9/9/2019

IOT(INTERNET OF THINGS)

-named by kelvin anston

what is iot?

-connecting everyday things embedded with electronics and software and sensors to the

internet enabling them to coolect and exchange data.

How it evolved?

-from the embedded to connect via cloud

-because of size and cost

Why?

-new business oppurtunities

-real time marketing product

(ex..weather predicition based on previous data analysis).

Benefits

-improved performance

-reduced cost

-new revenue stream(selling it)

-create innovative idea

(ex.. medbox(medicalbox),washing machine--msg system)

IOT architecture

SECURITY

APPLICATION

CLOUD PLATFORM

COMMUNICATION

DEVICE

COMMUNICATION

-device to device

-device to cloud(HTTP,MQTT)

PLATFORM

device device to device device to cloud platform model application

ARM,AVR NFC,BLUETOOTH, HTTP,MQTT,CAN IBM,AWS KERNAS, NODE.JS,DJANGO

INTEL,NVDIA, THREAD M-BUS TENSORFLOW IOS,ANDROID

RASBERRYPI

IOT BOARD

--arduino,rasberrypi,mosquito,android things

IOT SOLUTION DEVELOPMENT

--business case deveopment(use case)

--build vs buy decision(buy a board or built a hardware)

--proof of concept(document creation)

--initial pilot rollout(start of project based on document)

--commercial deployment(submit to the end user)

ARDUINO

--open source electonics platform based on easy to use hardware and software.

--compile,upload,new fille,open,save,serial monitor(serial data)

coding

--void setup()->define pin number,run or execute only one time.

--void loop()->execute or run repeatedly.

ADDIING A LIBRARY LIKE ARDUINI (ESP32)

--file-preference-path-json-additional based manager.

--tools-board-esp32 dev module.

ESP32

--espressif 32(bit processor)

--low cost,low opwer system

SPECIFICATION

Power supply-- usb,vin

GPIOS-general purpose input output system

Processors: Tensilica Xtensa 32-bit LX6 microprocessor,

operating at 160 or 240 MHz and performing at up to 600 DMIPS

Ultra low power (ULP) co-processor

Memory: 160 KiB static allocated + 160 KiB dynamic allocated DRAM

Wi-Fi: 802.11 b/g/n

Bluetooth: v4.2 BR/EDR and BLE

Peripheral interfaces:

12-bit SAR ADC up to 18 channels

2 × 8-bit DACs

10 × touch sensors (capacitive sensing GPIOs)

4 × SPI

2 × I²S interfaces

2 × I²C interfaces

3 × UART

16 pwm

10 capacitive sensing gpios

PINS

**--** Analog to Digital Converter (ADC)( ADC1\_CH0 (GPIO 36),ADC1\_CH1 (GPIO 37),ADC1\_CH2 (GPIO 38),ADC1\_CH3 (GPIO 39),ADC1\_CH4 (GPIO 32),ADC1\_CH5 (GPIO 33),ADC1\_CH6 (GPIO 34),ADC1\_CH7 (GPIO 35),ADC2\_CH0 (GPIO 4),ADC2\_CH1 (GPIO 0),ADC2\_CH2 (GPIO 2),ADC2\_CH3 (GPIO 15),ADC2\_CH4 (GPIO 13),ADC2\_CH5 (GPIO 12),ADC2\_CH6 (GPIO 14),ADC2\_CH7 (GPIO 27),ADC2\_CH8 (GPIO 25),ADC2\_CH9 (GPIO 26))

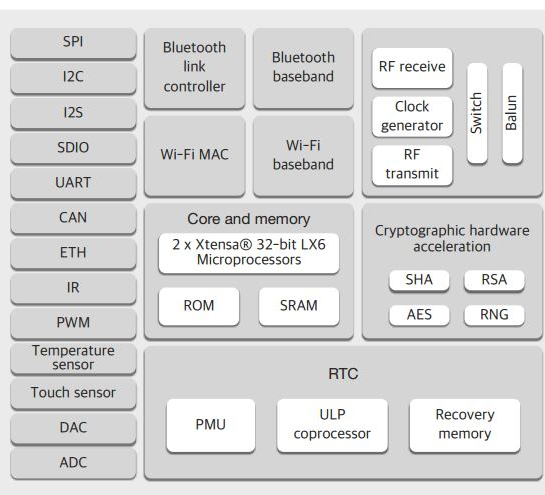
--DAC(DAC1 (GPIO25),DAC2 (GPIO26))

