





Forsk Team



Abhishek
Sr. Developer



Sourabh
Sr. Developer



Dr. Sylvester
Manager



Rohit C
Developer



Rohit M
Developer



Abul
Sr. Developer



Usha
Sr. UX Designer



Yogendra
Lead



Devendra
Developer



Kunal
Developer



IoT Bootcamp ?

- End to End coverage on the applications of IoT
- Understanding Business, Opportunity & Developments
- Covering the entire architecture of IoT
- Special emphasis on technologies in ecosystem



At the end of it

- You will have understanding of IoT, verticals, market potential, Alliances and Industry updates
- Understanding the hardware side of the IoT (Sensors, development Boards, Embedded Systems, protocols)
- Excellent understanding of the communication protocols & software technologies (Cloud, JSON, Web Services, Big Data, Analytics)



**INTERNET
of PEOPLE**



What exactly is the
**"INTERNET
of THINGS"?**



1 SENSORS & ACTUATORS

We are giving our world a digital nervous system. Location data using GPS sensors. Eyes and ears using cameras and microphones, along with sensory organs that can measure everything from temperature to pressure changes.

2 CONNECTIVITY

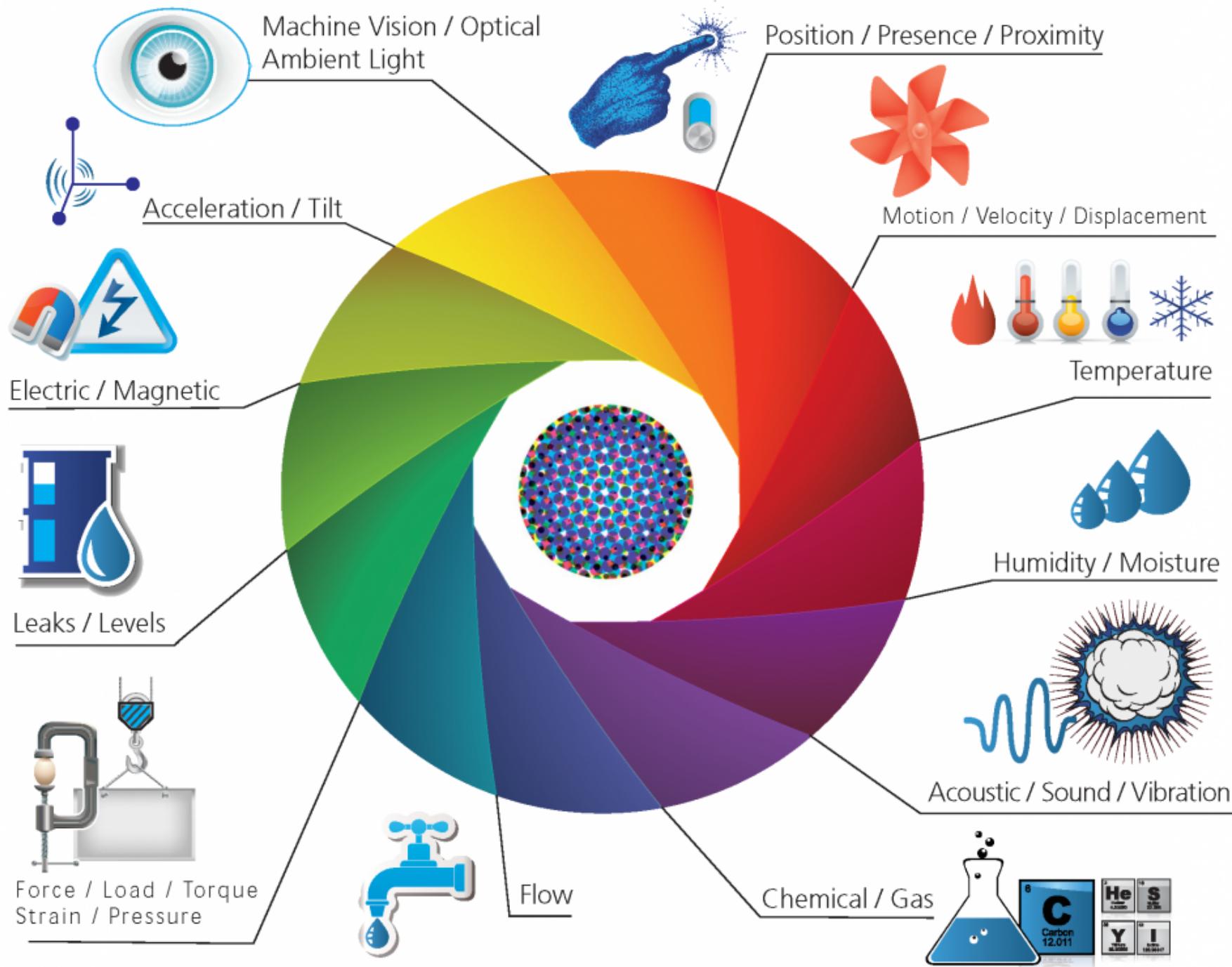
These inputs are digitized and placed onto networks.

3 PEOPLE & PROCESSES

These networked inputs can then be combined into bi-directional systems that integrate data, people, processes and systems for better decision making.

1 **SENSORS** & ACTUATORS

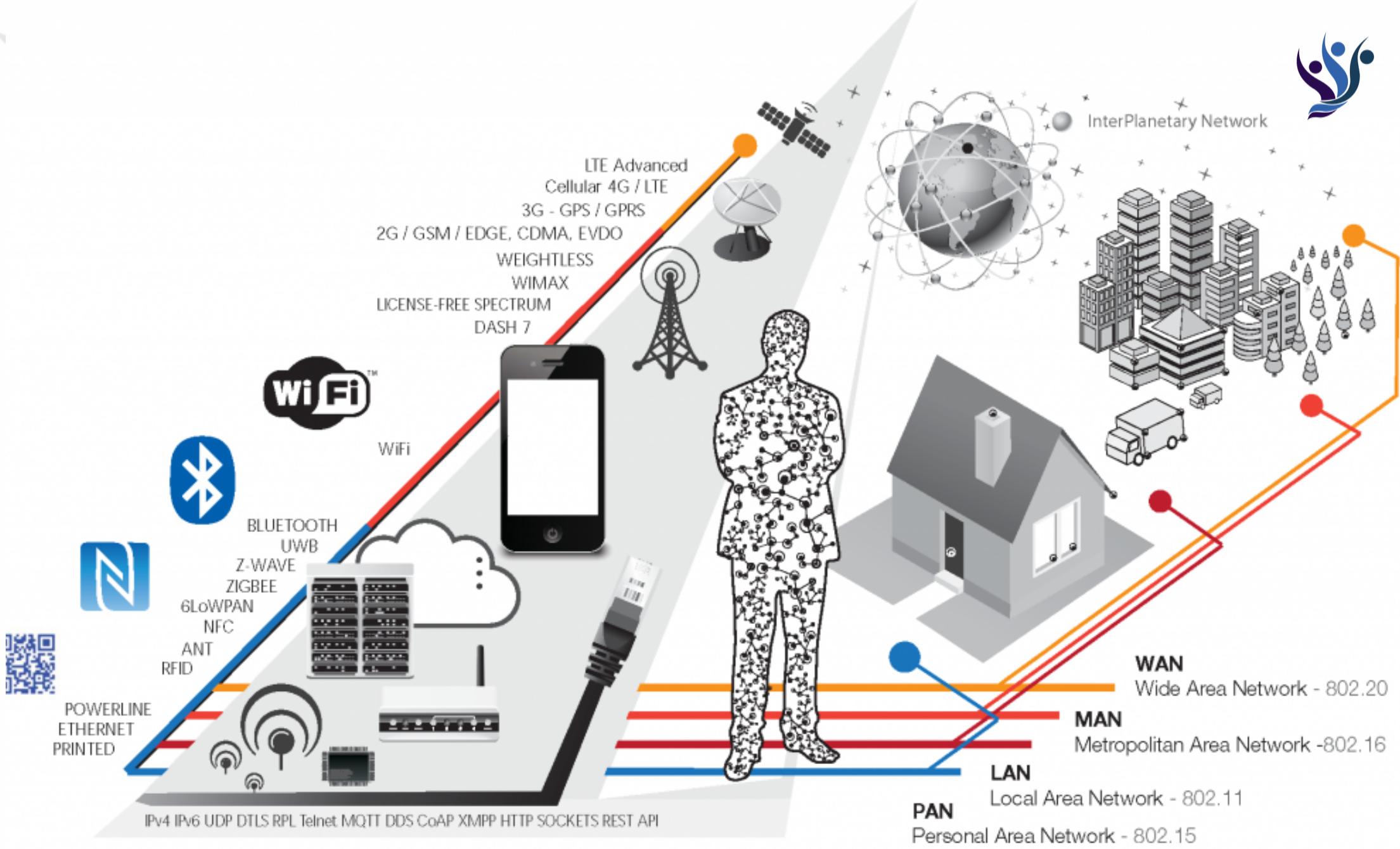
We are giving our world a **digital nervous system**. Location data using GPS sensors. Eyes and ears using cameras and microphones, along with sensory organs that can measure everything from temperature to pressure changes.



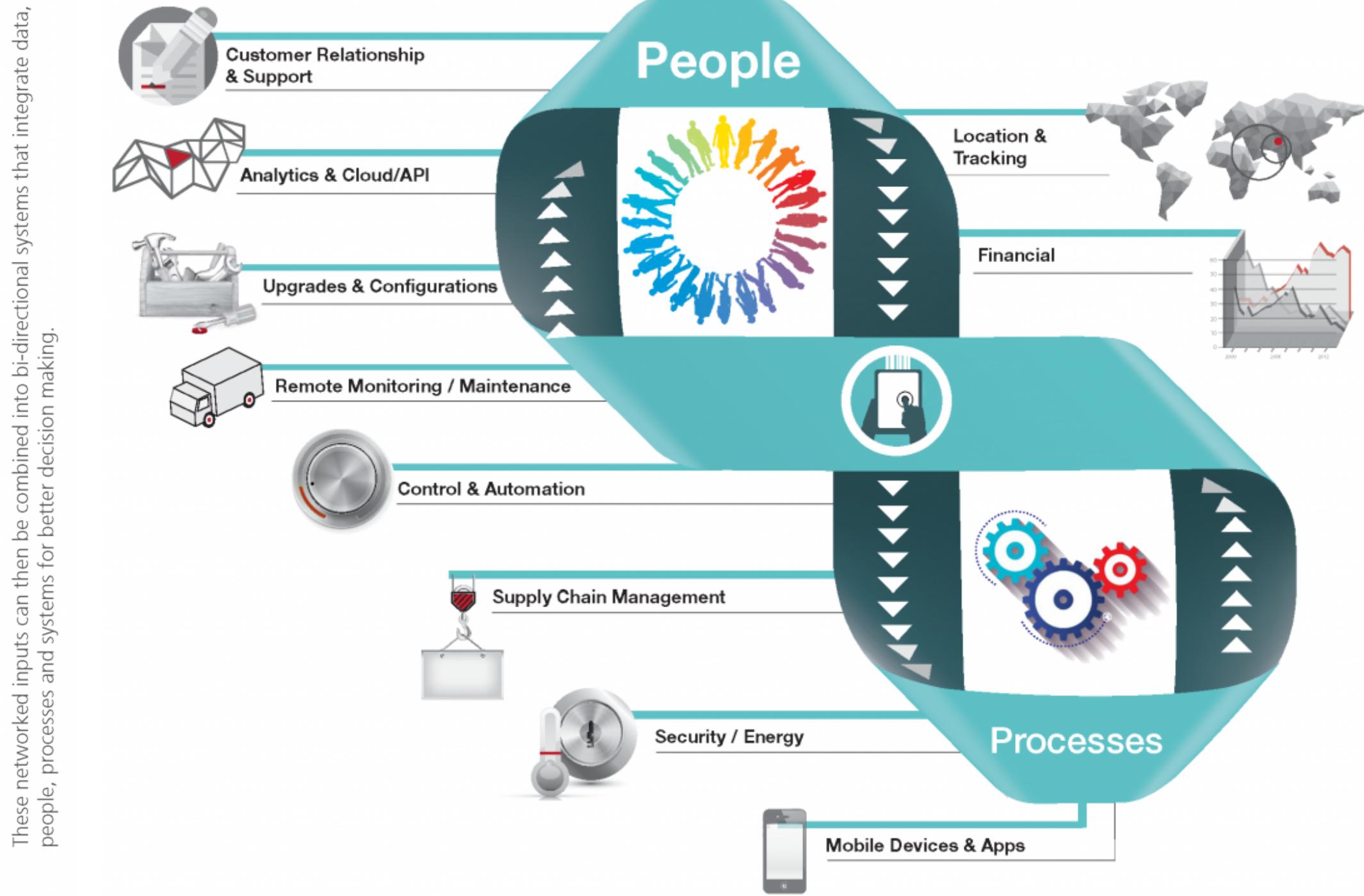
2 CONNECTIVITY

2

These inputs are digitized and placed onto networks.



3 PEOPLE & PROCESSES



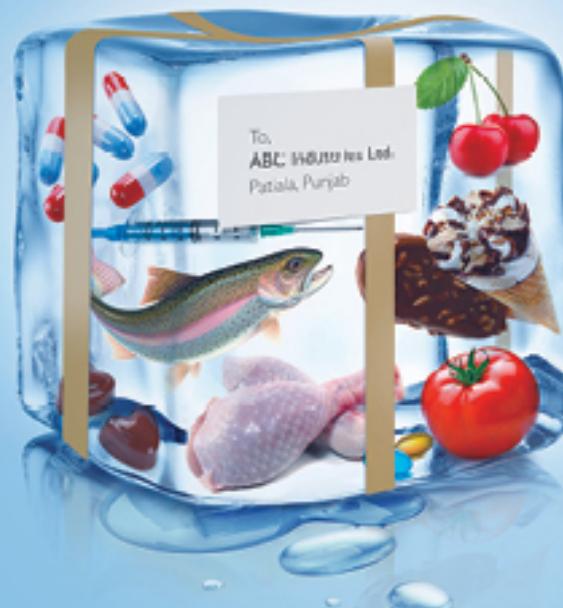
These networked inputs can then be combined into bi-directional systems that integrate data, people, processes and systems for better decision making.



Problem Statement



When technology leads the way,
freshness follows.

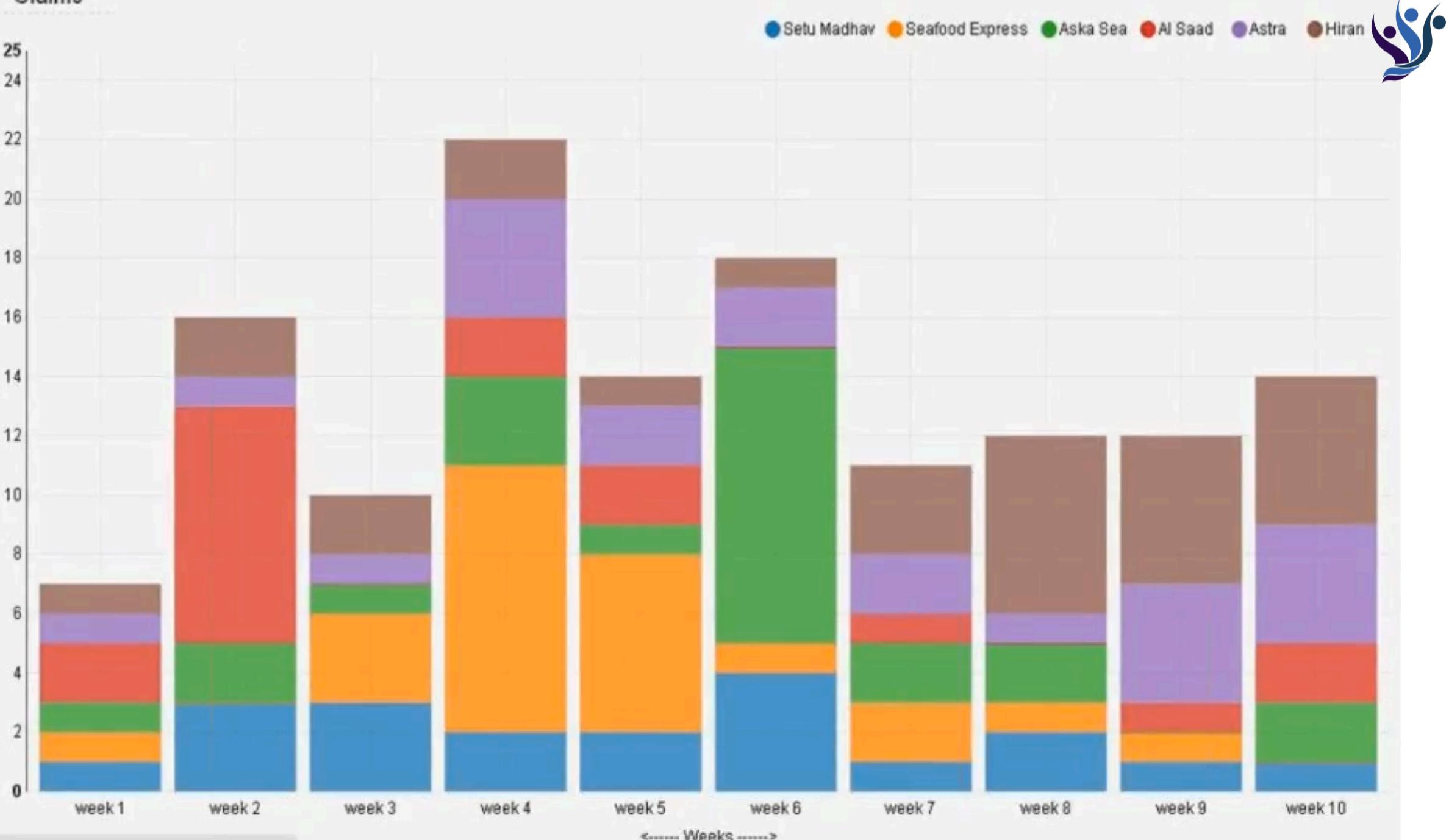


From Pickup to Destination:
Delivered Fresh, Delivered Safe.



