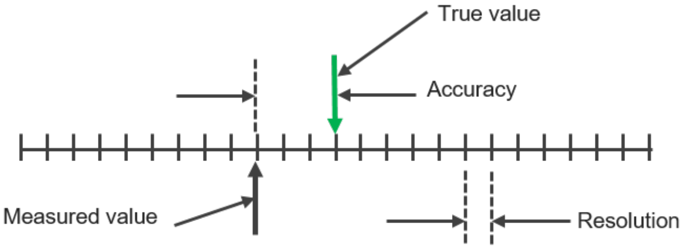
CHOOSING A SENSOR: TERMINOLOGIES

- **Accuracy:** Ability of a sensor to yield a measurement that matches the true value being measured

- **Sensitivity:** Ratio between the change in the output signal to a small change in the input physical signal

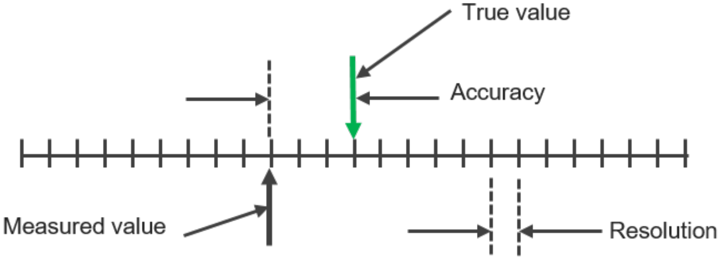
- **Repeatability/Precision:** Ability of the sensor to output the same value for the same input over several trials

- **Resolution:** Smallest increment of a measurement that a device can make



Source

CHOOSING A SENSOR: ACCURACY VS RESOLUTION



Source

Example 1

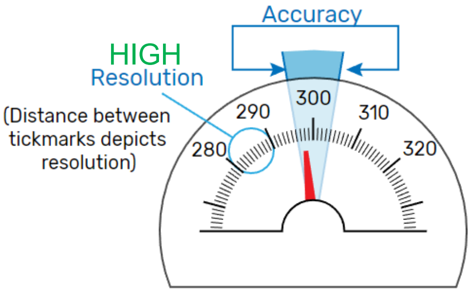
TRUE VALUE = 17.5 °C

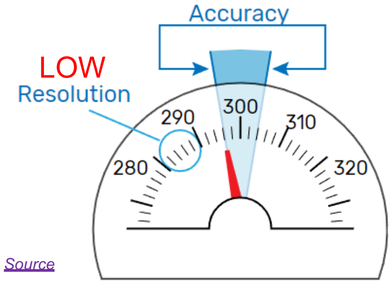
LOW RESOLUTION (no decimals)
MEASUREMENT = 17 °C
(error = 0.5 °C, Low Accuracy)

HIGH RESOLUTION (up to 2 decimal places)
MEASUREMENT = 17.48 °C
(error = 0.02 °C, High Accuracy)

Resolution is commonly expressed as either **decimal places** or **divisions/parts/counts**

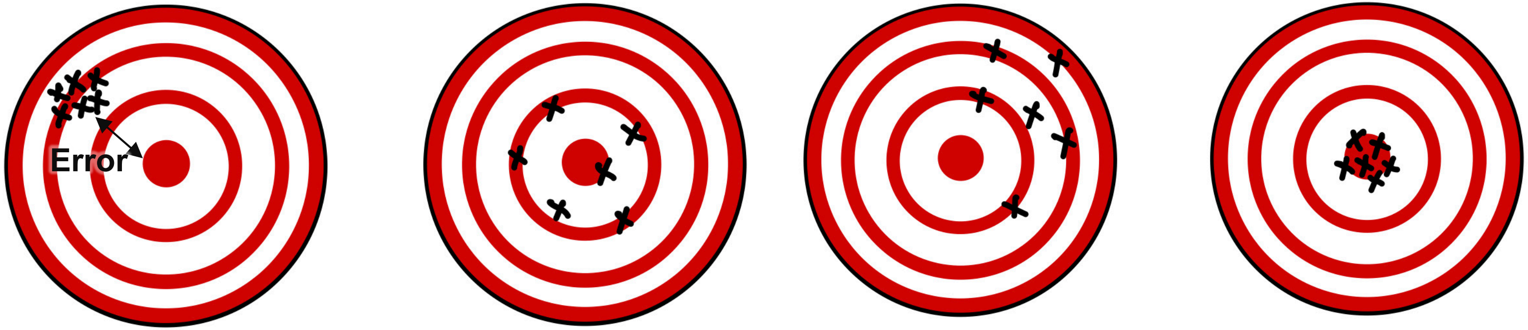
Example 2





Source

CHOOSING A SENSOR: ACCURACY VS PRECISION



High Precision
Low Accuracy

Low Precision
High Accuracy

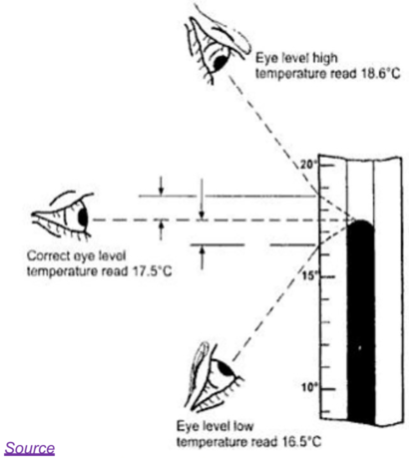
Low Precision
Low Accuracy

High Precision
High Accuracy

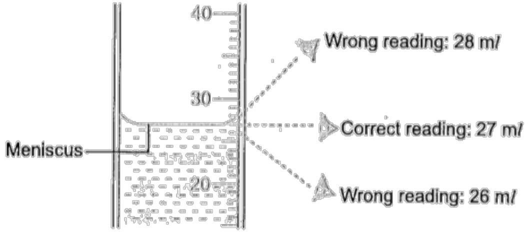
PARALLAX ERROR

Parallax error **occurs when the measurement is more or less than the true value because of your eye position being at an angle relative to the measurement markings**

For example, Parallax error in reading thermometer’s output or when looking at the meniscus of liquid in a measuring container



Source

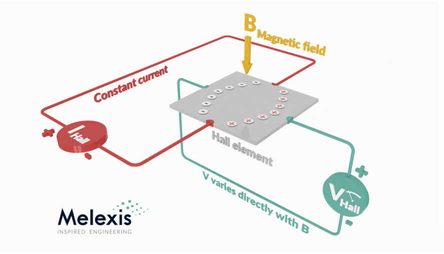


Source

CHOOSING A SENSOR: IMPORTANT FACTORS

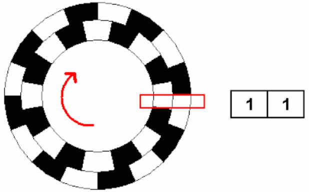
Environmental factors	Economic factors	Sensor characteristics
Temperature range	Cost	Sensitivity
Humidity effects	Availability	Range
Corrosion	Lifetime	Stability
Size		Repeatability
Overrange protection		Linearity
Power consumption		Error
Susceptibility to electro-magnetic (EM) interference		Response time