--INPUT(GPIO 34,GPIO 35,GPIO 36,GPIO 39)

--SPI(GPIO 6 (SCK/CLK),GPIO 7 (SDO/SD0),GPIO 8 (SDI/SD1),GPIO 9 (SHD/SD2),GPIO 10 (SWP/SD3),GPIO 11 (CSC/CMD))

--capacitive touch pin(T0 (GPIO 4),T1 (GPIO 0),T2 (GPIO 2),T3 (GPIO 15),T4 (GPIO 13),T5 (GPIO 12),T6 (GPIO 14),T7 (GPIO 27),T8 (GPIO 33),T9 (GPIO 32))

ESP



ESP SLEEP

ACTIVE-everything active(160-260ma)

MODEM-esp,elp,rtc(3-20mA)

LIGTH-enable and disable esp “

DEEP-elp,rtc(10microA)

Hibernation-rtc(2.5microA)

MOSI=master out slave in

BASIC PROGRAM

//digital write

void setup() {

pinMode(22,OUTPUT);

}

void loop() {

digitalWrite(22,HIGH);

// put your main code here, to run repeatedly:

delay(2000);//milliseconds delay

digitalWrite(22,LOW);//glow the led (digitalWrite for digitalsignal)

delay(2000);

}

//digital read

const int buttonPin=23;

const int ledPin=2;

int buttonState=0;

void setup() {

// put your setup code here, to run once:

pinMode(ledPin,OUTPUT);

pinMode(buttonPin,INPUT);

}

void loop() {

// put your main code here, to run repeatedly:

buttonState=digitalRead(buttonPin);

if(buttonState==HIGH)

digitalWrite(ledPin,HIGH);

else

digitalWrite(ledPin,LOW);

}

//serial read and write

const int ldrPin=4;

void setup() {

Serial.begin(9600);

}

void loop() {

// put your main code here, to run repeatedly:

int value=analogRead(ldrPin);

Serial.println(value);

delay(2000);

}

ULTRASONIC SENSOR

* Detect abtacles and its distance(2-400cm)

MODEL->HC-SR04

4PINS

VCC - input

TRIG - generate normal sound(output)

ECHO - receive the sound(INPUT)

GND - ground pin

Distance=(duration/2)\*velocity(340m/s)

SPECIFICATION

* Power Supply :+5V DC
* Quiescent Current : <2mA
* Working Current: 15mA
* Effectual Angle: <15°
* Ranging Distance : 2cm – 400 cm/1″ – 13ft
* Resolution : 0.3 cm
* Measuring Angle: 30 degree
* Trigger Input Pulse width: 10uS
* Dimension: 45mm x 20mm x 15mm

PROGRAM

#define trigPin 23

#define echoPin 22

void setup() {

// put your setup code here, to run once:

Serial.begin(9600);

pinMode(trigPin,OUTPUT);

pinMode(echoPin,INPUT);

pinMode(2,OUTPUT);

pinMode(4,OUTPUT);

Serial.println("us");

delay(1000);

}

void loop() {

// put your main code here, to run repeatedly:

int duration,distance;

digitalWrite(trigPin,HIGH);

delay(1000);

digitalWrite(trigPin,LOW);//

duration=pulseIn(echoPin,HIGH);

distance=(duration/2)\*0.0343;

if(distance<10)

{

digitalWrite(2,HIGH);

digitalWrite(4,LOW);

}

else

{

digitalWrite(4,HIGH);

digitalWrite(2,LOW);

}

Serial.println(distance);

Serial.println("cm");

delay(1000);

}

Ex---smart dustbin

DHT11

-- DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output

**Specification**

* Supply Voltage: +5 V
* Temperature range :0-50 °C error of ± 2 °C
* Humidity :20-90% RH ± 5% RH error
* Interface: Digital