Claims from Clients

Claims

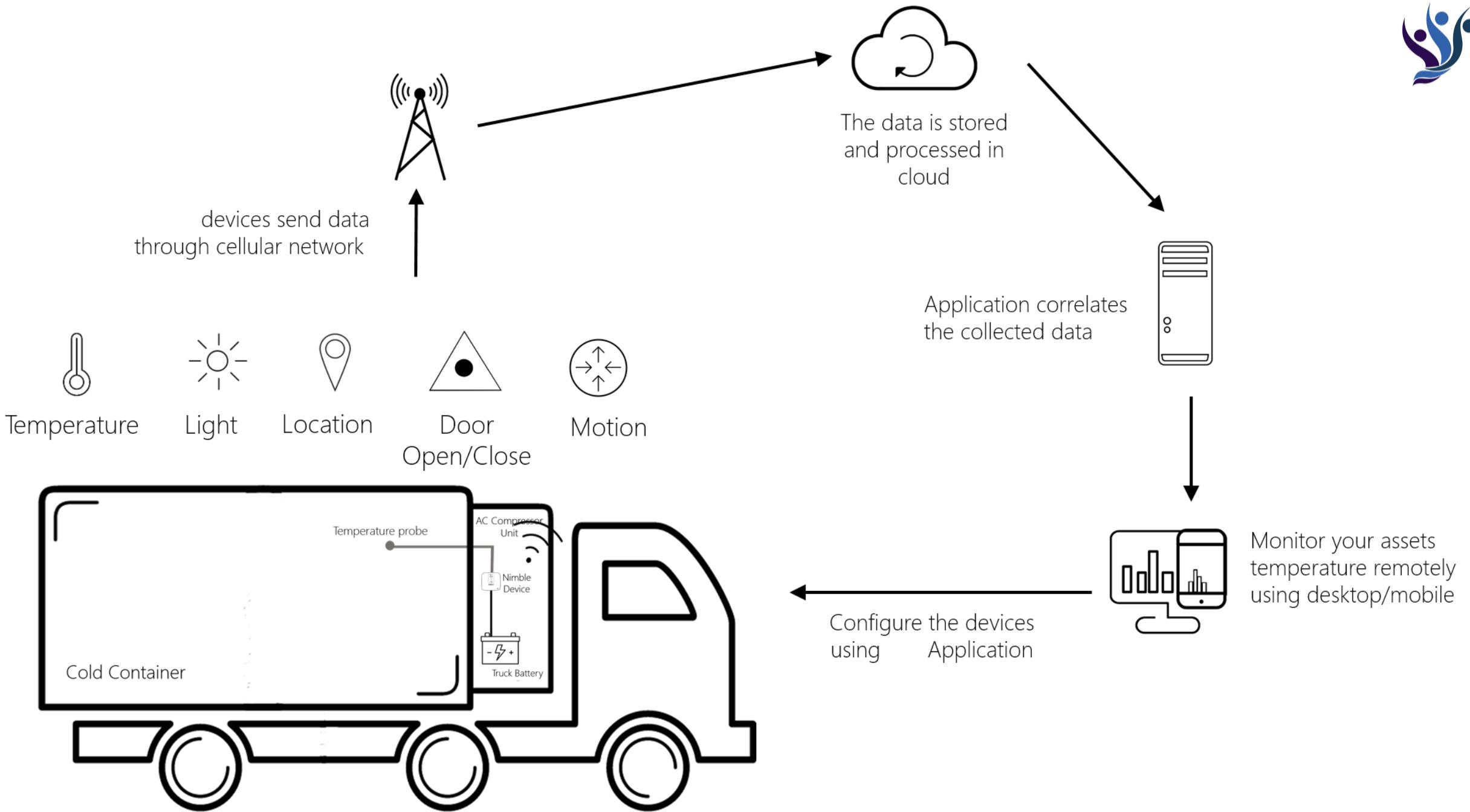




How To Filter False Claims



Solution

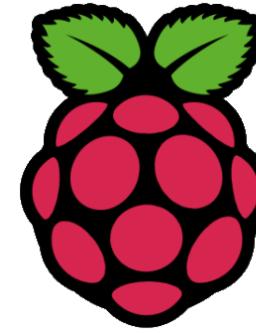
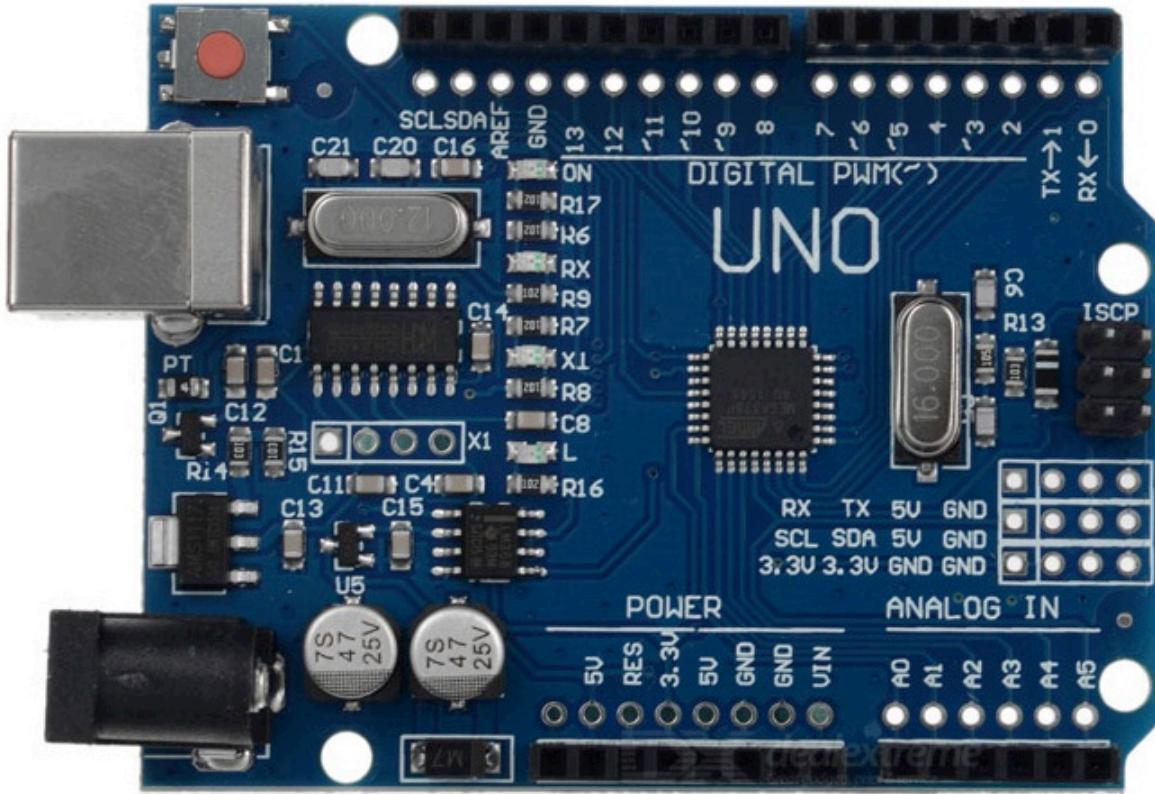




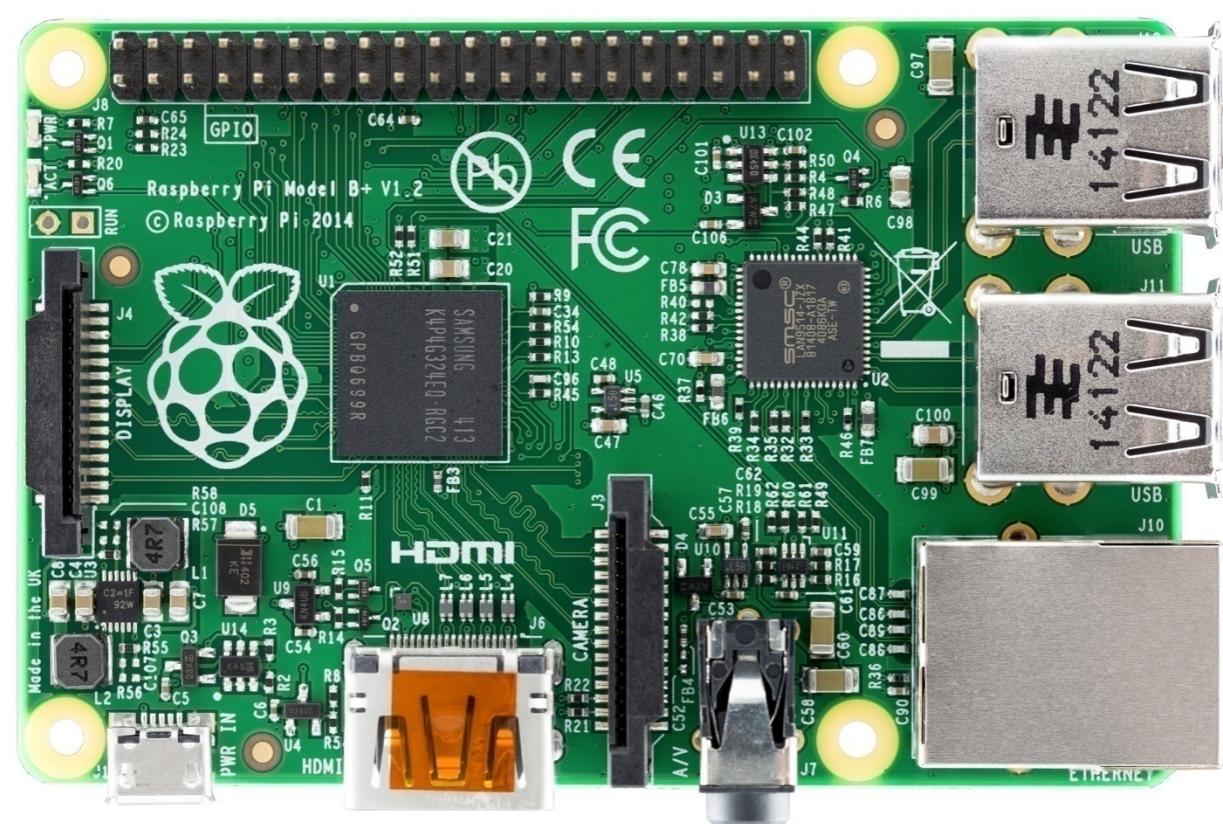
Execution



Development Boards



RaspberryPi





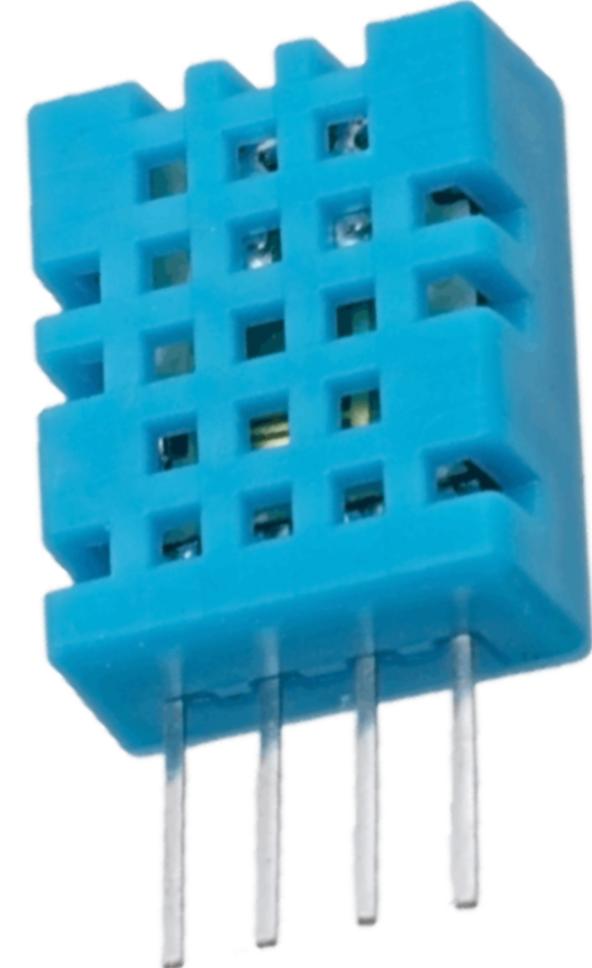
Hardware Requirements



LDR



LM35D



DHT11



Hardware Requirements



PIR



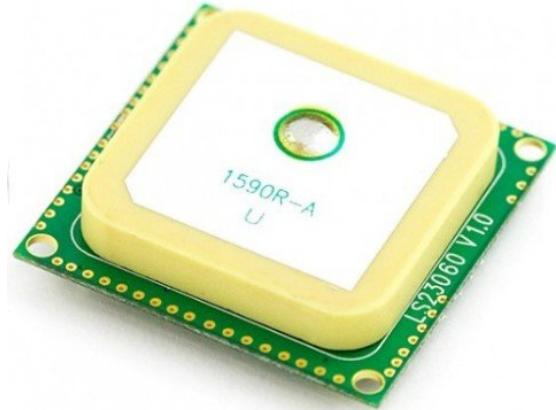
PC817



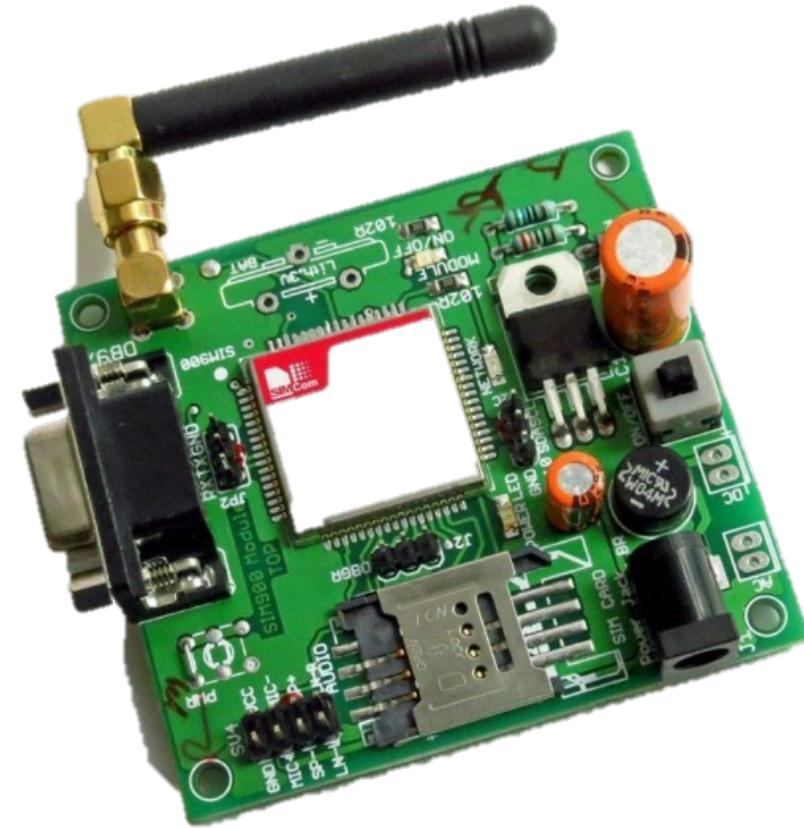
REED



Hardware Requirements

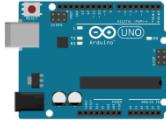
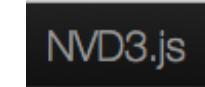


GPS



SIM800

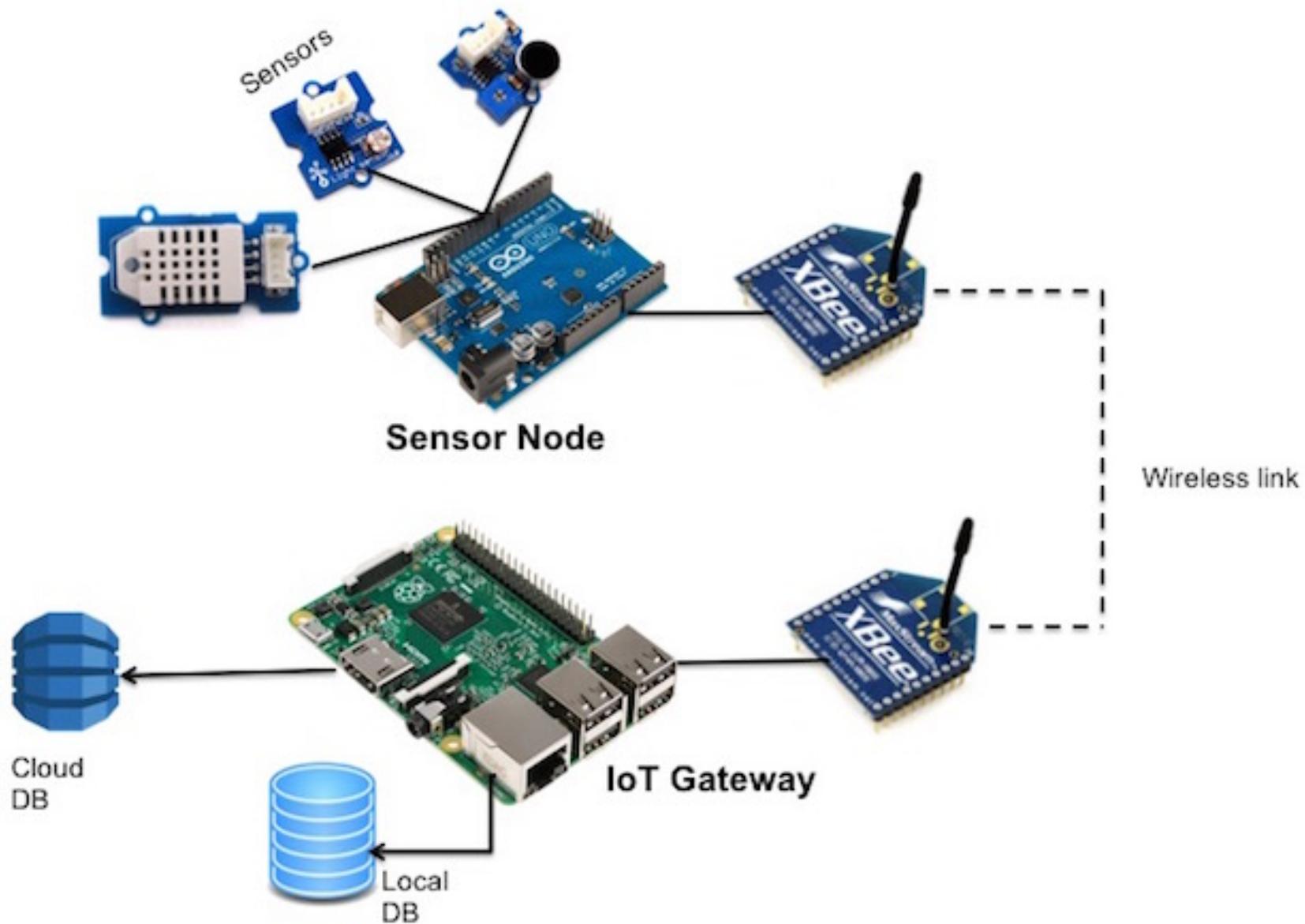
Practical Flow for Best Cold Chain Problem

Sensors	Nodes	Gateway	Web Services	Data Stores	Analytics	Visualization
 Read, Calibrate and Display data from sensors   	 Develop code for reading SMS through Arduino and trigger the Alarm.	 Send the data to cloud using GPRS	 Create a REST based Web Service to send truck sensor data on the cloud    Learn how to create cloud based programs	 Write the program to store the truck sensor data in NoSql Database.	 Analyze the data to solve business problem.	 Generate a neat report to visualize the data for management. 