CHOOSING A SENSOR: DIFFERENT TYPES



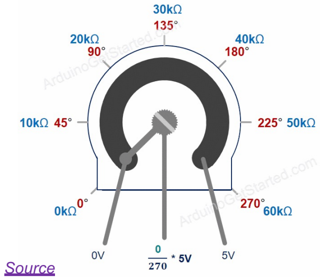
Source

Hall effect sensor



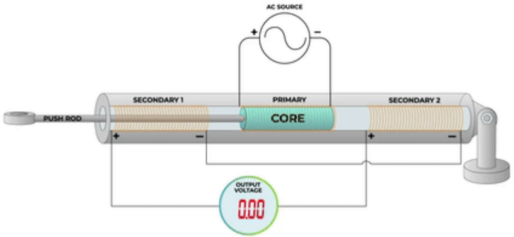
Source

Encoder



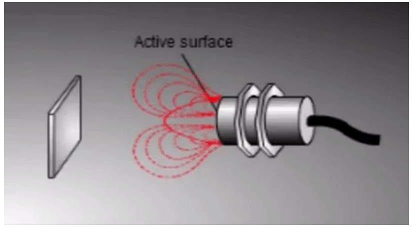
Source

Potentiometer



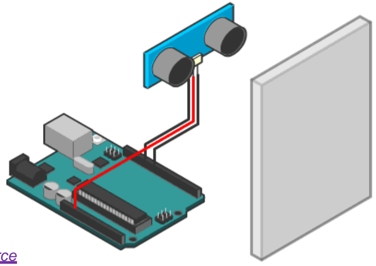
Source

Linear Variable Differential Transducer



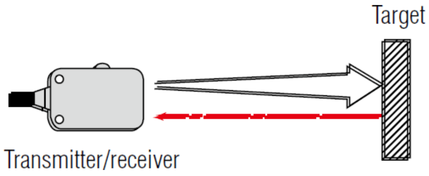
Source

Eddy current sensor



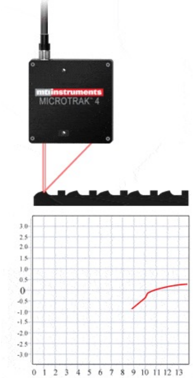
Source

Ultrasonic sensor



Source

Reflective sensor



Source

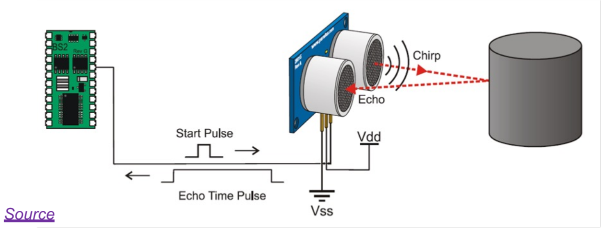
Laser sensor

CHOOSING A SENSOR: EXAMPLE

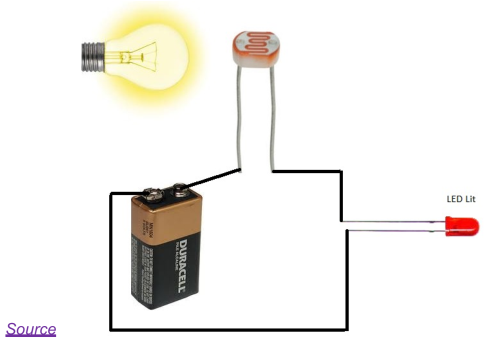
Sensor	Range	Environment	Accuracy	Sensitivity	Cost
Hall Effect Sensor	Short	Standard	On or off	On or off	Low
Optical Encoder – Linear / Rotary	Long	Standard	Variable	High	Variable
Potentiometer	Long	Standard	Medium	High	Low
Linear/Rotary Variable Differential Transformer (LVDT/RVDT)	Limited	Dirty environments	High	High	High
Eddy-Current Proximity Probe	Short	Dirty environments	Medium	Variable	Medium
Ultrasound	Long	Standard	Variable	Medium	Variable
Reflective Light Proximity Sensor	Long	Standard	Variable	High	Variable
Laser Sensor	Long	Dirty environments and high temp target	High	High	Variable

TYPE OF SENSORS

- **Active sensors:** Send signal into environment and measure interaction of signal with the environment (e.g., radar, ultrasonic sensor)
- **Passive sensors:** Record signals already present in environment (e.g., temperature sensor (LM35), photoresistor)
- **Analog sensors:** Sensors that produce continuous analog output signal, for digital processing, we usually need to convert such sensor output into digital domain (e.g., potentiometers, temperature sensor, accelerometer)
- **Digital sensors:** Sensors that produce discrete valued output signal (e.g., digital cameras, digital temperature sensor, push button switch)



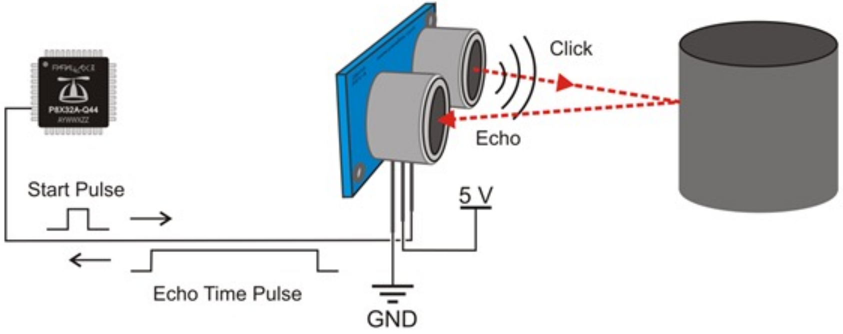
Active sensor: Ultrasonic sensor



Passive sensor: Photoresistor

ULTRASONIC DISTANCE SENSOR

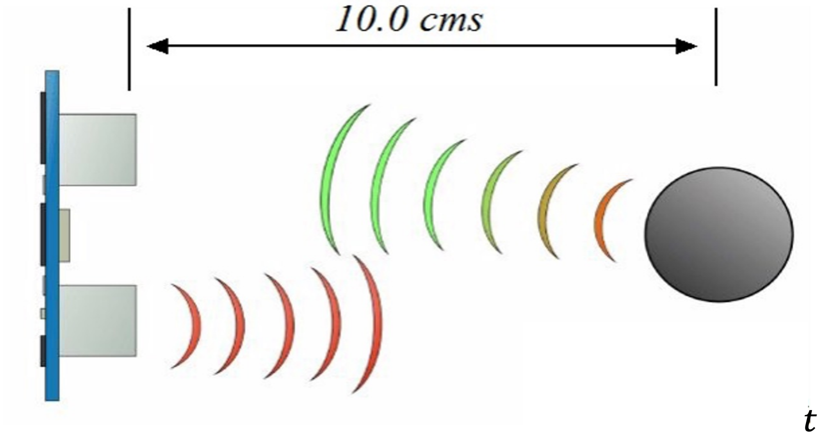
- The ultrasonic sensor emits short bursts of sound and waits for the waves to be reflected by (echoed) nearby objects
- The frequency of the sound is too high for humans to hear (40KHz–ultrasonic frequency)
- The ultrasonic sensor **measures the time of flight of the sound burst**
- The user then computes the distance to an object using this time of flight and the speed of sound in air (1,126 ft/s or **332 m/s**)