PINS

VCC,GND,OUTPUT

PROGRAM

// Example testing sketch for various DHT humidity/temperature sensors

// Written by ladyada, public domain

include "DHT.h"

define DHTPIN 2 // what pin we're connected to

// Uncomment whatever type you're using!

define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE); //dht--instance variable

void setup() {

Serial.begin(9600);

Serial.println("DHTxx test!");

dht.begin(); // dht get activated

}

void loop() {

// Wait a few seconds between measurements.

delay(2000);

float h = dht.readHumidity();

// Read temperature as Celsius (the default)

float t = dht.readTemperature();

// Read temperature as Fahrenheit (isFahrenheit = true)

float f = dht.readTemperature(true);

// Check if any reads failed and exit early (to try again).

if (isnan(h) || isnan(t) || isnan(f)) { //isnan--is not a number

Serial.println("Failed to read from DHT sensor!");

return;

}

// Compute heat index in Fahrenheit (the default)

float hif = dht.computeHeatIndex(f, h);

// Compute heat index in Celsius (isFahreheit = false)

float hic = dht.computeHeatIndex(t, h, false);

Serial.print("Humidity: ");

Serial.print(h);

Serial.print(" %\t");

Serial.print("Temperature: ");

Serial.print(t);

Serial.print(" \*C ");

Serial.print(f);

Serial.print(" \*F\t");

Serial.print("Heat index: ");

Serial.print(hic);

Serial.print(" \*C ");

Serial.print(hif);

Serial.println(" \*F");

}

10/9/19

**COMMUNICATION**

**Server-**store information,client side interaction

There are 2 types of connection

-http

-mqtt

HTTP(hypertext transfer protocol)

-tcp/ip based communication

-application level protocol

-deliever data(images/gif/png)

-client(web browser,search engine etc)

-request response model

-port-80,8080

User ---request---------------🡪server

Server---response------------🡪user

HTTP CLIENT REQUEST

-It take place over a tcp/ip protocol.

-the client opens a socket

-convert to http via port

-the command is routed to the server via internet

-the server receive the command and process something.

HTTP REQUEST INVOLVES

Request line

Get-url-http/1.0

Request header

Connection : keep-alive

User-agent : modzilla

Host : localhost 8080

Accept : image/gif/jpg

Accept-encoding : gzip

Accept-language: en

Accept-charset:iso8859,\*,UTF-8

HTTP method

Get-for push and get (secureless)

Post-for push and get(secured)

Delete-delete data in the server

Put-update data in the server

HTTP STATUS CODE

1XX—information(request receiver and continue process)

2XX-success(request received successfully)

3XX-redirection(further action must be taken in order to complete the request

4XX-cilent error(request for bad syntax)

5XX-server error(the server failed)

device

Motor on

Motor off

DB

SERVER

Motor on—request is updated to the server—that will not automatically to the device…the device want to request for it and get updated at every time.

Program

MQTT (MESSAGE QUEUE TELEMETRY TRANSPORT PROTOCOL)

--consist of publish/subscriber messaging

-- low bandwidth

-- less battery consumption

-- port 1883

(most protocol in mqtt)

WORKING

publishing lap

Temperature sensor----------------🡪mqtt

mobile

publishing and subscribing takes place

MQTT BROKER

--Transmit the message received to subscriber and queue the receiver message from publisher.

--they send message through TOPIC and QOS

QOS(QUALITY OF SERVICE)

QOS0-this doesn’t guarantee the subscriber receiver the message

QOS1-this guarantee the message is received by atleast one subscriber

QOS2-this guarantee all the subscriber receive the message

PROGRAM

/\*\*\*\*\*\*\*\*\*

Rui Santos

Complete project details at http://randomnerdtutorials.com

\*\*\*\*\*\*\*\*\*/

//one client @a time

//internet should be in same hotspot and wifi

//26/on(off)

//27/on(off)