Introduction to Sensor, Nodes & Gateway

Sensor Node and Gateway





IoT Clouds

- Device management and integration support
- Information security
- Data collection protocols
- Data analytics



IoT Software Platforms (Cloud)

Artik Cloud

ThingWorx

Autodesk Fusion Connect

Salesforce IoT Cloud

AWS IOT

Telit DeviceWise

GE Predix

Xively

Google Cloud IoT

Zebra Zatar Cloud

Microsoft Azure IoT Suite

Kaa (**Open Source**)

IBM Watson IoT

✓ ThingSpeak (**Open Source**)



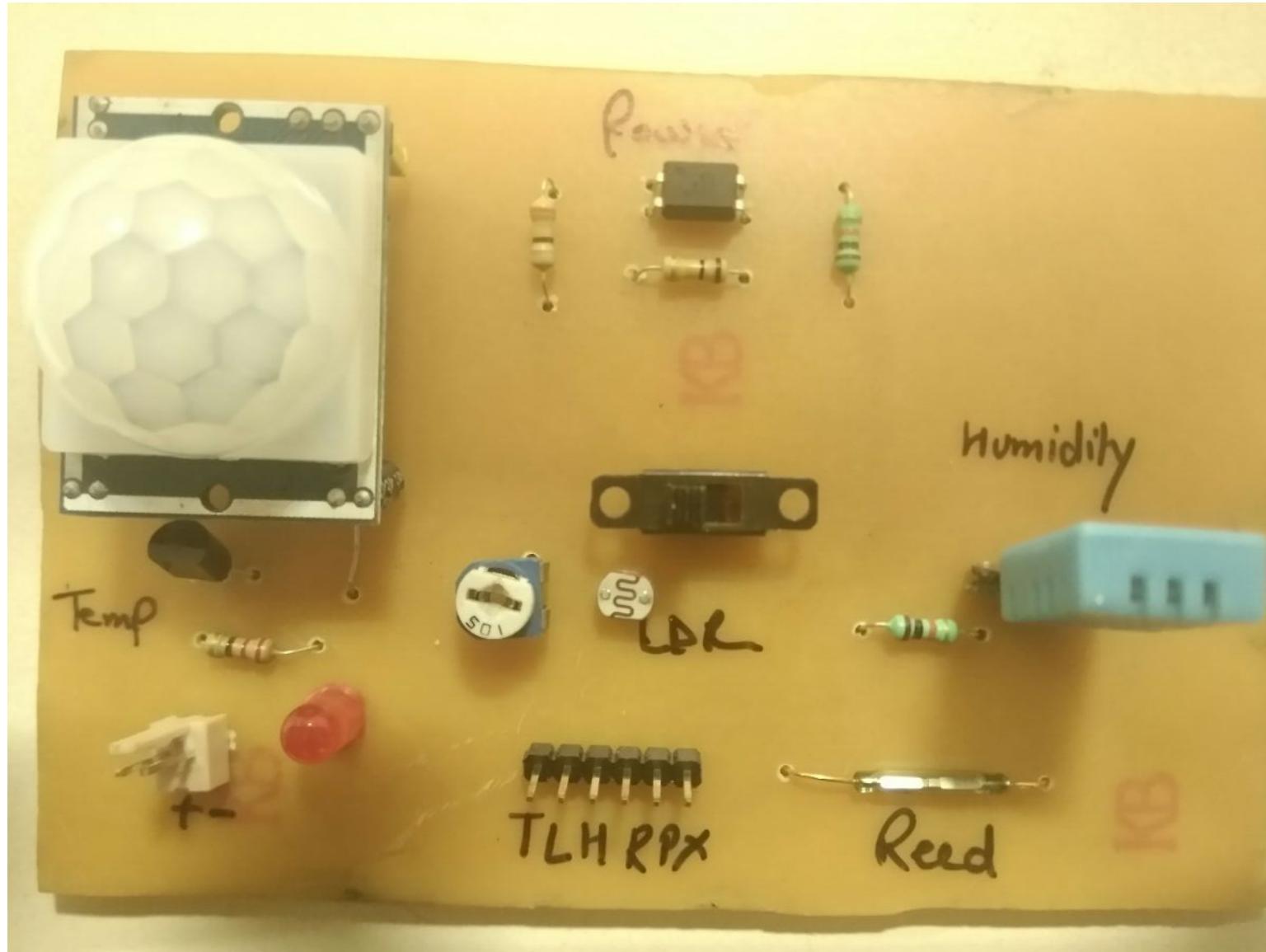
Hands On



Introduction to Sensor Board

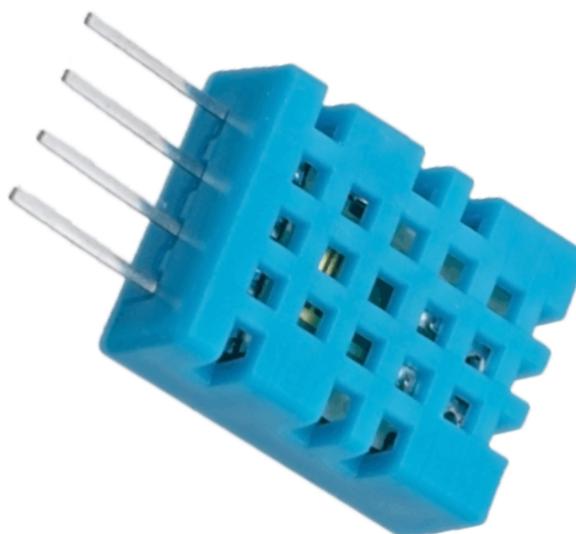
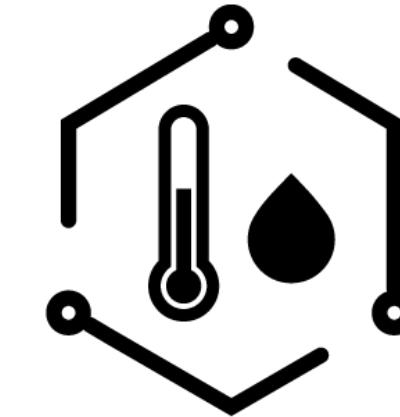
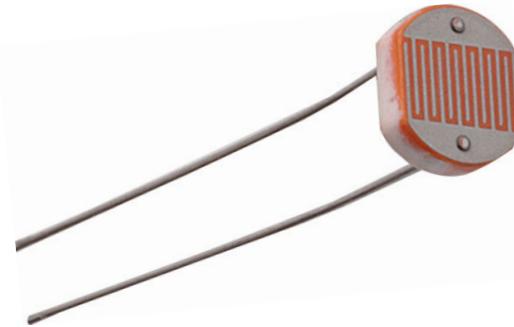
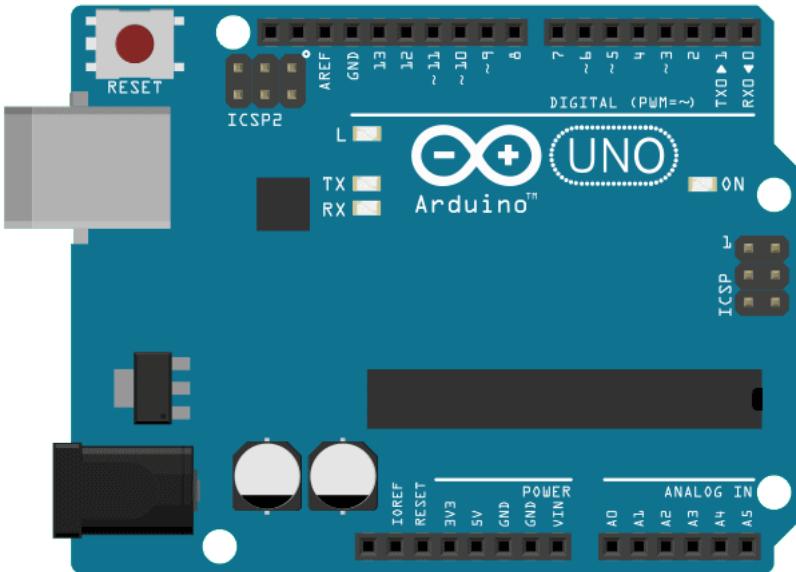


Sensor Board



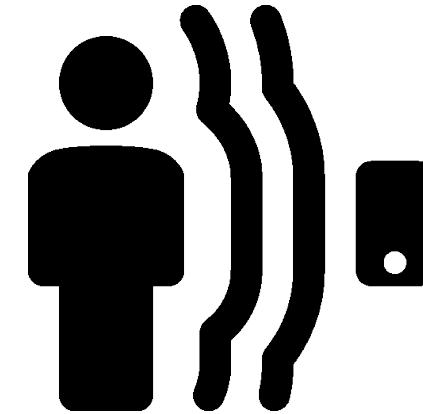
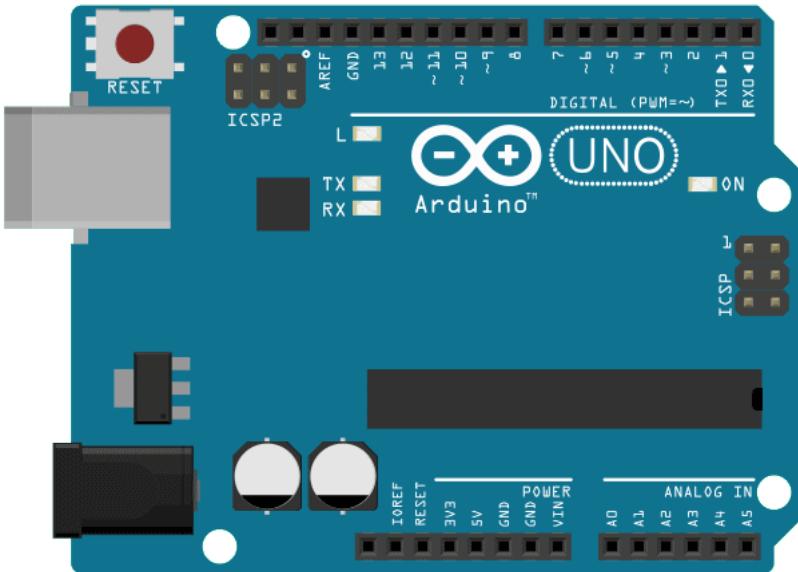


Analog Sensors





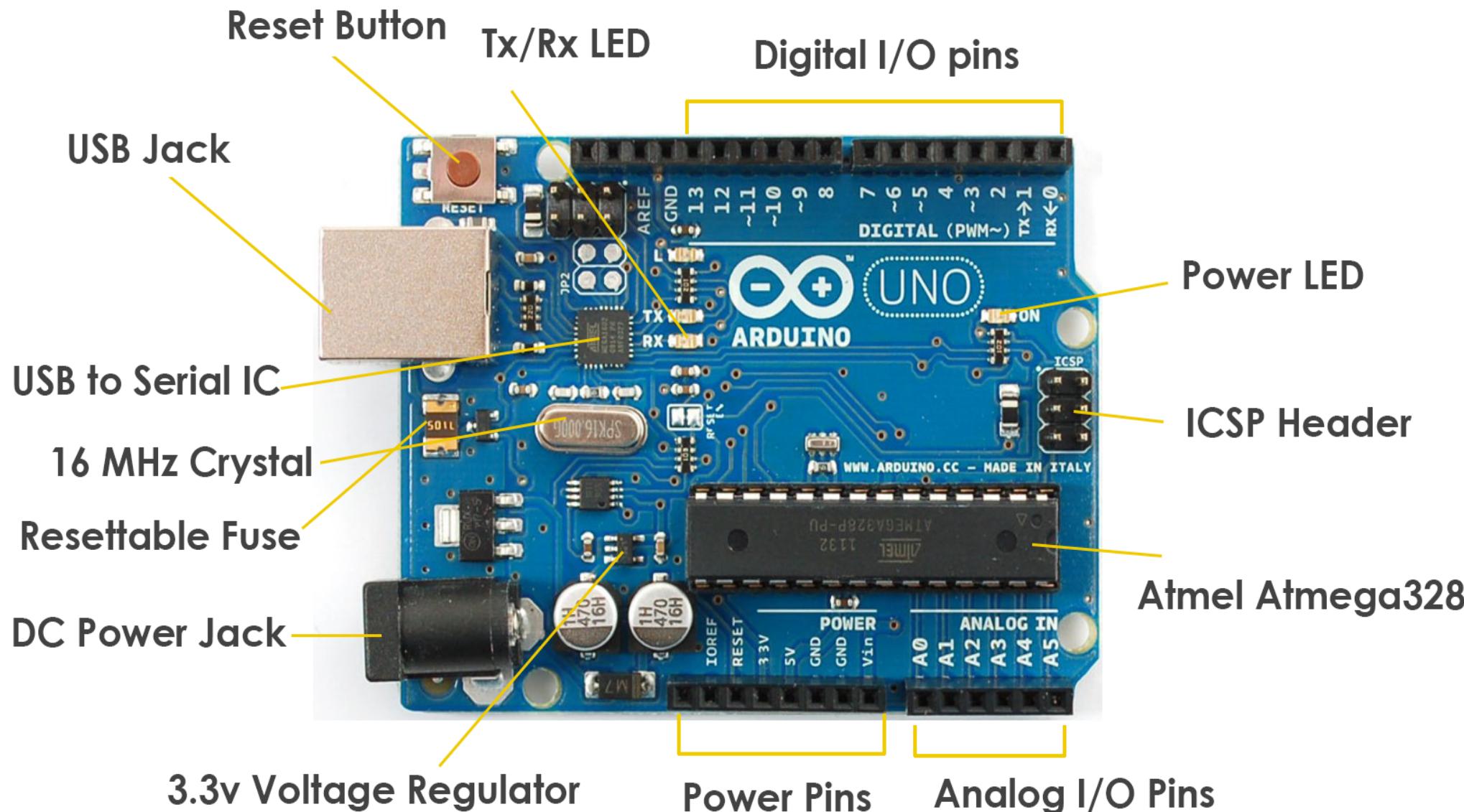
Digital Sensors





Introduction to Arduino





Programming Language Components

Structure

- Basic syntax
- Arithmetic operators
- Control structures
- Comparison Operators
- Boolean Operators

Variables

- Constants
- Data types
- Scope

Functions

- Digital I/O
- Analog I/O
- Math
- Serial communication
- Defining your own



Programming Language Components

Structure

- Basic syntax
- Arithmetic operators
- Control structures
- Comparison Operators
- Boolean Operators

Variables

- Constants
- Data types
- Scope

Functions

- Digital I/O
- Analog I/O
- Math
- Serial communication
- Defining your own

Structure: Basic Syntax

;
Each statement ends in a semicolon.

For example: `int a = 13;`

{ }
Curly braces always come in pairs; they are used to define the start and end of functions, loops, and conditional statements.

For example:

```
while (boolean expression)
{
    statement(s)
}
```

// Single line comment
/* */ Multi-line comment

#define Used to give a name to a constant value.

For example:

```
#define ledPin 3
```

Structure: Arithmetic Operators

Assignment operator stores the value to the right of the equal sign in the variable to the left of the equal sign: `sensorVal = analogRead(FSRPin);`

Addition, subtraction, multiplication, and division. For example:

+	result	= value1 + value2;
-	result	= value1 - value2;
*	result	= value1 * value2;
/	result	= value1 / value2;

where `value1` and `value2` are any variables or constants

Tips:

- Choose variable sizes that are large enough to hold the largest calculated result
- For math that requires fractions, use float variables (but there are drawbacks)
- Check for order of operations; use parentheses to enforce order

Structure: Control Structures

if Tests whether a certain condition has been reached. Used in conjunction with a comparison operator.

For example:

```
if (someVariable > 50)
{
    // do something here
}
```

if...else Allows you to do multiple tests.

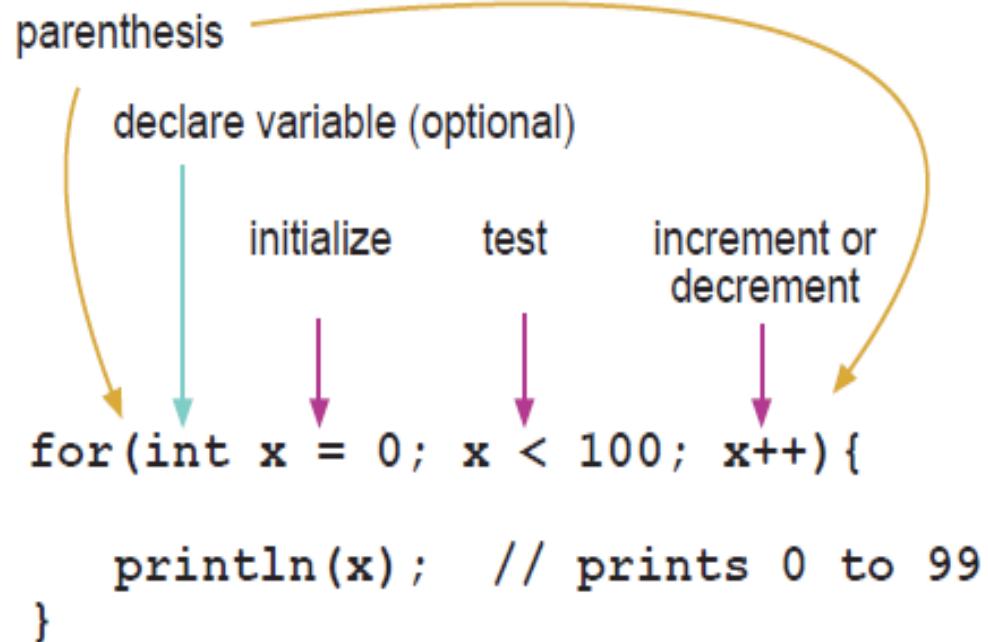
For example:

```
if (force > 1)
{
    // action A
}
else
{
    // action B
}
```

Structure: Control Structures

for Creates a loop for repetitive operations.

```
for (initialization; condition; increment) {  
    //statement(s);  
}
```



Structure: Control Structures

switch case Allows you to specify different code that should be executed in various conditions. For example:

```
switch (var)
{   case 1:
    //do something when var equals 1
    break;
    case 2:
    //do something when var equals 2
    break;
    default:
    // if nothing else matches, do the default
    // default is optional
}
```

Tips:

- Be careful not to accidentally use the assignment operator = instead of ==.
- Cannot use statements such as $0 < x < 1$; need to do each comparison separately

Structure: Comparison Operator

The result of a statement with a comparison operator is
either TRUE (1) or FALSE (2)

```
x == y (x is equal to y)
x != y (x is not equal to y)
x < y (x is less than y)
x > y (x is greater than y)
x <= y (x is less than or equal to y)
x >= y (x is greater than or equal to y)
```

Tips:

- Be careful not to accidentally use the assignment operator = instead of ==.
- Cannot use statements such as $0 < x < 1$; need to do each comparison separately

Structure: Boolean Operator

&&

Logical AND.True only if both operands are true,e.g.

```
if (digitalRead(2) == HIGH && digitalRead(3) == HIGH) {  
    // do this only if both inputs are high  
}
```

||

Logical OR.True if either operand is true,e.g.

```
if (x > 0 || y > 0) {  
    // do this if either x or y is greater than 0  
}
```

!

NOT.True if the operand is false, e.g.

```
if (!x) {  
    // do this if x is false (0)  
}
```

Programming Language Components

Structure

- Basic syntax
- Arithmetic operators
- Control structures
- Comparison Operators
- Boolean Operators

Variables

- Constants
- Data types
- Scope

Functions

- Digital I/O
- Analog I/O
- Math
- Serial communication
- Defining your own

Variables: Constants

HIGH When reading or writing to a digital pin, there are only two possible values a pin can take (or be set to): **HIGH** and **LOW**

true Logical levels (result of a comparison):
false false is defined as 0
 true is defined as 1 (but more broadly, anything but 0)

In addition, integer and floating-point constants can be used:

Floating point 10.0
2.34E5
67e-12

Variables: Data Types

void Used in function declarations to indicate that the function returns no information. For example:

```
void setup()
{
    // ...
}
```

```
void loop()
{
    // ...
}
```

boolean A boolean holds one of two values, true or false. For example:

```
boolean running = false;
if (running) {
// do something
}
```

Variables: Data Types

char A data type that stores a character value. For example:

```
char myChar = 'A';  
char myChar = 65; // both are equivalent
```

Coding is in this ASCII chart: <http://arduino.cc/en/Reference/ASCIIchart>

float Datatype for floating-point numbers, a number that has a decimal point.

double Floating-point numbers are often used to approximate analog and continuous values because they have greater resolution than integers. Floats have 6-7 decimal digits of precision. On the Hapkit board, double is the same as float.

Variables: Scope

Global vs. Local:

- A **global** variable is one that can be seen by every function in a program. Define it outside a function.
- A **local** variable is only visible to the function in which it is declared. Define it inside a function.
- For complex programs, local variables can help prevent programming errors. However, global variables are an easy way to share information across functions.

The **static** keyword is used to create variables that are visible to only one function. However unlike local variables that get created and destroyed every time a function is called, static variables persist beyond the function call, preserving their data between function calls.

For example: `static int a;`

Programming Language Components

Structure

- Basic syntax
- Arithmetic operators
- Control structures
- Comparison Operators
- Boolean Operators

Variables

- Constants
- Data types
- Scope

Functions

- Digital I/O
- Analog I/O
- Math
- Serial communication
- Defining your own

Functions: Digital I/O

For a description of the roles of different pins on the Arduino/Hapkit, see
<http://arduino.cc/en/Tutorial/DigitalPins>

`pinMode(pin, mode)` Configures the specified pin to behave either as an input or an output. `pin` is the pin number.

