DES535
Ubiquitous Computing

Dr. Pragma Kar
Assistant Professor
Department of Human-Centered Design

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY **DELHI**

Google Classroom Code : pcwnf5t

Ambient & Context-Aware Computing

Module III

Context-aware Computing

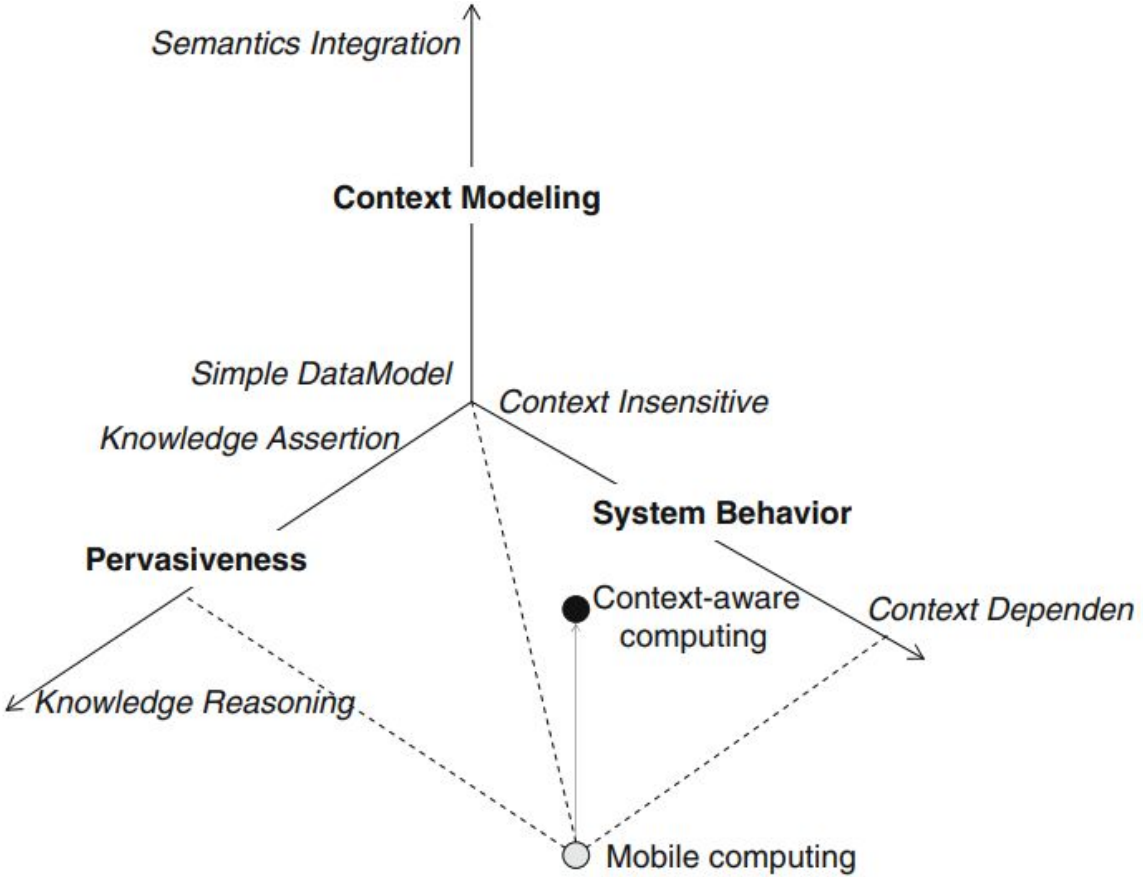
- A computing system that can **sense** and process information about the users' **context** and **adapt** its behaviour (response) (semi)**automatically** in (near)**realtime**.
- Example :
  - Smart Classroom : Dimming light based on whether the instructor is presenting or delivering a speech. Tracking students' activities to estimate their attention.
  - Smart Office : Sensing the number of occupants based on the level of CO2. Recognising occupant's stress and suggesting relaxing exercises.
  - Smart Vehicle : Alerting the sleepy driver. Understanding driver's driving pattern and suggest the optimal route.