//server esp32

// Load Wi-Fi library

//get the ip from serial monitor and get the ip

//paste the ip in browser with 26/on

#include <WiFi.h>

//wifi.h beacuse of board

// Replace with your network credentials

const char\* ssid = "POCOPHONE F1"; // hotspot name

const char\* password = "348siddiq348"; // hotspot password

// Set web server port number to 80

WiFiServer server(80); //defining the port

// Variable to store the HTTP request

String header; //http header

// Auxiliar variables to store the current output state

String output26State = "off";

String output27State = "off";

// Assign output variables to GPIO pins

const int output26 = 26; //esp32 pins

const int output27 = 27;

void setup() {

Serial.begin(115200); //baudrate analog signal

// Initialize the output variables as outputs

pinMode(output26, OUTPUT);

pinMode(output27, OUTPUT);

// Set outputs to LOW

digitalWrite(output26, LOW);

digitalWrite(output27, LOW);

// Connect to Wi-Fi network with SSID and password

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);//function call

while (WiFi.status() != WL\_CONNECTED) {

delay(500); // while continue until the wifi get connected

Serial.print(".");

}

// Print local IP address and start web server

Serial.println("");

Serial.println("WiFi connected.");

Serial.println("IP address: ");

Serial.println(WiFi.localIP()); //get the ip address

server.begin(); //wifiserver server(80)-- begginnning the server

}

void loop(){

WiFiClient client = server.available(); // Listen for incoming clients

if (client) { // If a new client connects,

Serial.println("New Client."); // print a message out in the serial port

String currentLine = ""; // make a String to hold incoming data from the client

while (client.connected()) { // loop while the client's connected

if (client.available()) { // if there's bytes to read from the client,

char c = client.read(); // read a byte, then (get the ip address link)

Serial.write(c); // print it out the serial monitor

header += c; //get 27/on

if (c == '\n') { // if the byte is a newline character

// if the current line is blank, you got two newline characters in a row.

// that's the end of the client HTTP request, so send a response:

if (currentLine.length() == 0) {

// HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

// and a content-type so the client knows what's coming, then a blank line:

client.println("HTTP/1.1 200 OK");

client.println("Content-type:text/html");

client.println("Connection: close");

client.println();

// turns the GPIOs on and off

if (header.indexOf("GET /26/on") >= 0) {

Serial.println("GPIO 26 on");

output26State = "on";

digitalWrite(output26, HIGH);

} else if (header.indexOf("GET /26/off") >= 0) {

Serial.println("GPIO 26 off");

output26State = "off";

digitalWrite(output26, LOW);

} else if (header.indexOf("GET /27/on") >= 0) {//get the html page

Serial.println("GPIO 27 on");

output27State = "on";

digitalWrite(output27, HIGH);

} else if (header.indexOf("GET /27/off") >= 0) {

Serial.println("GPIO 27 off");

output27State = "off";

digitalWrite(output27, LOW);

}

// Display the HTML web page

client.println("<!DOCTYPE html><html>");

client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

client.println("<link rel=\"icon\" href=\"data:,\">");

// CSS to style the on/off buttons

// Feel free to change the background-color and font-size attributes to fit your preferences

client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");

client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

client.println(".button2 {background-color: #555555;}</style></head>");

// Web Page Heading

client.println("<body><h1>ESP32 Web Server</h1>");

// Display current state, and ON/OFF buttons for GPIO 26

client.println("<p>GPIO 26 - State " + output26State + "</p>");

// If the output26State is off, it displays the ON button

if (output26State=="off") {

client.println("<p><a href=\"/26/on\"><button class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/26/off\"><button class=\"button button2\">OFF</button></a></p>");

}

// Display current state, and ON/OFF buttons for GPIO 27

client.println("<p>GPIO 27 - State " + output27State + "</p>");

// If the output27State is off, it displays the ON button

if (output27State=="off") {

client.println("<p><a href=\"/27/on\"><button class=\"button\">ON</button></a></p>");

} else {

client.println("<p><a href=\"/27/off\"><button class=\"button button2\">OFF</button></a></p>");

}

client.println("</body></html>");

// The HTTP response ends with another blank line

client.println();

// Break out of the while loop

break;

} else { // if you got a newline, then clear currentLine

currentLine = "";

}

} else if (c != '\r') { // if you got anything else but a carriage return character,

currentLine += c; // add it to the end of the currentLine

}

}

}

// Clear the header variable

header = "";

// Close the connection

client.stop();

Serial.println("Client disconnected.");

Serial.println("");

}

}

Search google🡪 hivemq broker🡪( <https://www.hivemq.com/public-mqtt-broker/)public> mqtt🡪try mqtt browser client🡪give connect

In public create topic (ex karo) and create add new topic subscriber(karo)

Then type messge in published(click publish) ….,u will get response from the subscriber.