`digitalWrite(pin, value)` Write a HIGH or a LOW value to a digital pin.

`digitalRead(pin)` Reads the value from a specified digital pin. The result will be either HIGH or LOW.

Functions: Analog I/O

`analogReference(type)` The default reference voltage is 5V. This can be changed to a different `type` and different resolution using this function.

`analogRead(pin)` Reads the value from the specified analog pin and returns a value between 0 and 1023 to represent a voltage between 0 and 5 volts (for default). It takes about 0.0001 seconds to read an analog pin.

`analogWrite(pin,value)` Writes an analog value (PWM wave) to a pin. `value` is the duty cycle: between 0 (always off) and 255 (always on). Works on pins 3,5,6,9,10, and 11.

Functions: Math

<code>min(x, y)</code>	Calculates the minimum of two numbers
<code>max(x, y)</code>	Calculates the maximum of two numbers
<code>abs(x)</code>	Computes the absolute value of a number
<code>pow(base, exponent)</code>	Calculates the value of a number raised to a power
<code>sqrt(x)</code>	Calculates the square root of a number
<code>map(value, fromLow, fromHigh, toLow, toHigh)</code>	Re-maps a number from one range to another. That is, a value of <code>fromLow</code> would get mapped to <code>toLow</code> , a value of <code>fromHigh</code> to <code>toHigh</code> , values in between to values in between.
Trigonometric functions such as <code>sin</code> , <code>cos</code> , and <code>tan</code> are also available.	

Functions: Serial Communication

Typically used for communication between an Arduino board and a computer via the USB port. Use the **serial monitor** to communicate with the board.

`Serial.begin(9600);` Used to begin serial communications, typically at a 9600 baud rate (bits per second)

`Serial.print(val, format);` Prints data to the serial port as human-readable ASCII text.
Examples:

`Serial.print(78)` gives "78"

`Serial.print(1.23456)` gives "1.23"

`Serial.println(1.23456, 4)` gives "1.2346"

`Serial.print("Hello world.")` gives "Hello world."

`Serial.println(val);` Prints val followed by carriage return

Functions: Defining your own

- Many other functions have been created by Arduino users; some are posted at
<http://playground.arduino.cc/Main/GeneralCodeLibrary>
 - You can also define your own function.
 - This could be used to make your code more organized and efficient.
- ```
int find_text(String needle, String haystack)
{
```

This is a function that searches for a given string within another string. If the search string is found its position is returned, otherwise -1 is returned.

```
}
```



# Installation of Arduino



# Hello World for Arduino



# Sensor Board Wiring

(+) to Arduino +5V

(-) to Arduino GND

## Analog Line

T – Temperature Sensor ( A2 )

H – Humidity Sensor ( A5 )

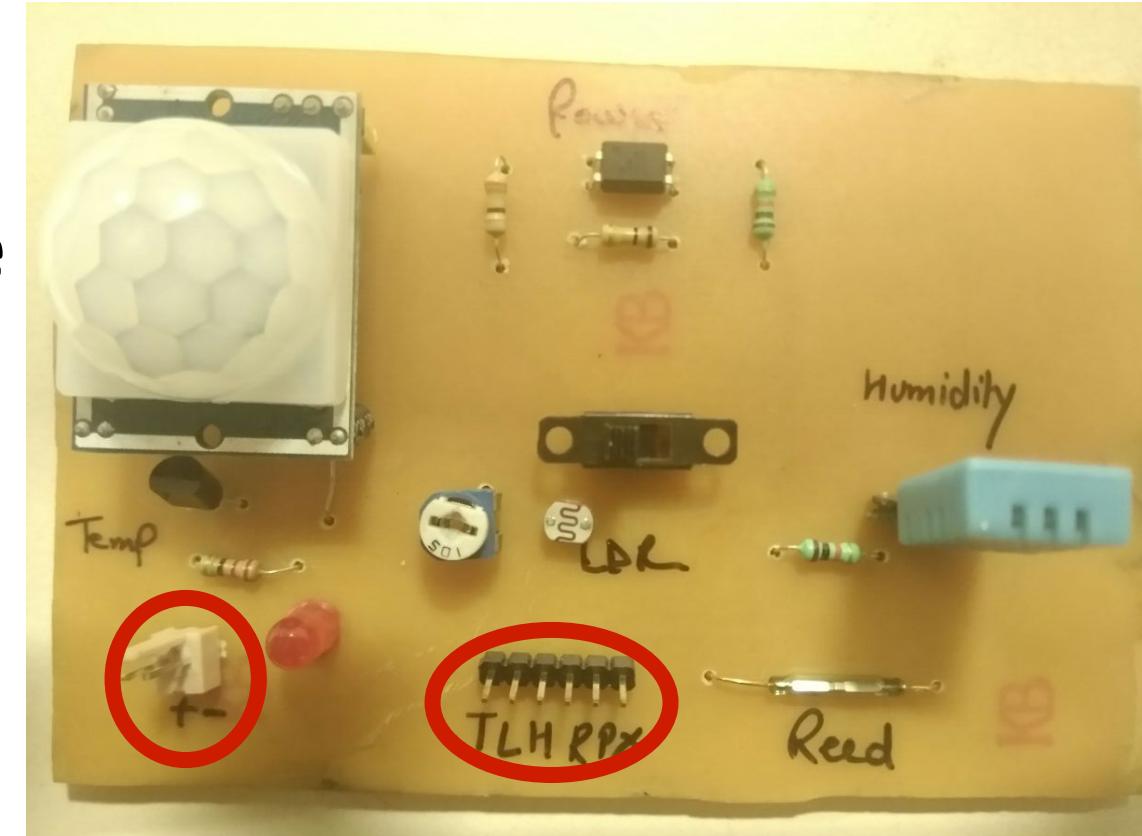
L – Light Sensor (LDR) ( A4 )

## Digital Line

P – Power Switch based on Optocoupler ( 4 )

R – Magnetically operated Reed Switch ( 5 )

X – PIR (Passive Infrared Sensor) ( 3 )





# Analog vs Digital



23:59:59





# PIR Sensor

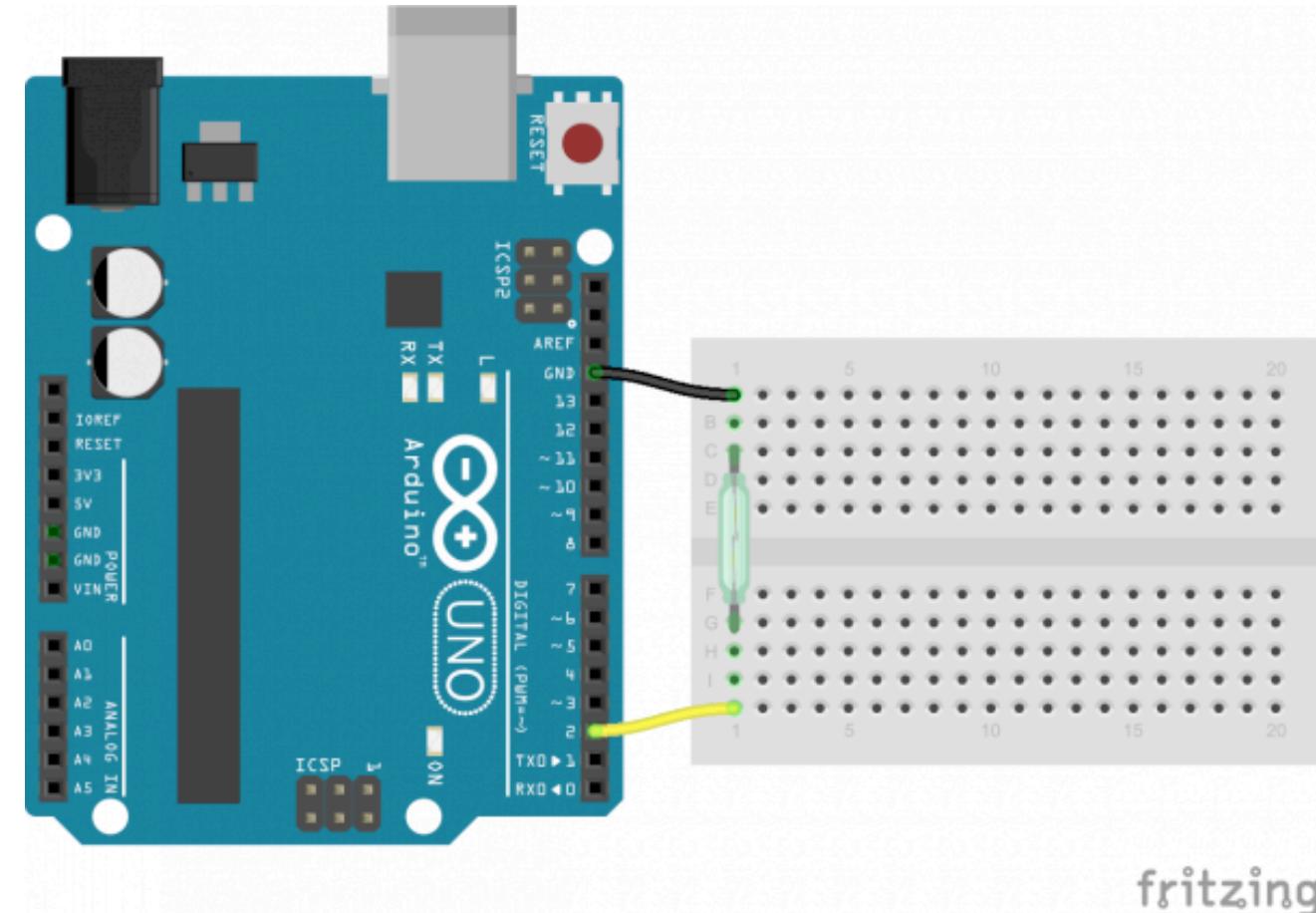
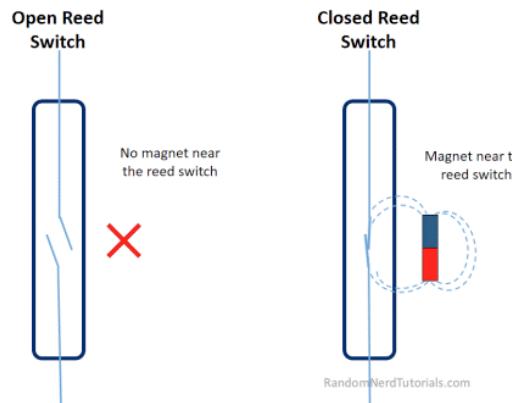


## Pin Wiring

| Pin Name | Wiring to Arduino Uno |
|----------|-----------------------|
| OUT      | Digital Pin 2         |
| GND      | GND                   |
| VCC      | 5V                    |



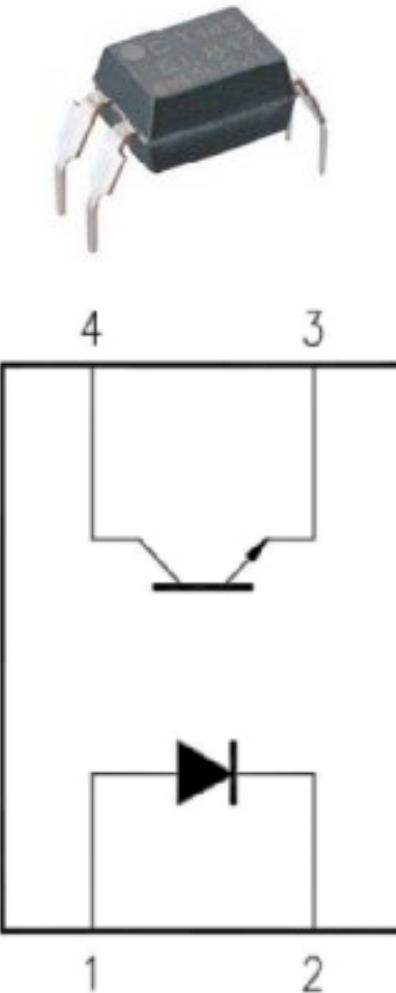
# Reed Sensor



fritzing



# PC817 Power Sensor



| Pin Name | Wiring to Arduino Uno |
|----------|-----------------------|
| D0       | Digital Pins          |
| GND      | GND                   |
| VCC      | 5V                    |

1. Anode  
2. Cathode

3. Emitter  
4. Collector



# Analog vs Digital



23:59:59





# What is ADC ?

The ADC on the Arduino is a 10-bit ADC meaning it has the ability to detect 1,024 ( $2^{10}$ ) discrete analog levels.

Some micro-controllers have 8-bit ADCs ( $2^8 = 256$  discrete levels).

Some have 16-bit ADCs ( $2^{16} = 65,536$  discrete levels).

The ADC reports a ratiometric value. This means that the ADC assumes 5V is 1023 and anything less than 5V will be a ratio between 5V and 1023.

$$\frac{\text{Resolution of the ADC}}{\text{System Voltage}} = \frac{\text{ADC Reading}}{\text{Analog Voltage Measured}}$$

$$\frac{1023}{5} = \frac{\text{ADC Reading}}{\text{Analog Voltage Measured}}$$



# What is ADC ?

If the analog voltage is 2.12V what will the ADC report as a value ?

$$\frac{1023}{5.00V} = \frac{x}{2.12V} \quad \frac{1023}{5.00V} * 2.12V = x \quad x = 434$$

If your system is 3.3V and your ADC is reporting 512, what is the voltage measured ?



# What is ADC ?

**What happens if you connect an analog sensor to a regular digital pin ?**

```
int x = analogRead(8);

//Try to read the analog value on digital pin 8 - compiles but does not give right value !
```

**What happens if I connect digital sensor to an analog pin ?**

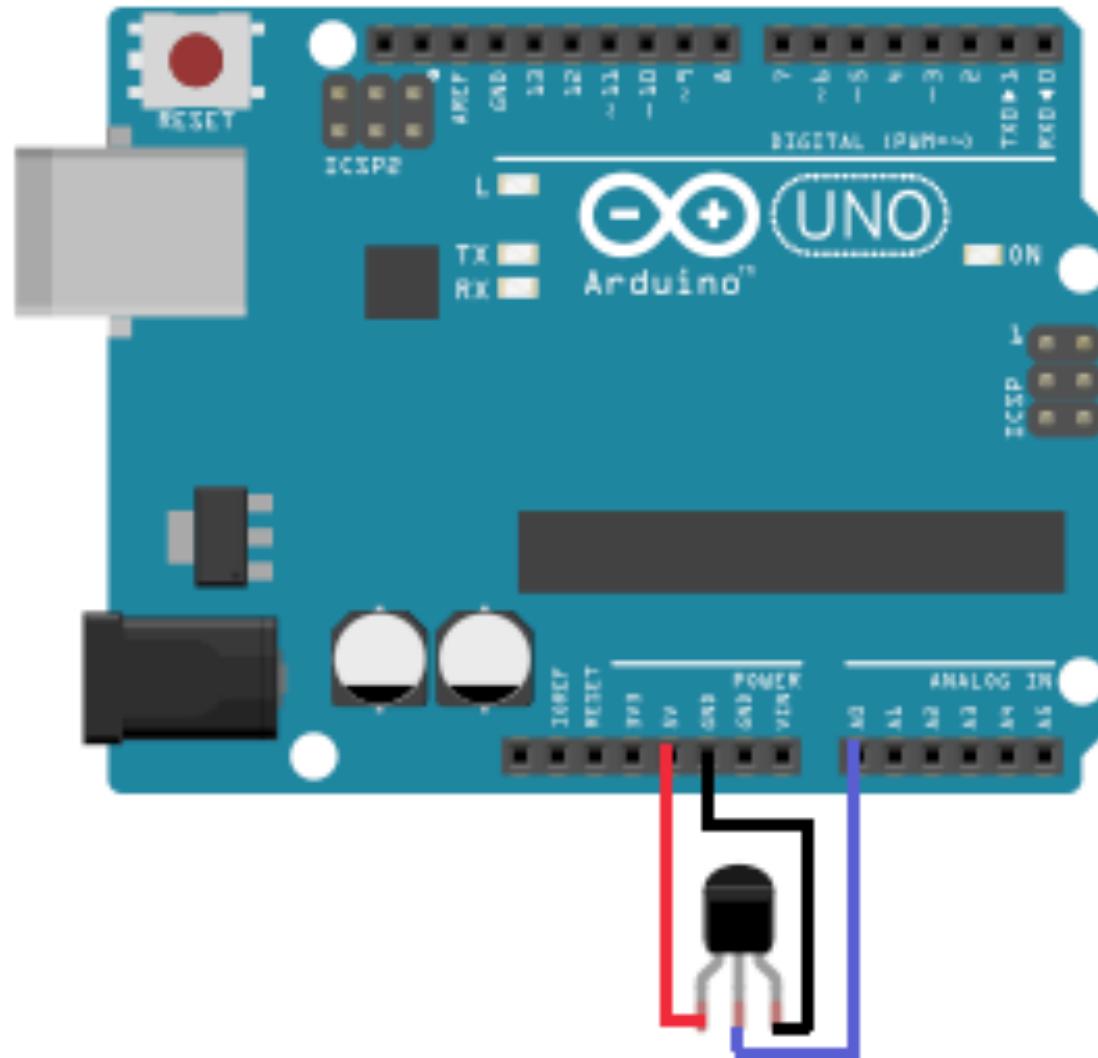
If you do an analog-to-digital conversion on a button, you will most likely see ADC values very close to 1023 (or 5V which is binary 1)

or very close to 0 (or 0V which is binary 0).

```
pinMode(A3, INPUT); // Setup the pin mode
int x = analogRead(A3); //Reads the analog value on pin A3 into x
Serial.print("Analog value: ");
Serial.println(x); // Prints the value on the Serial Monitor
```