Source

ULTRASONIC DISTANCE SENSOR

- The ultrasonic sensor sends out a high-frequency sound pulse and then times how long it takes for the echo of sound to reflect back
- The sensor has 2 openings on its front, one opening transmits ultrasonic waves, (like a tiny speaker), the other receives them (like a tiny microphone)



Source

Note: 1 μs = 10^{-6} s

Speed of sound in air

$$v = 340\ m/s$$
$$v = 0.034\ cm/\mu s$$

Total measured time:

$$t = 588\ \mu s$$

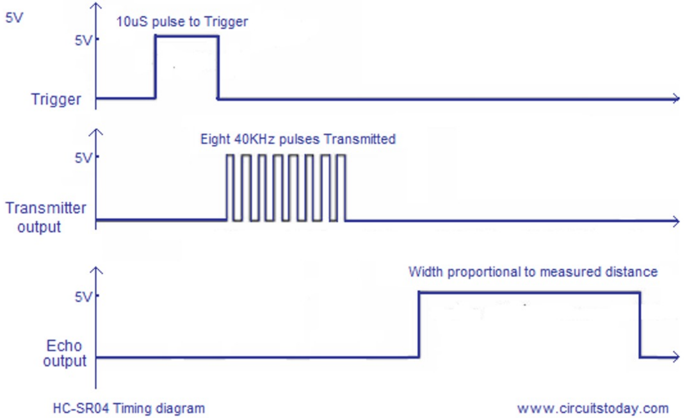
Distance

$$s = \frac{t}{2} \times 0.034 = 294 \times 10^{-6} \times 0.034 = 9.96\ cm$$

OPERATION – ULTRASONIC SENSOR

- Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst
- This burst travels through the air at about 1,130 feet per second, hits an object, and then bounces back to the sensor
- The ultrasonic sensor provides a **HIGH** output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target

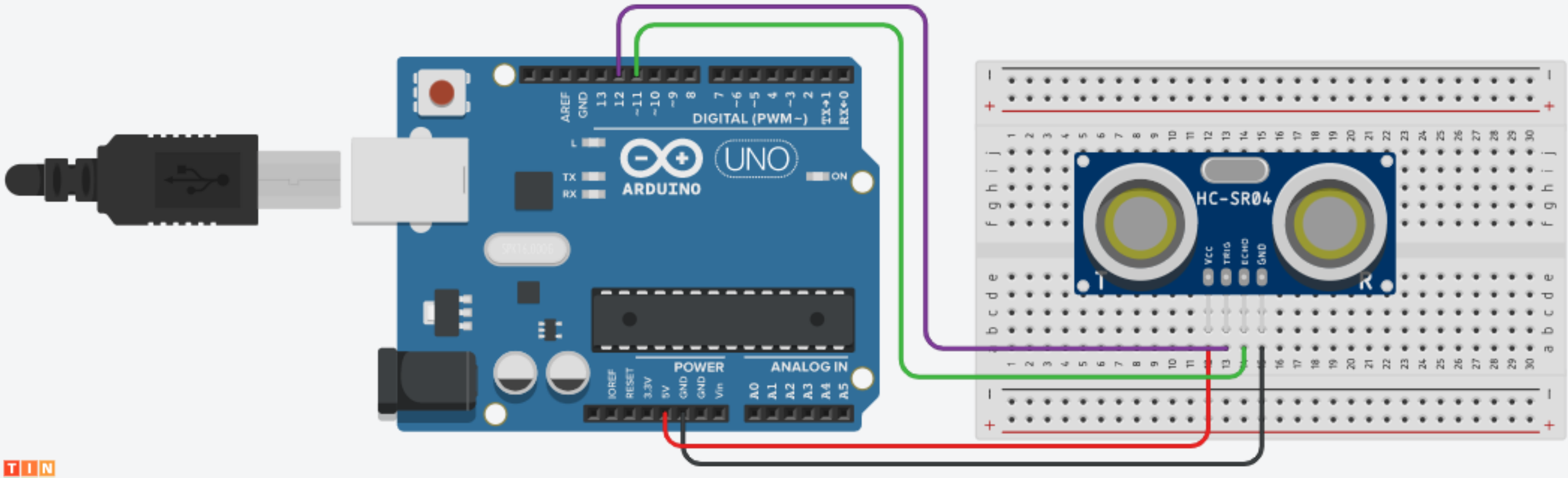
Source



Source

ACTIVITY - 1

Interfacing Ultrasonic Sensor with Arduino



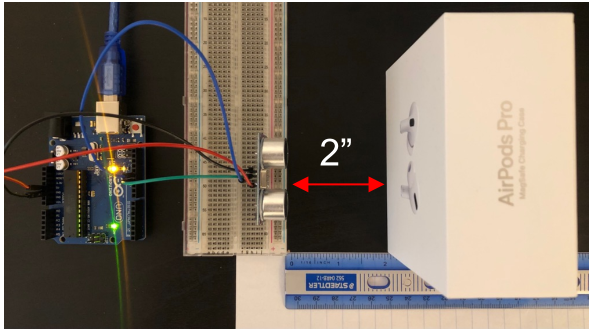
ARDUINO SKETCH – ULTRASONIC SENSOR

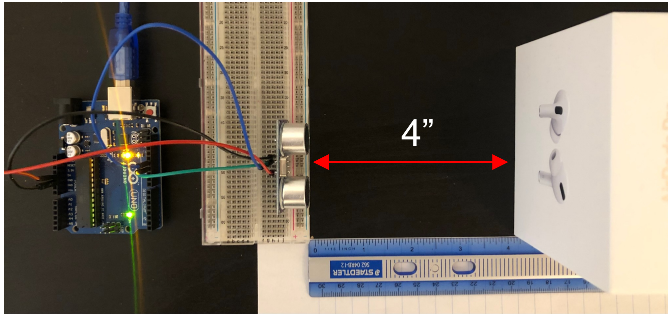
- Ultrasonic sensor is given a 5μs pulse by Arduino pin 12, configured as **OUTPUT**
- When the sensor receives the 5μs pulse, it emits a 40kHz burst of sound out of its "speaker"
- Sensor waits for sound to reflect off of something and return to "microphone" where it is detected; the sensor then sets Arduino pin 11 to **LOW**
- Arduino uses the **pulseIn** command to measure the time of flight of the sound wave in microseconds (the time for which pin 11 is **HIGH**)