CLOUD

it is a term used to describe a global network of servers, each with a unique function

SERVICES

IAAS- **Infrastructure as a service** (**IaaS**) is a form of cloud computing that provides virtualized computing resources over the internet.

PAAS- Platform as a service (**PaaS**) is a **cloud computing** model in which a third-party provider delivers hardware and software tools -- usually those needed for application development -- to users over the internet. A **PaaS** provider hosts the hardware and software on its own infrastructure

SAAS- Software as a service (**SaaS**) is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the Internet. **SaaS** is one of three main categories of **cloud computing**, alongside infrastructure as a service (IaaS) and platform as a service (PaaS).

USECASE OF IOT

Device🡪IBM IOT platform🡪node-RED🡪web app or mobile app

IBM IOT platform

--Go to cloud.ibm.com🡪ibm cloud(<https://cloud.ibm.com/>)

-->catalog🡪internet of things🡪internet of things platform

🡪service name(smart weather monitoring) ---(scroll down) pricing plans🡪lite🡪create🡪launch

It will direct to ibm watson

--Adding device (Device Type- smartweathermonitor,device id- esp32)🡪next🡪next🡪 Authentication Token(-123456789)🡪next🡪finish

COPY THE Device Credentials

Top of Form

Organization IDi59zwy

Bottom of Form

Top of Form

Device Type smartweathermonitor

Bottom of Form

Top of Form

Device ID esp32

Bottom of Form

Top of Form

Authentication Methoduse-token-auth

Bottom of Form

Top of Form

Authentication Token123456789

CREATING IBM IOT SIMULATOR

Search-ibm iot simulator an d select iot simulator sensor(<http://watson-iot-sensor-simulator.mybluemix.net/>)

--use the copied device credentials🡪save changes

Simulator is ready

THEN GO TO WHATSON IOT PLATFORM

--check the status of device (connected)

--recent events🡪output of temperature and humidity

CREATING A DEVICE/connecting to device

In ibm iot whatson click🡪board🡪new board🡪create new board(board name- smartweathermonitor)🡪submit

Select the smartweathermonitor board🡪add new🡪get the chart(line chart)🡪check(esp32)🡪next🡪connect to new dataset(Event- iotsensor, Property- temperature,type-number,degree-celcius🡪next🡪next🡪next(any changes can be made)

The output is shown by changing the value in iot sensor check the line chart

TO create web app

IBM Watson IoT Platform

Apps🡪generate api keys🡪next🡪standard application🡪generate keys

Copy the api key and authendication token

Api key—(a-i59zwy-w5ad0m64sh)

Authendication token – (LTyb1GB!y&L8XOheLq)

Go to ibm cloud

Catalog🡪node-RED🡪click it

Enter app name- bhavanisathishkriss(hostname)🡪create

🡪running/awake🡪visit app url🡪redirect to node page🡪 next🡪not recommended(tick it)🡪next🡪next🡪finish🡪click go to your node-red flow

🡪editor will appear(node workspace console)

Input->inject->duble tap->string->bhavanisathish->repeatmode->set seconds

Output->debug(connect inject with debug)(see output in debug message)

IOT TEMPERATURE AND HUMIDITY

Input🡪ibmiot🡪double tap🡪  Authentication(api key)🡪edit🡪paste the copied api and authendication token🡪add🡪uncheck device type(paste it),id(paste it))🡪add(check whether it connected or not).

Then add a debug and deploy it.we get output in json form ({d:object})

JSON(JAVASCRIPT OBJECT NOTATION)

-has {key : values}…., key value remain same but the values changes.