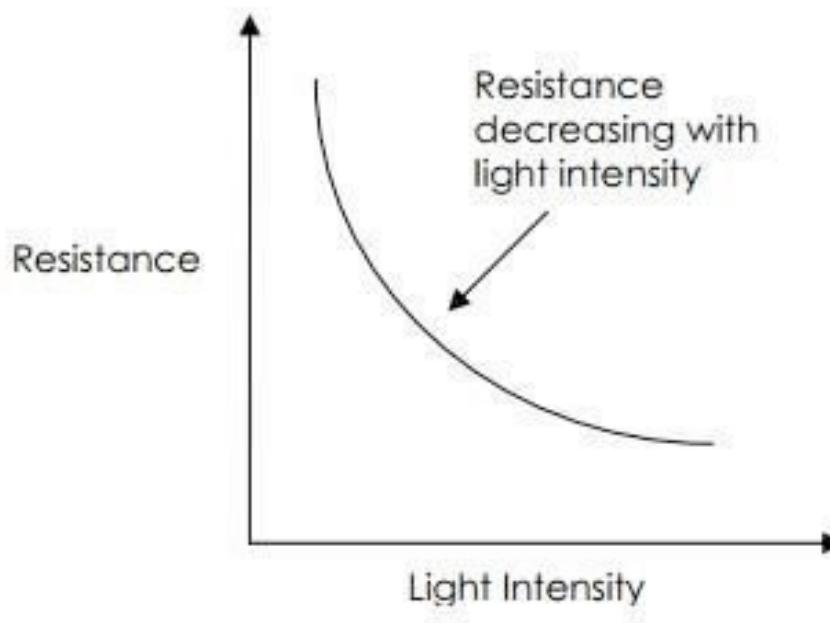


# LM35 Temperature Sensor





# LDR Light Sensor



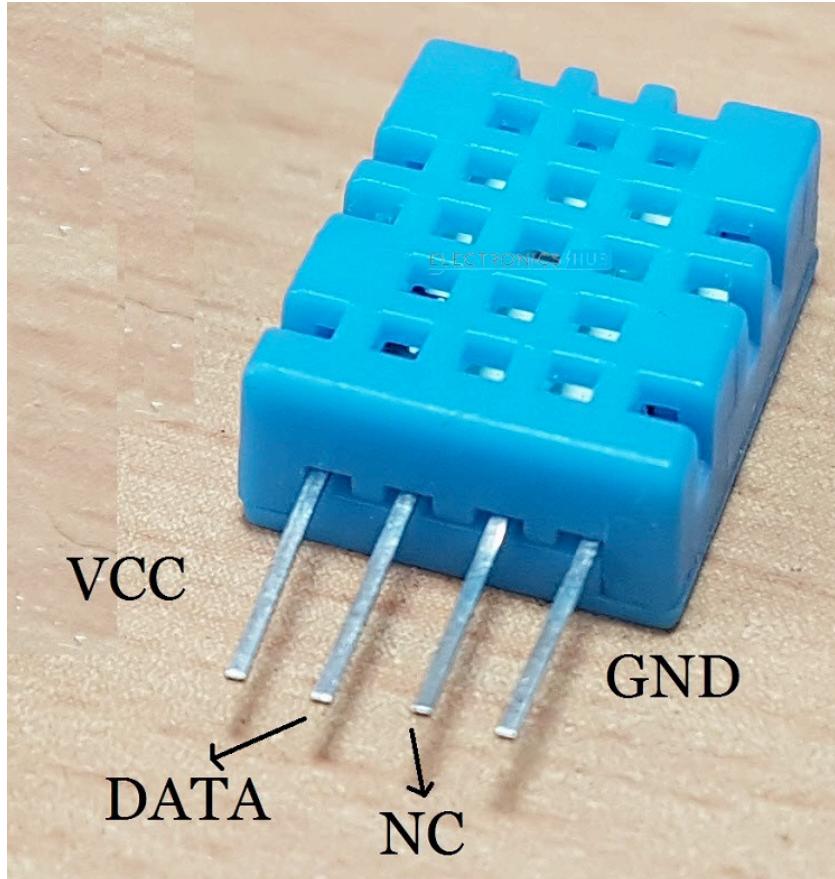
## Pin Wiring

Wiring your sensor to the Arduino is pretty simple:

| Pin              | Wiring to Arduino Uno |
|------------------|-----------------------|
| <b>Right Pin</b> | Analog Pin            |
| <b>Left Pin</b>  | 5V                    |



# DHT11 Humidity Sensor



## Pin Wiring

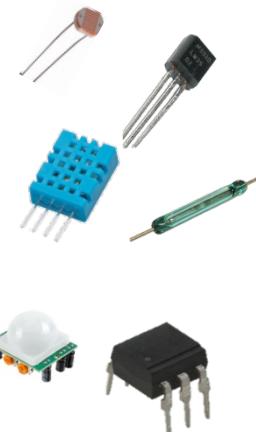
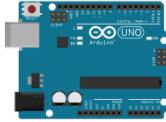
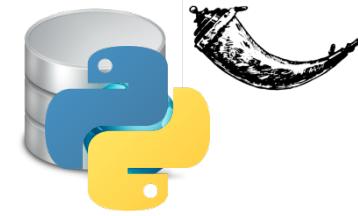
Wiring your sensor to the Arduino is pretty simple:

| Pin  | Wiring to Arduino Uno |
|------|-----------------------|
| Data | Analog Pin(A0-A5)     |
| NC   | Not Used              |
| GND  | GND                   |
| VCC  | 5V                    |



# **Installation of Third Party Libraries (DHT11.zip)**

# Practical Flow for Best Cold Chain Problem

| Sensors                                                                                                                                                                                                                | Nodes                                                                                                                                                    | Gateway                                                                                                                                                                                                                                           | Web Services                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Data Stores                                                                                                                                                | Analytics                                                                                                                          | Visualization                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <br>Read, Calibrate and Display data from sensors<br> | <br>Develop code for reading SMS through Arduino and trigger the Alarm. | <br><br>Send the data to cloud using GPRS<br>Send the data to cloud using WiFi | <br><br><br><br>Create a REST based Web Service to send truck sensor data on the cloud<br>Learn how to create cloud based programs | <br>Write the program to store the truck sensor data in NoSql Database. | <br>Analyze the data to solve business problem. | <br><br>Generate a neat report to visualize the data for management.<br>View and Control from Mobile App |



# Experiments

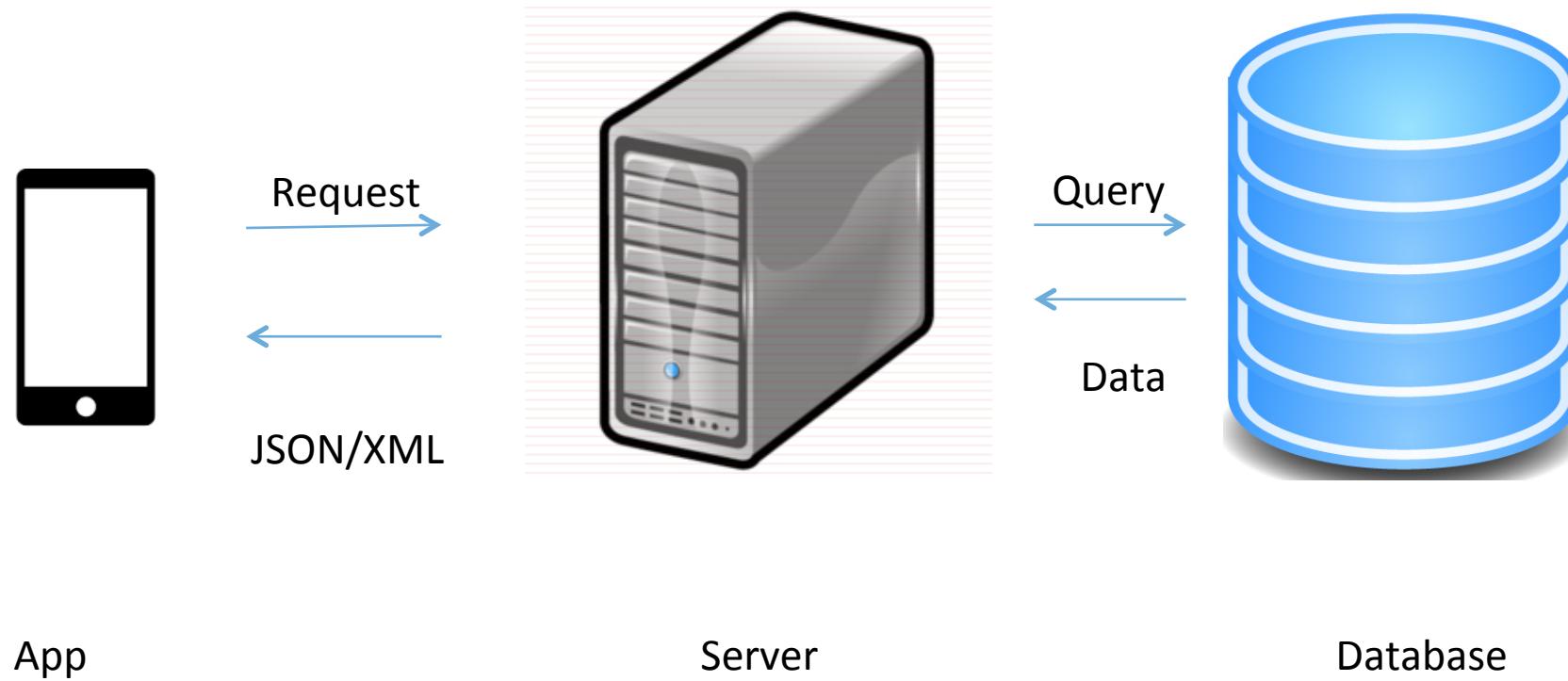
1. digital\_sensor\_check.ino
2. all\_digital\_sensor\_check.ino
3. analog\_sensor\_check.ino
4. all\_analog\_sensor\_check.ino
5. all\_sensor\_check.ino



# Data Exchange Mechanisms



# Data Exchange





# JSON

- JavaScript Object Notation
- Used to send data between server and mobile app
- Simple data interchange format
- Human readable
- Programming language independent



# JSON Data Structure

- Object
  - { "key1": "value1", "key2": "value2" }
- Array
  - [“first”, “second”, “third”, “fourth” ]
- Number
  - 42, 3.21
- String
  - “This is string”
- Boolean
  - True, False
- Null
  - Null



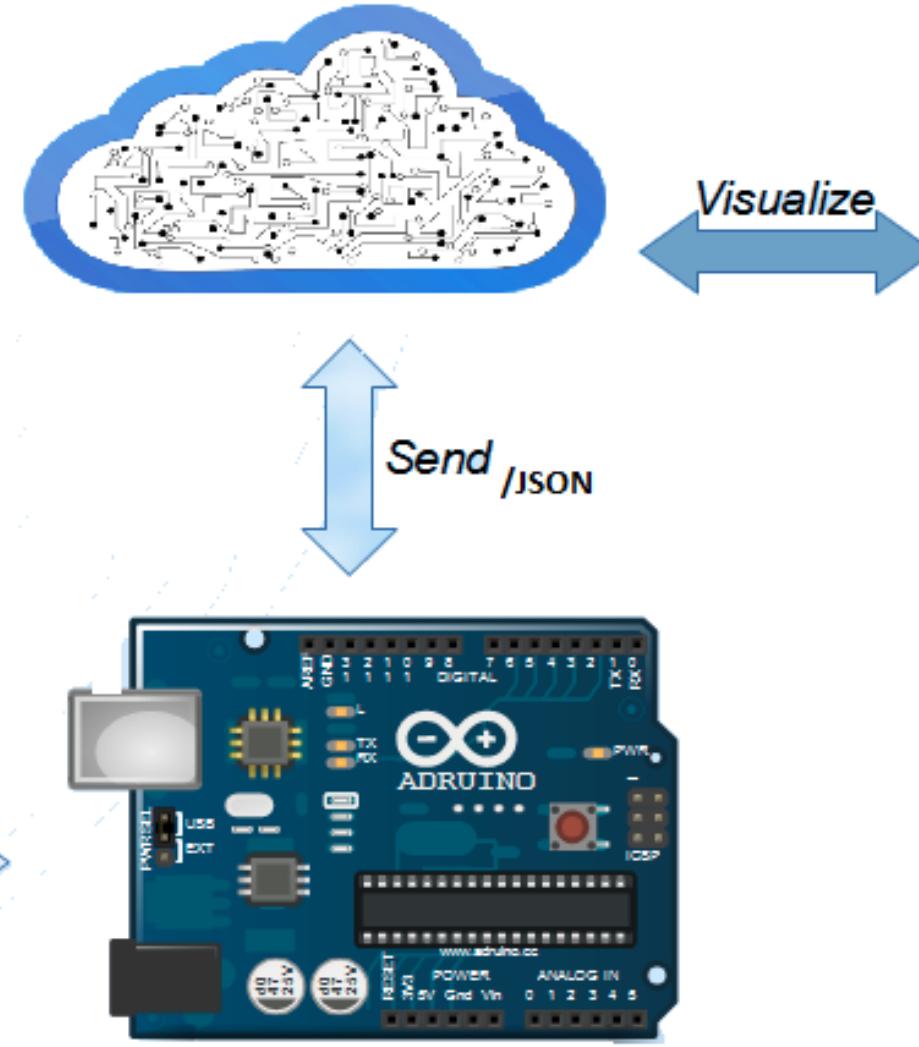
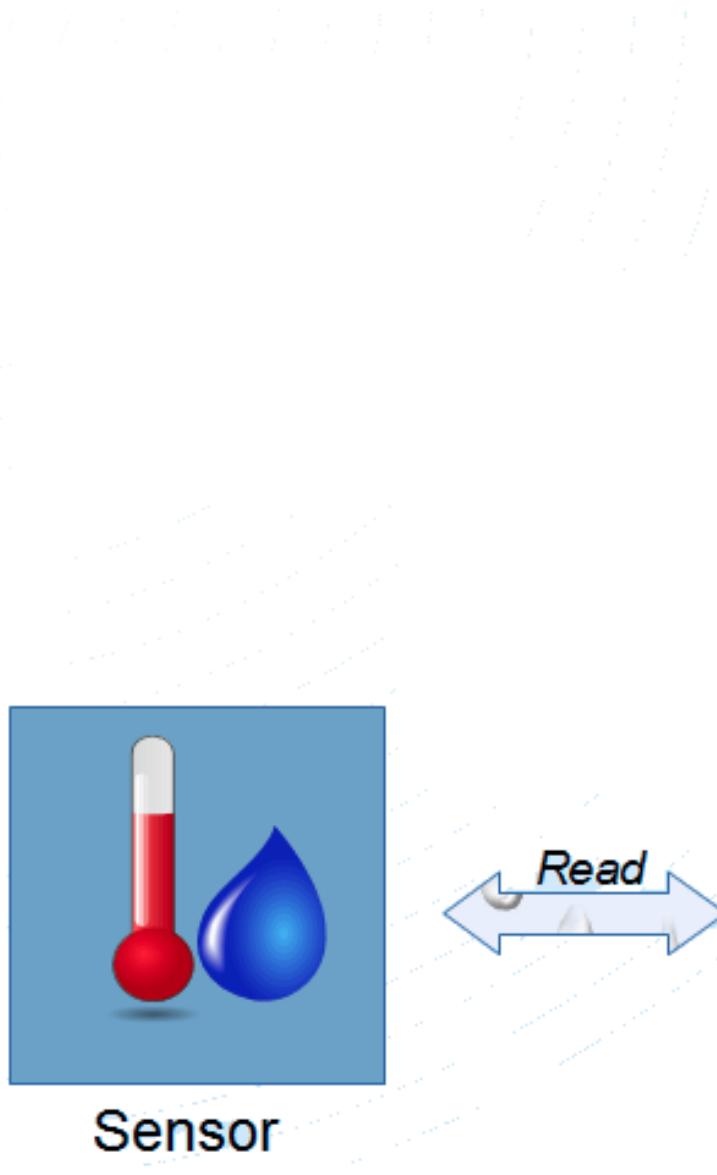
# JSON Example

```
{
 "door_operation": "1",
 "power_status": "1",
 "temperature": "30.5",
 "humidity": "37",
 "trip_id": "120",
 "device_type": "ColdChain",
 "device_no": "9928492120",
 "type": "transaction"
}
```

```
{
 "field1": "1",
 "field2": "1",
 "field3": "30.5",
 "field4": "37",
 "field5": "120",
 "field6": "ColdChain",
 "field7": "9928492120",
 "field8": "transaction"
}
```



# Summary



Arduino



# How to use REST API ?



# REST APIs – Weather Details

- **OpenWeatherMap** is an online service that provides weather data, including current weather data, forecasts, and historical data to the developers of web services and mobile applications
- <http://openweathermap.org/api>
- To access the API you need to sign up for an API key



# Jaipur Weather Data – By Name

- Calling API with City Name

- [http://api.openweathermap.org/data/2.5/weather?  
q=Jaipur&appid=e9185b28e9969fb7a300801eb026de9c](http://api.openweathermap.org/data/2.5/weather?q=Jaipur&appid=e9185b28e9969fb7a300801eb026de9c)

- Use your API Key to get the data in your browser

- Response

- ```
{"coord":{"lon":75.82,"lat":26.92},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":306.323,"pressure":975.65,"humidity":30,"temp_min":306.323,"temp_max":306.323,"sea_level":1021.8,"grnd_level":975.65},"wind":{"speed":3.49,"deg":315.504},"clouds":{"all":0},"dt":1476778298,"sys":{"message":0.0114,"country":"IN","sunrise":1476752290,"sunset":1476793496},"id":1269515,"name":"Jaipur","cod":200}
```



Cloud Signup



Step 1: Open <https://thingspeak.com/>

The screenshot shows the ThingSpeak website homepage. At the top, there's a navigation bar with links for File, Edit, View, History, Bookmarks, Tools, Window, Help, and a search bar. Below the navigation is a banner featuring a city skyline at night with a network of glowing nodes overlaid, symbolizing connectivity. The banner has the text "ThingSpeak™" and "Understand Your Things" along with a subtitle "The open IoT platform with MATLAB analytics." Two buttons are visible: a green "Get Started For Free" button and a white "Learn More" button. In the top right corner of the banner, there are links for "How to Buy", "Log In", and "Sign Up". The "Sign Up" link is specifically highlighted with a large red circle. At the bottom of the page, there are three main sections: "Collect" (represented by a cloud icon), "Analyze" (represented by a bar chart icon), and "Act" (represented by a gear icon). Each section has a brief description below it.

ThingSpeak™

Understand Your Things

The open IoT platform with MATLAB analytics.

Get Started For Free

Learn More

Sign Up

Collect

Analyze

Act

Send sensor data to the cloud.

Analyze and visualize your data with MATLAB.

Trigger a reaction.



Step 2: SignUp

Screenshot of a Firefox browser window showing the ThingSpeak sign-up page.

The browser header shows:

- Firefox menu bar: File, Edit, View, History, Bookmarks, Tools, Window, Help
- Address bar: Sign Up - ThingSpeak (https://thingspeak.com/users/sign_up)
- Toolbar icons: Back, Forward, Stop, Refresh, Home, etc.
- Status bar: U.S., Mon 6 Feb 12:55:57 PM

The ThingSpeak website navigation bar includes:

- ThingSpeak™
- Channels
- Apps
- Community
- Support
- How to Buy
- Log In
- Sign Up (highlighted in dark blue)

The main content area displays:

Sign up for ThingSpeak

In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your MathWorks Account. The ThingSpeak service is operated by The MathWorks, Inc.

A login form titled "Log in to your MathWorks Account" is shown, with fields for Email and Password, and a "Forgot your password?" link. A red circle highlights the "Create Account" button at the bottom left of the form.

At the bottom of the page, there is a link: "Already have a ThingSpeak account? Log In".



Step 3: Verify your email

service

Verify Email Address

To: Sylvester Fernandes

Inbox - Forsk

2 February 2017 at 6:14 PM

S

Thank you for registering with MathWorks!

To complete the registration process, verify your email address by clicking this link:

[Verify your email](#)

Sincerely,
MathWorks Customer Service Team

[Privacy policy](#)



Step 4: Login

Screenshot of a Firefox browser window showing the ThingSpeak login page. The URL in the address bar is <https://thingspeak.com/login>. The 'Log In' button in the top right corner is highlighted with a red circle.

Firefox menu bar: File Edit View History Bookmarks Tools Window Help

ThingSpeak navigation bar: Log In - ThingSpeak | https://thingspeak.com/login | Search | Channels Apps Community Support How to Buy Log In Sign Up

Log in to ThingSpeak

Email ID: sylvester@forsk.in

Password: [REDACTED]

Forgot your password?

Sign In

New user?
Sign up for the first time



Step 5: Create a New Channel

Screenshot of a Firefox browser window showing the ThingSpeak "New Channel" creation page. A red circle highlights the "Channels" button in the top navigation bar.

New Channel

Name:

Description:

Field 1: Field Label 1

Field 2:

Field 3:

Field 4:

Field 5:

Field 6:

Field 7:

Field 8:

Metadata:

Tags:

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Latitude:** Specify the position of the sensor or thing that collects data in decimal degrees. For example, the latitude of the city of London is 51.5072.
- **Longitude:** Specify the position of the sensor or thing that collects data in decimal degrees. For example, the longitude of the city of London is -0.1275.
- **Elevation:** Specify the position of the sensor or thing that collects data in meters. For example, the elevation of the city of London is 35.052.
- **Make Public:** If you want to make the channel publicly available, check this box.
- **URL:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Video ID:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.