Context Modeling

- The context modeling is a research area that tries to answer the following questions:
  - which are the most appropriate contextual information that can model well enough the specific context in a certain domain (e.g., epistemic field, such as mobility behavior of a user, network congestion control, early warning, and transportation traffic notification),
  - which are the relations among such pieces of information,
  - how can one take into account the information change, and, how can one react to such change, if necessary.
- Example : Tracking and maintaining attention
  - Contextual information/ elements : Time (t), Action (a), Location (l), User (u)
  - Relation : t is “night”, l is in “bedroom”, a of u is “writing”
  - Reaction : “put all notifications on silent”

https://link.springer.com/content/pdf/10.1007/s11277-006-9187-6.pdf

Context-aware Logic Space



https://link.springer.com/content/pdf/10.1007/s11277-006-9187-6.pdf

Example of an Application in the Context-aware Logic Space

- Automatic plant watering
  - Context Modeling : Do you need a simple data model of semantic integration to differentiate?
  - Pervasiveness : Can simple knowledge assertion work or knowledge reasoning is required?
  - System behaviour : Should the system be loosely context-aware or totally context dependent?

Example of an Application in the Context-aware Logic Space

- Context Modeling : Do you need a simple data model of semantic integration to differentiate?
- Pervasiveness : Can simple knowledge assertion work or knowledge reasoning is required?
- System behaviour : Should the system be loosely context-aware or totally context dependent?

Requirements of Context-Aware Applications

- **Context acquisition:** A mechanism to obtain context data from diverse context sources. Context acquisition could be dealt with hardware sensors delivering information that conforms to a low-level data model.
  - *Eg. ambient light level, IMU, noise level, CO2.*
- **Context aggregation**: A mechanism that provides context storing and integrity. In case of a shared context model, the context aggregation forms a basis for merging correlated contextual information. The context composition is a specific kind of context aggregation, when the involved contexts are compatible with the same, or equivalent, context model.
  - *Source 1 , Source 2, …. Source n-> aggregated value*
- **Context consistency:** Context consistency enables the rationality of dynamically changing distributed context models. Such mechanism, regarded as being an extended context aggregation mechanism, maintains the structure of the contextual model into higher levels of abstraction.
  - *Change in source 1 is reflected on the other sources and hence, the system.*

https://link.springer.com/content/pdf/10.1007/s11277-006-9187-6.pdf

Requirements of Context-Aware Applications

- **Context discovery:** The aim of context discovery is to locate, and access contextual sources, in terms of serving context requests (e.g., discovering the appropriate, or approximate, context pertinent to an entity).
  - *You enter C212, the class attendance app discovers the instructor, ambient details and CO2 level of the room.*
- **Context query:** By exploring contextual information, residing in distributed context repositories, a reference model needs a high-level mechanism for posing queries. Complex context retrieval tasks (e.g., queries as list all persons in the same conference hall whose presentation is at the same time with mine) must be transparent to end-users.
- **Context adaptation:** The application should be capable of adapting its behavior according to contextual information. Specifically, it, automatically, adapts the system configuration in response to a contextual change.
  - *You decide to leave C212. The app shows the CO2 level has decreased and the attendance has decreased too.*

https://link.springer.com/content/pdf/10.1007/s11277-006-9187-6.pdf

Requirements of Context-Aware Applications

- **Context reasoning:** Context can be elaborated with reasoning mechanisms. Context reasoning is a process for inferring new context, previously unidentified on the basis of a-priori known context. Reasoning tasks check context consistency and deduce high-level context.
  - *5 students leave the room. The app indicates the lecture is not engaging.*
- **Context quality indicators:** Context data can come from heterogeneous context sources, such as, sensors, and software services. A mechanism for maintaining predefined sets of quality indicators is very important. Such indicators may be resolution, accuracy, repeatability, frequency, and staleness of context.
- **Context integration:** Existing context models vary in the expressiveness they support, in semantics, and in the abstraction level of the conceptual entities. Contextual information integration (i.e., contextual semantic integration of the individuals of an ontology) can be conducted whenever different context models are in accordance, not only, with their semantics, but, also, with their similar domains of interest.
  - <Source 1 : Location, CO2><Source 2: lux, location> –>Integrated schema: Location, CO2, Lux

https://link.springer.com/content/pdf/10.1007/s11277-006-9187-6.pdf

Demonstration

How to sense context (simple data model)