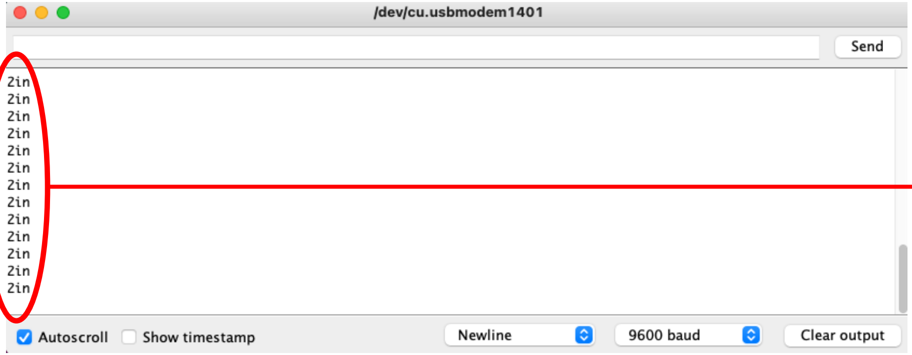
```
void setup() {
  Serial.begin(9600);
}

void loop()  {
  long duration, inches;

  pinMode(12, OUTPUT); // pin 12- connected to trigger pin
  pinMode(11, INPUT); // pin 11- connected to echo pin

  // send a 5 microsecond pulse out pin 12
  digitalWrite(12, LOW);
  delayMicroseconds(2);
  digitalWrite(12, HIGH);
  delayMicroseconds(5);
  digitalWrite(12, LOW);

  /* measure the time of flight of sound wave */
  duration = pulseIn(11, HIGH);

  /* 1130 ft/s * 12in/ft * 1s/1,000,000us = 74*/
  /* factor of 2 since sound travels out and back */

  inches = duration / 74 / 2;

  Serial.print(inches);          // display distance in inches
  Serial.print("in ");
  Serial.println();
}
```

Program

ACTIVITY 1 - DEMONSTRATION



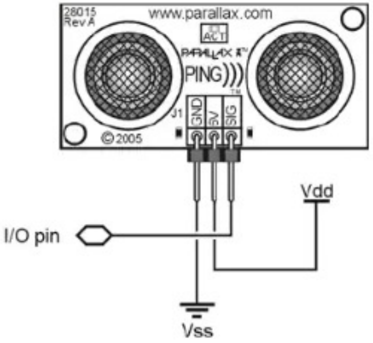




PING))) ULTRASONIC DISTANCE SENSOR

Source

Source



The PING))) sensor by Parallax Inc., has a range of 2cm to 3 meters, while the HC-SR04 has a range of 2cm to 4 meters

It requires only 3 pins (5V, SIG & GND) compared to HC-SR04 that required 4 pins (5V, TRIG, ECH & GND)

Source: PARALLAX PING SENSOR DATASHEET

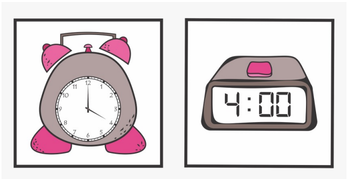
Source: HC-SR04 SENSOR DATASHEET

ANALOG TO DIGITAL CONVERSION (ADC)

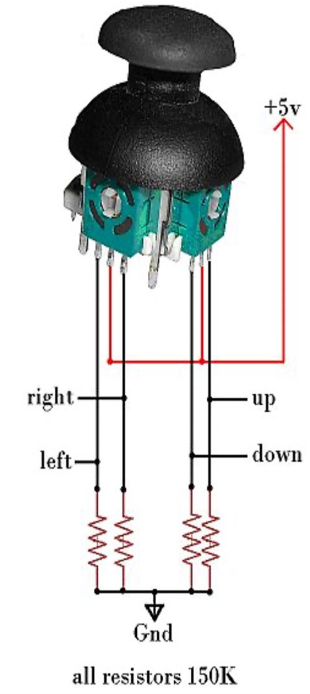
- **What is an Analog signal?**

It is continuous signal with a range of values (in case of Arduino, the voltage reading can be any value in the range of 0-5V, not just 0 or 5V)

- **Why convert to digital?**
    - A microcontroller only understands digital signals, such as whether a button is pressed or not
    - It considers zero volts (0V) as a binary 0 and a five volts (5V) as a binary 1
    - However, sensors can output 0.01V or 4.99V or anything in between, for example, 2.6V, 1.2V, etc.
    - Here is where ADC is necessary to convert that analog input into a binary signal to make decisions



Source



Source

ANALOG TO DIGITAL CONVERSION (ADC)

- Lately, most microcontrollers have built-in ADC modules

- The Arduino Uno board contains **6 pins for ADC**

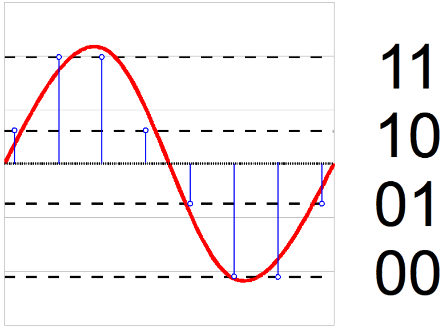
- It has a **10-bit analog to digital converter**: maps input voltages between 0 and 5 volts into integer values between 0 and 1023 (1024 steps)

- Older microcontrollers such as the Intel 8051 do not have a built-in ADC device, but external ADC module can be used (beyond the scope of this course)

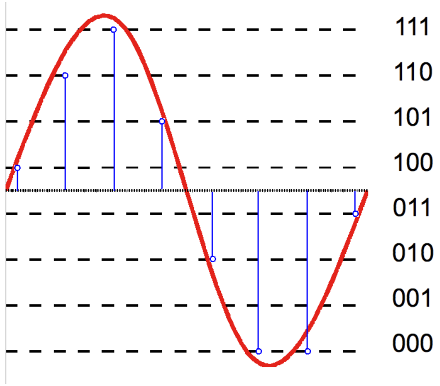
Source

ANALOG TO DIGITAL CONVERSION

- Resolution: number of different voltage levels
- Arduino's resolution is 10 bits, i.e., 2^{10} states = 1024 states
- Smallest measurable voltage change is (5/1023)V = 4.8mV
- The ADC turns the analog voltage into a digital value
- The function that you use to obtain the value of an analog signal is analogRead(pin)



Source

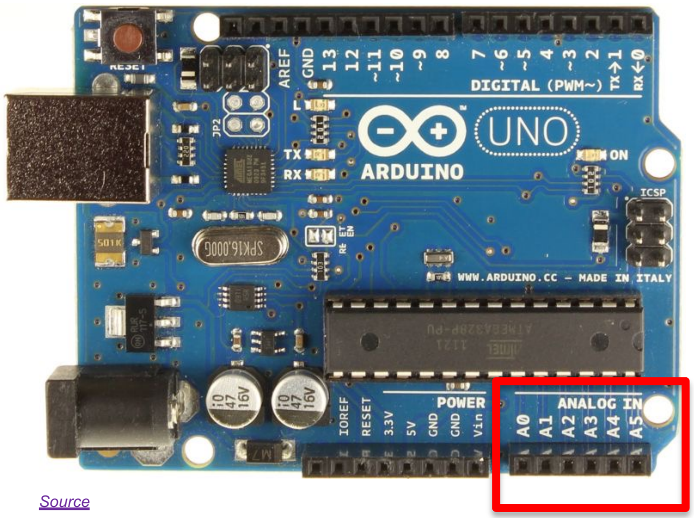


Source

READING ANALOG VALUES

Note: 1 μs = 10^{-6} s

- **analogRead(A0):**
  - This command is used to read the analog value from the specified analog pin (here, pin A0)
  - Reads an integer between 0 and 1023 from pin
  - The usable pins on the Arduino UNO for this function are A0 to A5
  - It takes about 100 μs for an Arduino to read an analog input