Using the Channel



Mapping

Field 1

id

Field 2

start_location

Field 3

end_location

Field 4

driver_name

Field 5

customer

Field 6

material

Field 7

device_no

Field 8

type

trips

Field 1

door_operation

Field 2

power_status

Field 3

temperature

Field 4

humidity

Field 5

trip_id

Field 6

device_type

Field 7

device_no

Field 8

type

transaction



Step 6: View Channel Status

Screenshot of a Firefox browser window showing the ThingSpeak channel details for "coldchain".

The browser header shows:

- Firefox File Edit View History Bookmarks Tools Window Help
- 100% U.S. Mon 6 Feb 1:00:50 PM

The ThingSpeak channel page includes:

- Channel ID: 222415
- Author: sylvesterf
- Access: Public
- Captures the data for the cold chain problem
- View Options: Private View (selected), Public View, Channel Settings, API Keys, Data Import / Export
- Buttons: Add Visualizations, Data Export, MATLAB Analysis, MATLAB Visualization
- Channel Stats: Created: 3 days ago, Updated: 2 minutes ago, Last entry: 2 minutes ago, Entries: 26
- Field 1 Chart (door_operation): A line chart showing a sharp drop from 1.0 to -0.1 at approximately 12:00 on Feb 5, followed by a slow decline to about -0.25 by 12:00 on Feb 6.
- Field 2 Chart (power_status): A line chart showing a sharp drop from 1.0 to 0.0 at approximately 12:00 on Feb 5, followed by a steady increase to about 0.9 by 12:00 on Feb 6.

A red circle highlights the "Public View" button in the View Options section.



Note : Write API Keys

Screenshot of a Firefox browser window showing the ThingSpeak API Keys page for Channel ID 222415.

The URL in the address bar is https://thingspeak.com/channels/222415/api_keys.

The page title is "API Keys - ThingSpeak".

The top navigation bar includes links for File, Edit, View, History, Bookmarks, Tools, Window, Help, Channels, Apps, Community, Support, How to Buy, Account, and Sign Out.

The main content area shows the channel details:

- Channel ID: 222415
- Author: sylvesterf
- Access: Public
- Description: Captures the data for the cold chain problem

The navigation tabs at the top of the content area are: Private View, Public View, Channel Settings, API Keys (which is highlighted with a red circle), and Data Import / Export.

Write API Key

Key: OC277GP1H26YHUJO

Generate New Write API Key

Read API Keys

Key: NB5UI22JS0VTPVEE

Note: (empty)

Save Note Delete API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click [Generate New Write API Key](#).
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click [Generate New Read API Key](#) to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

Create a Channel

```
POST https://api.thingspeak.com/channels.json  
api_key=IBEDPIGOJJGCVZV6  
name=My New Channel
```

Update a Channel

```
PUT https://api.thingspeak.com/channels/222415
```



How to Export Data in csv Format

Screenshot of a Firefox browser window showing the ThingSpeak channel page for "coldchain".

The browser toolbar at the top includes: Apple icon, Firefox, File, Edit, View, History, Bookmarks, Tools, Window, Help, and several status icons.

The address bar shows: Data Import / Export - Thin... | https://thingspeak.com/channels/222415/import_export

The ThingSpeak navigation bar includes: ThingSpeak™, Channels, Apps, Community, Support, How to Buy, Account, and Sign Out.

The main content area displays:

- Channel ID: 222415**
- Author: sylvesterf**
- Access: Public**
- Captures the data for the cold chain problem**

A red oval highlights the **Data Import / Export** tab in the navigation bar.

Import: Upload a CSV file to import data into this channel. A "Browse..." button shows "No file selected." Below it is a "Time Zone" dropdown set to "(GMT+00:00) UTC". A green "Upload" button is present.

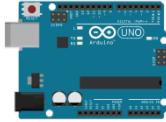
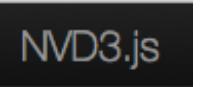
Export: Download all of this Channel's feeds in CSV format. A green "Download" button is present.

Help: Select a CSV file on your hard drive and import all of its data directly into this channel. Your CSV file should contain a date field in the first column. If your data doesn't contain timezone info, select one appropriately. [Learn More](#)

API Requests:

- Update Channel Feed - GET**: GET `https://api.thingspeak.com/update?api_key=0C277GP1H26YHUJO&field1=73`
- Update Channel Feed - POST**: POST `https://api.thingspeak.com/update.json`
`api_key=0C277GP1H26YHUJO`
`field1=73`
- Get a Channel Feed**: GET `https://api.thingspeak.com/channels/222415/feeds.json?results=1`
- Get a Channel Field Feed**: GET `https://api.thingspeak.com/channels/222415/fields/1.json`

Practical Flow for Best Cold Chain Problem

Sensors	Nodes	Gateway	Web Services	Data Stores	Analytics	Visualization
 Read, Calibrate and Display data from sensors 	 Develop code for reading SMS through Arduino and trigger the Alarm.	 Send the data to cloud using GPRS  Send the data to cloud using WiFi	 Create a REST based Web Service to send truck sensor data on the cloud Learn how to create cloud based programs	 Write the program to store the truck sensor data in NoSql Database.	 Analyze the data to solve business problem.	 Generate a neat report to visualize the data for management.  View and Control from Mobile App



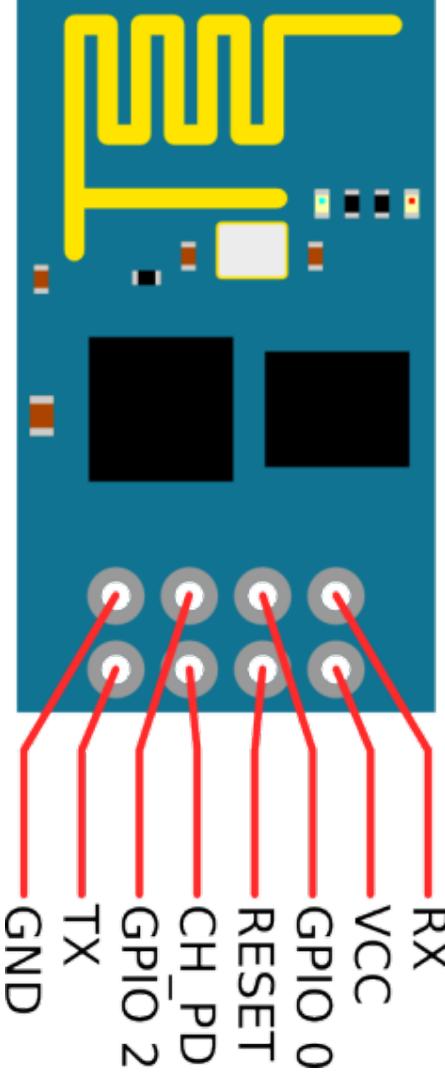
Gateway using WiFi

To View the Public Data

<https://thingspeak.com/channels/223662>



ESP8266 (WiFi) Notation

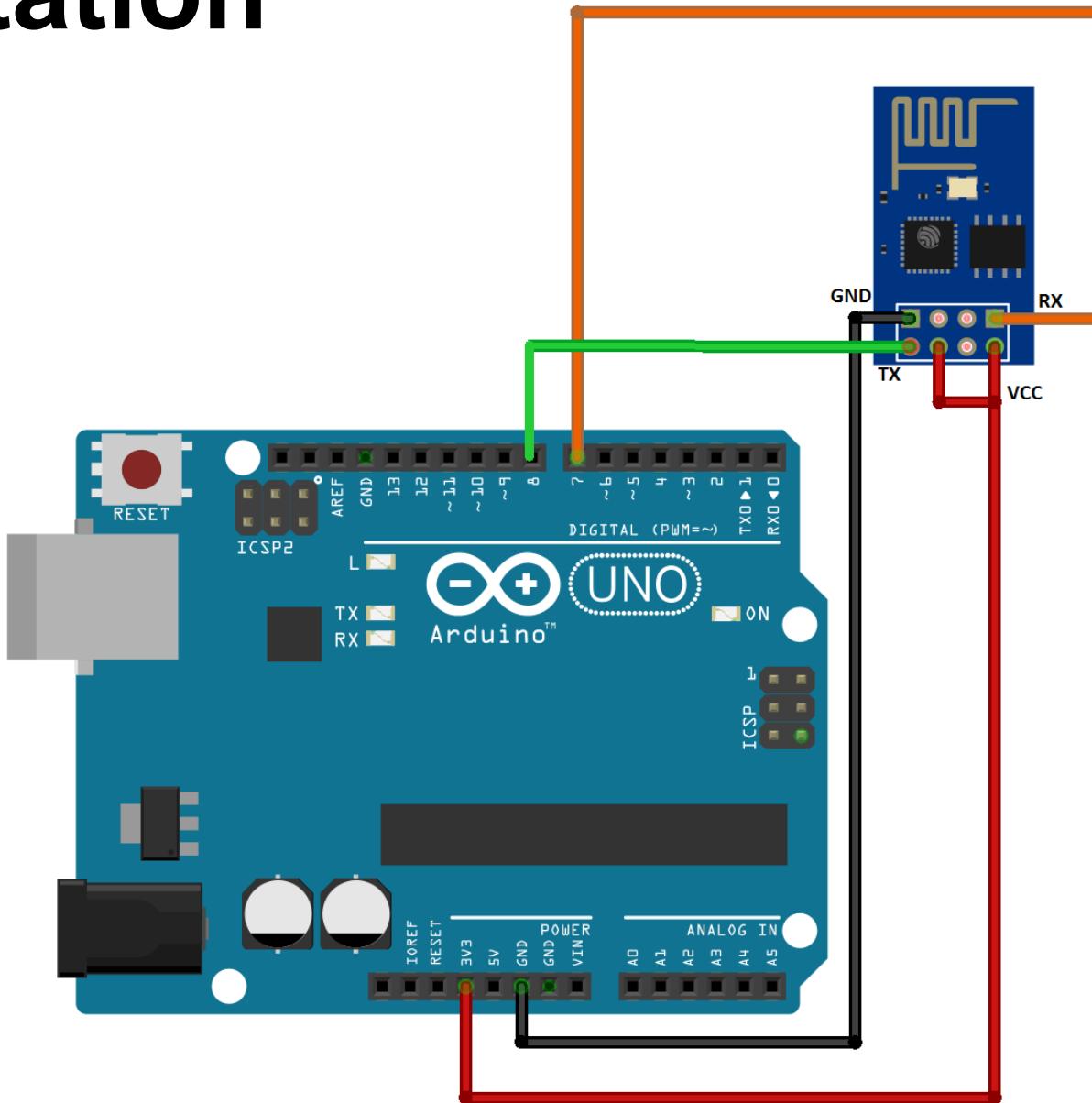


VCC to 3.3V of Arduino

GND to GND of Arduino

Rx – to Arduino PIN 7

Tx – to Arduino PIN 8





Logical Steps for WiFi

wifi_init()

connect_wifi()

void prepareData()

postToThingsSpeak();



Gateway using GSM

To View the Public Data

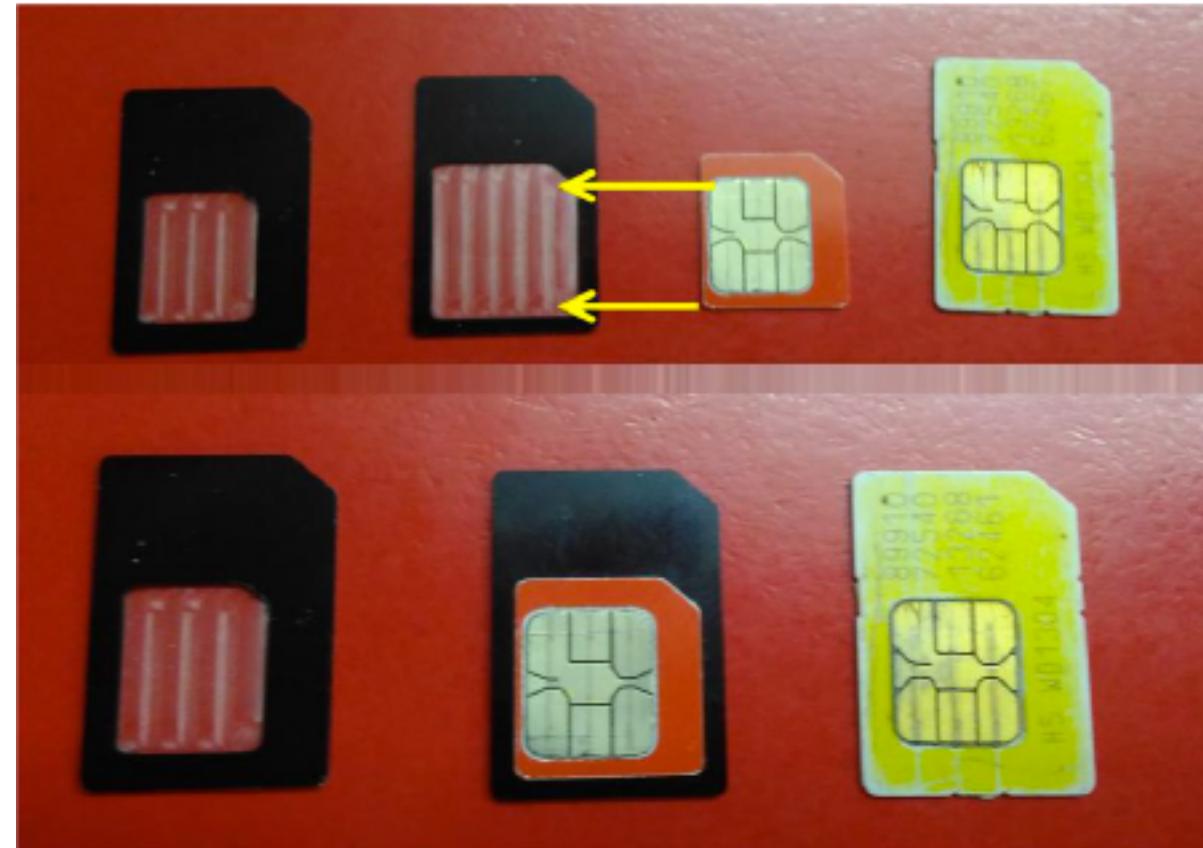
<https://thingspeak.com/channels/223662>



Inserting the SIM to Board

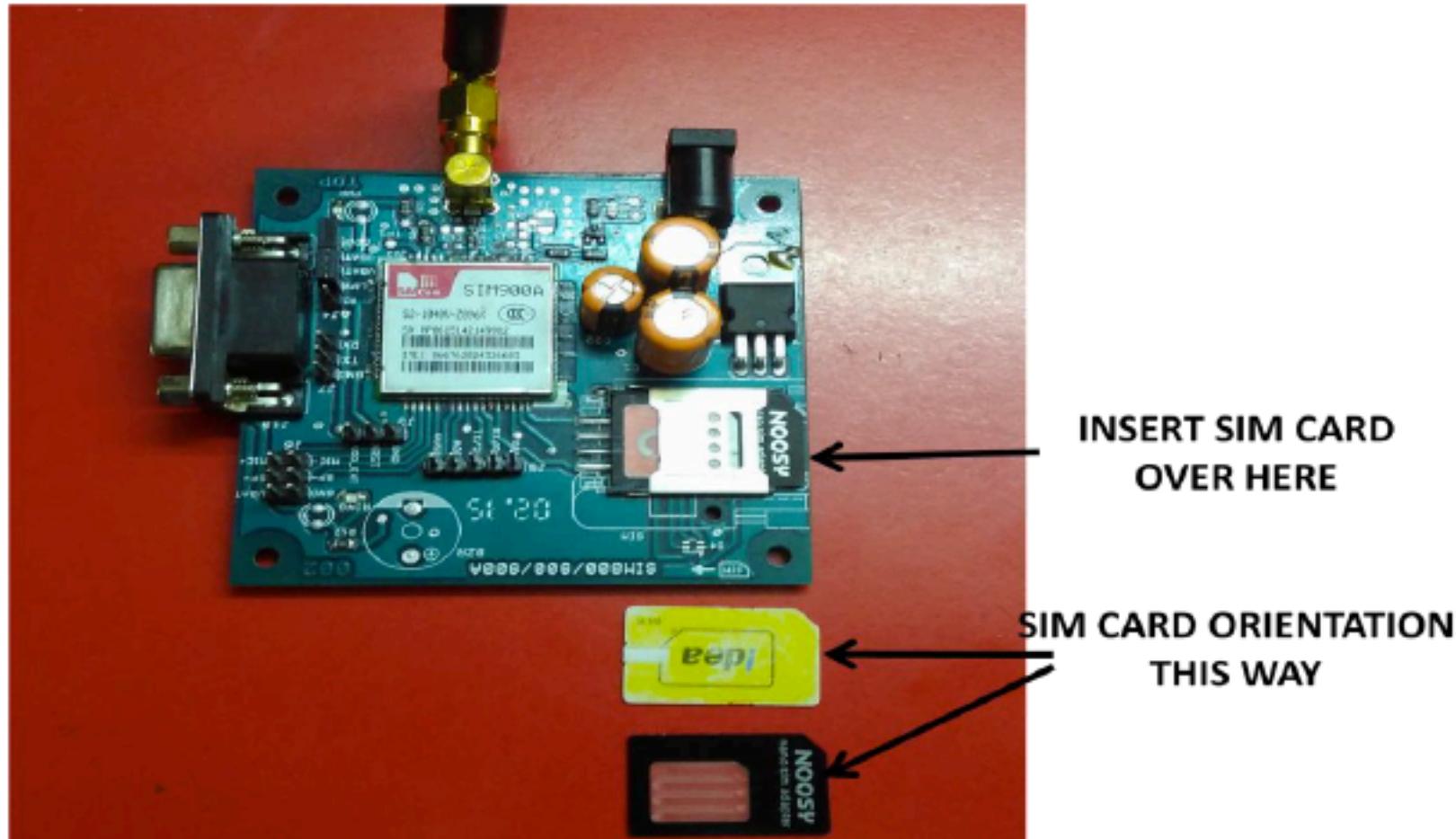


Step 1: Converting the GSM SIM





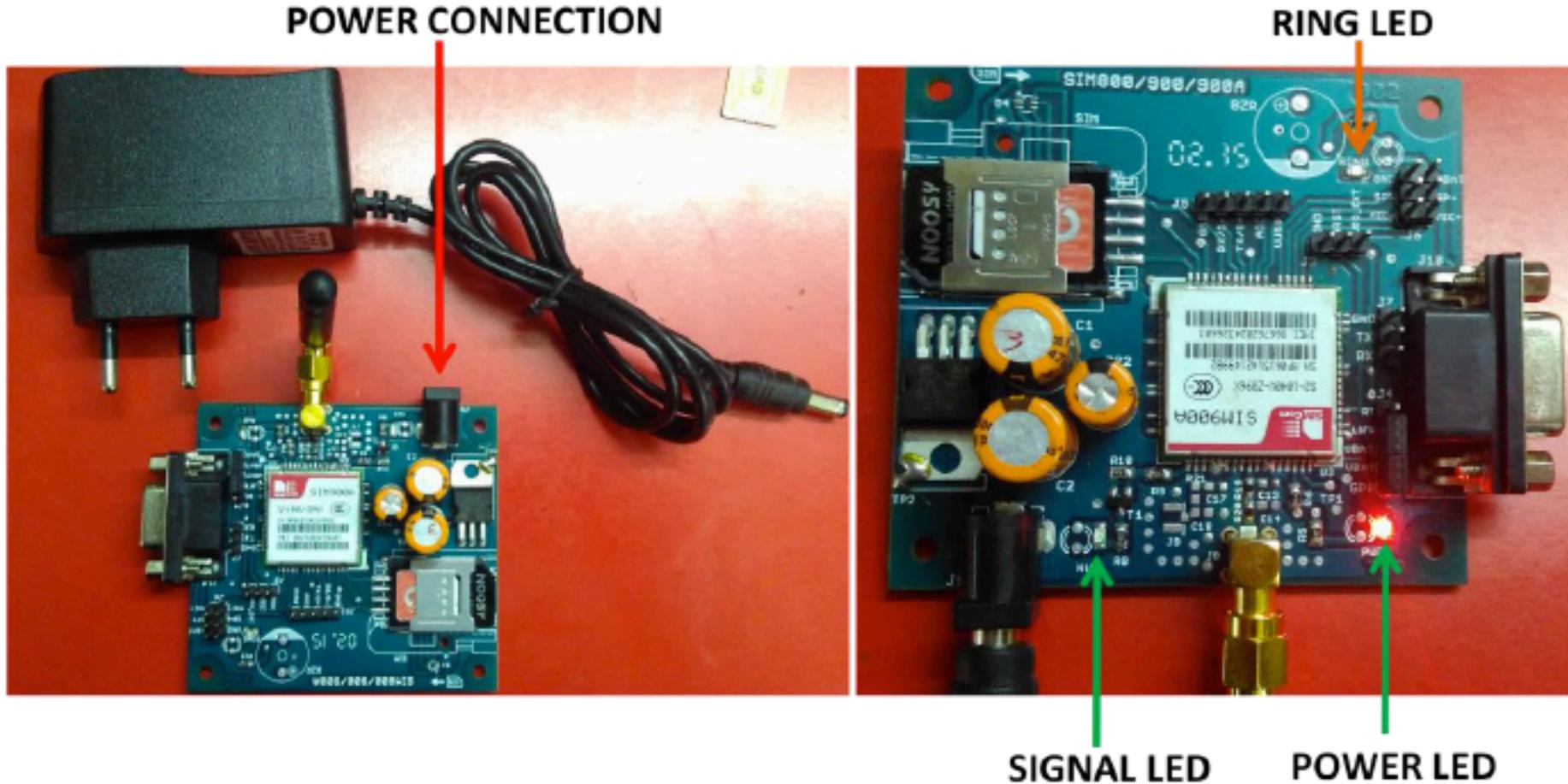
Step 2: Installing the GSM SIM



Note: Make sure to follow the Orientation of the SIM card with the copper pads of the SIM card touch the pins on the GSM Board.