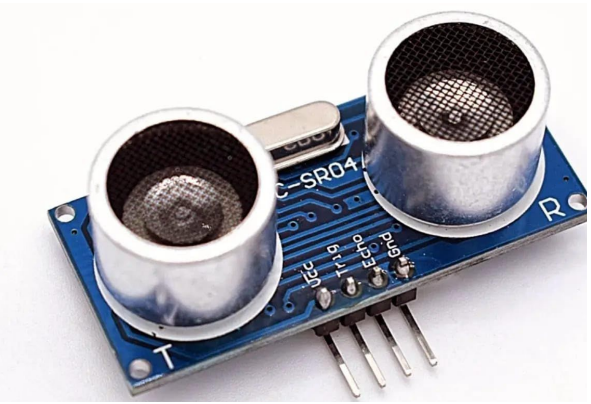
ARDUINO SENSORS

1. HCR
2. Soil Humidity
3. Obstacle Proximity

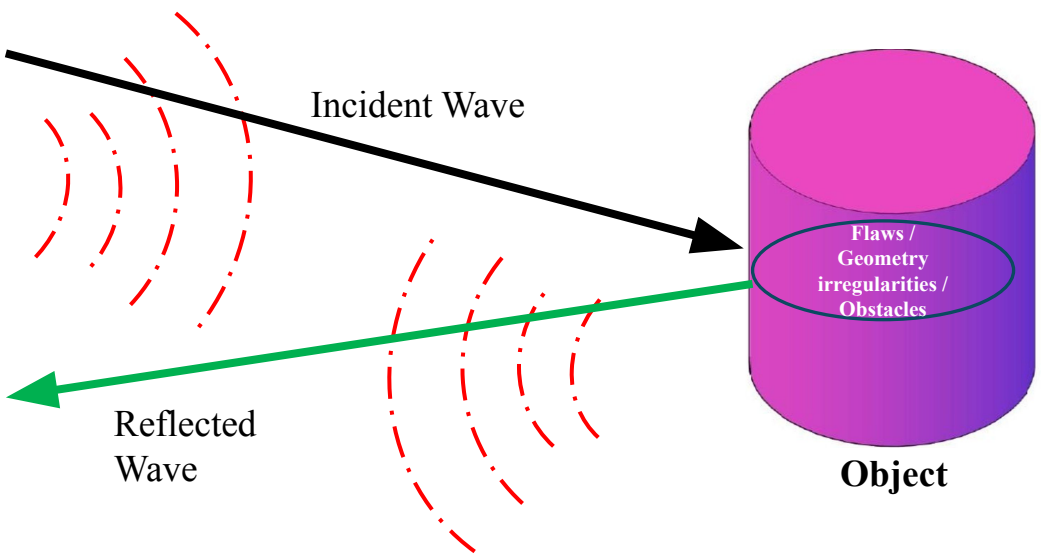
1.HC SR-SENSOR

ToF (Time of Flight): $\frac{Distance}{Speed}$

Receiver Pulse Duration: Time (T)
Speed of sound : Speed (V)