To get the temperature value we use function node

---In function mode give name=temperature and function as(msg.payload=msg.payload.d.temperature;) 🡪delopy it

--text node to add the text

CREATING WEB APP

--- menu🡪manage platte🡪install🡪node-red-dashboard->install🡪close

Gauge(node)🡪edit🡪edit🡪add🡪add-🡪label(temperature)🡪deploy

Copy the link upto net and add/ui([https://bhavanisathishkriss.eu-gb.mybluemix.net](https://bhavanisathishkriss.eu-gb.mybluemix.net/)/ui)

11/9/19

CREATING MOBILE APP

GETTING THE API(application program interface)

Resource list🡪cloud foundary apps🡪bhavanisathishkriss🡪visit app url🡪go to node-red flow edit

NODE-RED

--- take 2 http node(input(url🡪sensordata) and output) and connect it with template(bhavanisathish)🡪deploy🡪newtab([https://bhavanisathishkriss.eu-gb.mybluemix.net/sensordata)--->output(bhavanisathish)](https://bhavanisathishkriss.eu-gb.mybluemix.net/sensordata)---%3eoutput(bhavanisathish))..

OPEN IBM SIMULATOR🡪iot sensor🡪give the required details🡪savechanges

GO TO NODE-RED🡪get http (input(sensordata) and output) and function(create global variable and get the global variable in json)

---global.set("temp",msg.payload.d.temperature); in old temp function

--in new function(msg.payload={"temp":global.get("temp"),"hum":global.get("humidity")};)

---connect it and deploy it🡪newtab([https://bhavanisathishkriss.eu-gb.mybluemix.net/sensordata)--->](https://bhavanisathishkriss.eu-gb.mybluemix.net/sensordata)---%3e)output({"temp":20,"hum":78}

).

CREATING URL BY GET METHOD VALUE

--- get http (input(command) and output) and connect it with function🡪 deploy it🡪newtab([<https://bhavanisathishkriss.eu-gb.mybluemix.net/command?value=MOTORON&value1=MOTOROFF>)--->](https://bhavanisathishkriss.eu-gb.mybluemix.net/value1=MOTORON&value2=MOTOROFF)---%3e)output({"value":"MOTORON","value1":"MOTOROFF"}

)

--- create separate function to get indivial output like MOTORON(msg.payload=msg.payload.value;)from the old function and connect it with debug node🡪deploy it

Output(9/12/2019, 11:33:17 AM[node: 55f98f29.6c8ac](https://bhavanisathishkriss.eu-gb.mybluemix.net/red/)msg.payload : string[7]

"MOTORON"

9/12/2019, 11:33:18 AM[node: 55f98f29.6c8ac](https://bhavanisathishkriss.eu-gb.mybluemix.net/red/)msg.payload : string[8]

"MOTOROFF")

MOBILE APP CREATION

Mit app inventor🡪( [https://appinventor.mit.edu/) -->create](https://appinventor.mit.edu/)%20--%3ecreate) apps🡪choose account🡪accept🡪continue

Start new project🡪design ur ui—horizontal and vertical arrangement(width—fill parent)—import label and text box in horizontal arrangement and button in vertical arrangement🡪(edit and create it creatively)…🡪add 2 web (give the url in web1(<https://bhavanisathishkriss.eu-gb.mybluemix.net/sensordata>)) and 1 clock.

Click to blocks for backend coding

🡪control contain control statement,screen,clock etc

🡪clock🡪 when clock1 timer🡪web1🡪call web1 get

🡪go to web1🡪drag web1 got.text(response content)🡪textbox🡪drag textbox enabled(change enable to text)

🡪list🡪drag look up pairs in key

🡪web1🡪drag call web1 json textdecode jsontext(drag it in pairs of above)🡪drag and drop get response content

🡪text🡪empty🡪temp(join it with look uo in pair key)

🡪web2🡪call web2 .get

---🡪duplicate the text1 change the temp to hum

🡪button🡪drag when button 1 click do🡪copy the url([https://bhavanisathishkriss.eu-gb.mybluemix.net/command?value=MOTORON)-->web2-->set](https://bhavanisathishkriss.eu-gb.mybluemix.net/command?value=MOTORON)--%3eweb2--%3eset) web 2 url to🡪text🡪paste the url