Step 3: Power the GSM Board



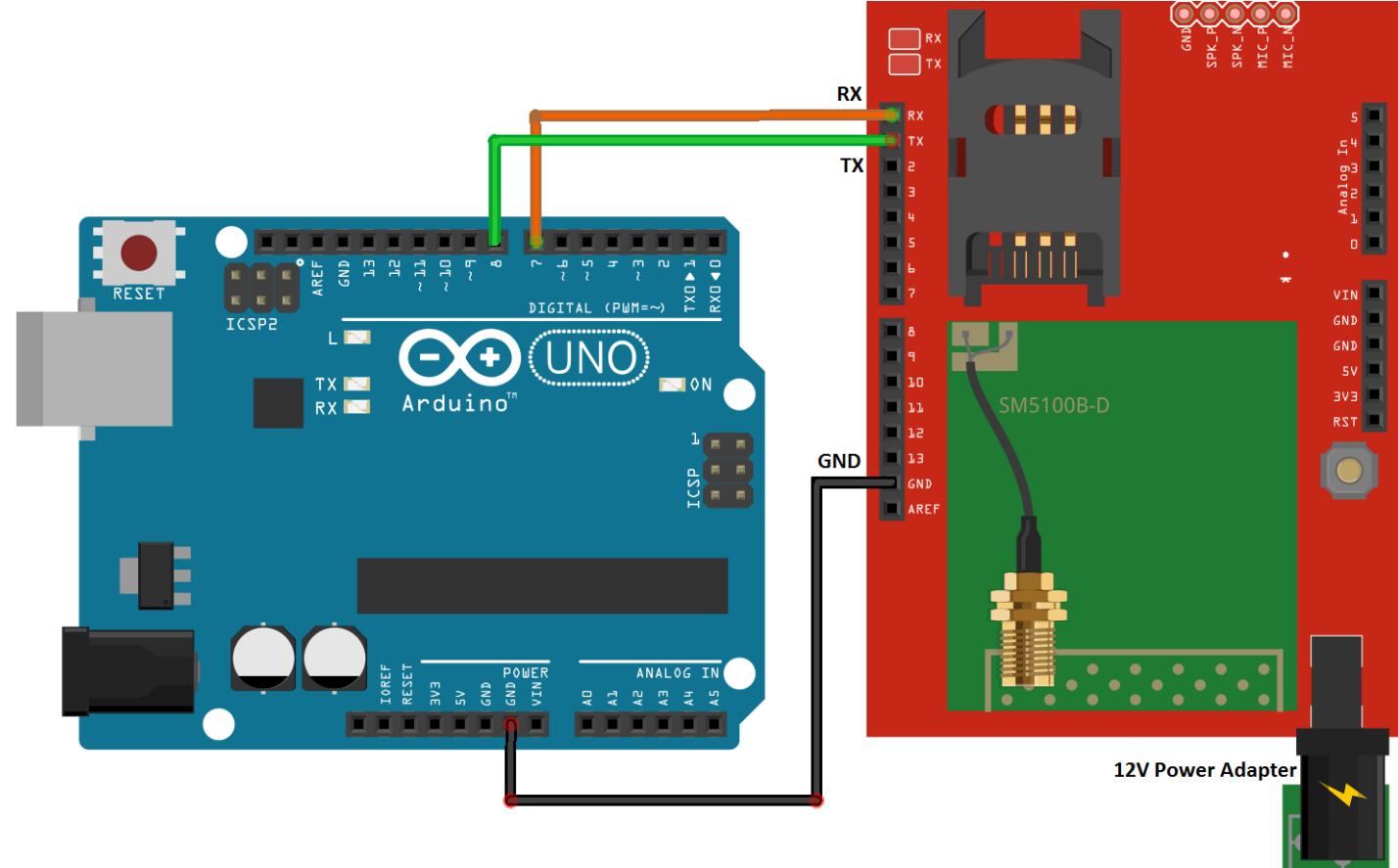


Step 4: Connecting the GSM to Arduino

- to GND on Arduino Board

Rx of GSM Board to PIN 7 on Arduino Board

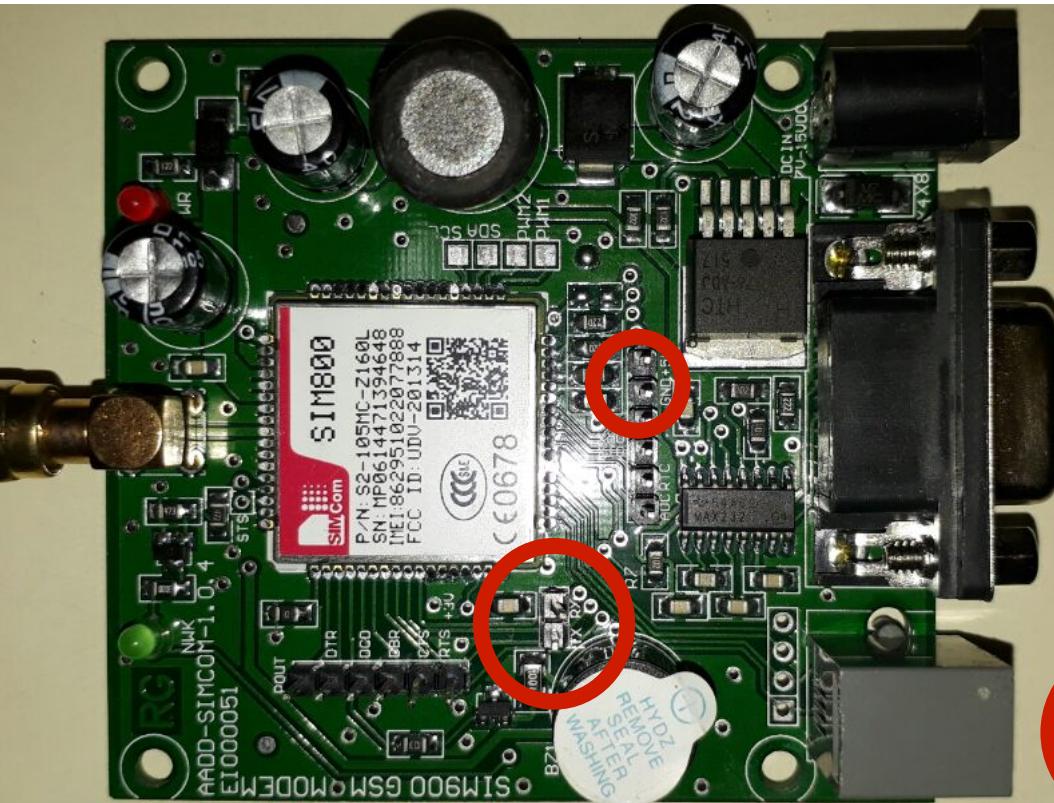
Tx of GSM Board to PIN 8 on Arduino Board



fritzing



Step 5: Different Type of GSM





Logical Steps for GSM

setup_gsm()

createJSONandpostData()



Experiments

6. wifi_http.ino (with hardcoded data)
7. wifi_http_sensor.ino
8. gsm_http.ino (optional)
9. gsm_http_sensor.ino (optional)



MQTT - History

Arlen Nipper (Arcom) &
Andy Stanford-Clark (IBM)
invent MQTT

royalty free

OASIS TC

MQTT 3.1.1 Release



1999

2010

2013

2014



MQ Telemetry Transport (MQTT)

- ❑ Lightweight messaging protocol for M2M communication
- ❑ Telemetry = Tele-Metering = Remote measurements
- ❑ Invented and sponsored by IBM.
Now Open source. Open Source libraries available.
- ❑ MQ originated from “message queueing (MQ)” architecture used by IBM for service oriented networks. There is **no** queueing in MQTT.
- ❑ Telemetry data goes from devices to a server or broker.
Uses a publish/subscribe mechanism.
- ❑ Lightweight = Low network bandwidth and small code footprint
- ❑ Many open source implementations of clients and brokers are available
Really small message broker (RSMB): C
Micro Broker



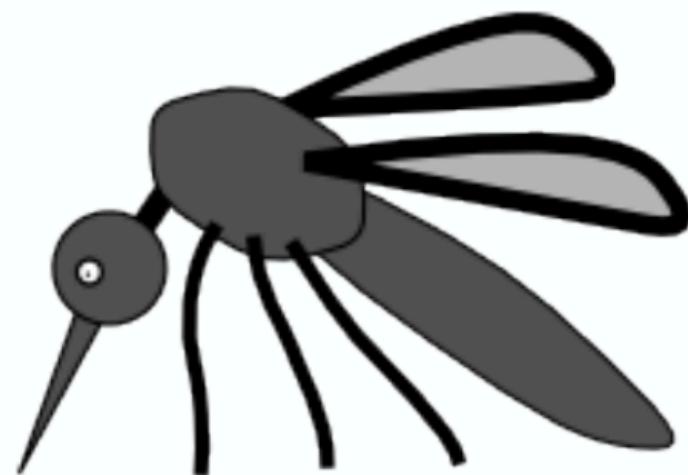
Eclipse Paho



- Open Source
- “Reference Implementation”
- Active Community
- Sync und Async API
- Java, C, C++, C#, Go,
Javascript, Python



Mosquitto



- Open Source
- Ideal for constrained devices
- Supports bridging
- Implemented in C



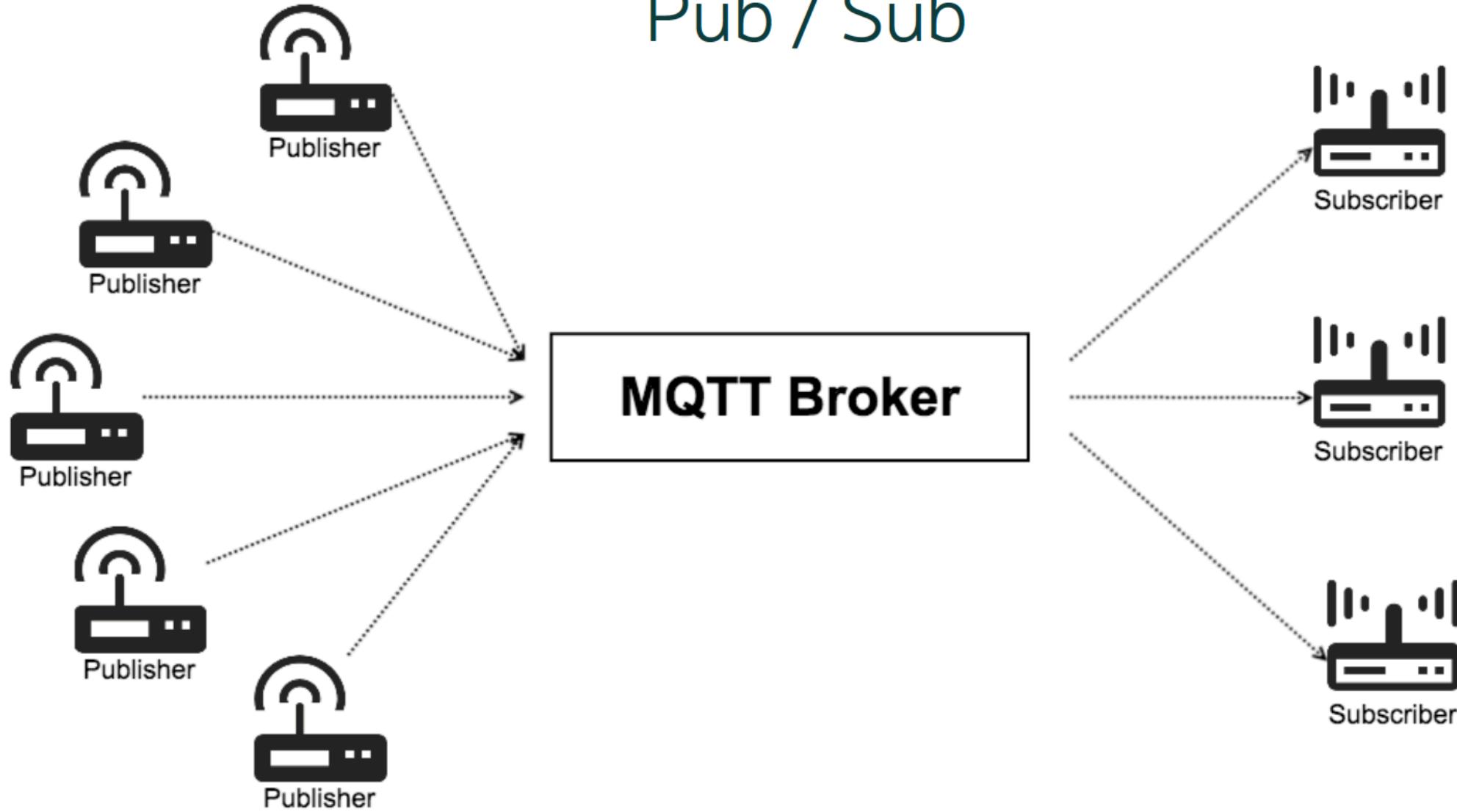
HiveMQ

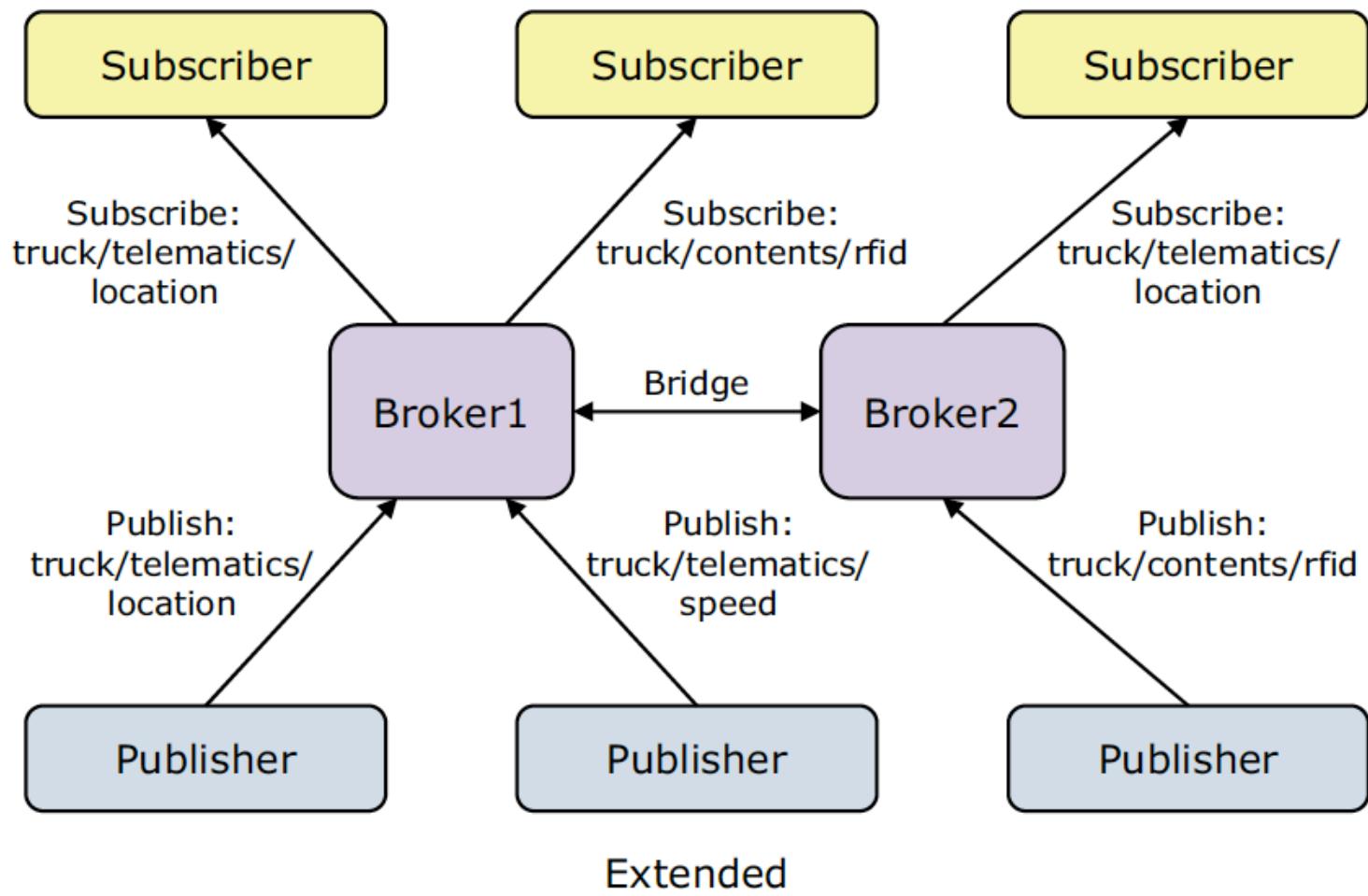
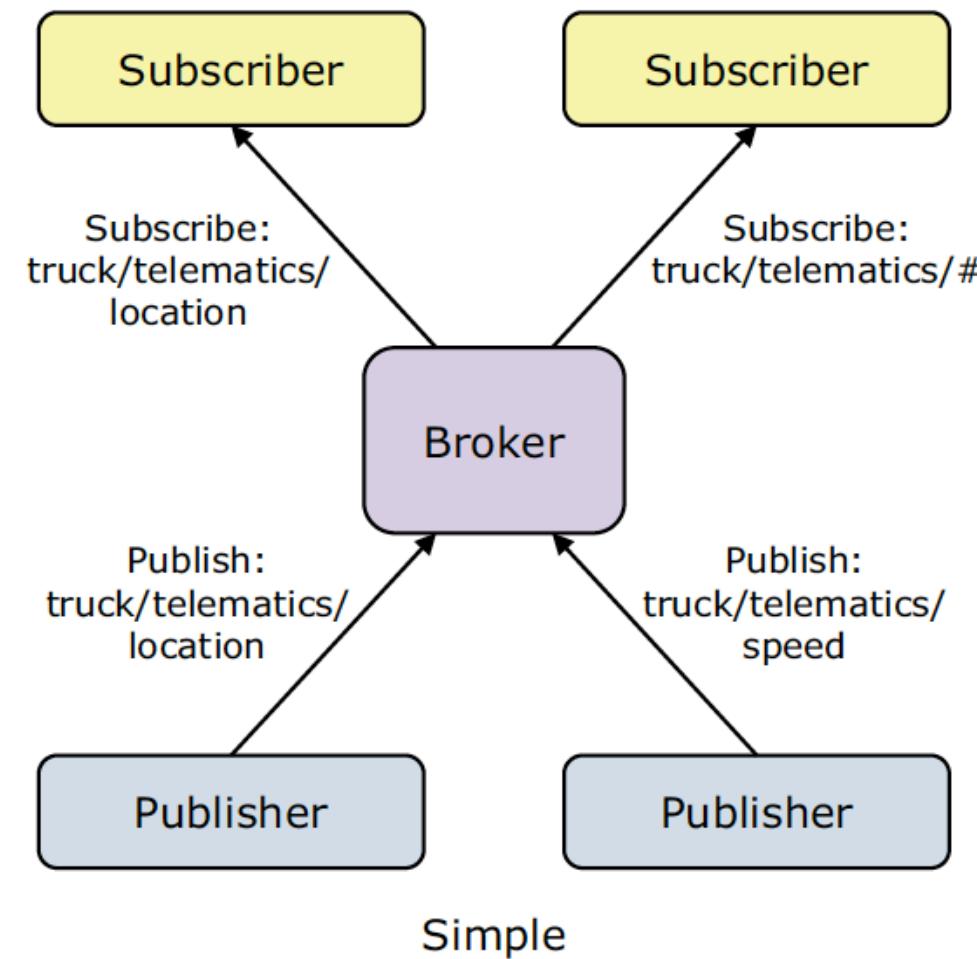