HCR Sensor

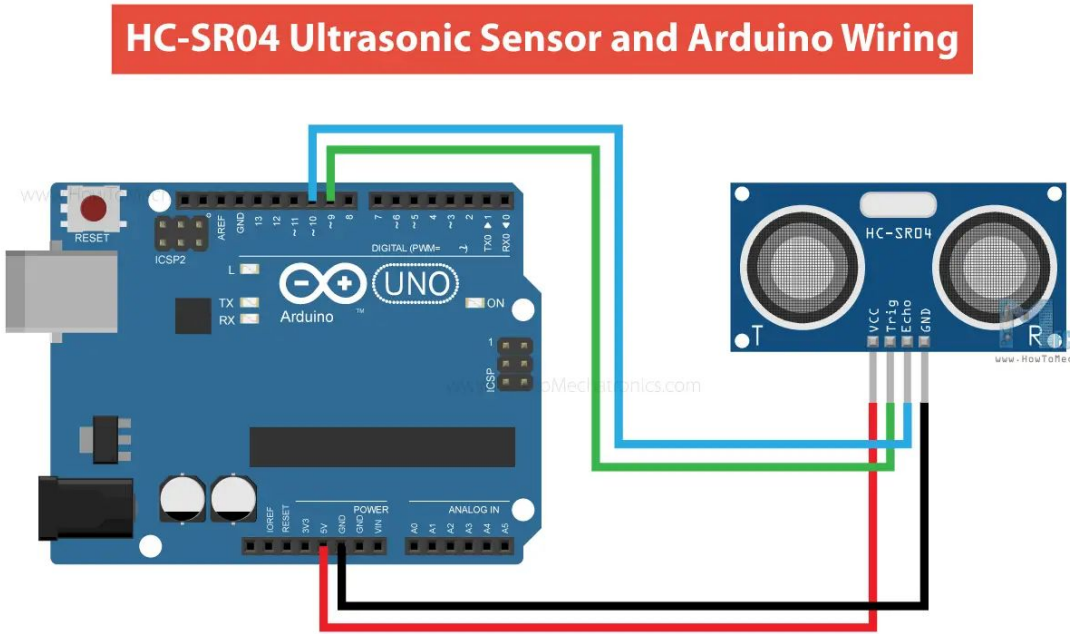


Distance = (Speed x Time) / 2 = (34cm/ms x 1.5ms) / 2 = 25.5cm

- ❑ Ultrasonic Sensor HC-SR04
- ❑ Arduino Board
- ❑ Breadboard and Jump Wires

WIRING CONNECTION:

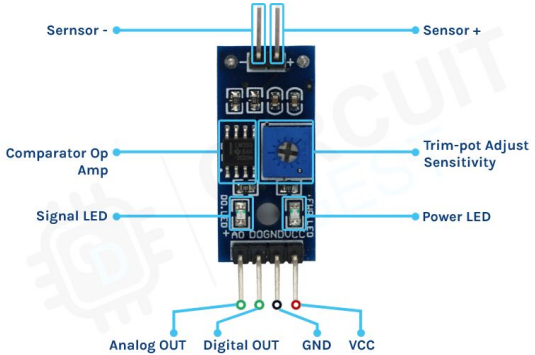
Distance measuring sensor which has a range from 2cm to 400cm



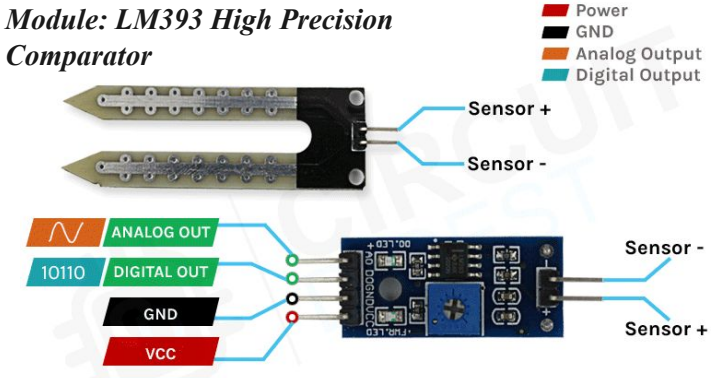
Code:/*Ultrasonic Sensor HC-SR04 and Arduino Tutorial*/

```
// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;
// defines variables
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}
void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in
microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);
}
```