🡪duplicate it for another button

🡪built🡪provide apk url🡪scan the code after downloading mit app inventer in play store

//pubsub

12/9

AI

-artificail intelligence in which machine think like a human

-machine learning is a subset AI

-coded in python

WATSON ASSISTANT

Cloud.ibm.com🡪catolg🡪AI🡪watson assistant🡪create 🡪manage🡪launch Watson assistant🡪delete the my first assistant(delete) and create new assistant🡪name(karo)🡪create🡪add dialog skill🡪skill->name(medbot)🡪create🡪double tap on it

Intent---- An intent is a collection of user statements that have the same meaning. It is denoted by #

Entities--- Entities are like nouns or keywords. By building out your business terms in entities your assistant can provide targeted responses to queries. It is denoted by @

Dialog- A dialog is where you develop branching interaction flows for conversations between your customers and your assistant.

Creating intents

Create intent🡪intentname(song)🡪create🡪user example(hi,hello)🡪press back

Creating entities

Create entity🡪entityname(@song)🡪create🡪give value and synonym🡪add value

Creating dialog

Create dialog🡪between welcome and anything else create the dialog by add node below🡪name(response song) , if assistant response(entities)🡪assistant(single response)(give word close it )🡪try it

🡪customize🡪multiple response🡪on🡪apply🡪is assistant response and response(give that)

SYSTEM ENTITY🡪go to entities enable sys-person

@sys-person🡪create child node with assistant response true

If you want to store the name value

Open context editor🡪give name and value($name “@sys-person”)🡪try it.

Ibm Watson assistant🡪karo🡪preview link🡪click it🡪deployed

WATSON VISAUAL RECOGNITION

cloud.ibm.com-->catalog-->ai-->visual recognition-->create-->

manage(api---i6CVH7DdoQ8PCt9rB8Kk5ubQmeNuuEkqQuEN041HA1Md

url--https://gateway.watsonplatform.net/visual-recognition/api)

-->launch watson studio-->food🡪test🡪drag and drop the image🡪see the output..

cloud.ibm.com-->catalog-->ai-->visual recognition-->create-->

manage(api---i6CVH7DdoQ8PCt9rB8Kk5ubQmeNuuEkqQuEN041HA1Md

url--https://gateway.watsonplatform.net/visual-recognition/api)

-->launch watson studio-->general🡪test🡪drag and drop the image🡪output

RASBERRYPI

--mini pc

--credit card sized,single board computer

SPECIFICATION

**SoC:** Broadcom BCM2837  
**CPU:** 4× ARM Cortex-A53, 1.2GHz  
**GPU:** Broadcom VideoCore IV  
**RAM:** 1GB LPDDR2 (900 MHz)  
**Networking:** 10/100 Ethernet, 2.4GHz 802.11n wireless  
**Bluetooth:** Bluetooth 4.1 Classic, Bluetooth Low Energy  
**Storage:** microSD  
**GPIO:** 40-pin header, populated  
**Ports:** HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

VNC-connecting pi with laptop

PINOUT—(<https://pinout.xyz/>)

RASBERRYPI PROGRAM

Python 2 instal packages🡪sudo pip install opencv

version

cv2.\_\_version\_\_

Terminal🡪idle🡪write and compile

//buttton.py

import RPi.GPIO as GPIO #Import GPIO library

import time #Import time library(for delay purpose)

GPIO.setwarnings(False)#clear the old content

GPIO.setmode(GPIO.BOARD) #Set GPIO pin numbering(else we can use bcm)

GPIO.setup(18, GPIO.OUT)

GPIO.setup(12, GPIO.IN, pull\_up\_down=GPIO.PUD\_UP) #Enable input and pull up resistors

while True:

input\_state = GPIO.input(12) #Read and store value of input to a variable

if (input\_state == 1): #Check whether pin is grounded

print"light is on"

GPIO.output(18, 1) #1--high

else:

print"light is off"

GPIO.output(18, 0)

time.sleep(5)