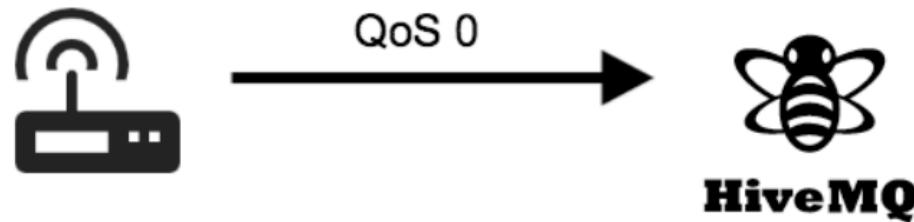
- High Performance MQTT Broker
- Built with security in mind
- Open Source Java Plugin System
- Supports Bridging
- Supports Clustering
- Designed for enterprise use



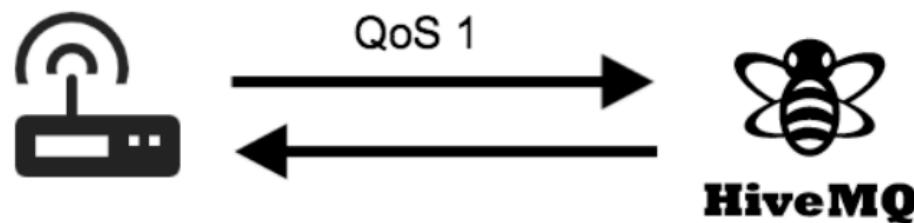
Pub / Sub



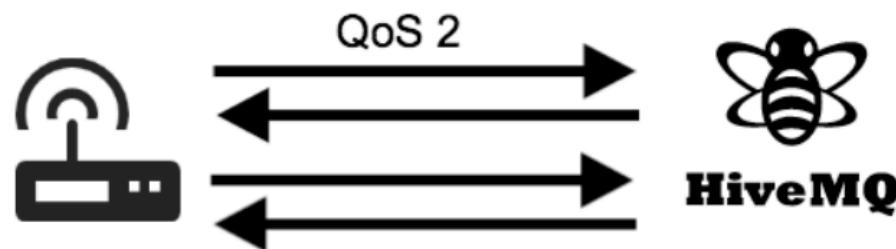




QoS 0 |
At most once delivery
(Best effort, No Ack)



QoS 1 |
At least once delivery
(Acked, retransmitted
if ack not received)



QoS 2 |
Exactly once delivery
Request to send (Publish)
Clear-to-send
(Pubrec), message (Pubrel)
ack (Pubcomp)

Retained Messages: Server keeps messages even after sending it to all subscribers. New subscribers get the retained messages

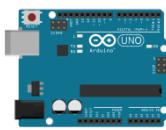


Experiments

10. wifi_mqtt.ino (optional)

Practical Flow for Best Cold Chain Problem

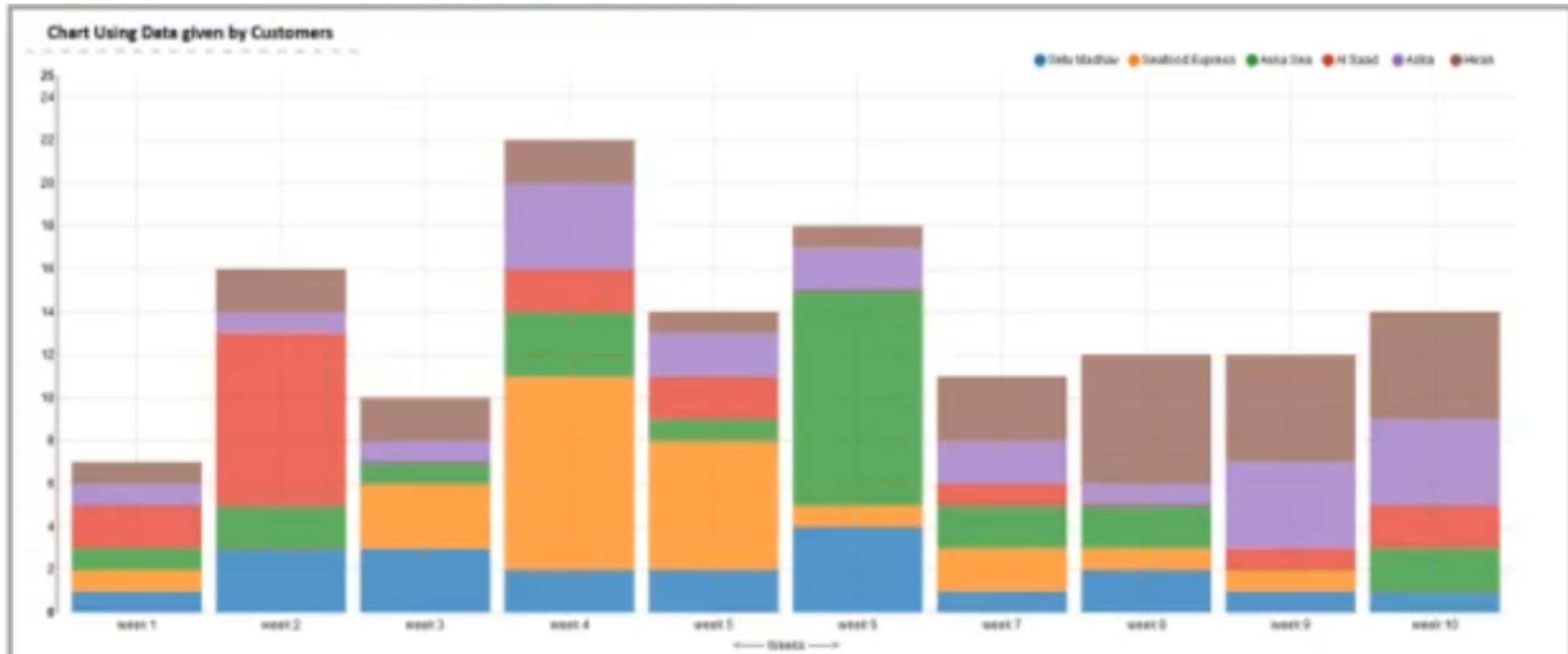


Sensors	Nodes	Gateway	Web Services	Data Stores ✓	Analytics	Visualization	
 Read, Calibrate and Display data from sensors 	 Develop code for reading SMS through Arduino and trigger the Alarm.	 Send the data to cloud using GPRS	 Send the data to cloud using WiFi	 Create a REST based Web Service to send truck sensor data on the cloud    Learn how to create cloud based programs	 Write the program to store the truck sensor data in NoSql Database.	 Analyze the data to solve business problem.	 Generate a neat report to visualize the data for management. 



Business Scenario

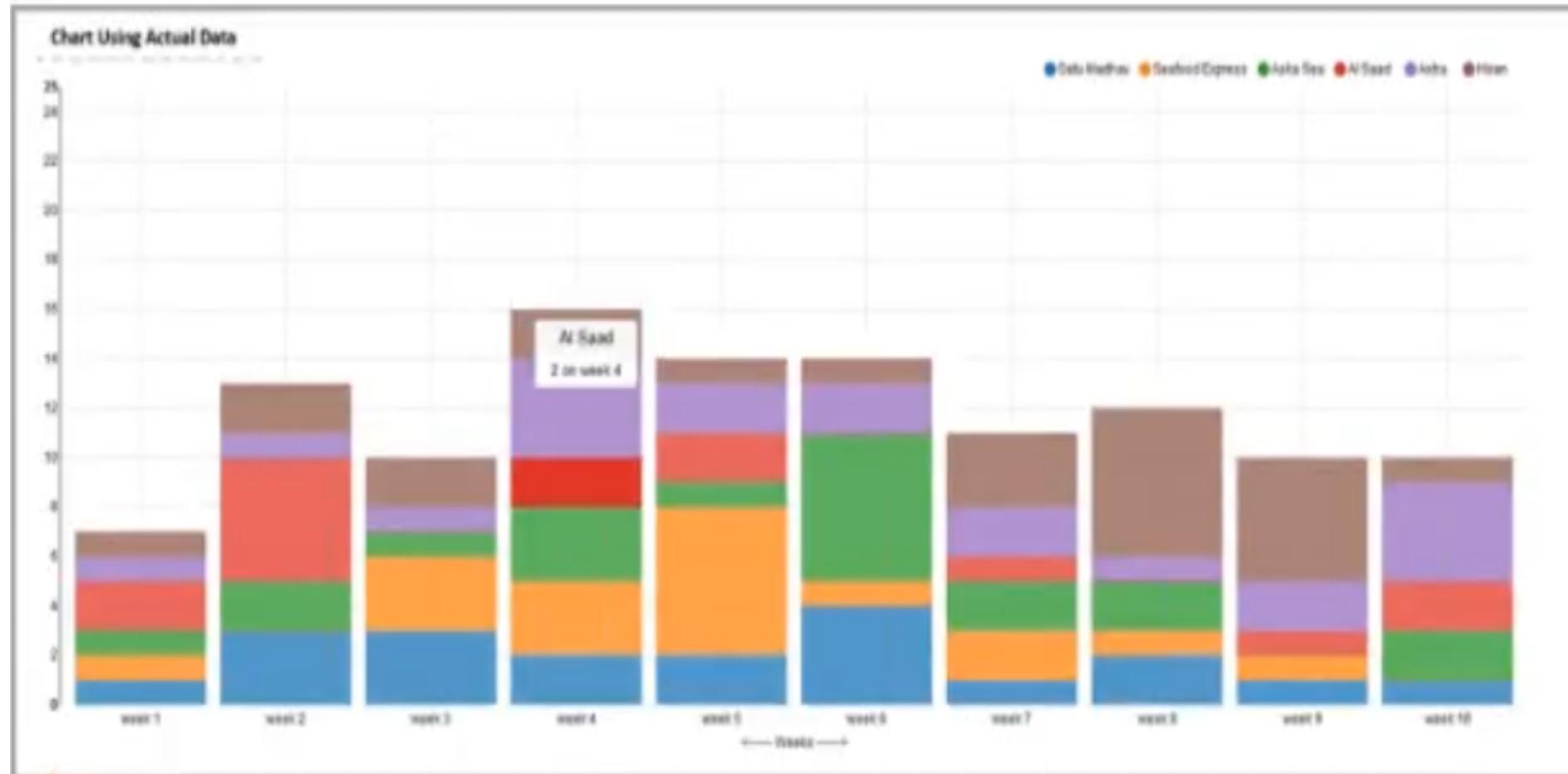
Below Chart describes the number of claims received in last 10 weeks





Business Scenario

Below chart describes the genuine claims which we got from the Cloud platform





Business Scenario

Below chart describes the fault claims



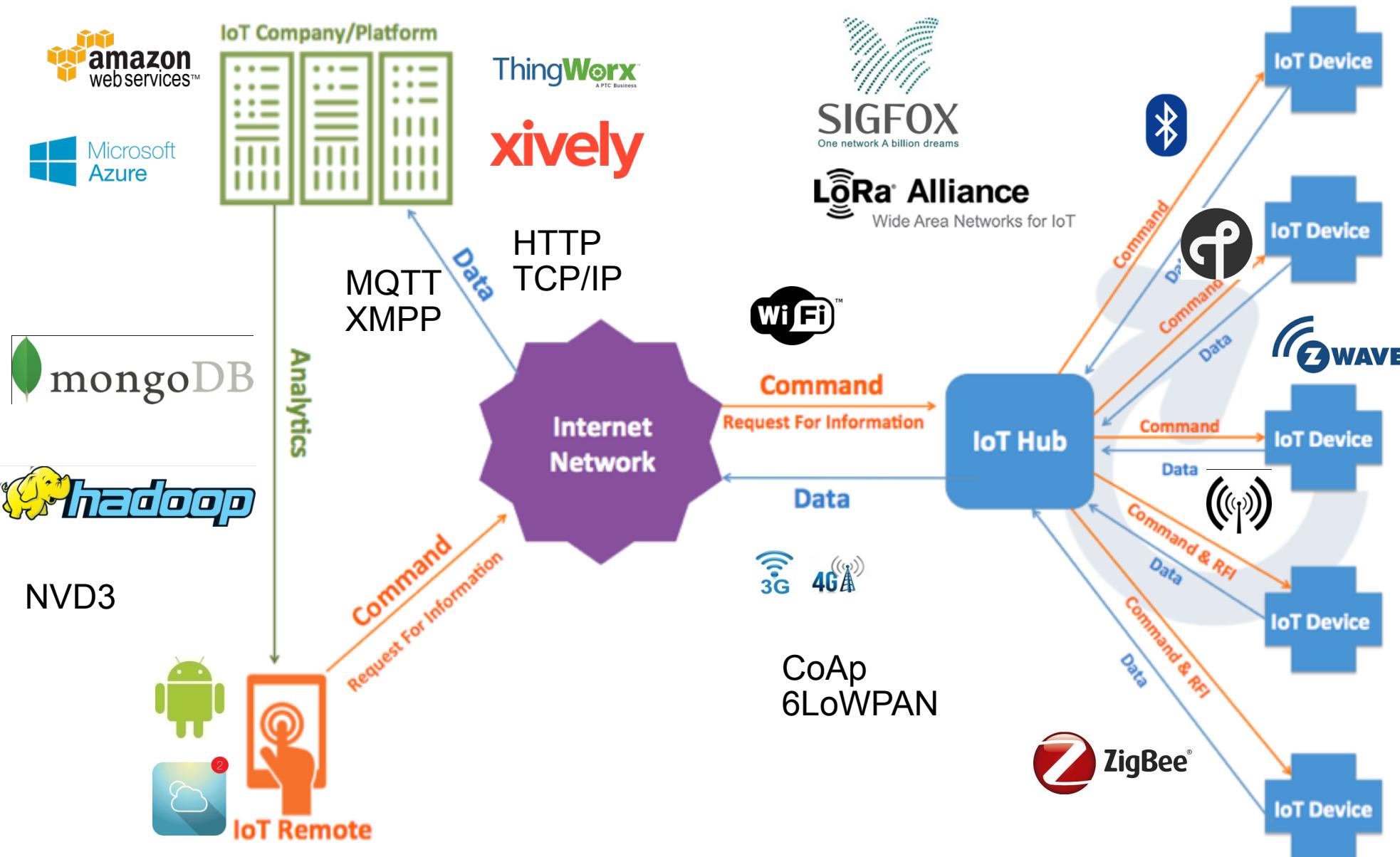


Enterprise IoT

(Bigger Picture)



IoT Solution Architecture





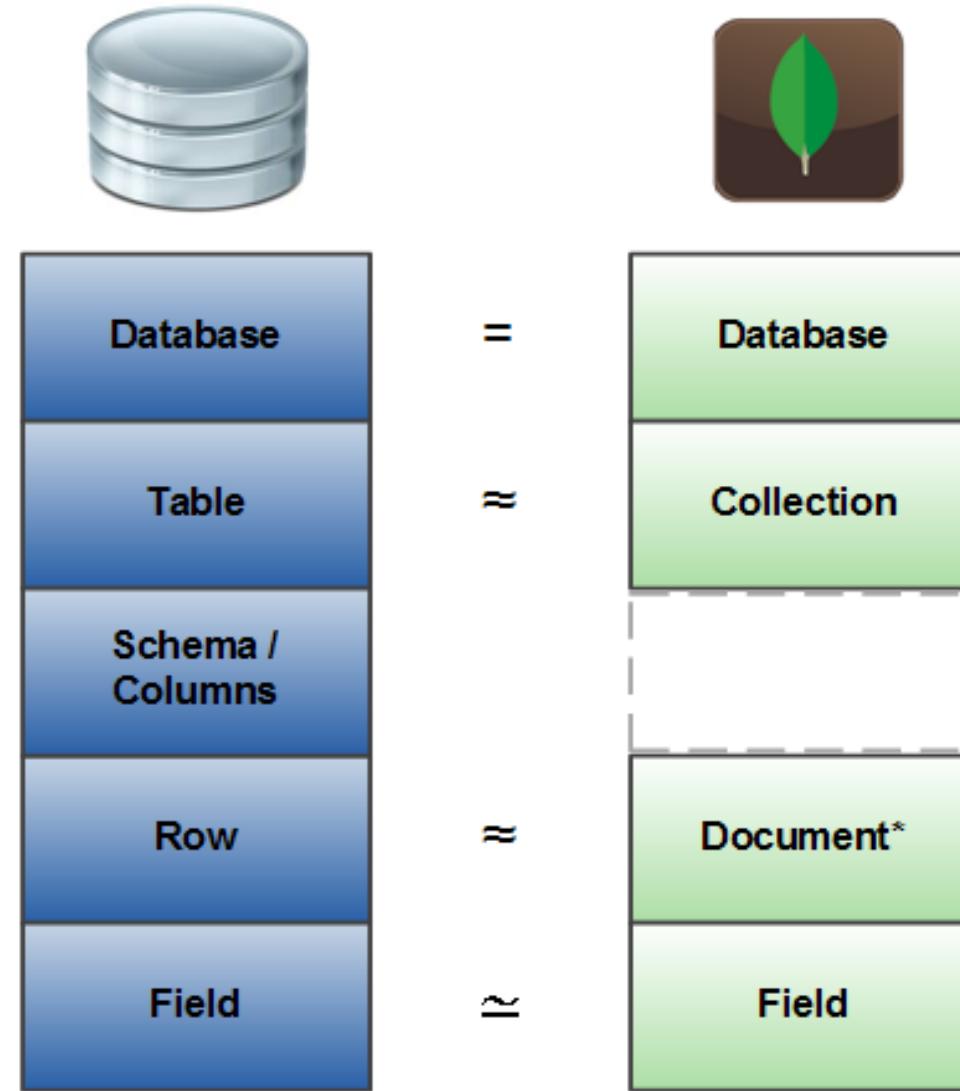
Introduction to Python



Introduction to matplotlib



RDBMS Vs MongoDB





Introduction to mlab



Demo of Raspberry Pi