2. SOIL MOISTURE SENSOR WITH ARDUINO:



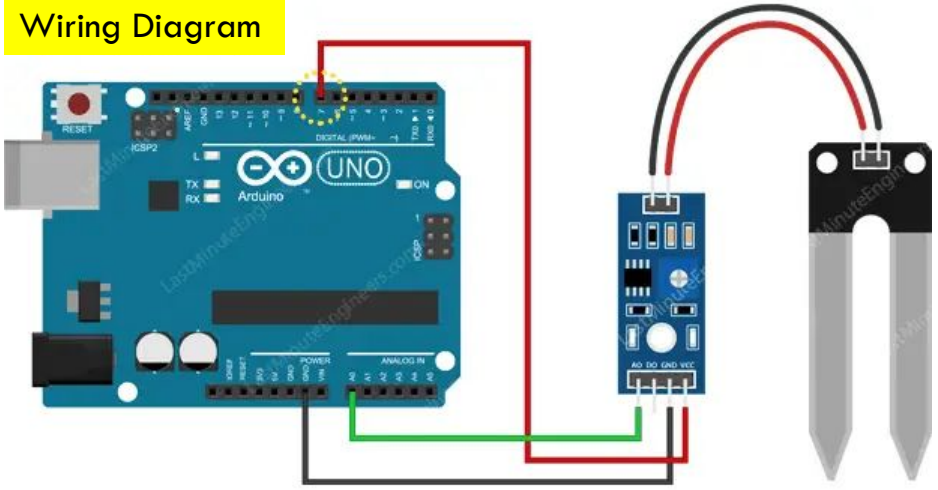
Module: LM393 High Precision Comparator



Electrode Probe

- *Resistance varies inversely with soil moisture*
- *The more water in the soil, the better the conductivity and the lower the resistance*
- *The less water in the soil, the lower the conductivity and thus the higher the resistance*

Wiring Diagram



CODE:

Moisture Value Measurement

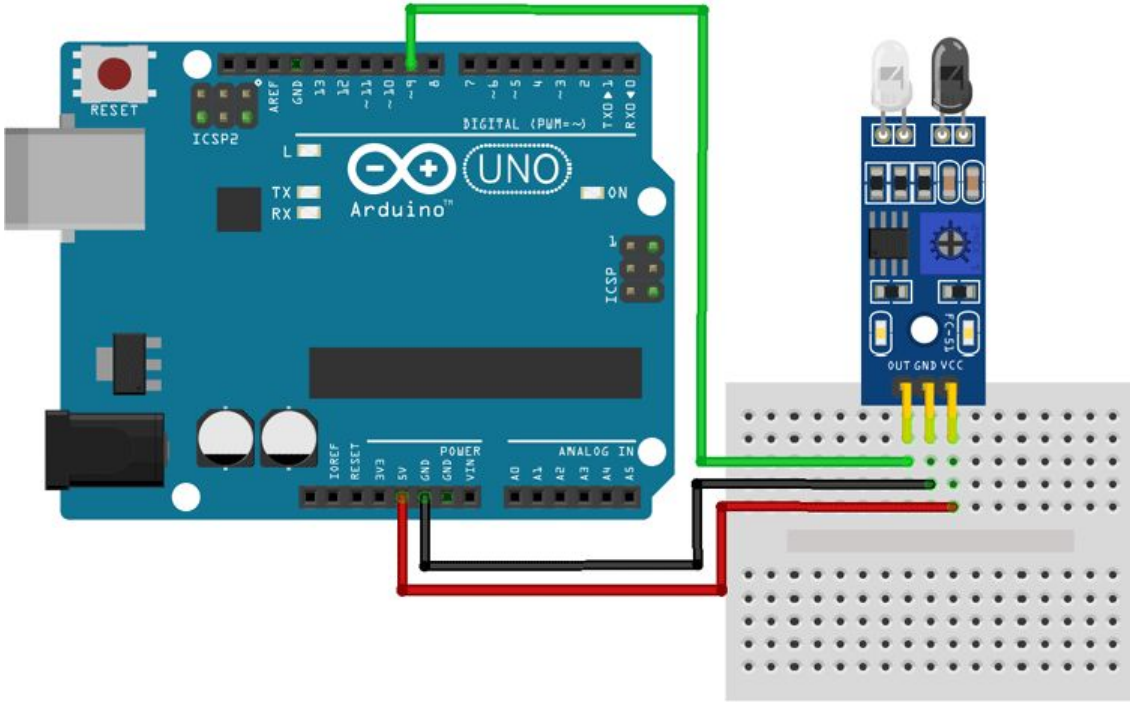
```
// Sensor pins
#define sensorPower 7
#define sensorPin A0

void setup() {
	pinMode(sensorPower, OUTPUT);

	// Initially keep the sensor OFF
	digitalWrite(sensorPower, LOW);

	Serial.begin(9600);
}

void loop() {
	//get the reading from the function below and print it
	Serial.print("Analog output: ");
	Serial.println(readSensor());

	delay(1000);
}

//  This function returns the analog soil moisture measurement
int readSensor() {
	digitalWrite(sensorPower, HIGH);	// Turn the sensor
ON
	delay(10);
				// Allow power to settle
	int val = analogRead(sensorPin);	// Read the
analog value form sensor
	digitalWrite(sensorPower, LOW);			// Turn
the sensor OFF
	return val;
				// Return analog moisture value
}
```