//stt.py

#from \_\_future\_\_ import print\_function

import json

#from os.path import join, dirname

from ibm\_watson import SpeechToTextV1

from ibm\_watson.websocket import RecognizeCallback, AudioSource

import threading

# If service instance provides API key authentication

service = SpeechToTextV1(

## url is optional, and defaults to the URL below. Use the correct URL for your region.

url='https://gateway-lon.watsonplatform.net/speech-to-text/api',

iam\_apikey='zZKq0SA4iR0\_RjPMO6gO5ipjteqYosg3SYBvnU9WQJ-r')

#models = service.list\_models().get\_result()

#print(json.dumps(models, indent=2))

#model = service.get\_model('en-GB\_NarrowbandModel').get\_result()

#print(json.dumps(model, indent=2))

with open('/home/pi/Documents/harish.wav','rb') as audio\_file:

a= json.dumps(

service.recognize(

audio=audio\_file,

content\_type='audio/wav',

timestamps=True,

word\_confidence=True).get\_result(),indent=2)

print(a)

//print(a['results'][0]['alternatives'][0]['word\_confidence'])

//dht.py

open terminal

paste the url--git clone https://github.com/szazo/DHT11\_Python-->download the files

-->filemanager-->DHT!!\_python-->run example in idle

pgm

import RPi.GPIO as GPIO

import dht11

import time

import datetime

# initialize GPIO

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)##bcm

GPIO.cleanup()

# read data using pin 14

instance = dht11.DHT11(pin = 14)##calling the library

while True:

result = instance.read()

if result.is\_valid():

print("date and time"+str(datetime.datetime.now())

print( "Temperature: %d C" % result.temperature ,"Humidity: %d %%" % result.humidity)

time.sleep(5)

//tts.py

#

import json

from ibm\_watson import TextToSpeechV1 #sudo pip install --upgrade "ibm-watson>=3.2.0"

from ibm\_watson.websocket import SynthesizeCallback

#To install the pydub command is sudo pip install pydub

from pydub import AudioSegment

from pydub.playback import play #importing the play function from libarary

# If service instance provides API key authentication

service = TextToSpeechV1(

## url is optional, and defaults to the URL below. Use the correct URL for your region.

url='https://gateway-lon.watsonplatform.net/text-to-speech/api',

iam\_apikey='igrizDNk7tf4yBN3o4s2LmyaUI65YEz-yOFJWmI88rbN')//replace ur api key

with open('/home/pi/Documents/harish.wav', 'wb') as audio\_file:

audio\_file.write(

service.synthesize(

'Hello prasad', #text can be written

voice='en-US\_AllisonVoice',

accept='audio/wav'

).get\_result().content)

song = AudioSegment.from\_wav("/home/pi/Documents/harish.wav")

play(song)

//image.py

import json

from watson\_developer\_cloud import VisualRecognitionV3

visual\_recognition = VisualRecognitionV3(

'2018-03-19',

iam\_apikey='exn28QfZQ-j8t\_RCmsAaDvmrWIK8XNPelOYLoQXUlNVk')

with open(r'C:\Users\bethi\Desktop\Visual recognition\Beagle\16182438211\_e078fbdf05\_b.jpg', 'rb') as images\_file:

classes = visual\_recognition.classify(

images\_file,

threshold='0.6',

classifier\_ids='DefaultCustomModel\_1888879731').get\_result()

print(classes['images'][0]['classifiers'][0]['classes'][0]['class'])

//servo.py

# -\*- coding: utf-8 -\*-

import RPi.GPIO as GPIO

import time

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BCM)

GPIO.setup(7, GPIO.OUT)

p = GPIO.PWM(7, 50)

p.start(7.5)

try:

while True:

p.ChangeDutyCycle(7.5) #Neutral (90°)

time.sleep(2)

print "Servo Rotates 90 °C"

p.ChangeDutyCycle(12.5) #180°

time.sleep(2)

print "Servo Rotates 180 °C"

p.ChangeDutyCycle(2.5) #0°

time.sleep(2)

print "Servo Rotates 0 °C"

except KeyboardInterrupt:

p.stop()

SESSION COMPLETED…