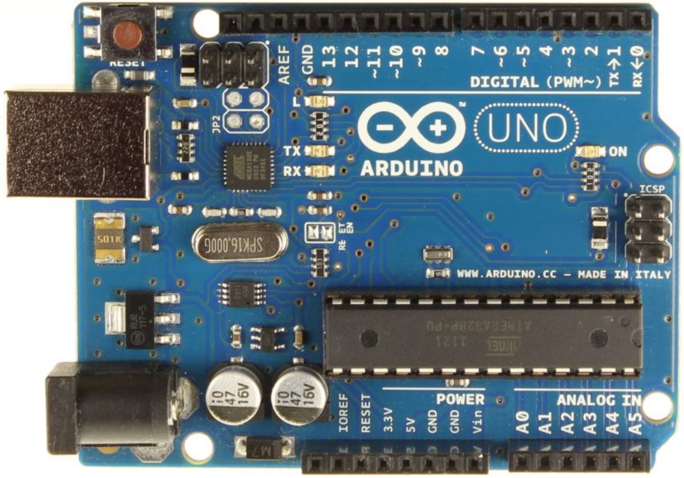


Source

WRITING ANALOG VALUES

- **analogWrite(3,128):**
  - Writes an analog value (PWM wave) to a pin (here, pin 3)
  - Writes an integer between 0 and 255 to a pin (here, the value is 128)
  - The Arduino does not have a built-in digital-to-analog converter (DAC), but it can pulse-width modulate (**PWM**) a digital signal to achieve some of the functions of an analog output
  - It can be used to control LED brightness or drive a motor at various speeds

PWM pins - 3,5,6,9,10,11



Source

CONVERTING ANALOG VALUE TO DIGITAL

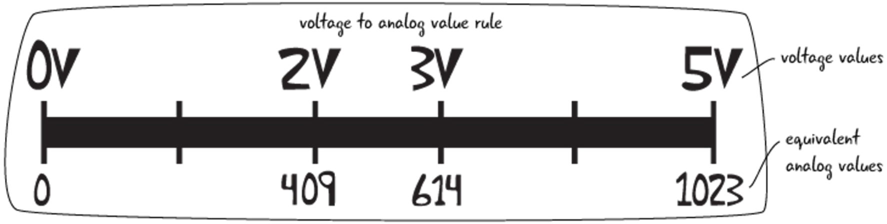
This is the variable will hold the actual voltage value

This is the value returned by analogRead(), it in the range 0 - 1023

```
float voltage = sensorValue * (5.0 / 1023.0);
```

A float is a data type that has a decimal point. Floats can be huge numbers but require double the memory (4 bytes) compared to integers (2 bytes)

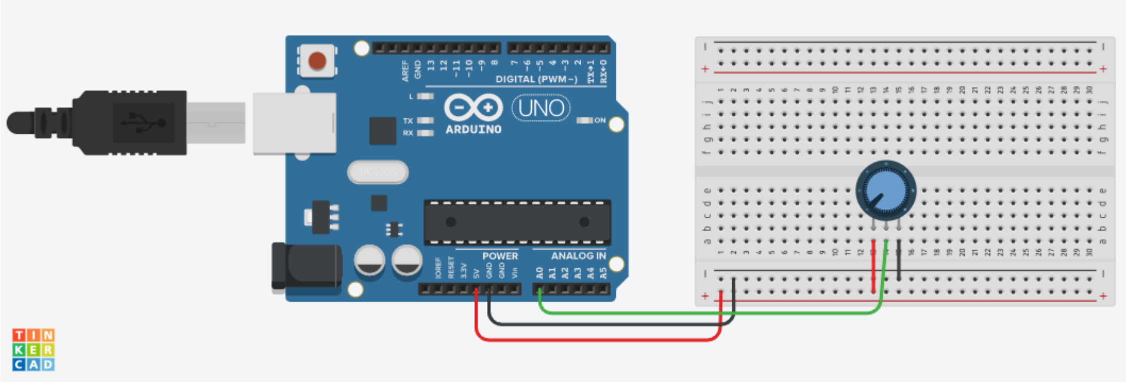
This is a conversion factor used to change the scale from 0 - 1023 range to 0 - 5 range



Source

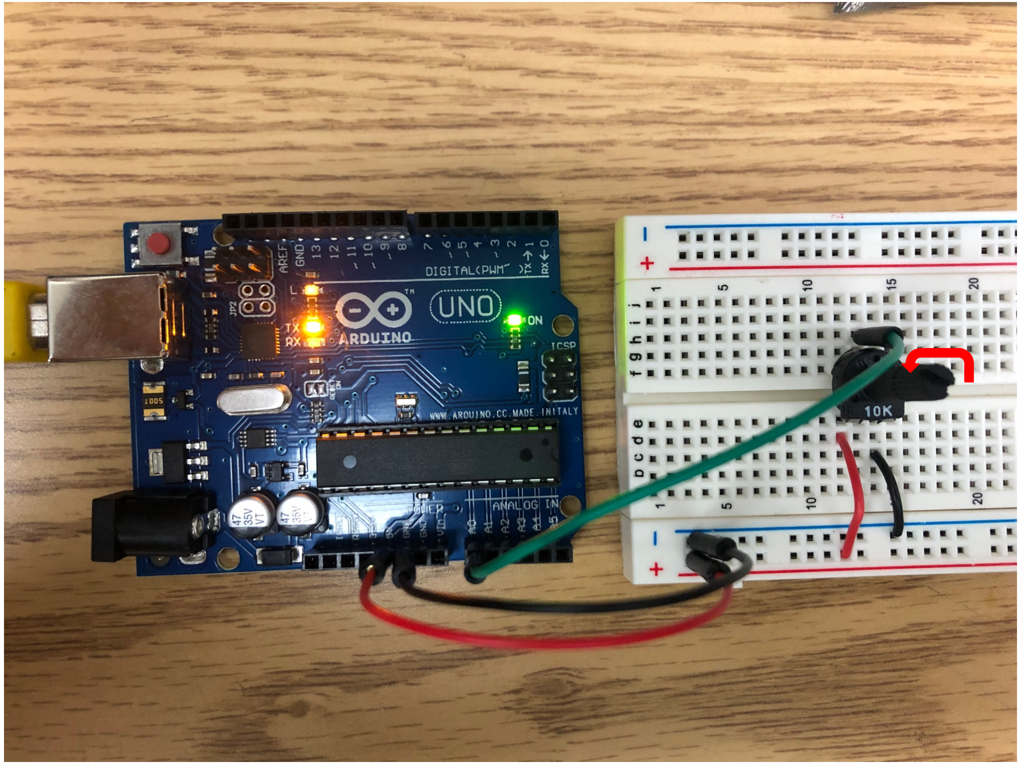
ADC EXAMPLE - POTENTIOMETER

```
int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
int sensorValue = 0;        // value read from the pot

float sensor_voltRead = 0;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  sensorValue = analogRead(analogInPin);          // read the discrete value:
  sensor_voltRead = (sensorValue/1023.0)*5.0;   //voltage reading of POT

  // print the results to the serial monitor:
  Serial.flush();
  Serial.print("sensor = " );
  Serial.println(sensorValue);
  Serial.print("sensor voltage reading = " );
  Serial.println(sensor_voltRead);
  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```



Program

ADC EXAMPLE - POTENTIOMETER

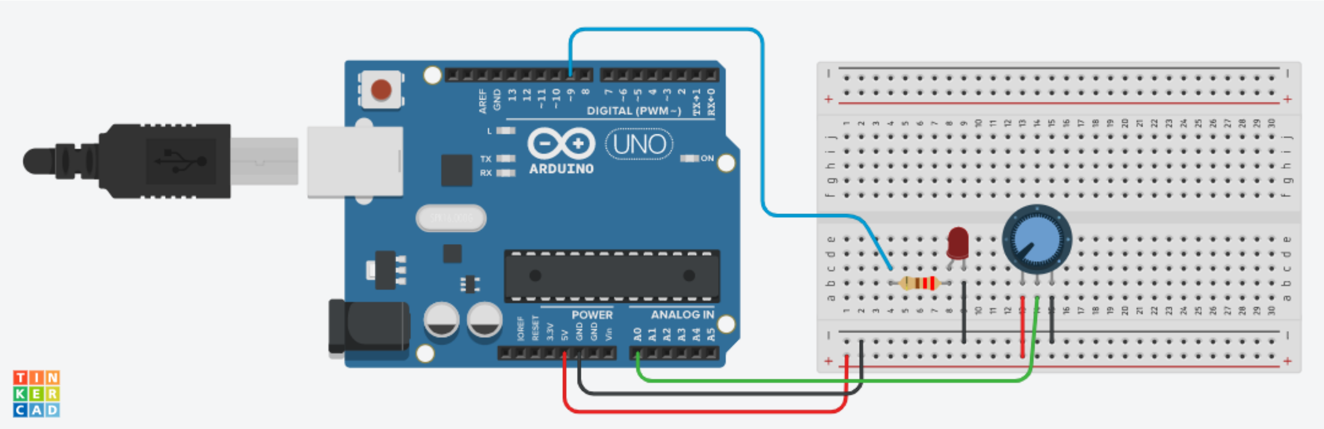