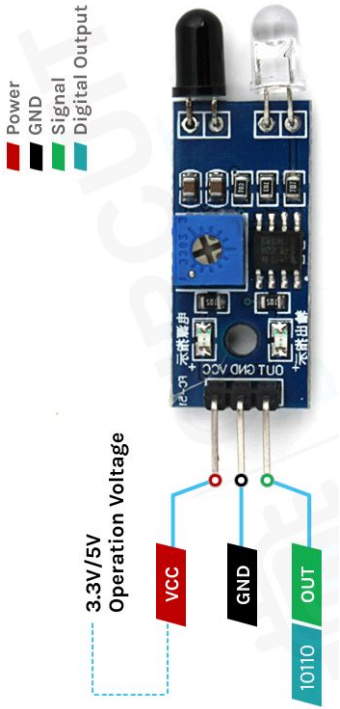
Quality of moisture

```
/* Change these values based on your calibration values */
#define soilWet 500   // Define max value we consider soil 'wet'
#define soilDry 750   // Define min value we consider soil 'dry'

// Sensor pins
#define sensorPower 7
#define sensorPin A0

void setup() {
	pinMode(sensorPower, OUTPUT);

	// Initially keep the sensor OFF
	digitalWrite(sensorPower, LOW);

	Serial.begin(9600);
}

void loop() {
	//get the reading from the function below and print it
	int moisture = readSensor();
	Serial.print("Analog Output: ");
	Serial.println(moisture);

	// Determine status of our soil
	if (moisture < soilWet) {
		Serial.println("Status: Soil is too wet");
	} else if (moisture >= soilWet && moisture < soilDry) {
		Serial.println("Status: Soil moisture is perfect");
	} else {
		Serial.println("Status: Soil is too dry - time to water!");
	}

	delay(1000);// Take a reading every second for testing
						// Normally you should take reading
perhaps once or twice a day
	Serial.println();
}

//  This function returns the analog soil moisture measurement
int readSensor() {
	digitalWrite(sensorPower, HIGH);	// Turn the sensor ON
	delay(10);
//
Allow power to settle
	int val = analogRead(sensorPin);	// Read the analog value form sensor
	digitalWrite(sensorPower, LOW);		// Turn the sensor OFF
	return val;
//
Return analog moisture value
}
```

3. PROXIMITY SENSOR :

A proximity sensor (often simply prox) is a sensor able to detect the presence of nearby objects without any physical contact





Code

```
int IRSensor = 9; // connect IR sensor module to Arduino pin D9
int LED = 13; // connect LED to Arduino pin 13
void setup()
{
  Serial.begin(115200); // Init Serial at 115200 Baud Rate.
  Serial.println("Serial Working"); // Test to check if serial is working or
not
  pinMode(IRSensor, INPUT); // IR Sensor pin INPUT
  pinMode(LED, OUTPUT); // LED Pin Output
}
void loop(){
  int sensorStatus = digitalRead(IRSensor); // Set the GPIO as Input
  if (sensorStatus == 1) // Check if the pin high or not
  {
    // if the pin is high turn off the onboard Led
    digitalWrite(LED, LOW); // LED LOW
    Serial.println("Motion Detected!"); // print Motion Detected! on the
serial monitor window
  }
  else  {
    //else turn on the onboard LED
    digitalWrite(LED, HIGH); // LED High
    Serial.println("Motion Ended!"); // print Motion Ended! on the serial
monitor window
  }
}
```