```
sensor = 0
sensor voltage reading = 0.00
sensor = 0
sensor voltage reading = 0.00
sensor = 0
sensor voltage reading = 0.00
sensor = 0
sensor voltage reading = 0.00
sensor = 0
sensor voltage reading = 0.00
sensor = 101
sensor voltage reading = 0.49
sensor = 188
sensor voltage reading = 0.92
sensor = 391
sensor voltage reading = 1.91
sensor = 517
sensor voltage reading = 2.53
sensor = 624
sensor voltage reading = 3.05
sensor = 750
sensor voltage reading = 3.67
sensor = 860
sensor voltage reading = 4.20
sensor = 955
sensor voltage reading = 4.67
sensor = 1022
sensor voltage reading = 5.00
sensor = 1023
sensor voltage reading = 5.00
sensor = 1023
```

ADC EXAMPLE – LED BRIGHTNESS CONTROL

Potentiometer to Control LED's Brightness

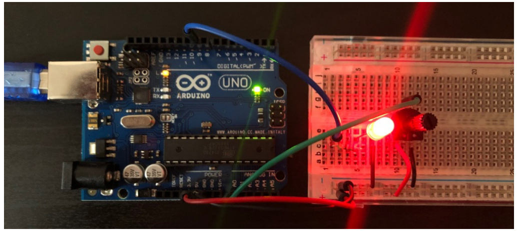
```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to
// declared as constants since the values do not change

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);

  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```



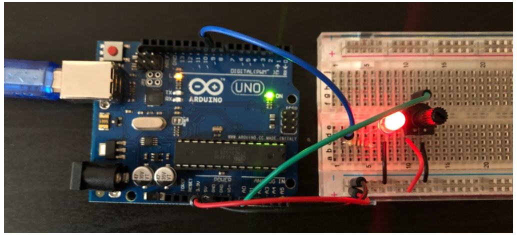
Program

LED BRIGHTNESS CONTROL

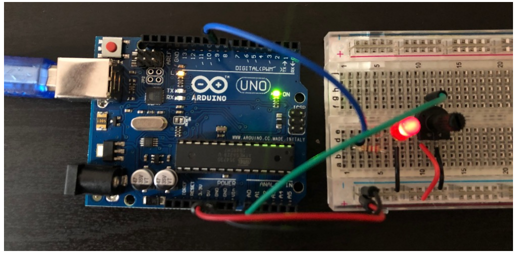
Potentiometer to Control LED's Brightness




```
sensor = 963    output = 240
sensor = 963    output = 240
sensor = 963    output = 240
```



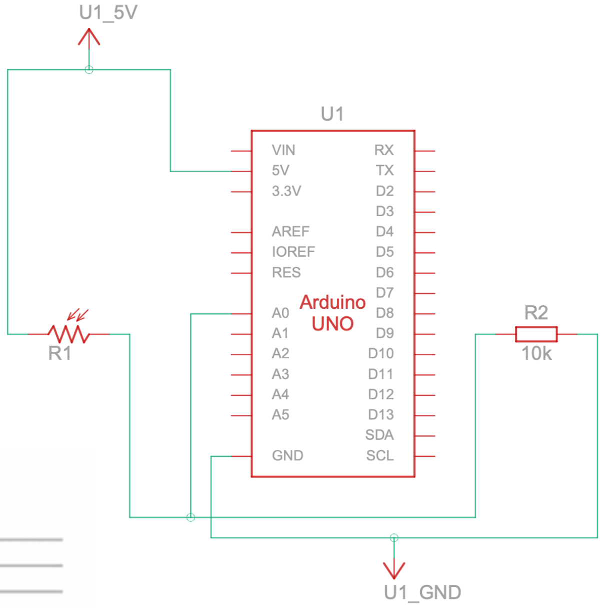

```
sensor = 165    output = 41
sensor = 165    output = 41
sensor = 165    output = 41
```



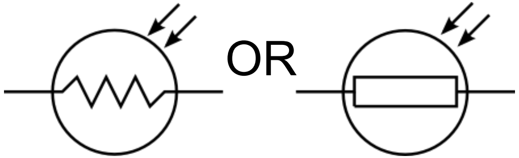

```
sensor = 10    output = 2
sensor = 10    output = 2
sensor = 10    output = 2
```

LIGHT SENSOR - PHOTORESISTOR

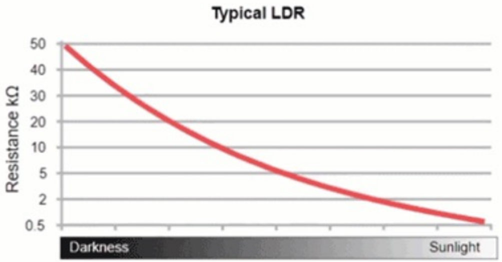
- **Photoresistors (LDR)** are used when we need a resistive analog response to a light level
- Photoresistors are analog devices that produce an analog change in the resistance value based on received light
- Photoresistors can be setup in an electric circuit to function as a digital sensor that turns on/off based on the light being above/below certain threshold



Circuit symbol:

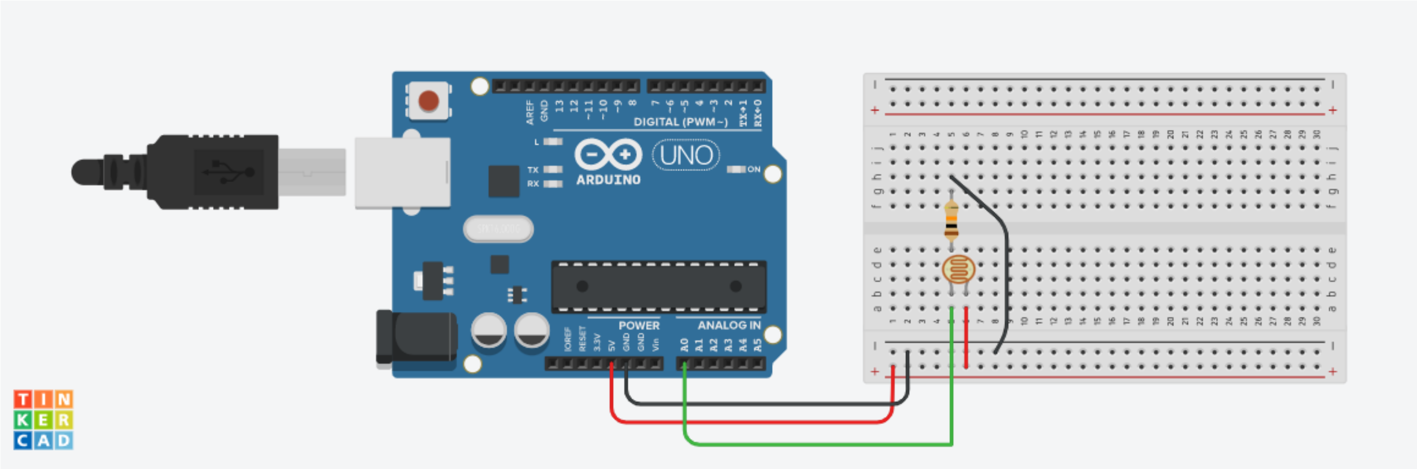


Source



Source

INTERFACING LDR WITH ARDUINO

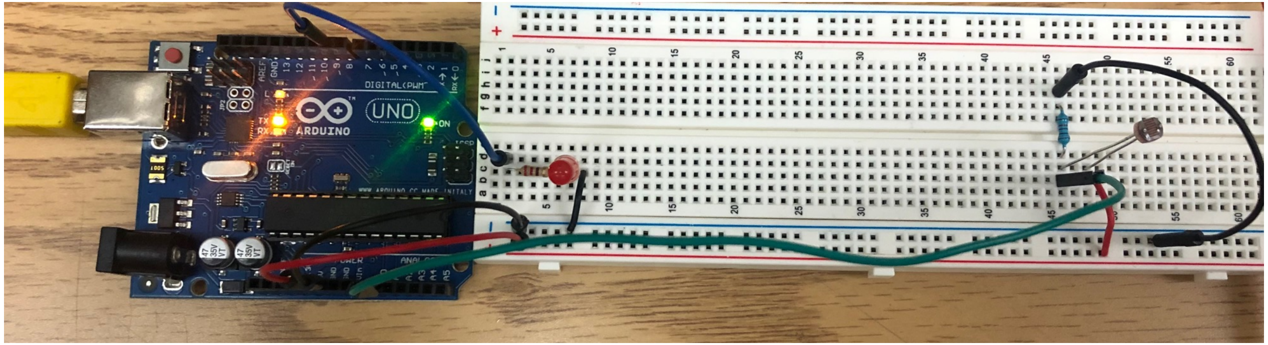


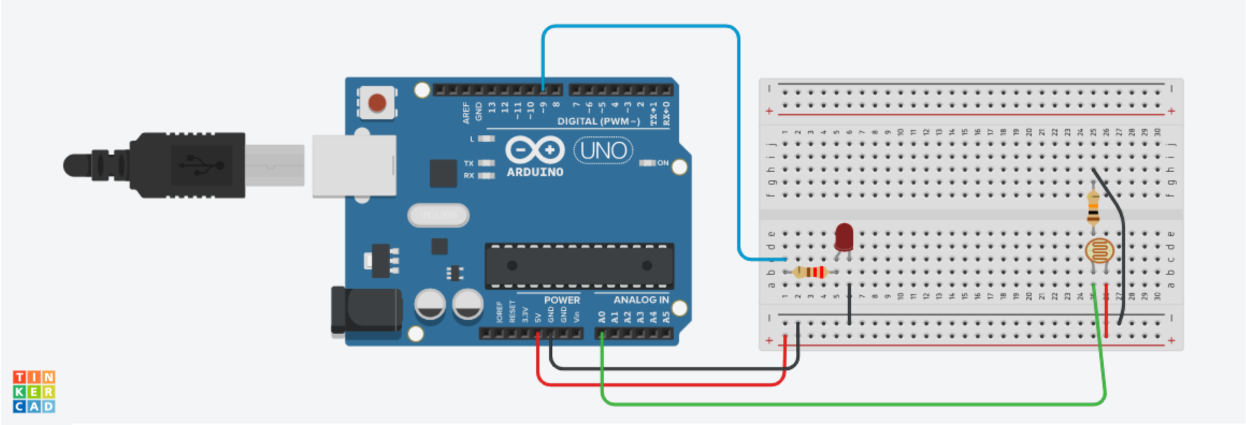
ACTIVITY - 2

Use LDR to turn a LED ON (if the light increases, LED switches off and if there is less light then LED switches on)

- Make relevant circuit diagram for LDR connection
- Use ADC of Arduino to accomplish the task (switch LED on and LED off)
- Same task- Rather than switching the LED on/ off – change the brightness of the LED depending upon the light intensity level at LDR

ACTIVITY 2 - CIRCUIT

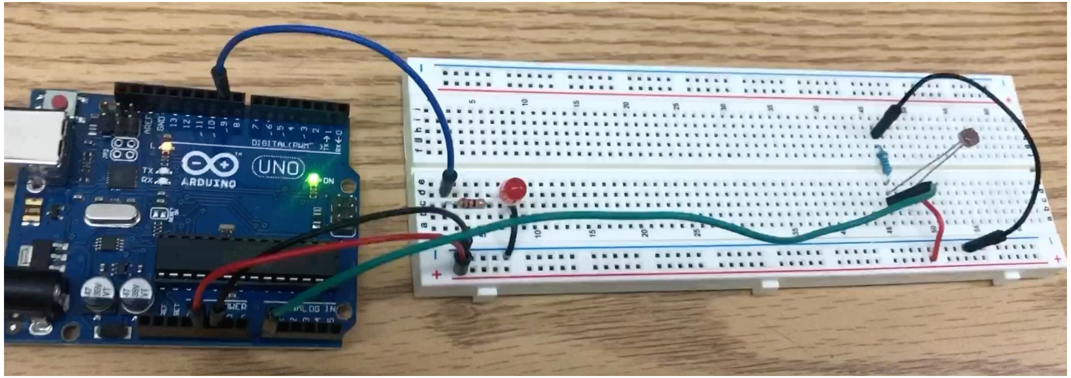




DEMO – LED ON/OFF USING LDR

```
const int ldr = A0; // Analog input pin that the LDR is attached to
const int led = 9; // Analog output pin that the LED is attached to
// declared as constant integers since the values do not change
int sensorValue = 0;

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

void loop() {
  // read the analog in value of LDR
  sensorValue=analogRead(ldr);

  if(sensorValue<500) // check for LDR thershold value
    {
      digitalWrite(led,HIGH);
      /* Makes the LED glow if it is dark enough
      (resitance less than threshold value)*/
    }
    else
    {
      digitalWrite(led,LOW);
      /*Turns the LED OFF when it is brighter
      (resitance greater than threshold value) */
    }

   // 2 milliseconds delay before the next loop
   // for ADC to settle after last reading
   delay(2);
}
```

Program

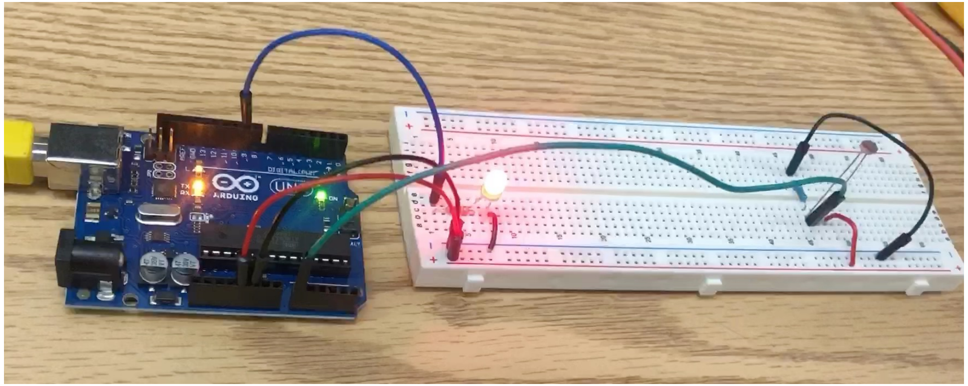


LED On/Off control using LDR (Video demo)

DEMO – LED BRIGHTNESS CONTROL USING LDR

```
const int ldr = A0; // Analog input pin that the LDR is attached to
const int led = 9; // Analog output pin that the LED is attached to
// declared as constant integers since the values do not change
int sensorValue = 0;        // value read from the LDR
int outputValue = 0;      // value output to the PWM (analog out)

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

void loop() {
  // read the analog in value of LDR
  sensorValue=analogRead(ldr);

  // map it to the range of the analog out:
  outputValue = map(sensorValue, 50, 700, 0, 255);
  analogWrite(led, outputValue);

  delay(2); // 2 milliseconds delay before the next loop
  // for ADC to settle after last reading
}
```

Program



LED brightness control using LDR (Video Demo)

SENSOR APPLICATIONS

Autonomous navigation

Source

Crop-line detection for autonomous harvesting

Source

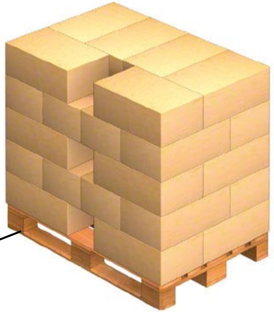
Plant monitoring & sample collection

Source

Automated guide vehicle (line or tag following)

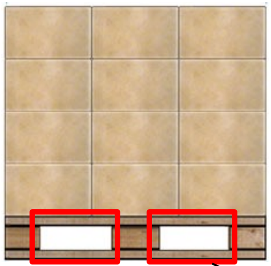
SENSOR APPLICATIONS

Warehouse (Material-handling)



Pallet

Source



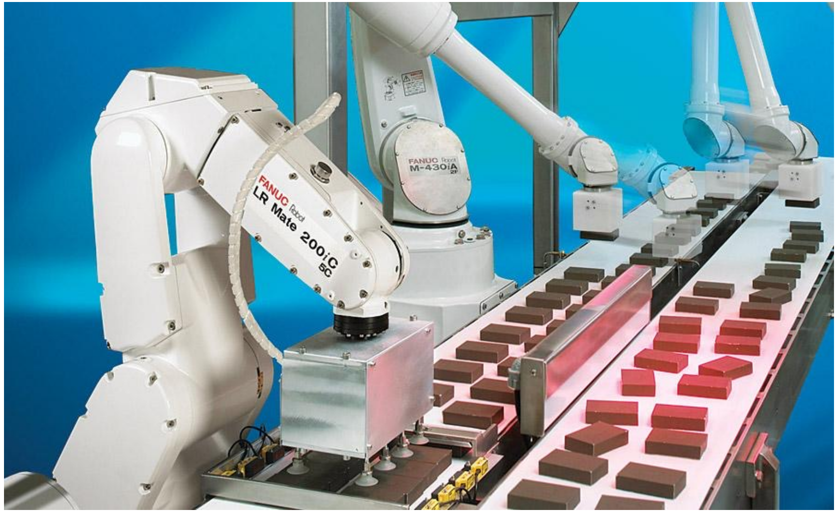
Source

Slot

Source

Estimation of pallet orientation (slots)

Manufacturing



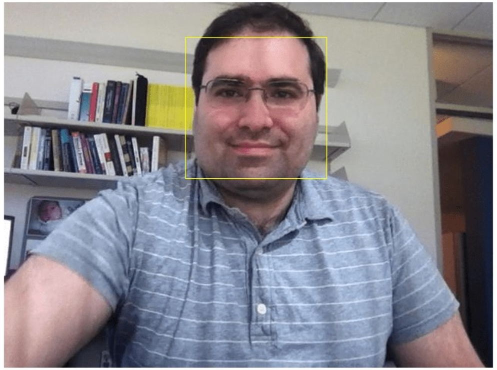
Source

Quality inspection and predictive maintenance

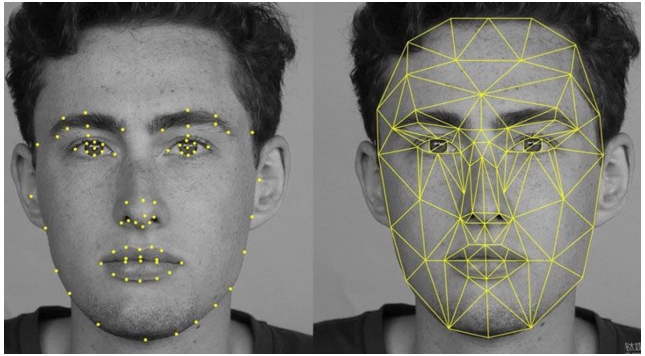
SENSOR APPLICATIONS

Face detection & tracking

Biometric data



Source



Source

Source

Detection a face in an image or tracking face in a video

Face recognition

Fingerprint scanner

Task / Activity:
(Reading sensors with Arduino)

TASK / ACTIVITY

- Objectives, goals, mission of task / to do activity: Learn how different types of Sensors work, and read sensor output using Arduino microcontroller
- Subtask 1: Use LDR to blink a LED (more light → led switch off, less light → led switch on)
  - Make relevant circuit diagram for LDR connection
  - Use ADC of Arduino to accomplish the task (switch LED on and LED off)
  - Same task- Rather than switching the LED on/ off – change the brightness of the LED depending upon the light intensity level at LDR
- Subtask 2: Interfacing ultrasonic sensor using Arduino to read object's distance

NYU TANDON SCHOOL OF ENGINEERING

Thank You!

Questions and Feedback?